Installation Wolf (Multi-User Cloud Gaming)

Objectif : Faire tourner plusieurs instances de Steam indépendantes sur un ou plusieurs GPU, accessibles via Moonlight avec support manettes, en utilisant Wolf et la méthode Nvidia (Manual) sur Unraid.

---

1. Prérequis Système (Unraid)

A. Plugin Nvidia

Installer le plugin **"Nvidia Driver"** (par ich777) depuis les Community Applications.

B. Correctifs NVENC et NvFBC pour les pilotes Nvidia

Nvidia bloque le nombre de flux NVENC au-delà de 3. Pour lever cette limite, on applique un patch au chargement du driver.

Installer le plugin **"User Scripts"** depuis les Community Applications.
Créer ensuite un script, par exemple nommé **"Nvidia NVENC Patch"** avec comme contenu :

```
#!/bin/bash
wget https://raw.githubusercontent.com/keylase/nvidia-patch/master/patch.sh -O /tmp/nvidia-patch.sh
chmod +x /tmp/nvidia-patch.sh
/tmp/nvidia-patch.sh
```

À lancer une première fois manuellement.
Dans User Scripts, régler le script pour qu'il s'exécute **à chaque démarrage de l'Array**.

C. Activation du "Kernel Mode Setting" (KMS)

Wolf a besoin de contrôler l'affichage au niveau du noyau pour créer les sessions graphiques. Sans ça, on risque l'écran noir.

1. Aller dans **Main > Flash > Syslinux Configuration**.
2. Dans l'entrée `Unraid OS`, ajouter à la fin de la ligne :

   ```
   nvidia-drm.modeset=1
   ```

3. **Redémarrer Unraid**.
4. Vérifier que le KMS est bien activé :

   ```
   cat /sys/module/nvidia_drm/parameters/modeset
   ```

   Il doit renvoyer : `Y`

D. Persistance des Manettes (uinput / uhid)

Pour que Unraid recrée correctement les règles de manettes virtuelles à chaque démarrage (sinon elles disparaissent après reboot).

1. Ouvrir le terminal Unraid.
2. Éditer le fichier de boot : `nano /boot/config/go`
3. Ajouter ce bloc à la fin du fichier :

```
# --- Wolf : Support Manettes Virtuelles ---
# 1. Permissions uinput
chmod 666 /dev/uinput || true

# 2. Règles UDEV
cat << 'EOF' > /etc/udev/rules.d/85-wolf-virtual-inputs.rules
KERNEL=="uinput", SUBSYSTEM=="misc", MODE="0660", GROUP="input", OPTIONS+="static_node=uinput"
KERNEL=="uhid", TAG+="uaccess"
SUBSYSTEMS=="input", ATTRS{id/vendor}=="ab00", MODE="0660", GROUP="input",
ENV{ID_SEAT}="seat9"
SUBSYSTEMS=="input", ATTRS{name}=="Wolf X-Box One (virtual) pad", MODE="0660", GROUP="input"
SUBSYSTEMS=="input", ATTRS{name}=="Wolf PS5 (virtual) pad", MODE="0660", GROUP="input"
SUBSYSTEMS=="input", ATTRS{name}=="Wolf gamepad (virtual) motion sensors", MODE="0660",
GROUP="input"
SUBSYSTEMS=="input", ATTRS{name}=="Wolf Nintendo (virtual) pad", MODE="0660", GROUP="input"
EOF

# 3. Recharger les règles
udevadm control --reload-rules && udevadm trigger
```

---

2. Installation de Wolf (Méthode Nvidia Manual sur Unraid)

Avec la méthode Nvidia Manual, on va :

- Construire une image Docker qui contient les fichiers du driver Nvidia.
- Pré-remplir un volume Docker `nvidia-driver-vol` avec ces fichiers.
- Monter ce volume dans le conteneur Wolf via un template Unraid.

A. Construire l'image `gow/nvidia-driver:latest`

Dans le terminal Unraid :

```
cd /root


curl https://raw.githubusercontent.com/games-on-whales/gow/master/images/nvidia-driver/Dockerfile \
  | docker build -t gow/nvidia-driver:latest -f - --build-arg NV_VERSION=$(cat /sys/module/nvidia/version) .
```

Cela crée l'image locale `gow/nvidia-driver:latest` avec la version de driver actuellement chargée.

B. Créer / mettre à jour le volume Docker `nvidia-driver-vol`

On pré-populue un volume Docker avec les libs Nvidia :

```
docker create --rm \
  --mount source=nvidia-driver-vol,destination=/usr/nvidia \
  gow/nvidia-driver:latest sh
```

Vérifier qu'il existe :

```
docker volume ls | grep nvidia-driver
```

Il doit afficher :**`nvidia-driver-vol`**

C. Script User Scripts : reconstruire le volume après chaque reboot

Important : à chaque mise à jour du driver Nvidia, il faut reconstruire l'image et repopuler le volume, sinon Wolf utilise des libs qui ne correspondent plus au driver chargé.

Le plus simple sur Unraid : utiliser le plugin **User Scripts** avec un script lancé au démarrage de l'Array.

1. Dans User Scripts, créer un script nommé par exemple **"Wolf - Rebuild Nvidia driver volume"**.
2. Contenu du script :

```
#!/bin/bash
# Wolf - (Re)construction de l'image et du volume driver Nvidia + restart Wolf

WOLF_CONTAINER_NAME="wolf"
DRIVER_IMAGE="gow/nvidia-driver:latest"
DRIVER_VOLUME="nvidia-driver-vol"
NOTIFY="/usr/local/emhttp/webGui/scripts/notify"

notify_msg() {
  # $1 = niveau (normal|warning|alert)
  # $2 = sujet
  # $3 = description
  if [ -x "${NOTIFY}" ]; then
    "${NOTIFY}" -e "Wolf Nvidia" -s "$2" -d "$3" -i "$1"
  fi
}

echo ">>> [Wolf] $(date) - Début de la reconstruction Nvidia (image + volume)"
notify_msg "normal" "Début mise à jour Wolf Nvidia" "Reconstruction de l'image Nvidia et du
volume ${DRIVER_VOLUME} en cours..."

# Vérif rapide de la version de driver en cours
if [ -r /sys/module/nvidia/version ]; then
  DRV_VER=$(cat /sys/module/nvidia/version)
  echo ">>> [Wolf] Driver Nvidia détecté : ${DRV_VER}"
else
  echo "!!! [Wolf] Impossible de lire /sys/module/nvidia/version (driver Nvidia non chargé ?)"
  notify_msg "warning" "Driver Nvidia introuvable" "Impossible de lire
/sys/module/nvidia/version. Vérifiez que le driver est bien chargé avant de relancer le
script."
fi

echo ">>> [Wolf] Reconstruction de l'image ${DRIVER_IMAGE}..."

curl -s https://raw.githubusercontent.com/games-on-whales/gow/master/images/nvidia-
driver/Dockerfile \
  | docker build -t "${DRIVER_IMAGE}" -f - \
      --build-arg NV_VERSION=$(cat /sys/module/nvidia/version) \
      .
```

```
if [ $? -ne 0 ]; then
  echo "!!! [Wolf] Échec du build de l'image ${DRIVER_IMAGE}"
  notify_msg "alert" "Échec build image Nvidia" "Le build de l'image ${DRIVER_IMAGE} a échoué.
Consultez les logs du script / de Docker pour plus de détails."
  exit 1
fi

echo ">>> [Wolf] (Re)population du volume ${DRIVER_VOLUME}..."

# On crée un conteneur temporaire pour initialiser le volume,
# comme dans la doc officielle Wolf, sans jamais l'exécuter.
CID=$(docker create \
  --mount source=${DRIVER_VOLUME},destination=/usr/nvidia \
  "${DRIVER_IMAGE}" sh 2>/dev/null)

if [ -z "${CID}" ]; then
  echo "!!! [Wolf] Échec de la création du conteneur temporaire pour le volume
${DRIVER_VOLUME}"
  notify_msg "alert" "Échec initialisation volume Nvidia" "Impossible de créer un conteneur
temporaire pour le volume ${DRIVER_VOLUME}. Vérifiez Docker et le volume."
  exit 1
fi

echo ">>> [Wolf] Conteneur temporaire ${CID} créé pour initialiser le volume ${DRIVER_VOLUME}"

# On supprime tout de suite le conteneur (le volume, lui, reste)
docker rm "${CID}" >/dev/null 2>&1 || true
echo ">>> [Wolf] Conteneur temporaire ${CID} supprimé"

echo ">>> [Wolf] Redémarrage du conteneur ${WOLF_CONTAINER_NAME}..."

# On essaye de le stopper (au cas où il tourne déjà)
if docker stop "${WOLF_CONTAINER_NAME}" 2>/dev/null; then
  echo ">>> [Wolf] Conteneur ${WOLF_CONTAINER_NAME} stoppé."
else
  echo ">>> [Wolf] Conteneur ${WOLF_CONTAINER_NAME} n'était pas en cours d'exécution."
fi

# Puis on le démarre
if docker start "${WOLF_CONTAINER_NAME}" 2>/dev/null; then
```

```
  echo ">>> [Wolf] Conteneur ${WOLF_CONTAINER_NAME} démarré."
  notify_msg "normal" "Wolf Nvidia en place" "La mise en place des bibliothèques Nvidia et le redémarrage de Wolf se sont terminés avec succès."
else
  echo "!!! [Wolf] Impossible de démarrer le conteneur ${WOLF_CONTAINER_NAME} (à vérifier dans l'UI Docker)."
  notify_msg "alert" "Échec redémarrage Wolf" "Impossible de démarrer le conteneur ${WOLF_CONTAINER_NAME}. Vérifiez la configuration dans l'UI Docker."
  exit 1
fi

echo ">>> [Wolf] $(date) - Opération terminée."
```

3. Configurer le script pour s'exécuter à **chaque démarrage de l'Array**.

À chaque reboot (après une maj de driver par exemple), le volume sera automatiquement synchronisé avec la bonne version.

> **Ce que fait le script :**
> - Vérifie la version du driver Nvidia actuellement chargée sur Unraid.
> - Rebuild l'image Docker gow/nvidia-driver:latest avec cette version de driver.
> - (Re)inititialise le volume Docker nvidia-driver-vol avec les librairies Nvidia de l'image.
> - Redémarre automatiquement le conteneur wolf pour qu'il utilise les librairies à jour.
> - Logue chaque étape pour faciliter le debug en cas de problème.

D. Création automatique du template Unraid `my-wolf.xml`

On va créer directement le template Unraid via une commande dans le terminal, pour qu'il apparaisse dans **Docker > Add Container**.

1. Dans le terminal Unraid, lancer :

```
mkdir -p /boot/config/plugins/dockerMan/templates-user

cat << 'EOF' > /boot/config/plugins/dockerMan/templates-user/my-wolf.xml
<?xml version="1.0"?>
<Container version="2">
    <Name>wolf</Name>
    <Repository>ghcr.io/games-on-whales/wolf:stable</Repository>
    <Registry>https://ghcr.io/</Registry>
    <Network>host</Network>
    <MyIP/>
```

```
    <Shell>bash</Shell>
    <Privileged>true</Privileged>
    <Support>https://games-on-whales.github.io/wolf/stable/</Support>
    <Project>https://github.com/games-on-whales/wolf</Project>
    <Overview>Wolf est un serveur de streaming Moonlight permettant de partager un hôte pour plusieurs sessions de jeu et bureaux virtuels.</Overview>
    <Category>GameServers:</Category>
    <WebUI>http://[IP]:8080/</WebUI>
    <TemplateURL/>
    <Icon>https://games-on-whales.github.io/assets/favicon.png</Icon>
    <!-- Volume Docker Nvidia + règle cgroup -->
    <ExtraParams>-v nvidia-driver-vol:/usr/nvidia:rw --device-cgroup-rule="c 13:* rmw"</ExtraParams>
    <PostArgs/>
    <CPUset/>
    <DateInstalled>0</DateInstalled>
    <DonateText/>
    <DonateLink/>
    <Description>Wolf est un serveur de streaming Moonlight permettant de partager un hôte pour plusieurs sessions de jeu et bureaux virtuels.</Description>

    <Networking>
        <Mode>host</Mode>
        <Publish/>
    </Networking>

    <!-- Volumes "bruts" utilisés par le conteneur -->
    <Data>
        <Volume>
            <HostDir>/mnt/user/appdata/wolf</HostDir>
            <ContainerDir>/etc/wolf</ContainerDir>
            <Mode>rw</Mode>
        </Volume>
        <Volume>
            <HostDir>/var/run/docker.sock</HostDir>
            <ContainerDir>/var/run/docker.sock</ContainerDir>
            <Mode>rw</Mode>
        </Volume>
        <Volume>
            <HostDir>/dev</HostDir>
```

```
            <ContainerDir>/dev</ContainerDir>
            <Mode>rw</Mode>
        </Volume>
        <Volume>
            <HostDir>/run/udev</HostDir>
            <ContainerDir>/run/udev</ContainerDir>
            <Mode>rw</Mode>
        </Volume>
    </Data>

    <!-- Variables d'environnement réelles -->
    <Environment>
        <Variable>
            <Value>nvidia-driver-vol</Value>
            <Name>NVIDIA_DRIVER_VOLUME_NAME</Name>
            <Mode/>
        </Variable>
    </Environment>

    <Labels/>

    <!-- Config = ce qui apparaît dans l'UI Unraid -->

    <!-- Appdata Wolf -->
    <Config Name="Wolf Config" Target="/etc/wolf" Default="/mnt/user/appdata/wolf" Mode="rw"
            Description="Dossier de configuration de Wolf"
            Type="Path" Display="always" Required="true"
Mask="false">/mnt/user/appdata/wolf</Config>

    <!-- Docker socket -->
    <Config Name="Docker socket" Target="/var/run/docker.sock" Default="/var/run/docker.sock"
Mode="rw"
            Description="Socket Docker (nécessaire pour lancer les apps GOW)"
            Type="Path" Display="advanced" Required="true"
Mask="false">/var/run/docker.sock</Config>

    <!-- /dev host -->
    <Config Name="Dev" Target="/dev" Default="/dev" Mode="rw"
            Description="Montage de /dev du host"
            Type="Path" Display="advanced" Required="true" Mask="false">/dev</Config>
```

```
    <!-- /run/udev host -->
    <Config Name="Udev" Target="/run/udev" Default="/run/udev" Mode="rw"
            Description="Montage de /run/udev du host"
            Type="Path" Display="advanced" Required="true" Mask="false">/run/udev</Config>

    <!-- Variable : nom du volume Nvidia -->
    <Config Name="NVIDIA_DRIVER_VOLUME_NAME" Target="NVIDIA_DRIVER_VOLUME_NAME"
Default="nvidia-driver-vol" Mode=""
            Description="Nom du volume Docker contenant les bibliothèques Nvidia (méthode
Manual)"
            Type="Variable" Display="advanced" Required="false" Mask="false">nvidia-driver-
vol</Config>

    <!-- Devices GPU / input -->

    <Config Name="Device /dev/dri" Target="/dev/dri" Default="/dev/dri" Mode=""
            Description="DRM / GPU devices"
            Type="Device" Display="advanced" Required="true" Mask="false">/dev/dri</Config>

    <Config Name="Device /dev/nvidiactl" Target="/dev/nvidiactl" Default="/dev/nvidiactl"
Mode=""
            Description="Nvidia control device"
            Type="Device" Display="advanced" Required="true"
Mask="false">/dev/nvidiactl</Config>

    <Config Name="Device /dev/nvidia0" Target="/dev/nvidia0" Default="/dev/nvidia0" Mode=""
            Description="GPU Nvidia principal"
            Type="Device" Display="advanced" Required="true"
Mask="false">/dev/nvidia0</Config>

    <Config Name="Device /dev/nvidia-uvm" Target="/dev/nvidia-uvm" Default="/dev/nvidia-uvm"
Mode=""
            Description="Nvidia UVM"
            Type="Device" Display="advanced" Required="true" Mask="false">/dev/nvidia-
uvm</Config>

    <Config Name="Device /dev/nvidia-uvm-tools" Target="/dev/nvidia-uvm-tools"
Default="/dev/nvidia-uvm-tools" Mode=""
            Description="Nvidia UVM tools"
```

```
            Type="Device" Display="advanced" Required="true" Mask="false">/dev/nvidia-uvm-
tools</Config>

    <Config Name="Device /dev/nvidia-cap1" Target="/dev/nvidia-caps/nvidia-cap1"
Default="/dev/nvidia-caps/nvidia-cap1" Mode=""
            Description="Nvidia capability 1"
            Type="Device" Display="advanced" Required="true" Mask="false">/dev/nvidia-
caps/nvidia-cap1</Config>

    <Config Name="Device /dev/nvidia-cap2" Target="/dev/nvidia-caps/nvidia-cap2"
Default="/dev/nvidia-caps/nvidia-cap2" Mode=""
            Description="Nvidia capability 2"
            Type="Device" Display="advanced" Required="true" Mask="false">/dev/nvidia-
caps/nvidia-cap2</Config>

    <Config Name="Device /dev/uinput" Target="/dev/uinput" Default="/dev/uinput" Mode=""
            Description="uinput (manettes virtuelles, clavier, etc.)"
            Type="Device" Display="advanced" Required="true" Mask="false">/dev/uinput</Config>

    <Config Name="Device /dev/uhid" Target="/dev/uhid" Default="/dev/uhid" Mode=""
            Description="UHID (entrée HID user space)"
            Type="Device" Display="advanced" Required="true" Mask="false">/dev/uhid</Config>

</Container>
EOF
```

2. Aller dans **Docker > Add Container** et choisir le template **wolf** dans la liste déroulante. Normalement, tout est déjà correctement renseigné.
3. Cliquer sur **Apply** pour créer et lancer le conteneur Wolf.

> **Note GPU :** ce template est prévu pour une configuration avec un seul GPU Nvidia : il expose explicitement le device `/dev/nvidia0`.
> Si tu ajoutes un second GPU, il suffira d'ajouter aussi le device `/dev/nvidia1`
> Via l'interface Unraid : Docker → Éditer le conteneur → "Add another device" pour le rendre visible dans le conteneur Wolf.
>
> J'explique plus loin le multi GPU. (en cours de rédaction)

---

3. Configuration de Wolf (config.toml & utilisateurs)

A. Préparation des permissions aux répertoires (méthode recommandée : ACL)

Par défaut, Unraid utilise l'utilisateur `nobody:users` (99:100) pour les partages.
Plutôt que de faire des `chown/chmod -R` à chaque nouvel utilisateur ou nouveau jeu, on peut utiliser les ACL pour donner en plus tous les droits au groupe `1000` (celui utilisé par les containers Wolf/Steam).

Donner les droits **rwx** au groupe 1000 sur l'appdata Wolf

```
setfacl -R -m g:1000:rwx /mnt/user/appdata/wolf
setfacl -R -d -m g:1000:rwx /mnt/user/appdata/wolf
```

Donner les droits **rwx** au groupe 1000 sur le dossier de jeux

```
setfacl -R -m g:1000:rwx /mnt/user/Games
setfacl -R -d -m g:1000:rwx /mnt/user/Games
```

> Certains outils Unraid comme **"New Permissions"** ou des scripts maison peuvent **écraser les ACL** si tu les lances sur ces dossiers → dans ce cas, il faudra relancer les `setfacl`.
>
> Si tu déplaces des données entre disques via certains outils externes (rsync sans `-A`, cp, etc.), les ACL peuvent ne pas être copiées.

Modifier le `config.toml`

Fichier cible (côté host) :

```
nano /mnt/user/appdata/wolf/cfg/config.toml
```

À titre perso, pour le moment, j'ai choisi d'épurer totalement l'affichage dans Moonlight. Mais il est possible de laisser l'app Wolf UI présente, qui permet de classer les apps par utilisateur.

Dans ce fichier, vous supprimez tout ce qu'il y a entre :

```
[[profiles]]
id = 'moonlight-profile-id'
```

et

```
[[profiles]]
id = 'user'
name = 'User'
```

Et vous collez à la place la configuration de votre premier utilisateur suivante :

> Remplacez bien les quatre **<NOM_DU_COMPTE>**.
> Remplacez **<url-image-png>** par une image PNG. Dimension idéale width=112px height=150px (facilement reconnaissable dans Moonlight).

```
# --- Utilisateur 1 : <NOM_DU_COMPTE> ---
    [[profiles.apps]]
    icon_png_path = '<url-image-png>'
    start_virtual_compositor = true
    title = '<NOM_DU_COMPTE>'

        [profiles.apps.runner]
        base_create_json = '''{
  "HostConfig": {
    "IpcMode": "host",
    "CapAdd": ["SYS_ADMIN", "SYS_NICE", "SYS_PTRACE", "NET_RAW", "MKNOD", "NET_ADMIN"],
    "SecurityOpt": ["seccomp=unconfined", "apparmor=unconfined"],
    "Ulimits": [{"Name":"nofile", "Hard":10240, "Soft":10240}],
    "Privileged": false,
    "DeviceCgroupRules": ["c 13:* rmw", "c 244:* rmw"]
  }
}
'''
        devices = []
        env = [
            'PROTON_LOG=1',
            'RUN_SWAY=true',
            'GOW_REQUIRED_DEVICES=/dev/input/* /dev/dri/* /dev/nvidia*',
            'XKB_DEFAULT_LAYOUT=fr',
            'TZ=Europe/Paris'
        ]
        image = 'ghcr.io/games-on-whales/steam:edge'
        mounts = [
            '/mnt/user/Games:/home/retro/games-unraid:rw',
```

```
            '/mnt/user/appdata/wolf/<NOM_DU_COMPTE>:/home/retro/.steam:rw'
        ]
        name = '<NOM_DU_COMPTE>'
        ports = []
        type = 'docker'
```

> **'XKB_DEFAULT_LAYOUT=fr'** → clavier AZERTY français.
> **'TZ=Europe/Paris'** → heure correcte dans l'app.

> **'/mnt/user/Games:/home/retro/games-unraid:rw'** → point de montage du répertoire d'installation des jeux.
> **'/mnt/user/appdata/wolf/<NOM_DU_COMPTE>:/home/retro/.steam:rw'** → rend l'installation de Steam persistante par utilisateur.

3b. Variante des gestions d'utilisateurs

Wolf UI

Wolf UI permet d'avoir un répertoire par utilisateur.

- Ranger toutes les apps par utilisateur.
- Utiliser le mode `co-op` (deux utilisateurs sur la même session en coop locale).

https://www.youtube.com/embed/8an-SvnD4pk?si=IFsfXUJj87F0c-a3

Activation de Wolf UI

Faire apparaître le menu Wolf UI dans Moonlight

Pour activer Wolf UI, il faut modifier `config.toml`, repérer la partie :

```
[[profiles]]
id = 'moonlight-profile-id'
```

Directement dessous, collez :

```
[[profiles.apps]]
    icon_png_path = 'https://raw.githubusercontent.com/games-on-whales/wolf-
ui/refs/heads/main/src/Icons/wolf_ui_icon.png'
```

```
    start_virtual_compositor = true
    title = 'Wolf UI'

        [profiles.apps.runner]
        base_create_json = '''{
      "HostConfig": {
        "IpcMode": "host",
        "CapAdd": ["NET_RAW", "MKNOD", "NET_ADMIN", "SYS_ADMIN", "SYS_NICE"],
        "Privileged": false,
        "DeviceCgroupRules": ["c 13:* rmw", "c 244:* rmw"]
      }
    }'''
        devices = []
        env = [
            'GOW_REQUIRED_DEVICES=/dev/input/event* /dev/dri/* /dev/nvidia*',
            'WOLF_SOCKET_PATH=/var/run/wolf/wolf.sock',
            'WOLF_UI_AUTOUPDATE=False',
            'LOGLEVEL=INFO',
            'XKB_DEFAULT_LAYOUT=fr'
        ]
        image = 'ghcr.io/games-on-whales/wolf-ui:main'
        mounts = [ '/var/run/wolf/wolf.sock:/var/run/wolf/wolf.sock' ]
        name = 'Wolf-UI'
        ports = []
        type = 'docker'
```

Création des espaces utilisateur

De base, dans `config.toml`, un utilisateur `user` est créé avec toutes les apps dedans.

Vous pouvez créer un nouvel espace en ajoutant :

```
[[profiles]]
id = 'user'
name = '<NOM_DU_COMPTE>'
icon_png_path = 'url_de_votre_image.png'
pin = [ 3, 2, 1, 4 ]
```

> L' `id` doit être unique, exemple : `user1`, `user2`, etc...
> Le `pin` est facultatif, il permet de mettre un password de 4 chiffres pour accéder à l'espace de l'utilisateur.

>

> Toutes les apps de cet utilisateur doivent être présentes sous ces lignes.

---

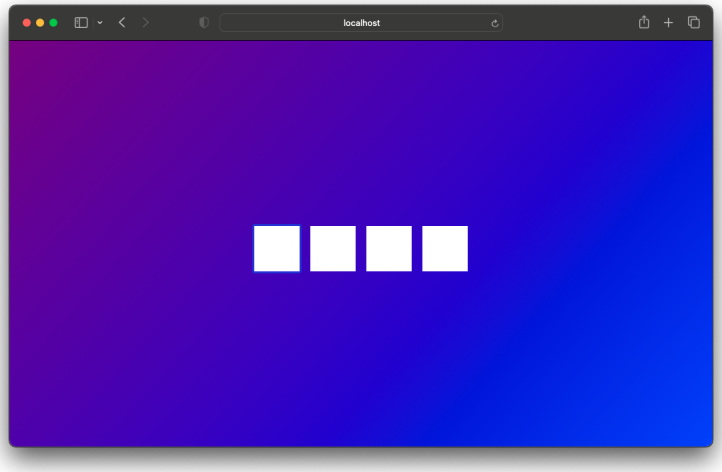
4. Configuration Post-Installation

A. Appairage Moonlight

1. Lancer Moonlight sur le client (PC, TV, Tablette).
2. Noter le code PIN affiché.
3. Consulter les logs Docker : `docker logs -f wolf`
4. Cliquer sur le lien `http://IP_UNRAID:47989/pin/ID` visible dans les logs et entrer le code PIN à 4 chiffres affiché dans Moonlight.

```
10:56:57.579042925 INFO  | Insert pin at http://192.168.1.40:47989/pin/#A264880E15D310F8
10:57:14.297393226 INFO  | Succesfully paired 192.168.1.126
```

5. La page web vous demande de renseigner le PIN pour déverrouiller Moonlight.



À la première ouverture de Steam, le dépôt s'installe, puis il faut vous connecter à votre compte avec Steam Guard.

B. Déclaration des jeux dans Steam

Par défaut, Steam ne connaît pas le disque d'installation des jeux configuré dans `config.toml`. Si vous laissez tel quel, il va installer les jeux dans le répertoire `appdata` (non partagé entre utilisateurs).

1. Fermer Big Picture.
2. Dans Steam, aller dans **Paramètres > Stockage**.
3. Ajouter un lecteur > remonter à `/Home` > sélectionner le dossier `/games-unraid`.
4. Le définir comme lecteur **par défaut** (étoile).

> À faire pour chaque compte Steam.

C. MangoHud

L’overlay Steam ne fonctionne pas dans le conteneur Wolf. Pour voir les FPS et autres stats, on utilise **MangoHud**.

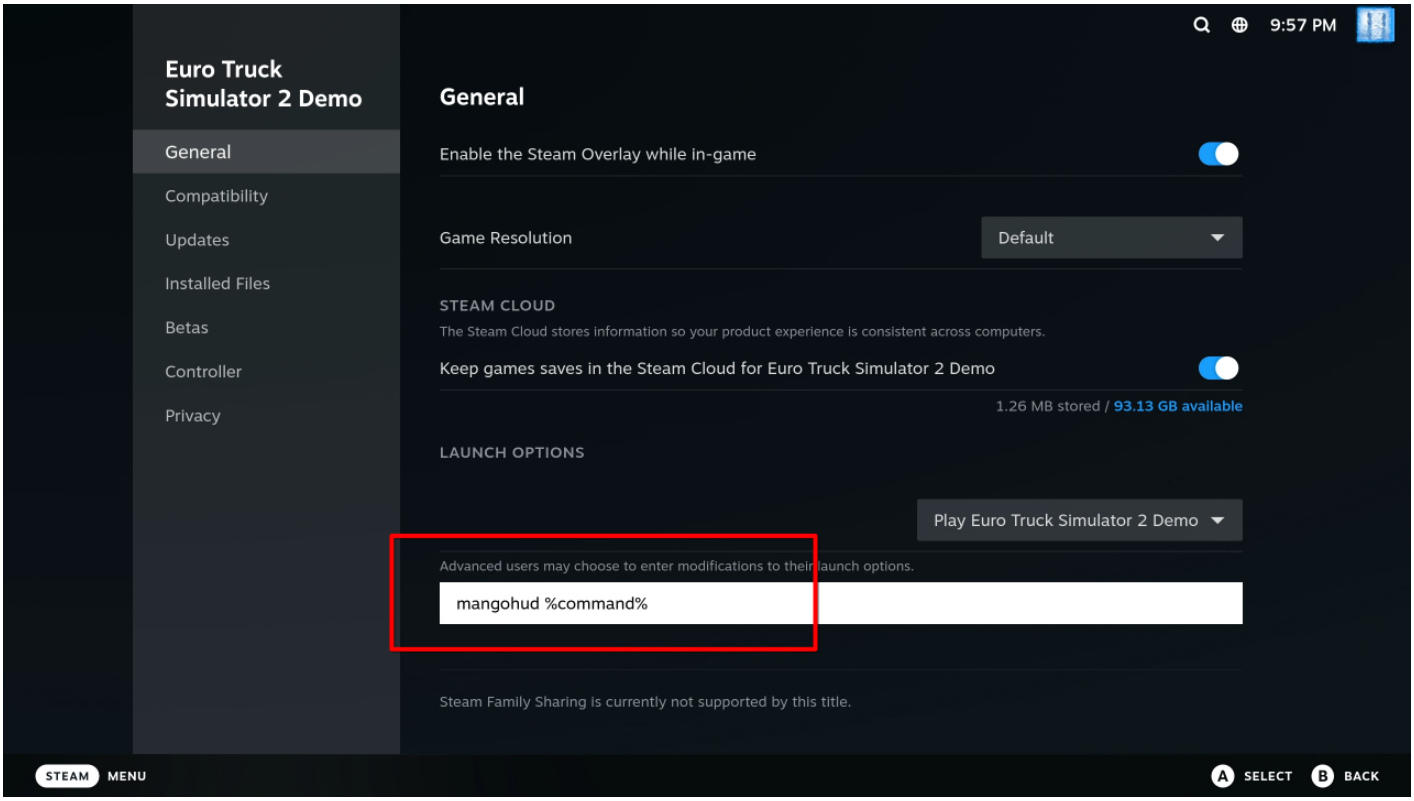
Il est déjà installé et activé par défaut pour tous les jeux Vulkan (y compris les jeux Proton).

MangoHud

Jeux OpenGL (Linux natif)

Pour les jeux OpenGL, vous pouvez l’activer jeu par jeu en ajoutant :

```
mangohud %command%
```



Activation et configuration

En jeu, vous pouvez appuyer sur `Maj droite + F12` pour afficher/masquer MangoHud.
Et sur `Maj droite + F11` pour changer sa position à l’écran.

---

5. Accès à Distance (Ouverture de Ports)

Wolf ne gère pas l’UPnP. Pour jouer depuis l’extérieur, deux options :

Option A : VPN (Recommandé)

Utiliser **Tailscale** ou **Wireguard** (plugins Unraid). C’est plus sécurisé et ne nécessite aucune ouverture de port. (Plus chiant côté FireTV ou autre, mais c’est la vie □□)

Option B : Ouverture de Ports (NAT)

Sans VPN, rediriger les ports suivants sur votre routeur vers l’IP du serveur Unraid :

- **TCP 47984** : HTTPS (sécurité)
- **TCP 47989** : HTTP (appairage Web)
- **TCP 48010** : RTSP (initialisation du stream)
- **UDP 47999** : Contrôle (manettes)
- **UDP 48100** : Flux vidéo (RTP - spécifique Wolf)
- **UDP 48200** : Flux audio (RTP - spécifique Wolf)

Revision #44
Created 2025-11-27 13:29:16 UTC by thymon
Updated 2026-01-05 08:24:46 UTC by thymon