Eden pour RomM

RomM — Emulator Streaming avec Eden (Nintendo Switch) sur Unraid (GPU NVIDIA)

Ce tutoriel explique comment ajouter un conteneur **Eden** (émulateur Nintendo Switch, fork de Yuzu) à une instance **RomM 5.1.0+** pour lancer des jeux Switch depuis RomM et y jouer dans le navigateur, avec rendu GPU NVIDIA.

Le principe est identique à l'intégration Dolphin : un *Docker Mod* injecte un **broker HTTP** (port 8000 du conteneur) que RomM contacte, et le flux est diffusé par **Selkies** en H.264 NVENC.

> ✅ Testé et fonctionnel avec **RomM 5.1.0 stable** (la branche de développement n'est pas nécessaire), Unraid 7, Ryzen 9 3900X + RTX 3060, `lscr.io/linuxserver/eden`. Résultat : **Mario Kart 8 Deluxe à 60 FPS stables** en 1080p dans le navigateur.

> ⚠️ Ce guide suppose que la page « **Emulator Streaming avec Dolphin** » a déjà été suivie : les prérequis hôte sont les mêmes et ne sont pas re-détaillés ici — plugin Nvidia Driver (branche Production, ≥ 580), paramètres noyau `nvidia-drm.modeset=1 nvidia_drm.fbdev=1`, et surtout les **deux correctifs NVIDIA (§2 de la page Dolphin)** : composant **egl-x11** et nœud `/dev/nvidia-modeset`, indispensables ici aussi.

>

> **Ce guide décrit une installation en réseau local.** Pour jouer depuis l'extérieur, pour l'isolation réseau des conteneurs, et pour le **portail nginx du mod Eden qui bloque le streaming avec RomM 5.1.0**, voir la page « **Accès distant, publication HTTPS et isolation réseau** ». Elle contient aussi l'avertissement de sécurité à lire avant toute publication.

---

1. Particularités d'Eden à connaître avant de commencer

1. **Contenu requis de votre propre console.** Eden ne lance aucun jeu commercial sans les `prod.keys` (clés de déchiffrement) et le **firmware** de la Switch — à extraire de votre propre console.
2. **Eden est Vulkan-first**, et son backend Vulkan a besoin du nœud `/dev/nvidia-modeset` (§2.2 de la page Dolphin). Sans lui : liste de GPU **vide** dans les menus, et au lancement d'un jeu l'erreur *« Eden has encountered an error while running the video core »* (log : `Invalid device index -1`, `VK_ERROR_INITIALIZATION_FAILED`).
3. **Les réglages faits dans l'interface ne sont jamais sauvegardés** quand Eden est lancé par le broker : celui-ci tue le processus avant qu'Eden n'écrive sa configuration. Tout réglage durable se fait **dans le fichier, conteneur arrêté** (§5).
4. **Deux réglages par défaut coûtent très cher en performances** : la précision GPU sur *High* et l'absence d'épinglage CPU sur les Ryzen multi-CCD (§5 et §6). Sans eux, comptez ~30 FPS au lieu de 60.

---

2. Conteneur Eden

Image : `lscr.io/linuxserver/eden`. Configuration identique à Dolphin, à trois différences près : le mod, les ports hôte (les deux conteneurs coexistent), et l'appdata dédié.

2.1 Ports (host:container)

Les services écoutent toujours sur **3000/3001/8000 côté conteneur** — seuls les ports hôte changent :

Port hôte (exemple)	Port conteneur	Rôle

3010	3000	Selkies HTTP
3011	3001	**Selkies HTTPS** (flux de jeu)
8001	8000	**Broker RomM**

2.2 Variables d'environnement

Variable	Valeur
`DOCKER_MODS`	`ghcr.io/loneangelfayt/eden-romm-integration-mod:latest`
`ROM_ROOT`	`/romm/library`
`BROKER_SECRET`	**le même secret que vos autres conteneurs émulateurs** — RomM n'a qu'un `STREAMING_BROKER_SECRET` global
`NVIDIA_VISIBLE_DEVICES`	UUID du GPU (ou `all`)
`NVIDIA_DRIVER_CAPABILITIES`	`all`
`DRINODE` / `DRI_NODE`	`/dev/dri/renderD128` (adapter)
`SELKIES_MANUAL_WIDTH` / `HEIGHT`	`1920` / `1080`
`__EGL_VENDOR_LIBRARY_FILENAMES`	`/usr/share/glvnd/egl_vendor.d/10_nvidia.json`
`PUID` / `PGID`	`99` / `100`

2.3 Extra Parameters

```
--gpus all --runtime nvidia --shm-size=1gb --device /dev/nvidia-modeset --cpuset-cpus=0-5,12-17
```

L'épinglage `--cpuset-cpus` est **obligatoire sur tout Ryzen multi-CCD** : sans lui, comptez environ un tiers de performances en moins (§6). Adaptez impérativement la liste à votre processeur — la valeur ci-dessus correspond au premier CCD d'un 3900X et serait **contre-productive** sur une autre topologie. Sur un CPU mono-CCD ou Intel, retirez l'option.

2.4 Volumes

Host Path	Container Path	Mode
(le même dossier de ROMs que RomM)	`/romm/library`	**ro**
(appdata dédié, ex. `/mnt/user/appdata/eden`)	`/config`	rw

Host Path	Container Path	Mode
+ les 4 montages du correctif egl-x11 (identiques à la page Dolphin : 2 libs + 2 JSON)		**ro**

> Rappel : le dossier de bibliothèque doit être monté **au même chemin conteneur** (`/romm/library`) dans RomM et dans Eden.

2.5 Vérification du broker

```
docker logs eden 2>&1 | grep -iE "broker|mod" | head -10
curl -s http://localhost:8001/health        # → {"status": "ok"}
```

---

3. Premier démarrage d'Eden

Ouvrir `https://IP_DU_SERVEUR:3011`, accepter le certificat auto-signé. Au premier lancement, Eden propose de forcer **X11** au lieu de Wayland : **accepter** (c'est le chemin réparé par le correctif egl-x11, et celui qu'utilise le broker).

---

4. Clés, firmware, jeux et mises à jour

Ordre important : les clés d'abord, le firmware ensuite (Eden a besoin des clés pour installer le firmware).

1. **`prod.keys`** : fichier décompressé, à poser dans l'appdata : `<appdata>/.local/share/eden/keys/prod.keys`. Redémarrer Eden ensuite.
2. **Firmware** : garder le ZIP tel quel, le poser dans l'appdata (visible sous `/config/` dans le conteneur), puis dans Eden : **Tools → Install Firmware** en pointant le ZIP.
3. **Jeux** : fichiers directs `.nsp` ou `.xci` (pas d'archives), dans le dossier de plateforme **`switch`** de la bibliothèque (fs_slug RomM : `switch`).
4. **Mises à jour et DLC** : Eden n'a **pas** de « dossier de patchs » — ils s'installent une fois pour toutes dans la NAND virtuelle : **File → Install Files to NAND**, sélectionner les `.nsp` de mise à jour. Vérification : clic droit sur le jeu → Properties → la version doit être celle de

la mise à jour. Tout persiste dans l'appdata (`.local/share/eden/nand/`) et profite à tous les joueurs. Ne placez **pas** les mises à jour dans la bibliothèque RomM (elles seraient scannées comme une fausse plateforme).

---

5. ⚠ Réglages Eden (à faire dans le fichier, conteneur arrêté)

L'interface ne sauvegarde rien quand Eden est lancé par le broker (§1.3). Éditez donc directement `<appdata>/.config/eden/qt-config.ini`, **conteneur arrêté**.

Chez Eden, chaque clé est accompagnée d'une ligne `clé\default=` : tant qu'elle vaut `true`, la valeur est ignorée au profit du défaut compilé. **Il faut donc toujours passer `\default` à `false`.**

> ⚠ **Après toute édition depuis l'hôte** (`sed -i`, `cat >`, redirection…), rétablissez le propriétaire : `chown -R 99:100 <appdata>`. Ces commandes recréent le fichier et peuvent le laisser en `root:root` ; Eden ne peut alors plus enregistrer sa configuration, ce qui ne se voit que par un `Config file could not be saved!` dans son log.

5.1 Les deux réglages qui comptent

```
docker stop eden
F=<appdata>/.config/eden/qt-config.ini

# Backend graphique : Vulkan
sed -i 's/^backend\\default=.*/backend\\default=false/; s/^backend=.*/backend=1/' "$F"

# Précision GPU : Normal (le défaut compilé est High = beaucoup plus lent)
sed -i 's/^gpu_accuracy\\default=.*/gpu_accuracy\\default=false/;
s/^gpu_accuracy=.*/gpu_accuracy=0/' "$F"

docker start eden
grep -nE "^backend|^gpu_accuracy" "$F"
```

`gpu_accuracy` **est le réglage le plus rentable de tout ce guide** : le défaut *High* force des chemins d'émulation beaucoup plus lents, pour une précision dont la quasi-totalité des jeux n'a pas besoin. Mesuré ici : **+33 % de FPS** en passant sur *Normal*.

5.2 Énumérations utiles

Depuis janvier 2026, Eden a fusionné le backend de rendu et le backend de shaders dans une seule clé `backend` :

`backend`	**Valeur**
0	OpenGL (GLSL)
1	**Vulkan** (recommandé)
2	Null
3	OpenGL GLASM
4	OpenGL SPIR-V

> La clé `shader_backend` est **obsolète** : si elle traîne dans votre fichier, elle est ignorée en silence. Supprimez-la.

`gpu_accuracy`	**Valeur**
0	**Normal** (recommandé)
1	High (défaut compilé)
2	Extreme

5.3 Réglages vérifiés, à laisser tels quels

`use_asynchronous_gpu_emulation=true` (indispensable), `use_multi_core=true`, `use_disk_shader_cache=true`, `speed_limit=100`.

`use_asynchronous_shaders` : laisser à **false** — testé ici, l'activer dégrade légèrement les performances.

5.4 Langue française et plein écran

Là encore, deux notions distinctes : la langue de **la console émulée** (celle que les jeux utilisent) et celle de **l'interface** d'Eden.

```
docker stop eden
F=<appdata>/.config/eden/qt-config.ini

# Langue de la console émulée (jeux) : 2 = français
sed -i 's/^language_index\\default=.*/language_index\\default=false/;
s/^language_index=.*/language_index=2/' "$F"

# Plein écran persistant
sed -i 's/^fullscreen\\default=.*/fullscreen\\default=false/;
s/^fullscreen=.*/fullscreen=true/' "$F"

docker start eden
grep -nE "^language_index|^fullscreen" "$F"
```

Valeurs de `language_index` : `0` japonais, `1` anglais US, `2` **français**, `3` allemand, `4` italien, `5` espagnol, `12` anglais UK, `13` français canadien.

Interface d'Eden en français : la clé `language` de la section `[UI]` stocke un code de locale dont le format varie selon les builds. Le plus fiable est de la faire écrire par Eden lui-même :

1. Lancer Eden **manuellement** (pas via RomM) depuis le bureau Selkies
2. Emulation → Configure → General → Interface language → *Français*
3. **Fermer Eden proprement** (File → Exit) — c'est ce qui déclenche l'écriture de la configuration
4. Vérifier : `grep -n "^language" <appdata>/.config/eden/qt-config.ini`

Si la ligne `language\default=true` subsiste, la passer à `false` conteneur arrêté.

> Rappel : le plein écran est de toute façon déclenché par le broker (il envoie F11 trois secondes après le lancement). Le réglage ci-dessus le rend cohérent aussi pour les lancements manuels. Pour en sortir ponctuellement — par exemple pour lire le compteur de FPS — appuyer sur **Échap**.

---

6. ⚠ Épinglage CPU — obligatoire sur Ryzen multi-CCD

Voir la section correspondante de la page Dolphin pour l'explication détaillée. En résumé : sur un Ryzen à plusieurs CCD, les threads d'Eden migrant d'un groupe de cache à l'autre coûtent très cher, sans que rien ne paraisse saturer.

Ce réglage n'est pas optionnel : il représente ici la moitié du chemin entre un jeu injouable et 60 FPS stables.

```
lscpu -e=CPU,CORE,L3      # identifier les groupes de cache
```

Puis épingler le conteneur sur un seul CCD via `--cpuset-cpus` (Extra Parameters) ou l'onglet **CPU Pinning** du template.

> Mesuré sur un 3900X avec Mario Kart 8 Deluxe : **40 → 60 FPS** grâce à ce seul réglage. C'est aussi ce qui explique qu'une VM avec vCPU épinglés paraisse bien plus rapide que le même émulateur en conteneur non épinglé.

Récapitulatif des gains mesurés (Mario Kart 8 Deluxe, 1080p) :

Étape	FPS
Configuration initiale	~30 (voire écran noir sans les correctifs §2 de la page Dolphin)
+ `gpu_accuracy=0`	~40
+ épinglage CCD	**60 stables**

---

7. Côté RomM

Ajouter l'entrée dans le bloc `streaming` du `config.yml` (adapter IP et ports) :

```
    - platform: switch
      host: https://IP_DU_SERVEUR:3011
      broker_host: http://IP_DU_SERVEUR:8001
      label: Eden
```

Redémarrer RomM, rescanner la plateforme Switch.

---

8. Test final

1. Fermer proprement toute instance d'Eden ouverte dans le bureau Selkies (le broker lance la sienne).
2. Fiche d'un jeu Switch dans RomM → bouton **Play on Eden**.
3. Le jeu doit s'afficher, avec le son.

Vérifier l'accélération GPU pendant qu'un jeu tourne :

```
docker exec -u abc eden sh -c 'for p in /proc/[0-9]*; do ls -l $p/fd 2>/dev/null | grep -q nvidia && echo "$(basename $p) $(tr "\0" " " < $p/cmdline | cut -c1-60)"; done'
```

Le processus `eden` doit apparaître dans la liste.

> **Astuce mesure :** le broker met Eden en plein écran (il envoie F11 après 3 s), ce qui masque la barre d'état. Appuyez sur **Échap** pour en sortir et lire le FPS et le pourcentage de vitesse.

---

9. Dépannage rapide

Symptôme	Cause	Solution
« error while running the video core », log : `Invalid device index -1`	Nœud `/dev/nvidia-modeset` absent → Vulkan sans présentation	§2.2 de la page Dolphin
Liste de GPU vide dans les menus Graphics	Idem	§2.2 de la page Dolphin
Réglages qui « ne tiennent pas »	Config jamais écrite (processus tué par le broker)	Éditer `qt-config.ini` conteneur arrêté (§5)
Réglage écrit mais sans effet	Ligne `clé\default=true` restée en place	Passer `\default` à `false` (§5)
~30 FPS au lieu de 60, sans que rien ne sature	`gpu_accuracy` sur High et/ou pas d'épinglage CPU	§5.1 et §6
Erreur de clés à l'installation du firmware ou d'un jeu	`prod.keys` absent ou mal placé	§4, puis redémarrer Eden
Jeu à la version 1.0.0 malgré la mise à jour	Mise à jour non installée en NAND	§4 — File → Install Files to NAND
Jeu très lent, `eden` absent du test des fd	Correctif egl-x11 manquant	§2.1 de la page Dolphin

Symptôme	Cause	Solution
Écran noir après une mise à jour du driver NVIDIA	Version des libs egl-x11 changée sur l'hôte	Mettre à jour les Host Path des montages
403 Forbidden nginx en ouvrant l'URL du flux	Portail du mod : normal sans session RomM ouverte	Page « Accès distant », §5
Écran noir depuis RomM, alors que le lancement réussit côté serveur	Portail du mod : RomM 5.1.0 n'envoie pas le `stream_token`	Page « Accès distant », §5
Le problème revient après chaque modification de template	Le conteneur est recréé, le portail est réinstallé	Script `custom-cont-init.d` — page « Accès distant », §5

Revision #3
Created 2026-07-31 11:30:36 UTC by thymon
Updated 2026-08-16 23:13:45 UTC by thymon