Lenovo

Design Guide: Hyperconverged Proxmox Virtual Environment

Last update: September 14, 2025

Version 1.0

Open Source Management of a Hyperconverged Infrastructure Environment

Virtualization with traditional Virtual Machines or Linux Containers

Using self-healing Ceph file system ensures data coherency at scale

3rd Party Applications can use Rest API to manage and monitor

Trick Hartman
Gary Cudak
Keith Adams

LENOVO PRESS

Table of Contents

1 Introducing Proxmox Virtual Environment 1

1.1 Why use the Proxmox Virtual Environment? 1

1.2 The key benefits 1

2 Before you get started 4

2.1 Functional prerequisites 4

2.2 Known limitations 8

2.3 Hardware configurations for Proxmox VE 8

3 Deployment requirements 9

4 Proxmox VE deployment steps 10

4.1 Deployment Preparations 10

5 Configuring Proxmox VE as an HCI Solution 14

5.1 Create the Proxmox VE Cluster - CLI 14

5.2 Configure Ceph Storage via CLI 16

5.3 Spinning up a VM via CLI 20

5.4 Graphical Configuration of the Solution 21

5.5 Configure Ceph Storage via Proxmox VE Web Interface 23

1 Introducing Proxmox Virtual Environment

The Proxmox Virtual Environment (VE) is an open-source software-defined management platform for enterprise virtualization and containerization applications. Using Linux KVM and Container (LXC) technology, Proxmox VE provides a resilient and stable platform for software-defined compute, network and storage functionality. Similarly to other closed source applications, Proxmox VE also provides a web-based user interface to monitor and manage virtual machines and containers along with integrated high-availability and integrated disaster recovery tools to comfortably handle clusters of several systems.

1.1 Why use the Proxmox Virtual Environment?

Proxmox VE combines both virtualization and containerization technologies onto a single platform. Thus, allowing for you to optimize existing resources and reduce costs in a hyperconverged scenario. Even the most demanding Linux and Windows workloads can be virtualized or containerized dynamically while allowing your storage to grow organically using proven software-defined applications like Ceph file system.

1.2 The key benefits

Many benefits of using the Proxmox Virtual Environment can raise the return on investment when using Lenovo ThinkSystem servers. Some of the main benefits of the Virtual Environment are as follows:

Cost Efficiency: A small or medium enterprise needs to have their costs under control. The Proxmox Virtual Environment (VE) allows a company to begin with the community edition for free and add support contracts as required when issues happen. The enterprise ready features are built in and supported starting from the community tier up to a premium tier with unlimited support and a promised 2-hour response time for any opened ticket.

Virtualization: Whether using traditional kernel based virtual machines (KVM) or container-based virtualization using Linux Containers (LXC), the Proxmox VE gives you two ways to run your business.

KVM is the leading technology for full virtualization which runs with near native performance on all modern x86_64 hardware like the Lenovo ThinkSystem family of servers. This allows you to create virtual instances of your Windows and Linux operating systems in virtual machines, where each VM has its own set of virtualized hardware and allows a server to run several application workloads on a single system.

Linux Containers (LXC) is a lightweight alternative, where only the specific libraries used by an application are wrapped together in an LXC package. This package is then run with all other container applications on a single host kernel operating system. Each of these packages are separated from all others running on the server and an administrator can easily manage the system separately from the application workloads.

Software-defined storage with Ceph: Among the more traditional storage types permitted by Proxmox VE, CephFS, an open-source distributed object store and file system, can easily be integrated to manage your software-defined infrastructure. Lenovo has Ceph storage solutions which combine the world's most reliable and secure Lenovo ThinkSystem SR650 V3 server with Ceph software to deliver a comprehensive software-defined storage platform for AI data lake houses, backup and archive, and data as-a-service use cases for hybrid cloud environments. Software-defined storage setups can be greatly enhanced using the

Ceph file system design and Proxmox VE allows an easy setup and management of the CephFS which can scale up to exabytes of storage and adds self-healing file system options.

Clustering: Proxmox VE allows a customer to start with a single node instance and expand it into a full-fledged clustered environment out of the box. This allows for live or online migrations without any apparent effect from the user side for nearly transparent maintenance. With a unique multi-master design, management of a cluster is greatly simplified so that an administrator may perform maintenance tasks from any node within a cluster. This allows a very simple management of all VMs and containers and storage from the Proxmox VE GUI.

Central Management: Proxmox central management service is a web interface which also provides a CLI and REST API so that it can be used for task automation. Whether via the multi-master web-based management interface, or even through a mobile application, you can manage your servers from one place. Proxmox VE also provides a command line interface which for advanced users, can provide all the functionality of the GUI with the ability to script and automate task creation. With the standardized REST API, third-party applications can also be written to handle management of the virtual environment.

Reliability, Availability & Security: With a built-in firewall, backup, restore and high-availability options, the Proxmox VE is well-suited to run your software-defined data center.

HCI Parity at a Glance (Ops Focus) of Key HCI Providers

This table maps core HCI capabilities across Proxmox VE 8.2, VMware vCenter/vSAN, and Nutanix Prism/AOS. Proxmox covers the fundamentals—compute HA and live migration; resilient, policy-driven storage via Ceph, integrated backup, and SDN overlays—while vCenter and Prism add deeper automation (e.g., DRS, one-click lifecycle/firmware, and advanced micro-segmentation). For many environments, Proxmox can achieve near-parity at lower software cost, trading a bit of operational effort for openness and flexibility.

CAPABILITY	PROXMOX VE (8.2)	VMWARE VSAN	NUTANIX PRISM/AOS
COMPUTE/HA	KVM/QEMU, live migration, HA manager	vMotion, HA, DRS	Live migration, HA, autonomous placement
STORAGE	Ceph RBD (RF/EC, CRUSH); CephFS for shares	vSAN (SPBM, dedupe/compress, EC)	AOS (RF2/RF3, EC-X, data reduction)
LIFECYCLE	GUI upgrades for PVE & Ceph; repo-based	vLCM for ESXi/drivers/firmware	LCM one-click (hypervisor+firmware+storage)

NETWORKING / SDN	VLAN, VXLAN/EVPN; firewall integration	NSX (overlay, micro-segmentation, services)	Flow (micro-segmentation, policy)
BACKUP / DR	Proxmox Backup Server + VM replication	vSphere Replication / SRM; rich ecosystem	Snapshots/replication; Leap DR
GPU / VGPU	SR-IOV/mdev/NVIDIA vGPU (more manual)	Mature vGPU; vMotion w/ vGPU (zero-copy)	AHV vGPU support (NVIDIA)

This document references ProxMox VE version 8.2 as the baseline for all instructions. Where we deviate for newer features, we will call out the specific version individually (for example, Proxmox VE 8.3) only when applicable.

2 Before you get started

To run the Proxmox VE system, the following prerequisites should be met. Using a Lenovo ThinkSystem or ThinkEdge server is the natural choice for such an environment.

2.1 Functional prerequisites

To successfully deploy Proxmox VE on Lenovo servers, we recommend one of the following server types:

Example Lenovo servers which can support Proxmox VE:

ThinkEdge	Edge	SE455 V3 (7DBY)
ThinkSystem	Rack 1U 1S	SR635 V3 [4th Gen EPYC] (7D9H/7D9G)
ThinkSystem	Rack 1U 2S	SR630 V3 [5th Gen Xeon] (7D72/7D73/7D74)
ThinkSystem	Rack 1U 2S	SR645 V3 [4th Gen EPYC] (7D9D/7D9C)
ThinkSystem	Rack 2U 1S	SR655 V3 [4th Gen EPYC] (7D9F/7D9E)
ThinkSystem	Rack 2U 2S	SR650 V3 [5th Gen Xeon] (7D75/7D76/7D77)
ThinkSystem	Rack 2U 2S	SR665 V3 [4th Gen EPYC] (7D9B/7D9A)
ThinkSystem	Rack 2U 4S	SR850 V3 (7D97/7D96/7D98)
ThinkSystem	Rack 3U 2S	SR675 V3 [4th Gen EPYC] (7D9Q/7D9R)
ThinkSystem	Rack 4U 4S	SR860 V3 (7D94/7D93/7D95)
ThinkSystem	Rack 8U 8S	SR950 V3 (7DC5/7DC4/7DC6)

For an up-to-date list of supported servers, check the Lenovo Operating System Interoperability Guide at: https://lenovopress.lenovo.com/osig#os_families=proxmox-virtual-environment&support=all

Ensure the server's firmware is updated to the latest version available from Lenovo's support site for optimal compatibility. Visit the Lenovo support site at: https://datacentersupport.lenovo.com/us/en/solutions/server-os-linux to ensure proper firmware levels.

2.1.1 Memory Requirements

A minimum of 2 GB of RAM is required for the Proxmox VE operating system and its services. Additional memory is necessary for guest VMs and containers—allocate at least 1 GB per VM or container, with higher amounts (e.g., 4–8 GB) recommended for demanding workloads. For Lenovo servers using advanced storage solutions like ZFS or Ceph, plan for an additional 1 GB of RAM per terabyte of storage.

2.1.2 Storage Configuration

Proxmox VE supports a variety of storage options compatible with Lenovo hardware, including local storage (DAS) and hardware RAID, or software-defined storage like ZFS or Ceph. Using SATA/SAS SSD drives for the OS is perfectly valid, although we recommend using NVMe SSD devices for the data drives to ensure you have the most efficient and fastest speeds for your virtual machines. Using external storage like the Lenovo ThinkSystem DM or DE Series All Flash SAN arrays is also possible. Lenovo ThinkSystem DE Series All-Flash systems are ideal for mixed workloads and demanding applications such as analytics, high-performance computing (HPC), and artificial intelligence (AI); whereas Lenovo ThinkSystem DM Series All-Flash arrays are ideal for performance critical applications such as Oracle, Microsoft SQL Server, MongoDB databases, VDI, and server virtualization, to data-intensive AI training, tuning, inferencing, and retrieval-augmented generation (RAG) workloads.

We suggest using ThinkSystem M.2 or 7mm SATA/SAS SSD 240GB (or larger) boot disks with a hardware raid controller to ensure a basic level of RAID1 resilience for the Proxmox VE operating system. When using ZFS or CephFS as the storage subsystem, both require raw disk access to the data drives and do not require a hardware RAID controller. These file systems require you to configure drives in JBOD mode when using them. In this document, we use Ceph RBD for VM/CT disks (block storage with snapshots/clones). Use CephFS for shared files (ISOs, templates, backup targets).

2.1.3 UEFI Settings

Secure boot considerations

Enabled by default on Proxmox VE 8.1 and later. Only disable Secure Boot if you intentionally run unsigned DKMS modules and choose not to sign them, or for temporary troubleshooting of rare firmware/key issues. For more information, visit the Proxmox Wiki at: https://pve.proxmox.com/wiki/Secure_Boot_Setup .Using the XClarity Controller (XCC), select this path:

- Settings > Security > Secure Boot Configuration

In the Secure Boot configuration screen, select the preferred Secure Boot setting for your use case, and ensure it is set as desired. By default, it is *Enabled*.

Ensure proper virtualization configuration

Activate IOMMU for passthrough (under System Settings > Devices). This is set to Enabled by default, but it is helpful to ensure that nothing has changed.

Performance Considerations

For best performance, ensure that the operating mode is set to High Performance, at System Settings -> Operating Modes The purpose for this setting, is to ensure dynamic power-saving features, such as C-states and P-states are disabled, CPU frequency is locked at maximum speed and that throughput and responsiveness are prioritized over power efficiency.

These settings impact the HCI elements of Proxmox VE cluster and Ceph in these manners:

- Improved write operations for Ceph OSDs, quicker PG recovery and balance and improved SCSI host responsiveness
- Reduced jitter for VM and container launch and memory allocation
- More predictable CPU response, under load
- Quicker reaction times for fencing, quorum changes and monitor elections

2.2 Known limitations

2.2.1 Driver Support:

Proxmox VE uses a custom Debian-based kernel, which generally includes drivers for Lenovo's common NICs (e.g., Broadcom NetXtreme, Intel I350) and storage controllers. However, newer Lenovo hardware (e.g., ThinkSystem servers with cutting-edge NVMe controllers introduced in late 2024) may require manual driver updates if not yet included into the Proxmox kernel. Check Lenovo's Linux compatibility matrix, at https://datacentersupport.lenovo.com/us/en/solutions/server-os-linux and the Proxmox community forums for updates.

2.2.2 System Health Considerations:

XCC's IPMI and Redfish APIs report hardware health, but Proxmox VE lacks a native agent to poll these endpoints. Manual integration via ipmitool or custom scripts ~~is~~ are needed for fan speed or temp alerts.

2.2.3 NVMe Hot-Swap:

ThinkSystem NVMe backplanes support physical hot-swap. On PVE 8.2, the kernel usually detects insert/remove events automatically for native NVMe devices. If a new drive doesn't appear, trigger an NVMe rescan with `echo 1 > /sys/class/nvme/nvme0/rescan_controller and nvme ns-rescan /dev/nvme0.`

2.3 Hardware configurations for Proxmox VE

Proxmox VE can be based on any of the Lenovo ThinkSystem servers available as long as they meet the basic system requirements[1] for Proxmox VE. The server we tested this design guide can be seen in Table 1.

Table 1. Test Hardware Configuration

Server	Lenovo ThinkSystem SR635 V3
Processor	1x ThinkSystem AMD EPYC 9334 32C 210W 2.7GHz Processor
Memory	192 GB (6x ThinkSystem 32GB TruDDR5 4800MHz (2Rx8) RDIMM-A)
NIC	1x ThinkSystem Intel E810-DA4 10/25GbE SFP28 4-Port OCP Ethernet Adapter
Disk	16x ThinkSystem E1.S 5.9mm 7450 PRO 7.68TB Read Intensive NVMe PCIe 4.0 x4 HS SSD

[1] Proxmox VE System Requirements: https://www.proxmox.com/en/products/proxmox-virtual-environment/requirements

3 Deployment requirements

Below are the recommended system specifications for optimal performance and reliability:

- Memory:
  - Minimum 2 GB for the OS and Proxmox VE services, plus designated memory for guests.
  - For Ceph and ZFS, approximately 1GB of memory for every TB of used storage.
- Fast and redundant storage, best results are achieved with NVMe SSDs.
- OS storage: Use ThinkSystem M.2/7mm internal boot options to store the operating system.
- VM storage:
  - For local storage, use either a hardware RAID with battery backed write cache (BBU) for local disks or non-RAID for ZFS and Ceph storage.
  - Neither ZFS nor Ceph are compatible with a hardware RAID controller.
- Shared and distributed storage is possible.
- NVMe SSDs with high performance solid-state drive (SSD) technology that provides high I/O throughput and low latency are recommended for best performance.
- Redundant (Multi-)Gbit NICs, with additional NICs depending on the preferred storage technology and cluster setup.
- For PCI(e) passthrough the CPU needs to support the VT-d/AMD-d flag.

4 Proxmox VE deployment steps

The installer ISO image includes the following:

- Complete operating system (Debian Linux, 64-bit)
- The Proxmox VE installer, which partitions the local disk(s) with ext4, XFS, BTRFS (technology preview), or ZFS and installs the operating system
- Proxmox VE Linux kernel with KVM and LXC support
- Complete toolset for administering virtual machines, containers, the host system, clusters and all necessary resources
- Web-based management interface

4.1 Deployment Preparations

4.1.1 Download Proxmox ISO

Visit the official Proxmox website to download[2] the latest Proxmox Virtual Environment (VE) available.

4.1.2 Mount Bootable ISO to XCC Remote Console

Using the instructions[3] from Lenovo on how to mount the Proxmox ISO, install the Proxmox VE system. This article introduces the procedure used to mount local media files as virtual drives in the XClarity Controller (XCC).

4.1.3 Boot Installation Media

Once the installer has booted, the Proxmox VE menu will be displayed.

- Install Proxmox VE (Graphical) – Starts the normal graphical installation
- Install Proxmox VE (Terminal UI) – Starts the terminal-mode installation wizard
- Install Proxmox VE (Terminal UI, Serial Console) – Starts terminal-mode with serial console

---

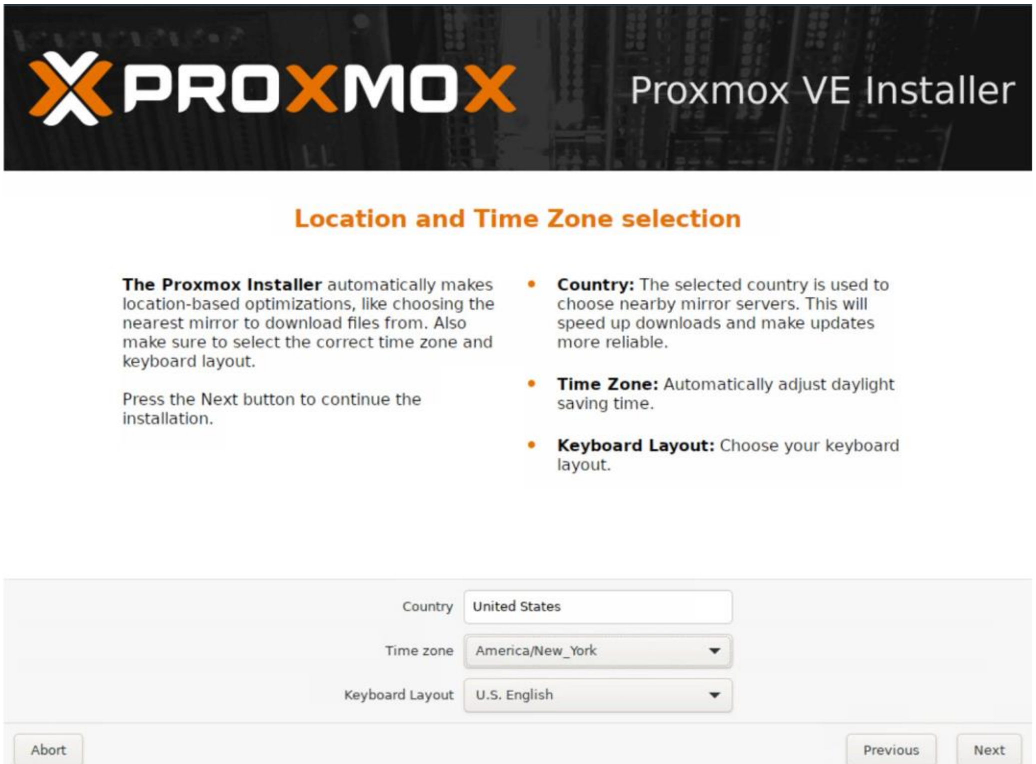
[2] Proxmox VE Download - https://www.proxmox.com/en/downloads/proxmox-virtual-environment

[3] XCC ISO Mounting instructions - https://support.lenovo.com/us/en/solutions/ht507498-how-to-mount-an-iso-image-locally-through-lxcc

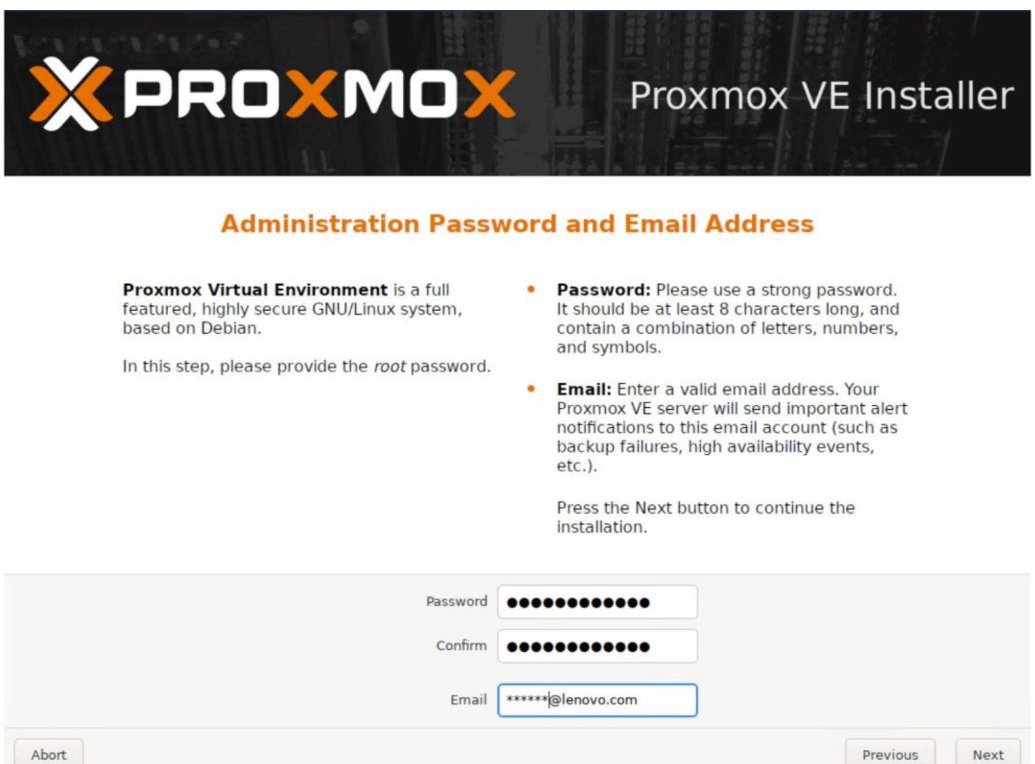
For this deployment, we selected **Install Proxmox VE (Graphical)** to start the installation. Accept the **End User License Agreement (EULA)**.

Select the target hard drive where Proxmox VE will be installed. We choose on the Lenovo ThinkSystem SR635 V3 the 7mm NVMe SSD device.

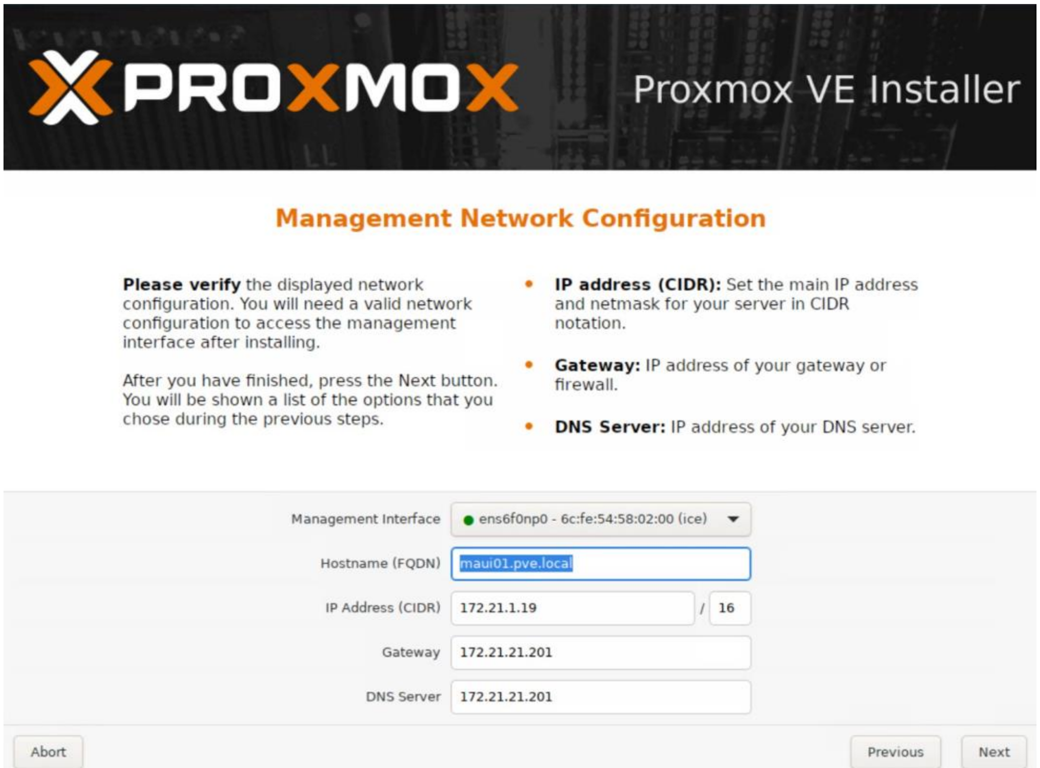
Choose your **country, time zone, and keyboard layout**.



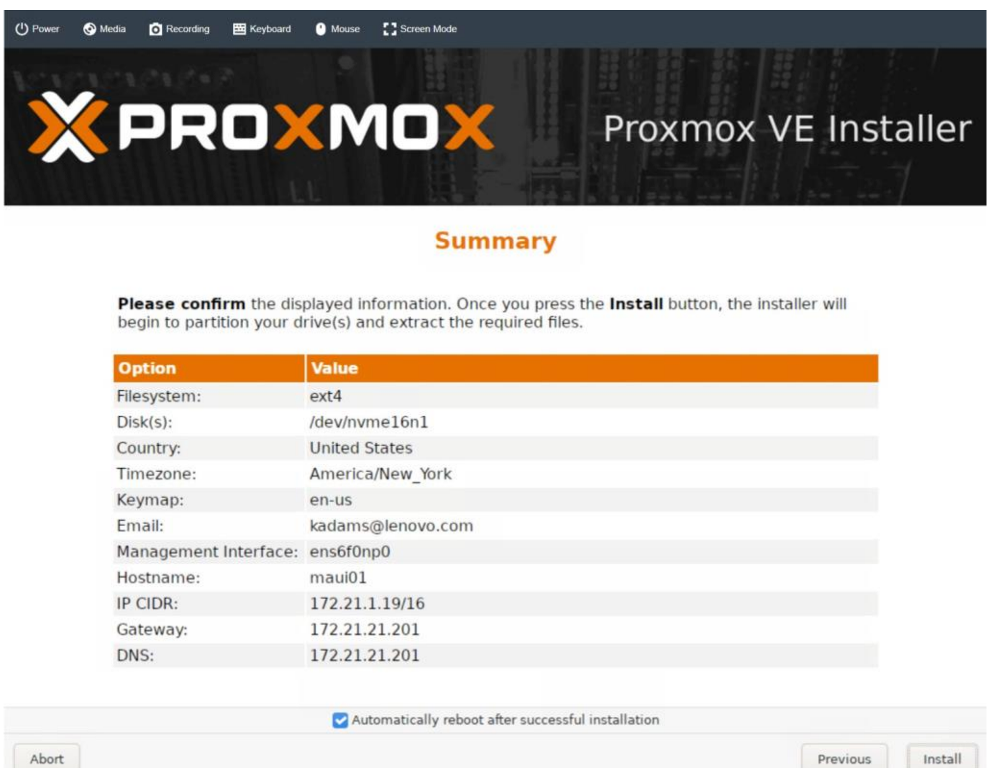
Set the **root password** and provide an **email address** for system alerts.



Set up your **network settings** (choose a static IP or DHCP). **TIP**: initially plug only a single port to install the Proxmox VE OS. Later, you can configure all network ports for redundancy or dedicated host networks.



Please review all settings carefully. Click Install to start the installation. This may take a few minutes to complete. Once installation finishes, the system will prompt you to remove the installation media and **Reboot**.



5 Configuring Proxmox VE as an HCI Solution

The goal is to create an HCI environment which is conceptually like a Nutanix software-defined or VMware VSAN storage setup. The order of operations will be:

- Create and verify the Proxmox VE Cluster
- Initialize Ceph and create monitors: Set up the Ceph cluster with monitors to manage the storage environment.
- Prepare and create Ceph OSDs (Disks): Configure disks as Ceph Object Storage Daemons (OSDs) for data storage.
- Create Ceph Storage Pools for Virtual Machines
- Integrate Ceph with Proxmox VE: Configure Proxmox to use Ceph storage for VMs and containers.

In general, we will approach the configuration process via both command line interface, initially, followed by a brief setup using the graphical interface. This document assumes that the nodes are installed, updated and networking is already configured.

5.1 Create the Proxmox VE Cluster - CLI

We will begin by creating and verifying the cluster from the CLI. Start by running `pvecm create <cluster name>`. In this instance, we are calling the cluster `ProxC1`. Once that has completed, run `pvecm status` to verify the creation. You should have output like the image that follows:

Next, we will add nodes to the cluster and verify again. Execute `pvecm add <IP address>` where IP address is the IP address of the head node, from each of the other nodes that are planned for inclusion.

Here, we have added the node.

```
root@prox3:~#
root@prox3:~# pvecm add 172.21.1.19
Please enter superuser (root) password for '172.21.1.19': ************
Establishing API connection with host '172.21.1.19'
The authenticity of host '172.21.1.19' can't be established.
X509 SHA256 key fingerprint is 03:9F:4C:2A:85:34:D7:62:A5:23:68:B8:0D:75:43:FB:96:08:A0:FC:C2:18:25:F1:DA:2B:50:C7:E8:A2:93:A7.
Are you sure you want to continue connecting (yes/no)? yes
Login succeeded.
check cluster join API version
No cluster network links passed explicitly, fallback to local node IP '172.21.1.21'
Request addition of this node
Join request OK, finishing setup locally
stopping pve-cluster service
backup old database to '/var/lib/pve-cluster/backup/config-1752592319.sql.gz'
waiting for quorum...OK
(re)generate node files
generate new node certificate
merge authorized SSH keys
generated new node certificate, restart pveproxy and pvedaemon services
successfully added node 'prox3' to cluster.
```

We will verify the status of the cluster from the node. We can see from the `pvecm status` command, that all 3 nodes are included, and the cluster is healthy.

```
root@prox3:~#
root@prox3:~# pvecm status
Cluster information
-------------------
Name:             ProxC1
Config Version:   3
Transport:        knet
Secure auth:      on

Quorum information
------------------
Date:             Tue Jul 15 06:14:07 2025
Quorum provider:  corosync_votequorum
Nodes:            3
Node ID:          0x00000003
Ring ID:          1.d
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate

Membership information
----------------------
    Nodeid      Votes Name
0x00000001          1 172.21.1.19
0x00000002          1 172.21.1.20
0x00000003          1 172.21.1.21 (local)
root@prox3:~# _
```

5.2 Configure Ceph Storage via CLI

On Proxmox VE 8.2, Ceph Quincy (17.2) is supported and commonly used; Reef (18.2) is also supported and is the default on newer installs since 8.1. Our build uses Quincy.

First, we will install the Ceph packages on each node. If you don't have a subscription, you will need to alter the apt repositories from the PVE enterprise link to the generic link. After those changes, run `pveceph install -repository no-subscription.`

```
root@prox3:~# pveceph install --repository no-subscription

HINT: The no-subscription repository is not the best choice for production setups.
Proxmox recommends using the enterprise repository with a valid subscription.
This will install Ceph Quincy - continue (y/N)?
```

A successful result will appear as:

```
installed ceph quincy successfully!

reloading API to load new Ceph RADOS library...
root@prox3:~#
```

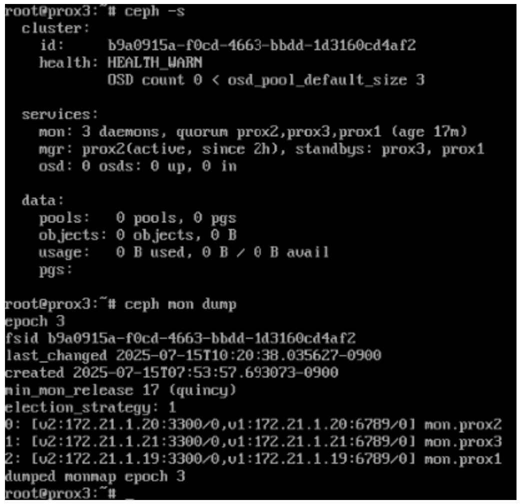
Next, we will initialize Ceph on the head node, which will set up the initial Ceph cluster configuration and the monitor and manager daemons. This is executed at the subnet level, running `pveceph init -network 172.21.1.0/24`.

```
root@prox1:~# pveceph init --network 172.21.1.0/24
creating /etc/pve/priv/ceph.client.admin.keyring
root@prox1:~#
```

On the remaining nodes, we will create monitors, to establish a distributed metadata and quorum system across the subnet by running `pveceph createmon`.

```
root@prox2:~# pveceph createmon
unable to get monitor info from DNS SRV with service name: ceph-mon
rados_connect failed - No such file or directory
creating new monitor keyring
creating /etc/pve/priv/ceph.mon.keyring
importing contents of /etc/pve/priv/ceph.client.admin.keyring into /etc/pve/priv/ceph.mon.keyring
monmaptool: monmap file /tmp/monmap
monmaptool: generated fsid a025e1be-3133-4b51-8d0d-2b64562c05e3
setting min_mon_release = octopus
epoch 0
fsid a025e1be-3133-4b51-8d0d-2b64562c05e3
last_changed 2025-07-15T07:53:57.693073-0900
created 2025-07-15T07:53:57.693073-0900
min_mon_release 15 (octopus)
election_strategy: 1
0: [v2:172.21.1.20:3300/0,v1:172.21.1.20:6789/0] mon.prox2
monmaptool: writing epoch 0 to /tmp/monmap (1 monitors)
created the first monitor, assume it's safe to disable insecure global ID reclaim for new setup
Configuring keyring for ceph-crash.service
Created symlink /etc/systemd/system/ceph-mon.target.wants/ceph-mon@prox2.service -> /lib/systemd/system/ceph-mon@.service.
creating manager directory '/var/lib/ceph/mgr/ceph-prox2'
creating keys for 'mgr.prox2'
setting owner for directory
enabling service 'ceph-mgr@prox2.service'
Created symlink /etc/systemd/system/ceph-mgr.target.wants/ceph-mgr@prox2.service -> /lib/systemd/system/ceph-mgr@.service.
starting service 'ceph-mgr@prox2.service'
root@prox2:~#
```

Verify the state of the Ceph cluster with `ceph -s` and `ceph mon dump`. You should see results like the image below.



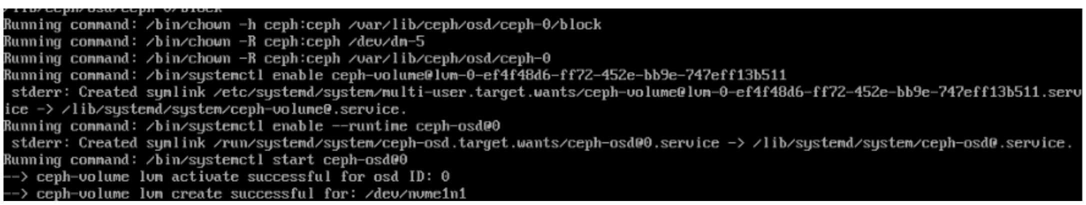
```
root@prox3:~# ceph -s
  cluster:
    id:     b9a0915a-f0cd-4663-bbdd-1d3160cd4af2
    health: HEALTH_WARN
            OSD count 0 < osd_pool_default_size 3

  services:
    mon: 3 daemons, quorum prox2,prox3,prox1 (age 17m)
    mgr: prox2(active, since 2h), standbys: prox3, prox1
    osd: 0 osds: 0 up, 0 in

  data:
    pools:   0 pools, 0 pgs
    objects: 0 objects, 0 B
    usage:   0 B used, 0 B / 0 B avail
    pgs:

root@prox3:~# ceph mon dump
epoch 3
fsid b9a0915a-f0cd-4663-bbdd-1d3160cd4af2
last_changed 2025-07-15T10:20:38.035627-0900
created 2025-07-15T07:53:57.693073-0900
min_mon_release 17 (quincy)
election_strategy: 1
0: [v2:172.21.1.20:3300/0,v1:172.21.1.20:6789/0] mon.prox2
1: [v2:172.21.1.21:3300/0,v1:172.21.1.21:6789/0] mon.prox3
2: [v2:172.21.1.19:3300/0,v1:172.21.1.19:6789/0] mon.prox1
dumped monmap epoch 3
root@prox3:~# _
```

Ensure you wipe the disks prior to creating the OSD, then create an OSD on each node, by running `pveceph createosd <device>`, for example using the second NVMe device: `pveceph createosd /dev/nvme1n1`.



```
Running command: /bin/chown -h ceph:ceph /var/lib/ceph/osd/ceph-0/block
Running command: /bin/chown -R ceph:ceph /dev/dm-5
Running command: /bin/chown -R ceph:ceph /var/lib/ceph/osd/ceph-0
Running command: /bin/systemctl enable ceph-volume@lvm-0-ef4f48d6-ff72-452e-bb9e-747eff13b511
 stderr: Created symlink /etc/systemd/system/multi-user.target.wants/ceph-volume@lvm-0-ef4f48d6-ff72-452e-bb9e-747eff13b511.service -> /lib/systemd/system/ceph-volume@.service.
Running command: /bin/systemctl enable --runtime ceph-osd@0
 stderr: Created symlink /run/systemd/system/ceph-osd.target.wants/ceph-osd@0.service -> /lib/systemd/system/ceph-osd@.service.
Running command: /bin/systemctl start ceph-osd@0
--> ceph-volume lvm activate successful for osd ID: 0
--> ceph-volume lvm create successful for: /dev/nvme1n1
```

Once these steps are complete, they can be verified using `ceph osd tree`, `ceph -s` *and* `ceph mon dump`. We can see that health is OK, that all 3 monitors are online and that the manager is active. Lastly, that each node has an OSD configured.

```
root@prox1:/var/lib/ceph/mon# ceph osd tree
ID  CLASS  WEIGHT    TYPE NAME       STATUS  REWEIGHT  PRI-AFF
-1         20.95889  root default
-5          6.98630      host prox1
 1    ssd   6.98630          osd.1       up   1.00000  1.00000
-7          6.98630      host prox2
 2    ssd   6.98630          osd.2       up   1.00000  1.00000
-3          6.98630      host prox3
 0    ssd   6.98630          osd.0       up   1.00000  1.00000
root@prox1:/var/lib/ceph/mon# ceph -s
  cluster:
    id:     b9a0915a-f0cd-4663-bbdd-1d3160cd4af2
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum prox2,prox3,prox1 (age 14m)
    mgr: prox2(active, since 14m), standbys: prox3, prox1
    osd: 3 osds: 3 up (since 41s), 3 in (since 52s)

  data:
    pools:   1 pools, 1 pgs
    objects: 2 objects, 577 KiB
    usage:   883 MiB used, 21 TiB / 21 TiB avail
    pgs:     1 active+clean

root@prox1:/var/lib/ceph/mon# ceph mon dump
epoch 3
fsid b9a0915a-f0cd-4663-bbdd-1d3160cd4af2
last_changed 2025-07-15T10:20:38.035627-0900
created 2025-07-15T07:53:57.693073-0900
min_mon_release 17 (quincy)
election_strategy: 1
0: [v2:172.21.1.20:3300/0,v1:172.21.1.20:6789/0] mon.prox2
1: [v2:172.21.1.21:3300/0,v1:172.21.1.21:6789/0] mon.prox3
2: [v2:172.21.1.19:3300/0,v1:172.21.1.19:6789/0] mon.prox1
dumped monmap epoch 3
root@prox1:/var/lib/ceph/mon#
```

One final check is to check the status of the Ceph cluster with: `ceph osd status`.

```
root@prox1:/var/lib/ceph/mon# ceph osd status
ID  HOST    USED  AVAIL  WR OPS  WR DATA  RD OPS  RD DATA  STATE
 0  prox3   295M  7153G      0        0       0        0   exists,up
 1  prox1   295M  7153G      0        0       0        0   exists,up
 2  prox2   295M  7153G      0        0       0        0   exists,up
```

You can repeat these steps for each drive, in each system which is planned to participate in the OSD. To see the list of disks available, run `lsblk -o NAME,SIZE,MODEL,TYPE,MOUNTPOINT`. Disks without partitions are potentially available, depending on the planned usage.

```
root@prox1:/var/lib/ceph/mon# lsblk -o NAME,SIZE,MODEL,TYPE,MOUNTPOINT
NAME                                                                    SIZE MODEL                        TYPE MOUNTPOINT
nvme16n1                                                              894.2G 7mm NVMe 2-Bay Rear RAID Kit disk
├─nvme16n1p1                                                           1007K                              part
├─nvme16n1p2                                                              1G                              part
└─nvme16n1p3                                                          893.2G                              part
  ├─pve--OLD--2BDA15F6-swap                                               8G                              lvm
  ├─pve--OLD--2BDA15F6-root                                              96G                              lvm
  ├─pve--OLD--2BDA15F6-data_tmeta                                       7.7G                              lvm
  │ └─pve--OLD--2BDA15F6-data                                         757.7G                              lvm
  └─pve--OLD--2BDA15F6-data_tdata                                     757.7G                              lvm
    └─pve--OLD--2BDA15F6-data                                         757.7G                              lvm
nvme1n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
└─ceph--bf044ece--52af--4990--a8cc--b6abf61d855b-osd--block--65677cb8--0f1c--4018--bb4c--dea180bc455f
                                                                          7T                              lvm
nvme0n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
├─nvme0n1p1                                                            1007K                              part
├─nvme0n1p2                                                               1G                              part /boot/efi
└─nvme0n1p3                                                               7T                              part
  ├─pve-swap                                                              8G                              lvm  [SWAP]
  ├─pve-root                                                             96G                              lvm  /
  ├─pve-data_tmeta                                                     15.9G                              lvm
  │ └─pve-data                                                          6.8T                              lvm
  └─pve-data_tdata                                                      6.8T                              lvm
    └─pve-data                                                          6.8T                              lvm
nvme3n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme2n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme13n1                                                                  7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme8n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme6n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme5n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme7n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme15n1                                                                  7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme14n1                                                                  7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme4n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme9n1                                                                   7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme10n1                                                                  7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme11n1                                                                  7T Micron_7450_MTFDKBZ7T6TFR    disk
nvme12n1                                                                  7T Micron_7450_MTFDKBZ7T6TFR    disk
```

Once the desired OSDs are configured, check the OSD tree again to verify.

```
root@prox3:~# ceph osd tree
ID  CLASS  WEIGHT     TYPE NAME       STATUS  REWEIGHT  PRI-AFF
-1         167.67114  root default
-5          55.89038      host prox1
 1    ssd    6.98630          osd.1       up   1.00000  1.00000
10    ssd    6.98630          osd.10      up   1.00000  1.00000
11    ssd    6.98630          osd.11      up   1.00000  1.00000
12    ssd    6.98630          osd.12      up   1.00000  1.00000
13    ssd    6.98630          osd.13      up   1.00000  1.00000
14    ssd    6.98630          osd.14      up   1.00000  1.00000
15    ssd    6.98630          osd.15      up   1.00000  1.00000
16    ssd    6.98630          osd.16      up   1.00000  1.00000
-7          55.89038      host prox2
 2    ssd    6.98630          osd.2       up   1.00000  1.00000
 3    ssd    6.98630          osd.3       up   1.00000  1.00000
 4    ssd    6.98630          osd.4       up   1.00000  1.00000
 5    ssd    6.98630          osd.5       up   1.00000  1.00000
 6    ssd    6.98630          osd.6       up   1.00000  1.00000
 7    ssd    6.98630          osd.7       up   1.00000  1.00000
 8    ssd    6.98630          osd.8       up   1.00000  1.00000
 9    ssd    6.98630          osd.9       up   1.00000  1.00000
-3          55.89038      host prox3
 0    ssd    6.98630          osd.0       up   1.00000  1.00000
17    ssd    6.98630          osd.17      up   1.00000  1.00000
18    ssd    6.98630          osd.18      up   1.00000  1.00000
19    ssd    6.98630          osd.19      up   1.00000  1.00000
20    ssd    6.98630          osd.20      up   1.00000  1.00000
21    ssd    6.98630          osd.21      up   1.00000  1.00000
22    ssd    6.98630          osd.22      up   1.00000  1.00000
23    ssd    6.98630          osd.23      up   1.00000  1.00000
root@prox3:~# ceph osd stat
24 osds: 24 up (since 4m), 24 in (since 4m); epoch: e126
```

Following this, we will create the placement groups (PG). In this scenario, we will use 512 PGs, since it is practical in this scenario where we have 24 OSDs. The general formula for PG quantity can be defined as:

Target PGs = (# OSDs * 100) / replication factor, and we have a replication factor of 3. While the math works out to 800 PGs, additional PGs require more background threads and impacts performance; additionally, PGs can be added later but cannot be removed. It is equally viable to start with less than 100 PGs for a small cluster. In the case of a cluster expected to have highly parallel workloads or heavy I/O, as well as running on servers with high performance drives, such as NVMe, higher PG counts are the better choice.

We will run, from our Ceph manager (prox2), `ceph osd pool create <pool name> 512 512`, which simply yields a message indicating the successful create of the pool. In this case, it will be called `vmstore`.

Alternatively, instead of fixing `pg_num` up front, you can enable Ceph's PG autoscaler and let it size/adjust PGs based on the cluster and each pool's expected share of data. This keeps small clusters lean and scales PGs up as the pool grows. In practice you set replica size, turn autoscaling on, and—if you have multiple pools—give each a target_size_ratio so Ceph knows how to divide PGs. You can read more at: https://docs.ceph.com/en/latest/rados/operations/placement-groups/ .

Next, we will declare the usage of the pool, since Ceph requires that information to ensure proper internal checks, optimizations and compatibility enforcement. Run the command `ceph osd pool application enable vmstore rbd`.

```
root@prox3:~# ceph osd pool create vmstore 512 512
pool 'vmstore' created
root@prox3:~# ceph osd pool application enable vmstore rbd
enabled application 'rbd' on pool 'vmstore'
```

Next, we will register the RBD pool in Proxmox as a storage backend, running:

```
pvesh create /storage --storage vmstore --type rbd --content images -monhost
"172.21.1.19;172.21.1.20;172.21.1.21" --pool vmstore --krbd 1
```

Key options in the command: `--type rbd` sets the RADOS block device flag; `--content images` sets the content expectation to VM disks; `--pool vmstore` aligns to the pool we created and `--krbd 1` ensures use of the kernel RBD driver.

And we can verify with `pvesh get /storage` and `pvesh get /storage/vmstore`.

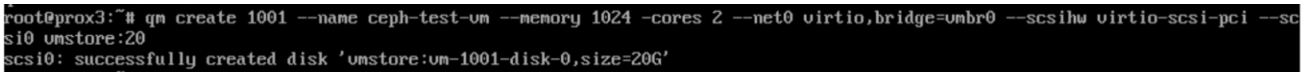
```
root@prox3:~# pvesh create /storage --storage vmstore --type rbd --content images --monhost "172.21.1.19;172.21.1.20;172.21.1.21
" --pool vmstore --krbd 1
┌─────────┬─────────┐
│ key     │ value   │
├─────────┼─────────┤
│ storage │ vmstore │
├─────────┼─────────┤
│ type    │ rbd     │
└─────────┴─────────┘
root@prox3:~# pvesh get /storage
┌───────────┐
│ storage   │
├───────────┤
│ local     │
├───────────┤
│ local-lvm │
├───────────┤
│ vmstore   │
└───────────┘
```

```
root@prox3:~# pvesh get /storage/vmstore
┌─────────┬──────────────────────────────────────────┐
│ key     │ value                                    │
├─────────┼──────────────────────────────────────────┤
│ content │ images                                   │
├─────────┼──────────────────────────────────────────┤
│ digest  │ 031b10585f5b1fd1521b77b8b1d13e431de2bd5f │
├─────────┼──────────────────────────────────────────┤
│ krbd    │ 1                                        │
├─────────┼──────────────────────────────────────────┤
│ monhost │ 172.21.1.19;172.21.1.20;172.21.1.21      │
├─────────┼──────────────────────────────────────────┤
│ pool    │ vmstore                                  │
├─────────┼──────────────────────────────────────────┤
│ shared  │ 1                                        │
├─────────┼──────────────────────────────────────────┤
│ storage │ vmstore                                  │
├─────────┼──────────────────────────────────────────┤
│ type    │ rbd                                      │
└─────────┴──────────────────────────────────────────┘
```

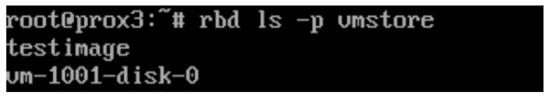
5.3 Spinning up a VM via CLI

Now that the PVE cluster is defined and Ceph storage is configured and linked to the PVE cluster as VM image storage, we will create a virtual machine. Specifically, we will define the new virtual machine, with identity 1001, with a 20GB virtual disk on the Ceph RBD pool, with 1GB of RAM and 2 CPU cores, with the virtio-scsi storage controller, for optimal I/O. Run the following command:

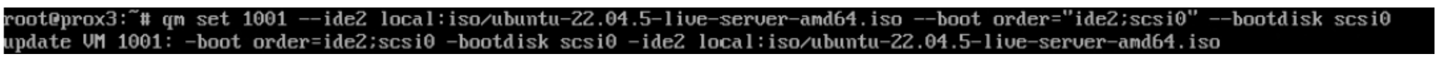
```
> qm create 1001 --name ceph-test-vm --memory 1024 --cores 2 --net0
virtio,bridge=vmbr0 --scsihw virtio-scsi-pci --scsi0 vmstore:20
```



We can verify the existence of the VM image via `rbd ls -p vmstore`.



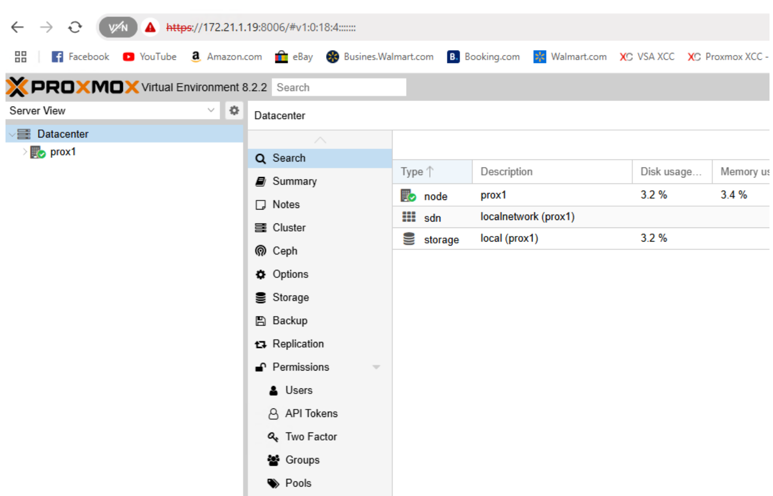
Next, we will attach the VM to an OS ISO. Run: `qm set 1001 --ide2 local:iso/ubuntu-22.04.5-live-server-amd64.iso --boot order=ide2;scsi0 --bootdisk scsi0`



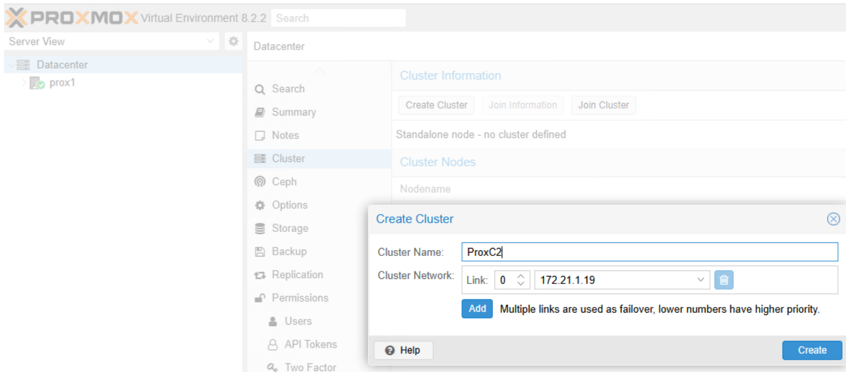
At this point, we can start the VM with `qm start 1001`.

5.4 Graphical Configuration of the Solution

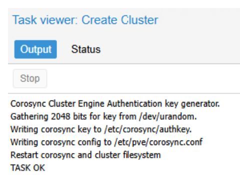
Here, we will rebuild the HCI solution, using the graphical components provided by Proxmox. We will begin by accessing the graphical web console at port 8006, e.g. http://172.21.1.19:8006. For this example, we will continue using the root user, but in a production environment, it is recommended to create specific user roles for PVE and Ceph. Once you have logged in, you will see the unconfigured Datacenter view for that specific node, as shown in the image below.



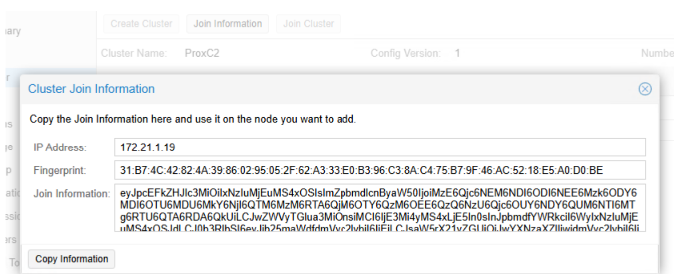
Begin by navigating to Datacenter -> Cluster and selecting “Create Cluster”. You’ll get a window where the cluster name can be entered, and if you have multiple links for redundancy, you can set them up by adding additional links prior to selecting “Create”. In this case, we will use the bridge we already have, **`vmbr0`**.



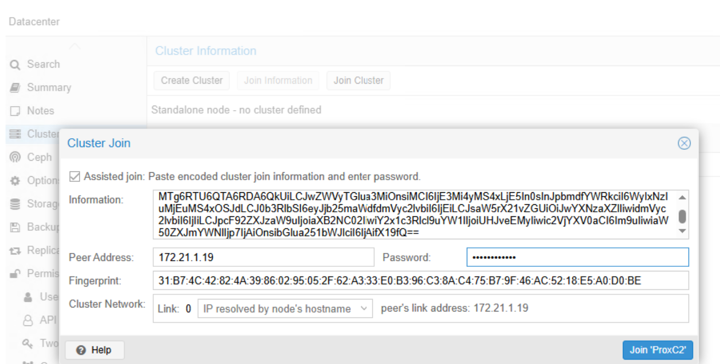
You will get a brief output, indicating the actions taken and the status of the request.



Before moving to the other nodes, we will select the “Join Information” button and copy the information.



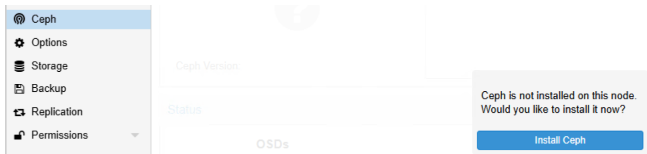
At each web interface for the other nodes, we will select Datacenter -> Cluster -> Join Cluster and input the join information into the window that appears. Be sure to verify each field.



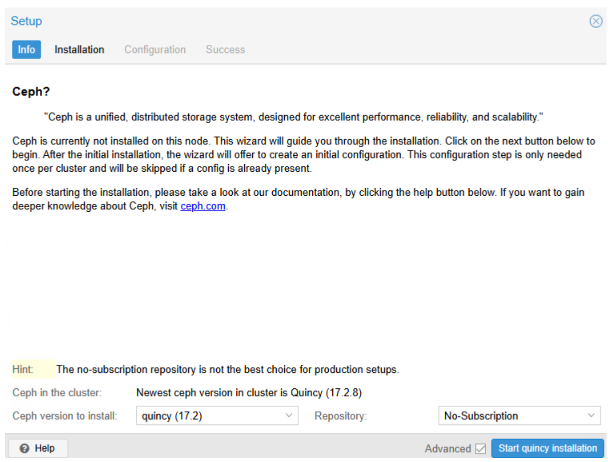
5.5 Configure Ceph Storage via Proxmox VE Web Interface

Next, we will set up Ceph again, using the Web UI this time. From Datacenter -> Ceph, you will be asked to install it, followed by a screen where you can select the version to install, as well as the repository.

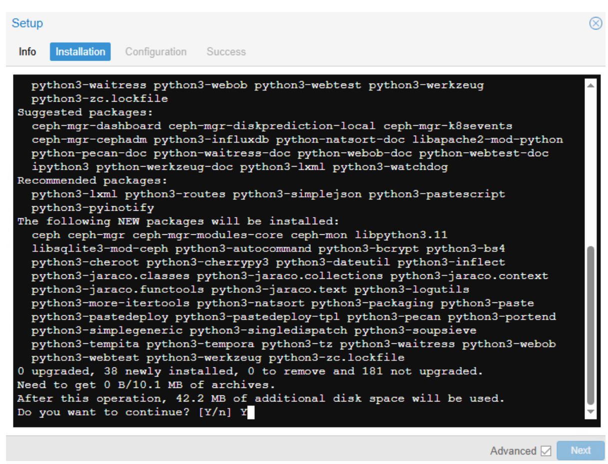
Install screen:



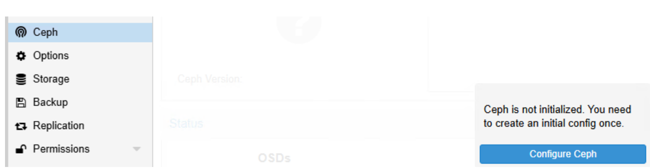
As we did earlier, for the repository, we will select “No Subscription”. We will also remain with Ceph 17.2 (quincy).



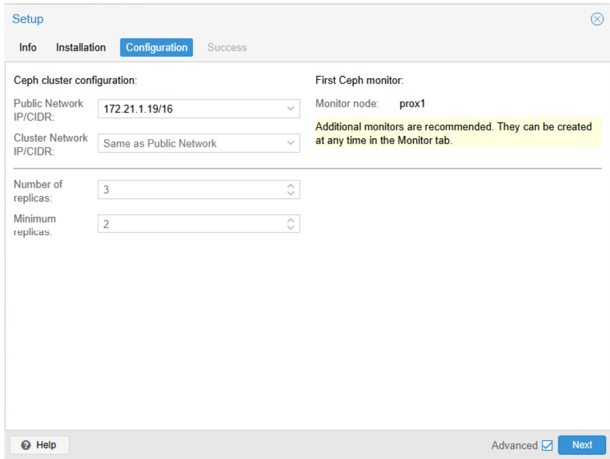
You’ll be presented with a terminal screen where you must accept the package installation.



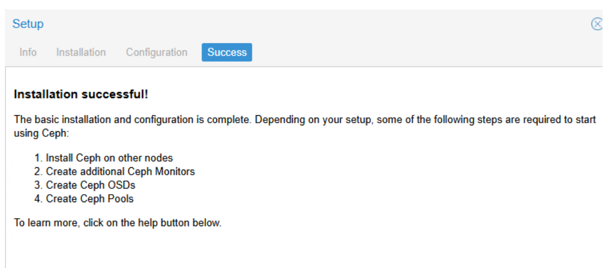
Once that process is completed, selecting Next takes you to the configuration.



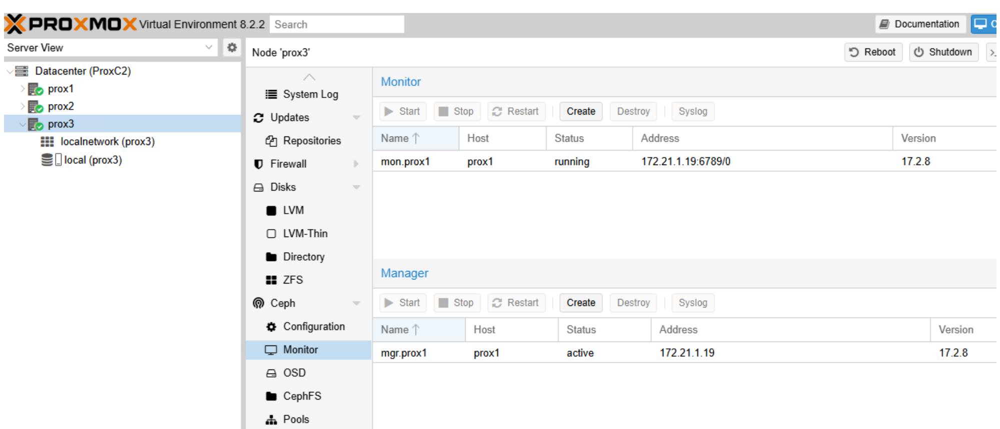
On the configuration screen, the Public Network should have the network already assigned to the node, from the Proxmox VE cluster configuration, in the drop-down list. If not, enter manually. If you have a specific desire regarding the replica quantity, ensure the “Advanced” option at the bottom is selected, and configure accordingly.



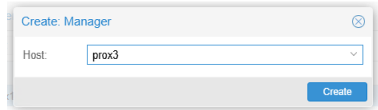
Upon completion, you’ll get this message, indicating the next steps. We will repeat the installation and configuration process on the remaining two nodes.



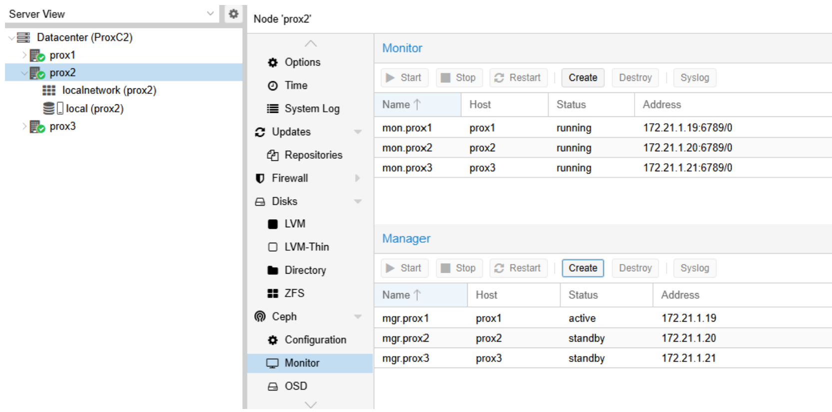
To configure the additional nodes as monitors and managers, we will need to select the node in the Datacenter view and navigate to the Ceph section:



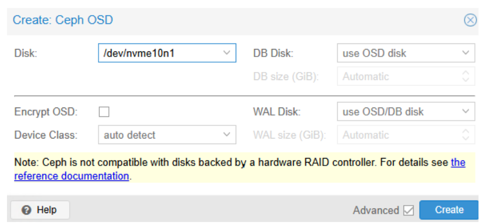
Select “Create” in both sections and add the node. Repeat for each node.



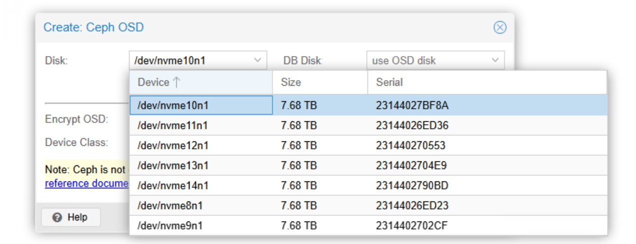
When complete, you will see all the nodes in the Monitor and Manager list.



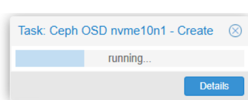
Next, navigate down to OSD and select “Create” at the top after you have first wiped the individual disks that will be used for the OSD. You will get a screen link the following, where the Disk drop-down will list the unconfigured disks available on that system.



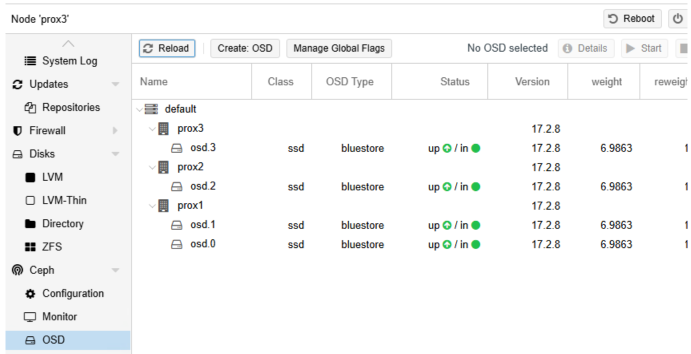
For each disk desired to participate as a member of the OSD pool, repeat the process.



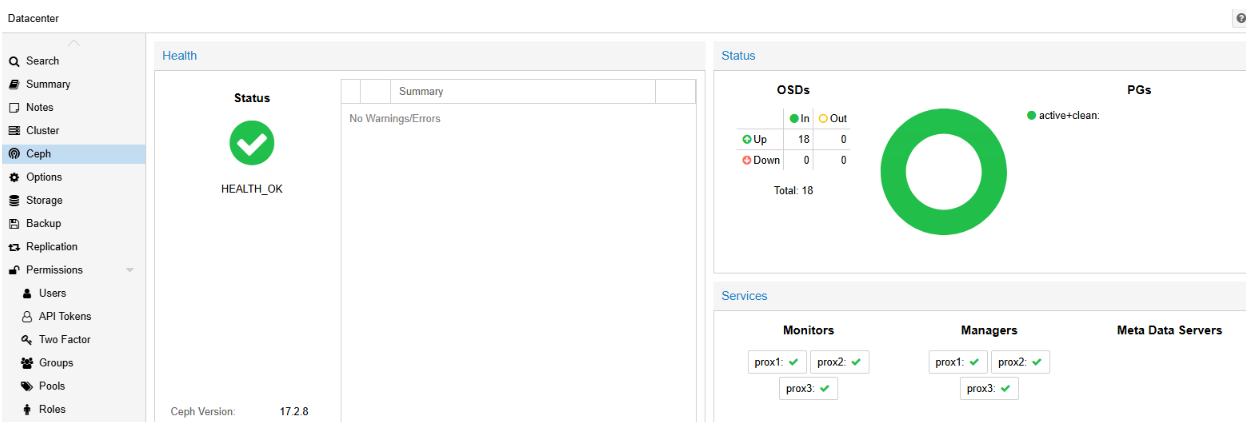
An image of the execution:



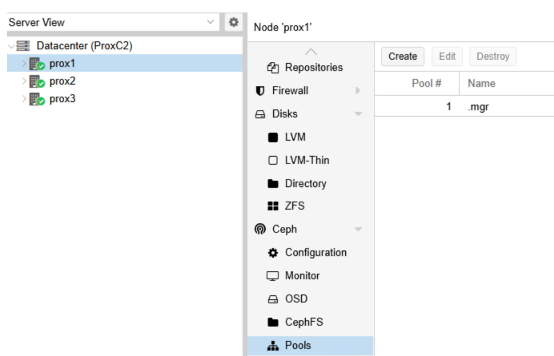
As the OSDs are created, you can refresh to see the current list:



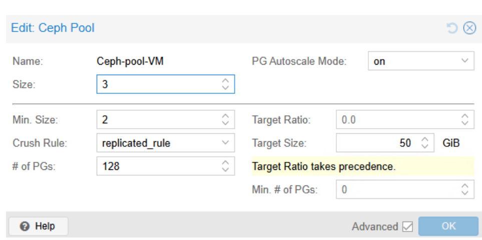
Once the desired disks are configured as OSDs, we can go back to Datacenter -> Ceph and check the overall health of the storage pool.



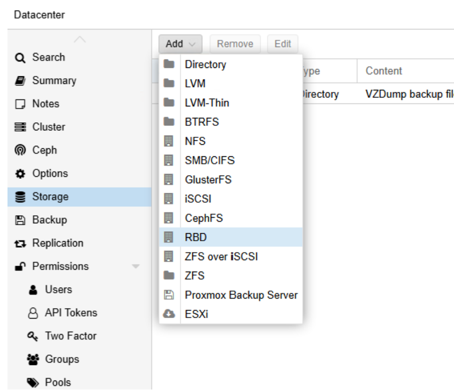
We will now create a pool from the configured OSDs, at <any node> -> Ceph -> Pools.



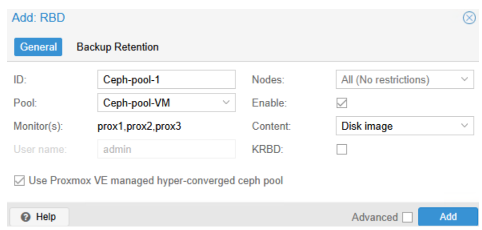
Here, we will accept defaults but set the target size to 50GiB.



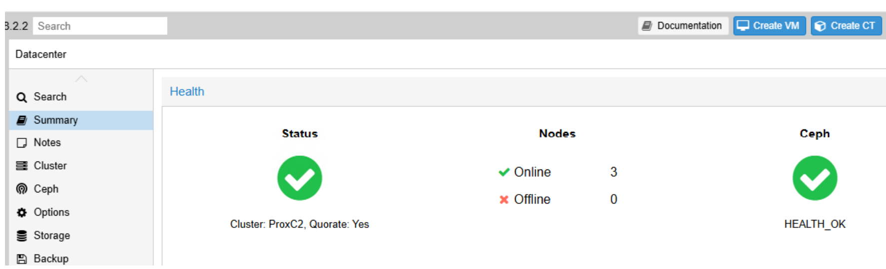
Next, we will configure the RBD storage for VMs, by navigating to Datacenter -> Storage and selecting Add -> RBD.



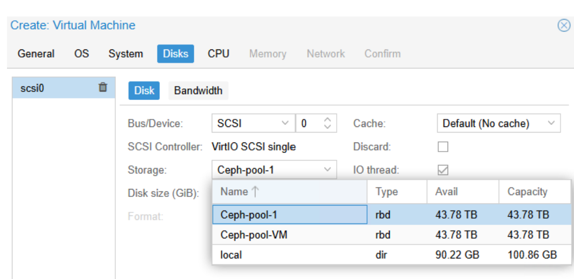
On the following screen, name the ID according to your preference. This step connects the storage pool to Proxmox VE's storage abstraction, like a datastore in ESXi or a storage container in Nutanix Prism.



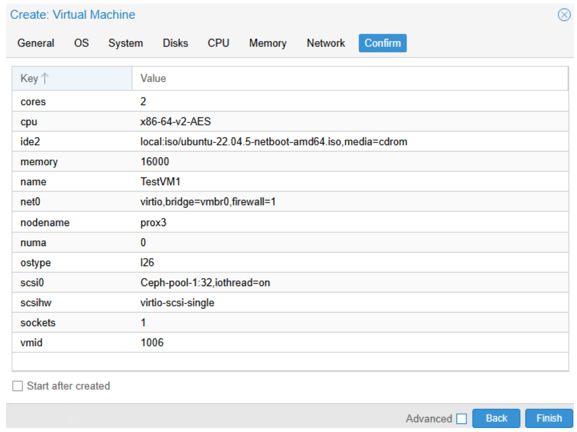
Lastly, we will create a VM on the newly created RBD store. At the upper-right hand side of the Web UI, you'll see options to create a VM or Container. Select the VM options.



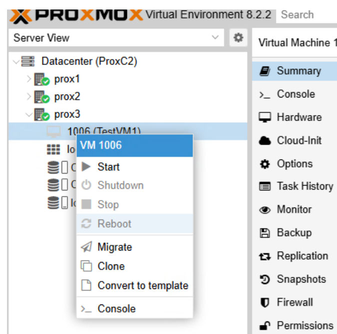
There are several screens to walk through. We will only include a few key pieces of the process. In this scenario, we're using the ubuntu-22.04.5 netboot image. In the case of the storage, we will select an RBD pool from the list.



Finally, we have the confirmation screen and we will select finish.



Next, we will navigate to Prox3, where we created the VM, and start it.



At this stage, you should be able to open a terminal and begin using your VM.

September 2025