OSTree : virtualisation avec systemd-nspawn

Objet	Création d'environnements virtuels avec systemd-nspawn
Niveau requis	débutant, avisé
Débutant, à savoir	
Suivi	:DRAFT:

Lorsqu'on a un serveur physique, et qu'on veut créer plusieurs environnements virtuels où on pourra tester différentes choses, les machines physiques sont difficiles à cloner, mais on peut configurer une sorte d'environnement virtuel par-dessus pour pouvoir démarrer une distribution de système d'exploitation plus tard.

Le lab décrit comment utiliser systemd-nspawn et systemd-machined pour gérer des machines “virtuelles” en utilisant une distribution OSTree en tant qu'hôte et invité. OSTree permet de partager un seul /ostree entre différents systèmes. voici comment je l'ai fait.

Créer le déploiement OSTree

Créer un nouveau déploiement OSTree sans impact sur l'hôte. Tout d'abord, créer un nouveau système d'exploitation dans OSTree:

```
[root@host host]# ostree admin os-init guest
ostree/deploy/guest initialized as OSTree root
```

On a maintenant un OS invité enregistré, mais il n'a aucun déploiement. Un déploiement est une image système que l'on superpose aux données résidentes (/var). Pour créer un déploiement, on a besoin d'un hachage de validation OSTree pour le référencer.

Faire un ostree admin status pour voir less déploiements:

```
# ostree admin status

  fedora 131642a5b3da24dc47f9ff8191f5856bbeced08275ea823fa543327fd20f4862.0
(staged)
    Version: 30.20190715.0
    origin: <unknown origin type>
* fedora 8e721e68bfa7c813361f3067d202b2fd907634e192059451ffa8365ca5733224.0
    Version: 30.20190713.0
    origin: <unknown origin type>
  fedora 38a1f86b1168dda0e8ed80782afb3e56bb1918b1984c1dc0d7216f63f865a7a0.0
(rollback)
    Version: 30.20190701.0
    origin: <unknown origin type>
```

On peut voit, qu'on a effectué une mise à jour (staged) mais qu'on n'a pas redémarré. On a également le déploiement actuel et le déploiement de restauration. les hachages ne sont pas encore

des ID de validation.

Faire un rpm-ostree status pour l'obtenir les ID de validation:

```
# rpm-ostree status

State: idle
AutomaticUpdates: disabled
Deployments:
  ostree://fedora:fedora/30/x86_64/silverblue
                   Version: 30.20190715.0 (2019-07-15T00:36:07Z)
                BaseCommit:
fa23b170d892d78935de422064e73a84057119ea315257a9030dc7659e726a7b
                 StateRoot: fedora
              GPGSignature: Valid signature by
F1D8EC98F241AAF20DF69420EF3C111FCFC659B9
                      Diff: 11 upgraded, 1 added
           LayeredPackages: net-tools strace systemd-container tcpdump tmux

● ostree://fedora:fedora/30/x86_64/silverblue
                   Version: 30.20190713.0 (2019-07-13T00:38:50Z)
                BaseCommit:
f8aeed3ee64bddecb6ebfd382db9c4b19c5f5502cb24fdb4eed788b9c4558f62
                 StateRoot: fedora
              GPGSignature: Valid signature by
F1D8EC98F241AAF20DF69420EF3C111FCFC659B9
           LayeredPackages: net-tools systemd-container tcpdump tmux

  ostree://fedora:fedora/30/x86_64/silverblue
                   Version: 30.20190701.0 (2019-07-01T00:36:21Z)
                BaseCommit:
f5efad3126a7bc90b88c0a3ba382c15eb3119e123eef3b1f1e2c4a7f7480fcc1
                 StateRoot: fedora
              GPGSignature: Valid signature by
F1D8EC98F241AAF20DF69420EF3C111FCFC659B9
           LayeredPackages: net-tools tcpdump tmux
```

On prend le commit fa23b170d892d78935de422064e73a84057119ea315257a9030dc7659e726a7b comme commit de base pour notre nouveau déploiement invité.

```
# ostree admin deploy --not-as-default --os=guest
fa23b170d892d78935de422064e73a84057119ea315257a9030dc7659e726a7b

Relabeling /var (no stamp file 'var/.ostree-selabeled' found)
Bootloader updated; bootconfig swap: yes; deployment count change: 0
```

Il faut spécifier `--not-as-default` sinon lorsque l'hôte va redémarrer, il redémarrera sur le déploiement invité. Ce ne serait pas bien.

Configurer systemd-nspawn pour démarrer l'invité

Parce que `systemd-nspawn` accède à `/sysroot` et que SELinux n'est pas configuré pour autoriser cela par défaut, on peut simplement régler SELinux en mode permissif:

```
# setenforce Permissive
```

Maintenant, il faut créer un dropin à systemd-nspawn@guest.service pour configurer l'option de systemd-nspawn:

```
# cat /etc/systemd/system/systemd-nspawn@guest.service.d/10-service.conf
[Service]
ExecStart=
Environment=SYSTEMD_NSPAWN_LOCK=0
ExecStart=/bin/sh -xc ' \
        deployment=`ostree admin status | sed -ne \'s/. %i \\(\\S*\\)\\(
.*\\)*$/\\1/; T; p; q\'`; \
        exec /usr/bin/systemd-nspawn --quiet --keep-unit --boot --link-
journal=try-guest --network-veth --settings=override --machine=%i \
        --directory=/sysroot \
        --bind /boot:/boot \
        --bind +/sysroot/ostree/deploy/%i/var:/var \
        --pivot-root /ostree/deploy/%i/deploy/$deployment:/sysroot \
        $SYSTEMD_NSPAWN_FLAGS \
        '
RestartForceExitStatus=
Restart=always
RestartSec=3s
```

- **-U** est supprimé car on ne veut pas que `systemd-nspawn` mette à jour l'UID/GID dans `/ostree`.
- **–directory=/sysroot** indique à systemd-nspawn de démarrer `/sysroot` en tant que répertoire racine. Il s'agit de la racine du système de fichiers.
- **–bind /boot:/boot** indique de garder la partition de démarrage montée. Cela est nécessaire pour que la ligne de commande ostree fonctionne (elle recherche les fichiers dans `/boot/loader/entries`)
- **–bind +/sysroot/ostree/deploy/%i/var:/var** indique de monter le répertoire `/var` persistant. Sinon, on aura un système complet en lecture seule.
- **–pivot-root /ostree/deploy/%i/deploy/$deployment:/sysroot** indique de pivoter-root vers le déploiement spécifique (calculé à partir du pipeline sed ci-dessus)
- **Restart=always** et **RestartSec=3s** donne suffisamment de temps au système-usiné pour annuler l'enregistrement de la machine afin que la machine puisse redémarrer sans erreurs (Échec d'enregistrement de la machine: la machine “invité” existe déjà)

Avec cela, on peut maintenant activer et démarrer le service, et on a une sous-machine prête.

```
# systemctl enable --now systemd-nspawn@guest.service
# machinectl list
MACHINE CLASS     SERVICE        OS     VERSION ADDRESSES
guest   container systemd-nspawn fedora 30      -

2 machines listed.
# machinectl shell guest
Connected to machine guest. Press ^] three times within 1s to exit session.

Welcome to Fedora Silverblue. This terminal is running on the
immutable host system. You may want to try out the Toolbox for a
more traditional environment that allows package installation
with DNF.

For more information, see the documentation.

# ostree admin status
  fedora 8e721e68bfa7c813361f3067d202b2fd907634e192059451ffa8365ca5733224.0
    Version: 30.20190713.0
    origin: <unknown origin type>
* guest fa23b170d892d78935de422064e73a84057119ea315257a9030dc7659e726a7b.0
    Version: 30.20190715.0
    origin refspec:
fa23b170d892d78935de422064e73a84057119ea315257a9030dc7659e726a7b
  fedora 38a1f86b1168dda0e8ed80782afb3e56bb1918b1984c1dc0d7216f63f865a7a0.0
    Version: 30.20190701.0
    origin: <unknown origin type>
```

Il faut également activer machines.target:

```
$ systemctl enable machines.target
Created symlink /etc/systemd/system/multi-user.target.wants/machines.target
→ /usr/lib/systemd/system/machines.target.
```

Configurer le pont réseau

Pour rendre la machine invitée accessible à partir du réseau, elle doit être pontée (ou autrement connectée) au réseau hôte. Par défaut, une paire d'interface réseau est créée mais non configurée.

Tout d'abord, créer un pont sur l'hôte avec `systemd-networkd` qui convient aux serveurs.

```
# systemctl status systemd-networkd
● systemd-networkd.service - Network Service
   Loaded: loaded (/usr/lib/systemd/system/systemd-networkd.service;
enabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-07-15 14:34:30 CEST; 22s ago
```

Ensuite, créer un périphérique de pont:

```
# cat /etc/systemd/network/br0.netdev
[NetDev]
Name=br0
Kind=bridge
```

Attribuer les interfaces réseau en amont au pont (attention, si vous avez des règles DHCP statiques, l'adresse MAC de l'hôte passera à l'adresse MAC du pont, qui est déterministe):

```
# cat /etc/systemd/network/br0-bind.network
[Match]
Name=en*

[Network]
Bridge=br0
```

Activer DHCP sur le pont:

```
# cat /etc/systemd/network/br0.network
[Match]
Name=br0

[Network]
DHCP=ipv4
```

Et ajouter un drapeau systemd-nspawn pour relier la machine:

```
# cat /etc/systemd/system/systemd-nspawn@guest.service.d/10-flags.conf
[Service]
Environment=SYSTEMD_NSPAWN_FLAGS=--network-bridge=br0
```

Ensuite, on peut redémarrer systemd-networkd et systemd-nspawn@guest.

Résolution des problèmes

Périphériques de bloc

Lorsqu'on utilise btrfs, on peut avoir des problèmes avec les répertoires de ces périphérique pour de données parce que `systemd-nspawn` masque les périphériques de bloc, il échoue.

La solution consiste à ajouter à la ligne de commande `systemd-nspawn` avant la racine pivot:

```
--bind /dev: /dev
```

Obtenir /proc /sys en écriture

Pour obtenir `/proc`, `/sys` accessible en écriture, il faut démarrer `systemd-nspawn` avec:

```
Environnement=SYSTEMD_NSPAWN_API_VFS_WRITABLE=1
```