Docker

- Objet : Mise en place et utilisation de docker.io
- Niveau requis :
  débutant, avisé
- Commentaires : Docker.io est un outil permettant de créer facilement des conteneurs pour certaines applications.
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  à-tester
  - Création par captnfab 13/08/2023
  - Testé par <...> le <...> Fix Me!
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Introduction

Les *conteneurs*, que ce soit *LXC* ou *docker* sont des sortes de mini-machines virtuelles, des sous-systèmes compartimentés, permettant d'exécuter des applications qui ne s'intègrent pas harmonieusement au reste du système, ou que l'on souhaite garder séparées de celui-ci.

Avec LXC, on peut installer par exemple une Archlinux dans un dossier de sa debian, et puis la maintenir au quotidien, l'utiliser quand bon nous semble. À la différence d'une VM, le matériel n'est pas virtualisé, et le noyau utilisé est en fait le noyau de l'hôte (l'OS qui fait tourner le LXC).

Avec Docker, le principe est un peu différent, l'idée est de générer une image toute faite destinée uniquement à faire tourner une application, pré-configurée comme il faut. Exemple: une vieille version de gnuradio qui ne compile plus sous une debian actuelle, qui nécessiterait d'installer des tas de lib pas forcément compatibles avec notre système, et que l'on ne veut surtout pas garder ensuite.

À noter que comme une image docker est figée, elle ne reçoit pas de mise à jour, en particulier de mise à jour de sécurité. Docker n'est donc en général pas une bonne solution pour des serveurs, à moins de mettre à jour régulièrement ses images dockers, ce qui nécessite qu'elles soient reconstruites intégralement en mettant à jour les libs utilisées, ce qui est un potentiel casse-tête.

Installation

```
apt install docker.io
```

Utilisation

Permissions

Pour utiliser docker, il faut soit être root, soit être membre du groupe `docker`.

Le plus simple est de rajouter son utilisateur au groupe `docker`, pour se faire:

```
adduser votre_username docker
```

À noter qu'il faut ensuite redémarrer, fermer la session ou taper `newgrp docker` pour que le shell hérite des bons droits, cf Ajouter un utilisateur à un groupe

Construire un conteneur

De nombreux conteneurs sont disponibles sur internet, mais il est souvent intéressant de construire ses propres conteneurs, souvent en se basant sur des conteneurs existants pour ne pas réinventer *toute* la roue.

On crée pour cela un fichier `Dockerfile`, dont voici un exemple

- qui se base sur une image ubuntu 20.04,
- crée un utilisateur `gnuradio` de base membre des groupes sudo et gnuradio, avec pour mot de passe `gnuradio`,
- lance des commandes de création de dossier, d'installation de paquets, etc.
- copie un fichier `exemple.grc` depuis le dossier courant vers un emplacement dans l'image docker (`/home/gnuradio/exemple.grc`)
- indique que le docker doit être lancé en tant que `gnuradio`,
- indique que le dossier de travail (`pwd`) doit être `/home/gnuradio`
- enfin, indique que la commande à lancer est `gnuradio-companion`.

```
FROM ubuntu:20.04

ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update

RUN apt-get install -y sudo
RUN useradd --create-home --shell /bin/bash -G sudo gnuradio
RUN echo 'gnuradio:gnuradio' | chpasswd

RUN mkdir /home/gnuradio/persistent  && chown gnuradio:gnuradio
/home/gnuradio/persistent
RUN apt-get install -y gir1.2-gtk-3.0 gnuradio gnuradio-dev cmake git
libboost-all-dev libcppunit-dev liblog4cpp5-dev swig liborc-dev libgsl-dev
vim xterm rtl-sdr gr-osmosdr

COPY --chown gnuradio:gnuradio --chmod 0644 exemple.grc
/home/gnuradio/exemple.grc

USER gnuradio
WORKDIR /home/gnuradio

CMD gnuradio-companion
```

Une fois ce fichier créé, on construit l'image que l'on va nommer, par exemple `gnuradio3.8`, via la commande suivante où le `.` indique le dossier comportant le `Dockerfile`.

```
docker build -t gnuradio3.8 .
```

Sauf erreur, l'image est créée et se trouve dans le dépôt local des images, consultable via `docker image ls`.

Pour en savoir plus:

```
man docker-build
man docker-image
```

Lancer une image dans un conteneur

On utilise `docker run` suivi du nom de l'image à lancer, exemple `debian:slim` et optionnellement de la commande à exécuter dans ce docker, par exemple `bash`.

```
docker run -it debian:slim bash
```

Note: `-it` permet d'avoir un terminal interactif vers cette commande.

Il est en général intéressant de partager un dossier ou un fichier entre le conteneur docker et l'hôte. Le mécanisme utilisé pour cela dans docker est basé sur la notion de `volume`.

Voici la commande un peu complexe qui serait utilisée ici pour l'exemple de gnuradio:

- GnuRadio a besoin d'accéder aux périphériques systèmes (`/dev/*`) on peut lui donner cet accès via `–privileged`,
- GnuRadio a besoin d'accéder à Xorg pour l'affichage, on peut lui donner cet accès via `–volume="/tmp/.X11-unix:/tmp/.X11-unix"`, et de manière similaire pour les sockets de dbus et avahi.
- GnuRadio est une application graphique que je veux lancer en tant qu'utilisateur, je dois donc partager mon `.Xauthority` avec l'utilisateur `gnuradio` du docker et indiquer le `DISPLAY` sur lequel il doit s'afficher en partageant ma variable d'environnement.
- Pour accéder aux périphériques audio et radio, je dois m'assurer que l'utilisateur est dans les groupes `audio` et `plugdev`.
- Enfin, je veux que mon dossier `~/partage-gnuradio` soit *monté* dans le dossier `/home/gnuradio/persistent` du conteneur afin de partager des fichiers.

Cela se traduit par:

```
docker run \
  --privileged \
  --volume="/tmp/.X11-unix:/tmp/.X11-unix" \
  --volume="$HOME/.Xauthority:/home/gnuradio/.Xauthority:rw" \
  --volume="$HOME/partage-gnuradio:/home/gnuradio/persistent" \
  --volume="/var/run/dbus:/var/run/dbus" \
  --volume="/var/run/avahi-daemon/socket:/var/run/avahi-daemon/socket" \
  --group-add audio \
  --group-add plugdev \
  --env="DISPLAY" \
  -it \
```

```
  gnuradio3.8
```

L'exemple est un peu complexe à cause de Xorg, dbus et avahi, mais dans le cas de serveurs ou d'applications texte, les choses sont plus simples.

Pour en savoir plus:

```
man docker-run
```

Résolution de problèmes

Droits insuffisants

Les commandes docker retournent un message ressemblant à cela et suggérant une permission refusée :

```
permission denied while trying to connect to the Docker daemon socket at
unix:///var/run/docker.sock:
  Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json":
    dial unix /var/run/docker.sock:
      connect:
        permission denied
```

Le shell utilisé pour lancer la commande ne dispose pas des droits suffisants. S'assurer d'avoir bien suivi le paragraphe Permissions et lu Ajouter un utilisateur à un groupe.

[1)]
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !