Fedora

Soluciones, personalizaciones y ajustes en Fedora. Para no volver a googlear lo mismo cada vez que reinstalo.

Conectividad

Conectividad

Puente de red en Fedora con nmcli

Este artículo explica cómo configurar un puente de red en Fedora utilizando NetworkManager a través de `nmcli`. La configuración permite que las máquinas virtuales (VMs) se conecten a la red a través del puente.

Pasos principales:

1. **Crear un dispositivo de tipo "bridge"** (br0, por ejemplo).
2. **Configurar la tarjeta de red física** (enp39s0) como esclava del puente.
3. **Asignar la dirección IP y puerta de enlace al puente**, en lugar de hacerlo a la interfaz física.

De esta forma, las VMs podrán conectarse a br0 y el sistema anfitrión seguirá teniendo conexión a la red.

Configuración con nmcli

1. **Eliminar y desactivar la configuración actual** de enp39s0 (Perfil 1) para evitar conflictos:

   ```
   nmcli con down "Perfil 1"
   nmcli con delete "Perfil 1"
   ```

   (Esto desconectará temporalmente la red; tener precaución si se usa SSH.)
2. **Crear el puente**:

   ```
   nmcli con add type bridge ifname br0 con-name br0
   ```

   - Esto genera una interfaz virtual denominada br0.
3. **Configurar una dirección IP estática (opcional)** en br0:

```
nmcli con modify br0 ipv4.method manual \
    ipv4.addresses "192.168.100.130/24" \
    ipv4.gateway "192.168.100.1" \
    ipv4.dns "192.168.100.3 192.168.100.240"
nmcli con modify br0 ipv6.method ignore
```

(Para DHCP, utilizar `ipv4.method auto` y omitir addresses, gateway y DNS.)

4. **Crear la conexión esclava** para la interfaz física enp39s0:

```
nmcli con add type bridge-slave ifname enp39s0 master br0 con-name br0-slave
```

(Este comando asigna enp39s0 al puente br0.)

5. **Activar el puente**:

```
nmcli con up br0
nmcli con up br0-slave
```

Con esta configuración, el sistema anfitrión tendrá la dirección IP en br0, y la interfaz física enp39s0 quedará como parte del puente. Las VMs podrán conectarse a br0 y obtener una dirección IP en la misma red.

Consideraciones adicionales:

- **Precaución**: Realizar cambios en la única interfaz de red puede provocar pérdida de conexión. En entornos SSH, se recomienda tomar precauciones o tener acceso físico a la máquina.
- **Conectividad de las VMs**: Las VMs deben configurarse para utilizar "br0" en lugar de "enp39s0" para conectarse a la red local.

Con esta configuración, el puente de red en Fedora quedará funcional y permitirá la conectividad esperada.

Conectividad

Usar resolv.conf directo en Fedora desactivando systemd-resolved

---

Este apunte documenta cómo forzar que Fedora utilice AdGuardHome como resolutor DNS directo, desactivando `systemd-resolved`, validando DNSSEC correctamente y asegurando que los cambios persistan tras reinicio.

Características

- Desactiva completamente `systemd-resolved`.
- Utiliza un `resolv.conf` estático y personalizado.
- Resuelve nombres DNS a través de AdGuardHome.
- Valida correctamente DNSSEC.
- Compatible con la resolución de nombres `.lan.internal` (o el que quieras) si se combina con reescrituras desde Tailscale.

Requisitos previos

- Fedora instalado y actualizado.
- AdGuardHome configurado como resolutor DNS.
- Haber verificado que AdGuard valida DNSSEC (por ejemplo, usando Cloudflare o NextDNS como upstreams).

Desactivar `systemd-resolved` en Fedora

Fedora utiliza por defecto un stub DNS en `127.0.0.53`, gestionado por `systemd-resolved`, lo cual puede interferir si queremos usar AdGuardHome como resolutor principal.

1. Comprobar estado actual

```
cat /etc/resolv.conf
```

Si ves algo como esto:

```
# This is /run/systemd/resolve/stub-resolv.conf managed by systemd-resolved(8)
nameserver 127.0.0.53
```

Estás usando el stub DNS. También puedes verificar con:

```
resolvectl status
```

Si ves `DNSSEC=no/unsupported`, significa que no se está validando correctamente.

2. Detener y deshabilitar systemd-resolved

```
sudo systemctl disable --now systemd-resolved
```

3. Eliminar el symlink de resolv.conf

```
sudo rm -f /etc/resolv.conf
```

4. Crear un resolv.conf estático personalizado (usa tus DNS locales)

```
sudo tee /etc/resolv.conf > /dev/null <<EOF
nameserver 192.168.1.5
nameserver 192.168.1.155
search lan.internal
options trust-ad edns0 timeout:1 attempts:2 rotate
EOF
```

Este archivo:

- Usa directamente tus resolutores locales (AdGuard).
- Habilita la confianza en firmas DNS (`trust-ad`).
- Permite extensiones modernas (`edns0`).
- Optimiza los tiempos de espera y el reintento.

Verificar funcionamiento

Resolución básica

```
dig google.com
```

DNSSEC fallido (debe dar `SERVFAIL`)

```
dig +dnssec dnssec-failed.org
```

DNSSEC correcto (debe dar `flags: ... ad`)

```
dig +dnssec sigok.verteiltesysteme.net
```

Resultado final

- Fedora ya no intercepta ni redirige peticiones DNS.
- `resolv.conf` permanece inmutable.
- Todo se resuelve a través de AdGuardHome.
- DNSSEC se valida correctamente.
- Puedes resolver hosts de Tailscale si has sincronizado `rewrites:` en AdGuard.
- En caso de que NetworkManager modifique nuestro resolv.conf, podemos aplicarle `sudo chattr +i /etc/resolv.conf` para que sea inmutable.

Enlaces de interés

- Evitar que resolv.conf sea modificado por aplicaciones externas

Herramientas de inteligencia artificial

Compilación de llama.cpp con HIP para GPUs AMD (ROCm)

Introducción

Este artículo documenta el proceso de **compilación de `llama.cpp` con backend HIP** para utilizar **aceleración por GPU en hardware AMD**, dentro del stack de inferencia local de modelos LLM. La integración se realiza sobre **ROCm**, interactuando directamente con la GPU mediante HIP y, opcionalmente, **rocWMMA** para optimizar Flash Attention en arquitecturas modernas.

El caso documentado corresponde a un sistema **Fedora 43**, con **Ryzen 9 5900X** y una **GPU RDNA3 (gfx1100)**. El enfoque es extrapolable a otras GPUs AMD compatibles, pero las decisiones y validaciones están hechas específicamente sobre este entorno.

Enfoque general / Arquitectura

El flujo seguido es intencionadamente directo:

- `llama.cpp` se compila **desde código fuente**.
- Se habilita el backend **GGML_HIP** mediante CMake.
- La compilación se ajusta explícitamente a la **arquitectura de la GPU** (`GPU_TARGETS`).
- El binario resultante (`llama-cli` o `llama-server`) ejecuta inferencia usando GPU vía HIP.

No se emplean contenedores ni capas intermedias. Se prioriza **control total del toolchain**, visibilidad de errores y reproducibilidad en el sistema base Fedora.

Requisitos previos

- Fedora 43.

- GPU AMD compatible con **ROCm / HIP** (RDNA2+, RDNA3 o CDNA).
- ROCm instalado y operativo.
- `clang` proporcionado por ROCm accesible vía `hipconfig`.
- CMake moderno.
- Espacio suficiente para la compilación.

Desarrollo

Clonado del repositorio

Se parte siempre del repositorio oficial de `llama.cpp`, evitando forks o parches externos que puedan introducir comportamientos no controlados.

Compilación con HIP habilitado

La compilación se realiza indicando explícitamente:

- El compilador HIP (`HIPCXX`).
- La ruta base de ROCm (`HIP_PATH`).
- Activación del backend HIP en GGML.
- Arquitectura concreta de la GPU (`gfx1100`).
- Build en modo `Release`.
- Paralelización elevada, acorde al Ryzen 9 5900X.

```
HIPCXX="$(hipconfig -l)/clang" HIP_PATH="$(hipconfig -R)" \
cmake -S . -B build \
  -DGGML_HIP=ON \
  -DGGML_HIP_ROCWMMA_FATTN=ON \
  -DGPU_TARGETS=gfx1100 \
  -DCMAKE_BUILD_TYPE=Release \
&& cmake --build build --config Release -- -j 24
```

Decisiones técnicas relevantes

- `GPU_TARGETS=gfx1100` Se fuerza la arquitectura RDNA3 para evitar builds genéricos innecesarios y reducir tanto tiempos de compilación como tamaño del binario.
- `GGML_HIP_ROCWMMA_FATTN=ON` Activa Flash Attention optimizado mediante `rocWMMA`, especialmente relevante en RDNA3.
- No se omite `GPU_TARGETS`: aunque es opcional, dejarlo implícito suele derivar en builds más lentas y menos predecibles.

Uso del binario resultante

Tras la compilación, los binarios quedan disponibles en:

```
build/bin/
```

La ayuda completa puede consultarse con:

```
./build/bin/llama-cli -h
```

Ejemplo mínimo de ejecución con `llama-cli`:

```
./build/bin/llama-cli \
  -m /ruta/a/tu/modelo.gguf \
  -p "Building a website can be done in 10 steps:"
```

La ruta del modelo depende del layout local de almacenamiento y no forma parte de esta documentación.

Ejecución como servicio con llama-server

Además del modo CLI, `llama.cpp` incluye **`llama-server`**, pensado para ejecutar inferencia persistente mediante una API HTTP (compatible con clientes tipo OpenAI).

Este modo es el habitual cuando se quiere:

- Mantener el modelo cargado en GPU.
- Evitar tiempos de arranque por inferencia.
- Consumir el modelo desde aplicaciones externas.

Ejemplo **básico y genérico**, válido para pruebas iniciales en GPU AMD con HIP:

```
./build/bin/llama-server \
  -m /ruta/a/tu/modelo.gguf \
  -ngl 99 \
  -c 4096 \
  --host 127.0.0.1 \
  --port 5000
```

Notas sobre los parámetros usados:

- `-m` Modelo GGUF a cargar.
- `-ngl 99` Intenta offloadear el mayor número posible de capas a la GPU. En sistemas con VRAM suficiente, equivale a "todo a GPU".

- `-c 4096` Tamaño de contexto moderado para uso general. Valores altos incrementan consumo de VRAM.
- `--host` / `--port` Dirección y puerto de escucha del servidor.

Este ejemplo evita opciones avanzadas (quantización de KV, Flash Attention explícito, buffers personalizados, plantillas de chat, etc.) para mantener un **baseline claro y portable**.

Aceleración adicional con rocWMMA

En arquitecturas **RDNA3+ o CDNA**, `rocWMMA` permite acelerar de forma significativa las operaciones de Flash Attention.

- Se incluye por defecto al instalar ROCm mediante el meta-paquete `rocm`.
- Alternativamente puede instalarse mediante paquetes `rocwmma-dev` / `rocwmma-devel`, según la distribución.

En escenarios no estándar, es posible añadir manualmente el include path:

```
-DCMAKE_CXX_FLAGS="-I<ruta>/rocwmma/library/include/"
```

Problemas detectados y resolución

Error: ROCm device library no encontrada

Error típico durante la compilación:

```
clang: error: cannot find ROCm device library;
```

Resolución aplicada:

1. Localizar dentro de `HIP_PATH` un directorio que contenga el archivo:
   ```
   oclc_abi_version_400.bc
   ```
2. Definir la variable `HIP_DEVICE_LIB_PATH` apuntando a dicho directorio:

```
HIPCXX="$(hipconfig -l)/clang" HIP_PATH="$(hipconfig -p)" \
HIP_DEVICE_LIB_PATH=<directorio_encontrado> \
cmake -S . -B build \
  -DGGML_HIP=ON \
```

```
  -DGPU_TARGETS=gfx1030 \
  -DCMAKE_BUILD_TYPE=Release \
&& cmake --build build -- -j 16
```

Arquitecturas GPU y detección

La arquitectura activa puede identificarse con:

```
rocminfo | grep gfx | head -1
```

Ejemplos habituales:

- `gfx1030` → RDNA2
- `gfx1100` → RDNA3 (RX 7900 XTX / XT / GRE)

Si la GPU no está oficialmente soportada:

- `HSA_OVERRIDE_GFX_VERSION=10.3.0` para RDNA2.
- `HSA_OVERRIDE_GFX_VERSION=11.0.0` para RDNA3.

Variables de entorno relevantes

- `HIP_VISIBLE_DEVICES` Limita qué GPU(s) son visibles para HIP.
- `HSA_OVERRIDE_GFX_VERSION` Fuerza compatibilidad con arquitecturas similares cuando el soporte no es oficial.

Resumen breve

- `llama.cpp` puede compilarse con **HIP** para GPUs AMD usando ROCm en Fedora 43.
- Forzar `GPU_TARGETS` mejora tiempos, tamaño y control del binario.
- `rocWMMA` aporta mejoras reales en Flash Attention sobre RDNA3.
- La compilación directa ofrece mayor control que soluciones encapsuladas.

Notas personales

HIP no es inmediato ni indulgente, pero una vez alineado el toolchain en Fedora, el resultado es estable y consistente. Para inferencia local seria en hardware AMD, ROCm sigue siendo el camino

más realista pese a sus fricciones.

Referencias

- llama.cpp – Build documentation
- Repositorio oficial de llama.cpp

Herramientas de inteligencia artificial

Configuración de OpenWebUI usando KoboldCpp-ROCm como API

Configuración de Open WebUI como interfaz web conectada a KoboldCpp-ROCm, encargado de la generación de texto mediante su API. Enfocado en tarjetas AMD; para NVIDIA existe el repositorio alternativo: KoboldCpp para NVIDIA.

1. Desplegar Open WebUI

El `docker-compose.yml` con el que lo despliego está disponible en mi Gitea:

docker-compose.yml en ChronosCMPS

Con ese archivo, basta ejecutar:

```
docker-compose up -d
```

Esto levanta la interfaz web en `http://localhost:3000`.

2. Instalar KoboldCpp-ROCm

Guía detallada aquí: KoboldCpp instalación y uso

Resumen rápido

1. Instalar KoboldCpp-ROCm en el sistema.
2. Descargar un modelo `.gguf` desde Hugging Face u otra fuente confiable.
3. Cargar el modelo y ejecutar KoboldCpp con parámetros acordes al hardware.

La API queda expuesta en:

```
http://localhost:5001
```

> **Nota:** sustituir `<IP-del-servidor>` por la IP local (ej: `http://192.168.1.x:5001/v1`).

---

3. Configurar Open WebUI

Acceder desde navegador: `http://localhost:3000`.

1. Crear usuario y contraseña inicial.
2. Entrar en **Configuración → Conexiones → Manage OpenAI API Connections**.
3. Añadir la URL de la API de KoboldCpp:

   ```
   http://192.168.1.x:5001/v1
   ```

   - En **key**, poner `0`.
   - Verificar conexión con el icono de recarga.
   - Guardar cambios.

---

4. Empezar a usarlo

1. Crear **Nuevo chat**.
2. Seleccionar el modelo cargado en KoboldCpp.
3. Escribir el primer mensaje.

Open WebUI ya funciona como interfaz web para KoboldCpp-ROCm.

Referencias

- Repositorio de Open WebUI
- Repositorio de KoboldCpp-ROCm
- Repositorio de KoboldCpp para NVIDIA
- docker-compose en mi Gitea

KoboldCPP: instalación y uso

Esta guía explica cómo configurar y compilar KoboldCPP optimizado para ROCm en Fedora, utilizando el fork koboldcpp-rocm. Está diseñado para aprovechar GPUs AMD con soporte para ROCm.

Instalación y configuración en Fedora

1. Clonar el repositorio

Primero, clona el repositorio de KoboldCPP-ROCm y accede al directorio:

```
git clone https://github.com/YellowRoseCx/koboldcpp-rocm.git
cd koboldcpp-rocm
```

2. Instalar herramientas necesarias

Ejecuta el siguiente comando para instalar las herramientas y librerías necesarias:

```
sudo dnf install rocm-*
```

> **Nota:** Si utilizas un shell como `zsh`, ejecuta este comando en un entorno `bash` para evitar problemas con el autocompletado:
>
> ```
> bash -c 'sudo dnf install rocm-*'
> ```

3. Instalar librerías de desarrollo específicas

Instala las librerías de desarrollo que necesitas para compilar KoboldCPP con soporte ROCm:

```
sudo dnf install rocblas-devel hipblas-devel
```

4. Crear un alias para actualizar KoboldCPP

Puedes crear un alias para compilar KoboldCPP con tus especificaciones de hardware. Por ejemplo:

```
alias updatekobold="make clean && make -j24 LLAMA_HIPBLAS=1 GPU_TARGETS=gfx1100"
```

> - `-j24`: Indica el número de núcleos de tu CPU para paralelizar la compilación. Ajusta el número según los núcleos disponibles en tu sistema.
> - `GPU_TARGETS=gfx1100`: Especifica el objetivo de tu GPU. Puedes identificar el objetivo de tu GPU utilizando el siguiente comando:
>
> ```
> rocminfo | grep "Name" | grep "gfx"
> ```
>
> **Ejemplo de salida:**
>
> ```
> Name:                    gfx1100
> Name:                    amdgcn-amd-amdhsa--gfx1100
> ```
>
> En este caso, la salida corresponde a una AMD 7900XTX.

5. Actualizar y compilar KoboldCPP

Cuando haya actualizaciones en el repositorio, sigue estos pasos para mantener tu instalación al día:

1. Accede al directorio del proyecto:
   ```
   cd koboldcpp-rocm
   ```
2. Actualiza el repositorio:
   ```
   git pull
   ```
3. Compila de nuevo el proyecto:
   ```
   make clean && make -j24 LLAMA_HIPBLAS=1 GPU_TARGETS=gfx1100
   ```

Mantente atento a las actualizaciones en el repositorio de YellowRoseCx, ya que pueden incluir correcciones y mejoras importantes.

6. Ejecutar la interfaz gráfica de KoboldCPP

Para utilizar la interfaz gráfica de KoboldCPP, ejecuta el siguiente comando desde el directorio del proyecto:

```
python koboldcpp.py
```

> **Nota sobre errores de tkinter:**
> Si al ejecutar la interfaz gráfica te encuentras con errores relacionados con **tkinter**, puedes probar a instalarlo con este comando:
>
> ```
> pip install ntk
> ```
>
> Esta solución me ha funcionado en mi caso, pero si persisten los problemas, puedes enviarme un mensaje y trataré de ayudar.

Características adicionales de KoboldCPP

- **API para conectarse a otros frontends**:
  KoboldCPP incluye una API que permite conectarse con otros frontends de procesamiento de texto. Esto amplía su funcionalidad al integrarlo con herramientas externas.
- **KoboldLite**:
  Además, KoboldCPP incluye una interfaz llamada **KoboldLite**, que permite probar los modelos de texto que cargues de forma sencilla. Es una forma rápida de experimentar con los modelos directamente.

Trabajando con modelos `.gguf`

KoboldCPP está diseñado para ejecutar modelos en formato `.gguf`. Puedes descargar modelos compatibles desde plataformas como Hugging Face. Escoge el modelo adecuado según el uso que vayas a darle. Una vez descargado, puedes cargar el modelo en KoboldCPP para comenzar a trabajar con él.

Nota importante

Esta guía se basa en mi experiencia personal instalando y configurando KoboldCPP en Fedora. Aunque me ha funcionado, puede que en otros casos no lo haga debido a diferencias en hardware o configuración del sistema. Por ello, recomiendo revisar siempre las instrucciones oficiales del repositorio de YellowRoseCx, donde encontrarás la documentación más actualizada y detallada.

Mantenimiento

Mantenimiento

Limpieza y mantenimiento Post-Actualización de Fedora

Después de actualizar Fedora, conviene hacer tareas de limpieza y mantenimiento para evitar conflictos, dejar el sistema optimizado y asegurarse de que no queden restos de configuraciones antiguas. Esta guía está basada en la documentación oficial de Fedora.

---

1. Archivos de configuración: rpmnew y rpmsave

Cuando Fedora detecta cambios en archivos de configuración, puede crear:

- `.rpmnew`: nueva versión del archivo.
- `.rpmsave`: respaldo del antiguo.

Para revisarlos:

```
sudo dnf install rpmconf
sudo rpmconf -a
```

Revisa especialmente:

- `/etc/ssh/sshd_config`
- `/etc/nsswitch.conf`
- `/etc/ntp.conf`
- `/etc/yum.repos.d/` (si tienes software de terceros)

---

2. Actualizar GRUB (solo en BIOS)

Si tu sistema usa BIOS (no UEFI), hay que reinstalar GRUB:

```
sudo grub2-install /dev/sda
```

(No pongas el número de partición).

---

3. Eliminar paquetes retirados

Algunos paquetes dejan de mantenerse entre versiones:

```
sudo dnf install remove-retired-packages
remove-retired-packages
# o si vienes de dos versiones anteriores:
remove-retired-packages 39
```

---

4. Limpiar duplicados

Primero asegúrate de estar al día:

```
sudo dnf upgrade
```

Luego elimina duplicados:

```
sudo dnf remove --duplicates
```

---

5. Eliminar paquetes obsoletos

Lista lo que ya no está en repositorios:

```
sudo dnf list --extras
```

Y limpia con:

```
sudo dnf remove $(sudo dnf repoquery --extras --exclude=kernel,kernel-\*)
```

---

6. Borrar kernels antiguos

Lista:

```
sudo dnf repoquery --installonly
```

Y elimina los antiguos:

```
old_kernels=($(dnf repoquery --installonly --latest-limit=-1 -q))
if [ "${#old_kernels[@]}" -gt 0 ]; then
    sudo dnf remove "${old_kernels[@]}"
else
    echo "No old kernels found"
fi
```

---

7. Limpiar llaves GPG viejas

```
sudo dnf install clean-rpm-gpg-pubkey
sudo clean-rpm-gpg-pubkey
```

---

8. Eliminar symlinks rotos

```
sudo dnf install symlinks
sudo symlinks -r /usr | grep dangling
sudo symlinks -r -d /usr
```

---

9. Actualizar kernel de rescate

```
sudo rm /boot/*rescue*
sudo kernel-install add "$(uname -r)" "/lib/modules/$(uname -r)/vmlinuz"
sudo dnf install dracut-config-rescue
```

Resumen

- Usa `rpmconf` para revisar configuraciones.
- Si tienes BIOS, reinstala GRUB.

- Limpia paquetes retirados, duplicados y extras.
- Revisa llaves GPG y symlinks.
- Actualiza el kernel de rescate si hace falta.

Todo esto deja tu Fedora al día y sin residuos de la versión anterior.

Mantenimiento

Pasos esenciales tras instalar Fedora

Guía de ajustes prácticos y mejoras recomendadas tras instalar Fedora desde cero. Pensado para dejar el sistema funcional, limpio y adaptado al uso diario.

Ajustes de DNF para acelerar descargas

```
sudo nano /etc/dnf/dnf.conf
```

Añadir:

```
max_parallel_downloads=10
fastestmirror=true
```

Actualizar el sistema

```
sudo dnf update
```

Repositorios RPM Fusion

```
sudo dnf install https://mirrors.rpmfusion.org/free/fedora/rpmfusion-free-release-$(rpm -E %fedora).noarch.rpm
sudo dnf install https://mirrors.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-$(rpm -E %fedora).noarch.rpm
sudo dnf update
```

Software esencial

Fuentes de Microsoft

```
sudo dnf install msttcore-fonts-installer
sudo msttcore-fonts-installer install
```

GNOME Tweaks

```
sudo dnf install gnome-tweaks
```

Extensiones de GNOME

```
sudo dnf install gnome-extensions-app
```

Programas básicos

```
sudo dnf install vlc gimp libreoffice
```

Flatpak y Flathub

```
sudo dnf install flatpak
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
```

Aceleración de video por hardware

Paquetes base

```
sudo dnf install ffmpeg-libs libva libva-utils
```

Intel

```
sudo dnf swap libva-intel-media-driver intel-media-driver --allowerasing
sudo dnf install libva-intel-driver
```

AMD

```
sudo dnf swap mesa-va-drivers mesa-va-drivers-freeworld
sudo dnf swap mesa-vdpau-drivers mesa-vdpau-drivers-freeworld
```

Verificación:

```
vainfo
vdpauinfo
```

H.264 en Firefox

```
sudo dnf install mozilla-openh264
sudo dnf config-manager --set-enabled fedora-cisco-openh264
```

Optimizaciones extra

Servicios innecesarios

```
systemctl list-unit-files --state=enabled
sudo systemctl disable nombre-del-servicio
```

---

zRAM

```
sudo dnf install zram-generator-defaults
```

Copias de seguridad (Deja Dup)

```
sudo dnf install deja-dup
```

Herramientas de desarrollo

```
sudo dnf groupinstall "Development Tools"
```

Firewall

```
sudo systemctl enable firewalld
sudo systemctl start firewalld
```

Compresión

```
sudo dnf install unzip unrar p7zip p7zip-plugins
```

Actualizaciones automáticas

```
sudo dnf install dnf-automatic
sudo systemctl enable --now dnf-automatic.timer
```

Notas

- Puedes adaptar o extender esta configuración según tu entorno o necesidades específicas.
- Fedora es bastante modular, así que conviene revisar qué componentes realmente necesitas.

Referencias:

- Guía original en GitHub
- It's FOSS - Things to do after installing Fedora

Multimedia

Multimedia

Aplicación de SpotX-Bash sobre Spotify en Flatpak

Introducción

Este artículo documenta la aplicación de **SpotX-Bash** sobre el cliente oficial de **Spotify distribuido vía Flatpak**, dentro del stack de aplicaciones de usuario.

La integración se realiza directamente sobre el binario oficial, aplicando parches locales que modifican el comportamiento del cliente sin introducir clientes alternativos, proxies externos ni reimplementaciones del protocolo.

Enfoque general / Arquitectura

El flujo se apoya en tres decisiones técnicas claras:

- **Spotify instalado vía Flatpak**, priorizando aislamiento, rutas predecibles y reversibilidad limpia.
- **Bloqueo explícito de actualizaciones** del paquete para evitar que el cliente sobrescriba los parches aplicados.
- **Ejecución directa de SpotX-Bash**, que actúa sobre los recursos del cliente oficial ya instalado.

No se intercepta tráfico de red ni se modifican APIs externas. Todo el cambio ocurre en local, sobre el cliente.

Características por defecto de SpotX-Bash

Con la configuración estándar (sin flags adicionales), SpotX-Bash aplica:

- Bloqueo de anuncios de audio.
- Eliminación de banners y elementos promocionales en la interfaz.

- Desactivación de componentes de tracking y logging innecesarios.
- Limpieza de recursos asociados a anuncios dentro del cliente.
- Conservación del cliente oficial y su flujo normal de ejecución.

Opcionalmente, SpotX-Bash permite (no aplicado por defecto):

- Activar modo desarrollador.
- Ocultar secciones no musicales (podcasts, audiolibros).
- Habilitar funciones experimentales.

Desarrollo

Instalación de Spotify vía Flatpak

Spotify se instala desde Flathub:

```
flatpak install flathub com.spotify.Client
```

Bloqueo de actualizaciones

Se bloquean futuras actualizaciones automáticas del paquete:

```
flatpak mask com.spotify.Client
```

Esto evita que Spotify sobrescriba los cambios aplicados por SpotX-Bash.

Aplicación de SpotX-Bash

SpotX-Bash se ejecuta directamente desde el repositorio oficial:

```
bash <(curl -sSL https://spotx-official.github.io/run.sh)
```

El comando se conserva documentado explícitamente para mantener trazabilidad y facilitar reaplicación futura.

Validación

Comprobaciones mínimas tras la aplicación:

- Spotify inicia correctamente desde Flatpak.

- No se reproducen anuncios de audio.
- No aparecen banners promocionales en la interfaz.
- El paquete permanece en estado `masked`.
- No se observan errores relevantes en la ejecución del cliente.

Resumen breve

Spotify se ejecuta desde Flatpak con actualizaciones bloqueadas y parches locales aplicados mediante SpotX-Bash, manteniendo el cliente oficial y priorizando control y estabilidad.

Referencias

- SpotX-Bash – GitHub

Multimedia

Configurar Fedora para audio Hi-Res con un DAC

Introducción

Este artículo documenta cómo está configurado Fedora para reproducir audio en alta resolución utilizando un DAC USB dedicado (en mi caso: **ADuM4160 → SMSL SU-1 → Douk U3**). Se describen los overrides aplicados a PipeWire y las reglas añadidas en WirePlumber para mejorar estabilidad, ampliar sample-rates permitidos y evitar autosuspend USB.

Características

- Reproducción estable con DAC USB aislado.
- Allowed-rates amplios para audio Hi-Res.
- Buffers optimizados para reducir pops, drift y cortes.
- Autosuspend USB desactivado completamente.
- Validación real del sample-rate mediante herramientas nativas de PipeWire.

Requisitos previos

- Fedora con PipeWire y WirePlumber (configuración predeterminada).
- DAC USB externo.
- Carpetas necesarias creadas previamente:
  - **PipeWire overrides:** `~/.config/pipewire/pipewire.conf.d/`
  - **WirePlumber rules:** `/etc/wireplumber/main.lua.d/`
- Permisos para modificar archivos del sistema.

Desarrollo / pasos

Qué se hizo y por qué

Fedora prioriza la compatibilidad general: mezcla todo a 48 kHz, limita sample-rates y aplica autosuspend USB. Con DACs externos esto genera resampling innecesario, latencia irregular y microcortes.

Para evitarlo se realizaron dos acciones principales:

1. **Overrides en PipeWire** para ampliar tasas permitidas, fijar un clock base estable y ajustar los buffers del DAC.
2. **Reglas en WirePlumber** para impedir la suspensión automática del DAC.

El resultado es una reproducción estable, sin conversiones innecesarias y adecuada para DACs conectados mediante aisladores.

Configuración utilizada (solo enlaces)

- 95-hifi-dac-usb.conf → Override completo de PipeWire.
- 99-usb-dac.lua → Regla de WirePlumber para desactivar autosuspend.

Validación

Para comprobar el sample-rate real durante la reproducción:

```
pw-top
```

Debe aparecer, por ejemplo, `rate: 96000` cuando se reproduce audio a 96 kHz.

Para ver sinks disponibles (en el idioma del sistema):

```
pactl list sinks | grep Name
```

Aplicar cambios tras modificar archivos:

```
systemctl --user restart pipewire pipewire-pulse wireplumber
```

Funcionamiento práctico

El sistema mantiene el DAC siempre activo, evita fluctuaciones del bus USB y acepta sample-rates altos sin forzar resampling. Con reproductores usando salida ALSA directa (DeadBeef, MPD), se respeta la frecuencia de origen siempre que el DAC la soporte.

Errores comunes o decisiones importantes

- **Editar `pipewire.conf` directamente**: no recomendable; los overrides son más estables y fáciles de mantener.
- **Usar `pactl` para validar sample-rate**: solo muestra la configuración del sink. La validación correcta es con `pw-top`.
- **Mantener autosuspend USB activado**: provoca cortes, reconexiones y drift del DAC.

Resumen breve

Fedora se ajusta para audio Hi-Res mediante overrides en PipeWire y reglas en WirePlumber. Se amplían sample-rates permitidos, se estabilizan buffers y se desactiva el autosuspend del DAC. La validación real del stream se realiza con `pw-top`.

Notas personales

- El aislador USB y el Douk U3 funcionan mejor con quantums amplios.
- La configuración actual elimina por completo los pops al cambiar de sample-rate.

Archivos de configuración

Los archivos se mantienen en Gitea:

- 95-hifi-dac-usb.conf
- 99-usb-dac.lua

Referencias

- Configuración en Gitea
- Documentación de PipeWire
- WirePlumber

Personalización

Personalización

Cambio del tamaño del cursor en Fedora

Introducción

En Fedora con entorno de escritorio Gnome, puede ocurrir que el tamaño del cursor varíe entre el escritorio y las aplicaciones. Esto suele suceder por diferencias en la escala de la interfaz o en la configuración de las aplicaciones que utilizan X11 y Wayland de manera simultánea.

Afortunadamente, podemos solucionar este problema ajustando manualmente el tamaño del cursor para que sea uniforme en todo el sistema.

Solución: Forzar el mismo tamaño de cursor en todo el sistema

La forma más sencilla de asegurarse de que el cursor tenga un tamaño consistente en todo el sistema es establecerlo manualmente con el siguiente comando:

1. **Abrir la terminal** y ejecutar:

   ```
   gsettings set org.gnome.desktop.interface cursor-size 32
   ```

   (Puedes cambiar el `32` por otro valor, como `48`, si necesitas que sea más grande.)
2. **Reiniciar la sesión** para que los cambios surtan efecto.

Este método asegura que el tamaño del cursor sea uniforme tanto en el escritorio como en las aplicaciones que se ejecutan bajo Wayland o X11.

Conclusión

Si notas diferencias en el tamaño del cursor entre el escritorio y las aplicaciones en Fedora con Gnome, el ajuste mediante `gsettings` es la forma más rápida y efectiva de solucionar el problema. Con este método, evitarás inconsistencias visuales y mejorarás la experiencia de uso del sistema.

Personalización

Personalizar nuestro GRUB

Introducción

GRUB (GRand Unified Bootloader) es el gestor de arranque utilizado en Fedora y muchas otras distribuciones de Linux. Cambiar su apariencia con un tema personalizado puede mejorar la estética del arranque del sistema. En esta guía, explicaremos cómo cambiar el tema de GRUB en Fedora 41.

Requisitos previos

Antes de comenzar, asegúrate de tener privilegios de superusuario (root o sudo) y de haber instalado `git` en tu sistema. Si no lo tienes, instálalo con:

```
sudo dnf install git -y
```

Descarga de temas para GRUB

Los temas de GRUB pueden descargarse desde el repositorio de distro-grub-themes. Para clonar este repositorio, usa el siguiente comando:

```
git clone https://github.com/AdisonCavani/distro-grub-themes.git
```

Una vez descargado, accede a la carpeta `themes` y elige el tema que desees. También puedes descargar un tema específico desde la sección de *Releases* del repositorio en GitHub.

Instalación del tema

1. **Mover el tema al directorio de GRUB**
   Accede al directorio de temas de GRUB. Si no existe, créalo:

```
sudo mkdir -p /boot/grub2/themes
```

Copia la carpeta del tema seleccionado al directorio de temas:

```
sudo cp -r [ruta_del_tema] /boot/grub2/themes/
```

(Reemplaza `[ruta_del_tema]` con la ruta del tema que hayas seleccionado.)

Configuración del archivo GRUB

Ahora debemos editar el archivo de configuración de GRUB para establecer el nuevo tema.

1. **Abrir el archivo de configuración de GRUB:**

   ```
   sudo nano /etc/default/grub
   ```

2. **Buscar la línea con `GRUB_THEME` o agregarla si no existe:**

   ```
   GRUB_THEME="/boot/grub2/themes/[nombre_del_tema]/theme.txt"
   ```

   (Reemplaza `[nombre_del_tema]` con el nombre de la carpeta del tema elegido.)
3. **Ejemplo de archivo `/etc/default/grub` modificado:**

   ```
   GRUB_TIMEOUT=5
   GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
   GRUB_DEFAULT=saved
   GRUB_DISABLE_SUBMENU=true
   #GRUB_TERMINAL_OUTPUT="console"
   GRUB_CMDLINE_LINUX="rhgb transparent_hugepage=always quiet mitigations=off"
   GRUB_DISABLE_RECOVERY="true"
   GRUB_ENABLE_BLSCFG=true
   GRUB_GFXMODE=2560x1440
   GRUB_THEME="/boot/grub2/themes/fedora/theme.txt"
   ```

Aplicar los cambios

Después de realizar las modificaciones necesarias, es importante regenerar la configuración de GRUB:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
sudo grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

Este proceso actualizará GRUB para reflejar los cambios en su apariencia.

Reiniciar y probar

Finalmente, reinicia tu sistema para ver el nuevo tema aplicado:

```
sudo reboot
```

Si algo sale mal y el sistema no arranca correctamente, puedes acceder al modo de recuperación de GRUB y revertir los cambios editando el archivo `/etc/default/grub` desde una sesión en modo rescate.

Conclusión

Personalizar el aspecto de GRUB en Fedora 41 es un proceso relativamente sencillo que permite mejorar la experiencia de arranque del sistema. Siguiendo esta guía, puedes cambiar y probar diferentes temas para encontrar el que mejor se adapte a tus preferencias.

Problemas y soluciones

Conectividad en máquinas virtuales KVM/QEMU falla tras actualizar a Fedora 41

Descripción del problema

Tras actualizar a Fedora 41, las máquinas virtuales (VMs) gestionadas por KVM/QEMU perdieron acceso a internet. Aunque las VMs podían obtener direcciones IP mediante DHCP, no lograban conectarse a redes externas. Este problema ocurrió sin haber realizado cambios explícitos en la configuración del sistema o las redes virtuales.

Causa identificada

Con la actualización a Fedora 41, el backend predeterminado del firewall cambió de **iptables** a **nftables**. Sin embargo, `libvirt`, la herramienta que gestiona las redes virtuales de KVM/QEMU, seguía esperando **iptables** como backend. Esta discrepancia impedía que las reglas NAT necesarias se aplicaran correctamente, causando la pérdida de conectividad.

Solución

La solución consiste en configurar explícitamente `libvirt` para que utilice **iptables** como backend de firewall, asegurándose de que sea compatible con el sistema.

Pasos a seguir

1. **Editar la configuración de `libvirt`:**
   Abre el archivo de configuración de red de `libvirt`:

   ```
   sudo nano /etc/libvirt/network.conf
   ```

2. **Configurar el backend del firewall:**
   Dentro del archivo, añade la siguiente línea (o descoméntala si ya está presente pero comentada):

   ```
   firewall_backend = "iptables"
   ```

   Esto obliga a `libvirt` a utilizar **iptables** como backend, en lugar de **nftables**.
3. **Reiniciar el servicio de `libvirtd`:**
   Aplica los cambios reiniciando el servicio de `libvirt`:

   ```
   sudo systemctl restart libvirtd
   ```

4. **Verificar la conectividad:**
   Reinicia tus VMs y prueba su conexión a internet. Deberían estar nuevamente en línea.

Consideraciones adicionales

- Si prefieres utilizar **nftables**, es posible configurar manualmente las reglas necesarias para permitir el tráfico de las VMs. Sin embargo, en la mayoría de los casos, volver a **iptables** es una solución más rápida y sencilla.
- Revisa si `libvirt` introduce soporte completo para **nftables** en futuras actualizaciones, ya que Fedora tiende a migrar hacia esta tecnología.

Referencias

- KVM Guests no longer can access past virb0 (Fedora Discussion)