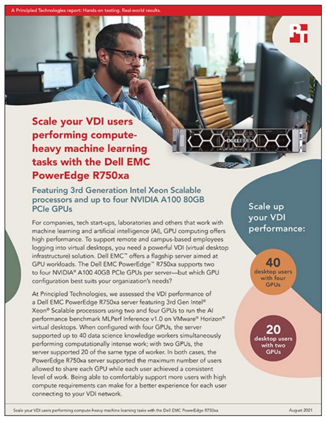
A Principled Technologies report: Hands-on testing. Real-world results.

Scale your VDI users performing compute-heavy machine learning tasks with the Dell EMC PowerEdge R750xa

Featuring 3rd Generation Intel Xeon Scalable processors and up to four NVIDIA A100 80GB PCIe GPUs

For companies, tech start-ups, laboratories and others that work with machine learning and artificial intelligence (AI), GPU computing offers high performance. To support remote and campus-based employees logging into virtual desktops, you need a powerful VDI (virtual desktop infrastructure) solution. Dell EMC™ offers a flagship server aimed at GPU workloads. The Dell EMC PowerEdge™ R750xa supports two to four NVIDIA® A100 40GB PCIe GPUs per server—but which GPU configuration best suits your organization's needs?

At Principled Technologies, we assessed the VDI performance of a Dell EMC PowerEdge R750xa server featuring 3rd Gen Intel® Xeon® Scalable processors using two and four GPUs to run the AI performance benchmark MLPerf Inference v1.0 on VMware® Horizon® virtual desktops. When configured with four GPUs, the server supported up to 40 data science knowledge workers simultaneously performing computationally intense work; with two GPUs, the server supported 20 of the same type of worker. In both cases, the PowerEdge R750xa server supported the maximum number of users allowed to share each GPU while each user achieved a consistent level of work. Being able to comfortably support more users with high compute requirements can make for a better experience for each user connecting to your VDI network.

Scale up your VDI performance:

40 desktop users with four GPUs

20 desktop users with two GPUs

Scale your VDI users performing compute-heavy machine learning tasks with the Dell EMC PowerEdge R750xa — August 2021

The science behind the report:

Scale your VDI users performing compute-heavy machine learning tasks with the Dell EMC PowerEdge R750xa

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Scale your VDI users performing compute-heavy machine learning tasks with the Dell EMC PowerEdge R750xa.

We concluded our hands-on testing on August 2, 2021. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on July 31, 2021 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Results of our VDI tests comparing a Dell EMC™ PowerEdge™ R750xa server with two NVIDIA® A100 GPUs (40GB PCIe®) to the same server with four GPUs. In MLPerf, scheduled queries per second is the rate at which the workload engine feeds inference samples to the implementation, and completed queries per second is the rate at which the implementation completes inference of the samples. The rate of scheduled and completed queries per second for both configurations matched the target rate we set (2,350) to within truncated precision of ±0.5 queries per second. Mean latency in MLPerf measures the average time between the engine supplying a sample for inference and the inference result's computation.

	2-GPU PowerEdge R750xa configuration	4-GPU PowerEdge R750xa configuration
Maximum concurrent DSKW VDI sessions	20	40
ESXi CPU usage	76%	88%
ESXi memory usage	28%	28%
MLPerf scheduled queries per second	2,350	2,350
MLPerf completed queries per second	2,350	2,350
MLPerf mean latency (milliseconds)	59.5	66.9

System configuration information

Table 2: Detailed information on the system we tested.

System configuration information	Dell EMC PowerEdge R750xa
BIOS name and version	Dell 1.1.1
Non-default BIOS settings	N/A
Operating system name and version/build number	VMware® ESXi 7.0 Update 2 Build-17630552 (A00)
Date of last OS updates/patches applied	4/22/21
System profile settings	Performance
Processor	
Number of processors	2
Vendor and model	Intel® Xeon® Gold 6330
Core count (per processor)	28
Core frequency (GHz)	2
Stepping	Model 106 Stepping 6
Memory module(s)	
Total memory in system (GB)	2,048
Number of memory modules	32
Vendor and model	Hynix® HMAA8GR7AJR4N-XN
Size (GB)	64
Type	PC3-12800R
Speed (MHz)	2,933
Speed running in the server (MHz)	2,933
Storage controller 1	
Vendor and model	Dell PERC H745 Front
Cache size (GB)	4
Firmware version	51.14.0-3707
Driver version	7.716.03.00
Storage controller 2	
Vendor and model	Dell BOSS-S2
Cache size (GB)	0
Firmware version	2.5.13.4008
Driver version	7.716.03.00-1vmw.702.0.0.17630552
Local storage 1	
Number of drives	4
Drive vendor and model	KIOXIA KPM5WVUG960G
Drive size (GB)	960
Drive information (speed, interface, type)	12 Gbps, SAS, SSD

System configuration information	Dell EMC PowerEdge R750xa
Local storage 2	
Number of drives	4
Drive vendor and model	Intel SSD D7-P5500
Drive size (GB)	1,920
Drive information (speed, interface, type)	PCIe 4.0, NVMe™, SSD
Network adapter 1	
Vendor and model	Broadcom® Gigabit Ethernet BCM5720
Number and type of ports	2 x 1Gb
Driver version	21.80.7
Network adapter 2	
Vendor and model	Intel Ethernet 25G 2P E810-XXV OCP
Number and type of ports	2 x 25Gb
Driver version	20.0.13
Cooling fans	
Vendor and model	Delta Electronics GFM0612HW-00
Number of cooling fans	6
Power supplies	
Vendor and model	Dell DS2400E-S1
Number of power supplies	2
Wattage of each (W)	2,400
Graphics processing unit	
Vendor and model	NVIDIA A100 Tensor Core GPU
Number of units	4
Memory (GB)	40
Form factor	PCIe
Firmware version	92.00.25.00.08
Driver version	460.73.02-1OEM.700.0.0.15525992

Table 3: Detailed configuration information for the network switches we used.

System configuration information	Dell EMC PowerEdge R750xa
Firmware revision	13.1530.0158
Operating system	MLNX-OS 3.6.5000
Number and type of ports	48 x 25GbE 8 x 100GbE
Number and type of ports used in test	6 x 25GbE
Non-default settings used	None

How we tested

Testing overview

We deployed VMware vSphere® 7.0 Update 2 to an infrastructure server (infra) and a system under test (SUT) server. The infrastructure server hosted virtual machines for cluster and VDI management with VMware vCenter® Server® 7.0 Update 2 and VMware Horizon® 2103, as well as test orchestration with VMware View Planner 4.6. We used View Planner to benchmark VDI performance for a hypothetical data science knowledge worker (DSKW) accessing the server under test. To emulate a DSKW, we created a View Planner Custom Workload and corresponding base image to run the workload. The base image used Ubuntu 18.04 as its operating system and included NVIDIA Docker and necessary NVIDIA GPU driver files. The machine learning workload (MLPerf v1.0) builds packaged in a Docker image deriving from NVIDIA's NGC Triton Server image, with simple custom logic to start the MLPerf benchmark. To expose the workload to View Planner, we created a small Python script to launch the workload container at the hook points defined by View Planner.

Note: For sections that reference deploying VMs from a template, we provide a table with VM template specifications below.

Table 4: Virtual machine details.

VM Name	Description	Operating System	vCPU count	Memory (GiB)	Disk (GiB)	vGPU (kind,count)	ESXi Server
DC1	Microsoft Active Directory domain controller	Microsoft Windows Server 2019 (64-bit)	4	8	40	N/A	infra
Jumpbox	Remote access proxy, deployment automation	Microsoft Windows Server 2019 (64-bit)	12	32	40	N/A	infra
nvlicsvr	NVIDIA License Server	Microsoft Windows Server 2019 (64-bit)	4	16	90	N/A	infra
vCenter	VMWare vCenter Server 7.02	VMware Photon OS (64-bit)	4	19	48.5	N/A	infra
view	VMware Horizon 2103 server	Microsoft Windows Server 2019 (64-bit)	4	8	40	N/A	infra
viewplanner-harness-4.6.0.0-16995088_OVF10	VMware View Planner 4.6 server	VMware Photon OS (64-bit)	8	8	64	N/A	infra
client-XXX	Client virtual machine for VMware View Planner 4.6 remote tests	Ubuntu 18.04 x86_64	2	4	64	N/A	client
desktopXXX	Desktop virtual machine for VMware View Planner 4.6 remote tests	Ubuntu 18.04 x86_64	2	32	64	4C,1	SUT

Enabling VT-d and SR-IOV on Dell EMC PowerEdge R750xa

1. In a web browser, connect to the server's iDRAC IP address.
2. Log into iDRAC.
3. In the main dashboard, click on Virtual Console to open a virtual console.
4. In the virtual console window, click Power → Power on system → Yes.
5. When the server prompts to press F2 to enter System Setup, press F2.
6. Click System BIOS → Processor Settings.
7. Ensure Virtualization Technology is enabled, and click Back.
8. Click Integrated Devices.
9. Ensure SR-IOV Global Enable is enabled, and click Back.
10. Click Finish.
11. Click Finish.
12. If prompted to confirm changes, click Yes.
13. When prompted to confirm exit, click Yes.
14. Allow the server to boot.

Installing vSphere 7.0 Update 2 on test and infrastructure servers

1. From the following link, download the Dell EMC Custom Image for ESXi 7.0 Update 2: https://my.vmware.com/group/vmware/evalcenter?p=vsphere-eval-7#tab_download.
2. Open a new browser tab, and connect to the IP address of the Dell EMC PowerEdge server iDRAC.
3. Log in with the iDRAC credentials. We used root/calvin.
4. In the lower left of the screen, click Launch Virtual Console.
5. In the console menu bar, click the Connect Virtual Media button.
6. Under Map CD/DVD, click the Browse… button, and select the image you downloaded in step 1. Click Open.
7. Click Map Device, and click Close.
8. On the console menu bar, click Boot, and select Virtual CD/DVD/ISO. To confirm, click Yes.
9. On the console menu bar, click the Power button. Select Power On System. To confirm, click Yes.
10. The system will boot to the mounted image and the Loading ESXi installer screen will appear. When prompted, press Enter to continue.
11. To Accept the EULA and Continue, press F11.
12. Select the storage device to target for installation. We selected the internal SD card. To continue, press Enter.
13. To confirm the storage targe, press Enter.
14. Select the keyboard layout, and press Enter.
15. Provide a root password, and confirm it. To continue, press Enter.
16. To install, press F11.
17. Upon completion, reboot the server by pressing Enter.

Installing vCenter Server Appliance 7.0 Update 2

1. From the VMware support portal, download VMware vCenter 7.0 Update 2: https://my.vmware.com.
2. Mount the image on your local system, and browse to the vcsa-ui-installer folder. Expand the folder for your OS. If the installer doesn't automatically begin, launch it yourself. When the vCenter Server Installer wizard opens, click Install.
3. To begin installation of the new vCenter server appliance, click Next.
4. To accept the license agreement, check the box, and click Next.
5. Enter the IP address of one of your newly deployed Dell EMC PowerEdge servers with ESXi 7.0 Update 2. Provide the root password, and click Next.
6. To accept the SHA1 thumbprint of the server's certificate, click Yes.
7. Accept the VM name, and provide and confirm the root password for the VCSA. Click Next.
8. Set the size for environment you're planning to deploy. We selected Medium. Click Next.
9. Select the datastore for installation. Accept the datastore defaults, and click Next.
10. Enter the FQDN, IP address information, and DNS servers you want to use for the vCenter server appliance. Click Next.
11. To begin deployment, click Finish.
12. When Stage 1 has completed, click Close. To confirm, click Yes.
13. Open a browser window, and navigate to `https://<vcenter.FQDN>:5480/`
14. On the Getting Started - vCenter Server page, click Set up.
15. Enter the root password, and click Log in.
16. Click Next.

17. Enable SSH access, and click Next.
18. To confirm the changes, Click OK.
19. Enter `vsphere.local` for the Single Sign-On domain name. Enter a password for the administrator account, confirm it, and click Next.
20. Click Next.
21. Click Finish.

Creating a cluster in vSphere 7.0 Update 2

1. Open a browser, and enter the address of the vCenter server you deployed. For example: `https://<vcenter.FQDN>/ui`
2. In the left panel, select the vCenter server, right-click, and select New Datacenter.
3. Provide a name for the new data center, and click OK.
4. Select the data center you just created, right-click, and select New Cluster.
5. Give a name to the cluster, and enable vSphere DRS. Click OK.
6. In the cluster configuration panel, under Add hosts, click Add.
7. Check the box for Use the same credentials for all hosts. Enter the IP Address and root credentials for the first host, and the IP addresses of all remaining hosts. Click Next.
8. Check the box beside Hostname/IP Address to select all hosts. Click OK.
9. Click Next.
10. Click Finish.

Configuring private/data network

Configuring vCenter for private/data network

1. Navigate to `https://<vCenter IP>:5480`
2. Log into vCenter as administrator@vsphere.local.
3. Click Networking.
4. In the top right corner of the Network Settings page, click Edit.
5. Select your preferred NIC. We used NIC 1, as we were already using NIC 0 for a public management IP.
6. Under Hostname and DNS, leave Obtain DNS settings automatically selected.
7. Under NIC settings, leave IPV4 enabled, and disable IPV6.
8. Enter a static IP address and prefix to use for vCenter management on the private network.
9. Leave the gateway blank.
10. Click Next.
11. Click Finish.

Configuring ESXi hosts for private/data network

Note: Repeat these steps for all infrastructure servers and the hosts for the ESXi server under test.

1. Log into vCenter as administrator@vsphere.local.
2. In the left pane, select the server under test.
3. Click the Configure tab.
4. Under Networking, click VMkernel adapters.
5. Click Add Networking.
6. Leave VMkernel Network Adapter selected, and click Next.
7. If you don't already have a vSwitch associated with your second NIC, select New standard switch.
8. Change MTU (Bytes) to `9000`
9. Click Next.
10. Enter a Network label and VLAN if applicable.
11. Under Available services, check Management, and click Next.
12. Click User static IPv4 settings, and enter an IP and subnet below.
13. Click Next, and click Finish.

Downloading NVIDIA evaluation license of NVIDIA vGPU

1. In a web browser, connect to https://www.nvidia.com/object/vgpu-evaluation.html.
2. Click Register for trial.
3. Enter your email and personal details.
4. In the Environment section, select the following:

• **Certified Server:**	Dell
• **VDI Hypervisor:**	VMware vSphere
• **VDI Seats:**	1-99
• **NVIDIA GPUs:**	A100
• **VDI Remoting Client:**	VMware Horizon
• **Primary Application:**	Other

5. Click Register.
6. Check your email for registration email, and click the set password link.
7. In the broswer window that pops up, set your password.
8. Log into https://nvid.nvidia.com.
9. Click Licensing Portal.
10. In the menu on the right, click Software Downloads.
11. Set the following search filter:

• **Product Family:**	vGPU
• **Platform:**	VMware vSphere
• **Platform Version:**	7.0

12. Click the download link for NVIDIA vGPU for vSphere 7.0 version 12.2.

Configuring GPU support on the ESXi server under test

Enabling SSH on the ESXi server under test

1. In vCenter, in the right panel, under Hosts and Clusters, locate the server under test. Right-click the server under test, and click Maintenance Mode → Enter Maintenance Mode.
2. Wait for the server to enter maintenance mode.
3. Click the server under test.
4. On the Configure tab, under System, click Services.
5. In the Services panel, click SSH → START.

Copying the NVIDIA vGPU driver zip file

1. Open an SSH connection to the ESXi server under test:

```
ssh root@<SUT ESXi Server IP>
```

2. Create a folder for installer files on the ESXi server under test:

```
mkdir /vmfs/volumes/nvme-ds-0/nvidia
```

3. On your local machine, use SCP to copy the NVIDIA zip file to the ESXi host:

```
scp /local/path/to/NVIDIA-GRID-vSphere-7.0-460.73.02-460.73.01-462.31.zip root@<SUT ESXi Server IP>:/
    vmfs/volumes/nvme-ds-0/nvidia/.
```

Installing the NVIDIA vGPU driver on the ESXi server under test

1. Open an SSH session to the ESXi server:

```
ssh root@<SUT ESXi server IP>
```

2. Change directories to the NVIDIA installer folder:

```
cd /vmfs/volumes/nvme-ds-0/nvidia/
```

3. Extract the zip file to a version-specific directory:

```
mkdir NVIDIA-GRID-vSphere-7.0-460.73.02-460.73.01-462.31
cd NVIDIA-GRID-vSphere-7.0-460.73.02-460.73.01-462.31
mv ../NVIDIA-GRID-vSphere-7.0-460.73.02-460.73.01-462.31.zip .
unzip NVIDIA-GRID-vSphere-7.0-460.73.02-460.73.01-462.31.zip
```

4. Extract the ESXi VIB file:

```
mkdir esxi
cd esxi
mv ../NVD-VGPU_460.73.02-1OEM.700.0.0.15525992_17944526.zip .
unzip NVD-VGPU_460.73.02-1OEM.700.0.0.15525992_17944526.zip
```

5. Install NVIDIA vGPU:

```
cd vib20
esxcli software vib install -v /vmfs/volumes/nvme-ds-0/nvidia/NVIDIA-GRID-vSphe
    re-7.0-460.73.02-460.73.01-462.31/esxi/vib20/NVIDIA-VMware_ESXi_7.0_Host_Driver/NVIDIA_bootbank_
    NVIDIA-VMware_ESXi_7.0_Host_Driver_460.73.02-1OEM.700.0.0.15
```

Note: you should see the following:

```
Installation Result
  Message: Operation finished successfully.
  Reboot Required: false
  VIBs Installed: NVIDIA_bootbank_NVIDIA-VMware_ESXi_7.0_Host_Driver_460.73.02-1OEM.700.0.0.15525992
  VIBs Removed:
  VIBs Skipped:
```

6. Reboot the ESXi server under test.

Confirming the NVIDIA vGPU driver is installed correctly on the ESXi server under test

1. Open an SSH session to the ESXi server under test.
2. Confirm the NVIDIA driver is loaded:

```
vmkload_mod -l | grep nvidia
```

you should see the following:

```
nvidia                         NNN   MMM
```

where NNN and MMM are non-zero numbers

3. Confirm the GPUs are visible to nvidia-smi:

```
nvidia-smi
```

Note: you should see the following:

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.73.02    Driver Version: 460.73.02    CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  A100-PCIE-40GB      On   | 00000000:17:00.0 Off |                    0 |
| N/A   50C    P0   105W / 250W |      0MiB / 40536MiB |    100%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  A100-PCIE-40GB      On   | 00000000:65:00.0 Off |                    0 |
| N/A   53C    P0   110W / 250W |      0MiB / 40536MiB |    100%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   2  A100-PCIE-40GB      On   | 00000000:CA:00.0 Off |                    0 |
| N/A   45C    P0   104W / 250W |      0MiB / 40536MiB |    100%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   3  A100-PCIE-40GB      On   | 00000000:E3:00.0 Off |                    0 |
| N/A   46C    P0   104W / 250W |      0MiB / 40536MiB |    100%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

4. Remove the ESXi server under test from maintenance mode.

Enabling vGPU hot migration in VMware vCenter

1. In VMware vCenter, in the Hosts and Clusters page, click the vCenter Server (the server itself; not the VM). For this deployment, the server is the root of the navigation tree on the left side in the Hosts and Clusters tab.
2. Click the Configure tab.
3. Under Settings, click Advanced Settings.
4. In the Advanced vCenter Server Settings panel, click EDIT SETTINGS.
5. In the Edit Advanced vCenter Server Settings modal dialog, in the Name column, click the filter icon.Type `vgpu`
   Note: The vgpu.hotmigrate.enabled setting should be listed in the table of options.
6. In the Value column for vgpu.hotmigrate.enabled, ensure the checkbox is checked.
7. Click Save.

Configuring GPU default sharing mode in VMware vCenter

1. In VMware vCenter, in the Hosts and Clusters page, click on the ESXi server under test.
2. Click the Configure tab.
3. In the Hardware section, click Graphics.
4. Click Host Graphics, and click EDIT...
5. Click the Shared Direct radio button.
6. Click the Spread VMs across GPUs (best performance) radio button.
7. Click OK.
8. Click Graphics Devices.
9. For each of the NVIDIA A100 GPUs:
   a. Click the row of the GPU.
   b. Click Edit...
   c. Click the Shared Direct radio button.
   d. Ensure the Restart X.Org server is checked.
   e. Click OK.
10. Once all GPUs have been configured in Shared Direct mode, restart the ESXi server.

Preparing the domain controller

Installing Windows Server 2019 with Active Directory

1. Log into the vSphere client as administrator@vsphere.local.
2. On the infra server, deploy a Windows Server 2019 VM from the template using 4 vCPU and 8 GB of memory. Name it `DC1` and log in as Administrator.
3. Launch Server Manager.
4. Click Manage → Add Roles and Features.
5. At the Before you begin screen, click Next.
6. At the Select installation type screen, leave Role-based or feature-based installation selected, and click Next.
7. At the Server Selection Screen, select the server from the pool, and click Next.
8. At the Select Server Roles screen, select Active Directory Domain Services.
9. When prompted, click Add Features, and click Next.
10. At the Select Features screen, click Next.
11. At the Active Directory Domain Services screen, click Next.
12. At the Confirm installation selections screen, check Restart the destination server automatically if required, and click Install.

Configuring Active Directory and DNS

1. After the installation completes, a screen should pop up with configuration options. If a screen does not appear, in the upper-right section of Server Manager, click the Tasks flag.
2. Click Promote this server to a Domain Controller.
3. At the Deployment Configuration screen, select Add a new forest.
4. In the Root domain name field, type `test.local` and click Next.
5. At the Domain Controller Options screen, leave the default values, and enter a password twice.
6. To accept default settings for DNS, NetBIOS, and directory paths, click Next four times.
7. At the Review Options screen, click Next.
8. At the Prerequisites Check dialog, allow the check to complete.
9. If there are no relevant errors, check Restart the destination server automatically if required, and click Install.
10. When the server restarts, log on using TEST\Administrator and the password you chose in step 5.

Configuring Windows Time service

1. To ensure reliable time, we pointed our Active Directory server to a physical NTP server.
2. Open a command prompt.
3. Configure and restart w32time:

```
W32tm /config /syncfromflags:manual /manualpeerlist:"<ip address of a NTP server>"
W32tm /config /reliable:yes
W32tm /config /update
W32tm /resync
Net stop w32time
Net start w32time
```

Configuring DHCP Server

1. Open Server Manager.
2. Select Manage, and click Add Roles and Features.
3. Click Next twice.
4. At the Select server roles screen, select DHCP Server.
5. When prompted, click Add Features, and click Next.
6. At the Select Features screen, click Next.
7. Click Next.
8. Review your installation selections, and click Install.
9. Once the installation completes, click Complete DHCP configuration.
10. On the Description page, click Next.
11. On the Authorization page, use the Domain Controller credentials you set up previously (TEST\Administrator). Click Commit.
12. On the Summary page, click Close.
13. On the Add Roles and Features Wizard, click Close.
14. In Server Manager, click Tools → DHCP.
15. In the left pane, double-click your server, and click IPv4.
16. In the right pane, under IPv4, click More Actions, and select New Scope.
17. Click Next.
18. Enter a name and description for the scope, and click Next.
19. Enter the following values for the IP Address Range, and click Next.
    - Start IP address: `172.16.10.1`
    - End IP address: `172.16.100.254`
    - Length: `16`
    - Subnet mask: `255.255.0.0`
20. At the Add Exclusions and Delay page, leave the defaults, and click Next.
21. Set the Lease Duration, and click Next. We used 30 days.
22. At the Configure DHCP Options page, leave Yes selected, and click Next.
23. At the Router (Default Gateway) page, leave the fields blank, and click Next.
24. At the Specify IPv4 DNS Settings screen, type `test.local` for the parent domain.
25. Type the preferred DNS server IPv4 address, and click Next.

26. At the WINS Server page, leave the fields empty, and click Next.
27. At the Activate Scope page, leave Yes checked, and click Next.
28. Click Finish.

Configuring the Active Directory SSL Certificate

1. Log onto DC1 as administrator@test.local.
2. Open Server Manager.
3. Select Manage, and click Add Roles and Features.
4. When the Add roles and Features Wizard begins, click Next.
5. Select Role-based or feature-based installation, and click Next.
6. Select DC1.test.local, and click Next.
7. At the server rolls menu, select Active Directory Certificate Services.
8. When prompted, click Add Features, and click Next.
9. Leave Select features as is, and click Next.
10. At the Active Directory Certificate Services introduction page, click Next.
11. Select Certificate Authority and Certificate Authority Web Enrollment.
12. When prompted, click Add Features, and click Next.
13. Click Next twice more. Click Install, and click Close.
14. In Server Manager, click the yellow triangle icon for Post-deployment configuration.
15. On the destination server, click Configure Active Directory Certificate Services.
16. Leave credentials as TEST\administrator, and click Next.
17. Select Certificate Authority and Certificate Authority Web Enrollment, and click Next.
18. Select Enterprise CA, and click Next.
19. Select Root CA, and click Next.
20. Select Create a new private key, and click Next.
21. Select SHA256 with a 2048 Key length, and click Next.
22. Leave the names fields and defaults, and click Next.
23. Change expiration to 10 years, and click Next.
24. Leave Certificate database locations as default. Click Next.
25. Click Configure.
26. When the configuration completes, click Close.
27. Open a command prompt, and type `ldp`
28. Click Connection, and click Connect.
29. For server, type `dc1.test.local`
30. Change the port to `636`
31. Check SSL, and click OK.

Configuring LDAP service

1. Open Administrative Tools, and click Certification Authority.
2. Click test-DC1-CA → Certificate Templates.
3. Right-click Manage.
4. Right-click Kerberos Authentication, and select Duplicate Template.
5. Click Request Handling.
6. Check the box for Allow private key to be exported, and click OK.
7. Right-click the new template, and rename it `LDAPoverSSL`
8. Return to the Certificates console. In the right pane, right-click New → Certificate Template to issue.
9. Select LDAPoverSSL, and click OK.

Preparing the VMware Horizon 2103 Connection Server virtual machine

Deploying the Windows Server 2019 virtual machine with VMware Horizon 2103 Connection Server

1. On the infra server, deploy a Windows Server 2019 VM from the template using 4 vCPUs and 8 GB of memory. For the VM name, type `view` and log in as an administrator.
2. Browse to VMware View installation media, and click VMware-viewconnectionserver-x86_64-7.12.0-15770369.exe.
3. Click Run.
4. At the Welcome screen, click Next.
5. Agree to the End User License Agreement, and click Next.
6. Keep the default installation directory, and click Next.
7. Select View Standard Server, and click Next.
8. At the Data Recovery screen, enter a backup password, and click Next.
9. Allow View Server to configure the Windows Firewall automatically, and click Next.
10. Authorize the local administrator to administer View, and click Next.
11. Choose whether to participate in the customer experience improvement program, and click Next.
12. Complete the installation wizard to finish installing View Connection Server.
13. Click Finish.
14. Reboot the server.
15. Join the VM to the test.local domain.

Configuring VMware Horizon 2103 Connection Server

1. Open a web browser, and navigate to http://<view connection1 FQDN>/admin.
2. Log in as administrator.
3. Under Licensing, click Edit License…
4. Enter a valid license serial number, and click OK.
5. Open View Configuration → Servers.
6. In the vCenter Servers tab, click Add…
7. Enter vCenter server credentials, and edit the following settings:
   - Max concurrent vCenter provisioning operations: 20
   - Max concurrent power operations: 50
   - Max concurrent View Composer maintenance operations: 20
   - Max concurrent View Composer provisioning operations: 20
   - Max concurrent Instant Clone Engine provisioning operations: 20
8. Click Next.
9. Uncheck Reclaim VM disk space.
10. At the ready to complete screen, click Finish.

Deploying the VMware View Planner 4.6 test harness

Deploying the VMware View Planner 4.6 test harness virtual machine

1. Download the viewplanner-harness-4.6.0.0-16995088_OVF10.ova file from VMware.
2. From the vCenter client, select the infra host, and right-click Deploy OVF Template…
3. In the Deploy OVF Template wizard, select local file, and click Browse…
4. Select viewplanner-harness-4.6.0.0-16995088_OVF10.ova, click Open, and click Next.
5. Select a DataCenter, and click Next.
6. Select the infra host, and click Next.
7. Review details, and click Next.
8. Accept the license agreements, and click Next.
9. Select the local DAS datastore, and click Next.
10. Select the priv-net network, and click Next.
11. Click Next, and click Finish to deploy the harness.
12. Power on the new VM, and note the IP address.

Configuring the VMware View Planner 4.6 test harness

1. Open a browser, and navigate to http://<ip address of the harness>:3307/vp-ui/.
2. Log in as follows:
   - Username: `vmware`
   - Password: `viewplanner`
3. Click Log in.
4. Click Servers.
5. Select infra, and click Add New.
6. Enter the following information:
   - Name: `vCenter`
   - IP: (The IP of vCenter)
   - Type: `vcenter`
   - DataCenter: `Datacenter`
   - Domain: `vsphere.local`
   - Username: `administrator`
   - Password: (The SSO password for vCenter)
7. Click Save.
8. Click Identity server.
9. Click Add new.
10. Enter the following information:
    - Name: `test.local`
    - IP: (The IP of DC)
    - Type: `microsoft_ad`
    - Username: `administrator`
    - Password: (The password for administrator@test.local)
11. Click Save.

Preparing the Ubuntu 18.04 VDI Data Science Knowledge Worker image

Creating a new virtual machine with Ubuntu 18.04 Desktop x86_64 installation media mounted

1. Using the VMware Web client, log into vCenter.
2. In the Hosts and Virtual Machines sidebar menu, right-click the infrastructure server, and select New Virtual Machine.
3. Under Select a creation type, select Create a new virtual machine, and click Next.
4. Under Select a name and folder, set the virtual machine name to `VDI-base-ubuntu`, and click Next.
5. Under Select a compute resource, select the infrastructure server, and click Next.
6. Under Select storage, select the local datastore, and set the VM Storage Policy to Management Storage policy - Thin. Click Next.
7. Under Select compatibility under Compatible with, select ESXi 7.0 or later, and click Next.
8. Under Select a guest OS, set Guest OS Family to Linux, and set Guest OS Version to Ubuntu Linux (64-bit). Click Next.
9. Under Customize hardware, set the following:
   - **CPU:** 2
   - **Memory:** 32 GB
   - **New Hard Disk:** 64 GB
   - **New Network:** VM Network
   - **"Connected" checkbox:** Checked
   - **New CD/DVD Drive:** Datastore ISO File
10. In the window that pops up, select the Ubuntu 18.04.3 Desktop x86_64 ISO file you previously uploaded.
11. Ensure the Connected checkbox is checked.
12. Click Add New Device → PCI Device.
13. Under New PCI Device, ensure NVIDIA GRID vGPU is selected.
14. In the NVIDIA GRID vGPU Profile drop-down menu, select grid_a100-4c.
15. Click Next.
16. Review details, and click Finish.
17. Wait for the Create virtual machine task to complete.

Installing Ubuntu 18.04 Desktop x86_64

1. In the Hosts and Virtual Machines sidebar menu, locate the newly created virtual machine.
2. Right-click the virtual machine, click Power, and click Power On.
3. Open the VMware virtual console.
4. Wait for the system to boot to the live installer desktop.
5. Click Install Ubuntu.
6. For the keyboard layout, select English (US), and click Continue.
7. For Updates and other software, perform the following steps:
    a. Select the Minimal installation radio button.
    b. Ensure the Download updates... checkbox is checked.
    c. Ensure the Install third-party... checkbox is not checked.
8. Click Continue.
9. For Installation type, perform the following steps:
    a. Select the Erase disk and install Ubuntu radio button.
    b. Ensure the Encrypt the new... checkbox is not checked.
    c. Ensure the Use LVM... checkbox is checked.
10. Click Install Now.
    Note: If a window pops up asking if you want to write changes to disks, click Continue.
11. In the window that asks for your location, select the New York time zone, and click Continue.
12. In the window that asks for your identity, fill in the following:
    - Your name: `ubuntu`
    - Your computer's name: `VDI-base-ubuntu`
    - Pick a username: `ubuntu`
    - Choose a password: `ubuntu`
    - Confirm your password: `ubuntu`
13. Click the radio button for Log in automatically.
14. Click Continue.
15. Wait for installation to complete, and click Restart Now.
16. After the VM restarts, in the Hosts and Virtual Machines sidebar menu, locate the VDI-base-ubuntu virtual machine.
17. Click the Summary tab.
18. Expand the VM Hardware pane.
19. To disconnect CD/DVD Drive 1, next to the plugs icon, click the drop-down, and click Disconnect.
20. Once the system has rebooted, you will receive a prompt to upgrade to Ubuntu 20.04. Decline to upgrade, and close the window.

Opening a terminal in VMware remote console

1. Using VMware Remote Console, connect to the VM.
2. Click the quick launch menu.
3. In the search box, type `Terminal`
4. Click the Terminal application icon.

Enabling password-less sudo

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Edit the sudoers file:

```
visudo
```

4. Add the following line to the end of the file.
   Note: Use a tab (i.e., not spaces) between ubuntu and ALL.

```
ubuntu	ALL=(ALL) NOPASSWD:ALL
```

5. To save the file, press CTRL+O, and press Enter.
6. To exit, press CTRL+X.

Updating Ubuntu package cache

1. Open a terminal.
2. Update APT's cache:

```
sudo apt-get update -y
```

3. Close the terminal.

Disabling the Gnome Welcome screen

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Modify XDG autostart file to disable first login autostart script:

```
sed -ri 's/^(Exec=.*)$/#\1/' /etc/xdg/autostart/gnome-initial-setup-first-login.desktop
```

Disabling Ubuntu Firewall

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Disable the firewall:

```
ufw disable
```

4. Close the terminal, and restart the VM.

Disabling IPv6

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Modify sysctl settings to disable IPv6:

```
sysctl -w net.ipv6.conf.all.disable_ipv6=1
sysctl -w net.ipv6.conf.default.disable_ipv6=1
```

4. Open a text editor for /etc/default/grub:

```
nano /etc/default/grub
```

5. To disable IPv6 in the default grub command, change the line:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
```

   to read as follows:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash ipv6.disable=1"
```

6. To disable IPv6 in the grub command, change the line:

```
GRUB_CMDLINE_LINUX=""
```

   to read

```
GRUB_CMDLINE_LINUX="ipv6.disable=1"
```

7. To save, press CTRL+O.
8. To exit, press CTRL+X.
9. Update grub with the new configuration:

```
update-grub
```

10. Reboot:

```
reboot
```

Disabling the Ubuntu lock screen

1. Connect to the VM using VMware Remote Console.
2. Click the quick launch menu.
3. In the search box, type `Settings`
4. Click the Settings application icon.
5. Click Privacy.
6. Click Screen Lock.
7. Click the Automatic Screen Lock slider button, and ensure it is in the Off position.
8. Close the application.

Disabling Ubuntu automatic updates

1. Open a terminal.
2. Become root:
   ```
   sudo su
   ```
3. Reconfigure the unattended-upgrades package:
   ```
   dpkg-reconfigure unattended-upgrades
   ```
4. Click or select No.
5. Press Enter.

Configuring VM networking

1. Open a terminal.
2. Become root:
   ```
   sudo su
   ```
3. Modify dhclient.conf to remove superseding DNS servers added by VMware tools:
   ```
   sed -ri -e '/supersede domain-name .*/d' -e '/supersede domain-name-servers .*/d' \
       /etc/dhcp/dhclient.conf
   ```
4. Open a text editor to modify the host's network plan:
   ```
   nano /etc/netplan/01-network-manager-all.yaml
   ```
5. Edit the file so that it matches the following (be sure to replace the adapter name, active directory domain, and server IP address with appropriate values for your deployment):
   ```
   ---
       network:
         version:             2
         renderer:            networkd
         ethernets:
           <ETHERNET ADAPTER NAME>:
             dhcp4:           true
             link-local:      []
             dhcp-identifier: mac
             nameservers:
               search:        [ <AD Domain NAME> ]
               addresses:     [ <AD Server IP> ]
   ```
6. Generate and apply the network plan:
   ```
   netplan generate
   netplan apply
   ```

Determining the virtual machine's IP Address

1. Open a terminal.
2. Become root:
   ```
   sudo su
   ```
3. List IP addresses (the IP address of instance is theone on the public/VM network):
   ```
   ip addr show Close the terminal.
   ```

Configuring /etc/hosts

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Add entries to /etc/hosts:

```
sed -ri \
  -e 's/^127\.0\.0\.1\s.*/127.0.0.1\tlocalhost/' \
  -e 's/^127\.0\.1\.1\s.*/127.0.1.1\t<FQDN>\t<HOSTNAME>/' \
  -e 's/^::1\s+.*/::1\t<FQDN>\t<HOSTNAME>/' \
    /etc/hosts
echo -e "<AD_SERVER_IP> <AD_FQDN>" >> /etc/hosts
echo -e "<HORIZON_VIEW_SERVER_IP> < HORIZON_VIEW_SERVER_FQDN>" >> /etc/hosts
```

Note: make sure to replace FQDN and HOSTNAME for the VM, the AD server, and the View Server with values appropriate to your deployment.

Installing Utilities

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Install system packages:

```
apt-get install -y wget curl htop git tree jq
```

Installing Python

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Update APT cache:

```
apt-get update -y
```

4. Install system packages for Python 2:

```
apt-get install -y python python-dev python-virtualenv python-pip
```

5. Add the deadsnakes PPA:

```
apt-add-repository ppa:deadsnakes/ppa
```

6. Install system packages for Python 3:

```
apt-get install -y software-properties-common
apt-get update -y
apt-get install -y python3.7 python3.7-dev python3.7-venv python3-pip
```

Adding the VM to Active Directory

1. Open a terminal.
2. Become root

```
sudo su
```

3. Install prerequisites:

```
apt-get update -y
apt-get install -y \
  krb5-user \
  winbind \
  samba \
  tdb-tools \
  libnss-winbind \
  libpam-winbind \
  libwbclient0
```

4. Disable IPv6 remote desktop in VMware tools:

```
nano /etc/vmware-tools/tools.conf
```

5. In the section for guestinfo, add or modify the max-ipv6-routes value to be 0. For example:

```
[guestinfo]
    max-ipv6-routes=0
```

6. Save and close the file.
7. Configure Kerberos:

```
nano /etc/krb5.conf
```

8. Edit the file so that it matches the following:

Note: be sure to replace domain names and Active Directory fully qualified domain name in bold.
Note: these entries are case sensitive; use the case as shown in bold.

```
[libdefaults]
    default_realm       =           MYDOMAIN.LOCAL
    clockskew           =           300
    ticket_lifetime     =           1d
    forwardable         =           true
    proxiable           =           true
    dns_lookup_realm    =           true
    dns_lookup_kdc      =           true
    krb4_config        =           /etc/krb.conf
    krb4_realms         =           /etc/krb.realms
    kdc_timesync        =           1
    ccache_type         =           4
[realms]
    MYDOMAIN.LOCAL         = {
        kdc             = ACTIVEDIRECTORY.mydomain.local
        admin_server    = ACTIVEDIRECTORY.abani.local
        default_domain  = MYDOMAIN.LOCAL
    }
[domain_realm]
    mydomain.local         = MYDOMAIN.LOCAL
    . mydomain.local       = MYDOMAIN.LOCAL
[appdefaults]
    pam                 = {
    ticket_lifetime     = 1d
    renew_lifetime      = 1d
    forwardable         = true
    proxiable           = false
    retain_after_close  = false
    minimum_uid         = 0
    debug               = false
```

9. Save and close the file.
10. Configure Samba:

```
nano /etc/samba/smb.conf
```

11. Make the file look like the following:

    Note: Be sure to replace domain names in bold.
    Note: These entries are case sensitive; use the case as shown in bold.

```
 [global]
   workgroup                  = MYDOMAIN
   realm                      = MYDOMAIN.LOCAL
   security                   = ADS
   encrypt passwords          = true
   socket options             = TCP_NODELAY
   domain master              = no
   local master               = no
   preferred master           = no
   os level                   = 0
   domain logons              = 0
   server string              = %h server (Samba, Ubuntu)
   dns proxy                  = no
   log file                  = /var/log/samba/log.%m
   max log size               = 1000
   panic action               = /usr/share/samba/panic-action %d
   server role                = standalone server
   passdb backend             = tdbsam
   obey pam restrictions      = yes
   unix password sync         = yes
   passwd program             = /usr/bin/passwd %u
   passwd chat                = *Enter\snew\s*\spassword:* %n\n *Retype\snew\s*\
    spassword:* %n\n *password\supdated\ssuccessfull* .
   pam password change        = yes
   map to guest               = bad user
   usershare allow guests     = yes
   idmap uid                  = 10000-20000
   idmap gid                  = 10000-20000
   winbind enum users         = yes
   template homedir           = /home/%D/%U
   template shell             = /bin/bash
   client use spnego          = yes
   client ntlmv2 auth         = yes
   encrypt passwords          = yes
   winbind use default domain = yes
   restrict anonymous         = 2
```

12. Configure nsswitch:

```
nano /etc/nsswitch.conf
```

13. Edit the file so that it matches the following (leave all instances of "myhostname" as is; do not replace the text with any other text.):

```
passwd: files winbind
group: files winbind
shadow: files winbind
gshadow:        files
hosts:          files mdns4_minimal [NOTFOUND=return] dns myhostname
networks:       files
protocols:      db files
services:       db files
ethers:         db files
rpc:            db files
netgroup:       nis
```

14. Save and close the file.

15. Open text editor to configurine nsswitch:

    ```
    nano /etc/nsswitch.conf
    ```

16. At the end of the file, add the following lines:
    Note: Be sure to replace domain names in bold. The entries are case sensitive; you must use the case as shown in bold.

    ```
    Administrator ALL=(ALL) NOPASSWD:ALL
    Administrator@mydomain.local ALL=(ALL) NOPASSWD:ALL
    MYDOMAIN.LOCAL\administrator ALL=(ALL) NOPASSWD:ALL
    %Administrators ALL=(ALL) NOPASSWD:ALL
    %Administrators@mydomain.local ALL=(ALL) NOPASSWD:ALL
    %MYDOMAIN.LOCAL\administrators ALL=(ALL) NOPASSWD:ALL
    %MYDOMAIN.LOCAL\view_planner_users ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_0 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_1 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_2 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_3 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_4 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_5 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_6 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_7 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_8 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_9 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_10 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_11 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_12 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_13 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_14 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_15 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_16 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_17 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_18 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_19 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_20 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_21 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_22 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_23 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_24 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_25 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_26 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_27 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_28 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_29 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_30 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_31 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_32 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_33 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_34 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_35 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_36 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_37 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_38 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_39 ALL=(ALL) NOPASSWD:ALL
    remote_desktop_wg_40 ALL=(ALL) NOPASSWD:ALL
    ```

17. Save and close the file.
18. Reboot the VM:

    ```
    reboot
    ```

19. Wait for the VM to reboot.
20. Join the user to the AD domain:

```
net ads join -U Administrator@mydomain.local
```

21. When prompted, enter the administrator's password.

Installing OpenSSH server

1. Open a terminal.
2. Become root:

```
sudo su
```

3. Install the OpenSSH SSH server package:

```
apt-get install -y openssh-server
```

4. Close the terminal.

Connecting to the virtual machine via SSH

1. Using SSH, configure a connection to the IP address of the VM.
2. Optionally, copy a key file. For openssh-client:

```
ssh-copy-id ubuntu@<IP ADDRESS>
```

3. Connect to the VM. For openssh-client:

```
ssh ubuntu@<IP ADDRESS>
```

Mounting CIFS Share with VMware Horizon 2103 and View Planner 4.6 installation media

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Install system packages for CIFS:

```
apt-get update -y
    apt-get install -y cifs-utils
```

4. Make mountpoint directory:

```
mkdir -p /media/installers
```

5. Mount the CIFS share:

```
mount                                            \
       -t cifs                                       \
       //HOSTNAME/path/to/install/media              \
       /media/installers                             \
       -o user=USER,password=PASSWORD,domain=DOMAIN
```

Note: Make the following replacements:

- Change `HOSTNAME` to the CIFS share host name
- Change `USER` to the CIFS share username (use guest for anonymous)
- Change `PASSWORD` to the CIFS share password (omit for anonymous)
- Change `DOMAIN` to the CIFS share domain name (omit for anonymous)
- Change `/path/to/install/media` to the path to the installation media within the CIFS share

Installing NVIDIA drivers

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Disable Gnome desktop, and ban nouveau drivers:

```
systemctl stop gdm
systemctl stop gdm3
systemctl disable --now gdm
systemctl disable --now gdm3
systemctl set-default multi-user.target
echo blacklist nouveau > /etc/modprobe.d/blacklist-nvidia-nouveau.conf
echo options nouveau modeset=0 >> /etc/modprobe.d/blacklist-nvidia-nouveau.conf
apt-get install -y build-essential linux-headers-$(uname -r) dkms pkg-config libglvnd-dev
```

4. Reboot:

```
reboot
```

5. Once the VM has rebooted, connect to the VM via SSH.
6. Become root:

```
sudo su
```

7. Create directories for NVIDIA installers:

```
mkdir -p /opt/nvidia
chmod 777 /opt/nvidia
```

8. Copy the NVIDIA vGPU zip file from the machine to which it was previously downloaded:

```
scp /local/path/to/NVIDIA-GRID-vSphere-7.0-460.73.02-460.73.01-462.31.zip ubuntu@<VM
    IP>:/opt/nvidia/.
```

Note: Replace `/local/path/to/` with the path to the folder containing the NVIDIA vGPU driver zip file.
Note: Replace `<VM IP>` with the IP address of the VDI-base-ubuntu virtual machine.

9. Run the installer:

```
cd /opt/nvidia
unzip ./NVIDIA-GRID-vSphere-7.0-460.73.02-460.73.01-462.31.zip
sh ./NVIDIA-Linux-x86_64-460.73.01-grid.run
```

Note: You will need to disregard a few error messages as we describe in the following steps.

10. The following error message should appear: "The distribution-provided pre-install script failed! Are you sure you want to continue?" Select Continue installation, and press Enter.
11. The installer will indicate a mismatch between the kernel compiler and current compiler with the following message: "The kernel was built with gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1), but the current compiler version is cc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0." When this happens, select Ignore CC version check, and press Enter.
12. The installer will prompt you to install NVIDIA's 32-bit compatibility libraries. Select Yes, and press Enter.
13. Reboot the VM:

```
reboot
```

14. Once the VM has rebooted, connect to the VM via SSH.
15. Become root:

```
sudo su
```

16. List loaded kernel modules with nvidia in the name:

```
lsmod | grep nvidia
```

Note: You should see the following:

```
nvidia                        NNN   MMM
```

where NNN and MMM are non-zero numbers.

17. Run nvidia-smi to report GPU statistics:

```
nvidia-smi
```

Note: You should see output similar to the following:

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.73.01    Driver Version: 460.73.01    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GRID A100-4C        On   | 00000000:02:00.0 Off |                    0 |
| N/A   N/A    P0    N/A /  N/A |    407MiB /  4091MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

18. Reconfigure the default systemd target:

```
systemctl set-default graphical.target.
```

19. Reboot the VM:

```
reboot
```

Installing Docker

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Update APT cache, install cURL, and install Docker:

```
apt-get update -y
apt-get install -y curl
curl https://get.docker.com | sh && sudo systemctl --now enable docker
```

4. Test Docker functionality:

```
docker run hello-world
```

Note: You should see the following:

```
Hello from Docker
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
     (amd64)
 3. The Docker daemon created a new container from that image which runs the
     executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
     to your terminal.

 To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash
```

```
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Installing NVIDIA Docker

1. Connect to the VM via SSH.
2. Become root:

   ```
   sudo su
   ```

3. Add NVIDIA APT repositories:

   ```
   distribution=$(. /etc/os-release;echo $ID$VERSION_ID) && \
       curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey |\
       apt-key add - &&\
       curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list |\
       tee /etc/apt/sources.list.d/nvidia-docker.list &&\
       curl -s -L https://nvidia.github.io/nvidia-container-runtime\
        /experimental/$distribution/nvidia-container-runtime.list |\
       tee /etc/apt/sources.list.d/nvidia-container-runtime.list
   ```

4. Update te APT cache and install NVIDIA Docker:

   ```
   apt-get update -y
   apt-get install -y nvidia-docker2
   ```

5. Restart Docker:

   ```
   systemctl restart docker
   ```

6. Test NVIDIA Docker:

   ```
   docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
   ```

   Note: You should see the following:

   ```
   +-----------------------------------------------------------------------------+
   | NVIDIA-SMI 460.73.01    Driver Version: 460.73.01    CUDA Version: 11.2     |
   |-------------------------------+----------------------+----------------------+
   | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
   | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
   |                               |                      |               MIG M. |
   |===============================+======================+======================|
   |   0  GRID A100-4C        On   | 00000000:02:00.0 Off |                    0 |
   | N/A   N/A    P0    N/A /  N/A |    407MiB /  4091MiB |      0%      Default |
   |                               |                      |             Disabled |
   +-------------------------------+----------------------+----------------------+

   +-----------------------------------------------------------------------------+
   | Processes:                                                                  |
   |  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
   |        ID   ID                                                   Usage      |
   |=============================================================================|
   |  No running processes found                                                 |
   +-----------------------------------------------------------------------------+
   ```

Configuring Docker DNS settings

1. Connect to the virtual machine via SSH.
2. Become root:

```
sudo su
```

3. Add DNS servers to docker's daemon.json configuration file:

```
cat /etc/docker/daemon.json | \
      jq '.dns = <DNS_SERVERS>' -eM | \
      jq '.dns-search = <DNS_SEARCH>' \
      > /tmp/daemon.json
    mv /tmp/daemon.json /etc/docker/daemon.json
    <DNS_SERVERS><DNS_SEARCH>
```

4. Restart Docker:

```
systemctl restart docker.service
```

Downloading and preprocessing the dataset

1. Follow the download and preprocessing instructions at the following URL: https://github.com/mlcommons/inference_results_v1.0/tree/master/closed/NVIDIA/code/resnet50/tensorrt.
2. Copy your preprocessed data to a new folder on the VM, such as /data/mlperf/resnet-dataset/.

Installing MLPerf Inference v1.0

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Clone the MLPerf Inference v1.0 results repository:

```
mkdir -p /opt/mlperf-inference-v1.0/{code,user}
cd /opt/mlperf-inference-v1.0
git clone --depth=1 https://github.com/mlcommons/inference_results_v1.0.git ./code
cd code
```

4. Patch the makefile to avoid trying to invoke a missing script:

```
sed -ri '#\t@python3 scripts/print_harness_result.py.*#d' closed/DellEMC/Makefile
```

Defining custom system for MLPerf Inference v1.0.

1. Connect to the virtual machine via SSH.
2. Become root:

```
sudo su
```

3. Edit the known systems list:

```
nano /opt/mlperf-inference-v1.0/code/closed/DellEMC/code/common/system_list.py
```

4. At the bottom of the file, just a few lines below the line starting with `class KnownSystems`, remove all of the declarations for other systems (lines of the form `XXX=SystemClass(…)`), and add a new system class definition:

```
R750XA_GRID_A100_4C = SystemClass("R750xa_GRID-A100-4C", ["GRID A100-4C"], [],
    Architecture.Ampere, [1] )
```

5. To save the file, press Ctrl+O, and press Enter.
6. To exit, press Ctrl+X.

Defining custom test configurations for MLPerf Inference v1.0.

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Edit the server configurations file:

```
nano /opt/mlperf-inference-v1.0/code/closed/DellEMC/configs/resnet50/Server/config.json
```

4. Edit the file so that it matches the following:

```
{
  "R750xa_GRID-A100-4Cx1": {
    "config_ver": {},
    "use_cuda_thread_per_device": true,
    "use_graphs": true,
    "gpu_batch_size": 64,
    "gpu_copy_streams": 4,
    "gpu_inference_streams": 4,
    "server_target_qps": 2350,
    "deque_timeout_usec": "4000"
  }
  "benchmark": "resnet50",
  "default": {
    "active_sms": 100,
    "input_dtype": "int8",
    "input_format": "linear",
    "map_path": "data_maps/dataset/val_map.txt",
    "precision": "int8",
    "tensor_path": "${PREPROCESSED_DATA_DIR}/dataset/ResNet50/int8_linear",
    "use_deque_limit": true
  },
  "scenario": "Server",
}
```

5. To save the file, press Ctrl+O, and press Enter.
6. To exit, press Ctrl+X.

Installing MLPerf Inference v1.0 Utility script

1. Connect to the virtual machine via SSH.
2. Become root:

```
sudo su
```

3. Create a folder for the utility script:

```
mkdir -p /opt/mlperf-runnerEdit the utility script:
nano /opt/mlperf-runner/run-mlperf-container.sh
```

4. Edit the file so that it matches the following (Note: be sure to replace DNS server IP addresses and search domains):

```
#!/bin/bash
#! --------------------------------------------------------
#!         !!! Require root privileges !!!
#! --------------------------------------------------------
if [[ $UID -ne 0 ]]; then
  echo "Script started as ''whoami'' instead of 'root'. "
  echo "Restarting script as 'root' user using sudo."
  sudo bash $0 "$@"
  exit $?
fi

echo "SCRIPT:      $0"
echo "ARGUMENTS:   $@"
echo "WORKING DIR: 'pwd'"
echo "RUNNING AS:  ''whoami''"
echo "ENVIRONMENT:"
env | egrep -v '^(PATH|LS_COLORS)' | sed 's/^/  | /' | column -ntxs '='
echo "PATH"
echo "$PATH" | sed 's/:/\n/g' | sed 's/^/  | /'
```

```
# -------------------------------------------------------
# Argument initialization
# -------------------------------------------------------
IMAGE_NAME=
CONTAINER_NAME=
CONTAINER_HOSTNAME=
OUTPUT_IMAGE=
declare -a EXTRA_ARGS
declare -a TEST_EXTRA_ARGS
declare -a CMD

# outer directories (base)
DATA_DIR=/data/mlperf/resnet-dataset
CODE_DIR=/opt/mlperf-inference-v1.0/code
USER_DIR=/opt/mlperf-inference-v1.0/user
LOG_DIR=/var/log/mlperf

# -------------------------------------------------------
# Error handling
# -------------------------------------------------------
function errexit(){
  echo "Error: $@" 1>&2
  echo "Aborting..." 1>&2
  exit 1
}

# -------------------------------------------------------
# Parse Arguments
# -------------------------------------------------------
while [ $# -gt 0 ]; do
  arg="$1"
  shift
  case "$arg" in
    -i|--image)        IMAGE_NAME=$1;         shift ;;
    -h|--hostname)     CONTAINER_HOSTNAME=$1; shift ;;
    -n|--name)         CONTAINER_NAME=$1;     shift ;;
    -o|--output-image) OUTPUT_IMAGE=$1;       shift ;;
    -d|--data-dir)     DATA_DIR=$1;           shift ;;
    -c|--code-dir)     CODE_DIR=$1;           shift ;;
    -u|--user-dir)     USER_DIR=$1;           shift ;;
    --log-dir)         LOG_DIR="$1"           shift ;;
    -x|--extra-args)
      # Consume remaining arguments up to a '--' argument which resumes normal parsing.
      while [ $# -gt 0 ]; do
        xarg="$1"
        shift
        if [ "$xarg" == "--" ]; then
          break
        else
          EXTRA_ARGS+=("${xarg}")
        fi
      done
    ;;
    -t|--test-args)
      # Consume remaining arguments up to a '--' argument which resumes normal parsing.
      while [ $# -gt 0 ]; do
```

```
        arg="$1"
        shift
        if [ "$arg" == "--" ]; then
          break
        else
          TEST_EXTRA_ARGS+=("${arg}")
        fi
      done
    ;;
    --|--cmd)
      # start of command. Consume remaining arguments
      while [ $# -gt 0 ]; do
        CMD+=("${1}")
        shift
      done
    ;;
    -b|--build)
      CMD=(bash -c 'export DEBIAN_FRONTEND=noninteractive && apt-get install -y tree jq htop &&
    make download_model BENCHMARKS=resnet50 && make build && make generate_engines RUN_ARGS="--
    benchmarks=resnet50 --scenarios=Offline,Server --config_ver=default"')
        ;;
    --run-server)       RUN=yes; SCENARIO=Server        ;;
    --run-offline)      RUN=yes; SCENARIO=Offline       ;;
    --bash)
                        CMD=(bash);
                        EXTRA_ARGS+=("-it")
                        CONTAINER_HOSTNAME=mlperf-bash
                        CONTAINER_NAME=mlperf-bash
                        OUTPUT_IMAGE=
    ;;
    --remove|--rm)      EXTRA_ARGS+=("--rm"); OUTPUT_IMAGE= ;;
    *)
      # unrecognized option, must be start of command. Consume remaining arguments
      CMD+=("${arg}")
      while [ $# -gt 0 ]; do
        CMD+=("${1}")
        shift
      done
    ;;
  esac
done

LOG_DIR_INNER=/mlperf-logs
mkdir -p "${LOG_DIR}"

if [ "$RUN" == yes ]; then
  CMD=(python3 code/main.py --benchmarks=resnet50 \
    --scenarios=$SCENARIO --config_ver=default \
    --test_mode=PerformanceOnly --action=run_harness \
    --log_dir=${LOG_DIR_INNER} \
  )
  CMD+=( "${TEST_EXTRA_ARGS[@]}" )
  EXTRA_ARGS+=("-e" "PREPROCESSED_DATA_DIR=/scratch/preprocessed_data" "--rm")
  CONTAINER_HOSTNAME=mlperf
  CONTAINER_NAME=mlperf
  OUTPUT_IMAGE=
fi
```

```
# -------------------------------------------------------
# Argument checks and defaults
# -------------------------------------------------------
[ -z "${CMD}" ] && errexit "Missing command! Note: Specify with --cmd X Y Z, -- X Y Z, or just X Y Z
at the end of all other options."
if [ -z "${IMAGE_NAME}" ]; then IMAGE_NAME=mlperf-inference:dell-latest; fi
if [ -z "${CONTAINER_NAME}" ]; then CONTAINER_NAME=${IMAGE_NAME/:*/}; fi
if [ -z "${CONTAINER_HOSTNAME}" ]; then CONTAINER_HOSTNAME=${CONTAINER_NAME}; fi

[ ! -d "${DATA_DIR}" ] && errexit "Missing data directory! Note: Specify with --data-dir DIRECTORY."
[ ! -d "${CODE_DIR}" ] && errexit "Missing code directory! Note: Specify with --code-dir DIRECTORY."
[ ! -d "${USER_DIR}" ] && errexit "Missing user directory! Note: Specify with --user-dir DIRECTORY."

# -------------------------------------------------------
# Derived Arguments and constants
# -------------------------------------------------------
# outer directories (subdirs)
WORK_DIR=${CODE_DIR}/closed/DellEMC
MAPS_DIR=${CODE_DIR}/closed/NVIDIA/data_maps/imagenet

# inner directories
USER_DIR_INNER=/mnt/user
WORK_DIR_INNER=/work
MAPS_DIR_INNER=${WORK_DIR_INNER}/data_maps/imagenet
SCRATCH_DIR_INNER=/scratch
DATA_DIR_INNER=${SCRATCH_DIR_INNER}/preprocessed_data

# -------------------------------------------------------
# Derived Docker invocation arguments
# -------------------------------------------------------
VOLUME_ARGS="-v ${LOG_DIR}:${LOG_DIR_INNER} -v ${DATA_DIR}:${DATA_DIR_INNER} -v ${WORK_DIR}:${WORK
DIR_INNER} -v ${USER_DIR}:${USER_DIR_INNER} -v ${MAPS_DIR}/cal_map.txt:${MAPS_DIR_INNER}/cal_map
txt:ro -v ${MAPS_DIR}/val_map.txt:${MAPS_DIR_INNER}/val_map.txt:ro -v /etc/timezone:/etc/timezone:ro
-v /etc/localtime:/etc/localtime:ro"
GPU_ARGS="--gpus=all"
SECURITY_ARGS="--security-opt apparmor=unconfined --security-opt seccomp=unconfined
--cap-add SYS_ADMIN"
ENVIRONMENT_ARGS="-w ${WORK_DIR_INNER} -e MLPERF_SCRATCH_PATH=${SCRATCH_DIR_INNER} -e NVIDIA_MIG
CONFIG_DEVICES=all"
DEVICE_ARGS="--device /dev/fuse"
HOST_ARGS="-h ${CONTAINER_HOSTNAME} --add-host ${CONTAINER_HOSTNAME}:127.0.0.1"
NET_ARGS="--net host"
DNS_ARGS="--dns <AD_SERVER_IP> --dns <ORG_DNS_SERVER_IP> --dns-search <AD_DOMAIN> --dns-search <ORG
DNS SEARCH DOMAIN>"
CONTAINER_ARGS="--name ${CONTAINER_NAME}"

function run(){
  echo "RUNNING COMMAND: $@"
  "${@}"
}

# -------------------------------------------------------
# Docker run function
# -------------------------------------------------------
function run_container(){
  run                \
  docker run          \
  ${VOLUME_ARGS}      \
  ${GPU_ARGS}         \
  ${SECURITY_ARGS}    \
  ${ENVIRONMENT_ARGS} \
  ${DEVICE_ARGS}      \
  ${HOST_ARGS}        \
  ${NET_ARGS}         \
  ${DNS_ARGS}         \
  ${CONTAINER_ARGS}   \
  "${EXTRA_ARGS[@]}"  \
  ${IMAGE_NAME} \
  "${CMD[@]}"
}

# -------------------------------------------------------
# Docker commit function
# -------------------------------------------------------
function commit_container(){
  if [ -z "${OUTPUT_IMAGE}" ]; then
    echo "Not committing container to image as no output was specified."
  else
    run docker stop "${CONTAINER_NAME}";
    run docker commit "${CONTAINER_NAME}" "${OUTPUT_IMAGE}" &&
    run docker rm "${CONTAINER_NAME}"
  fi
}

# -------------------------------------------------------
# main entry point
# -------------------------------------------------------
# Run the specified command in the specified mlperf-based
# container, optionally commiting the container as a new image
run_container && commit_container || errexit "Failed to run container or commit container to image."
```

5. To save the file, press CTRL+O, and press Enter.
6. To exit, press CTRL+X.

Building MLperf Inference v1.0

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Invoke the prebuild process:

```
cd /opt/mlperf-inference-v1.0/code
make prebuild
```

4. Wait for the prebuild to complete.
5. Build the MLPerf container:

```
cd /opt/mlperf-inference-v1.0/code/closed/DellEMC
    /opt/mlperf-runner/run-mlperf-container.sh \
      -i mlperf-inference:dell-latest \
      -h mlperf-inference-userv1.0 \
      -n mlperf-inference-user \
      -o mlperf-inference-v1.0-dellemc \
      --build
```

6. Wait for the container to finish building.

Installing VMware Tools

Mounting the VMware Tools ISO

1. In the Hosts and Virtual Machines sidebar menu, locate the newly created VM.
2. Click the Summary tab.
3. In the Actions drop-down menu, click Guest OS, and click Install VMware Tools...
4. Click Mount.

Extracting and installing VMware Tools

1. Connect to the VM via SSH.
2. Mount the VMware Tools ISO, extract and run the installer:

```
mkdir /media/vmware-tools
mount /dev/sr0 /media/vmware-tools
mkdir /opt/vmware-tools
cd /opt/vmware-tools
cp /media/vmware-tools/VMwareTools-10.3.23-17030940.tar.gz .
tar -xvf ./VMwareTools-10.3.23-17030940.tar.gz
cd vmware-tools-distrib
./vmware-install.pl
yes
```

3. For all remaining prompts, to accept the defaults, press Enter.

Installing the VMware Horizon 2103 agent

1. Locate the installation media for VMware Horizon within the CIFS share.
2. Connect to the VM via SSH.
3. Become root:

```
sudo su
```

4. Make a folder for the Horizon 2103 agent installer:

```
mkdir /opt/horizon-agent
    cd /opt/horizon-agent
```

5. Copy the Horizon 2103 agent installer to the virtual machine
Note: replace `/path/to/installers` **with the relevant folder on your CIFS/Samba share:**

```
cp /media/installers/path/to/installers/VMware-horizonagent-linux-x86_64-2103-8.2.0-17771892.tar.gz \
    /opt/horizon-agent/.
```

6. Extract the installer:

```
tar -xvf VMware-horizonagent-linux-x86_64-2103-8.2.0-17771892.tar.gz
```

7. Run the installer:

```
cd VMware-horizonagent-linux-x86_64-2103-8.2.0-17771892
apt-get -y install python-dbus python-gobject
./install_viewagent.sh
```

8. To confirm that you want to install the agent, press Enter.

Installing the VMware View Planner agent

1. Locate the installation media for VMware View Planner Agent within the CIFS share.
2. Connect to the virtual machine via SSH.
3. Become root:

```
sudo su
```

4. Make a folder for the VMware View Planner 4.6 Agent installer:

```
mkdir /opt/view-planner-agent
    cd /opt/view-planner-agent
```

5. Copy the VMware View Planner 4.6 Agent installer to the virtual machine.
Note: replace `/path/to/installers` **with the relevant folder on your CIFS/Samba share:**

```
cp /media/installers/path/to/installers/viewplanner-linux-agent-c-4.6.0.0-16995088.tar.gz /opt\
    /view-planner-agent/.
```

6. Extract and run the installer:

```
tar -xvf viewplanner-linux-agent-c-4.6.0.0-16995088.tar.gz
cd agent
./linux_agent_setup.sh
```

7. When prompted, enter the IP address for the View Planner harness VM, and press Enter.

Installing Google Chrome (optional)

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Make a folder for the Google Chrome installer:

```
mkdir /opt/google-chrome
    cd /opt/google-chrome
```

4. Download the Google Chrome installer:

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

5. Update APT, install prerequisites, and install Google Chrome:

```
apt-get update -y
apt-get install -y xz-utils
dpkg -I google-chrome-stable_current_amd64.deb
```

Installing the View Planner MLPerf Inference v1.0 custom workload

1. Connect to the VM via SSH.
2. Become root:

```
sudo su
```

3. Create a new workload directory:

```
plugins_home=~/ vp_agent/workload/vdi_user/
plugin_dir=${ plugins_home }/vp_mlperf_inference_1.0
mkdir -p $plugin_dir
cd $plugin_dirCreate the workload configuration file:nano workload.config
```

4. Edit the configuration file so that it matches the following:

Note: replace `/path/to/network/share` **(on line 36) with the mountpoint to a CIFS/Samba share which is persistently mounted in /etc/fstab. This will hold some debugging log files.**

```
[DEFAULT]
NAME             = vp_mlperf_inference
VERSION          = 1.0
EXECUTABLE_NAME  = python workload_script.py
EMBEDDED_PYTHON  = 1
TIMEOUT_SEC      = 7200
THINK_TIME_SEC   = 5
TYPE             = 0
OPERATING_SYSTEM = LINUX
DETAILS          = "MLPerf inference v1.0 workload"
SHARED_FOLDER    = /path/to/network/share
LAUNCHER_DIR     = /opt/mlperf-runner
LAUNCHER_SCRIPT  = run-mlperf-container.sh
LAUNCHER_ARGS    = -i mlperf-inference-v1.0-dellemc:latest --run-offlineCreate the workload python
    script:nano workload_script.py
```

5. Edit the script so that it matches the following:

   Note: replace `/path/to/network/share` **(on line 36) with the mountpoint to a CIFS/Samba share which is persistently mounted in /etc/fstab. This will hold some debugging log files.**

```
##########################################################################
# Copyright 2020 Principled Technologies, Inc.  All rights reserved.
# Name: workload_script.py
# Author : cwolfe@principledtechnologies.com
# Description : This is a custom workload for view planner which invokes mlperf inference 1.0
##########################################################################

import sys
import os
import logging
import subprocess
import threading
import time
import socket
import shlex
import datetime
import signal
import traceback
from enum import Enum
from queue import Queue, Empty
from typing import Optional

__WLOAD_CODE_DIR__  = os.path.dirname(os.path.abspath(__file__))
__CODE_ROOT_DIR__   = os.path.abspath(os.path.join(__WLOAD_CODE_DIR__, '..', '..', '..'))
__SHARED_CODE_DIR__ = os.path.abspath(os.path.join(__CODE_ROOT_DIR__, 'shared'))
__GLOGS_DIR__       = os.path.abspath(os.path.join(__CODE_ROOT_DIR__, 'logs'))
__WLOGS_DIR__       = os.path.abspath(os.path.join(__WLOAD_CODE_DIR__, 'logs'))
sys.path.append( __SHARED_CODE_DIR__ )
sys.path.append( __WLOAD_CODE_DIR__ )
from workload_include import WorkloadHelper, operation_group,  SUCCESS, FAILURE
from configparser import SafeConfigParser
prevumask=os.umask(0o000)
try:
    logger = logging.getLogger("workload")
    logger.setLevel(logging.DEBUG)
    fh = logging.FileHandler('/path/to/network/share/debug/%s.log' % socket.getfqdn())
    fh.setLevel(logging.DEBUG)
    fh2 = logging.FileHandler('./MLPERF_WORKLOAD.log')
    fh2.setLevel(logging.DEBUG)
    ch = logging.StreamHandler()
    ch.setLevel(logging.ERROR)
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
    fh.setFormatter(formatter)
    fh2.setFormatter(formatter)
    ch.setFormatter(formatter)
    logger.addHandler(fh)
    logger.addHandler(fh2)
    logger.addHandler(ch)
    logger.info("Logger initialized...")

    logger.info("Parsing config ...")
    try:
```

```
    parser = SafeConfigParser()
    parser.read('workload.config')
    launcher_dir         = parser.get('DEFAULT', 'LAUNCHER_DIR')
    launcher_script      = parser.get('DEFAULT', 'LAUNCHER_SCRIPT')
    launcher_args        = parser.get('DEFAULT', 'LAUNCHER_ARGS')
    shared_folder        = parser.get('DEFAULT', 'SHARED_FOLDER')
    timeout_sec          = int( parser.get('DEFAULT', 'TIMEOUT_SEC') ) - 5
    current_directory    = os.getcwd()
    if timeout_sec < 1:
        timeout_sec = 1

except Exception as e:
    try:
        logger.exception('Exception while reading config file {}'.format(str(e)))
    finally:
        e = None
        del e

logger.info("Defining time functions ...")
def now() -> datetime.datetime

    return datetime.datetime.utcnow().replace( tzinfo=datetime.timezone.utc )

def dt(start:datetime.datetime, end:Optional[datetime.datetime]=None) -> None:
    if end is None:
        end = now()

    return (end - start).total_seconds()

def timestamp():
    return datetime.datetime.strftime(now(), "%d%b%y-%H%M%S.%f").upper()

logger.info("Defining globals ...")
hostname        = socket.getfqdn()
runtime         = timestamp()
host_dir        = os.path.join(shared_folder, hostname)
capture_dir     = os.path.join(host_dir, f'RUN-{ runtime }' )
#capture_dir     = __WLOGS_DIR__
launcher        = os.path.join(launcher_dir, launcher_script)
nvidiaSmiExeLoc = 'nvidia-smi'

logger.info("Initializing workload helper ...")
try:
    #initialize helper class
    workloadHelper = WorkloadHelper()
except Exception as e:
    logger.exception("Failed to initialize helper")
    sys.exit()

logger.info("Registering operations ..."

op_count = 1
op_details = [
    ["run_mlperf", operation_group.OP_GROUP_A, 1]
]
workloadHelper.register(op_details)
```

```
logger.info("Defining program class ...")
class Channel(Enum):
    """ Identifies the output stream of a child processes. """
    STDOUT = 0
    STDERR = 1

class Program(object):
    """ A child process. """

    def __str__(self):
        return ' '.join([shlex.quote(arg) for arg in self.args])

    def __repr__(self):
        return "Program(%s)" % self

    def __init__(
            self,
            args,
            capture_stdout = True,
            capture_stderr = True,
            display_stdout = True,
            display_stderr = True,
            timeout_seconds=None,
            listener=None,
            env = None,
            log_prefix = None,
            silent = True,
            copy_shell_env = True,
            logger = None
    ):
        self.logger           = logger if logger else logging.getLogger()
        self.args             = [str(arg) for arg in args]
        self.capture_stdout   = capture_stdout
        self.capture_stderr   = capture_stderr
        self.display_stdout   = display_stdout
        self.display_stderr   = display_stderr
        self.timeout          = timeout_seconds
        self.listener         = listener
        self.env              = env
        self.log_prefix      = log_prefix
        self.silent           = silent
        self._proc            = None
        self.return_code      = None
        self._output          = []
        self._stopped         = threading.Event()
        self._stderr_thread   = None
        self._stdout_thread   = None
        self._handler_thread  = None
        self._watchdog_thread = None
        self._queue           = Queue()
        self._T_start         = None
        self.timed_out        = threading.Event()

        if (env is not None) and copy_shell_env:
            for k in [
                'LANG',
                'LANGUAGE',
                'LC_ALL',
                'DISPLAY',
                'LS_COLORS',
                'HOST_TYPE',
                'USER',
                'HOME',
                'NAME',
                'SHELL',
                'DOCKER_HOST',
                'TERM',
                'PATH'
            ]:
                v = os.environ.get(k, None)
                if v is not None:
                    env[k] = v

    def _watchdog(self):
        if self.timeout is not None:
            while self._proc.poll() is None:
                t_now   = now()
                elapsed = dt(self._T_start, t_now)
                if elapsed > self.timeout:
                    self.logger.warning(
"Program failed to complete in %s seconds." % self.timeout)
                    self.timed_out.set()
                    self.terminate()
                    while self._proc.poll() is None:
                        time.sleep(0.01)
                else:
                    time.sleep(0.01)

    def _pipe_reader(self, queue, pipe, channel, capture, display):
        try:
            with pipe:
                for line in iter(pipe.readline, b''):
                    queue.put(
                        (channel, capture, display, line, now())
                    )
        finally:
            queue.put(
                (channel, capture, display, None, now())
            )

    def _handler(self, queue, stopped):
        prefix = (self.log_prefix + ' ') if self.log_prefix else ''
        while not stopped.isSet():
            try:
                (channel, capture, display, line, timestamp) = queue.get_nowait()
                if line is not None:
                    if isinstance(line, bytes):
                        line = line.decode('utf-8').rstrip()
                    if capture:
                        self._output.append((channel, line, timestamp))
                    if display:
                        if channel==Channel.STDOUT:
                            self.logger.info(prefix+line)
                        else:
                            self.logger.error(prefix+line)
                    if self.listener:
                        try:
                            self.listener(channel, line, timestamp)
                        except Exception: #pylint: disable=broad-except
                            self.logger.exception(
                                "Error during user-supplied message"
                                " handler. (IGNORED)"
                            )
                queue.task_done()

            except Empty:
                time.sleep(0.05)

            except Exception:
                self.logger.exception("Error processing program output")

    def start(self):
        if not self.silent:
            self.logger.info("Starting program: %s" % self)

        self._T_start = now()
        self._proc = subprocess.Popen(
            self.args,
            stdout             = subprocess.PIPE,
            stderr             = subprocess.PIPE,
            bufsize            = 1,
            close_fds          = False,
            env                = self.env,
            universal_newlines = False
        )
        if self.timeout:
            self.watchdog_thread = threading.Thread( target=self._watchdog )
            self.watchdog_thread.start()

        # start the handler thread
        self.handler_thread = threading.Thread(
            target=self._handler,
            args=[self._queue, self._stopped]
        )
        self.handler_thread.start()

        # start the reader threads
        self.stdout_thread  = threading.Thread(
            target=self._pipe_reader,
            args=[
                self._queue, self._proc.stdout, Channel.STDOUT,
                self.capture_stdout, self.display_stdout
            ]
        )
        self.stdout_thread.start()

        self.stderr_thread  = threading.Thread(
            target=self.pipe_reader,
            args=[
                self._queue, self._proc.stderr, Channel.STDERR,
                self.capture_stderr, self.display_stderr
                ]
            )
        self._stderr_thread.start()

    def wait(self)->int:
        self._stdout_thread.join()
        self._stderr_thread.join()
        self._queue.join()
        self._stopped.set()
        self._handler_thread.join()
        if self.timeout:
            self._watchdog_thread.join()
        self.return_code = self._proc.wait()
        if not self.silent:
            self.logger.info(
                "Program completed: %s --> %s" % (
                self, self.return_code
                )
            )
        return self.return_code

    def send_signal(self, sig):
        if not self.silent:
            self.logger.info("Terminating program: %s with signal %s" % (self, sig))
        self._proc.send_signal(sig)

    def terminate(self):
        if not self.silent:
            self.logger.info("Terminating program: %s" % self)
        self._proc.terminate()

    @property
    def is_finished(self):
        return self._proc.poll() is not None

    @property
    def records(self):
        for x in self._output:
            yield x

    @property
    def lines(self):
        for (_, line, __) in self._output

            yield line

    @property
    def stdout_lines(self):
        for (channel, line, _) in self._output:
            if channel == Channel.STDOUT:
                yield line

    @property
    def stderr_lines(self):
        for (channel, line, _) in self._output:
            if channel == Channel.STDERR:
                yield line

    @property
    def output(self):
        return '\n'.join(self.lines)

    @property
    def stdout(self):
        return '\n'.join(self.stdout_lines)

    @property
    def stderr(self):
        return '\n'.join(self.stderr_lines)

class MlPerfProgram(Program):
    def __init__( self ):
        super( MlPerfProgram, self ).__init__(
            [
                '/bin/bash',
                '-c',
                "%s %s --log-dir %s" % (launcher, launcher_args, capture_dir)
            ],
            capture_stdout   = False,
            capture_stderr   = False,
            display_stdout   = True,
            display_stderr   = True,
            timeout_seconds  = timeout_sec,
            listener         = None,
            env              = None,
            log_prefix      = None,
            silent           = False,
            copy_shell_env   = True,
            logger           = logging.getLogger('mlperf')
        )

class NVSMIProgram(Program):
    def __init__( self ):
        super( NVSMIProgram, self ).__init__(
            [
                nvidiaSmiExeLoc,
                "-l",
                "-q",
                "--display=MEMORY,UTILIZATION"
            ],
            capture_stdout   = False,
            capture_stderr   = False,
            display_stdout   = True,
            display_stderr   = True,
            timeout_seconds  = None,
            listener         = None,
            env              = None,
            log_prefix      = None,
            silent           = False,
            copy_shell_env   = True,
                logger          = logging.getLogger('nvsmi')
            )

    logger.info("Starting operations loop ...")
    overall_result = SUCCESS

    for counter in range(op_count):
        op = op_details[counter]
        op_name, op_group, op_taskcount = op
        logger.debug("counter %d, opcount %d"%(counter,op_count))
        workloadHelper.schedule()
        workloadHelper.startOpLog(op_details[counter][0])
        if (counter == 0):
            logger.debug("Starting MLperf run")
            try:

                os.makedirs(capture_dir, mode=777, exist_ok=True)

                nvsmi = NVSMIProgram()
                nvsmi.start()

                mlperf = MlPerfProgram()
                try:
                    mlperf.start()
                    mlperf_rc = mlperf.wait()
                    if mlperf_rc != 0:
                        logger.exception("MLPerf script returned error status: %s." % mlperf_rc)
                        workloadHelper.logError(op_
    name, "MLPerf script returned error status: %s." % mlperf_rc, FAILURE)
                        overall_result = FAILURE
                except Exception as e:
                    logger.exception("Error launching or waiting for MLPerf script: %s." % str(e))
                    workloadHelper.logError(op
name, "Error launching or waiting for MLPerf script: %s." % str(e), FAILURE)
                    overall_result = FAILURE
                finally:
                    del mlperf
                    mlperf=None

                nvsmi.send_signal(signal.SIGINT)
                nvsmi_rc = nvsmi.wait()
            except Exception as e:
                logger.exception("Unhandled exception when running MLPerf script or nvidia
smi: %s." % str(e))
                workloadHelper.logError(op
name, "Unhandled exception when running MLPerf script or nvidia-smi.", FAILURE)
                overall_result = FAILURE
            finally:
                del nvsmi
                nvsmi=None
        else:
            break

        # End watermark and log timestamp [DO NOT REMOVE]
        workloadHelper.endOpLog(op_details[counter][0])
    workloadHelper.finished(overall_result)
except Exception as e:
    logger.exception("Unhandled exception: %s" % str(e))
finally:
    logging.shutdown()
    os.umask(prevumask)
```

6. Set file and folder permissions for the custom workload:chown -R root:root $d

   ```
   chmod -R 777 $d
   ```

Finalizing the VDI base image

1. In vCenter, on the left pane, locate and right-click the VM.
2. Click Template.
3. Click Convert to template.
4. In the popup, click Okay.
5. Rename the template `mp-a1004cx1`

Creating a VMware Customization Specification for VDI Desktop virtual machines

1. In vCenter, in the drop-down menu, select Profiles and Policies.
2. In the left sidebar, click VM Customization Specifications.
3. Click +New.
4. For Name, type `a1004cx1`
5. For Target Guest OS, select Linux.
6. Click Next.
7. Select Use Virtual Machine Name.
8. For Domain Name, enter your Active Directory domain name.
9. Click Next.
10. Select your Area and Location, and ensure Hardware clock set to is set to UTC.
11. Click Next.
12. In the Customization script field, enter the following text.
    Note: make sure to replace domain name and administrator password suitable for your deployment.

```
#!/bin/bash
function errexit(){
    echo "ERROR: $@\nAborting..." 1>&2;
    exit 1
}
function join_ad(){
    kdestroy
    yes password | kinit Administrator@mydomain.local
    yes password | net ads leave -U Administrator@mydomain.local
    yes password | net ads join -U Administrator@mydomain.local
}
function autologin(){
    sed -r 's/^(#\s*)?AutomaticLoginEnable\s*=.*/AutomaticLoginEnable = True/' -i /etc
gdm3/custom.conf
    sed -r 's/^(#\s*)?AutomaticLogin\s*=.*/AutomaticLogin = ubuntu/' -i /etc/gdm3/custom.conf
}
function precust(){
    echo "Performing VM precustomization..."
    autologin
}
function postcust(){
    echo "Performing VM postcustomization..."
    join_ad; join_ad; usermod -a -G adm,sudo,dip,plugdev,lpadmin administrator; autologin
}
case "$1" in
```

```
        precustomization) precust; ;;
        postcustomization) postcust; ;;
        *) errexit "Unknown action: '$1'"; ;;
    esac
```

13. On the Network page, select Use standard network settings…, and click Next.
14. For DNS settings, enter your Active Directory DNS settings. Optionally, you may include your organizational DNS settings.
15. Click Next.
16. On the Ready to complete page, verify the settings match expectations, and click Finish.

Preparing the Ubuntu 18.04 VDI Client image

Follow the instructions for Preparing the Ubuntu 18.04 VDI Data Science Knowledge Worker image (on page 13), with the following changes:

1. When deploying, deploy to the Client VMware ESXi Server.
2. Name the resulting template `client`
3. Omit the following sections:
    - Install nVidia Drivers
    - Installing Docker
    - Installing NVIDIA Docker
    - Configuring Docker DNS settings
    - Downloading and preprocessing the dataset
    - Installing MLperf Inference 1.0
    - Defining custom system for MLPerf Inference 1.0
    - Defining custom test configurations for MLPerf Inference 1.0
    - Building MLperf Inference 1.0
    - Installing VMware Horizon 2103 agent

    Note: this will include deploying the View Planner agent, and the custom workload, but does not need to include MLperf and related software, nor Horizon Agent.

Creating a VMware Customization Specification for VDI Client virtual machines

1. Repeat the process in Creating a VMware Customization Specification for VDI Desktop virtual machines, but set the specification's name to `client`. Alternatively, click duplicate on the desktop customization specification and set the new name to `client`.

Defining the VDI virtual machine pool in VMware Horizon 2103 Connection Server

1. Open Horizon 2103 View Administrator.
2. Log in as an administrator.
3. Under Inventory, click Desktops, and click Add.
4. Select Automated Desktop Pool, and click Next.
5. Select Full Virtual Machines, and click Next.
6. Select Dedicated, and leave Enable Automatic Assignment checked. Click Next.
7. On the Storage Policy Management page, leave defaults, and click Next.
8. For the pool ID and display name, type `a1004cx1_x40` and click Next.
9. Under Naming Pattern, type `desktop`
10. Under Provision Machines, select All Machines Up-Front.
11. Under Desktop Pool Sizing, in the Maximum Machines field, type `40`
12. To advance to vCenter Settings, click Next.
13. To the right of the Template field, click Browse, and select mp-a1004cx1. Click Submit.
14. For VM Folder location, select a suitable folder to contain the provisioned desktop VMs, and click Submit.
15. For Host or Cluster, select the VMware ESXi server under test, and click Submit.
16. For Resource Pool, select the VMware ESXi server under test, and click Submit.
17. For Datastores, click browse. Select all four NVMe datastores, and click Submit.
18. Click Next.
19. On the Desktop Pool Settings page, leave defaults, and click Next.

20. On the Remote Display Protocol page, leave defaults, and click Next.
21. On the Advanced Storage Options page, leave defaults, and click Next.
22. Under Guest customization, select Use this customization specification.
23. In the customization specification panel, select a1004cx1.
24. Click Next.
25. On the Ready to Complete step, ensure that Entitle Users After Adding Pool is checked, and click Submit.
26. In the Add users page, click Add.
27. In the Find User or Group dialog, in the Name/User Name search field, type `ADMIN`
28. Click Find.
29. Select Administrator@mydomain.local and Administrators@mydomain.local.
30. Ensure all listed users are selected, and click OK.
31. Click OK.
32. Wait for the desktop pool to display. To check the status, navigate to the Desktop pool, and click the Machines tab. The pool is ready when all hosts indicate show as Available in the Status column (on all pages).

Defining the MLPerf Inference custom workload in View Planner

1. Log into the View Planner webpage.
2. In the sidebar, click WORK LOAD.
3. In the top-right of the WorkLoad page, click Add New.
4. For Name, type `vp_mlperf_inference`
5. For version, type `1.0`
6. Click Save.

Defining a custom Work Profile that includes MLPerf Inference in View Planner

1. Log into the View Planner webpage.
2. In the sidebar, click WORK PROFILE.
3. In the top-right of the Work Profile page, click Add New.
4. For Name, type `Ubuntu_mlperf`
5. In the workloads section, in the list on the left, click vp_mlperf_inference.
6. Hold CTRL, and click vp_MoveFiles.
7. Hold CTRL, and click vp_RandomFileGenerator.
8. Click Save.

Defining benchmark run profiles for VMware View Planner

Defining benchmark run profiles

1. Log into the View Planner run harness GUI as [runharness IP]/admin.
2. Log in with the following credentials:
    - username: `vmware`
    - password: `viewplanner`
3. In the left pane, click Run Profile.
4. In the top-right of the window, click Add New.
5. In the Name field, type `remote_desktop`
6. For Run Mode, Select remote.
7. For VM/Session count, enter `40`
8. Set Iteration Count to 2.
9. Leave Rampup Time blank, and Think Time at 5. Click Next.
10. For workgroup name, enter `remote_desktop_wg`
    Note: this must match the name of the numbered entries you added to /etc/sudoers when creating the VDI DSKW VM Template.
11. Select the Workprofile you created in the section Defining a custom Work Profile that includes MLPerf Inference in View Planner.
12. Leave Percent VM at 100%.
13. Set the display protocol to blast.
14. Under desktop type, select VDI
15. Next to Desktop VM, click the plus sign.

16. Enter in a prefix name that matches the prefix that the pool above uses.
17. From the Infraserver drop-down, select your vCenter server.
18. From the Vdiserver Name drop down, select your VMware Horizon 2103 Server. Click the check mark.
19. Next to Desktop VM, click the plus sign.
20. Enter in a prefix name that matches the prefix you used previously when deploying the client pool.
21. From the Infraserver drop-down, select your vCenter server.
22. Click the Checkmark button.
23. Click NEXT.
24. Review your profile, and click FINISH.

Running the VMware View Planner benchmark

1. Open a browser, and navigate to http:/<View Planner IP address>/vp-ui
2. Log in as vmware.
3. Click New run.
4. Select the remote_desktop run profile.
5. Enter a run name.
6. Click START.

Read the report at http://facts.pt/y8wl0M3

This project was commissioned by Dell Technologies.