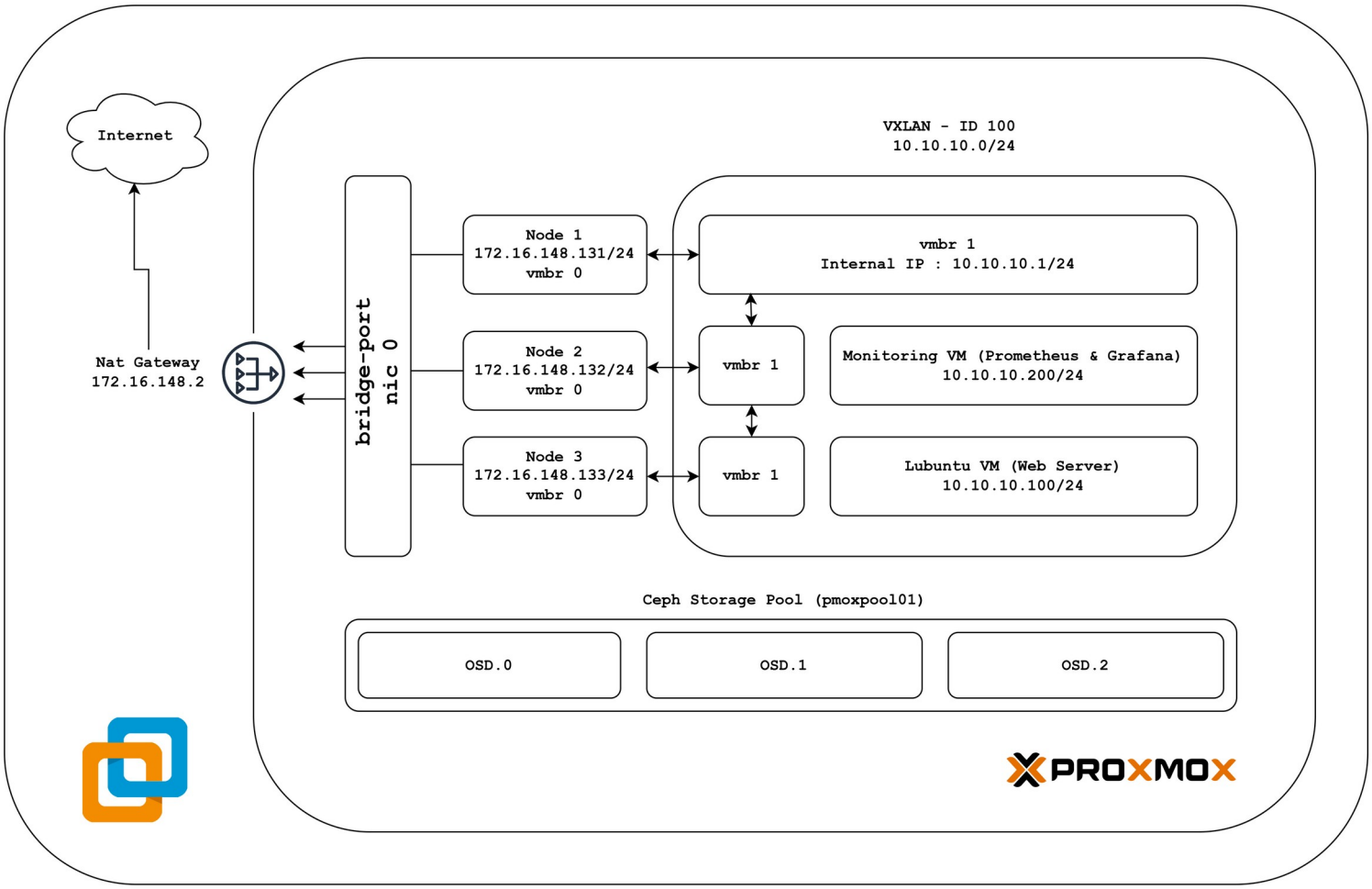
Design and Implementation of a High-Availability Proxmox Ve 3-Nodes Cluster

Ceph, VXLAN Isolation and Prometheus/Grafana Monitoring



Aymane Elouafi

IRIC2 – Ensa Khouribga

April 15, 2026

Introduction :



In the modern IT landscape, the demand for scalable, resilient, and highly available infrastructure has made hyper-converged environments the industry standard. Organizations increasingly rely on virtualization, distributed storage, and software-defined networking (SDN) to build robust private clouds that can sustain hardware failures and isolate tenant workloads securely.

This project aims to design, implement, and validate a complete, high-availability (HA) virtualized infrastructure from the ground up. The core of this environment is built upon a three-node **Proxmox Virtual Environment (PVE)** cluster, which provides the enterprise-grade hypervisor layer. To achieve true high availability and enable seamless virtual machine migration, **Ceph** was integrated to provide a distributed, software-defined storage backend, eliminating any single point of failure at the storage level.

Furthermore, a modern infrastructure requires strict network isolation and deep observability. To address this, a **VXLAN (Virtual Extensible LAN)** overlay network was deployed to stretch a secure, isolated Layer 2 network across the Layer 3 cluster nodes. Finally, a centralized monitoring stack utilizing **Prometheus and Grafana** was implemented to provide real-time visibility into the health, performance, and network utilization of the virtualized workloads.

Report Outline :

- Key Terminology & Concepts
- The Core Hypervisor: Proxmox Virtual Environment (PVE)
- Infrastructure Architecture and Deployment
- Storage Architecture: Distributed Software-Defined Storage via Ceph
- Logical Overlay Network: Software-Defined Networking via VXLAN
- Services Deployed & Monitoring
- High Availability and Fault Tolerance Validation
- Conclusion

Key Terminology & Concepts :

- **Virtualization**: The process of abstracting physical hardware resources—such as compute, network, and storage—to create multiple, independent software-based environments that can run concurrently on a single physical host.

- **Hypervisor**: The specialized software layer (also known as a Virtual Machine Monitor) that enables virtualization. It sits between the physical hardware and the virtual environments, pooling the physical resources and dynamically allocating them to the virtual machines.

- **Virtual Machine (VM)**: An isolated, software-defined computer that operates independently within a virtualized environment. It possesses its own virtual hardware (CPU, memory, storage) and runs its own operating system and applications, completely unaware that it is sharing a physical host.

- **Software-Defined Networking (SDN)**: A network architecture approach that decouples the network control plane (how traffic is routed) from the data plane (how traffic is forwarded). This allows administrators to centrally program, manage, and secure network behavior through software interfaces rather than individual hardware configurations.

- **Software-Defined Storage (SDS)**: A storage architecture that separates the storage management software from the underlying physical storage hardware. This enables the pooling of diverse storage resources into a unified, highly scalable system that can be dynamically provisioned as needed.

- **VLAN (Virtual Local Area Network)**: A logical network configuration that groups a subset of devices on a physical switch into their own isolated broadcast domain. It is used to segment network traffic for improved performance, organization, and security without requiring separate physical cables and switches.

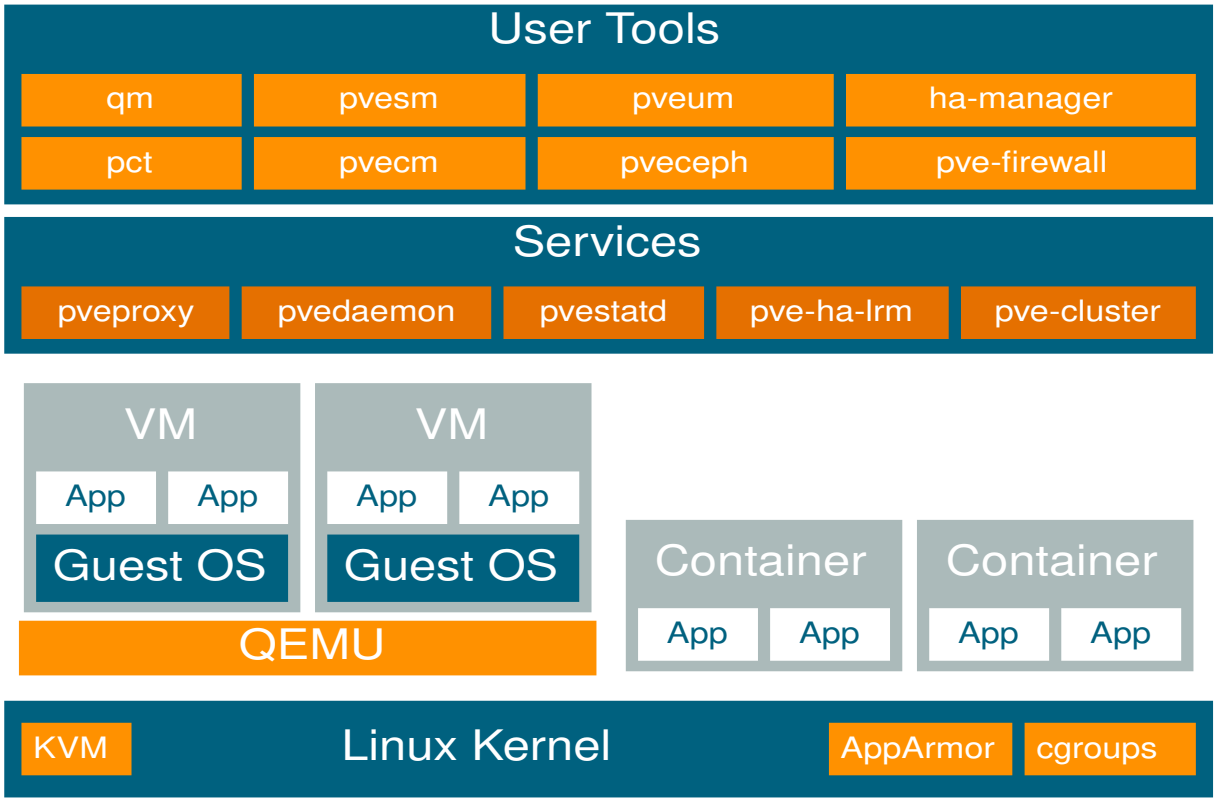
- **VXLAN (Virtual Extensible LAN):** An advanced network virtualization technology designed to solve the scalability limits of traditional VLANs. It encapsulates Layer 2 Ethernet frames within Layer 3 UDP packets, allowing for the creation of up to 16 million isolated logical networks across complex cloud and datacenter infrastructures.

- **Shared Storage:** A storage architecture where multiple independent servers or host machines are connected to and can access the same centralized pool of storage data simultaneously. This is a fundamental requirement for advanced virtualized operations like live VM migration and high availability.

- **Ceph:** A highly scalable, open-source software-defined storage platform that delivers object, block, and file storage from a single distributed cluster. It is designed to be self-healing and self-managing with no single point of failure.

- **High Availability (HA):** A system design and operational protocol that ensures a high level of operational performance and continuous uptime for an agreed-upon period. In virtualization, HA relies on clustering and shared storage to automatically restart virtual machines on functional hosts if their original physical host fails.

The Core Hypervisor: Proxmox Virtual Environment (PVE)



Before detailing the physical and logical topologies, it is essential to define the foundational software layer of this infrastructure: **Proxmox Virtual Environment (PVE)**.

Proxmox is a complete, open-source enterprise server virtualization management platform. Built on top of a Debian GNU/Linux distribution, it acts as a Bare-Metal (Type 1) hypervisor, providing the compute, network, and storage management capabilities required to build a modern datacenter.

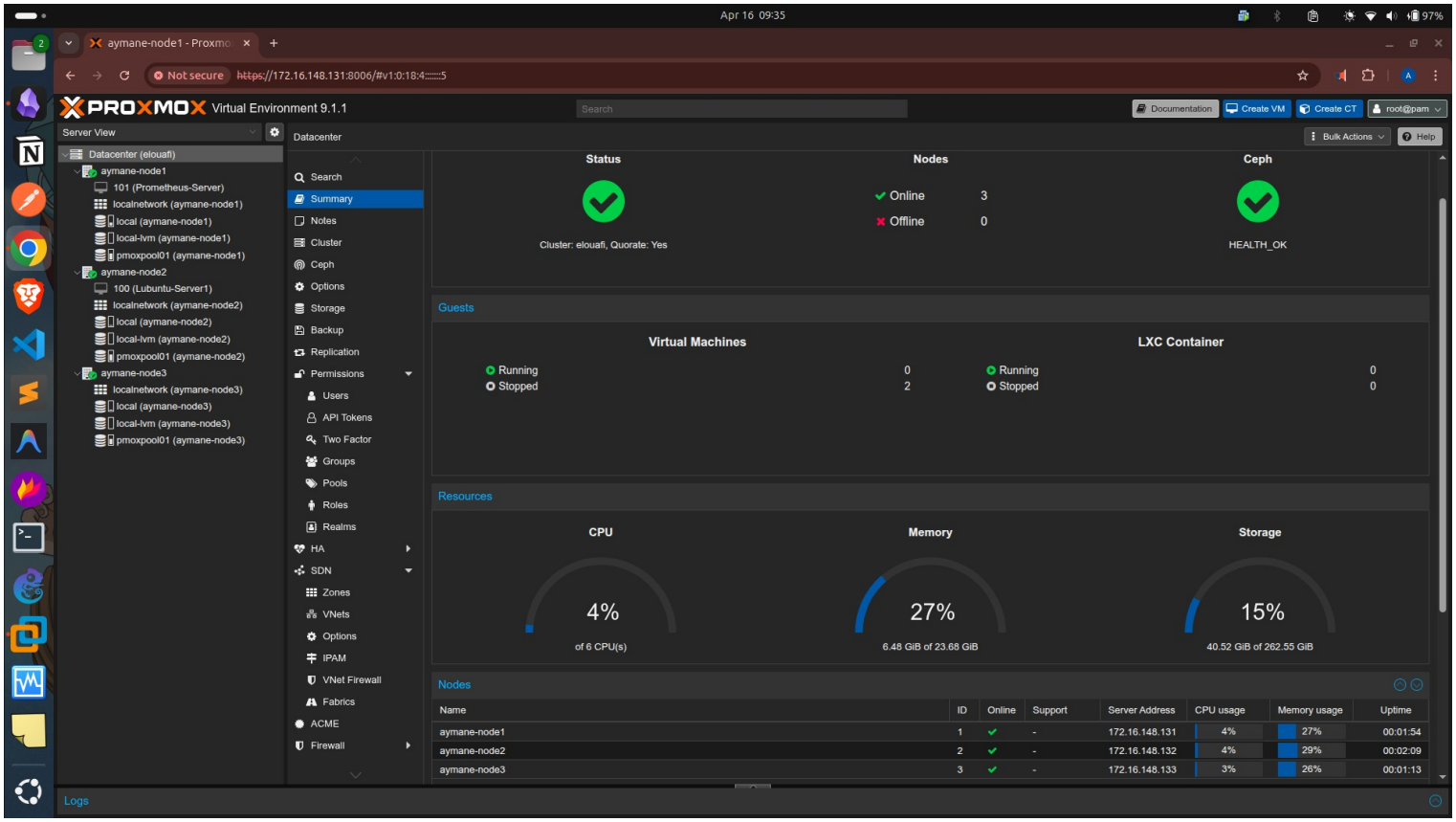
Unlike proprietary solutions that rely on separate management servers, Proxmox tightly integrates its management layer directly into each node via a unified web interface and REST API. Technically, Proxmox leverages a custom Linux kernel to support two distinct virtualization technologies simultaneously:

1. **KVM (Kernel-based Virtual Machine):** A full virtualization solution for Linux on x86 hardware containing virtualization extensions. In this project, KVM is utilized to run fully isolated, complete Virtual Machines (such as the Web Server and Monitoring VM) with their own dedicated virtual hardware.
2. **LXC (Linux Containers):** An operating-system-level virtualization method for running multiple isolated Linux

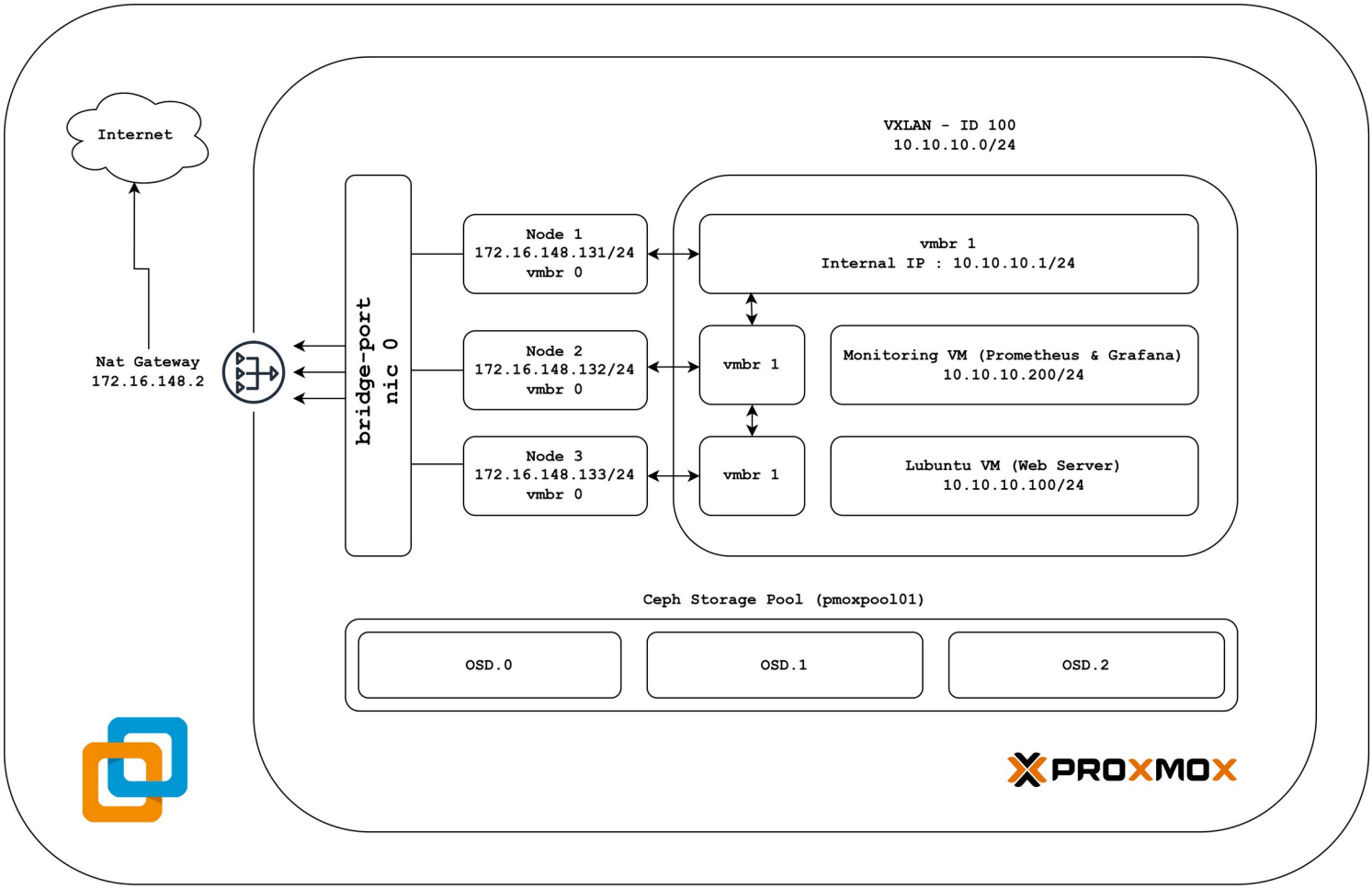
systems (containers) on a single control host. *(While supported, this project primarily focuses on KVM for strict tenant isolation).*

Why Proxmox for this Architecture?

Proxmox was explicitly chosen for this hyper-converged infrastructure project because of its native, out-of-the-box support for clustering and advanced software-defined technologies. Instead of requiring external plugins or expensive licenses, Proxmox natively integrates the **Corosync** cluster engine, **Ceph** distributed storage, and advanced Linux networking (including **VXLAN** and SDN capabilities). This native integration is what makes the High Availability (HA) and live migration features of this project possible.



Infrastructure Architecture :



The foundation of this highly available infrastructure is built upon a nested virtualization environment hosted within VMware Workstation. This "underlay" network simulates a traditional physical datacenter fabric, providing base Layer 3 connectivity, management access, and internet routing for the hypervisors.

The Underlay Fabric: The physical nodes are connected to a NAT-configured virtual network operating on the 172.16.148.0/24 subnet. Within this fabric, the upstream gateway and primary DNS resolution are provided by the VMware NAT router located at 172.16.148.2. This gateway is critical for allowing the nodes and nested workloads to fetch updates and communicate with external services (such as Grafana template repositories).

The Compute Nodes: The compute layer consists of a three-node Proxmox Virtual Environment (PVE) cluster. Each node utilizes a primary physical network interface (nic0) that is bound to the

external Linux bridge (vmbr0). This bridge acts as the management and public-facing switch for the nodes.

The cluster is composed of the following nodes:

- **aymane-node1 (Cluster Gateway):** This node acts as the primary ingress and egress point for the cluster's internal isolated networks. Its external vmbr0 interface is statically assigned the IP address **172.16.148.131/24**.
- **aymane-node2 (Cluster Member):** Operates as a standard compute and distributed storage node. Its external vmbr0 interface is assigned the IP address **172.16.148.132/24**.
- **aymane-node3 (Cluster Member):** Operates identically to node 2, providing the third point of failure tolerance for the quorum. Its external vmbr0 interface is assigned the IP address **172.16.148.133/24**.

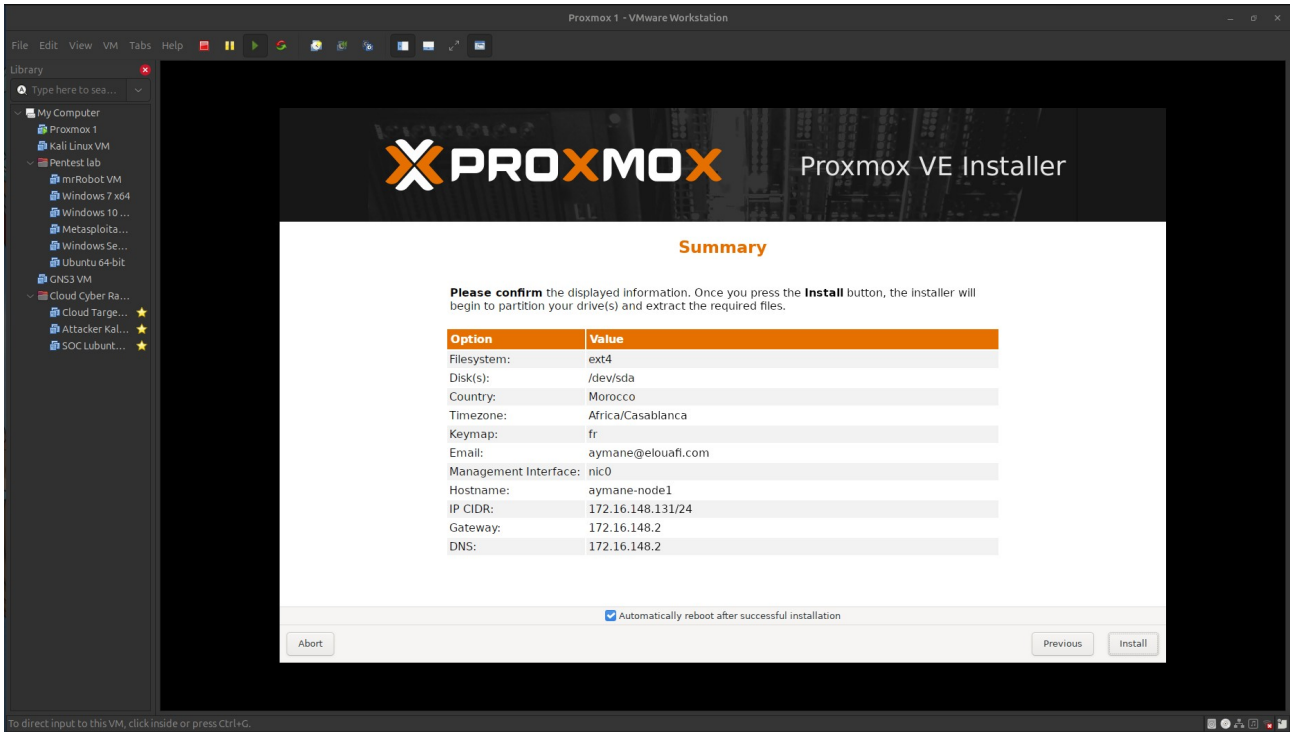
This baseline network configuration ensures that all three nodes have reliable, low-latency communication with one another. This underlying stability is mandatory for establishing the Proxmox Corosync cluster heartbeat and synchronizing the Ceph distributed storage traffic before any overlay networks are introduced.

Implementation: Proxmox Installation and Cluster Configuration

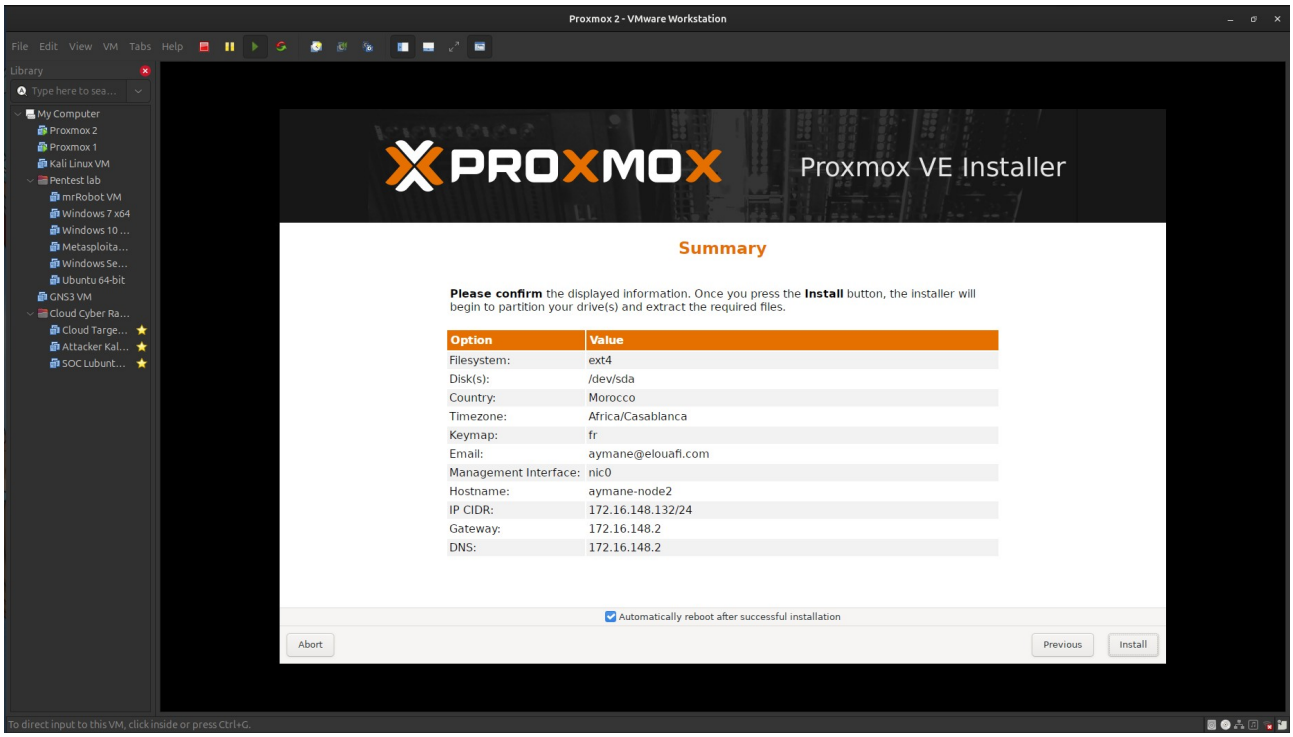
Base Installation and Network Initialization

The Proxmox VE ISO was installed on all three nodes.During the installation phase, the primary network interfaces were configured to align with the underlay architecture.

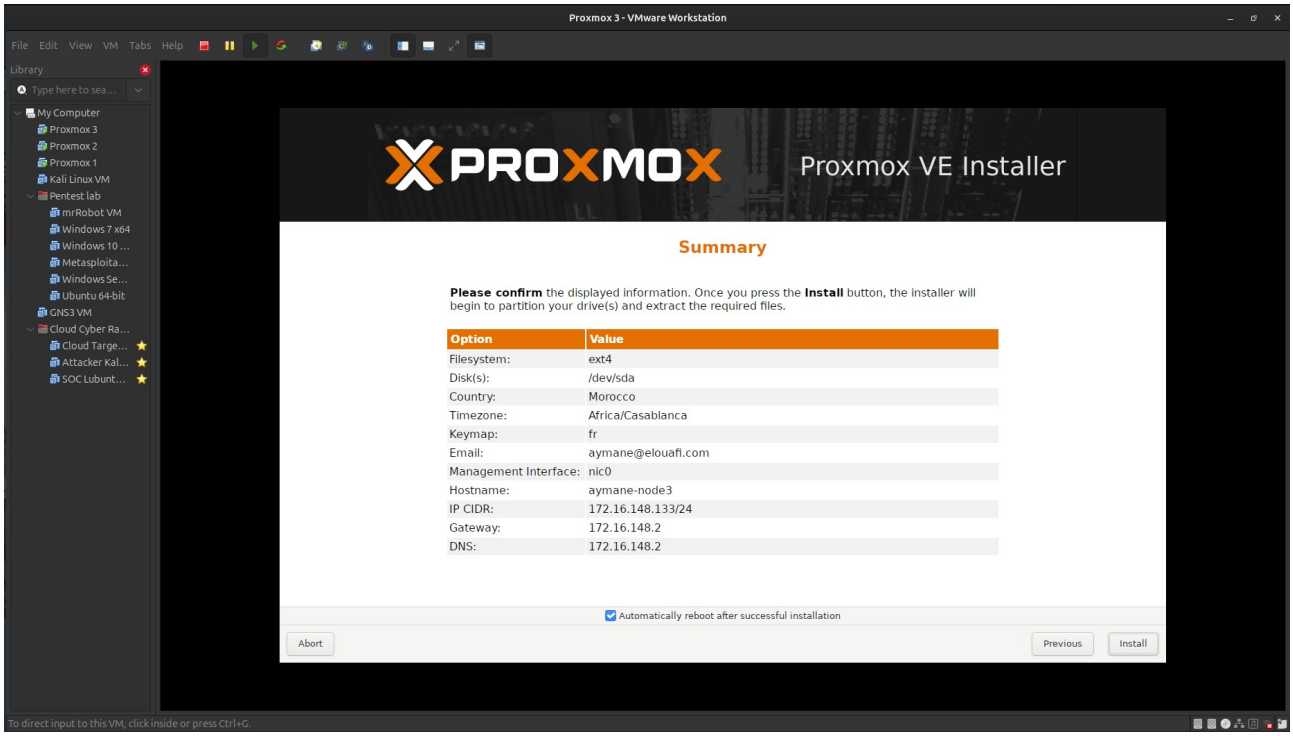
- Node 1 :


Proxmox 1 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
Proxmox 1
Kali Linux VM
Pentest lab
mrRobot VM
Windows 7 x64
Windows 10 ...
Metasploita...
Windows Se...
Ubuntu 64-bit
GNS3 VM
Cloud Cyber Ra...
Cloud Targe...
Attacker Kal...
SOC Lubunt...
PROXMOX
Proxmox VE Installer
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: Morocco
Timezone: Africa/Casablanca
Keymap: fr
Email: aymane@elouafi.com
Management Interface: nic0
Hostname: aymane-node1
IP CIDR: 172.16.148.131/24
Gateway: 172.16.148.2
DNS: 172.16.148.2
Automatically reboot after successful installation
Abort
Previous
Install
To direct input to this VM, click inside or press Ctrl+G.

- Node 2 :

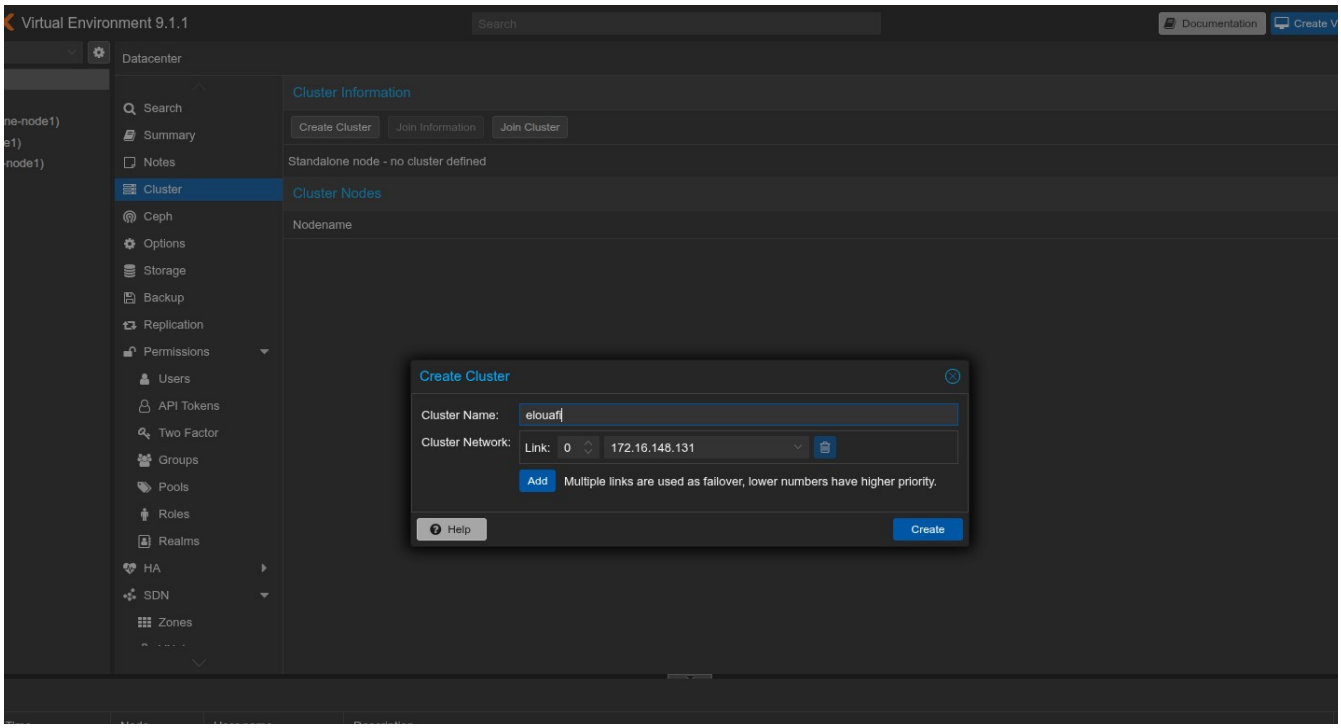

Proxmox 2 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
Proxmox 2
Proxmox 1
Kali Linux VM
Pentest lab
mrRobot VM
Windows 7 x64
Windows 10 ...
Metasploita...
Windows Se...
Ubuntu 64-bit
GNS3 VM
Cloud Cyber Ra...
Cloud Targe...
Attacker Kal...
SOC Lubunt...
PROXMOX
Proxmox VE Installer
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: Morocco
Timezone: Africa/Casablanca
Keymap: fr
Email: aymane@elouafi.com
Management Interface: nic0
Hostname: aymane-node2
IP CIDR: 172.16.148.132/24
Gateway: 172.16.148.2
DNS: 172.16.148.2
Automatically reboot after successful installation
Abort
Previous
Install
To direct input to this VM, click inside or press Ctrl+G.

- Node 3 :

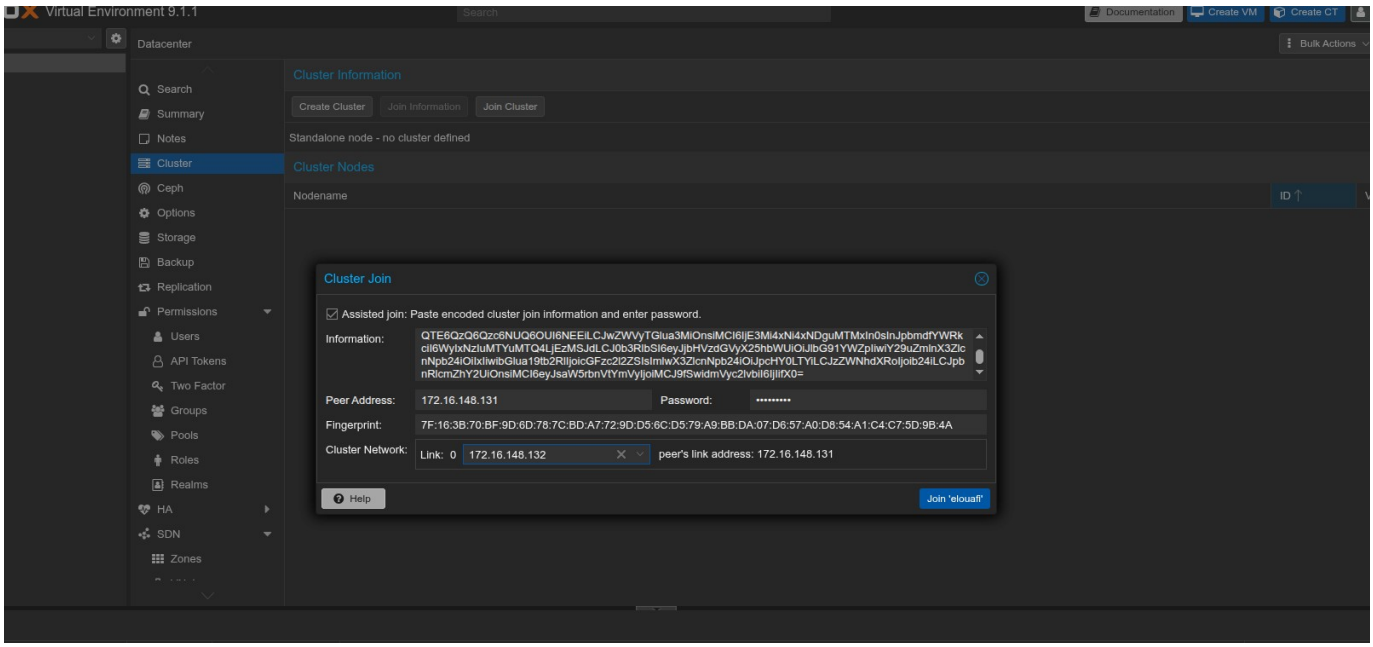

Proxmox 3 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
Proxmox 3
Proxmox 2
Proxmox 1
Kali Linux VM
Pentest lab
mrRobot VM
Windows 7 x64
Windows 10 ...
Metasploita...
Windows Se...
Ubuntu 64-bit
GNS3 VM
Cloud Cyber Ra...
Cloud Targe...
Attacker Kal...
SOC Lubunt...
To direct input to this VM, click inside or press Ctrl+G.
PROXMOX
Proxmox VE Installer
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:
Morocco
Timezone:
Africa/Casablanca
Keymap:
fr
Email:
aymane@elouafi.com
Management Interface:
nic0
Hostname:
aymane-node3
IP CIDR:
172.16.148.133/24
Gateway:
172.16.148.2
DNS:
172.16.148.2
Automatically reboot after successful installation
Abort
Previous
Install

Cluster Creation :

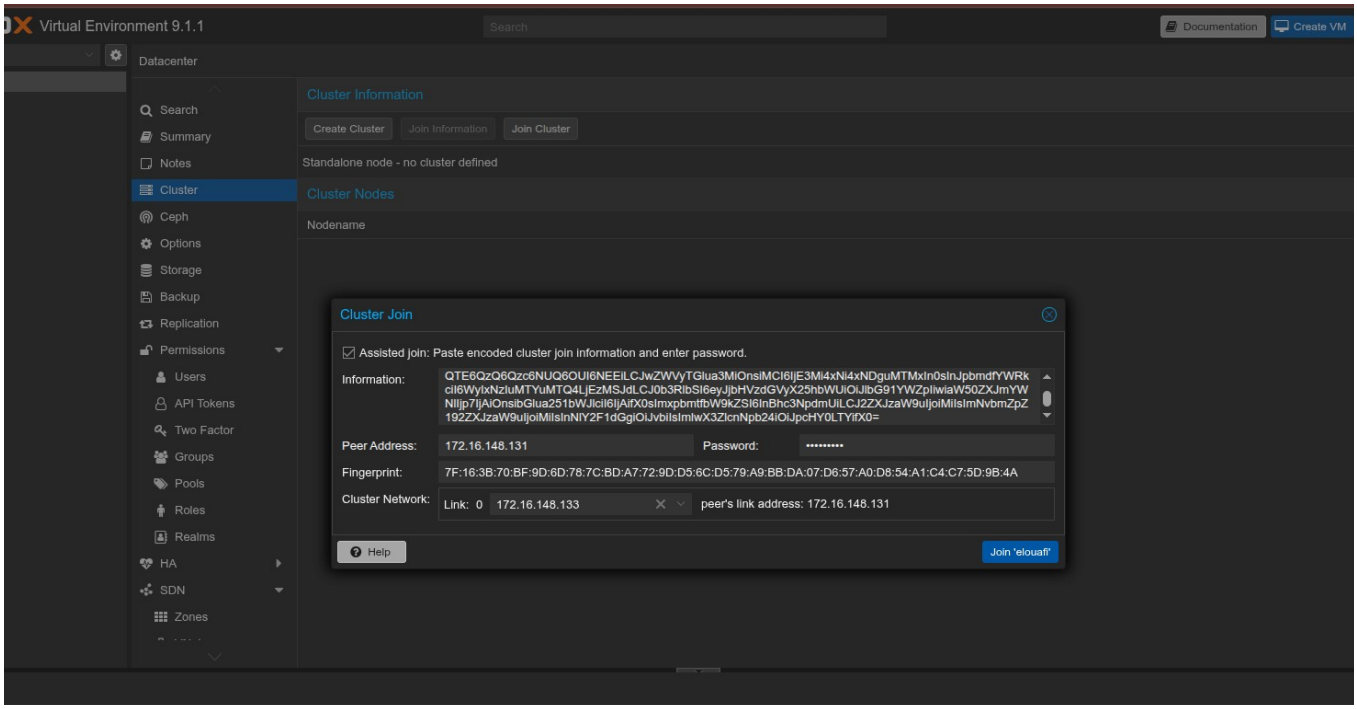
With the base networking established, the independent nodes were joined into a unified High-Availability cluster. Proxmox utilizes Corosync as its cluster engine to maintain state and quorum.

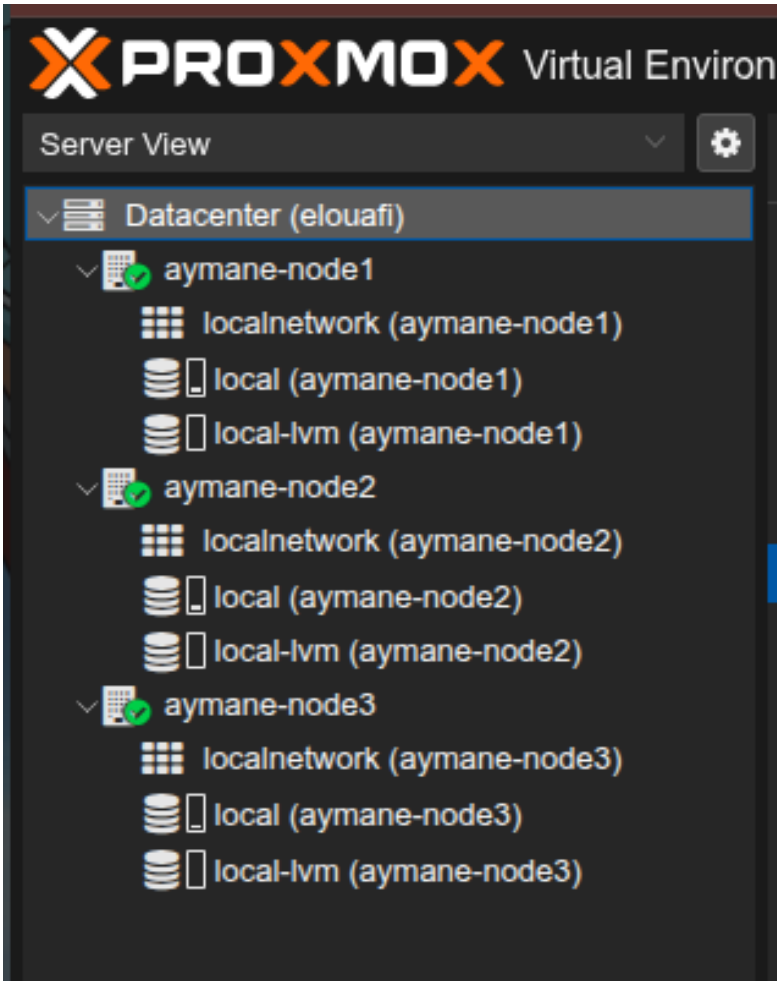


- Node 1 and Node 2 :

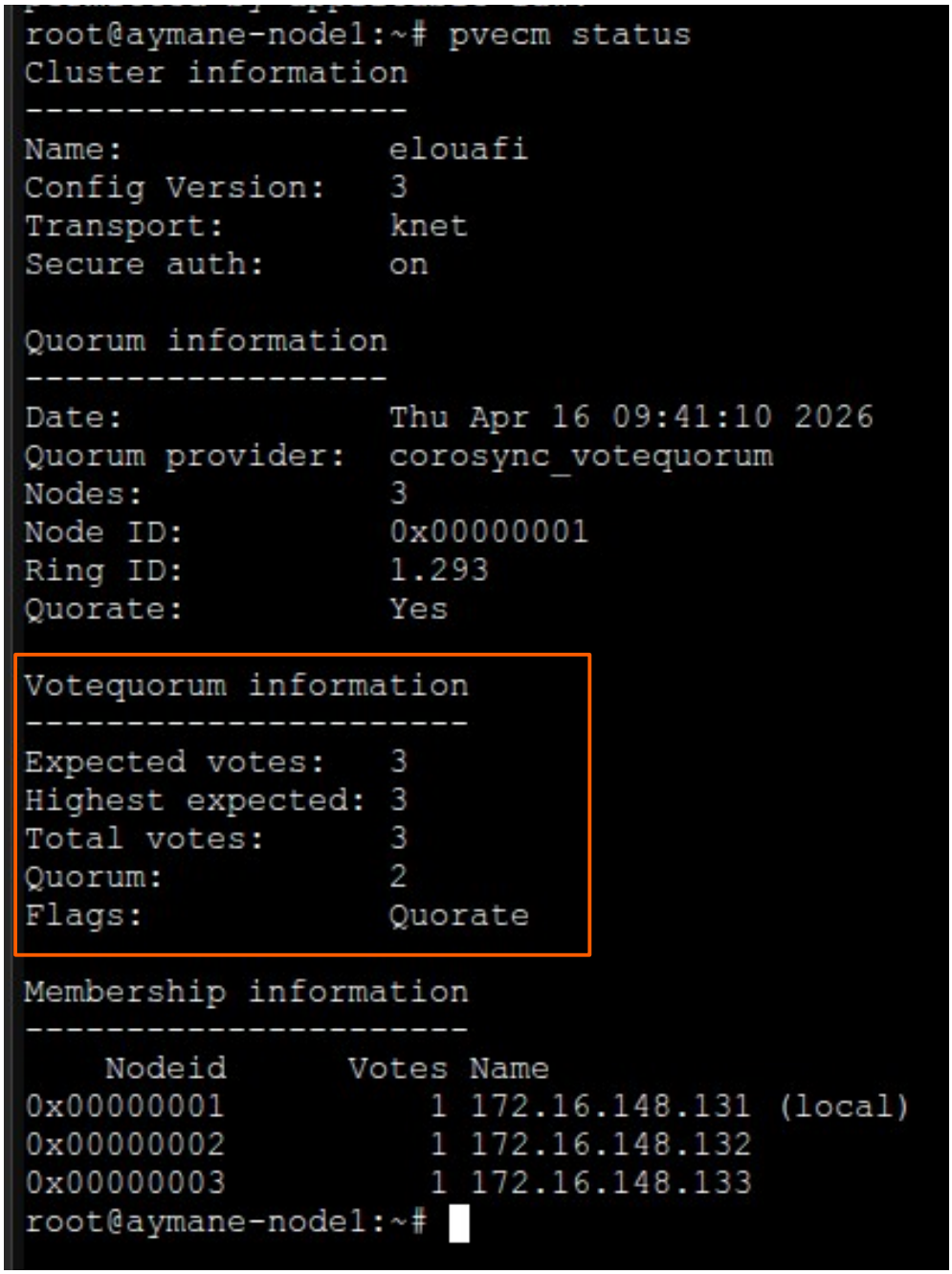


- Node 1 , Node 2 and Node 3 :



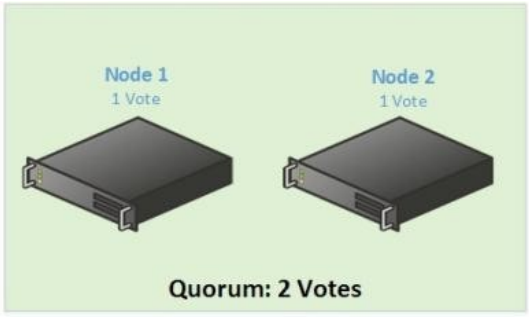


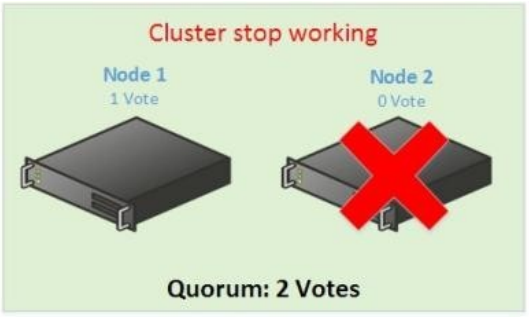
Quorum Validation :



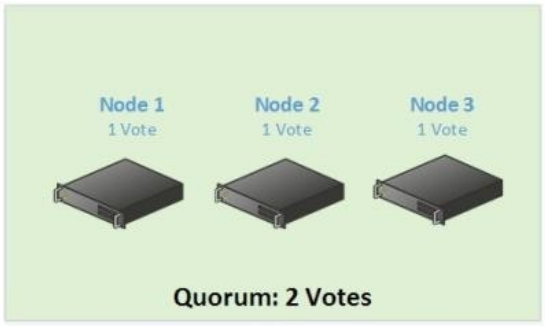
A 3-node architecture was deliberately chosen over a 2-node setup to satisfy the Corosync quorum requirement (N/2 + 1). In a 3-node cluster, a strict majority (2 votes) is required to make cluster-level decisions or execute High Availability failovers. This prevents "split-brain" scenarios where two disconnected nodes both attempt to take control of the same virtual machine.

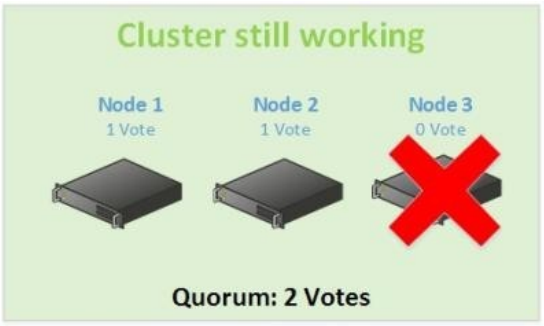
- Example of 2 nodes Cluster :

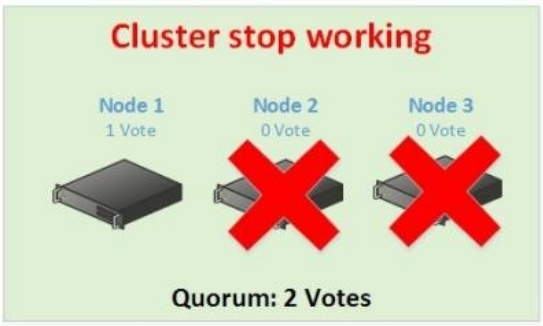




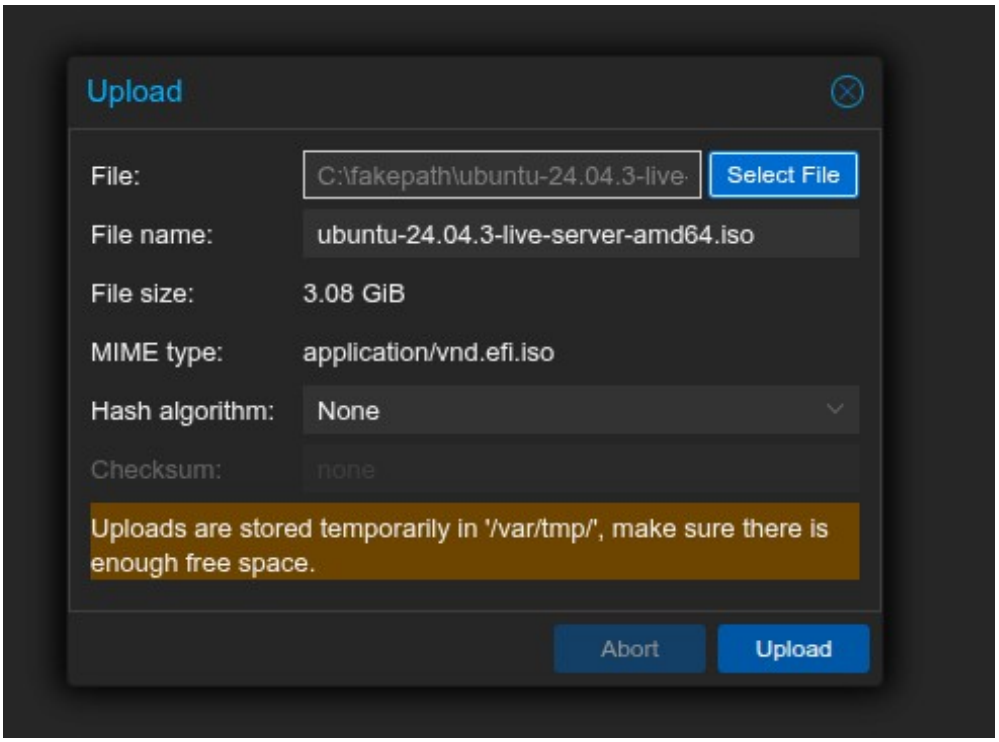
- Example of 3 nodes Cluster :

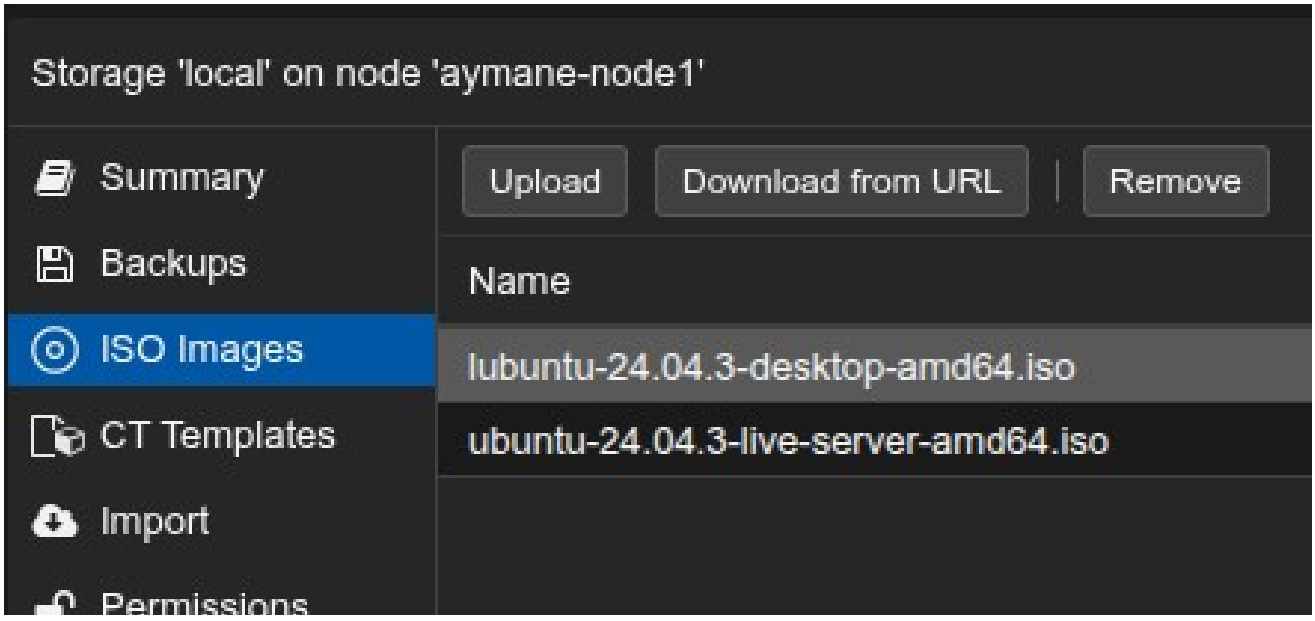




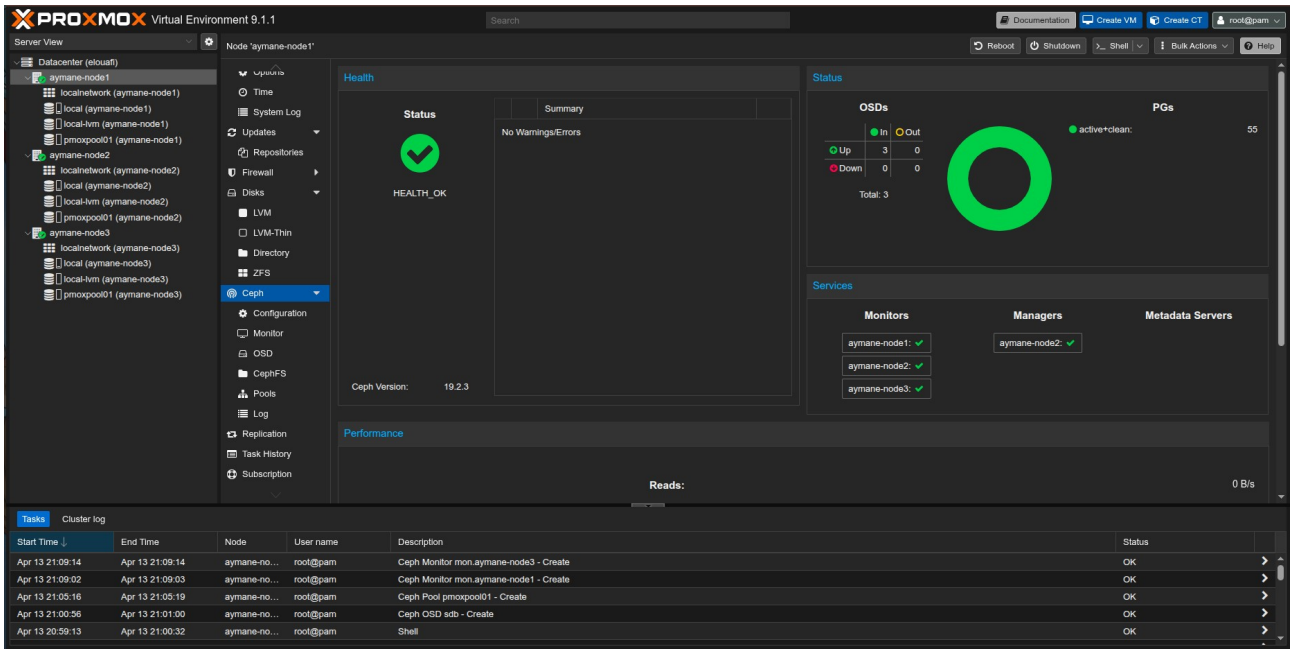


Upload ISO images to local Storage :





Storage Architecture: Distributed Software-Defined Storage via Ceph



In a traditional virtualization environment, Virtual Machines (VMs) are stored on local hypervisor disks or expensive centralized Storage Area Networks (SANs), creating single points of failure.To achieve true High Availability (HA) for this project, the infrastructure utilizes **Ceph**.

What is Ceph ?

Ceph is an enterprise-grade, open-source Software-Defined Storage (SDS) platform. Instead of relying on a centralized storage appliance, Ceph aggregates the local, direct-attached storage (hard drives or SSDs) of multiple independent servers into a single, massive, distributed storage cluster. It provides block, object, and file storage unified under one system.

Ceph relies on an intelligent algorithm called **CRUSH** (Controlled Replication Under Scalable Hashing). Rather than using a central lookup table to find where data is stored (which would be a bottleneck), the CRUSH algorithm allows every node to mathematically calculate exactly where data should be written or read, resulting in immense scalability and fault tolerance.

To function properly, Ceph relies on three primary daemon types deployed across the Proxmox cluster:

1. **OSD (Object Storage Daemon):** The workhorse of Ceph. OSDs interact directly with the physical storage drives, handling data read/write operations, peer-to-peer replication, and automatic data recovery if a disk fails.

2. **MON (Monitor):** Monitors maintain the "cluster map" and the overall health state of the infrastructure. They operate strictly on a quorum basis; in this 3-node architecture, at least two MONs must be online to prevent a "split-brain" scenario and maintain data integrity.
3. **MGR (Manager):** Runs alongside the monitors to track runtime metrics, cluster capacity, and system utilization.

Connecting the Cluster and Creating pmoxpool01 :

In this architecture, aymane-node1, aymane-node2, and aymane-node3 all act as hyper-converged nodes—meaning they provide both the CPU/RAM for the virtual machines and the physical disks for the storage cluster.

By deploying Ceph OSDs across the three nodes, their individual drives are fused into a logical storage fabric over the underlying network. Within this fabric, a logical storage pool named **pmoxpool01** was created to act as the backend for the virtual machines.

pmoxpool01 is configured with a **replica size of 3**. This is a critical engineering decision: every time a block of data is written to a VM's virtual hard drive, Ceph transparently and simultaneously writes a copy of that data to an OSD on Node 1, an OSD on Node 2, and an OSD on Node 3.

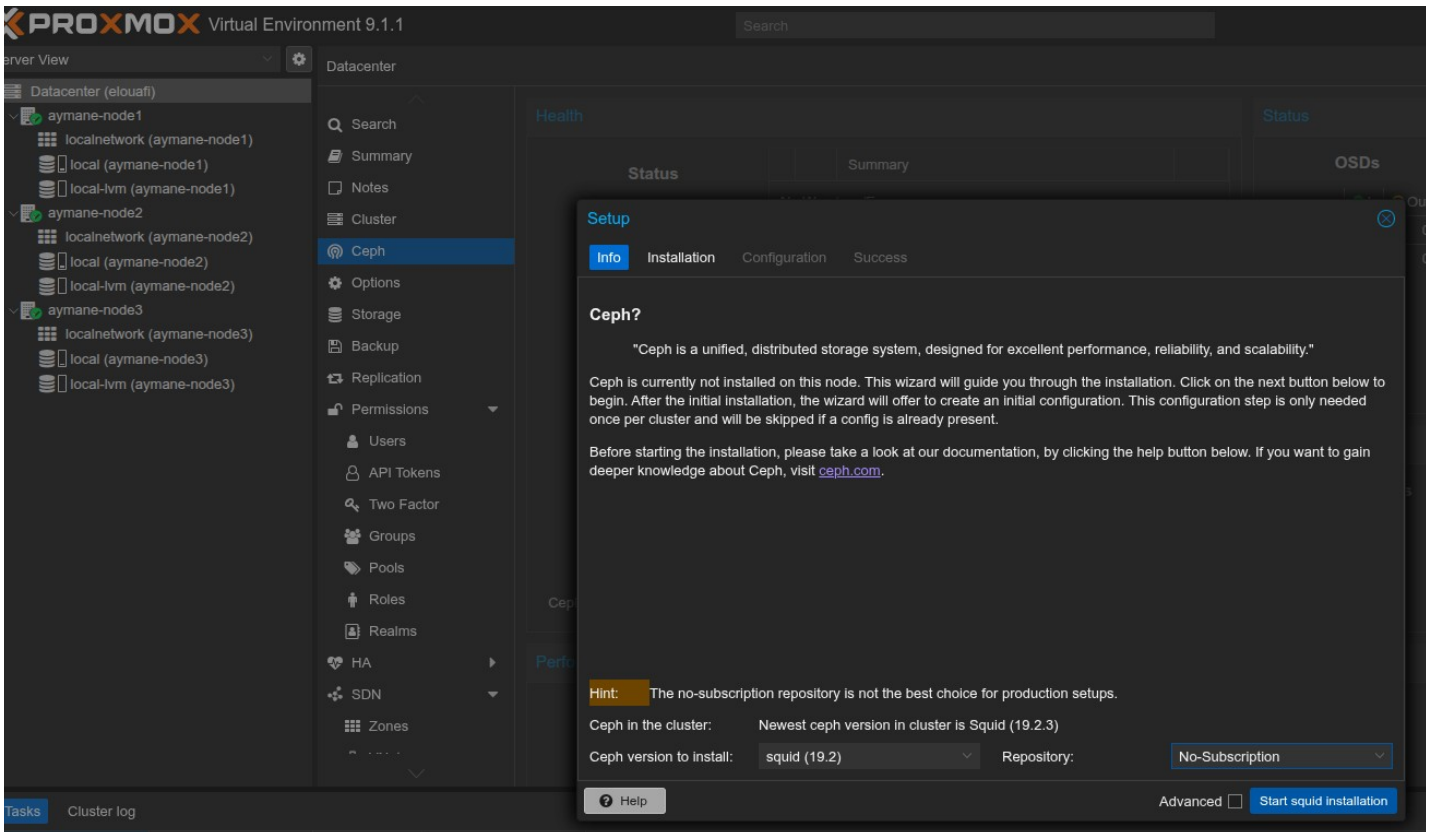
Enabling Redundancy and Live Migration :

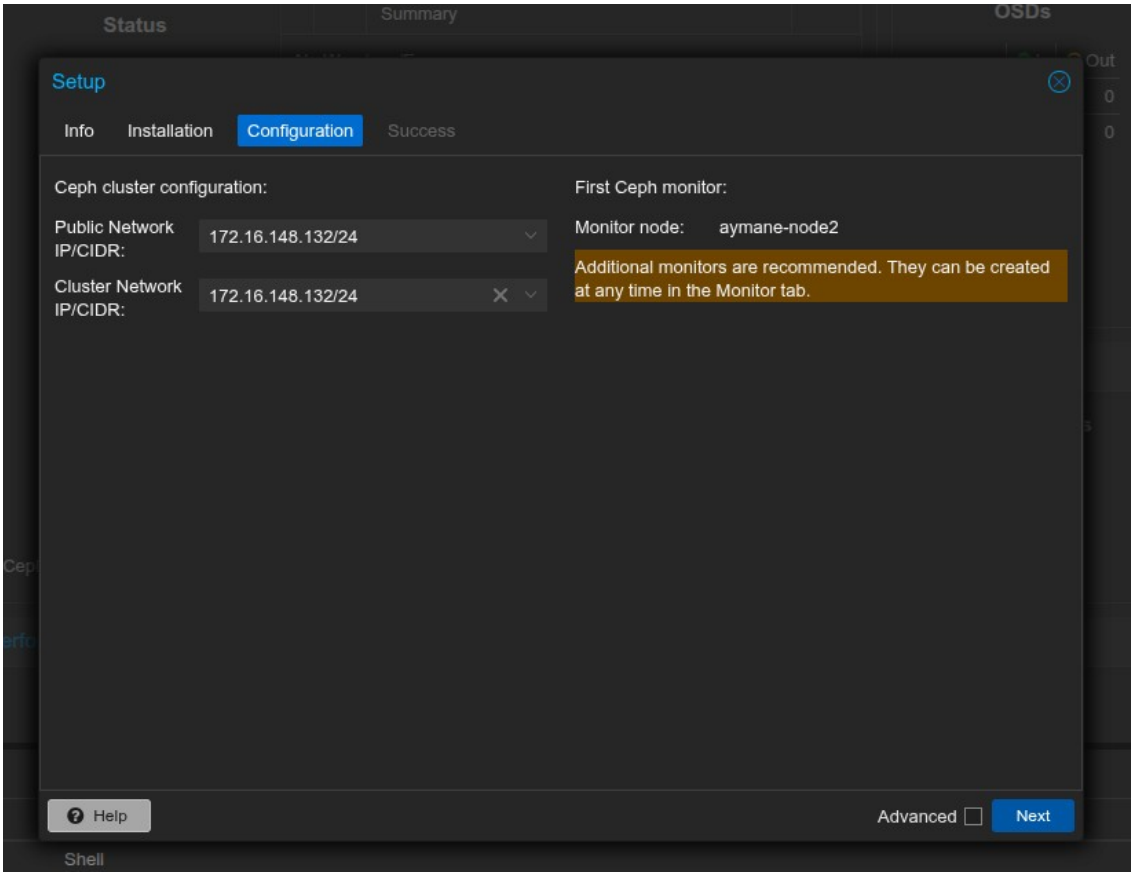
This distributed architecture provides two massive operational advantages:

1. **Fault Tolerance (VM Redundancy):** Because every block of data exists simultaneously on all three nodes, the cluster can survive the catastrophic failure of an entire physical node. If aymane-node2 suffers a hardware failure, no data is lost. The Proxmox High Availability manager can immediately restart aymane-node2's virtual machines on aymane-node1 or aymane-node3, pulling the identical data from the surviving OSDs.
2. **Zero-Downtime Live Migration:** In a standalone server, moving a VM to a new host requires copying gigabytes of disk data over the network, causing heavy downtime. Because pmoxpool01 is a *shared* pool, all three hypervisors have simultaneous access to the exact same virtual disks. Therefore, migrating a running VM (like the Web Server) from aymane-node1 to aymane-node3 only requires transferring the live RAM state across the network.The storage does not need to move, resulting in a live migration with zero perceived downtime for the application.

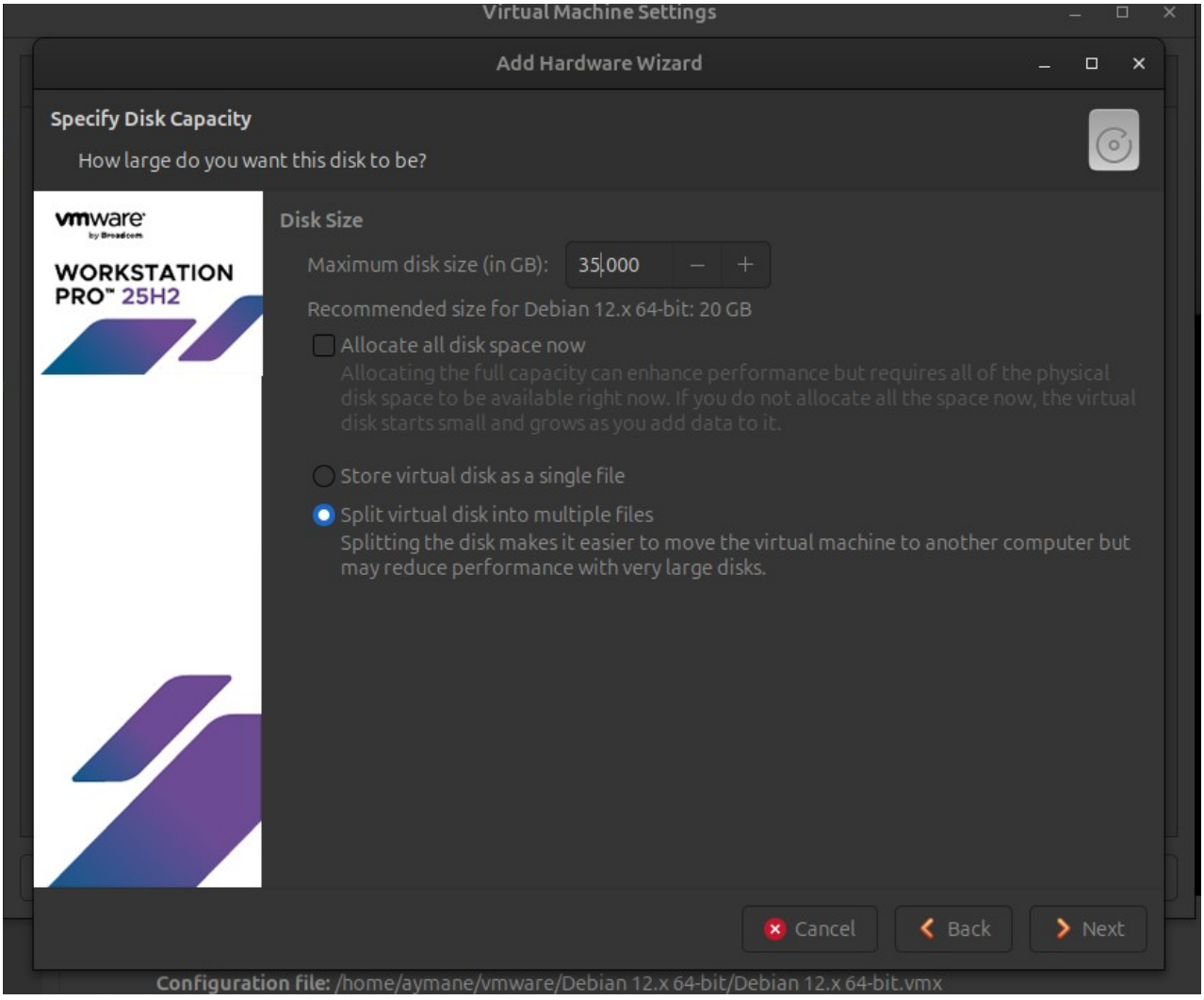
Implementation: Ceph Storage Cluster Configuration

Ceph Installation and Initialization in ALL the nodes :



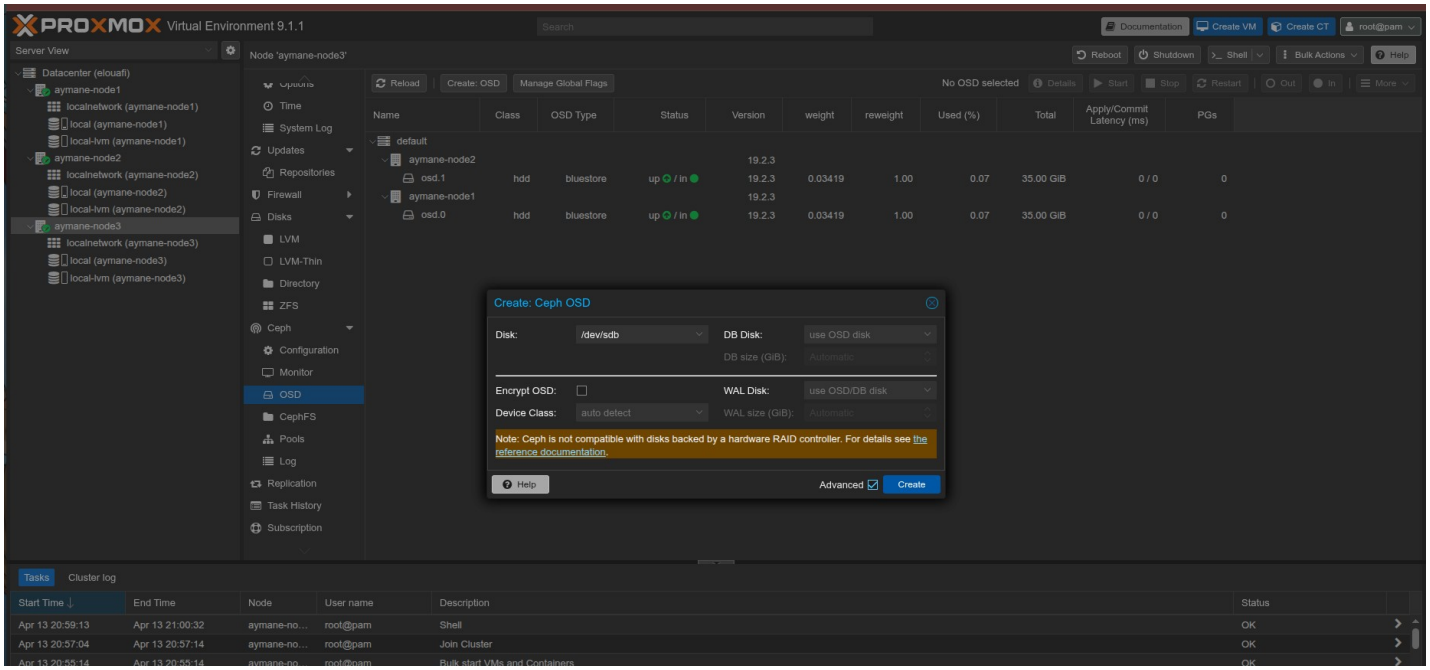


In each Proxmox VE VM we need to add a 35GB Hard Disk :



Provisioning Object Storage Daemons (OSDs) : Ceph requires dedicated, unformatted block devices (raw disks) to function. During the VMware provisioning phase, secondary virtual hard disks were attached to each Proxmox node specifically for storage purposes.

These raw disks were initialized and formatted as OSDs using the BlueStore backend (Ceph's default and most performant storage engine). This process was repeated on all three nodes to contribute their local capacity to the global cluster:



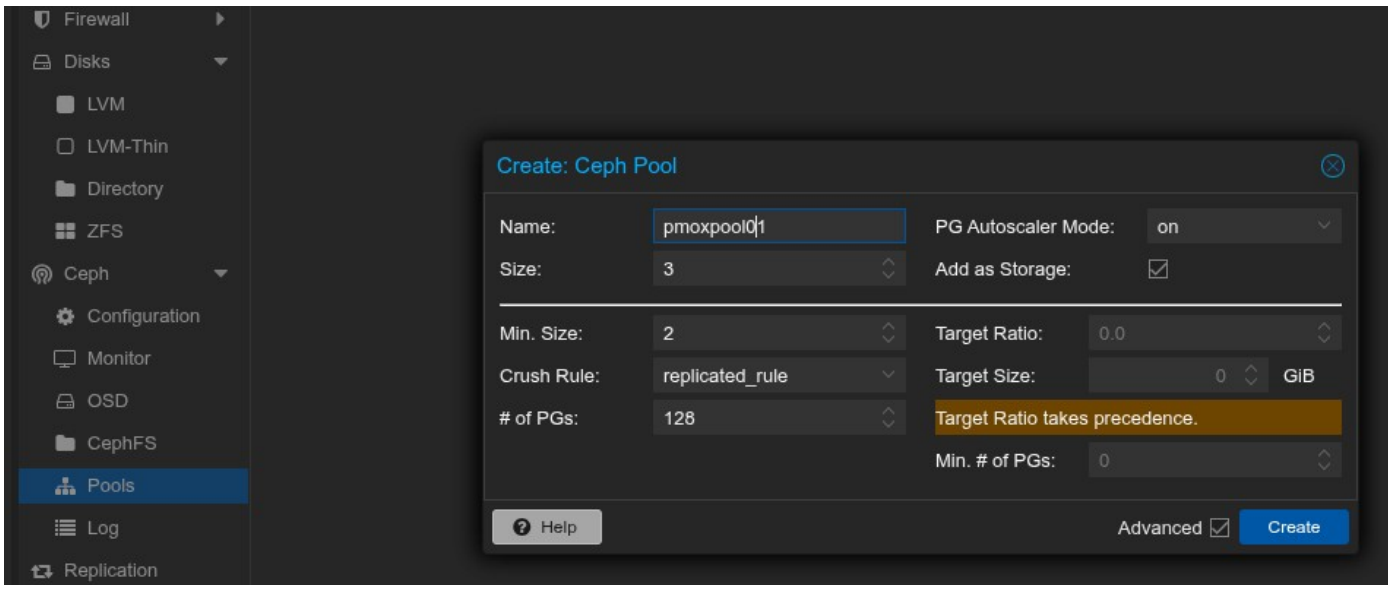
Name	Class	OSD Type	Status	Version	weight	reweight	Used (%)	Total	Apply/Commit Latency (ms)	PGs
default										
aymane-node3				19.2.3						
osd.2	hdd	bluestore	up / in	19.2.3	0.03419	1.00	1.19	35.00 GiB	1 / 1	0
aymane-node2				19.2.3						
osd.1	hdd	bluestore	up / in	19.2.3	0.03419	1.00	0.07	35.00 GiB	5 / 5	0
aymane-node1				19.2.3						
osd.0	hdd	bluestore	up / in	19.2.3	0.03419	1.00	0.07	35.00 GiB	1 / 1	0

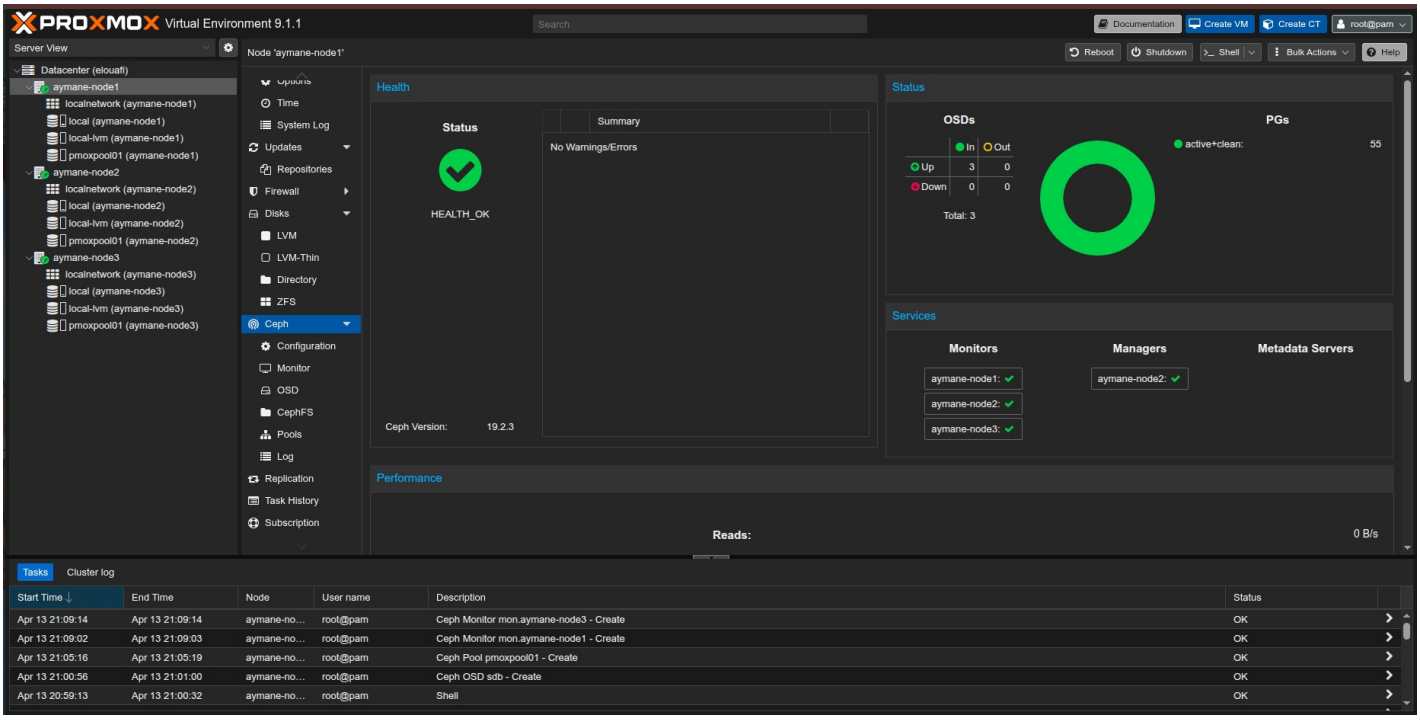
Storage Pool Creation (pmoxpool01) :

With the hardware layer recognized by Ceph, the logical storage pool was created to host the virtual machines. A pool named pmoxpool01 was provisioned with specific replication rules to guarantee High Availability:

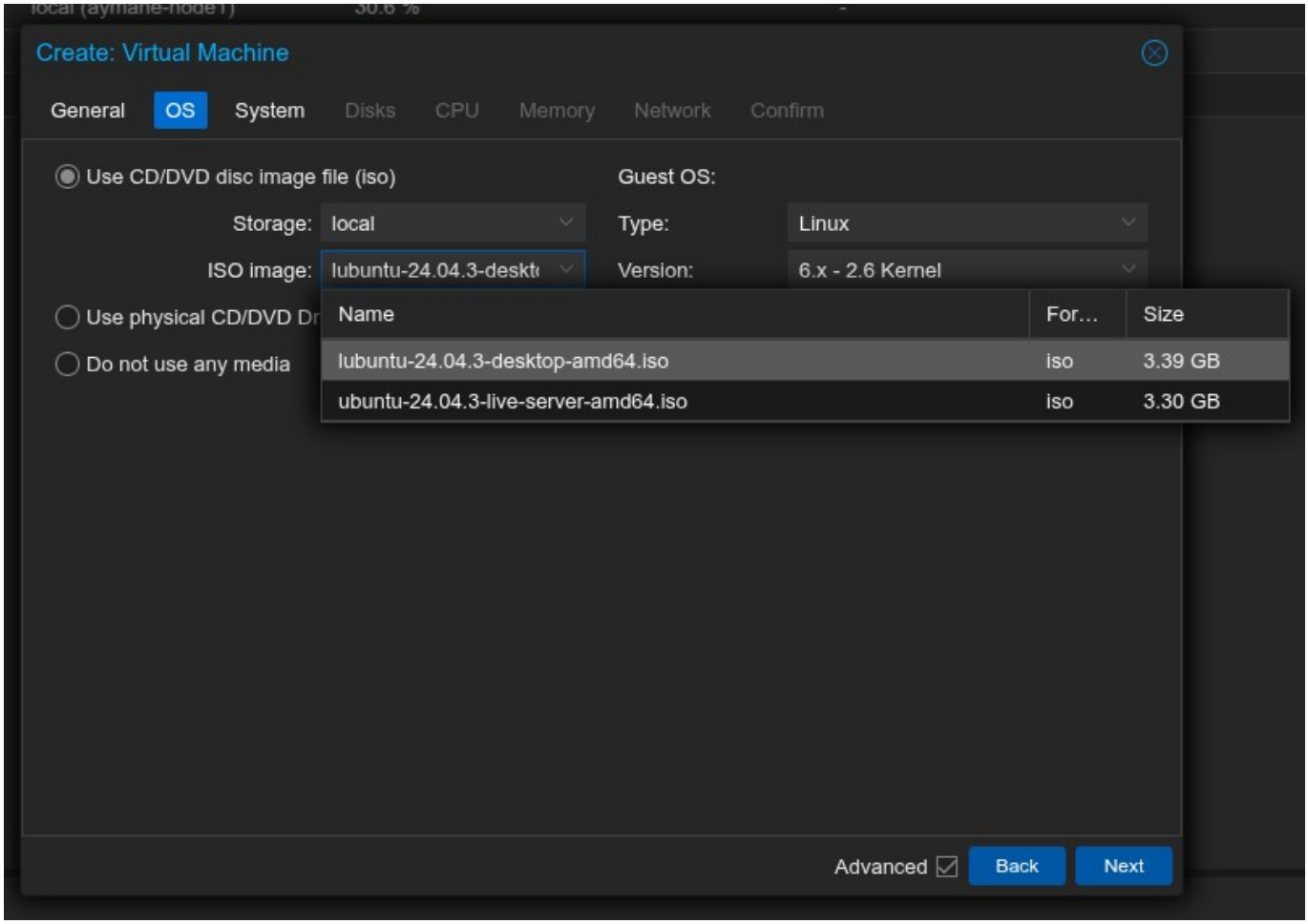
- **Size (Replicas) = 3:** Every block of data written to a VM is automatically copied to three different OSDs (one on each node).
- **Min_Size = 2:** The cluster will continue to accept write requests as long as at least two copies of the data can be successfully written. If two nodes fail simultaneously, the storage pool will lock itself to prevent data corruption.

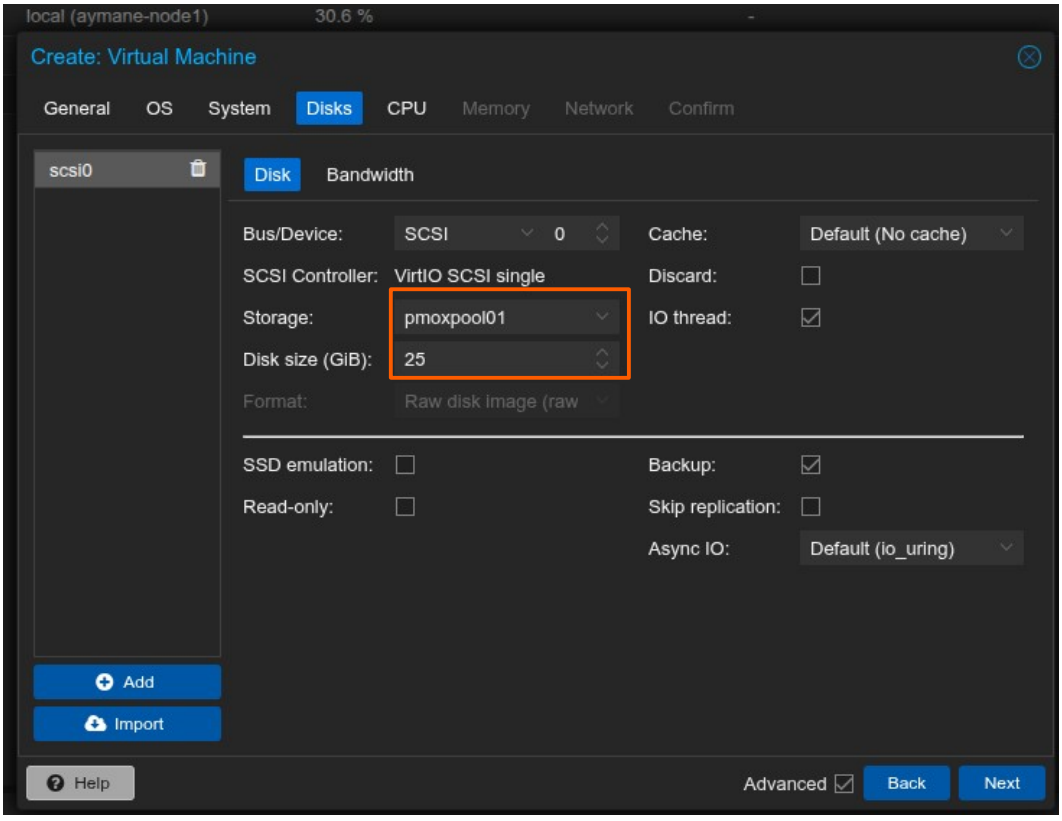
This pool was then added to the Proxmox Datacenter storage configuration, making it visible and available to all nodes for provisioning the Web Server and Monitoring VMs.

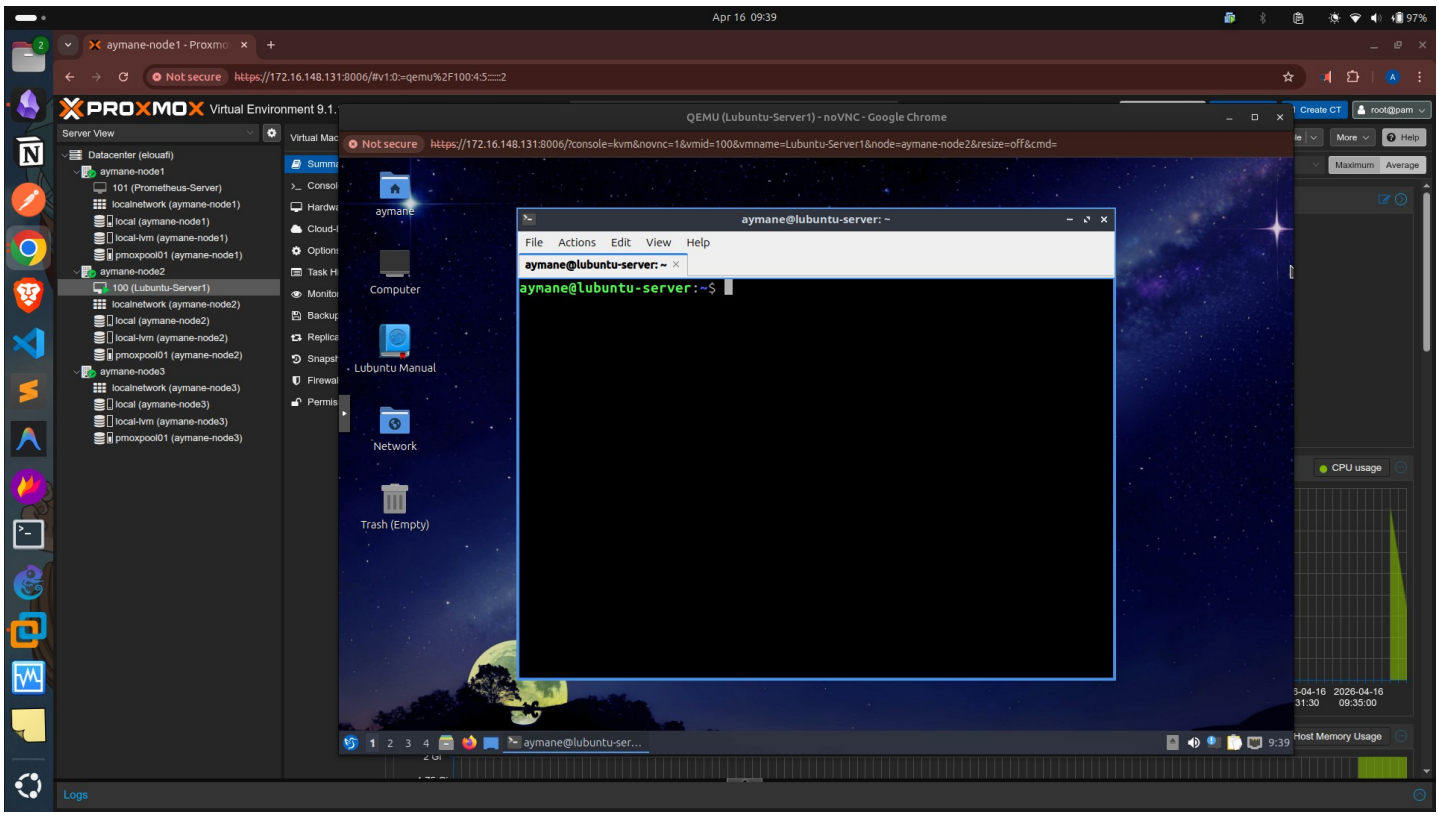

Firewall
Disks
LVM
LVM-Thin
Directory
ZFS
Ceph
Configuration
Monitor
OSD
CephFS
Pools
Log
Replication
Create: Ceph Pool
Name:
pmoxpool01
Size:
3
Min. Size:
2
Crush Rule:
replicated_rule
# of PGs:
128
PG Autoscaler Mode:
on
Add as Storage:
Target Ratio:
0.0
Target Size:
0
GiB
Target Ratio takes precedence.
Min. # of PGs:
0
Help
Advanced
Create


PROXMOX Virtual Environment 9.1.1
Search
Documentation
Create VM
Create CT
root@pam
Server View
Node 'aymane-node1'
Reboot
Shutdown
Shell
Bulk Actions
Help
Datacenter (elouafi)
aymane-node1
localnetwork (aymane-node1)
local (aymane-node1)
local-lvm (aymane-node1)
pmoxpool01 (aymane-node1)
aymane-node2
localnetwork (aymane-node2)
local (aymane-node2)
local-lvm (aymane-node2)
pmoxpool01 (aymane-node2)
aymane-node3
localnetwork (aymane-node3)
local (aymane-node3)
local-lvm (aymane-node3)
pmoxpool01 (aymane-node3)
Time
System Log
Updates
Repositories
Firewall
Disks
LVM
LVM-Thin
Directory
ZFS
Ceph
Configuration
Monitor
OSD
CephFS
Pools
Log
Replication
Task History
Subscription
Health
Status
HEALTH_OK
Summary
No Warnings/Errors
Ceph Version:
19.2.3
Status
OSDs
In
Out
Up
3
0
Down
0
0
Total: 3
PGs
active+clean:
55
Services
Monitors
aymane-node1:
aymane-node2:
aymane-node3:
Managers
aymane-node2:
Metadata Servers
Performance
Reads:
0 B/s
Tasks
Cluster log
Start Time
End Time
Node
User name
Description
Status
Apr 13 21:09:14
Apr 13 21:09:14
aymane-no...
root@pam
Ceph Monitor mon.aymane-node3 - Create
OK
Apr 13 21:09:02
Apr 13 21:09:03
aymane-no...
root@pam
Ceph Monitor mon.aymane-node1 - Create
OK
Apr 13 21:05:16
Apr 13 21:05:19
aymane-no...
root@pam
Ceph Pool pmoxpool01 - Create
OK
Apr 13 21:00:56
Apr 13 21:01:00
aymane-no...
root@pam
Ceph OSD sdb - Create
OK
Apr 13 20:59:13
Apr 13 21:00:32
aymane-no...
root@pam
Shell
OK

Create A Virtual Machine using the Ceph :







Logical Overlay Network: Software-Defined Networking via VXLAN

```
auto vxlan1
iface vxlan1 inet manual
        vxlan-id 100
        vxlan-remoteip 172.16.148.131
        vxlan-remoteip 172.16.148.133

auto vmbr1
iface vmbr1 inet static
        address 10.10.10.1/24
        bridge-ports vxlan1
        bridge-stp off
        bridge-fd 0
```

While the physical underlay provides the base connectivity for the hypervisors, deploying virtual machines directly onto the underlay (vmbr0) is considered poor architectural practice. It exposes internal workloads to the external network and limits mobility. To address this, an isolated, Software-Defined Network (SDN) was required.

The Engineering Problem: The Layer 2 Boundary Limitation

The core challenge in a clustered virtual environment is network mobility. If a virtual machine (e.g., the Web Server) is live-migrated from Node 2 to Node 3 using the Ceph shared storage, it must retain its IP address and network connections. To do this, the VM must believe it is plugged into the exact same physical Layer 2 switch, regardless of which physical hypervisor it resides on.

However, standard Layer 2 broadcast domains (like ARP requests and standard MAC-based switching) cannot cross Layer 3 routed networks. If we simply create a private virtual switch on Node 1 and another private virtual switch on Node 2, they are completely isolated. Traffic cannot flow between them.

The Foundation: The Internal Bridge (vmbr1)

To build the isolated tenant network, a secondary Linux bridge, **vmbr1**, was created on all three nodes. vmbr1 acts as the internal virtual switch where the virtual machines connect their virtual network interface cards (vNICs).

- **On Node 2 and Node 3:** vmbr1 operates purely at Layer 2 (Switching). It has no IP address assigned (inet manual). It simply receives frames from the VMs and forwards them.
- **On aymane-node1:** vmbr1 operates at both Layer 2 and Layer 3 (Routing). It is assigned the static IP address **10.10.10.1/24**. This makes Node 1 the designated Default Gateway for the entire internal subnet.

The Initial Challenge: The Nested MAC Security Limitation

During the initial deployment phase, virtual machines were bridged directly to the external vmbr0 interface to access the 172.16.148.0/24 underlay. However, this resulted in immediate network failure: the VMs could not obtain DHCP leases and, even with static IPs assigned, could not route to the internet. They could only successfully ping their immediate hypervisor host.

The root cause of this failure was traced to the security mechanisms of the upstream hypervisor (VMware Workstation). In a nested virtualization environment, the VMware virtual switch enforces strict MAC address filtering. It only permits traffic originating from the authorized MAC address assigned to the Proxmox host's nic0 interface. When the nested Proxmox VMs attempted to send traffic using their own generated MAC addresses, the VMware vSwitch dropped the packets as unauthorized "MAC spoofing."

The Intermediate Fix: The Local Private Bridge

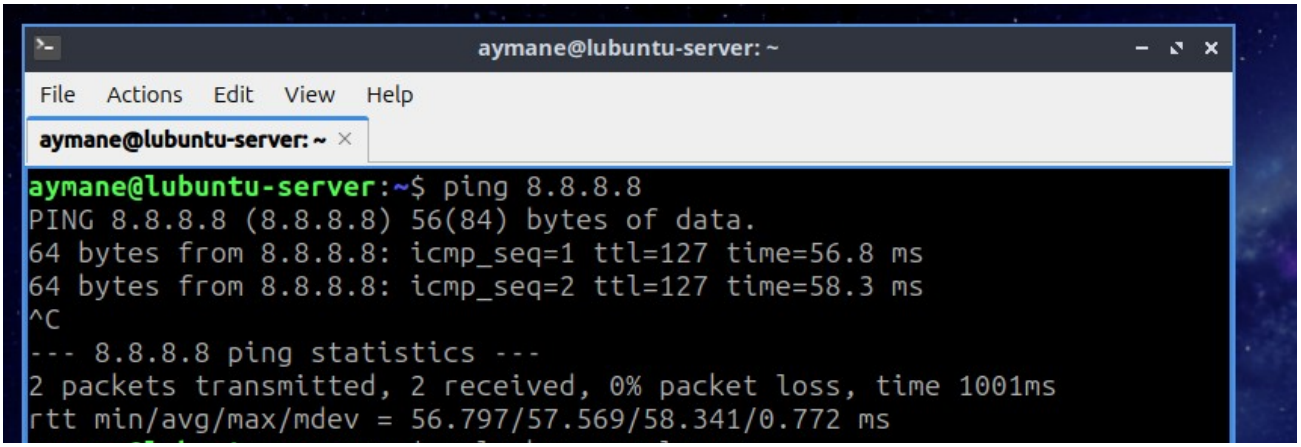
To bypass this security restriction without compromising the VMware configuration, a secondary, isolated Linux bridge (**vmbr1**) was created in each Node.Their traffic was forced through Nodes NAT firewall (MASQUERADE). The NAT process rewrote the packets to use Nodes authorized MAC and IP address, successfully tricking VMware and granting the VMs internet access.

On each Node :

```
# 1. Add the bridge configuration
cat << 'EOF' >> /etc/network/interfaces
auto vmbr1
iface vmbr1 inet static
      address 10.10.10.1/24
      bridge-ports none
      bridge-stp off
      bridge-fd 0
EOF
# 2. Apply the network changes
ifreload -a
```

Firewall Rules :

```
# 1. Enable IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2. Add the Masquerade rule (routes internal traffic out through vmbr0)
iptables -t nat -A POSTROUTING -s 10.10.10.0/24 -o vmbr0 -j MASQUERADE
# 3. Allow traffic to flow out to the external bridge
iptables -A FORWARD -i vmbr1 -o vmbr0 -j ACCEPT
# 4. Allow the return traffic back in
iptables -A FORWARD -i vmbr0 -o vmbr1 -m state --state RELATED,ESTABLISHED -j ACCEPT
```



- **Problem** : vmbr1 on node1 and vmbr1 on node2 or node3 are completely isolated virtual switches. They have no physical wire connecting them.

The Final Architectural Problem: Node Isolation

While the vmbr1 NAT configuration solved the internet routing issue, it created a severe operational limitation: vmbr1 was physically isolated to each node. A Web Server VM on Node 2's vmbr1 could not ping the Monitoring VM on Node 3's vmbr1. The private bridges were disconnected islands. To achieve true cluster mobility and cross-node communication, a technology was required to securely connect these isolated islands over the physical network.

The Solution: VXLAN (Virtual Extensible LAN)

To connect the isolated vmbr1 bridges across the three distinct hypervisors, the architecture utilizes **VXLAN** (Virtual Extensible LAN) using VNI (VXLAN Network Identifier) **100**.

VXLAN is an advanced network virtualization technology that provides a solution to the Layer 2 boundary problem through a technique called **MAC-in-UDP encapsulation**. Here is how it functions within the cluster:

1. **VTEP Creation:** Each Proxmox node acts as a VXLAN Tunnel Endpoint (VTEP). A virtual interface (vxlan1) is created and plugged into the vmbr1 bridge alongside the VMs.
2. **Encapsulation:** When a VM on Node 2 sends a packet to the Gateway on Node 1, the packet enters vmbr1. The VTEP intercepts this standard Layer 2 Ethernet frame and "wraps" it inside a new UDP packet.
3. **Transit:** This newly encapsulated UDP packet is then routed over the physical underlay network (172.16.148.x), addressed directly to Node 1's external IP address.

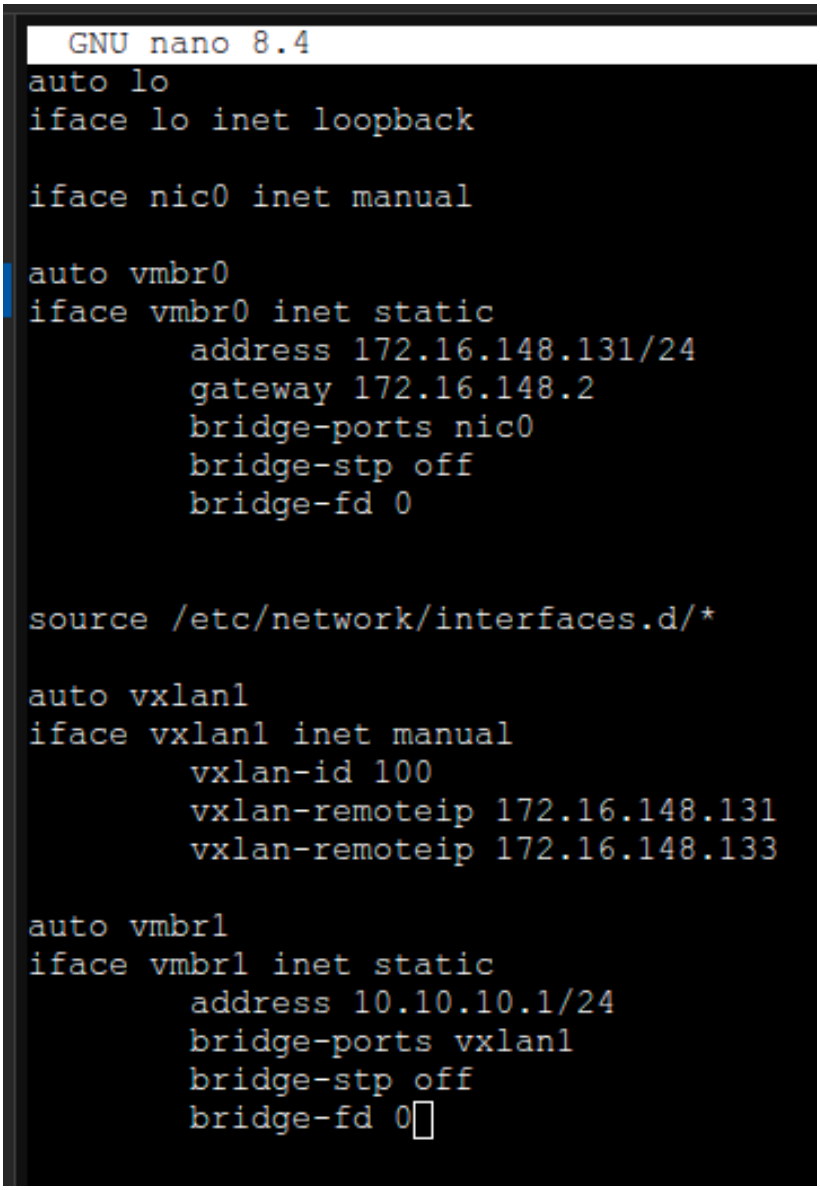
4. **Decapsulation:** When Node 1 receives the UDP packet, its VTEP strips away the VXLAN header, revealing the original, intact 10.10.10.x packet, and drops it onto its own vmbr1 bridge.

Architectural Benefits

By implementing VXLAN, the infrastructure successfully achieves **"Layer 2 stretching"** over a Layer 3 underlay. To the virtual machines, the 10.10.10.0/24 subnet appears as a single, contiguous, flat physical switch, completely hiding the complexity of the underlying Proxmox cluster. This enables absolute tenant isolation, secure internal traffic flow, and seamless zero-downtime VM migrations across the Ceph storage pool.

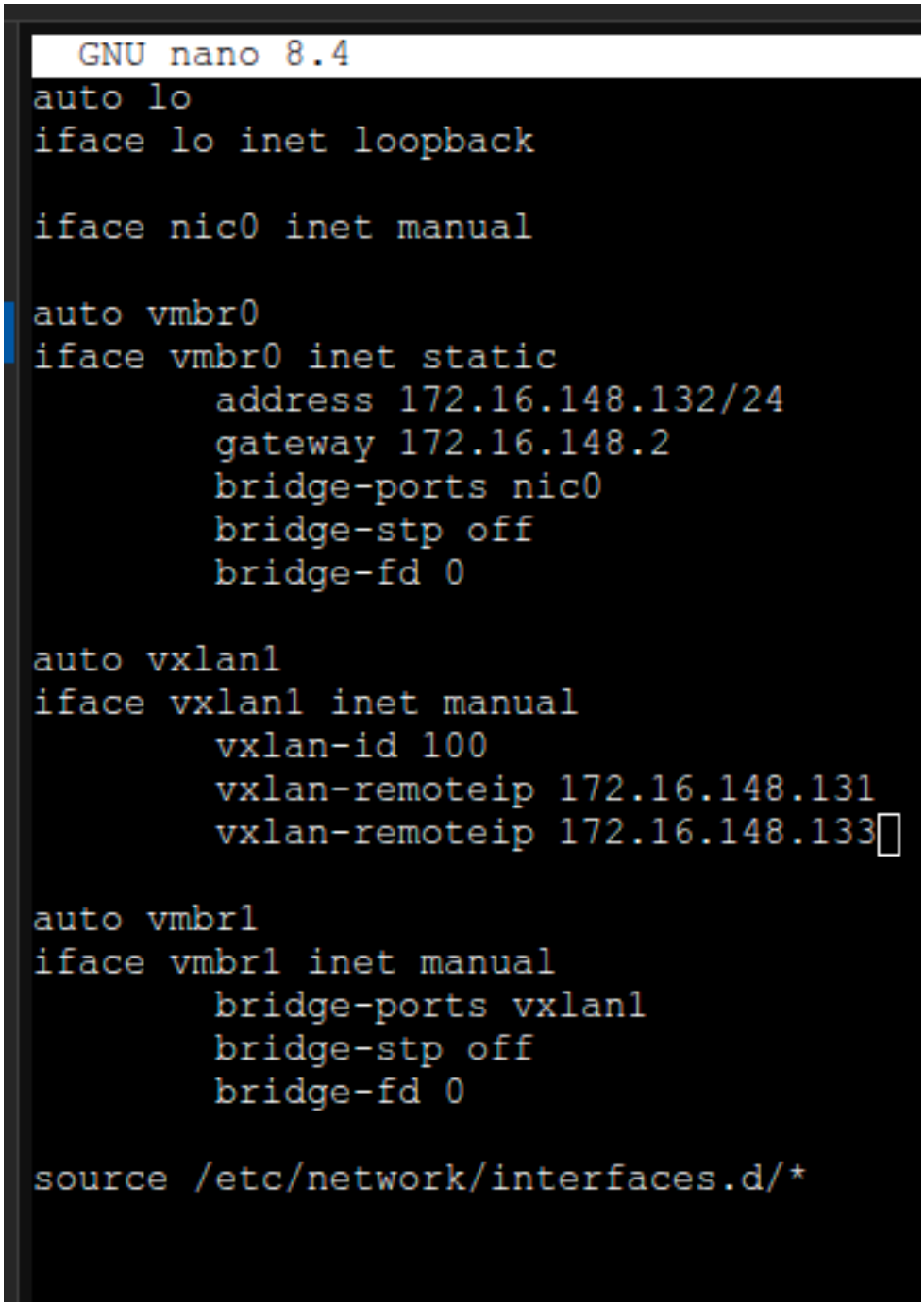
Reconfigure Node 1 (The Master Gateway)

We need to attach a VXLAN tunnel to your existing vmbr1 so it can talk to the other nodes.



Configure Node 2 & Node 3 (The Forwarders)

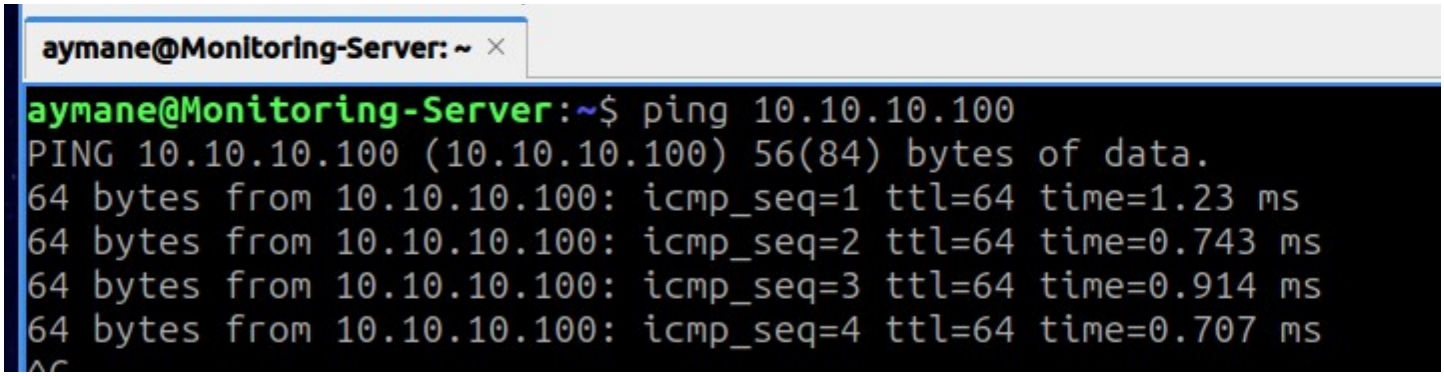
Since Node 1 is holding the gateway IP (10.10.10.1) and doing the NAT to the internet, Node 2 and Node 3 **do not need an IP address** on their internal bridges. They just need the bridge to exist so the VMs can plug into it.

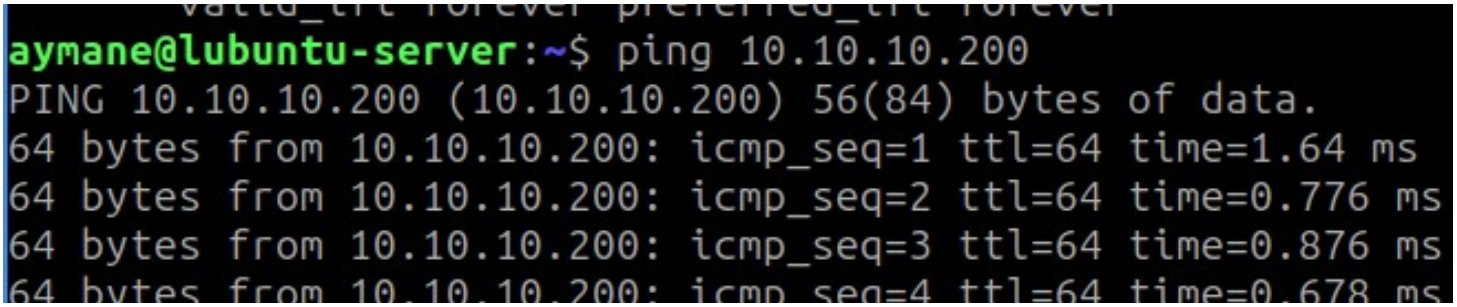


Firewall Rules :

```
# 1. Enable IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2. Add the Masquerade rule (routes internal traffic out through vmbr0)
iptables -t nat -A POSTROUTING -s 10.10.10.0/24 -o vmbr0 -j MASQUERADE
# 3. Allow traffic to flow out to the external bridge
iptables -A FORWARD -i vmbr1 -o vmbr0 -j ACCEPT
# 4. Allow the return traffic back in
iptables -A FORWARD -i vmbr0 -o vmbr1 -m state --state RELATED,ESTABLISHED -j ACCEPT
# VXLAN fix
iptables -t mangle -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

Ping Tests :





```
aymane@Monitoring-Server:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=123 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=56.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=126 ms
^C
--- 8.8.8.8 ping statistics ---
```

```
aymane@lubuntu-server:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=60.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=57.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=56.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=127 time=60.5 ms
^C
```

Services Deployed & Monitoring:

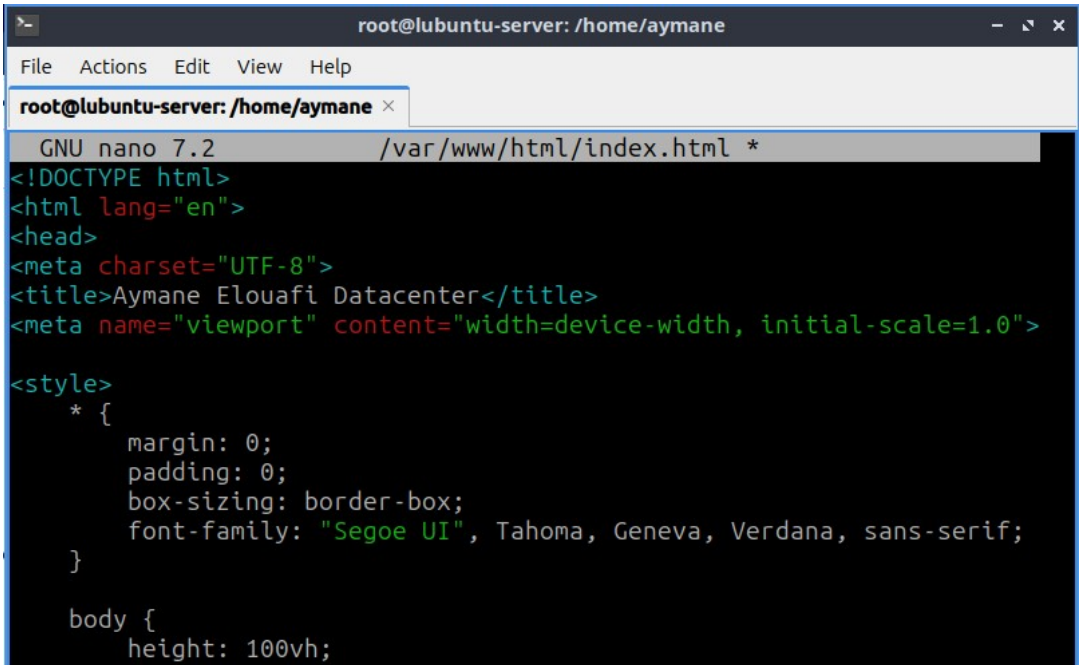
Web Server Virtual Machine :

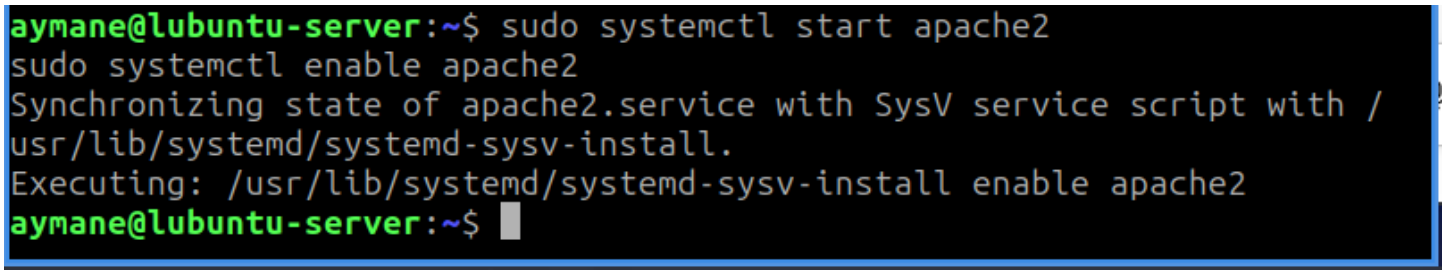
To validate the functionality of the underlying Ceph storage cluster and the isolated VXLAN network, a primary tenant workload was deployed in the form of a Web Server Virtual Machine.

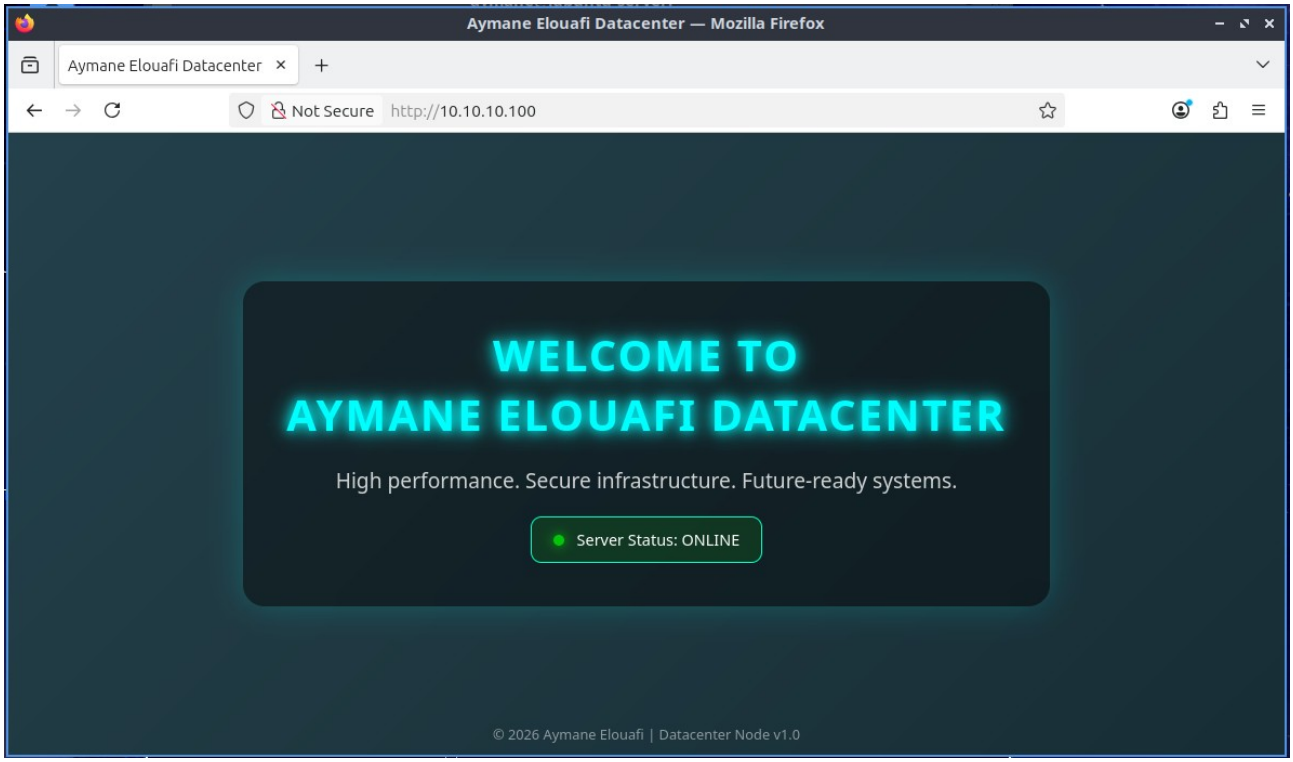
Setup Static Ip address for the web server :

```
aymane@Monitoring-Server:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gr
oup default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether bc:24:11:3c:8b:7d brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 10.10.10.200/24 brd 10.10.10.255 scope global noprefixroute e
ns18
       valid_lft forever preferred_lft forever
    inet6 fe80::5531:e6d9:dfe0:3ec7/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
aymane@Monitoring-Server:~$
```

Apache Web Server :







Observability Stack: Prometheus and Grafana Monitoring

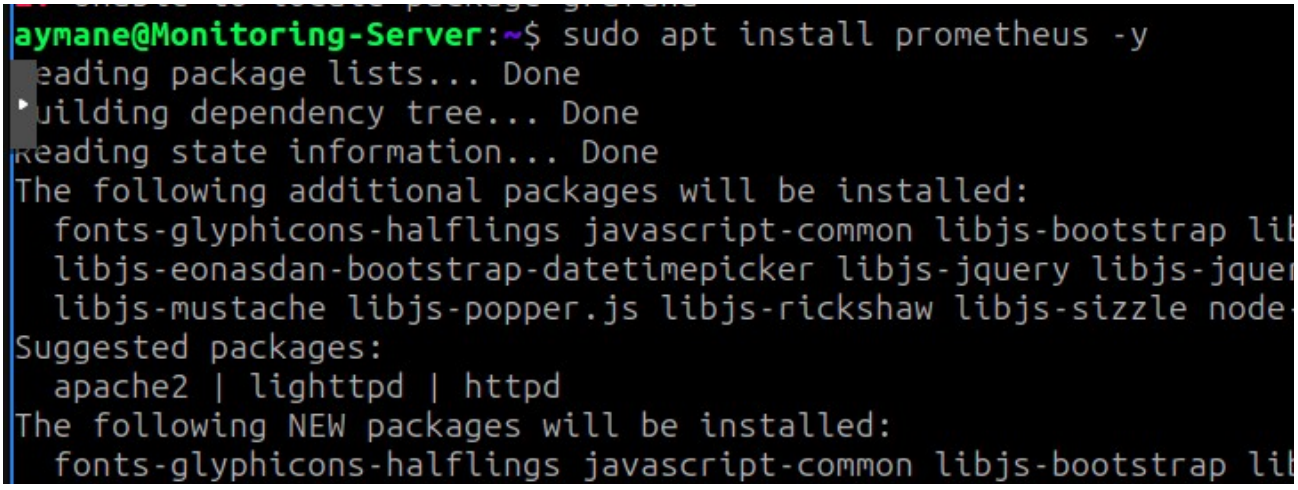
In a distributed virtual architecture, providing deep visibility into tenant workloads and isolated services is critical for identifying bottlenecks and preventing outages. To achieve this, a centralized observability stack was deployed. Before detailing the implementation, it is essential to define the two core technologies driving this stack.

What is Prometheus?



Prometheus is an industry-standard, open-source systems monitoring and alerting toolkit. Unlike traditional monitoring systems that passively wait for agents to send them data (a "push" model), Prometheus operates on an active "pull" architecture. At its core, it is a highly efficient Time-Series Database (TSDB). It relies on lightweight agents called "Exporters" (such as Node Exporter) installed on target machines. These exporters translate hardware and OS metrics into a standardized format and expose them over a simple HTTP endpoint. The central Prometheus server then periodically "scrapes" (pulls) this data, storing it with precise timestamps.

Install Prometheus on the Monitoring Server :



Install & Enable Prometheus Node Exporter in the target VMS :

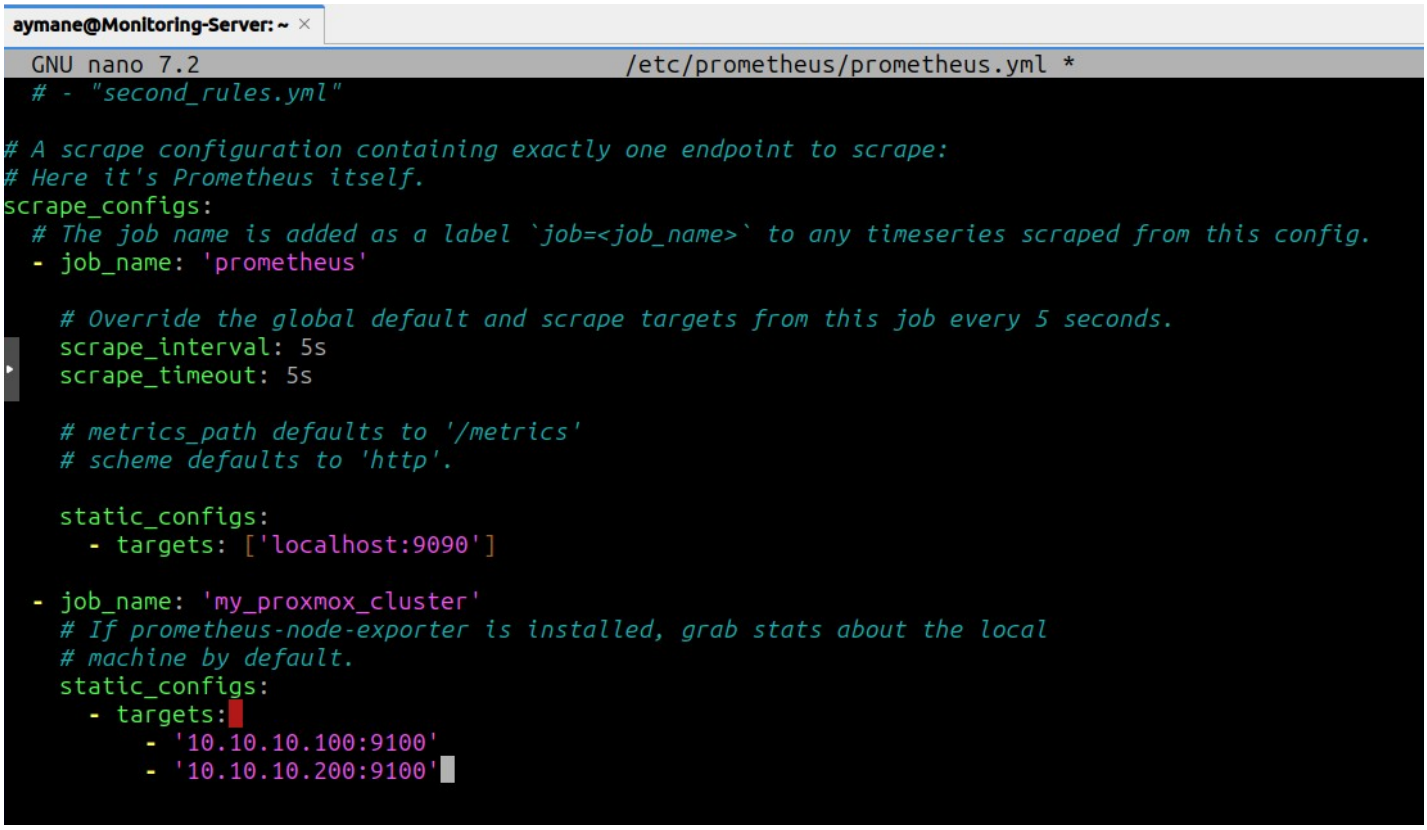
```
aymane@lubuntu-server:~$ sudo apt install prometheus-node-exporter
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  freeipmi-common ipmitool libfreeipmi17 libio-pty-perl libipc-run-per
  moreutils nvme-cli openipmi prometheus-node-exporter-collectors pyth
  smartmontools
```

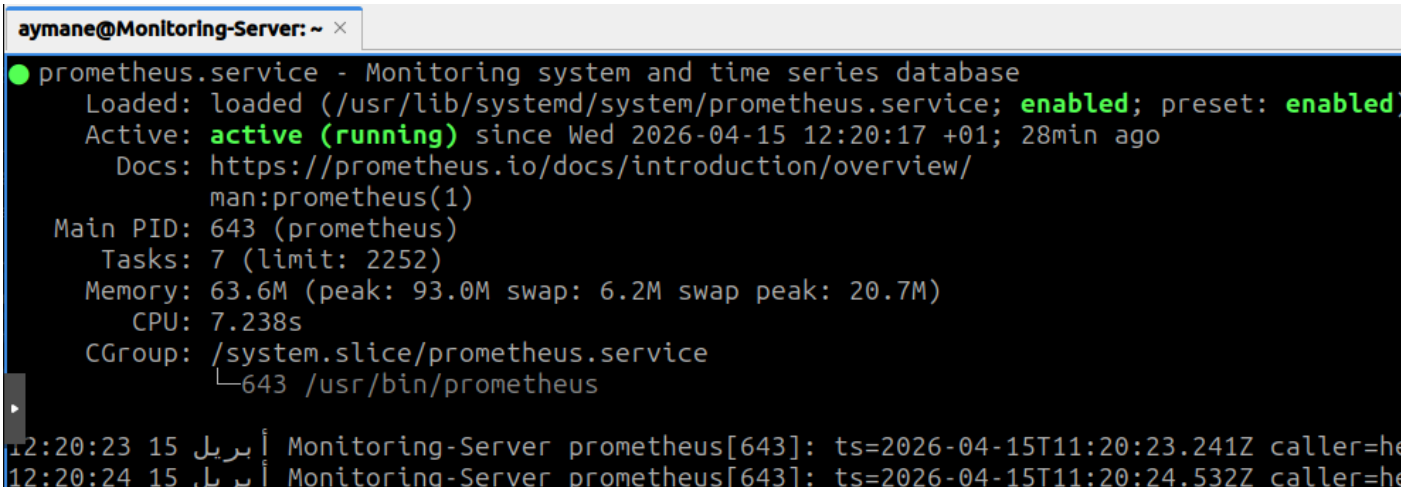
```
aymane@Monitoring-Server:~$ sudo apt install prometheus-node-exporter -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  freeipmi-common ipmitool libfreeipmi17 libio-pty-perl libipc-run-perl libc
  moreutils nvme-cli openipmi prometheus-node-exporter-collectors python3-de
  smartmontools
```

```
Processing triggers for libc-bin (2.39-0ubuntu8.5) ...
aymane@lubuntu-server:~$ sudo systemctl enable --now prometheus-node-exporter
Synchronizing state of prometheus-node-exporter.service with SysV service scri
v-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable prometheus-node-export
aymane@lubuntu-server:~$
```

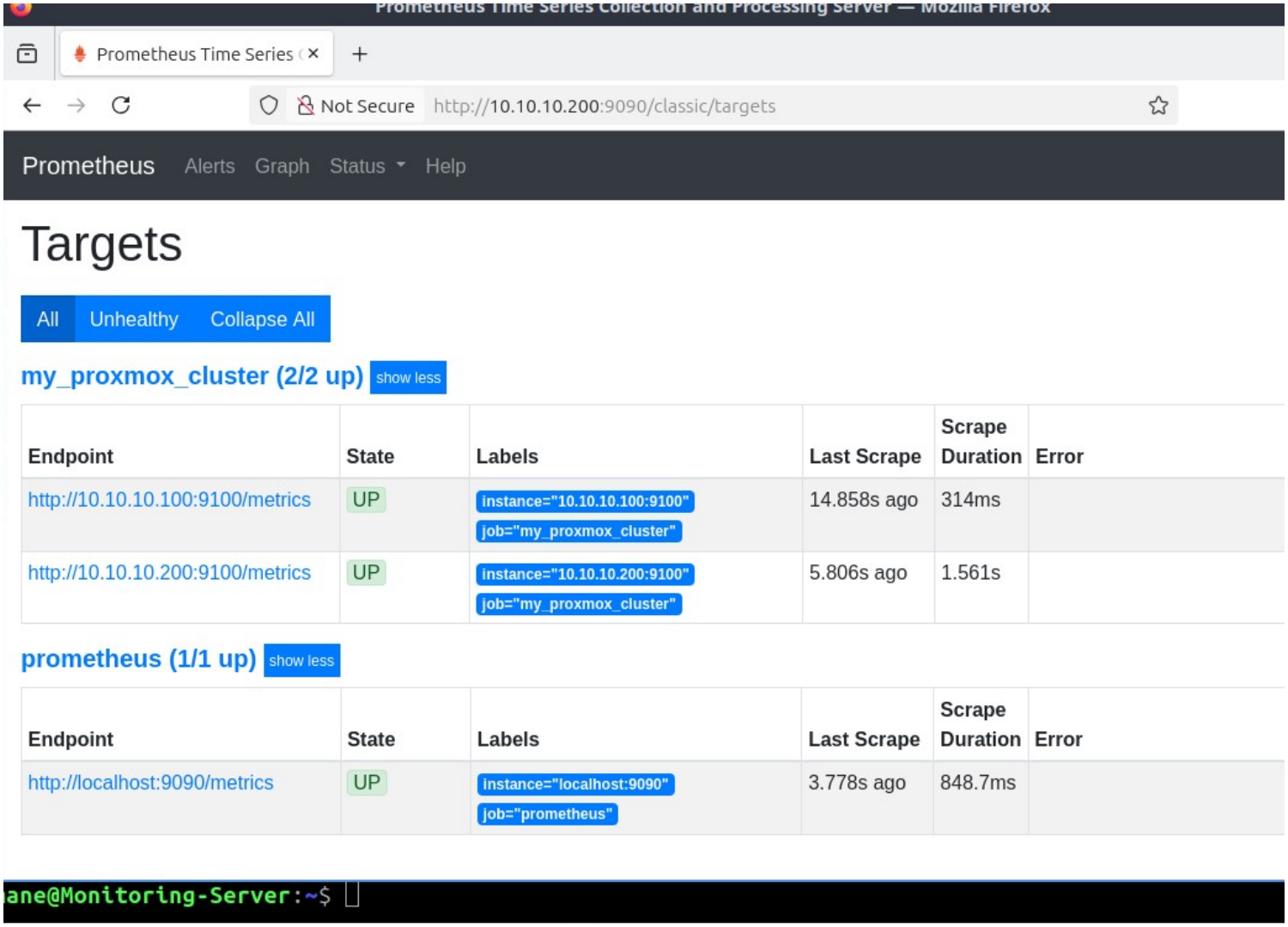
```
aymane@Monitoring-Server:~$ sudo systemctl enable --now prometheus-node-exporter
Synchronizing state of prometheus-node-exporter.service with SysV service script
v-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable prometheus-node-exporter
aymane@Monitoring-Server:~$
```

Add the targets on /etc/prometheus/prometheus.yml :





Prometheus Dashboard :



What is Grafana?



While Prometheus is exceptionally powerful at collecting and storing vast amounts of metric data, its native interface is not designed for complex data visualization. Grafana is an open-source analytics and interactive visualization platform that bridges this gap. It connects natively to Prometheus (acting as its primary data source) and translates the raw time-series queries into highly readable, dynamic, and customizable dashboards. Together, Prometheus and Grafana form the standard backbone of modern cloud-native observability.

Install & Enable Grafana on the Monitoring Server :

```
aymane@Monitoring-Server:~$ sudo apt-get install -y adduser libfontconfig1 musl

wget https://dl.grafana.com/grafana/release/13.0.0/grafana_13.0.0_24384745231_l
inux_amd64.deb
sudo dpkg -i grafana_13.0.0_24384745231_linux_amd64.deb
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
adduser is already the newest version (3.137ubuntu1).
libfontconfig1 is already the newest version (2.15.0-1.1ubuntu2).
musl is already the newest version (1.2.4-2).
0 upgraded, 0 newly installed, 0 to remove and 301 not upgraded.
--2026-04-15 11:15:14--  https://dl.grafana.com/grafana/release/13.0.0/grafana_
13.0.0_24384745231_linux_amd64.deb
Resolving dl.grafana.com (dl.grafana.com)... 199.232.198.217, 199.232.194.217,
2a04:4e42:4c::729, ...
Connecting to dl.grafana.com (dl.grafana.com)|199.232.198.217|:443... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 274205968 (262M) [application/octet-stream]
Saving to: ‘grafana_13.0.0_24384745231_linux_amd64.deb’

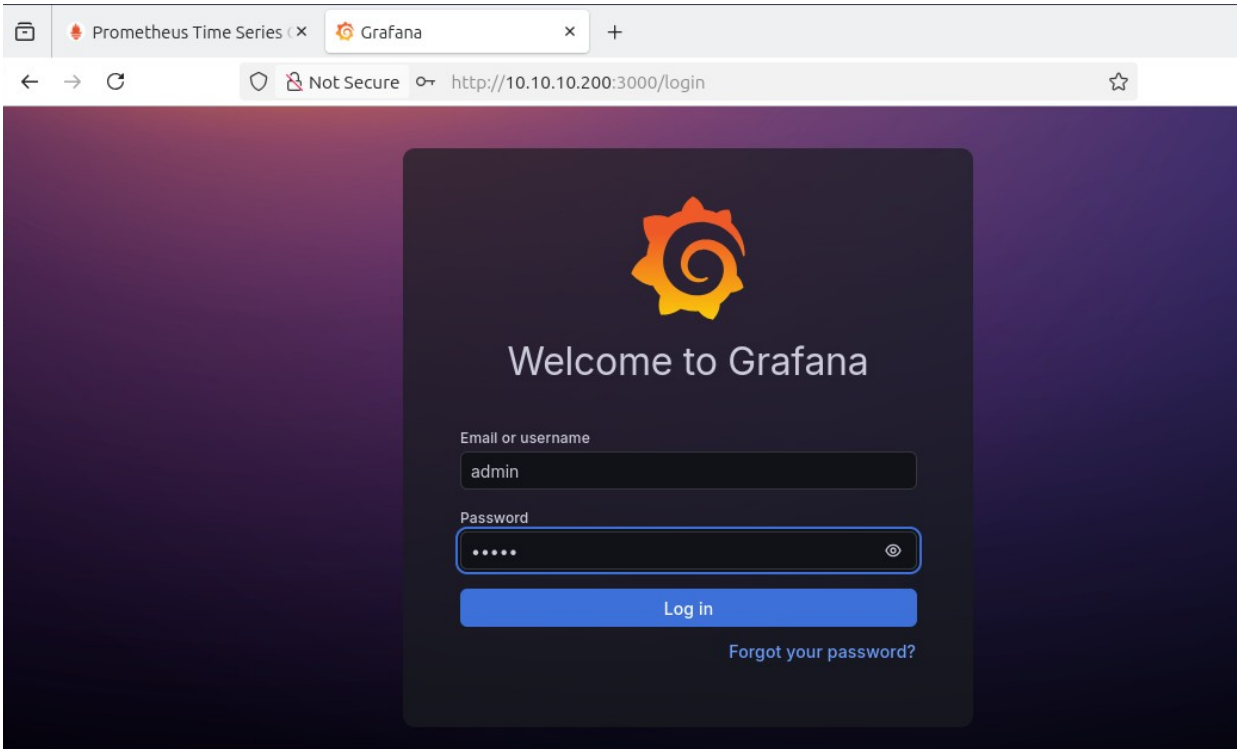
.0_24384745231_linu   1%[                    ]   2.78M   132KB/s    eta 8m 30s
```

```
aymane@Monitoring-Server:~$ sudo systemctl enable --now grafana-server
Synchronizing state of grafana-server.service with SysV service script w
r/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-serv
ice → /usr/lib/systemd/system/grafana-server.service.
```

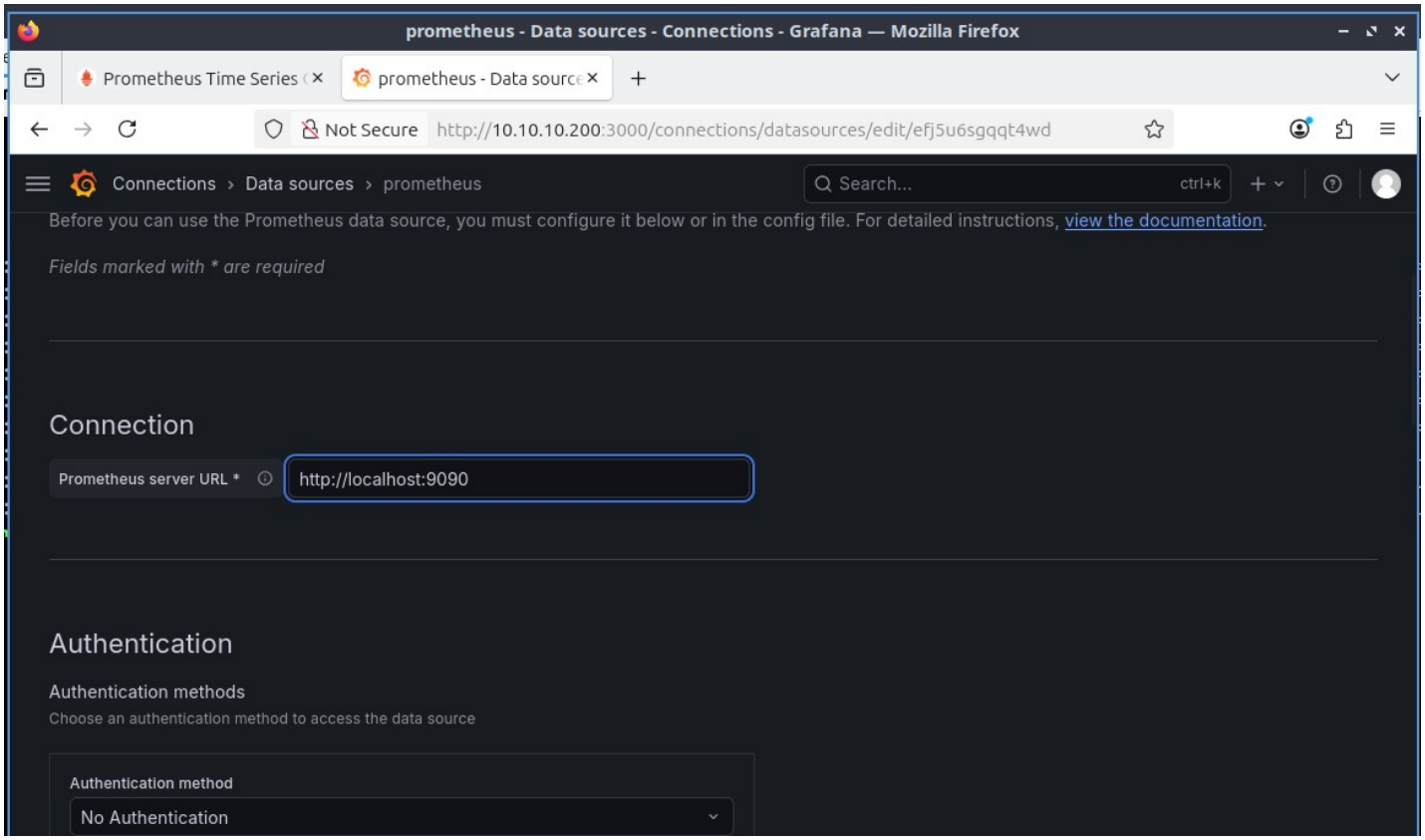
```
aymane@Monitoring-Server: ~
aymane@Monitoring-Server:~$ sudo systemctl start grafana-server
aymane@Monitoring-Server:~$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; >
     Active: active (running) since Wed 2026-04-15 11:49:23 +01; 25s ago
       Docs: http://docs.grafana.org
   Main PID: 32829 (grafana)
      Tasks: 4 (limit: 2253)
     Memory: 44.0M (peak: 44.3M)
        CPU: 2.996s
     CGroup: /system.slice/grafana-server.service
             └─32829 /usr/share/grafana/bin/grafana server --config=/etc/grafa>

11:49:49 15 أبريل Monitoring-Server grafana[32829]: logger=migrator t=2026-04->
11:49:49 15 أبريل Monitoring-Server grafana[32829]: logger=migrator t=2026-04->
```

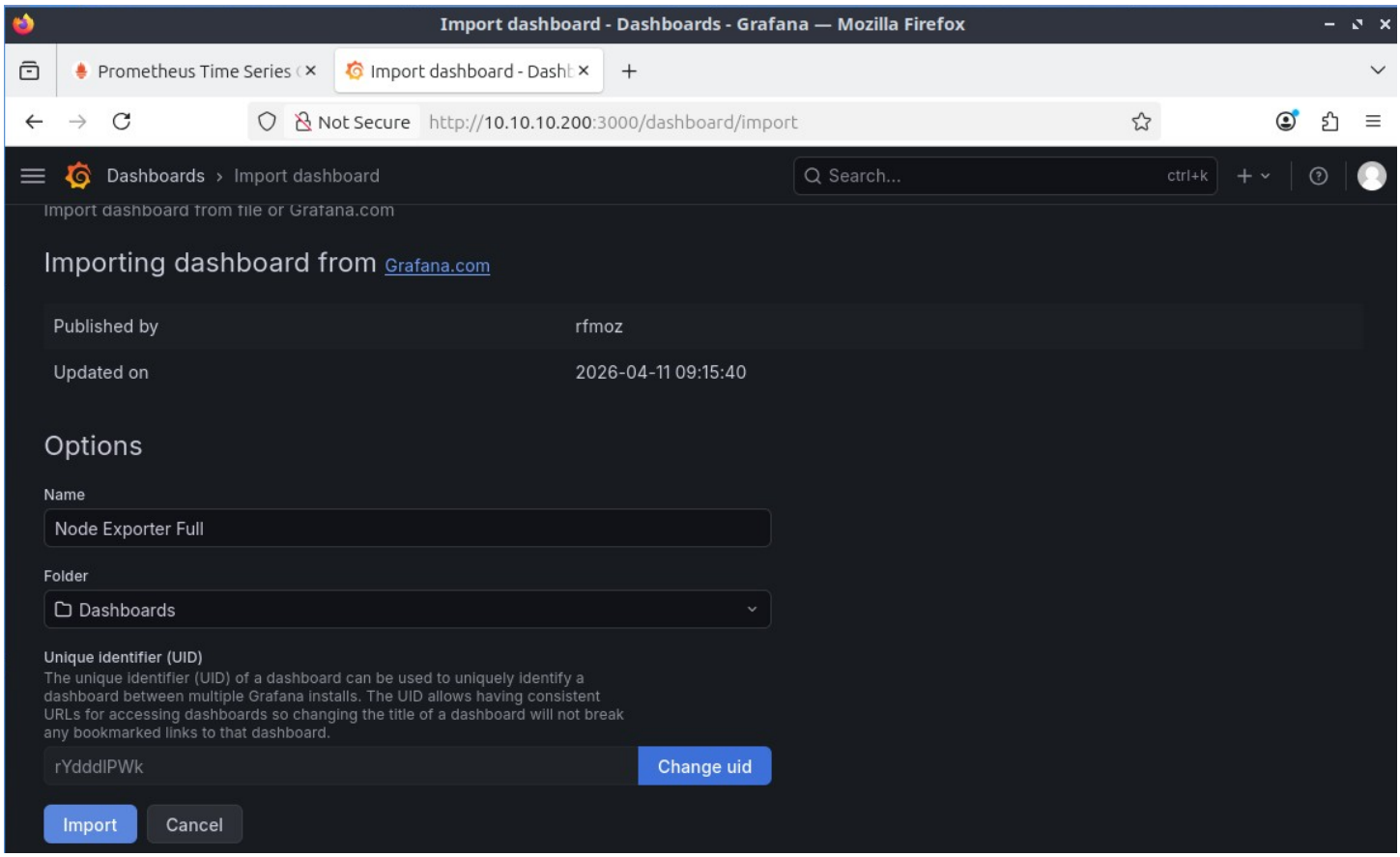
Connect to Grafana with admin/admin :



Add Prometheus as data source :

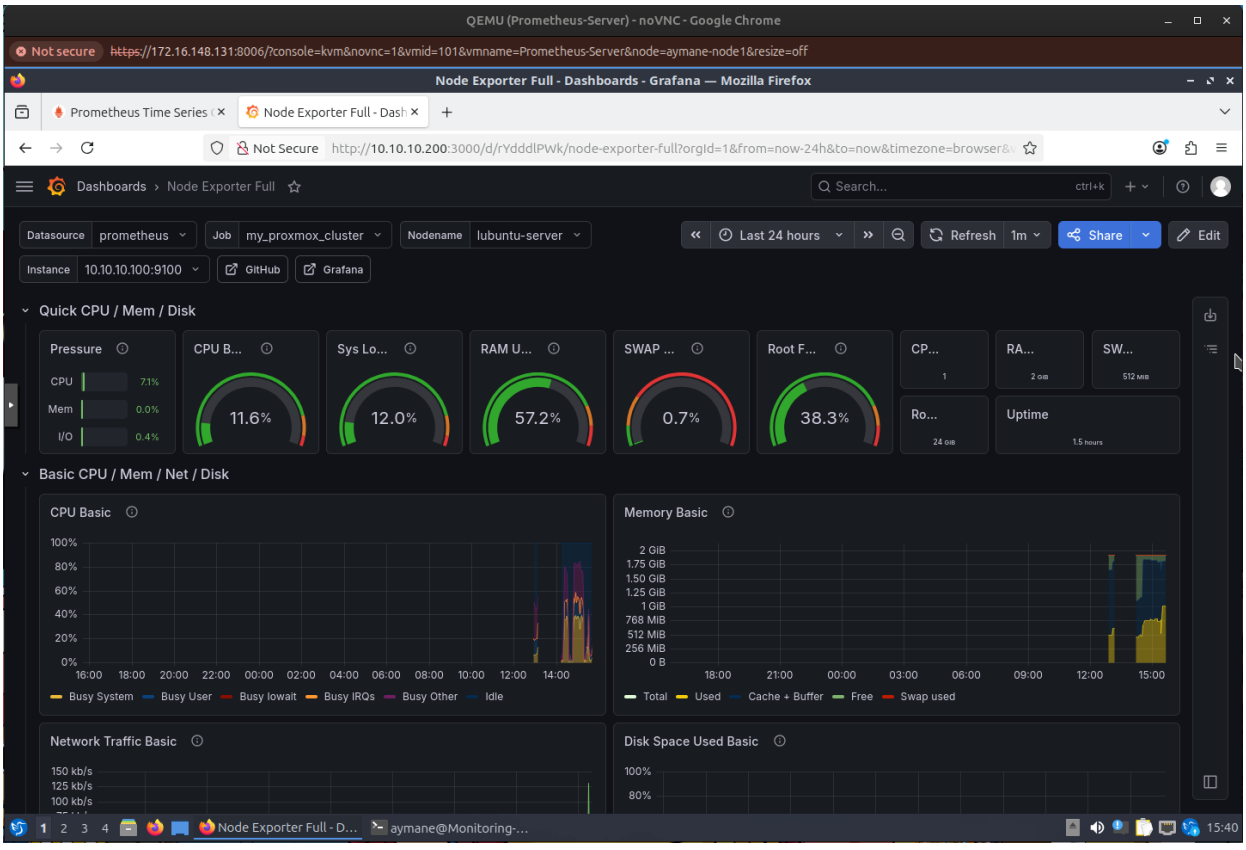


Import the node exporter dashboard using the id=1860 :

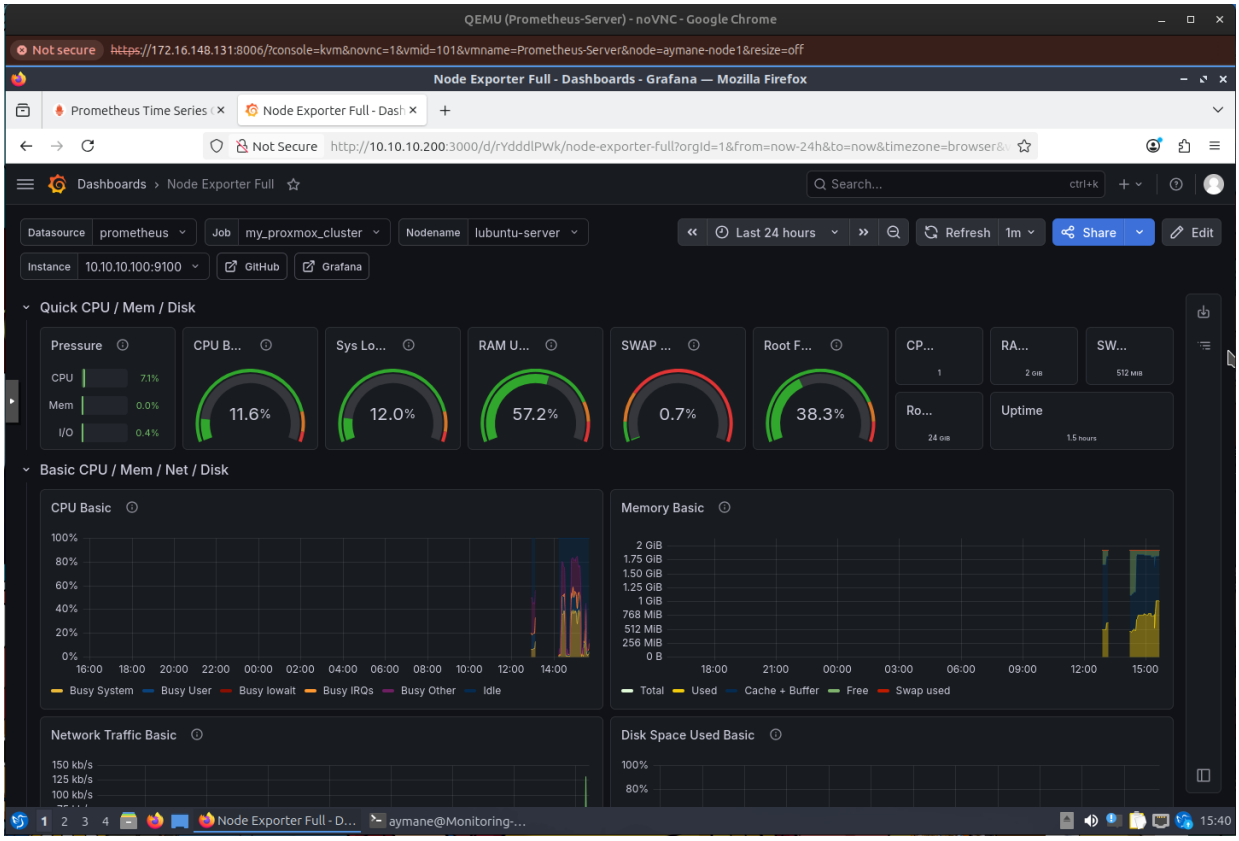


Grafana Dashboard :

- Lubuntu Server (WebServer) :



- Monitoring Server :



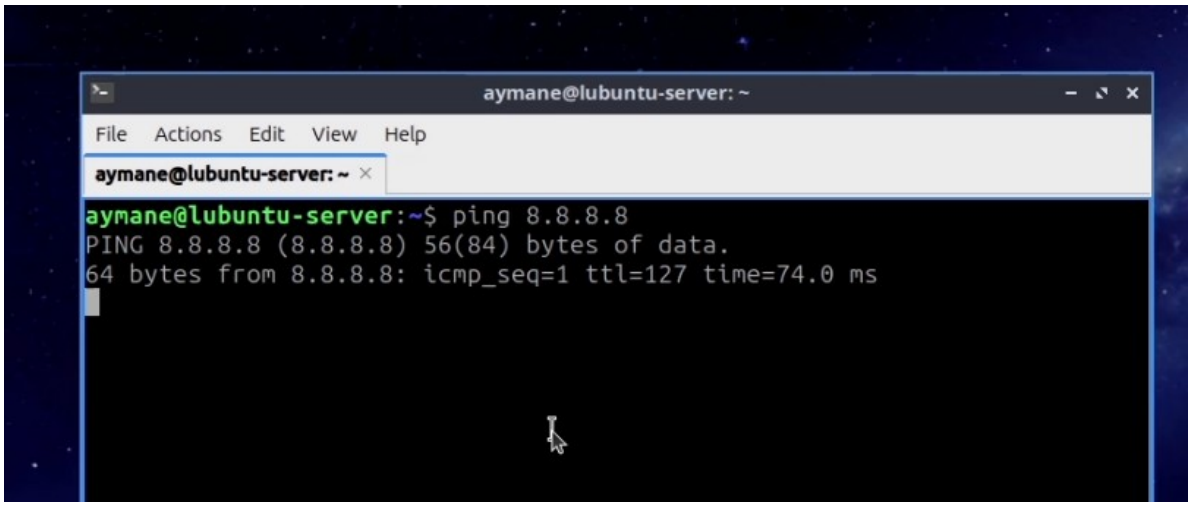
High Availability and Fault Tolerance Validation

The primary engineering objective of deploying a hyper-converged Proxmox cluster with Ceph storage is to ensure that tenant workloads can survive catastrophic hardware failures. To validate the resilience of this architecture, a simulated hardware outage (chaos test) was executed against the active cluster.

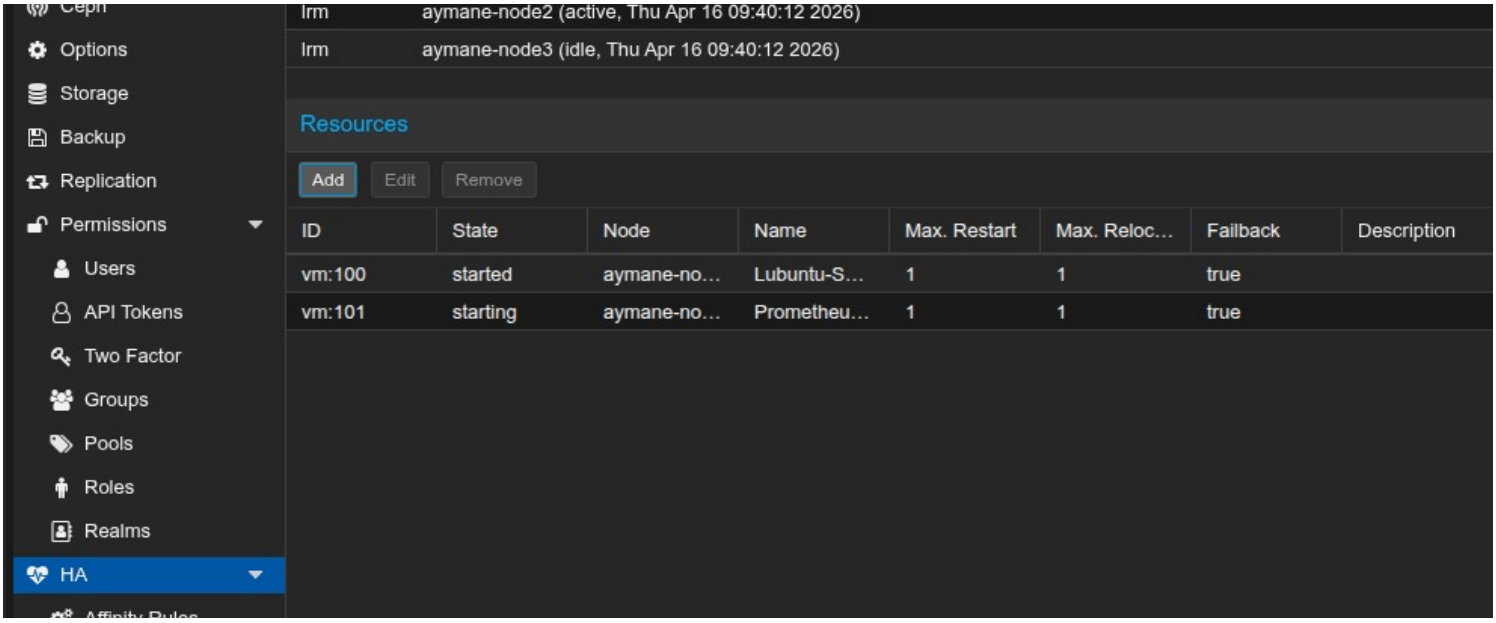
Phase 1: Live Migration Test

To accurately measure the impact of a node failure, a continuous baseline of connectivity was established.

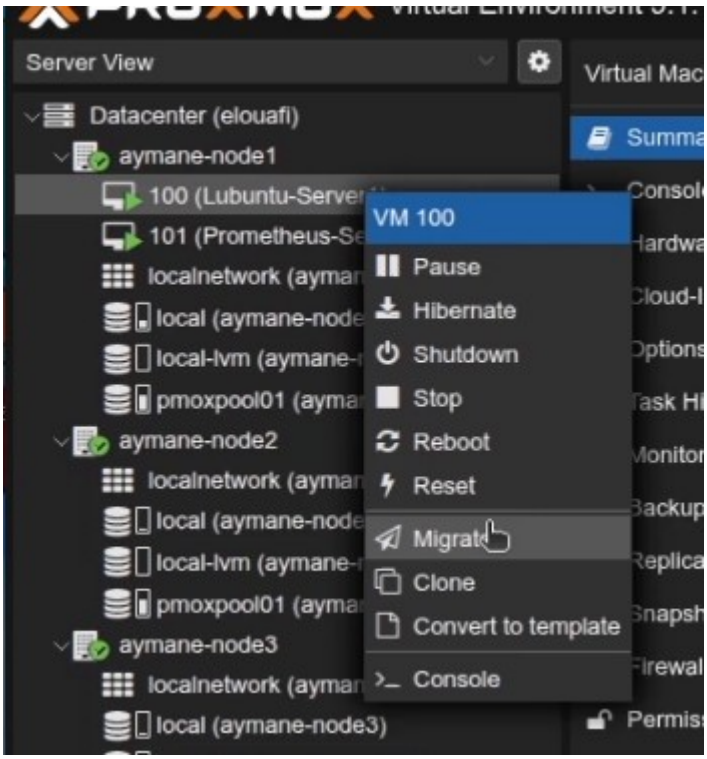
1. The **Web Server VM** was verified to be actively running on **node1**.

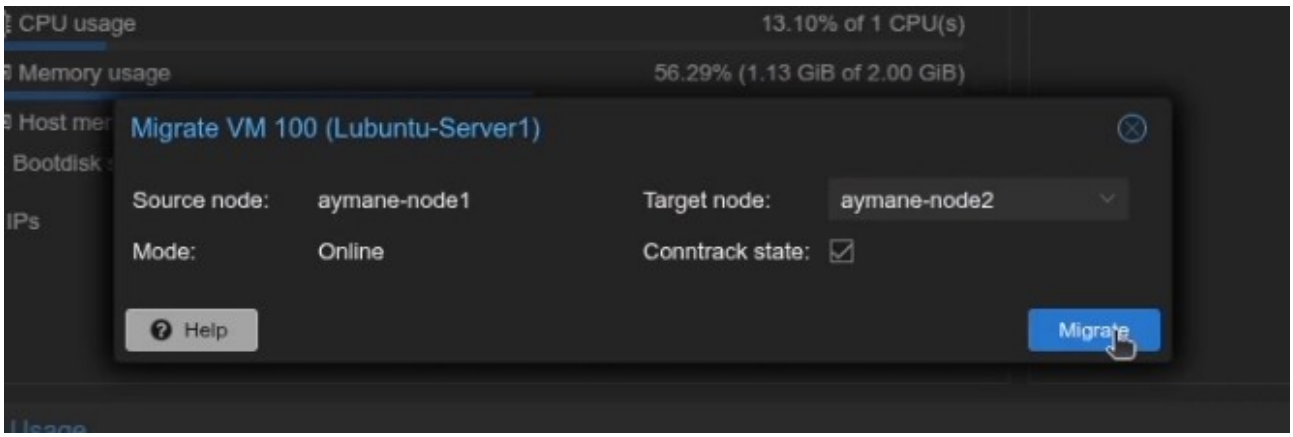


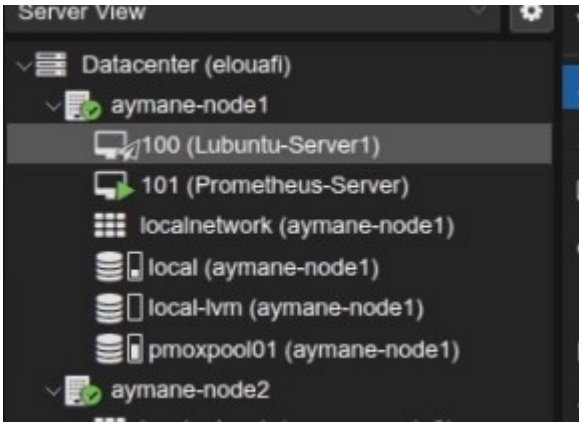
2. The VM was formally added to the Proxmox High Availability (HA) Manager resources, instructing the cluster to automatically manage its state.

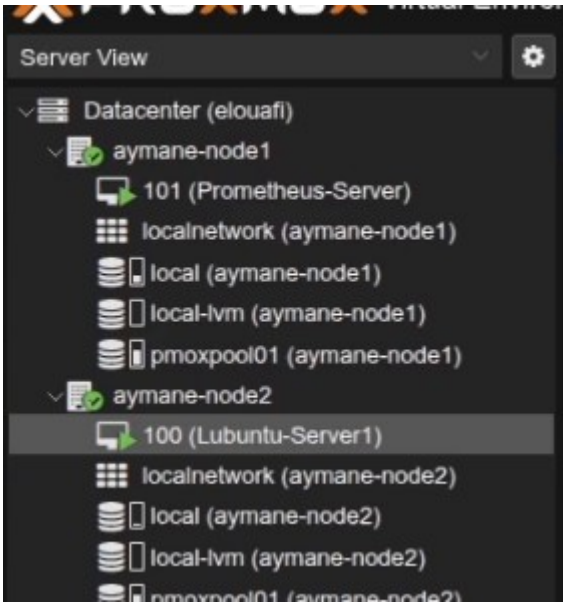


3.Migration Test :



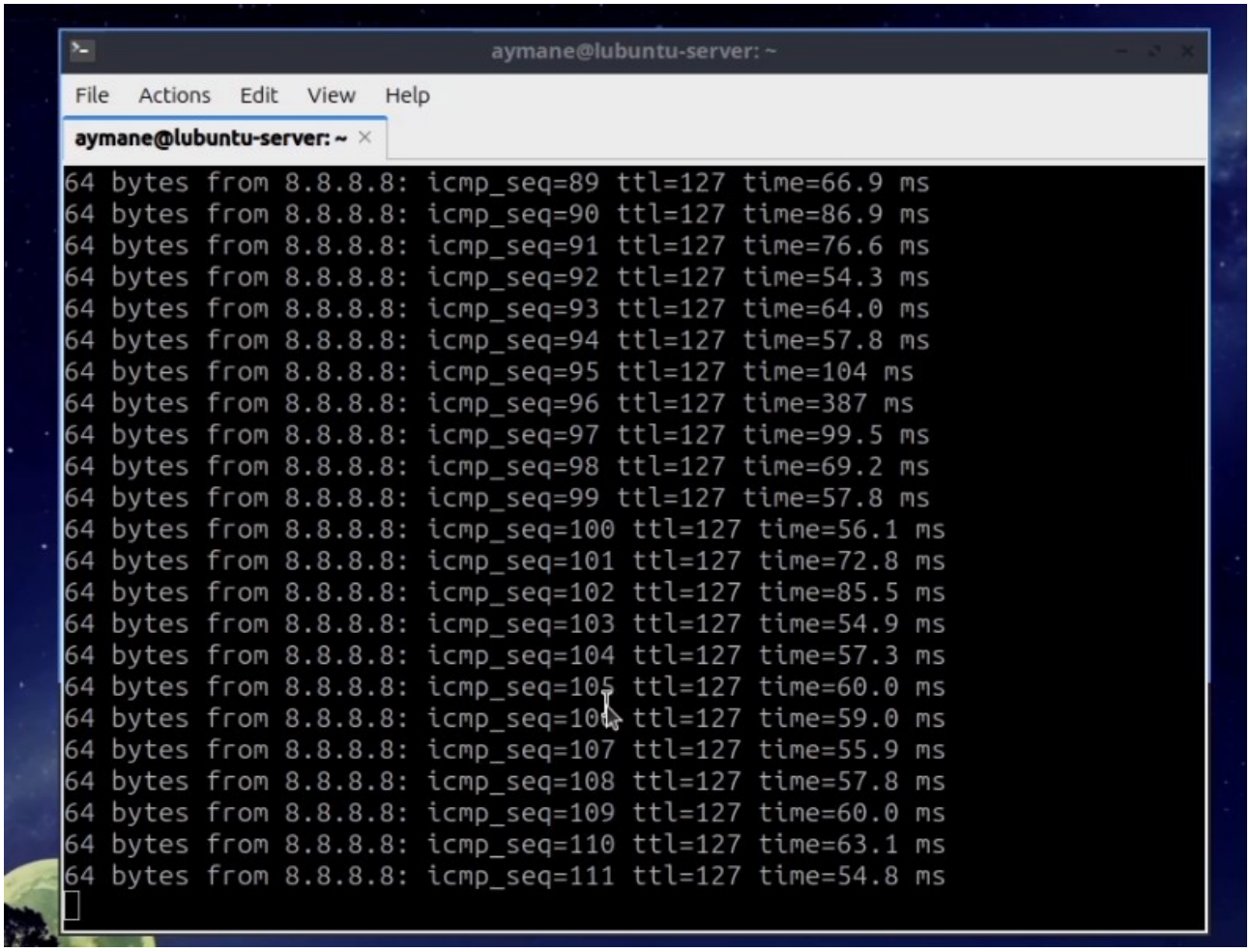






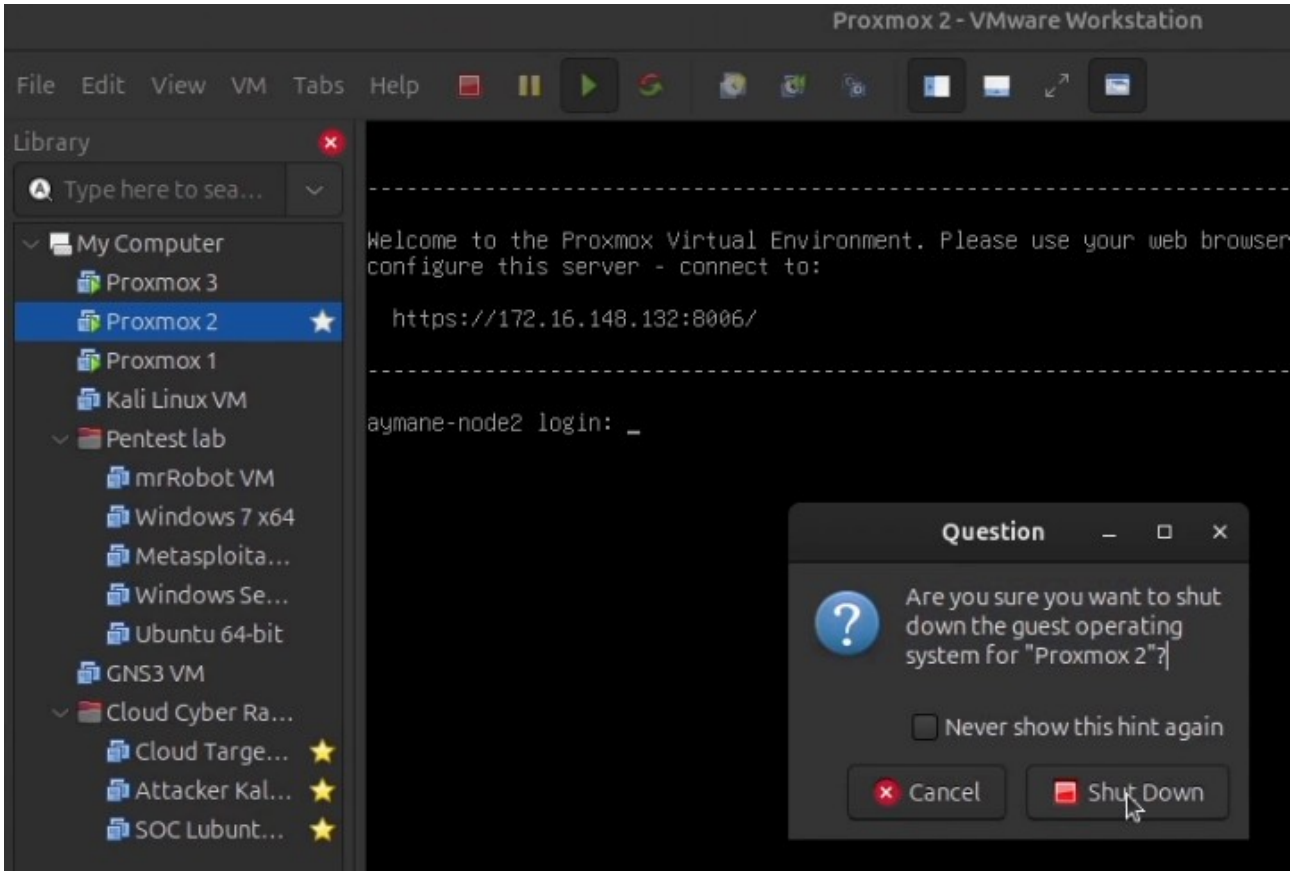
aymane-no…	root@pam	VM 100 - Start	OK
aymane-no…	root@pam	VM 100 - Migrate	OK
aymane-no…	root@pam	HA 100 - Migrate	OK

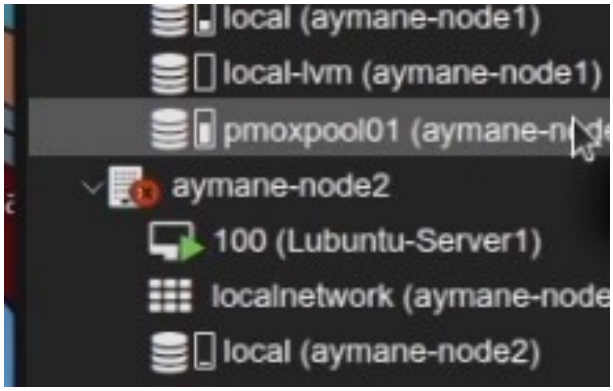
4.The ping returned consistent replies :



Phase 2: The Catastrophic Failure Simulation

With the network baseline established, a "hard power-off" was simulated on the host running the Web Server. **aymane-node2** (172.16.148.132) was abruptly shut down at the hypervisor level, simulating a sudden motherboard or power supply failure without allowing the operating system to perform a graceful shutdown.



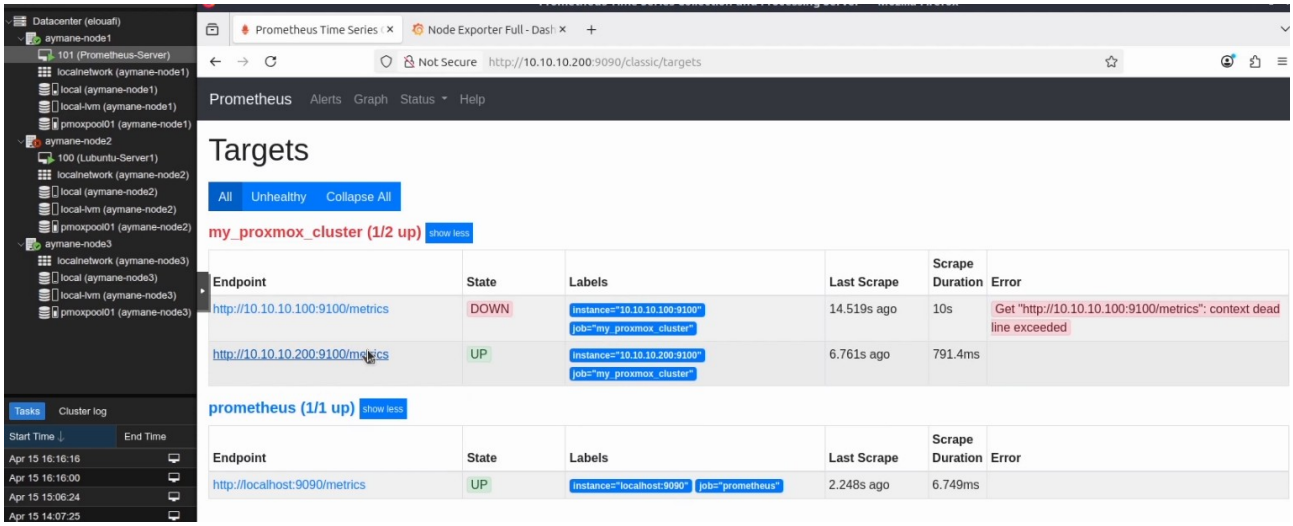


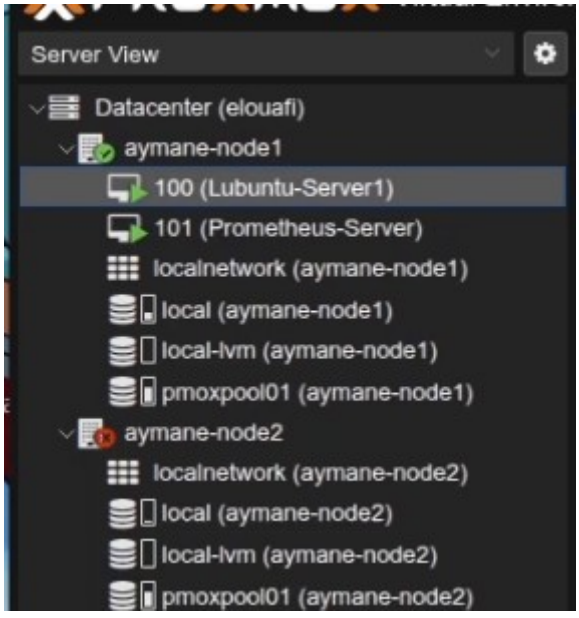
Phase 3: Cluster Response and Automatic Recovery

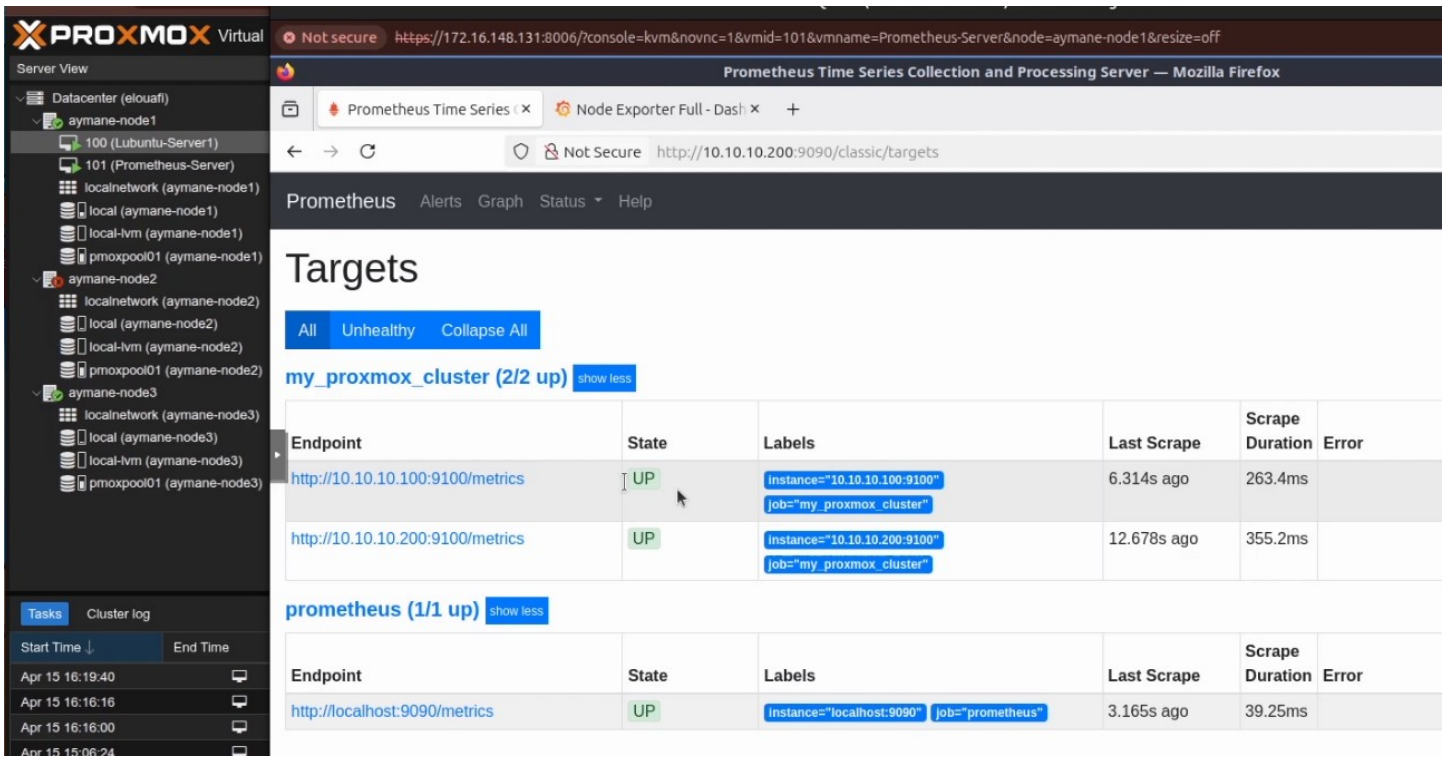
The infrastructure reacted to the node loss through a sequence of automated recovery steps:

1. **Heartbeat Loss Detection:** Within seconds, the Corosync cluster engine detected the loss of communication (heartbeat) from node2.

2. **Quorum Maintenance:** Because the architecture was designed with three nodes, the surviving nodes (aymane-node1 and node3) maintained a strict majority (2 out of 3 votes). The cluster remained fully operational, successfully preventing a split-brain scenario.
3. **Ceph Rebalancing:** The Ceph monitors detected the loss of the OSDs located on node2. Because the pmoxpool01 pool was configured with a replication size of 3, no data was lost. Ceph marked the missing OSDs as 'down' and continued serving I/O requests using the surviving replicas on Node 1 and Node 3.
4. **HA Resource Fencing and Relocation:** The Proxmox HA Manager recognized that the Web Server VM was no longer accessible on node2. To ensure data integrity, it logically "fenced" the failed node and automatically issued a start command for the Web Server VM on a surviving node (**aymane-node1**).







The Web Server was successfully restored on node1 without any manual intervention. Because the VM's virtual hard disk was securely stored on the Ceph cluster, it booted perfectly with no data corruption. Furthermore, because the vmbr1 VXLAN overlay seamlessly stretches across all nodes, the VM retained its exact IP address (10.10.10.100) upon booting on the new host. This test conclusively validates the fault tolerance, storage redundancy, and network mobility of the engineered architecture.

Conclusion :

The primary objective of this project was to design, implement, and validate a highly available, hyper-converged infrastructure capable of sustaining hardware failures while providing secure, software-defined network isolation. Through the successful deployment of a three-node Proxmox Virtual Environment cluster, these objectives were comprehensively met.

By replacing traditional centralized storage with **Ceph Software-Defined Storage**, the infrastructure successfully eliminated single points of failure at the storage layer. The distributed pmoxpool01 pool guaranteed strict data redundancy and provided the shared storage backbone necessary for seamless virtual machine mobility across the compute nodes.

The most significant engineering achievements of this project lie in the iterative resolution of complex networking constraints. Confronted with the MAC-spoofing security limitations of the nested VMware underlay, an intermediate NAT-based bridging solution (vmbr1) was initially deployed. While this successfully granted workloads external internet access, it resulted in strict node isolation, preventing cross-cluster VM communication.

To overcome this final architectural hurdle, a **VXLAN overlay network** was engineered. By utilizing MAC-in-UDP encapsulation, the architecture successfully stretched a secure Layer 2 broadcast domain across the routed Layer 3 physical nodes.

Furthermore, the integration of the **Prometheus and Grafana** observability stack ensured deep, real-time visibility into the performance telemetry of the virtualized workloads operating within this isolated tenant network.

Ultimately, the simulated hardware failure test validated the resilience of the entire architecture. Upon the abrupt shutdown of a cluster node, Corosync successfully maintained quorum, Ceph dynamically redirected storage I/O, and the High Availability manager automatically relocated and restored the Web Server VM with zero data corruption.