

Chapitre 5: Administration des systèmes

INF1070

Utilisation et administration des systèmes informatiques

Jean Privat & Alexandre Blondin Massé

Université du Québec à Montréal

Hiver 2021

Plan

- 1 Processus
- 2 Services et démons (*daemon*)
- 3 Installation
- 4 Démarrage et services
- 5 Configuration
- 6 Naviguer sur le *web*
- 7 Cybersécurité
- 8 Multiplexeur de terminaux et sessions
- 9 Docker

Administration système

L'**administrateur système** est la personne **responsable**

- des serveurs et postes informatique
- au niveau **logiciel**

Tâches habituelles

- Installation et désinstallation des logiciels
- Configuration des logiciels
- Mise-à-jour des logiciels
- Supervision des services logiciels

Autres tâches habituelles

- Configuration des disques, organisation des partitions, etc.
- Gestion des sauvegardes et des restaurations (*backups*)
- Configuration des périphériques (écrans, claviers, souris, etc.)
- Ajout et suppression des utilisateurs
- Configuration des comptes utilisateurs
- Configuration du réseau
- Responsable de la sécurité des services administrés

Programmation?

- Pas nécessairement expert en programmation
- Capable de programmer et de comprendre les scripts shell
- Sait compiler et installer des programmes

Processus



Processus UNIX =

Un programme en cours d'exécution =

- Le programme exécuté (fichier exécutable)
- De l'état et des ressources (mémoire, CPU, etc)
- Un utilisateur (et un groupe)
- Un identifiant = numéro de processus (pid)
- Un processus parent dont il hérite ses caractéristiques
- Un début... et une fin
- D'autres informations utiles à sa bonne gestion

Un processus est un concept **important** du **système d'exploitation**



Les processus sont isolés les uns des autres

- Un processus est autonome et cohérent
- Plein de processus existent en même temps (multitâche)
- Un processus ne peut pas corrompre un autre processus (du moins sans respecter les règles)
- Un processus peut collaborer avec d'autres processus

Exemple de collaboration: tubes

```
$ grep lol /usr/share/dict/french | lolcat
```

2 processus: grep et lolcat

Lister les processus



Commande `ps`: instantané des processus

```
$ ps
  PID TTY          TIME CMD
 4517 pts/3        00:00:00 bash
24535 pts/3        00:00:00 ps
```

Par défaut `ps` affiche la liste des processus

- De l'utilisateur courant
- Dans le terminal en cours
- Avec peu d'information

```
$ ps | head -n 50 | sort -n | grep ' '
```


Arborescence des processus

Un nouveau processus est créé par un autre processus

- Un créateur est appelé **processus parent**
 - Le premier processus (dit racine) n'a pas de parent
C'est `init` (ou `systemd`) de PID=1
 - Sous UNIX la relation de parenté est préservée
- On a une arborescence de processus

Commande `ps tree` vue arborescente des processus (extra)

```
$ ps tree -phT
```

Options de ps

Trois familles, plein d'options

- Traditionnelle (POSIX), avec un tiret « - »
 - BSD, sans tiret
 - Extensions extra, avec un ou deux tirets « -- »
- Beaucoup de confusion

```
$ ps -eF
```

```
$ ps aux
```

- -e, -A afficher tous les processus
- -f afficher plus de colonnes
- -F afficher encore plus de colonnes (extra)
- a tous les processus (avec un terminal)
- ax tous les processus (même sans terminal)
- u afficher des colonnes orientées utilisateur

Information des processus

```
$ ps -F
```

| UID | PID | PPID | C | SZ | RSS | PSR | STIME | TTY | TIME | CMD |
|------|------|------|---|------|------|-----|-------|-------|------|-------|
| jean | 1435 | 356 | 0 | 3832 | 2960 | 7 | 15:24 | pts/0 | 0:00 | ps -F |

```
$ ps u
```

| USER | PID | %CPU | %MEM | VSZ | RSS | TTY | STAT | START | TIME | COMMAND |
|------|-----|------|------|-----|-----|------|------|-------|------|---------|
| jean | 252 | 0.0 | 0.0 | 384 | 353 | tty1 | R+ | 10:08 | 0:00 | ps u |

- UID (et USER): utilisateur du processus
 - PID: identifiant du processus
 - PPID: identifiant du processus parent
 - STIME (et START): date et heure de démarrage
 - CMD (et COMMAND): ligne de commande
- Les autres colonnes une autre fois

Autres options pratiques de ps

Filtrer

- `-p` par PIDs
- `-C` par noms de commande (extra)
- `-u` par utilisateurs
- `-x` par l'utilisateur courant (extra)

Afficher

- `-o` indiquer les colonnes voulues
- `L` lister les colonnes possibles (BSD)
- `--forest` affiche l'arborescence (extra)
- `--sort` trie les lignes (extra)

Suivre en temps réel les processus

Commande `top`: processus en temps réel (extra)

- Liste les processus par utilisation processeur
- Interface interactive
- Plein de commandes pour filtrer et trier

Quelques commandes

- `q` (ou `Ctrl+C`) quitter
- `h` affiche l'aide
- `P` trier par consommation CPU (défaut)
- `M` trier par consommation mémoire
- `N` trier par PID
- `T` trier par temps CPU total
- `k` terminer un processus

Services et démons (*daemon*)

Services et démons (*daemon*)

En général / souvent

- Application qui attend et répond à un événement
- Répond à des requêtes réseau et/ou locales
- Pas invoqué explicitement, ne dépend pas d'un terminal
- Processus démarré automatiquement par `init` ou `systemd`
- Isolé dans des utilisateurs dédiés (dit *système*)
- Nom terminé par `d` (pour *daemon*)



- `init` ou `systemd`: premier processus du système
- `crond`: planifie des tâches
- `dhcpd`: configuration TCP/IP automatique des clients
- `httpd`: sert les ressources HTTP
- `sshd`: accepte les connexions SSH entrantes
- `lpd` ou `cups`: gère les impressions
- `gdm`: gestionnaire de connexion graphique (*Gnome display manager*)
- `mpd`: démon pour jouer de la musique

Exemple: serveur web

- Logiciel qui répond aux requêtes du World Wide Web
- Utilise principalement le protocole HTTP
- Voir chapitre 8 pour les détails

Apache

- Serveur web (HTTP) le plus utilisé 43% en octobre 2019
- Première version en 1995
- Site officiel: <https://httpd.apache.org/>
- Licence Apache 2.0

Nginx

- Une alternative à Apache, 2e plus utilisé, en croissance (30%)
- Première version en 2004
- Site officiel: <https://nginx.org/>
- Licence BSD

Installation

Installation d'Apache

Sur Debian et ses dérivées (Ubuntu, Mint, etc.)

```
$ sudo apt update  
$ sudo apt install apache2
```

`apt` est une interface du système de gestion de paquets (extra)

- `update` télécharger les informations sur les paquets à partir des sources configurées.
- `install` pour installer un paquet
- `upgrade` (et `full-upgrade`) pour mettre à jour les paquets
- `remove` (et `purge`) supprime un paquet
- `search`, `show`, `list` cherche et affiche des paquets

Paquet (ou paquetage)



Archive incluant

- des **fichiers** binaires et textes
- des **informations** et
- des **procédures**

nécessaires à l'installation et à la désinstallation d'un logiciel sur un système d'exploitation.

Contenu d'un paquet

- fichiers exécutables ou sources
- fichiers de configuration
- documentation
- dépendances logicielles
- scripts d'installation
- scripts de désinstallation

Gestionnaire de paquets

- L'approche recommandée pour installer un logiciel sous Unix
- Installe et gère les programmes et fichiers accessoires
- Gère les versions et les dépendances entre paquets
- Permet de mettre à jour et de désinstaller **proprement**
- Exemples: apt, dnf, pacman

De nombreux **langages** de programmation fournissent aussi des gestionnaires de paquets spécifiques à leurs écosystèmes

Exemples:

- pip: Python
- gem: Ruby
- cargo: Rust
- npm: Node.js (Javascript)
- cabal: Haskell

Installer à partir des sources



- ① Télécharger le code source de l'application
 - ② Compiler l'application
- Nécessite d'avoir déjà les compilateurs et bibliothèques
- ③ Installer l'application pour permettre son exécution
- Par les gens (utilisateurs) ou automatiquement (services)

Pour experts

- Lisez la documentation
- Pas de mises-à-jour automatiques
- La désinstallation n'est pas toujours simple
- Pour les conflits, débrouillez-vous!

Démarrage et services

Démarrage du système

Au démarrage, plusieurs services sont lancés par `init` (le premier programme)

Plusieurs gestionnaires de services existent:

`init` Système V (1983)

- Les services sont gérés par les scripts dans `/etc/init.d`
- Doivent supporter minimalement les commandes `start` et `stop`
- Exemple: `/etc/init.d/apache2` pour Apache

Systemd (2010)

- Sur la majorité des distributions Linux modernes
- Centralise la gestion de nombreux comportements
- Les services sont gérés via des fichiers de configuration
- Exemple: `/lib/systemd/system/apache2.service`
- Plus ou moins **rétro-compatible** avec `init` système V

Gestion des services

- `service` exécute une commande d'un service système V (extra)
- `systemctl` gère les services Systemd (extra)

Liste les services

```
$ sudo service --status-all
```

```
$ systemctl list-units
```

État d'un service

```
$ sudo service apache2 status
```

```
$ systemctl status apache2.service
```

Autres actions possibles sur les services

- `status` — informations sur l'état du service
- `start` — démarrer le service
- `stop` — arrêter le service
- `reload` — mettre à jour l'état du service et recharger la configuration
- `restart` — redémarrer le service

Configuration

Configuration des programmes et services

Configuration globale et pour les services

`/etc/` contient les fichiers de configuration des services

- Fichiers textes simples facilement éditables
- permet de configurer/réparer un système minimal
- Formats spécifiques aux applications
- Lisez la documentation

Exemple: `/etc/bash.bashrc` configuration générale du shell interactif

Pour les utilisateurs

Fichiers de configuration dans le répertoire maison `~`

- Directement dans `~` en tant que fichier caché (classique)
- Dans `~/.config` (moderne)

Exemple: `~/.bashrc` configuration locale du shell interactif

Configuration du shell

Les fichiers de configurations du shell sont en fait des **scripts shell**.

- Chaque ligne est une commande shell
- Les commandes sont exécutées dans l'ordre
- Les lignes vides sont ignorées
- Les commentaires sont ignorés (commencent par #)

Scripts shell

Voir le chapitre 7

Exemple .bashrc

① Ajouter dans ~/.bashrc

```
alias lla='ls -la' # Alias shell
export WEBSERV_DIR='/etc/apache2' # var. d'environnement
```

- `alias` — définit ou affiche les synonymes (*alias*) du shell (POSIX)
- `export` — définit des variables d'environnement (POSIX)

② Recharger la configuration (ou ouvrir un nouveau shell)

```
$ . .bashrc
```

③ Profiter

```
$ lla /etc
$ echo "$WEBSERV_DIR"
$ ls "$WEBSERV_DIR"
```

Variables du shell et d'environnement

Noms associés à des valeurs qui affectent l'environnement du shell et éventuellement celui des commandes exécutées

Quelques exemples:

- HOME: Le répertoire d'accueil de l'utilisateur (utilisé par ~ et cd)
 - PATH: La liste des répertoires où chercher les commandes
 - PS1: L'invite de commande principale
 - PWD: Le répertoire de travail courant (mis à jour par cd)
- Pour plus de détail, voir chapitre 7

Attention

Toujours protéger avec des **guillemets** doubles ("")

```
$ export FOO="la vie"  
$ echo "J'aime $FOO" > "$FOO"  
$ cat "$FOO"
```

Configuration d'Apache

Fichiers et dossiers de configuration

```
$ ls /etc/apache2
```

```
apache2.conf      # Configuration globale
ports.conf        # Ports sur lesquels écouter
sites-available/  # Hôtes virtuels disponibles
sites-enabled/    # Hôtes virtuels activés
mods-available/   # Modules disponibles
mods-enabled/     # Modules activés
```

- `apache2.conf` appelé souvent <httpd.conf> peut aussi contenir les directives d'inclusion des autres fichiers et dossiers de configuration.
- La directive `Listen` permet de spécifier le port et/ou l'adresse IP par défaut pour accéder au site.
- Documentation configuration:
<http://httpd.apache.org/docs/current/>

Hôte par défaut

```
$ cd /etc/apache2/sites-available  
$ sudo vim 000-default.conf
```

Configuration hôte par défaut:

```
<VirtualHost *:80>  
    ServerAdmin webmaster@localhost  
    DocumentRoot /var/www/html  
    ErrorLog ${APACHE_LOG_DIR}/error.log  
    CustomLog ${APACHE_LOG_DIR}/access.log combined  
</VirtualHost>
```

Redémarrage du serveur

Il faut recharger le serveur web pour que le site web soit accessible

```
$ sudo service apache2 reload  
ou  
$ sudo systemctl reload apache2.service
```




- Un VirtualHost par nom de domaine différent.

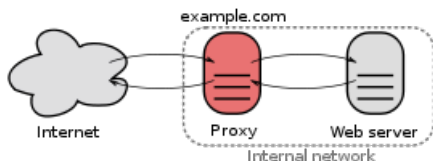
```
ServerName www.example.com # Ajouter dans VirtualHost
```

- La directive DocumentRoot spécifie le dossier accessible sur le web.

Proxys inverse (reverse proxy ou gateway)



Intermédiaire entre client et serveur, mais côté serveur



(source: [Wikipedia](#))

- Reçoit les requêtes de l'utilisateur et les envoie au serveur
- Reçoit la réponse du serveur et la retourne au client
- Le client peut ne pas avoir accès direct au serveur
- Proxy: `mod_proxy`
- Load balancing: ajouter `mod_proxy_balancer`

```
# Reverse Proxy dans la configuration Apache
ProxyPass "/foo" "http://foo.example.com/bar"
ProxyPassReverse "/foo" "http://foo.example.com/bar"
```

Contenu accessible

Contenu *web*

- Dans le répertoire indiqué par la directive `DocumentRoot`.
→ `/var/www/html` par défaut
- Dossier utilisateur: spécifié par la directive `UserDir`
→ `UserDir public_html` permet aux utilisateurs de créer un sous dossier `public_html` et d'y mettre le contenu accessible sur le web.

Accéder au site web

Ouvrir un navigateur à l'adresse suivante.

- Par défaut `http://localhost` ou `http://nom.domaine.ext`.

Exemple: `http://labunix.uqam.ca`

- Utilisateur `http://nom.domaine.ext/~nomutilisateur`

Exemple `http://labunix.uqam.ca/~ab123456`

Naviguer sur le *web*

Naviguer sur le web

Navigateur (ou client web)

- Logiciel qui simplifie la navigation
- Aussi appelé **fureteur** (ou *browser*)
- Saisit les requêtes des clients
- Communique avec le serveur
- Affiche le contenu demandé

Exemples

- **Graphiques**: Firefox, Chromium, Safari, IE, ...
- **Console**: lynx, elinks, w3m
- **Ligne de commande**: wget, curl



URL (*uniform resource locator*)

schéma: [//autorité] chemin[?requête] [#fragment]

- **schéma**: http, https, ftp, mailto, file, etc.
 - **autorité**: de la forme [utilisateur@]hôte[:port]
 - **chemin**: chemin vers la ressource
 - **requête**: suite de paires attribut-valeur (souvent)
 - **fragment**: identifie une partie spécifique de la ressource
- détails au chapitre 8, au INF3190 et au INF3271

Exemples

- <http://www.wikipedia.org/>
- <https://alice@abc.com:99/forum/?tag=bash&order=newest#top>

Télécharger des ressources

- `wget` (GNU)
- `curl` (Extra)

```
$ wget "https://fr.wikipedia.org/wiki/Shell_Unix"  
$ curl "https://fr.wikipedia.org/wiki/Shell_Unix"
```

Wget ou cURL?

cURL

- Plus portable
- Basé sur une bibliothèque `libcurl`
- Supporte plus de protocoles
- Licence MIT

Wget

- Souvent déjà installé
- Commande autonome
- Récursif (miroir)
- Licence GPL



```
$ curl url
```

Beaucoup d'options

- `-o`, `--output` sauvegarde dans le fichier spécifié
- `-O`, `--remote-name` sauvegarde dans le nom de fichier de l'URL
- `-L`, `--location` suit la redirection vers une autre page
- `-C`, `--continue-at` reprend un téléchargement interrompu
- `-I`, `--head` télécharge seulement l'en-tête
- `-z`, `--time-cond` télécharge si modifié depuis une certaine date
- `-v`, `--verbose` mode verbeux
- `-s`, `--silent` mode silencieux
- `-x`, `--proxy` utilise un proxy
- `--limit-rate` vitesse limite la vitesse (en secondes)
- `--trace` affiche une trace (débuguer)

Exemple

Vérifier le type de ressource:

```
$ curl -s https://www.linux.org/styles/uix/uix/logo.png |  
> grep '^content-type'  
content-type: image/png
```

Télécharger et sauvegarder sous logo.png:

```
$ curl -sO https://www.linux.org/styles/uix/uix/logo.png  
$ display logo.png
```

Cybersécurité

Cybersécurité

Exposer un service c'est prendre un risque.

Il faut respecter les **bonnes pratiques**

- Limiter les services exposés
 - Contrôler les données exposées [OWASP 2017 A3-Sensitive Data Exposure](#)
 - Configurer correctement (et éviter les modes *debug*) [OWASP 2017 A6-Security Misconfiguration](#)
 - Utiliser des logiciels fiables et à jour [OWASP 2017 A9-Using Components with Known Vulnerabilities](#)
 - Surveiller les journaux et les alertes [OWASP 2017 A10-Insufficient Logging & Monitoring](#)
- OWASP: *Open Web Application Security Project*, organisme qui promeut la sécurité du web.

Plus dans le chapitre 8 et dans INF4471 et INF600C



- Un programme interactif communique constamment et directement à l'utilisateur qui l'a configuré et l'exécute.
 - Un service ne communique pas directement avec l'administrateur
- Il écrit un **journal** des **événements** et des **problèmes**

Traditionnellement, les journaux des services sont dans `/var/log/`

- Fichiers textes simples
- Ayant souvent l'extension `.log` (exemple `/var/log/auth.log`)
- Parfois archivés automatiquement (exemple `/var/log/auth.log.2.gz`)
- Lisibles avec un système minimal
- *grepable* et compatibles avec les outils Unix habituels

Journaux Apache

Dans `/var/log/apache2/`

```
$ ls /var/log/apache2
access.log          # Enregistrement des requêtes
error.log           # Enregistrement des erreurs
$ grep " 404 " /var/log/apache2/access.log
$ tail -f /var/log/apache2/*.log
```

`tail` — Affiche les dernières lignes

- `-f`, `--follow` affiche les données en continu

Note

L'emplacement et le contenu des *logs* sont configurables

SSH = secure shell



- Permet de se connecter à un serveur
- Avec une connexion sécurisée
- Anciennement, on utilisait `telnet`, qui n'est pas sécurisé
- Requiert de s'**authentifier**
- Utilise une **clé publique** pour identifier le serveur
- Implémenté dans OpenSSH: `ssh` (BSD)

```
$ ssh ab123456@java.labunix.uqam.ca #connexion
$ exit # deconnexion
```

Transférer des fichiers

`scp` — copie des fichiers par connexion sécurisée

- `-r` copie récursivement (suit les liens symboliques)
- `-p` préserve les dates et les droits
- `-P` spécifie un port

```
$ scp img/debian.png img/ubuntu.png\  
> ab123456@java.labunix.uqam.ca:~/Pictures  
Password:  
debian.png                100%   18KB  333.8KB/s   00:00  
ubuntu.png                100%   886    28.1KB/s   00:00  
$ scp -r ab123456@java.labunix.uqam.ca:~/Pictures/\  
> Pictures  
[...]  
$ ls Pictures/
```

Voir aussi `rsync` (extra) — synchronise des fichiers locaux et distants



- On génère une **paire de clés** (publique, privée)
 - `ssh-keygen`
 - Dans `~/.ssh/id_rsa` et `~/.ssh/id_rsa.pub` par défaut
 - On entre une **phrase de passe** (optionnelle)
 - Permet de **déchiffrer** la clé privée
 - Utilisée par le client
 - On **copie** la clé publique sur la machine distante
 - `ssh-copy-id`
 - Elles vont dans `~/.ssh/authorized_keys`
 - On se connecte sans mot de passe

Autres utilisations

- Automatiser les connexion SSH
- Identités multiples (une par paire de clés)
- Protocole `git` via SSH (on enregistre ses clés publiques)

Multiplexeur de terminaux et sessions



Quitte une session

- Si on quitte une connexion SSH
- Tous les processus lancés terminent
- On peut toujours utiliser `disown` ou `nohup`
- Mais pas pratique de les récupérer plus tard

Solution

Utiliser un serveur de sessions

- `screen` (GNU)
- `tmux` (extra)

Manipulation de sessions

- `tmux` démarre une nouvelle session
- `tmux new -s <nom>` démarre une session nommée
- `tmux a`, `tmux at`, `tmux attach` charge la dernière session
- `tmux a -t <nom>` charge une session nommée
- `tmux ls` liste les sessions existantes
- `tmux kill-session -t <nom>` termine une session nommée

Lorsque Tmux tourne, on peut entrer `Ctrl` + `B` puis:

- `:new` crée une nouvelle session
- `s` liste les sessions
- `$` pour nommer la session courante
- `d` se détacher de la session
- `t` affiche l'heure dans la fenêtre
- `?` affiche de l'aide

Manipulation de l'interface

Panneaux

- % séparation horizontale
- " séparation verticale
- o inverse les panneaux
- q affiche la numérotation des panneaux
- x tue le panneau
- Espace change la disposition des panneaux

Fenêtres

- c nouvelle fenêtre
- , nommer une fenêtre
- w lister les fenêtres
- f trouver une fenêtre
- & tuer une fenêtre
- . déplacer une fenêtre

Exemple

Je lance une session:

```
$ ssh ab123456@java.labunix.uqam.ca
$ tmux new -s masession
# Je lance maintenant un long calcul
$ cat /dev/urandom | tr -cd 0-9
# Je "détache" la session (Ctrl + B puis d)
$ exit
```

Puis je la récupère:

```
$ ssh ab123456@java.labunix.uqam.ca
$ tmux a -t masession
```

Docker

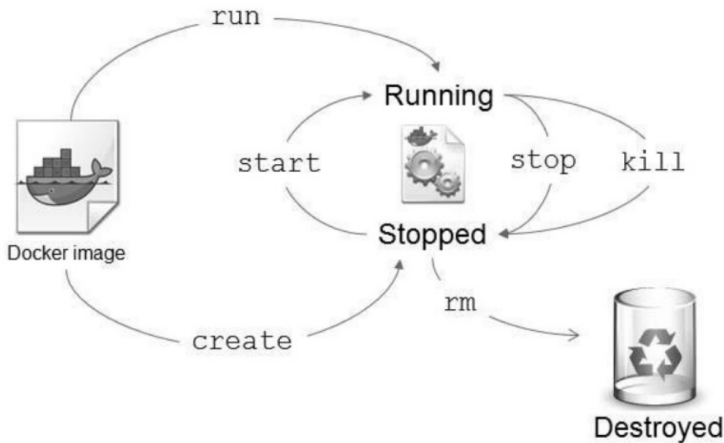
C'est quoi

« Docker est une technologie de *conteneurisation* qui permet la création et l'utilisation de conteneurs Linux »

- Un **conteneur** est un ensemble de processus isolés d'un système
- Il partage le même noyau de système d'exploitation que son hôte
- Il doit être compatible avec le système d'exploitation sous-jacent
- Il contient tous les fichiers nécessaires à son exécution

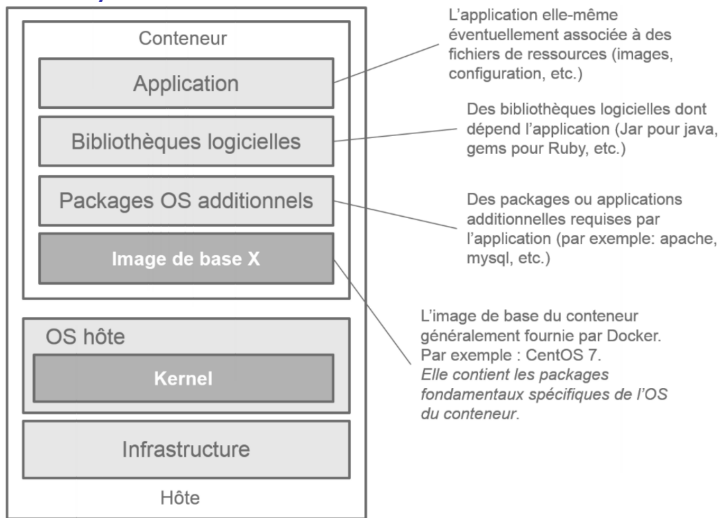
Terminologie

- **Image** une archive en lecture seule contenant un ensemble de fichiers (exécutables, bibliothèques, etc.)
- **Conteneur** est une instance active (started) ou inactive (stopped) d'une image
- **Registry** est un annuaire de stockage des images Docker
→ autrement dit, c'est un dépôt d'images Docker
- **Docker HUB** dépôt public d'images mises à disposition par Docker
- **Dockerfile** fichier texte de description d'une image



Source: "Docker - Pratique des architectures à base de conteneurs",
par Pierre-Yves Cloux, Thomas Garlot et Johann Kohler

Comment ça fonctionne



Source: "Docker - Pratique des architectures à base de conteneurs",
par Pierre-Yves Cloux, Thomas Garlot et Johann Kohler



- Les principales technologies sur lesquelles repose Docker sont `cgroup`, `namespace` et `UnionFS`
- Il est possible de faire collaborer plusieurs conteneurs avec `docker-compose`
- Administrer un grand nombre de conteneurs est une tâche complexe, il est recommandé d'utiliser un orchestrateur comme `Kubernetes`
- Le partage du noyau de l'hôte avec les conteneurs ouvre une brèche de sécurité

Alternatives

- Il existe d'autres technologies de conteneurisation comme par exemple `LXC` ou `Singularity`