systemd-nspawn

Table of Contents

systemd-nspawn est comme la commande **chroot**, il peut être utilisé pour exécuter une commande ou un système d'exploitation dans un conteneur d'espace léger. Il est plus puissant que **chroot** car il virtualise entièrement la hiérarchie du système de fichiers, ainsi que l'arborescence des processus, les différents sous-systèmes IPC et l'hôte et le nom de domaine.

systemd-nspawn limite l'accès à diverses interfaces du noyau dans le conteneur en lecture seule, telles que `/sys`, `/proc/sys` ou `/sys/fs/selinux`. Les interfaces réseau et l'horloge système ne peuvent pas être modifiées depuis le conteneur. Les nœuds de périphérique ne peuvent pas être

créés. Le système hôte ne peut pas être redémarré et les modules du noyau ne peuvent pas être chargés depuis le conteneur.

systemd-nspawn est un outil plus simple à configurer que **LXC** ou **Libvirt**.

Installation

systemd-nspawn fait partie et est fourni avec **systemd**.

Création et démarrage d'un conteneur

Créer et démarrer un conteneur Arch Linux minimal

Installer d'abord **arch-install-scripts**:

```
sudo apt install arch-install-scripts
```

Ensuite, créer un répertoire pour contenir le conteneur. Dans cet exemple, on utilisera `~/MyContainer`:

```
sudo mkdir ~/MyContainer
```

Ensuite, utiliser **pacstrap** pour installer un système Arch de base dans le conteneur. Au minimum, il faut installer le package de base:

```
pacstrap -c ~/MyContainer base [paquets/groupes supplémentaires]
```

Le package de base ne dépend pas du package du noyau Linux et est prêt pour le conteneur.

Une fois l'installation terminée, se chrooter dans le conteneur et définir un mot de passe root :

```
systemd-nspawn -D ~/MonContainer
passwd
logout
```

Enfin, démarrer dans le conteneur :

```
systemd-nspawn -b -D ~/MonContainer
```

L'option **-b** démarrera le conteneur (c'est-à-dire qu'il exécutera systemd en tant que PID=1), au lieu d'exécuter simplement un shell, et **-D** spécifie le répertoire qui devient le répertoire racine du conteneur.

Une fois le conteneur démarré, on peut se connecter en tant que “root” avec le mot de passe.

Si la connexion échoue avec “Connexion incorrecte”, le problème est probablement lié à la liste blanche des périphériques securetty TTY

Le conteneur peut être mis hors tension en exécutant **poweroff** depuis l'intérieur du conteneur. Depuis l'hôte, les conteneurs peuvent être contrôlés par l'outil **machinectl**.

Pour mettre fin à la session depuis le conteneur, maintenir la touche Ctrl enfoncée et appuyer rapidement trois fois sur]. Les utilisateurs de clavier non américains doivent utiliser % au lieu de].

Créer un environnement Debian ou Ubuntu

Installer **debootstrap** et **debian-archive-keyring** ou **ubuntu-keyring** selon la distribution souhaitée.

systemd-nspawn nécessite que le système d'exploitation du conteneur utilise systemd init (il s'exécute en tant que PID 1) et que systemd-nspawn soit installé dans le conteneur. Il faut s'assurer que le package systemd-container soit installé sur le système de conteneur.

A partir de là, il est plutôt facile de mettre en place des environnements Debian ou Ubuntu :

```
cd /var/lib/machines
debootstrap --include=systemd-container --components=main,universe nom-de-code container-name repository-url
```

Pour Debian, les noms de code valides sont soit les noms roulants comme “stable” et “testing” ou les noms de version comme “stretch” et “sid”, pour Ubuntu, le nom de code comme “xenial” ou “zesty” doit être utilisé. Une liste complète des noms de code se trouve dans /usr/share/debootstrap/scripts. Dans le cas d'une image Debian, “l'url du référentiel” peut être https://deb.debian.org/debian/. Pour une image Ubuntu, le “repository-url” peut être http://archive.ubuntu.com/ubuntu/. “référentiel-url” ne doit pas contenir de barre oblique à la fin.

Par exemple pour télécharger un système Debian Arm64 de base pour Raspberry Pi:

```
sudo debootstrap --arch arm64 buster /var/lib/machines
http://ftp.uk.debian.org/debian
```

Tout comme Arch, Debian et Ubuntu ne laisseront pas se connecter sans mot de passe. Pour définir le mot de passe root, exécuter **systemd-nspawn** sans l'option **-b**:

```
cd /var/lib/machines
systemd-nspawn -D ./nom-du-conteneur
passwd
logout
```

Gestion

Les conteneurs situés dans `/var/lib/machines/` peuvent être contrôlés par la commande machinectl, qui contrôle en interne les instances de l'unité **systemd-nspawn@.service**. Les sous-répertoires dans `/var/lib/machines/` correspondent aux noms des conteneurs, c'est-à-dire `/var/lib/machines/container-name/`.

Si le conteneur ne peut pas être placé dans `/var/lib/machines/` pour une raison quelconque, il peut être lié symboliquement.

Options systemd-nspawn par défaut

Les conteneurs démarrés via **machinectl** ou **systemd-nspawn@.service** utilisent des options par défaut différentes de celles des conteneurs démarrés manuellement par la commande **systemd-nspawn**. Les options supplémentaires utilisées par le service sont :

- **-b/–boot**: Les conteneurs gérés recherchent automatiquement un programme init et l'invoquent en tant que PID 1.
- **–network-veth**: (implique **–private-network**)Les conteneurs gérés obtiennent une interface réseau virtuelle et sont déconnectés du réseau hôte.
- **-U**: Les conteneurs gérés utilisent la fonctionnalité user_namespaces(7) par défaut si elle est prise en charge par le noyau. Voir Conteneurs #Unprivileged pour les implications.
- **–link-journal=try-guest**

Le comportement peut être remplacé dans les fichiers de configuration par conteneur,

machinectl

L'outil **machinectl** nécessite que **systemd** et **dbus** soient installés dans le conteneur.

Les conteneurs peuvent être gérés par la commande **machinectl subcommand container-name**. Par exemple, pour démarrer un conteneur :

```
machinectl start container-name
```

De même, il existe des sous-commandes telles que **poweroff**, **reboot**, **status** et **show**.

Les opérations de mise hors tension et de redémarrage peuvent être effectuées depuis le conteneur à l'aide des commandes **poweroff** et **reboot**.

Les autres commandes courantes sont :

- **machinectl list**: affiche une liste des conteneurs en cours d'exécution
- **machinectl login container-name**: ouvre une session de connexion interactive dans un conteneur
- **machinectl shell [username@]container-name**: ouvre une session shell interactive dans un conteneur (cela appelle immédiatement un processus utilisateur sans passer par le processus de connexion dans le conteneur)
- **machinectl enable container-name** et **machinectl disable container-name**: activer ou désactiver un conteneur pour qu'il démarre au démarrage,

machinectl a également des sous-commandes pour gérer les images de conteneur (ou de machine virtuelle) et les transferts d'images.

Une grande partie de la chaîne d'outils de base de **systemd** a été mise à jour pour fonctionner avec des conteneurs. Les outils qui fournissent généralement une option **-M**, **–machine=** qui prendra un nom de conteneur comme argument.

Voir les journaux de journal pour une machine particulière	`journalctl -M container-name`
Afficher le contenu du groupe de contrôle	`systemd-cgls -M container-name`
Voir l'heure de démarrage du conteneur	`systemd-analyze -M container-name`

Configuration

Paramètres par conteneur

Pour spécifier des paramètres par conteneur et non des remplacements globaux, les fichiers `.nspawn` peuvent être utilisés:

- Les fichiers `.nspawn` peuvent être supprimés de manière inattendue de `/etc/systemd/nspawn/` lorsqu'on exécute `machinectl remove`.
- L'interaction des options réseau spécifiées dans le fichier `.nspawn` et sur la ligne de commande ne fonctionne pas correctement lorsqu'il y a **–settings=override** (qui est spécifié dans le fichier `systemd-nspawn@.service`). Comme solution de contournement, il faut inclure l'option **VirtualEthernet=on**, même si le service spécifie **–network-veth**.

Activer le démarrage du conteneur au démarrage

Lorsqu'on utilise fréquemment un conteneur, on peut le démarrer au démarrage. Il faut d'abord s'assurer que **machines.target** est activé.

Les conteneurs découvrables par **machinectl** peuvent être activés ou désactivés :

```
machinectl enable container-name
```

- Cela a pour effet d'activer l'unité systemd **systemd-nspawn@container-name.service**.
- Les conteneurs démarrés par **machinectl** obtiennent une interface Ethernet virtuelle.

Contrôle des ressources

Pour un aperçu de l'utilisation des ressources :

```
systemd-cgtop
```

On peut tirer parti des groupes de contrôle pour implémenter les limites et la gestion des ressources des conteneurs avec `systemctl set-property`. Par exemple, on peut souhaiter limiter la quantité de mémoire ou l'utilisation du processeur. Pour limiter la consommation mémoire duconteneur à 2 Gio :

```
systemctl set-property systemd-nspawn@container-name.service MemoryMax=2G
```

Ou pour limiter l'utilisation du temps CPU à environ l'equivalent de 2 cœurs :

```
systemctl set-property systemd-nspawn@container-name.service CPUQuota=200%
```

Cela créera des fichiers permanents dans `/etc/systemd/system.control/systemd-nspawn@container-name.service.d/`.

Selon la documentation, **MemoryHigh** est la méthode préférée pour contrôler la consommation de mémoire, mais elle ne sera pas strictement limitée comme c'est le cas avec **MemoryMax**. On peut utiliser les deux options en laissant **MemoryMax** comme dernière ligne de défense. Il faut également tenir compte du fait qu'on ne limitera pas le nombre de processeurs que le conteneur peut voir, mais on obtiendra des résultats similaires en limitant le temps que le conteneur obtiendra au maximum, par rapport au temps CPU total.

Si on souhaite que ces modifications ne soient que temporaires, On peut passer l'option

–runtime. On peut vérifier leurs résultats avec `systemd-cgtop`.

La mise en réseau

Les conteneurs systemd-nspawn peuvent utiliser soit la mise en réseau hôte, soit la mise en réseau privée :

- En mode réseau hôte, le conteneur dispose d'un accès complet au réseau hôte. Cela signifie que le conteneur pourra accéder à tous les services réseau sur l'hôte et que les paquets provenant du conteneur apparaîtront sur le réseau extérieur comme provenant de l'hôte (c'est-à-dire partageant la même adresse IP).
- En mode réseau privé, le conteneur est déconnecté du réseau de l'hôte. Cela rend toutes les interfaces réseau indisponibles pour le conteneur, à l'exception du périphérique de bouclage et de ceux explicitement affectés au conteneur. Il existe plusieurs façons de configurer des interfaces réseau pour le conteneur :
  - une interface existante peut être affectée au conteneur (par exemple, si vous aver plusieurs périphériques Ethernet),
  - une interface réseau virtuelle associée à une interface existante (c'est-à-dire une interface VLAN) peut être créée et affectée au conteneur,
  - une liaison Ethernet virtuelle entre l'hôte et le conteneur peut être créée.

Dans ce dernier cas, le réseau du conteneur est entièrement isolé (du réseau extérieur ainsi que des autres conteneurs) et il appartient à l'administrateur de configurer la mise en réseau entre l'hôte et les conteneurs. Cela implique généralement la création d'un pont réseau pour connecter plusieurs interfaces (physiques ou virtuelles) ou la configuration d'une traduction d'adresses réseau entre plusieurs interfaces.

Le mode de mise en réseau hôte convient aux conteneurs d'applications qui n'exécutent aucun logiciel de mise en réseau qui configurerait l'interface affectée au conteneur. La mise en réseau hôte est le mode par défaut lorsque vous exécuter systemd-nspawn à partir du shell.

D'autre part, le mode réseau privé convient aux conteneurs système qui doivent être isolés du système hôte. La création de liens Ethernet virtuels est un outil très flexible permettant de créer des réseaux virtuels complexes. Il s'agit du mode par défaut pour les conteneurs démarrés par machinectl ou systemd-nspawn@.service.

Les sous-sections suivantes décrivent des scénarios courants. Voir systemd-nspawn(1) §Options réseau pour plus de détails sur les options systemd-nspawn disponibles. Utiliser le réseau hôte

Pour désactiver le réseau privé et la création d'un lien Ethernet virtuel utilisé par les conteneurs démarrés avec machinectl, ajouter un fichier `/etc/systemd/nspawn/container-name.nspawn` avec l'option suivante :

```
[Network]
VirtualEthernet=no
```

Cela remplacera l'option **-n/–network-veth** utilisée dans **systemd-nspawn@.service** et les conteneurs nouvellement démarrés utiliseront le mode réseau hôte.

Utiliser une liaison Ethernet virtuelle

Si un conteneur est démarré avec l'option **-n/–network-veth**, **systemd-nspawn** créera un lien Ethernet virtuel entre l'hôte et le conteneur. Le côté hôte du lien sera disponible en tant qu'interface réseau nommée **ve-container-name**. Le côté conteneur du lien sera nommé **host0** (cette option implique **–private-network**).

Si le nom du conteneur est trop long, le nom de l'interface sera raccourci (par exemple, **ve-long-conKQGh** au lieu de **ve-long-container-name**) pour tenir dans la limite de **15** caractères. Le nom complet sera défini comme la propriété altname de l'interface et pourra toujours être utilisé pour référencer l'interface.

Lors de l'examen des interfaces avec ip link, les noms d'interface seront affichés avec un suffixe, tel que **ve-container-name@if2** et **host0@if9**. Le **@ifN** ne fait pas partie du nom de l'interface ; à la place, ip link ajoute ces informations pour indiquer à quel « emplacement » le câble Ethernet virtuel se connecte à l'autre extrémité.

Par exemple, une interface Ethernet virtuelle hôte affichée sous la forme **ve-foo@if2** est connectée au conteneur **foo**, et à l'intérieur du conteneur à la deuxième interface réseau - celle affichée avec l'index **2** lors de l'exécution de ip link à l'intérieur du conteneur. De même, l'interface nommée **host0@if9** dans le conteneur est connectée à la **9ème** interface réseau sur l'hôte.

Lorsqu'on démarre le conteneur, une adresse IP doit être attribuée aux deux interfaces (sur l'hôte et dans le conteneur). Si on utilise **systemd-networkd** sur l'hôte ainsi que dans le conteneur, cette opération est prête à l'emploi :

- le fichier `/usr/lib/systemd/network/80-container-ve.network` sur l'hôte correspond à l'interface **ve-container-name** et démarre un serveur DHCP, qui attribue des adresses IP à l'interface hôte ainsi qu'au conteneur,
- le fichier `/usr/lib/systemd/network/80-container-host0.network` dans le conteneur correspond à l'interface **host0** et démarre un client DHCP, qui reçoit une adresse IP de l'hôte.

Si on n'utilise pas **systemd-networkd**, On peut configurer des adresses IP statiques ou démarrer un serveur DHCP sur l'interface hôte et un client DHCP dans le conteneur.

Pour donner au conteneur l'accès au réseau extérieur, on peut configurer NAT. Dans **systemd-networkd**, cela se fait (partiellement) automatiquement via l'option **IPMasquerade=both** dans `/usr/lib/systemd/network/80-container-ve.network`. Cependant, cela n'émet qu'une seule règle iptables (ou nftables) telle que

```
-t nat -A POSTROUTING -s 192.168.163.192/28 -j MASQUERADE
```

La table de filtrage doit être configurée manuellement. On peut utiliser un caractère générique pour faire correspondre toutes les interfaces commençant par **ve-**:

```
iptables -A FORWARD -i ve-+ -o internet0 -j ACCEPT
```

systemd-networkd et systemd-nspawn peuvent s'interfacer avec iptables (en utilisant la bibliothèque libiptc) ainsi qu'avec nftables [4][5]. Dans les deux cas, IPv4 et IPv6 NAT sont pris en charge.

De plus, on doit ouvrir le port UDP 67 sur les interfaces ve-+ pour les connexions entrantes vers le serveur DHCP (exploité par systemd-networkd) :

```
iptables -A ENTRÉE -i ve-+ -p udp -m udp --dport 67 -j ACCEPT
```

Utiliser un pont réseau

Si on a configuré un pont réseau sur le système hôte, On peut créer une liaison Ethernet virtuelle pour le conteneur et ajouter son côté hôte au pont réseau. Cela se fait avec l'option `--network-bridge=bridge-name`, cela implique `--network-veth`, c'est-à-dire que la liaison Ethernet virtuelle est créée automatiquement. Cependant, le côté hôte du lien utilisera le préfixe **vb-** au lieu de **ve-**, donc les options **systemd-networkd** pour démarrer le serveur DHCP et le masquage IP ne seront pas appliquées.

La gestion du pont est laissée à l'administrateur. Par exemple, le pont peut connecter des interfaces virtuelles à une interface physique, ou il peut connecter uniquement des interfaces virtuelles de plusieurs conteneurs.

Il existe également une option **–network-zone=zone-name** qui est similaire à **–network-bridge** mais le pont réseau est géré automatiquement par **systemd-nspawn** et **systemd-networkd**. L'interface de pont nommée **vz-zone-name** est automatiquement créée lorsque le premier conteneur configuré avec **–network-zone=zone-name** est démarré, et est automatiquement supprimé lorsque le dernier conteneur configuré avec **–network-zone=zone-name** se termine . Par conséquent, cette option facilite le placement de plusieurs conteneurs associés sur un réseau virtuel commun. Les interfaces **vz-*** sont gérées par **systemd-networkd** de la même manière que les interfaces **ve-*** en utilisant les options du fichier `/usr/lib/systemd/network/80-container-vz.network`.

Utiliser une interface "macvlan" ou "ipvlan"

Au lieu de créer une liaison Ethernet virtuelle (dont le côté hôte peut ou non être ajouté à un pont), On peut créer une interface virtuelle sur une interface physique existante (c'est-à-dire une interface VLAN) et l'ajouter au conteneur. L'interface virtuelle sera reliée à l'interface hôte sous-jacente et ainsi le conteneur sera exposé au réseau extérieur, ce qui lui permet d'obtenir une adresse IP distincte via DHCP à partir du même LAN auquel l'hôte est connecté.

systemd-nspawn propose 2 options :

- **–network-macvlan=interface**: l'interface virtuelle aura une adresse MAC différente de celle de l'interface physique sous-jacente et sera nommée **mv-interface**.
- **–network-ipvlan=interface**: l'interface virtuelle aura la même adresse MAC que l'interface physique sous-jacente et sera nommée **iv-interface**.

Les deux options impliquent **–private-network**.

Utiliser une interface existante

Si le système hôte possède plusieurs interfaces réseau physiques, On peut utiliser `--network-interface=interface` pour attribuer une interface au conteneur (et la rendre indisponible pour l'hôte pendant le démarrage du conteneur). Cela implique `--private-network`.

La transmission d'interfaces réseau sans fil aux conteneurs systemd-nspawn n'est actuellement pas prise en charge.

Mappage des ports

Lorsque la mise en réseau privée est activée, des ports individuels sur l'hôte peuvent être mappés sur des ports sur le conteneur à l'aide de l'option **-p/–port** ou en utilisant le paramètre Port dans un fichier `.nspawn`. Cela se fait en envoyant des règles iptables à la table nat, mais la chaîne FORWARD dans la table de filtrage doit être configurée manuellement.

Par exemple, pour mapper un port TCP 8000 sur l'hôte au port TCP 80 dans le conteneur définir dans `/etc/systemd/nspawn/container-name.nspawn` une unité:

```
[Network]
Port=tcp:8000:80
```

systemd-nspawn exclut explicitement l'interface de bouclage lors de la mapports de ping. Par conséquent, pour l'exemple ci-dessus, localhost:8000 se connecte à l'hôte et non au conteneur. Seules les connexions à d'autres interfaces sont soumises au mappage de port.

Résolution de nom de domaine

La résolution de nom de domaine dans le conteneur peut être configurée par l'option **–resolv-conf** de **systemd-nspawn** ou l'option correspondante **ResolvConf=** pour les fichiers `.nspawn`. Il existe de nombreuses valeurs possibles.

La valeur par défaut est **auto**, ce qui signifie que :

- Si **–private-network** est activé, le fichier `/etc/resolv.conf` est laissé tel quel dans le conteneur.
- Sinon, si **systemd-resolved** s'exécute sur l'hôte, son fichier stub resolv.conf est copié ou monté en liaison dans le conteneur.

- Sinon, le fichier `/etc/resolv.conf` est copié ou monté en liaison de l'hôte vers le conteneur.

Dans les deux derniers cas, le fichier est copié, si la racine du conteneur est accessible en écriture, et monté en liaison s'il est en lecture seule.

Trucs et astuces

Télécharger d'une image et ouverture d'un shell

- Téléchargement d'une image Ubuntu et y ouvrir un shell

```
machinectl pull-tar
https://cloud-images.ubuntu.com/trusty/current/trusty-server-cloudimg-amd64-
root.tar.gz
systemd-nspawn -M trusty-server-cloudimg-amd64-root
```

Cela télécharge et vérifie l'image .tar spécifiée, puis utilise systemd-nspawn(1) pour y ouvrir un shell.

- Téléchargement d'une image Fedora, définission du mot de passe root, et démarrage en tant que service

```
machinectl pull-raw --verify=no \
https://download.fedoraproject.org/pub/fedora/linux/releases/35/Cloud/x86_64
/images/Fedora-Cloud-Base-35-1.2.x86_64.raw.xz\
      Fedora-Cloud-Base-35-1.2.x86-64
systemd-nspawn -M Fedora-Cloud-Base-35-1.2.x86-64
passwd
logout
machinectl start Fedora-Cloud-Base-35-1.2.x86-64
machinectl connect Fedora-Cloud-Base-35-1.2.x86-64
```

Ceci télécharge l'image **.raw** spécifiée avec la vérification désactivée. Ensuite, un shell y est ouvert et un mot de passe root est défini. Ensuite, le shell est laissé et la machine a démarré en tant que service système. Avec la dernière commande, une invite de connexion au conteneur est demandée.

Exporter une image de conteneur sous forme de fichier tar

```
machinectl export-tar fedora myfedora.tar.xz
```

Exporte le conteneur “fedora” en tant que fichier tar compressé xz myfedora.tar.xz dans le répertoire courant.

Créer une nouvelle session shell

```

```

```
machinectl shell --uid=lennart
```

Cela crée une nouvelle session shell sur l'hôte local pour l'ID utilisateur « lennart », à la manière de su(1).

Conteneurs non privilégiés

systemd-nspawn prend en charge les conteneurs non privilégiés, bien que les conteneurs doivent être démarrés en tant que root.

cette fonctionnalité nécessite **user_namespaces**

La façon la plus simple de le faire est de laisser **systemd-nspawn** choisir automatiquement une plage inutilisée d'UID/GID en utilisant l'option -U :

```
systemd-nspawn -bUD ~/MyContainer
```

Si le noyau prend en charge les espaces de noms d'utilisateurs, l'option **-U** est équivalente à `--private-users=pick --private-users-chown`. Cela implique que les fichiers et les répertoires du conteneur sont associés à la plage sélectionnée d'UID/GID privés au démarrage du conteneur.

On peut également spécifier manuellement la plage UID/GID du conteneur, mais cela est rarement utile.

Une fois qu'on a démarré un conteneur avec une plage UID/GID privée, il faut continuer à l'utiliser de cette façon pour éviter les erreurs d'autorisation. Alternativement, il est possible d'annuler l'effet de `--private-users-chown` (ou `-U`) sur le système de fichiers en spécifiant une plage d'ID commençant à 0 :

```
systemd-nspawn -D ~/MonContainer --private-users=0 --private-users-chown
```

Utiliser un environnement X

Il faut définir la variable d'environnement DISPLAY dans la session de conteneur pour se connecter au serveur X externe.

X stocke certains fichiers requis dans le répertoire /tmp. Pour que le conteneur affiche quoi que ce soit, il doit avoir accès à ces fichiers. Pour ce faire, ajouter l'option `--bind-ro=/tmp/.X11-unix` au démarrage du conteneur.

Depuis la version 235 de systemd, le contenu de `/tmp/.X11-unix` doit être monté en lecture seule, sinon il disparaîtra du système de fichiers. L'indicateur de montage en lecture seule n'empêche pas l'utilisation de l'appel système connect() sur le socket. Si on a également lié `/run/user/1000`, il faut lier explicitement `/run/user/1000/bus` en lecture seule pour empêcher la suppression du socket **dbus**.

xhost ne fournit que des droits d'accès plutôt grossiers au serveur X. Un contrôle d'accès plus fin est possible via le fichier **$XAUTHORITY**. Malheureusement, le simple fait de rendre le fichier **$XAUTHORITY** accessible dans le conteneur ne suffira pas : le fichier **$XAUTHORITY** est spécifique à l'hôte, mais le conteneur est un hôte différent. L'astuce suivante peut être utilisée pour que le serveur X accepte le fichier **$XAUTHORITY** d'une application X exécutée à l'intérieur du conteneur :

```
$XAUTH=/tmp/container_xauth
$xauth nextract - "$DISPLAY" | sed -e 's/^..../ffff/' | xauth -f
"$XAUTH" nmerge -
systemd-nspawn -D monContainer --bind=/tmp/.X11-unix --
bind="$XAUTH" -E DISPLAY="$DISPLAY" -E XAUTHORITY="$XAUTH" --as-
pid2 /usr/ bin/xeyes
```

La deuxième ligne ci-dessus définit la famille de connexion sur “FamilyWild”, valeur 65535, ce qui fait que l'entrée correspond à chaque affichage.

Utilisation du wrapper X/Xephyr

Un autre moyen simple d'exécuter des applications X et d'éviter les risques d'un bureau X partagé consiste à utiliser le wrapper X. Les avantages ici sont d'éviter entièrement l'interaction entre les applications dans le conteneur et les applications non conteneur et de pouvoir exécuter un environnement de bureau ou un gestionnaire de fenêtres différent, les inconvénients sont moins de performances et le manque d'accélération matérielle lors de l'utilisation de **Xephyr**.

Démarrer **Xephyr** en dehors du conteneur en utilisant :

```
Xephyr :1 -resizeable
```

Démarrer ensuite le conteneur avec les options suivantes :

```
--setenv=DISPLAY=:1 --bind-ro=/tmp/.X11-unix/X1
```

Aucune autre liaison n'est nécessaire.

Il faut peut-être toujours définir manuellement **DISPLAY=:1** dans le conteneur dans certaines circonstances (principalement si utilisé avec **-b**).

Exécuter une application

Pour exécuter Firefox en tant que PID 1:

```
systemd-nspawn --setenv=DISPLAY=:0 \
               --setenv=XAUTHORITY=~/.Xauthority \
               --bind-ro=$HOME/.Xauthority:/root/.Xauthority \
               --bind=/tmp/.X11-unix \
               -D ~/containers/firefox \
               firefox
```

Alternativement, On peut démarrer le conteneur et laisser par ex. **systemd-networkd** configurer l'interface réseau virtuelle :

```
systemd-nspawn --bind-ro=$HOME/.Xauthority:/root/.Xauthority \
               --bind=/tmp/.X11-unix \
               -D ~/containers/firefox \
               --network-veth -b
```

Une fois le conteneur démarré, exécuter le binaire Xorg comme suit :

```
systemd-run -M firefox --setenv=DISPLAY=:0 firefox
```

Accélération graphique 3D

Pour activer les graphiques 3D accélérés, il peut être nécessaire de lier `/dev/dri` au conteneur en ajoutant la ligne suivante au fichier `.nspawn`:

```
Bind=/dev/dri
```

Ceci permet de résoudre notamment l'erreur suivante:

```
libGL error: MESA-LOADER: failed to retrieve device information
libGL error: Version 4 or later of flush extension not found
libGL error: failed to load driver: i915
```

On peut confirmer qu'il a été activé en exécutant **glxinfo** ou **glxgears**.

Accéder au système de fichiers hôte

Par exemple lorsque l'hôte et le conteneur sont Arch Linux, alors on pourrait, partager le cache pacman :

```
systemd-nspawn --bind=/var/cache/pacman/pkg
```

On peut également spécifier une liaison par conteneur dans le fichier `/etc/systemd/nspawn/mon-container.nspawn`:

```
[Files]
Bind=/var/cache/pacman/pkg
```

Pour lier le répertoire à un chemin différent dans le conteneur, ajouter le chemin en le séparant par deux-points. Par example:

```
systemd-nspawn --bind=/path/to/host_dir:/path/to/container_dir
```

Utiliser le sous-volume Btrfs comme racine de conteneur

Pour utiliser un sous-volume Btrfs comme modèle pour la racine du conteneur, utiliser l'indicateur `--template`. Cela prend un instantané du sous-volume et remplit le répertoire racine du conteneur avec celui-ci.

Si le chemin du modèle spécifié n'est pas la racine d'un sous-volume, l'arborescence entière est copiée. Cela prendra beaucoup de temps.

Par exemple, pour utiliser un instantané situé dans `/.snapshots/403/snapshot` :

```
systemd-nspawn --template=/.snapshots/403/snapshots -b -D mon-conteneur
```

où **my-container** est le nom du répertoire qui sera créé pour le conteneur. Après la mise hors tension, le sous-volume nouvellement créé est conservé.

On peut utiliser l'indicateur `--ephemeral` ou `-x` pour créer un instantané btrfs temporaire du conteneur et l'utiliser comme racine du conteneur. Toutes les modifications apportées lors du démarrage dans le conteneur seront perdues. Par example:

```
systemd-nspawn -D my-container -xb
```

où **my-container** est le répertoire d'un conteneur ou d'un système existant. Par exemple, si **/** est un sous-volume btrfs, on peut créer un conteneur éphémère du système hôte en cours d'exécution en faisant :

```
systemd-nspawn -D / -xb
```

Après la mise hors tension du conteneur, le sous-volume btrfs qui a été créé est immédiatement supprimé.

Exécuter docker dans systemd-nspawn

Depuis Docker 20.10, il est possible d'exécuter des conteneurs Docker dans un conteneur systemd-nspawn non privilégié avec cgroups v2 activé (par défaut dans Arch Linux) sans compromettre les mesures de sécurité en désactivant les cgroups et les espaces de noms d'utilisateurs. Pour ce faire, modifier `/etc/systemd/nspawn/myContainer.nspawn` (créer si absent) et ajouter les configurations suivantes.

```
[Exec]
SystemCallFilter=add_key keyctl
```

Ensuite, Docker devrait fonctionner tel quel à l'intérieur du conteneur.

La configuration ci-dessus expose les appels système add_key et keyctl au conteneur, qui ne sont pas dans un espace de noms. Cela pourrait toujours être un risque de sécurité, même s'il est bien inférieur à la désactivation complète de l'espacement des noms d'utilisateurs comme ce qu'il fallait faire avant cgroups v2.

Étant donné que **overlayfs** ne fonctionne pas avec les espaces de noms d'utilisateurs et n'est pas disponible dans **systemd-nspawn**, par défaut, Docker utilise le vfs inefficace comme pilote de stockage, ce qui crée une copie de l'image à chaque démarrage d'un conteneur. Cela peut être contourné en utilisant **fuse-overlayfs** comme pilote de stockage. Pour ce faire, il faut d'abord exposer `/dev/fuse` au conteneur dans le fichier `/etc/systemd/nspawn/myContainer.nspawn`:

```
[Files]
Bind=/dev/fuse
```

puis autoriser le conteneur à lire et à écrire le nœud de l'appareil :

```
systemctl set-property systemd-nspawn@myContainer DeviceAllow='/dev/fuse
rwm'
```

Enfin, installer le package **fuse-overlayfs** à l'intérieur du conteneur. Vous dever redémarrer le conteneur pour que toute la configuration prenne effet.

Dépannage

La connexion root échoue

Si on a l'erreur suivante lorsqu'on essaye de se connecter(c'est-à-dire en utilisant `machinectl login <name>`):

```
arch-nspawn login: root
Login incorrect
```

Et le journal indique :

```
pam_securetty(login:auth): access denied: tty 'pts/0' is not secure !
```

Il est possible de supprimer `/etc/securetty` et `/usr/share/factory/etc/securetty` sur le système de fichiers du conteneur, ou simplement d'ajouter les terminaux **pty** souhaités (comme **pts/0**), si nécessaire, à `/etc/securetty` sur le système de fichiers du conteneur. Toutes les modifications seront annulées au prochain démarrage, il est donc nécessaire de supprimer également l'entrée `/etc/securetty` de `/usr/lib/tmpfiles.d/arch.conf` sur le système de fichiers du conteneur. Si on opte pour la suppression, On peut également éventuellement mettre les fichiers sur liste noire (**NoExtract**) dans `/etc/pacman.conf` pour éviter qu'ils ne soient réinstallés.

execv(...) failed: Permission denied

Lorsqu'on essaye de démarrer le conteneur via `systemd-nspawn -bD /path/to/container` (ou qu'on exécute quelque chose dans le conteneur), l'erreur suivante apparaît :

```
execv(/usr/lib/systemd/systemd, /lib/systemd/systemd, /sbin/init) failed:
Permission denied
```

même si les autorisations des fichiers en question (c'est-à-dire `/lib/systemd/systemd`) sont correctes, cela peut être le résultat d'avoir monté le système de fichiers sur lequel le conteneur est stocké en tant qu'utilisateur non root. Par exemple, si on monte un disque manuellement avec une entrée dans fstab qui a les options **noauto,user,..., systemd-nspawn** n'autorisera pas l'exécution des fichiers même s'ils appartiennent à root.

Le type de terminal dans TERM est incorrect (couleurs brisées)

Lors de la connexion au conteneur via la connexion **machinectl**, les couleurs et les frappes dans le terminal du conteneur peuvent être brisées. Cela peut être dû à un type de terminal incorrect dans la variable d'environnement TERM. La variable d'environnement n'est pas héritée du shell sur l'hôte, mais revient à une valeur par défaut fixée dans **systemd** (vt220), sauf si elle est explicitement configurée. Pour configurer, dans le conteneur, créer une superposition de configuration pour le service **container-getty@.service systemd** qui lance la connexion **getty** pour **machinectl login**,

et définir TERM sur la valeur qui correspond au terminal hôte à partir duquel on se connecte dans le fichier `/etc/systemd/system/container-getty@.service.d/term.conf`:

```
[Service]
Environment=TERM=xterm-256color
```

On peut également utiliser le shell **machinectl**. Il hérite correctement de la variable d'environnement TERM du terminal.