Conteneurs et Virtualisation

M1 - CHPS
Architecture Interne des Systèmes d'exploitations (AISE)

Jean-Baptiste Besnard
<jean-baptiste.besnard@paratools.com>

Julien Adam
<julien.adam@paratools.com>

UNIVERSITÉ DE
VERSAILLES
ST-QUENTIN-EN-YVELINES

Programme du Semestre

- 1 - Généralités sur les OS et Utilisation de base

- 2 - Processus, Threads & Synchronisation

- 3 - Compilation et représentation Binaire

- 4 - Architecture Mémoire d'un processus

- 5 - Programmation réseau et entrées/sorties avancées

- **6 - Virtualisation et Conteneurs**

- **7 - Examen + Noyau Linux et modules**

- **8 - Démo de projets**

Machines Virtuelles

Virtualisation

Une machine virtuelle:

- Émule une machine de manière logicielle pour permettre l'exécution isolée d'un programme;
- Execute du code dans un contexte spécifique (souvent avec l'aide du matériel) pour le contraindre en terme d'accès;

Application / Serveur

OS Virtualisé

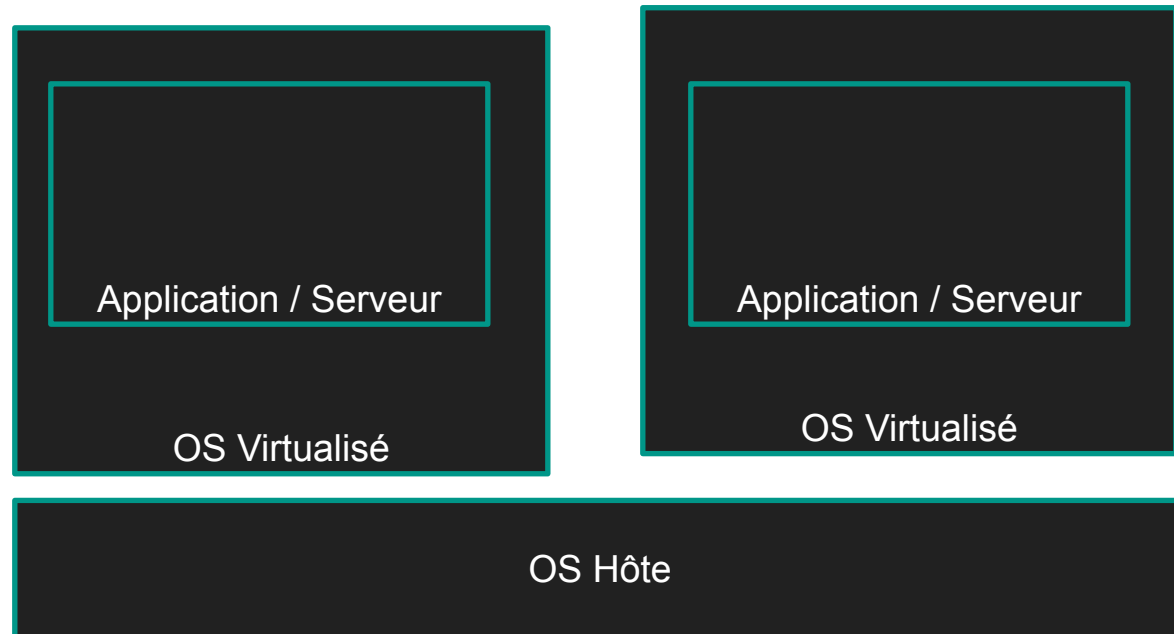OS Hôte
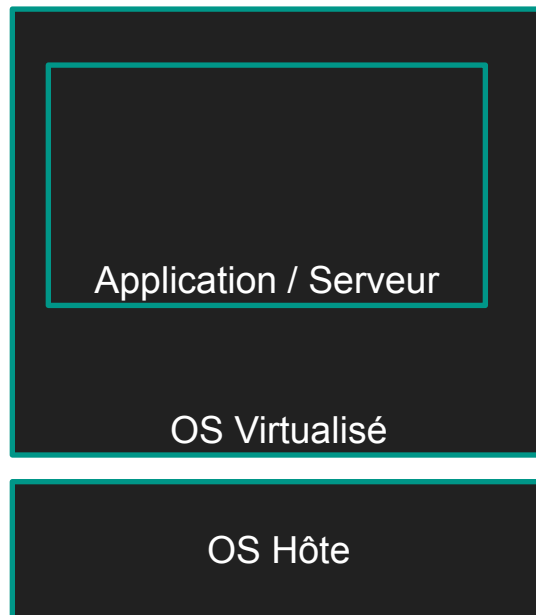
Virtualisation

Une machine virtuelle:

- Émule une machine de manière logicielle pour permettre l'exécution isolée d'un programme;
- Execute du code dans un contexte spécifique (souvent avec l'aide du matériel) pour le contraindre en terme d'accès;

Avantages de la Virtualisation

- Regroupement des serveurs sur une même machine. De plus certains serveurs sont faibles en consommation CPU.
- Isolation FORTE des serveurs avec des **systèmes d'exploitation différents et des systèmes de fichiers distincts**;
- **Isolation** y compris vis à vis du matériel (carte réseau virtuelle) et contraintes mémoire cpu explicite (exposition partielle des resources) — **gestion dynamique des resources** (CPU Hotplug);
- **Réplication** et **sauvegarde** facilitée (on sauve l'image disque dans sa totalité), rétablir le système c'est rétablir une image plus récente;
- Facilité d'administration il devient possible de **migrer** un serveur/ service donné.

Inconvénients de la Virtualisation

- Les abstractions matérielles ont un coût en performance non négligeable;
- L'OS est totalement répliqué en stockage et en mémoire dans les différentes VMs;
- Si un serveur avec de nombreuses VMs tombe toutes les VMs associées sont inopérantes (besoin de redondance);
- Il y a un overhead d'administration important du fait de la complexité additionnelles des machines séparées.

Votre Première VM

Nous utiliserons qemu installez le !

- (Ubuntu) sudo apt-get install qemu-kvm qemu virt-manager virt-viewer libvirt-bin
- (Centos 7) yum install -y qemu-kvm qemu-img virt-manager libvirt libvirt-python libvirt-client virt-install virt-viewer
- Votre distrib (go Google)

Créer une image disque de 5GB:

```
qemu-img create -f qcow2 mydebian.qcow2 5G
```

Lancer la VM:

```
qemu-system-XXX [IMAGE]
```

qemu-system-aarch64	qemu-system-i386	qemu-system-microblazeel	qemu-system-mipsel	qemu-system-ppc64	qemu-system-sh4eb	qemu-system-unicore32
qemu-system-alpha	qemu-system-lm32	qemu-system-mips	qemu-system-moxie	qemu-system-ppcemb	qemu-system-sparc	**qemu-system-x86_64**
qemu-system-arm	qemu-system-m68k	qemu-system-mips64	qemu-system-or32	qemu-system-s390x	qemu-system-sparc64	qemu-system-xtensa
qemu-system-cris	qemu-system-microblaze	qemu-system-mips64el	qemu-system-ppc	qemu-system-sh4	qemu-system-tricore	qemu-system-xtensaeb

Votre Première VM

```
Boot failed: could not read the boot disk

Booting from DVD/CD...
Boot failed: Could not read from CDROM (code 0003)
Booting from ROM...
iPXE (PCI 00:03.0) starting execution...ok
iPXE initialising devices...ok


iPXE 1.0.0+git-20161027.b991c67-1 -- Open Source Network Boot Firmware -- http:/
/ipxe.org
Features: DNS HTTP iSCSI NFS TFTP AoE ELF MBOOT PXE bzImage Menu PXEXT

net0: 52:54:00:12:34:56 using 82540em on 0000:00:03.0 (open)
  [Link:up, TX:0 TXE:0 RX:0 RXE:0]
Configuring (net0 52:54:00:12:34:56)...... ok
net0: 10.0.2.15/255.255.255.0 gw 10.0.2.2
net0: fec0::5054:ff:fe12:3456/64 gw fe80::2
net0: fe80::5054:ff:fe12:3456/64
Nothing to boot: No such file or directory (http://ipxe.org/2d03e13b)
No more network devices

No bootable device.
```

Quel est le problème ??

Votre Première VM

```
Boot failed: could not read the boot disk

Booting from DVD/CD...
Boot failed: Could not read from CDROM (code 0003)
Booting from ROM...
iPXE (PCI 00:03.0) starting execution...ok
iPXE initialising devices...ok




iPXE 1.0.0+git-20161027.b991c67-1 -- Open Source Network Boot Firmware -- http:/
/ipxe.org
Features: DNS HTTP iSCSI NFS TFTP AoE ELF MBOOT PXE bzImage Menu PXEXT

net0: 52:54:00:12:34:56 using 82540em on 0000:00:03.0 (open)
  [Link:up, TX:0 TXE:0 RX:0 RXE:0]
Configuring (net0 52:54:00:12:34:56)...... ok
net0: 10.0.2.15/255.255.255.0 gw 10.0.2.2
net0: fec0::5054:ff:fe12:3456/64 gw fe80::2
net0: fe80::5054:ff:fe12:3456/64
Nothing to boot: No such file or directory (http://ipxe.org/2d03e13b)
No more network devices

No bootable device.
```

Pas d'OS ! Nous avons passé une image vide !!

Votre Première VM

Démarrer avec un CDROM inséré (comme une vrai machine):

https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/

Image d'installation par le réseau de Debian « netinst »:

https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-10.8.0-amd64-netinst.iso

Téléchargez l'image:

wget https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-10.8.0-amd64-netinst.iso

Démarrez la VM avec l'image disque:

qemu-system-x86_64 --cdrom ./debian-10.3.0-amd64-netinst.iso -hda mydebian.qcow2 -m 1024
                    -netdev user,id=eth0,hostfwd=tcp::10022-:22 -device e1000,netdev=eth0
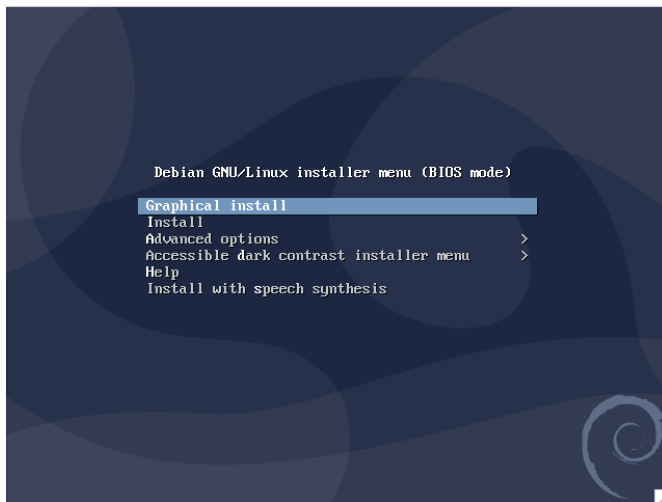
CTRL+ALT / CTRL + ALT + g pour sortir la souris

Image Debian avec Docker

Récupérez une image avec debian:

root MDP toto
chps MDP toto

https://france.paratools.com/chps.qcow2

Votre Première VM

Donner plus de resources à la VM: `-smp 2 -m 2048`

```
1  [|                                    2.0%]   Tasks: 26, 49 thr; 1 running
2  [                                     0.0%]   Load average: 0.00 0.04 0.14
Mem[|||||||||||||||||             197M/1.95G]   Uptime: 00:21:09
Swp[                                   0K/0K]
```

Se connecter en ssh:

ssh root@localhost -p 10022

CTRL+ALT pour sortir la souris

Votre Première VM

Démarrer le système sans affichage: `-nographic`

```
CTRL + a puis x pour quitter
CTRL + a puis c pour ouvrir la console qemu (qemu-monitor)
```

Hotplug CPU

qemu-system-x86_64 [IMG] **-smp 2,maxcpus=4 -nographic**

```
CTRL + a puis x pour quitter
CTRL + a puis c pour ouvrir la console qemu (qemu-monitor)

CTRL + a puis c
cpu-add 3 (0 et 1 sont déjà présents)

Sur l'hôte:
$ ssh root@localhost -p 10022 #mdp toto

Dans la VM le nouveau CPU n'est pas directement actif:
$ echo 1 > /sys/devices/system/cpu/cpu2/online
```

```
1  [                                              0.0%]
2  [|                                             2.6%]
Mem[|||||||||||||                          133M/1.95G]
Swp[                                           0K/0K]
```

```
1  [|                                             1.3%]
2  [|                                             1.3%]
3  [                                              0.0%]
Mem[|||||||||||||                          133M/1.95G]
Swp[                                           0K/0K]
```

```
1  [|                                             1.3%]
2  [|                                             0.6%]
3  [                                              0.0%]
4  [||                                            2.0%]
Mem[|||||||||||||                          136M/1.95G]
Swp[                                           0K/0K]
```

b

qemu-system-x86_64 [IMG] **-device virtio-baloon**

```
# Dépend du chargement des modules virtio dans le kernel cible !
#CTRL + a puis c pour ouvrir la console qemu (qemu-monitor)

CTRL + a puis c
```

```
(qemu) balloon 1024
```

```
1  [                                                    0.0%]
2  [|                                                   1.3%]
Mem[||||||||||||                                135M/1.95G]
Swp[                                                 0K/0K]
```

```
1  [                                                    0.0%]
2  [||                                                  2.0%]
Mem[|||||||||||||||||||||||||||||               134M/971M]
Swp[                                                 0K/0K]
```

Le Conteneur

Conteneurs ?

- C'est beaucoup de chose, cela veut principalement dire que les ressources d'un processus sont isolées;
- Ceci utilise la notion de namespace:
  - ➡ Mount namespaces
  - ➡ User-namespaces
  - ➡ Network namespaces
  - ➡ (…)
- En général le conteneur consiste en un changement de système de fichier tout en gardant le même noyau à la différence d'une VM par exemple.

Liste des Namespaces

« man namespaces »

Namespace	Constant	Isolates
Cgroup	CLONE_NEWCGROUP	Cgroup root directory
IPC	CLONE_NEWIPC	System V IPC, POSIX message queues
Network	CLONE_NEWNET	Network devices, stacks, ports, etc.
Mount	CLONE_NEWNS	Mount points
PID	CLONE_NEWPID	Process IDs
User	CLONE_NEWUSER	User and group IDs
UTS	CLONE_NEWUTS	Hostname and NIS domain name

Un Conteneur

Aspects Runtime

Container
Runtime

Bash

./a.out

Root 2

Root 2

unshare()
or **clone()**

Root 1
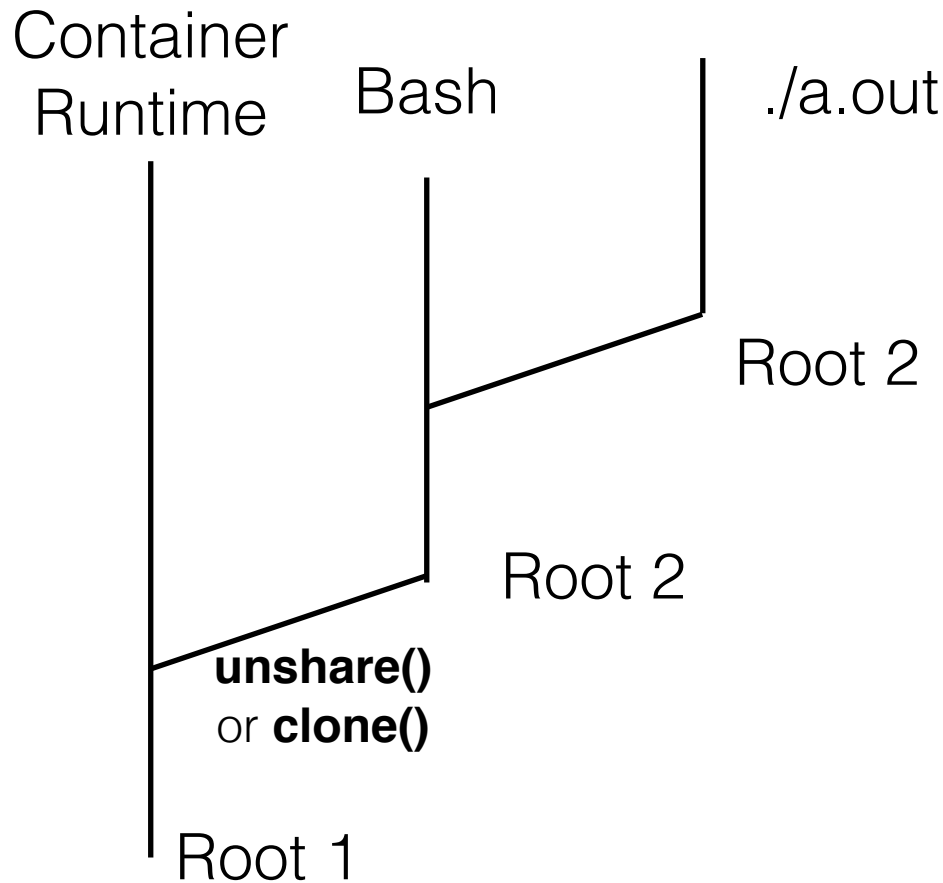
Le Conteneur vu de l'utilisateur
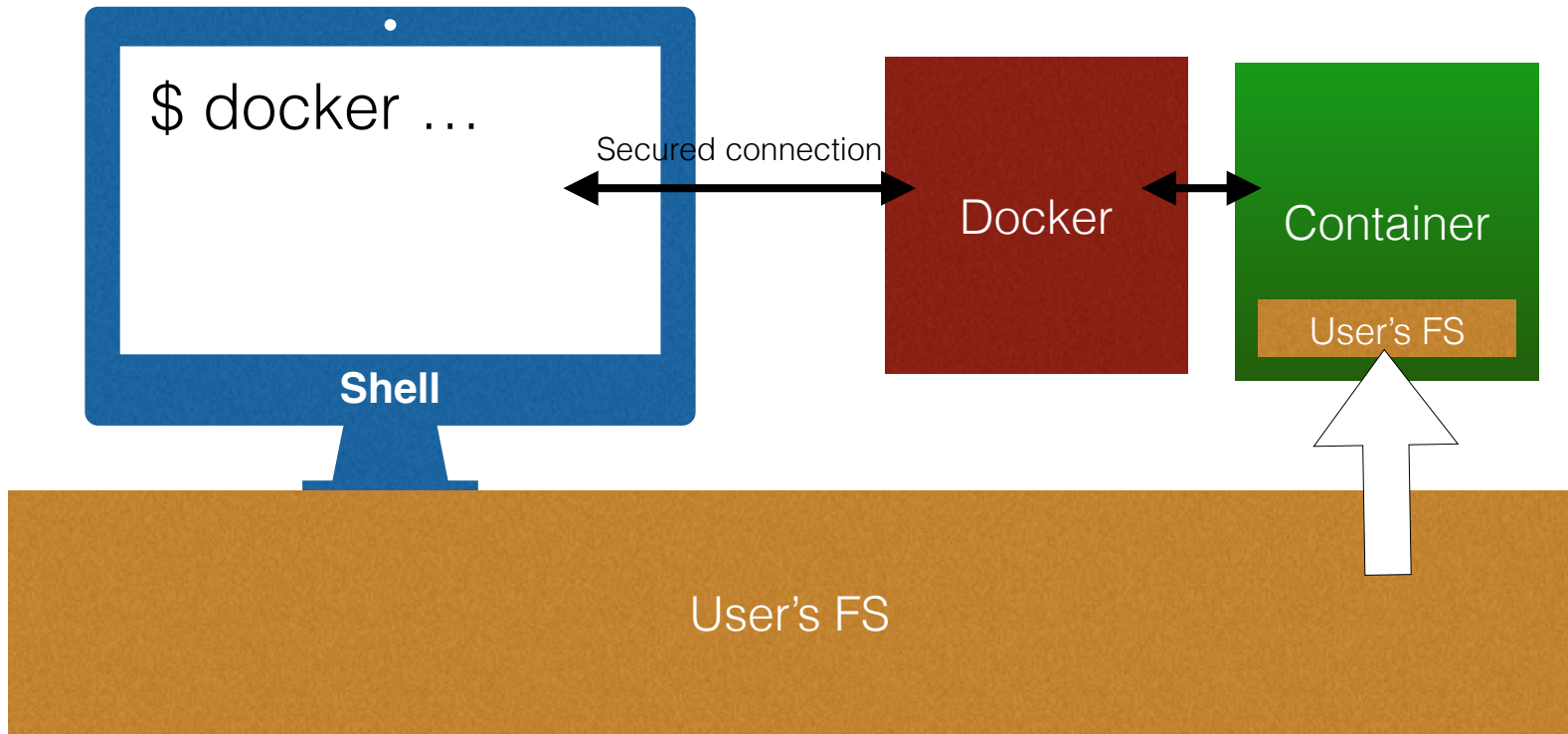
RUN <IMAGE> <COMMAND>

RUN <IMAGE>

Une image contient toutes les dépendances pour lancer un programmes (binaires bibliothèques …) mais PAS le kernel.
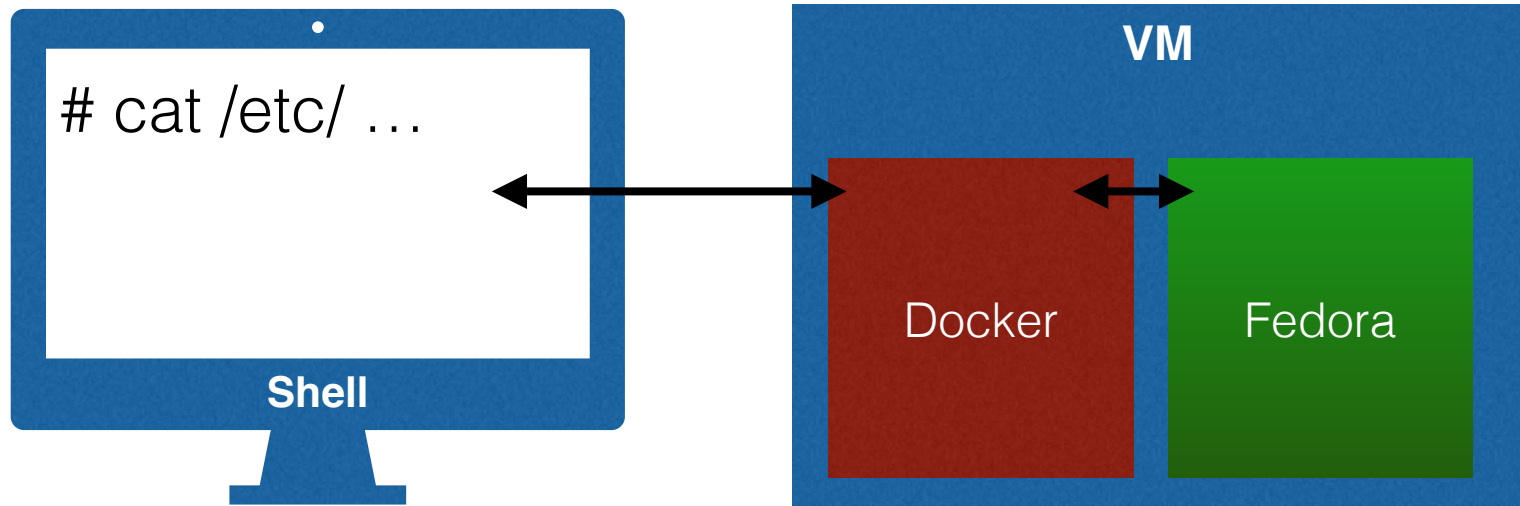
Docker

Commencer avec Docker

```
$ docker version
Client: Docker Engine – Community
 Version:               18.09.5
(…)
```

Le Démon Docker

$ docker …

Secured connection

Docker

Container

User's FS

Shell

User's FS

First Run inside Docker

```
$ docker run -ti fedora
(Entering container … )
# cat /etc/redhat-release
Fedora release 31 (Thirty One)
```

cat /etc/ …

Shell

VM

Docker

Fedora

Syntaxe Docker Run

```
docker run -ti --rm [IMAGE] [COMMAND] [ARGS]
```

▸ -i : mode interactif   (par défaut Docker tourne en arrière plan);

▸ -t : pour ouvrir un TTY et donc avoir un support terminal complet pour VIM par exemple;

▸ --rm: supprimer le conteneur à la sortie du programme.

Lister les Images Docker

Pour voir les images disponibles allez sur Docker HUB
https://hub.docker.com/search?q=&type=image

```
$ docker image list

REPOSITORY  TAG     IMAGE ID      CREATED       SIZE
debian      latest  b5d2d9b1597b  11 days ago   114MB
alpine      latest  cc0abc535e36  2 weeks ago   5.59MB
ubuntu      latest  549b9b86cb8d  2 weeks ago   64.2MB
fedora      latest  f0858ad3febd  2 months ago  194MB
```

Lister les Conteneurs

```
$ docker image list
REPOSITORY              TAG              IMAGE ID         CREATED         SIZE
jupyter/scipy-notebook  latest           295a5802d799     2 days ago      3.45GB
pcocc/umoci             v0.1             27530b76a409     7 months ago    11.5MB
pcocc/umoci             v0.1-amd64       27530b76a409     7 months ago    11.5MB
pcocc/tzdata            v0.1             4a126134da60     7 months ago    1.53MB
pcocc/tzdata            v0.1-amd64       4a126134da60     7 months ago    1.53MB
pcocc/squashfs-tools    v0.1             0da08da32ae5     7 months ago    2.11MB
pcocc/squashfs-tools    v0.1-amd64       0da08da32ae5     7 months ago    2.11MB
(…)
```

Lancer un Conteneur Détaché

```
$ docker run -ti -d  --rm ubuntu sleep 10000

$ docker ps
CONTAINER ID      IMAGE          COMMAND          CREATED          STATUS          PORTS          NAMES
8aa628c5982a      ubuntu          "sleep 100000"    About a minute ago   Up About a minute                    recursing_herschel
$ docker attach recursing_herschel
(…)
# CTRL + p puis CTRL + q
read escape sequence
```
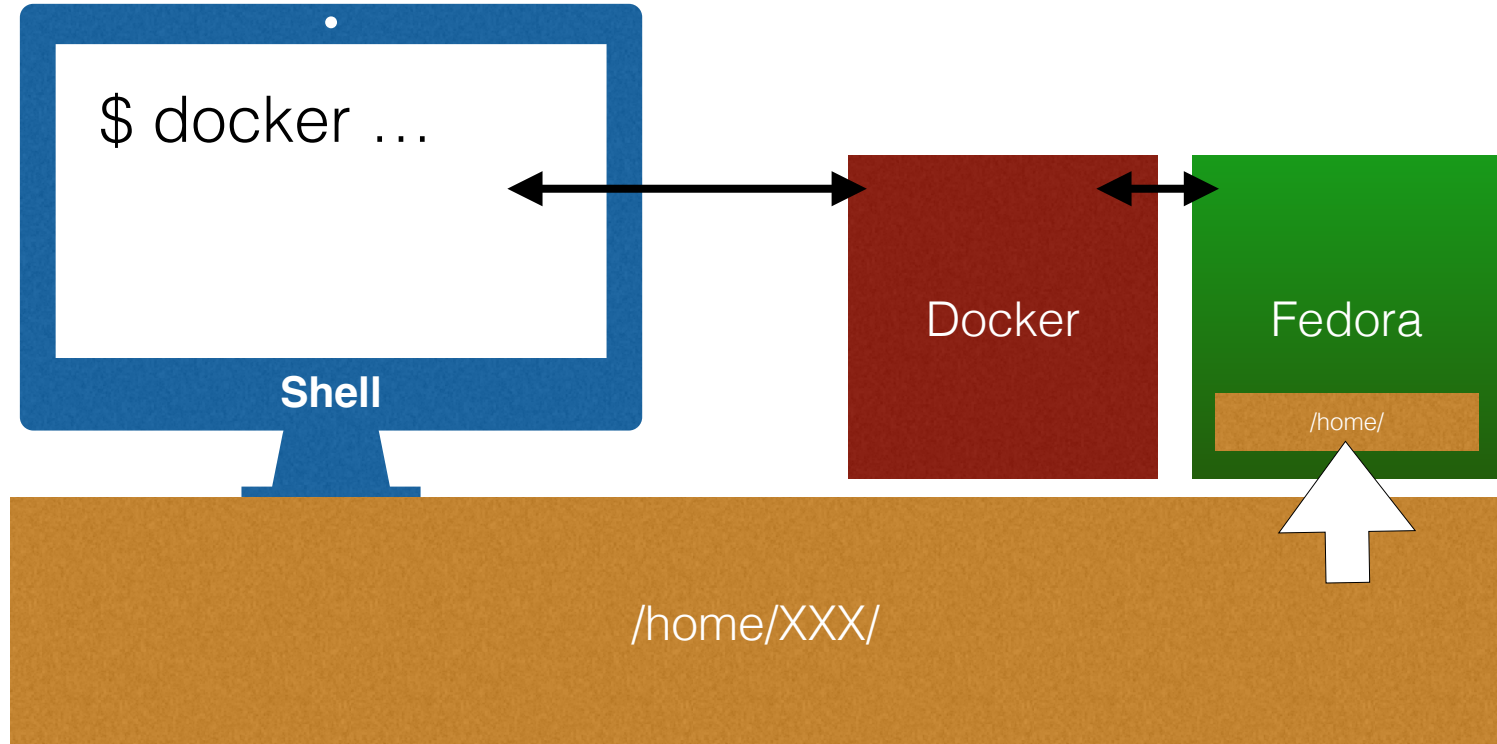
Pour se détacher CTRL+p puis CTRL+q
Uniquement si lancé avec -ti !!!!

Monter le $HOME

```
cd $HOME; touch ./hello_docker
docker run -ti \
  --rm -v $HOME:/home -w /home fedora
ls ./hello_docker
./hello_docker
```

‣ -v : Volume (mounting dir & files)
    [FROM]:[TO]

‣ -w : work directory (the CWD of the
  command being run) here bash by default.

Monter le $HOME

$ docker ...

Shell

Docker

Fedora

/home/

/home/XXX/

Voir la Commande par Défaut

$ docker inspect nginx

```
"Env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "NGINX_VERSION=1.17.0",
    "NJS_VERSION=0.3.2",
    "PKG_RELEASE=1~stretch"
],
"Cmd": [
    "/bin/sh",
    "-c",
    "#(nop) ",
    "CMD [\"nginx\" \"-g\" \"daemon off;\"]"
],
```

Altérer la Commande par Défaut

```
$ docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]
```

$ docker run -ti --rm nginx [CMD] [ARGS]

$ docker run -ti --rm nginx /bin/bash

```
$ docker run -ti  --rm nginx /bin/bash
root@2fa50e296016:/#

Exit ou CTRL + D pour quitter
```

Editer une image existante

```
$ docker run -ti \
    --name mycont -v $HOME:/home/ fedora
cp /home/my_data /data
CTRL+D
```

$ docker ps -a #-a for terminated cont.

CONTAINER ID	IMAGE COMMAND CREATED	STATUS	PORTS
044ed20ab55b	fedora « /bin/bash" 32 seconds ago	Exited (0) 20 seconds ago	hungry_ros

$ docker commit hungry_rosalind myfed

CONTAINER ID	IMAGE COMMAND CREATED	STATUS	PORTS
044ed20ab55b	fedora « /bin/bash" 32 seconds ago	Exited (0) 20 seconds ago	hungry_ros

$ docker image list

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
myfed	latest	93006aa38912	7 seconds ago	194MB
fedora	latest	f0858ad3febd	2 months ago	194MB

Lancer un Serveur

```
docker run -ti --rm -p 8080:80 nginx
```

Redirige 8080 vers le port 80 du conteneur.

```
$ docker run -d  -p 8080:80  nginx

$ curl localhost:8080
(…)
<title>Welcome to nginx!</title>
(…)

$ docker ps
CONTAINER ID       IMAGE            COMMAND                  PORTS             NAMES
2211a94ebf61       nginx            "nginx -g 'daemon of…"        0.0.0.0:8080->80/tcp   magical_nash

$ docker kill magical_nash
```

Dockerfile
Créez vos propre conteneurs

Le Dockerfile

- Décrit la recette d'un conteneur
- Vise à être reproductible (partager la recette et non le conteneur)
- Est la base du partage sur le HUB Docker:

```
1   ADD file ... in /                                24.65 MB

2   CMD ["bash"]                                         0 B

3   LABEL maintainer=NGINX Docker Maintainers            0 B

4   ENV NGINX_VERSION=1.17.8                             0 B

5   ENV NJS_VERSION=0.3.8                                0 B

6   ENV PKG_RELEASE=1~buster                             0 B

7   /bin/sh -c set -x                                22.72 MB

8   /bin/sh -c ln -sf /dev/stdout                      202 B

9   EXPOSE 80                                            0 B

10  STOPSIGNAL SIGTERM                                   0 B

11  CMD ["nginx" "-g" "daemon                            0 B
```

FROM

```
FROM [--platform=<platform>] <image> [AS <name>]
```
Or

```
FROM [--platform=<platform>] <image>[:<tag>] [AS <name>]
```
Or

```
FROM [--platform=<platform>] <image>[@<digest>] [AS <name>]
```

The `FROM` instruction initializes a new build stage and sets the *Base Image* for subsequent instructions. As such, a valid `Dockerfile` must start with a `FROM` instruction. The image can be any valid image – it is especially easy to start by **pulling an image** from the *Public Repositories*.

- `ARG` is the only instruction that may precede `FROM` in the `Dockerfile`. See Understand how ARG and FROM interact.
- `FROM` can appear multiple times within a single `Dockerfile` to create multiple images or use one build stage as a dependency for another. Simply make a note of the last image ID output by the commit before each new `FROM` instruction. Each `FROM` instruction clears any state created by previous instructions.
- Optionally a name can be given to a new build stage by adding `AS name` to the `FROM` instruction. The name can be used in subsequent `FROM` and `COPY` `--from=<name|index>` instructions to refer to the image built in this stage.
- The `tag` or `digest` values are optional. If you omit either of them, the builder assumes a `latest` tag by default. The builder returns an error if it cannot find the `tag` value.

The optional `--platform` flag can be used to specify the platform of the image in case `FROM` references a multi-platform image. For example, `linux/amd64`, `linux/arm64`, or `windows/amd64`. By default, the target platform of the build request is used. Global build arguments can be used in the value of this flag, for example automatic platform ARGs allow you to force a stage to native build platform (`--platform=$BUILDPLATFORM`), and use it to cross-compile to the target platform inside the stage.

https://docs.docker.com/engine/reference/builder/

RUN

RUN has 2 forms:

- `RUN <command>` (*shell* form, the command is run in a shell, which by default is `/bin/sh -c` on Linux or `cmd /S /C` on Windows)
- `RUN ["executable", "param1", "param2"]` (*exec* form)

The `RUN` instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the `Dockerfile`.

Layering `RUN` instructions and generating commits conforms to the core concepts of Docker where commits are cheap and containers can be created from any point in an image's history, much like source control.

The *exec* form makes it possible to avoid shell string munging, and to `RUN` commands using a base image that does not contain the specified shell executable.

The default shell for the *shell* form can be changed using the `SHELL` command.

In the *shell* form you can use a `\` (backslash) to continue a single RUN instruction onto the next line. For example, consider these two lines:

```
RUN /bin/bash -c 'source $HOME/.bashrc; \
echo $HOME'
```

Together they are equivalent to this single line:

```
RUN /bin/bash -c 'source $HOME/.bashrc; echo $HOME'
```

EXPOSE

```
EXPOSE <port> [<port>/<protocol>...]
```
The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.
You can specify whether the port listens on TCP or UDP, and the default is TCP if the protocol is not
specified.

The EXPOSE instruction does not actually publish the port. It functions as a type of documentation between the
person who builds the image and the person who runs the container, about which ports are intended to be
published. To actually publish the port when running the container, use the –p flag on docker run to publish
and map one or more ports, or the –P flag to publish all exposed ports and map them to high-order ports.

By default, EXPOSE assumes TCP. You can also specify UDP:

```
EXPOSE 80/udp
```
To expose on both TCP and UDP, include two lines:

```
EXPOSE 80/tcp
EXPOSE 80/udp
```

$ docker run -d -P nginx

```
$ docker ps
CONTAINER ID     IMAGE          COMMAND                CREATED          STATUS          PORTS                NAMES
bdc0d96b31b7     nginx          "nginx -g 'daemon of…"  14 seconds ago   Up 13 seconds    0.0.0.0:32768->80/tcp   boring_brattain
```

https://docs.docker.com/engine/reference/builder/#expose

ENV

```
ENV <key> <value>
ENV <key>=<value> ...
```

The ENV instruction sets the environment variable <key> to the value <value>. This value will be in the environment for all subsequent instructions in the build stage and can be replaced inline in many as well.

The ENV instruction has two forms. The first form, ENV <key> <value>, will set a single variable to a value. The entire string after the first space will be treated as the <value> - including whitespace characters. The value will be interpreted for other environment variables, so quote characters will be removed if they are not escaped.

The second form, ENV <key>=<value> ..., allows for multiple variables to be set at one time. Notice that the second form uses the equals sign (=) in the syntax, while the first form does not. Like command line parsing, quotes and backslashes can be used to include spaces within values.

For example:

```
ENV myName="John Doe" myDog=Rex\ The\ Dog \
    myCat=fluffy
```

and

```
ENV myName John Doe
ENV myDog Rex The Dog
ENV myCat fluffy
```

will yield the same net results in the final image.

The environment variables set using ENV will persist when a container is run from the resulting image. You can view the values using docker inspect, and change them using docker run --env <key>=<value>.

COPY

COPY has two forms:

- `COPY [--chown=<user>:<group>] <src>... <dest>`
- `COPY [--chown=<user>:<group>] ["<src>",... "<dest>"]` (this form is required for paths containing whitespace)

The `COPY` instruction copies new files or directories from `<src>` and adds them to the filesystem of the container at the path `<dest>`.

Multiple `<src>` resources may be specified but the paths of files and directories will be interpreted as relative to the source of the context of the build.

ADD

ADD has two forms:

- `ADD [--chown=<user>:<group>] <src>... <dest>`
- `ADD [--chown=<user>:<group>] ["<src>",... "<dest>"]` (this form is required for paths containing whitespace)

The `ADD` instruction copies new files, directories or remote file URLs from `<src>` and adds them to the filesystem of the image at the path `<dest>`.

Multiple `<src>` resources may be specified but if they are files or directories, their paths are interpreted as relative to the source of the context of the build.

Identique à COPY mais:
- **Supporte les URLs**
- **Extrait les archives !**

https://docs.docker.com/engine/reference/builder/#add

CMD

The `CMD` instruction has three forms:

- `CMD ["executable","param1","param2"]` (*exec* form, this is the preferred form)
- `CMD ["param1","param2"]` (as *default parameters to ENTRYPOINT*)
- `CMD command param1 param2` (*shell* form)

There can only be one `CMD` instruction in a `Dockerfile`. If you list more than one `CMD` then only the last `CMD` will take effect.

The main purpose of a `CMD` is to provide defaults for an executing container. These defaults can include an executable, or they can omit the executable, in which case you must specify an `ENTRYPOINT` instruction as well.

ENTRYPOINT

ENTRYPOINT has two forms:

- `ENTRYPOINT ["executable", "param1", "param2"]` (*exec* form, preferred)
- `ENTRYPOINT command param1 param2` (*shell* form)

An `ENTRYPOINT` allows you to configure a container that will run as an executable.

Un conteneur exécute:
[ENTRYPOINT] [CMD]

WORKDIR

```
WORKDIR /path/to/workdir
```
The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile. If the WORKDIR doesn't exist, it will be created even if it's not used in any subsequent Dockerfile instruction.

Exemple de Dockerfile

Alpine avec VIM

```
from alpine
RUN apk add vim
CMD ["vim"]
```

$ docker build . -t alpvim

```
Sending build context to Docker daemon  3.072kB
Step 1/3 : from alpine
 ---> 055936d39205
Step 2/3 : RUN apk add vim
 ---> Running in 21c3189729ac
fetch http://dl-cdn.alpinelinux.org/alpine/v3.9/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.9/community/x86_64/APKINDEX.tar.gz
(1/5) Installing lua5.3-libs (5.3.5-r2)
(2/5) Installing ncurses-terminfo-base (6.1_p20190105-r0)
(3/5) Installing ncurses-terminfo (6.1_p20190105-r0)
(4/5) Installing ncurses-libs (6.1_p20190105-r0)
(5/5) Installing vim (8.1.1365-r0)
Executing busybox-1.29.3-r10.trigger
OK: 40 MiB in 19 packages
Removing intermediate container 21c3189729ac
 ---> e47638747864
Step 3/3 : CMD ["vim"]
 ---> Running in ac86ae31b7a3
Removing intermediate container ac86ae31b7a3
 ---> ee258ef6e23e
Successfully built ee258ef6e23e
Successfully tagged alpvim:latest
```

$ docker run -ti alpvim

Ubuntu Serveur Web

from ubuntu

RUN apt-get update && \
    apt-get install nginx -y && \
    apt-get clean

EXPOSE 80/tcp

CMD ["nginx", "-g", "daemon off;"]

```
$ docker build . -t unginx
$ docker run -d -P --rm unginx
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5dd75be0ed8a	unginx	"nginx -g 'daemon of…"	18 seconds ago	Up 16 seconds	**0.0.0.0:32771->80/tcp**	exciting_ride

Debian Serveur Web

from **debian**

RUN apt-get update && \
    apt-get install nginx -y && \
    apt-get clean

EXPOSE 80/tcp

CMD ["nginx", "-g", "daemon off;"]

```
$ docker build . -t unginx
$ docker run -d -P --rm unginx
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5dd75be0ed8a	unginx	"nginx -g 'daemon of…"	18 seconds ago	Up 16 seconds	**0.0.0.0:32771->80/tcp**	exciting_ride

La Composition De Conteneurs
« docker-compose »

Considérons La Configuration Suivante

Comment démarrer deux conteneurs qui doivent partager des services ?

Docker Compose

docker-compose.yml

```yaml
version: '3.3'

services:
    myredis:
      image: redis
      restart: always
    web:
      depends_on:
         - myredis
      image: mywebsebver
      ports:
         - « 8080:80"
      restart: always
      volumes:
               - ./html/:/var/www/html/
```

Syntaxe YAML !

https://fr.wikipedia.org/wiki/YAML

Serveur Web et Redis

```
version: '3.3'

services:
  myredis:
    image: redis
    restart: always
  web:
    depends_on:
      – myredis
    image: mywebsebver
    ports:
      – « 8080:80"
    restart: always
    volumes:
            – ./html/:/var/www/html/
```

Conteneur base de donnée @ myredis

Conteneur « web » @ web

https://cheatography.com/tasjaevan/cheat-sheets/redis/

Redis

- Stockage clef-valeur avec des structure de donnée prédéfinies:
  ➡ List
  ➡ Hash
  ➡ Sets
  ➡ …
- Liste de commandes : https://cheatography.com/tasjaevan/cheat-sheets/redis/
- Interface extrêmement simple:
- Via netcat 127.0.0.1 6379
- Via redis-cli (du paquet redis-tools)

Exemple de Base

```yaml
version: '3.3'

services:
  myredis:
    image: redis
    restart: always

  client:
    depends_on:
      - myredis
    image: redis-cli
```

Conteneur base de donnée @ myredis

Script client @ client

Définition de redis-cli :

```dockerfile
FROM php:7.4.15-apache
RUN apt-get update && apt-get install -y redis-tools && apt-get clean

COPY ./script.sh /

CMD ["/bin/sh", "/script.sh"]
```

Exemple de Base

Contenu de script.sh :

```sh
#!/bin/sh


for i in $(seq 1 1 100)
do
    echo "SET $i"
    echo "SET key_$i 0" | redis-cli -h myredis
done



while true
do
    for i in $(seq 1 1 100)
    do
        echo "INCR $i"
        echo "INCR key_$i" | redis-cli -h myredis
    done
done
```

Le serveur tourne dans « myredis »

On lance

Depuis un dossier (ou sous dossier du *docker-compose.yml*)

« `docker-compose up` » (`-d` pour être en background)

```
$ docker-compose up
Creating network "redis_default" with the default driver
Creating redis_myredis_1 ... done
Creating redis_client_1  ... done
Attaching to redis_myredis_1, redis_client_1
myredis_1  | 1:C 01 Mar 2021 13:46:08.259 # oO0OoO00oO00o Redis is starting oO0
OoO00oO00o
myredis_1  | 1:C 01 Mar 2021 13:46:08.259 # Redis version=6.2.0, bits=64, commi
t=00000000, modified=0, pid=1, just started
myredis_1  | 1:C 01 Mar 2021 13:46:08.259 # Warning: no config file specified,
using the default config. In order to specify a config file use redis-server /p
ath/to/redis.conf
myredis_1  | 1:M 01 Mar 2021 13:46:08.260 * monotonic clock: POSIX clock_gettim
e
myredis_1  | 1:M 01 Mar 2021 13:46:08.260 * Running mode=standalone, port=6379.
myredis_1  | 1:M 01 Mar 2021 13:46:08.260 # WARNING: The TCP backlog setting of
 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lowe
r value of 128.
myredis_1  | 1:M 01 Mar 2021 13:46:08.260 # Server initialized
myredis_1  | 1:M 01 Mar 2021 13:46:08.260 # WARNING overcommit_memory is set to
 0! Background save may fail under low memory condition. To fix this issue add
'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the comma
nd 'sysctl vm.overcommit_memory=1' for this to take effect.
myredis_1  | 1:M 01 Mar 2021 13:46:08.260 * Ready to accept connections
client_1   | SET 1
client_1   | OK
client_1   | SET 2
client_1   | OK
```

Pour Stopper

Depuis un dossier (ou sous dossier du docker-compose.yml)

« docker-compose down »

```
$ docker-compose down
Stopping redis_client_1  ... done
Stopping redis_myredis_1 ... done
Removing redis_client_1  ... done
Removing redis_myredis_1 ... done
Removing network redis_default
```

Serveur Web et Redis

```
version: '3.3'

services:
  myredis:
    image: redis
    restart: always
    ports:
      - "6379:6379"
  web:
    depends_on:
      - myredis
    image: mywebsebver
    ports:
      - "8080:80"
    restart: always
    volumes:
            - ./html/:/var/www/html/
```

Conteneur base de donnée @ myredis

On expose le redis

Conteneur « web » @ web

Serveur Web et Redis

```
$ redis-cli
127.0.0.1:6379> GET key_12
"11"
```

On peut maintenant s'y connecter depuis la machine hôte !

Docker Compose Serveur WEB

```yaml
version: '3.3'

services:
  myredis:
    image: redis
    restart: always
  web:
    depends_on:
      - myredis
    image: mywebsebver
    ports:
      - « 8080:80"
    restart: always
    volumes:
          - ./html/:/var/www/html/
```

Conteneur base de donnée @ myredis

Conteneur « web » @ web

Considérons La Configuration Suivante

Notre exemple: compter les occurrences des mots dans « guerre et Paix », stocker cette information dans une base de donnée « clef-valeur » et afficher le tout via une page web propulsée par PHP.

La Partie Python

Dockerfile pour python-redis

```
FROM python:3
RUN pip install redis
WORKDIR /scripts/
CMD python3 /scripts/client.py
```

docker-compose.yml

```
pythonclient:
 depends_on:
    - myredis
 image: python-redis
 restart: "no"
 volumes:
    - ./python_dir/:/scripts/
```

On redémarre pas si le script se stop

La Partie Python (Script)

client.py

```python
import redis
import string

r = redis.Redis(host="myredis")

# Read the full file
with open("getp.txt") as f:
    data=f.read()

# This will eventually contain
# all words
words = []

# Split on lines
lines=data.split("\n")

# Iterate on lines
for l in lines:
    # This hack removes punctuation
    l = l.translate(str.maketrans('', '', string.punctuation))
    # Now split on spaces
    words += l.split(" ")

# This will add all words to the redis
for w in words:
    print(w)
    r.hincrby("words", w, 1)
```

La Partie Redis

docker-compose.yml

```
myredis:
  image: redis
  restart: always
  #ports:
  #  - "6379:6379"
```

Note: on utilise l'image officielle sans modification !

La Partie Serveur Web PHP

Dockerfile pour php-redis

```
FROM php:7.4.15-apache
RUN /usr/local/bin/pecl install redis-5.1.1 && docker-php-ext-enable redis
```

docker-compose.yml

```yaml
web:
  depends_on:
    - myredis
  image: php-redis
  ports:
    - "8080:80"
  restart: always
  volumes:
        - ./php_dir/:/var/www/html/
```

La Partie Serveur Web PHP

Le script index.php

```php
<html>
<body>

<?php
$redis = new Redis();
$redis->connect('myredis', 6379);
$range = $redis->hkeys("words");

$words = array();

foreach($range as $e)
{
    $count = $redis->hget("words", $e);
    array_push($words, array("word" => $e, "count" => $count));
}

function cmp($a, $b)
{
    return $b["count"] - $a["count"];
}

usort($words, "cmp")

?>

<h1>REDIS</h1>

<table>
<?php
foreach($words as $e)
{
    echo "<tr><td>".$e['word']."</td><td>".$e["count"]."</td></tr>";
}
?>
</table>

</body>
</html>
```

On Lance !

```
$ docker-compose up
Starting php_myredis_1 ... done
Starting php_web_1          ... done
Starting php_pythonclient_1 ... done
Attaching to php_myredis_1, php_web_1, php_pythonclient_1
myredis_1      | 1:C 01 Mar 2021 14:09:23.876 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
myredis_1      | 1:C 01 Mar 2021 14:09:23.876 # Redis version=6.2.0, bits=64, commit=00000000,
 modified=0, pid=1, just started
myredis_1      | 1:C 01 Mar 2021 14:09:23.876 # Warning: no config file specified, using the d
efault config. In order to specify a config file use redis-server /path/to/redis.conf
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 * monotonic clock: POSIX clock_gettime
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 * Running mode=standalone, port=6379.
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 # WARNING: The TCP backlog setting of 511 cannot
 be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 # Server initialized
web_1          | AH00558: apache2: Could not reliably determine the server's fully qualified d
omain name, using 172.29.0.3. Set the 'ServerName' directive globally to suppress this message
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 # WARNING overcommit_memory is set to 0! Backgro
und save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1'
to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this
 to take effect.
web_1          | AH00558: apache2: Could not reliably determine the server's fully qualified d
omain name, using 172.29.0.3. Set the 'ServerName' directive globally to suppress this message
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 * Loading RDB produced by version 6.2.0
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 * RDB age 2501 seconds
myredis_1      | 1:M 01 Mar 2021 14:09:23.877 * RDB memory usage when created 1.80 Mb
web_1          | [Mon Mar 01 14:09:24.605510 2021] [mpm_prefork:notice] [pid 1] AH00163: Apach
e/2.4.38 (Debian) PHP/7.4.15 configured -- resuming normal operations
web_1          | [Mon Mar 01 14:09:24.605563 2021] [core:notice] [pid 1] AH00094: Command line
: 'apache2 -D FOREGROUND'
myredis_1      | 1:M 01 Mar 2021 14:09:23.893 * DB loaded from disk: 0.016 seconds
myredis_1      | 1:M 01 Mar 2021 14:09:23.893 * Ready to accept connections
pythonclient_1 | The
pythonclient_1 | Project
pythonclient_1 | Gutenberg
pythonclient_1 | EBook
pythonclient_1 | of
pythonclient_1 | La
pythonclient_1 | guerre
```

La sortie sur http://127.0.0.1:8080

REDIS

de	15275
	10910
et	10894
la	8772
le	8215
à	7933
en	5324
les	4615
se	3929
que	3781
il	3568
qui	3564
un	3463
son	3413
lui	3177
ne	2869
pas	2661
des	2595
sa	2553
du	2401
une	2282
dans	2174
pour	2150