vSphere Project

VMware vSphere Infrastructure Implementation

Team Members:

Islam Yasser

Fatma Ahmed

Tasneem Adel

Supervisor:

Eng. Ekram AbdelWahab

Table Of Contents:

Project Overview

This documentation details the architecture, deployment, and configuration of a resilient, highly available, and automated VMware vSphere virtualization infrastructure. Designed to simulate an enterprise-grade datacenter environment.

The project focuses on resource optimization, continuous availability, and efficient virtual machine lifecycle management.

Key Infrastructure Highlights:

- **Centralized Compute Cluster:** The core environment is built upon a cluster of three VMware ESXi hosts, centrally administered and monitored via a vCenter Server appliance.
- **Optimized Network Topology:** Network traffic is physically and logically segregated using Standard Virtual Switches (vSwitches). Management and virtual machine data are isolated on vSwitch0 , while high-bandwidth storage (NFS) and live-migration (vMotion) traffic are dedicated to vSwitch1.
- **Shared Storage Backend:** A Red Hat Enterprise Linux (RHEL 9.2) server acts as the storage backend, providing centralized NFS datastores. This shared storage architecture enables cluster-wide features and hosts Content Libraries for standardized, rapid VM provisioning.
- **Automated Workload Balancing:** vSphere Distributed Resource Scheduler (DRS) is configured to ensure continuous, automated load balancing and optimal resource allocation across the cluster.

- **Disaster Recovery:** The infrastructure demonstrates robust fault tolerance and recovery capabilities. vSphere High Availability (HA) is enabled to automatically restart workloads in the event of a physical host failure. Furthermore, vSphere Fault Tolerance (FT) is implemented for critical workloads to ensure continuous availability with zero downtime.
- **Lifecycle Management:** The project covers end-to-end virtual machine administration, including live vMotion migrations, state preservation via snapshots, and the creation of standardized VM templates for scalable deployments.

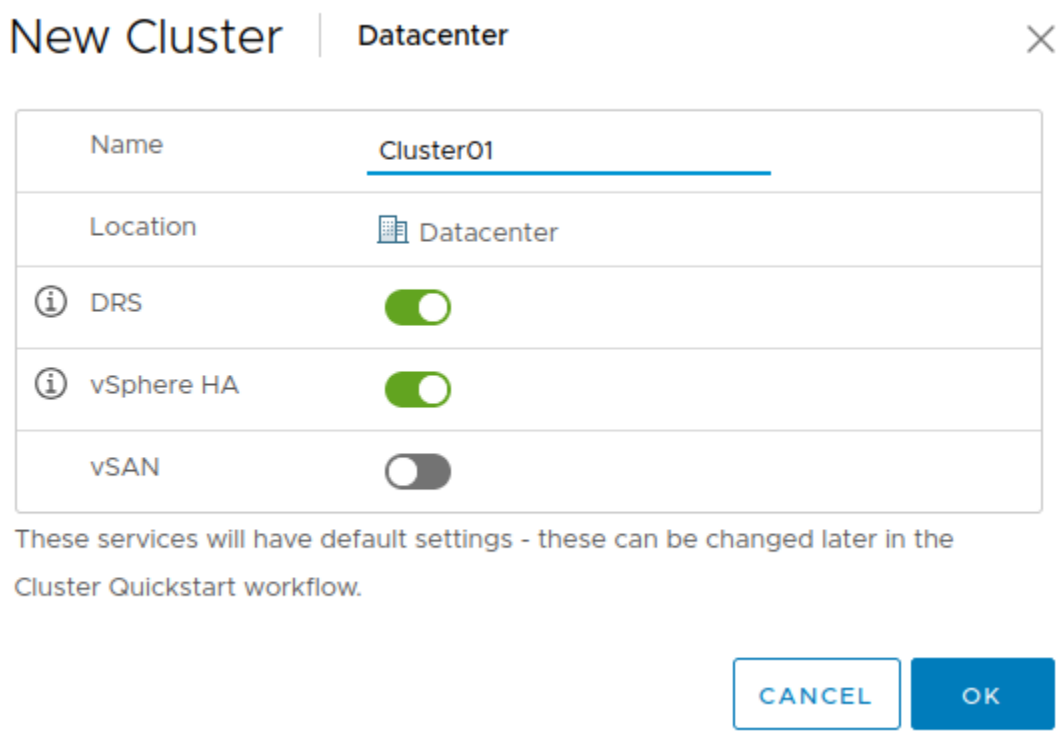
Infrastructure Setup

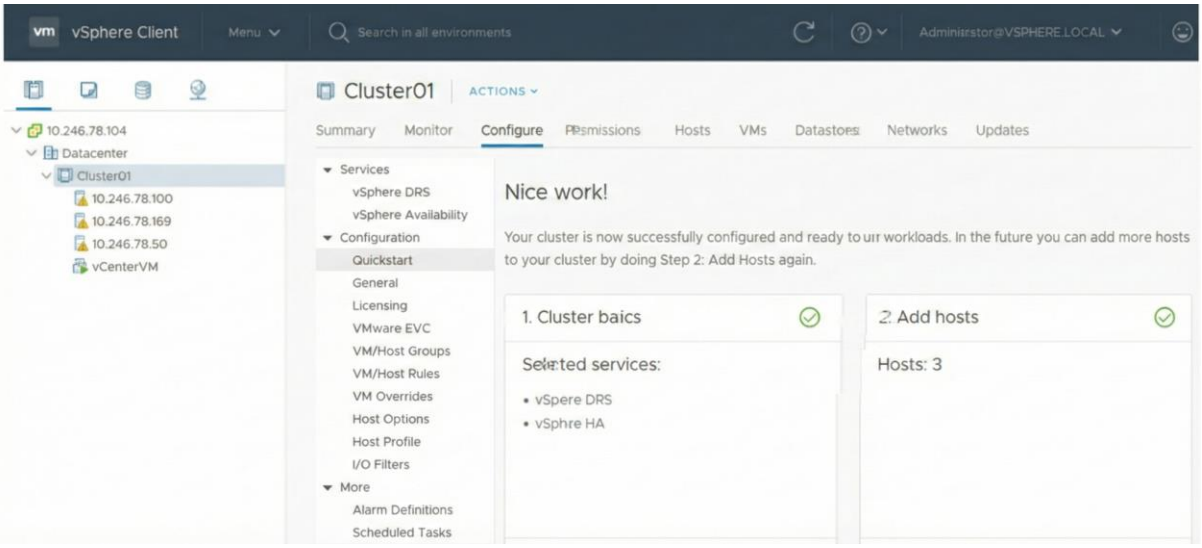
- **vCenter Server:** Configured with the management IP address `10.246.78.104`.
- **ESXi Hosts:** Three ESXi hosts were deployed to form the compute cluster.
  - **ESXi 1:** IP `10.246.78.50`.
  - **ESXi 2:** IP `10.246.78.100`.
  - **ESXi 3:** IP `10.246.78.169`.
- **Network Gateway:** All management interfaces share the default IPv4 gateway of `10.246.78.158` with a subnet mask of `255.255.255.0`.

Cluster Configuration

A cluster named `Cluster01` was established within the datacenter to pool the resources of the three ESXi hosts.

- **Both vSphere HA and vSphere DRS were successfully enabled on Cluster01.**





Networking Configuration

Network traffic was logically and physically separated using two Standard Virtual Switches (`vSwitch0` and `vSwitch1`) across all hosts.

vSwitch0 (Management & VM Traffic)

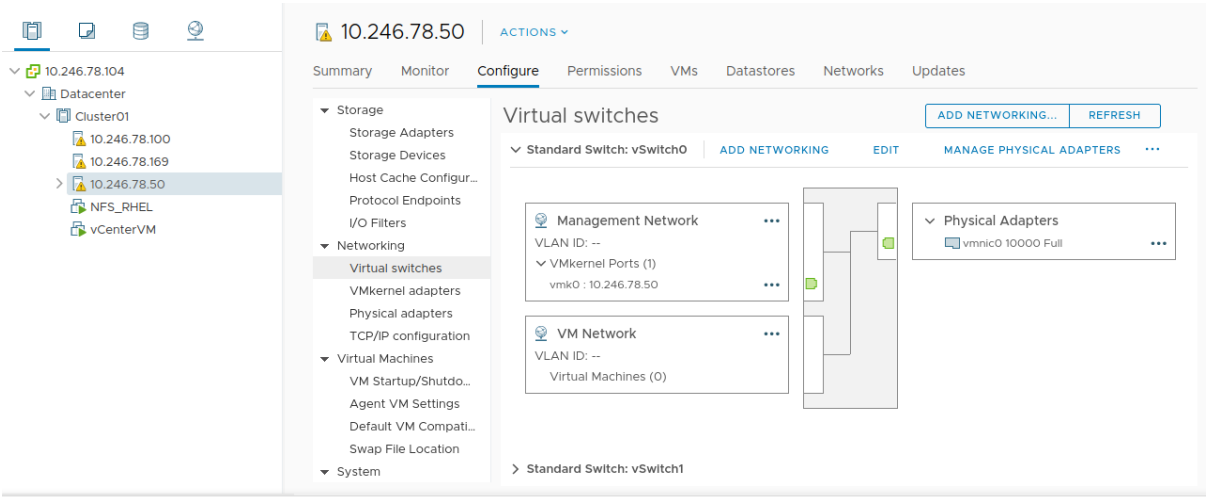
- **Physical Uplink:** Assigned to physical adapter `vmnic0`.
- **Port Groups:**
  - **Management Network:** Configured on `vmk0` with the respective host management IPs (`10.246.78.50`, `10.246.78.100`, `10.246.78.169`).
  - **VM Network:** Dedicated to virtual machine data traffic.

vSwitch1 (Storage & vMotion Traffic)

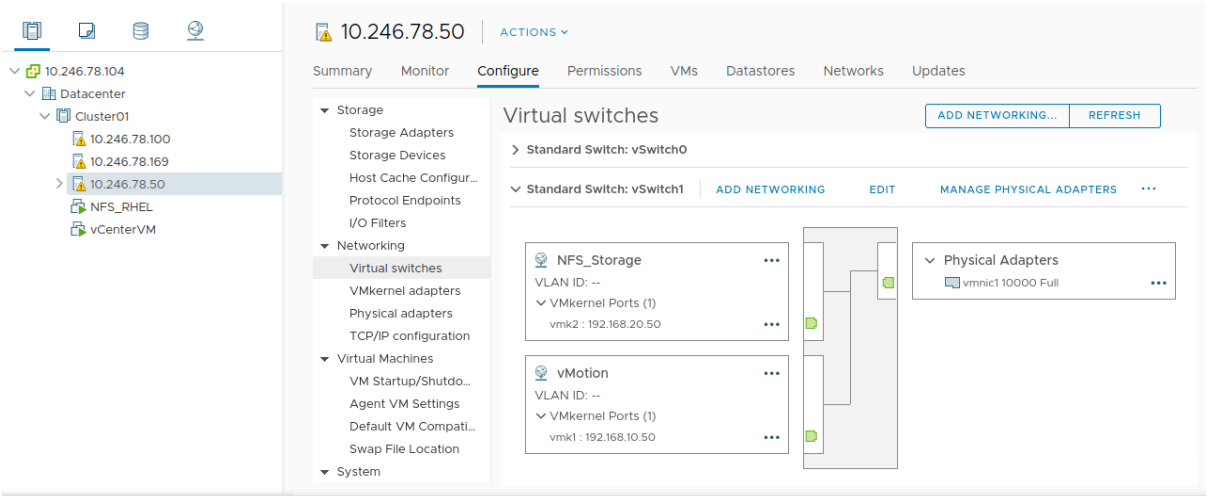
- **Physical Uplink:** Assigned to physical adapter `vmnic1`.
- **Port Groups & VMkernel Adapters:**
  - **NFS_Storage (`vmk2`):** Used to connect to the shared storage backend. The IP addresses were set to `192.168.20.x` and subnet 255.255.255.0 (`192.168.20.50`, `192.168.20.100`, `192.168.20.169`). The MTU was set to the default `1500`.
  - **vMotion (`vmk1`):** Used for live migration of virtual machines. Configured on the `192.168.10.x` subnet 255.255.255.0 (`192.168.10.50`, `192.168.10.100`, `192.168.10.169`). Enabled services on this port included **vMotion** and **Provisioning**. The MTU was set to `1500`.

ESXI 10.246.78.50:

vSwitch0:

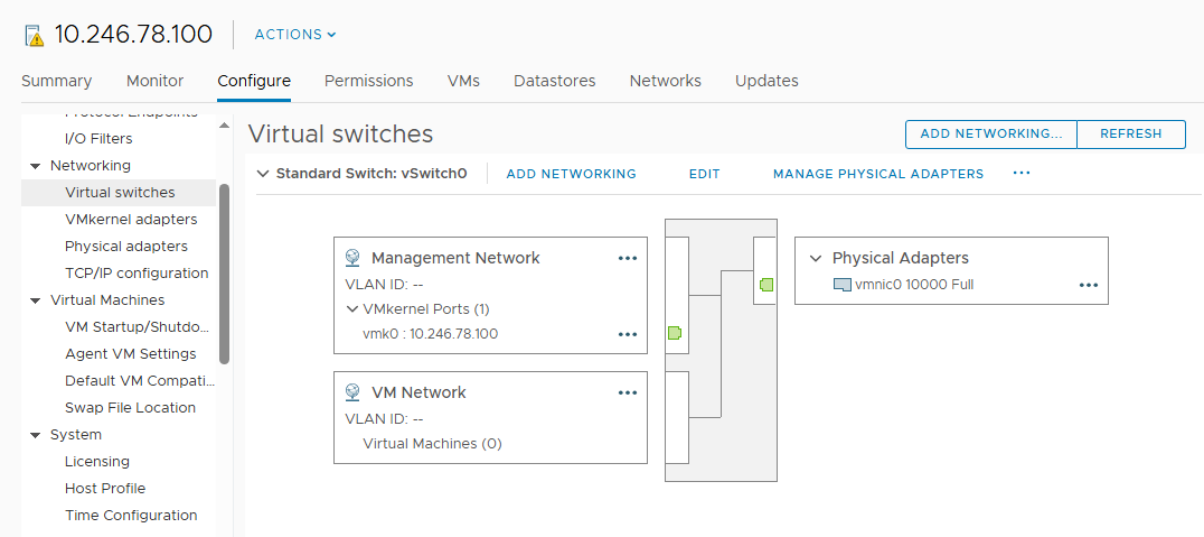


vSwitch1:

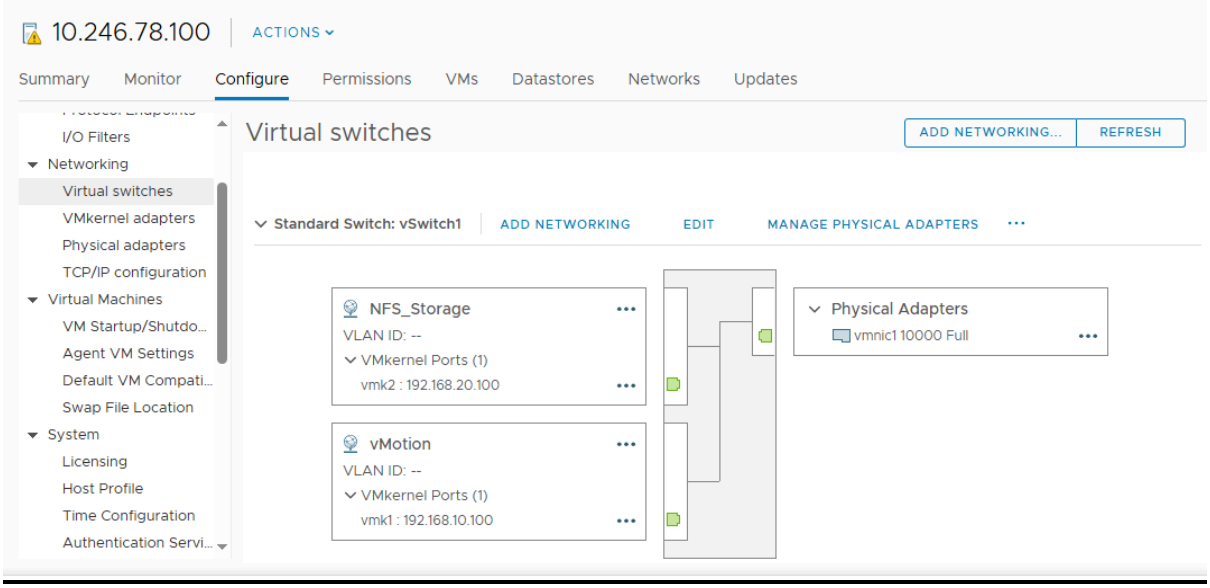


ESXI 10.246.78.100:

vSwitch0:

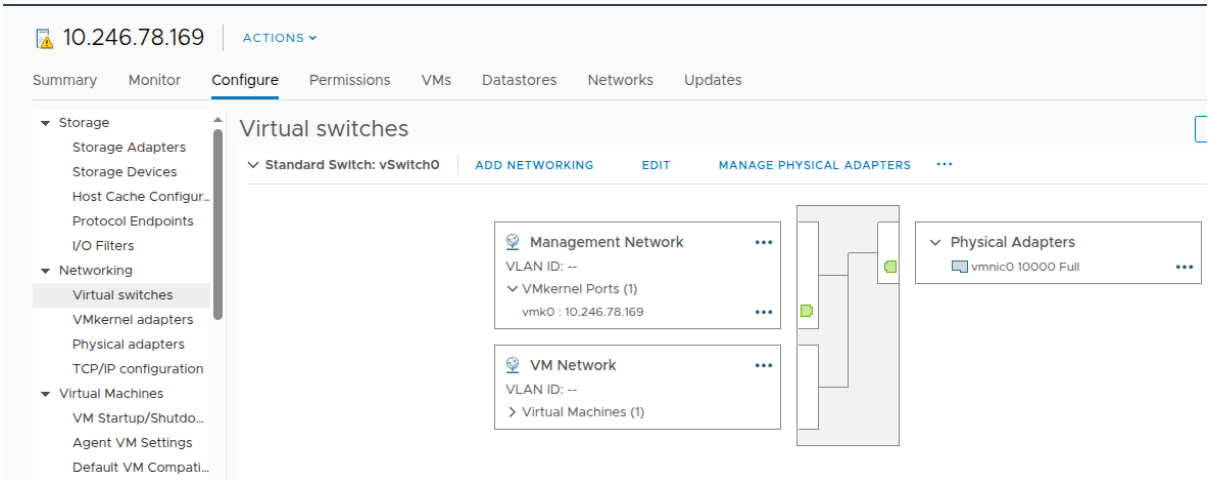


vSwitch1:

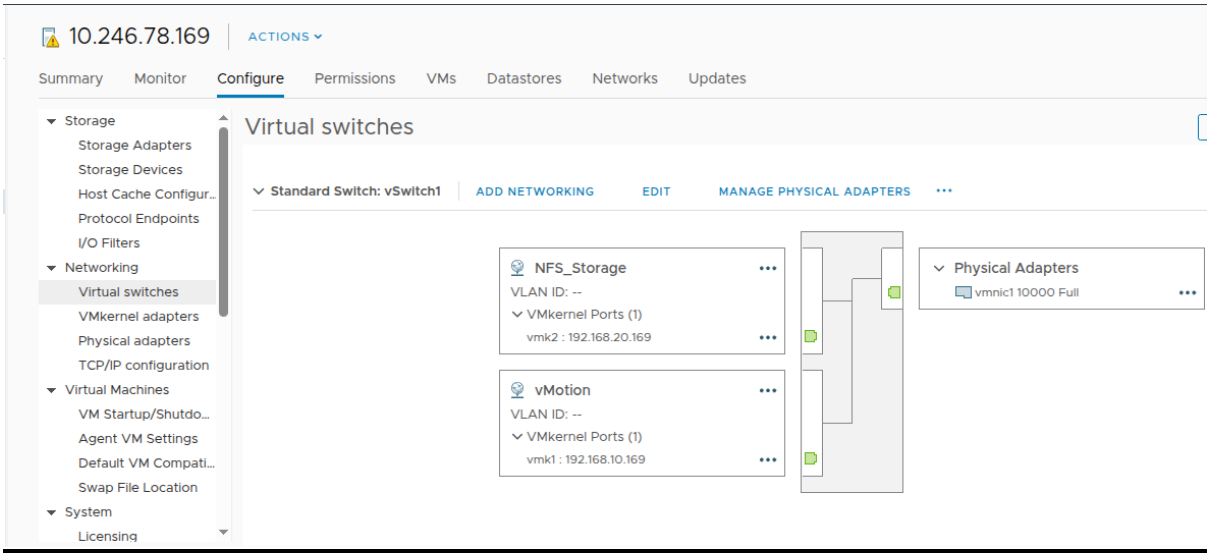


ESXI 10.246.78.169:

vSwitch0:

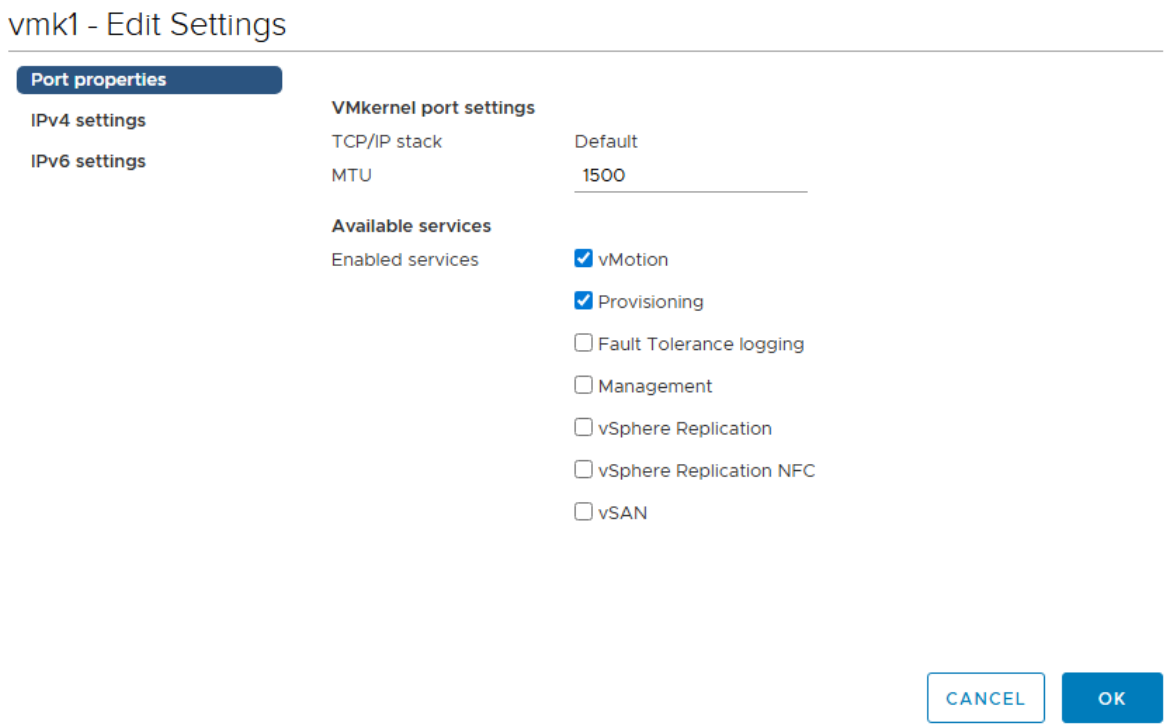


vSwitch1:



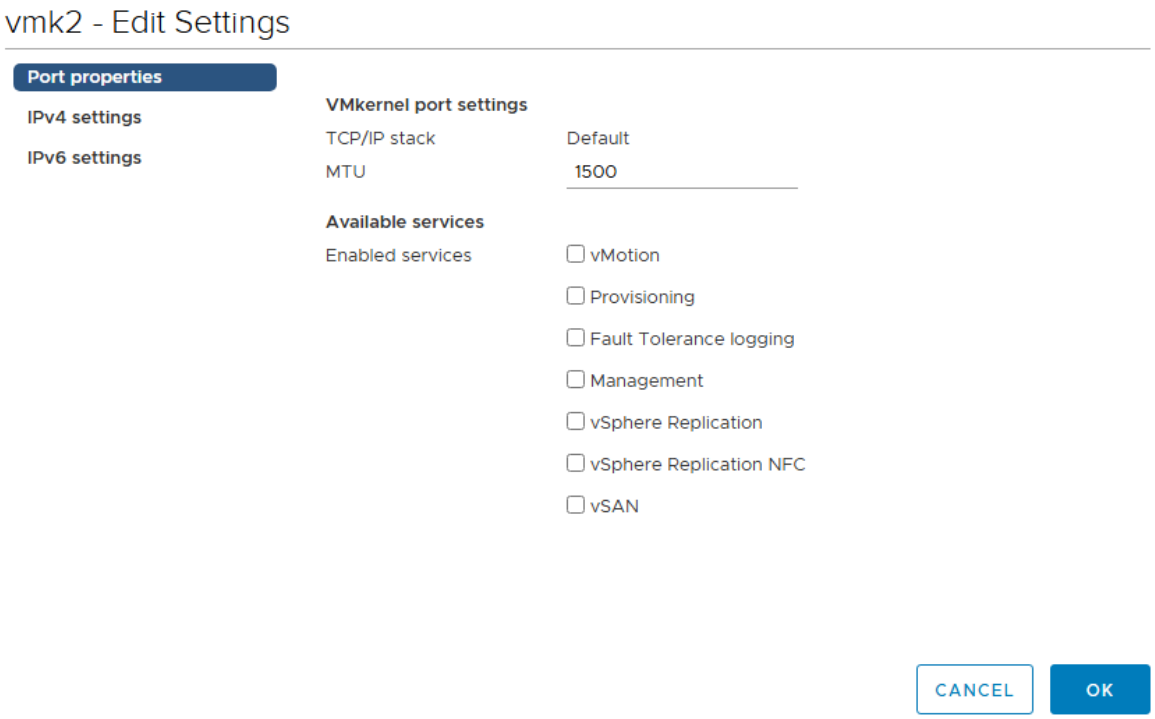
vMotion:

The two vSwitches have been configured for all ESXI's hosts.



NFS_Storage:

The two vSwitches have been configured for all ESXI's hosts.

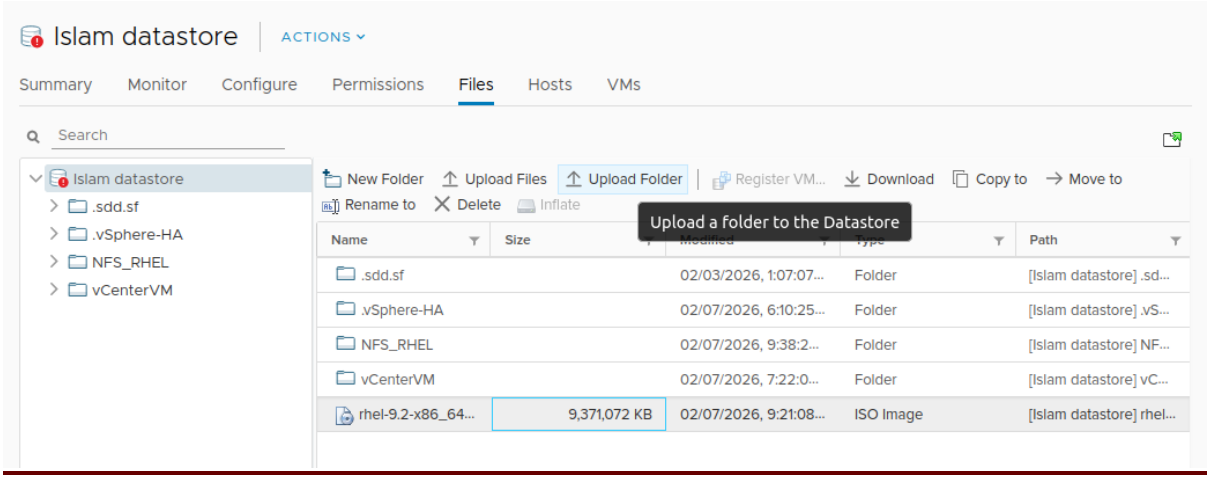


Storage Infrastructure

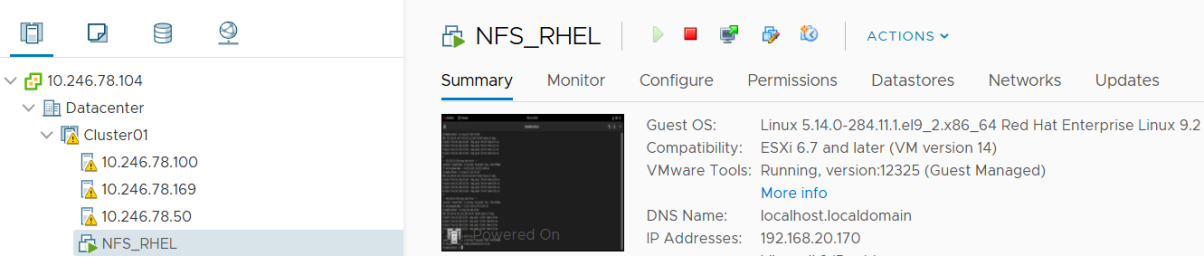
The environment relies on a centralized NFS server to provide shared storage capabilities necessary for cluster services like HA and DRS.

- **NFS Server Details:** A Red Hat Enterprise Linux (RHEL) 9.2 virtual machine (`NFS_RHEL`) was deployed as the storage backend. It was assigned the IP address `192.168.20.170`.
- **Connectivity Verification:** Successful ICMP ping tests were executed from the RHEL server to the `vmk2` IP interfaces of all three ESXi hosts, confirming routing and connectivity on the storage network.
- **Datastores Mounted:**
    - `Islam datastore`: A local datastore utilized to host the RHEL 9.2 installation ISO (`rhel-9.2-x86_64`, sized at approximately 9.3 GB).
    - `NFS_Shared_Storage`: The primary shared datastore mounted across the cluster.

Uploading the RHEL .Iso image into the Datastore



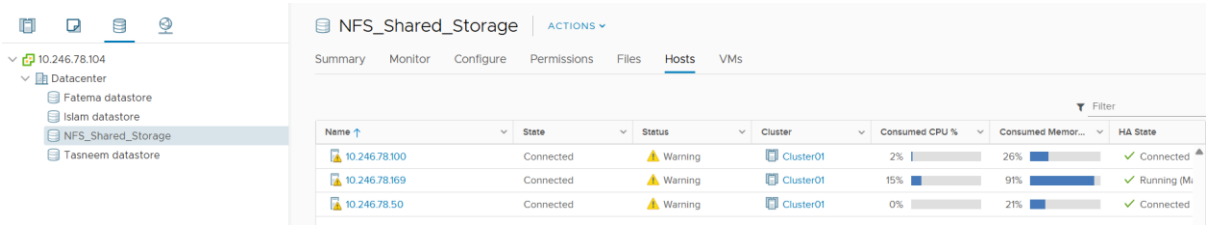
RHEL Installation for NFS Storage:



Pinging from NFS server to all ESXI's hosts

```
[islam@localhost ~]$ ping 192.168.20.100
PING 192.168.20.100 (192.168.20.100) 56(84) bytes of data.
64 bytes from 192.168.20.100: icmp_seq=1 ttl=64 time=8.73 ms
64 bytes from 192.168.20.100: icmp_seq=2 ttl=64 time=6.33 ms
64 bytes from 192.168.20.100: icmp_seq=3 ttl=64 time=11.8 ms
64 bytes from 192.168.20.100: icmp_seq=4 ttl=64 time=9.92 ms
^C
--- 192.168.20.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 6.332/9.201/11.819/1.988 ms
[islam@localhost ~]$ ping 192.168.20.169
PING 192.168.20.169 (192.168.20.169) 56(84) bytes of data.
64 bytes from 192.168.20.169: icmp_seq=1 ttl=64 time=0.734 ms
64 bytes from 192.168.20.169: icmp_seq=2 ttl=64 time=0.547 ms
64 bytes from 192.168.20.169: icmp_seq=3 ttl=64 time=0.741 ms
64 bytes from 192.168.20.169: icmp_seq=4 ttl=64 time=0.512 ms
^C
--- 192.168.20.169 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.512/0.633/0.741/0.104 ms
[islam@localhost ~]$ ping 192.168.20.50
PING 192.168.20.50 (192.168.20.50) 56(84) bytes of data.
64 bytes from 192.168.20.50: icmp_seq=1 ttl=64 time=5.40 ms
64 bytes from 192.168.20.50: icmp_seq=2 ttl=64 time=28.6 ms
64 bytes from 192.168.20.50: icmp_seq=3 ttl=64 time=8.75 ms
64 bytes from 192.168.20.50: icmp_seq=4 ttl=64 time=9.58 ms
^C
--- 192.168.20.50 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 5.398/13.090/28.637/9.111 ms
[islam@localhost ~]$
```

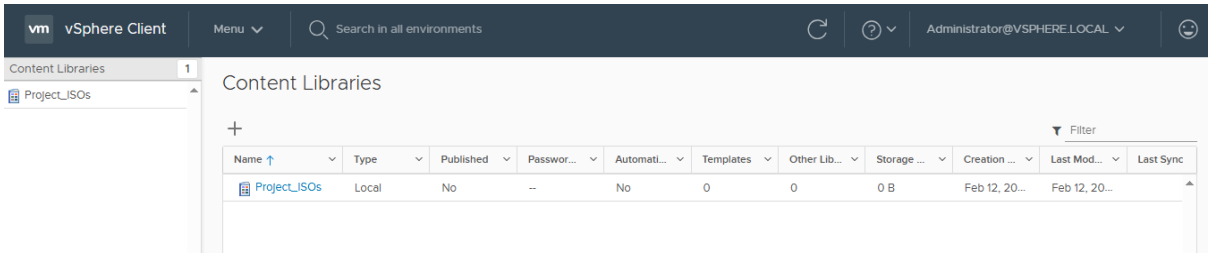
Shared DataCenter:



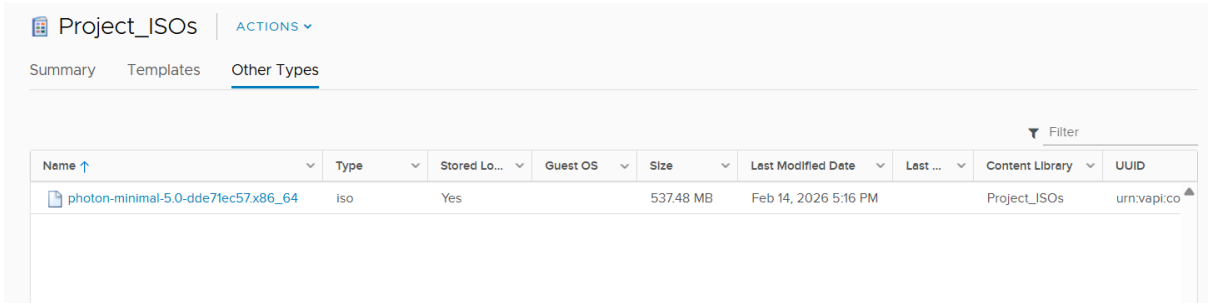
Content Library:

- A content library named `Project_ISOs` was created.
- A Photon OS installation medium (`photon-minimal-5.0...`, sized at 537.48 MB) was uploaded to this library for rapid VM deployment.

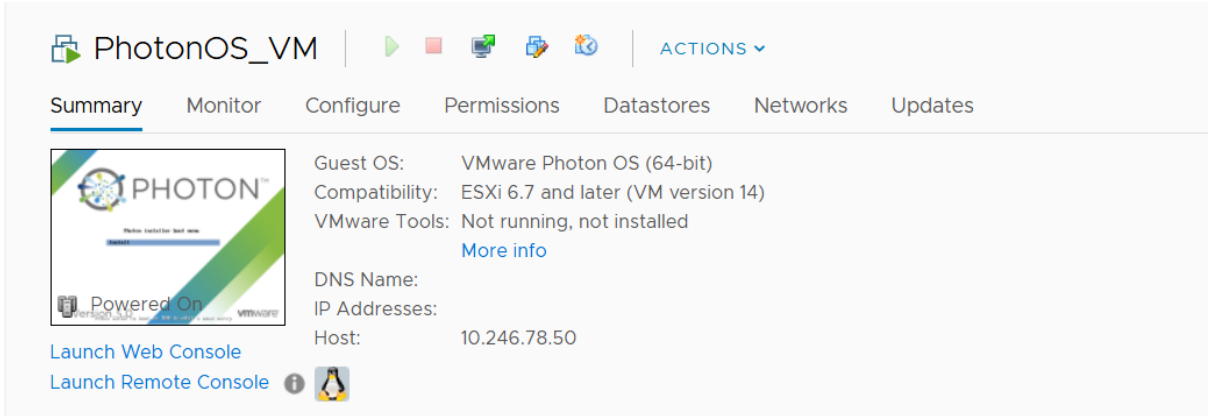
Content Library Creation:



Uploading Iso image to content library



Install Photon OS:



VMware Tools Installation and Verification

Ensuring VMware Tools (or Open VM Tools) are installed and running is a critical step in VM deployment, as it provides essential drivers and enables vCenter to manage the guest operating system effectively. The project involved checking and verifying these tools across different guest operating systems.

Red Hat Enterprise Linux 9.2 (NFS Server):

For the basic virtual machine running RHEL 9.2, which serves as the NFS storage backend, an explicit installation and verification process was documented.

- **Installation Attempt:** The package manager was utilized via the command line to attempt the installation using the command `sudo dnf install open-vm-tools`.
- **Service Verification:** To ensure the toolset was actively functioning in the background, the command `sudo systemctl status vmtoolsd` was executed.
- **Service Status:** The output confirmed that the `vmtoolsd.service` (Service for virtual machines hosted on VMware) was loaded, enabled, and active (running).

```
[islam@localhost yum.repos.d]$ sudo dnf install open-vm-tools
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscription-manager to register.

RHEL 9 Base OS                                                  5.2 MB/s | 1.7 MB     00:00
RHEL 9 AppStream                                                 32 MB/s | 6.3 MB     00:00
Package open-vm-tools-12.1.5-1.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

```
[islam@localhost yum.repos.d]$ sudo systemctl status vmtoolsd
● vmtoolsd.service - Service for virtual machines hosted on VMware
     Loaded: loaded (/usr/lib/systemd/system/vmtoolsd.service; enabled; preset: enabled)
     Active: active (running) since Thu 2026-02-12 15:26:06 EET; 32min ago
       Docs: http://github.com/vmware/open-vm-tools
   Main PID: 867 (vmtoolsd)
      Tasks: 4 (limit: 10973)
     Memory: 5.1M
        CPU: 3.055s
     CGroup: /system.slice/vmtoolsd.service
             └─867 /usr/bin/vmtoolsd

Feb 12 15:26:06 localhost systemd[1]: Started Service for virtual machines hosted on VMware.
[islam@localhost yum.repos.d]$
```

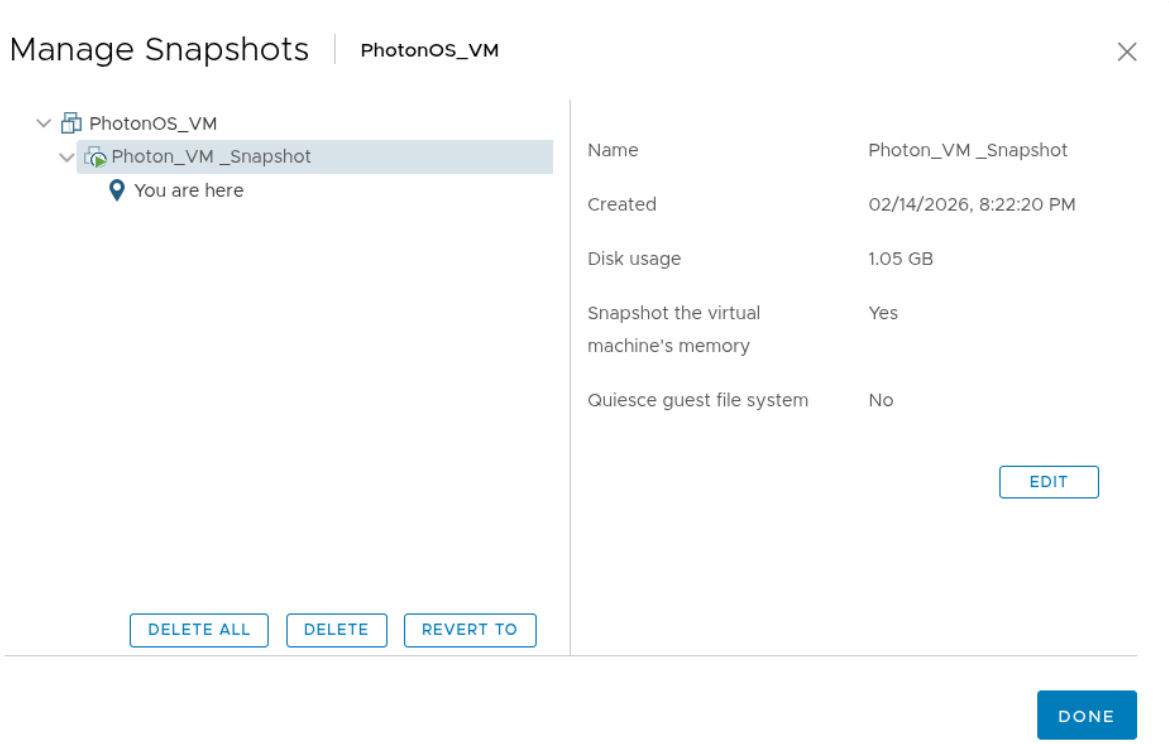
VMware Photon OS Virtual Machines:

- Because Photon OS is optimized for vSphere, VMware Tools are baked into the OS automatically.
- Immediately following the deployment of `Photon0S_VM`, the vSphere Client briefly showed the tools as Not running, not installed prior to the OS completing its first full boot sequence.
- Shortly after, the status automatically updated to Running, without any manual installation commands being executed.

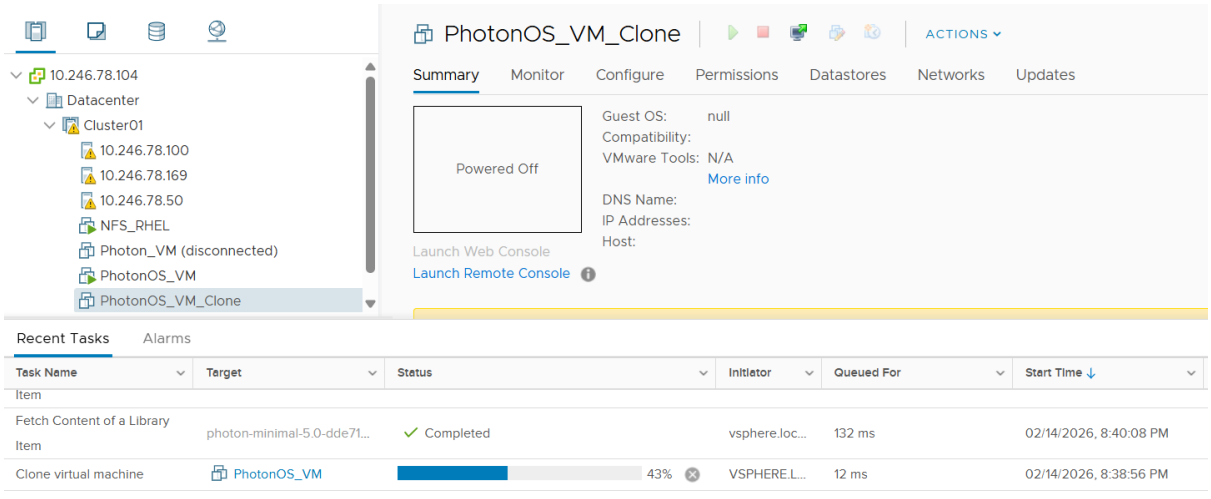
Virtual Machine Operations & Lifecycle Management

Extensive testing of VM lifecycle and mobility features was conducted using the lightweight `PhotonOS_VM`.

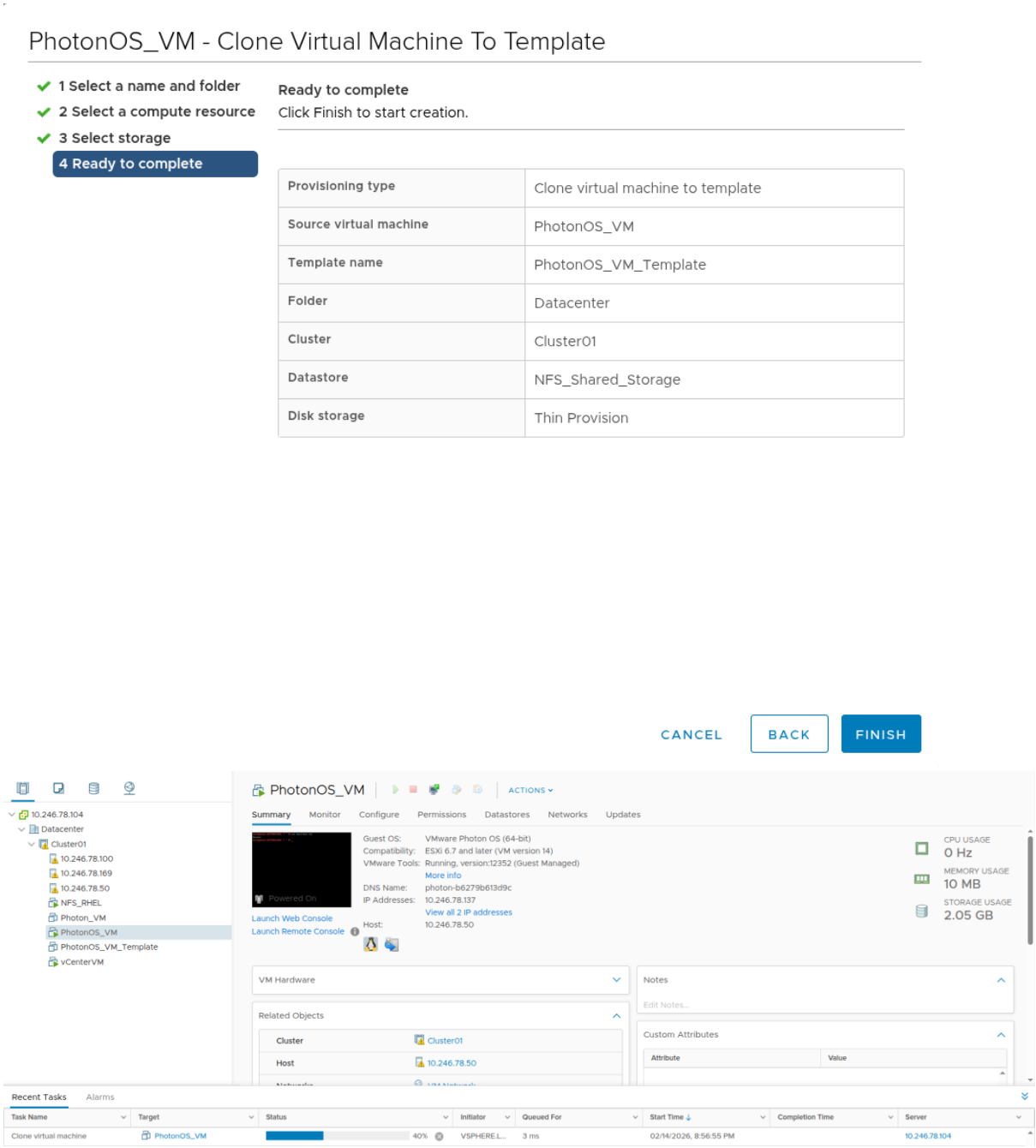
- **Snapshots:** A snapshot named `Photon_VM_Snapshot` was subsequently taken, capturing 1.05 GB of disk usage and preserving the virtual machine's memory state.



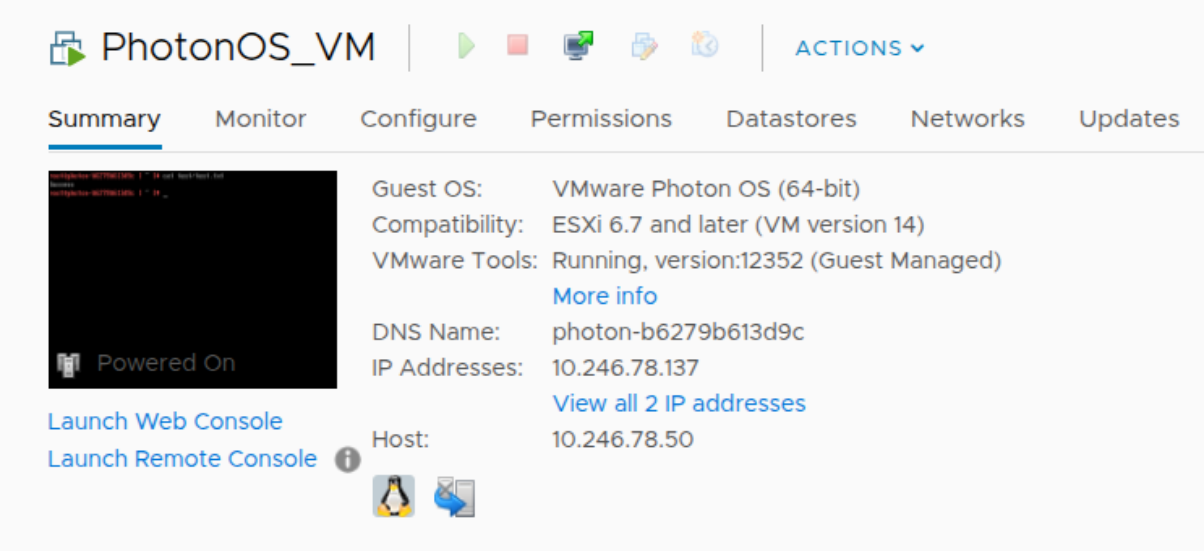
- **Cloning:** The VM was successfully cloned to a new instance named `PhotonOS_VM_Clone`.

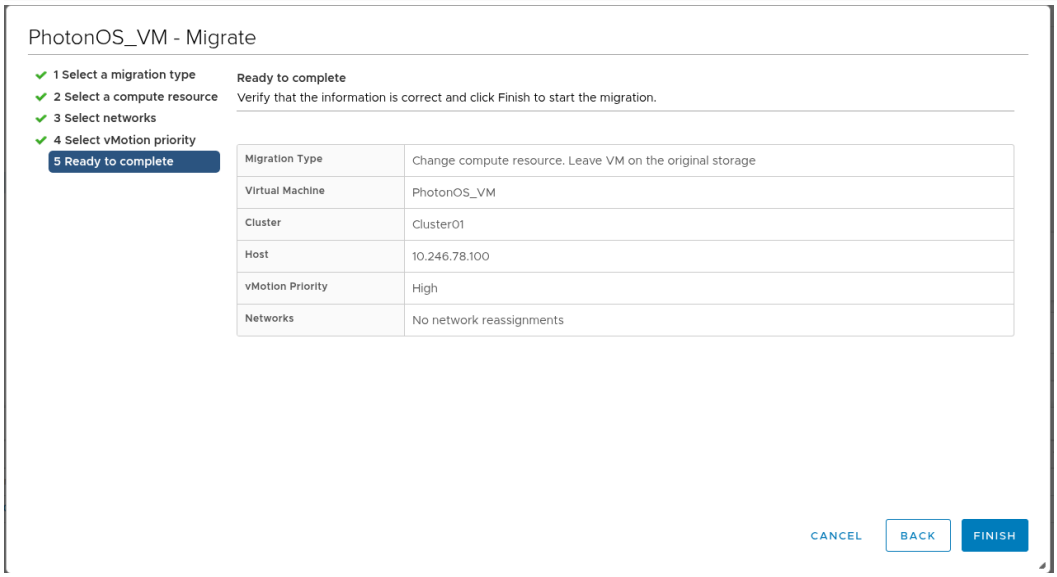


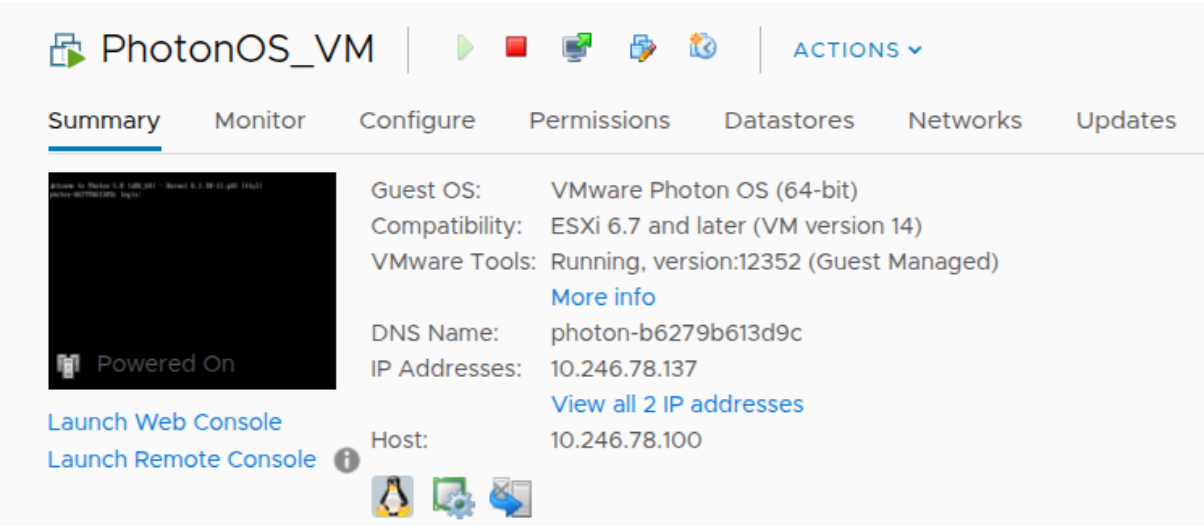
- **Templates:** To standardize future deployments, the VM was cloned into a template named `PhotonOS_VM_Template`. This template was stored on `NFS_Shared_Storage` utilizing Thin Provisioning.



- **Live Migration (vMotion):** `PhotonOS_VM` was successfully live-migrated from host `10.246.78.50` to host `10.246.78.100`. The migration priority was set to "High".



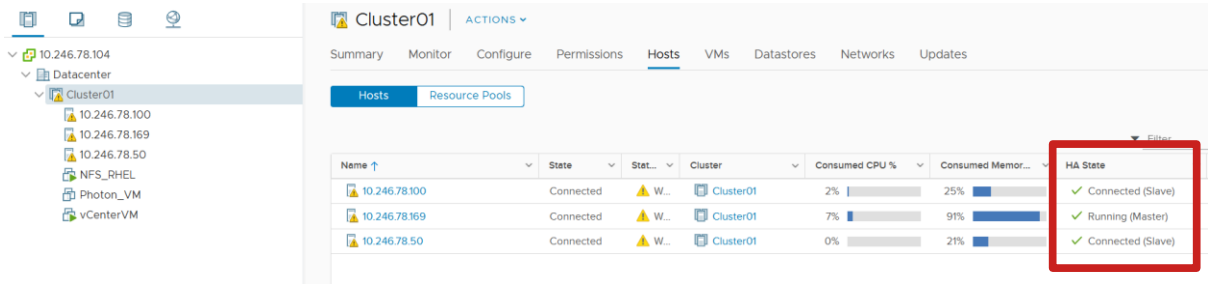




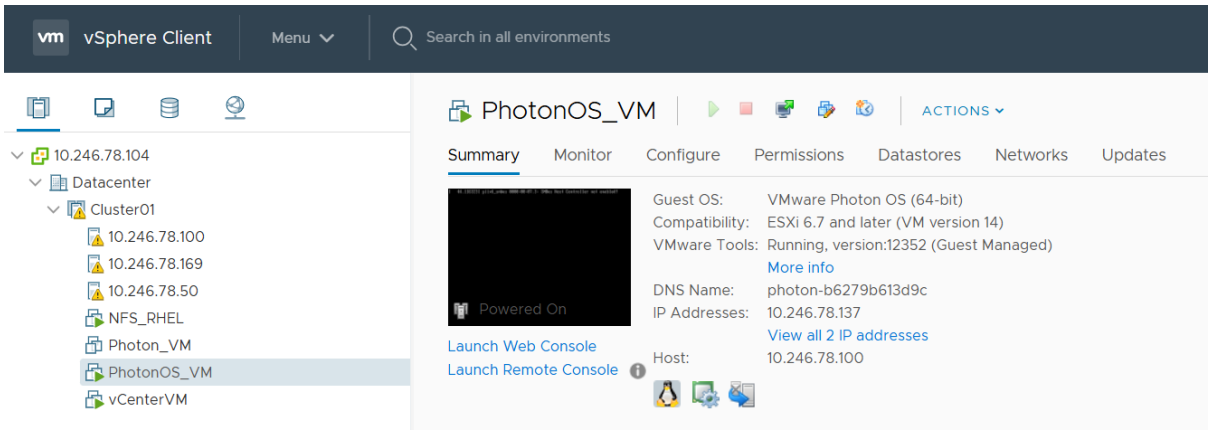
vSphere High Availability (vSphere HA)

During the validation phases, several specific behaviors and simulated failures were tested:

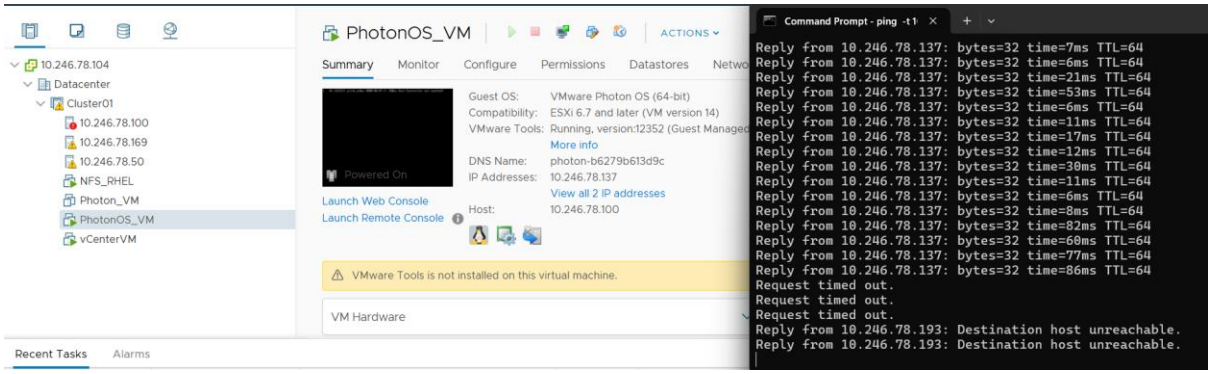
- After Enabled vSphere HA The election finished and there is the master and slaves.



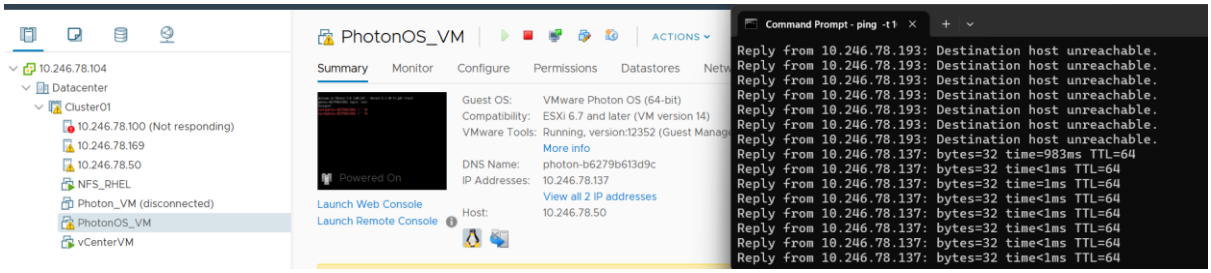
- **HA Failover Test and VM Downtime:** To test HA functionality, a crash was simulated on host `10.246.78.100`, which was actively running `PhotonOS_VM`.



- The host entered a Not responding state. During this time, continuous ping requests to the VM dropped, showing *Request timed out* and *Destination host unreachable*.



- vSphere HA detected the host failure and automatically restarted `PhotonOS_VM` on a surviving host (`10.246.78.50`).
- Once the VM rebooted, ping replies stabilized, confirming successful automated disaster recovery.

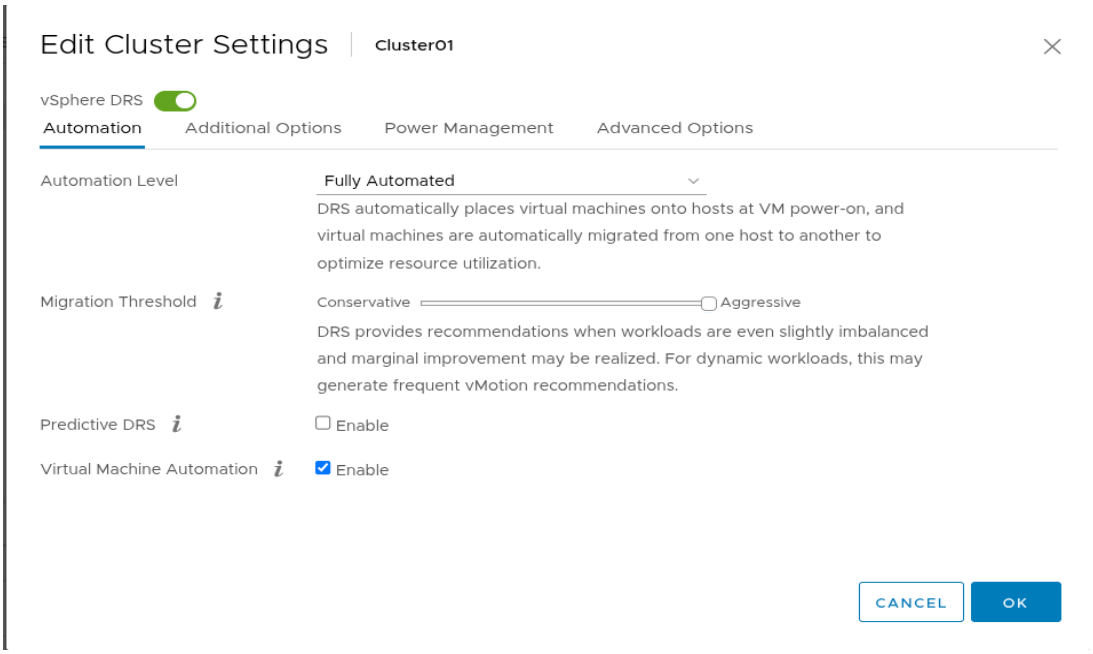


vSphere Distributed Resource Scheduler (DRS)

vSphere DRS was enabled during the initial configuration of `Cluster01`, alongside vSphere HA, to ensure balanced resource allocation and optimize the performance of the virtual machines across the ESXi hosts.

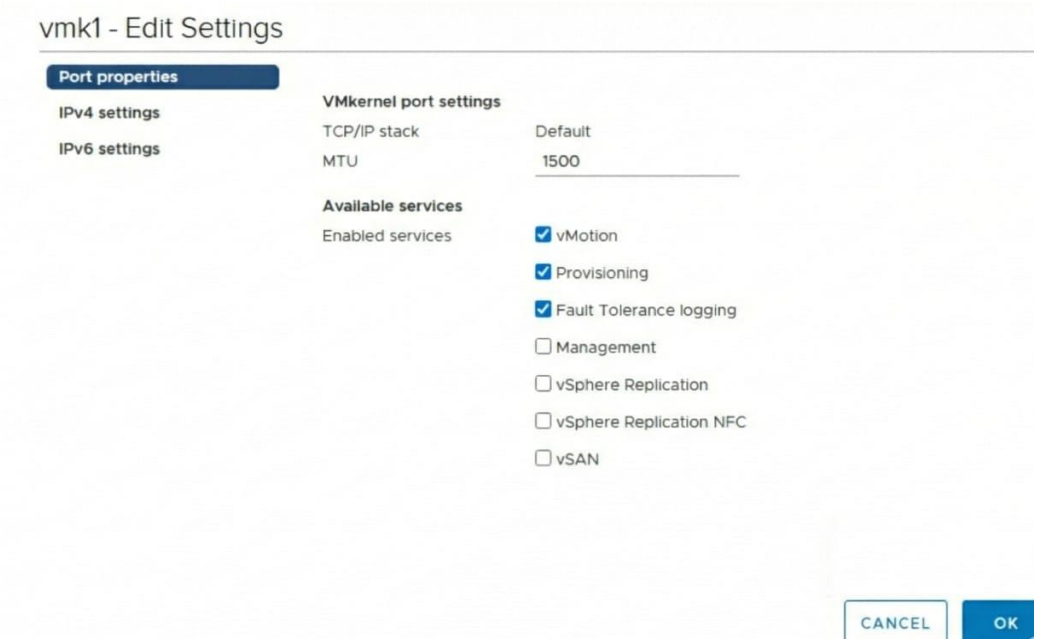
The cluster settings were specifically tuned to handle workload distribution without requiring manual intervention:

- **Automation Level**: The automation level was set to Fully Automated. Under this setting, DRS handles operations without manual prompts in two key ways:
  - Initial Placement: It automatically places virtual machines onto the most optimal hosts when they are powered on.
  - Load Balancing: It automatically live-migrates virtual machines from one host to another to continuously optimize resource utilization across the cluster.
- **Migration Threshold:** The migration threshold was configured to be Aggressive. With this high-sensitivity setting, DRS provides recommendations when workloads are even slightly imbalanced and marginal improvement may be realized. For dynamic workloads, this aggressive stance may generate frequent vMotion recommendations.
- **Additional Automation Options:**
  - Virtual Machine Automation: This setting was explicitly enabled to allow DRS to apply automation at the individual VM level.
  - Predictive DRS: This advanced feature was left disabled.

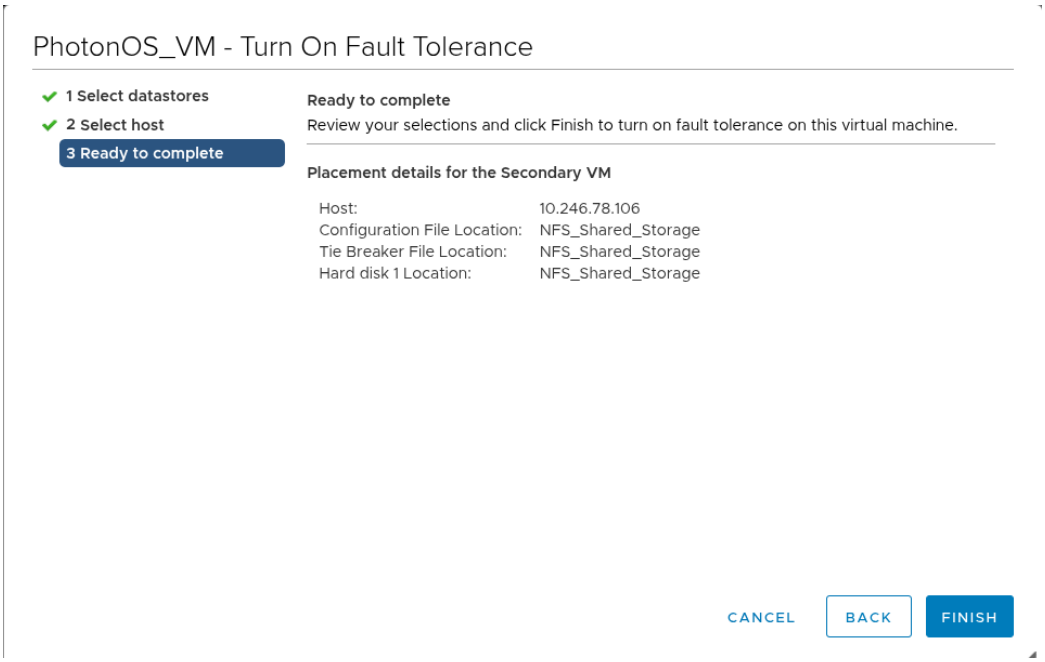


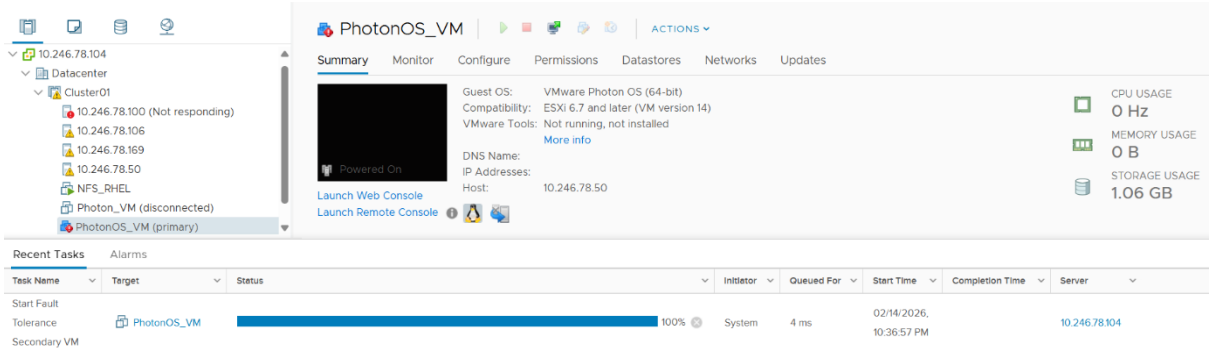
vSphere Fault Tolerance (vSphere FT)

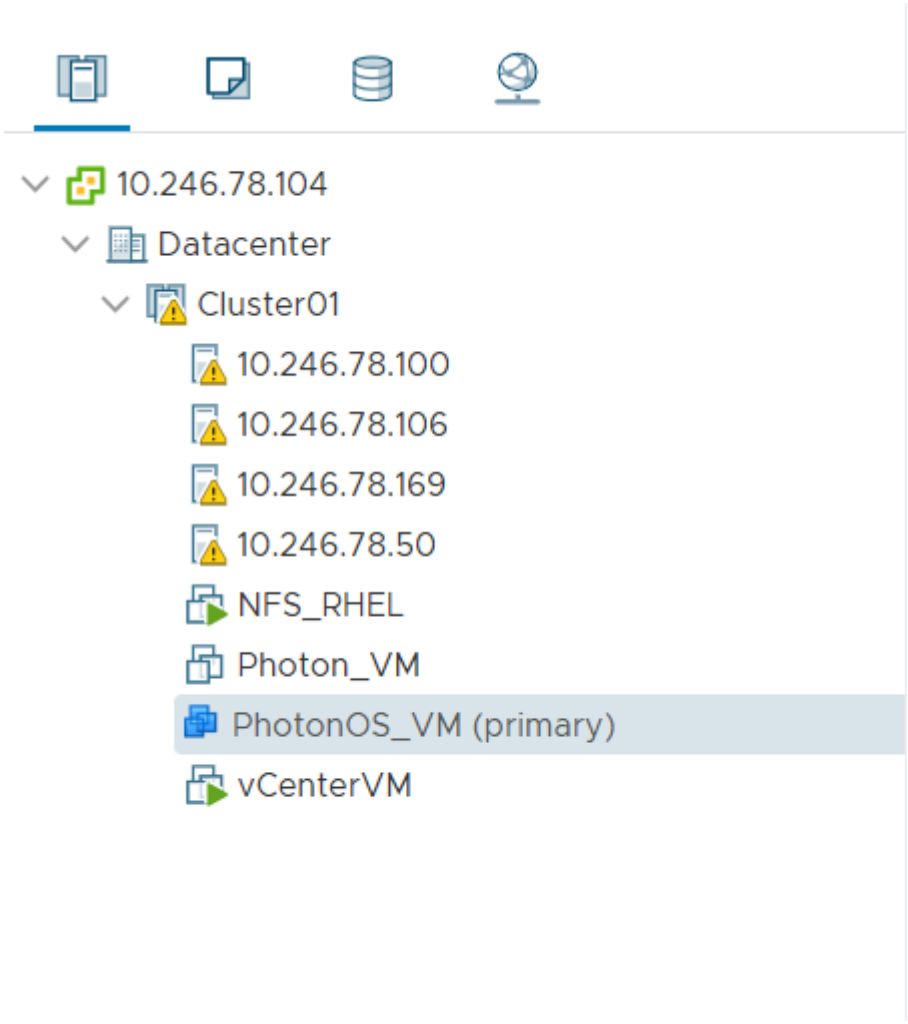
- **On the real lab environment we were working on:**
  - Zero-downtime protection was configured for `PhotonOS_VM`. Prior to turning on vSphere FT, the Fault Tolerance logging service was explicitly enabled on the `vmk1` VMkernel adapter on vSwitch1.



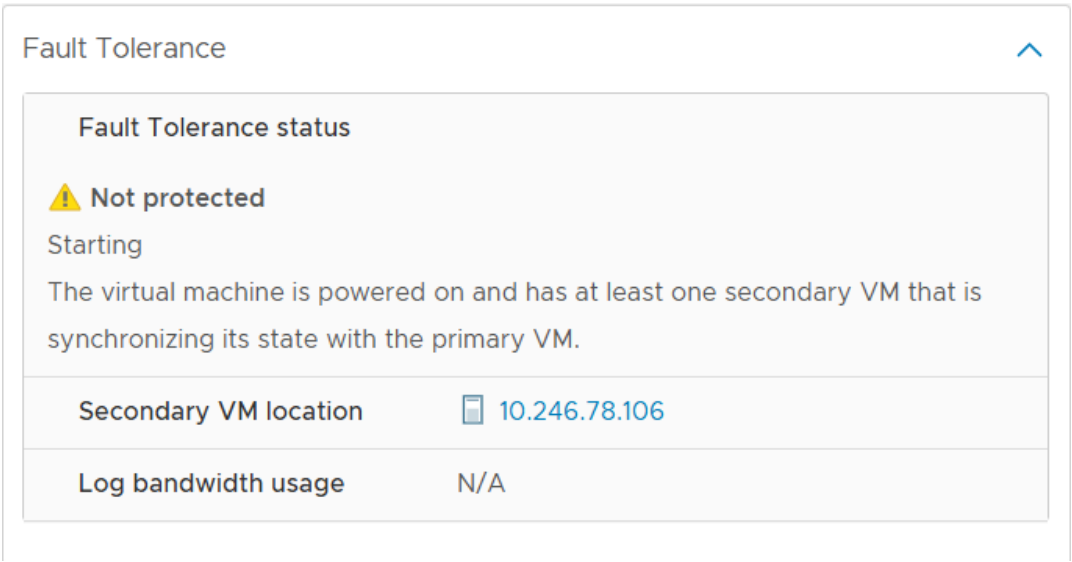
  - Immediately after enabling vSphere FT, the Primary VM remained on its host while the Secondary VM was placed on `10.246.78.106`.




10.246.78.104
Datacenter
Cluster01
10.246.78.100 (Not responding)
10.246.78.106
10.246.78.169
10.246.78.50
NFS_RHEL
Photon_VM (disconnected)
PhotonOS_VM (primary)
PhotonOS_VM
ACTIONS
Summary
Monitor
Configure
Permissions
Datastores
Networks
Updates
Powered On
Launch Web Console
Launch Remote Console
Guest OS: VMware Photon OS (64-bit)
Compatibility: ESXi 6.7 and later (VM version 14)
VMware Tools: Not running, not installed
More info
DNS Name:
IP Addresses:
Host: 10.246.78.50
CPU USAGE
0 Hz
MEMORY USAGE
0 B
STORAGE USAGE
1.06 GB
Recent Tasks
Alarms
Task Name
Target
Status
Initiator
Queued For
Start Time
Completion Time
Server
Start Fault Tolerance Secondary VM
PhotonOS_VM
100%
System
4 ms
02/14/2026, 10:36:57 PM
10.246.78.104


10.246.78.104
Datacenter
Cluster01
10.246.78.100
10.246.78.106
10.246.78.169
10.246.78.50
NFS_RHEL
Photon_VM
PhotonOS_VM (primary)
vCenterVM

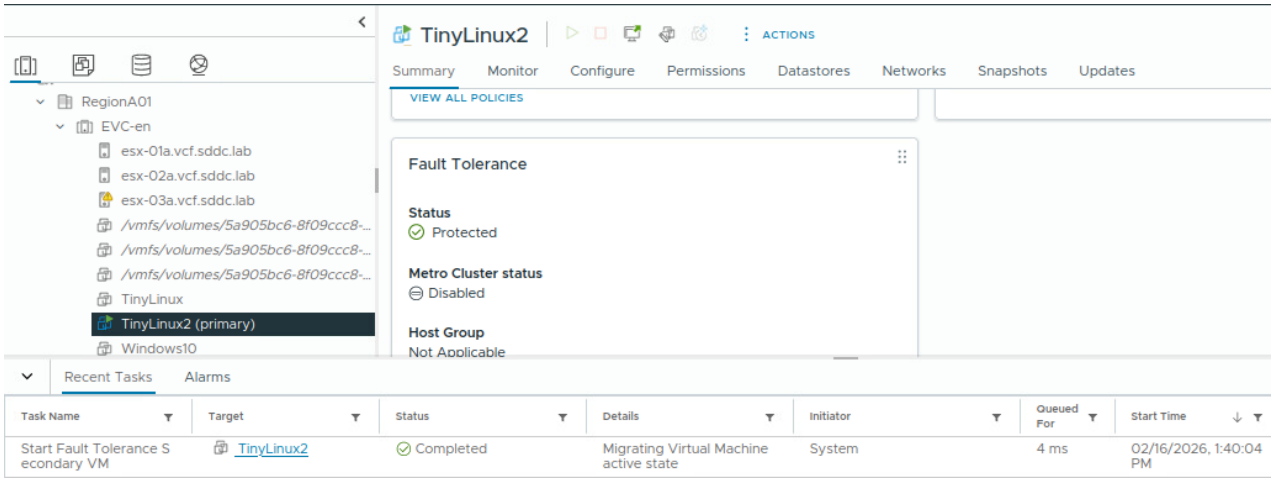
- The vSphere client displayed a yellow warning: *Fault Tolerance status: Not protected.* This was verified to be a normal transitional state. The interface noted the status was Starting because the powered-on primary virtual machine was actively Synchronizing its state to the newly deployed secondary VM.



- **Successful Implementation via VMware Hands-on Labs**
  - To conclusively validate our Fault Tolerance deployment methodology and overcome the local physical bottlenecks, we transitioned our execution to the **VMware Hands-on Labs (HOL)** environment.

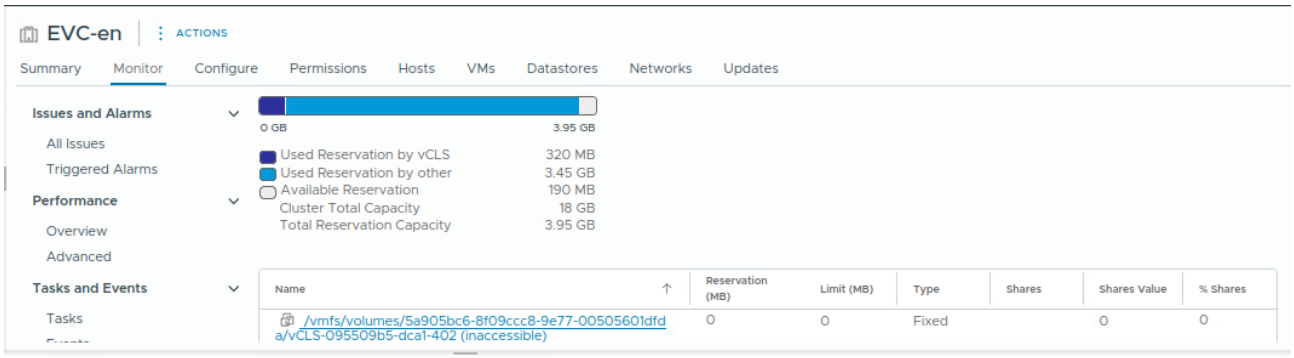
The following steps were successfully implemented:

- **Dedicated EVC Cluster Creation:** A new, separate cluster named EVC-en was created specifically for Fault Tolerance testing. Enhanced vMotion Compatibility (EVC) was successfully enabled on this dedicated cluster to guarantee a uniform CPU baseline.



- **Isolating the Management Plane:** Two ESXi hosts (esx-01a.vcf.sddc.lab and esx-02a.vcf.sddc.lab) were added to this new cluster. Crucially, these hosts were deliberately selected because they were not hosting the vCenter Server. This successfully bypassed the vCenter paradox, allowing the EVC baseline to be applied without disrupting the management plane.

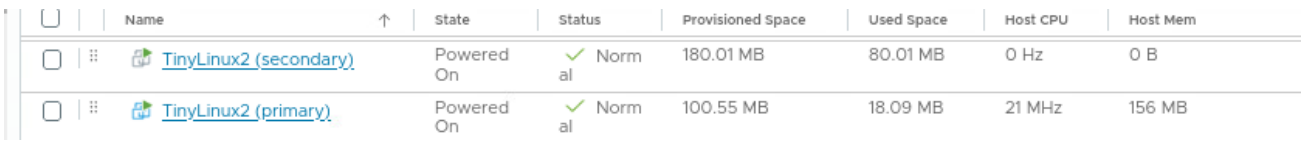
- **Overcoming Admission Control Limits:** After configuring the required FT Logging networks, the allocated RAM for our test virtual machine (TinyLinux2) was strategically reduced. This memory tuning ensured the VM's footprint complied with vSphere's strict Admission Control policies within the lab.



- The cluster only had 190 MB of memory available.
- Enabling Fault Tolerance on a 512 MB VM required more memory than the cluster had left.
- Therefore, we decreased the VM's memory from 512 MB to 128 MB. This allowed it to easily fit into the 190 MB limit and power on successfully.

Successful Execution and Protection: With the CPU architecture unified and resources optimized, Fault Tolerance was initiated. The vCenter task Start Fault Tolerance Secondary VM completed successfully.

Verification: The vSphere Client confirmed the synchronization, transitioning the FT Status to Protected. Both the TinyLinux2 **Primary** and **Secondary** virtual machines operated simultaneously under a Normal status, successfully demonstrating a fully functional FT configuration.



Challenges Faced

- ## VMkernel Network Connectivity Issue

  One of the main challenges faced during the project was related to **VMkernel network connectivity** used for NFS storage.

  All ESXi hosts were configured with VMkernel adapters in the same subnet 192.168.20.x.
  Communication between ESXi hosts was successful.
  However, the NFS virtual machine was unable to communicate with one specific host 192.168.20.50.

  This caused issues in accessing shared storage and prevented proper datastore functionality.

  → **Root Cause**
  After extensive troubleshooting of:
  - vSwitch configuration
  - Port Groups
  - VMkernel adapters
  - Physical uplinks

  It was found that the issue was not related to the ESXi configuration itself, but rather to the **host operating system environment**.

  The ESXI's Host was initially running on a Linux-based system using VMware Workstation, which caused unexpected network behavior and instability in the virtual networking layer.

  → **Solution**
  - The host operating system was changed from Linux to Windows
  - The environment was rebuilt using VMware Workstation on Windows
  - All ESXi hosts and networking configurations were recreated

  After switching to Windows:
  - Network connectivity became stable
  - VMkernel adapters were reachable from the NFS VM
  - NFS datastore was successfully configured

→ **Lesson Learned**

The issue highlighted that:

- The underlying host OS can significantly impact virtualization performance
- Some networking issues may originate outside the ESXi configuration
- Using a stable and fully supported host environment is critical for lab setups

- ## FT Deployment Challenges & Troubleshooting

  Deploying Fault Tolerance in this simulated enterprise environment presented significant hardware and resource challenges, which required strategic workarounds.

- **Challenge 1:**

  Enhanced vMotion Compatibility (EVC) and Hardware Mismatch

  vSphere FT requires the Primary and Secondary VMs to execute in strict lockstep, which demands identical CPU instruction sets and hardware compatibility between the physical hosts. During the initial FT deployment across the original cluster hosts, an EVC/hardware compatibility error occurred.

- **Solution:** To bypass this hardware mismatch and fulfill FT's strict CPU requirements, a completely new nested ESXi host was provisioned with the IP address 10.246.78.106. This new host was specifically created on the **same physical PC** hosting the primary VM. By running both the primary and secondary hosts on the exact same underlying physical hardware, the CPU architecture was guaranteed to be identical, resolving the EVC conflict and allowing the FT configuration to proceed.

  The Optimal Enterprise Solution (Enabling VMware EVC): The best practice to resolve this in a production environment is to enable Enhanced vMotion Compatibility (EVC) on `Cluster01`. EVC establishes a baseline CPU feature set across all hosts in the cluster, masking hardware differences and allowing features like vMotion and FT to function seamlessly across heterogeneous processors.

Steps to Enable EVC:

1. Navigate to `Cluster01` in the vSphere Client.
2. Select the **Configure** tab.
3. Under the **Configuration** menu, select **VMware EVC**.
4. Click **EDIT** and change the EVC state to **Enable** for the respective CPU vendor (Intel or AMD).
5. Select an EVC mode (baseline) that corresponds to the oldest processor architecture present among the ESXi hosts.

The vCenter Paradox & Our Lab Workaround: To apply a lower EVC baseline successfully, any running virtual machines utilizing advanced CPU instructions must be powered off. However, because our `vCenterVM` resides inside the cluster it manages, powering it off creates a paradox where we lose the management interface required to enable EVC. Resolving this normally requires complex migrations to hosts outside the cluster.

To bypass this paradox without disrupting the nested environment, we provisioned a completely new nested ESXi host (`10.246.78.106`) on the same physical PC that was hosting the primary VM. Running both the primary and secondary hosts on the exact same underlying physical hardware guaranteed identical CPU architecture out-of-the-box, allowing us to seamlessly proceed with the FT configuration.

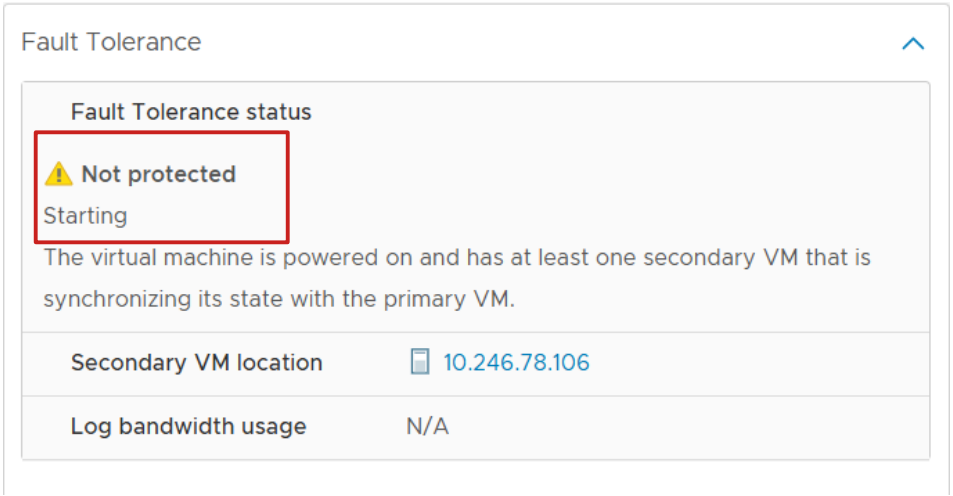
- **Challenge 2:**

  Severe Resource Contention

  Because the environment utilizes nested virtualization running multiple ESXi hosts, a vCenter Server appliance, and an NFS server, the hardware's compute and memory resources were heavily oversubscribed.

  - When FT was enabled, it attempted to deploy and power on the Secondary VM on the new host (10.246.78.106). However, due to the severe lack of available physical resources, the Secondary VM could not fully instantiate or complete its initial synchronization.

- As a result of this resource exhaustion, the vSphere client displayed a yellow warning: Fault Tolerance status: Not protected. The interface remained stuck in the Starting state, noting that the VM was attempting to synchronize its state to the newly deployed secondary VM, but could not complete the process due to the hardware bottleneck.



- **Linux Kernel Update Breaking VMware Workstation**

  Another major issue occurred after updating the host operating system.
  A Linux kernel update caused VMware Workstation to stop working properly.
  Virtual machines failed to start due to incompatibility between VMware modules and the updated kernel.

  → **Root Cause**
  VMware Workstation relies on kernel modules that must be compatible with the running Linux kernel.
  After the update, these modules were no longer supported.

  → **Solution**
  - Reverted to a previous, compatible Linux kernel version
  - Ensured VMware modules could compile correctly
  - Avoided automatic kernel updates during the project timeline