Proxmox VE 9.0: Windows 11 vGPU (VT-d) Passthrough with Intel Alder Lake

July 28, 2024 in **Proxmox**

Tags: GPU Passthrough SR-IOV vGPU VT-d Windows 11

Interested in sharing your Intel Alder Lake GPU between multiple Proxmox 8.x or 9.x VMs? This post covers configuring Intel vGPU virtual functions on Proxmox 9.0 using an Intel 12th Generation CPU (Alder Lake) with Windows 11 Pro 25H2. I will walk you through the Proxmox 9.0 kernel configuration, modifying GRUB, Installing Windows 11 Pro, and then setting up the Intel graphics inside Windows.

I've completely updated the content from my Proxmox 8.0 vGPU post which appeared on my blog back in June 2024. While the process is generally the same as before, I've added new sections and refreshed all the prior content. I've redirected the old blog post URL to this post. In case you need the old post you can view it here: Proxmox VE 8: Windows 11 vGPU (VT-d) Passthrough with Intel Alder Lake. It has a lot of user comments, which you might find helpful. However, use this post for the installation process on Proxmox 9.0.

Update October 4, 2025: Validated that Windows 11 25H2 works as expected. Totally revamped the local account procedure for RDP. Added a new section to prevent Windows update from installing old Intel drivers, breaking vGPU. Various other screenshot updates.

Update September 30, 2025: I've done Proxmox 9.0 testing, and found that no changes to my procedure are needed. I used Intel drivers 32.0.101.7076 with Kernel 6.14.11-3-pve. Updated text to reflect Proxmox 9.0, and updated Community scripts URL.

Update January 1, 2025: Significant updates to the post for Windows 11 24H2, Proxmox kernel 6.8.12-5 and Proxmox 8.3. Minor tweak to kernel configuration steps. Updated Proxmox community script link for kernel cleanup. Several updated screenshots.

Update July 28, 2024: Significant update to the steps, as Linux kernels 6.1-6.9 are now supported by the DKMS module. I used PVE Kernel 6.8.8-4 without issue. As always, you can find the latest official repo here: i915-sriov-dkms.

Also, to clear out some 'junk' from my previous 8.0/8.1 install with DKMS modules I had to perform a clean install of Proxmox PVE 8.2. Once I did the clean install the steps in this post worked like charm. Restoring my VMs and LXCs from Proxmox Backup Server (PBS) was a piece of cake and very quick.

Finally, on this updated stack and latest Intel driver for Windows, the Parsec app is working for me.

Update May 5, 2024: Kernel 6.5.13-5 was giving me problems, as well as readers, on some systems with secure boot. A reader suggested 6.5.13-3, which DID work on the one system that 6.5.13-5 had issues on. So I updated the steps to use 6.5.13-3, as that seems more widely compatible for users with secure boot.

Update May 4, 2024: I've updated the instructions for Proxmox 8.2. This requires pinning kernel 6.5.13-5 in order for vGPU/DKMS to work. You can NOT use the 6.8.x kernel branch with vGPU.

Update April 29, 2024: Added installing sysfsutils in the first Proxmox CLI section. I also updated the tested Proxmox kernel and Arc driver versions.

Update April 25, 2024: Proxmox 8.2 is now out, but ships with Kernel 6.8x. DKMS is broken on this kernel. DO NOT use this procedure on a vanilla Proxmox 8.2 install with Kernel 6.8. Either wait to upgrade to Proxmox 8.2, or pin the kernel to a prior 6.5 kernel that still works with DKMS and then upgrade.

Update Dec 7, 2023: Proxmox has released a minor kernel upgrade. Kernel 6.5.11-7-pve. If you update from the Proxmox UI, it looks like DKMS rebuilds the kernel and you should be good to go. However, in my experience your 7 VFs will vanish if you do nothing. As my article states, just re-run the all the GRUB steps which rebuild DKMS from scratch, then reboot. Now my 7 VFs are back. This is why it is wise to pin your kernel version and only upgrade as needed. I also added a bit to the troubleshooting section.

Update December 2, 2023: Added a screenshot for the secure boot MOK setup.

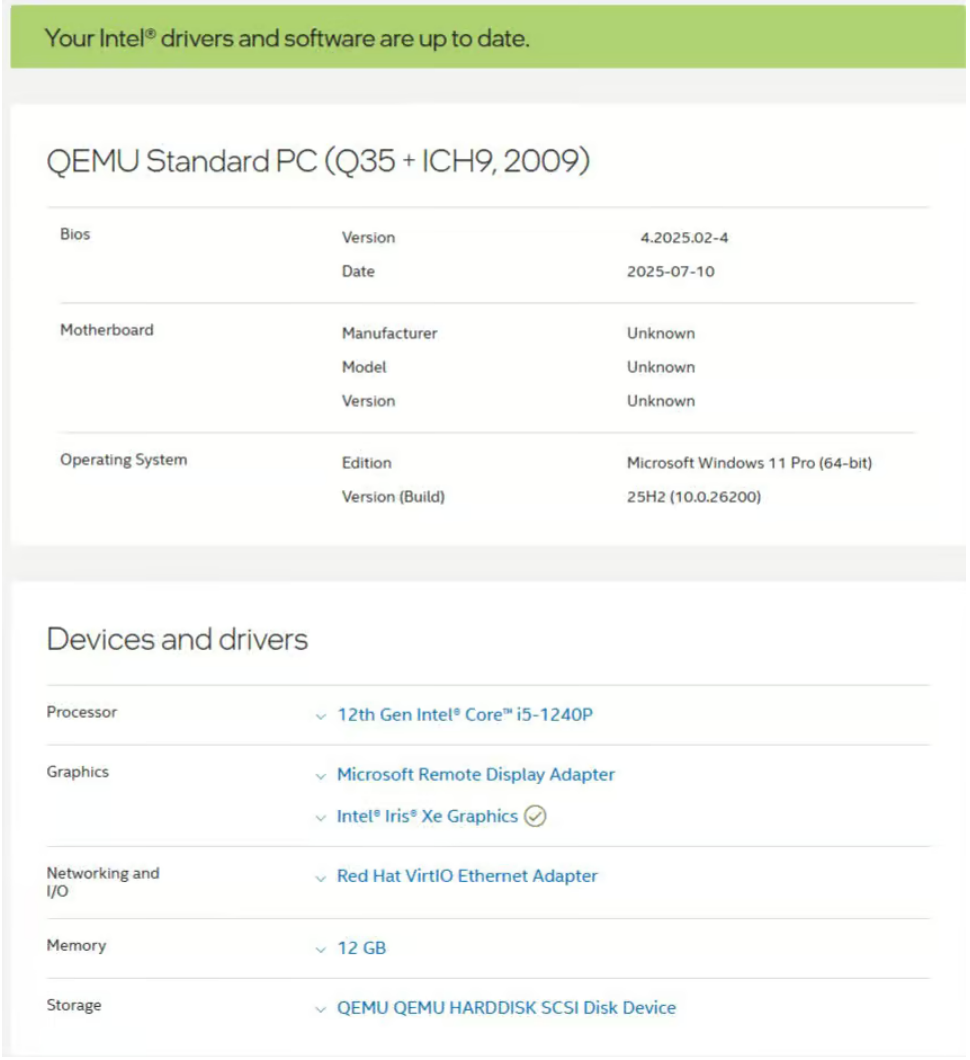
Update November 27, 2023: Added an additional step for Proxmox 8.1 or later installs that use secure boot. Updated Plex information, as their stance has now changed. They will work on SR-IOV (vGPU) support for a future release.

Update November 26, 2023: I've modified both the DKMS and GRUB sections so they are now responsive to the kernel version your Proxmox host is running. It also performs more system cleanup, and is pretty much just a copy/paste endeavor now. The changes were inspired by a Github comment, which I modified for even more automation.

Update November 25, 2023: The DKMS repo has been updated to address the minor tweak needed for Kernel 6.5. So I removed the step to modify one of the DKMS files. I also added a LXC vGPU VF section to address other services like Plex and their vGPU support.

Proof is in the Pudding

By the time you finish following my guide, you should see something similar to the screenshot below for your Windows 11 Pro 25H2 Proxmox VM. This is using the latest Proxmox 9.0 6.14 kernel, Intel drivers, and Windows updates as of September 30, 2025.



Working vGPU Support on Windows Pro 11 25H2 + Proxmox 9.0

GPU Passthrough vs Virtualization

Typical GPU passthrough passes the entire PCIe graphics device to the VM. This means only one VM can use the GPU resources. This has an advantage of being able to output the video display via the computer's HDMI or DisplayPort ports to an external monitor. But, you are limited to one VM using the GPU. This can be useful if you want Proxmox on your desktop computer, but also want to fully use the GPU resources for your primary desktop OS with an external monitor.

However there is a way to share the GPU among VMs. This technology called Intel VT-d (Intel Virtualization Technology for Directed I/O) which enables the virtualization of the GPU resources and a present VF (virtual function) to up to 7 VMs. This enables (up to) 7 VMs to concurrently use the GPU, but you lose the ability to use a physically connected monitor. Thus you are limited to Remote Desktop access only for these VMs.

Full GPU passthrough and vGPU setups both have their place. It totally depends on what you want your setup to do. This post will focus on vGPU configuration and sharing your GPU with up to 7 Proxmox VMs. I've only tested this with Windows 11 VMs. Linux VMs, in the past, have had Intel driver issues that caused some issues with vGPUs. So test thoroughly if you want to try vGPU with Linux VMs.

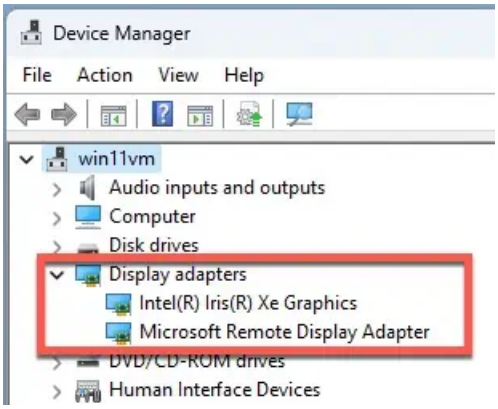
What GPU does Windows 11 see?

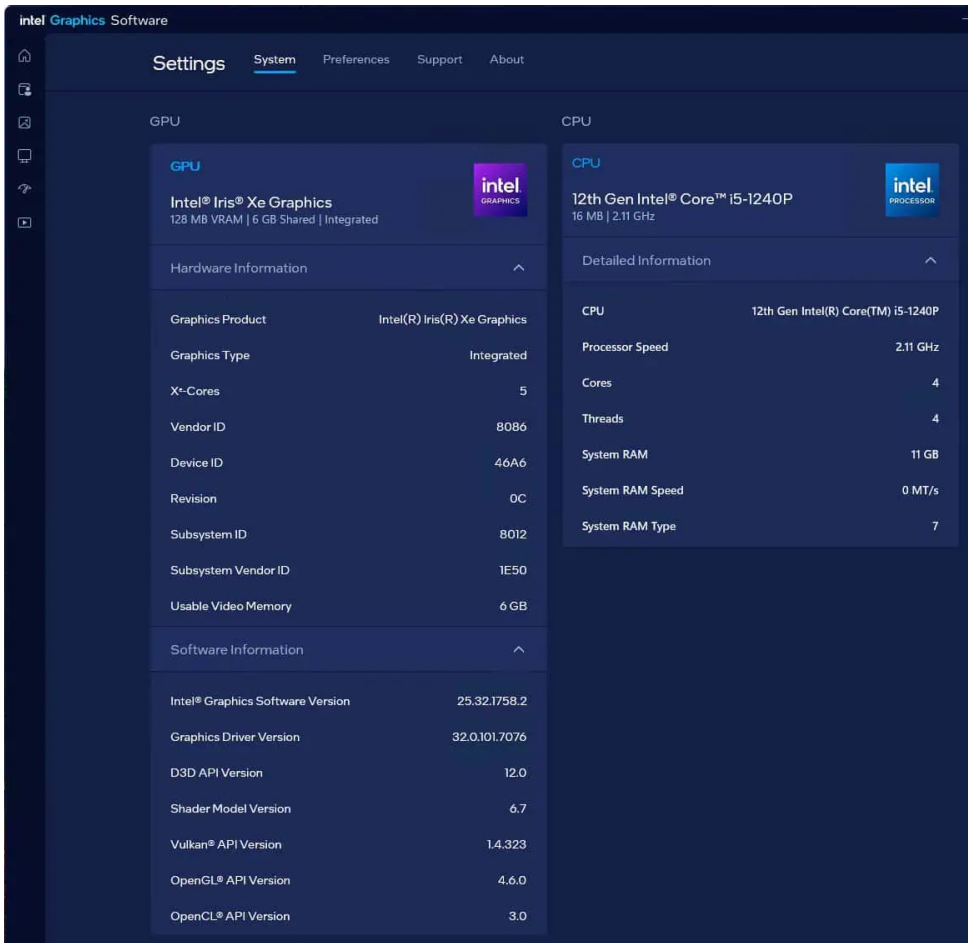
When using a vGPU VF the OS does not know you have virtualized the GPU. The

screenshots below are from my Beelink i5-1240P Proxmox VE 8.1 host. The “root” GPU is at PCIe 02.0, and you can see 7 GPU VFs (virtual functions).

ID ↑	IOMM…	Vendor	Device	Medi…
0000:00:02.0	0	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.1	18	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.2	19	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.3	20	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.4	21	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.5	22	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.6	23	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.7	24	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:0d.0	5	Intel Corporation	Alder Lake-P Thunderbolt 4 USB Controller	No
0000:00:0d.2	5	Intel Corporation	Alder Lake-P Thunderbolt 4 NHI #0	No
0000:00:0d.3	5	Intel Corporation	Alder Lake-P Thunderbolt 4 NHI #1	No

I’ve assigned one GPU VF to a Windows 11 Pro VM (00:02.1). From the Windows perspective the stock WHQL Intel driver works as normal. The GPU even shows up as an Intel Iris Xe. You can also look at the Intel Arc control application and view the various hardware details of the GPU.





My Working Configuration

Not all Intel CPUs support VT-d with their GPU. Intel only supports it on 11th Generation and later CPUs. It considers prior generations "legacy". I've seen forum posts that users have issues with Intel 11th Gen, so that may or may not work for you. My latest configuration is as follows:

- Beelink SEi12 Pro (Intel 12th Gen Core i5-1240P)
- Proxmox VE 9.0 (**with Linux 6.14.11-3-pve**)
- Windows 11 Pro (25H2)
- Intel GPU Driver 32.0.101.7076 (09/12/2025)

Note: Your Proxmox host must have **Intel VT-d** (or whatever they call it) enabled in your BIOS, and your motherboard must properly implement it. Not all motherboards have working VT-d for GPUs. If this is disabled you will see "NO IOMMU detected" when you try and add the VF PCI device to the Windows VM. You BIOS may also have a feature called something like **SR-IOV** that needs to be enabled as well.

Note: Several readers left comments on my 8.0 post about the Parsec app not working. However, with as of 10/04/2025 using the above configuration the Parsec app launches just fine.

What about LXC vGPU VF Compatibility?

If you are nerdy enough to run a Windows 11 VM with a vGPU, you might also have some Linux LXCs that can use GPU resources as well (such as Plex). Linux LXC compatibility with vGPU VFs might be problematic.

As of the publication date of this post, a Plex LXC with Ubuntu 22.04 has problems with HDR tone mapping. Basically the Linux Intel Media Driver (IMD) doesn't like using a vGPU VF. If you enable HDR tone mapping on the Plex server and you are viewing HDR content on a device that needs server side HDR tone mapping, the video stream will likely be corrupted.

However, **Plex hardware transcoding will still work**. Hardware transcoding uses the XE graphics module in the CPU (not GPU), whereas HDR tone mapping uses the GPU itself. Meaning, these two hardware offloads use different APIs and the GPU offload for HDR tone mapping is broken when using a vGPU VF.

Chuck from Plex posted on the forums on that he consulted with his engineers, and they will work on making Plex compatible SR-IOV. However, the needed kernel mods or GRUB updates will be up to the user to make. Intel has mainline SR-IOV for the Linux 6.4 kernel, and is working on 6.5. You can follow the progress in SR-IOV Mainling.

Bottom line, if you have LXCs on a Proxmox host that need GPU resources, they may not work with a vGPU VF. When Intel releases their official 6.5 package, it should offer wider compatibility. Configuring a LXC to use a vGPU VF requires modifying the LXC config file. Since this post is about Windows 11 vGPU, that procedure is out of scope for this post. I'll cover the required LXC config changes in a separate post.

Proxmox Kernel Configuration

Note: If you have previously pinned your Proxmox kernel, unpin the kernel then reboot the Proxmox host into the new kernel that you want to modify for vGPU support. Pinning and unpinning instructions are at the end of this post.

1. On your Proxmox host open a **shell** and run the following commands. First we need to install Git, kernel headers and do a bit of cleanup.

```
apt update && apt install git sysfsutils pve-headers mokutil -y
rm -rf /usr/src/i915-sriov-dkms-*
rm -rf /var/lib/dkms/i915-sriov-dkms
rm -rf ~/i915-sriov-dkms*
find /lib/modules -regex ".*/updates/dkms/i915.ko" -delete
```

2. Now we need to clone the DKMS repo and do a little build work.

```
cd ~
git clone https://github.com/strongtz/i915-sriov-dkms.git
apt install build-* dkms
cd ~/i915-sriov-dkms
dkms add .
```

3. Let's now build the new kernel and check the status. Validate that it shows **installed**.

```
VERSION=$(dkms status -m i915-sriov-dkms | cut -d':' -f1)
dkms install -m $VERSION --force
dkms status
```

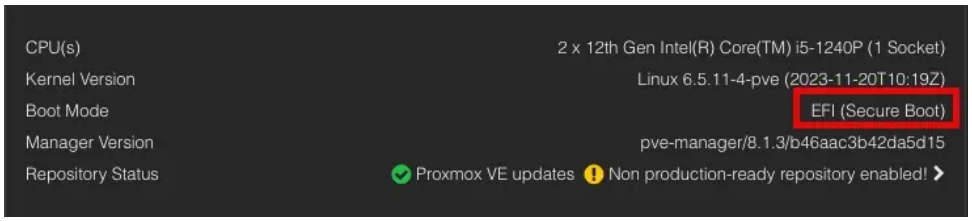
```
root@proxmox1:~/i915-sriov-dkms# apt update && apt install git sysfsutils pve-headers mokutil -y
rm -rf /usr/src/i915-sriov-dkms-*
rm -rf /var/lib/dkms/i915-sriov-dkms
rm -rf ~/i915-sriov-dkms*
find /lib/modules -regex ".*/updates/dkms/i915.ko" -delete
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://security.debian.org/debian-security bookworm-security InRelease
Hit:3 http://deb.debian.org/debian bookworm-updates InRelease
Hit:4 http://download.proxmox.com/debian/pve bookworm InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.39.5-0+deb12u1).
sysfsutils is already the newest version (2.1.1-4).
pve-headers is already the newest version (8.3.0).
mokutil is already the newest version (0.6.0-2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@proxmox1:~/i915-sriov-dkms# cd ~
git clone https://github.com/strongtz/i915-sriov-dkms.git
apt install build-* dkms
cd ~/i915-sriov-dkms
dkms add .
Cloning into 'i915-sriov-dkms'...
remote: Enumerating objects: 3675, done.
remote: Counting objects: 100% (1039/1039), done.
remote: Compressing objects: 100% (383/383), done.
remote: Total 3675 (delta 767), reused 665 (delta 654), pack-reused 2636 (from 1)
Receiving objects: 100% (3675/3675), 4.82 MiB | 15.52 MiB/s, done.
Resolving deltas: 100% (2236/2236), done.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'build-essential' for glob 'build-*'
build-essential is already the newest version (12.9).
dkms is already the newest version (3.0.10-8+deb12u1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Creating symlink /var/lib/dkms/i915-sriov-dkms/2024.12.30/source -> /usr/src/i915-sriov-dkms-2024.12.30
root@proxmox1:~/i915-sriov-dkms# VERSION=$(dkms status -m i915-sriov-dkms | cut -d':' -f1)
dkms install -m $VERSION --force
dkms status
Sign command: /lib/modules/6.8.12-5-pve/build/scripts/sign-file
Signing key: /var/lib/dkms/mok.key
Public certificate (MOK): /var/lib/dkms/mok.pub

Building module:
Cleaning build area...
make -j16 KERNELRELEASE=6.8.12-5-pve -C /lib/modules/6.8.12-5-pve/build M=/var/lib/dkms/i915-sriov-dkms/2024.12.30/build.....
Signing module /var/lib/dkms/i915-sriov-dkms/2024.12.30/build/i915.ko
Cleaning build area...

i915.ko:
Running module version sanity check.
 - Original module
 - Installation
   - Installing to /lib/modules/6.8.12-5-pve/updates/dkms/
depmod...
i915-sriov-dkms/2024.12.30, 6.8.12-5-pve, x86_64: installed
root@proxmox1:~/i915-sriov-dkms#
```

5. For fresh Proxmox 8.1 and later installs, secure boot may be enabled. Just in case it is, we need to load the DKMS key so the kernel will load the module. Run the following

command, then enter a password. This password is only for MOK setup, and will be used again when you reboot the host. After that, the password is not needed. It does NOT need to be the same password as you used for the root account.



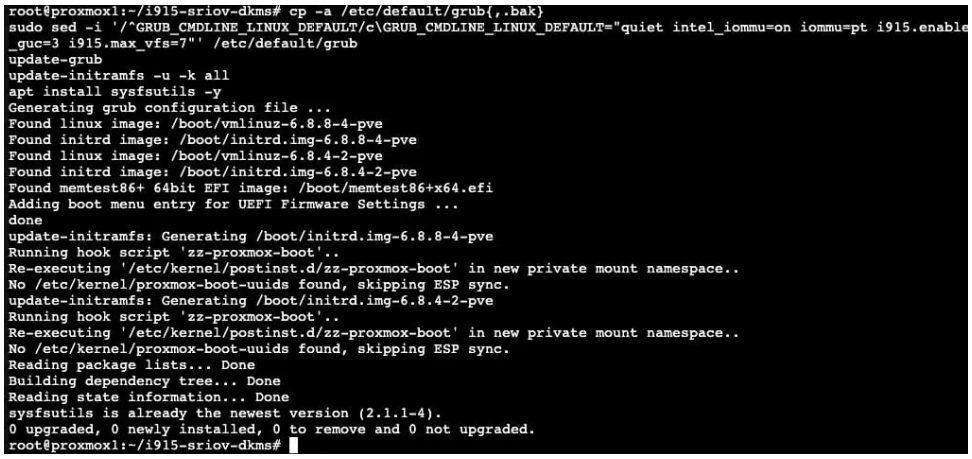

```
mokutil --import /var/lib/dkms/mok.pub
```

Proxmox GRUB Configuration

Note: Default installations of Proxmox use the GRUB boot loader. If that's your situation, follow the steps in this section. If you are using ZFS or another config that uses **systemd bootloader**, skip down to the systemd section below.

1. Back in the Proxmox shell run the following commands if you **DO NOT** have a **Google Coral PCIe TPU** in your Proxmox host. You would know if you did, so if you aren't sure, run the first block of commands. If your Google Coral is USB, use the first block of commands as well. Run the second block of commands if your Google Coral is a PCIe module.

```
cp -a /etc/default/grub{,.bak}
sudo sed -i '/^GRUB_CMDLINE_LINUX_DEFAULT/c\GRUB_CMDLINE_LINUX_DEF
update-grub
update-initramfs -u -k all
```



If your Proxmox host **DOES** have a **Google Coral PCIe TPU** and you are using PCIe passthrough to a LXC or VM, use this command instead. This will blacklist the Coral device at the Proxmox host level so that your LXC/VM can get exclusive access.

```
cp -a /etc/default/grub{,.bak}
```

```
sudo sed -i '/^GRUB_CMDLINE_LINUX_DEFAULT/c\GRUB_CMDLINE_LINUX_DEF
update-grub
update-initramfs -u -k all
apt install sysfsutils -y
```

Proxmox SystemD Bootloader

Note: If your Proxmox host doesn't use GRUB to boot (default), but rather uses systemd, then follow these steps. This is likely the case if you are using ZFS. Skip this section if you are using GRUB.

1. Let's modify the kernel loader command line:

```
nano /etc/kernel/cmdline
```

2. Add the following text to the **END** of the current line. Do NOT add a second line.

```
intel_iommu=on iommu=pt i915.enable_guc=3 i915.max_vfs=7
```

3. Run the following command to update the boot loader.

```
proxmox-boot-tool refresh
```

Finish PCI Configuration

1. Now we need to find which PCIe bus the VGA card is on. It's typically **00:02.0**.

```
root@proxmox1:~# lspci | grep VGA
00:02.0 VGA compatible controller: Intel Corporation Alder Lake-P Integrated Graphics Controller (rev 0c)
root@proxmox1:~#
```

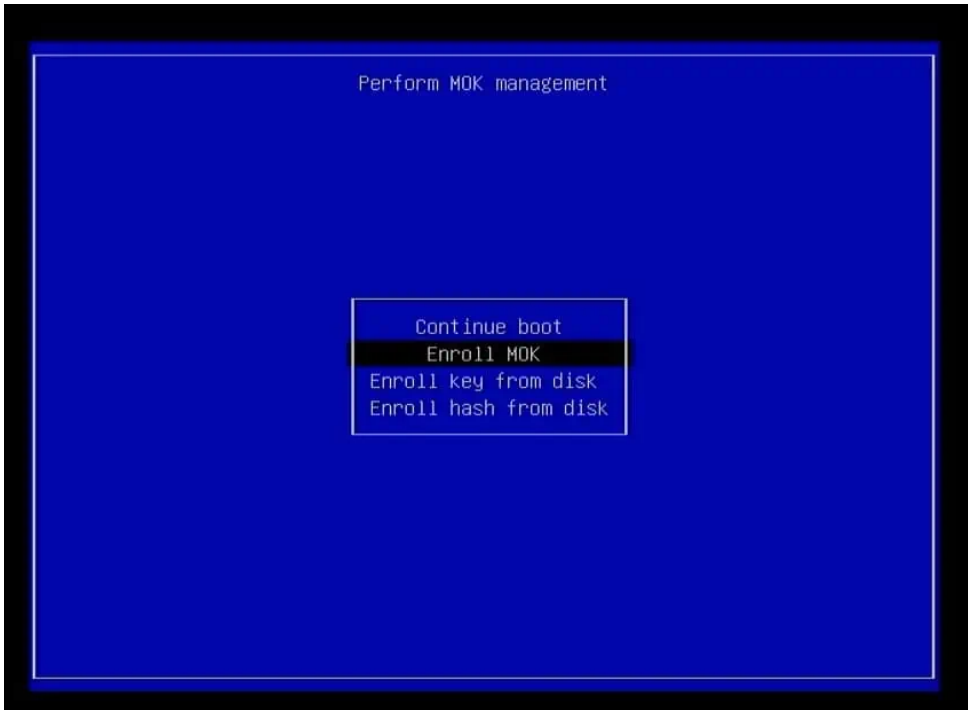
2. Run the following command and modify the PCIe bus number if needed. In this case I'm using **00:02.0**. To verify the file was modified, cat the file and ensure it was modified.

```
echo "devices/pci0000:00/0000:00:02.0/sriov_numvfs = 7" > /etc/sys
```

```
cat /etc/sysfs.conf
```

```
root@proxmox3:~# cat /etc/sysfs.conf
devices/pci0000:00/0000:00:02.0/sriov_numvfs = 7
root@proxmox3:~#
```

3. Reboot the Proxmox host. If using Proxmox 8.1 or later with secure boot you **MUST** setup MOK. As the Proxmox host reboots, monitor the boot process and wait for the **Perform MOK management** window (screenshot below). If you miss the first reboot you will need to re-run the mokutil command and reboot again. The DKMS module will NOT load until you step through this setup.

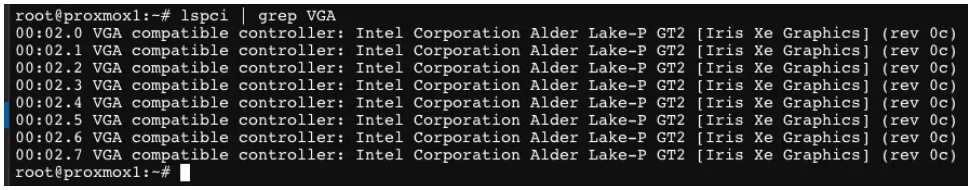


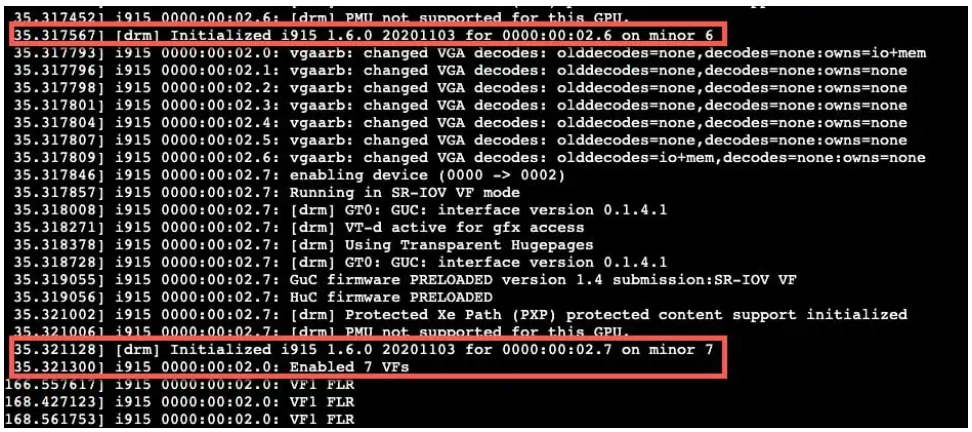
Secure Boot MOK Configuration (Proxmox 8.1+)

4. Select **Enroll MOK, Continue, Yes, *<password>*, Reboot.**

5. Login to the Proxmox host, open a **Shell**, then run the commands below. The first should return eight lines of PCIe devices. The second command should return a lot of log data. If everything was successful, at the end you should see minor **PCIe IDs 1-7** and finally **Enabled 7 VFs**. If you are using secure boot and do NOT see the 7 VFs, then the DKMS module is probably not loaded. Troubleshoot as needed.

```
lspci | grep VGA
dmesg | grep i915
```

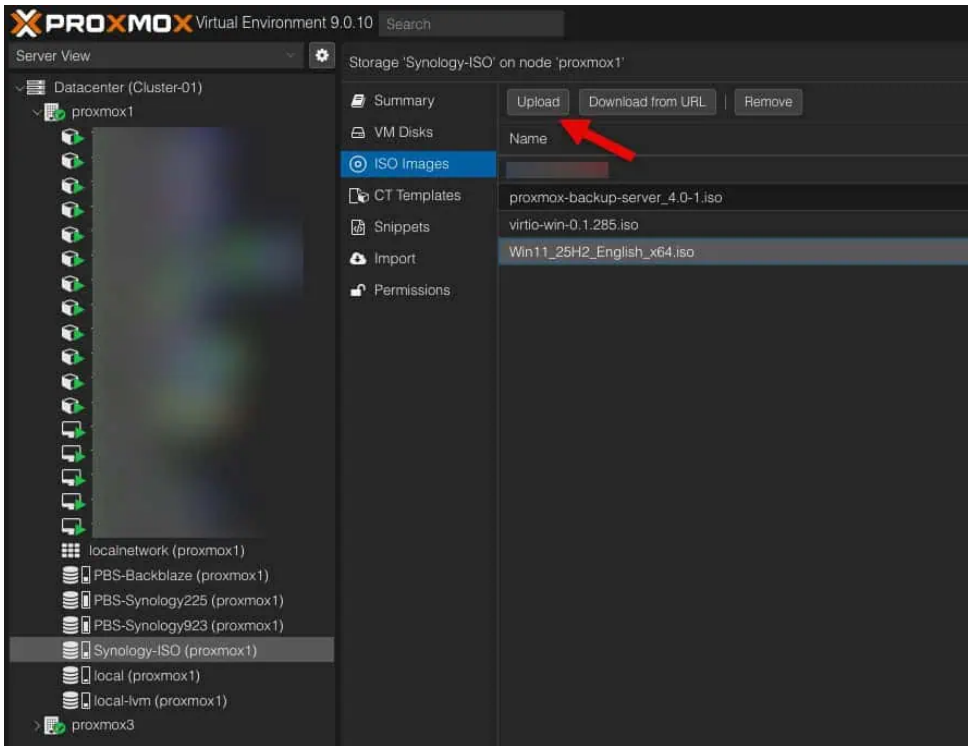




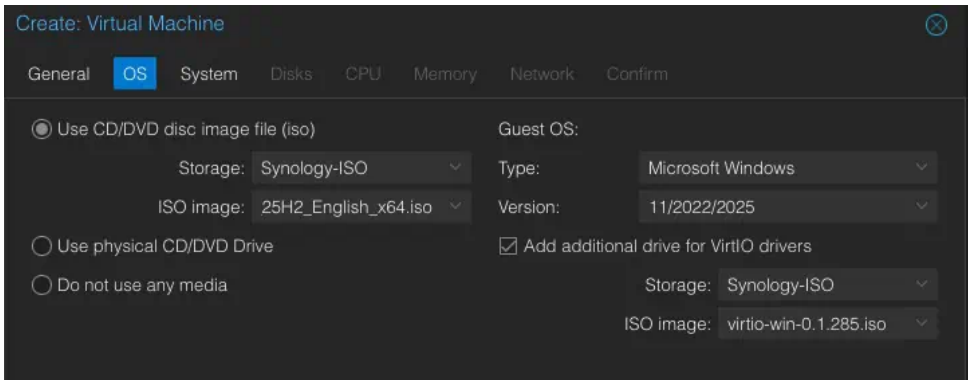
6. Now that the Proxmox host is ready, we can install and configure Windows 11. If you do NOT see 7 VFs enabled, stop. Troubleshoot as needed. Do not pass go, do not collect $100 without 7 VFs. If you are using secure boot and you aren't seeing the 7 VFs, double check the MOK configuration.

Windows 11 Installation

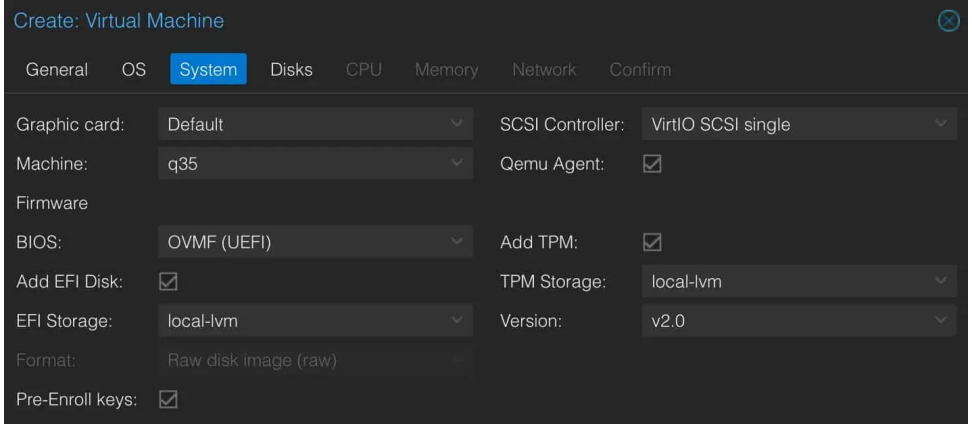
1. Download the latest Fedora Windows VirtIO driver ISO from here.
2. Download the Windows 11 ISO from here. Use the **Download Windows 11 Disk Image (ISO) for x64 devices** option.
3. Upload both the VirtIO and Windows 11 ISOs to the Proxmox server. You can use any Proxmox storage container that you wish. I uploaded them to my Synology. If you don't have any NAS storage mapped, you probably have "**local**", which works.



4. Start the VM creation process. On the **General** tab enter the name of your VM. Click **Next**.
5. On the **OS** tab select the Windows 11 ISO. Change the Guest OS to **Microsoft Windows, 11/2022/2025**. Tick the box for the VirtIO drivers, then select your Windows VirtIO ISO. Click **Next**. **Note:** The VirtIO drivers option is new to Proxmox 8.1.

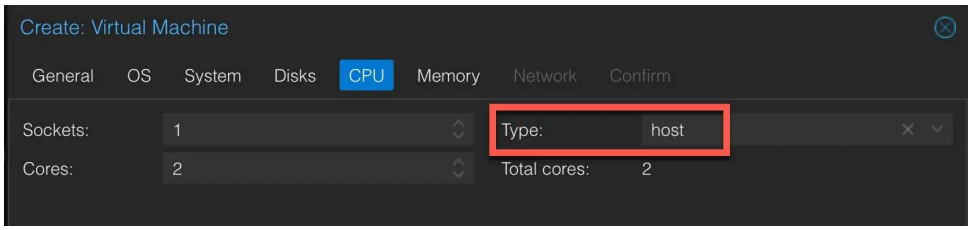


6. On the **System** page modify the settings to match EXACTLY as those shown below. If your local VM storage is named differently (e.g. NOT **local-lvm**, use that instead).

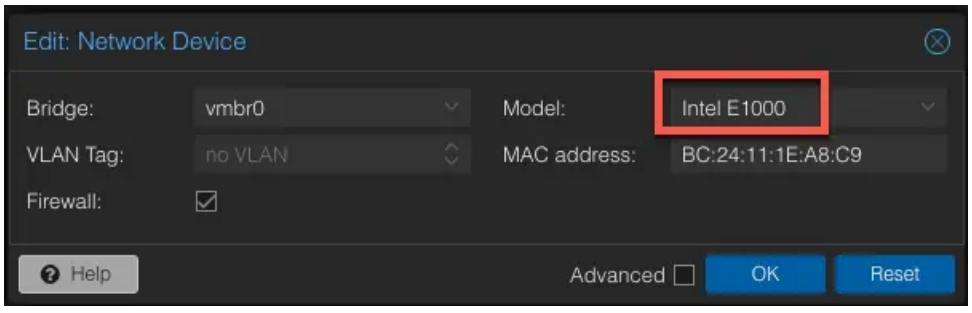


7. On the **Disks** tab, modify the size as needed. I suggest a minimum of 64GB (probably closer to 100GB for future Windows updates). Modify the **Cache** and **Discard** settings as shown. Only enable **Discard** if using SSD/NVMe storage (not a spinning disk).

8. On the **CPU** tab, change the **Type** to **host**. Allocate however many cores you want. I chose 2.



9. On the **Memory** tab allocated as much memory as you want. I suggest 8GB or more.
10. On the **Network** tab change the model to **Intel E1000**. **Note:** We will change this to VirtIO later, after Windows is configured.

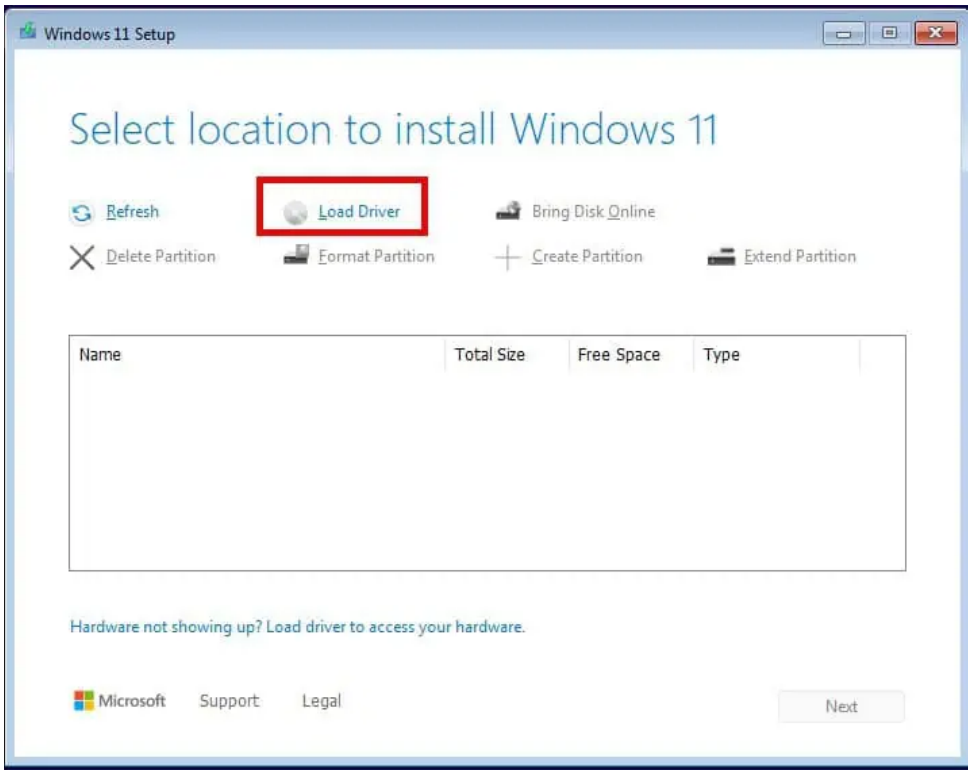


11. Review your VM configuration. Click **Finish**. Do not start the VM.

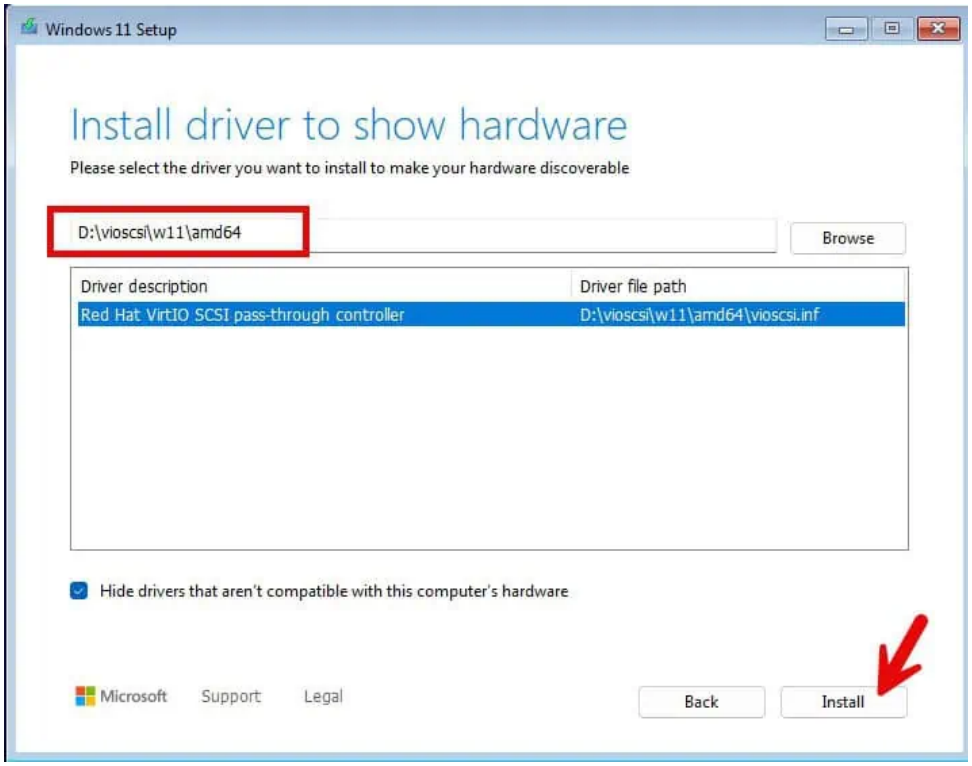
Key ↑	Value
agent	1
bios	ovmf
boot	order=scsi0;ide0;ide2;net0
cores	2
cpu	host
efidisk0	local-lvm:1,efitype=4m,pre-enrolled-keys=1
ide0	Synology:iso/virtio-win-0.1.240.iso,media=cdrom
ide2	Synology:iso/Win11_23H2_English_x64.iso,media=cdrom
machine	q35
memory	8192
name	Win11
net0	e1000,bridge=vmbr0,firewall=1
nodename	proxmox1
numa	0
ostype	win11
scsi0	local-lvm:64,discard=on,iothread=on,cache=writethrough
scsihw	virtio-scsi-single
sockets	1
tpmstate0	local-lvm:1,version=v2.0
vmid	112

Windows 11 Installation

1. In Proxmox click on the Windows 11 VM, then open a console. Start the VM, then press **Enter** to boot from the CD. If you miss the prompt, just “reset” the VM and watch the console closely.
2. Select your language, time, currency format. Click **Next**. Select keyboard. Click **Next**. Click Install Windows 11. Check the box to delete everything. Click **Next**.
3. Click **I don’t have a product key**.
4. Select **Windows 11 Pro**. Click **Next**.
5. Tick the box to accept the license agreement. Click **Next**.
6. Click **Load driver**.



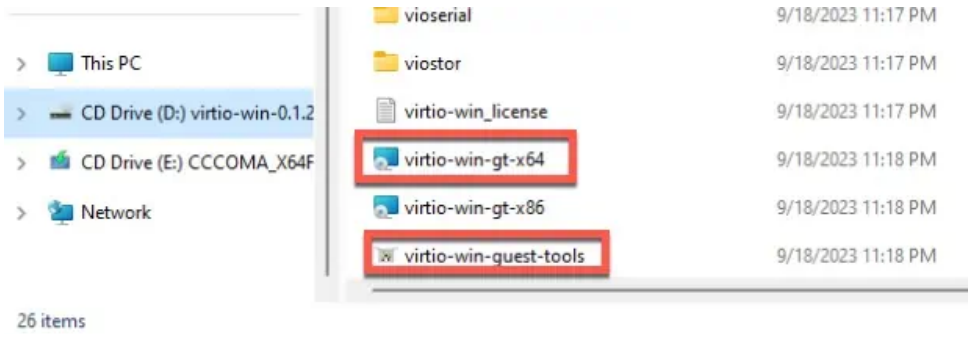
8. Browse to the D: drive, navigate to vioscsi\w11\amd64 and click **OK**.
9. Select the **w11** driver. Click **Next**.



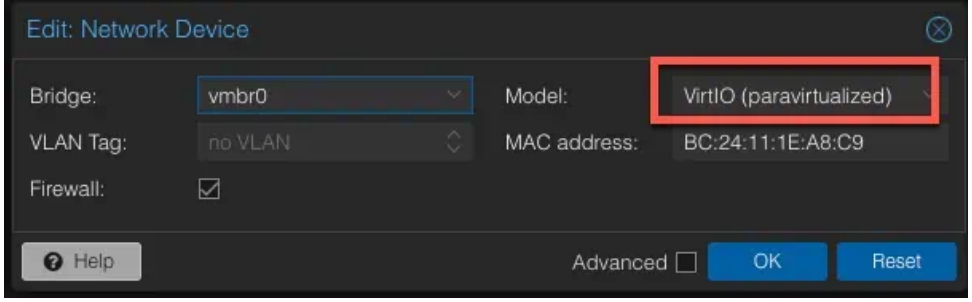
10. On **Select location to install Windows 11** click **Next**.
11. On **Ready to install** click **Next**. Sit back and wait for Windows 11 to install.

Windows 11 Initial Configuration

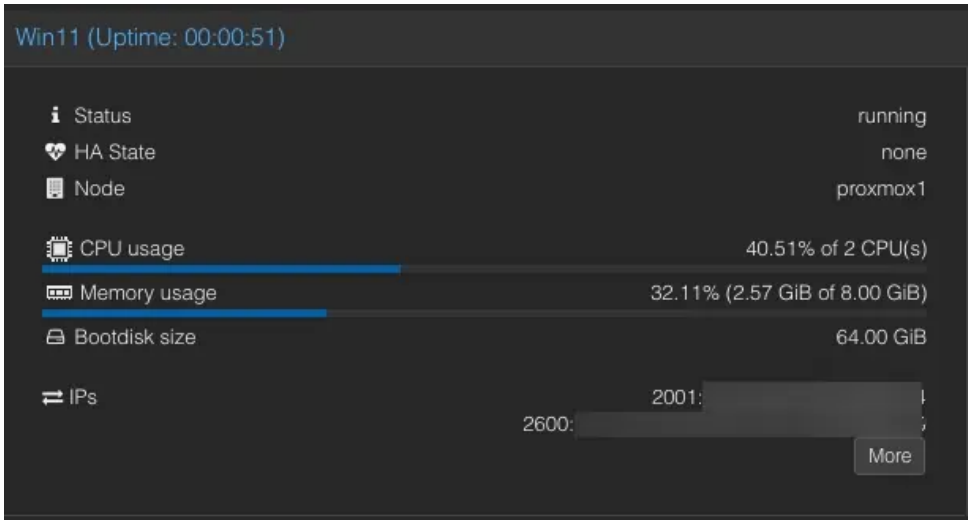
1. Once Windows boots you should see a screen confirming your country or region. Make an appropriate selection and click **Yes**.
2. Confirm the right keyboard layout. Click **Yes**. Add a second keyboard layout if needed.
3. Wait for Windows to check for updates. Windows may reboot.
4. Enter the name of your PC. Click **Next**. Wait for Windows to reboot.
5. Click **Set up for personal use**. Click **Next**. Wait for any Windows updates to install.
6. Click **Sign in**.
7. Choose your privacy settings for your device and click **Next**. Click **Accept.**
8. In File Explorer navigate to the D: drive. Run **virtio-win-gt-x64** and **virtio-win-guest-tools**. Use all default options.
9. Shutdown (NOT reboot) Windows.



13. In Proxmox modify the Windows 11 VM settings and change the NIC to **VirtIO**.



14. Start the Windows 11 VM. Verify at least one IP is showing in the Proxmox console.

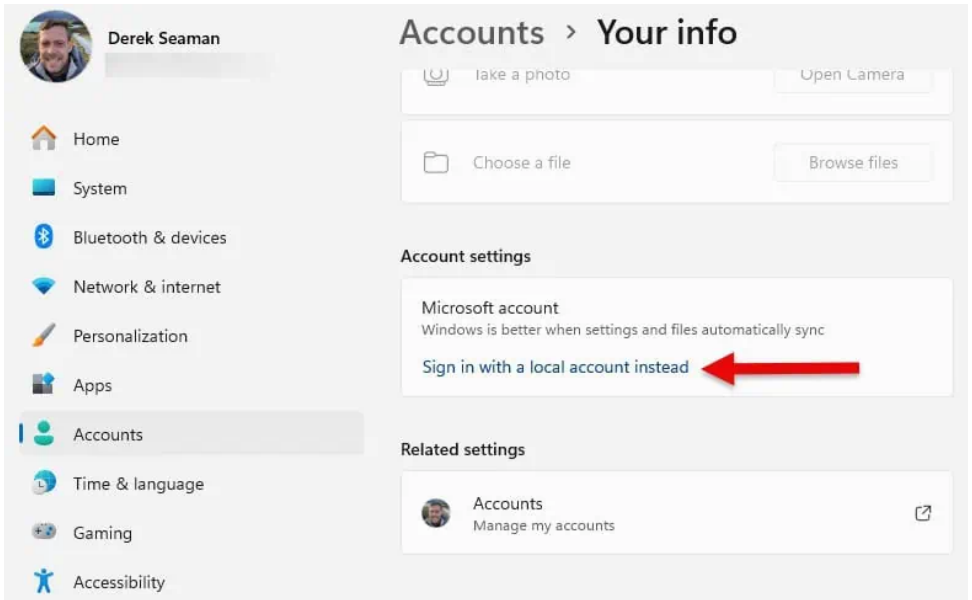


15. You can now unmount the Windows 11 and VirtIO ISOs from the Proxmox console.

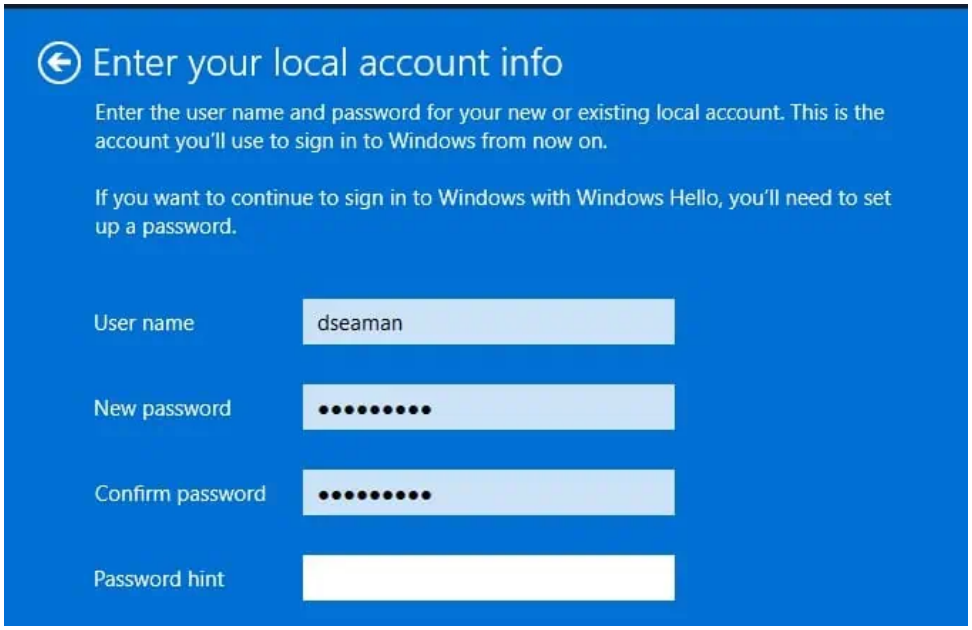
Enabling Remote Desktop

When we enable the vGPU feature, you will be unable to open the Proxmox console to the Windows VM. So we need to enable Remote Desktop (RDP) so you can access Windows over the network. We will also enable signing in with a local account so that RDP doesn't have deal with your Microsoft online account.

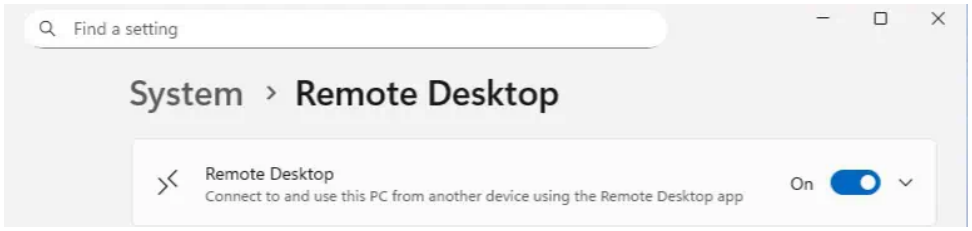
1. In the Windows start search box type "**your info**". Click on **Your account info** under **Best match**.
2. Under Account settings click on **Sign in with a local account instead**.



3. Walk through the wizard to create a local account with password. I would select a unique username to avoid confusion with whatever user name Microsoft used for your online account.



4. After your local account is created, in the Start search box type **remote desktop** then click on **Remote desktop settings** under **Best match**.
5. Enable **Remote Desktop**.

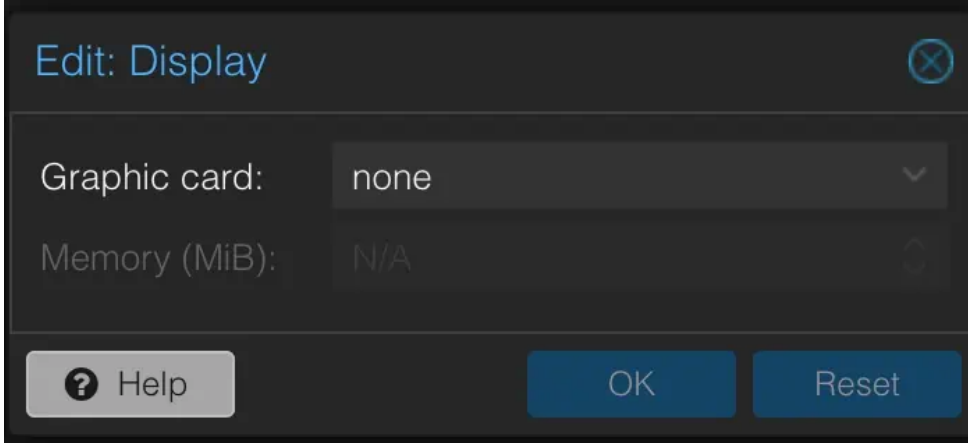


6. You will want to disable local account password expiration, as RDP will fail when your password expires with no easy way to reset. You'd need to re-enable the Proxmox console to reset your password (see later in this post for a how to). Open an **elevated** PowerShell command prompt and enter the following command using the local user name you specified in step 3.

```
Set-LocalUser -Name "username" -PasswordNeverExpires $true
```

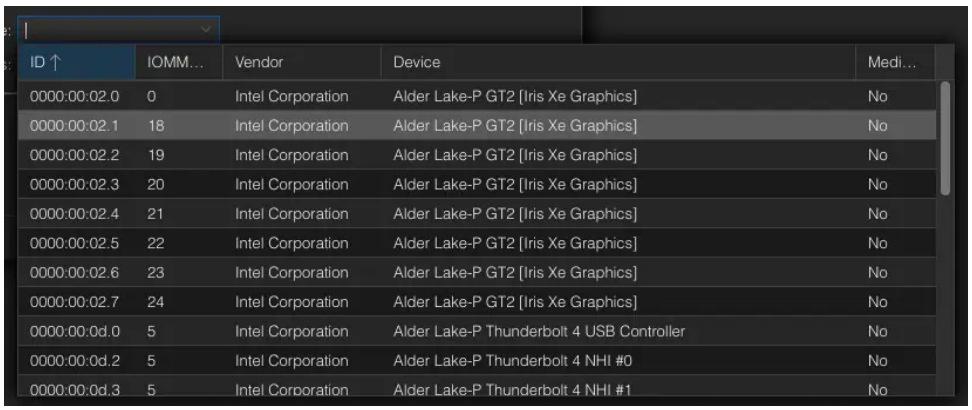
Windows 11 vGPU Configuration

1. Open your favorite RDP client and login using the user name and credentials you setup. You should now see your Windows desktop and the Proxmox console window should show the lock screen.
2. Inside the Windows VM open your favorite browser and download the latest Intel "Recommended" graphics driver from here. In my case I'm grabbing "**Intel® 11th – 14th Gen Processor Graphics – Windows**" version **32.0.101.7076**.
3. Shutdown the Windows VM. Wait for the VM to power off.
4. In the Proxmox console click on the Windows 11 VM in the left pane. Then click on **Hardware**. Click on the **Display** item in the right pane. Click **Edit**, then change it to **none**.



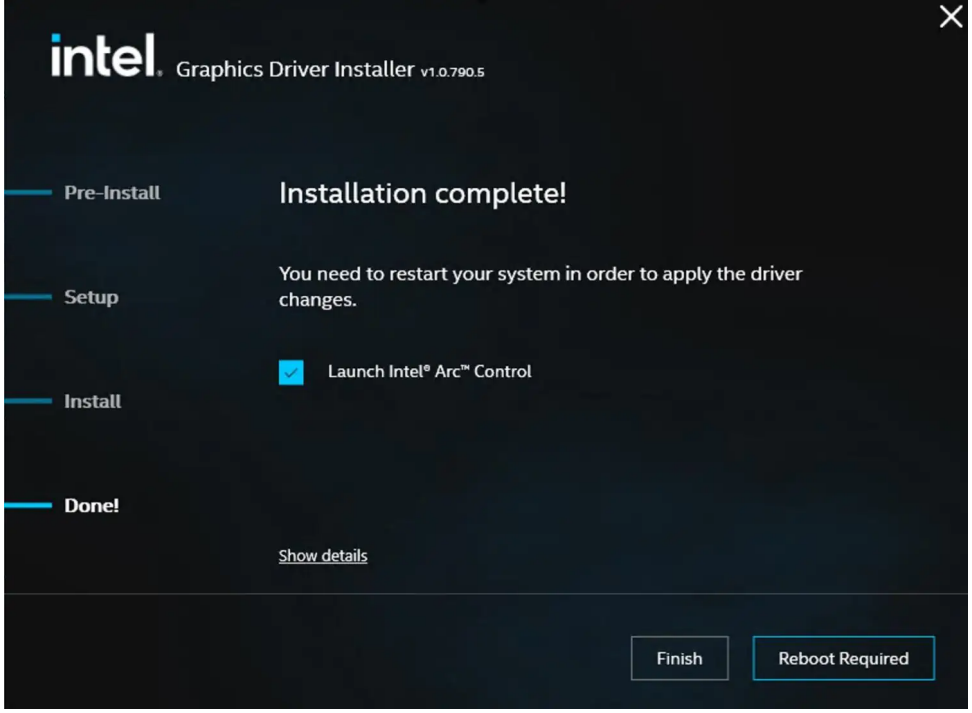
Note: If in the next couple of steps the 7 GPU VFs aren't listed, try rebooting your Proxmox host and see if they come back. Then try adding one to your Windows VM again.

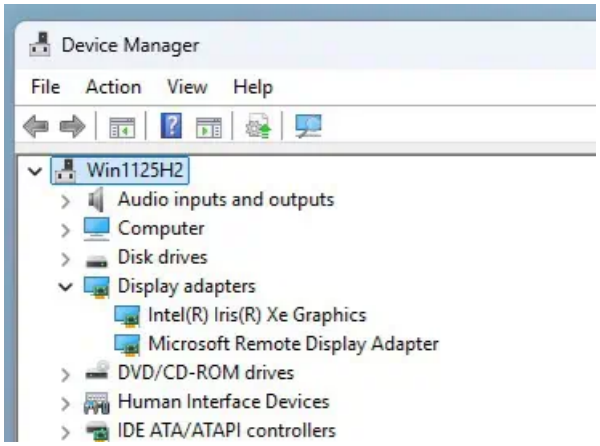
5. In the top of the right pane click on **Add**, then select **PCI Device**.
6. Select **Raw Device**. Then review all of the PCI devices available. Select one of the sub-function (.1, .2, etc..) graphics controllers (i.e. ANY entry except the 00:02.0). Do **NOT** use the root "0" device, for ANYTHING. I chose **02.1**. Click **Add**. **Do NOT** tick the "All Functions" box. Tick the box next to **Primary GPU**. Click **Add**.



ID ↑	IOMM…	Vendor	Device	Medi…
0000:00:02.0	0	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.1	18	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.2	19	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.3	20	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.4	21	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.5	22	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.6	23	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:02.7	24	Intel Corporation	Alder Lake-P GT2 [Iris Xe Graphics]	No
0000:00:0d.0	5	Intel Corporation	Alder Lake-P Thunderbolt 4 USB Controller	No
0000:00:0d.2	5	Intel Corporation	Alder Lake-P Thunderbolt 4 NHI #0	No
0000:00:0d.3	5	Intel Corporation	Alder Lake-P Thunderbolt 4 NHI #1	No

7. Start the Windows 11 VM and wait a couple of minutes for it to boot and RDP to become active. Note, the Proxmox Windows console will NOT connect since we removed the virtual VGA device. You will see a **Failed to connect to server** message. You can now ONLY access Windows via RDP.
8. RDP into the Windows 11 VM. **Note:** If the RDP session opens but the screen is black, try rebooting the VM and see if that helps.
9. Locate the Intel Graphics driver installer and run it. Accept all defaults.
10. If all goes well, you will be presented with an **Installation complete!** screen. Note: For the gfx_win_101.7076 package, the installer never seems to complete, however the Intel drivers do install and appear to work fine. If the Intel installer appears to hang, give it 15-20 minutes, and check task manager for any activity. Once things quiet down, proceed with the next section. You can also open **Device Manager** and validate the **Intel(R) Iris(R) Xe Graphics** device is in a healthy state.
11. Do not yet reboot. Follow the steps in the next section before you reboot.





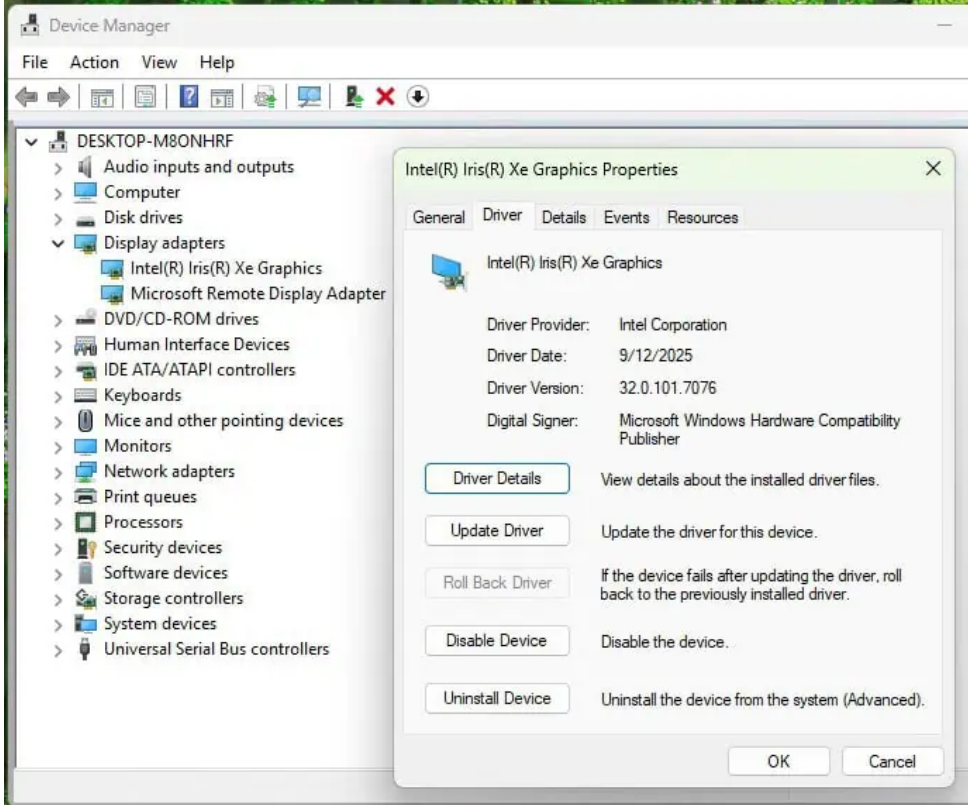
Preventing Automatic Driver Updates

Windows 11 seems to have the nasty habit of thinking the latest Intel GPU drivers need to be "updated" to a version from 2022. **This breaks vGPU functionality.** To prevent Windows update from automatically update drivers follow this procedure (Windows 11 Pro/Enterprise):

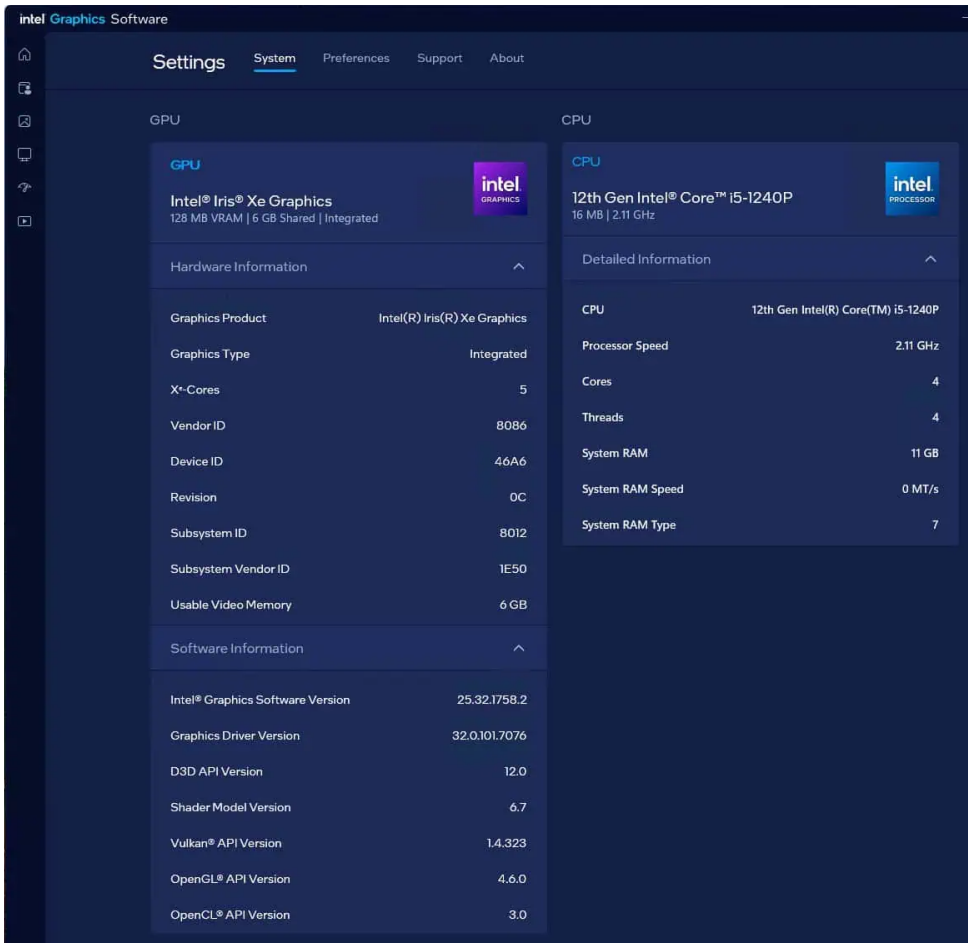
1. Type **Group** in the **Start** search bar and click on **Edit group policy**.
2. Navigate to **Computer Configuration > Administrative Templates > Windows Components > Windows Update > Manage updates offered from Windows Update.**
3. Locate **Do not include drivers with Windows Update** and **enable** the setting.
4. Reboot.

Windows 11 vGPU Validation

1. RDP into Windows and launch **Device Manager**.
2. Expand **Display adapters** and verify there's an Intel adapter in a healthy state (e.g. no error 43). **Note**: If you DO see an error, immediately reboot the VM and check again. If there's still an error, re-install the Intel drivers and reboot once or twice.



3. Launch **Intel Graphics Software**. Click on the **gear icon (Settings)**, **System**, and expand GPU.



4. Launch **Task Manager**, then watch a YouTube video. Verify the GPU is being used.

Troubleshooting Intel Driver Installation

The first time I did this on my N100 Proxmox server the Intel drivers had issues installing. For some reason the RDP session would freeze mid way through the install, or would get disconnected and then fail to connect. I had to reboot the VM from the Proxmox UI and then re-start the Intel installer using their “clean” option. After a couple of re-installs, it ran just fine. It ran flawlessly the first time on my i5-1240P server. If after a VM reboot RDP can’t connect after a few minutes, reboot the VM and try again.

On rare occasions if you reboot the Proxmox host and the Windows 11 VM gets a GPU device error, try rebooting the Proxmox host again and see if it clears. Re-installing the Intel drivers might help too.

Also, if you see the following message in the dmesg logs, this likely means you have secure boot enabled and did not properly configure MOK or enter the MOK password after your first host reboot. If this is the case, re-run the mok utility command, connect a physical monitor/keyboard to your Proxmox host, reboot, and run through the MOK setup.

i915: module verification failed: signature and/or required key missing – tainting kernel

I’ve also observed that sometimes Windows tries to replace your installed Intel ARC driver with a version that is years older, and does NOT work. If you see driver errors, check the version that Windows is trying to use. Make sure it’s the latest version that you manually installed, not one from years ago.

How to Use Intel GPU VFs

You can configure up to 7 VMs to use vGPU resources on the Proxmox host. Each VM MUST be assigned a unique PCIe VF. In addition, no VMs can use the “root” PCIe GPU device. If a running VM is assigned the root GPU, the VFs will not function properly.

Some readers asked if they could connect a HDMI cable to their mini PC and access the Windows 11 desktop. As far as I know this is not possible, as the HDMI output is tied to the primary PCIe GPU device, which we are not using. You will be limited to using RDP to access your desktop(s). You would need to use full GPU PCIe

passthrough for that.

Future Proxmox Kernel Upgrades

As Proxmox gets updated over time with newer Linux Kernel versions, you WILL need to reconfigure DKMS to patch the new kernel. Thankfully this is a pretty simple process. Just follow the section **Proxmox Kernel Configuration**. This will rebuild the new kernel with the latest DKMS module.

There are dependencies between the DKMS module and the Linux kernel. Sometimes the DKMS module breaks with newer kernels, or manual tweaks to the DKMS files are needed. **DO NOT assume** the very latest Proxmox kernel will be successfully patched by DKMS. You can check out the dkms GitHub Issues page and see if there are known issues, or report an issue if you are having problems. If you want to play it safe, after a new kernel comes out I would wait a few days or weeks to see if any issues pop up on the DKMS repo.

There is a neat solution to prevent Proxmox updates from installing a new kernel. You can use the **pin** command, as shown below, to allow Proxmox to update everything EXCEPT your kernel. Of course you eventually do need to update the kernel, but this way you can update it on your schedule and after you've reviewed forums to see if anyone ran into an issue with the latest Proxmox kernel.

To use the pin command, run the command below to pin your current kernel version.

```
proxmox-boot-tool kernel pin $(uname -r)
```

```
root@proxmox1:~# proxmox-boot-tool kernel pin $(uname -r)
Setting '6.8.12-5-pve' as grub default entry and running update-grub.
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.8.12-5-pve
Found initrd image: /boot/initrd.img-6.8.12-5-pve
Found memtest86+ 64bit EFI image: /boot/memtest86+x64.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
root@proxmox1:~#
```

When you are ready to upgrade your kernel, run the unpin command. Update your Proxmox host then re-pin the new kernel.

```
proxmox-boot-tool kernel unpin
```

Unable to connect via RDP

If for some reason you can't connect via RDP to your VM, there is a way to regain a local Proxmox console. This can happen if your password expires, for example. To enable the Proxmox console (and disable vGPU):

1. Shutdown the Windows VM.
2. Remove the PCIe VF device attached to your GPU.
3. Modify the **Display** hardware property and change it to **Default**.
4. Start the VM and wait for the Proxmox console to connect.

Do whatever you need to do to troubleshoot the issue. To re-enable vGPU:

1. Shutdown the VM.
2. Change **Display** to **None**.
3. Re-assign the PCIe GPU VF device.
4. Start the VM.

Kernel Cleanup (Optional)

Over time as you run Proxmox and do routine upgrades, your system can get littered with old kernels. This isn't a problem per say, but does waste storage space. If you want to clean up unused kernels, you can use the awesome Community scripts to remove old versions. No need to reboot after the cleanup. This is entirely optional and is only mentioned for good housekeeping.

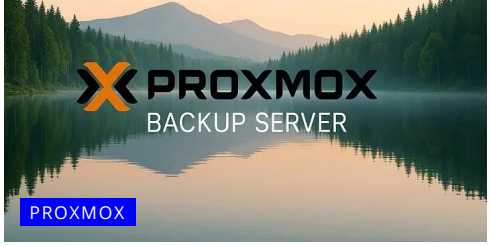
```
bash -c "$(curl -fsSL https://raw.githubusercontent.com/community-
```

Summary

The process for configuring Windows 11 for vGPU using VT-d on Proxmox VE 9.0 is a bit tedious, but it has worked very well for me. By using the virtual functions (VFs), you can share the Intel GPU with up to seven VMs at the same time. Remember that each VM on the Proxmox host that you want to use a vGPU must use a unique VF. Using GPU VFs means you can't use a physically connected monitor to your Proxmox host and access your VM's desktop. You have to use remote desktop to access your Windows 11's desktop.

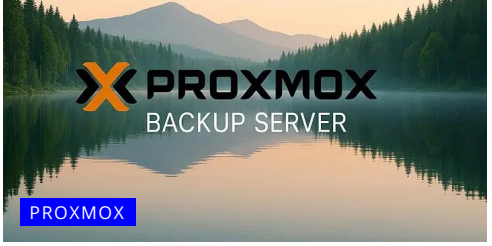
Configuring your Proxmox host for vGPU with VFs may have a negative impact on Linux based VMs or LXCs that need GPU resources, such as Plex. If you are using other VM/LXCs that need GPU resources, be sure to test them thoroughly including HDR tone mapping. Also remember that when you do Proxmox host updates that install a newer kernel, you will have to re-patch the kernel with dkms or your vGPU VFs won't function. Using the kernel pin command you upgrade kernels on your schedule, while allowing other Proxmox updates to install.

Related Posts



Proxmox Backup Server (PBS) 4.0 Blog Series

August 21, 2025

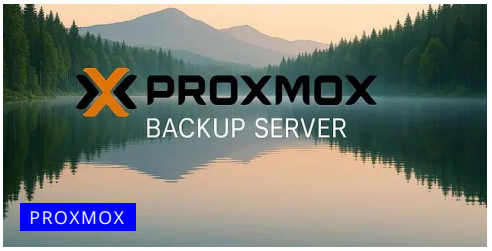


How To: Proxmox Backup Server 4 as a Synology VM

August 17, 2025

How To: Backblaze B2 as a Proxmox Backup Server 4 S3 Datastore

August 17, 2025



How To: Synology iSCSI LUN for Proxmox Backup Server Datastore

August 17, 2025

Subscribe

Join the discussion

1024

175 COMMENTS Oldest

Devedse November 25, 2023 2:30 pm

Thanks for the update, it helped me update my cluster again.

0 Reply

Miles November 25, 2023 4:08 pm

Hello, Nice rework of your previous post (june), that I had followed to get rid of the error 43 in intel GPU driver in my Win11 Pro install. In reading this post, I understand why I couldn't get my connected screen to show something. It's just not possible when using vGPU. Ok. But for my NUC, I didn't manage to get the iGPU passthrough working nor having a display on the connected screen... I own a NUC, Geekom Mini-IT13, with an Intel i9-13900H with an Iris XE for iGPU. Do you have a working guide lie yours to follow for... Read more »

Last edited 1 year ago by Miles

0 Reply

Author

Derek Seaman November 25, 2023 5:04 pm