Arch Linux

Información útil

1. Compartir audio

Al iniciar Hyprland se crea un *micrófono virtual*, que podemos enlazar con el audio de cualquier dispositivo o aplicación. Podemos así, compartir el audio de las aplicaciones junto con nuestra voz. Útil para aplicaciones que no tienen implementada esta funcionalidad.

Para esto tendremos que elegir *my-virtualmic* como nuestro micrófono predeterminado. Y añadir el audio de nuestro micrófono real y nuestras aplicaiones mediante el script `pipewire-virtualmic-select (Alt+F11)`.

2. Cronie y scripts útiles

El repositorio incluye una variedad de scripts para automatizar tareas. Por defecto no se usan, pero podemos hacer que se ejecuten de forma automática con crond.

2.1. convert-2m4a y convert-2mp3

Estos scripts convierten toda la música del directorio que damos como primer argumento a formato *.m4a* o *.mp3* en un mirror que replica al estructura del directorio original.

Si ejecutamos:

```
convert-2mp3 /musica/biblioteca /musica/mp3
```

Toda la música de `/musica/biblioteca` se convertira en mp3 en la carpeta `/musica/mp3`

- El script puede usarse con el flag *-p* para convertir varios archivos de forma paralela, lo que reduce el tiempo necesario para la conversión de archivos, pero consume más recursos y probablemente ocupe todo el tiempo de CPU hasta que el script se termine de ejecutar.
- El script también puede usarse con el flag *-l* lo que hará que además de convertir las canciones a otro formato, se busque la letra de la canción y se incluya dentro del archivo de audio.

2.2. corruption-check

Este script comprueba que no haya archivos corruptos en nuestra biblioteca de música, corrige falsos positivos de corrupción y nos escribe una lista con los archivos que no se pueden reproducir correctamente en `/tmp/corruption.log`. Necesita como argumento el directorio cuyos archivos de audio queremos analizar.

2.3. exif-remove

Este script necesita borra toda la información EXIF de las imágenes que contiene el directorio que se le da como argumento.

2.4. wake

Este script comprueba si hay alguna máquina virtual en ejecución, y si no encuentra ninguna, suspende nuestro equipo y lo reanuda a las 7 de la mañana del día siguiente. Útil para ahorrar energía y no tener que preocuparte por suspender tu equipo, ni de encenderlo por las mañanas.

El script te avisa de que el sistema se va suspender y pasados 10 minutos desde dicho aviso, suspende el sistema. *Si el script se ejecuta pasándole el argumento "now", entonces el sistema se supenderá inmediatamente.*

2.5. wakeme

Este script funciona como un despertador, hace sonar el archivo de audio especificado hasta que le damos al un botón que apaga nuestra alarma.

2.6. compressed-backup

Este script crea un fichero comprimido `tar.gz` con una copia de seguridad del directorio que se le da por primer argumento en el directorio que se le da por segundo argumento. Además se encarga de borrar las copias de seguridad que tienen mas de un mes automáticamente.

3. VFIO GPU passthrough

Con VFIO (Virtual Function I/O) GPU passthrough podemos pasarle una tarjeta gráfica física a una máquina virtual. Lo que nos permite tener gráficos acelerados dentro de dicha máquina virtual.

3.1. Pasos Iniciales

- Primero activamos VT-d o AMD-v *dependiendo de si tenemos un procesador Intel o AMD*.
- Debemos de tener IOMMU activado. En la mayoría de placas bases activar VT-d o AMD-v, también activa IOMMU.
- Debemos tener desactivado CSM (Compatibility Support Module) en los ajustes de arranque de nuestra placa base.

3.2. Pre-configurar el gestor de arranque

El gestor de arranque GNU GRUB, es el programa que se encarga de cargar el kernel de nuestro sistema operativo.

Para usar VFIO necesitamos configurar el arranque del kernel. Las opciones de configuración globales de GRUB están en `/etc/default/grub`

Este archivo contiene las opciones de arranque del kernel en: *GRUB_CMDLINE_LINUX*.

```
GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet"
GRUB_CMDLINE_LINUX=""
```

Tendremos que añadir las siguientes opciones:

- `iommu=pt`
- `amd_iommu=on` o `intel_iommu=on`, dependiendo de si tenemos una cpu Intel o AMD.
- `video=efifb:off`

```
GRUB_CMDLINE_LINUX="intel_iommu=on iommu=pt video=efifb:off"
GRUB_CMDLINE_LINUX="amd_iommu=on iommu=pt video=efifb:off"
```

Una vez editado el archivo, debemos actualizar nuestra configuración de GRUB con:

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

3.3. Identificar el ID de nuestra gráfica y los grupos IOMMU

Después de esto ya podemos asignar el driver VFIO a nuestra tarjeta gráfica para poder usarla en nuestra máquina virtual.

Necesitamos conocer el identificador de nuestra gráfica, podemos obtener esta informaciónejecutando en BASH:

```
shopt -s nullglob
for g in /sys/kernel/iommu_groups/*; do
    echo "IOMMU Group ${g##*/}:"
    for d in $g/devices/*; do
        echo -e "\t$(lspci -nns ${d##*/})"
    done;
done;
```

```
IOMMU Group 15:
  08:00.0 VGA compatible controller [0300]: NVIDIA TU116 [GeForce GTX 1660] [10de:21c4] (rev a1)
  08:00.1 Audio device [0403]: NVIDIA TU116 High Definition Audio Controller [10de:1aeb] (rev a1)
  08:00.2 USB controller [0c03]: NVIDIA Device [10de:1aec] (rev a1)
  08:00.3 Serial bus controller [0c80]: NVIDIA TU116 [GeForce GTX 1650] [10de:1aed] (rev a1)
```

Debemos añadir cada uno de los IDs de los dispositivos que se encuentran en el mismo grupo que nuestra gráfica (en nuestro ejemplo: `10de:21c4`, `10de:1aeb`, `10de:1aec` y `10de:1aed`) a nuestro archivo de configuración de GRUB, diciéndole que le asigne a estos dispositivos el driver vfio.

```
GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet"
GRUB_CMDLINE_LINUX="amd_iommu=on iommu=pt video=efifb:off
vfio-pci.ids=10de:21c4,10de:1aeb,10de:1aec,10de:1aed"
```

Ejecutamos de nuevo:

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

3.4. Módulos del kernel

Ahora cada vez que iniciemos nuestro ordenador el kernel intentará asignarle a esos dispositivos PCI el driver VFIO, pero el driver VFIO por defecto no se carga durante el arranque del sistema.

Debemos de configurar nuestro kernel para que nuestra imagen de arranque incluya el driver VFIO, para poder asignarselo a nuestra gráfica.

Para esto debemos editar `/etc/mkinitcpio.conf` y añadir los módulos del kernel que vamos a usar.

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

Una vez editado nuestro archivo debemos regenerar el initramfs con:

```
# mkinitcpio -P
```

Para comprobar que hemos seguido los pasos correctamente reiniciamos nuestro ordenador y cuando arranque de nuevo, ejecutamos:

```
# dmesg | grep -i vfio
```

Si tenemos una salida parecida a:

```
[3.416692] vfio_pci: add [10de:21c4[ffffffff:ffffffff]] class 0x000000/00000000
[3.433353] vfio_pci: add [10de:1aeb[ffffffff:ffffffff]] class 0x000000/00000000
[3.450019] vfio_pci: add [10de:1aec[ffffffff:ffffffff]] class 0x000000/00000000
[3.466953] vfio_pci: add [10de:1aed[ffffffff:ffffffff]] class 0x000000/00000000
```

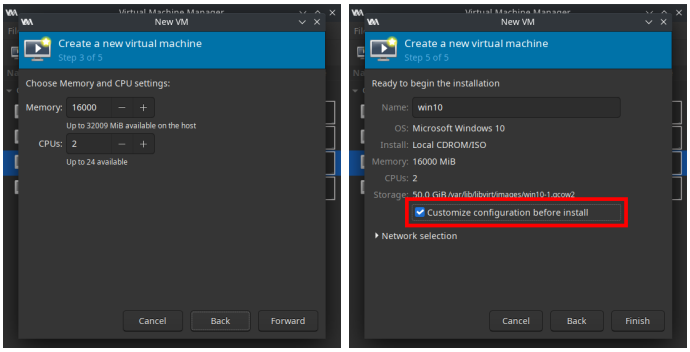
donde podemos ver que el driver `vfio_pci` se ha cargado correctamente para nuestros dispositivos, entonces hemos realizado correctamente todos los pasos.

3.5. Instalación del sistema operativo

Ya tenemos lista nuestra gráfica para ser usada por nuestra máquina virtual, queda instalar nuestra máquina virtual y configurarla para usarla cómodamente.

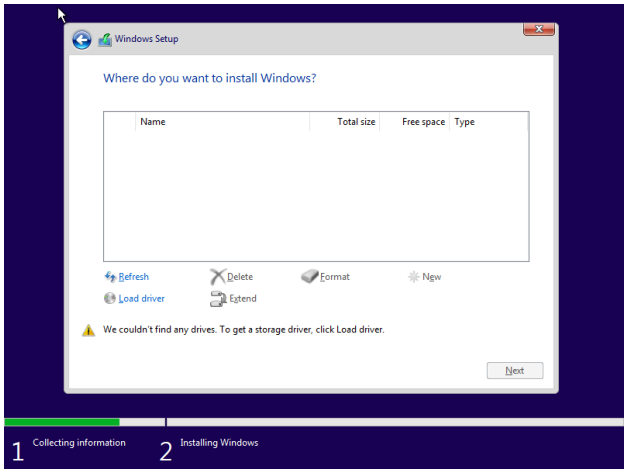
Descargamos la ISO de Windows 10 desde https://www.microsoft.com, y los drivers que necesitará Windows desde https://fedorapeople.org. Una vez descargadas las ISO, abrimos *virt-manager* y creamos una máquina virtual:

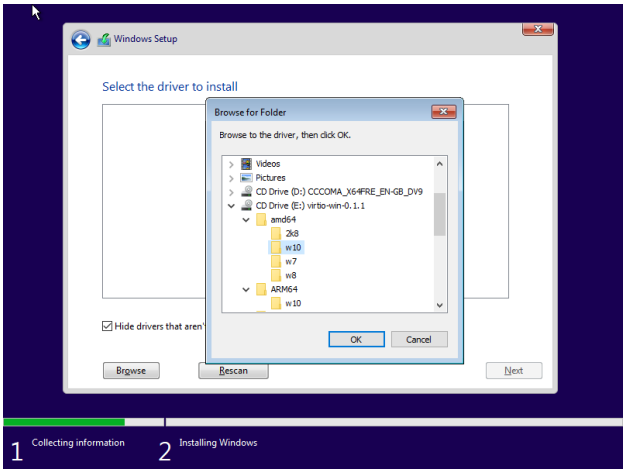
Asignamos la memoria RAM *(8GB como mínimo)* a nuestra máquina virtual. Y en el último paso, elegimos *customizar nuestra máquina virtual* antes de la instalación.



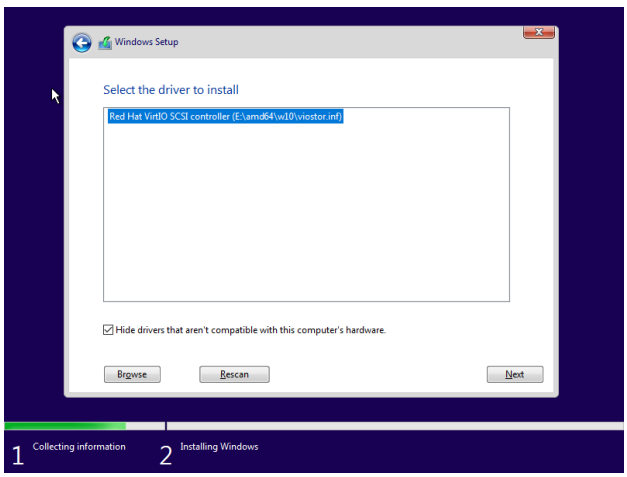
Ahora deberás hacer las siguientes modificaciones:

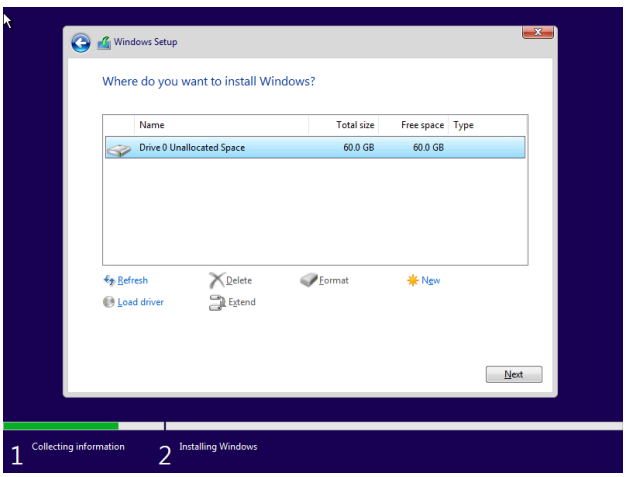
- Cambiar el chipset a *Q35* y elegir el firmware `UEFI:x86_64:/usr/share/edk2/x64/OVMF_CODE.fd`
- Ajustar la *topología* del procesador de la siguiente manera:
    - 1 *socket*, tantos *centros* comos núcleos tenga tu procesador y tantos *hilos* como hilos tenga tu procesador por núcleo (ej. para un procesador de 8 núcleos, 16 hilos, elegiríamos: *1 Socket, 8 Centros, 2 Hilos*).
- Cambiar el bus de nuestro disco virtual de *SATA* a *VirtIO*.
- Cambiar el modelo del *NIC* a *virtio*
- Añadir el ISO con los drivers virtio.





Durante la instalación debemos cargar nuestro driver *virtio* para que nuestro disco virtual sea detectado. *Cuando la instaliación de windows finalize, apaga la máquina virtual.*





3.6. Looking Glass en el Host

Para usar looking glass sin tener dos monitores, necesitamos un display dummy.

Looking Glass nos muestra una el output de nuestra gráfica con una latencia muy baja. Logra esto mandando la información a través de un archivo compartido entre máquina virtual y host.

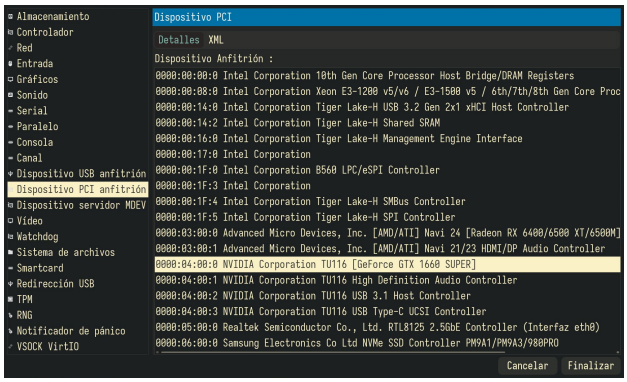
Necesitamos configurar *tmpfilesd* para que cada vez que iniciamos el sistema, se cree el archivo que compartido que Looking Glass necesita. Para eso ejecutamos los siguientes comandos:

```
echo "f /dev/shm/looking-glass 0660 $USER kvm -" | \
sudo tee -a /etc/tmpfiles.d/looking-glass.conf
```

3.7. Añadir Gráfica y Looking Glass en el Guest

Ya tenemos casi todo listo, nos queda instalar los drivers de vídeo y Looking Glass en nuestra máquina virtual.

Tenemos que añadir nuestra tarjeta gráfica a la máquina virtual. En *"Añadir Dispositivo"*, buscamos la pestaña para añadir dispositivos PCI y añadimos la tarjeta gráfica. Después debemos de configurar el servidor Spice para que use el puerto 6000 en vez de asignar un puerto automáticamente.

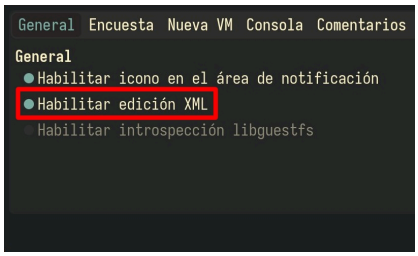


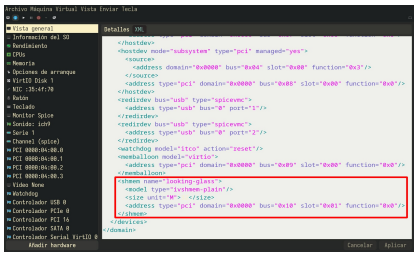
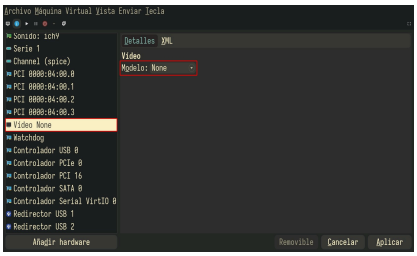
Ahora tenemos que activar la edición de XML en Virt-Manager *(Editar, Preferencias)* y editar el código XML de nuestra máquina virtual añadiendo estas líneas justo después de las sección `</membaloon>`:

```
<shmem name='looking-glass'>
  <model type='ivshmem-plain'/>
  <size unit='M'>32</size>
</shmem>
```

Iniciamos nuestra máquina virtual e instalamos nuestros drivers de NVIDIA o AMD y Looking Glass Host.

Finalmente cambiamos el driver de vídeo de *QXL* a *None* y borramos el dispositivo de tableta táctil. La próxima vez que iniciemos la máquina virtual desde Virt-Manager podremos usarla através de Looking Glass con gráficos acelerados por hardware.
Para que esto funcione debemos reiniciar nuestro ordenador o ejecutar el comando.



```
sudo install -g kvm -o $(whoami) -m 0660 /dev/null /dev/shm/looking-glass
```

Nota

Para más información sobre como configurar un VFIO passthought puede consultar:

- https://qqq.ninja/blog/post/vfio-on-arch/
- https://gitlab.com/risingprismtv/single-gpu-passthrough/-/wikis/home
- https://wiki.archlinux.org/title/PCI_passthrough_via_OVMF

4. VirtioFS

Podemos usar virtiofs para compartir directorios del host con nuestra máquina virtual. Para ello, añadimos los directorios deseados desde Virt-Manager (Añadir hardware, Sistema de archivos) eligiendo la *ruta de origen* (la carpeta que queremos compartir) y la *ruta objetivo* (el nombre con el que aparecerá en nuestra máquina invitada).

Necesitamos también los controladores VirtIO y el software de integración con el host. Podemos instalarlos desde la ISO que descargamos en https://fedorapeople.org. Esta contiene los instaladores necesarios *(virtio-win-gt-x64.msi y virtio-win-guest-tools.exe)*

Después de instalar los controladores, debemos instalar WinFSP para poder montar sistemas de archivos VirtioFS. Podemos activar el servicio *virtiofs* para que windows detecte nuestra carpeta compartida.

4.0.1. Más de una carpeta

Para compartir múltiples carpetas con nuestra máquina virtual, en lugar de utilizar el servicio *virtiofs*, es necesario crear un script que monte las carpetas deseadas directamente en la máquina virtual.

Para ello, primero ejecutamos desde cmd como Administrador:

```
"C:\Program Files (x86)\WinFsp\bin\fsreg.bat" virtiofs "C:\Program Files\Virtio-Win\VioFS\virtiofs.exe" "-t%1 -m%2"
```

Una vez que tenemos todo instalado, creamos un script '.bat' que monte nuestras carpetas en la máquina invitada. Si tenemos dos carpetas cuyas rutas objetivo son "documentos" y "videos" y queremos montarlas en "Y:" y "Z:", respectivamente, podemos crear un script para montar ambas carpetas:

```
"C:\Program Files (x86)\WinFsp\bin\launchctl-x64.exe" start virtiofs documentos documentos Y:
"C:\Program Files (x86)\WinFsp\bin\launchctl-x64.exe" start virtiofs videos videos Z:
```

5. Conexión bridge y RDP

Adicionalmente, podemos configurar nuestra máquina virtual para, a efectos prácticos, comportarse como un ordenador distinto en la red. En vez de usar un NAT y que nuestro host se encargue del re-direccionamiento al guest, podemos usar una conexión puente y hacer que nuestro router reconozca nuestra máquina virtual como un dipositivo distinto y le asigne su propia IP. De esta forma podemos abrir servicios desde nuestro guest y poder acceder a ellos.

5.1. Creación de nuestra red puente

Para crear nuestra conexión de puente, utilizaremos *NetworkManager*:

- Ejecutamos `nmtui`, seleccionamos *Modificar una conexión* y elegimos añadir una nueva conexión *(de tipo Puente)*.
- Debemos elegir que tarjeta de red queremos *añadir* para la conexión puenteada. Elegiremos *Ethernet*.
- Debemos eliminar la conexión Ethernet original de la lista de conexiones en (el host seguirá teniendo acceso a Internet; tanto la máquina anfitriona como la máquina invitada usarán la conexión de puente para acceder a Internet).

5.2. Invitado con conexión puenteada

Ahora tenemos que decirle a virt-manager que queremos usar nuestra conexión puenteada para la máquina virtual, podemos incluso borrar la red NAT *'default'* si no tenemos ninguna otra máquina virtual que queramos conectar a Internet.

Para usar la conexión puenteada en nuestra máquina virtual; en las propiedades de nuestra máquina virtual, cambiamos la *'Fuente de red'* de nuestro NIC a la conexión puenteada.

Adicionalmente, podemos ajustar la dirección MAC y establecer una IP estática asociada a esa dirección MAC en nuestro router. Lo cual facilitará configurar servicios y acceder a nuestra máquina invitada desde la red.

5.3. Configurar Escritorio remoto

Ahora podemos configurar nuestra máquina invitada para acceder remotamente a ella mediante RDP, para ello en Windows *(Sólo si tenemos Windows 10/11 Pro)* en Ajustes/Escritorio Remoto, activaremos el escritorio remoto.

Para conectaros a la máquina virtual de nuestra red debemos configurar el Firewall de nuestro guest para permitir las conexiones entrantes de RDP, para ello ejecutaremos este comando en cmd como Administrador: `netsh advfirewall firewall add rule name="RDP" protocol=TCP dir=in localport=3389 action=allow`

Podemos conectarnos a nuestra máquina usando el script `rdp-connect`

6. SSH

Para conectarnos a nuestro equipo remotamente debemos configurar OpenSSH. Podemos configurar SSH de forma muy básica con el script:

```
ssh-configure
```

Mi recomendación es no conformarse con esta configuración básica. Desactiva el login con el usuario root, desactiva el login por contraseña y usa claves públicas para conectarte.

6.1. VNC através de SSH

Podemos usar SSH y VNC para acceder a nuestro entorno gráfico de forma remota. Por defecto el script `autostart.sh` inicia un servidor VNC para poder conectarnos remotamente a nuestro equipo. Para hacer uso del servidor VNC necesitamos usar tunneling para que el tráfico de nuestro puerto 5900 *(servidor vnc)* le llegue a nuestro equipo remoto:

```
ssh usuario@255.255.255.255 -L 5900:localhost:5900
```

Sustituye *usuario* por el usuario de tu máquina, y *255.255.255.255* por la dirección IP de tu máquina. Para conectarte al equipo remotamente debes tener instalado algún cliente VNC, mi recomendación es remmina.

7. Firewall

Para instalar el firewall ejecuta:

```
sudo sh -c 'pacman -Sy ufw; systemctl enable ufw'
```

Podemos añadir unas reglas muy básicas a nuestro firewall *(que se aplicarán cuando reiniciemos nuestro equipo)* con:

```
sudo ufw limit 22/tcp; sudo ufw allow 80/tcp
sudo ufw allow 443/tcp; sudo ufw allow syncthing
sudo ufw default deny incoming; sudo ufw default allow outgoing
```

Nota

Si configuraste ssh con el script `bin/utils/ssh-configure` el firewall ya se configuró automáticamente

Aviso de Uso

Este proyecto es para uso personal. Se comparte con la intención de que pueda ser útil para otros, pero se proporciona tal cual, sin ninguna garantía. No se asume responsabilidad por ninguna pérdida de datos o problemas que puedan surgir del uso de este proyecto.