Conteneurs et reproductibilité

—

Ensimag 1A

Grégory Mounié (CC-BY-SA)

2024-2025

Le but de ce TP porte sur les processus, la mémoire et les fichiers. Mais il se place dans le cas des conteneurs sous Linux.

Les conteneurs ont plusieurs usages, mais dans ce TP, vous fabriquerez une image permettant d'exécuter de manière fiable, un programme distribué dans l'image. Le but est de garantir l'exécution sur n'importe quelle machine exécutant la technologie de conteneur appropriée.

Le TP est calibré pour occuper les plus rapides d'entre vous jusqu'à la fin du TP. Le sujet donne des indications générales avec un peu d'aide. Une partie de l'exercice est que les plus rapides entrainent les autres pour que l'ensemble du groupe termine le sujet. Travaillez en groupe et échanger sur votre compréhension avec vos camarades. Vous poser des questions sur des cas non triviaux est une bonne façon d'apprendre. Si vous êtes bloqué, vous trouverez à la fin du sujet la liste des commandes.

Ce document peut être téléchargé depuis l'adresse suivante : `http://systemes.pages.ensimag.fr/unix-gitlab/atelier/conteneurs.pdf`.

1. Un conteneur n'est pas une machine virtuelle.

Les concepts d'une machine virtuelle sont simples : les programmes sont exécutés par une machine (virtuelle) qui est simulé par l'exécution d'un programme (le simulateur) sur la véritable machine physique.

Les conteneurs sont différents des machines virtuelles. Ils s'appuyent sur les concepts fondamentaux centraux classiques d'un système d'exploitation, mais ils ont repoussé leur usage un peu plus loin pour obtenir un résultat proche des machines virtuelles, plus souple et moins gourmand en ressource, bref, plus efficace.

Les concepts de bases sont :

Seul au monde Chaque processus s'exécute isolé dans son propre espace de mémoire (sa mémoire virtuelle), comme s'il était seul sur la machine.

Le système d'exploitation est un programme qui s'exécute qu'à la demande (un programme comme les autres, avec une programmation par évènement, comme une interface graphique). Soit à la demande du matériel : presser une touche du clavier ; arrivée d'un message réseaux, fin d'une lecture ou d'une écriture sur un disque ou un SSD, etc.). Soit à la demande des programmes en exécution : (**les appels systèmes**).

Les appels systèmes sont standardisés et stables Sous les différents Unix, une fois créés, ils sont stables. Un programme compilé il y a 30 ans, sous Linux peut s'exécuter aujourd'hui, moyennant d'avoir les bonnes bibliothèques partagées. Les conteneurs résolve cette limitation en embarquant toutes les bibliothèques utilisées par le programme.

Toutes les interactions d'un processus passent par le mécanisme des appels systèmes Le système peut donc changer son comportement en fonction du processus. Cela sert en particulier pour isoler les bugs, les droits et garantir la sécurité.

En conséquence, **la vision d'un processus du reste du monde n'est que ce que le système lui en a dit. Les actions d'un processus sur le reste du monde sont toutes filtrées et réalisées par le système**. C'est ce qui est au centre de l'implémentation des conteneurs.

L'idée centrale des conteneurs est d'exploiter l'isolation des processus pour leur donner des informations et des accès complètement différent les uns des autres. Cela est réalisé principalement avec deux changements. L'arborescence de fichiers vu dans un conteneur n'est pas celle de l'ensemble du système hôte. La liste des fichiers qui sont visibles, et donc qui peuvent être lus ou écris est différente. Cela inclue la liste des programmes. Le deuxième changement est que la liste des processus dans le conteneur, avec lesquels un processus peut donc interagir, n'est qu'un sous-ensemble de la liste totale des programmes en exécution.

Le noyau du système est le même (un seul noyau de système d'exploitation) pour toutes les conteneurs et le système hôte. Le système peut donc tout contrôler et optimiser simultanément. Il connait ce qui se passe dans chaque conteneur puisque c'est lui qui décide de tout.

Pour une meilleure compréhension des conteneurs, ou si vous vous demandez pourquoi nous parlons de ça si tôt dans ce TP, vous pouvez regarder et tester le module 2 du MOOC sur `https://fun-mooc.fr` « Reproducible Research II : Practices and tools for managing computations and data » [Arnaud Legrand, Konrad Hinsen, Christophe Pouzat et al]. Il est l'inspiration de ce TP. Il explique l'usage des conteneurs pour un public de scientifiques (pas uniquement pour des informaticiens, Konrad Hinsen est biophysicien moléculaire, Christophe Pouzat est mathématicien appliqué).

2. Promenade au pays des conteneurs : être dedans, dehors, ou les deux en même temps

Après Debian, utilisée dans le TP précédent, nous allons utiliser Alpine Linux, qui est une distribution visant le minimalisme et la grande qualité. Elle est l'une des distributions phare dans les nuages (Cloud).

2.1. Singularity

Les différentes technologies utilisent des formats d'image assez similaire à ce que vous avez fabriqué pour la machine virtuelle.

À l'Ensimag, sur les machines vous trouverez les conteneurs *singularity*. L'outil le plus connu est *docker*, car il vient avec son attirail d'images bien fourni. Nous utiliserons justement une image docker. Celle d'Alpine Linux. Elle ne sera téléchargée qu'une fois et gardé en cache dans votre HOME. Il faudra epnser à nettoyer ce cache à la fin de la séance.

2.2. Préparation

Ouvrir deux terminaux différents et le sujet. Comme terminal, nous conseillons de prendre deux fois `terminology` et de changer le thème ou la couleur dans l'un des deux pour bien les différencier (bouton droit de la souris pour faire apparaitre la configuration graphique de terminology dans "settings" puis "colors").

Arrangez les pour pouvoir lire les terminaux et le sujet simultanément. Une méthode rapide consiste à utiliser le tuilage (tiling) de Gnome 3 (Super (Windows) + ← ou → , ↑ , ↓)

2.3. Cas 1 : mise en conteneur en gardant votre HOME

Lancez dans l'un des deux terminaux Alpine Linux (image venant de docker.io) dans un conteneur singularity par défaut

```
singularity run docker://alpine:latest
```

Lancez les commandes suivantes, l'une après l'autre, dans les deux terminaux. La séquence de commandes essayent de montrer les différences entre « dedans » et « dehors ». Les commentaires indiquent si le résultat devrait être identique ou différents.

```
cd                          # identique
pwd                         # identique
ls                          # identique
ps                          # différent
ps -e                       # différent
ps | grep terminology       # différent
```

```
which ps                    # différent
echo $(which ps)            # différent
ls -l $(which ps)           # différent
ps --version                # différent
busybox ps                  # similaire
ls /                        # différent
ls -i /                     # différent
stat /                      # différent
df                          # différent
ls /user/<1 2 ou 3>/        # différent
```

Lancez la commande `exit` pour terminer le conteneur.

2.4. Cas 2 : remplacer les programmes standard et ne pas avoir accès à vos fichiers

Avec singularity viennent trois niveaux d'isolation. Celui par défaut, qui vous permet de manipuler les fichiers de votre véritable HOME, une isolation un peu plus forte avec l'option `-c` (contain) qui remplace votre HOME par un HOME vide dans le conteneur et une isolation encore plus forte avec l'option `-C` (contain all).

Lancez dans l'un des deux terminaux Alpine Linux (image venant de docker.io) dans un conteneur singularity avec une isolation très forte

```
singularity run -C docker://alpine:latest
```

Relancez la même séquence de commandes.

```
cd                          # identique
pwd                         # identique
ls                          # différent
ps                          # différent
ps -e                       # différent
ps | grep terminology       # différent
which ps                    # différent
echo $(which ps)            # différent
ls -l $(which ps)           # différent
ps --version                # différent
busybox ps                  # similaire
ls /                        # différent
ls -i /                     # différent
stat /                      # différent
df                          # différent
ls /user/<1 2 ou 3>/        # différent
```

Lancez la commande `exit` pour terminer le conteneur.

2.5. Nettoyer votre cache

Combien cela vous a-t-il coûté en place disque de télécharger un conteneur Alpine Linux ? Pas très cher, mais il est inutile de gacher votre précieux espace disque pour rien.

```
singularity cache list -v
singularity cache clean # et répondre Yes :-)
singularity cache list -v
```

3. Fabriquer une image en environnement contraint

Les machines de l'Ensimag ne sont pas vos machines. Vous n'avez pas de droits d'administrateurs dessus (sauf les Bug Busters). Vos droits sont limités pour vous empêcher de pouvoir changer la configuration d'une machine.

Pour ce TP, vous ne pouvez donc pas utiliser `singularity` pour construire une image sur une machine de l'Ensimag. Cela demande de vous ouvrir des droits que vous n'avez pas sur les machines de l'Ensimag.

Enfin, du moins, pas directement.

Dans une machine virtuelle, vous pouvez être administrateur, installer singularity, et construire l'image. Cette image pourra être copiée en utilisant SSH vers votre machine physique.

L'énorme avantage de cette image construite avec singularity est sa **reproductibilité**. Elle pourra être exécutée par n'importe quel singularity sur n'importe quel linux, indépendamment de ce qui y est installé. Vous pouvez donc facilement la passer à l'un de vos camarades fan de Debian, Arch, Manjaro, Fedora, Mint, Guix, NixOS, etc.

Les conteneurs docker et singularity ont une grosse limitation. Ils n'ont pas l'accès direct au matériel, pour éviter des soucis de sécurité, mais ce qui peut pose des soucis de configuration pour les applications graphiques ou audio.

Pour distribuer un logiciel de manière reproductible, il existe d'autres technologies que les conteneurs docker et singularity. Elles ont d'autres limitations. Snap (très utilisé dans Ubuntu), Flatpak, Guix et Appimage sont quatres exemples de canaux de distribution de logiciels (y compris audio et graphique). Firejail est un exemple du contraire : il prend un logiciel déjà installé chez vous, et l'exécute dans un conteneur avec les limitations que vous choisissez.

3.1. Installer Alpine linux dans une VM

Nous allons pouvoir installer Alpine avec une image plus petite. Il faut garder un peu de place pour singularity, puis pour pouvoir construire l'image. Une image de 1.5 Gio devrait suffire.

```
dd of=3mmunix_alpine.img bs=1536M seek=1 count=0
```

Nous n'avons pas vraiment besoin du réseau vers la VM (le réseau depuis la VM marche par défaut) si vous utilisez l'écran de qemu. Il faudra le rajouter si vous en avez le besoin.

Lancez `qemu` avec 8Gio de mémoire, comme cdrom `~jdpunix/iso/alpine-virt-3.20.2-x86_64.iso` et comme disque le nouveau fichier `3mmunix_alpine.img`.

Connectez-vous comme « root » (il n'y a pas de mot de passe). Attention le clavier est pour l'instant en QWERTY ! Lancez l'installation avec la commande `setup-alpine`. en QWERTY, le [a] est à la place du [q] et [-] sur la touche [)] ou utilisez le pavé numérique [1]

Après avoir choisi le clavier "fr" puis "fr" vous devriez être en AZERTY pour le reste de l'installation.

Les défauts devraient vous convenir pour le reste, sauf ATTENTION au disque. Il faut bien s'installer sur `sda`, avec une installation `sys`

À la fin de l'installation, il faut `reboot`er.

3.2. Installer singularity dans la VM

Reconnectez-vous comme root.

C'est une très mauvaise habitude de travailler comme root. Il est alors facile détruire son installation. Prenez grand soin aux commandes que vous tapez, au risque de devoir tout recommencer.

La commande de gestion des paquets est `apk`. Singularity est disponible dans les entre-pôts `community` de la distribution. Ils ne sont pas utilisés par défaut. Il faut donc changer le défaut. Pour cela il faut éditer le fichier de configuration `/etc/apk/repositories`. Pour éditer, soit vous utilisez vi (pas vim, vi), soit vous installer un des petits éditeurs disponibles de base (vim, nano, mg).

Il faut enlever le # (qui transforme la ligne en commentaire) en début de la ligne *community*.

```
# apk search singularity
# apk update # mettre à jour la liste de paquet disponible
# apk add vim nano mg # installer vim, nano et mg (clone petit
↪  d'emacs)
# vim /etc/apk/repositories # ou nano, ou mg suivant vos goûts
# apk update
# apk search singularity
....
# apk add singularity
```

1. Merci à Betty Holberton, pionnière, au sens propre, de l'Informatique, pour ce pavé numérique !.

3.3. Fabriquer une recette pour singularity et construire l'image

Il serait possible de lancer une image alpine dans le container singularity dans votre VM alpine, avec les droits d'écriture. Il faut alors de la modifier à la main, puis l'extraire du cache de singularity pour pouvoir l'envoyer à vos camarades.

En pratique, l'alternative utilisée est de fabriquer une recette, puis de demander à singularity de construire l'image correspondant à la recette. Il est alors trivial de partager les quelques lignes de recette.

3.4. Fabriquer l'image dans la VM à partir d'une recette

Nous allons mettre dans la recette lolcat, cowsay et fortune. Pour cela, il va faloir utiliser la branche `edge` d'Alpine et d'ajouter le reprétoire `testing` en plus de *main* et de *community*.

La recette est relativement simple. Néanmoins, il va donc falloir programmer le changement du fichier `/etc/apt/repositories`. pour cela nous allons utiliser deux outils de bases dans la trousse du programmeur sous GNU Linux/Unix : `sed` et `GNU awk`. Prennez un peu de temps un jour prochain pour faire quelques tutoriels de ces deux outils.

La recette est un simple fichier `moo_VOTRE_LOGIN.def` (exemple : `moo_arecoque.def`) à mettre dans la VM. Cela tombe bien, vous avez dejà choisi un éditeur. Mais vous pouvez aussi en rajouter un nouveau qui est dans la partie community (comme neovim, kakoune et emacs).

Le fichier aura le contenu suivant :

```
1  Bootstrap: docker
2  From: alpine:latest
3  Stage: build
4
5  %post
6    apk update && apk add gawk && sed -i 's/v3.20/edge/g'
     ↪ /etc/apk/repositories && gawk -i inplace '/community/ { print $1;
     ↪ gsub(/community/, "testing", $1);} {print}' /etc/apk/repositories
     ↪ && apk update && apk add fortune install cowsay
7
8  %labels
9    Author Alice Recoque <Alice.Recoque@grenoble-inp.org>
10   Version v1.0
```

La cuisson de votre délicieuse image est obtenu en demandant simplement à singularity (celui de votre vm) de

```
1  singularity build moo_arecoque.sif moo_arecoque.def
```

Vous obtenez une image `moo_arecoque.sif`, au format de singularity.

Il ne vous reste plus qu'à copier l'image vers votre HOME sur votre machine physique, avec `scp`.

Sur les machines de l'ensimag, vous pouvez donc lancer maintenant de nouvelles applications indispensables.

```
arecoque@ensipc$ ls -lh  moo_arecoque.sif
arecoque@ensipc$ singularity run moo_arecoque.sif fortune |
↪  singularity run moo_arecoque.sif cowsay | singularity run
↪  moo_arecoque.sif lolcat
arecoque@ensipc$ singularity shell moo_arecoque.sif
Singularity> fortune | cowsay | lolcat
Singularity> exit
arecoque@ensipc$
```

A. Solutions

```
# prendre deux terminaux visuellement différents
#
terminology &
terminology & # bouton droit, changer la coiuleur
# Conteneur: ceci n'est pas une VM
singularity run docker://alpine:latest
# comparez dedans et dehors
cd                          # identique
pwd                         # identique
ls                          # différent
ps                          # différent
ps -e                       # différent
ps | grep terminology       # différent
which ps                    # différent
echo $(which ps)            # différent
ls -l $(which ps)           # différent
ps --version                # différent
busybox ps                  # similaire
ls /                        # différent
ls -i /                     # différent
stat /                      # différent
df                          # différent
ls /user/<1 2 ou 3>/        # différent


# comparer dedans et dehors
singularity run -C docker://alpine:latest
# comparer dedans et dehors
```

```
  cd                          # identique
pwd                         # identique
ls                          # différent
ps                          # différent
ps -e                       # différent
ps | grep terminology       # différent
which ps                    # différent
echo $(which ps)            # différent
ls -l $(which ps)           # différent
ps --version                # différent
busybox ps                  # similaire
ls /                        # différent
ls -i /                     # différent
stat /                      # différent
df                          # différent
ls /user/<1 2 ou 3>/        # différent

# singularity cache
singularity cache list -v
singularity cache clean

# Alpine a singularity. On peut le mettre dans une VM (plus petite !)
# but construire une image sif, avec fortune, lolcat et cowsay et la télécharger
# sur la ubuntu pour la faire tourner.

# Les fichiers sont partiellement ou totalement différents
# Les processus sont imbriqués
dd of=3mmunix_vmdisk.img bs=2G seek=1 count=0
qemu-system-x86_64  -machine q35,accel=kvm -m 8G -cdrom ~jdpunix/alpine-virt-3.20.2-x86_64.iso -nic user,hostfwd=tcp:127.0.0.1:2222-:22 3mmunix_vmdisk.img

# dans alpine
# liste des paquets  https://pkgs.alpinelinux.org/packages
se connecter comme root (sans mot de passe)
setup-alpine
# le reste va ressembler à l'install de la Debian
# creer une autre utilisateur (alternative: autoriser root a se connecter en ssh avec un mot de passe)
# choisr le disque sda lorsque
# choisir une installation sys

```

```
# REBOOTER ! Sinon, vous changer uniquement le système en mode transitoire

apk add vim # ou utiliser nano
# remplacer dans le /etc/apk/repositories, dans la liste des miroirs, "v3.20" par "edge"
# edge permet d'avoir un singularity plus à jour
apk add emacs neovim helix kakoune # choisir le votre !
# bien ajouter les liens community en décommentant les lignes
apk update
apk upgrade
apk add singularity

# singularity maintenant permet de passer root, on va pouvoir éditer
# l'image singularity, la sauver en sif et la télécharger sur votre
# machine.

# Fabriquer une image pour faire tourner fortune cowsay et lolcat
# utilisation d'une sigularity recipes (venant du MOOC d'Arnaud Legrand)
# Bootstrap: docker
# From: alpine:latest
# Stage: build
#
# %post
#   apk update && apk add gawk && sed -i 's/v3.20/edge/g' /etc/apk/repositories && gawk -i inplace '/community/ { print $1; gsub(/community/, "testing", $1);} {print}' /etc/apk/repositories && apk update && apk add fortune install cowsay
#
# %labels
#   Author Alice Recoque <Alice.Recoque@grenoble-inp.org>
#   Version v1.0

# fabriquer le SIF (pas besoin de fakeroot, on est root)
singularity build moo_arecoque.sif moo_arecoque.def

# ramener le sif en dehors de

# et faire tourner les trois applications
singularity run moo_arecoque.sif fortune | singularity run moo_arecoque.sif cowsay | singularity run moo_arecoque.sif lolcat

```