Module Linux

Enseignante

H.Slimani
S. BEN YAALA

Conteneurisation dans Red Hat

Plan

- Docker vs. Podman
- Manipulation des images
- Manipulation des conteneurs

Introduction

LXC

LXC, Technically first implementation of Container

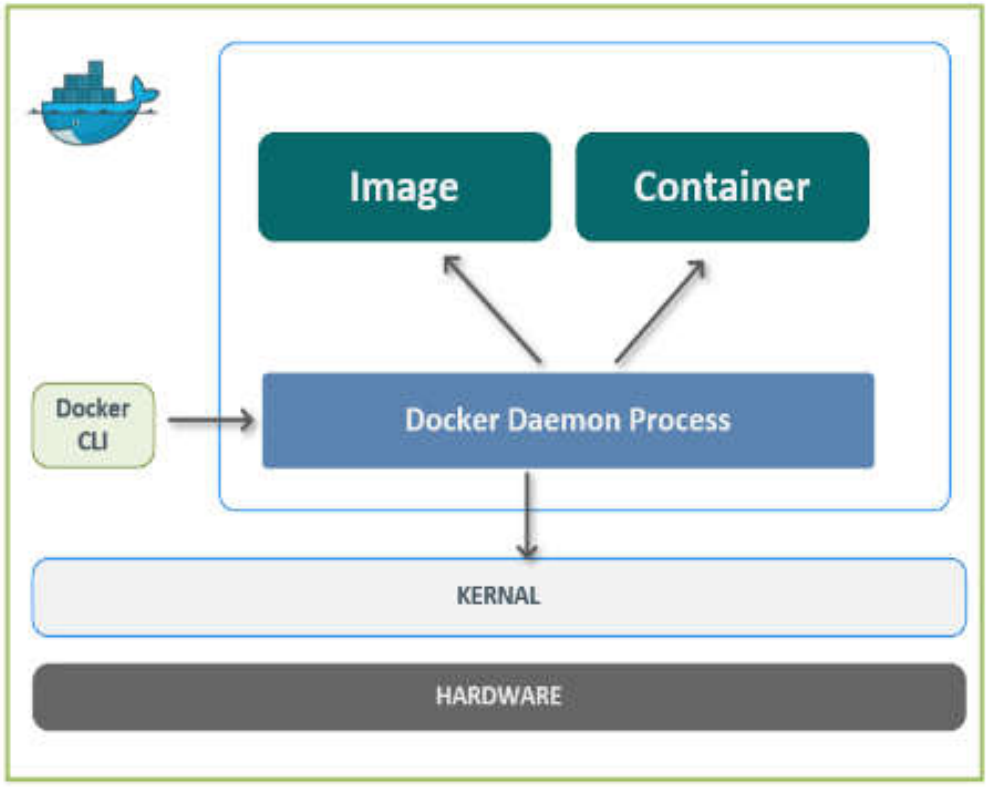
Daemon process based Container Runtime Engine, Standardize container implementation

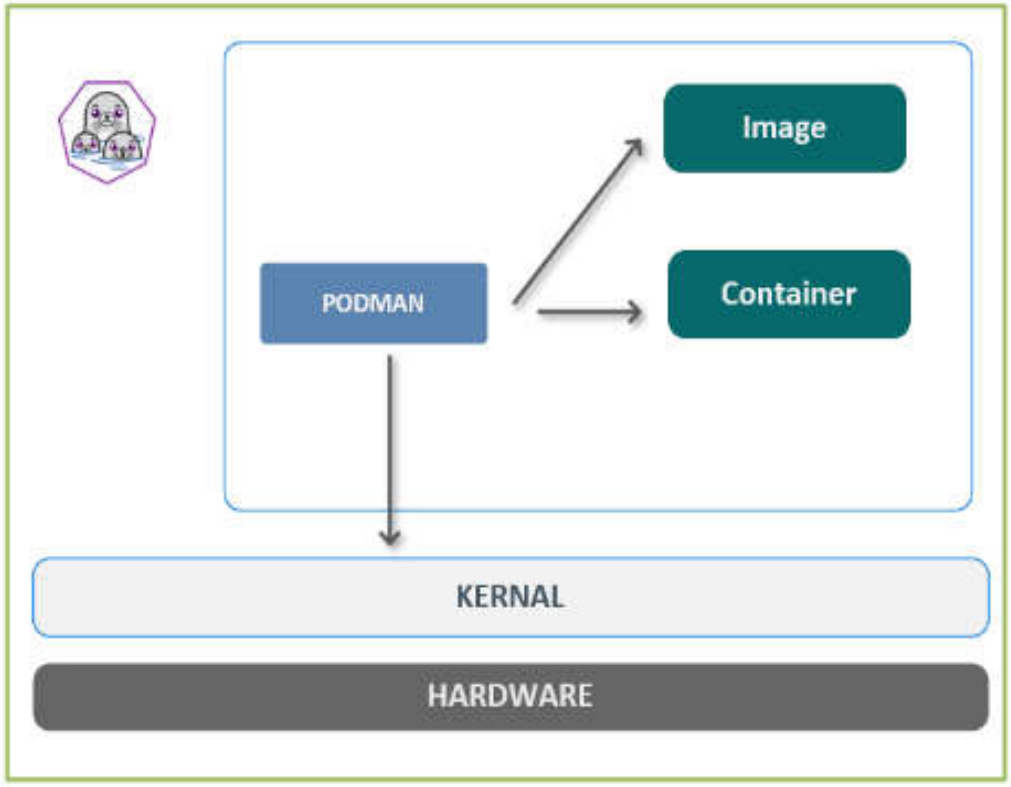
Opensource Solution for Containers, No Daemon Process, Security focused

Introduction

- Depuis son lancement en 2014, Docker est la solution leader pour l'exécution de conteneurs.
- Jusqu'à RHEL 7, Docker était la pile de conteneurs par défaut utilisée sur RHEL.
- Avec la sortie de RHEL 8, Red Hat a décidé de ne plus prendre en charge Docker et de proposer sa propre pile.
- Cette pile utilise podman comme outil principal pour exécuter les conteneurs.

Docker vs. Podman





Avantages de podman

- La nouvelle solution offre quelques avantages par rapport à la solution Docker :
- Dans RHEL 8, les conteneurs s'exécutent par défaut en tant qu'utilisateur non root.
- Les utilisateurs peuvent exécuter leurs propres conteneurs et, donc, les conteneurs s'exécutent dans un espace de noms d'utilisateur où ils sont strictement isolés et inaccessibles aux autres utilisateurs.
- La solution RHEL 8 n'a pas besoin de démon, mais les conteneurs s'exécutent au-dessus de l'environnement d'exécution léger du conteneur CRI-o.

Avantages de podman

- Un avantage important de l'utilisation de podman est le rootless container.
- Les conteneurs rootless sont démarrés par des utilisateurs non root et ne nécessitent pas de privilèges root. Cela rend l'exécution des conteneurs beaucoup plus sécurisée, mais cela s'accompagne également de certains défis.
- Les conteneurs rootless ne peuvent pas accéder aux composants du système d'exploitation hôte qui nécessitent un accès root. Par exemple, les conteneurs sans racine n'ont pas d'adresse IP (car ils nécessitent des privilèges root pour allouer une adresse IP) et ne peuvent se lier qu'à un port non privilégié.
- En outre, si le conteneur sans racine a besoin d'accéder au stockage basé sur l'hôte, l'utilisateur qui exécute le conteneur doit être propriétaire du répertoire qui fournit le stockage.

Registres d'images

- Les images de conteneur sont généralement extraites des registres de conteneurs, qui sont spécifiés dans le fichier de configuration **/etc/containers/registries.conf.**
- Un utilisateur qui exécute un conteneur sans racine peut créer un fichier **~/.config/containers/registries.conf**. En cas de conflit, les paramètres du fichier spécifique à l'utilisateur remplaceront les paramètres du fichier générique.
- Dans le fichier **registries.conf**, trois registres sont utilisés par défaut. Les deux premiers sont des registres Red Hat qui donnent accès à des logiciels sous licence. Vous devez saisir vos identifiants Red Hat pour accéder à ces registres. Pour le troisième registre, le registre Docker est utilisé. Docker héberge le plus grand registre de conteneurs actuellement disponible, et l'ajout du registre Docker comme dernier registre augmentera vos chances de trouver l'image de conteneur souhaitée.

Installation et login

- Installation de module container-tools

```
yum module install container-tools -y
```

- Login au registre d'images redhat

```
podman login
```

Création d'un compte sur le site de redhat

+ Manipulation des images

$ podman info

$ podman search mariadb

$ podman search --filter is-official=true alpine (filtrer les images officielles)

$ podman inspect busybox (inspect images en local)

$ Skopio inspect busybox (inspect image dans le register)

Manipulation des images

- **podman run** : extrait d'abord l'image (pull), la stocke sur votre système local, puis exécute le conteneur.

- **podman pull** : stocker l'image sans l'exécuter, et après l'avoir extraite, vous pouvez toujours l'exécuter. Cette deuxième méthode est plus sécurisée car elle vous permet d'inspecter le contenu de l'image avant de l'exécuter.

- **podman rmi** : supprimer des images de conteneur (le conteneur doit etre arrété).

+ Manipulation des conteneurs

- La manipulation d'un conteneur englobe :
  - La gestion de l'état des conteneurs
  - L'exécution des commandes dans un conteneur
  - La gestion des ports
  - La gestion de stockage
  - Exécution des conteneurs comme un service systemd

Gestion de l'état des conteneurs

- **podman stop :** envoie un signal SIGTERM au conteneur. Si cela ne donne aucun résultat après 10 secondes, une commande SIGKILL est envoyée.
- **podman kill** : envoie immédiatement une commande SIGKILL. Dans la plupart des cas, ce n'est pas nécessaire car l'arrêt du podman enverra également un SIGKILL après 10 secondes.
- **podman restart** : redémarre un conteneur en cours d'exécution.
- **podman start** : démarre un conteneur qui n'est pas en cours d'exécution (stoppé).
- **podman rm** <container> : supprimer un conteneur
- **podman run –rm** <container> : après son exécution, les fichiers du conteneur sont automatiquement nettoyés.

Exécution des commandes dans un conteneur

- **podman exec** mycontainer uname -r : exécuter la commande et écrire sa sortie dans STDOUT
- **podman exec -it** myncontainer /bin/bash : pour ouvrir un shell Bash dans le conteneur et exécuter plusieurs commandes à partir de là,

Gestion des ports de conteneurs

- Les **« rootless containers »** dans podman s'exécutent sans adresse réseau car ils ne dispose pas de privilèges suffisants pour allouer une adresse réseau.
- Pour rendre le service exécuté dans le conteneur accessible de l'extérieur, vous devez configurer la redirection de port, où un port sur l'hôte du conteneur est utilisé pour accéder à un port dans l'application de conteneur.
- Notez que lorsque vous exécutez un « rootless container », vous ne pouvez adresser que des ports non privilégiés sur l'hôte : les ports 1 à 1024 sont accessibles uniquement par l'utilisateur root.

Gestion des ports de conteneurs

- Pour exécuter un conteneur avec transfert de port, vous ajoutez l'option -p à la commande podman run :

```
podman run --name nginxport -d -p 8080:80 nginx
```

pour exécuter l'image nginx en tant que conteneur et rendre le processus nginx accessible sur le port hôte 8080, qui sera transmis au port http standard 80 sur lequel nginx offre son prestations de service.

- Ouvrir également le port dans le pare-feu :

```
sudo firewall-cmd --add-port 8080/tcp --permanent
sudo firewall-cmd --reload
```

- Après avoir exposé un conteneur de serveur Web sur un port hôte, vous pouvez utiliser :

```
curl localhost:8080        (pour vérifier l'accès).
```

Gestion du stockage dans les conteneurs

- Pour ajouter un stockage persistant aux conteneurs Podman, vous montez par un répertoire sur le système d'exploitation hôte dans le conteneur.
- Un montage lié est un type spécifique de montage, où un répertoire est monté au lieu d'un périphérique bloc. Cela garantit que le contenu du répertoire sur le système d'exploitation hôte est accessible dans le conteneur.
- Ainsi, lorsque des fichiers sont écrits dans le conteneur dans le répertoire monté par liaison, ils sont également validés dans le système d'exploitation hôte, ce qui garantit que les données seront disponibles au-delà de la durée de vie du conteneur.

Gestion du stockage dans les conteneurs

Pour monter le volume, vous utilisez la commande :

--volume rép_hôte:rép_conteneur .

Si l'utilisateur exécutant le conteneur est le propriétaire ou si le conteneur est un conteneur root, vous pouvez utiliser :

--volume rép_hôte: rép_conteneur:Z

comme alternative à la définition automatique du contexte SELinux. mais cela ne fonctionne que si l'utilisateur qui exécute le conteneur est propriétaire du répertoire. Ce n'est pas suffisant si l'utilisateur a des droits d'écriture sur le répertoire !

Exécuter des conteneurs comme services Systemd

- Dans systemd, les services sont facilement démarrés et activés avec les autorisations root à l'aide de commandes telles que

  systemctl enable --now myservice.service.

- Si aucune autorisation root n'est disponible, vous devez utiliser **systemctl --user**. L'option **--user** permet aux utilisateurs d'exécuter les commandes systemd courantes, mais uniquement dans l'espace utilisateur. Cela fonctionne pour n'importe quel service qui peut s'exécuter sans les autorisations root ; par exemple, utilisez la cde suivante pour démarrer le service myservice:

  systemctl --user start myservice.service

Exécuter des conteneurs comme services Systemd

- Par défaut, lorsque systemctl --user est utilisé, les services ne peuvent être démarrés automatiquement qu'au démarrage d'une session utilisateur.
- Pour définir une exception à cela, vous pouvez utiliser le gestionnaire de session loginctl, qui fait partie de la solution systemd pour activer le « linger» pour un compte d'utilisateur spécifique.
- **loginctl enable-linger myuser** : activer le linger pour l'utilisateur myuser. Lorsque « linger » est activée, les services systemd activés pour cet utilisateur spécifique seront démarrés au démarrage du système, pas seulement lorsque l'utilisateur se connecte.

Exécuter des conteneurs comme services Systemd

podman generate systemd --name mycontainer –files -- new

→ générer un fichier d'unité systemd pour démarrer les conteneurs.

→ ce fichier conteneur doit être généré dans le répertoire ~/.config/systemd/user/

→ Il faut créer ce répertoire et y accéder avant d'exécuter la commande podman generate.

- La commande systemd : podman generate suppose qu'un conteneur portant le nom mycontainer a déjà été créé et se traduira par un fichier unité (unit file) qui peut être activé à l'aide de :

systemctl --user enable container-mycontainer.service

Exécuter des conteneurs comme services Systemd


```
# container-mynginx.service
# autogenerated by Podman 1.9.3
# Thu Sep 24 16:00:53 CEST 2020

[Unit]
Description=Podman container-mynginx.service
Documentation=man:podman-generate-systemd(1)
Wants=network.target
After=network-online.target

[Service]
Environment=PODMAN_SYSTEMD_UNIT=%n
Restart=on-failure

ExecStart=/usr/bin/podman start mynginx
ExecStop=/usr/bin/podman stop -t 10 mynginx
PIDFile=/run/user/1001/containers/overlay-containers/554ead617fd6928
cd8dbcd38c3f53727d91f84be8d9785c503f71136df1f9/userdata/conmon.pid
KillMode=none
23
~z
[Install]
```

```
WantedBy=multi-user.target
```

Avant l'activation du service utilisateur systemd, vous devez vous assurer que la ligne **WantedBy** dans le fichier d'unité de service est modifiée. Par défaut, il lit WantedBy=multi-user.target.

L'environnement utilisateur systemd, cependant, n'a pas de multiuser.target, c'est pourquoi cette ligne doit être modifiée pour lire **WantedBy=default.target.**