AMD Instinct™ MI350X/MI355X on DVX for VMware ESXi 9.1 User Guide

Version: v1.1
Date: July 2026

Contents

List of Figures

Chapter 1: Overview

1.1 Getting started with virtualization

AMD's virtualization stack for ESXi centers on the `amdgpuv` driver together with SR-IOV (Single Root I/O Virtualization). SR-IOV is the standard pattern for exposing *virtual functions* (VFs) from a physical PCIe device so guests can use efficient, direct device access rather than relying on full software emulation for the accelerator data path.
That direction addresses demanding workloads—high-performance computing (HPC), artificial intelligence (AI), machine learning (ML), and graphics-intensive applications—and, when sharing is enabled in a given release, can support finer isolation and allocation across multiple tenants. This guide introduces the VMware vSphere, DVX, and Instinct MI350X/MI355X context in which `amdgpuv` runs on ESXi.

MI350X and MI355X: The **configuration** steps in this guide are written for **MI350X**. Unless a section states otherwise, the same procedures apply **equivalently** to **MI355X** on officially qualified platforms (Table 3.1, Chapter 3). For simplicity, the text refers to **MI350X** only.

Release scope: This user guide documents **VMware ESXi 9.1** only; other ESXi releases are out of scope for this publication.

The `amdgpuv` version covered by this document does *not* support concurrent sharing of one physical GPU across multiple virtual machines. For this publication, the **only supported deployment model** is **one Linux virtual machine that owns all eight MI350X adapters** on a fully populated host (for example the full XGMI hive). The **default** frame-buffer sharing mode for that hive is **MODE_8** (see Chapter 2); **end users cannot change** frame-buffer sharing mode at this point, and this guide documents no procedure to do so. Patterns such as eight single-GPU guests, one GPU per VM, or other VM×GPU splits are **out of scope** here even if your platform program qualifies them elsewhere; follow your platform release notes for any roadmap beyond this guide. AMD intends to extend `amdgpuv` in future releases where platform qualification and VMware compatibility allow; do not assume additional assignment models from this document alone.

The following chapters cover accelerator architecture, hardware and software requirements, system configuration, and operational practices for `amdgpuv` on vSphere with MI350X/MI355X-class accelerators.

1.2 Understanding SR-IOV

PCI-SIG defines SR-IOV as an approach to I/O virtualization in which a single physical PCIe device can present multiple lightweight virtual functions on the bus. On qualified AMD Instinct platforms, firmware and the host stack (here, `amdgpuv` on ESXi) work together so those VFs can

be assigned to guests or orchestrated via DVX where supported. How many VFs are exposed, and whether multiple VMs may share one physical GPU at the same time, depend on the specific driver release; see the **Release scope** note in §1.1.
Single root means the hierarchy under one PCI Express root complex—the tree of buses and endpoints anchored at the host. A design goal of SR-IOV is to reduce hypervisor mediation on the fast path: each VF can carry its own address space, interrupts, and DMA context, which helps virtual machines approach native PCIe device performance when the device, firmware, and drivers all cooperate.
The SR-IOV standard is maintained by PCI-SIG, ensuring its relevance and effectiveness through cross-industry collaboration and funding in the evolving landscape of virtualization technology.

1.3 VMware vSphere

VMware vSphere is VMware's suite for running and managing virtual infrastructure in the data center. Throughout this document, **vSphere** denotes the *integrated product suite*—ESXi, vCenter Server, and related licensed components—that you deploy and operate as a coordinated platform, not a single installable image.

- **ESXi** (§1.4) is the Type-1 hypervisor that runs directly on each physical host. It schedules VMs, owns PCIe and GPU resources, and hosts the ESXi-side driver stack (including DVX-related paths and vendor GPU drivers such as `amdgpuv` on AMD MI-GPUs).
- **vCenter Server** (§1.6) is the centralized management service for a fleet of ESXi hosts. Through vCenter you define clusters, inventories, VM placement, policies, and features such as vMotion and DRS; it does not replace the data path on the host, but it coordinates how DVX and SR-IOV-style assignments are applied consistently across hosts (§1.7).
- **DVX** (§1.5) is a VMware mechanism on the ESXi side for virtualizing certain PCIe-class devices so that guests see a virtual device model while the platform can still participate in orchestration (for example migration readiness) where the stack supports it—in contrast to raw SR-IOV or pass-through, which favors performance and direct assignment over the same style of mobility.

This user guide assumes a **vSphere** deployment: one or more ESXi hosts with MI350X, managed by vCenter, with Linux guests consuming ROCm as described in later chapters. When the text refers to "the cluster," "the host," or "vCenter," those roles are all within that vSphere model.

1.4 ESXi

ESXi is the hypervisor software installed on each physical server. At runtime it applies CPU and memory scheduling to guests, exposes PCIe accelerators—including general-purpose and compute-oriented GPUs—through hypervisor drivers and assignment policies, and keeps tenant workloads separated from the host management domain. Vendor stacks such as `amdgpuv` on AMD hardware, SR-IOV, and DVX-related paths execute in this layer. vCenter (§1.6) orchestrates inventory and cluster policy but does not substitute for ESXi on the per-host execution path.

Key features

- **CPU and memory scheduling** for concurrent VMs, including fairness, reservations, limits, and shares.
- **Isolation and a minimal trusted computing base** suitable for multi-tenant and regulated environments.

- **GPU and accelerator mediation:** physical GPUs and other PCIe accelerators are owned by ESXi and exposed to VMs through hypervisor drivers and an assignment model appropriate to the device. Hardware and firmware offload, together with VMware device virtualization, help compute-heavy guests retain efficient accelerator access while CPU and memory remain fairly scheduled across concurrent VMs.
- **Virtualized networking:** guest traffic typically uses virtual switches, port groups, and paravirtual or emulated NICs; where the design requires lower latency or higher throughput to the wire, ESXi also supports assigning physical network adapters to selected VMs.

1.5 DVX driver

The DVX (Device Virtualization Extensions) is a component of VMware which allows the driver stack to provide PCI Express virtual device implementations for accelerators and high-performance devices. Virtual devices created through the DVX interface automatically become available for consumption by virtual machines. Figure 1.1 illustrates the DVX device virtualization context at a high level.

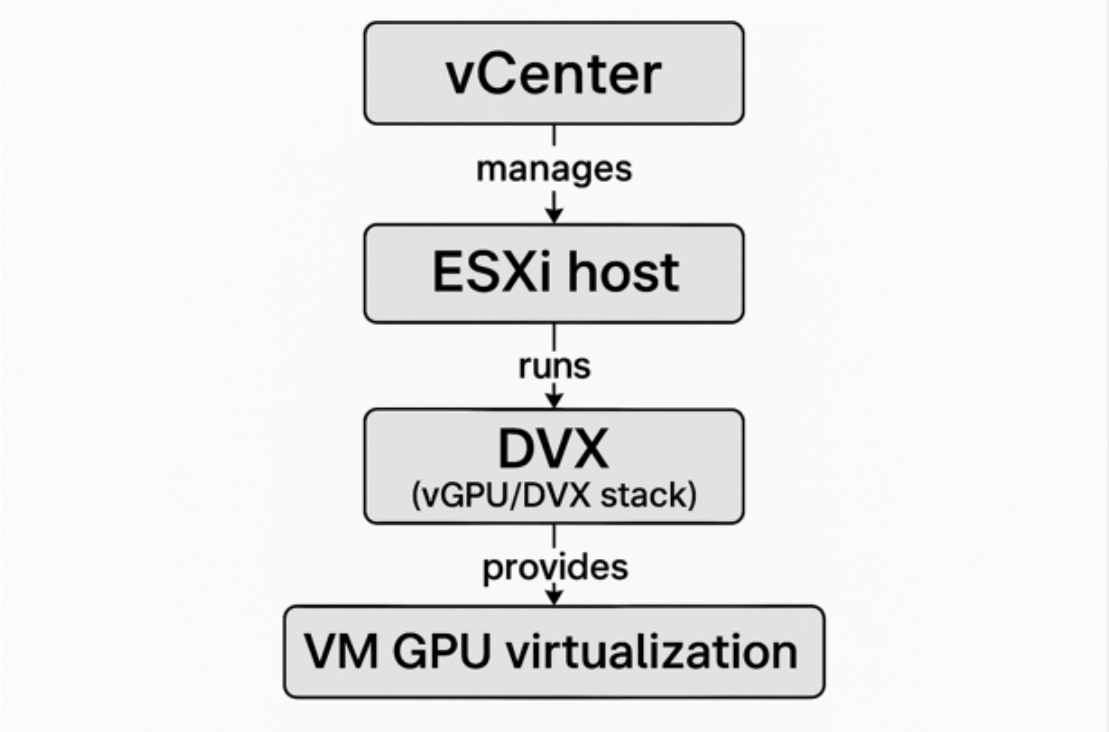


Figure 1.1: DVX / device virtualization context.

The DVX supports suspend/resume, save/restore, and vMotion readiness, and it includes virtual shared graphics acceleration (vSGA). This interface may be compatible with multiple ESXi versions in the broader VMware portfolio; **this guide addresses ESXi 9.1 only**. It is optimized for high performance by offloading certain device functions to hardware/firmware paths. It ensures smooth migration and policy-driven offloads across a cluster of ESXi hosts managed by vCenter, facilitating scalable, hardware-accelerated virtualization.

1.6 vCenter

vCenter is a service that acts as a centralized management layer responsible for coordinating ESXi hosts (and their VMs) connected in a network. It provides one place for provisioning, monitoring, and policy enforcement across the data center or edge deployments. When used with DVX, vCenter coordinates device virtualization protocols, offload configurations, and VM migrations, enabling consistent DVX behavior (throughput, latency, and device policy). This combination simplifies management, improves orchestration of hardware-accelerated workloads, and supports features like vMotion, DRS, and high availability with DVX-enabled devices.
Figure 1.2 summarizes, at a high level, how the pieces in this guide relate: VMware ESXi on GPU hosts, the `amdgpuv` driver stack on those hosts, and vCenter as the management plane that orchestrates hosts, virtual machines, and DVX-capable accelerator assignments.

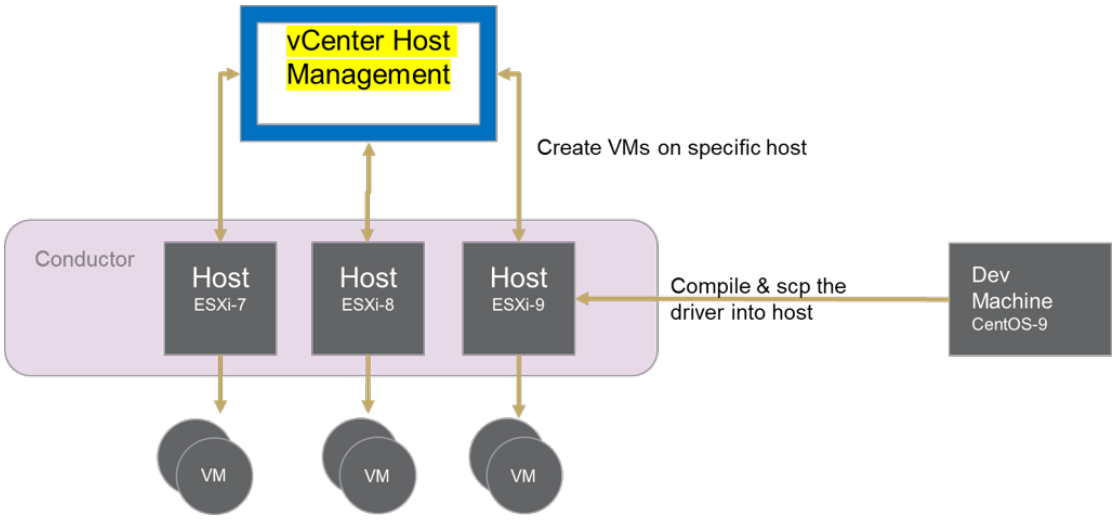


Figure 1.2: High-level view of ESXi, `amdgpuv`, vCenter, and DVX-related accelerator workloads.

1.7 vCenter, DVX, and SR-IOV

In vCenter, data-path processes are not directly managed; instead, it provides a control-oriented layer that coordinates SR-IOV and DVX functionality at the cluster level. DVX is a software mechanism within the VMware platform that enables virtualization of PCIe devices to support features such as vMotion and highly portable, orchestratable virtualization. SR-IOV is a hardware mechanism that allows a single physical device to expose multiple Virtual Functions (VFs) directly assigned to virtual machines, prioritizing performance over mobility and orchestration.

Chapter 2: AMD Instinct MI350X/MI355X Architecture

Leveraging the latest CDNA 4 architecture, the AMD Instinct MI350X and MI355X series accelerators are engineered for the most demanding AI and HPC workloads. Featuring up to 256 compute units, these GPUs harness 288 GB of HBM3E memory to manage large data sets efficiently, achieving up to 8 TB/s of memory bandwidth for complex, data-driven applications. Advanced AMD Infinity Fabric interconnects enhance multi-GPU scalability and performance, supporting ultra-dense configurations for large-scale AI training.
The AMD Instinct MI355X GPU is purpose built for high density computing environments, offering more peak performance than its counterpart the MI350X. This increased power enables the MI355X to sustain higher performance over time, minimizing throttling and maximizing throughput during prolonged or intensive workloads.
Designed for high-performance, energy-efficient operation, these accelerators maintain robust computational output across a wide range of workloads. With comprehensive support for various precision modes and data types, the MI350X and MI355X deliver flexible and efficient computation for scientific, AI, and HPC tasks.

2.1 Understanding XGMI

XGMI, or External Global Memory Interconnect, is AMD's high-speed GPU-to-GPU interconnect based on Infinity Fabric™ technology. It plays a crucial role in creating a coherent memory space across multiple GPUs, enabling efficient data transfer for high-performance computing (HPC) and artificial intelligence (AI) workloads.

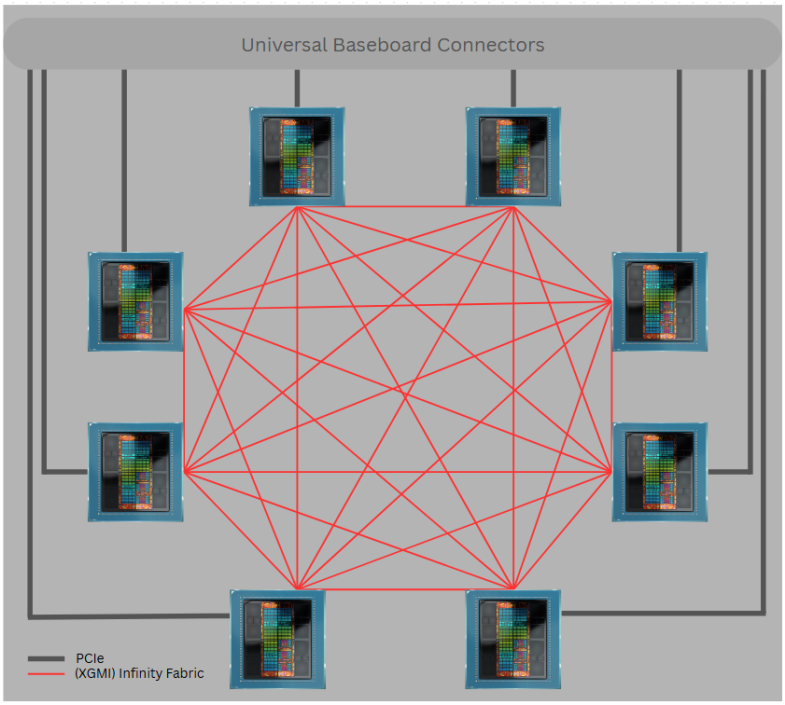


Figure 2.1: Preview of XGMI link between multiple GPUs

According to the XGMI specification, XGMI serves as an inter-socket interface that maintains a bi-directional communication channel. This channel bridges coherent fabrics between multiple silicon instances, allowing for seamless data transfer. The physical XGMI connector is present on supported AMD GPUs, enabling the connection of multiple GPU cards into a homogeneous memory space. This memory space is constructed by joining the local VRAM of each peer GPU. XGMI enables P2P transfers that are generally faster than those over PCIe, although performance may vary depending on the platform.

Note: It is important to note that XGMI is only available in a single Virtual Function (VF) mode. This means that the advantages of XGMI cannot be utilized if one GPU Physical Function (PF) is partitioned into multiple GPU VFs during driver load time.

2.2 Frame Buffer Sharing Modes

On multi-GPU Instinct platforms, frame buffer (FB) memory can be described in terms of predefined *sharing modes* that govern how adapters see one another's frame buffers. For the ESXi + `amdgpuv` stack in this user guide, the **default** FB sharing mode is **MODE 8** (all GPUs in one sharing domain). **End users cannot change** FB sharing mode at this point; nothing in Chapter 4 alters it.
Completeness. The other modes below appear **only** so you recognize the names in Instinct and `amd-smi` documentation elsewhere. They are *not* selectable deployment options in this publication.

- **MODE 8:** All GPUs share their frame buffers with each other.
- **MODE 4:** GPUs are divided into two groups, with each group of four sharing their frame buffers.
- **MODE 2:** GPUs are divided into four groups, with each group of two sharing their frame buffers.
- **MODE 1:** No sharing of frame buffers between GPUs.

Chapter 3: Hardware and software requirements

This chapter collects **minimum and example requirements** for building and testing the MI350X DVX stack on VMware ESXi with Linux guests. For the MI350X/MI355X GPU architecture summary, see Chapter 2. The material here is organized in two layers:

Hardware (§3.1) What the physical host and the guest VMs need: officially qualified host platforms (Table 3.1), host CPU/memory/NVM/networking/GPU rules (§3.1.1), and illustrative guest sizing under §3.1.2 / §3.1.2 (Table 3.3).

Software (§3.2) Three stacks: the *management* plane where vCenter runs (§3.2.1), the *data-center* ESXi hosts that hold MI350X or MI355X and run workloads (§3.2.2), and *guest* Linux VMs (§3.2.3) (Tables 3.4–3.6).

All figures in the tables are **template or example values** from a reference lab configuration. Replace them with targets that match your own qualification matrix, server and board guides, and the ESXi and ROCm versions you certify for production.

3.1 Hardware requirements

The **ESXi host** owns the physical GPUs, PCIe topology, and firmware settings (including SR-IOV, IOMMU, and related BIOS options summarized later in Chapter 4). It must provide enough CPU, memory, and I/O capacity to run the hypervisor, vCenter-connected services, and the assigned accelerator resources without contending with guest workloads.

Guest VMs consume virtual CPUs, RAM, and disk you allocate per VM. When GPUs are exposed via DVX-style virtual devices, each guest still needs sufficient CPU and memory for its driver stack, ROCm user space, and application space. For the `amdgpuv` release described in this guide, the **only in-scope assignment** is **eight MI350X devices in one Linux VM** on a fully populated host (one guest owns all eight adapters). Table 3.3 gives *example* vCPU, RAM, and disk shapes for that topology; scale values for your workload mix, page-cache behavior, and logging.

How this section is organized: §3.1.1 cites the officially qualified server platforms in Table 3.1, then gives template *floor* values for the host machine. §3.1.2 (§3.1.2) gives a guest sizing template for the supported **one VM × eight GPU** model.

3.1.1 Host

The host must be a known-good SR-IOV platform for MI350X or MI355X: correct PCIe widths and speeds, stable platform firmware, and enough DIMM capacity to cover ESXi, VM memory reservations, and any large BAR or DMA mapping pressure from multiple GPUs. Network bandwidth to vCenter and between hosts matters if you use features that migrate or orchestrate VMs across the cluster. Use only platforms that appear in Table 3.1 as officially qualified for this stack unless your support agreement explicitly adds another system.

Qualified platforms

Table 3.1 lists server platforms that are **officially qualified** for the DVX configuration described in this guide. The **Accelerators** column states whether that qualification uses **MI350X** or **MI355X** GPUs. Treat the table as the authoritative list of supported OEM systems for this program; other SR-IOV-capable servers are outside the scope of this document unless and until they are added to that qualification set.

Table 3.1: Officially qualified host platforms

OEM	System	Accelerators
Supermicro	AS-4126GS-NMR-LCC	MI355X
Supermicro	AS-8126GS-TNMR	MI350X

Minimum host resources (template)

Table 3.2 states **template minimums** for a dense MI350X configuration. Treat NVM and network entries as placeholders: match NVMe and NIC classes to your datastore design, vSAN or NFS traffic, and guest image placement. GPU counts assume SR-IOV is enabled in firmware and that XGMI hive topologies follow your board vendor's cabling rules.

Table 3.2: Host hardware requirements

Item	Requirement
CPU	>2 × 96 cores (example template)
System memory	>1.5 TiB
NVM	>2.5 T
Network adapter	>1000 M
GPU adapter	1–8× MI350X with SR-IOV enabled; or 8× MI350X in XGMI hive

3.1.2 Guest

Guest sizing is driven by how many GPUs each VM owns, how much host CPU and RAM you reserve for parallel jobs, and how large root disks must be for ROCm, logs, and datasets. The template below does **not** include scratch space for large checkpoints or container images—extend disks or add data vDisks if your benchmarks require it.

Guest virtual hardware template

Table 3.3 belongs to **§3.1.2** and lists **example** vCPU, RAM, and boot-disk sizes for that single supported pattern. Adjust every row for your own qualification plan; these numbers are a starting point only.

Table 3.3: Guest sizing — one VM with eight GPUs (example template)

Resource	1 VM × 8 GPU (example)
CPU	128
RAM	1 T
Disk capacity	1 T

3.2 Software requirements

VMware splits responsibilities across a **management** tier (vCenter and its supporting services), one or more **ESXi hosts** in a datacenter or cluster that actually run VMs and own the GPUs, and the **guest operating systems** inside those VMs. The subsections below follow that split so requirements are not mixed between the machine that hosts vCenter UI/APIs and the machines that execute GPU workloads.

3.2.1 Management host (vCenter Server)

The **management server** is where vCenter Server runs. In most labs it is deployed as the VMware vCenter Server Appliance (VCSA) on an ESXi host or management cluster that is *separate* from the GPU-heavy datacenter you use for MI350X testing, though small setups sometimes collapse roles onto fewer physical boxes. This tier does not load `amdgpuv`; it must only meet VMware's pairing rules for ESXi 9.1 and the vCenter build you pair with it, provide reliable DNS, time, and backup for the vCenter VM, and expose the UI or APIs your operators use to attach devices, run workflows, and monitor inventory.

Obtaining vCenter images: Download the vCenter Server Appliance (VCSA) installation media (ISO and associated payloads) from the Broadcom Support Portal using your entitled Broadcom account. Pick the vCenter build certified for ESXi 9.1 per §3.2.2, following the current VMware/Broadcom interoperability matrix for that release pair.

Table 3.4: Management plane — vCenter (template)

Component	Example / template requirement
vCenter Server	vCenter Server Appliance (VCSA) or form factor approved by your IT standard
vCenter version	Release certified with ESXi 9.1 (use VMware interoperability matrix)
Identity / time	NTP to lab stratum; DNS forward/reverse for vCenter FQDN
Client access	Supported browser and VMware Remote Console builds per VMware documentation
Network	Stable management VLAN/L3 path from operators to vCenter and from vCenter to every ESXi host

3.2.2 Datacenter hosts (ESXi)

Datacenter hosts are the ESXi systems that join the vCenter inventory, host the MI350X adapters, and run the guest VMs. For MI350X on ESXi, **host setup** spans the hypervisor build, the VMware GPU virtualization stack (`amdgpuv`), and the **publisher-delivered** GPU/baseboard firmware bundle described below. Requirements in this subsection apply per *GPU host*, not to the vCenter appliance.

PLDM / BKC / platform firmware: **PLDM** (*Platform Level Data Model*) and **BKC** (*Best Known Configuration*) are used interchangeably with **platform firmware** in this guide for the same publisher-delivered bundle. The floor for that bundle is **26.00**. Obtain and apply updates *only* through your **server or infrastructure publisher**'s qualified release and change process; do not self-fetch, inventory, or reflash outside that channel unless your program explicitly authorizes it.

Obtaining ESXi images: Download ESXi installation ISOs from the Broadcom Support Portal under the same entitlement flow as vCenter. For deployments on a *specific server vendor and platform*, strongly consider **vendor-customized images** (OEM images): these ESXi builds are tailored to particular hardware generations and typically bundle the right inbox or OEM drivers, management agents, and validation against that vendor support matrix. Use the generic ESXi image only when your qualification plan explicitly allows it and you accept responsibility for driver and firmware alignment on that SKU.

Table 3.5: ESXi datacenter host software (template)

Component	Requirement
Hypervisor	VMware ESXi 9.1 (only release supported by this guide)
`amdgpuv` driver	Officially signed VIB (match ESXi 9.1 and the firmware baseline your publisher qualifies)
Firmware bundle	At least **26.00**, publisher-qualified; see the callout **PLDM / BKC / platform firmware** (p. 13)

While AMD publishes drivers and ROCm user space components, your server or infrastructure provider publishes the GPU and baseboard firmware by bundling AMD's firmware releases via AMD's **Platform Level Data Model (PLDM)** bundle, which includes the **Integrated Firmware Image (IFWI)**.
GPU and baseboard firmware versioning might differ across GPU families.

Obtaining the `amdgpuv` VIB (AMD)

The ESXi `amdgpuv` driver package is distributed by AMD separately from Broadcom's ESXi and vCenter media. To find releases, start from the **AMD Instinct accelerators** portfolio page (link below), then open the **device product page** that matches your deployment, for example MI350X or MI355X. On that page, use the **Driver & support** section to locate the **ESXi** `amdgpuv` VIB for your qualified build. The deliverable is a **VMware-format VIB that AMD signs**; install it with normal ESXi signature verification. If your program's entitlement uses a partner-specific path, use that in addition to AMD's public listings.

- **AMD Instinct accelerators (portfolio entry point):**
  `https://www.amd.com/en/products/accelerators/instinct.html`

3.2.3 Guest virtual machines

Guest VMs run the Linux stack that consumes ROCm and user workloads. They are independent of the ESXi build except through virtual hardware and pass-through/DVX device models you assign in vCenter or host profiles.

The guest operating system named in Table 3.6 is the distribution that was **qualified** for this guide's stack. The guest `amdgpu` driver and ROCm user space may behave similarly on other Linux distributions, but those combinations are outside the scope of this document unless you qualify them under your own program.

Table 3.6: Guest VM software (template)

Component	Requirement
Guest OS	Ubuntu 24.04
`amdgpu` driver / ROCm	ROCm 7.2.1 and later (guest driver support)

Chapter 4: System configuration

This chapter describes **system configuration** for the MI350X DVX stack. On the **ESXi host** it covers applicable BIOS settings, **enabling GPU DVX** (SR-IOV and virtual-function count on the MI350X adapters) as a prerequisite to installing the `amdgpuv` driver VIB, the VIB install itself, and (where needed) AMD SMI for monitoring or diagnostics. For the qualified XGMI hive, the **default** frame-buffer sharing mode is **MODE 8**. It is **not** something the operator can change through this guide or through documented steps in this publication at this time; Chapter 2 explains the terminology. For **vSphere and the Linux guest** it covers attaching hosts to a datacenter, provisioning the Linux VM (ISO install, clone, or template), attaching the DVX `amdgpuv` **Vendor Device Group (VDG)** for the eight-GPU hive to that VM, and preparing the guest for ROCm. It builds on the accelerator overview in Chapter 2 and the hardware and software requirements in Chapter 3.

4.1 Host setup

Installation of VMware ESXi on the server is *not* part of this document. The host is expected to already run **VMware ESXi 9.1** (aligned with the hypervisor requirement in Chapter 3; this guide does not support other ESXi releases). For media, licensing, and installation or upgrade procedures, refer to the Broadcom Support Portal.

4.1.1 Enable the following extensions in system BIOS

On many server platforms, recent BIOS or platform firmware revisions already enable virtualization, IOMMU, and related PCI options by default. Use Table 4.1 as a checklist: for each listed capability, if a matching control exists in your firmware menus, configure it as required. If a named setting does not appear, no further action is ordinarily needed—the feature is expected to be enabled by default or provided through an equivalent vendor implementation; confirm with your OEM documentation if you are unsure.

Table 4.1: BIOS settings for host preparation

Setting	Location / notes
Virtualization extensions	BIOS option
SR-IOV support	Advanced → PCI Subsystem Setting
Above 4G decoding	Advanced → PCI Subsystem Setting
PCIe ARI support	Advanced → PCI Subsystem Setting
IOMMU	Advanced → NB Configuration
ACS enabled	Advanced → NB Configuration

4.1.2 Enabling GPU DVX devices (host SR-IOV and VFs)

For the sequence in this guide, preparing the physical MI350X adapters for DVX is a **mandatory step before** you install the ESXi `amdgpuv` host driver (§4.1.3). On each adapter, enable **SR-IOV** and set the virtual-function count as your program requires (this document uses **one VF per physical GPU**). Per-VF **Dynamic Direct Path I/O** / passthrough toggles are **not** required for the flow in this guide; vSphere exposes the `amdgpuv` Vendor Device Group to the

VM without that extra host step. The flow below uses the host **Configure** → **PCI devices** UI (vSphere Client or equivalent). Wording and tab names can vary by release; match the intent to your build.

Open PCI devices on the host. Select the ESXi host in inventory, open the **Configure** tab, and choose **PCI devices**. You should see the full list of PCI endpoints the host reports (Figure 4.1).

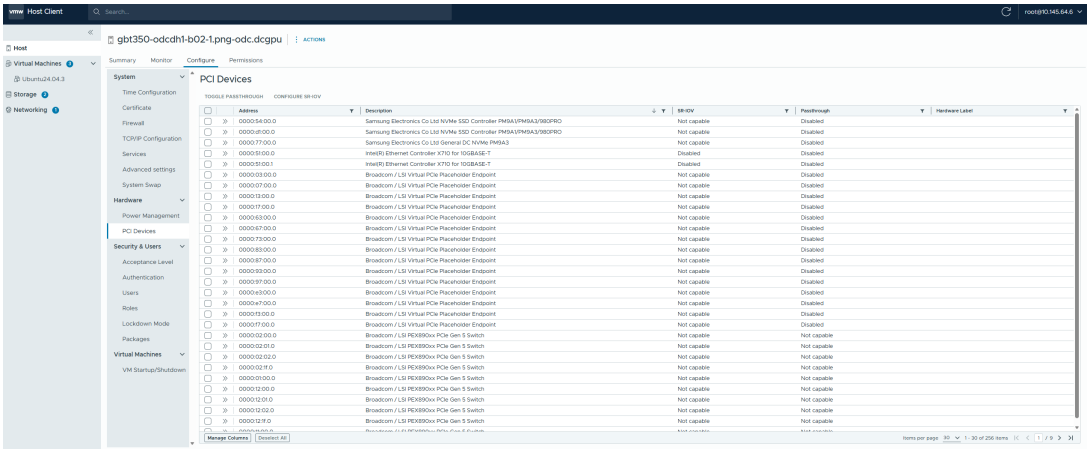


Figure 4.1: Host **Configure**: **PCI devices** list (all devices).

Focus on MI350X adapters. The list can be long. Use the table's **Description** (or equivalent) filter and search terms so only the **MI350X** GPU entries remain visible (Figure 4.2). Select the adapter you are configuring if several are present.

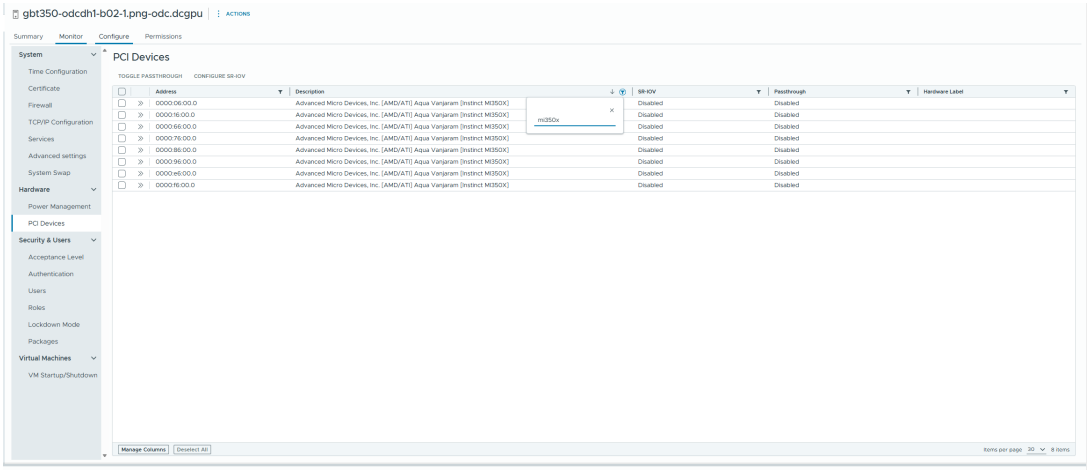


Figure 4.2: Filter PCI devices to MI350X adapters before changing SR-IOV settings.

Enable SR-IOV on the GPU. With the target MI350X row selected, turn on **Enable SR-IOV** (or the vendor-equivalent control) for that device (Figure 4.3).

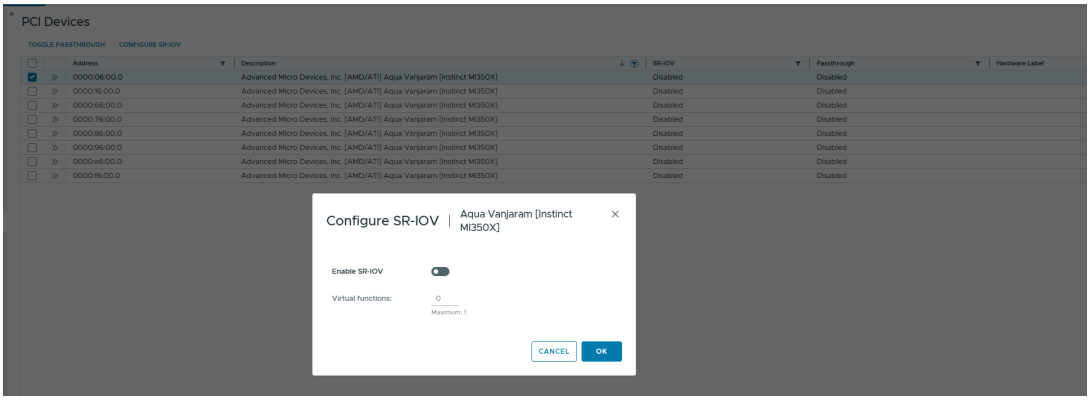


Figure 4.3: Enable SR-IOV for the selected MI350X PCI device.

Set the VF count. Set the number of virtual functions to **1**. This guide is written and qualified for a **single VF per physical GPU** on the host; other counts may be valid for your program but are outside the scope of this document (Figure 4.4).

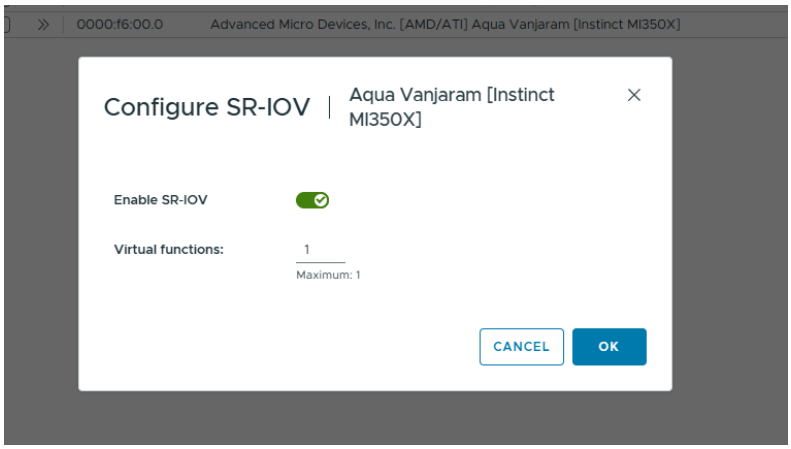


Figure 4.4: SR-IOV enabled with VF count set to one.

Apply changes and reboot if required. After you save or confirm, the UI may show a state such as **Enabled / needs reboot** if the platform stack is not yet bound as expected (Figure 4.5). Repeat this step for each MI350X on the host. After all adapters are configured, proceed with host driver installation in §4.1.3.

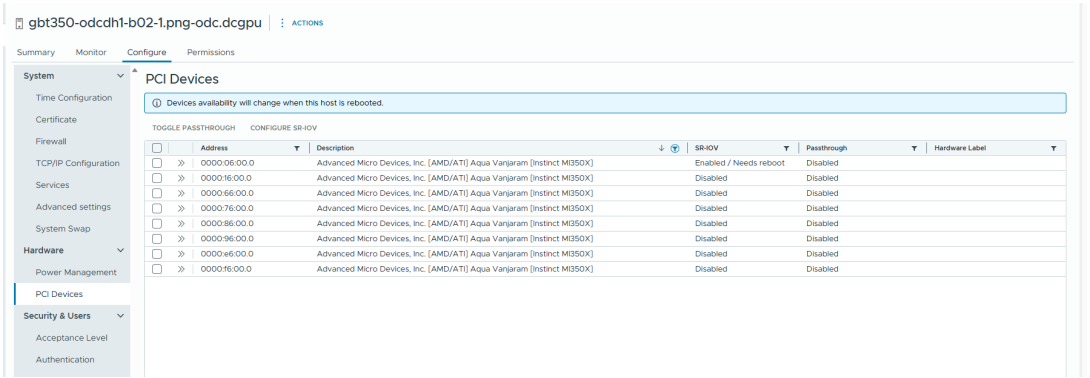


Figure 4.5: Device may report **Enabled / needs reboot** until the host is restarted.

4.1.3 Installing amdgpuv host driver

Prerequisite: Complete §4.1.2 (SR-IOV and VF count on all MI350X adapters) **before** you download or install this VIB.

Download the VIB

Download the ESXi `amdgpuv` driver package from AMD using the official path for your program. This guide points to that location in Chapter 3 under **Obtaining the `amdgpuv` VIB (AMD)** (§3.2.2, item **`amdgpuv` driver (ESXi)**). The production VIB from AMD is **officially signed** for VMware ESXi; the install steps in §4.1.3 use the normal `esxcli software vib install` path so the host can **verify that signature**. Save the file to a system that can reach the ESXi host and its datastores. If AMD delivers a `.zip` or similar archive, extract it and identify the `amdgpuv-*.vib` payload. Upload the `.vib` to a datastore the host can access (for example the vSphere Client **Datastore** browser, or `scp`/SFTP to `/vmfs/volumes/<datastore>/` when your policy allows it). Record the full path to the `.vib`; the install commands in §4.1.3 use that path.

Install on the host

On each GPU host, use the DCUI, SSH, or ESXi Shell with sufficient privileges and apply the procedure in §4.1.3 to install the VIB you copied to the datastore.

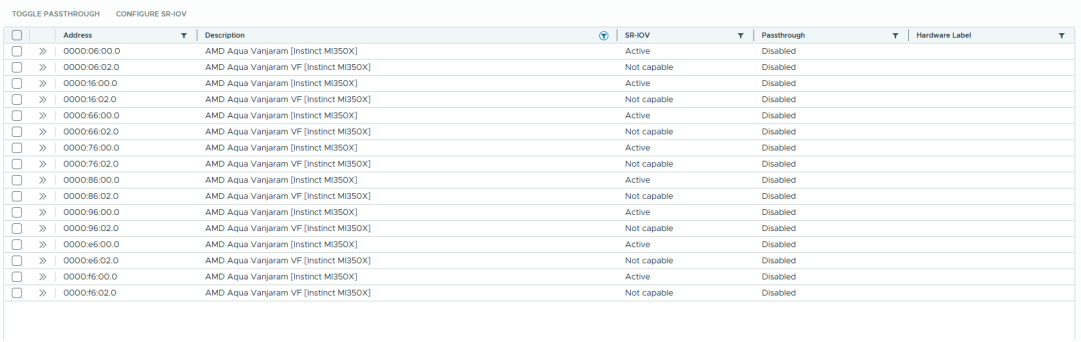
Maintenance mode and VIB install

Before installing or removing VIBs or changing kernel modules, place the host in maintenance mode so that running VMs can be migrated or powered down in a controlled way. From the vSphere Client, right-click the host and choose **Enter Maintenance Mode**, or use the CLI equivalents below.

ESXi host shell: amdgpuv VIB install and verification

```
# Optional: esxcli system maintenanceMode set --enable true
vim-cmd /hostsvc/maintenance_mode_enter
# --- maintenance mode (install VIB) ---
# Replace <number> with your datastore index; adjust path and amdgpuv-*.vib name.
# AMD qualification VIB is signed; omit --no-sig-check unless AMD support directs a one-off
    exception.
esxcli software vib install -v /vmfs/volumes/datastore<number>/amdgpuv-*.vib
# Example installer transcript (values will match your build):
# Installation Result
#   Message: The update completed successfully, but the system needs to be rebooted for the
    changes to be effective.
#   VIBs Installed: AMD_bootbank_amdgpuv_7.0.28.0-1OEM.803.0.0.24022510
#   Reboot Required: true
# Enable module autoload after reboot: -m module name, -e enable
esxcli system module set -m amdgpuv -e true
# If you will reboot immediately, you may remain in maintenance mode.
# Optional: esxcli system maintenanceMode set --enable false
vim-cmd /hostsvc/maintenance_mode_exit
reboot
# After reboot: confirm module and VIB (example output shown in comments)
esxcli system module list | grep "amdgpuv\|Is Loaded"
# amdgpuv                               true        true
esxcli software vib get -n amdgpuv | grep "Creation Date\|Version:"
#   Version: 7.0.28.0-1OEM.803.0.0.24022510
#   Creation Date: 2025-10-16
date
```

After the `amdgpuv` VIB is installed and the host has rebooted, each MI350X in **PCI devices** should report SR-IOV as active and list its virtual functions under the adapter—ready for the guest assignment steps in §4.2.4.


TOGGLE PASSTHROUGH CONFIGURE SR-IOV

Address	Description	SR-IOV	Passthrough	Hardware Label
0000:06:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:06:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	
0000:16:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:16:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	
0000:66:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:66:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	
0000:76:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:76:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	
0000:86:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:86:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	
0000:96:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:96:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	
0000:e6:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:e6:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	
0000:f6:00.0	AMD Aqua Vanjaram [Instinct MI350X]	Active	Disabled	
0000:f6:02.0	AMD Aqua Vanjaram VF [Instinct MI350X]	Not capable	Disabled	

Figure 4.6: Virtual functions listed under the adapters after `amdgpuv` is active.

4.1.4 Uninstalling `amdgpuv` from the ESXi host

Remove the `amdgpuv` VIB when you are backing out a qualification image, replacing the driver with another AMD-signed build (when a clean remove is required before reinstall), or returning the server to a configuration that must not load the MI350X ESXi stack. The steps below remove the **same signed VIB** you installed with `esxcli software vib install`.

Before you remove the VIB

- **Guests:** Power off or migrate any VMs that consume MI350X, DVX, or related SR-IOV assignments on this host so that no workload holds the GPU or its virtual functions.
- **Maintenance mode:** Put the host in maintenance mode using the vSphere Client or the same CLI pattern as in §4.1.3.
- **Cluster policy:** If the host is in a DRS-enabled cluster, confirm that maintenance mode and any required evacuations match your change window.

Remove the VIB and reboot

Run the following on the host shell. The example transcript in comments is illustrative; your `VIBs Removed` line reflects the build you had installed. If `esxcli` reports that the module is busy, ensure all GPU consumers are stopped and that the host is in maintenance mode before retrying; use VMware documentation for any `-f`/`-force` option only when your support process explicitly allows it.

ESXi host shell: `amdgpuv` VIB removal

```
# Host must be in maintenance mode; no VM should be using MI350X on this server.
esxcli software vib remove -n amdgpuv
# Example removal transcript (version strings match your prior install):
# Removal Result
#   Message: The update completed successfully, but the system needs to be rebooted for the
    changes to be effective.
#   VIBs Removed: AMD_bootbank_amdgpuv_7.0.28.0-1OEM.803.0.0.24022510
#   Reboot Required: true
reboot
# After reboot: confirm the VIB and module are gone (expect no amdgpuv line)
esxcli software vib list | grep -i amdgpuv || true
esxcli system module list | grep amdgpuv || true
```

Exit maintenance mode when the host is ready to rejoin the cluster, using the vSphere Client or `vim-cmd /hostsvc/maintenance_mode_exit` / the matching `esxcli` maintenance-mode command, consistent with your operational runbook.

4.1.5 AMD SMI (System Management Interface)

AMD SMI is AMD's **System Management Interface**. The `amd-smi` utility is a command-line tool that uses AMD SMI Library APIs to monitor and configure AMD GPUs and NICs. In a virtualized deployment such as the one this guide describes, it lets host administrators check GPU and NIC state from the ESXi host environment and provides broad GPU management capabilities appropriate to that role.
Output can be plain text, JSON, or CSV; you can show results in the console or write them to a file you specify, depending on the options you pass to `amd-smi`.
With the `amdgpuv` stack installed as in §4.1.3, the following components are available at the paths below (as delivered with this configuration, not build-variant paths):

- **AMD SMI Library:** `/usr/lib/libamdsmi.so`. Use this shared object when you build or link custom tools against the AMD SMI API.
- **`amd-smi` CLI:** `/bin/amd-smi`. This is the command-line entry point that exercises those APIs for monitoring and configuration from a shell.

For syntax, subcommands, and format options, see AMD's Instinct documentation: AMD SMI: `amd-smi` CLI usage. This guide does not document the C or Python API surfaces.
Frame-buffer sharing. The stack assumes the hive is already running with the default **MODE 8** frame-buffer sharing arrangement. End users **cannot** change that mode using the workflows in this document at this point. Do not treat `amd-smi` or other external references as a license to reconfigure FB sharing on the host unless your platform program explicitly directs it outside this guide.

4.2 Guest setup

This section covers how to add the ESXi host to vCenter, create the Linux guest VM, attach the MI350X `amdgpuv` VDG (`com.amd.amdgpuv.xgmi8`) to that VM, and configure the guest for ROCm.

4.2.1 Create new datacenter on vSphere and connect host server

Log in to vSphere and open the vSphere IP address. Select “New Datacenter”, then name it.

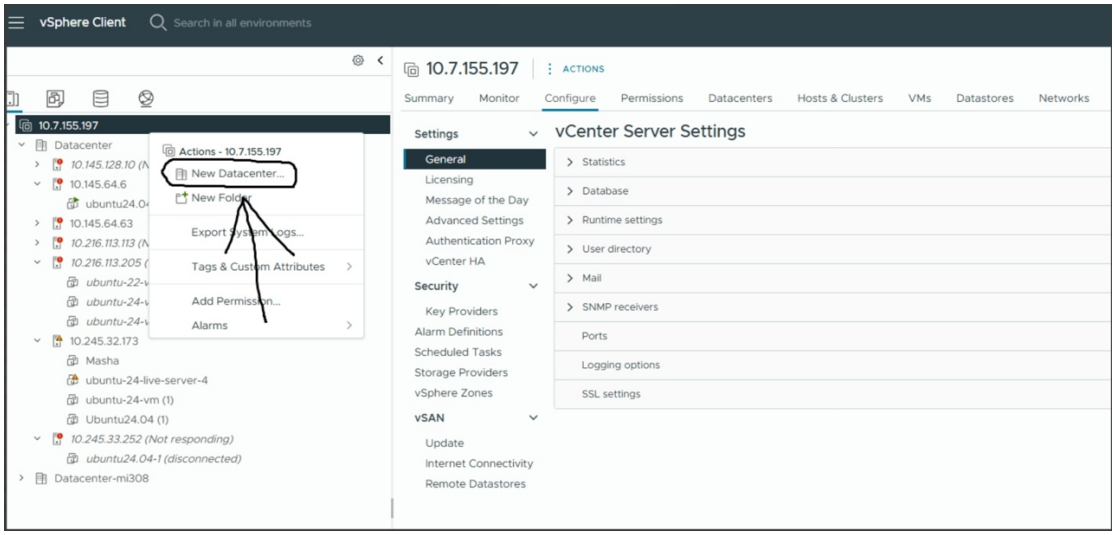


Figure 4.7: vSphere Client: create a new datacenter (initial step).

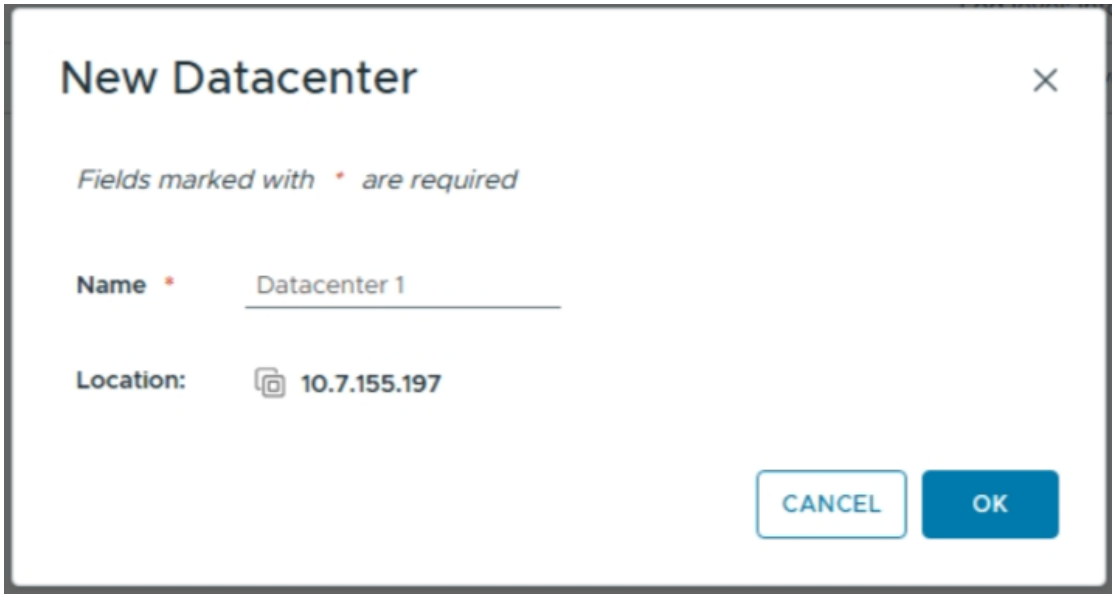


Figure 4.8: vSphere: new datacenter (following screen).

Add the host server to vCenter: select the datacenter name and choose “Add Host”.

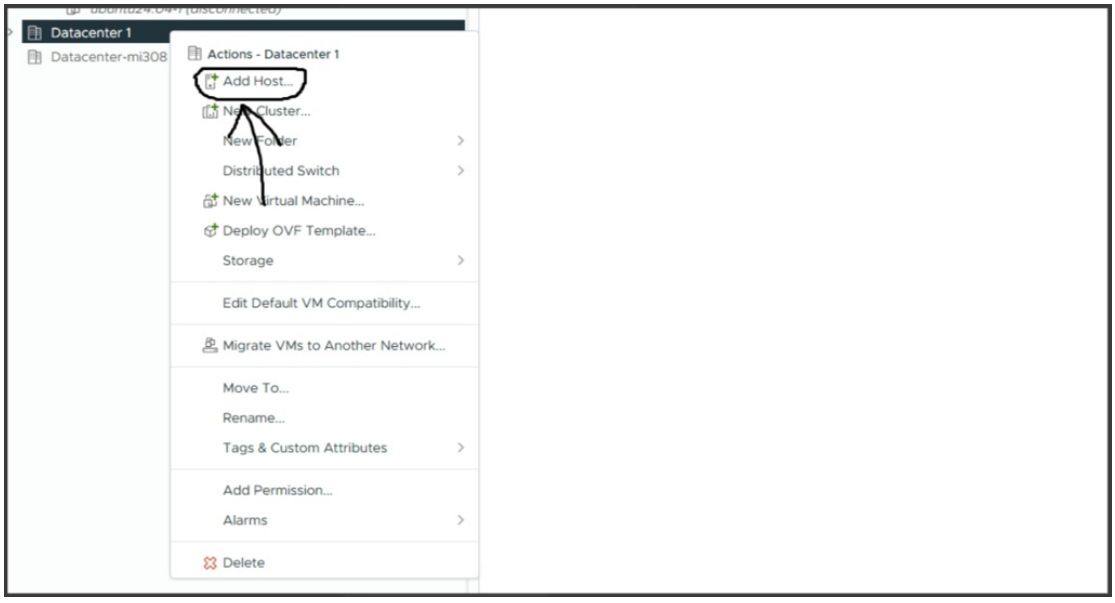


Figure 4.9: vSphere: add host wizard.

Add the host server IP address.

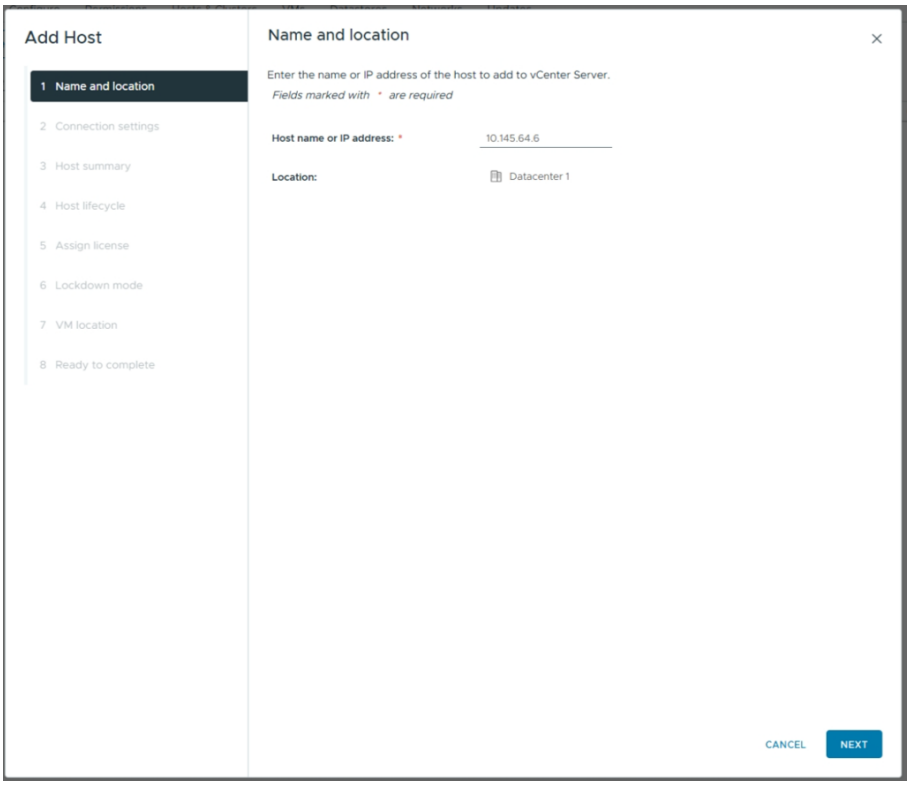


Figure 4.10: vSphere: host connection settings (IP address).

Press **Next** in the following tabs until you reach **Host lifecycle**. Choose “Extract the image on the host”.

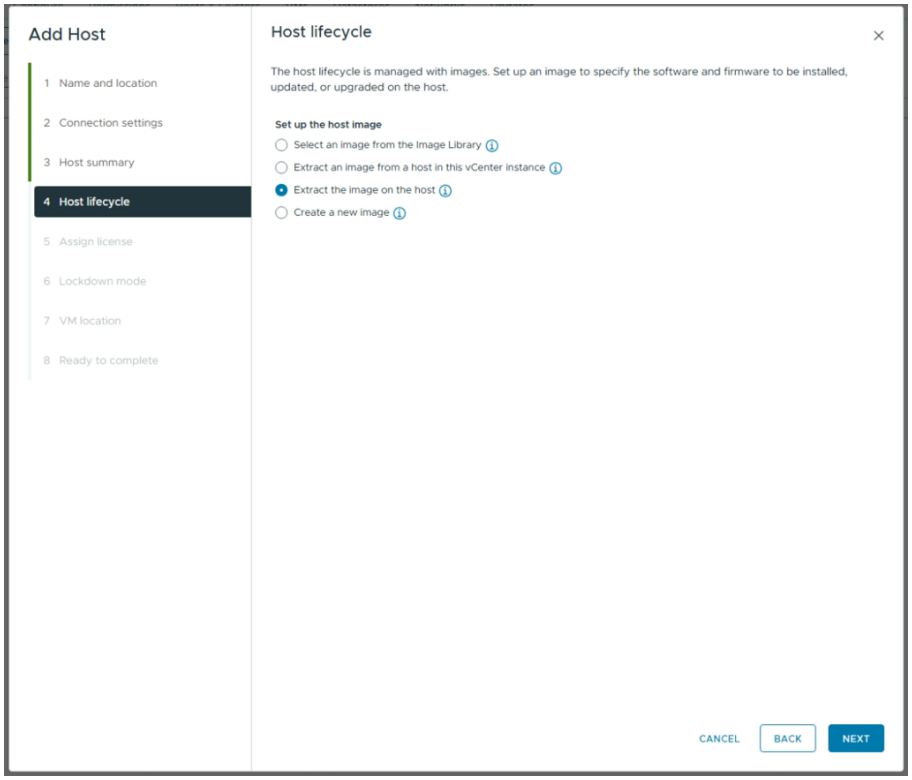


Figure 4.11: vSphere: host lifecycle — extract the image on the host.

Press **Next** in the following tabs until you reach **Lockdown mode**. Choose “Disabled”.

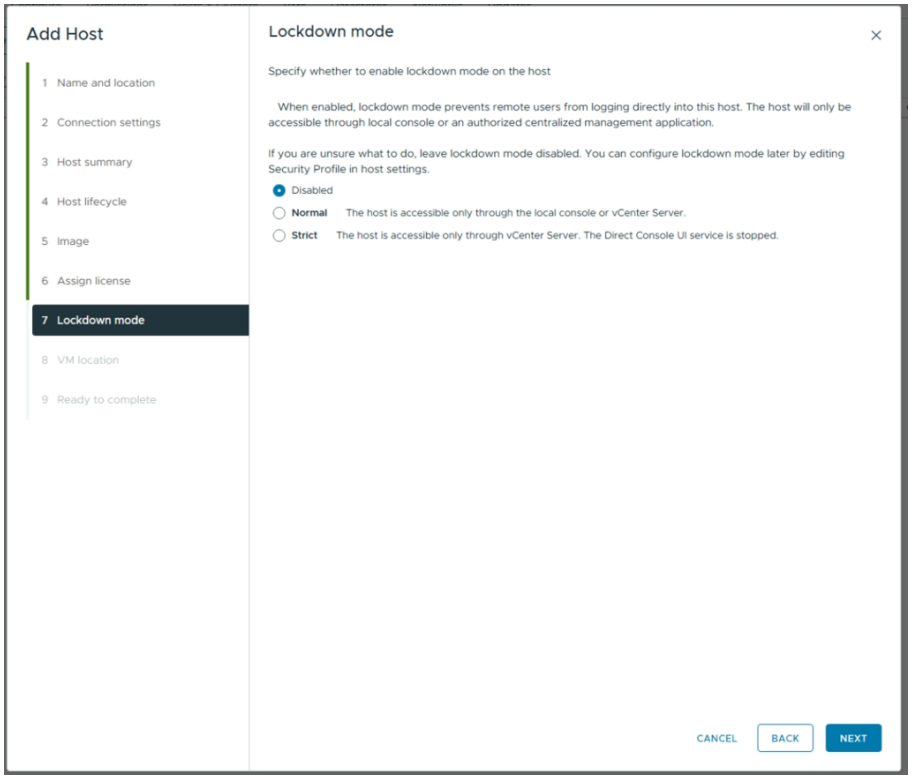


Figure 4.12: vSphere: lockdown mode — disabled.

Press **Next** on the remaining tabs to finish setup. After a few seconds, the host IP address

appears under the datacenter.

4.2.2 Create virtual machines for Linux guests

Provision the **guest operating system** that will run ROCm and your workloads on MI350X after you attach the GPU in §4.2.4. Align the guest type and resources with Chapter 3; this guide qualifies **Ubuntu 24.04** as the Linux distribution for that stack.

vSphere offers several supported ways to obtain an empty or preinstalled guest. Pick the workflow that matches your lab process:

§4.2.2 Create from ISO on the host when you are performing a clean OS install from installation media—typically the *first* guest before you capture a template or clone source.

§4.2.2 Clone an existing VM when a reference machine already reflects your patched, right-sized image.

§4.2.2 Deploy from a template when you maintain a golden image and need repeatable, fast rollout.

The figures in this subsection illustrate representative vSphere Client and ESXi host UI screens. Exact labels and wizard order can vary by vSphere release; translate the intent (ISO media, clone source, datastore, firmware) to your build. Sizing values shown in the ISO example are *illustrative*—apply the guest template from Chapter 3 and your qualification plan. The following subsections follow the order above: **ISO install first**, then **clone**, then **template** deployment.

Create a new virtual machine from an ISO (host UI)

Use this workflow when you perform a *clean* Ubuntu install from installation media in the ESXi host client (or equivalent host-local UI), typically for the first reference guest before you capture a clone source or template. The figures below follow one representative wizard; names and tab order may differ slightly by ESXi build, but the sequence is the same: create the VM shell, size it, set firmware, attach the ISO, then finish and install the OS.

Start the new-VM workflow. Open create/register VM (Figure 4.13), then choose **Create a new virtual machine**.

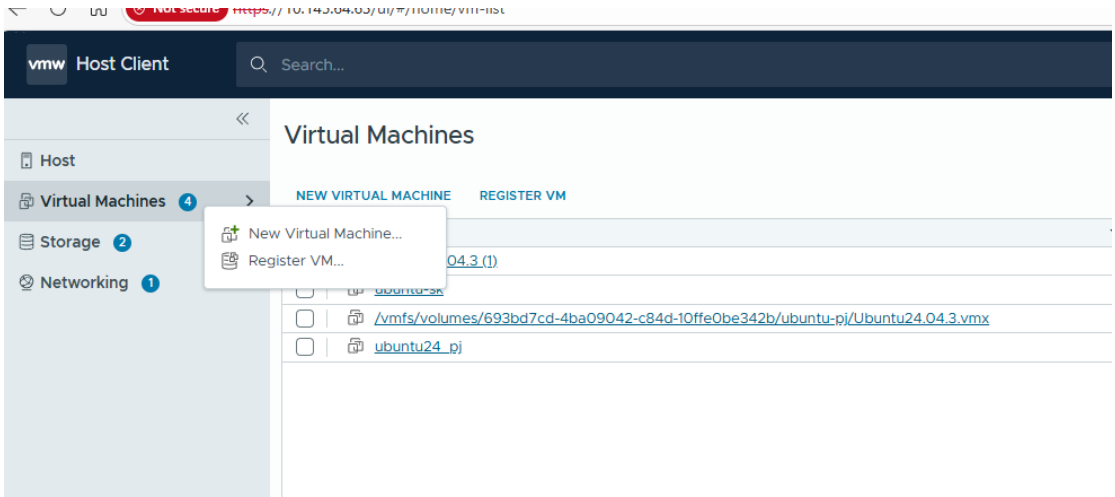


Figure 4.13: ESXi host UI: start create or register virtual machine.

VM Name and compatibility. Set name and hardware compatibility per your cluster (Figure 4.14).

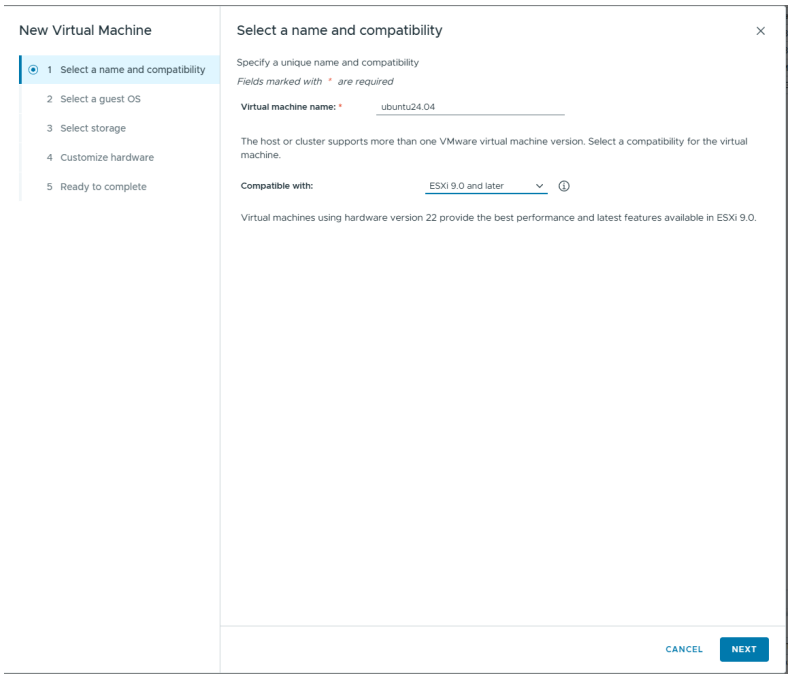


Figure 4.14: New VM: name and compatibility.

Guest OS family and version. Choose **Linux** and the closest 64-bit Ubuntu type offered (e.g. 22.04 if 24.04 is not listed); install Ubuntu 24.04 from the ISO anyway per Chapter 3 (Figure 4.15).

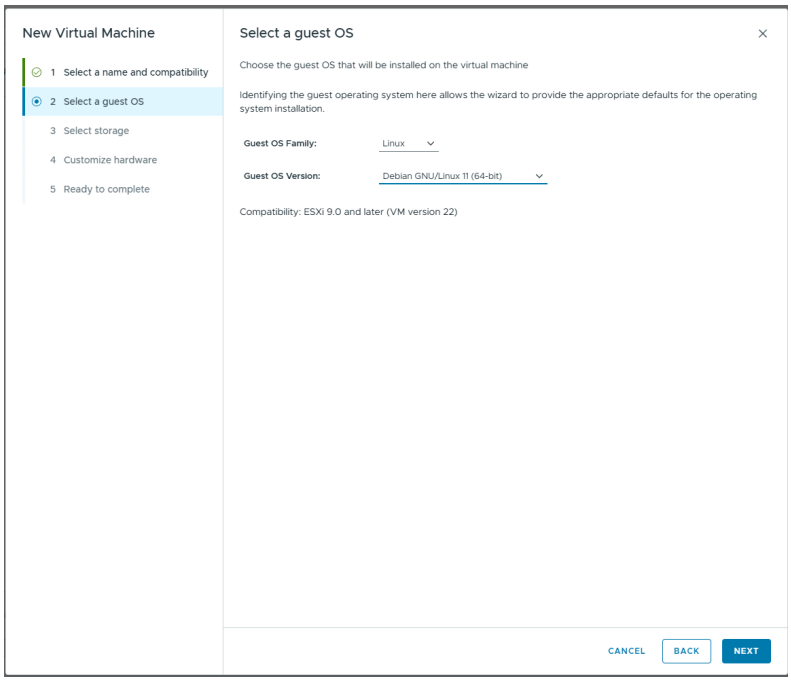


Figure 4.15: New VM: guest OS family and version.

Storage for VM files. Select a datastore for VM files (Figure 4.16); the ISO may live elsewhere if the host can still reach it when you mount CD/DVD.

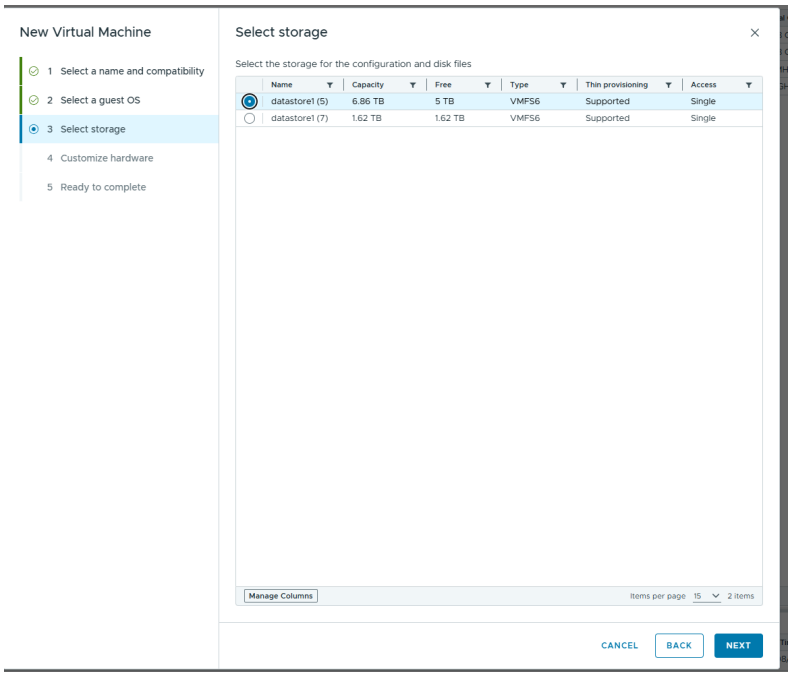


Figure 4.16: Datastore for the new VM's configuration and disks.

CPU, memory, and memory reservation. On the hardware customization pages, set vCPU and RAM to the guest template your qualification plan requires (Chapter 3). For GPU workloads, many labs **reserve all guest memory** (100% reservation) so the hypervisor does not reclaim RAM under pressure (Figure 4.17).

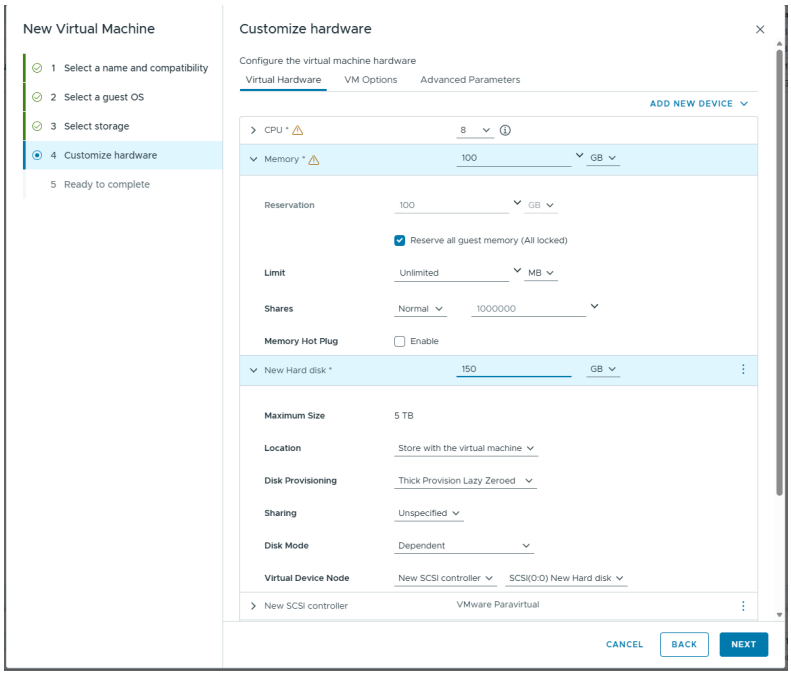


Figure 4.17: CPU, memory, and (if shown) memory reservation.

Firmware and boot. Use **UEFI** (or what your Ubuntu image needs) so the installer boots from virtual optical media (Figure 4.18).

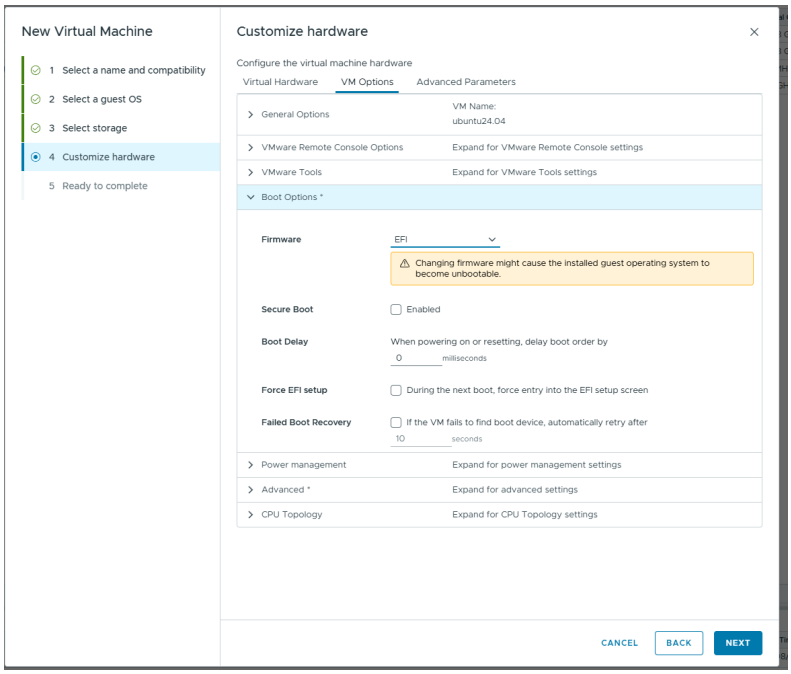


Figure 4.18: Firmware: UEFI for the new VM.

Attach the installer ISO. Mount the Ubuntu ISO on the virtual CD/DVD; set boot order if needed so the guest starts from it (Figure 4.19).

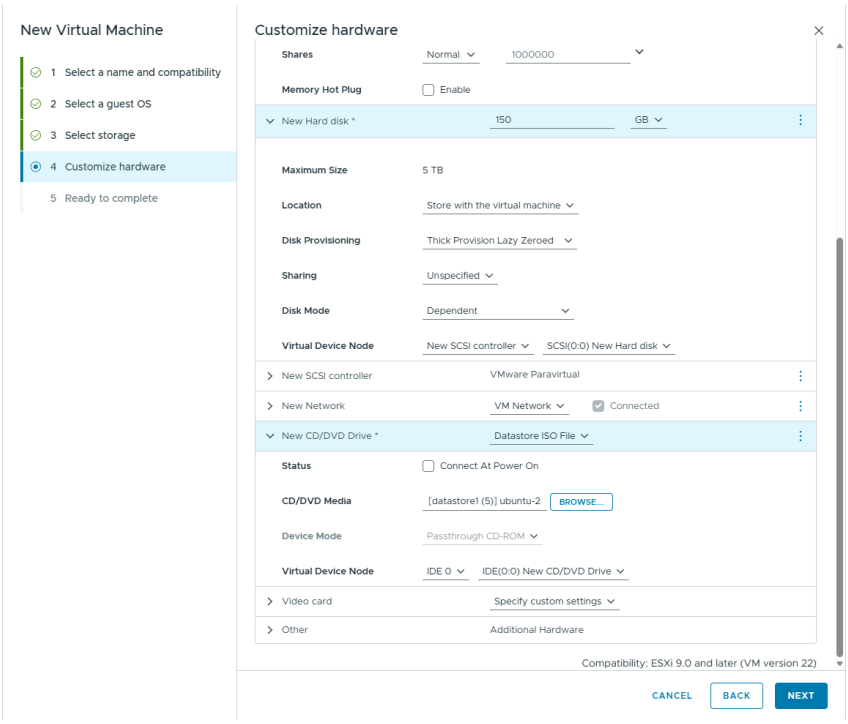


Figure 4.19: Mount the Ubuntu installer ISO on the virtual CD/DVD.

Review and create. Confirm hardware and devices, then finish (Figure 4.20).

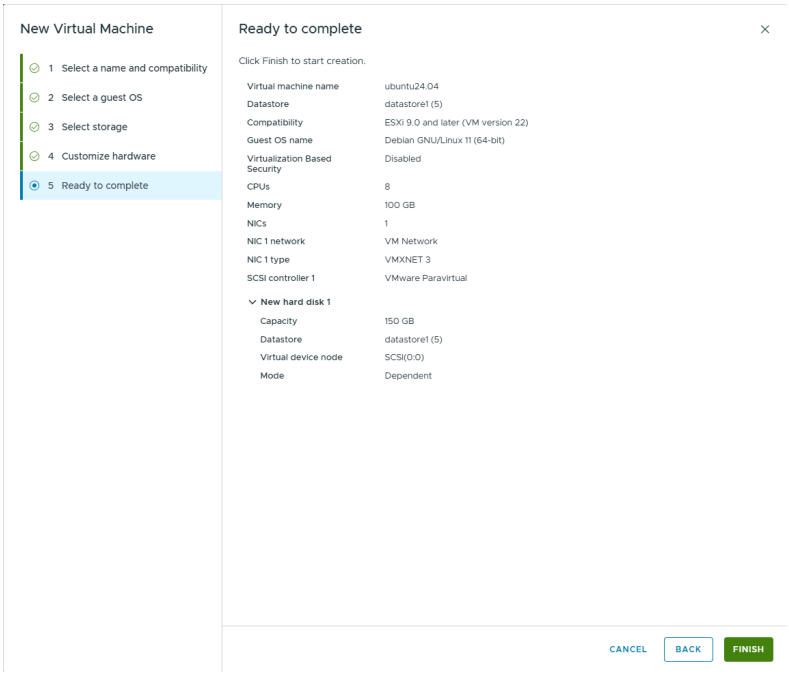


Figure 4.20: Review settings before completing VM creation.

Power on the VM and run through the Ubuntu installer from the virtual CD/DVD. After the OS is stable, then continue with §4.2.3 for guest configuration (or use the VM as the source for §4.2.2 or a template per §4.2.2).

Clone from an existing virtual machine

Use cloning when a source VM already has the correct OS build, tools, and baseline configuration—for example a machine you installed with §4.2.2—and you need additional identical instances for testing. In the vSphere Client, browse to the datacenter and cluster that contain the target ESXi host, select the host, and start the new virtual machine workflow.

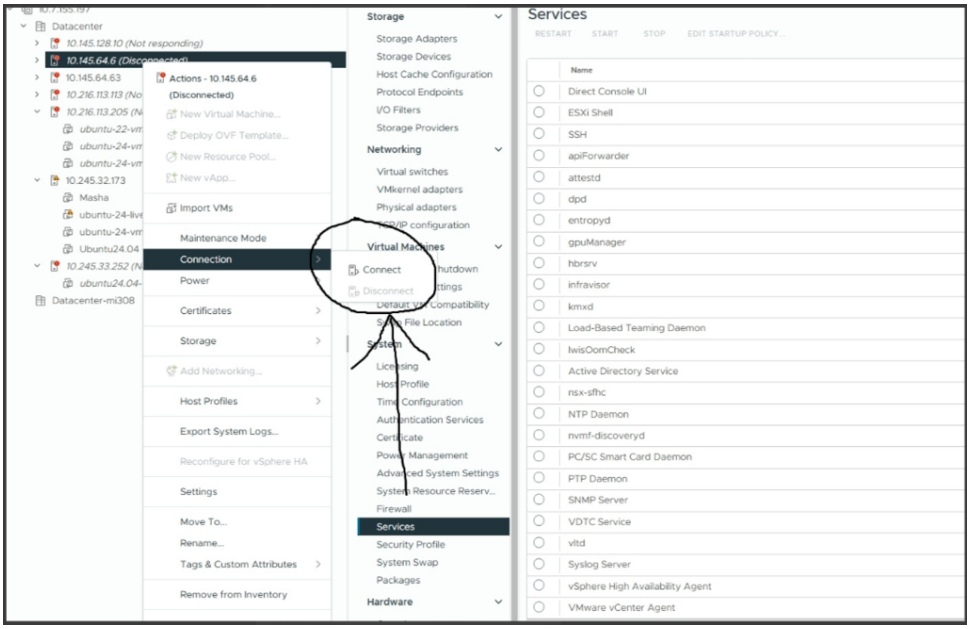


Figure 4.21: Datacenter inventory and host selection (clone workflow).

Open the host context menu or **Actions** menu to create a new virtual machine.

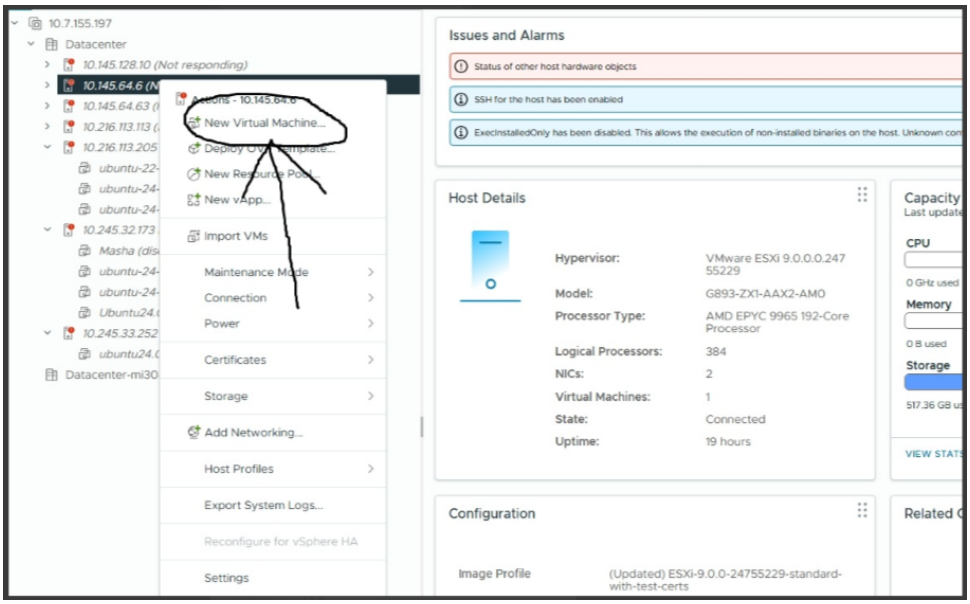


Figure 4.22: Start the new virtual machine wizard from the host.

Choose **Clone an existing virtual machine** and pick the VM that will serve as the source.

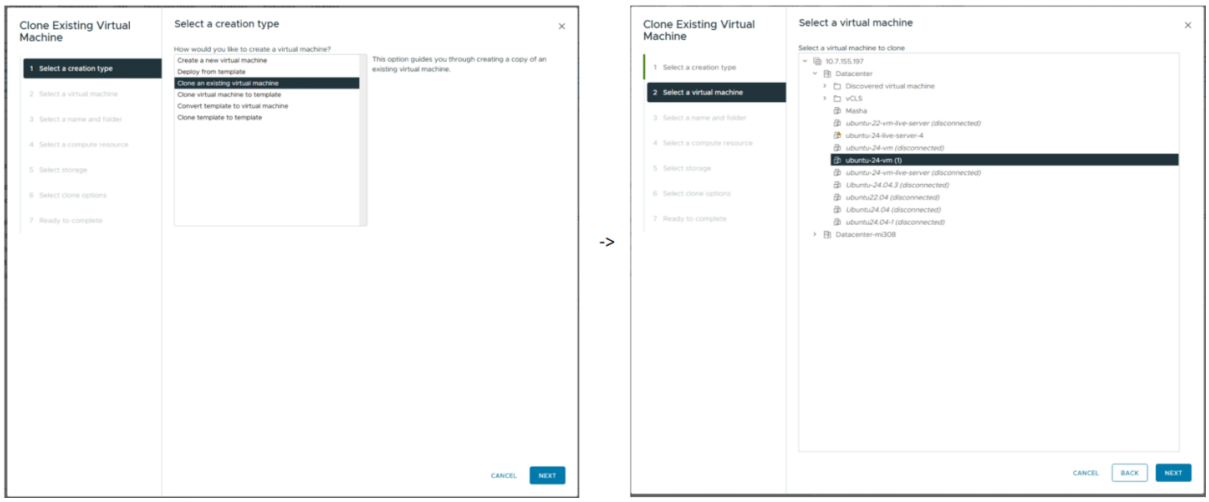


Figure 4.23: Select clone source virtual machine.

Assign a name for the new VM and the compute resource (host or cluster) where it will run.

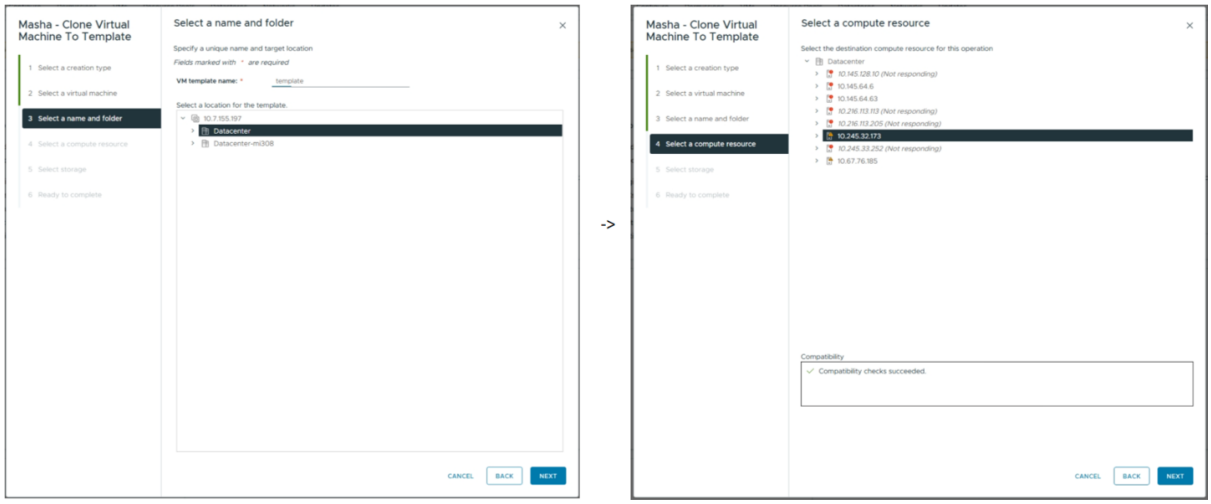


Figure 4.24: Name and placement for the cloned VM.

Select storage (for example a VMFS or vSAN datastore on the target host).

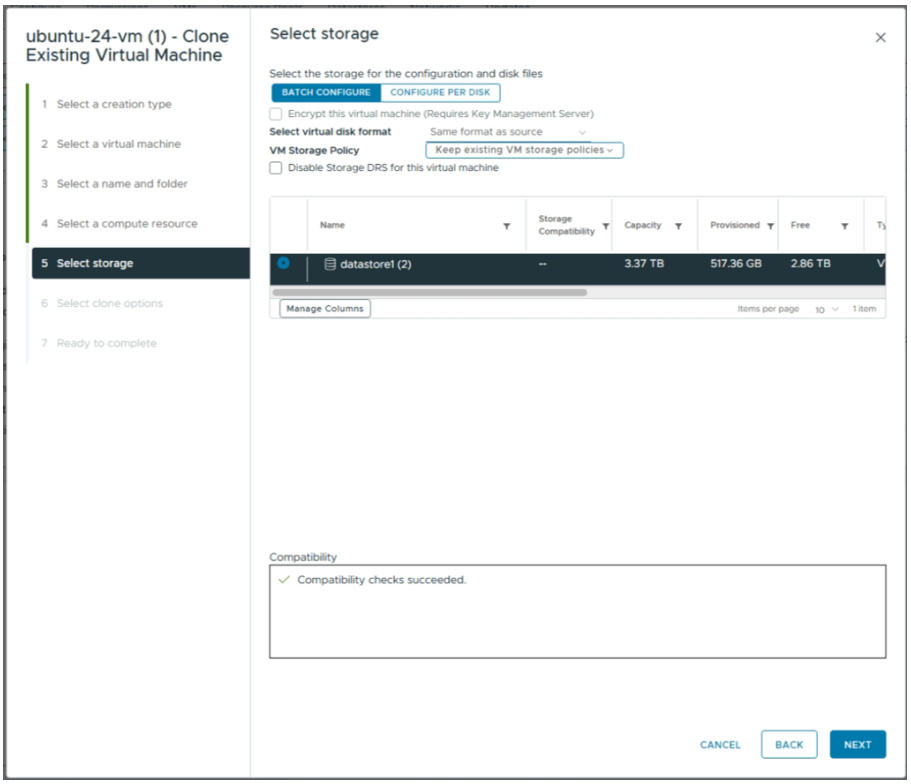


Figure 4.25: Datastore selection for the clone.

Complete any optional customization pages your organization requires, then finish the wizard.

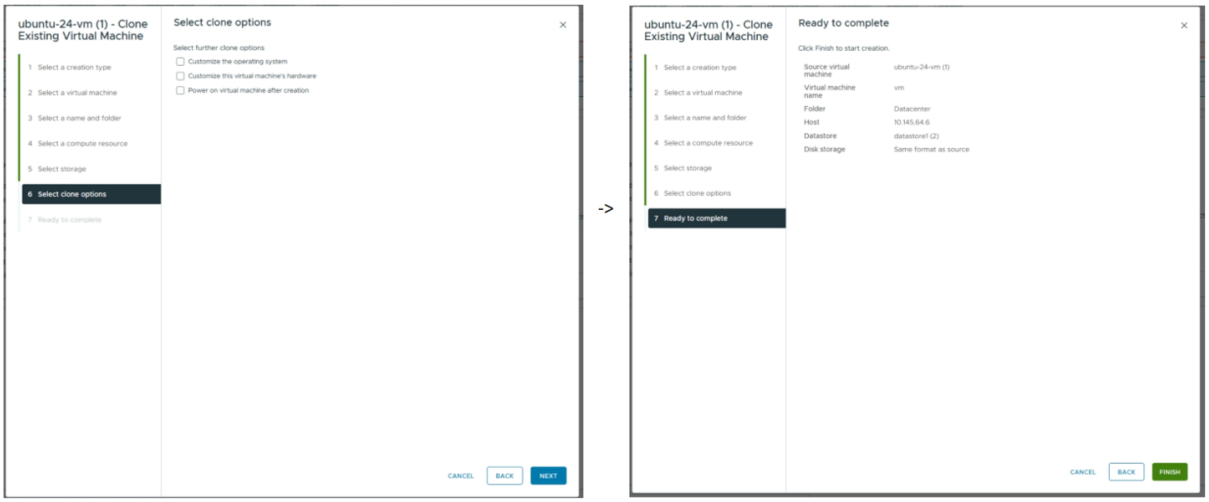


Figure 4.26: Complete the clone wizard.

Deploy from a virtual machine template

A **template** is a powered-off, deployable image derived from a prepared guest. Typical practice is to install and harden the OS once (commonly with the ISO path in §4.2.2), convert or copy the VM to a template, then deploy new VMs from that template for scale and consistency. The first figures show selecting the VM that will supply the template content.

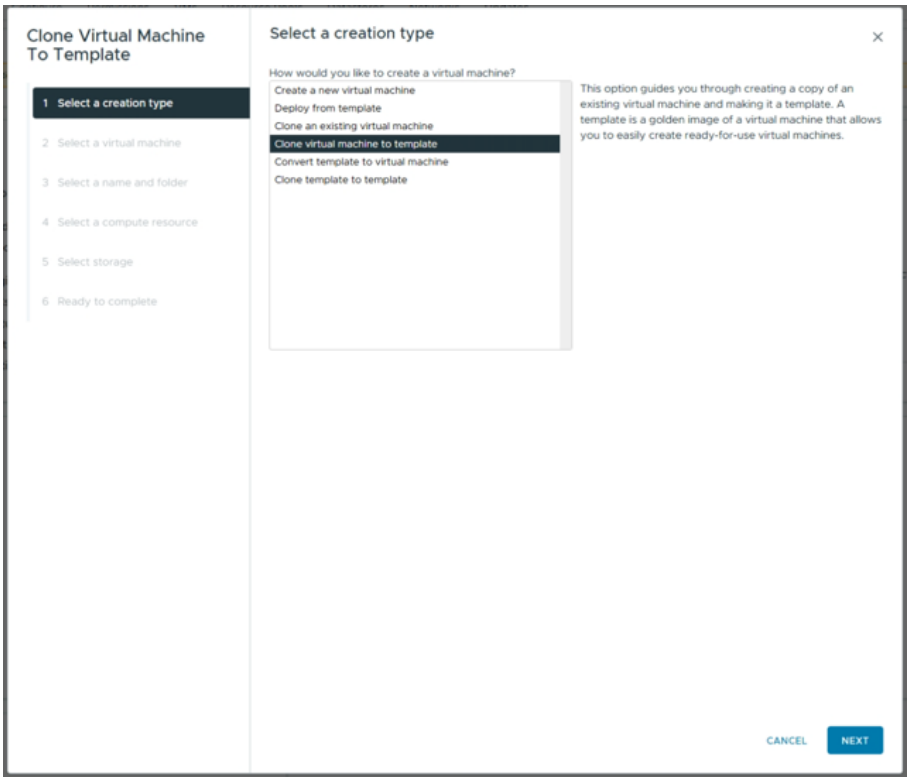


Figure 4.27: Choose the virtual machine to convert or use as template source.

Confirm the source VM for the template operation.

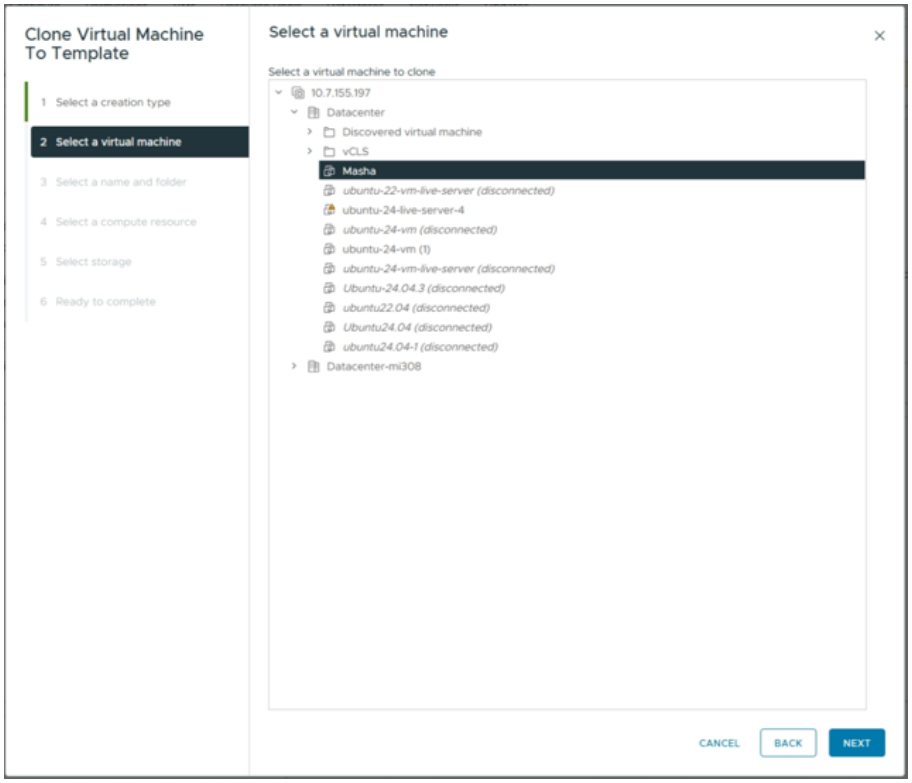


Figure 4.28: Template source virtual machine.

After the template exists in inventory, create a new virtual machine and select **Deploy from template**. Naming, placement, storage, and completion steps parallel the clone workflow in §4.2.2.

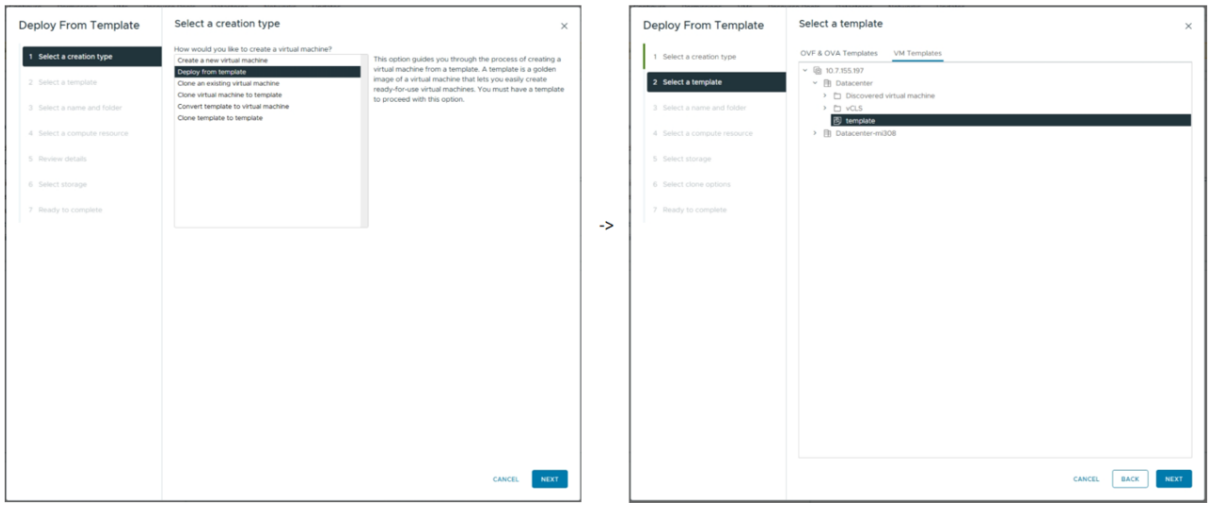


Figure 4.29: Deploy new VM from template.

Work through the remaining wizard pages using the same datastore and resource choices you use for other MI350X guests.

4.2.3 Set up VM environment

Install driver-related packages

Log in to the guest VM (console or SSH) and apply a minimal administrative baseline so you can install software and edit configuration files remotely.

Linux guest shell: base packages and SSH

```
sudo passwd root
su root
sudo apt update
sudo apt upgrade
apt-get install vim ssh dcore -y
vi /etc/ssh/sshd_config
# Edit "PermitRootLogin" to "yes"
service sshd restart
```

Blacklist inbox `amdgpu` before ROCm install

Some bring-up flows blacklist the distribution's default `amdgpu` kernel module so it does not autoload at boot *before* you install the qualified AMD ROCm / `amdgpu` stack with `amdgpu-install`. Apply the following on Ubuntu, then reboot.

Linux guest shell: GRUB and module blacklist

```
vi /etc/default/grub
# GRUB_CMDLINE_LINUX_DEFAULT="quiet splash modprobe.blacklist=amdgpu"
update-grub2
update-initramfs -u
reboot
```

Why blacklisting helps in this guide's context:

- It reduces the chance that an *inbox* or mismatched `amdgpu` module binds to the MI350X device presented in the guest (`amdgpuv` / DVX or related assignment) ahead of the ROCm packages you intend to install.
- It keeps early reboots predictable while kernel packages, headers, and user-space components are aligned with your qualified ROCm build.
- It matches lab practice for GPU validation where you want explicit control over when the stack loads, rather than the default Ubuntu driver path.

After `amdgpu-install` completes, follow AMD's documentation and your qualification plan for whether to remove the blacklist entry and allow the supported module load path.

Optional: multi-user (headless) target

If your test matrix requires a text-only default runlevel (for example guest reload or headless automation), set the default systemd target and reboot.

Linux guest shell: multi-user default target

```
sudo systemctl set-default multi-user.target
reboot
```

4.2.4 Assigning the DVX `amdgpuv` device group to the virtual machine

Chapter 3 documents **one supported** DVX assignment model for this guide: **eight MI350X devices in one Linux VM** (Table 3.3, **1 VM × 8 GPU**). You do **not** add each GPU as a separate `amdgpuv` PCI entry. Instead, add the **Vendor Device Group (VDG)** once: in the device picker, select the group named `com.amd.amdgpuv.xgmi8`, which represents the full XGMI eight-GPU hive as a single assignment (Figure 4.32).
Power the VM **off** while you change PCI / device-group assignments unless your vSphere build explicitly allows hot-add; confirm for your client version and any host compliance warnings.

In the vSphere Client, select the target virtual machine, open **Actions → Edit Settings**.

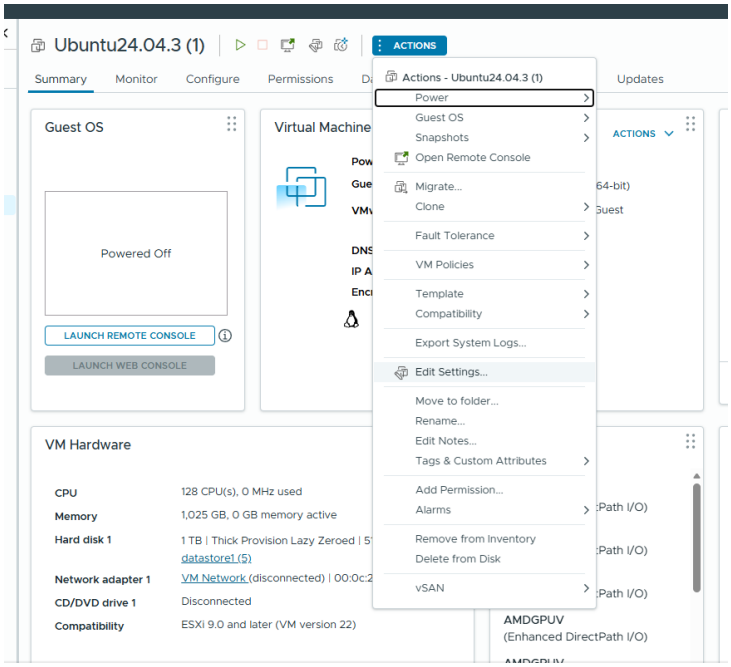


Figure 4.30: Open virtual machine settings from the VM **Actions** menu.

On the **Virtual Hardware** tab, use **Add new device** (or **Add other device**, depending on release). Choose **PCI device** to open the list of host DVX resources you can attach to the VM.

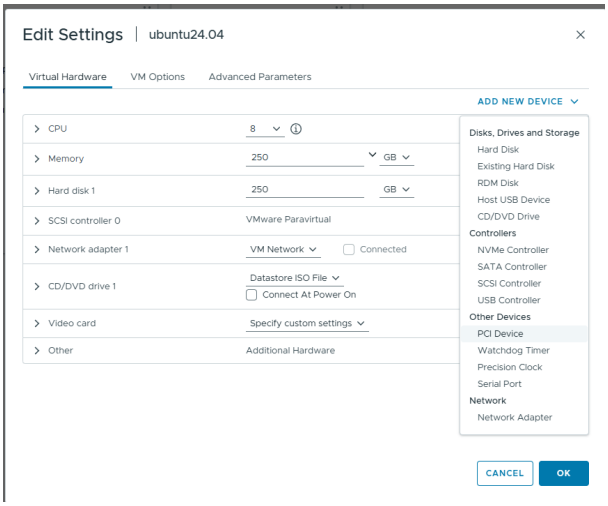


Figure 4.31: Add a new device: select **PCI Device** from the device list.

From the list, select the **Vendor Device Group** `com.amd.amdgpuv.xgmi8` (not eight individual `amdgpuv` lines).

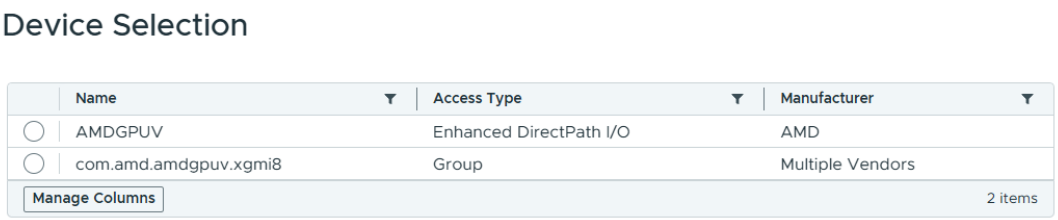


Figure 4.32: Select VDG `com.amd.amdgpuv.xgmi8` to attach the full eight-GPU `amdgpuv` hive in one step.

Power on the VM. In **Summary** or **Edit Settings**, confirm that the `com.amd.amdgpuv.xgmi8` group (or its equivalent hardware summary) is present and that the guest sees all eight adapters before ROCm or validation work.

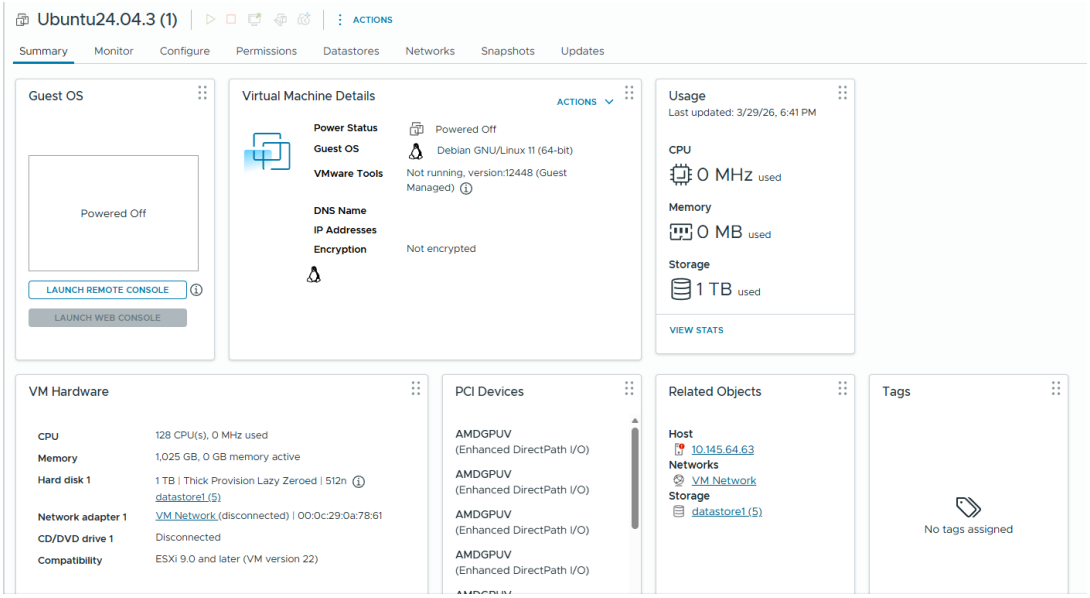


Figure 4.33: Verify the assigned DVX / PCI device group in the VM hardware view.

4.2.5 Install guest driver and ROCm

With Ubuntu 24.04 running and the MI350X exposed through `amdgpuv`, install the guest **Linux `amdgpu` driver and ROCm user space** at the revision your program qualifies. The requirement floor in Table 3.6 (Chapter 3) is **ROCm 7.2.1 and later**; pick the exact ROCm release, repository, and optional components that match your program's validation matrix.

The AMD ROCm™ Software Stack and related Radeon™ software for Linux are installed using the `amdgpu-install` utility, which resolves a coherent set of packages for your distribution and ROCm version. Use AMD's current documentation: ROCm installation for Linux (*ROCm installation (Linux)*). Choose Ubuntu 24.04–appropriate install options and the ROCm build you certify; complete any reboot or `modprobe` steps the guide requires before workload or validation activities.

Chapter 5: Reference

This chapter lists external sites and documentation that the guide cites for initial bring-up. Replace program-specific placeholders when AMD publishes final download locations for the ESXi `amdgpuv` package (publisher firmware rules: Chapter 3).

5.1 Web resources

Broadcom Support Portal `https://support.broadcom.com/`

Use your entitled account to obtain VMware ESXi installation ISOs, the vCenter Server Appliance (VCSA) media, and related Broadcom/VMware support content referenced in Chapter 2 (accelerator architecture), Chapter 3 (management and datacenter host software), and Chapter 4 (ESXi on the GPU host).

ROCm installation (Linux) `https://rocm.docs.amd.com/projects/install-on-linux/en/latest/`

AMD's current guide for installing the ROCm stack on Linux distributions, including use of `amdgpu-install`; follow this in the guest after the MI350X is exposed through `amdgpuv`, as described in Chapter 4.

AMD Instinct Virtualization Driver `https://instinct.docs.amd.com/projects/virt-drv/en/latest/index.html`

AMD documentation for Instinct GPU virtualization with SR-IOV, including host and VM setup workflows and related topics (for example XGMI configuration). The site is oriented to QEMU/KVM deployments; use it alongside this guide's VMware ESXi and `amdgpuv` material.

AMD `amdgpuv` driver (ESXi); publisher firmware The `amdgpuv` VIB is distributed by AMD outside Broadcom's ESXi and vCenter catalogs. See Chapter 3, **Obtaining the `amdgpuv` VIB (AMD)**, for the **Instinct accelerators** link and the **Driver & support** path on MI350X / MI355X (and related) product pages. For the publisher firmware bundle (terminology, version **26.00** floor, and delivery rules), see the **PLDM / BKC / platform firmware** callout in §3.2.2 (Chapter 3).

- **`amdgpuv` (ESXi):** `https://www.amd.com/en/products/accelerators/instinct.html`

Additional environment parameters, lab operations, and test cases can be added here or in later revisions as the guide expands.

Appendix A: Revision History

The following table shows the revision history for this document.

Table A.1: Revision history

Date	Version	Summary
April 2026	1.0	Release 1.0: overview, hardware and software requirements, MI350X/MI355X architecture, and system configuration (host SR-IOV/VF, `amdgpuv` VIB, VDG `com.amd.amdgpuv.xgmi8`, guest ROCm). Scope: VMware ESXi 9.1 only; one Linux VM with eight MI350X devices (`MODE_8`); other VM×GPU splits out of scope.
July 2026	1.1	Release 1.1: title and scope cover MI350X/MI355X; MI350X/MI355X naming note (procedures written for MI350X, equivalent on MI355X); qualified-platform accelerators column; editorial updates.

Appendix B: Notices

© Copyright 2025 Advanced Micro Devices, Inc.

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED "AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

B.1 Trademarks

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Ubuntu and the Ubuntu logo are registered trademarks of Canonical Ltd.

Dell is a trademark of Dell Inc. or its subsidiaries.

Broadcom is a trademark of Broadcom Limited in the United States and/or other countries.

Supermicro is a trademark or registered trademark of Super Micro Computer, Inc. or its subsidiaries in the United States and other countries.