VIRCL Project Documentation

Server Virtualization Systems

Technical report for virtualization systems





VMware	Proxmox	Shared storage
ESXi hosts, vCenter, HA/DRS cluster results	PVE hosts, GUI/CLI workflows, migration and backup results	CLOIF2 NAS / Unraid using NFS and iSCSI

Project title	Server Virtualization Systems
Team members	Andrea Girotto and Guilherme Videira Marques
Specialisation	Clustering
Hypervisors	VMware ESXi with vCenter and Proxmox VE
Servers	HP13, HP14, HP15, HP16
Storage backend	Unraid using NFS and iSCSI
Date	June 2026

Prepared by Andrea Girotto · Guilherme Videira Marques

Table of Contents

Table of Contents 2
Table of Figure 6
Project environment 9
1. Hypervisor deployment 10
Concept 10
VMware ESXi (+ vCenter) 10
Proxmox VE 11
Comparison 11
Impediments / lessons learned 12
Implementation walkthrough screenshots 12
2. VM creation via GUI 18
Concept 18
VMware ESXi (+ vCenter) 18
Proxmox VE 18
Comparison 18
Impediments / lessons learned 19
Implementation walkthrough screenshots 19
3. VM creation via CLI 24
Concept 24
VMware ESXi (+ vCenter) 24
Proxmox VE 24
Comparison 25
Impediments / lessons learned 25
Implementation walkthrough screenshots 26
4. Guest OS installation 29
Concept 29
VMware ESXi (+ vCenter) 29
Proxmox VE 29
Comparison 30
Impediments / lessons learned 30
Implementation walkthrough screenshots 31
5. VM resource reconfiguration 33
Concept 33

VMware ESXi (+ vCenter) 33
Proxmox VE 33
Comparison 33
Impediments / lessons learned 34
Implementation walkthrough screenshots 34
6. Cloning a VM 42
Concept 42
VMware ESXi (+ vCenter) 42
Proxmox VE 42
Comparison 42
Impediments / lessons learned 43
Implementation walkthrough screenshots 43
7. Exporting a VM 49
Concept 49
VMware ESXi (+ vCenter) 49
Proxmox VE 49
Comparison 50
Impediments / lessons learned 50
Implementation walkthrough screenshots 51
8. Hypervisor configuration backup 55
Concept 55
VMware ESXi (+ vCenter) 55
Proxmox VE 55
Comparison 56
Impediments / lessons learned 56
Implementation walkthrough screenshots 56
9. VM backup 58
Concept 58
VMware ESXi (+ vCenter) 58
Proxmox VE 58
Comparison 59
Impediments / lessons learned 59
Implementation walkthrough screenshots 60
10. Snapshots 65

Concept 65
VMware ESXi (+ vCenter) 65
Proxmox VE 65
Comparison 66
Impediments / lessons learned 66
Implementation walkthrough screenshots 66
11. Templates 70
Concept 70
VMware ESXi (+ vCenter) 70
Proxmox VE 70
Comparison 71
Impediments / lessons learned 71
Implementation walkthrough screenshots 72
12. Remote storage 76
Concept 76
Protocol decision 76
VMware ESXi (+ vCenter) 76
Proxmox VE 77
Comparison 78
Impediments / lessons learned 78
Implementation walkthrough screenshots 79
13. Live storage migration 83
Concept 83
VMware ESXi (+ vCenter) 83
Proxmox VE 83
Comparison 83
Impediments / lessons learned 84
Implementation walkthrough screenshots 84
14. Live hypervisor migration 88
Concept 88
VMware ESXi (+ vCenter) 88
Proxmox VE 88
Comparison 89
Impediments / lessons learned 89

Implementation walkthrough screenshots 90
15. Users and permissions 94
Concept 94
VMware ESXi (+ vCenter) 94
Proxmox VE 94
Comparison 95
Impediments / lessons learned 95
Implementation walkthrough screenshots 95
16. Hypervisor patching 99
Concept 99
VMware ESXi (+ vCenter) 99
Proxmox VE 99
Comparison 100
Impediments / lessons learned 100
Implementation walkthrough screenshots 101
17. Specialisation: Clustering 104
Requirement coverage 104
Architecture decision 105
Implementation summary 105
Theoretical concept 105
Implementation in VMware ESXi (+ vCenter) 106
Implementation in Proxmox VE 107
Do's and Don'ts 107
Implementation walkthrough screenshots 108
Personal conclusion 114
Gui 114
Andrea 114
Sources 116
Glossary 117

Table of Figure

Figure 1 - HP14 Proxmox host summary showing CPU, RAM, kernel, manager version, and storage usage. 12
Figure 2 - HP16 Proxmox host summary showing the same physical hardware class as HP14. 13
Figure 3 - VCSA Stage 2 complete....13
Figure 4 - Both hosts in cluster 14
Figure 5 - Management network configuration 14
Figure 6 - Proxmox web UI login....15
Figure 7 - Final LVM-Thin list....15
Figure 8 - Disk picker 15
Figure 9 - ESXi Host Client first login 16
Figure 10 - TCP/IP edit - final values before save 16
Figure 11 - Datastores result - `HP15-datastore2` created 17
Figure 12 - Windows 10 client VM running successfully in the vCenter console....19
Figure 13 - vCenter New Virtual Machine summary for the HP13 Windows 10 VM. 20
Figure 14 - TinyCore Linux VM booted successfully in the Proxmox noVNC console. 21
Figure 15 - `VM 100` summary - running 21
Figure 16 - `VM 100` summary - stopped, just created 22
Figure 17 - Create VM wizard - Disks tab, `vmdata` selected 22
Figure 18 - Create VM wizard - Confirm tab, final config....23
Figure 19 - Installed Linux guest OS booted successfully from the VM disk. 26
Figure 20 - Linux installer running inside `VM 101` through the Proxmox noVNC console. 26
Figure 21 - Proxmox `VM 101` created with the Linux guest profile before OS installation....27
Figure 22 - PowerCLI output confirming creation of the ESXi CLI VM....27
Figure 23 - PowerCLI session connected to vCenter for CLI-based VM creation. 27
Figure 24 - CLI-created VM visible in the vCenter inventory. 28
Figure 25 - Installed Linux guest OS booted successfully from the VM disk. 31
Figure 26 - Linux installer running inside `VM 101` through the Proxmox noVNC console. 31
Figure 27 - Proxmox `VM 101` created with the Linux guest profile before OS installation....32
Figure 28 - ESXi VM resource baseline before CPU, memory, disk, and network changes. 34
Figure 29 - ESXi VM hardware after CPU, memory, disk, and network reconfiguration. 35
Figure 30 - Guest OS verification of updated CPU and RAM values. 36
Figure 31 - Guest OS verification of the added virtual disk....37
Figure 32 - ESXi VM settings after temporary test resources were cleaned up. 38
Figure 33 - Proxmox `VM 101` booted successfully after resource changes. 39
Figure 34 - Proxmox `VM 101` after adding a disk and second network adapter. 39
Figure 35 - Linux guest verification after restoring the original resource profile....40
Figure 36 - Linux guest OS detecting both Proxmox virtual NICs. 40
Figure 37 - Proxmox `VM 101` baseline hardware before resource changes. 40
Figure 38 - Proxmox `VM 101` hardware after cleanup and restoration. 41
Figure 39 - ESXi source VM powered off before cloning. 43
Figure 40 - vCenter clone wizard summary before creating the ESXi clone. 44
Figure 41 - vCenter clone task completed successfully. 44
Figure 42 - Source VM and cloned VM visible in the vCenter inventory. 45
Figure 43 - Cloned ESXi VM booted successfully after cloning. 46

Figure 44 - Proxmox full-clone dialog for `VM 101`. 47
Figure 45 - Proxmox source VM stopped before clone creation. 47
Figure 46 - Proxmox clone task completed successfully. 48
Figure 47 - Proxmox cloned `VM 102` hardware visible after cloning. 48
Figure 48 - ESXi source VM powered off before export. 51
Figure 49 - vCenter export dialog configured for the ESXi VM. 52
Figure 50 - Downloaded ESXi OVF export files on the local machine. 52
Figure 51 - Proxmox backup dialog used to export `VM 102`. 53
Figure 52 - Proxmox export backup task completed successfully. 53
Figure 53 - Exported Proxmox backup archive visible in storage. 54
Figure 54 - Proxmox `VM 102` stopped before export. 54
Figure 55 - HP13 ESXi configuration backup command and generated download URL. 56
Figure 56 - HP13 ESXi configuration backup bundle downloaded. 57
Figure 57 - HP15 ESXi configuration backup command and generated download URL. 57
Figure 58 - HP15 ESXi configuration backup bundle downloaded. 57
Figure 59 - Extracted ESXi configuration backup bundle contents. 57
Figure 60 - ESXi source VM powered off before VM backup/export. 60
Figure 61 - vCenter export dialog used for the ESXi VM backup. 61
Figure 62 - Downloaded ESXi VM backup files. 61
Figure 63 - ESXi VM restore/deploy task completed successfully. 62
Figure 64 - Restored ESXi VM visible in the vCenter inventory. 62
Figure 65 - Restored ESXi VM booted successfully. 63
Figure 66 - Proxmox VM backup archive visible before restore. 63
Figure 67 - Proxmox restore dialog configured for new `VM 112`. 64
Figure 68 - Restored Proxmox `VM 112` visible after restore. 64
Figure 69 - vCenter snapshot creation task completed successfully. 66
Figure 70 - ESXi snapshot visible in Snapshot Manager. 67
Figure 71 - Test marker file created inside the guest after snapshot. 67
Figure 72 - vCenter snapshot revert task completed successfully. 67
Figure 73 - Guest marker file removed after reverting the ESXi snapshot. 68
Figure 74 - ESXi Snapshot Manager clean after snapshot deletion. 68
Figure 75 - Proxmox snapshot created and listed for `VM 102` 68
Figure 76 - Test marker created inside the Proxmox guest after snapshot. 69
Figure 77 - Guest marker removed after Proxmox snapshot rollback. 69
Figure 78 - ESXi VM template visible in the vCenter inventory. 72
Figure 79 - vCenter deployment from template completed successfully. 72
Figure 80 - VM deployed from ESXi template visible in inventory. 73
Figure 81 - VM deployed from ESXi template booted with hostname and IP. 73
Figure 82 - Proxmox template clone dialog for new `VM 111`. 74
Figure 83 - Proxmox menu option used to convert `VM 110` into a template. 74
Figure 84 - Proxmox template icon confirming `VM 110` became a template. 74
Figure 85 - Proxmox `VM 111` created successfully from template. 75
Figure 86 - CLOIF2 NFS datastore visible to both ESXi hosts. 79
Figure 87 - ESXi NFS write test proving datastore access. 79
Figure 88 - iSCSI LUN visible from the ESXi storage adapter. 79
Figure 89 - ESXi iSCSI datastore created successfully. 80

Figure 90 - Final Proxmox storage list with iSCSI and LVM entries. 80
Figure 91 - Proxmox iSCSI storage visible after configuration. 81
Figure 92 - CLOIF2 NFS storage visible in Proxmox. 81
Figure 93 - Unraid NFS share settings used for Proxmox access. 82
Figure 94 - ESXi VM running before live storage migration. 84
Figure 95 - ESXi storage migration task completed successfully. 84
Figure 96 - ESXi VM summary confirming the new datastore after migration. 85
Figure 97 - Proxmox `VM 111` disk location before storage migration. 85
Figure 98 - Proxmox Move Disk dialog with target storage selected. 86
Figure 99 - Proxmox storage migration task completed successfully. 86
Figure 100 - Proxmox `VM 111` still running after storage migration. 87
Figure 101 - ESXi VM running on HP13 before vMotion. 90
Figure 102 - vCenter vMotion task completed successfully. 90
Figure 103 - ESXi VM running on HP15 after vMotion. 91
Figure 104 - Proxmox cluster view with HP14 and HP16 before the nested node was added. 91
Figure 105 - Proxmox online migration dialog targeting HP16. 92
Figure 106 - Proxmox online migration task completed successfully. 92
Figure 107 - Proxmox `VM 111` running on HP16 after online migration. 93
Figure 108 - Shared NFS storage enabled for Proxmox migration. 93
Figure 109 - vCenter local user created for permission testing. 95
Figure 110 - vCenter permission assignment dialog for the test user. 96
Figure 111 - vCenter login test with the restricted Viewer user. 96
Figure 112 - Proxmox test user created for permission validation. 97
Figure 113 - Proxmox permission assignment created for the test user. 97
Figure 114 - Proxmox login test with the restricted viewer user. 97
Figure 115 - Proxmox correctly blocks a write action for the restricted user. 98
Figure 116 - HP13 ESXi version output captured for patching verification. 101
Figure 117 - HP15 ESXi version output captured for patching verification. 101
Figure 118 - vSphere Lifecycle Manager compliance result for ESXi patching. 102
Figure 119 - Available Proxmox updates shown before patching HP16. 102
Figure 120 - HP16 Proxmox upgrade completed and requested reboot. 103
Figure 121 - HP14 Proxmox upgrade completed and requested reboot. 103
Figure 122 - Nested ESXi VM running on HP14 Proxmox for the ESXi cluster design. 108
Figure 123 - vCenter cluster view showing the three ESXi hosts. 108
Figure 124 - vSphere HA enabled on the ESXi cluster. 109
Figure 125 - vSphere DRS configured in fully automated mode. 109
Figure 126 - Shared datastores visible in the ESXi cluster. 109
Figure 127 - Nested Proxmox installer summary with hostname, IP address, gateway, and DNS settings. 110
Figure 128 - Proxmox Datacenter tree showing HP14, HP16, and pve-nested in cluster `vircl-team4`. 110
Figure 129 - Proxmox `VM 100` hardware showing local disk and ISO before migration cleanup. 111
Figure 130 - Proxmox migration warning caused by attached local CD/DVD media. 111
Figure 131 - Proxmox storage list showing CLOIF2 iSCSI, LVM, and NFS entries. 111
Figure 132 - Proxmox cluster storage view showing shared NFS enabled across nodes. 112
Figure 133 - Proxmox online migration dialog after local CD/DVD media was removed. 112

Figure 134 - Proxmox HA page showing quorum OK before adding the HA resource. 112
Figure 135 - Proxmox dialog for adding `VM 100` as an HA-managed resource. 113
Figure 136 - Proxmox HA status showing `VM 100` as a started HA resource. 113
Figure 137 - Proxmox HA status with HP16 as active manager and `VM 100` still protected. 113

Project environment

Role	Server	Platform	Address / note
ESXi physical host	HP13	VMware ESXi + vCenter host	10.0.10.131, vCenter 10.0.10.132
Proxmox physical host	HP14	Proxmox VE	*10.0.10.141*
ESXi physical host	HP15	VMware ESXi	*10.0.10.151*
Proxmox physical host	HP16	Proxmox VE	*10.0.10.161*
Remote storage	Unraid	NFS and iSCSI	Used for shared storage, backups, and migration steps
Specialisation	Nested nodes	ESXi nested on HP14, Proxmox nested on HP13	Used to explain three-node clustering and quorum

1. Hypervisor deployment

This chapter covers the installation and initial configuration of both hypervisors across the project's four HP servers. Since all physical nodes share the same hardware class, one representative workflow is documented per platform — HP15 for ESXi, HP13 for vCenter/VCSA, and HP14 for Proxmox VE. Node-specific notes are included where the outcome differed, such as HP14's post-install network correction and the storage initialisation steps on HP14 and HP16.

Representative walkthrough screenshots and results in this chapter focus on:

- VMware ESXi installation and post-install configuration, using HP15 as the clean install example;
- vCenter/VCSA deployment and cluster creation, using the HP13 implementation;
- Proxmox VE installation and post-install configuration, using HP14 as the detailed example.

HP14 and HP16 both report the same physical Proxmox host profile in the supplied dashboard screenshots: 80 logical CPUs from Intel Xeon Gold 6138 processors, about 62.52 GiB usable RAM, Proxmox VE 9.2.3, and Linux kernel 7.0.6-2-pve. The nested Proxmox node is intentionally different and is documented under the clustering specialisation, not as a physical server.

Concept

Type-1 (bare-metal) hypervisors run directly on the server hardware and host multiple guest VMs. VMware ESXi is a closed-source commercial hypervisor; cluster-level management requires vCenter, which is licensed separately and deployed as a dedicated appliance (VCSA). Proxmox VE is open-source with no licence required — it is built on Debian/KVM with LXC container support and includes a built-in web UI that handles both single-node and cluster management without an additional management server. The install target is the same (a bare HP server), but the two installers, licensing models, network configuration steps, and post-install workflows differ significantly.

VMware ESXi (+ vCenter)

The ESXi installer ISO (~340 MB, VMware-VMvisor-Installer-8.0U2b-23305546.x86_64.iso) was sourced from the school NAS at \\\\nas.btslgk.lu and mounted via iLO 5 virtual media - no physical USB drive was needed. Both HP13 and HP15 booted from the iLO virtual CD-ROM by selecting the one-time boot menu (F11) at POST. The installer is a text/graphical hybrid driven from the DCUI. The target disk was the 223.54 GiB HPE Smart Array logical volume; the larger 1.09 TiB logical volume was left untouched for the **VMFS** datastore.

Network configuration is split across two separate steps: the management IP (/16 subnet mask, static gateway 10.0.0.1) is set in the DCUI (F2 -> **Configure Management Network** -> IPv4 Configuration), while DNS, search domain, and the TCP/IP stack default gateway source are corrected afterwards in the host web UI. One important quirk: the DCUI hostname field rejects dots - the short hostname (hp13, hp15) and the domain name (vircl.local) must be entered in two separate fields. After the first boot, a **VMFS** datastore was created on the 1.09 TiB logical volume using the ESXi host client (**Storage -> New Datastore -> VMFS**).

HP13 was designated as the vCenter host. The vCenter Server Appliance (VCSA 8.0, VMware-VCSA-all-8.0U3f-24322831.iso) was deployed in two stages from a Windows workstation. Stage 1 deploys the appliance VM onto HP13's local **VMFS**; Stage 2 runs the first-boot configuration wizard inside the

appliance. The PNID (Primary Network Identifier) was set to the vCenter IP address (10.0.10.132) rather than a hostname, because Stage 2 runs a DNS reverse-lookup against the PNID and our lab DNS was not fully populated at that point. After Stage 2 completed, the **VIRCL-Cluster** was created in vCenter and both HP13 and HP15 were added to it, enabling DRS (Fully Automated) and HA.

Proxmox VE

The Proxmox VE installer ISO (~1.3 GB, proxmox-ve_9.1-1.iso) was sourced from the same school NAS and mounted via iLO 5 virtual media, identical to the ESXi workflow. Proxmox uses a graphical Debian-based installer with a single FQDN field (unlike ESXi's split hostname/domain fields). The target disk selection must be done carefully: HPE Smart Array logical volumes appear as `/dev/sdX`, and the letter assignments change between the installer environment and the first post-reboot boot - the correct disk must be identified by size, not by device letter.

The subnet mask must be set to /16 during installation — the installer defaults to /24, which leaves the node unreachable outside its own /24 segment. The gateway (10.0.0.1) and DNS (10.0.0.62) must also be entered correctly at this stage.

Post-install steps were identical on both nodes: the enterprise subscription repository was disabled and replaced with the pve-no-subscription repository, then all pending updates were applied:

```
apt dist-upgrade
```

Storage layout: the 223.54 GiB logical volume holds the Proxmox OS (ext4); the 1.09 TiB logical volume was initialised as an LVM-thin pool (vmdata on HP14, hp16-vmstore on HP16) for VM disk allocation. On HP14, the larger disk retained a residual VMFS partition from a previous ESXi installation, which blocked pvcreate until the partition table was wiped with: `wipefs -a`

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Install media size	~340 MB	~1.3 GB
Installer style	Text/graphical DCUI; hostname and domain entered as two separate fields	Graphical Debian-based; single FQDN field
Default network mask	/24 - must be changed to /16 for the school lab	/24 - must be changed to /16 for the school lab
Post-install management	vSphere Client (web UI) per host; vCenter Server for cluster-level management	Built-in web UI at port 8006; no additional management server required
Time from boot to usable	~10 min for ESXi alone; +60 min for VCSA deployment and Stage 2 configuration	~15-20 min including repo switch and LVM-thin storage setup

Impediments / lessons learned

- Both installers default to /24, but the school lab is /16 - the subnet mask must be set to 255.255.0.0 during installation (or corrected immediately after), otherwise nodes on other 10.0.x.0/24 subnets are unreachable. Discovered on HP14 (Proxmox); applied correctly to HP15 (ESXi) and HP16 (Proxmox) from then on.
- ESXi hostname field rejects dots - the installer DCUI splits hostname and domain into two separate fields. Entering hp13.vircl.local in the hostname field fails silently or is truncated; the correct input is hp13 in the hostname field and vircl.local in the domain field.
- vCenter PNID must resolve in DNS - the Stage 2 configuration wizard validates the PNID (Primary Network Identifier) with a reverse DNS lookup. Using the vCenter IP address (10.0.10.132) as the PNID avoids this dependency when lab DNS is not yet fully populated.
- Default route source mismatch on HP15 - after switching HP15's management network from DHCP to static, outbound internet access broke because the default route still carried source DHCP instead of MANUAL. The fix is to manually re-add the default route: *`esxcli network ip route ipv4 add -n 0.0.0.0 -g 10.0.0.1`*
- Proxmox disk device letters shift on first reboot - during installation the OS disk may appear as /dev/sdc, but after the first reboot the HPE Smart Array re-enumerates the logical volumes and the letter changes. Always identify the target disk by size, not by `/dev/sdX` letter.
- Residual VMFS on HP14 blocked LVM-thin creation - the 1.09 TiB logical volume on HP14 retained a VMFS partition table from a previous ESXi group. pvcreate refused to initialise the device until the partition table was cleared with: *`wipefs -a`*

Implementation walkthrough screenshots

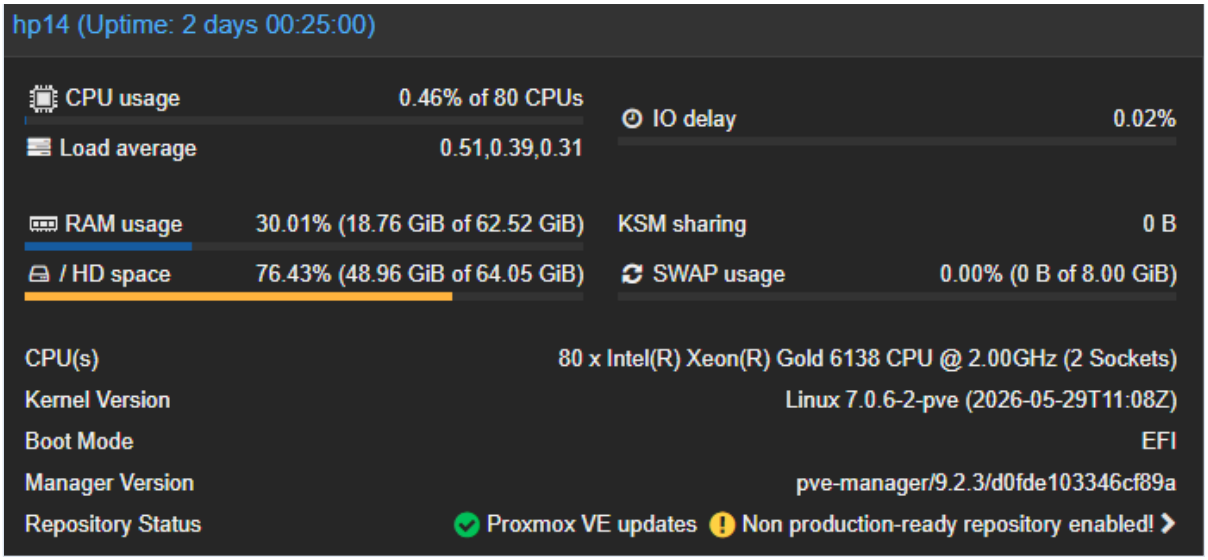


Figure 1 - HP14 Proxmox host summary showing CPU, RAM, kernel, manager version, and storage usage.

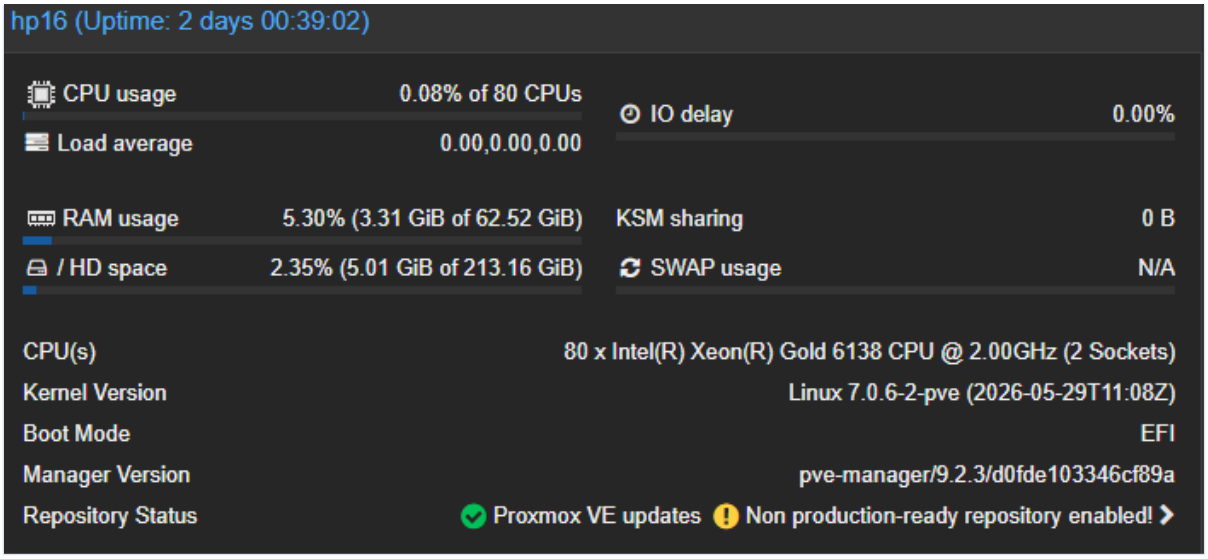


Figure 2 - HP16 Proxmox host summary showing the same physical hardware class as HP14.

Network Details	
Network configuration	Assign static IP address
IP version	IPv4
IP address	10.0.10.132
Subnet mask	16
Host name	vcenter.vircl.local
Gateway	10.0.0.1
DNS servers	10.0.0.62,10.0.0.1
vCenter Server Details	
Time synchronization mode	Synchronize time with the ESXi host
SSH access	Activated
SSO Details	
SSO Details	vsphere.local
Username	administrator
Customer Experience Improvement Program	
CEIP setting	Opted out

Figure 3 - VCSA Stage 2 complete

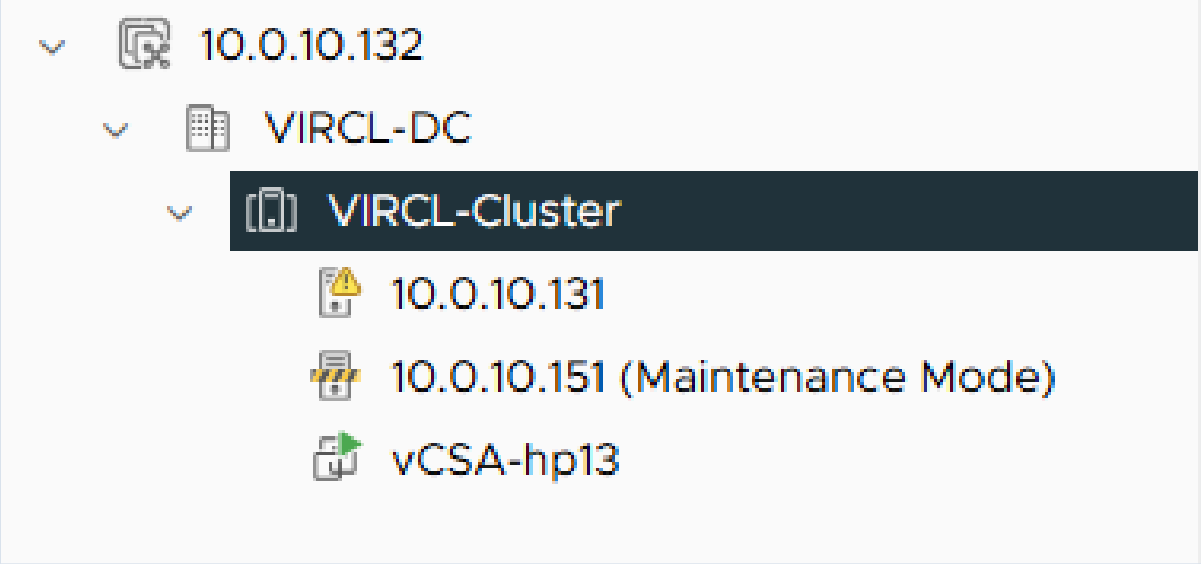


Figure 4 - Both hosts in cluster

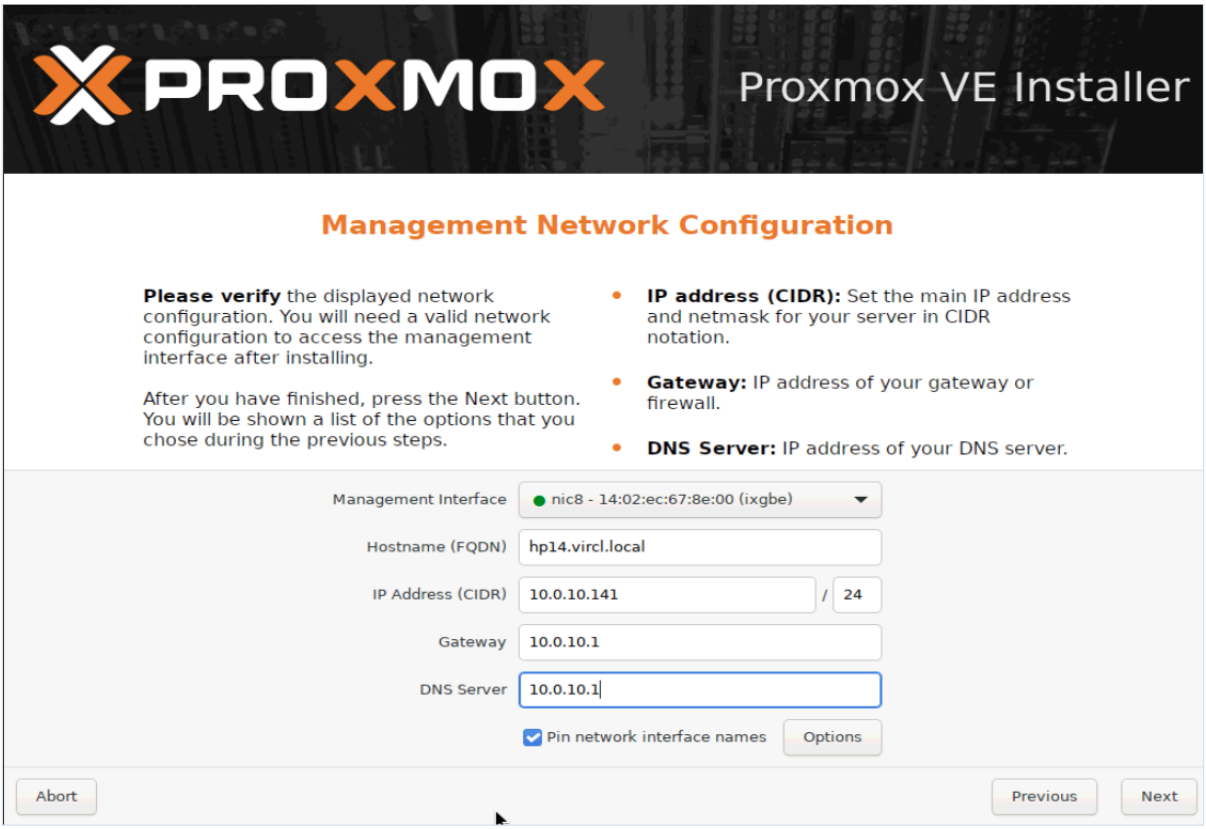


Figure 5 - Management network configuration

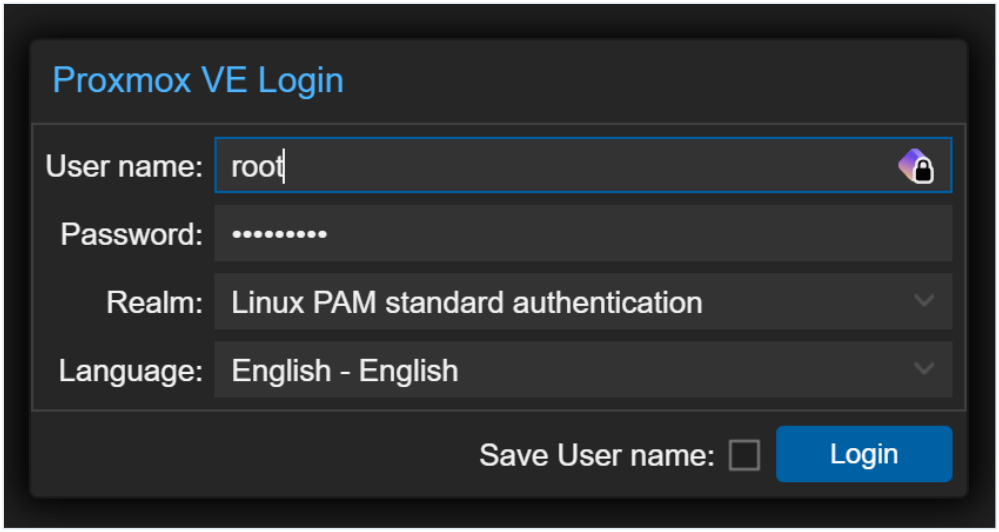


Figure 6 - Proxmox web UI login

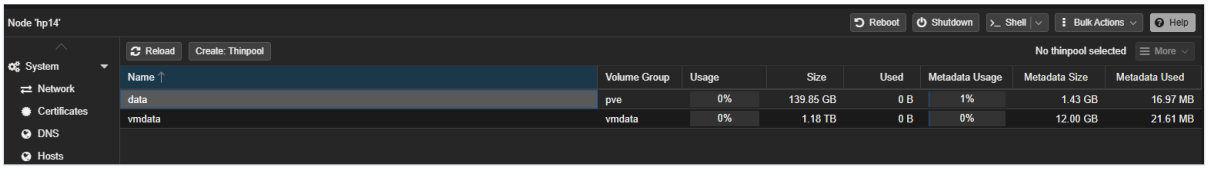


Figure 7 - Final LVM-Thin list

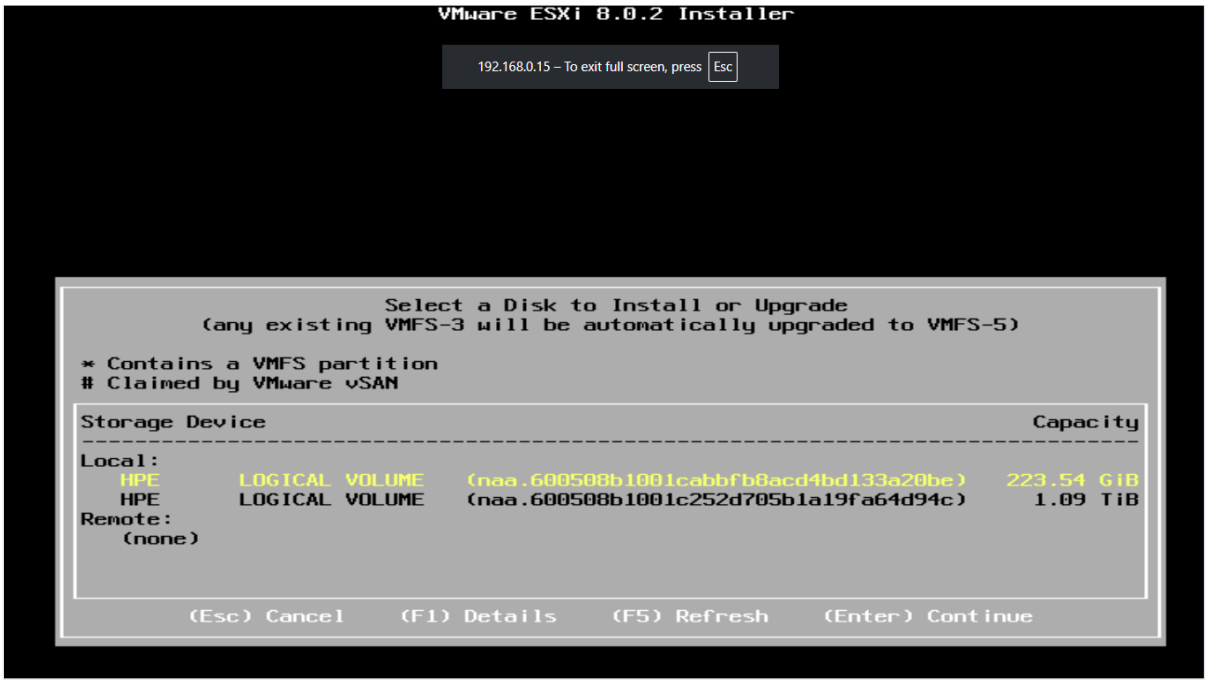


Figure 8 - Disk picker

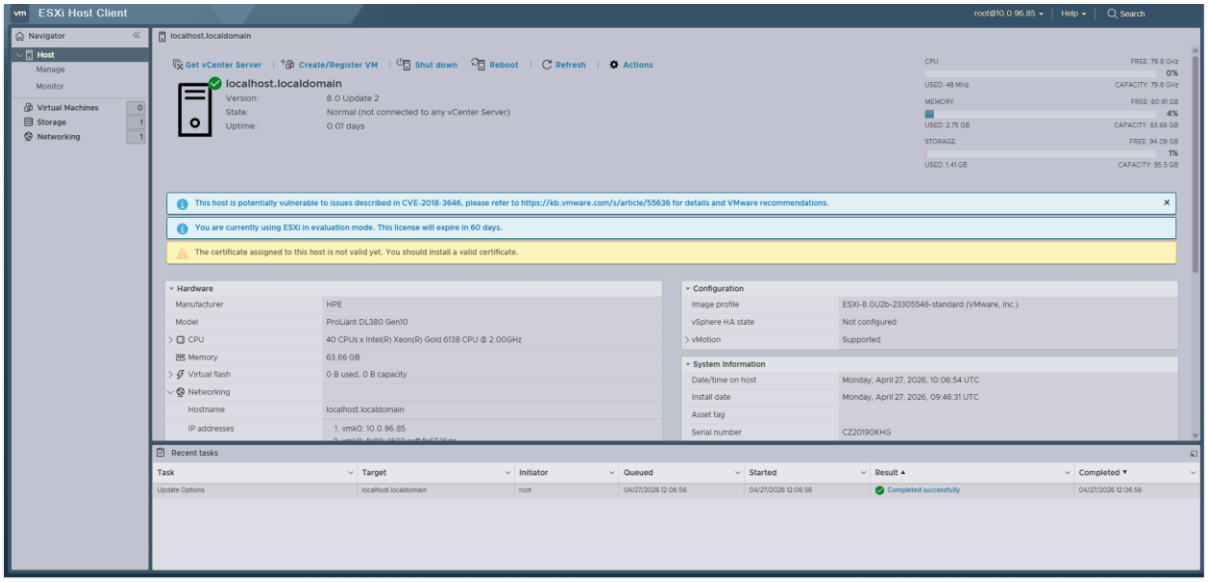


Figure 9 - ESXi Host Client first login

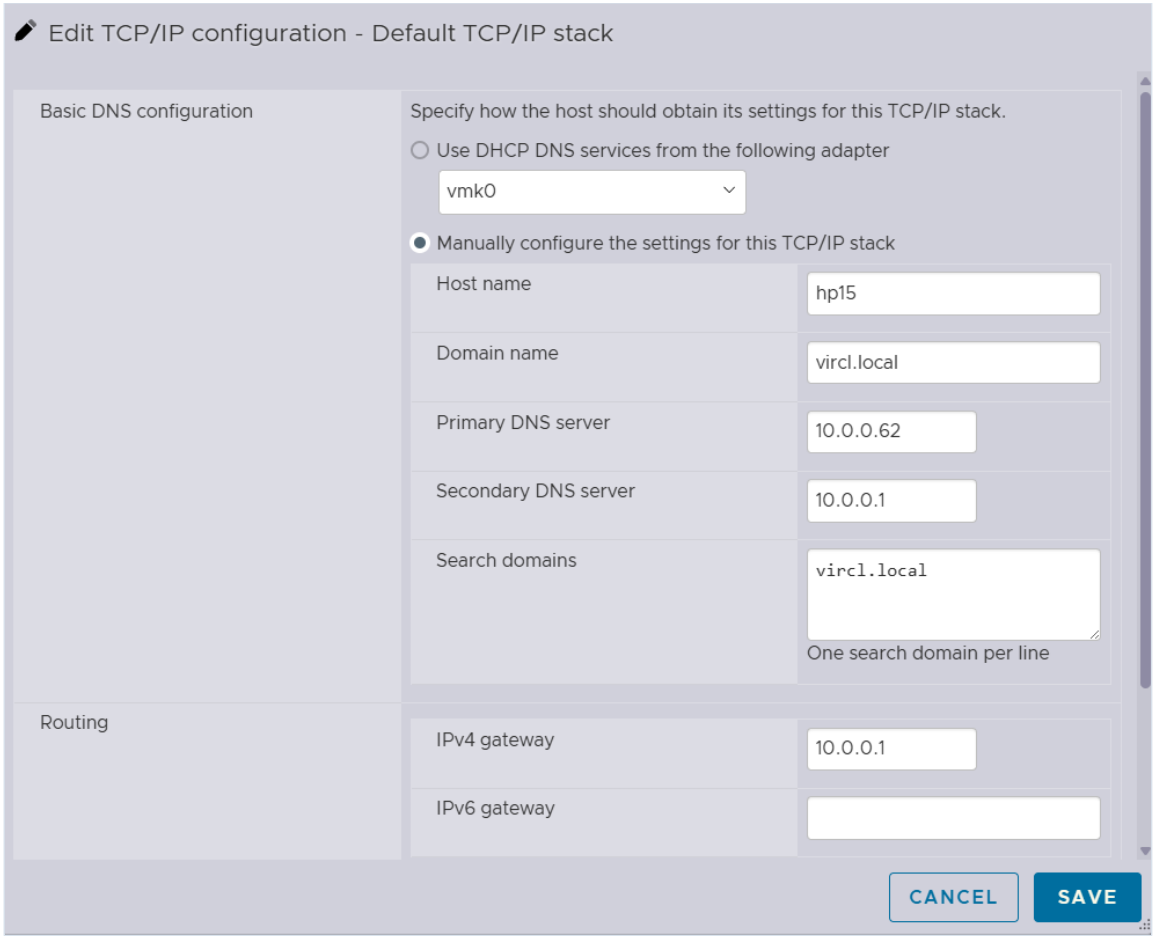


Figure 10 - TCP/IP edit - final values before save

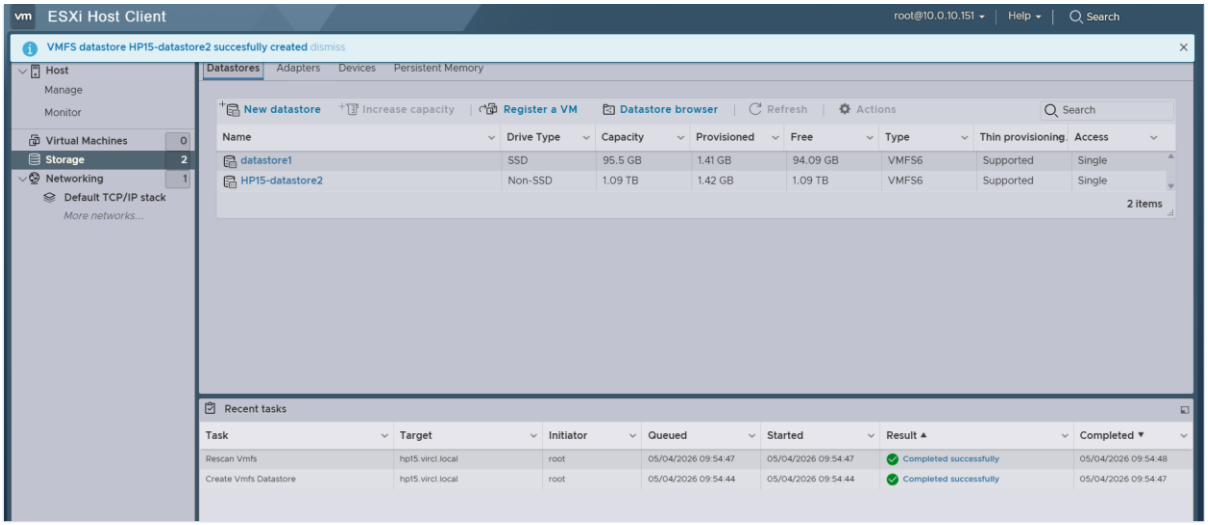


Figure 11 - Datastores result - `HP15-datastore2` created

2. VM creation via GUI

Concept

Creating a VM through the graphical interface proves the normal administrator workflow for each hypervisor. The GUI wizard makes the important VM design choices visible: target host, datastore/storage, guest OS type, CPU, memory, disk, network adapter, and installation ISO. This chapter compares how those choices are made in vCenter and in Proxmox VE.

VMware ESXi (+ vCenter)

The VMware side was completed through the vCenter web UI because HP13 and HP15 were managed by vCenter. VM `win10-vm-01` was created on HP15 through the vCenter New Virtual Machine wizard. A Windows 10 Enterprise Evaluation ISO was uploaded to datastore `HP15-datastore2`. The VM was configured with 2 vCPUs, 4 GB RAM, a 48 GB thin-provisioned disk, and an E1000E network adapter on the `VM Network` port group. The VM powered on successfully and the Windows installer loaded through the vCenter remote console.

Proxmox VE

The Proxmox side was completed on HP14 with the Proxmox web UI. A TinyCore Linux ISO was uploaded to local ISO storage and `VM 100` (`tiny-linux-vm-01`) was created with the Create VM wizard. The VM used the dedicated `vmdata` LVM-thin pool for its disk, VirtIO networking on vmbr0, and booted successfully through the noVNC console.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Management interface	vCenter vSphere Client	Proxmox VE web UI
ISO handling	ISO uploaded as a normal datastore file	ISO uploaded under storage content type iso
Storage selected for VM disk	VMFS datastore such as hp13-vmstore or HP15-datastore2	LVM-thin storage vmdata
Console access	vCenter remote console	Built-in noVNC console
Main advantage	Strong central management for clustered ESXi hosts	Fast integrated workflow on the node/cluster UI
Main caution	Direct write operations via the standalone host client may be blocked once the host is managed by vCenter — always create and modify VMs through the vCenter UI	The wizard may pre-select a storage pool that differs from the intended target — always verify disk placement on the Confirm tab before finishing

Impediments / lessons learned

- Attempting to create or edit a VM via the standalone ESXi host client after the host has been added to vCenter will fail or be blocked — use the vCenter UI exclusively once a host is cluster-managed.
- ISO handling differs between platforms: ESXi treats ISOs as regular datastore files, while Proxmox registers them under a dedicated ISO content type in storage. Uploading to the wrong location makes the ISO invisible in the wizard.
- In Proxmox, the wizard's final Confirm tab is important because it clearly shows whether the disk is going to the correct storage pool.

Implementation walkthrough screenshots

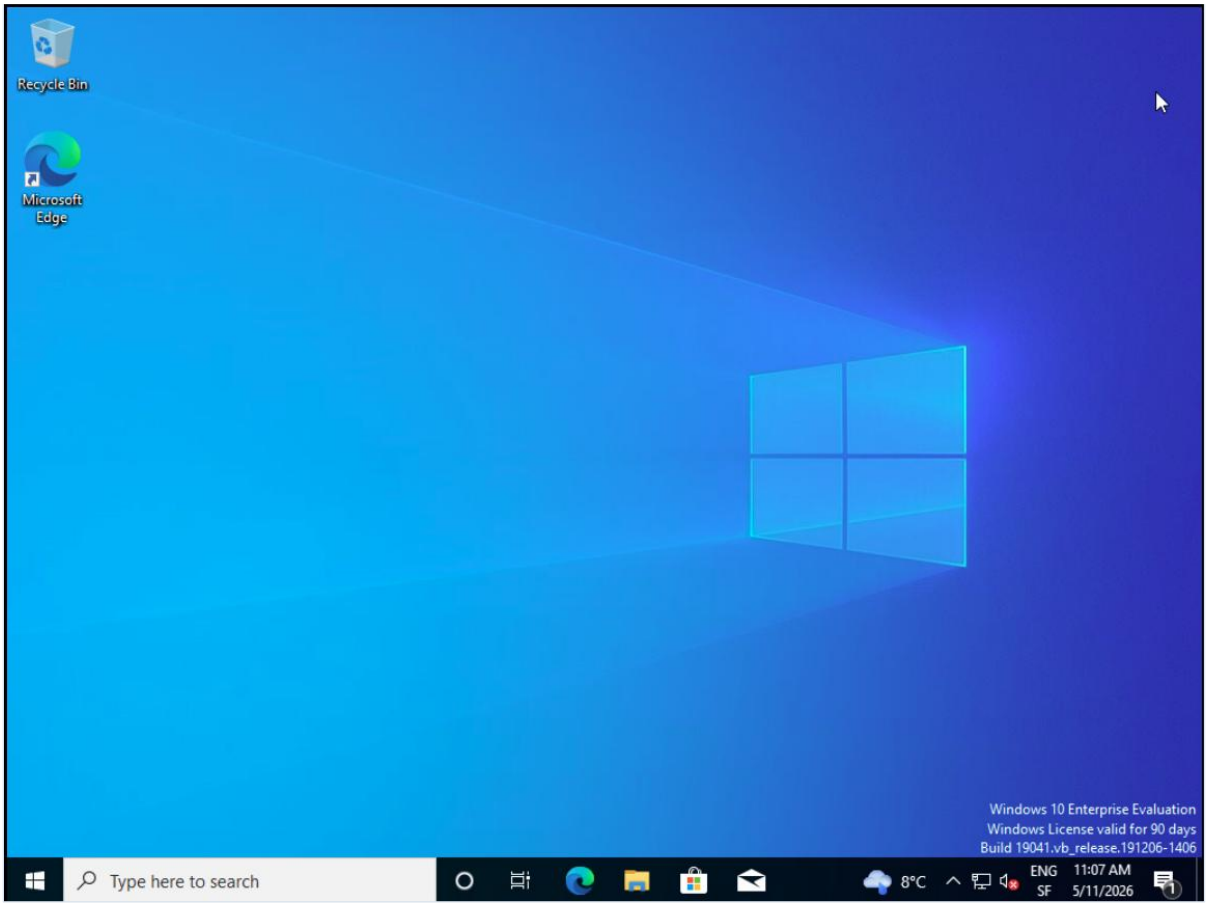


Figure 12 - Windows 10 client VM running successfully in the vCenter console.

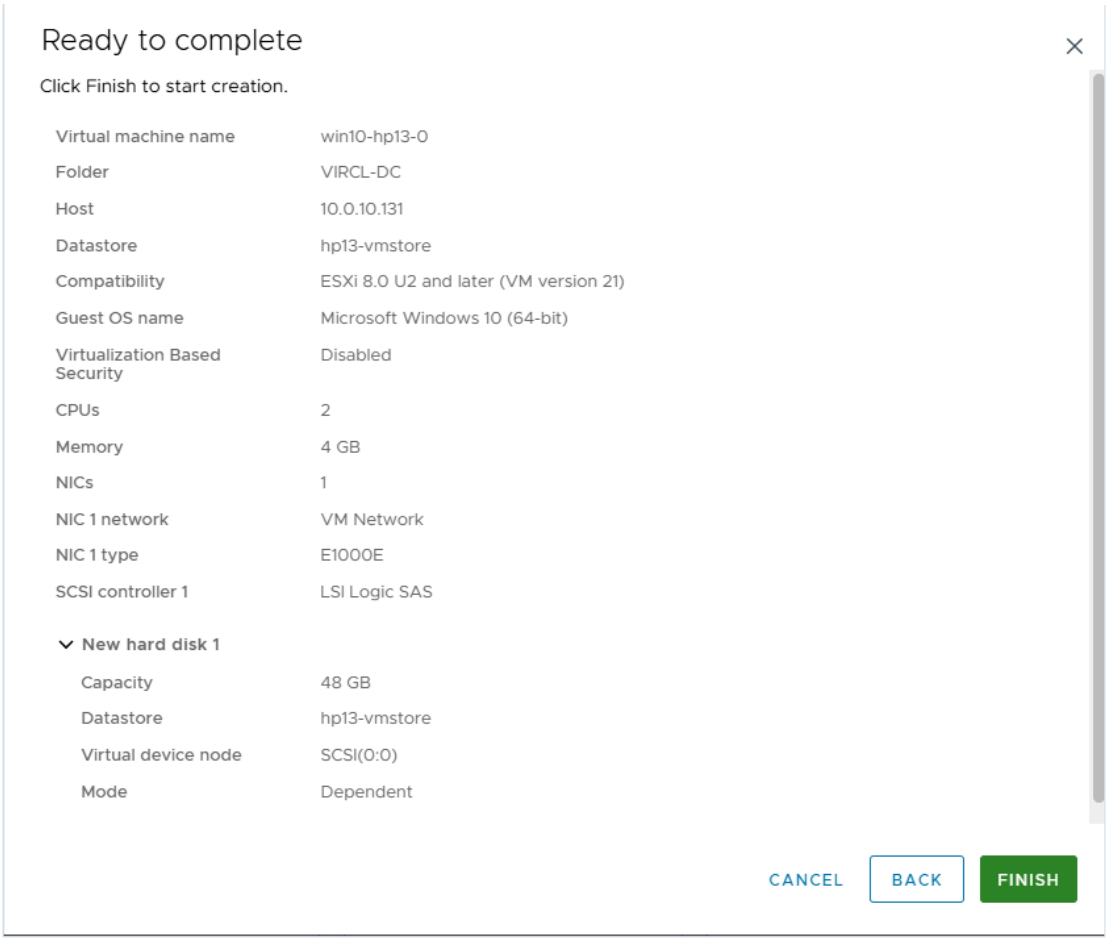


Figure 13 - vCenter New Virtual Machine summary for the HP13 Windows 10 VM.

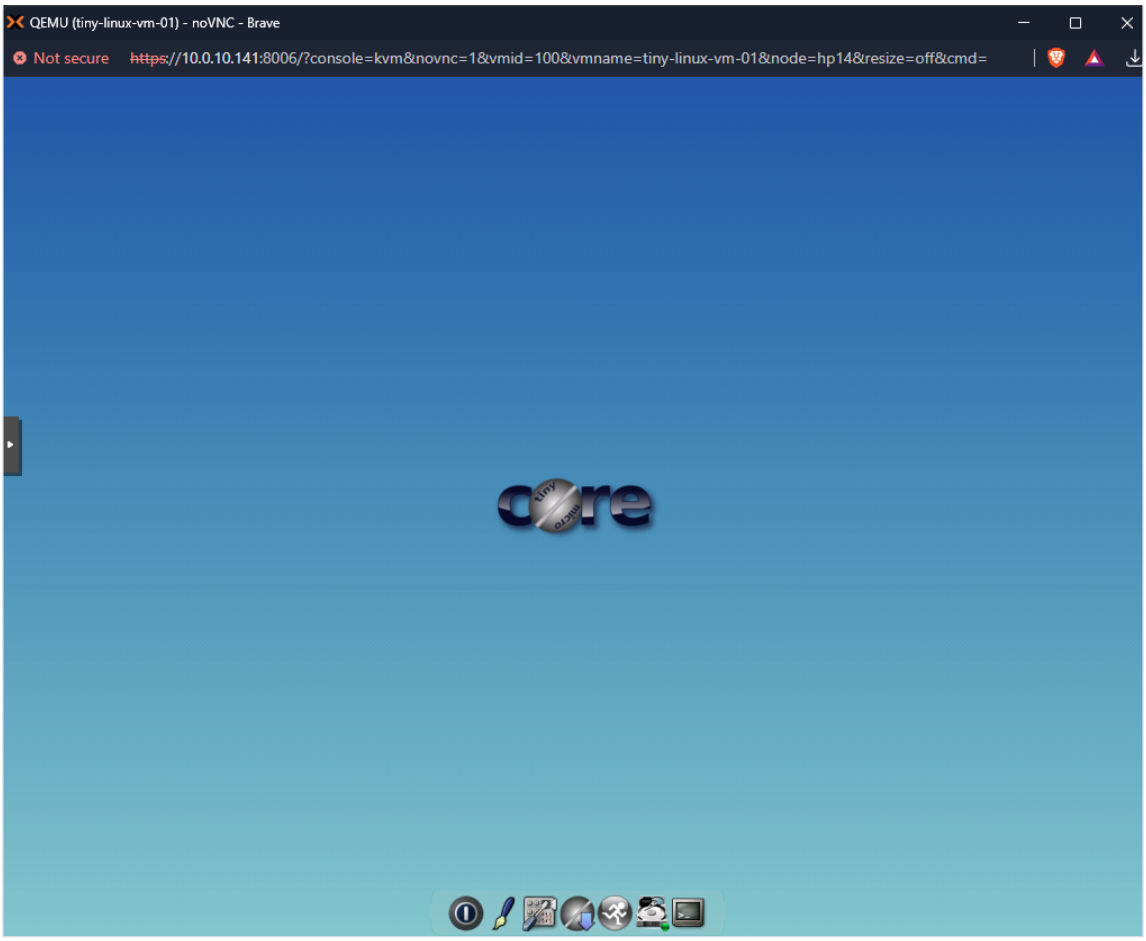


Figure 14 - TinyCore Linux VM booted successfully in the Proxmox noVNC console.

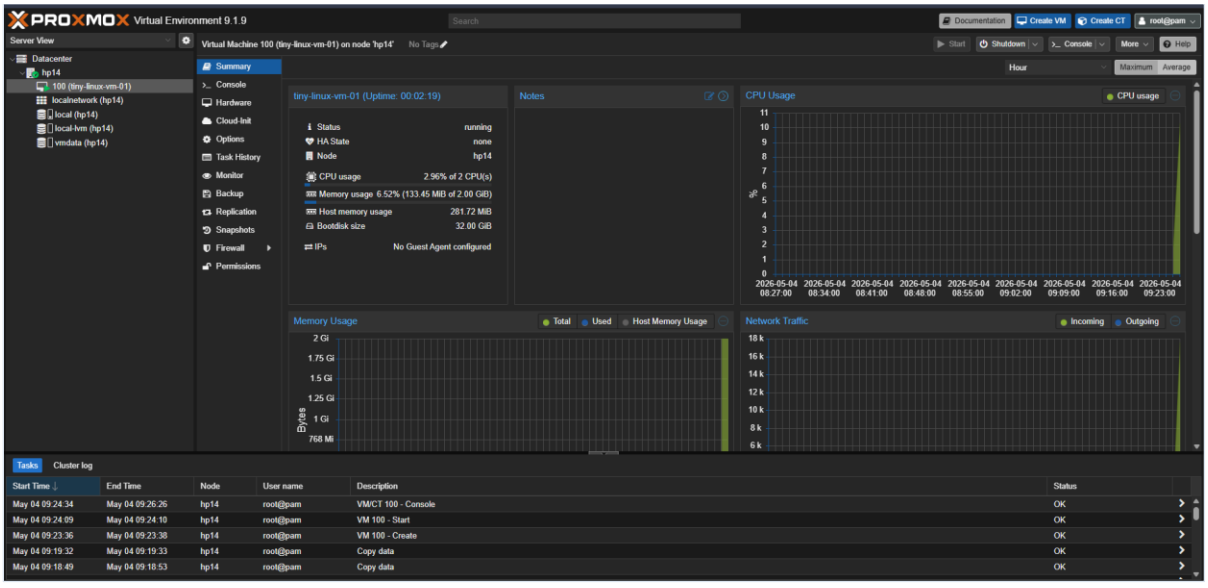


Figure 15 - `VM 100` summary - running

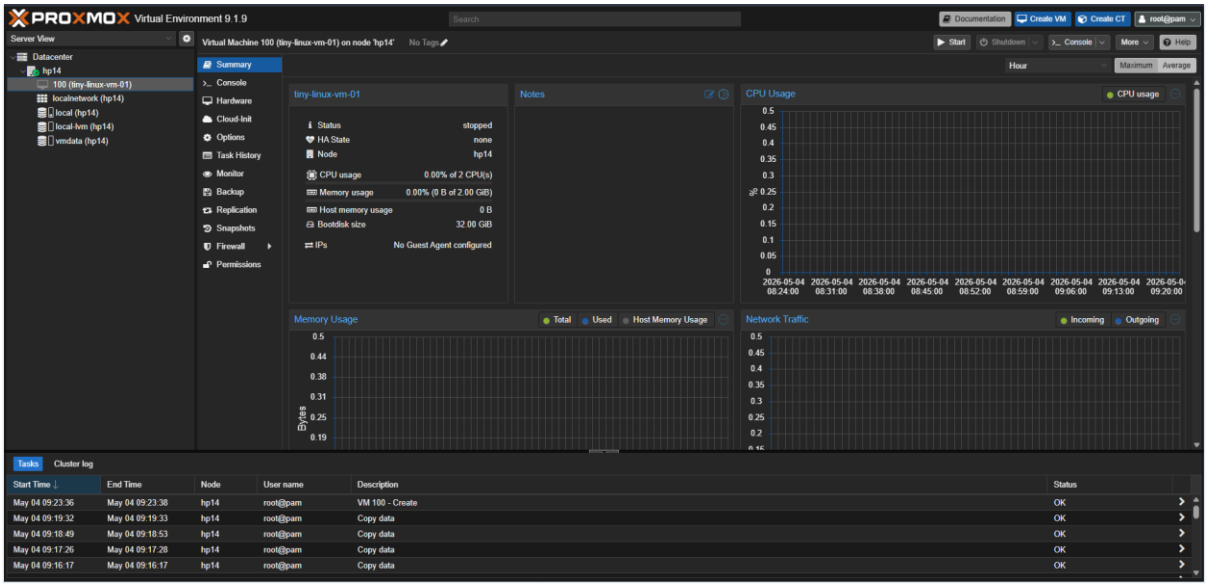


Figure 16 - `VM 100` summary - stopped, just created

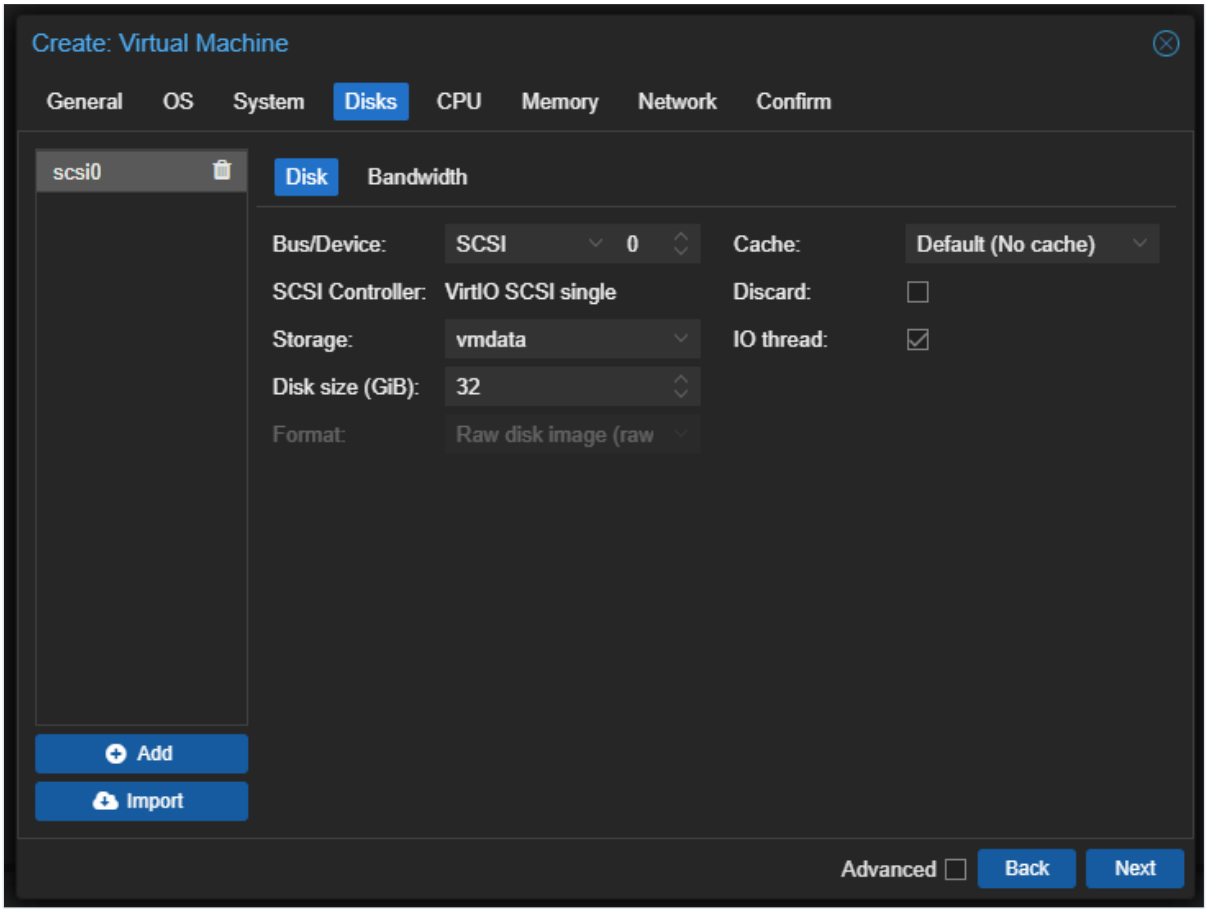


Figure 17 - Create VM wizard - Disks tab, `vmdata` selected

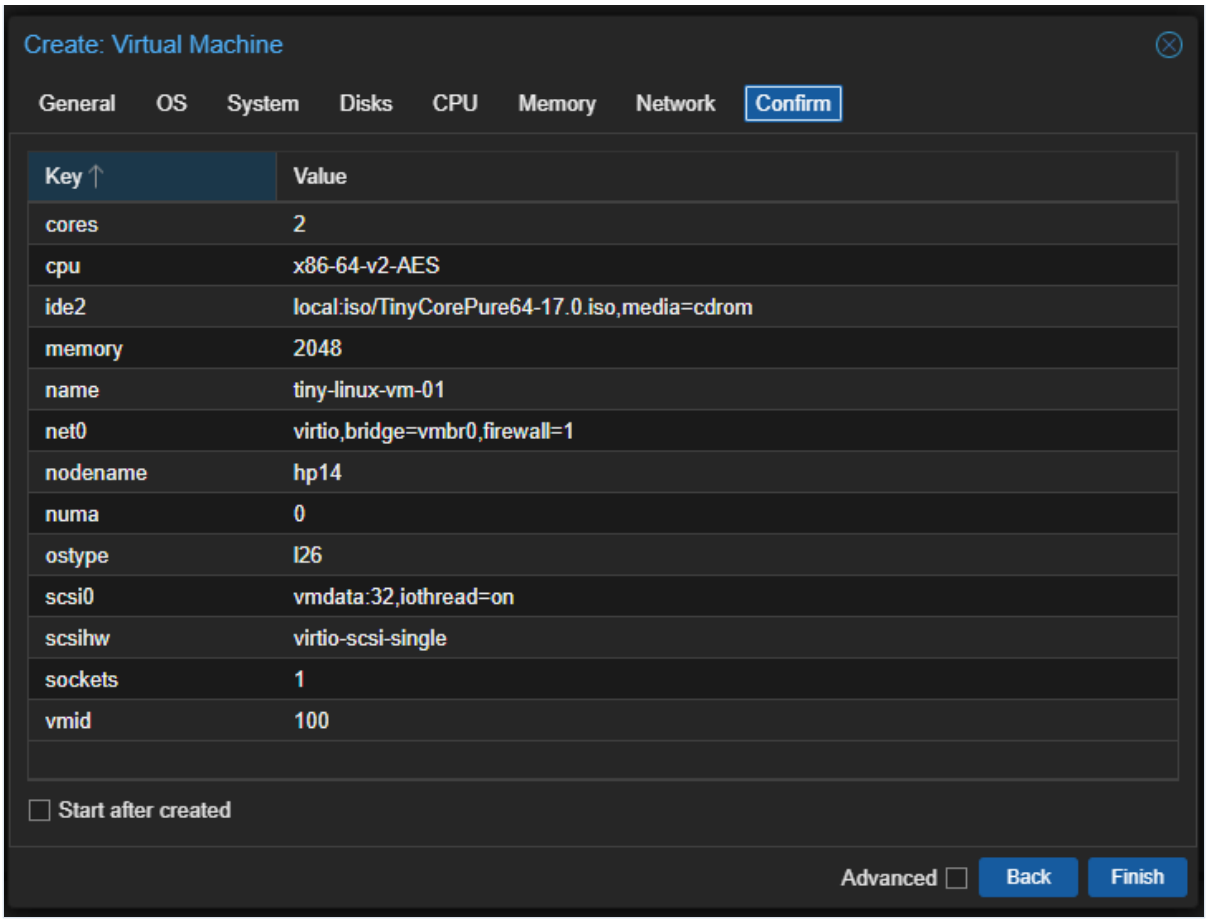


Figure 18 - Create VM wizard - Confirm tab, final config

3. VM creation via CLI

Concept

Creating a VM from the command line proves that the platform can be automated and reproduced without relying only on manual GUI steps. CLI creation is useful for repeatable builds, scripting, troubleshooting, and documenting exact parameters such as CPU count, memory, disk placement, boot order, network bridge, and ISO path.

VMware ESXi (+ vCenter)

The VMware side used PowerCLI from a Windows workstation connected to vCenter. Because HP13 and HP15 were managed by vCenter, PowerCLI was the correct CLI path for VM creation. VM ubuntu-hp15-01 was created on HP15 with HP15-datastore2, 2 vCPUs, 2 GiB RAM, a 20 GiB thin disk, and the VM Network; the Ubuntu ISO was attached; and the VM was powered on using the following cmdlets:

New-VM

Set-CDDrive

Start-VM

Proxmox VE

The Proxmox side used SSH on HP14 and the native qm command. VM 101 (lubuntu-vm-01) was created with one command including CPU, memory, ISO, disk, SCSI controller, network bridge, and boot order. The VM was verified and started; after OS installation the boot order was updated to boot from scsi0 (disk) instead of the CD-ROM device:

qm create

qm config

qm start

Comparison

Aspect	ESXi/vCenter	Proxmox VE
CLI tool	VMware PowerCLI	qm over SSH
Management target	vCenter (Connect-VIServer)	Proxmox node/cluster shell
VM creation command	New-VM	qm create
ISO attachment	Set-CDDrive after VM creation	ISO can be included directly in qm create
Verification	vCenter inventory / PowerCLI output	qm config, qm status, Proxmox task log
Main caution	Use the host name/IP exactly as registered in vCenter	Include the installed disk in the boot order after installation

Impediments / lessons learned

- For vCenter-managed ESXi hosts, direct esxcli or vim-cmd write operations are not the normal VM creation path. PowerCLI through vCenter is the practical CLI equivalent.
- PowerCLI host selection depends on how the host was registered in vCenter. In this lab, HP15 had to be referenced by IP address because the lab DNS was not fully populated at the time of registration, so the hostname did not resolve correctly from the PowerCLI workstation.
- On Proxmox, qm start and qm stop may return no output on success, so qm status <vmid> is needed for explicit verification.
- The Proxmox boot order matters: after installing from ISO, the VM must boot from the virtual disk, not only from the CD-ROM device.

Implementation walkthrough screenshots

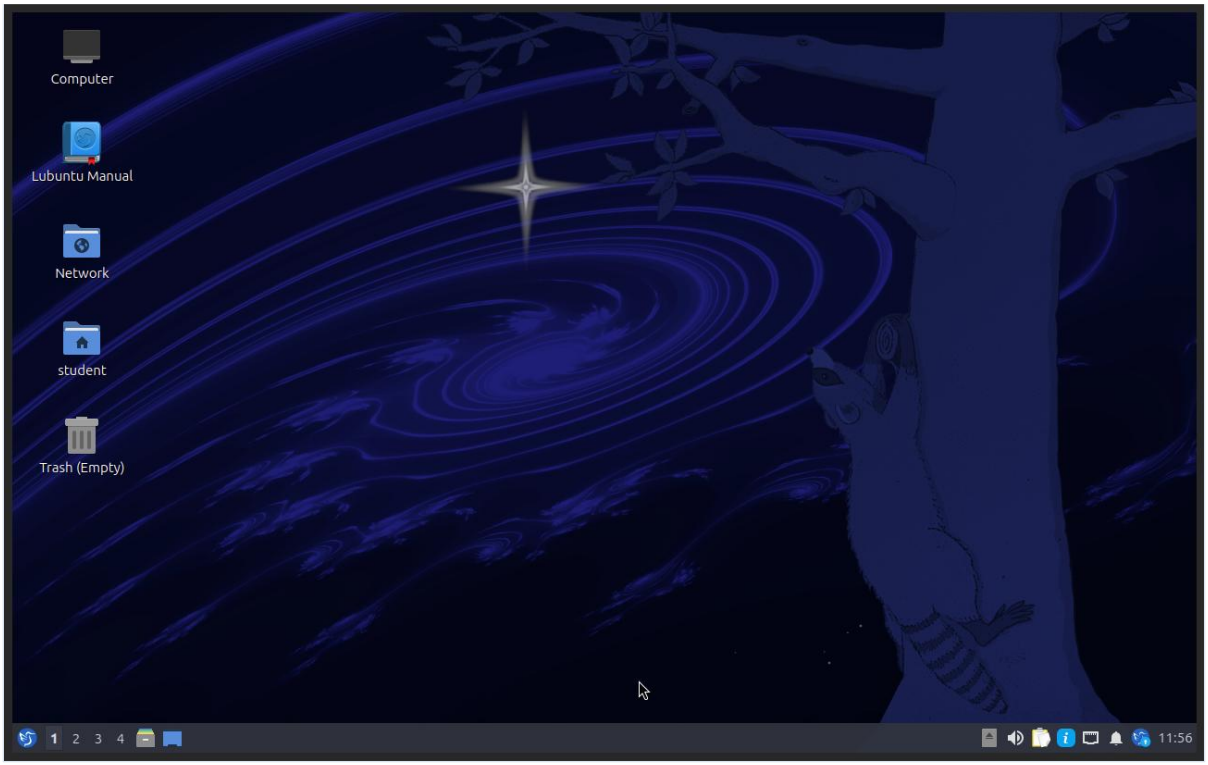


Figure 19 - Installed Linux guest OS booted successfully from the VM disk.

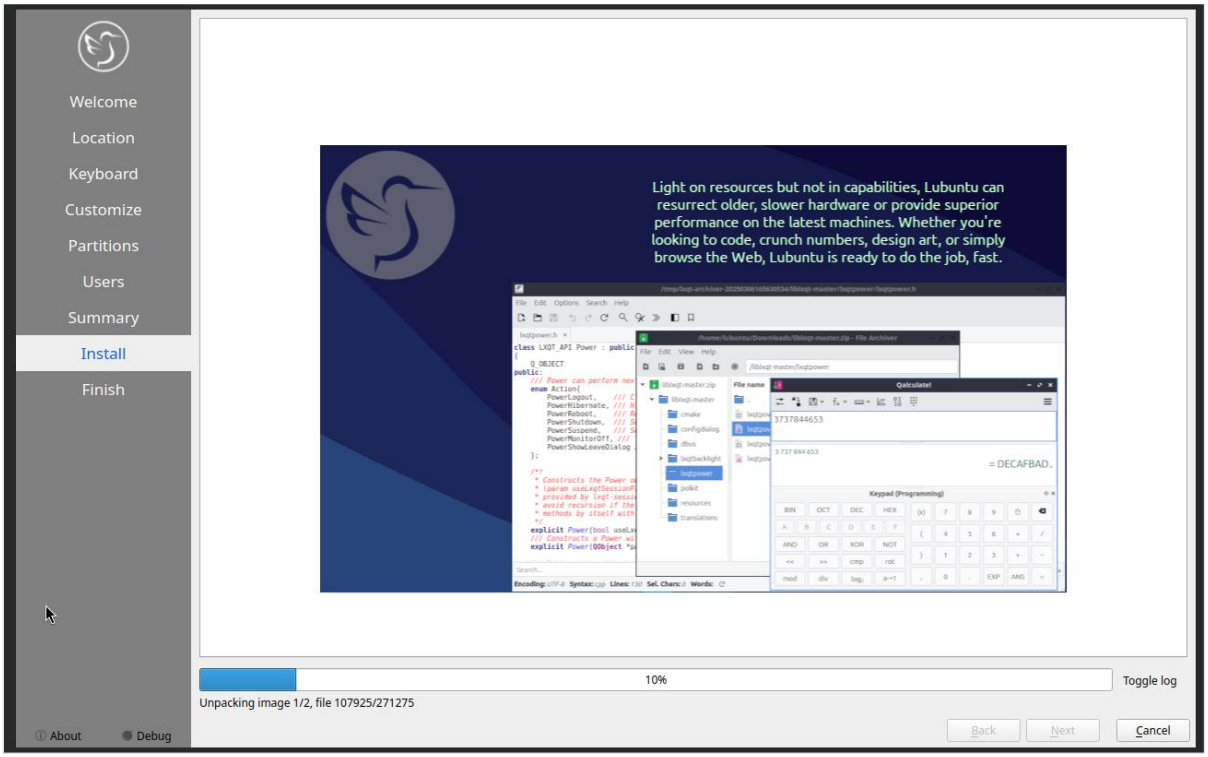


Figure 20 - Linux installer running inside `VM 101` through the Proxmox noVNC console.

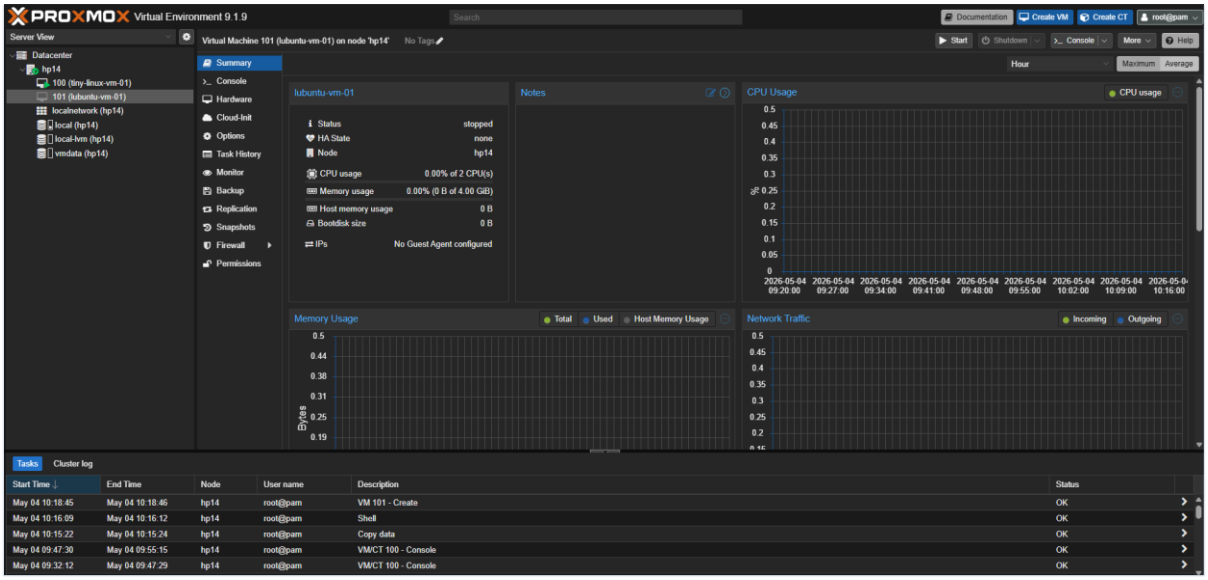


Figure 21 - Proxmox `VM 101` created with the Linux guest profile before OS installation.

```
PS C:\Users\andre> New-VM `
>>       -Name "ubuntu-hp15-01" `
>>       -VMHost "10.0.10.151" `
>>       -Datastore "HP15-datastore2" `
>>       -GuestId "ubuntu64Guest" `
>>       -NumCpu 2 `
>>       -MemoryMB 2048 `
>>       -DiskMB 20480 `
>>       -DiskStorageFormat Thin `
>>       -NetworkName "VM Network" `
>>       -CD

Name                 PowerState Num CPUs MemoryGB
----                 ---------- -------- --------
ubuntu-hp15-01       PoweredOff 2        2.000
```

Figure 22 - PowerCLI output confirming creation of the ESXi CLI VM.

```
PS C:\Users\andre>  Connect-VIServer -Server 10.0.10.132 -User administrator@vsphere.local -Password Test..123

Name                           Port  User
----                           ----  ----
10.0.10.132                    443   VSPHERE.LOCAL\Administrator
```

Figure 23 - PowerCLI session connected to vCenter for CLI-based VM creation.

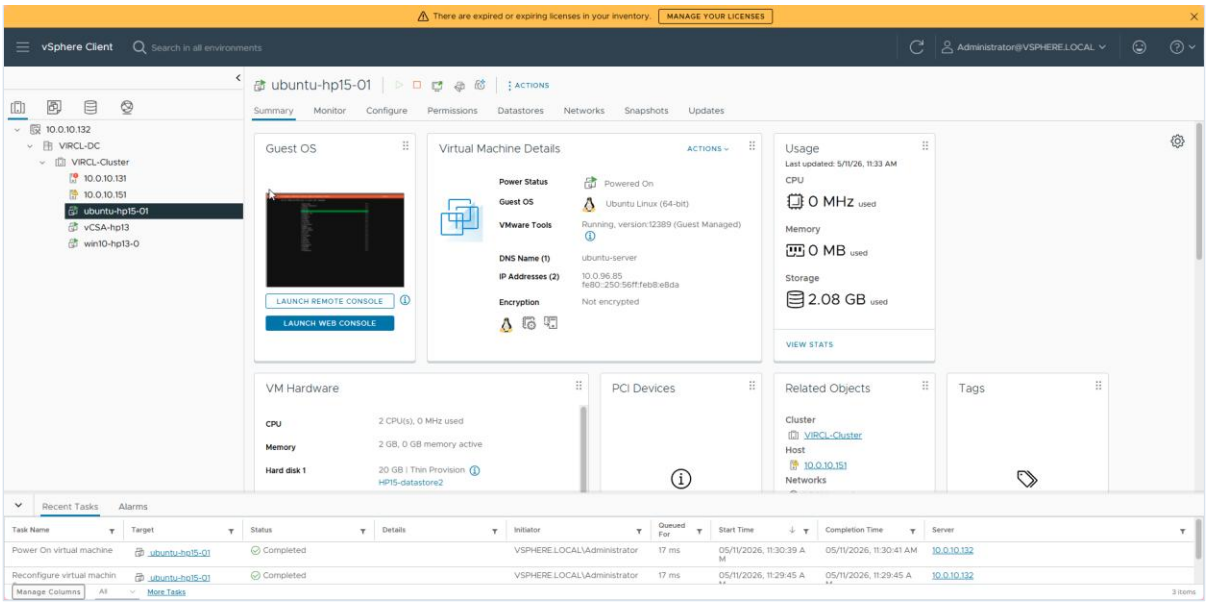


Figure 24 - CLI-created VM visible in the vCenter inventory.

4. Guest OS installation

Concept

A guest OS is the operating system installed inside a virtual machine. The hypervisor provides virtual hardware: CPU, memory, disk controller, network adapter, firmware, and console access. The guest OS was installed onto that virtual hardware in the same general way as an installation on a physical computer.

This chapter is separate from VM creation because creating a VM only prepares the virtual machine container. A usable VM also needs an operating system installed from an ISO, the boot order corrected after installation, and ideally guest tools installed so the hypervisor can manage the VM cleanly.

Examples in this project:

Requirement	Guest OS	Hypervisor implementation
Windows client OS	Windows 10 Enterprise Evaluation	ESXi / vCenter on HP15, shown through the GUI VM creation and installation screenshots
Linux guest OS	Lubuntu 26.04 Linux guest	Proxmox VE on HP14, VM 101, shown through the CLI VM creation and installation screenshots

VMware ESXi (+ vCenter)

Implementation result: Windows 10 Enterprise Evaluation was installed on `win10-vm-01` on HP15 through vCenter. The installation completed successfully, the VM booted to the Windows desktop, and the guest OS was verified as functional through the vCenter remote console.

Proxmox VE

Outcome: Lubuntu 26.04 was installed as a Linux guest on `lubuntu-vm-01` (`VM 101`) on HP14. The VM was created in Chapter 3, but the actual guest OS installation steps and screenshots are referenced here because they satisfy the guest OS installation chapter.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Guest OS selection	Guest OS family and version are selected during VM creation, e.g., Windows / Microsoft Windows 10 (64-bit).	Guest OS type is selected as Linux / l26 for modern Linux guests.
Installation media	ISO uploaded to a datastore and mounted as a virtual CD/DVD drive.	ISO uploaded to local storage and mounted as ide2 virtual CD/DVD media.
Console	vCenter remote console / web console.	Built-in noVNC console in the Proxmox web UI.
Post-install action	Disconnect ISO and boot from virtual disk. Install VMware Tools if needed.	Detach ISO and set boot order to scsi0. Install qemu-guest-agent later if guest-management verification is needed.
Main risk	Leaving the virtual CD/DVD drive connected after installation may cause the VM to boot from the ISO again on next start — disconnect the ISO and verify the boot order points to the virtual disk after the OS is installed.	Boot order can accidentally stay ISO-only; then the VM will not boot after ejecting the ISO.

Impediments / lessons learned

- A hypervisor's "Guest OS" field is not the installed OS. It is a compatibility/profile setting for the virtual hardware.
- Tiny live ISOs are useful for boot tests, but they do not satisfy the persistent guest OS installation requirement unless the OS is installed to disk.
- For Proxmox, setting the boot order to include both ide2 (CD-ROM) and scsi0 (disk) during installation avoids the 'no bootable disk' problem on first reboot — the VM boots from the ISO to install, then falls through to the disk. Once installation is complete, simplify the boot order to scsi0 only so the VM no longer tries to boot from a disconnected ISO.

Implementation walkthrough screenshots

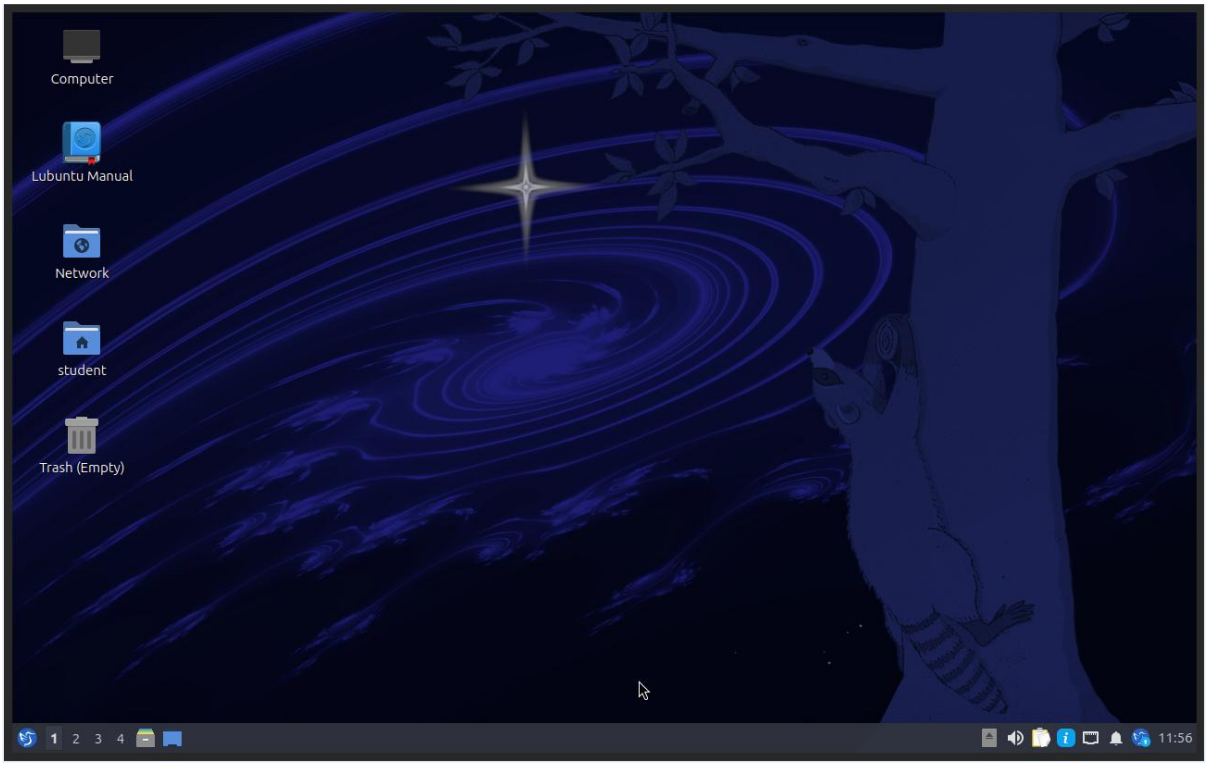


Figure 25 - Installed Linux guest OS booted successfully from the VM disk.

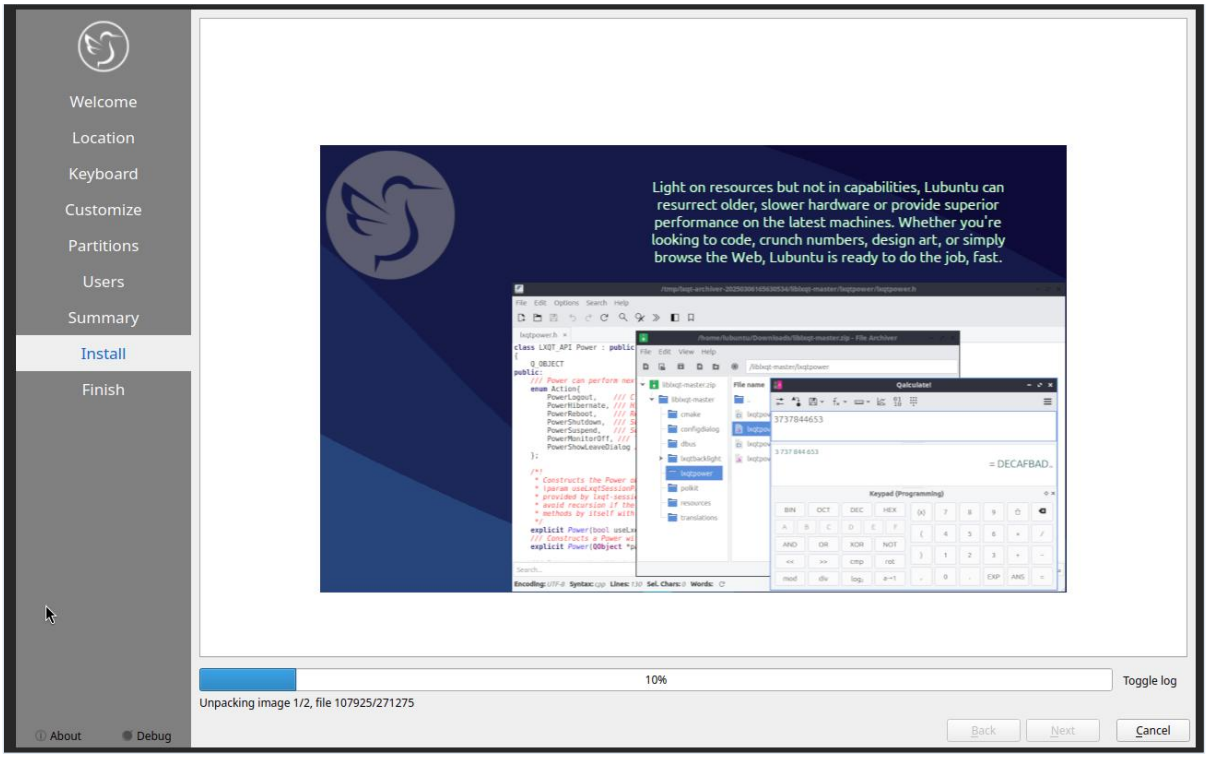


Figure 26 - Linux installer running inside `VM 101` through the Proxmox noVNC console.

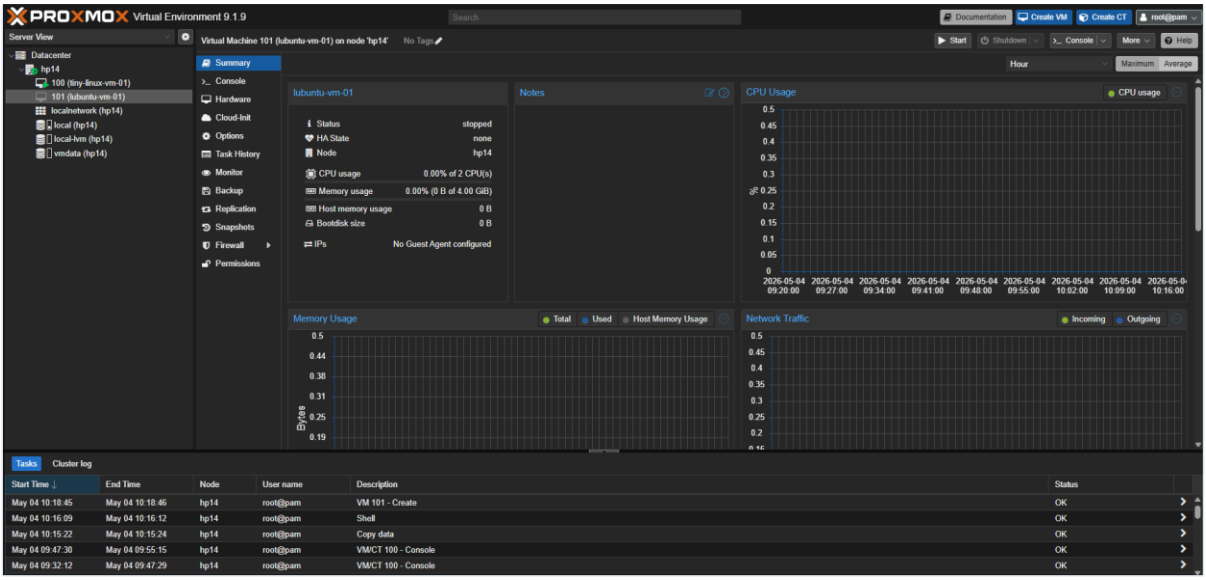


Figure 27 - Proxmox `VM 101` created with the Linux guest profile before OS installation.

5. VM resource reconfiguration

Concept

VM resource reconfiguration means changing the virtual hardware assigned to a VM after it has already been created. Typical resources include CPU count, memory size, virtual disks, and network adapters.

This matters because VM sizing is rarely perfect on the first attempt. Both hypervisors allowed administrators to adjust resources as workload requirements changed without rebuilding the VM from scratch.

VMware ESXi (+ vCenter)

Result: Andrea reconfigured VM `win10-hp13-01` on HP13 through vCenter. The VM started with 2 vCPUs, 4 GB RAM, one 48 GB disk, and one E1000E network adapter. It was powered off, changed to 4 vCPUs and 8 GB RAM, and a temporary 10 GB thin-provisioned disk plus a second VMXNET3 network adapter were added. Windows booted successfully afterward. The guest showed the new CPU/RAM and the new 10 GB disk, and then the temporary disk and NIC were removed again.

Proxmox VE

Outcome: `VM 101` (`lubuntu-vm-01`) on HP14 was reconfigured in Proxmox. CPU and RAM were increased. A temporary disk and temporary NIC were added, and the VM booted successfully afterward. The temporary NIC was removed, and the temporary disk was detached and deleted from Unused Disk 0. CPU/RAM were restored to the original VM size.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Management interface	vCenter VM settings editor.	Proxmox VM Hardware tab or qm set CLI.
CPU/RAM changes	Usually done while powered off unless hot-add is configured and supported by the guest.	Easiest while powered off; hotplug exists but is not required for this lab.
Disk changes	Add disk in VM settings, then initialize/format inside the guest if needed.	Add disk as scsi1/similar, then initialize/format inside the guest if needed.
NIC changes	Add/remove virtual NICs connected to port groups.	Add/remove netX adapters connected to Linux bridges such as vmbr0.
Cleanup risk	Removing a disk from VM config may not delete datastore files unless explicitly deleted.	Detaching a disk can leave an unusedX entry that must be removed for full cleanup.

Impediments / lessons learned

- Hypervisor-side resource changes and guest-side configuration are separate. Adding a disk or NIC to the VM does not automatically make it usable inside the OS.
- For a safe verification step, use a temporary second disk instead of resizing the OS disk. This avoids irreversible disk-growth work and keeps the installed guest clean for later chapters.
- Shut down the VM before the resource changes unless the chapter is specifically trying to demonstrate hotplug.

Implementation walkthrough screenshots

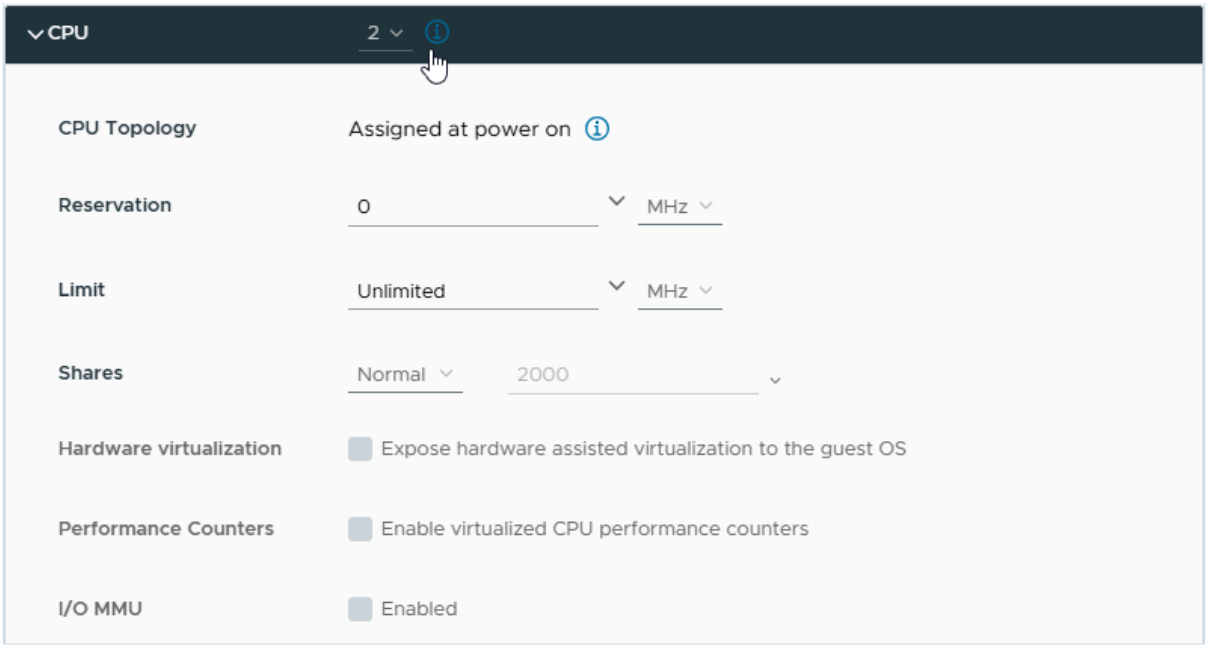


Figure 28 - ESXi VM resource baseline before CPU, memory, disk, and network changes.

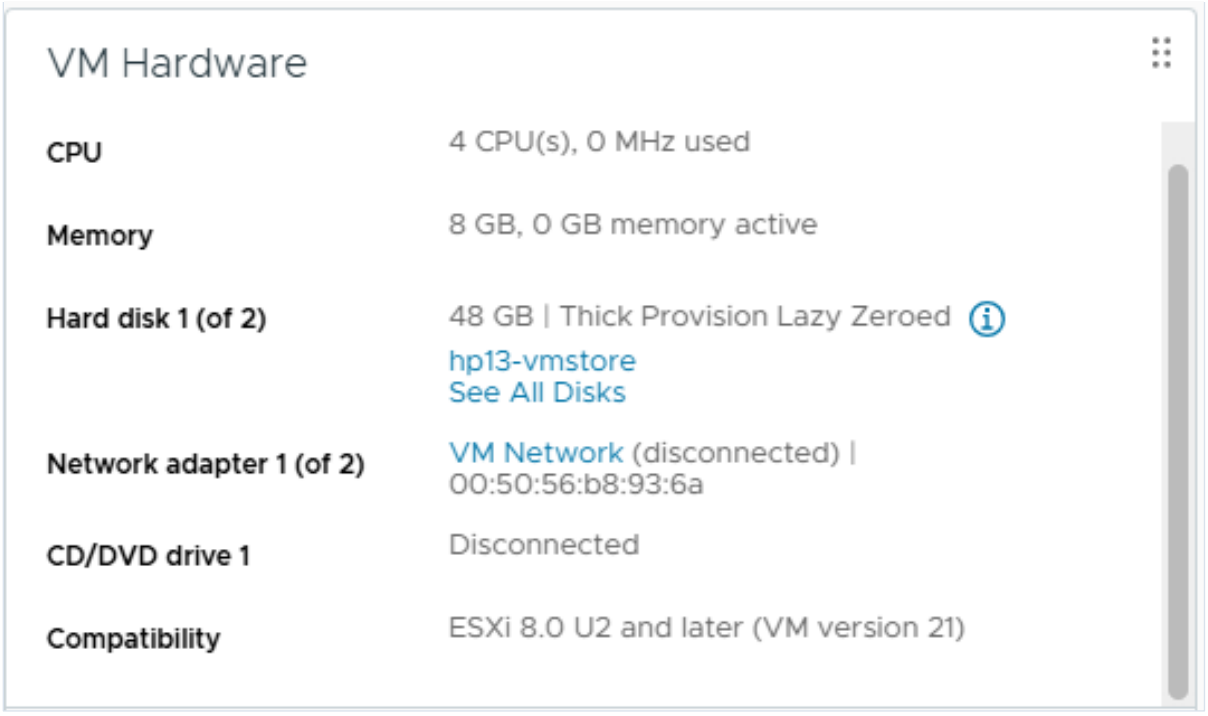


Figure 29 - ESXi VM hardware after CPU, memory, disk, and network reconfiguration.



Figure 30 - Guest OS verification of updated CPU and RAM values.

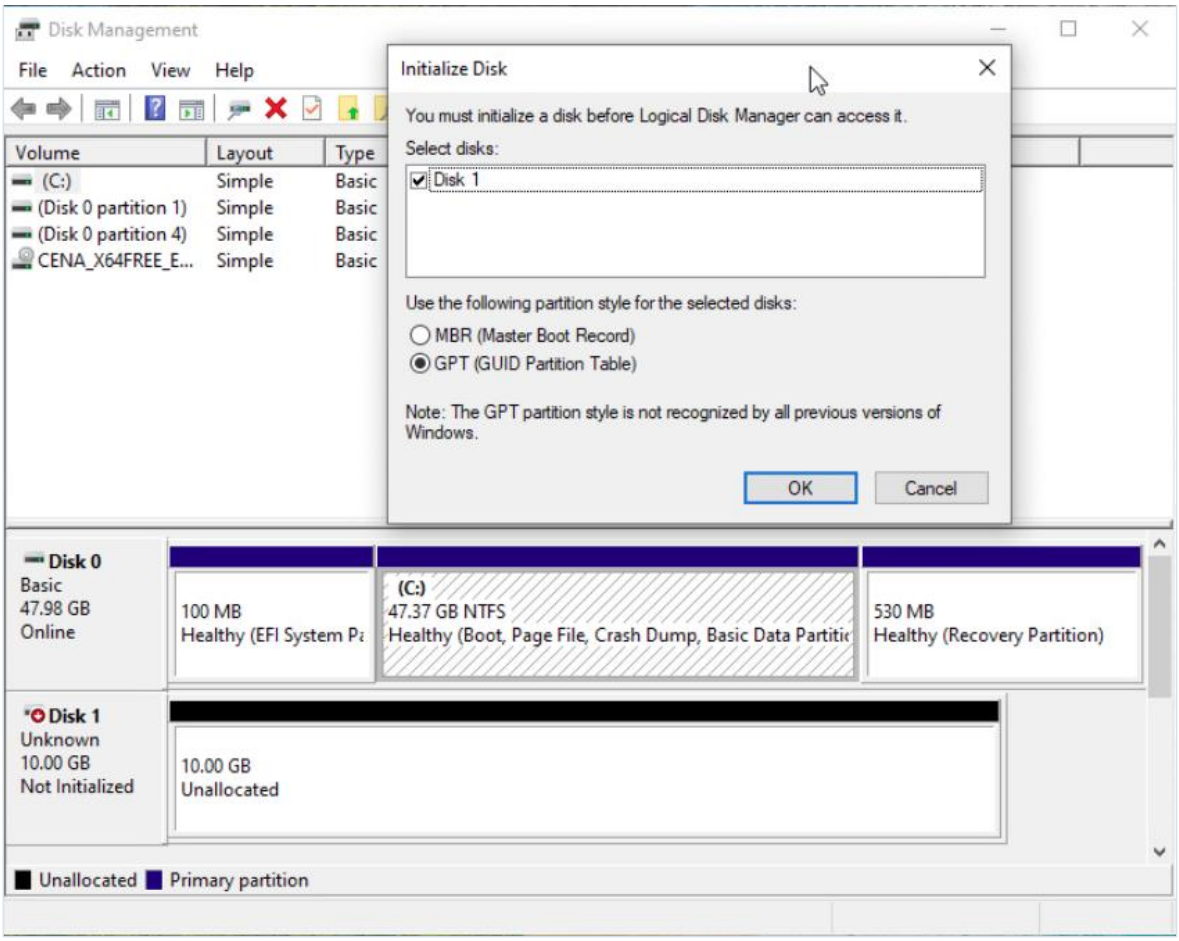


Figure 31 - Guest OS verification of the added virtual disk.

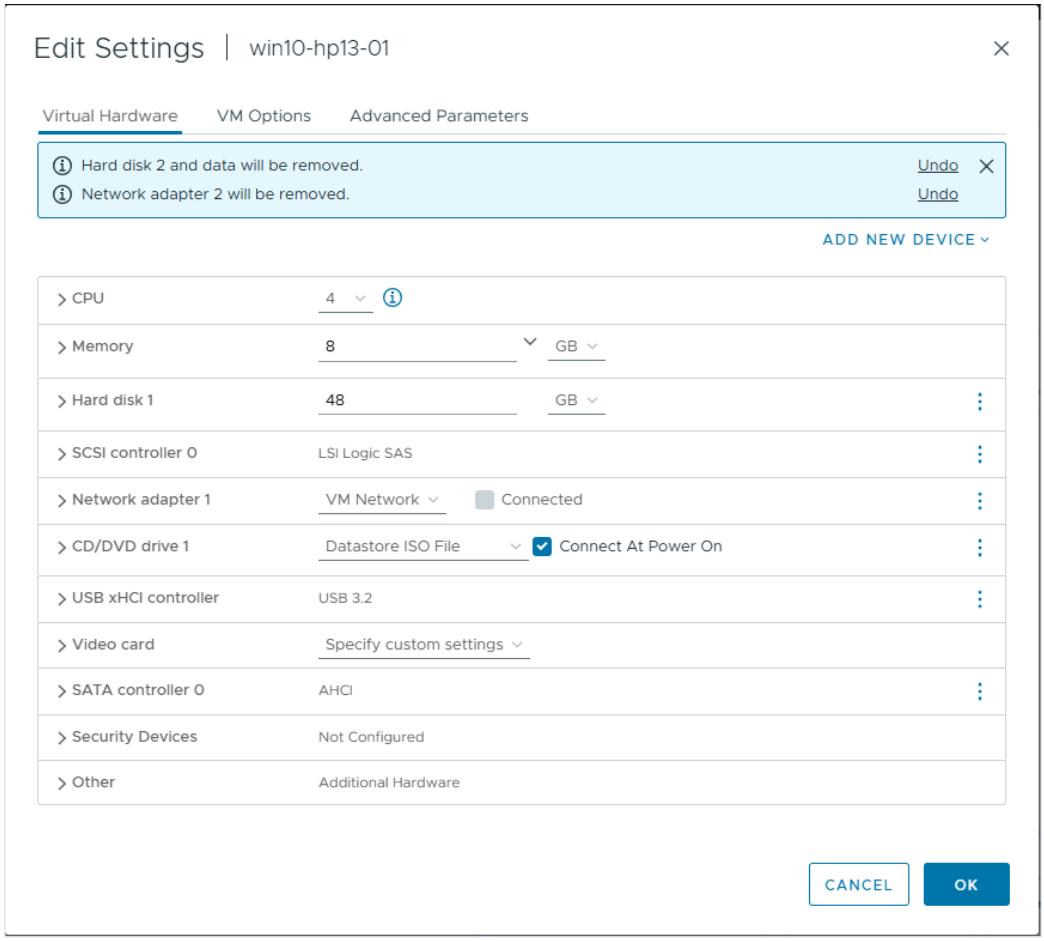


Figure 32 - ESXi VM settings after temporary test resources were cleaned up.

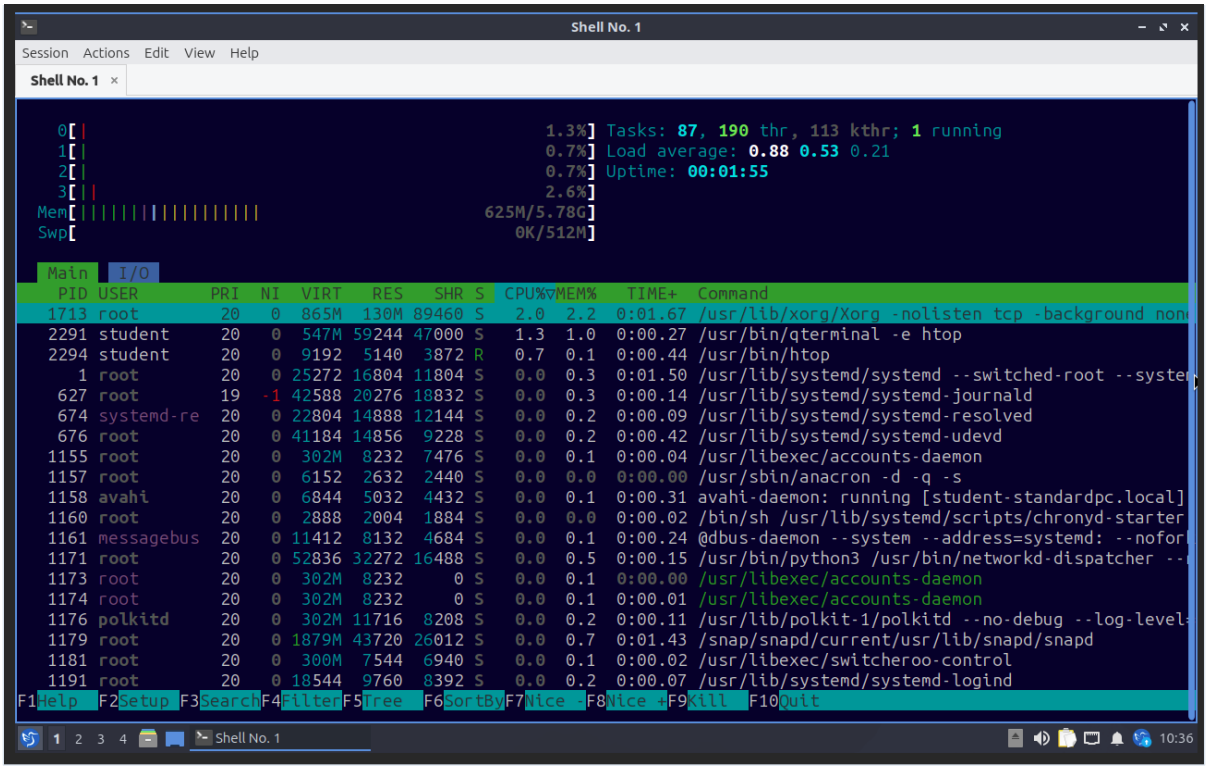


Figure 33 - Proxmox `VM 101` booted successfully after resource changes.

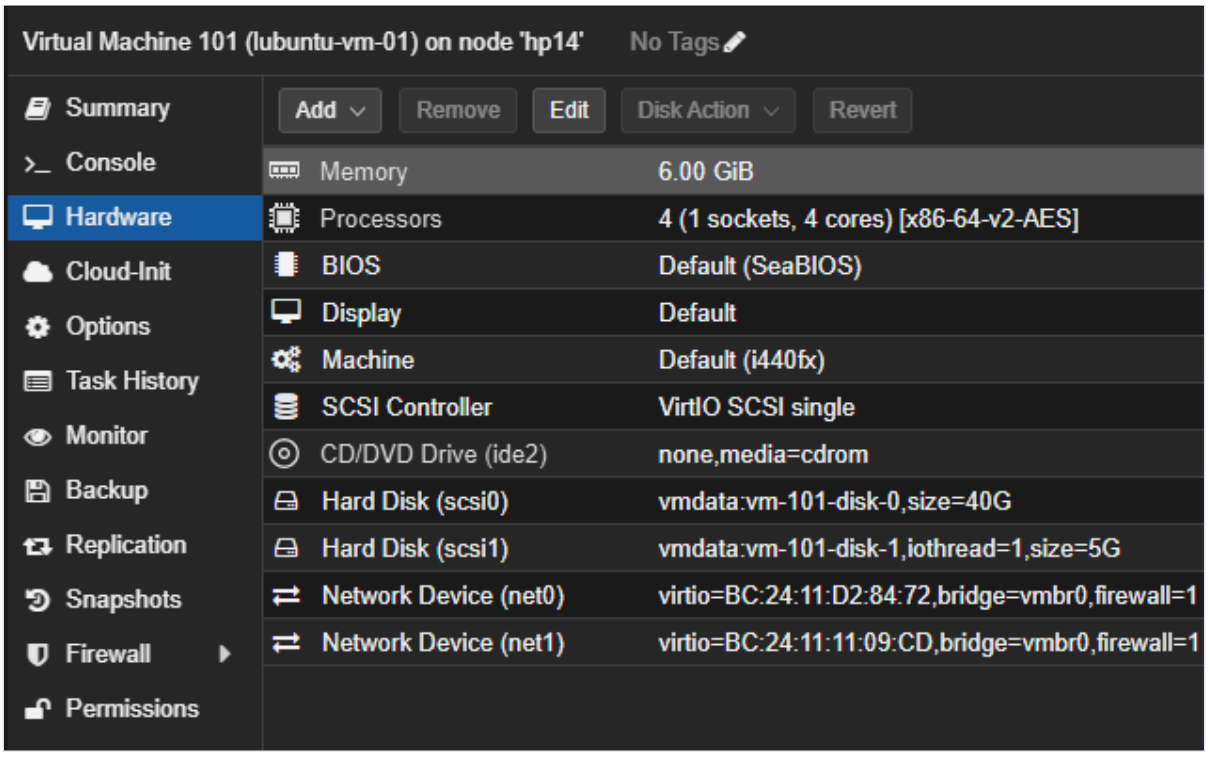


Figure 34 - Proxmox `VM 101` after adding a disk and second network adapter.

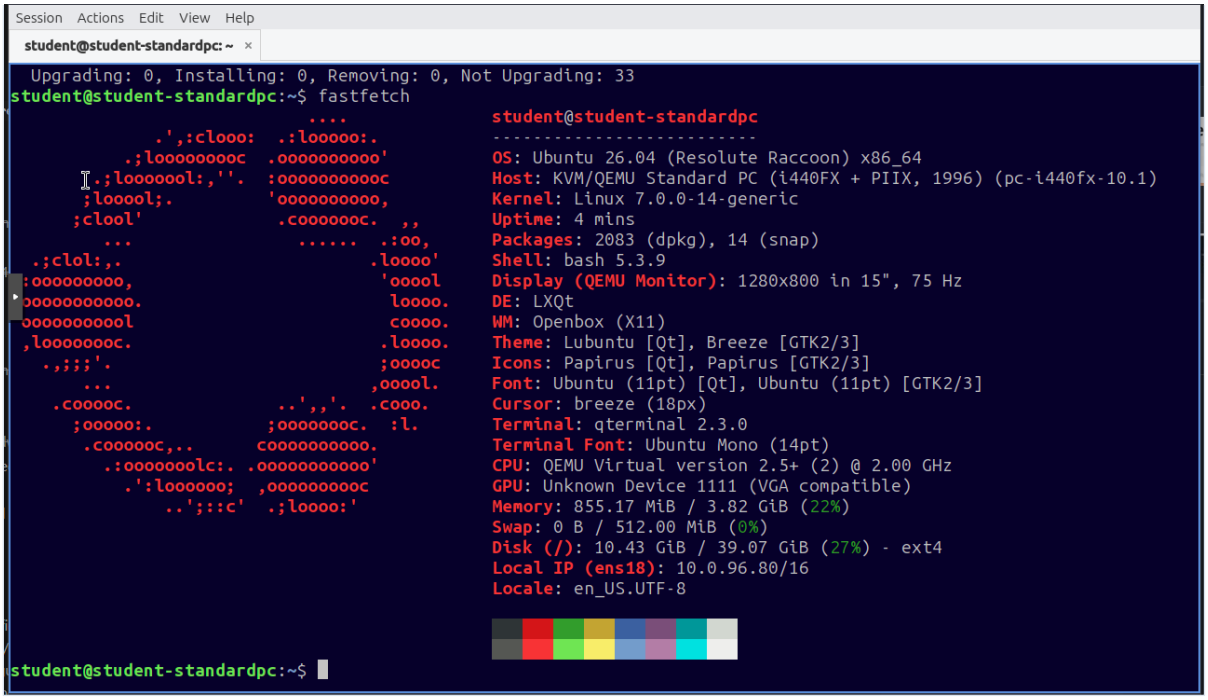


Figure 35 - Linux guest verification after restoring the original resource profile.

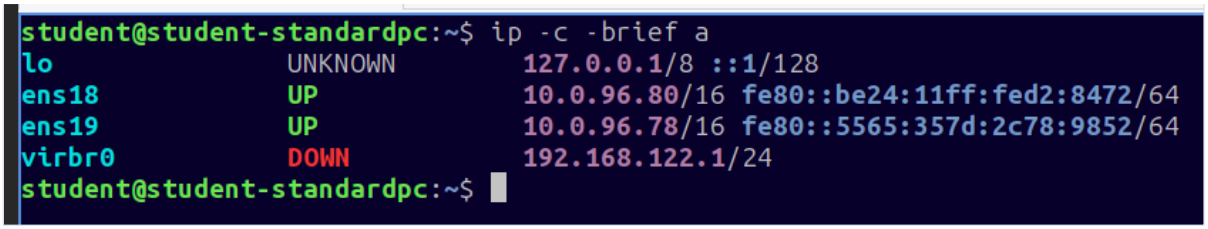


Figure 36 - Linux guest OS detecting both Proxmox virtual NICs.

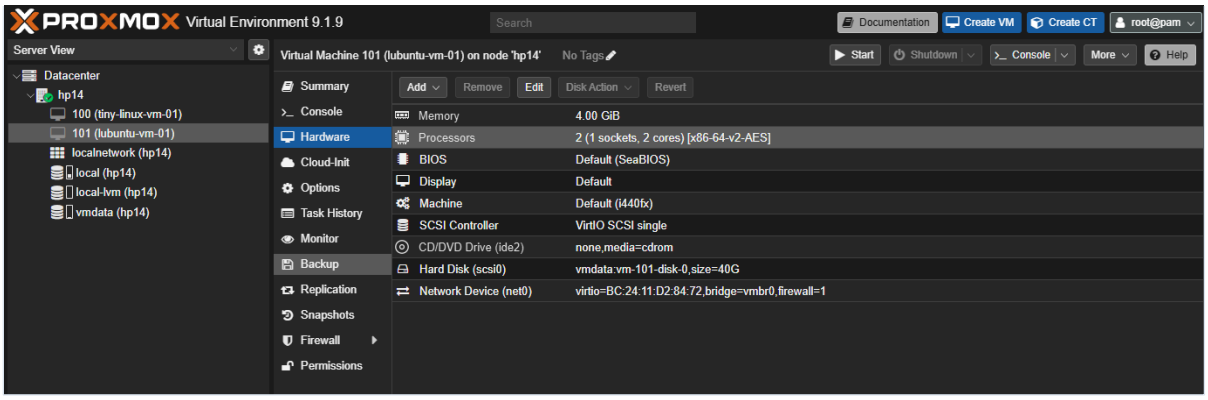


Figure 37 - Proxmox `VM 101` baseline hardware before resource changes.

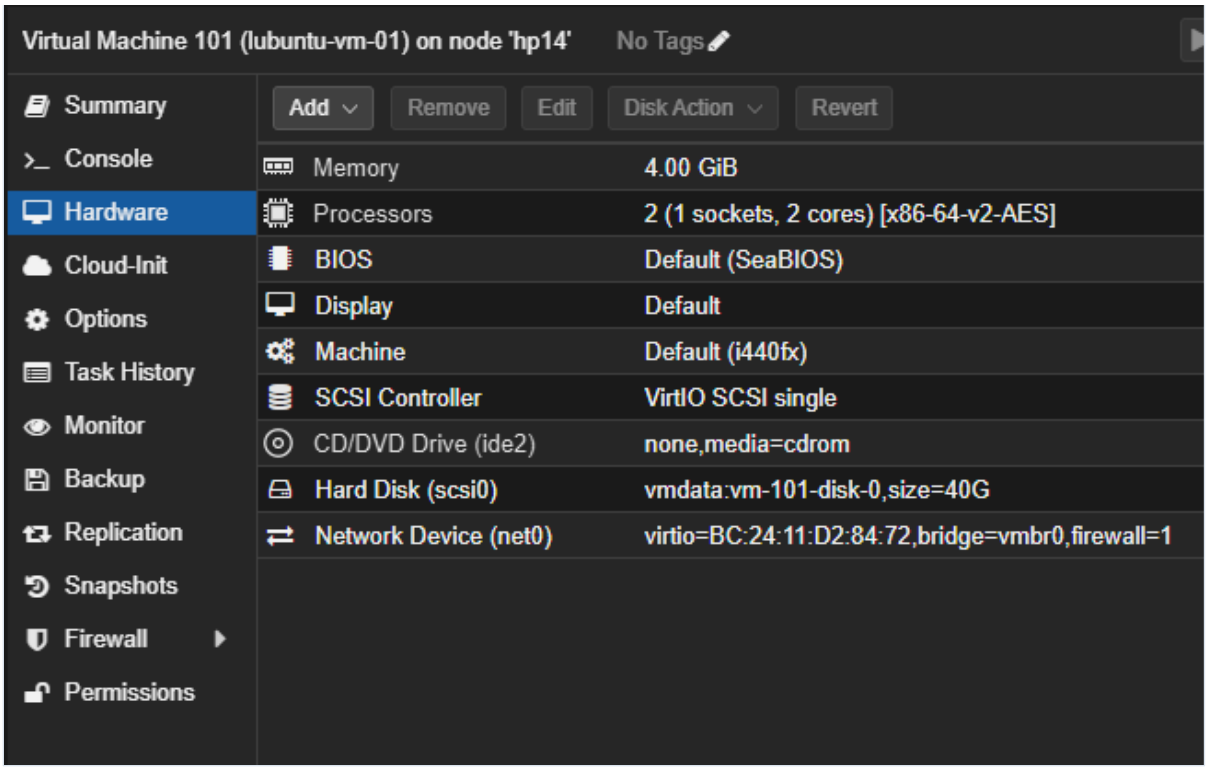


Figure 38 - Proxmox `VM 101` hardware after cleanup and restoration.

6. Cloning a VM

Concept

VM cloning creates a new VM from an existing one. Instead of installing an operating system and configuring hardware from zero, the administrator copies a known-working VM and then adapts the clone.

Cloning is useful for testing, repeatable deployments, and preparing multiple similar machines. The main design choice is whether the clone is a full clone or a linked clone:

- A full clone has its own independent disk copy. It uses more storage but is safer and easier to move/delete independently.
- A linked clone depends on the source/template disk. It saves storage but creates a dependency on the original backing disk.

VMware ESXi (+ vCenter)

Implementation result: VM `win10-hp13-01` on HP13 was cloned through vCenter using Clone → Clone to Virtual Machine. The clone was named `win10-hp13-01-clone`, placed on datastore `HP13-vmstore`, and the clone task completed successfully. The cloned VM appeared in the vCenter inventory and booted without errors.

Proxmox VE

Outcome: `VM 101` (`lubuntu-vm-01`) was full-cloned into `VM 102` (`lubuntu-vm-01-clone`) on HP14. The clone task completed with TASK OK, `VM 102` appeared with its own disk on `vmdata`, and the cloned VM booted successfully.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Entry point	VM context menu in vCenter.	VM context menu in Proxmox web UI, or qm clone CLI.
Full clone	Creates independent VM files/disks.	Creates independent Proxmox VM disks.
Linked clone	Requires an existing snapshot or a VM template as the backing source; not used in this lab.	Available when source conditions support it, but full clone is simpler for this lab.
Storage choice	Select datastore during clone wizard.	Select target storage such as vmdata.
Main risk	Cloned guest may keep same hostname/IP identity; vCenter's built-in Guest Customization spec can automate hostname and IP reassignment.	Cloned guest may keep same hostname/IP identity; must be corrected manually inside the guest OS after cloning.

Impediments / lessons learned

- Full clones are easiest to document because they have no dependency on the source VM after cloning.
- Running a source VM and its clone at the same time may create duplicate hostnames inside the network. The lab clone was used only for verification, while production clones require guest customisation.

Implementation walkthrough screenshots

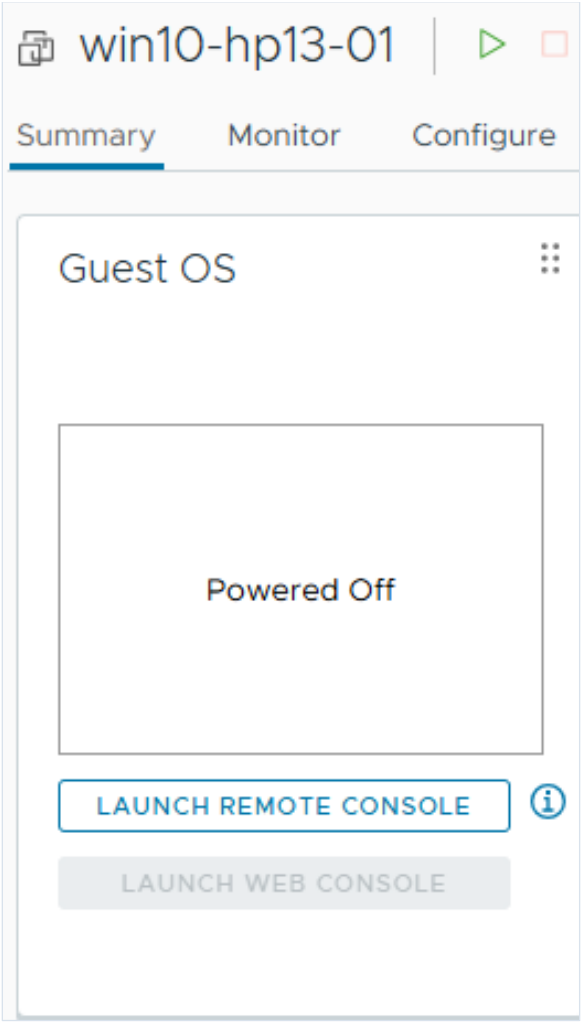


Figure 39 - ESXi source VM powered off before cloning.

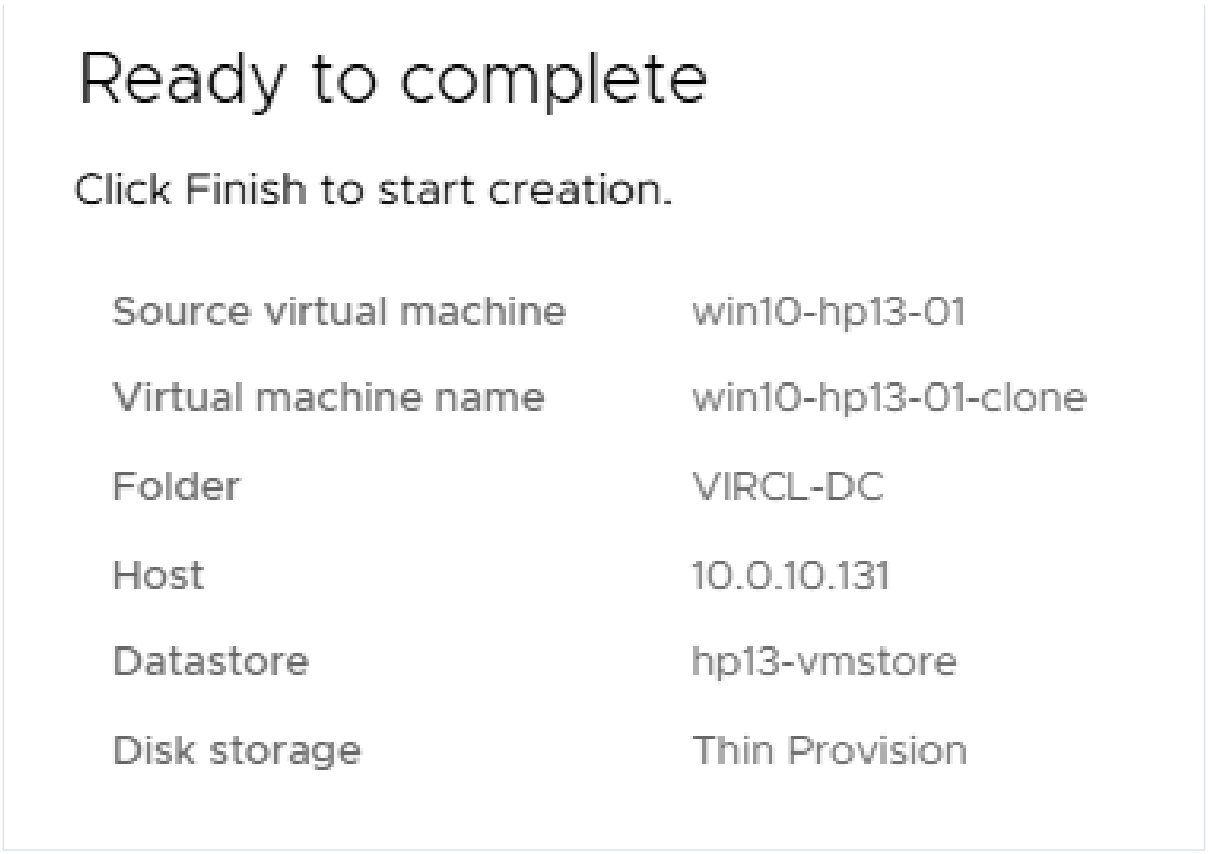


Figure 40 - vCenter clone wizard summary before creating the ESXi clone.

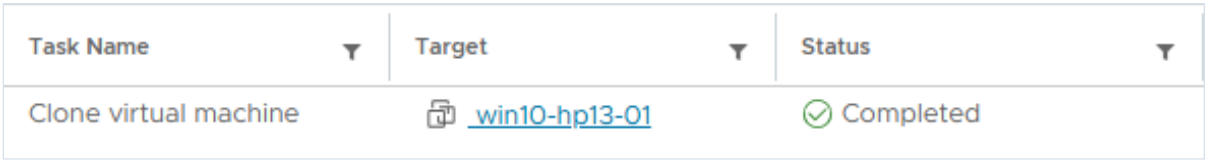


Figure 41 - vCenter clone task completed successfully.

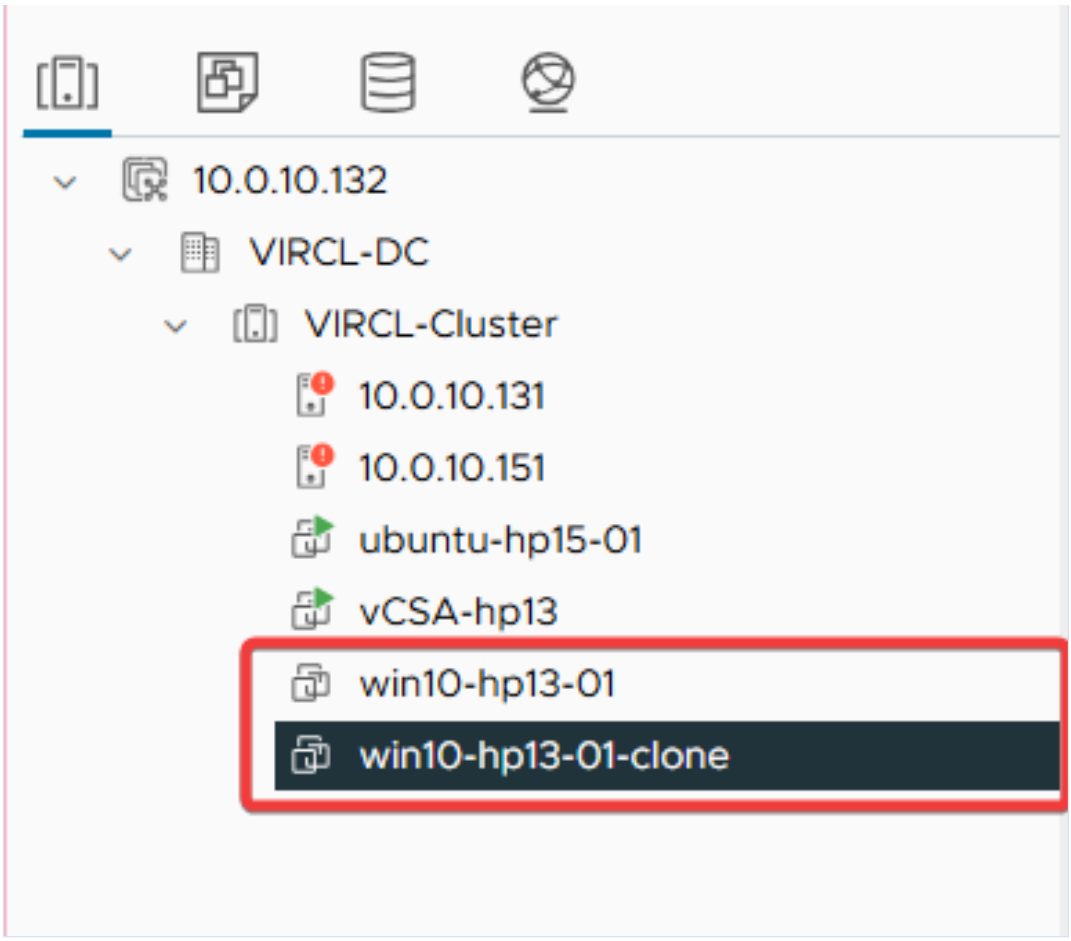


Figure 42 - Source VM and cloned VM visible in the vCenter inventory.

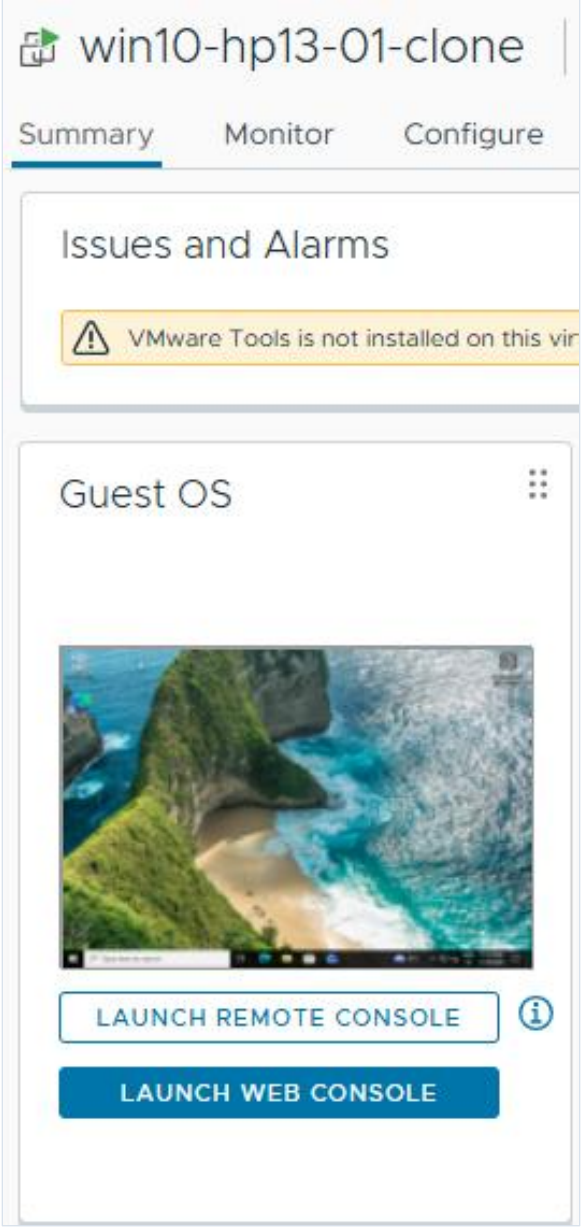


Figure 43 - Cloned ESXi VM booted successfully after cloning.

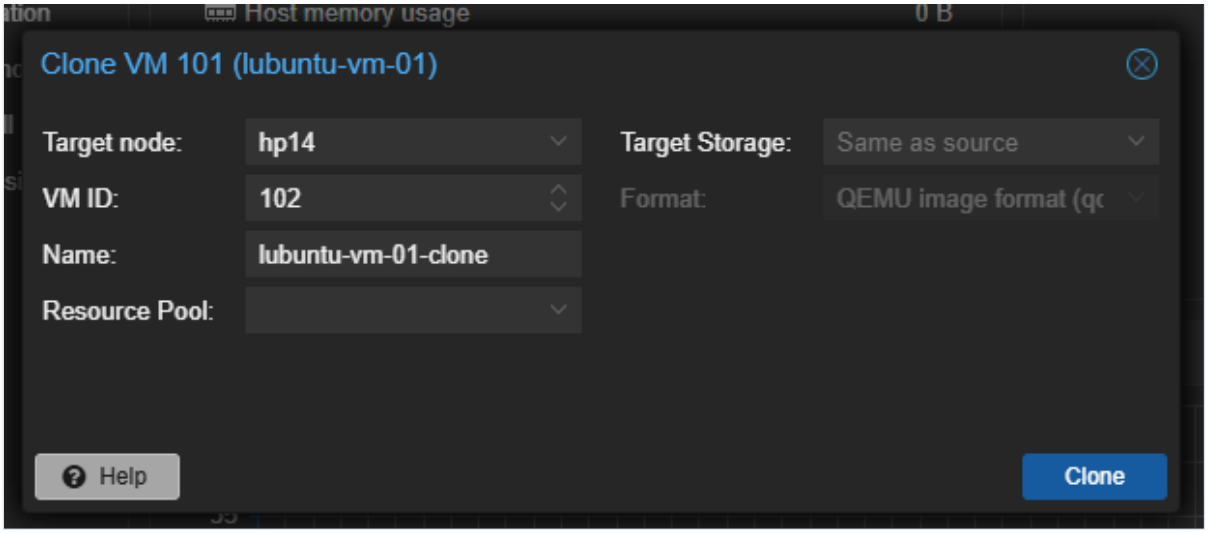


Figure 44 - Proxmox full-clone dialog for `VM 101`.

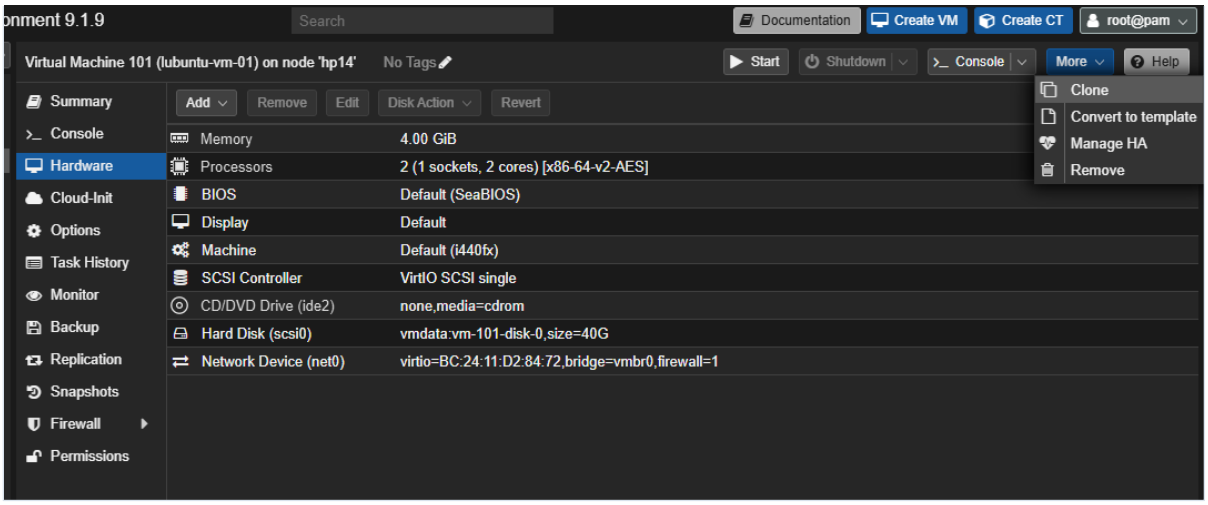


Figure 45 - Proxmox source VM stopped before clone creation.

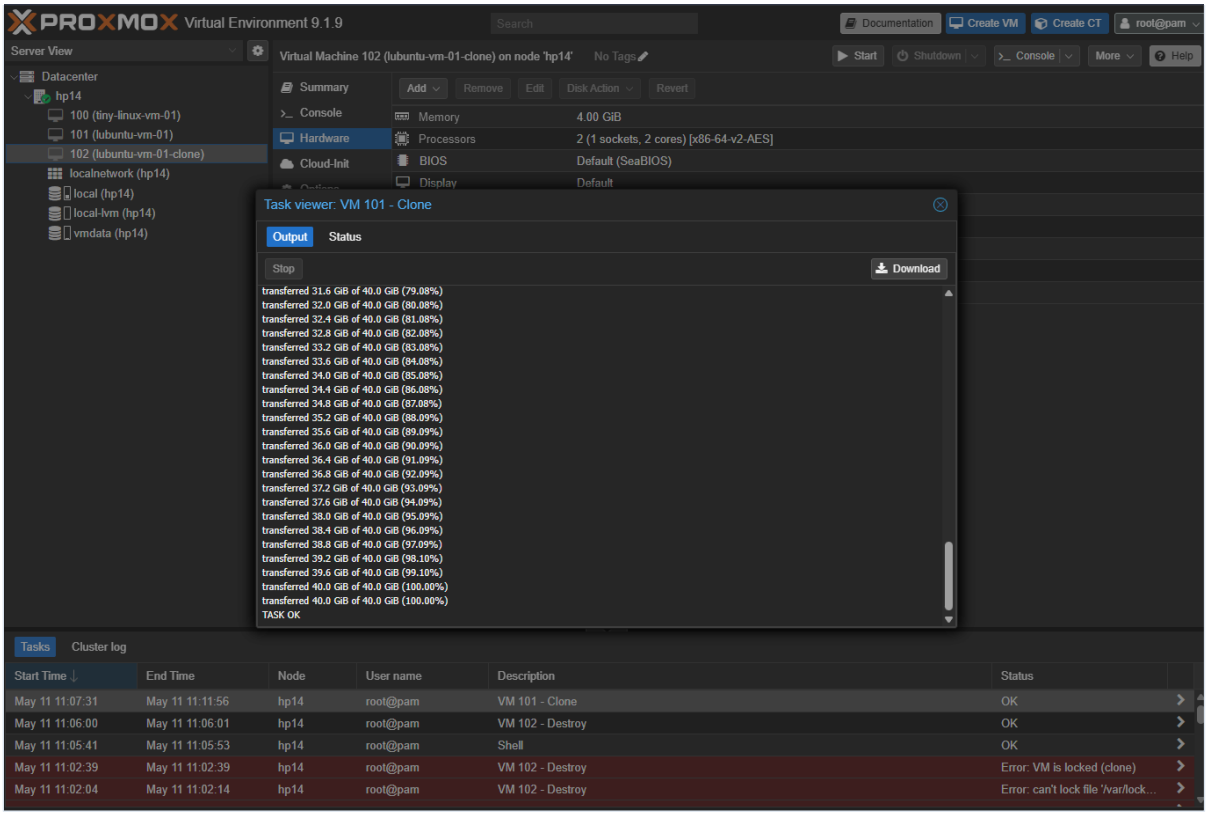


Figure 46 - Proxmox clone task completed successfully.

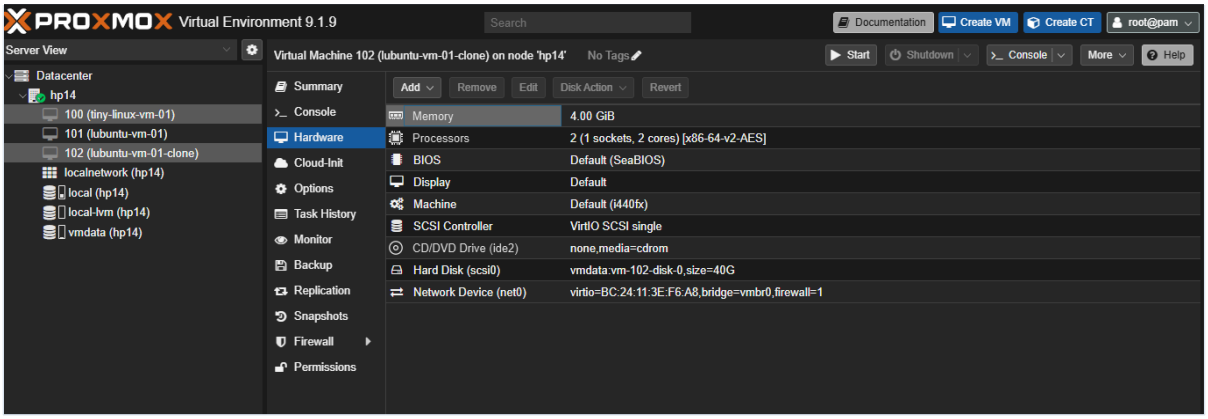


Figure 47 - Proxmox cloned `VM 102` hardware visible after cloning.

7. Exporting a VM

Concept

Exporting a VM means packaging the VM configuration and virtual disks so the VM can be moved, archived, or imported somewhere else. It is different from cloning: a clone stays inside the same virtualisation platform, while an export creates files that can be stored outside the inventory.

For this project, the most important point is that VMware ESXi/vCenter and Proxmox VE use different export formats:

- VMware exports VMs as OVF/OVA-style packages. OVF is a portable description format with one or more virtual disk files.
- Proxmox normally exports a QEMU VM through vzdump. This creates a Proxmox backup archive that can be restored with qmrestore or through the Proxmox web UI.

In both systems, the cleanest export is made while the VM is powered off. This avoids filesystem inconsistency inside the guest OS and makes the export step easier to explain.

VMware ESXi (+ vCenter)

Result: Andrea exported the vCenter VM `win10-hp13-01-clone` as an OVF package. The VM was powered off first, Export OVF Template was opened from vCenter, the export name `win10-hp13-01-clone-export` was used, vCenter showed the export task in progress, and File Explorer showed the downloaded OVF package files (.ovf, .mf, .vmdk, .iso, and .nvram).

Proxmox VE

Outcome: Gui exported the HP14 Lubuntu clone `VM 102` (`lubuntu-vm-01-clone`) using a one-off Proxmox backup on 11.05.2026. The backup used storage local, mode Stop, and compression ZSTD. The task finished with TASK OK, and the archive vzdump-qemu-102-2026_05_11-11_34_56.vma.zst is visible in hp14 -> local -> Backups.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Typical export format	OVF package, or OVA when available.	vzdump backup archive such as .vma.zst.
Main UI entry point	VM actions menu in the vSphere Client.	VM Backup tab or vzdump CLI.
Best VM state	Powered off before export.	Powered off, using stop mode for the export.
Portability	OVF is designed for cross-platform import, although device mapping may still need adjustment.	Native Proxmox format; easiest to restore back into Proxmox.
Restore/import path	Deploy OVF Template in vCenter/ESXi.	Restore through Proxmox UI or qmrestore.
Main risk	Browser download interruptions and large multi-file exports.	Treating vzdump as a universal OVF export; it is mainly a Proxmox-native artifact.

Impediments / lessons learned

- Ensure sufficient free storage outside the hypervisor datastore before starting an export — even small guest operating systems produce multi-gigabyte export files. Running out of space mid-export leaves a partial and unusable archive.
- Powering off the VM before export produces a consistent archive without risk of guest filesystem inconsistency — live exports introduce additional complexity that is rarely worth the brief period of uptime saved.
- VMware gives the more portable export format. Proxmox gives a very practical native export/restore workflow, but it is not the same as an OVF export.
- A successful export is not fully proven until the exported files are visible and the matching import/restore method is known.

Implementation walkthrough screenshots

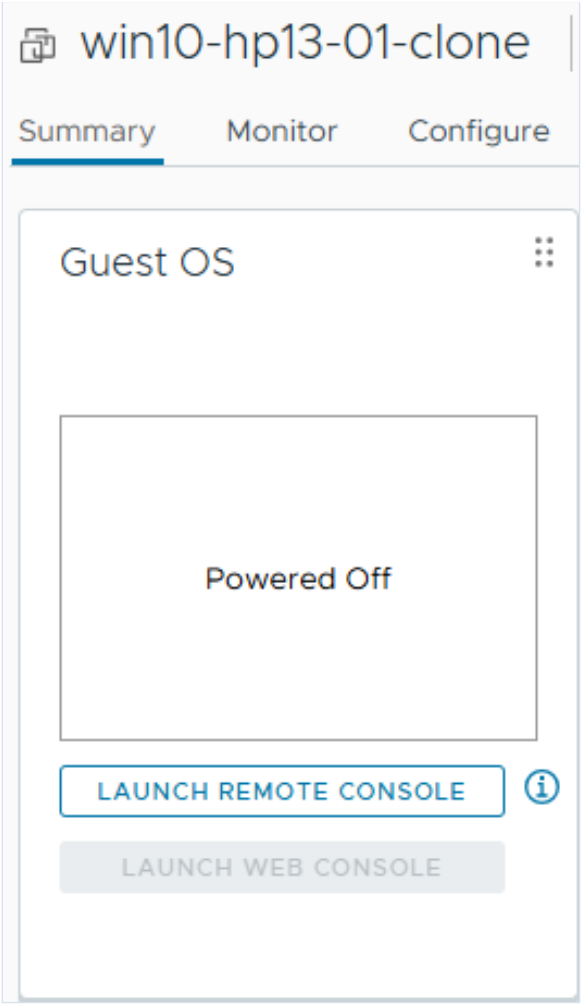


Figure 48 - ESXi source VM powered off before export.

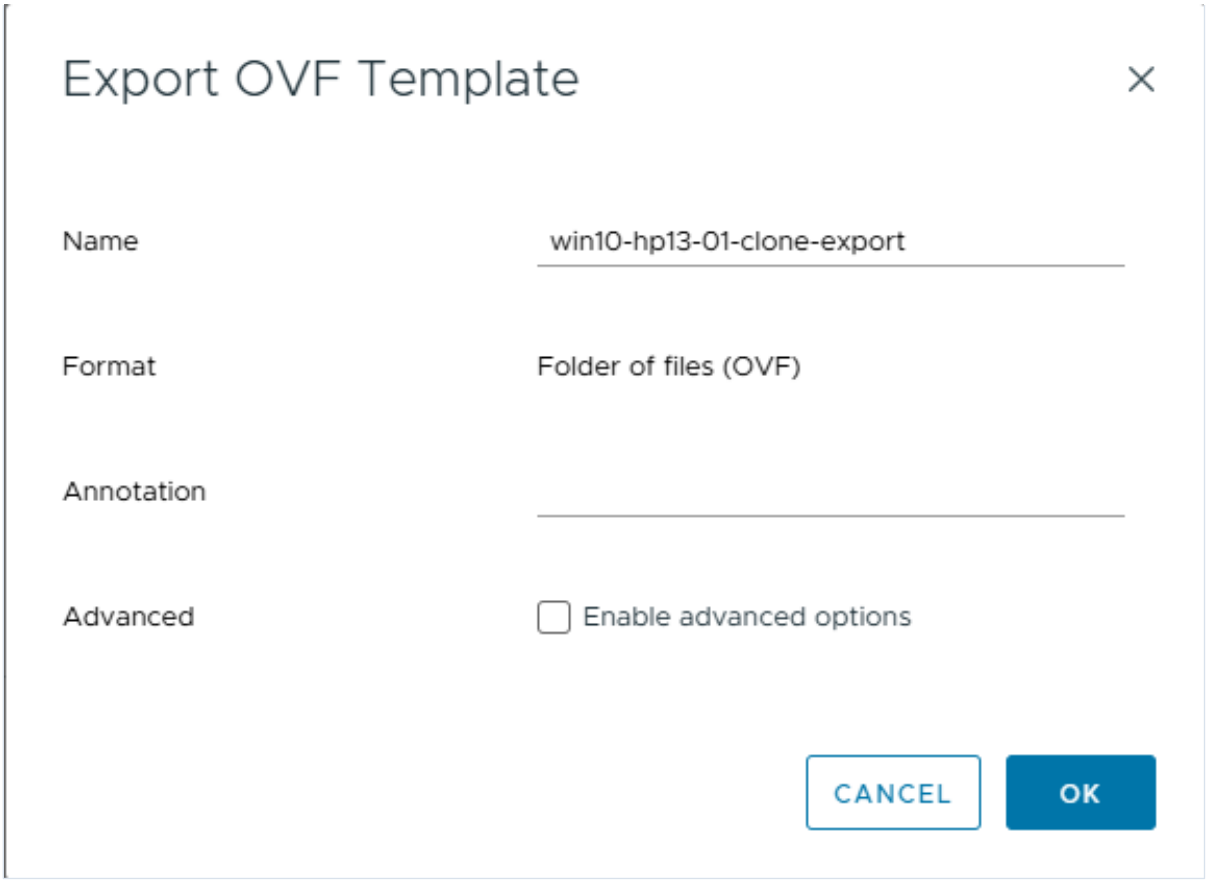


Figure 49 - vCenter export dialog configured for the ESXi VM.

win10-hp13-01-clone-export.mf	18/05/2026 11:37	MF File	1 KB
win10-hp13-01-clone-export-1.vmdk	18/05/2026 11:37	Virtual Machine Di...	18,297,699 ...
win10-hp13-01-clone-export-3.iso	18/05/2026 11:27	Disc Image File	5,420,408 KB
win10-hp13-01-clone-export.ovf	18/05/2026 11:22	Open Virtualizatio...	9 KB
win10-hp13-01-clone-export-2.vmdk	18/05/2026 11:22	Virtual Machine Di...	68 KB
win10-hp13-01-clone-export-4.nvram	18/05/2026 11:22	NVRAM File	265 KB

Figure 50 - Downloaded ESXi OVF export files on the local machine.

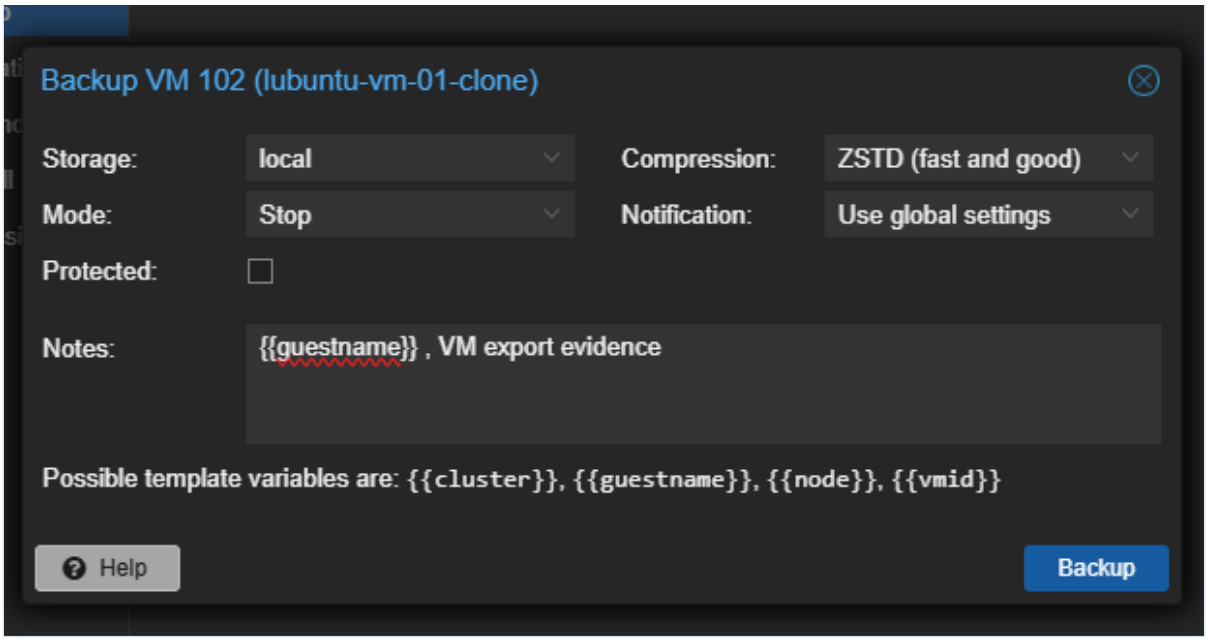


Figure 51 - Proxmox backup dialog used to export `VM 102`.

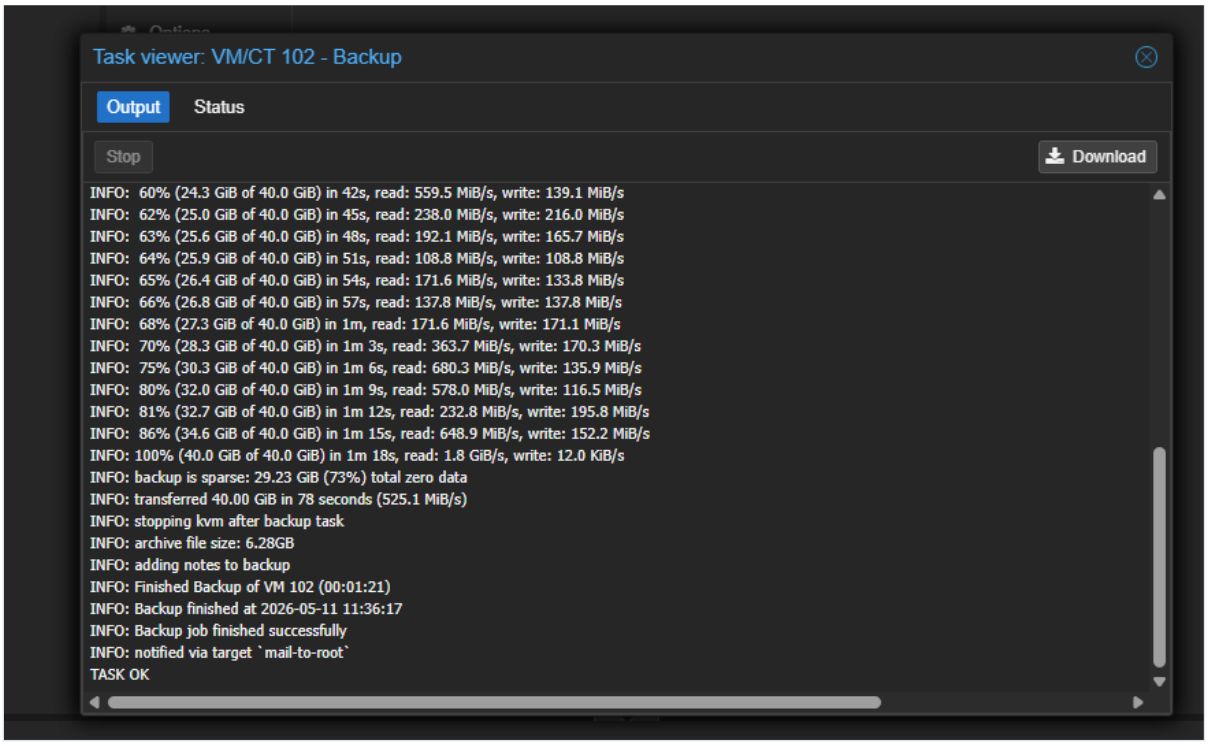


Figure 52 - Proxmox export backup task completed successfully.

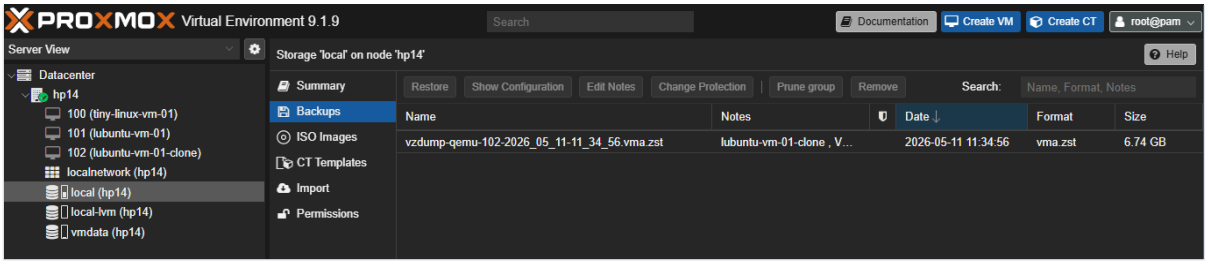


Figure 53 - Exported Proxmox backup archive visible in storage.

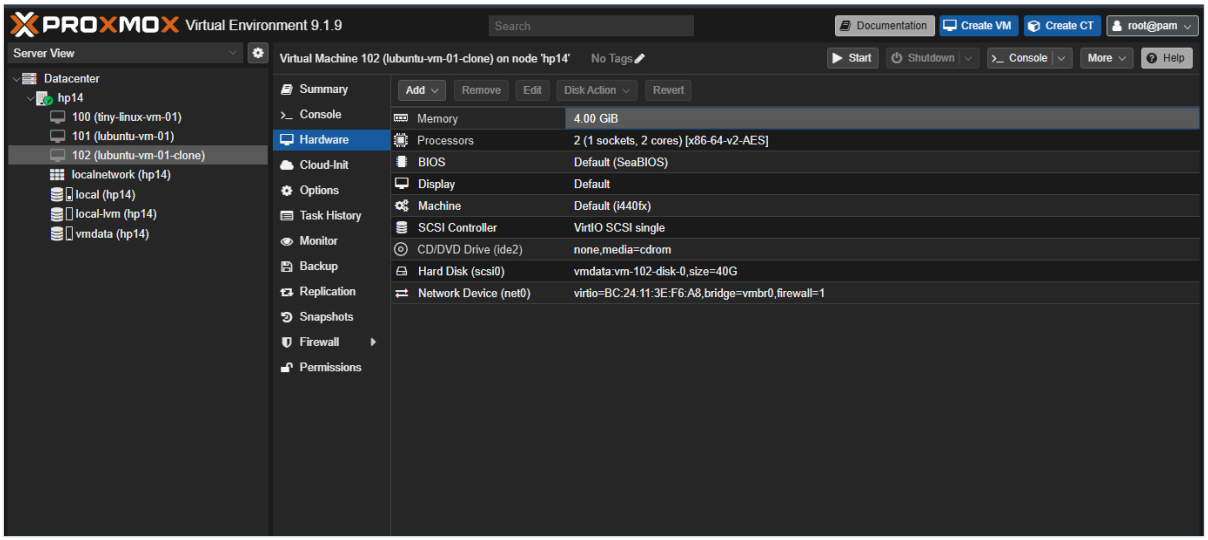


Figure 54 - Proxmox `VM 102` stopped before export.

8. Hypervisor configuration backup

Concept

Hypervisor configuration backup protects the management state of the host itself. It is different from a VM backup: it captures settings needed to rebuild or restore the hypervisor configuration, such as networking, storage definitions, services, users, permissions, repository configuration, and cluster configuration. VM disks and guest operating systems are backed up separately in Chapter 9.

VMware ESXi (+ vCenter)

The ESXi side was completed for HP13 and HP15 using SSH and vim-cmd. On each host, the following commands flushed the configuration and generated a downloadable bundle:

```
vim-cmd hostsvc/firmware/sync_config
```

```
vim-cmd hostsvc/firmware/backup_config
```

The resulting .tgz bundles were downloaded and one bundle was inspected with tar -tzf to prove it contained ESXi configuration files.

Proxmox VE

The Proxmox side was completed for HP14 and HP16 with a shell script that archives /etc/pve and node-local configuration files such as network, hostname, APT repository, filesystem, and backup configuration. The archives were written to the CLOIF2 NFS share under config-backups/proxmox/, so the backup copy is stored outside the node being backed up. Archive contents were verified with tar -tzf.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Backup method	*vim-cmd hostsvc/firmware/backup_config*	Shell script plus tar
Backup output	VMware host configuration .tgz bundle	Custom .tgz archive
Scope	ESXi host configuration	/etc/pve cluster config plus node-local files
Storage target	Downloaded to workstation	CLOIF2 NFS remote storage
Verification	Downloaded bundle and listed archive contents	Listed archive contents and confirmed remote NFS location
VM disks included	No	No

Impediments / lessons learned

- ESXi provides a host configuration bundle through vim-cmd, but the lab build did not expose a simple backup button in the Host Client.
- sync_config was run before backup_config on ESXi so the generated bundle reflected the current host state.
- Proxmox has no single built-in host config bundle action, so the practical solution is to archive /etc/pve plus node-local files.
- A backup stored only on the hypervisor being backed up is weak recovery practice. The Proxmox archives were therefore copied to CLOIF2 NFS storage.

Implementation walkthrough screenshots

```
[root@hp13:~] vim-cmd hostsvc/firmware/sync_config
[root@hp13:~] vim-cmd hostsvc/firmware/backup_config
Bundle can be downloaded at : http://*/downloads/520fbbb6-51d4-1dd7-245a-3c080d8fbdbd/configBundle-hp1
3.vircl.local.tgz
```

Figure 55 - HP13 ESXi configuration backup command and generated download URL.

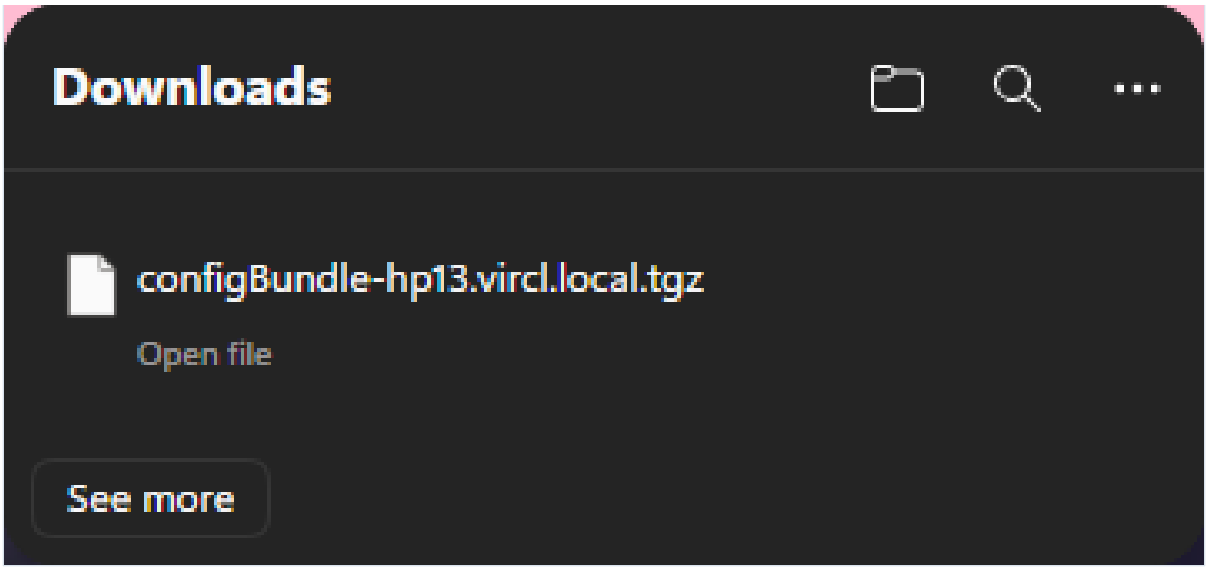


Figure 56 - HP13 ESXi configuration backup bundle downloaded.

```
[root@hp15:~] vim-cmd hostsvc/firmware/sync_config
[root@hp15:~] vim-cmd hostsvc/firmware/backup_config
Bundle can be downloaded at : http://*/downloads/52dacf86-b22c-2e76-3cdb-78aa63d08e41/configBundle-hp1
5.vircl.local.tgz
```

Figure 57 - HP15 ESXi configuration backup command and generated download URL.

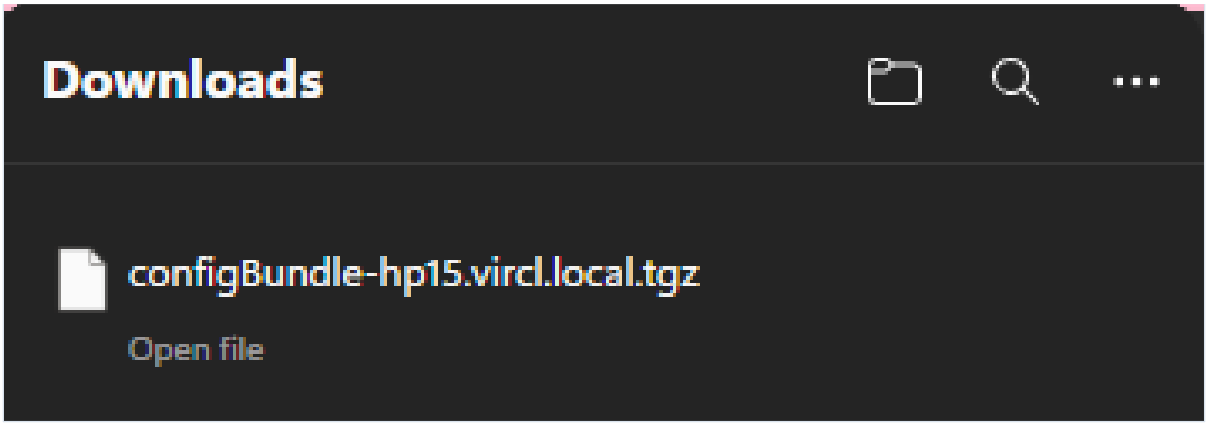


Figure 58 - HP15 ESXi configuration backup bundle downloaded.

```
PS C:\Users\andre> tar -tzf "C:\Users\andre\Downloads\configBundle-hp13.vircl.local.tgz"
./
./Manifest.txt
./jumpstrt.gz
./state.tgz
```

Figure 59 - Extracted ESXi configuration backup bundle contents.

9. VM backup

Concept

Backing up a virtual machine means creating an independent copy of the VM configuration and virtual disks so the VM can be restored after deletion, corruption, failed maintenance, or storage loss. A backup is different from a snapshot: a snapshot is a short-term rollback point that normally still depends on the same VM and datastore, while a backup is a separate recovery artifact.

For this project, a VM backup is considered proven when three things are shown:

- a backup job or export process completed successfully;
- the resulting backup files are visible on backup-capable storage or on the workstation;
- the VM can be restored or the exact restore/import path is documented.

The clean restore method is to restore to a new VM ID/name so the original VM is not overwritten.

VMware ESXi (+ vCenter)

Implementation result: VM `win10-hp13-01-clone` was cold-backed up by exporting it as an OVF package through vCenter. The export completed successfully, producing .ovf, .mf, .vmdk, and .nvram files. To verify recoverability, the OVF package was deployed back into vCenter as a new VM named `win10-hp13-01-restored`. The deployment task completed successfully and the restored VM booted to the Windows desktop, confirming the backup artifact was valid.

Proxmox VE

Proxmox has a built-in VM backup workflow through the web UI and the vzdump command. It can run manual or scheduled backup jobs and stores VM backups as archive files on backup-capable storage. A VM restore can be performed through the web UI or with qmrestore.

Outcome: Gui created a fresh Proxmox backup archive for HP14 `VM 102` (`lubuntu-vm-01-clone`) on 18.05.2026 using vzdump. The archive is vzdump-qemu-102-2026_05_18-11_33_41.vma.zst, stored under local (hp14) -> Backups, and the CLI output ended with Backup job finished successfully.

The archive was restored through the Proxmox web UI to new `VM ID 112` on storage `vmdata`, with the Unique option enabled. `VM 112` appeared in the Proxmox inventory as `lubuntu-vm-01-clone`, confirming that the backup archive was usable for recovery.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Built-in VM backup in this lab	No simple native VM backup job workflow was available in the vCenter/ESXi lab setup.	Built-in backup through the VM Backup tab, Datacenter backup jobs, and vzdump.
Practical lab method	Manual OVF export and verified OVF redeployment.	Manual or scheduled vzdump backup.
Backup artifact	OVF package: .ovf, .vmdk, and optional .mf; OVA if created with a tool.	Proxmox backup archive such as .vma.zst.
Restore path	Deploy OVF Template in vCenter/ESXi.	Web UI restore or qmrestore.
Scheduling	Requires an external backup product, script, or automation outside the basic export workflow.	Built into Proxmox backup jobs.
Consistency model	Cold export is simplest; production backups normally rely on VMware backup APIs and guest quiescing through backup software.	Backup modes include stop/suspend/snapshot depending on storage and guest support; guest agent improves consistency.
Portability	OVF is more portable across VMware and some third-party platforms.	Native to Proxmox; easiest to restore into Proxmox.
Best verification	Export files visible, then deploy them as a new VM.	Archive visible, then restore it as a new VM.

Impediments / lessons learned

- VM backup must be proven by recovery, not only by job completion. A backup that has never been restored is only an assumption.
- Proxmox made the VM backup workflow much faster for the lab because vzdump is integrated into the web UI.
- VMware's OVF export is useful for a manual cold backup/export, but it is not equivalent to a production backup system with scheduling, retention, and application-aware consistency.
- Restoring into a new VM ID/name avoids overwriting the original VM and keeps the test low risk.
- Backup storage needs enough free space. The HP14 Proxmox `VM 102` backup compressed a 40 GiB virtual disk to about 6.28 GiB, but larger VMs can quickly exceed local storage.

Implementation walkthrough screenshots

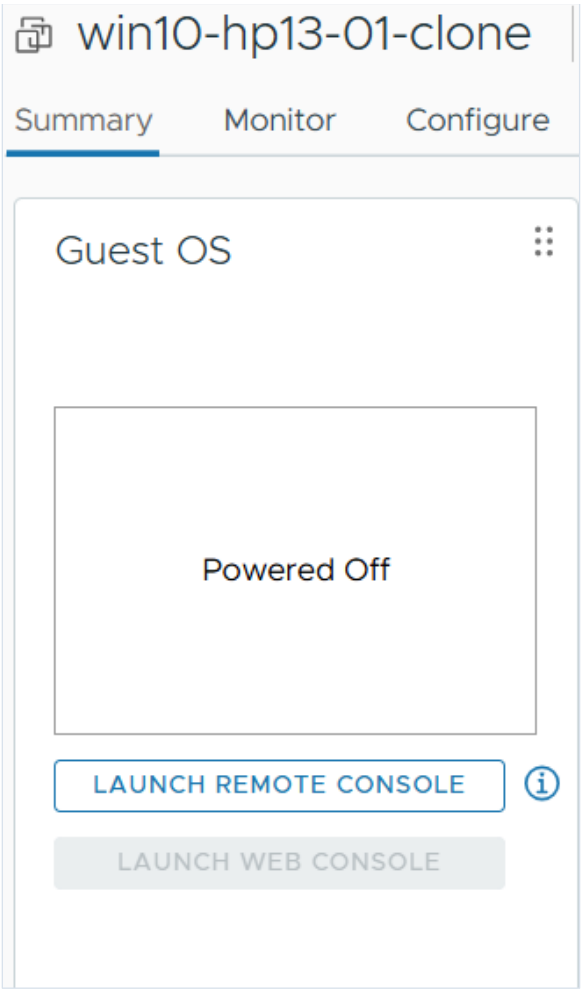


Figure 60 - ESXi source VM powered off before VM backup/export.

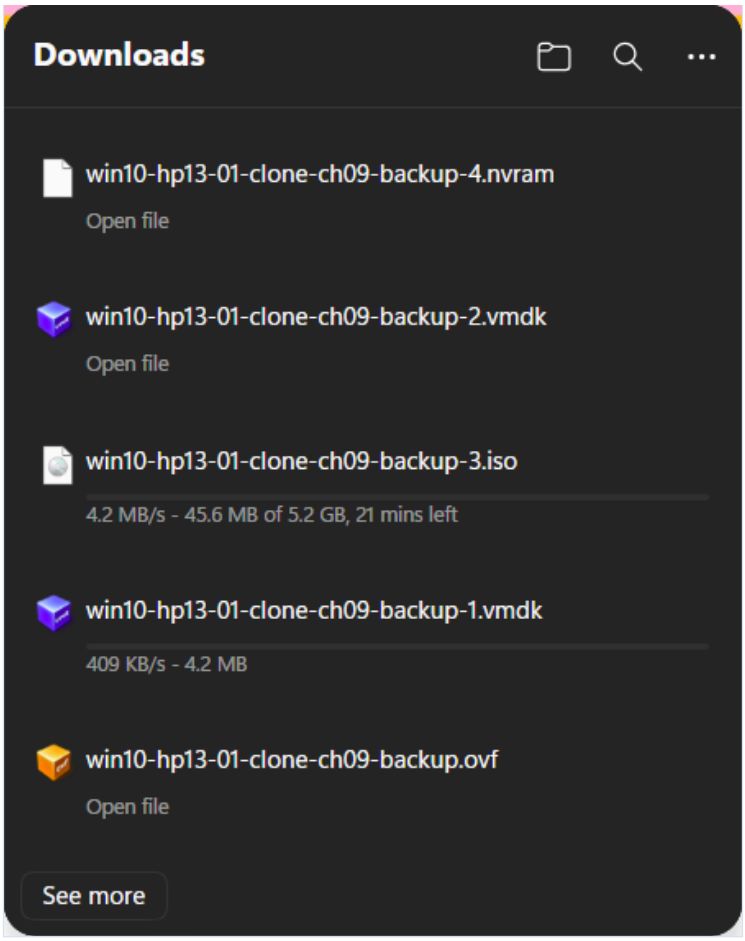


Figure 61 - vCenter export dialog used for the ESXi VM backup.

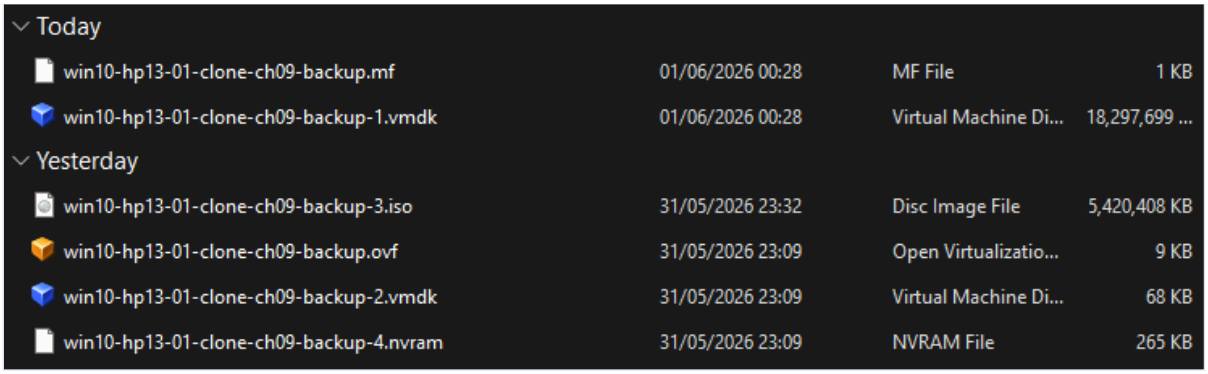


Figure 62 - Downloaded ESXi VM backup files.

Task Name	Target	Status
Deploy OVF template	VIRCL-Cluster	Completed
Import OVF package	10.0.10.131	Completed

Figure 63 - ESXi VM restore/deploy task completed successfully.

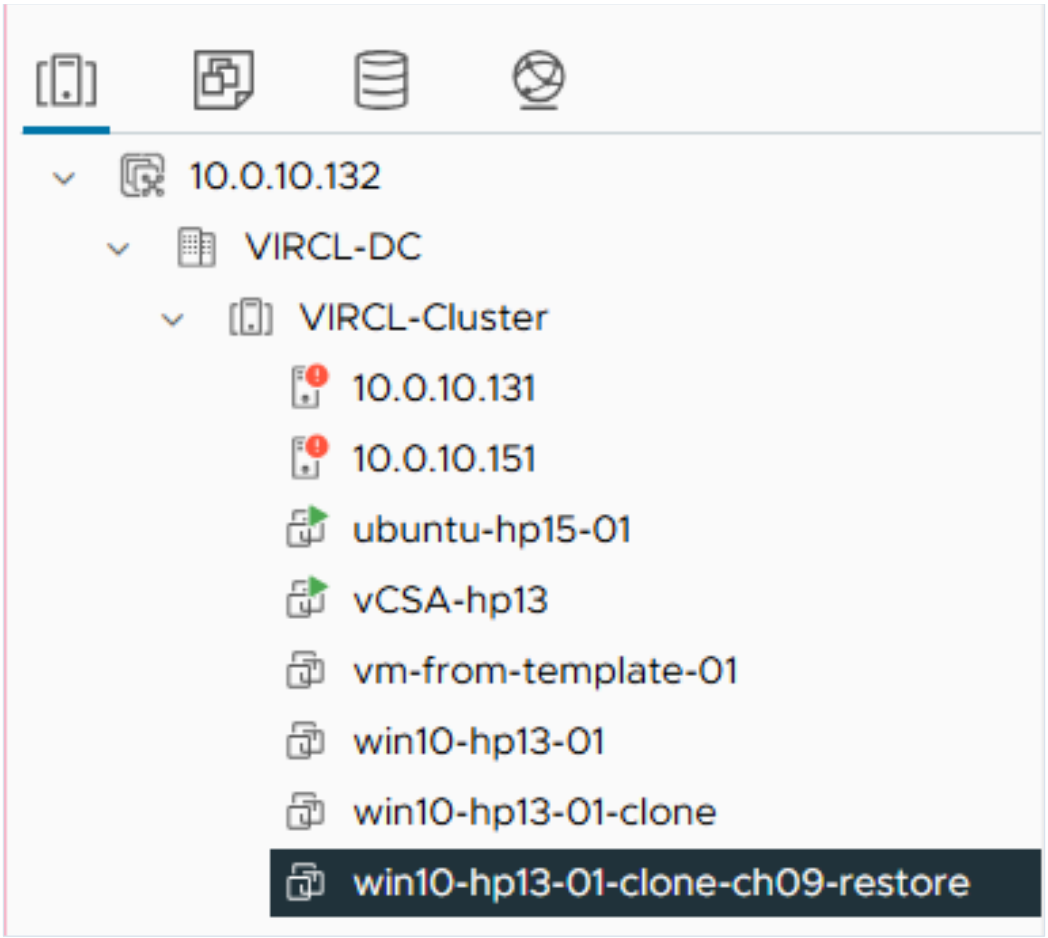


Figure 64 - Restored ESXi VM visible in the vCenter inventory.

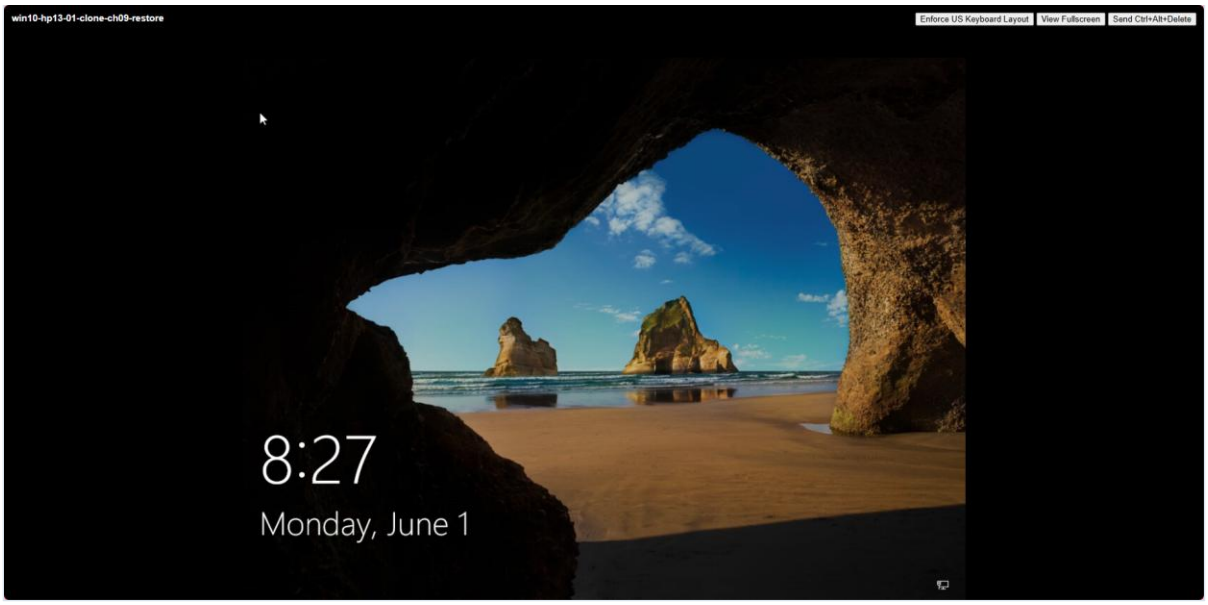


Figure 65 - Restored ESXi VM booted successfully.

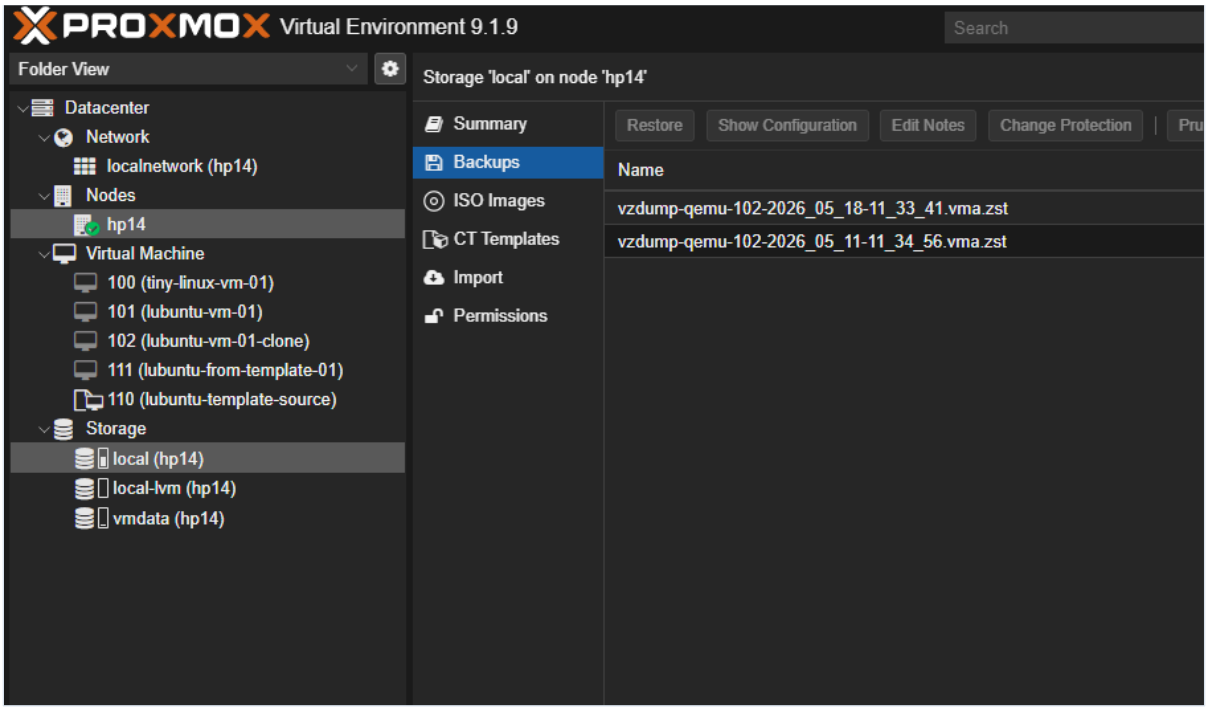


Figure 66 - Proxmox VM backup archive visible before restore.

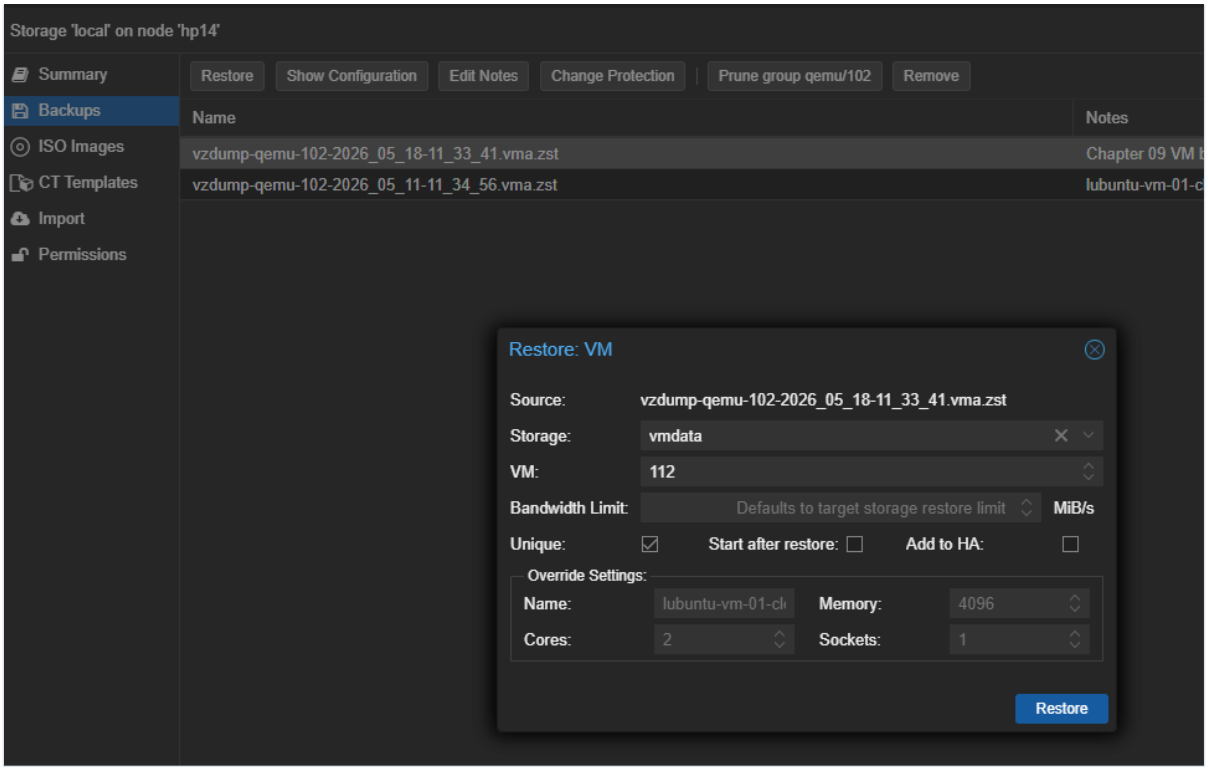


Figure 67 - Proxmox restore dialog configured for new `VM 112`.

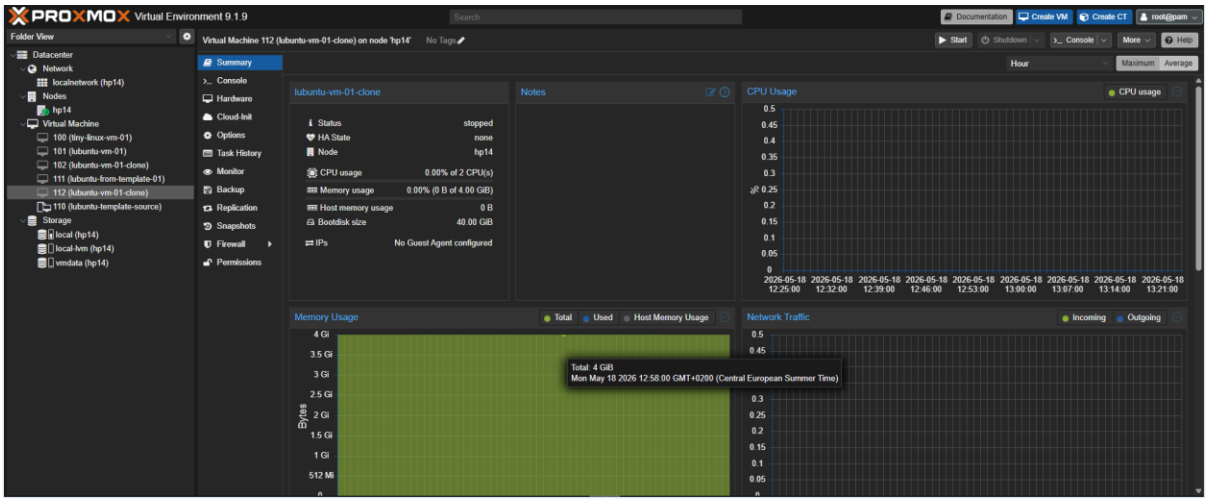


Figure 68 - Restored Proxmox `VM 112` visible after restore.

10. Snapshots

Concept

A VM snapshot records the state of a virtual machine at a specific point in time. It normally captures the VM configuration and virtual disk state, and can optionally include the running memory state. Snapshots are useful before risky changes because they give the administrator a quick rollback point.

Snapshots are not a replacement for backups. A backup creates an independent copy that can survive datastore loss or VM deletion. A snapshot usually remains attached to the original VM and depends on the same storage. For this project, the important operations are:

- a snapshot was created before a test change;
- a test marker file was created inside the VM;
- the VM was reverted to the snapshot and the marker disappearance proved the rollback;
- the snapshot was deleted after the test so it did not keep consuming storage.

VMware ESXi (+ vCenter)

Result: A snapshot named `ch10-test-snapshot` was created on VM `win10-hp13-01-clone` running on HP13. After snapshot creation, a test marker file `snapshot-proof.txt` was created on the Windows desktop. The VM was then reverted to `ch10-test-snapshot`. Upon reboot, the marker file was absent, confirming the rollback restored the pre-snapshot state. The snapshot was then deleted and the Snapshot Manager showed a clean state.

Proxmox VE

Outcome: Gui used HP14 `VM 102` (`lubuntu-vm-01-clone`) as the safe test VM. A snapshot named `ch10_before_marker` was created while the VM was stopped. After the snapshot, a marker file named chapter10_after_snapshot.txt was created on the Lubuntu desktop. The VM was rolled back to `ch10_before_marker`, then booted again; the marker file was gone, proving that the rollback restored the earlier VM state. Snapshot deletion was confirmed through the Proxmox web UI; the snapshot list returned to empty after cleanup.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Main UI entry point	VM context menu or VM Snapshots tab in vCenter.	VM Snapshots tab in the Proxmox web UI.
Snapshot content	VM disk/configuration state, optionally memory.	VM disk/configuration state, optionally RAM state.
Typical lab proof	Snapshot created, guest state changed, reverted to snapshot, and guest state verified as returned to pre-snapshot.	Snapshot created, guest state changed, rolled back to snapshot, and guest state verified as returned to pre-snapshot.
Deletion behavior	Consolidates/removes snapshot delta data back into the base disk chain.	Removes the snapshot and its delta/reference data from the VM storage.
Main risk	Long-lived snapshots can grow and affect datastore space/performance.	Long-lived snapshots can grow and consume storage; rollback discards newer VM state.
Best practice	Use short-lived snapshots before maintenance, then delete them.	Use short-lived snapshots before maintenance, then delete them. VM disks must be on qcow2 or LVM-thin storage — raw-format disks do not support snapshots.

Impediments / lessons learned

- Snapshots are best used as temporary rollback points, not long-term protection.
- The revert test must use a disposable VM or clone, because rollback intentionally discards changes made after the snapshot.
- For clean verification, create a simple visible marker inside the guest OS after the snapshot, then show that it disappears after rollback.
- Delete the snapshot after testing. Leaving snapshots open creates hidden storage growth and can confuse later backup/export tests.

Implementation walkthrough screenshots

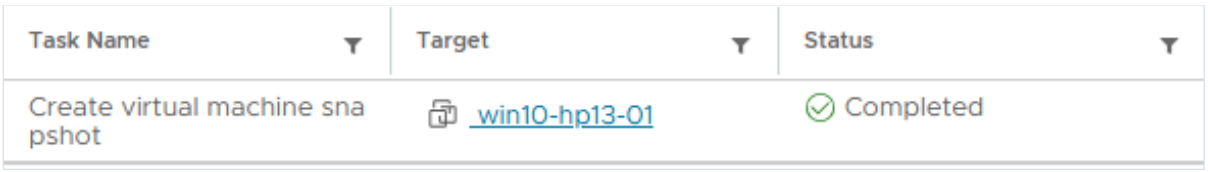


Figure 69 - vCenter snapshot creation task completed successfully.

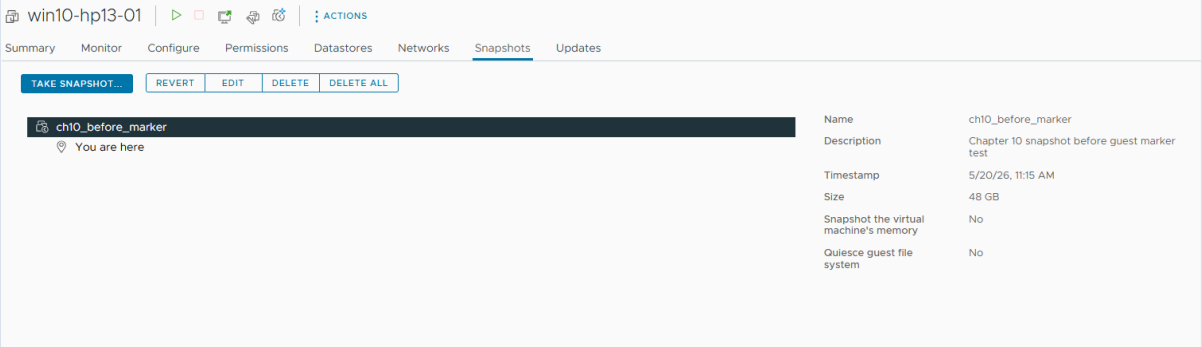


Figure 70 - ESXi snapshot visible in Snapshot Manager.

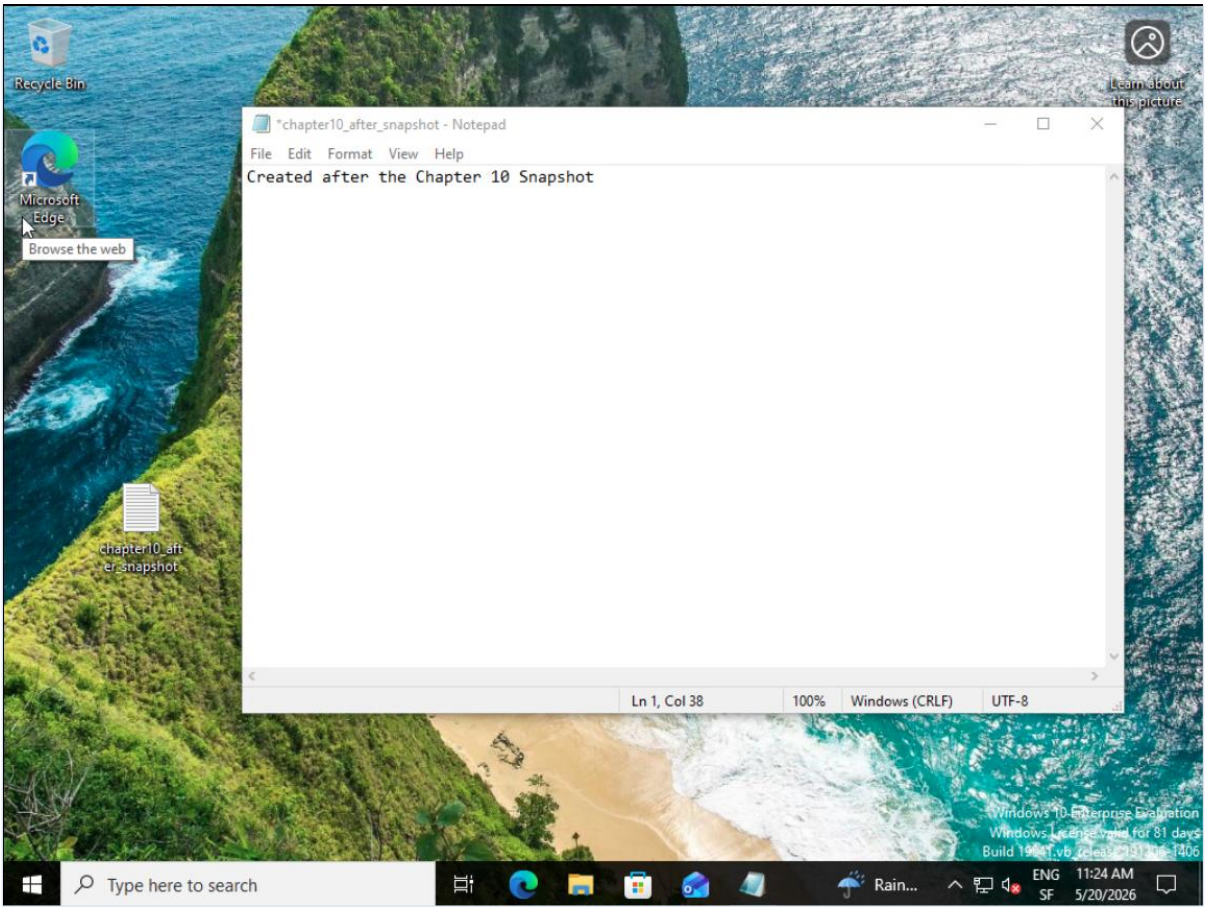


Figure 71 - Test marker file created inside the guest after snapshot.

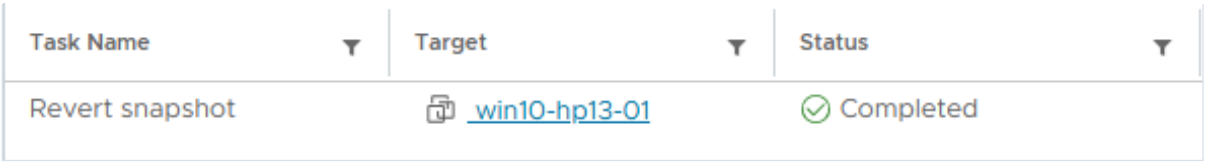


Figure 72 - vCenter snapshot revert task completed successfully.

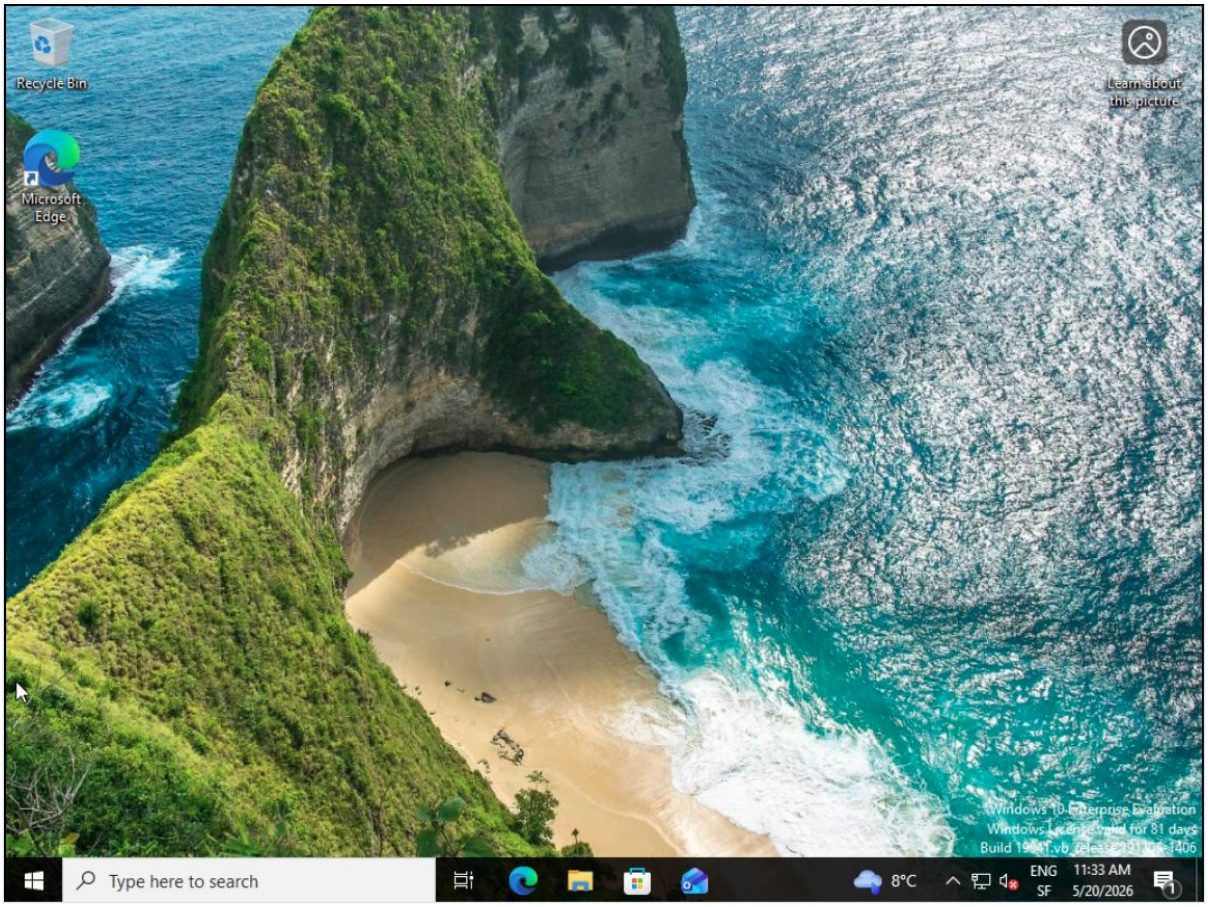


Figure 73 - Guest marker file removed after reverting the ESXi snapshot.

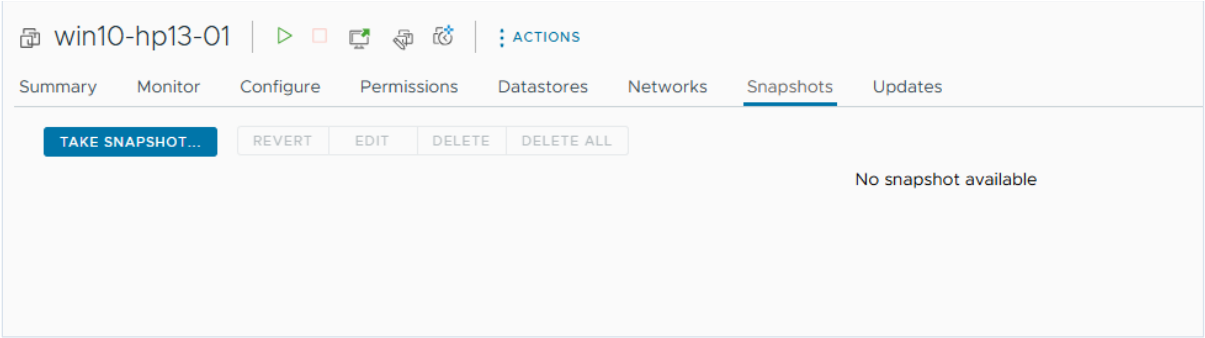


Figure 74 - ESXi Snapshot Manager clean after snapshot deletion.

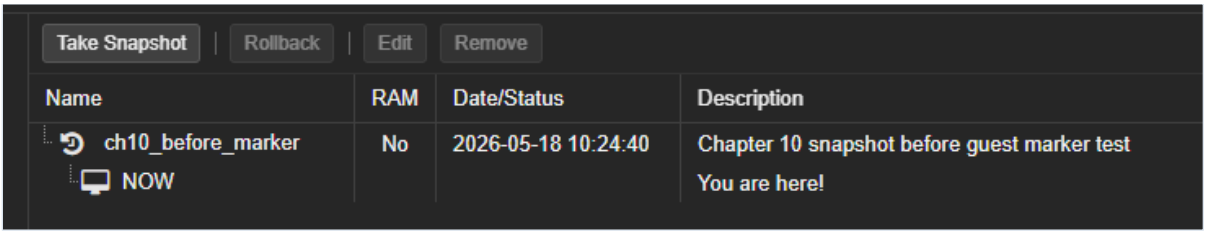


Figure 75 - Proxmox snapshot created and listed for `VM 102`.

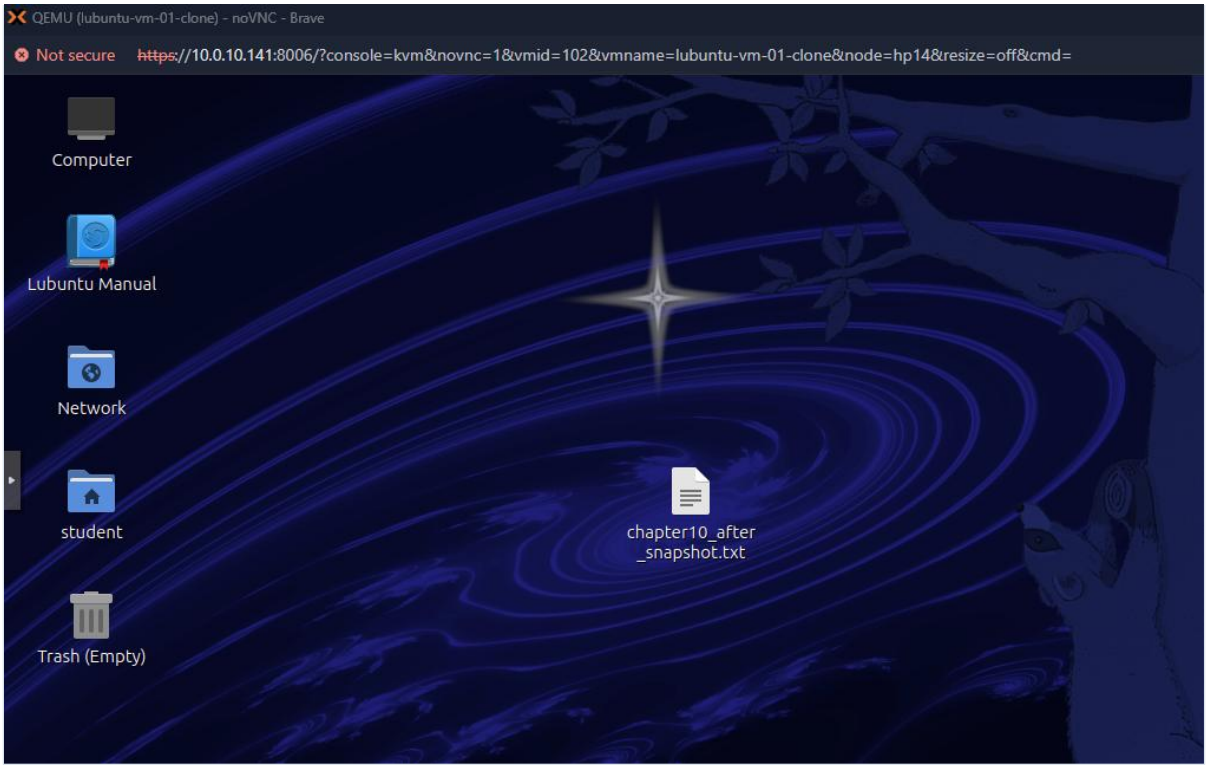


Figure 76 - Test marker created inside the Proxmox guest after snapshot.

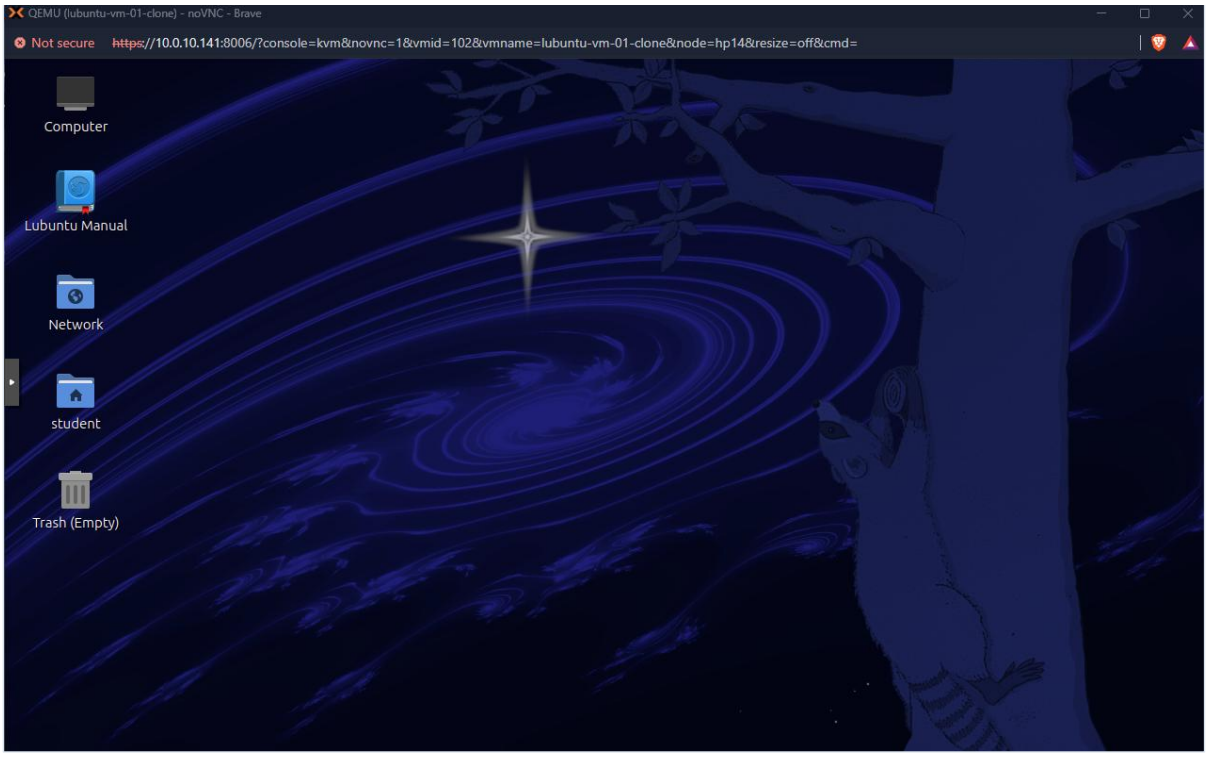


Figure 77 - Guest marker removed after Proxmox snapshot rollback.

11. Templates

Concept

A VM template is a prepared virtual machine image used as a repeatable starting point for new VMs. Instead of installing the operating system and applying the same base configuration every time, the administrator prepares one clean source VM, converts or clones it into a template, and then deploys new VMs from that template.

Templates are useful when many similar VMs are needed, because they reduce manual work and make the deployed machines more consistent. The lab template was kept as a clean base image for repeated deployment. In production, Windows templates are normally generalized with Sysprep and Linux templates are cleaned so they do not carry duplicated identity values such as stale SSH host keys or /etc/machine-id.

For this lab, the proof is:

- a clean source VM was cloned into a template;
- a new VM was deployed from that template;
- the deployed VM appeared as a normal VM;
- the template remained available for future deployments.

VMware ESXi (+ vCenter)

Implementation result: VM `win10-hp13-01-clone` was cloned to template `win10-template-01` using Clone → Clone to Template in vCenter, preserving the original VM. A new VM `win10-from-template-01` was then deployed from `win10-template-01` onto datastore `HP13-vmstore`. The deployment completed successfully, the new VM appeared in the vCenter inventory, and it booted to the Windows desktop with the expected hostname and IP configuration.

Proxmox VE

Outcome: on HP14, the existing Lubuntu VM was cloned into disposable `VM 110`, `VM 110` was converted into a Proxmox template, and `template 110` was cloned into `VM 111`. `VM 111` appears as a normal VM in the Proxmox inventory, while `template 110` remains visible and reusable.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Main UI entry point	vCenter VM context menu: Clone → Clone to Template or Template → Convert to Template.	Proxmox VM context menu: Convert to template, then clone from the template.
Template behavior	Template is a separate inventory object when cloned to template; it is deployed through New VM from This Template.	Template is a VM marked as a template; it cannot be started directly and is used as a clone source.
Safest lab approach	Clone an existing VM to a template so the original VM stays usable.	Clone an existing VM to a disposable source first, then convert only that disposable source into a template.
Deployment operation	Deploy a new VM from template through the vCenter wizard.	Clone the template into a new VM ID, usually as a full clone for independence.
Guest customization	vCenter can use customization specifications, especially for Windows/Sysprep workflows.	Proxmox usually relies on manual cleanup, cloud-init images, or post-clone guest changes.
Main risk	Converting the wrong VM to a template removes it from normal powered-on VM use until converted back or cloned.	Converting a VM to template is irreversible without cloning back — the source VM can no longer be started directly. Always convert a disposable clone, not the only working copy.
Best practice	Patch and generalize the source VM before templating; keep templates clean and versioned.	Use a minimal clean source, prefer cloud-init for scalable deployments, and give cloned VMs unique identity/network settings.

Impediments / lessons learned

- A template is not a backup. It is a deployment source and was protected through the separate backup/export workflow.
- Do not convert the only useful VM directly into a template. Create a disposable clone or use the platform's clone-to-template workflow.
- Template creation alone is not sufficient proof — a working VM must be deployed from the template to demonstrate it functions as a repeatable deployment source.
- Guest identity matters after deployment. Two cloned VMs may share hostname, machine ID, or Windows SID unless the guest is generalized.
- For this project, a full clone from the template is easier to explain than a linked clone because the deployed VM is independent from the template storage chain.

Implementation walkthrough screenshots

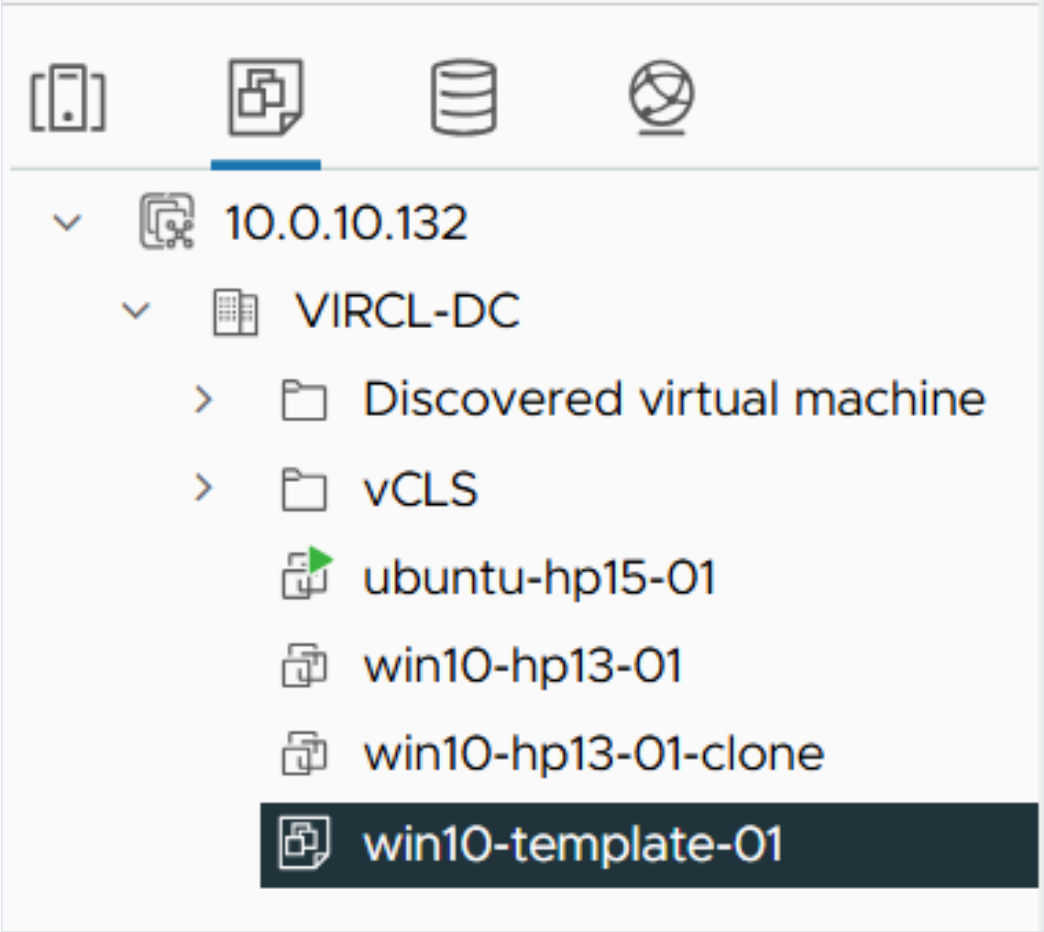


Figure 78 - ESXi VM template visible in the vCenter inventory.

Task Name	Target	Status
Clone virtual machine	win10-template-01	Completed

Figure 79 - vCenter deployment from template completed successfully.

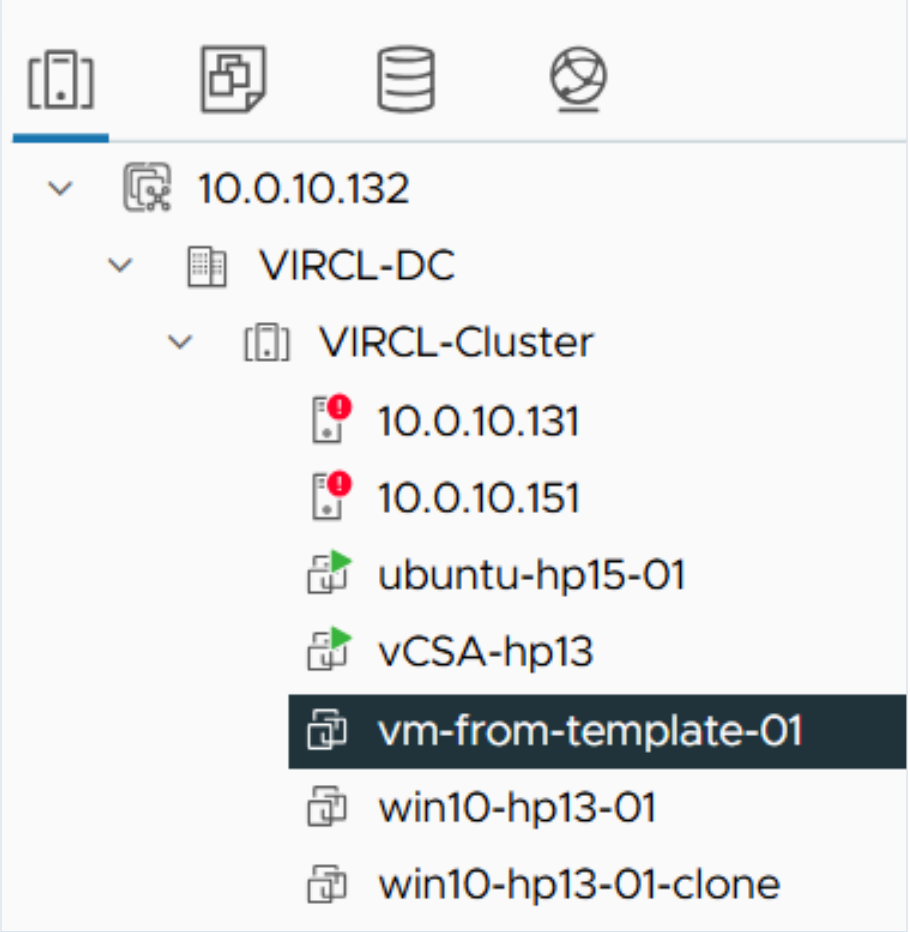


Figure 80 - VM deployed from ESXi template visible in inventory.

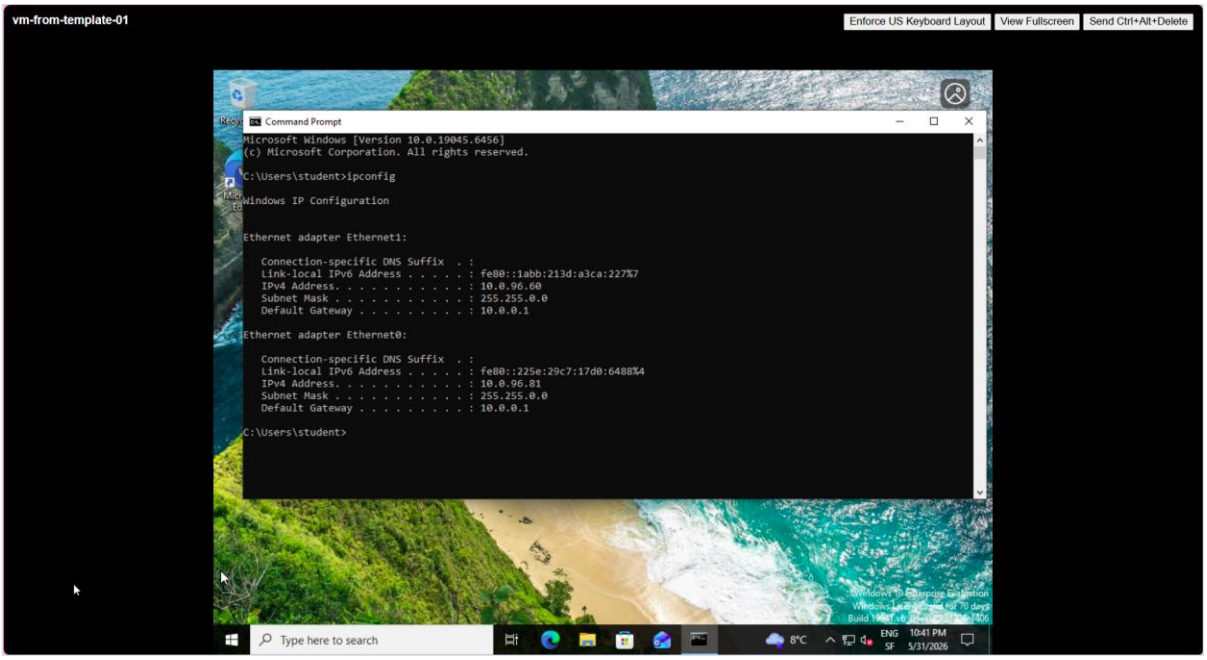


Figure 81 - VM deployed from ESXi template booted with hostname and IP.

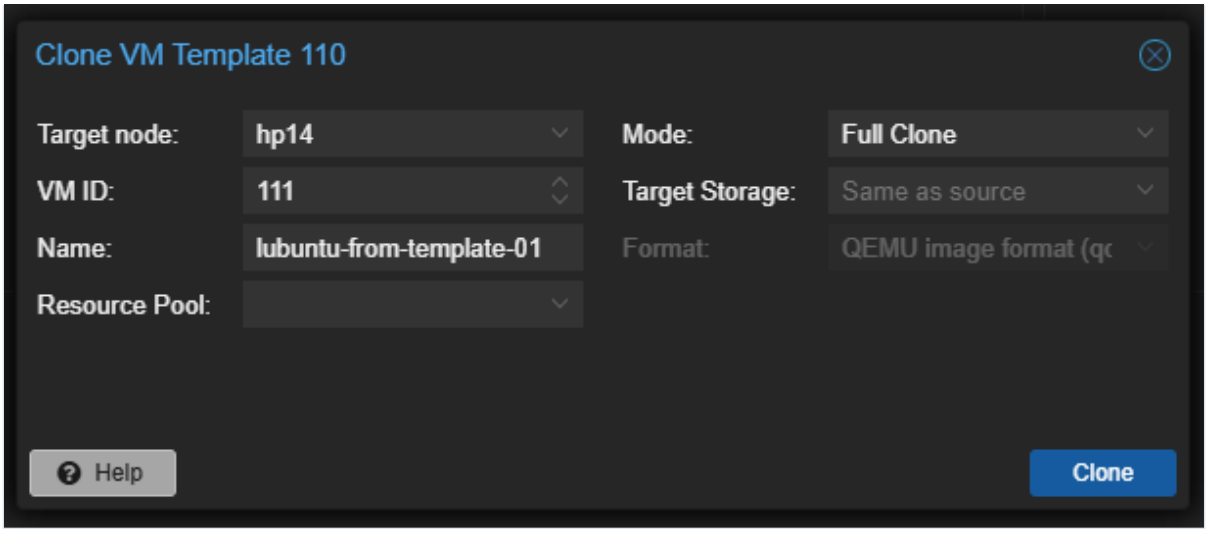


Figure 82 - Proxmox template clone dialog for new `VM 111`.

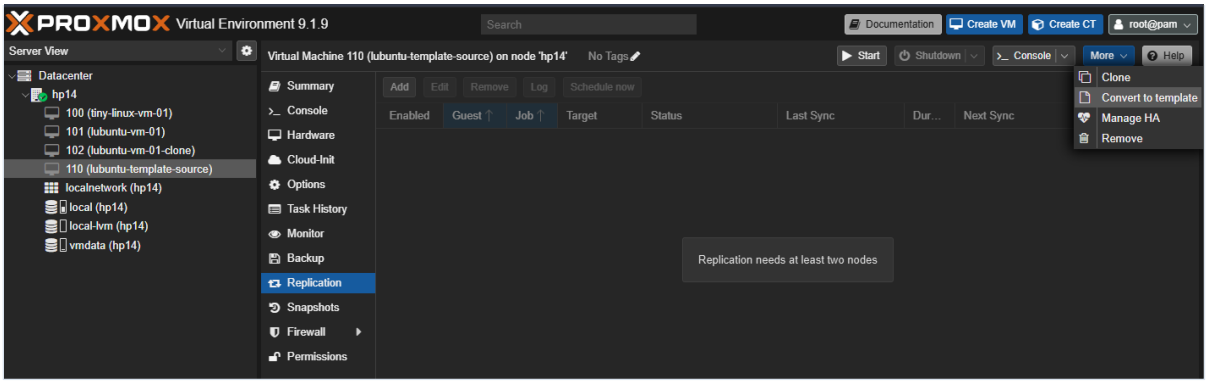


Figure 83 - Proxmox menu option used to convert `VM 110` into a template.

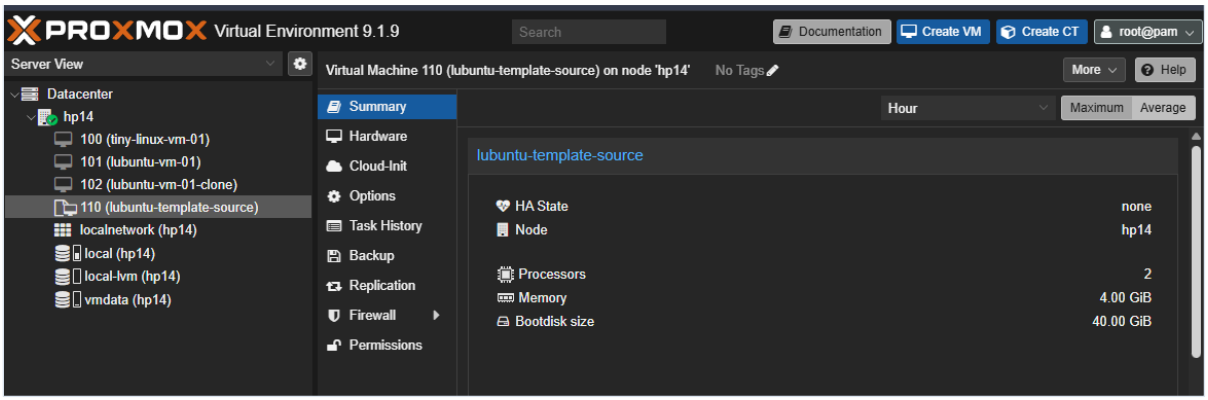


Figure 84 - Proxmox template icon confirming `VM 110` became a template.

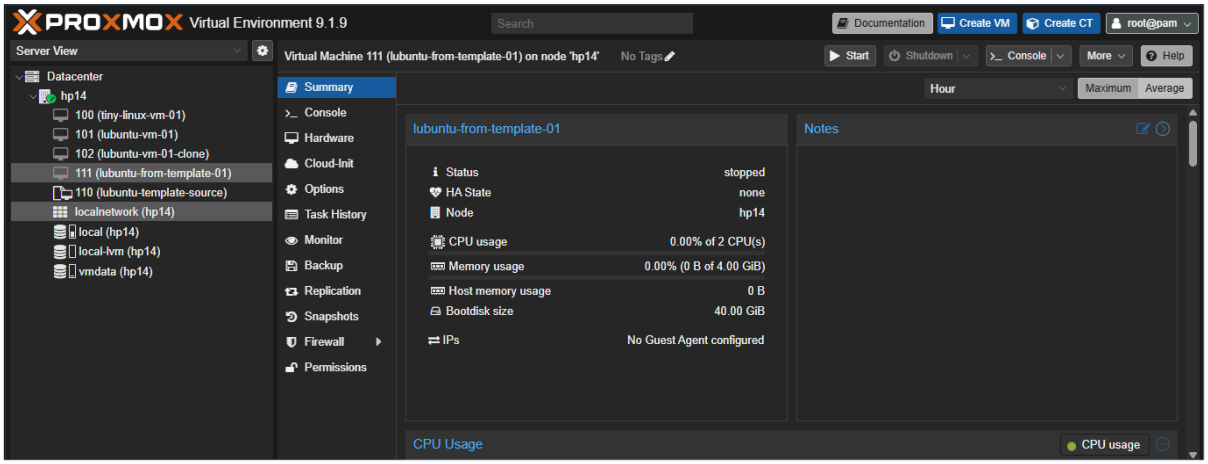


Figure 85 - Proxmox `VM 111` created successfully from template.

12. Remote storage

Concept

Remote storage lets VM files live on the CLOIF2 NAS instead of only on local disks inside HP13-HP16. This is required before storage migration, backup-to-NAS workflow, and clustering/high-availability tests make sense. It also separates VM storage from the local hypervisor disks, which is closer to how shared storage is used in production virtualized environments.

Protocol decision

Selected protocols: NFS + iSCSI.

Reasoning:

- NFS is the fastest practical path for the lab. It is file-based, simple to export from Unraid, and usable by both VMware ESXi/vCenter and Proxmox VE as shared storage.
- iSCSI gives the required contrast with NFS: block storage instead of file storage. ESXi can place VMFS on an iSCSI LUN, and Proxmox can use iSCSI directly or with LVM on top.
- SMB/CIFS is not selected as a main hypervisor datastore protocol. It remains useful for normal NAS file access and ISO copying, but it is weaker for the ESXi datastore side and would not demonstrate the storage model as cleanly as NFS + iSCSI.
- Unraid note: SMB and NFS are standard Unraid share services. iSCSI is available through the Unraid Community Applications plugin. Both NFS and iSCSI were configured successfully on CLOIF2 for both the ESXi/vCenter and Proxmox VE sides.

VMware ESXi (+ vCenter)

The ESXi/vCenter side uses CLOIF2 as remote storage through NFS and iSCSI. The walkthrough shows:

- vmkping connectivity from HP13 and HP15 to the NAS.
- NFS datastore creation through the vCenter datastore wizard.
- The datastore creation task completing successfully.
- The datastore visible from HP13 and from both ESXi hosts.
- A write test proving the datastore is usable.
- iSCSI software adapter creation.
- iSCSI target discovery and LUN visibility.
- VMFS datastore creation on the iSCSI LUN.
- Final iSCSI datastore summary in vCenter.

Proxmox VE

HP14 uses dedicated Unraid storage objects for the Proxmox-side configuration:

Protocol	Unraid object	Proxmox object	Purpose
NFS	Share vircl-proxmox-nfs, export /mnt/user/vircl-proxmox-nfs	cloif2-proxmox-nfs	File-based remote VM storage
iSCSI	Target iqn.2026-05.vircl.team4:proxmox-hp14, FileIO iscsi_proxmox_hp14	cloif2-proxmox-iscsi + cloif2-proxmox-iscsi-lvm	Block-based remote VM storage

The walkthrough shows:

- The Unraid share was created specifically for Proxmox to avoid confusion with the team/general or ESXi-side share.
- NFS was enabled with private security and restricted to HP14.
- HP14 could see the export using showmount -e 10.0.13.4.
- Proxmox added the export as storage ID `cloif2-proxmox-nfs`.
- Proxmox marked the storage as shared and enabled.
- df -h showed the mounted NFS filesystem.
- A file named proxmox-write-test.txt was created on the mount, proving write access.
- A Proxmox-only iSCSI target was created instead of reusing the existing iscsi_cloif target that was already mapped to VMware/Microsoft.
- HP14's initiator iqn.1993-08.org.debian:01:fb284f7ddd91 was mapped to the Proxmox iSCSI target.
- A dedicated FileIO volume iscsi_proxmox_hp14 was created and mapped as LUN 0.
- Proxmox added the target as `cloif2-proxmox-iscsi`.
- Proxmox added LVM on top as `cloif2-proxmox-iscsi-lvm`, which is the usable VM disk storage.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
NAS platform	CLOIF2 / Unraid	CLOIF2 / Unraid
Main protocol shown	NFS + iSCSI	NFS + iSCSI
Management path	vCenter datastore wizard	Proxmox Datacenter storage configuration
Storage object name	vCenter datastore / VMFS datastore	Proxmox storage IDs cloif2-proxmox-nfs, cloif2-proxmox-iscsi, cloif2-proxmox-iscsi-lvm
Client verification	vmkping, datastore visibility, write test	showmount, df -h, write test, iSCSI target + LVM visible
Current iSCSI status	Complete: target, LUN, VMFS datastore and summary shown	Complete: target, FileIO LUN, initiator mapping, Proxmox iSCSI and LVM storage shown

Impediments / lessons learned

- Naming matters. The Proxmox share was named `vircl-proxmox-nfs` and the Proxmox storage ID was named `cloif2-proxmox-nfs` so it cannot be confused with Andrea's ESXi work or the general `vircl-team4` share.
- NFS is the right first remote-storage protocol in this lab because it is quick to validate end to end on both platforms.
- iSCSI is now complete on both sides. On Proxmox, the important detail is that the raw iSCSI target is not the final usable VM storage by itself; LVM was added on top of the LUN.
- The existing CLOIF iSCSI backstore iscsi_cloif was not reused for Proxmox because it was already mapped to VMware/Microsoft. A separate Proxmox-only FileIO volume avoided unsafe multi-system writes to the same block device.
- The Unraid iSCSI plugin UI is not very explicit: the FileIO volume must be created on the FileIO tab first, then attached as a LUN, then mapped to the initiator. The mapping field required the full `fileio;iscsi_proxmox_hp14` value.

Implementation walkthrough screenshots

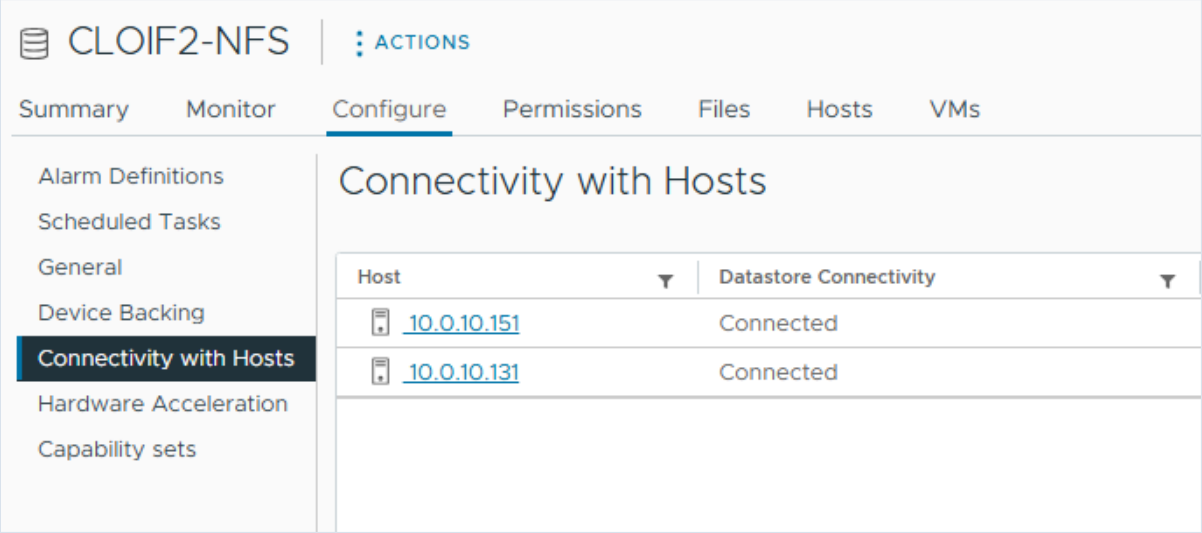


Figure 86 - CLOIF2 NFS datastore visible to both ESXi hosts.

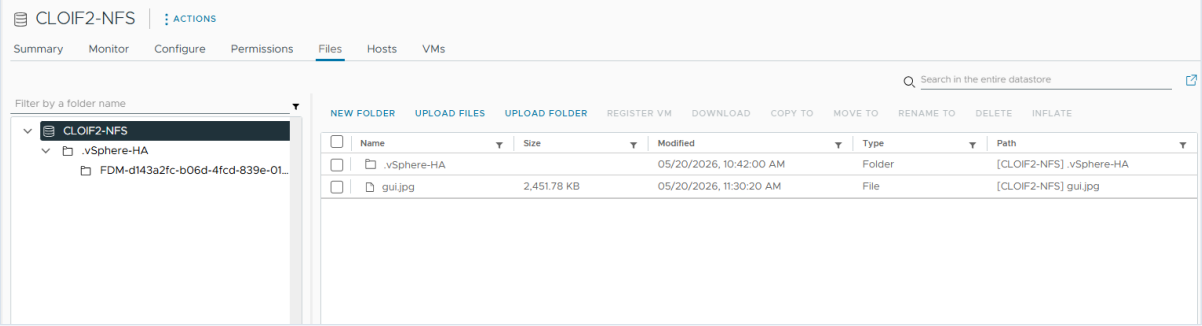


Figure 87 - ESXi NFS write test proving datastore access.

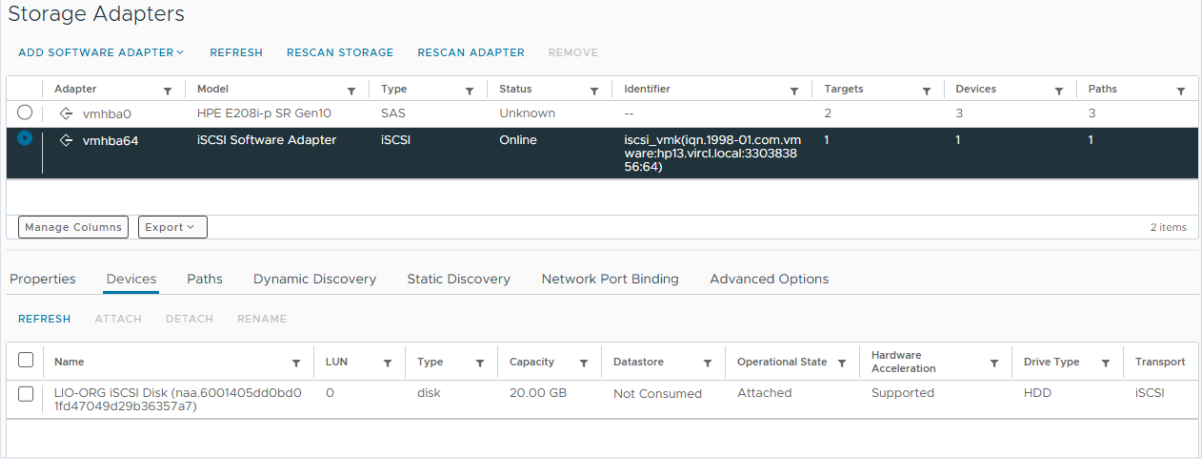


Figure 88 - iSCSI LUN visible from the ESXi storage adapter.

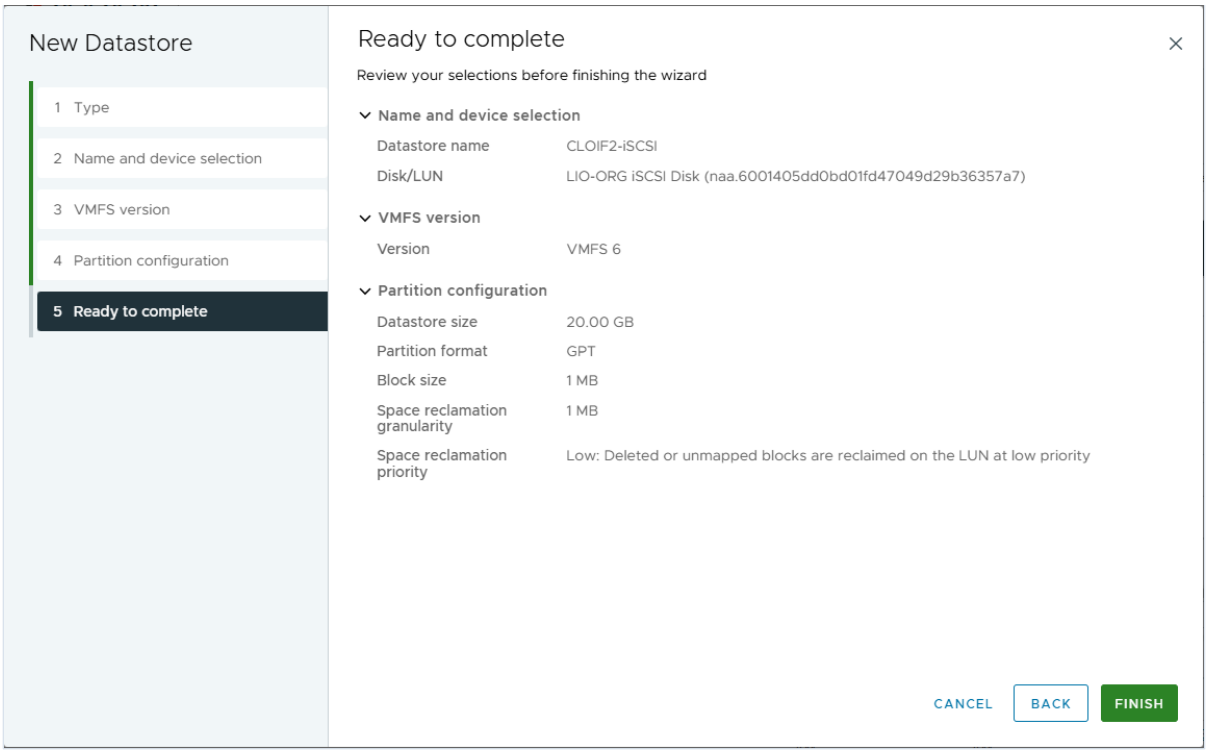


Figure 89 - ESXi iSCSI datastore created successfully.

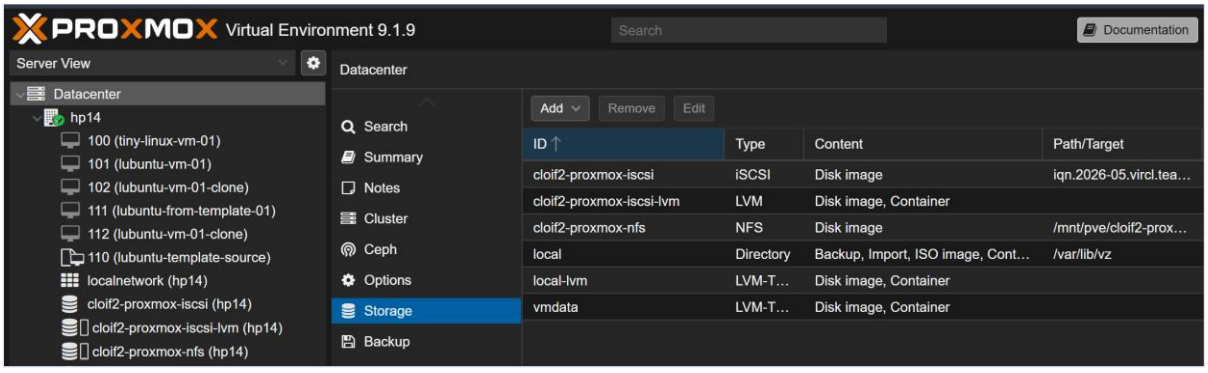


Figure 90 - Final Proxmox storage list with iSCSI and LVM entries.

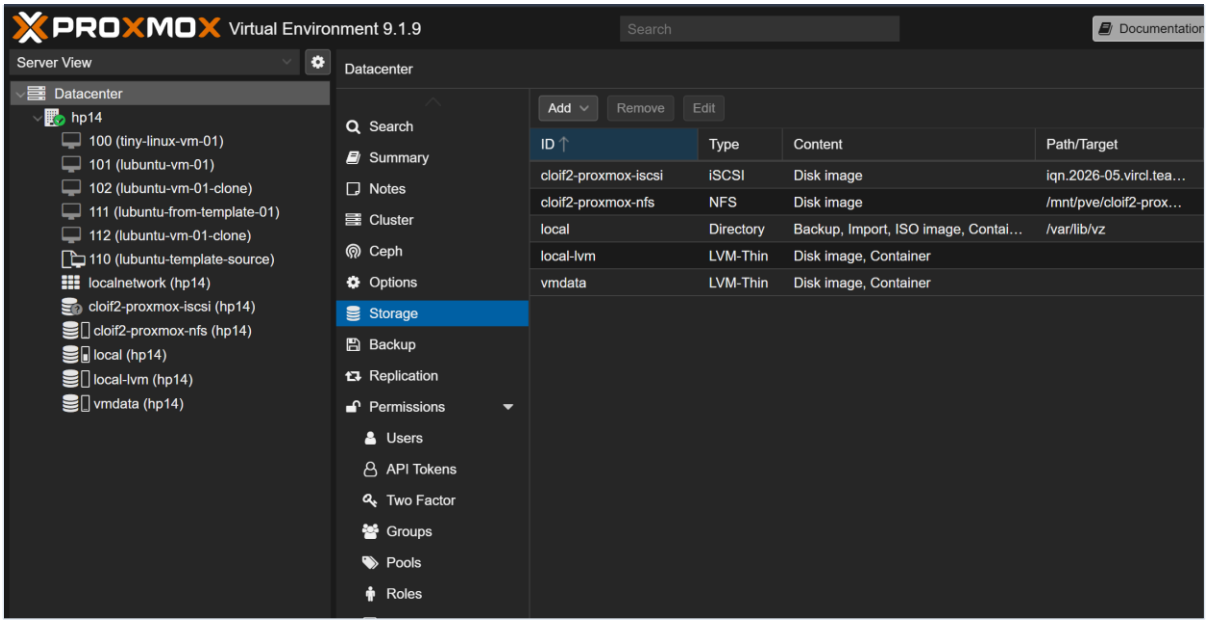


Figure 91 - Proxmox iSCSI storage visible after configuration.



Figure 92 - CLOIF2 NFS storage visible in Proxmox.

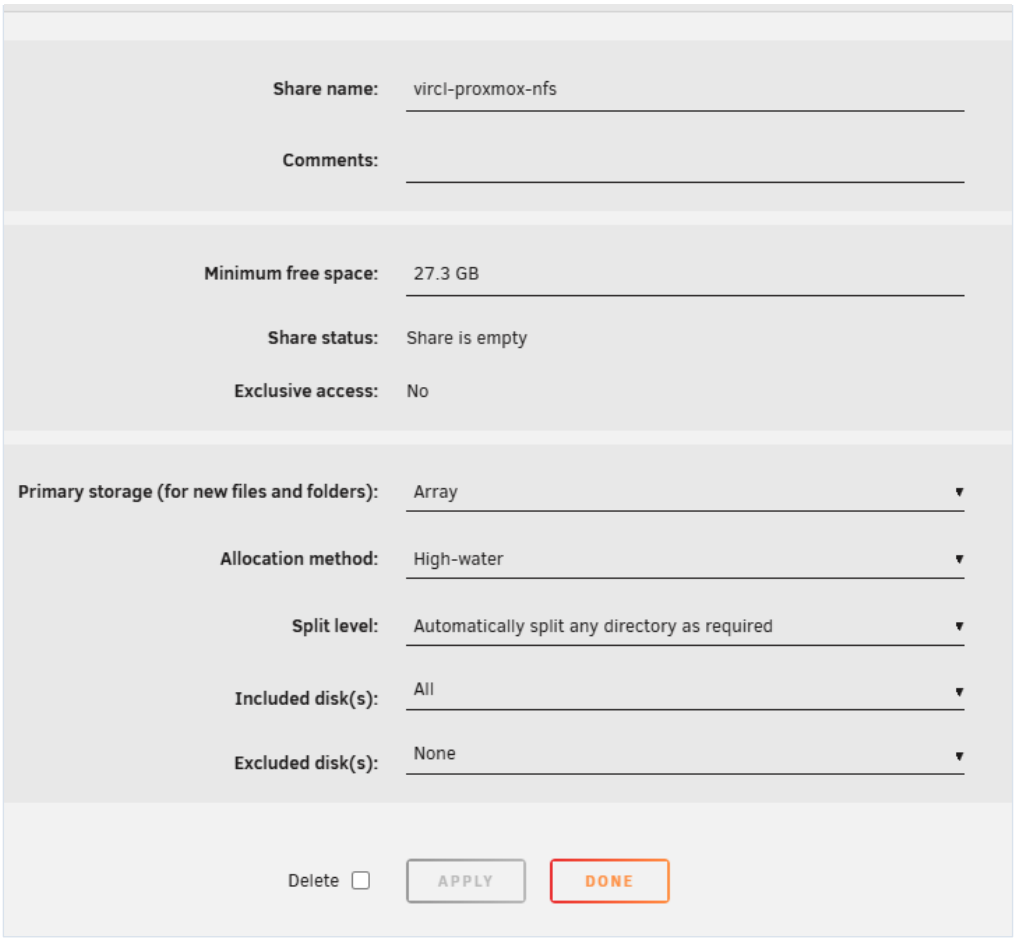


Figure 93 - Unraid NFS share settings used for Proxmox access.

13. Live storage migration

Concept

Storage migration means moving a VM's virtual disk files from one datastore or storage backend to another while keeping the VM definition and compute placement the same. The VM does not move to a different host in this chapter; only its storage location changes.

This is useful when a datastore is running out of space, when storage must be replaced, or when a VM must be prepared for another operation such as vMotion / hypervisor migration. When the platform supports live storage migration, the VM can remain powered on while the disk is copied and switched over.

VMware ESXi (+ vCenter)

The VMware side used vCenter Storage vMotion. VM `win10-hp13-01-clone` stayed powered on while its storage changed from `hp13-vmstore` to the remote `CLOIF2-NFS` datastore. vCenter showed the relocation task in progress and then completed, and the VM summary confirmed the new datastore after migration.

Proxmox VE

The Proxmox side used Disk Action → Move Storage on `VM 111` (`lubuntu-from-template-01`). The VM remained running on node hp14 while the active disk scsi0 moved from local storage `vmdata` to remote storage `cloif2-proxmox-nfs`. The Proxmox task log showed the live mirror operation and ended with TASK OK.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Tool used	vCenter Migrate wizard / Storage vMotion	VM Hardware tab / Disk Action / Move Storage
Test VM	win10-hp13-01-clone	111 - lubuntu-from-template-01
Source storage	hp13-vmstore	vmdata
Destination storage	CLOIF2-NFS	cloif2-proxmox-nfs
VM state	Powered on	Running
Result	Relocate task completed	Move disk task ended with TASK OK
Main verification point	vCenter Recent Tasks and VM Summary datastore	Proxmox task log, Hardware tab, and VM Summary

Impediments / lessons learned

- Storage migration moves only the VM's disk to a different storage location — the VM stays on the same host. This is distinct from hypervisor migration, where the compute workload moves to a different host.
- Shared or remote storage is important preparation for Chapter 14, because live host migration is easier when the destination host can access the same datastore.
- On Proxmox, leaving Delete source unchecked keeps the old disk as Unused Disk 0. This is safe but consumes storage until it is removed.
- The Proxmox iSCSI/LVM storage was available, but its capacity was close to the VM disk size. NFS was selected for the live move to avoid a space-related failure during the lab run.

Implementation walkthrough screenshots

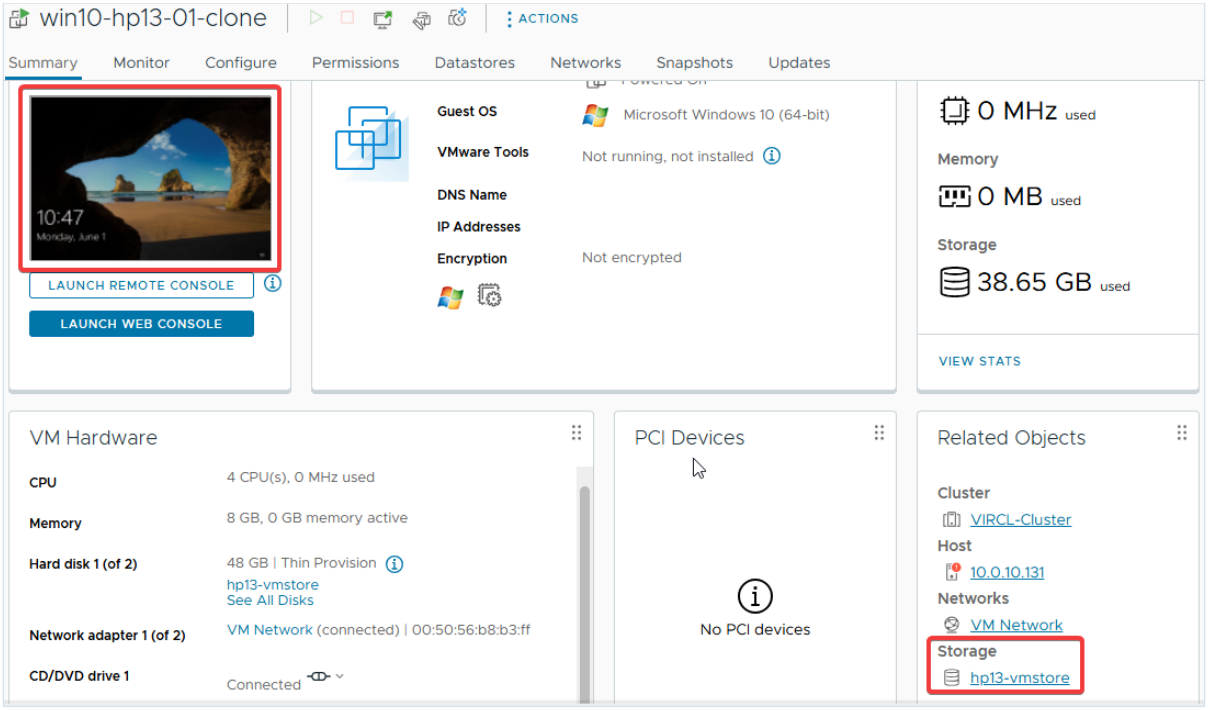


Figure 94 - ESXi VM running before live storage migration.

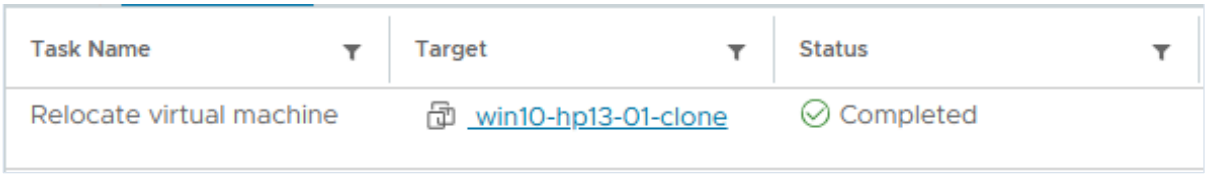


Figure 95 - ESXi storage migration task completed successfully.

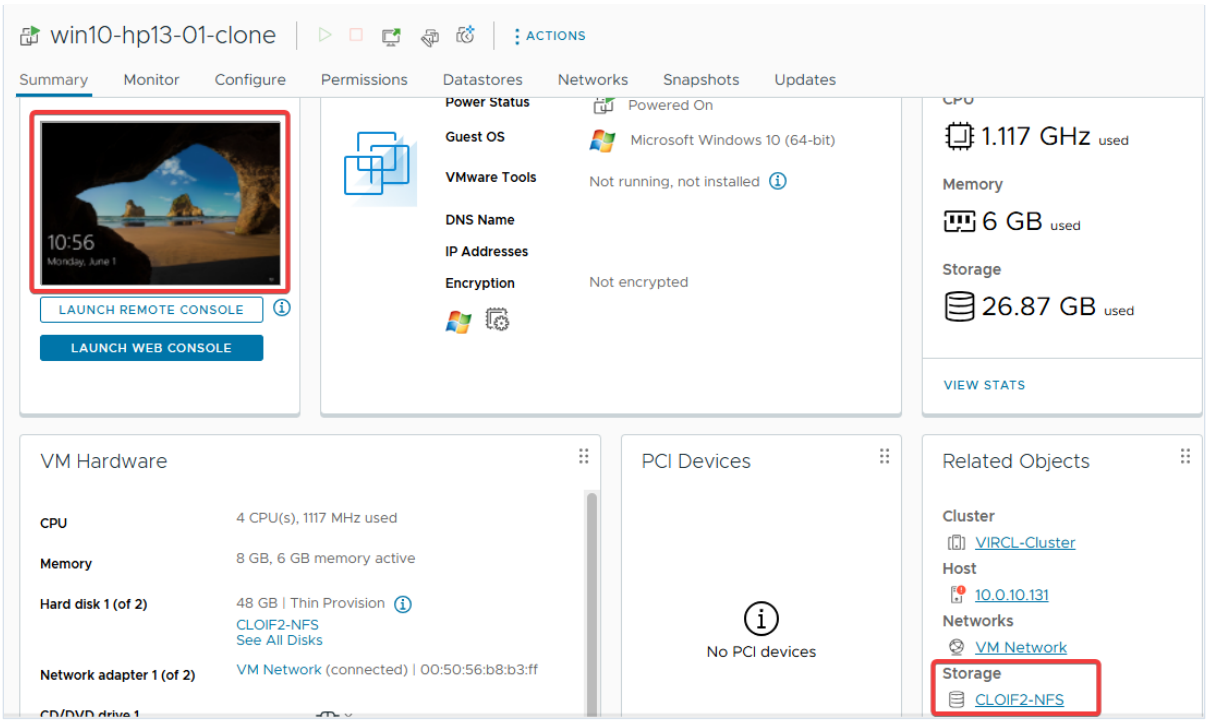


Figure 96 - ESXi VM summary confirming the new datastore after migration.

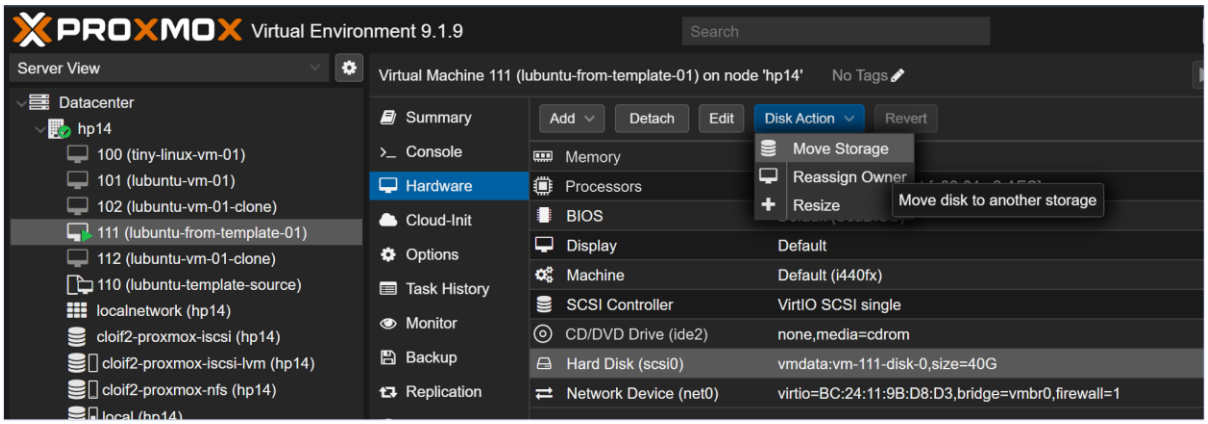


Figure 97 - Proxmox `VM 111` disk location before storage migration.

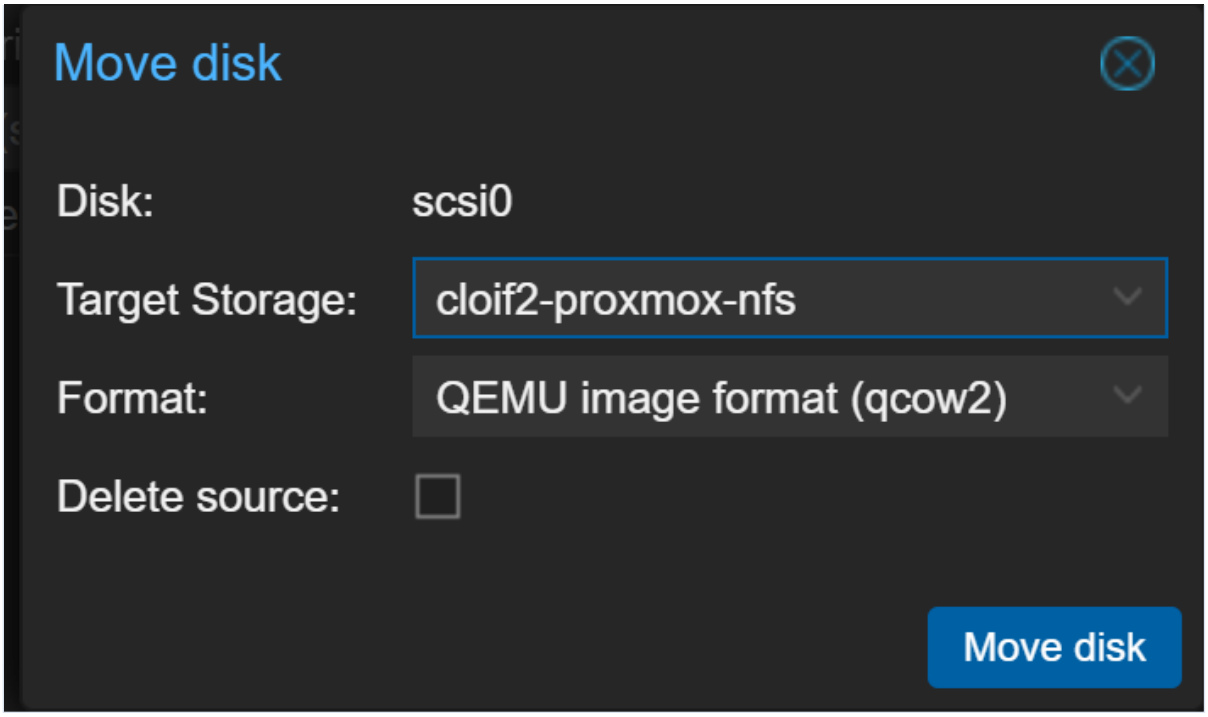


Figure 98 - Proxmox Move Disk dialog with target storage selected.

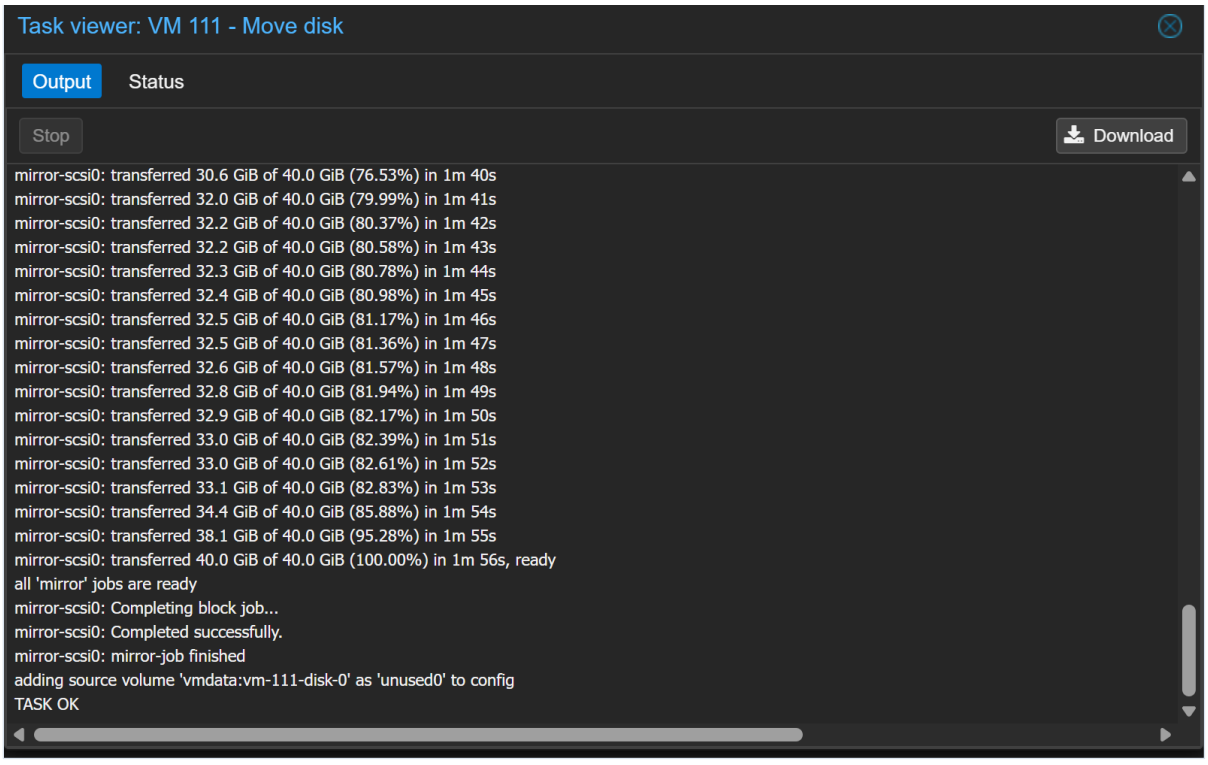


Figure 99 - Proxmox storage migration task completed successfully.

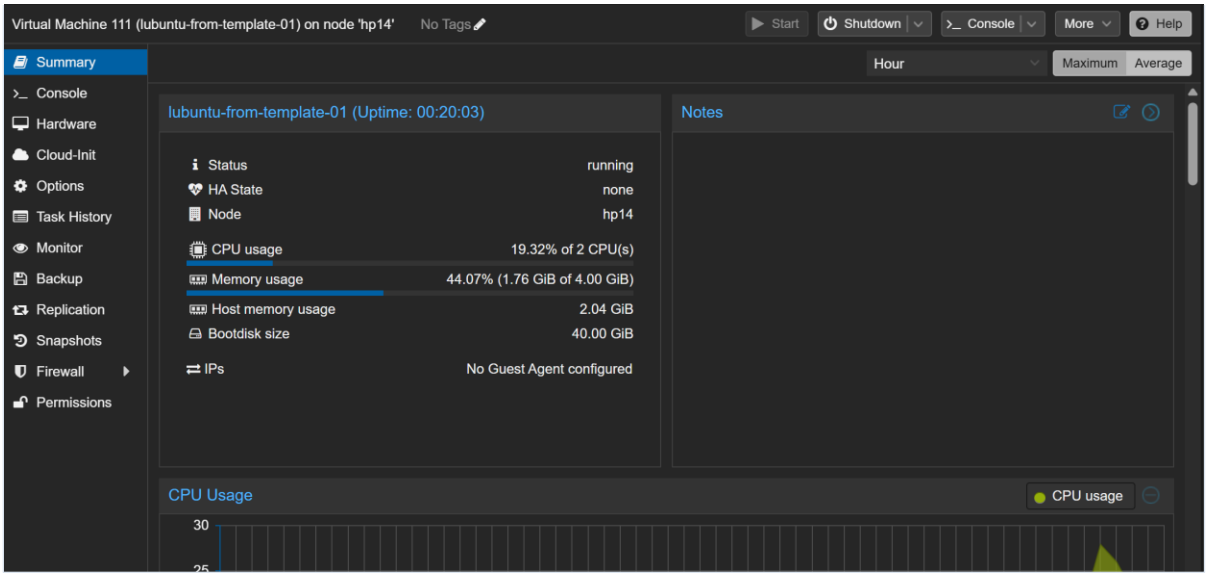


Figure 100 - Proxmox `VM 111` still running after storage migration.

14. Live hypervisor migration

Concept

Live hypervisor migration moves a running virtual machine from one physical host to another. The VM keeps running while the hypervisor transfers the active compute state to the destination host. This is useful for maintenance, load balancing, and host replacement because services can remain available while the underlying server changes.

In this project, the feature is demonstrated differently on each platform:

- VMware ESXi uses vCenter vMotion between HP13 and HP15.
- Proxmox VE requires HP14 and HP16 to be members of the same Proxmox cluster before a VM can be live-migrated between nodes.

Cross-hypervisor migration between ESXi and Proxmox is not treated as live migration. That case is documented as an export/import or backup/restore path, because the platforms use different management stacks and VM formats.

VMware ESXi (+ vCenter)

The ESXi side was completed with vCenter vMotion. The VM `win10-hp13-01-clone` was moved from HP13 to HP15 while powered on. The VM disk stayed on the shared `CLOIF2-NFS` datastore, so the migration changed only the compute host.

Proxmox VE

The Proxmox side was completed with an online migration from HP14 to HP16. HP14 and HP16 were first joined into the Proxmox cluster `vircl-team4`, because Proxmox migration targets are cluster nodes. During the setup, a storage configuration issue appeared on HP16 because HP14-only LVM storages were visible cluster-wide. Those storages were restricted to hp14, while the shared NFS storage `cloif2-proxmox-nfs` was left available for both nodes.

`VM 111` was used as the migration test VM because its active disk was already on `cloif2-proxmox-nfs`. The task completed with TASK OK, and `VM 111` appeared under HP16 while remaining powered on.

After the first migration, HP16 remained in an unreliable/unknown GUI state. The node was removed from the cluster and cleaned without reinstalling Proxmox.

The recovery included backing up VM 111's configuration, removing stale Corosync and /etc/pve/nodes state, and cold-starting HP16 after a stuck iSCSI daemon prevented normal service shutdown. HP16 was then rejoined through the Proxmox GUI. The recovered cluster showed vircl-team4 at configuration version 4 with both HP14 and HP16 healthy. VM 111 was restored from its saved configuration and started successfully on HP16.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Management layer	vCenter	Proxmox VE cluster
Source host	HP13	HP14
Destination host	HP15	HP16
Shared storage used	CLOIF2-NFS	cloif2-proxmox-nfs
Migration prerequisite	vMotion enabled on both ESXi hosts	Both Proxmox nodes joined in one cluster
Current status	Completed	Completed

Impediments / lessons learned

- ESXi vMotion required vMotion traffic to be enabled on both VMkernel adapters and required the VM to avoid an inaccessible ISO mounted from HP13-only storage.
- Proxmox migration requires both nodes to be in the same cluster. This section uses the cluster solely as a migration prerequisite; full cluster configuration and HA are covered in section 17.
- After HP16 joined the Proxmox cluster, cluster-wide storage configuration exposed HP14-only LVM storage to HP16. This caused the error `no such logical volume pve/data`. The fix was to restrict HP14-only storages to node hp14.
- The Unraid NFS export initially allowed only HP14, so HP16 received `mount.nfs: access denied by server`. The NFS rule was extended to include 10.0.10.161.
- Shared storage is the cleanest path for live migration on both platforms. In this lab, NFS from CLOIF2 is the common shared storage used for the migration result.
- Proxmox node removal is not a normal GUI workflow. HP16 had to be removed and rejoined using a mix of GUI inspection and shell cleanup. The final lesson was that cluster traffic is cleaner when separated from normal/storage traffic using a second NIC or VLAN, while the lab cluster used the assigned 10.0.10.x LAN1 addresses.

Implementation walkthrough screenshots

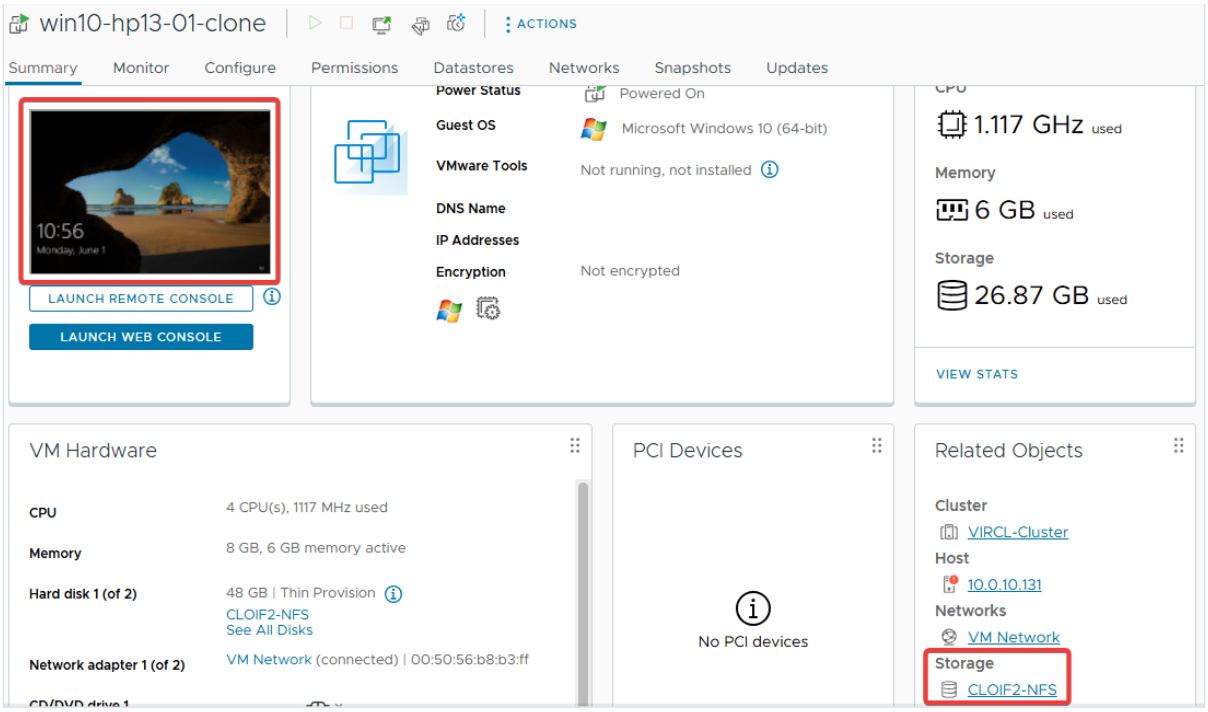


Figure 101 - ESXi VM running on HP13 before vMotion.

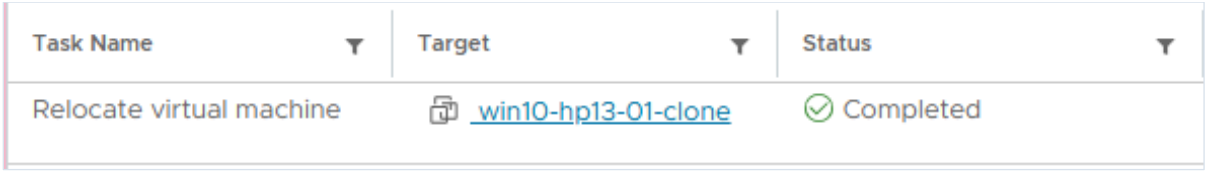


Figure 102 - vCenter vMotion task completed successfully.

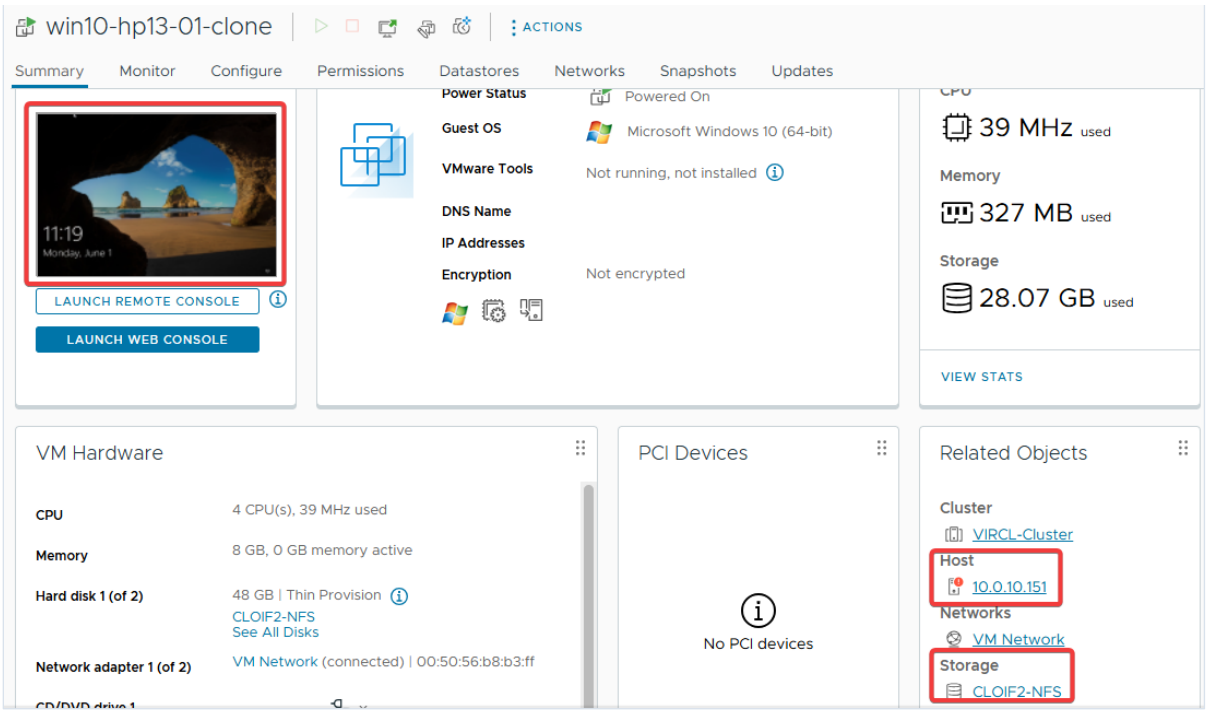


Figure 103 - ESXi VM running on HP15 after vMotion.

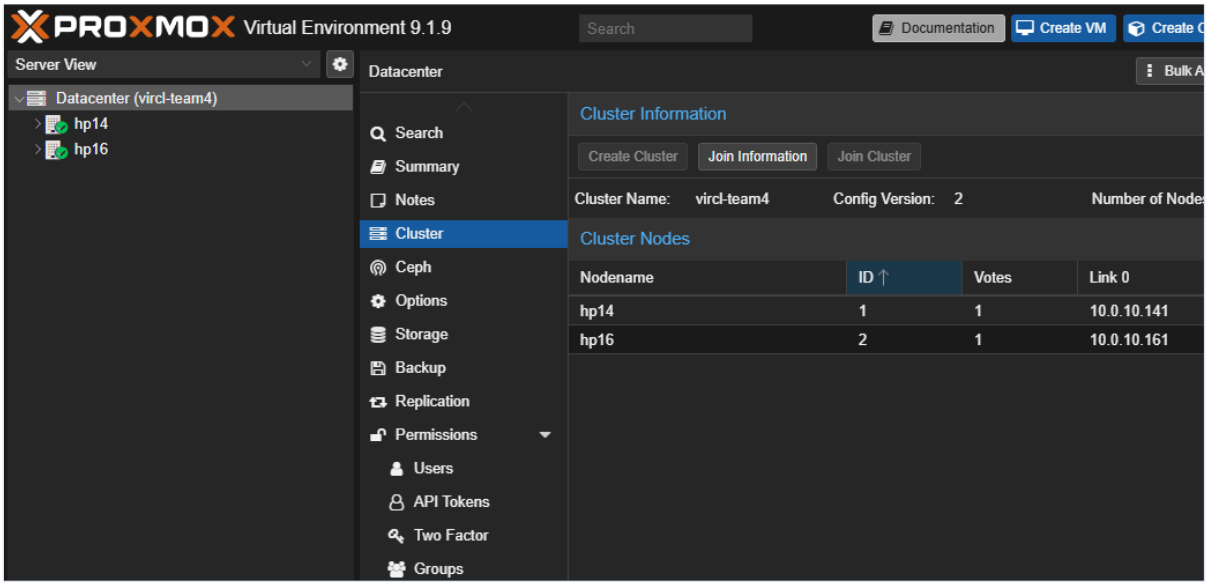


Figure 104 - Proxmox cluster view with HP14 and HP16 before the nested node was added.



Figure 105 - Proxmox online migration dialog targeting HP16.

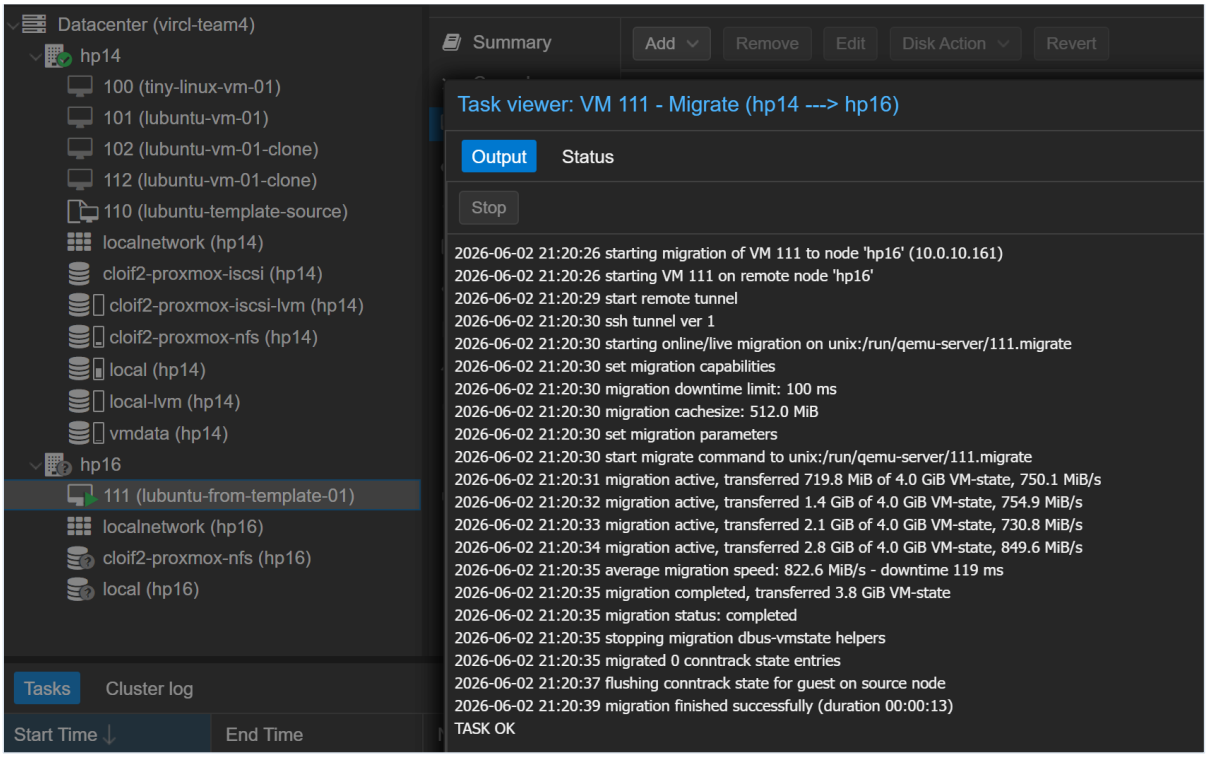


Figure 106 - Proxmox online migration task completed successfully.

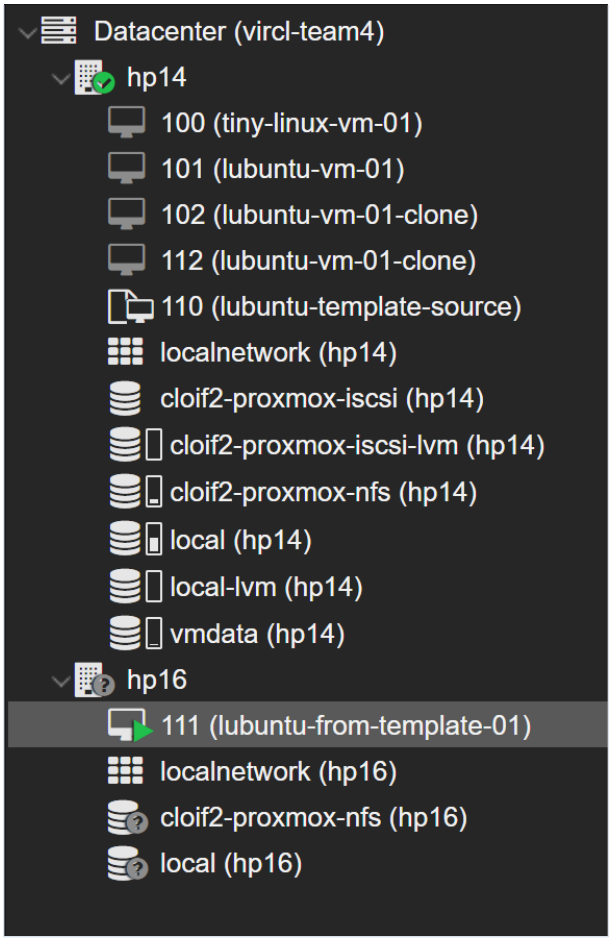


Figure 107 - Proxmox `VM 111` running on HP16 after online migration.

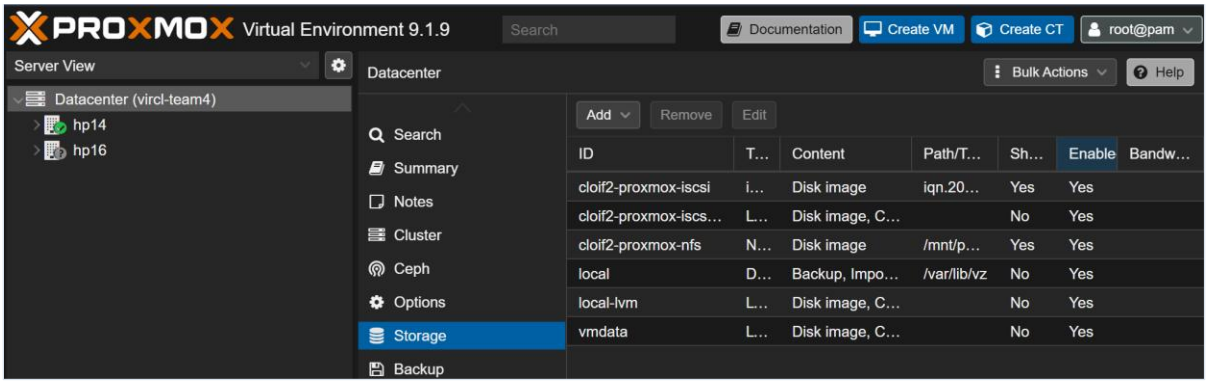


Figure 108 - Shared NFS storage enabled for Proxmox migration.

15. Users and permissions

Concept

Users and permissions control who can log in to the virtualization platform and what actions they are allowed to perform. The important idea is that access is not only attached to a user account. It is the combination of an identity, a role, and a scope.

In this project, both platforms were tested with a temporary read-only account. The account had enough rights to view inventory and status information, but not enough rights to change VM or host configuration. This proves that the management layer can separate observation from administration.

VMware ESXi (+ vCenter)

The ESXi/vCenter side was completed in vCenter at https://10.0.10.132. A temporary SSO user Viewer@vsphere.local was created in the vsphere.local domain. The built-in Read-Only role was assigned on the VIRCL-DC datacenter object with propagation enabled, so the permission applied to the child cluster, hosts, and VMs.

The account was then used for a verification login. The inventory was visible, but write actions such as VM power operations and settings changes were unavailable. After the screenshots were captured, the permission was removed and the temporary user was deleted.

Proxmox VE

The Proxmox side was completed from the HP14 Proxmox GUI in the `vircl-team4` cluster. A temporary local Proxmox user viewer@pve was created in the pve realm. The built-in PVEAuditor role was assigned at the Proxmox datacenter root path / with propagation enabled.

Because HP14 and HP16 are members of the `vircl-team4` cluster, the user and ACL entry applied to the cluster-wide Proxmox view. The verification login as viewer@pve showed that the account could see the datacenter, both nodes, storage entries, and VMs. Administrative actions were unavailable: on `VM 111`, hardware-add options and VM actions such as start, shutdown, and migrate were greyed out. After verification, the temporary user was removed.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Identity format	Viewer@vsphere.local	viewer@pve
Identity source	vCenter SSO domain vsphere.local	Proxmox local pve realm
Read-only role	Read-Only	PVEAuditor
Permission scope	Datacenter object VIRCL-DC	Datacenter root path /
Propagation	Enabled to include child objects	Enabled to include child paths
Verification method	Log in as viewer and confirm inventory visible, write actions blocked	Log in as viewer and confirm cluster resources visible, write actions blocked
Cleanup order	Permission removed before SSO user deletion	Temporary Proxmox user deleted after testing; no viewer@pve permission was kept

Impediments / lessons learned

- vCenter permissions depend on three parts: user, role, and inventory object. Missing propagation can make the datacenter visible while hiding the objects below it.
- Proxmox permissions use paths such as / and user IDs with realms such as viewer@pve. The realm is part of the username and must be selected correctly at login.
- ESXi/vCenter uses the term permission for the user-role-object assignment. Proxmox uses ACL entries for the same concept.
- Built-in read-only roles are useful because they make the result easy to verify: the user can inspect resources but cannot modify them.
- Temporary accounts were cleaned up after the screenshots so the lab environment was left in the same state as before.

Implementation walkthrough screenshots

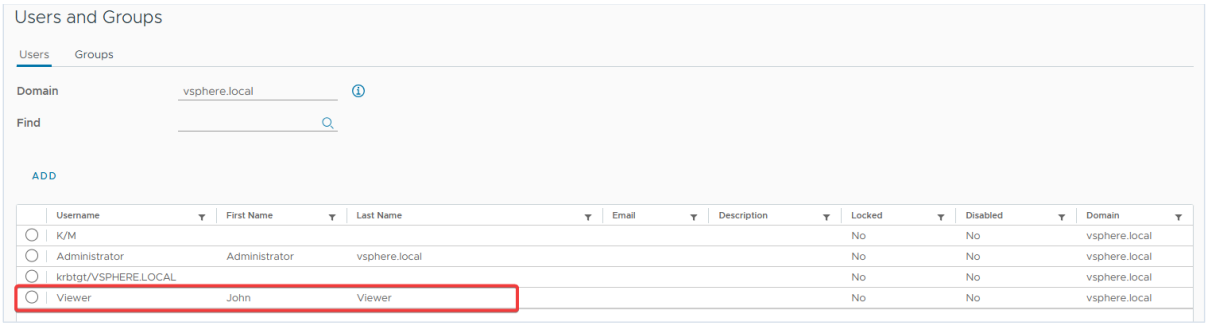


Figure 109 - vCenter local user created for permission testing.

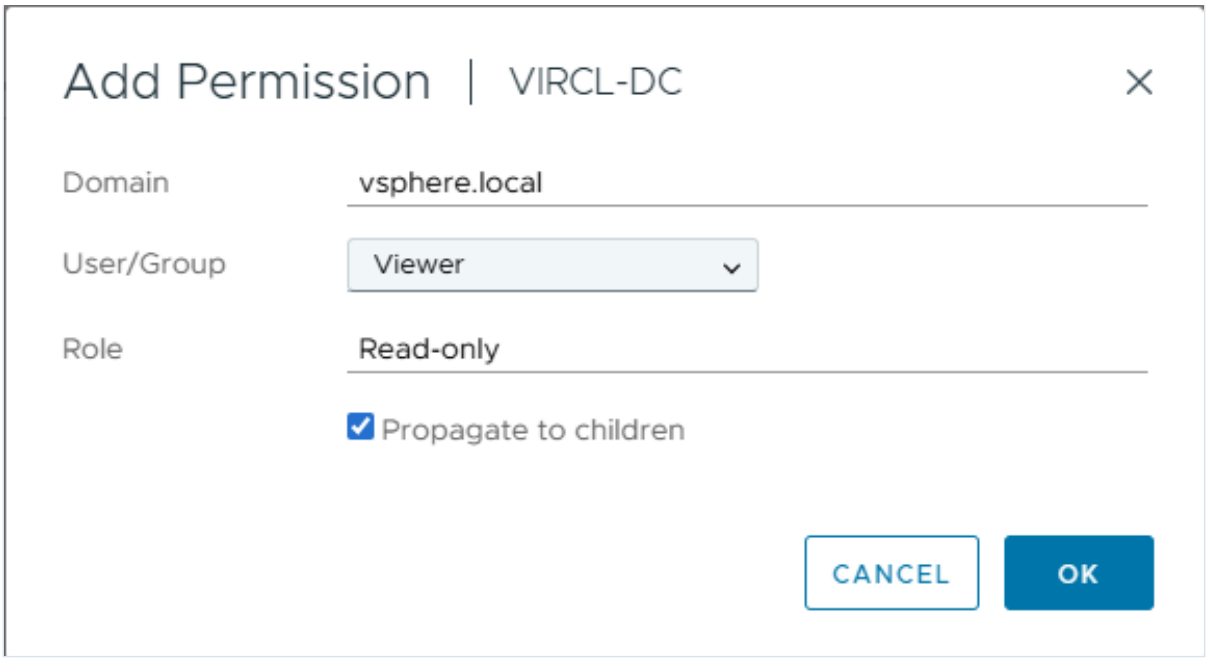


Figure 110 - vCenter permission assignment dialog for the test user.

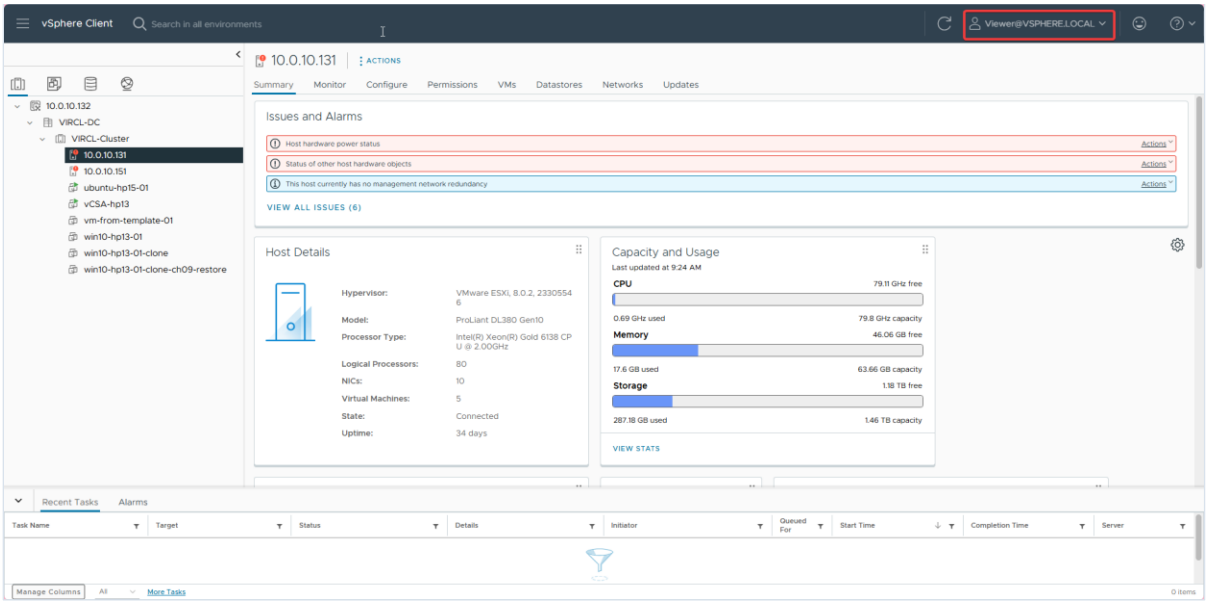


Figure 111 - vCenter login test with the restricted Viewer user.

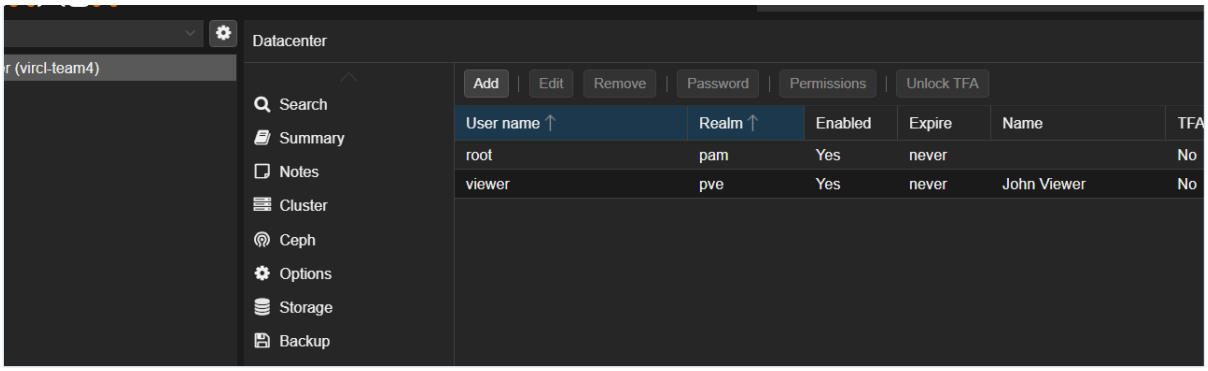


Figure 112 - Proxmox test user created for permission validation.

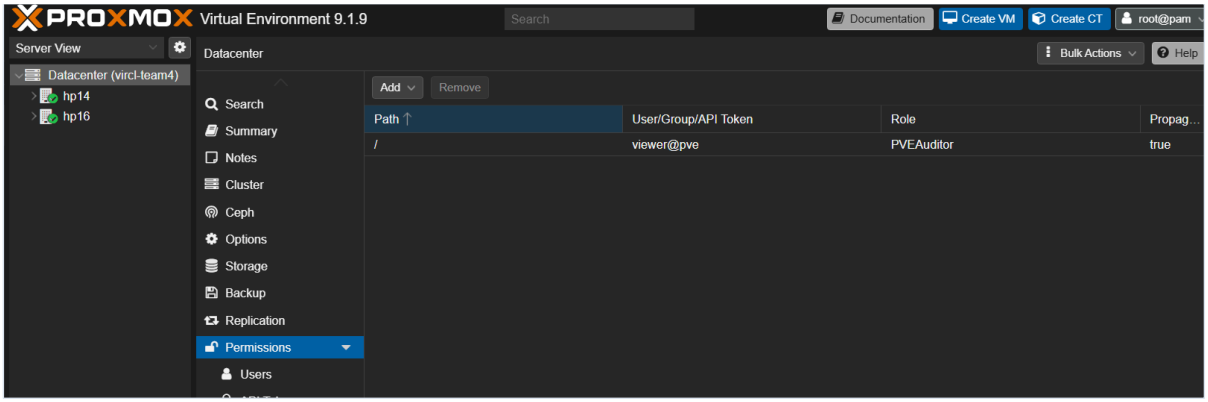


Figure 113 - Proxmox permission assignment created for the test user.

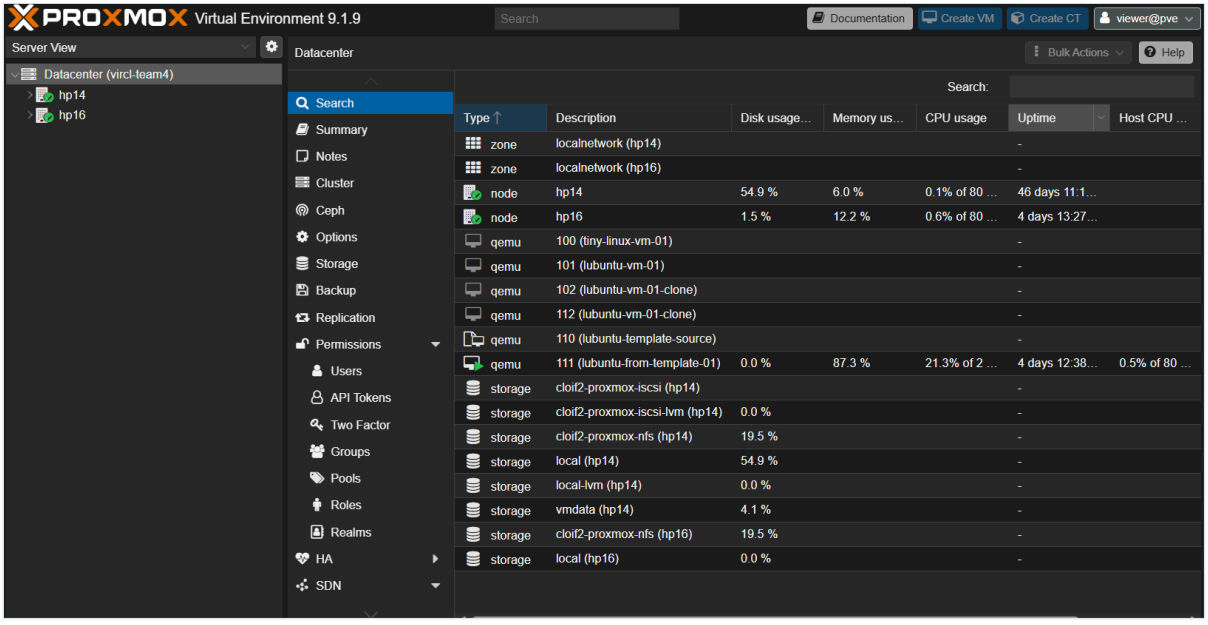


Figure 114 - Proxmox login test with the restricted viewer user.

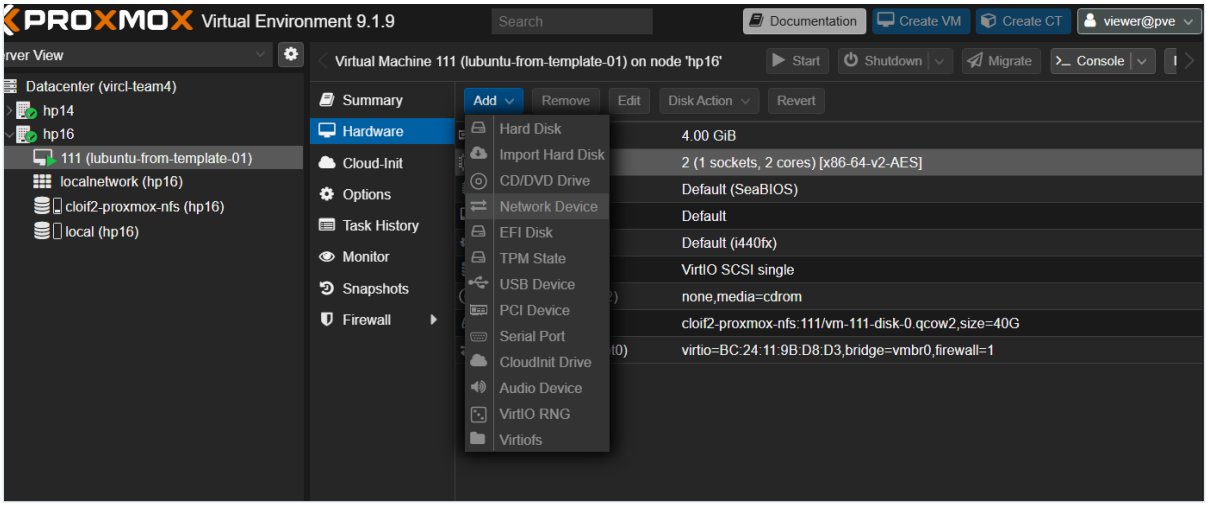


Figure 115 - Proxmox correctly blocks a write action for the restricted user.

16. Hypervisor patching

Concept

Hypervisor patching keeps the virtualization layer secure, stable, and supported. For this project, the important result is the host's current patch/build state, the configured update mechanism, and the practical limitation of applying patches in a shared lab where maintenance mode or reboot can interrupt running VMs and other project work.

VMware ESXi (+ vCenter)

The ESXi side has patch-state verification for HP13 and HP15. The following SSH commands captured the installed ESXi version, build number, and image profile:

```
esxcli system version get
```

```
esxcli software profile get
```

vSphere Lifecycle Manager was also opened and a compliance check was captured from the host Updates workflow. Patch state was verified and documented; remediation was deferred to avoid disrupting the shared lab environment and running VMs. The captured version and compliance data confirm the hosts are at a known patch level and the update mechanism is functional.

Proxmox VE

The Proxmox side was completed on the clustered HP14/HP16 environment. Repository setup had already been done during Chapter 1: enterprise repositories were disabled and the Proxmox/Ceph no-subscription repositories were enabled. This section then verified the current update state in the Proxmox GUI and used the node Updates -> Upgrade workflow.

HP16 showed pending updates in the GUI before patching, including Debian-origin packages and Proxmox packages from the no-subscription repository. The upgrade task was started through the GUI and completed with "Your System is up-to-date." HP14 was upgraded through the same workflow and also completed with "Your System is up-to-date." Both nodes reported that a kernel update had been installed and that a reboot was needed to activate the new kernel. After the upgrades, both Proxmox nodes were rebooted in the lab.

Because HP14 and HP16 are members of the same Proxmox cluster `vircl-team4`, the operational lesson is that Proxmox patching is node-local but cluster-aware: before rebooting a node, running VMs and cluster health were checked. In production, the same approach keeps node reboots sequential.

Comparison

Aspect	ESXi/vCenter	Proxmox VE
Update mechanism	vSphere Lifecycle Manager and ESXi image profiles	Debian/APT repositories plus Proxmox packages
Version verification	esxcli system version get, esxcli software profile get	pveversion -v, APT repository and package status
Applying updates	Usually requires maintenance mode and reboot	Usually runs through apt, with reboot required for kernel updates
Lab status	Patch/build state captured; no remediation applied	HP14 and HP16 upgraded through the Proxmox GUI; kernel reboot required and nodes rebooted

Impediments / lessons learned

- Patching a hypervisor can disrupt VMs and cluster work. In the lab, patch-state verification is safer than unplanned remediation.
- ESXi remediation through Lifecycle Manager normally requires maintenance mode and may require moving or powering off VMs.
- Proxmox package updates are transparent through APT and the GUI Updates page, but kernel or core Proxmox package updates still require careful reboot planning.
- Repository switching and patching are separate tasks. The Proxmox no-subscription repositories were configured during deployment; this section documents the later update check, upgrade, and reboot requirement.

Implementation walkthrough screenshots

```
[root@hp13:~] esxcli system version get
   Product: VMware ESXi
   Version: 8.0.2
   Build: Releasebuild-23305546
   Update: 2
   Patch: 30
[root@hp13:~] esxcli software profile get
(Updated) ESXi-8.0U2b-23305546-standard
   Name: (Updated) ESXi-8.0U2b-23305546-standard
   Vendor: VMware, Inc.
   Creation Time: 2026-05-06T08:45:36
   Modification Time: 2026-06-01T07:34:01
   Stateless Ready: True
   Description:

      2026-05-06T08:45:35.451462+00:00: The following VIBs are
      installed:
        vmware-fdm    8.0.2-23504390
```

Figure 116 - HP13 ESXi version output captured for patching verification.

```
[root@hp15:~] esxcli system version get
   Product: VMware ESXi
   Version: 8.0.2
   Build: Releasebuild-23305546
   Update: 2
   Patch: 30
[root@hp15:~] esxcli software profile get
(Updated) ESXi-8.0U2b-23305546-standard
   Name: (Updated) ESXi-8.0U2b-23305546-standard
   Vendor: VMware, Inc.
   Creation Time: 2026-05-11T08:09:32
   Modification Time: 2026-06-01T08:36:40
   Stateless Ready: True
   Description:

      2026-05-11T08:09:32.567571+00:00: The following VIBs are
      installed:
        vmware-fdm    8.0.2-23504390
```

Figure 117 - HP15 ESXi version output captured for patching verification.

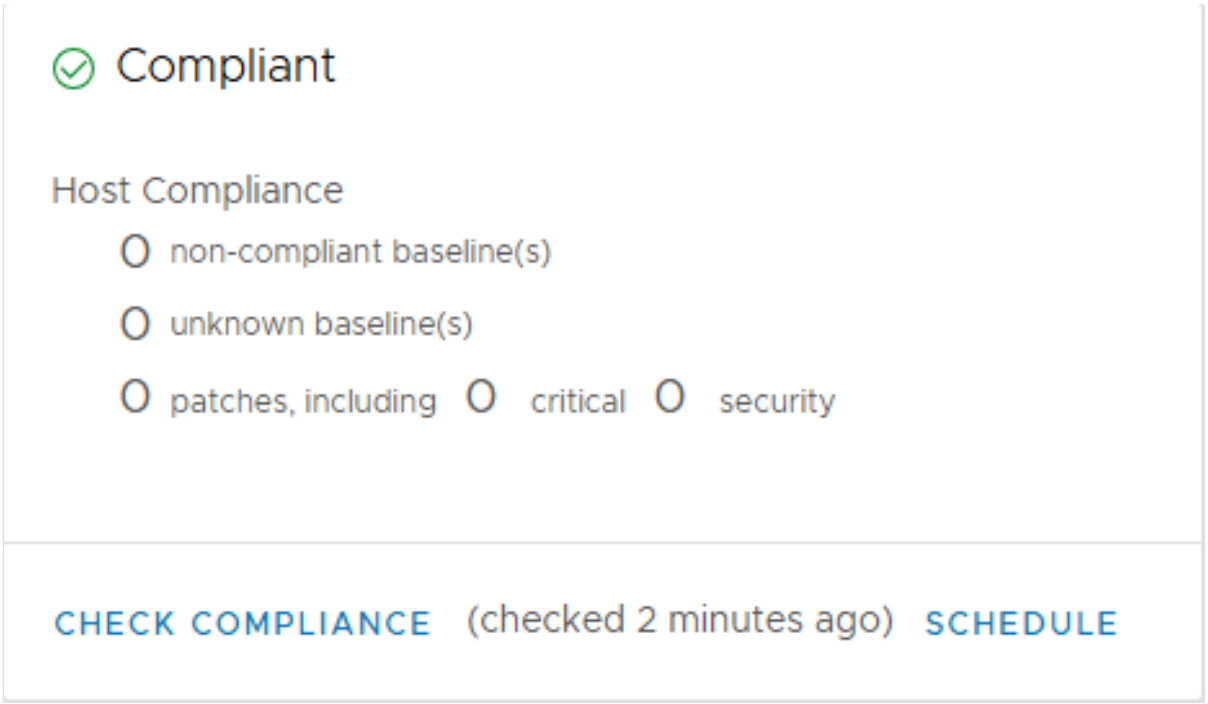


Figure 118 - vSphere Lifecycle Manager compliance result for ESXi patching.

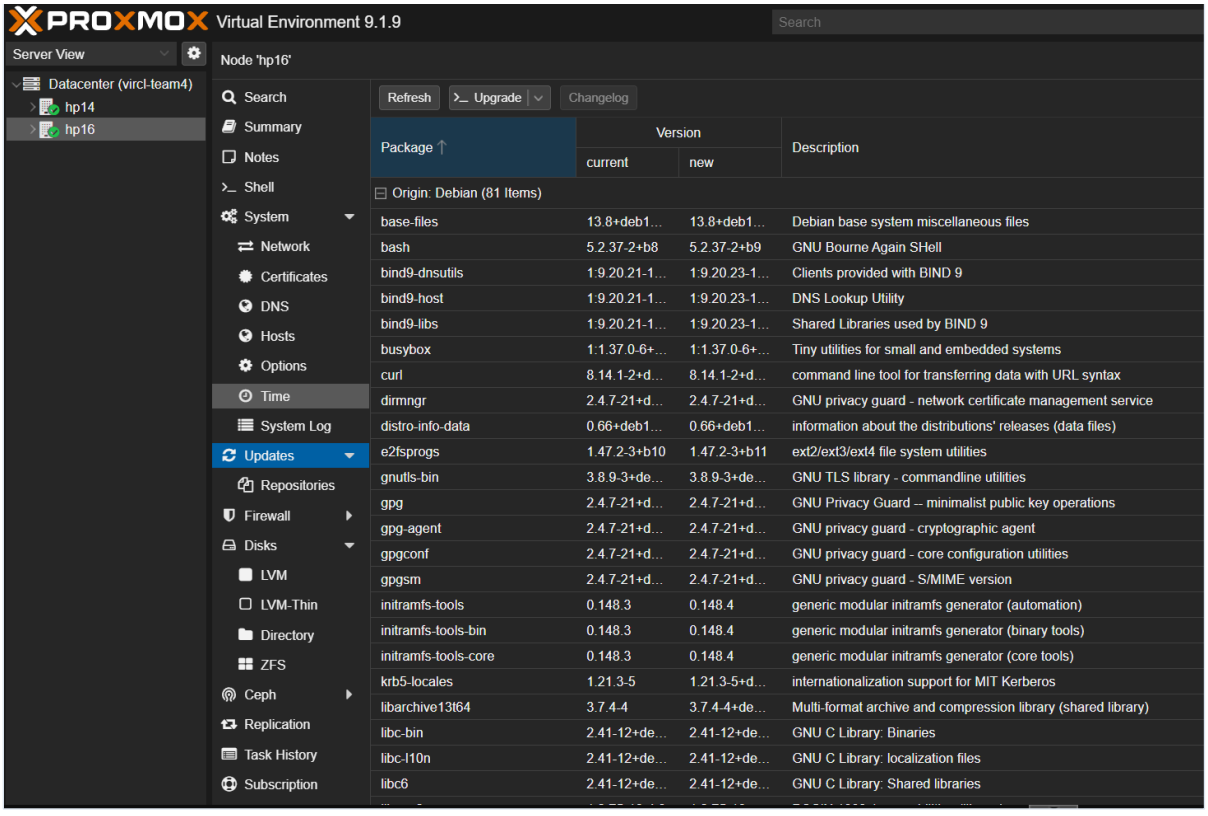


Figure 119 - Available Proxmox updates shown before patching HP16.

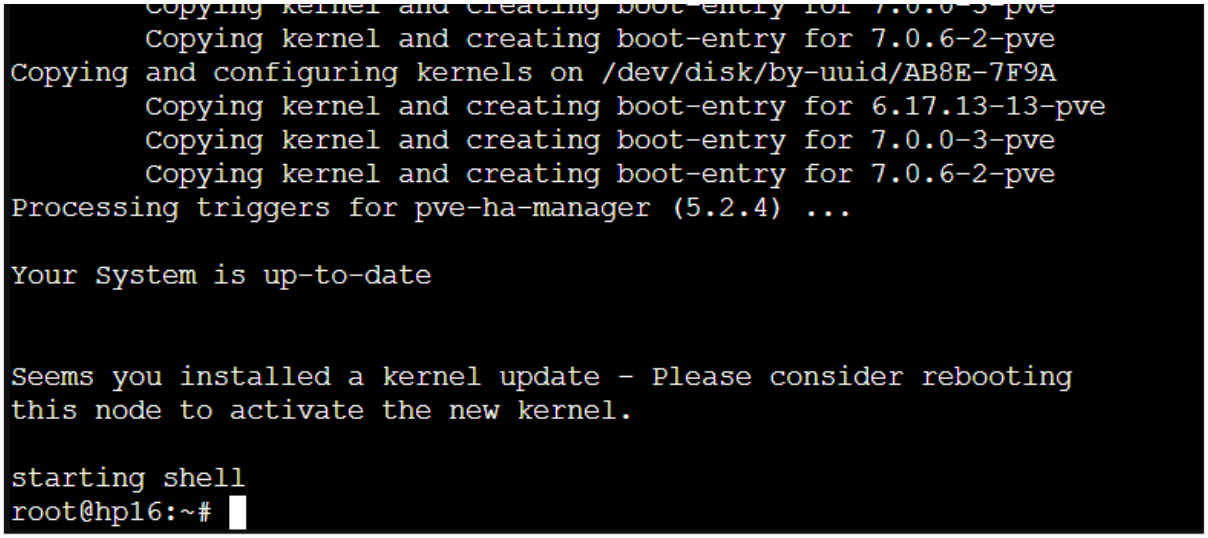


Figure 120 - HP16 Proxmox upgrade completed and requested reboot.

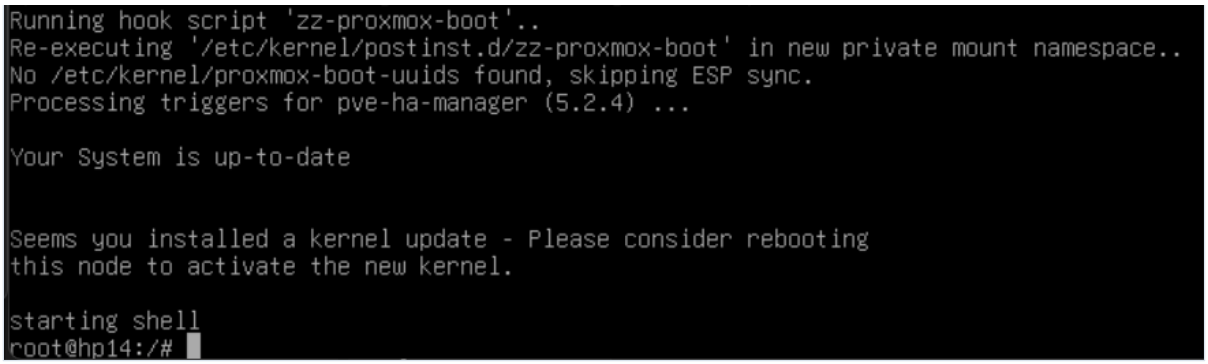


Figure 121 - HP14 Proxmox upgrade completed and requested reboot.

17. Specialisation: Clustering

Team 4 selected Clustering as the project specialisation. The practical goal is to show how VMware ESXi/vCenter and Proxmox VE can group multiple hypervisor hosts together, provide central management, support live migration, and prepare for high availability.

This chapter covers the design decisions, implementation, screenshots, limitations, and presentation flow for the clustering specialisation.

The final architecture uses nested virtualisation to reach a 3-node design on both platforms. The ESXi/vCenter cluster uses HP13, HP15, and a nested ESXi VM running on HP14 Proxmox. The Proxmox VE cluster uses HP14, HP16, and a nested Proxmox VM running on HP13 ESXi. The chapter follows this design in its explanation: quorum first, then shared storage, live migration, HA placement, and the limitations of nested nodes.

Requirement coverage

The project brief says that the specialisation must prove three things:

Requirement	Where it is covered	Verification / output
Understand and explain the concept	This chapter, especially the theory, quorum, migration, HA, shared storage, and do's/don'ts sections	Written explanation and presentation slides
Implement it in both hypervisors	ESXi/vCenter and Proxmox VE implementation sections in this chapter	vCenter/ESXi cluster screenshots, Proxmox cluster screenshots, nested nodes, shared storage, and migration results
Name do's and don'ts	Do's and Don'ts section below	Practical design rules and limitations

Architecture decision

The project has four physical HP servers and two required hypervisors. A simple split gives only two physical nodes per platform:

- ESXi: HP13 and HP15.
- Proxmox: HP14 and HP16.

That is enough to demonstrate basic live migration, but it is weak for a clustering specialisation because many cluster decisions depend on quorum. A two-node cluster can easily become ambiguous if one node is isolated. To make the design stronger, each platform receives a third node by nesting it inside the opposite hypervisor:

Cluster	Physical nodes	Nested third node	Nested host
ESXi/vCenter	HP13, HP15	esxi-nested	HP14 Proxmox
Proxmox VE	HP14, HP16	pve-nested	HP13 ESXi

This creates a symmetrical cross-platform architecture: Proxmox hosts an ESXi VM, and ESXi hosts a Proxmox VM.

Implementation summary

Platform	Cluster implementation	Practical proof
VMware ESXi/vCenter	HP13 and HP15 are physical ESXi hosts managed by vCenter. esxi-nested runs as an ESXi VM on HP14 Proxmox and is visible in the vCenter cluster as 10.0.10.143.	vCenter three-host cluster view, vSphere HA enabled, DRS fully automated, shared datastore visibility, and vMotion result between HP13 and HP15.
Proxmox VE	HP14 and HP16 are physical Proxmox nodes in cluster vircl-team4. pve-nested runs as a Proxmox VM on HP13 ESXi and acts mainly as a third vote/witness-style node.	pvecm status, three-node Datacenter view, shared CLOIF2 storage, online migration from HP14 to HP16, HA group physical-hosts-only.

Theoretical concept

A hypervisor cluster is a group of hosts managed as one environment. Instead of treating every server as an isolated island, the administrator can place hosts into a shared management scope, attach common storage, and move or restart workloads between hosts.

Important clustering concepts:

- Central management: one interface manages several hosts, VMs, permissions, and tasks.
- Live migration: a running VM moves from one host to another with minimal interruption. In VMware this is vMotion; in Proxmox this is online migration.

- High Availability (HA): if a host fails, the cluster can restart protected VMs on another host. This is not instant live migration because the failed host cannot transfer the VM memory state.
- DRS/load balancing: workloads can be placed or moved based on resource pressure. VMware exposes this through DRS; Proxmox can use HA groups and manual placement, with more advanced scheduling depending on configuration.
- Quorum: the cluster needs enough votes to make decisions safely. Quorum prevents split-brain, where separated nodes both believe they are allowed to control the same resources.
- Shared storage: VM disks must be visible to the target host for reliable migration and HA. Without shared storage, the system either has to copy disks during migration or cannot restart the VM on another host.
- Nested virtualisation: one hypervisor runs as a VM inside another hypervisor. It is useful for lab topology and cluster membership, but performance and failure behaviour are not equal to physical hosts.

The most important practical difference for this specialisation is planned migration versus crash failover:

Situation	Cluster feature	Practical result
A host is healthy but needs maintenance	Live migration	The running VM can move to another host with minimal interruption.
A host suddenly crashes	HA failover	The VM is restarted on another host. There is downtime because the guest OS must boot again.
One management UI is unavailable	Multi-node management	The cluster can be opened from another online node if quorum remains.
A node is isolated from the others	Quorum	Only the side with enough votes can safely continue changing cluster state.

Implementation in VMware ESXi (+ vCenter)

The ESXi side uses vCenter as the management plane. HP13 and HP15 are physical ESXi hosts, while esxi-nested is installed as a VM on HP14 Proxmox and appears in the cluster as 10.0.10.143. This gives vCenter three ESXi hosts to manage.

Result: The nested ESXi VM `esxi-nested` (10.0.10.143) running on HP14 Proxmox was added to the `VIRCL-Cluster` in vCenter. The cluster then contained three ESXi hosts: HP13, HP15, and `esxi-nested`. vSphere HA was enabled with host monitoring, DRS was set to Fully Automated, and shared datastores `CLOIF2-NFS` and `CLOIF2-iSCSI` were visible to all hosts. The host icons displayed warnings due to the nested VM's limited resources, which is expected for a lab environment. vMotion between HP13 and HP15 was successfully demonstrated in Chapter 14. The nested node provided the third cluster member required for quorum demonstration.

Implementation in Proxmox VE

The Proxmox side uses cluster `vircl-team4`. HP14 and HP16 are physical Proxmox hosts, while pve-nested is installed as a VM on HP13 ESXi. Corosync handles cluster membership and quorum.

Outcome: Section 14 proved online migration of `VM 111` from HP14 to HP16 after its disk and ISO were both moved to shared CLOIF2 NFS storage. The specialisation added the third Proxmox node so the final cluster had three votes. The nested node was treated as a witness/lightweight node because the lab evidence showed it was useful for quorum but not suitable as a production workload target.

The extended showcase adds an HA placement design:

- HA group `physical-hosts-only` contains HP14 and HP16.
- pve-nested is excluded from real workload placement.
- A small disposable Linux VM was used as the HA resource candidate on shared storage.
- Planned migration was demonstrated through the completed HP14-to-HP16 migration result.
- HA failover was validated by intentionally powering off the pve-nested node to confirm that the cluster detected the outage and restarted the protected VM on one of the remaining available hosts.

Do's and Don'ts

- Three nodes were used when explaining quorum.
- Nested nodes were kept lightweight because their performance and storage access were limited.
- Network reachability was verified before joining nodes to the cluster.
- Shared storage was verified before claiming migration or HA support.
- The nested virtualisation settings were captured, not only the final cluster view.
- Planned migration was separated from unplanned HA restart in the explanation.
- HA groups kept real workloads on physical hosts.
- Limitations were stated directly where HA or DRS could not be tested under production conditions.
- ESXi-to-Proxmox live migration was not claimed. Cross-hypervisor movement was treated as export/import or conversion, not vMotion/online migration.
- Zero-downtime crash failover was not claimed. Normal HA restarted the VM and did not preserve RAM state.
- HA failover was tested in a controlled manner outside the live presentation by intentionally shutting down pve-nested and verifying that the protected VM restarted on an available host.
- Important workloads were not placed on nested hosts.
- Local-only storage restrictions in Proxmox were handled before migration and HA claims were made.
- The two-node cluster was not treated as equivalent to a proper quorum design.
- In production, a dedicated and separated IP/network interface is recommended for cluster communication (Corosync heartbeat / vCenter management network) so that cluster traffic is isolated from VM and general management traffic. In this lab environment, no additional interfaces were available, so all traffic shared the same interface — this was a hardware constraint, not a design choice.

Implementation walkthrough screenshots

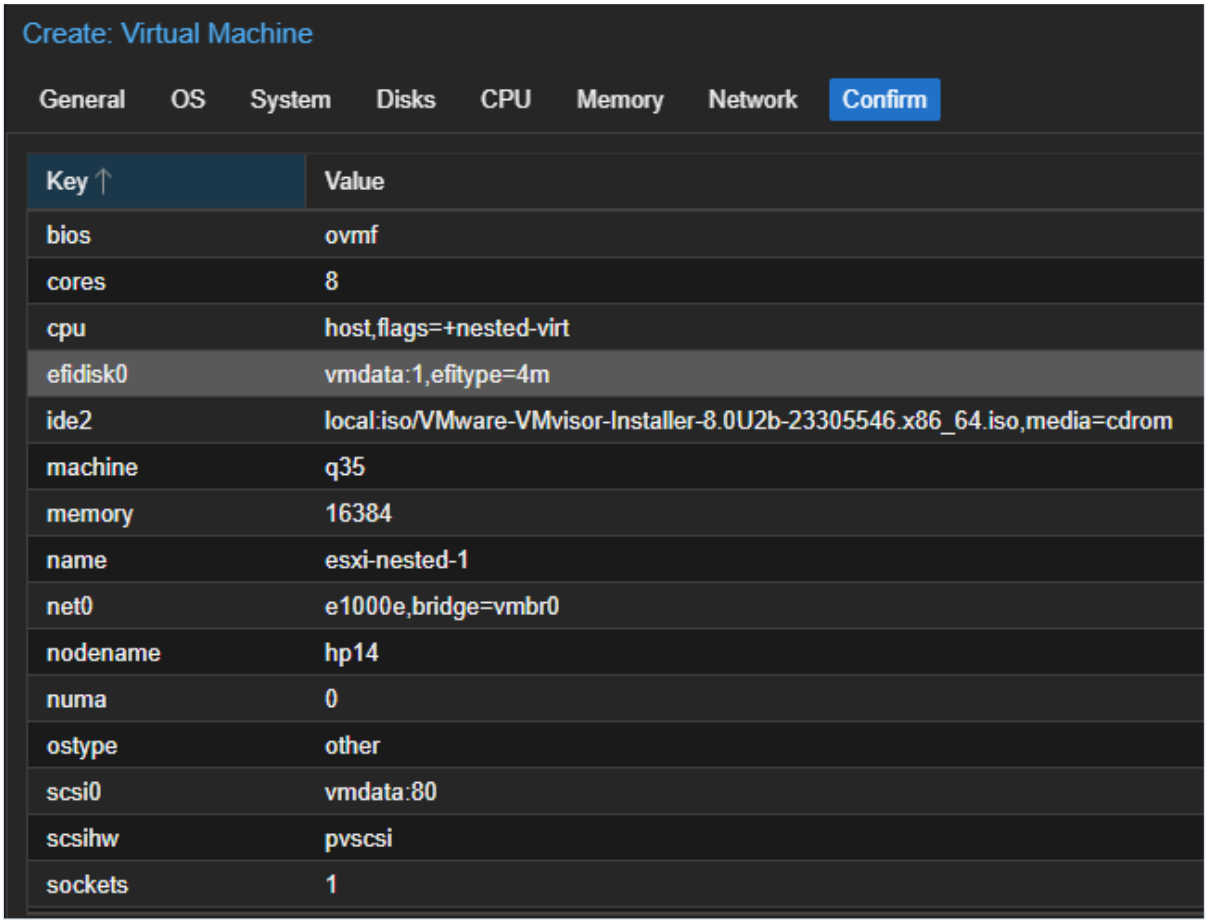


Figure 122 - Nested ESXi VM running on HP14 Proxmox for the ESXi cluster design.

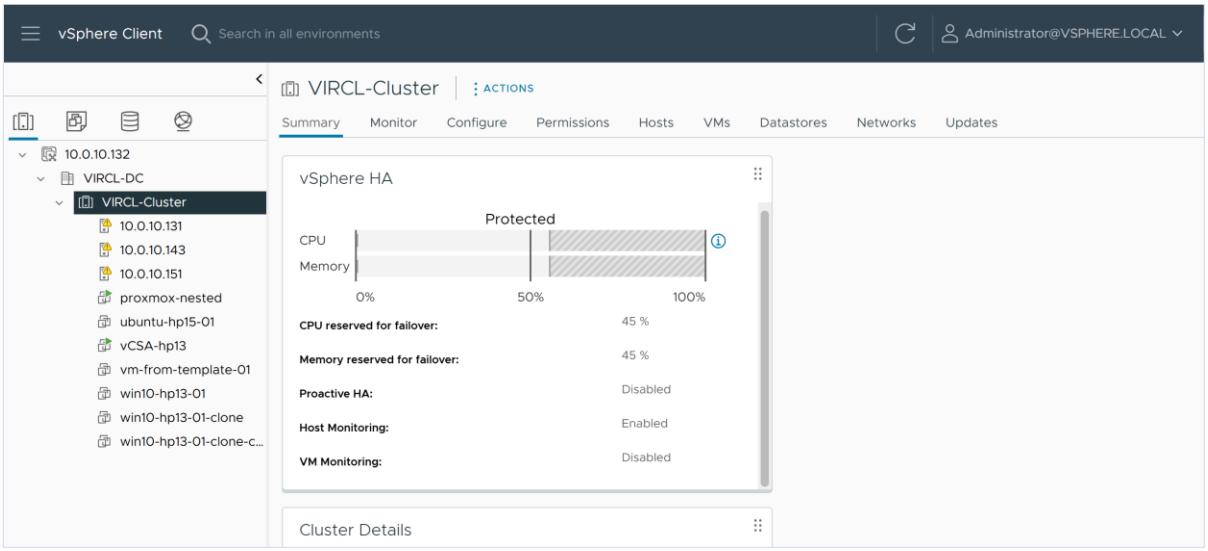


Figure 123 - vCenter cluster view showing the three ESXi hosts.

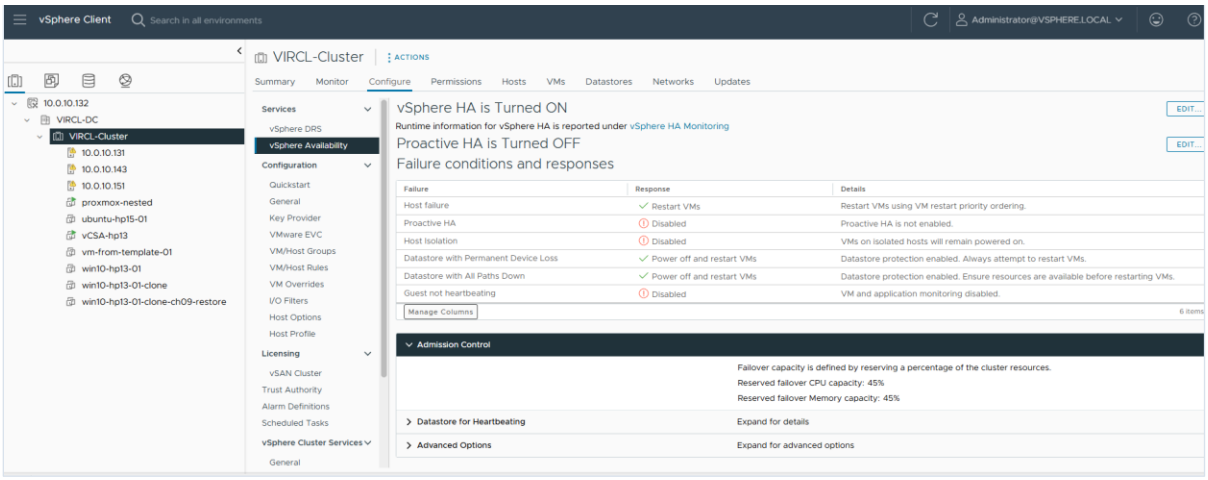


Figure 124 - vSphere HA enabled on the ESXi cluster.

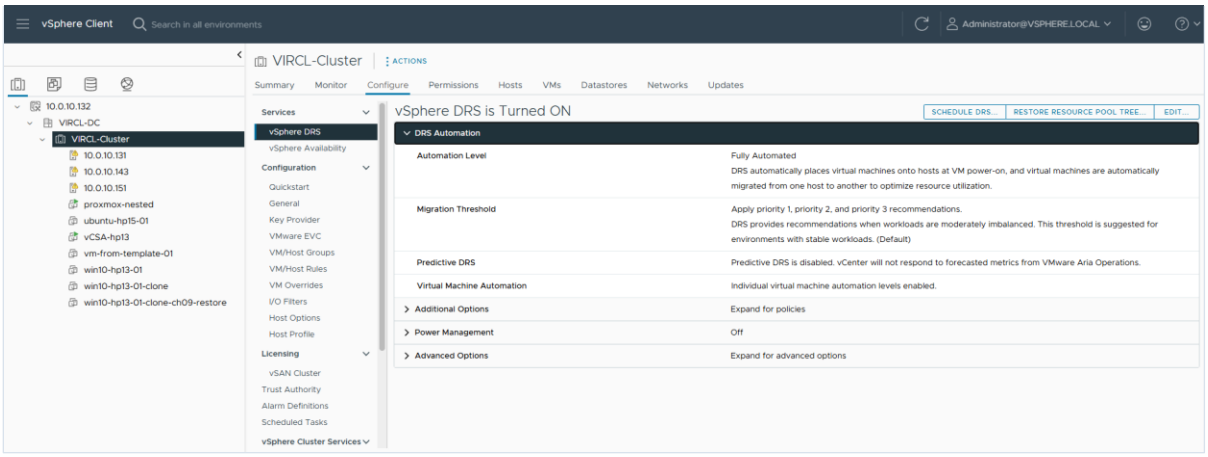


Figure 125 - vSphere DRS configured in fully automated mode.

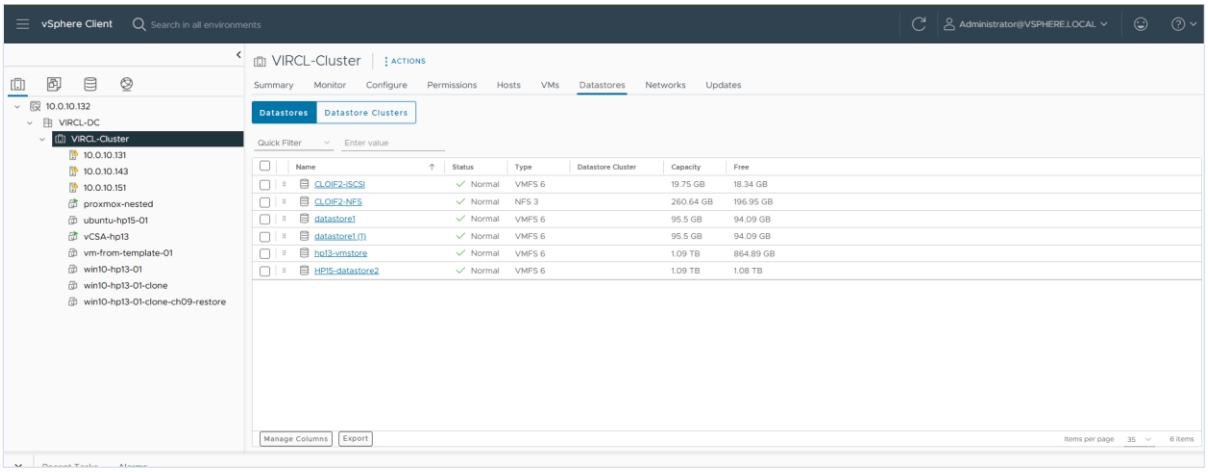


Figure 126 - Shared datastores visible in the ESXi cluster.

Summary

Please confirm the displayed information. Once you press the **Install** button, the installer will begin to partition your drive(s) and extract the required files.

Option	Value
Filesystem:	ext4
Disk(s):	/dev/sda
Country:	Luxembourg
Timezone:	Europe/Luxembourg
Keymap:	fr-ch
Email:	girotto2005@gmail.com
Management Interface:	nic0
Hostname:	pve-nested
IP CIDR:	10.0.10.133/16
Gateway:	10.0.0.1
DNS:	10.0.0.62

Figure 127 - Nested Proxmox installer summary with hostname, IP address, gateway, and DNS settings.

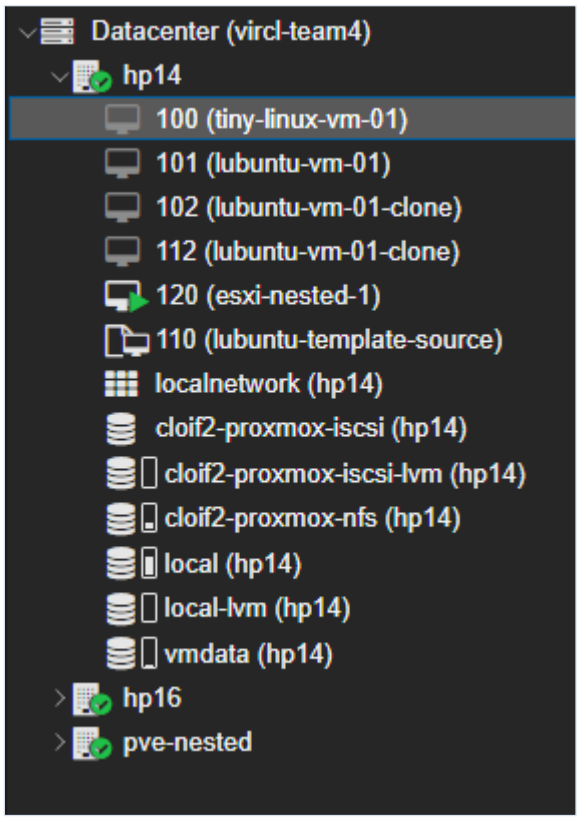


Figure 128 - Proxmox Datacenter tree showing HP14, HP16, and pve-nested in cluster `vircl-team4`.

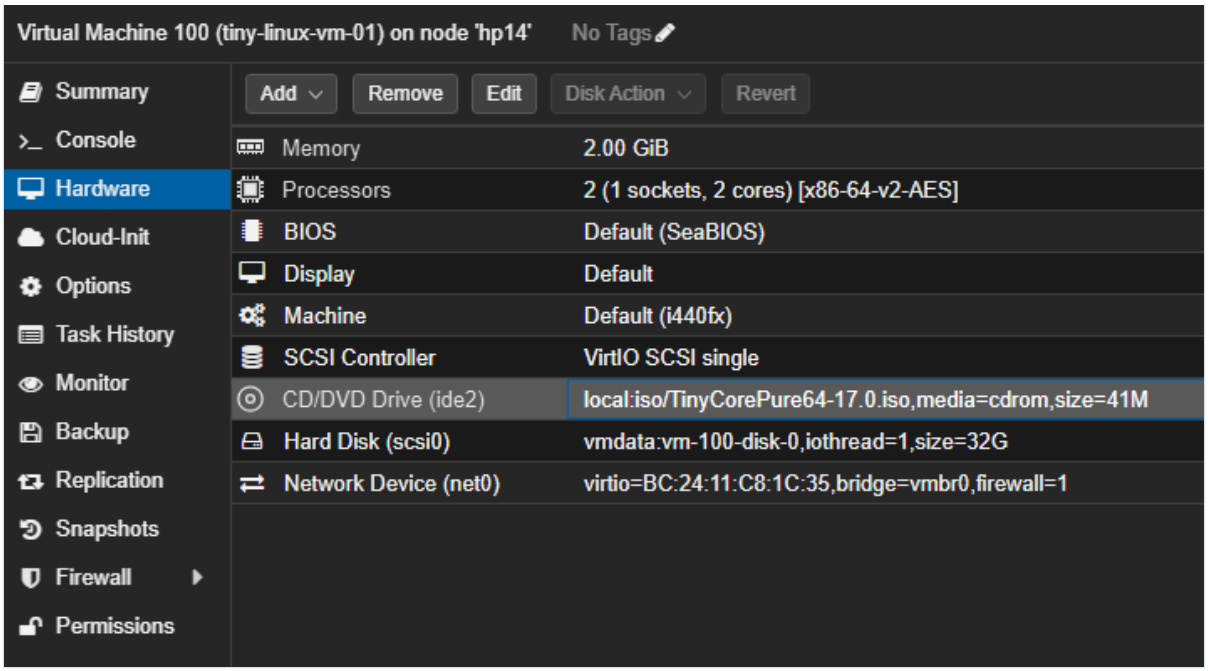


Figure 129 - Proxmox `VM 100` hardware showing local disk and ISO before migration cleanup.

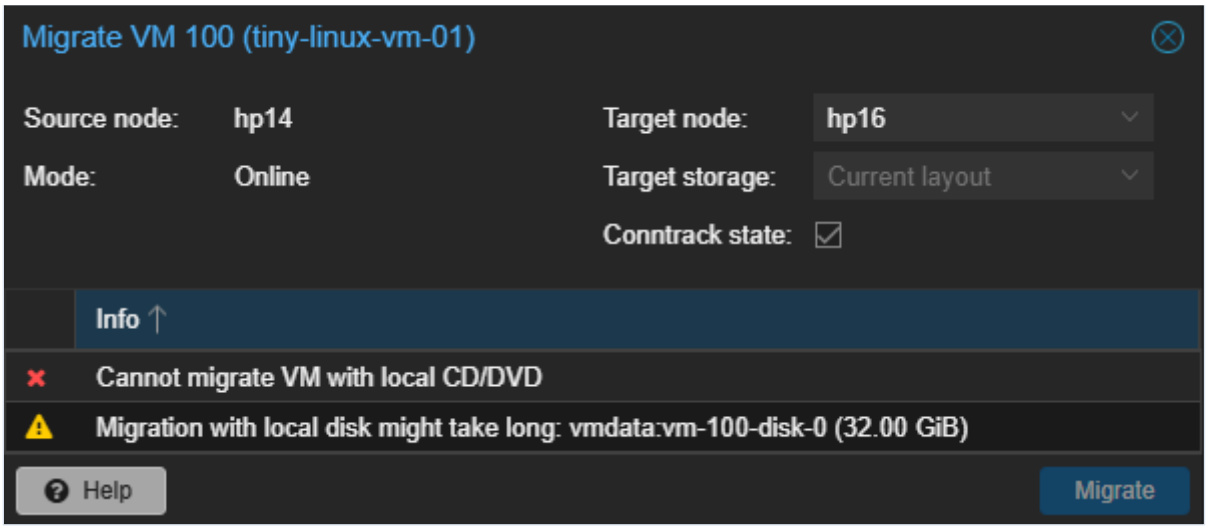
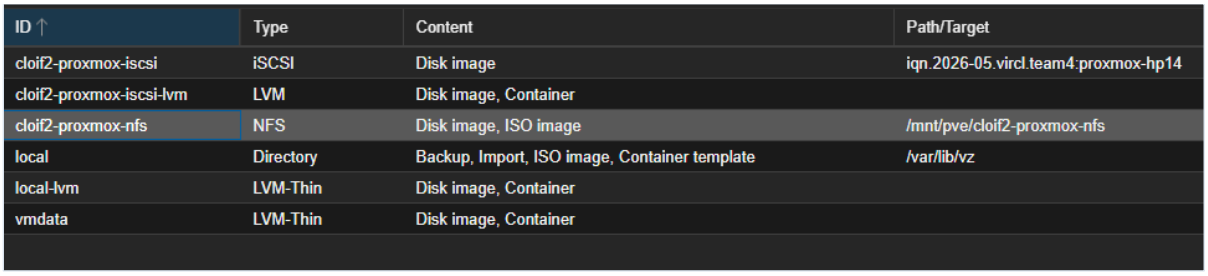


Figure 130 - Proxmox migration warning caused by attached local CD/DVD media.



ID	Type	Content	Path/Target
cloif2-proxmox-iscsi	iSCSI	Disk image	iqn.2026-05.vircl.team4:proxmox-hp14
cloif2-proxmox-iscsi-lvm	LVM	Disk image, Container	
cloif2-proxmox-nfs	NFS	Disk image, ISO image	/mnt/pve/cloif2-proxmox-nfs
local	Directory	Backup, Import, ISO image, Container template	/var/lib/vz
local-lvm	LVM-Thin	Disk image, Container	
vmdata	LVM-Thin	Disk image, Container	

Figure 131 - Proxmox storage list showing CLOIF2 iSCSI, LVM, and NFS entries.

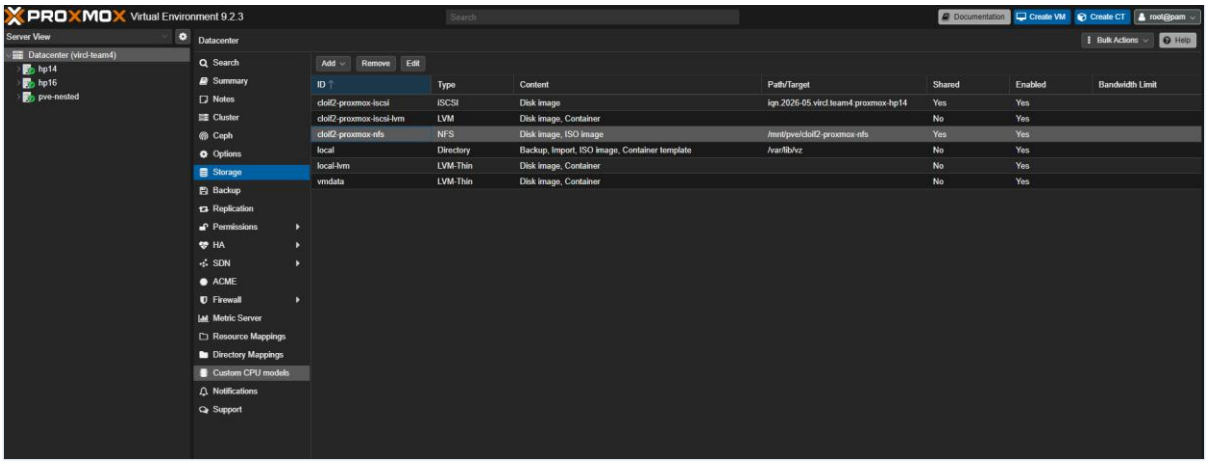


Figure 132 - Proxmox cluster storage view showing shared NFS enabled across nodes.

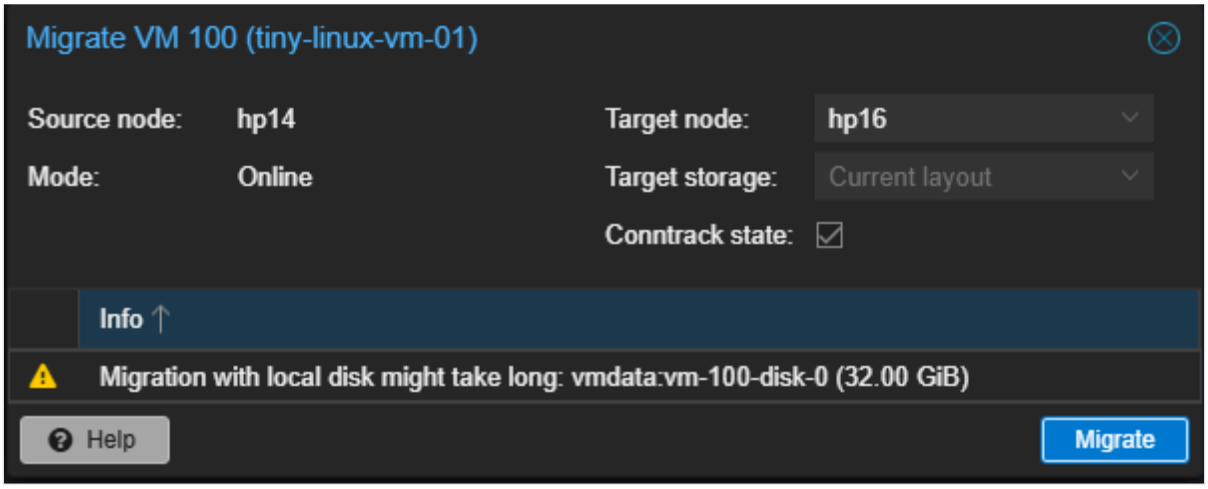


Figure 133 - Proxmox online migration dialog after local CD/DVD media was removed.

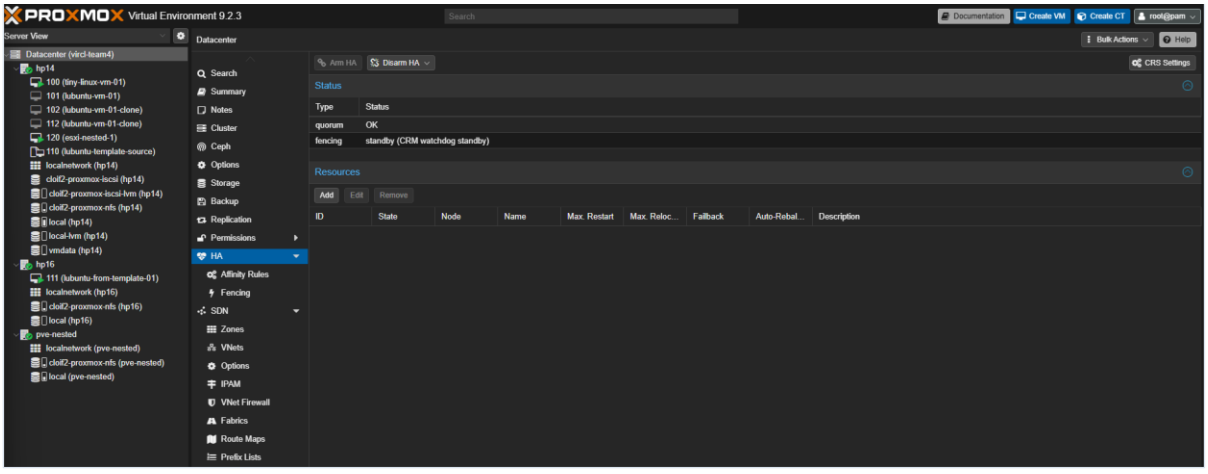


Figure 134 - Proxmox HA page showing quorum OK before adding the HA resource.

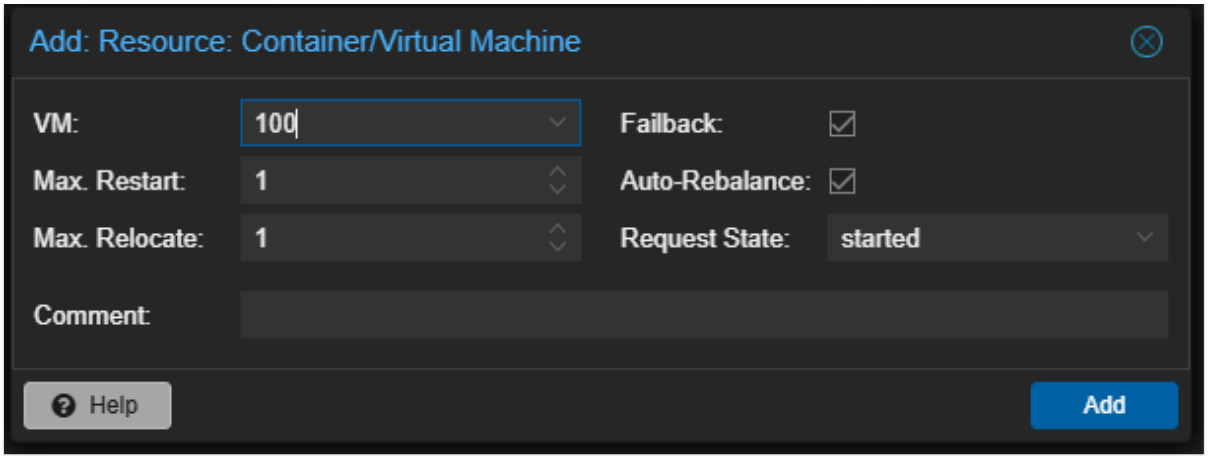


Figure 135 - Proxmox dialog for adding `VM 100` as an HA-managed resource.

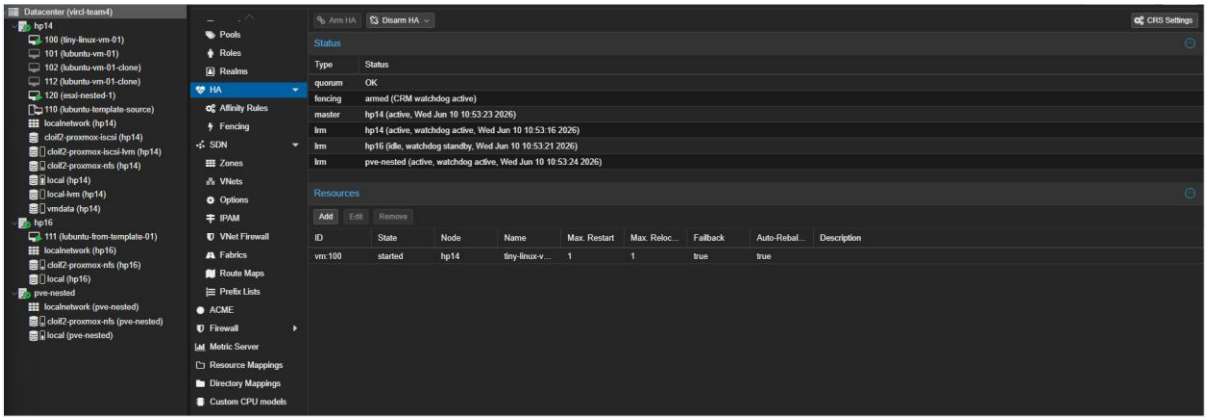


Figure 136 - Proxmox HA status showing `VM 100` as a started HA resource.

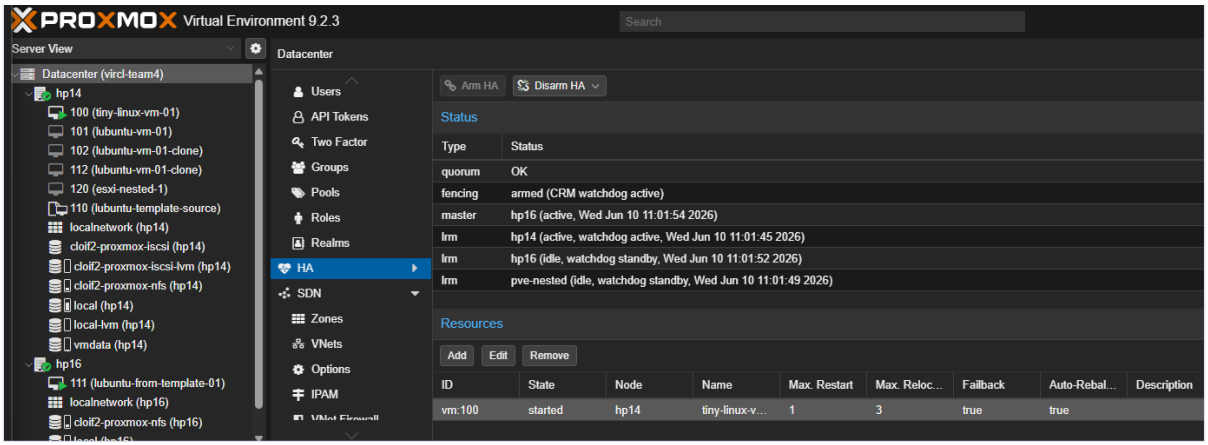


Figure 137 - Proxmox HA status with HP16 as active manager and `VM 100` still protected.

Personal conclusion

Gui

This project was useful because it forced the same virtualization ideas to be implemented in two different ecosystems rather than only described in theory. On both platforms I worked through the full stack — from installation and storage configuration through VM lifecycle, migration, backup, and clustering. On the ESXi side, handling HP15 installation and VM creation through vCenter made the difference between standalone host management and centralised vCenter management much clearer.

The biggest lesson for me was that the technical feature is often only half of the task. The other half is documentation: screenshots, exact commands, names, paths, and explaining limitations honestly. Several operations were straightforward once the storage and cluster state were correct, but they were difficult to document properly if the screenshot or command output was not captured at the right moment. I also learned that production-style virtualization depends heavily on shared storage, consistent networking, and central management. Features like vMotion, Proxmox online migration, templates, and backups only make sense when the underlying storage and network design is clean.

If I had to repeat the project, I would keep the same chapter-based documentation structure, but I would update the documentation checklist immediately after every lab block. Waiting even one week makes it harder to remember which screenshot proves which requirement. Overall, the project gave me a more practical understanding of how ESXi/vCenter and Proxmox solve the same administrator problems with different tools and assumptions.

Andrea

On the ESXi side, I was responsible for HP13 and, together with Gui, for HP15. HP13 was the most complex server to bring up: it required a full ESXi installation, the vCenter Server Appliance deployment in two stages, and then building the `VIRCL-Cluster` that both HP13 and HP15 joined. The vCenter deployment taught me how much depends on getting the foundation right before anything else works. The PNID had to be set to the IP address rather than the hostname because Stage 2 validation runs a DNS reverse lookup, and our lab DNS was not yet fully configured at that point. Every feature that came later — vMotion, HA, templates, Storage vMotion — only worked once vCenter, shared storage, and the correct network configuration were all in place at the same time.

On the Proxmox side, I installed HP16. The /24 vs /16 subnet issue had already been documented from Gui's HP14 install, so HP16 was the first server in the project where the network was configured correctly on the first attempt. I also worked on the clustering specialisation, which meant installing a third Proxmox node as a nested VM inside HP13's ESXi. That brought its own unexpected problem: the vSwitch port group on ESXi has a "Forged Transmits" security policy that is set to Reject by default, and that silently drops replies from the Linux bridge inside the VM. The fix is simple once you know what to look for, but it took a full troubleshooting session to identify.

The project confirmed something I did not expect going in: the two hypervisors solve the same administrator problems in fundamentally different ways. vCenter is a separate, centralised service that you deploy first and then everything else registers to it. Proxmox is peer-to-peer — any node can join the cluster and Corosync handles membership and quorum without a separate management server. Both approaches work, but they fail in different ways and require different troubleshooting instincts. Working

with both in the same project, on adjacent physical servers, made the contrast much clearer than any textbook comparison would have.

Sources

#	Source	Link
1	Broadcom / VMware vSphere Documentation	https://docs.vmware.com/en/VMware-vSphere/
2	Broadcom TechDocs - VMware vSphere product documentation	https://techdocs.broadcom.com/us/en/vmware-cis/vsphere.html
3	Broadcom Developer Portal - PowerCLI / VCF PowerCLI	https://developer.broadcom.com/powercli
4	Proxmox VE Administration Guide	https://pve.proxmox.com/pve-docs/pve-admin-guide.html
5	Proxmox VE qm manual	https://pve.proxmox.com/pve-docs/qm.1.html
6	Proxmox VE pveum manual	https://pve.proxmox.com/pve-docs/pveum.1.html
7	Proxmox VE vzdump manual	https://pve.proxmox.com/pve-docs/vzdump.1.html
8	Proxmox VE storage documentation	https://pve.proxmox.com/wiki/Storage
9	Microsoft Learn - Windows technical documentation	https://learn.microsoft.com/en-us/windows/
10	Ubuntu official documentation	https://help.ubuntu.com/
11	Lubuntu Manual	https://manual.lubuntu.me/stable/
12	Debian Reference	https://www.debian.org/doc/manuals/debian-reference/
13	Debian APT User's Guide	https://www.debian.org/doc/manuals/apt-guide/
14	Unraid Documentation	https://docs.unraid.net/
15	School project brief: Project_Server Virtualization Systems.pdf	N/A (internal document)
16	Course/lab material and teacher instructions provided during VIRCL lab sessions.	N/A (internal document)
17	Anthropic Claude Code & OpenAI Codex: research, deployment assistance, and document editing	https://www.anthropic.com/claude , https://openai.com/codex

Glossary

Term	Meaning
Type-1 hypervisor	A hypervisor installed directly on server hardware, without a normal host operating system.
VM	Virtual machine. A software-defined computer running on a hypervisor.
Guest OS	The operating system installed inside a VM, such as Windows, Ubuntu, or Lubuntu.
ESXi	VMware's bare-metal hypervisor used on HP13 and HP15.
vCenter	VMware's central management platform for ESXi hosts, clusters, permissions, vMotion, HA, and DRS.
VCSA	vCenter Server Appliance, the appliance-based deployment of vCenter.
Proxmox VE	Open-source virtualization platform based on Debian, KVM/QEMU, LXC, and a web management UI.
Datastore	A storage location used by a hypervisor to store VM files, ISO files, templates, and backups.
VMFS	VMware Virtual Machine File System, used by ESXi datastores.
NFS	Network File System, used to mount shared storage over the network.
iSCSI	Storage protocol that presents remote block storage over IP networking.
LVM-thin	Linux thin-provisioned logical volume storage used by Proxmox for VM disks.
NAS	Network-attached storage. In this project, CLOIF2/Unraid provided shared storage.
ISO	Disk image used as virtual installation media for operating systems.
vCPU	Virtual CPU assigned to a VM.
NIC	Network interface card. In a VM, this is a virtual network adapter.
Bridge	A virtual network switch used by Proxmox to connect VMs to the physical network.
vSwitch	VMware virtual switch used to connect VM networks and VMkernel networks.
Snapshot	A point-in-time VM state used to revert a VM after a test or change.
Clone	A copy of an existing VM.
Template	A reusable golden VM image used to deploy new VMs consistently.
OVF/OVA	VM export format used mainly on the VMware side. OVA is a single-file package; OVF is a folder/package format.

Term	Meaning
Backup	A copy of VM or hypervisor data used for recovery.
Restore	The process of recreating a VM or configuration from a backup.
Live migration	Moving a running VM from one host to another with minimal interruption.
vMotion	VMware's live migration feature.
Storage migration	Moving a VM disk from one datastore/storage target to another.
Cluster	A group of hypervisor hosts managed together for migration, resource management, and availability.
Quorum	The minimum number of cluster votes needed to safely make cluster decisions.
HA	High Availability. Restarts protected VMs on another host after a host failure.
DRS	Distributed Resource Scheduler. VMware feature for automated workload placement and balancing.
Corosync	Cluster communication and membership layer used by Proxmox.
PowerCLI	PowerShell module used to automate VMware/vCenter tasks.
qm	Proxmox command-line tool for managing QEMU/KVM virtual machines.
pveum	Proxmox command-line tool for users, roles, and permissions.
vzdump	Proxmox backup tool for VMs and containers.
noVNC	Browser-based console used by Proxmox to access VM screens.
EFI/BIOS	VM firmware modes that control how a virtual machine boots.