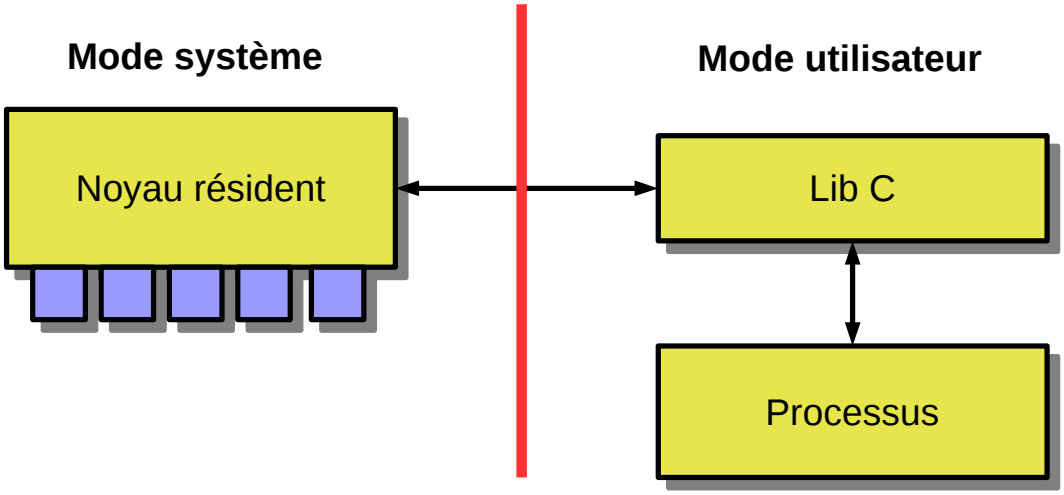
Des conteneurs au cloud : principes et administration

Table des matières

1 Caractéristiques du système UNIX

- Système **multi-utilisateurs simultanés**
  - ▷ connexion via la console, des terminaux, le réseau
  - ▷ authentification **forte** et **souple**
- Système **multitâches** (en temps partagé)
  - ▷ pour les architectures **multi-processeurs** et/ou **multi-coeurs**
  - ▷ pour les applications **multi-threads**
- Système **simple** et peu **consommateur de ressources**
  - ▷ aspect **brut** (pas de fioriture)
  - ▷ **rigoureux** et **minimaliste**
  - ▷ **boîte blanche** (tout est visible)
    - la position des objets est normalisé
    - les fichiers de configuration sont au format texte
    - les traces sont au format texte
- Système **portable** (90% à 98% écrit en langage C)
  - ▷ il initie les plateformes **ouvertes** pour contrer l'approche propriétaire
  - ▷ il permet la conception d'applications **portables** :
    - indépendance vis-à-vis d'un OS
    - indépendance vis-à-vis de l'architecture matérielle
- **Prolifération** des versions d'UNIX

1.1 Structure



Processus interactifs
- shells
- utilitaires

Processus en arrière-plan
- démons
- services

1.2 Processus UNIX

Un **processus UNIX** c'est :

- un numéro d'identification (PID) et le numéro du père (PPID)
- un répertoire courant (CWD)
- un code qui s'exécute et des zones de données (pile, tas)
- un tableau (`fd` pour *file descriptor*) des fichiers ouverts avec
  - `fd[0]` : entrée standard (stdin),
  - `fd[1]` : sortie standard (stdout),
  - `fd[2]` : sortie d'erreur standard (stderr),
- un ensemble de variables d'environnement

```
PATH=/bin:/usr/bin:/usr/local/bin
HOME=/home/alfred
UID=2000
...
```

La création d'un nouveau processus se réalise par duplication (`fork`) du processus courant.

```
$ export MSG="Hello"                shell père
$ printenv                            |
...                                 printenv
MSG=Hello                             |
..                                    |
$ bash                                |
```

```
$ echo $MSG                         shell fils
Hello                                 |
$ export MSG="Salut"                  |
$ echo $MSG                           |
Salut                                 |
$ exit                                |
```

```
$ echo $MSG                              shell père
Hello                                      |
$                                          |
```

Chaque processus possède un répertoire courant (hérité du père) :

```
$ cd /home; pwd; (pwd ; cd /var ; pwd) ; pwd
/home
/home
/var
/home
$
```

1.3 Historique

1969/70 : v1 en BCPL/B par K. Thompson et D. Ritchie sur PDP7

1972/73 : v2 en C sur PDP11

- **1980** : Berkley Software Distribution achète une licence et prépare la version **BSD 1.0** (association avec **Sun** et **IBM**)
  - ▷ **1988** : évolution vers **NEXTSTEP** (S. Jobs) et **1999** **MacOS X**
  - ▷ **Années 1990 à 2010** évolution vers **FreeBSD 9.x**, **OpenBSD** et **NetBSD 5.0**
- **1981** : AT&T achète Bell Labs et sort la version **System III** qui évolue en **System IV**
  - ▷ **Début 1980** : évolution vers **System V release 1.0**, puis **release 2.0**, puis **3.0**. Association avec **HP** (**HP/UX**), **IBM** (**Aix**) et **DEC** (**true64**). DEC est acheté depuis par Compaq puis par HP.
- **1988** : fusion des versions BSD et System V release 3.0 pour donner la version **System V release 4.0**
  - ▷ **1990** : Duplication du System V r4.0 qui devient **OSF1** (**Open Software Foundation**). Association de **DEC**, **HP** et **IBM**.

1987 : **Minix** de A. Tenenbaum (Unix Like pédagogique). Réécriture à partir de rien.

1991 : V0 de **Linux** de L. Torvalds (basé en partie sur **minix**), V1 en **1994**, V2 en **1996**, V3 en **2011** et V4 en **2015**. Apparition des **distributions** (assemblage d'un noyau Linux, des utilitaires GNU et de diverses applications).

- **Slackware**, **Debian**
- **RedHat** qui se divise en
  - ▷ Suse, Fedora
  - ▷ CentOS (versions 2, ..., 8 et 9)
    - Alma Linux (**celui que nous allons utiliser**)
    - Rocky Linux

Actuellement : BSD (Free/Net/Open), System V release 4.0, Linux

1.4 Arborescence

`/bin` commandes utilisées pendant la phase de démarrage (peu utilisé)

```
$ ls -l /bin
lrwxrwxrwx. 1 root root 7  7 juil. 21:27 /bin -> usr/bin
$
```

Notez le type `l` pour les liens symboliques.

`/boot` répertoire contenant les images du noyau Linux et le logiciel d'amorçage

```
$ ls /boot
config-3.10.0-229.20.1.el7.x86_64
grub
grub2
initramfs-0-rescue-1b39418727f14f13b7a4ebd58cf29827.img
initramfs-3.10.0-229.20.1.el7.x86_64.img
initrd-plymouth.img
symvers-3.10.0-229.20.1.el7.x86_64.gz
System.map-3.10.0-229.20.1.el7.x86_64
vmlinuz-0-rescue-1b39418727f14f13b7a4ebd58cf29827
vmlinuz-3.10.0-229.20.1.el7.x86_64
...
```

`/root` répertoire d'accueil de l'administrateur

`/sbin` répertoire des commandes destinées à l'administrateur

```
$ ls -l /sbin
lrwxrwxrwx. 1 root root 8  7 juil. 21:27 /sbin -> usr/sbin
```

`/tmp` répertoire de stockage temporaire

- Notez le type `d` (directory) et le **sticky bit** `t` dans les permissions.

```
$ ls -ld /tmp
drwxrwxrwt. 19 root root 4096 22 nov.  18:26 /tmp
```

- Il ne doit jamais être plein.
- Tous les utilisateurs peuvent écrire :

```
$ find $HOME -type f > /tmp/$$.mes.fichiers.txt
```

- Une meilleure version (avec `mktemp`) :

```
$ tmp=$(mktemp)
$ find $HOME -type f > $tmp
$ cat $tmp
$ rm $tmp
```

- Les fichiers ne survivent pas à un redémarrage.

`/dev` répertoire des **fichiers spéciaux** associés aux périphériques.

- Chaque fichier spécial **corresponds à un périphérique** : écrire/lire sur ce fichier revient à écrire/lire sur le périphérique.
- Les périphériques sont identifiés par un couple majeur/mineur.
- Notez le périphérique bloc (`b`), le majeur (`8`) et les permissions.

Un exemple pour les disques SCSI/SATA

```
$ ls -l /dev/sd*
brw-rw----. 1 root disk 8, 0 22 nov. 17:13 /dev/sda
brw-rw----. 1 root disk 8, 1 22 nov. 17:13 /dev/sda1
brw-rw----. 1 root disk 8, 2 22 nov. 17:13 /dev/sda2
brw-rw----. 1 root disk 8, 3 22 nov. 17:13 /dev/sda3
brw-rw----. 1 root disk 8, 16 22 nov. 17:13 /dev/sdb
brw-rw----. 1 root disk 8, 17 22 nov. 17:13 /dev/sdb1
```

- Ce répertoire est préparé à l'installation (avant) ou mis à jour dynamiquement (maintenant) par le démon `udevd`.

D'autres exemples de fichiers spéciaux :

- `/dev/tty*` Les terminaux virtuels (notez le `c` pour les périphériques caractères).

```
$ ls -l /dev/tty*
crw-rw-rw-. 1 root tty  5,  0 22 nov. 17:13 /dev/tty
crw--w----. 1 root tty  4,  0 22 nov. 17:13 /dev/tty0
crw--w----. 1 root tty  4,  1 22 nov. 17:13 /dev/tty1
crw--w----. 1 root tty  4,  2 22 nov. 17:13 /dev/tty2
...
```

Un exemple d'utilisation

```
$ tty
/dev/pts/2
$ ls -l /dev/pts/2
crw--w----. 1 massat tty 136, 2 22 nov. 19:00 /dev/pts/2
$ echo salut > /dev/pts/2
salut
$
```

Toujours autour des terminaux :

- `/dev/tty` le terminal associé au processus qui utilise ce fichier

```
$ echo salut > /dev/tty
salut
$
```

- `/dev/stdout` `/dev/stdin` `/dev/stderr` les entrées/sorties standards du processus qui utilise ces fichiers

```
$ ( echo réussite; echo echec > /dev/stderr ) 2> erreurs.txt
réussite
$ cat erreurs.txt
echec
```

```
$ ( echo réussite; echo echec > /dev/stderr ) 2>&1
réussite
echec
```

```
$ ( echo réussite; echo echec > /dev/stderr ) &> /dev/stdout
réussite
echec
```

- `/dev/ttyS*` les ports séries (à tester avec les consoles texte Ctrl+Alt+F1 à F4)

```
$ ls -l /dev/ttyS*
crw-rw----. 1 root dialout 4, 64 22 nov.  17:13 /dev/ttyS0
crw-rw----. 1 root dialout 4, 65 22 nov.  17:13 /dev/ttyS1
crw-rw----. 1 root dialout 4, 66 22 nov.  17:13 /dev/ttyS2
crw-rw----. 1 root dialout 4, 67 22 nov.  17:13 /dev/ttyS3
```

- `/dev/lp*` les ports parallèles

```
$ ls -l /dev/lp*
crw-rw----. 1 root lp 6, 0 22 nov.  17:13 /dev/lp0
crw-rw----. 1 root lp 6, 1 22 nov.  17:13 /dev/lp1
crw-rw----. 1 root lp 6, 2 22 nov.  17:13 /dev/lp2
crw-rw----. 1 root lp 6, 3 22 nov.  17:13 /dev/lp3
```

Les fichiers **très spéciaux** :

- `/dev/null` déversoir de données et contenu vide.

Exemple 1 : Vider un fichier

```
$ cat /dev/null > Un_fichier_vide.txt
```

Exemple 2 : supprimer des erreurs

```
$ une_commande_qui_genere_des_données_et_des_erreurs 2> /dev/null
```

- `/dev/zero` source sans fin de zéro binaire.

Videz un périphérique (dangereux)

```
# cat /dev/zero > /dev/sdb3      # à éviter
```

Construire un fichier (avec dd)

```
$ dd if=/dev/zero of=Un_fichier_de_10mo bs=1M count=10
```

`/etc` répertoire des fichiers de configuration. Quelques exemples

- `/etc/passwd` base de données des utilisateurs et `/etc/group` base de donnes des groupes.
- `/etc/sysconfig/*` fichiers de configuration de RedHat.
- `/etc/login` script exécuté par `csh` (`/usr/bin/tcsh` sous Linux) au démarrage d'une session utilisateur
- `/etc/profile` script exécuté par `sh` (`/usr/bin/bash` sous Linux) au démarrage d'une session utilisateur
- `/etc/profile.d/*.{sh,csh}` autres scripts exécutés par `sh` ou `csh` au démarrage d'une session.

`/home` comptes utilisateurs (localisés ou accessibles)

`/lib` les librairies (au sens large), librairies dynamiques

- `/lib/modules` modules du noyau Linux

`/usr` un point d'installation des logiciels

- `/usr/{bin,sbin,lib}` exécutables et librairies
- `/usr/include` fichiers `.h` à inclure
- `/usr/share` fichiers partagés indépendants de l'architecture
  - ▷ `/usr/share/info` page d'info (commande `info`)
  - ▷ `/usr/share/doc` documentation
  - ▷ `/usr/share/man` page de manuel (commande `man`)
    - `/usr/share/man/man1` commandes
    - `/usr/share/man/man2` appels système

    - `/usr/share/man/man3` fonctions de librairie
    - `/usr/share/man/man4` fichiers spéciaux
    - `/usr/share/man/man5` format de fichier
    - `/usr/share/man/man8` commandes administrateur
- `/usr/X11R6/{bin,sbin,lib,...}` installation des logiciels graphiques
- `/usr/local/{bin,sbin,lib,...}` installation des logiciels supplémentaires

`/var` partie variable du système

- `/var/tmp` espace temporaire permanent
- `/var/lib` stockage des données applicatives
    - ▷ `/var/lib/mysql` les BD MySQL
    - ▷ `/var/lib/php` les sessions PHP
- `/var/run` informations sur les processus en cours d'exécution ou les utilisateurs connectés

```
$ cat /var/run/sshd.pid
1177
$ ps aux | grep 1177
root  1177  0.0  0.0  82508  3596 ?  Ss   12:57   0:00 /usr/sbin/sshd -D
$
```

- `/var/cache` cache des applications
    - ▷ `/var/cache/man` les pages déjà formatées
    - ▷ `/var/cache/dnf` les packages déjà téléchargés
- `/var/log` traces de l'activité du système et des démons
    - ▷ `/var/log/messages` messages du noyau
    - ▷ `/var/log/maillog` activité SMTP
    - ▷ `/var/log/security` alertes de sécurité
    - ▷ `/var/log/boot.log` message du démarrage
- `/var/spool`
    - ▷ `/var/spool/mail` boîtes aux lettres
    - ▷ `/var/spool/lp` spooler d'impression

`/proc` pseudo répertoire de publication des informations du noyau

- `/proc/numéro_PID/cmdline` ligne de commande
- `/proc/numéro_PID/environ` variables d'environnement

```
$ cat /proc/$$/cmdline
bash
$
```

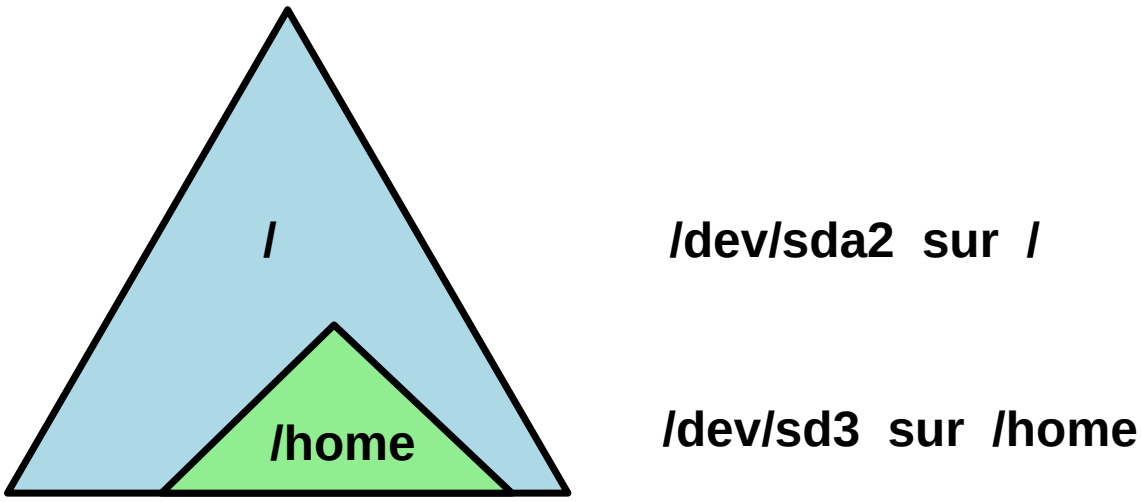
```
$ AA=aa cat /proc/self/environ
AA=aa
SHELL=/bin/bash
USER=massat
LANG=fr_FR.utf8
LOGNAME=massat
HOME=/home/massat
PATH=/usr/local/bin:/usr/bin
MAIL=/var/mail/massat
$
```

1.5 Points de montage

Montage : C'est une association entre un répertoire (le point de montage) et un périphérique de type bloc (une partition).

Sur un **poste de travail** :

- `/` obligatoire
- `/home` pour séparer les données des utilisateurs



Sur un **serveur** :

- `/` obligatoire
- `/home` pour séparer les données des utilisateurs
- `/var` par sécurité, pour placer des quotas, pour un disque rapide
- `/boot` pour faciliter les accès
- `/tmp` pour limiter la croissance et placer des quotas

1.6 Package RedHat

Les packages RedHat (fichiers `.rpm`) sont gérés par la commande `rpm` :

- `rpm -i nedit-5.5-23.el6.x86_64.rpm` installation
- `rpm -U nedit-5.5-23.el6.x86_64.rpm` installation ou mise à jour
- `rpm -q nedit` interrogation
- `rpm -q -l nedit` interrogation (version longue)
- `rpm -q -i -l nedit` interrogation (version très longue)
- `rpm -e nedit` suppression

- `rpm -V nedit` vérification des signatures

Un package RPM c'est

- des fichiers (avec une signature) codés avec `cpio`
- des dépendances (offertes ou nécessaires)
- des scripts de pre/post install/uninstall/update
- une description

Rechercher des informations :

```
$ ps aux                          # j'utilise la commande ps
...

$ type ps                         # où est la commande ps
ps est /usr/bin/ps

$ rpm -qf /usr/bin/ps             # à quel package appartient-elle ?
procps-ng-3.3.10-3.el7.x86_64

$ rpm -ql procps-ng               # quoi d'autre dans ce package ?
/usr/bin/ps
/usr/bin/top
/usr/share/man/man1/ps.1.gz
...
```

1.7 Mise à jour du système

La **mise à jour** implique

- la recherche et le chargement d'une nouvelle version (RPM),
- la mise à jour (`rpm -Uvh fichier.rpm`)

La commande `dnf` réalise toutes ces opérations

- `dnf list *nedit*` pour chercher un package
- `dnf install nom_de_package` chargement et installation
- `dnf remove nom_de_package` suppression
- `dnf update` recherche des mises à jour, chargement et application
- `dnf info nom_de_package`
- `dnf clean all` suppression des caches
- `/etc/dnf/` fichiers de configuration de dnf
- `/var/cache/dnf` lieu de stockage des RPM
- `/var/log/dnf.log` trace des opérations

La mise à jour

- **ne modifie pas** les fichiers de configuration,
- ajoute l'extension `.rpmnew` aux nouveaux fichiers de configuration
- sauvegarde les fichiers de configuration modifiés avec l'extension `.rpmsave`

2 Démarrage du système

Le démarrage d'un système de type UNIX se déroule en cinq étapes :

- Démarrage de la machine et chargement d'un logiciel d'amorçage
- Exécution du logiciel d'amorçage (GRUB) qui se charge du chargement du noyau

- Exécution du noyau et initialisation des périphériques
- Création du premier processus (`init` ou `systemd`)
- Activation des services

2.1 Démarrage du BIOS

- Sur les ordinateurs de type PC :
  - ▷ quatre **partitions principales**
  - ▷ un partition principale peut être découpée en **partitions étendues**
- Chargement du MBR (**Master Boot Record**) secteur de boot du disque
- ou recherche d'une partition active (**bootable**) puis
  - ▷ chargement du secteur de boot
  - ▷ lancement du logiciel d'amorcage

2.2 Démarrage de GRUB

GRUB (**GRand Unified Bootloader**) est un logiciel d'amorcage généraliste (il remplace **LiLo** : **Linux Loader**).

- Propose plusieurs options de démarrage (menu)
- Permet de modifier dynamiquement les options de démarrage
- Assure le chargement du noyau
- Prévoie le passage des paramètres au noyau

GRUB est configuré par `/etc/grub2.conf` (lien vers `/boot/grub2/grub.conf`) :

```
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub2-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#

### BEGIN /etc/grub.d/10_linux ###
menuentry 'CentOS Linux (3.10.0-327.el7.x86_64) 7 (Core)' ... {
    ...
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos5'

    ...

    linux16 /boot/vmlinuz-3.10.0-327.el7.x86_64
       root=UUID=2a7d55d9-1331-448b-a64e-ec8203d1659d ro crashkernel=auto
       rhgb quiet LANG=fr_FR.UTF-8 systemd.debug
    initrd16 /boot/initramfs-3.10.0-327.el7.x86_64.img
}
### END /etc/grub.d/10_linux ###
```

Quelques commandes utiles :

- `grub2-mkconfig -o /boot/grub2/grub.conf` pour refaire le fichier de configuration

- `grub2-install /dev/sda2` pour installer GRUB dans la deuxième partition d'un premier disque

2.3 Démarrage de Noyau

- Montage du disque RAM (`initrd`)
- Initialisation des périphériques (commande `lspci`)
- Montage de la racine (lecture seule)
- Lancement du premier processus (PID 1) :
  - ▷ `/sbin/init` ou `/usr/lib/systemd/systemd`

Les messages du noyau au démarrage :

```
$ dmesg
...
[    0.000000] Linux version 3.10.0-327.el7.x86_64 (...
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-3.10.0-327.el7.x86_64 ...
...
[    0.000000] DMI: Dell Inc. Precision WorkStation T5500  ...
...
[    0.654973] pci 0000:3f:00.0: [8086:2c70] type 00 class 0x060000
[    0.655006] pci 0000:3f:00.1: [8086:2d81] type 00 class 0x060000
...
[    1.234239] systemd[1]: systemd 219 running in system mode. ...
...

$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-3.10.0-327.el7.x86_64 ...
```

2.4 systemd

Systemd (qui remplace `/sbin/init` des systèmes V) assure (via des fichiers de configuration) **l'initialisation du système** et la mise en place des **services**

Systemd est organisé en unités :

```
$ systemctl list-units
...
cups.socket     loaded active running   CUPS Printing Service Sockets
home.mount      loaded active mounted   /home
crond.service   loaded active running   Command Scheduler
sshd.service    loaded active running   OpenSSH server daemon
...
```

il existe plusieurs types d'unité :

- `socket` : service accessible par un socket UNIX
- `service` : pour un démon
- `mount` : pour un point de montage
- `target` : pour définir un niveau de service (regroupement d'unités)
- ...

Le répertoire `/usr/lib/systemd/system` contient un fichier pour chaque **unité**.

```
$ ls -l /usr/lib/systemd/system/cups*
-r--r--r--. 1 root root 126 20 nov.  16:07 /usr/lib/systemd/system/cups.path
-r--r--r--. 1 root root 198 20 nov.  16:07 /usr/lib/systemd/system/cups.service
-r--r--r--. 1 root root 131 20 nov.  16:07 /usr/lib/systemd/system/cups.socket
```

Un autre exemple pour le serveur `ssh` :

```
$ rpm -ql openssh-server
...
/usr/lib/systemd/system/sshd-keygen.service
/usr/lib/systemd/system/sshd.service
/usr/lib/systemd/system/sshd.socket
...
```

Le fichier `/usr/lib/systemd/system/sshd.service` :

```
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process

[Install]
WantedBy=multi-user.target
```

Arrêter ou démarrer un service :

```
systemctl stop  sshd.service
systemctl start sshd.service
```

Rendre le démarrage automatique (ou pas) :

```
systemctl enable  sshd.service
systemctl disable sshd.service
```

Questionner un service :

```
systemctl status     sshd
systemctl is-enabled sshd
```

Tuer un service :

```
systemctl kill  sshd
```

Les unités sont regroupés dans des **niveaux** ou **cibles** (des unités `target`) :

Au démarrage, le démon `systemd` va traiter le répertoire `/etc/systemd/` et atteindre la cible `/etc/systemd/system/default.target`

```
$ cd /etc/systemd/system
$ ls default.target
total 12
lrwxrwxrwx. 1 root root   36 28 août  12:26 default.target ->
     /lib/systemd/system/graphical.target
```

Si nous explorons le répertoire `/usr/lib/systemd/system`

```
$ cd /usr/lib/systemd/system
$ ls -l *.target
...
-rw-r--r--. 1 root root 552 20 nov.  05:49 poweroff.target
-rw-r--r--. 1 root root 543 20 nov.  05:49 reboot.target
-rw-r--r--. 1 root root 486 20 nov.  05:49 rescue.target
-rw-r--r--. 1 root root 492 20 nov.  05:49 multi-user.target
-rw-r--r--. 1 root root 558 20 nov.  05:49 graphical.target
...
lrwxrwxrwx. 1 root root  16  2 déc.  06:53 default.target -> graphical.target
...
lrwxrwxrwx. 1 root root  15  2 déc.  06:53 runlevel0.target -> poweroff.target
lrwxrwxrwx. 1 root root  13  2 déc.  06:53 runlevel1.target -> rescue.target
lrwxrwxrwx. 1 root root  17  2 déc.  06:53 runlevel2.target -> multi-user.target
lrwxrwxrwx. 1 root root  17  2 déc.  06:53 runlevel3.target -> multi-user.target
lrwxrwxrwx. 1 root root  17  2 déc.  06:53 runlevel4.target -> multi-user.target
lrwxrwxrwx. 1 root root  16  2 déc.  06:53 runlevel5.target -> graphical.target
lrwxrwxrwx. 1 root root  13  2 déc.  06:53 runlevel6.target -> reboot.target
```

- niveau `0` : arret de la machine (ou redémarrage pour `6`)
- niveau `1` : mode dépannnage (**single user**)
- niveau `3` : système fonctionnel en mode texte (`5` pour le graphique)

Connaitre ou choisir la cible par défaut :

```
systemctl get-default
systemctl set-default multi-user.target
```

Activer une cible (peu utile) :

```
systemctl isolate runlevel0.target
```

C'est le travail effectué par les commandes `reboot`, `poweroff` et `halt` :

```
$ ls -l /usr/sbin/reboot /usr/sbin/poweroff /usr/sbin/halt
lrwxrwxrwx. 1 root root 16  2 déc.  06:53 /usr/sbin/halt     -> ../bin/systemctl
lrwxrwxrwx. 1 root root 16  2 déc.  06:53 /usr/sbin/poweroff -> ../bin/systemctl
lrwxrwxrwx. 1 root root 16  2 déc.  06:53 /usr/sbin/reboot   -> ../bin/systemctl
```

3 Gestion des disques

3.1 Volumes UNIX

Un **volume UNIX** c'est une table des **i-nodes** et des blocs de données ou d'index sur le disque.

Chaque **fichier** (de données ou répertoire) est réprésenté par un **i-node**. Lui même est identifié par sa position dans la table :

```
$ touch mon_fichier.txt
$ ls -li mon_fichier.txt
641265 -rw-r--r--. 1 root root 0  3 déc.  15:58 mon_fichier.txt
$ cp -l mon_fichier.txt le_meme.txt
$ ls -li le_meme.txt
641265 -rw-r--r--. 2 root root 0  3 déc.  15:58 le_meme.txt
$
```

Le **répertoire racine** d'un volume a toujours le numéro **deux** (**128** sur XFS) :

```
$ ls -lid / /home /extra
128 dr-xr-xr-x. 18 root   root   4096  3 déc.  06:12 /
  2 drwx------. 10 root   root   4096 11 sept. 09:44 /extra
  2 drwxr-xr-x.  5 root   root   4096 28 août  16:03 /home
```

Le **formatage** passe par l'utilitaire `mkfs` :

```
# mkfs -t ext4 /dev/sd4
```

Cette commande effectue une redirection vers une commande liée au codage choisi : `mkfs.ext2`, `mkfs.ext3`, `mkfs.xfs`, etc.

La **vérification** d'un volume passe par la commande `fsck`. Cette commande analyse

- la table des i-nodes
- le codage physique des fichiers
- la structure des répertoires

Les **fichiers récupérés** sont placés dans le répertoire `lost+found` à la racine du volume.

3.2 Montage

Un **montage** est une association entre un **répertoire** et un **fichier spécial** (donc un périphérique).

```
# mkdir /tmp/mon-disque
# mount /dev/sdc3 /tmp/mon-disque
# ls -l /tmp/mon-disque
...
le contenu du disque
...
# umount /dev/sdc3
# ls -l /tmp/mon-disque
#
```

une fois le **montage effectué**, le contenu du périphérique est accessible via le répertoire. Les montages actifs sont visibles avec

```
# mount
/dev/sda5 on /     type xfs  (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda6 on /home type ext3 (rw,relatime,seclabel,data=ordered)
```

A chaque montage est associé un jeu d'options (voir `man mount`).

Au démarrage **systemd** effectue les montages listés dans le fichier `/etc/fstab` :

```
# cat /etc/fstab
UUID=2a7d55d9-1331-448b-a64e-ec8203d1659d   /        xfs     defaults   0 0
UUID=c56e95b4-10d5-4717-a4f4-6f843b0ea2fc   /extra   ext3    defaults   1 2
UUID=caac63b3-a40b-4e07-bc42-443bab351803   /home    ext3    defaults   1 2
UUID=a0928620-e279-4ac7-9b38-5863ee21e607   swap     swap    defaults   0 0
#
```

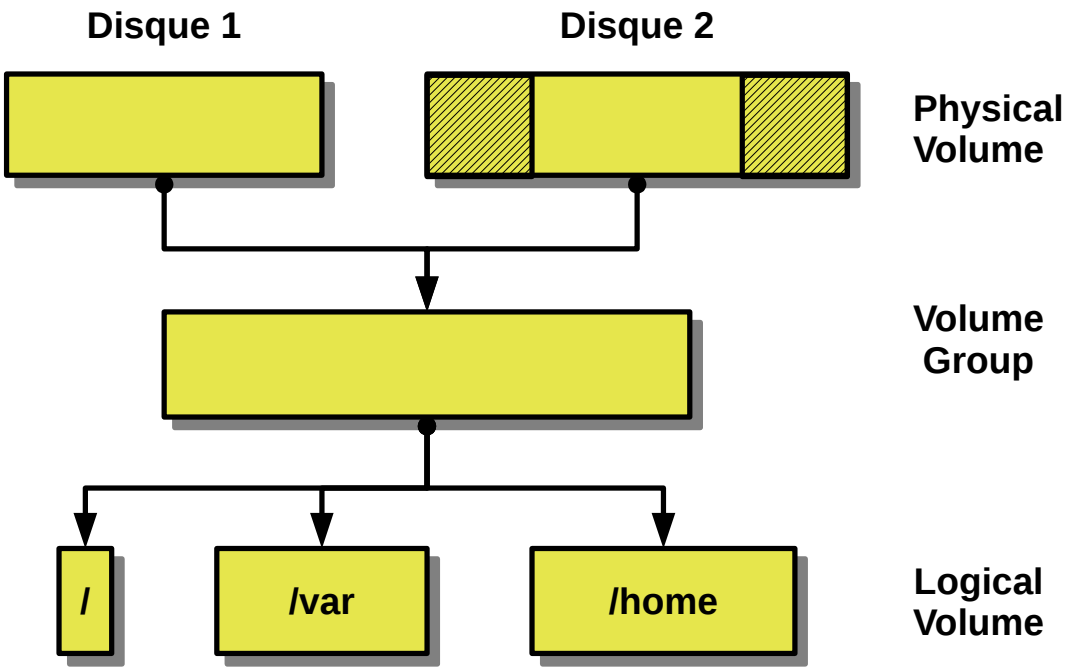
Dans cet exemple, les périphériques sont identifiés par leur UUID (**Universally Unique Identifier**) :

```
# ls -l /dev/disk/by-uuid/
total 0
lrwxrwxrwx.  ...  2a7d55d9-1331-448b-a64e-ec8203d1659d -> ../../sda5
lrwxrwxrwx.  ...  a0928620-e279-4ac7-9b38-5863ee21e607 -> ../../sdb1
lrwxrwxrwx.  ...  c56e95b4-10d5-4717-a4f4-6f843b0ea2fc -> ../../sdb2
lrwxrwxrwx.  ...  caac63b3-a40b-4e07-bc42-443bab351803 -> ../../sda6
#
```

3.3 Quota

Nous verrons ce point en travaux pratiques.

3.4 LVM : Logical Volume Manager



Commandes associées :

- Création : `{pv,vg,lv}create`
- Liste : `{pv,vg,lg}display` / `pvs` / `vgs` / `lvs` / `{pv,vg,lv}scan`
- Suppression : `{pv,vg,lv}remove`
- Modification d'un PV : `pvmove`
- Modification d'un VG : `vg{extend,reduce,merge}`
- Modification d'un LV : `lv{change,extend,reduce}`

Les opérations peuvent se dérouler **en même temps** que l'utilisation.

4 Définition des utilisateurs

4.1 Les fichiers

Les utilisateurs sont définis dans `/etc/passwd`.

Un exemple d'exploration

```
$ id
uid=1001(massat) gid=1001(massat) groupes=1001(massat)
$ grep massat /etc/passwd
massat:x:1001:1001:Jean-Luc Massat:/home/massat:/bin/bash
```

Les champs sont

- le login (8 caractères historiquement)
- le mot de passe crypté (DES, SHA, MD5)
- le UID (un entier)
- le GID (un entier pour identifier le groupe principal)
- le nom GECOS
- le répertoire d'accueil
- le shell utilisé

La liste des **shells autorisés** se trouve dans `/etc/shells`. La liste des **terminaux autorisés pour root** se trouve dans `/etc/securetty`.

Les groupes sont définis dans `/etc/group`. Un exemple :

```
massat:x:1001:user5,user9
```

Les champs sont

- le groupe (8 caractères historiquement)
- le mot de passe crypté (DES, SHA, MD5)
- le GID (un entier)
- la liste des logins des utilisateurs pour lesquels ce groupe est supplémentaire (séparés par des virgules)

Dans cette exemple, il existe un groupe pour chaque utilisateur (configuration de RedHat).

4.2 Les commandes

Commandes pour l'administrateur :

- `useradd` création d'un utilisateur (création du répertoire d'accueil à partir du modèle `/etc/skel`)
- `userdel` suppression d'un utilisateur
- `usermod` modification un utilisateur
- `groupadd` / `groupdel` création/suppression d'un groupe

Commandes pour l'administrateur et les utilisateurs :

- `passwd` changement du mot de passe (sans confirmation pour **root**)
- `chfn` changement du nom
- `chsh` changement du **shell**
- `id` identité (avec un paramètre éventuel)

Un exemple de création :

```
# useradd essai
# groupadd m2idl
# usermod essai -G m2idl
# id essai
uid=1002(essai) gid=1002(essai) groupes=1002(essai),1003(m2idl)
#
# passwd essai
Nouveau mot de passe : ...
Retapez le nouveau mot de passe : ...
#
# chsh -s /bin/tcsh essai
# su - essai
$ pwd
/home/essai
$ exit
#
```

4.3 Les mots de passe masqués

Le fichier `/etc/passwd` est lisible par tous les utilisateurs. Le **cryptage du mot de passe** n'est pas suffisant. Par sécurité, les mots de passe sont délocalisés dans `/etc/shadow`. ce fichier n'est pas lisible.

```
massat:mot-de-passe-crypté:16623:0:99999:7:::
```

Les champs sont

- le login
- le mot de passe crypté (DES, SHA, MD5)
- age du mot de passe (en secondes depuis 1970)
- âge minimum du mot de passe
- âge maximum du mot de passe
- ... voir `man 5 shadow`

Ces informations sont modifiables par la commande `chage`.

4.4 Le processus de Login

- `init` ou `systemd` assure la gestion de la bannière de login (fichier `/etc/issue`)
- `/usr/bin/login` assure la phase d'authentification :
  - ▷ utilisation de `/etc/{passwd,shadow,group}`
  - ▷ vérifier `/etc/nologin` qui doit être vide (sinon pas de login)
  - ▷ vérifier `/etc/securetty` (pour **root**)
  - ▷ afficher `/etc/motd` (message de bienvenue)
  - ▷ traces :
    - utilisateurs connectés : `/var/run/utmp` (à exploiter avec `w` ou `who`)
    - liste des connections : `/var/log/wtmp` (à exploiter avec `last`)
    - dernières connections : `/var/log/lastlog` (à exploiter avec `lastlog`)
  - ▷ Exécution du Shell de l'utilisateur : **le login shell**

4.5 Le shell de login

- Pour le **C-Shell** :
  - ▷ exécution de `/etc/csh.cshrc` (pour tous les shells)
  - ▷ exécution de `/etc/csh.login` (pour les shells de login)
    - exécution des scripts `/etc/profile.d/*.csh`
  - ▷ exécution de `$HOME/.cshrc` (tous les shells)
  - ▷ exécution de `$HOME/.login` (les login shells)
- Pour le **Bourne-Shell** :
  - ▷ exécution de `/etc/bashrc` (pour les `bash` sauf login)
  - ▷ exécution de `/etc/profile` (pour les shells de login)
    - exécution des scripts `/etc/profile.d/*.sh`
  - ▷ exécution de `$HOME/.bashrc` (pour les `bash` sauf login)
  - ▷ exécution de `$HOME/.profile` (les login shells)

4.6 Pluggable Authentication Modules (PAM)

Depuis 1995 l'authentification par fichiers est remplacée par une librairie accompagnée de **modules** (**PAM**). Le but est **d'assouplir** le paramétrage.

PAM prend en charge quatre opérations :

- `account` : vérification de la validité d'un compte
- `auth` : attestation d'identité
- `session` : préparation de la session utilisateur
- `password` : mise-à-jour du mot de passe

PAM est configuré par `/etc/pam.conf` ou une série de fichiers dans `/etc/pam.d`.

Étudions `/etc/pam.d/system-auth` :

```
# définiton des variables d'environnement
auth        required      pam_env.so
# si authentification par empreinte digitale, c'est OK
auth        sufficient    pam_fprintd.so
# si authentification UNIX, c'est OK
auth        sufficient    pam_unix.so nullok try_first_pass
# réussite sous condition
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
# échec sinon
auth        required      pam_deny.so

...

# vérifier la qualité
password    requisite     pam_pwquality.so try_first_pass local_users_only
# réussite si le mot de passe est modifié (notez le cryptage et shadow)
password    sufficient    pam_unix.so sha512 shadow nullok ...
# échec sinon
password    required      pam_deny.so

...
```

Ces fichiers sont modifiés par `/usr/sbin/authconfig` (ou `authconfig-tui`).

5 Configuration du réseau

5.1 Configuration ethernet

Les cartes sont détectés par le noyau :

```
# lspci | fgrep -i ethernet
06:00.0 Ethernet controller: Broadcom Corporation NetXtreme ...
#
```

Trace de la détection de la carte réseau :

```
# journalctl | fgrep 06:00
...
kernel: tg3 0000:06:00.0 eth0: Tigon3 [BCM95761) MAC address b8:ac:6f:4a:22:a2
kernel: tg3 0000:06:00.0 eth0: attached PHY is 5761 (10/100/1000Base-T Ethernet)
kernel: tg3 0000:06:00.0 eth0: dma_rwctrl[76180000] dma_mask[64-bit]
kernel: tg3 0000:06:00.0 enp6s0: Link is up at 1000 Mbps, full duplex
...
```

État des cartes réseau : commande `/sbin/ifconfig`

```
# ifconfig
enp6s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 139.124.5.44  netmask 255.255.255.0  broadcast 139.124.5.255
        inet6 fe80::baac:6fff:fe4a:22a2  prefixlen 64  scopeid 0x20<link>
        ether b8:ac:6f:4a:22:a2  txqueuelen 1000  (Ethernet)
        ...

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        ...

...
```

ou `ip` :

```
# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 ...
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
    link/ether b8:ac:6f:4a:22:a2 brd ff:ff:ff:ff:ff:ff
```

La configuration manuelle d'une carte utilise la commande `/sbin/ifconfig` :

```
ifconfig CARTE ADRESSE-IP netmask MASQUE-RESEAU
```

Par exemple :

```
ifconfig enp6s0 139.124.14.26 netmask 255.255.255.0
```

5.2 Configuration de la RedHat

Dans la distribution **RedHat**, la configuration du réseau est rangée dans : `/etc/sysconfig/network` et `/etc/sysconfig/network-scripts/ifcfg*`

```
# cd /etc/sysconfig/network-scripts/
# ls -l ifcfg-*
-rw-r--r--. 1 root root 294 28 août  12:26 ifcfg-enp6s0
-rw-r--r--. 1 root root 254 16 sept. 13:51 ifcfg-lo
#
# cat ifcfg-lo
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
BROADCAST=127.255.255.255
ONBOOT=yes
NAME=loopback
#
# cat ifcfg-enp6s0
TYPE=Ethernet
BOOTPROTO=dhcp
NAME=enp6s0
UUID=f7dbb30f-9a50-4a33-b52f-36a53269fb79
DEVICE=enp6s0
ONBOOT=yes
...
```

5.3 Manipulation des routes

L'interrogation et le changement des routes (commande `route`) :

```
# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
default         gateway         0.0.0.0         UG    100    0        0 enp6s0
139.124.5.0     0.0.0.0         255.255.255.0   U     100    0        0 enp6s0
```

Ajout d'une route :

```
# route add -net 192.168.0.0 netmask 255.255.0.0 dev enp6s0
# route
...
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 enp6s0
```

Ajoutez une route par défaut pour utiliser une passerelle :

```
# route add default gw 139.124.5.250
```

Pour supprimer une route `route del ADRESSE`

5.4 Autres commandes

- `netstat` : obtenir des informations sur
  - ▷ les routes (`-r`),
  - ▷ les statistiques (`-i`),
  - ▷ les connexions TCP (`-t`)
- `arp` : obtenir des informations sur la table ARP (correspondance entre adresses MAC et adresses IP)
- `tcpdump` : suivre l'activité réseau

```
tcpdump -i enp6s0 host sol.dil.univ-mrs.fr
```

- `nmap` : quels sont les ports ouverts sur une machine ?

```
nmap sol.dil.univ-mrs.fr
```

5.5 Les fichiers de nommage

Correspondance nom de machine / adresse IP : `/etc/hosts`

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

Correspondance nom de réseau / adresse IP : `/etc/networks`

```
default 0.0.0.0
loopback 127.0.0.0
link-local 169.254.0.0
```

Correspondance nom de service / numéro de port : `/etc/services`

```
...
http    80/tcp    www www-http    # WorldWideWeb HTTP
http    80/udp    www www-http    # HyperText Transfer Protocol
...
```

5.6 Nommage du système

Quelques commandes :

- `uname -a` : nom de la machine (et autre informations)
- `hostname` : nom de la machine
- `hostname nom-de-machine` : changer le nom de la machine
- `hostnamectl` : interroger/contrôler le nom de la machine (**systemd**)

5.7 Utiliser le DNS

Configuration du client DNS : `/etc/resolv.conf`

```
# Generated by NetworkManager
search lidil.univ-mrs.fr
nameserver 139.124.5.132
nameserver 139.124.5.131
```

pour interroger le DNS :

```
$ host www.dil.univ-mrs.fr
www.dil.univ-mrs.fr is an alias for sol.dil.univ-mrs.fr.
sol.dil.univ-mrs.fr has address 139.124.14.122
```

Choisir entre le DNS et le fichier `/etc/hosts` : `/etc/nsswitch.conf`

```
hosts:      files dns myhostname
networks:   files
services:   files sss
...
passwd:     files sss
```

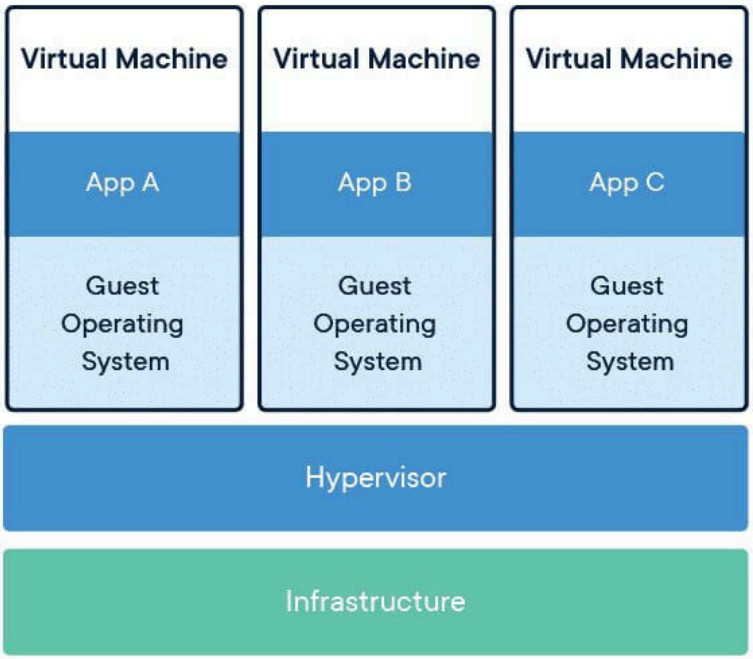
6 Les conteneurs

Objectifs : Il faut un moyen, simple, portable et paramétrable pour déployer des services de manière automatisée.

- L'installation **directe** est une solution compliquée.
- Un script peut faciliter l'installation.
- On doit donc livrer une procédure en plus du produit.

6.1 Déployer avec des VMs

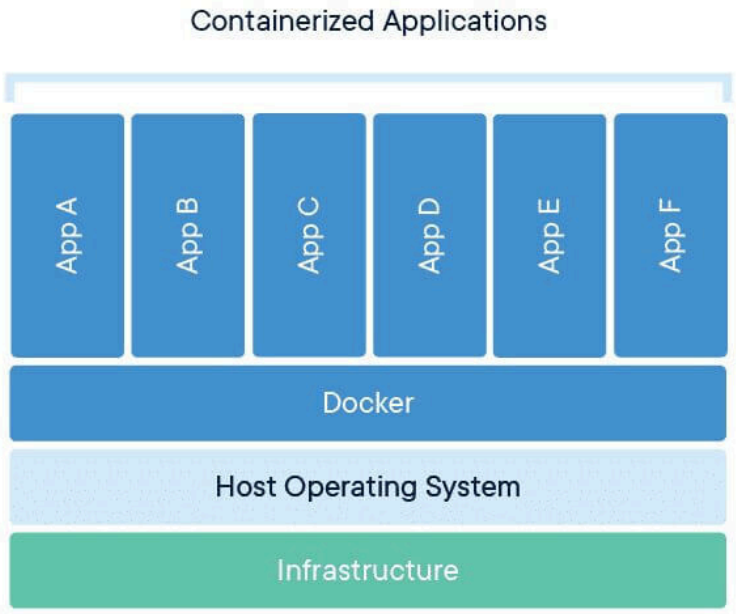
- Le montage et l'exportation d'une machine virtuelle est une solution pour packager le produit et la procédure d'installation.



Inconvénients. C'est une solution complexe mais automatisable. Elle reste néanmoins gourmande en ressources.

6.2 Déployer avec des conteneurs

Définition. Un **conteneur** est un espace qui
- accueille le déploiement d'un logiciel,
- isole les processus (indépendance des applications),
- alloue les ressources (découpage et arbitrage),
- émule les appels au noyau (indépendance vis-a-vis du système).



6.3 Conteneurs versus Machines virtuelles

Machines virtuelles :

- Isolation au niveau matériel
- Plusieurs OS

- Démarrage en minutes
- Taille en Giga-octets
- Utilisent beaucoup de ressources

- Sont déplaçables
- Les VM prêtes sont rares

Conteneurs :

- Isolation au niveau OS
- Partage d'un OS

- Démarrage en secondes
- Taille en Méga-octets
- Utilisent peu de ressources

- Facile a créer
- Les images pré-construites sont courantes

Note : Les VM et les conteneurs sont complémentaires. Il arrive souvent que des conteneurs soient déployés dans des machines virtuelles afin d'exploiter un serveur physique de forte capacité.