Table des matières

Installation de Docker

Prérequis

1. Internet sur votre VM
2. Debian 11 Bullseye
3. Utilisateur dans sudoers
4. Accès au terminal

Dépendances

Il vous faudra installer les dépendances, avant de pouvoir installer Docker!

```
sudo apt update
```

Puis:

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg
```

Clé GPG Docker

Ces clé sont nécéssaire afin de pouvoir ajouter le dépôt et contrôle les dépendances de paquet.

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Dépôt docker

De cette façon nous pouvons ajouter en une seule commande l'accès au dépôt.

```
echo \
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Mettez à jour les sources:

```
sudo apt update
```

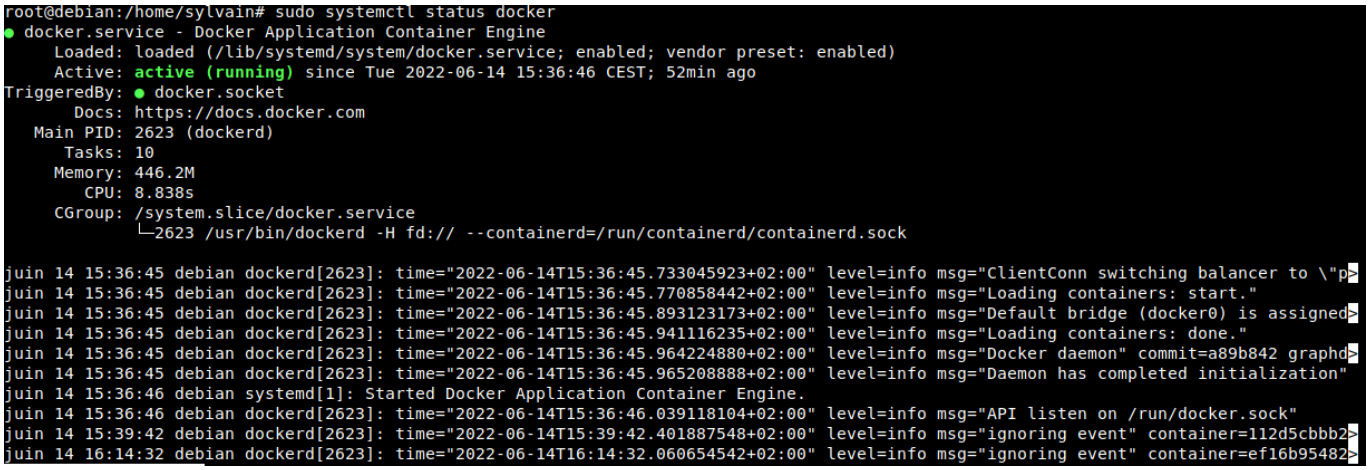
Installer Docker

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Service status

Vérifier qu'après l'installation celui-ci soit bien executer.

```
sudo systemctl status docker
```



Start/Stop

Voici comment lancer ou stopper manuellement Docker.

```
sudo systemctl start docker
```

```
sudo systemctl stop docker
```

Lancer Docker

Afin de ne plus passer par l'appel de la commande sudo, voici ce qu'il faut faire:

```
sudo usermod -aG docker ${USER}
```

Installer l'Autocomplétion

```
sudo apt update
sudo apt install bash-completion curl
sudo mkdir /etc/bash_completion.d/
sudo curl -L https://raw.githubusercontent.com/docker/docker-
ce/master/components/cli/contrib/completion/bash/docker -o
```

```
/etc/bash_completion.d/docker.sh
sudo curl -L
https://raw.githubusercontent.com/docker/compose/1.24.1/contrib/completion/b
ash/docker-compose -o /etc/bash_completion.d/docker-compose
```

Désinstaller Docker

Cette commande désinstallera docker et ces composants (ceux de l'installation)

```
sudo apt purge docker-ce docker-ce-cli containerd.io
```

Il vous restera à supprimer les répertoires de config et de conteneurs.

```
sudo rm -rf /var/lib/docker
sudo rm -rf /var/lib/containerd
```

Lister les images

```
docker images
```

ou

```
docker image ls
```

```
root@debian:~# docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
hello-world         latest    feb5d9fea6a5   8 months ago    13.3kB
riftbit/serviio     latest    acd769d353e0   18 months ago   360MB
root@debian:~#
```

Construire une image

Simple

Pour construire une image on utilise le format Dockerfile.

```
nano DockerFile
```

On y colle le contenu suivant :

```
FROM alpine:3.10.2
RUN apk update
RUN apk upgrade
```

Lancer la Construction

Pour lancer la construction de l'image :

```
docker build .
```

ou

```
docker builf -f Dokerfile .
```

```
Sending build context to **Docker** daemon  2.048kB
Step 1/3 : FROM alpine:3.10.2
 ---> 961769676411
Step 2/3 : RUN apk update
 ---> Running in ff0b9c41c6e5
fetch http://dl-cdn.alpinelinux.org/alpine/v3.10/ma
in/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.10/co
mmunity/x86_64/APKINDEX.tar.gz
v3.10.2-80-g68e4e4a13a [http://dl-cdn.alpinelinux.o
rg/alpine/v3.10/main]
v3.10.2-83-g64319a6606 [http://dl-cdn.alpinelinux.o
rg/alpine/v3.10/community]
OK: 10336 distinct packages available
Removing intermediate conteneur ff0b9c41c6e5
 ---> 497b57586909
Step 3/3 : RUN apk upgrade
 ---> Running in 874225a869dd
(1/2) Upgrading libcrypto1.1 (1.1.1c-r0 -> 1.1.1d-r
0)
(2/2) Upgrading libssl1.1 (1.1.1c-r0 -> 1.1.1d-r0)
OK: 6 MiB in 14 packages
Removing intermediate conteneur 874225a869dd
 ---> 9c5a9a6e761f
Successfully built 9c5a9a6e761f
```

On voit qu'il réalise les opérations demandées et que pour chaque instruction du fichier Dockerfile il indique à partir de quelle image ID il réalise la commande demandée.

```
    FROM alpine : 961769676411 qui correspond à celui de l'image téléchargée précédemment.  \\
    RUN apk update : ff0b9c41c6e5
    RUN apk upgrade : 874225a869dd
    Successfully built : 9c5a9a6e761f
```

Info importante
Docker par défaut détruit les conteneurs intermédiaires. Donc si on lance la commande Docker images on doit obtenir ce ID dans la liste retournée :

REPOSITORY TAG IMAGE ID CREATED SIZE 9c5a9a6e761f 5 minutes ago 10.2MB

On voit que cette image n'a pas de TAG. Pour taguer une image lors de la construction il suffit d'ajouter l'option -t tag:xxx

Il est possible d'ajouter un tag pendant la construction de l'image ou bien après, si vous avez oublier de le faire pendant.

```
docker build . -t test:0.1
...
docker images
REPOSITORY                      TAG
IMAGE ID            CREATED              SIZE
test                            0.1
9c5a9a6e761f        7 minutes ago        10.2MB
```

```
docker tag 9c5a9a6e761f test:`latest`

REPOSITORY                      TAG
IMAGE ID            CREATED              SIZE
test                            0.1
9c5a9a6e761f        9 minutes ago        10.2MB
test                            `latest`
9c5a9a6e761f        9 minutes ago        10.2MB
```

Lancer un conteneur

```
docker run -d -p 8895:80 riftbit/serviio
```

Voir le Logs

executer un **docker ps** pour voir les conteneur actif et relever l'ID.

```
sylvain@debian:~$ docker ps
```

```
CONTAINER ID   IMAGE             COMMAND                  CREATED
STATUS          PORTS
NAMES
bca1b049cd33   riftbit/serviio   "/bin/sh -c 'tail -f…"   24 minutes ago
Up 24 minutes   8895/tcp, 23423-23424/tcp, 1900/udp, 23523-23524/tcp,
0.0.0.0:8895->80/tcp, :::8895->80/tcp   funny_saha
```

Pour avoir le log executer :

```
docker logs bca1b049cd33
```

Vous devirer avoir une sortie du genre:

```
2022-06-15 13:39:46,323 INFO  [MediaServer] ----------------------------------
----------------------------------------
2022-06-15 13:39:46,324 INFO  [MediaServer] Serviio DLNA media streaming
server v 2.1 (rev. 64e78caaeacb5eafe11b93197db80b99f812bfb8)
2022-06-15 13:39:46,324 INFO  [MediaServer] Petr Nejedly 2009-2018
2022-06-15 13:39:46,324 INFO  [MediaServer] http://www.serviio.org
2022-06-15 13:39:46,324 INFO  [MediaServer]
2022-06-15 13:39:46,324 INFO  [MediaServer] Java 1.8.0_272-IcedTea amd64
2022-06-15 13:39:46,324 INFO  [MediaServer] OS Linux 5.10.0-15-amd64
2022-06-15 13:39:46,324 INFO  [MediaServer] File encoding: UTF-8
2022-06-15 13:39:46,363 INFO  [MediaServer] Headless mode enabled: true
2022-06-15 13:39:46,363 INFO  [MediaServer] User: root
2022-06-15 13:39:46,363 INFO  [MediaServer] User home dir: /root
2022-06-15 13:39:46,363 INFO  [MediaServer] Temp dir: /tmp
2022-06-15 13:39:46,363 INFO  [MediaServer] ----------------------------------
----------------------------------------
2022-06-15 13:39:46,372 INFO  [DBSchemaUpdateExecutor] Checking if DB schema
needs to be updated
2022-06-15 13:39:46,380 INFO  [DatabaseManager] Using DERBY database
language
2022-06-15 13:39:47,418 INFO  [DBSchemaUpdateExecutor] Updating DB schema
```

Entrer dans un conteneur

Exécuter une console Bash depuis mon conteneur:

```
docker exec -it nom-de-mon-conteneur bash
```

Commandes Docker

info

```
docker info
```

affiche plein d'information sur l'engine avec lequel vous êtes en contact

Etat conteneur

```
docker ps
```

affiche les conteneurs en train de tourner

```
docker ps -a
```

affiche également les conteneurs arrêtés

Version

```
docker --version
```

Rechercher une image

```
docker search imageName
```

ex: pour cacti

```
root@debian:~# docker search cacti
NAME                          DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
smcline06/cacti               Cacti v1+ in docker. Feedback and pull reque…   29                   [OK]
polinux/cacti                 Cacti Server with HTTP2/support and SSL term…   18                   [OK]
admindean/cacti               Cacti 1.2.20 with weathermap,thold and monit…   7
leniy/cacti                                                                   5                    [OK]
chestersgarage/cacti          Cacti all-in-one container, based on Alpine …   5                    [OK]
giuseppeiannelli/cacti        Cacti network manager rrdtool-based. This im…   4                    [OK]
thedollarsign/cacti           Cacti® - The Complete RRDTool-based Graphing…   3                    [OK]
krknopp/cacti                                                                 1
idle/cacti-1-minute           Cacti with 1 minute poller. Percona template…   1                    [OK]
cactiw/jira_notify                                                            0
oems/cacti                    Cacti is a complete network graphing solutio…   0                    [OK]
i5js/cacti_centos                                                             0
cactice/jupyter-evcxr-rust                                                    0
vault/cacti                   cacti for docker                                0                    [OK]
daduy/cacti                   Cacti on a docker! (Ubuntu + Apache2 + Mysql…   0
irasnyd/cacti                 Dockerized Cacti RRDTool network monitoring …   0                    [OK]
cactice/sverchok                                                              0
cacticouncil/pidgin-builders  Builder containers for Pidgin 3                 0
fblgit/cacti                  Configurable Cacti Instance of Frontend with…   0                    [OK]
cactiw/rocket-bot                                                             0
analogic/cacti                Full Cacti installation (S6+Apache+Cacti+Cac…   0                    [OK]
vyg1/cacti                    get cacti running with one command              0
fblgit/cacti-db               Cacti Database                                  0                    [OK]
analogic/cacti-thold          Docker container with full Cacti installatio…   0                    [OK]
mrlesmithjr/cacti             Build [Docker] image for [Cacti]..              0                    [OK]
root@debian:~#
```

Installer une image

```
docker pull imageName
```

Run une image

```
docker run -it imageName
```

Nettoyage system

Voir ce qui prend de la place

```
docker system df
docker system df -v  # version détaillée
```

Voir les images par taille

```
docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.Size}}" | sort -k 3 -h
```

Supprimer les containers arrêtés

```
docker container prune
```

Supprimer les images sans tag (dangling)

```
docker image prune
```

Supprimer les networks non utilisés

```
docker network prune
```

Supprimer les volumes non utilisés

```
docker volume prune
```

Tout faire d'un coup (safe)

```
docker system prune
```

Full Command Line

```
docker --help
```

```
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default
"/home/sylvain/.docker")
  -c, --context string     Name of the context to use to connect to the
daemon (overrides DOCKER_HOST env var and default
                           context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
"/home/sylvain/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default
"/home/sylvain/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default
"/home/sylvain/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  app*        Docker App (Docker Inc., v0.9.1-beta3)
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.8.2-docker)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  scan*       Docker Scan (Docker Inc., v0.17.0)
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a
running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local
```

```
filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's
filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem
image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT
by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage
statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit
codes

Run 'docker COMMAND --help' for more information on a command.
```

Commandes Docker Image

```
docker image --help
```

```
Usage:  docker image COMMAND

Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem
image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT
by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

API

Note: après chaque mise à jour du Docker, il faudra remettre ce paramètre s'enlève à chaque fois.

Activer l'API pour le gestion depuis un poste distant:

```
nano /lib/systemd/system/docker.service
```

Ajouter le paramètre

```
-H=tcp://0.0.0.0:2375
```

après la ligne **ExecStart=/usr/bin/dockerd**

```
ExecStart=/usr/bin/dockerd -H fd:// --
```

```
containerd=/run/containerd/containerd.sock -H=tcp://0.0.0.0:2375
```

-H=tcp:0.0.0.0:2375 permet à n'importe quel hôte d'interroger l'API !
*-H=tcp:*192.168.5.33:2375 permet seulement un hôte en particulier d'interroger l'API !

Recharger le service Docker:

```
systemctl daemon-reload
```

Relancer docker:
system initv:

```
sudo service docker restart
```

systemd:

```
systemctl restart docker.service
```

Test depuis une machine distante:

```
docker -H=IP-du-serveur-docker:2375 stats
```

retourne:

```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
b4c6fcf5f5db        adguardhome         0.03%               181.1MiB / 1.908GiB   9.27%               141kB / 28.6kB      0B / 8.19kB         10
f08a2b93c190        portainer           0.06%               12.22MiB / 1.908GiB   0.63%               560kB / 822kB       8.19kB / 0B         7
```

Script

Voici un Script qui permet d'automatiser la modification:

```bash
#!/bin/bash

# Chemin vers le fichier de configuration
FILE="/lib/systemd/system/docker.service"
SEARCH_VALUE="-H=tcp://0.0.0.0:2375"

# Vérifier si le fichier existe
if [ ! -f "$FILE" ]; then
  echo "Erreur : Le fichier $FILE n'existe pas."
  exit 1
```

```
fi

# Vérifier si la valeur est déjà présente
if grep -qF -- "$SEARCH_VALUE" "$FILE"; then
  echo "La valeur $SEARCH_VALUE est déjà présente dans le fichier."
else
  # Ajouter la valeur à la ligne concernée
  sudo sed -i '/ExecStart=\/usr\/sbin\/dockerd -H fd:\/\/ --
containerd=\/run\/containerd\/containerd.sock \$DOCKER_OPTS/s/$/
'"$SEARCH_VALUE"'/' "$FILE"

  # Redémarrer le service systemd pour prendre en compte les modifications
  sudo systemctl daemon-reload
  sudo systemctl restart docker

  echo "Modification terminée et service Docker redémarré."
fi
```

Dépannage

Si après une mise à jour de votre docker aucun conteneur n'apparait

```
docker info | grep "Storage Driver"
```

Doit afficher : “Storage Driver: vfs”
Si c'est pas le cas :

```
sudo nano /etc/docker/daemon.json
```

```
{
  "storage-driver": "vfs",
  "data-root": "/var/lib/docker"
}
```

Permet de force le driver par defaut pour vos Virtual File System (Layer)
