Complete Installation

Windows 11 VM + Looking Glass on a Fresh CachyOS Host

English edition of the structured and expanded guide.
Updated: April 8, 2026. This document combines machine-specific practice from the tested host with current Looking Glass and CachyOS installation guidance.

Host	CachyOS + RTX 3060 Ti	Guest	Windows 11 + GTX 1660 Ti
Display transport	Looking Glass via KVMFR	Display strategy	Physical dummy plug on the guest GPU
Guide type	Full guide with standard steps and local workarounds separated	Target boot flow	UEFI, libvirt, QEMU, OVMF, swtpm

How to read this guide

This is not just a command dump. The guide is split into standard steps — required or recommended for a typical Looking Glass installation — and machine-specific steps that were necessary on the tested host with an i7-13700K, ASUS PRIME Z790-P, and a GTX 1660 Ti at PCI address group 05:00.x.

Key point. The most important correction relative to the earlier draft is this: Looking Glass assumes that working GPU passthrough through libvirt/QEMU already exists. Looking Glass sits on top of that setup; it is not a replacement for VFIO configuration. This guide therefore stabilizes passthrough first and only then adds KVMFR, SPICE, and the Looking Glass host/client components.

What was added in this revised version

- Full SPICE configuration for input, audio, and clipboard in line with current Looking Glass guidance.
- KVMFR sizing based on target resolution and SDR/HDR mode instead of a hard-coded 128 MB value.
- A clear separation between shared steps and local workarounds such as maxphysaddr=39 and a forced rebinding of the 05:00.x IOMMU group.
- Separate kernel-parameter examples for GRUB, systemd-boot, rEFInd, and Limine on CachyOS.
- A dedicated section about virtio devices, vioinput, and the practical role of a hardware dummy plug.

Table of contents

1. Goal, architecture, and assumptions

The target architecture is simple: CachyOS remains the host OS and keeps the RTX 3060 Ti active for Linux, while Windows 11 runs inside a VM and uses the passed-through GTX 1660 Ti. The Windows image is not shown by switching a physical monitor input; it is captured by Looking Glass and displayed inside a Linux window.

This guide assumes a physical HDMI or DisplayPort dummy plug connected to the GTX 1660 Ti. A dummy plug is not a hard requirement of the Looking Glass project itself, but in practice it is a stable way to keep an active display on the guest GPU without relying on a virtual display driver.

Assumptions

- The host monitor is connected to the RTX 3060 Ti.
- The GTX 1660 Ti is reserved exclusively for the VM.
- VT-d / IOMMU works correctly in BIOS/UEFI.
- The stack is libvirt + QEMU + OVMF + swtpm.
- Looking Glass uses KVMFR instead of a regular shared-memory file.
- Windows 11 must have an active display attached to the guest GPU, ideally through a dummy plug.

What is standard and what is local

Item	Status	Comment
Q35 + OVMF + swtpm	Standard	Typical baseline for a modern Windows 11 passthrough VM.
KVMFR	Recommended	Currently the preferred Looking Glass transport path.
SPICE for input/audio/clipboard	Standard for a full LG setup	Needed for practical keyboard, mouse, audio, and clipboard integration.
Full VFIO binding of 05:00.x	Local but often necessary	Required here because the GPU functions share the same IOMMU group.
maxphysaddr=39	Machine-specific workaround	Only keep this if Looking Glass transport triggers DMA mapping failures on your host.

2. BIOS / UEFI

Before starting, the machine should boot in pure UEFI mode with CSM disabled. Secure Boot should be disabled unless you have already built and signed every custom component you plan to use. CachyOS also documents that the selected boot manager changes where kernel parameters are configured later.

BIOS settings that should be enabled

- VT-d = Enabled
- Above 4G Decoding = Enabled
- Primary Display = PEG or the host GPU
- iGPU = optionally Disabled if it is not needed for anything

Practical note. If you only have two roles — RTX 3060 Ti for Linux and GTX 1660 Ti for the VM — an iGPU is not required. The important part is that the host boots on the Linux GPU, not on the card intended for passthrough.

3. Host packages and libvirt services

On a fresh CachyOS installation you need the virtualization stack itself, OVMF firmware for UEFI guests, and a TPM emulator for Windows 11. The package set below is a reasonable baseline.

```
sudo pacman -Syu
sudo pacman -S qemu-desktop libvirt virt-manager edk2-ovmf swtpm dnsmasq dkms unzip
xorriso
```

After installing the packages, enable libvirtd and start the default NAT network.

```
sudo systemctl enable --now libvirtd
sudo virsh net-autostart default
sudo virsh net-start default
```

If you want to use virt-manager without root, add your user to the appropriate groups and then sign out and back in.

```
sudo usermod -aG libvirt,kvm cachyos
```

4. IOMMU and device identification

Before touching VFIO, identify the exact PCI addresses of the guest GPU and inspect its IOMMU group. On the tested machine, the GTX 1660 Ti exposed four functions at 05:00.x.

PCIe device identification

```
lspci -nn
```

IOMMU group inspection

```
for d in /sys/kernel/iommu_groups/*/devices/*; do
  echo "$d"
done
```

On the tested host, the GTX 1660 Ti presented the following functions:

- 05:00.0 — GPU — 10de:2182
- 05:00.1 — HD Audio — 10de:1aeb
- 05:00.2 — USB controller — 10de:1aec
- 05:00.3 — UCSI — 10de:1aed

Rule of thumb. If all 05:00.x functions sit in one IOMMU group, bind all of them to vfio-pci. In most cases you only pass 05:00.0 and 05:00.1 through to the VM itself, but the whole group must remain viable on the host side.

5. VFIO for the guest GPU

5.1 Modprobe configuration

Create /etc/modprobe.d/vfio-1660ti.conf and list the IDs for the full IOMMU group. The softdep lines below help make sure vfio-pci claims the guest card before the usual NVIDIA or related host drivers do.

```
options vfio-pci ids=10de:2182,10de:1aeb,10de:1aec,10de:1aed

softdep nvidia pre: vfio-pci
softdep nvidia_drm pre: vfio-pci
softdep nvidia_modeset pre: vfio-pci
softdep nvidia_uvm pre: vfio-pci
softdep nouveau pre: vfio-pci
softdep snd_hda_intel pre: vfio-pci
softdep xhci_pci pre: vfio-pci
softdep i2c_nvidia_gpu pre: vfio-pci
```

5.2 Initramfs

In /etc/mkinitcpio.conf, make sure the VFIO modules are loaded early.

```
MODULES=(vfio_pci vfio vfio_iommu_type1)
```

Then rebuild the initramfs.

```
sudo mkinitcpio -P
```

5.3 Kernel parameters — boot-manager-specific variant

The earlier version of the guide assumed GRUB only. On fresh CachyOS installations you may also be using systemd-boot, rEFInd, or Limine, so the location of the kernel parameters depends on the boot manager.

Parameters to add. Kernel parameters used in this guide: intel_iommu=on iommu=pt vfio-pci.ids=10de:2182,10de:1aeb,10de:1aec,10de:1aed

GRUB

```
# /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="... intel_iommu=on iommu=pt vfio-
pci.ids=10de:2182,10de:1aeb,10de:1aec,10de:1aed"

sudo grub-mkconfig -o /boot/grub/grub.cfg
```

systemd-boot

```
# /etc/sdboot-manage.conf
LINUX_OPTIONS="... intel_iommu=on iommu=pt vfio-
pci.ids=10de:2182,10de:1aeb,10de:1aec,10de:1aed"

sudo sdboot-manage gen
```

rEFInd

```
# /boot/refind_linux.conf
"Boot using default options" "root=... rw ... intel_iommu=on iommu=pt vfio-
pci.ids=10de:2182,10de:1aeb,10de:1aec,10de:1aed"
```

Limine

```
# /etc/default/limine
KERNEL_CMDLINE[default]="... intel_iommu=on iommu=pt vfio-
pci.ids=10de:2182,10de:1aeb,10de:1aec,10de:1aed"

sudo limine-mkinitcpio
```

5.4 Forced rebinding of the full IOMMU group — only if needed

On the tested machine, vfio-pci.ids=... did not always keep 05:00.2 on vfio-pci after boot. That is not a universal requirement, but it was necessary here and may help on similar hosts.

/usr/local/sbin/vfio-rebind-1660ti.sh

```
#!/bin/bash
set -euo pipefail

for dev in 0000:05:00.0 0000:05:00.1 0000:05:00.2 0000:05:00.3; do
    if [[ ! -e "/sys/bus/pci/devices/${dev}" ]]; then
        continue
    fi

    if [[ -L "/sys/bus/pci/devices/${dev}/driver" ]] && [[ "$(basename "$(readlink -f
"/sys/bus/pci/devices/${dev}/driver")")" == "vfio-pci" ]]; then
        continue
    fi

    echo vfio-pci > "/sys/bus/pci/devices/${dev}/driver_override"

    if [[ -L "/sys/bus/pci/devices/${dev}/driver" ]]; then
        echo "${dev}" > "/sys/bus/pci/devices/${dev}/driver/unbind"
    fi

    echo "${dev}" > /sys/bus/pci/drivers/vfio-pci/bind
done
```

```
sudo chmod 755 /usr/local/sbin/vfio-rebind-1660ti.sh
```

/etc/systemd/system/vfio-rebind-1660ti.service

```
[Unit]
Description=Rebind GTX 1660 Ti IOMMU group to vfio-pci
After=systemd-modules-load.service
Before=libvirtd.service

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/vfio-rebind-1660ti.sh
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable vfio-rebind-1660ti.service
```

5.5 Host reboot and VFIO verification

```
cat /proc/cmdline
for d in 05:00.0 05:00.1 05:00.2 05:00.3; do
  echo "== $d =="
  lspci -nnk -s "$d"
done
```

Expected state after reboot:

- /proc/cmdline shows intel_iommu=on and iommu=pt.
- All 05:00.x functions are bound to vfio-pci.
- The host RTX 3060 Ti still works normally with the NVIDIA driver.

6. KVMFR for Looking Glass

Current Looking Glass guidance recommends KVMFR instead of a regular shared-memory file because it provides better handling for the host/guest transport path. On this host, plain ivshmem together with the default setup caused QEMU to fail when the transport was added.

6.1 Choosing the memory size

Do not set 128 MB blindly. KVMFR size should match the maximum target resolution and whether you intend to use SDR or HDR.

Resolution	SDR	HDR
1920×1080	32 MB	64 MB
1920×1200	32 MB	64 MB
2560×1440	64 MB	128 MB
3840×2160	128 MB	256 MB

How to read the table. For this specific setup, 128 MB is correct only if you target 4K SDR or 1440p HDR. For 1080p or 1440p SDR it is simply extra headroom, not a requirement.

6.2 Building the kvmfr module through DKMS

The safest route is to download source that matches the Looking Glass host/client build you intend to use, then register the module with DKMS.

```
sudo mkdir -p /usr/local/src/looking-glass-b7-190
cd /usr/local/src/looking-glass-b7-190
sudo curl -fL -o looking-glass-source-B7-190-7f31ecf5.tar.gz
https://looking-glass.io/artifact/B7-190-7f31ecf5/source
sudo tar -xzf looking-glass-source-B7-190-7f31ecf5.tar.gz
sudo rm -rf /usr/src/kvmfr-0.0.12
sudo cp -r /usr/local/src/looking-glass-b7-190/looking-glass-B7-190-7f31ecf5/module
/usr/src/kvmfr-0.0.12
sudo dkms remove -m kvmfr -v 0.0.12 --all || true
sudo dkms add -m kvmfr -v 0.0.12
sudo dkms install -m kvmfr -v 0.0.12
```

6.3 Module configuration and permissions

```
# /etc/modprobe.d/kvmfr.conf
options kvmfr static_size_mb=128
```

```
# /etc/modules-load.d/kvmfr.conf
kvmfr
```

```
# /etc/udev/rules.d/99-kvmfr.rules
SUBSYSTEM=="kvmfr", GROUP="kvm", MODE="0660"
```

```
sudo modprobe kvmfr
sudo udevadm control --reload
sudo udevadm trigger
ls -l /dev/kvmfr0
```

The expected result is a character device at /dev/kvmfr0. If, after a QEMU failure, you see a regular file at that path instead of a character device, remove it and reload the module.

6.4 libvirt / QEMU access to /dev/kvmfr0

```
# /etc/libvirt/qemu.conf
cgroup_device_acl = [
    "/dev/null", "/dev/full", "/dev/zero",
    "/dev/random", "/dev/urandom",
    "/dev/ptmx", "/dev/userfaultfd",
    "/dev/kvmfr0"
]
```

AppArmor. The official documentation also describes adding /dev/kvmfr0 to AppArmor rules. On a typical CachyOS install this may not be required, but if you use a stricter profile set, add the device there as well.

7. Windows 11 VM configuration

You can create the VM in virt-manager, but the final result is still best checked and refined in the domain XML. The most important thing is to think of the VM as a full passthrough machine first and a Looking Glass guest second.

7.1 Base parameters

- Chipset: Q35
- Firmware: OVMF
- TPM: swtpm
- CPU: host-passthrough
- RAM: for example 16 GB
- vCPU: for example 8
- Disk: qcow2 or raw
- GPU passthrough: at minimum 05:00.0 and 05:00.1

If you use virtio devices for the disk or network, also attach the virtio-win ISO during Windows installation. You will need it to load the storage, network, and input drivers.

7.2 SPICE, input, audio, and clipboard — required for a complete LG setup

This was the biggest gap in the previous document. Looking Glass uses SPICE for keyboard, mouse, audio, and fallback display. For a full and comfortable setup, do not skip this block.

```
<graphics type='spice' autoport='yes' listen='127.0.0.1'>
  <listen type='address' address='127.0.0.1'/>
</graphics>

<video>
  <model type='vga' vram='16384' heads='1' primary='yes'/>
</video>

<input type='mouse' bus='virtio'/>
<input type='keyboard' bus='virtio'/>

<sound model='ich9'/>
```

```
<audio id='1' type='spice'/>

<channel type='spicevmc'>
  <target type='virtio' name='com.redhat.spice.0'/>
</channel>
```

What not to do. Do not leave a final <input type='tablet'/> in place just because virt-manager added it by default. Looking Glass recommends virtio mouse and keyboard with the Windows vioinput driver instead. Also prefer <model type='vga'/> over QXL for the fallback display path.

You can keep a simple virtual display temporarily during installation and early troubleshooting, but the actual image used by Looking Glass should come from the active display attached to the passed-through GPU.

7.3 CPU and memory

```
<cpu mode='host-passthrough' check='none' migratable='on'>
  <topology sockets='1' dies='1' clusters='1' cores='8' threads='1'/>
</cpu>

<memballoon model='none'/>
```

Disabling memballoon matters. Looking Glass documentation describes it as a source of serious performance problems in VFIO setups.

CPU tuning. Do not give every core to the VM. Leave at least two cores or four threads for the host, and later consider CPU pinning and IRQ tuning only after the base setup is stable.

7.4 Hiding KVM from NVIDIA

This remains a practical and safe part of the configuration, especially if you want to minimize the chance of NVIDIA driver issues inside the guest.

```
<features>
  <acpi/>
  <apic/>
  <ioapic driver='kvm'/>
  <kvm>
    <hidden state='on'/>
  </kvm>
  <hyperv mode='custom'>
    <relaxed state='on'/>
    <vapic state='on'/>
    <spinlocks state='on' retries='8191'/>
    <vpindex state='on'/>
    <runtime state='on'/>
    <synic state='on'/>
    <stimer state='on'/>
    <reset state='on'/>
    <vendor_id state='on' value='1A2B3C4D5E6F'/>
  </hyperv>
  <vmport state='off'/>
  <smm state='on'/>
</features>
```

7.5 KVMFR / Looking Glass in XML

With current QEMU/libvirt releases, use the JSON-style syntax. Remember that the value in size is given in bytes.

```
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
  ...
  <qemu:commandline>
    <qemu:arg value="-device"/>
    <qemu:arg value='{"driver":"ivshmem-plain","id":"shmem0","memdev":"lg"}'/>
    <qemu:arg value="-object"/>
    <qemu:arg value='{"qom-type":"memory-backend-file","id":"lg","mem-path":"/dev/
kvmfr0","size":134217728,"share":true}'/>
  </qemu:commandline>
</domain>
```

7.6 maxphysaddr=39 — workaround for this host only

On this machine, ordinary GPU passthrough worked, but after adding the Looking Glass transport QEMU failed with vfio: DMA mapping failed. Limiting maxphysaddr to 39 solved that on this host.

```
<cpu mode='host-passthrough' check='none' migratable='on'>
  <maxphysaddr mode='passthrough' limit='39'/>
</cpu>
```

Important. Do not treat this as a standard Looking Glass installation step. Keep it in the XML only if you actually hit the DMA mapping failure without it.

7.7 hostdev entries for the GPU

Add at least the GPU and audio functions of the GTX 1660 Ti to the VM, which means 05:00.0 and 05:00.1. The remaining group functions — 05:00.2 and 05:00.3 — do not normally need to be attached to the guest, but on this host they still had to stay bound to vfio-pci.

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x05' slot='0x00' function='0x0'/>
  </source>
</hostdev>

<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x05' slot='0x00' function='0x1'/>
  </source>
</hostdev>
```

8. Windows installation and drivers

8.1 Installation media

- Windows 11 ISO
- virtio-win ISO if you use virtio for disk, network, or input
- Optionally, a small tools ISO with Looking Glass Host and extra utilities

In practice, a separate tools ISO is very convenient. It can contain looking-glass-host-setup.exe, VDD Control.exe, and any fallback utilities you want to keep nearby.

8.2 Guest OS and driver installation

- Install Windows 11.
- If the disk or network uses virtio, load the appropriate driver from the virtio-win ISO during installation.
- After the first boot, install the remaining virtio drivers, including vioinput.
- Install the NVIDIA driver for the GTX 1660 Ti.

- In Device Manager, confirm that the GPU does not show Code 43.

8.3 Dummy plug and active display

On the Windows side, the most important practical condition is that the GTX 1660 Ti must have an active display. In this setup, the simplest solution is the physical dummy plug connected to the guest GPU.

> **If the image is still black.** If Windows sees the GPU but Looking Glass still shows a black screen, first check whether the display attached to the GTX 1660 Ti is active in Windows display settings and whether Looking Glass Host is running.

9. Looking Glass host and client

9.1 Looking Glass Host in Windows

In current B7 documentation, the Windows host installer includes the IVSHMEM driver, so you do not have to hunt it down separately the way many older tutorials still describe.

```
# In Windows, as Administrator:
looking-glass-host-setup.exe
```

- Run the installer as Administrator.
- Keep the default options unless you intentionally need different ones.
- Reboot the Windows VM after installation.

9.2 Looking Glass Client on CachyOS

The upstream documentation describes building the client from source. On CachyOS or Arch, exact package names depend on the repositories you use, so the cleanest final check is simply to confirm that the client binary starts.

```
# Launch the client after installation
looking-glass-client
```

> **Version compatibility.** If you use a bleeding-edge client, Looking Glass expects a matching host build. For stable releases, keep the host and client within the same branch/build family.

10. Final verification

10.1 Host

```
for d in 05:00.0 05:00.1 05:00.2 05:00.3; do
  echo "== $d =="
  lspci -nnk -s "$d"
done

ls -l /dev/kvmfr0
virsh -c qemu:///system domstate win11-igpu
```

10.2 Guest

- The GTX 1660 Ti works without Code 43.
- The dummy plug provides an active display.

- Looking Glass Host is running.
- After launching the client on Linux, the Windows desktop is visible.

11. Common problems

Problem	Likely cause or fix
VFIO PCI device assignment is not supported by the host	The host did not boot with intel_iommu=on and iommu=pt, or IOMMU is not enabled correctly in firmware.
group is not viable	Not every function in the same IOMMU group is bound to vfio-pci.
/dev/kvmfr0 is a regular file	The VM started before the kvmfr module loaded; remove the file and reload the module.
vfio: DMA mapping failed, unable to continue	On this host, KVMFR plus maxphysaddr limit='39' solved it.
Looking Glass works but there is no audio or input	Check the XML for SPICE graphics, VGA fallback video, virtio mouse/keyboard, and SPICE audio.
Clipboard does not work	SPICE guest tools are missing in Windows or the spicevmc channel is not present in XML.
Black screen even though the VM runs	No active display is attached to the GTX 1660 Ti, the Windows display is assigned incorrectly, or the Looking Glass Host service is not running.

12. Fast checklist

Step	Done	Notes
VT-d and Above 4G Decoding enabled in BIOS	☐	
Host boots on the RTX 3060 Ti	☐	
05:00.x functions identified correctly	☐	
vfio-pci binds the full 05:00.x group	☐	
The host still has working graphics after reboot	☐	
kvmfr loads and /dev/kvmfr0 exists as a character device	☐	
QEMU has permission to access /dev/kvmfr0	☐	
Windows VM starts with Q35 + OVMF + swtpm	☐	
SPICE block is present for input, audio, and clipboard	☐	
GPU and audio are attached as hostdev devices	☐	
Windows has NVIDIA + virtio + vioinput drivers installed	☐	

Dummy plug creates an active display on the guest GPU	☐	
Looking Glass Host is installed and running	☐	
looking-glass-client shows the Windows desktop	☐	

13. Final conclusion for this host

On this specific machine, a working and comfortable setup does not end at “GPU passthrough itself”. A stable result required a clean passthrough baseline, the full SPICE integration layer, KVMFR for Looking Glass, and one local workaround for the DMA mapping failure.

In practice. The three things that mattered most in practice were: 1) the full 05:00.x group bound to vfio-pci, 2) a complete SPICE + KVMFR block for Looking Glass, and 3) maxphysaddr=39 only if the transport fails without it.

14. Sources

- Looking Glass B7 — libvirt/QEMU Installation — https://looking-glass.io/docs/B7/install_libvirt/
- Looking Glass B7 — IVSHMEM with the KVMFR module — https://looking-glass.io/docs/B7/ivshmem_kvmfr/
- Looking Glass B7 — Host Application Installation — https://looking-glass.io/docs/B7/install_host/
- Looking Glass B7 — Client Application Installation — https://looking-glass.io/docs/B7/install_client/
- Looking Glass B7 — Building — https://looking-glass.io/docs/B7/build/
- CachyOS Wiki — Boot Manager Configuration — https://wiki.cachyos.org/configuration/boot_manager_configuration/
- CachyOS Wiki — Offered Boot Managers — https://wiki.cachyos.org/installation/boot_managers/
- CachyOS Wiki — Desktop/Laptop Installation — https://wiki.cachyos.org/installation/installation_on_root/