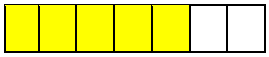
Outils

Version 1.0.0
Niveau requis : 5/7



Conteneurisation avec Docker (sous Linux MINT)

Sommaire

I. Préambule

I.I. *Objet*

L'objet de ce document est de présenter succinctement l'usage de Docker et comment s'en servir.

Dans la première version de ce document nous n'aborderons pas l'utilisation de docker-compose qui permet de relier / associer différents conteneurs pour segmenter les outils indépendants.

Nous entrerons ici dans l'exemple de mise en place d'un unique conteneur contenant toutes les briques permettant d'initialiser et fournir un site Drupal (hors sécurisation SSL) en localhost.

I.II. *Pré-requis*

Avoir suivi le cours sur la Virtualisation d'OS avec la mise en place de Linux MINT sous Virtual Box.

Un complément : après avoir suivi le cours de Linux MINT, il est recommandé de lancer l'installation de Linux MINT avec les paramètres de base.

Bien se souvenir pendant l'installation du mot de passe admin que vous devrez définir.

I.III. *Versions du document*

Version	Date	Auteur	Description
1.0.0	11/08/2022	Péquignat.eu	Création du document

I.IV. *Documents de référence*

#	Document	Version	Auteur(s)
[R1]	Virtualisation v1.0.0.pdf	1.0.0	Péquignat.eu
[R2]	https://linuxhint.com/install_docker_linux_mint/		Adnan Shabbir
[R3]	https://devopssec.fr/article/cours-complet-apprendre-technologie-docker#begin-article-section		ajdaini-hatim

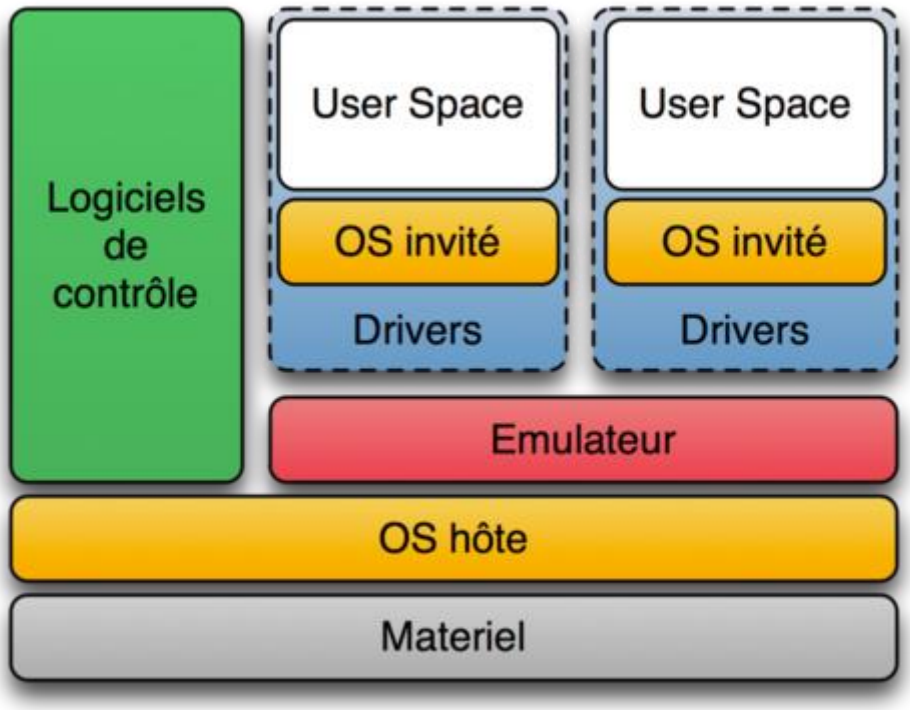
II. Virtualisation VS Containerisation

II.I. *Virtualisation avantages et limitations*

La virtualisation a pour principe d'exploiter sur une machine physique, un logiciel qui va permettre de lancer des Machines Virtuelles avec chacun leur propre OS et paramétrage.

Un des avantages, c'est que comme les machines virtuelles sont stockées sous forme de fichier il est facilement réalisable de dupliquer, reprendre, sauvegarder l'intégralité de la machine virtuelle.

L'inconvénient : c'est que toutes les machines virtuelles installé sur la même et unique machine physique doivent donc se répartir la puissance de cette machine physique. De plus, les accès réseaux ne sont pas non plus à laisser de côté, car bien souvent il n'y a que très peu de carte réseaux sur ces machines physique et surement pas une carte réseau par machine virtuelle. Donc si ces machines virtuelles consomment chacune une très grande bande passante, le réseau va ralentir sur chacune de ces machines.

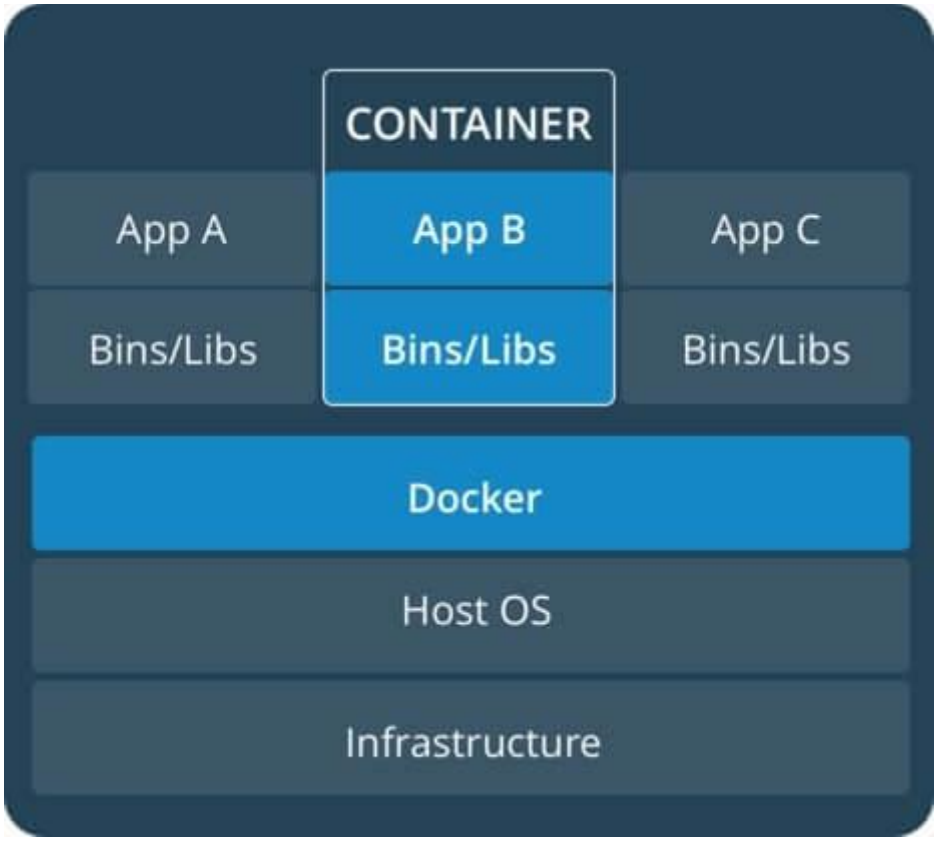


II.II. *Containerisation*

La containerisation, vient pour avoir l'intégralité des dépendances que nécessite un programme en termes de logiciels et bibliothèques d'outils permettant l'usage d'un programme.

Avec la containerisation, il est possible de définir la puissance maximale que peut consommer une application englobée dans un conteneur. L'avantage, c'est que cela permet d'être moins gourmand en ressource et en licence que de devoir lancer une VM complète par programme développé.

Aussi, la conteneurisation permet de limiter les ressources consommées tout en facilitant le déploiement avec toutes les dépendances d'un programme.



III. Notions nécessaire de connaître sous Linux

III.I. *Les namespaces*

Les namespaces sous Linux permettent d'isoler la vue qu'on les processus sur leurs ressources. Aussi il est possible de définir un namespace permettant de limiter la vue sur le système par les processus intégrés dans ce namespace.

Il existe plusieurs types de nanespaces :

- PID : Un entier définissant le processus. Celui-ci peut lancer des sous processus. En tuant le processus parent de PID par exemple 1 cela tue tous les processus descendants
- IPC : Permet de bloquer l'échange d'information entre processus
- NET : Tout ce qui est en rapport avec le réseau : IP, Routage, Parfeux etc…
- MOUNT : permettre de monter ou démonter des espaces disque ou réseau d'un autre système sans impacter le système hôte
- USER : Défini les utilisateurs
- UTS : associe un nom d'hôte et de domaine au processus pour avoir son propre hostname.

III.II. *cgroups*

Les cgroups permettent de limiter les ressources allouées à un groupe.

Quelques types de cgroup :

- cgroup cpuset : assigne des processeurs individuels et des nœuds de mémoire à des groupes de contrôle
- cgroup cpu : planifie un accès aux ressources du processeur
- cgroup cpuacct : génère des rapports sur les ressources du processeur utilisées
- cgroup devices : autorise ou refuse l'accès aux périphériques
- cgroup net_prio : permet de définir la priorité du trafic réseau
- cgroup memory : définit la limite d'utilisation de la mémoire
- cgroup blkio : limite de la vitesse E/S (lecture/écriture) sur périphériques de type bloc (ex : disque dur)
- cgroup pid : limite le nombre de processus

Allé pour s'amuser encore un peu plus, créons un cgroup qui limite la mémoire !

Commençons par créer un cgroup limitant l'utilisation de la mémoire :

```
sudo cgcreate -a <nom_d_utilisateur> -g memory:<nom_du_cgroup>
```

Voyons ce qu'il y a dedans :

```
ls -l /sys/fs/cgroup/memory/<nom_du_cgroup>/
```

Résultat :

```
-rw-r--r-- 1 <nom_d_utilisateur> root 0 Okt 10 23:16 memory.kmem.limit_in_bytes
-rw-r--r-- 1 <nom_d_utilisateur> root 0 Okt 10 23:14
memory.kmem.max_usage_in_bytes
```

Ensuite, on va limiter notre cgroup à 20 mégaoctets :

```
sudo echo 20000000 >
/sys/fs/cgroup/memory/<nom_du_cgroup>/memory.kmem.limit_in_bytes
```

Maintenant utilisons notre cgroup sur notre programme bash :

```
sudo cgexec  -g memory:<nom_du_cgroup> bash
```

Voilà le processus bash ne peut plus dépasser 20 Mo de mémoire.

IV.Installation Docker

Une fois que vous avez réalisé l'installation de Linux MINT

Lancer dans un terminal la commande :

```
sudo apt update
```

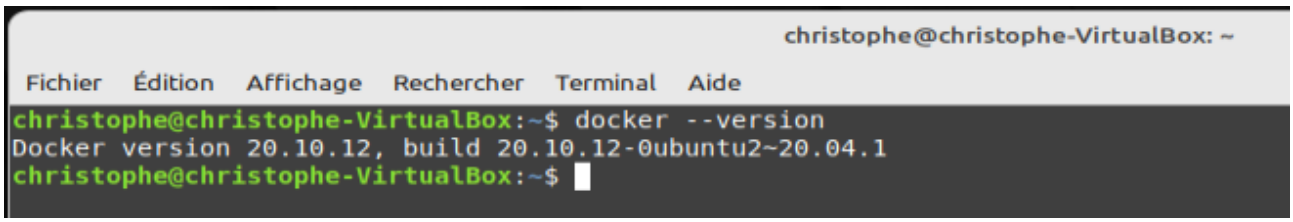
Puis lancer l'installation des packages de docker :

```
sudo apt install docker*
```

Il vous ait demandé de confirmer l'installation : O (Si français) ou Y (Si Anglais)

Vérifier la version installée :

```
docker --version
```



V. Activation de docker

```
sudo systemctl start docker
```

```
christophe@christophe-VirtualBox:~$ sudo systemctl start docker
[sudo] Mot de passe de christophe :
christophe@christophe-VirtualBox:~$
```

Pour activer le service docker automatiquement après un boot

```
sudo systemctl enable docker
```

```
christophe@christophe-VirtualBox:~$ sudo systemctl enable docker
christophe@christophe-VirtualBox:~$
```

Pour tester que docker fonctionne bien, lancer la commande :

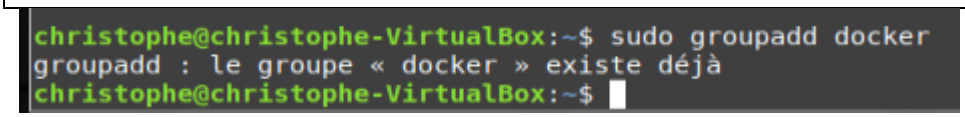
```
sudo docker run hello-world
```

```
christophe@christophe-VirtualBox:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dca625
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

christophe@christophe-VirtualBox:~$
```

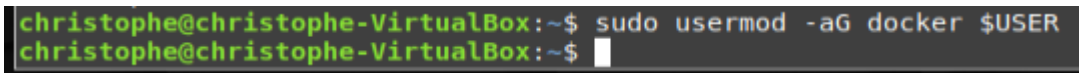
VI. Astuce Linux

Afin d'éviter à devoir faire systématiquement un sudo pour lancer une commande docker, il est possible d'ajouter dans le groupe docker l'utilisateur courant.

```
sudo groupadd docker
```




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

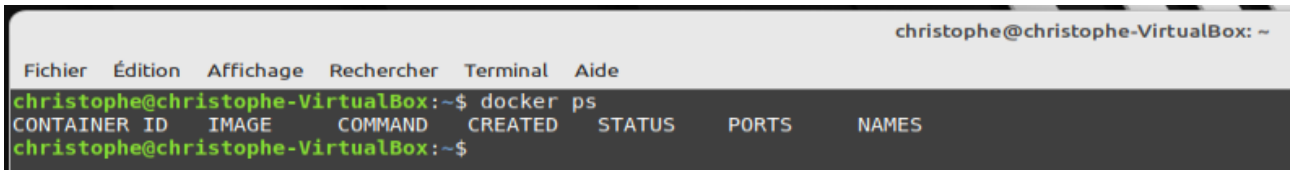


Il faut redémarrer la VM pour prise en compte.

```
sudo reboot
```

Test de bon fonctionnement

```
docker ps
```



VII. Image docker

Sur Docker, un conteneur est lancé en exécutant une image.

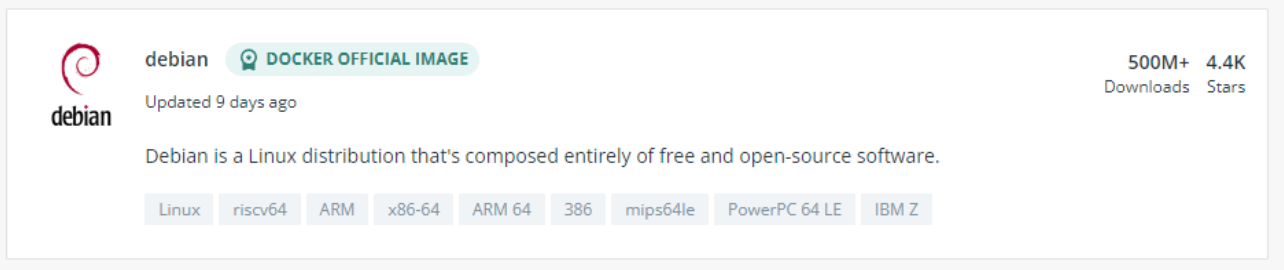
Une image est un package qui inclut tout ce qui est nécessaire à l'exécution d'une application, à savoir :

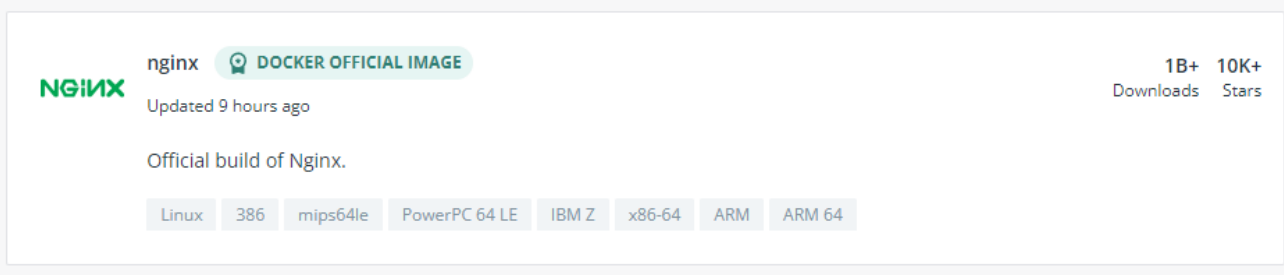
- Le code
- L'exécution
- Les variables d'environnement

- Les bibliothèques
- Les fichiers de configuration

Par la suite nous verrons plus sur les conteneurs. Pour le moment tout ce que vous devez retenir, c'est qu'une image docker est créée à partir d'un fichier nommé le Dockerfile. Une image est un modèle composé de plusieurs couches, ces couches contiennent notre application ainsi que les fichiers binaires et les bibliothèques requises. Lorsqu'une image est instanciée, son nom est un conteneur, un conteneur est donc une image en cours d'exécution.

Pour avoir un exemple de liste d'images disponibles officielles :
https://hub.docker.com/search?image_filter=official&q=&type=image&operating_system=linux&architecture=amd64&page=1





VII.I. *Récupérer des informations*

Pour commencer on va d'abord récupérer la liste des commandes possible :

```
docker help
```

```
christophe@christophe-VirtualBox:~$ docker help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/christophe/.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/christophe/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/christophe/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/christophe/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  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
  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.

To get more help with docker, check out our guides at https://docs.docker.com/go/guides/
christophe@christophe-VirtualBox:~$
```

```
docker info
```

```
christophe@christophe-VirtualBox:~$ docker info
Client:
 Context:    default
 Debug Mode: false

Server:
 Containers: 1
  Running: 0
  Paused: 0
  Stopped: 1
 Images: 1
 Server Version: 20.10.12
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runtime.v1.linux runc io.containerd.runc.v2
 Default Runtime: runc
 Init Binary: docker-init
 containerd version:
 runc version:
 init version:
 Security Options:
  apparmor
  seccomp
   Profile: default
 Kernel Version: 5.4.0-122-generic
 Operating System: Linux Mint 20.3
 OSType: linux
 Architecture: x86_64
 CPUs: 4
 Total Memory: 3.839GiB
 Name: christophe-VirtualBox
 ID: C4GC:PMP5:ER27:3Y6D:MZ2M:ILJ4:7NXP:DXSR:ZCBY:2WJG:HAT6:RCJS
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

WARNING: No swap limit support
christophe@christophe-VirtualBox:~$
```

```
docker image ls
```

```
christophe@christophe-VirtualBox:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
hello-world   latest    feb5d9fea6a5   10 months ago   13.3kB
christophe@christophe-VirtualBox:~$
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
Le titre REPOSITORY peut porter à confusion, c'est essentiellement le nom de l'image.	un tag ici est une façon de faire référence à votre image, ils sont utilisés principalement pour affecter une version à une image	L'identifiant de l'image (unique pour chaque image téléchargée)	Date de la dernière modification de l'image	Taille de l'image

VII.II. *Supprimer une image Docker*

Maintenant si on souhaite supprimer une image Docker, on aura besoin soit de son IMAGE ID soit de son nom. Une fois qu'on aura récupéré ces informations, on peut passer à la suite en lançant la commande suivante :

avec l'id de l'image :

```
christophe@christophe-VirtualBox:~$ docker rmi feb5d9fea6a5
```

avec le nom de l'image :

```
christophe@christophe-VirtualBox:~$ docker rmi hello-world
```

Une erreur apparait !?

```
christophe@christophe-VirtualBox:~$ docker rmi hello-world
Error response from daemon: conflict: unable to remove repository reference "hello-world" (must force) - container 8038b1ca0d48 is using its referenced image feb5d
9fea6a5
christophe@christophe-VirtualBox:~$
```

Ce message d'erreur nous explique, qu'on ne peut pas supprimer notre image Docker car des conteneurs ont été instanciés depuis notre image "hello-world". En gros si on jamais on supprime notre image "hello-world", ça va aussi supprimer nos conteneurs car ils se basent sur cette image. Dans notre cas ça ne pose aucun problème, d'ailleurs pour résoudre ce problème l'erreur nous informe qu'on doit forcer la suppression pour éliminer aussi les conteneurs liés à notre image (must force).

```
christophe@christophe-VirtualBox:~$ docker rmi -f hello-world
```

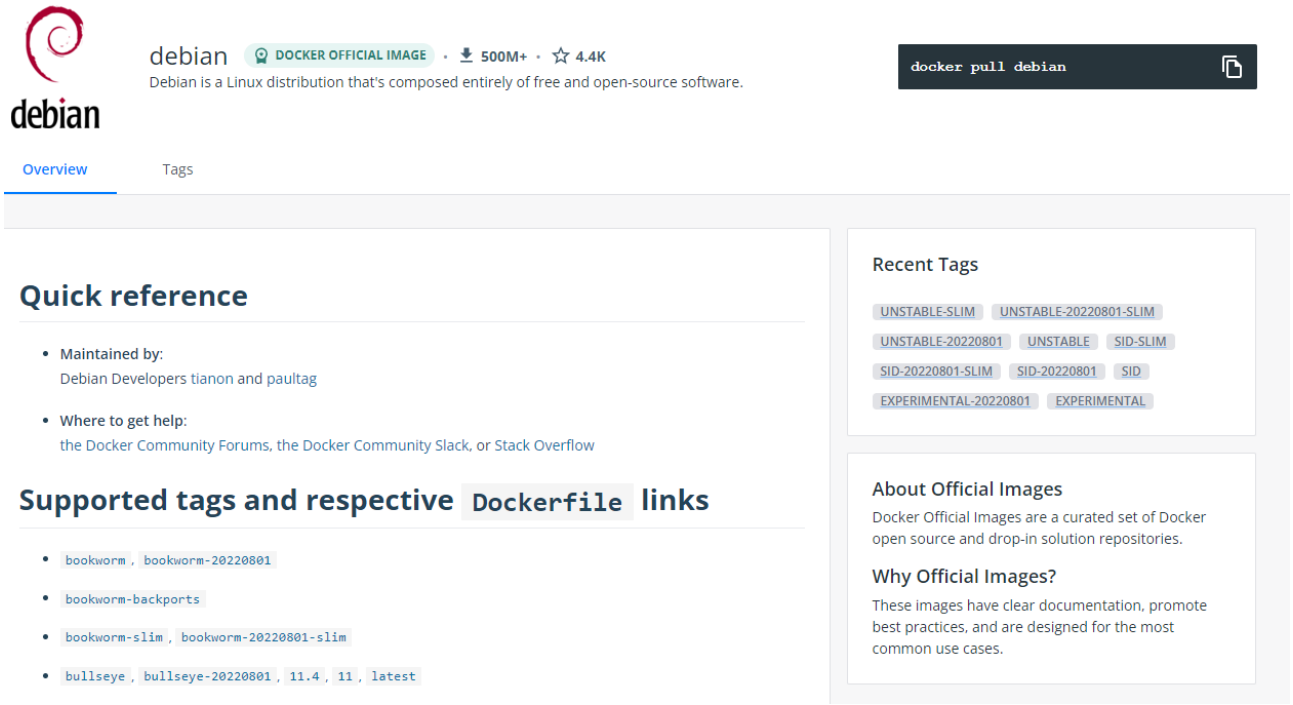
```
christophe@christophe-VirtualBox:~$ docker rmi -f hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dca625
Deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
christophe@christophe-VirtualBox:~$
```

```
christophe@christophe-VirtualBox:~$ docker image ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
christophe@christophe-VirtualBox:~$
```

VIII. Installation images docker

VIII.I. *Installation image debian*

https://hub.docker.com/_/debian



En haut à droite la ligne de commande à lancer pour installer l'image debian.

```
docker search debian
```

```
christophe@christophe-VirtualBox:~$ docker search debian
NAME                              DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
ubuntu                            Ubuntu is a Debian-based Linux operating sys…   14752     [OK]
debian                            Debian is a Linux distribution that's compos…   4401      [OK]
neurodebian                       NeuroDebian provides neuroscience research s…   92        [OK]
bitnami/debian-base-buildpack     Debian base compilation image                   2                    [OK]
treehouses/debian                                                                 2
ustclug/debian                    Official Debian Image with USTC Mirror          1
osrf/debian_armhf                 Debian Armhf Base Images                        1
osrf/debian_arm64                 Debian arm64 Base Images                        1
rancher/debianconsole                                                             1
dokken/debian-11                  Debian 11 image for use with kitchen-dokken     0
dokken/debian-8                   EOL: Debian 8 image for kitchen-dokken          0
dokken/debian-10                  Debian 10 image for use with kitchen-dokken     0
dokken/debian-9                   Debian 9 image for kitchen-dokken               0
corpusops/debian-bare             https://github.com/corpusops/docker-images/     0
corpusops/debian                  debian corpusops baseimage                      0
datadog/debian-i386                                                               0
mirantis/debian-build-ubuntu-trusty                                               0
mirantis/debian-build-ubuntu-xenial                                               0
dokken/debian-7                   EOL DISTRO: For use with kitchen-dokken, Bas…   0
treehouses/debian-tags                                                            0
dokken/debian-12                                                                  0
galaxy/debian32-wheel                                                             0
galaxy/debian-wheel                                                               0
galaxy/debian32                                                                   0
apache/couchdbci-debian           Apache CouchDB CI Debian                        0
christophe@christophe-VirtualBox:~$
```

Pour voir que les images officielles :

```
docker search --filter "is-official=true" debian
```

```
christophe@christophe-VirtualBox:~$ docker search --filter "is-official=true"  debian
NAME          DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
ubuntu        Ubuntu is a Debian-based Linux operating sys…   14752     [OK]
debian        Debian is a Linux distribution that's compos…   4401      [OK]
neurodebian   NeuroDebian provides neuroscience research s…   92        [OK]
christophe@christophe-VirtualBox:~$
```

Télécharger l'image debian :

```
docker pull debian
```

```
christophe@christophe-VirtualBox:~$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
001c52e26ad5: Pull complete
Digest: sha256:82bab30ed448b8e2509aabe21f40f0607d905b7fd0dec72802627a20274eba55
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest
christophe@christophe-VirtualBox:~$
```

IX. Fonctionnement et exploitation des conteneurs Docker

IX.I. *Différence entre image et conteneur*

Lorsque vous utilisez des fonctionnalités d'isolation du processus, on parle de conteneur.
Dans un conteneur, on peut avoir plusieurs programmes avec des dépendances de manière à les maintenir isolés du système hôte.

Nous avons aussi vu que sur docker, un conteneur est une instance d'exécution d'une image. Dans ce cas, l'instance de l'image est le conteneur.

IX.II. *Récupérer l'image de debian*

Etape réalisée, dans le § VIII.I.

Télécharger l'image debian :

```
docker pull debian
```

```
christophe@christophe-VirtualBox:~$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
001c52e26ad5: Pull complete
Digest: sha256:82bab30ed448b8e2509aabe21f40f0607d905b7fd0dec72802627a20274eba55
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest
christophe@christophe-VirtualBox:~$
```

IX.II.1 Instanciation de l'image debian

Essayons de lancer

```
docker run debian
```

```
christophe@christophe-VirtualBox: ~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
christophe@christophe-VirtualBox:~$ docker run debian
```

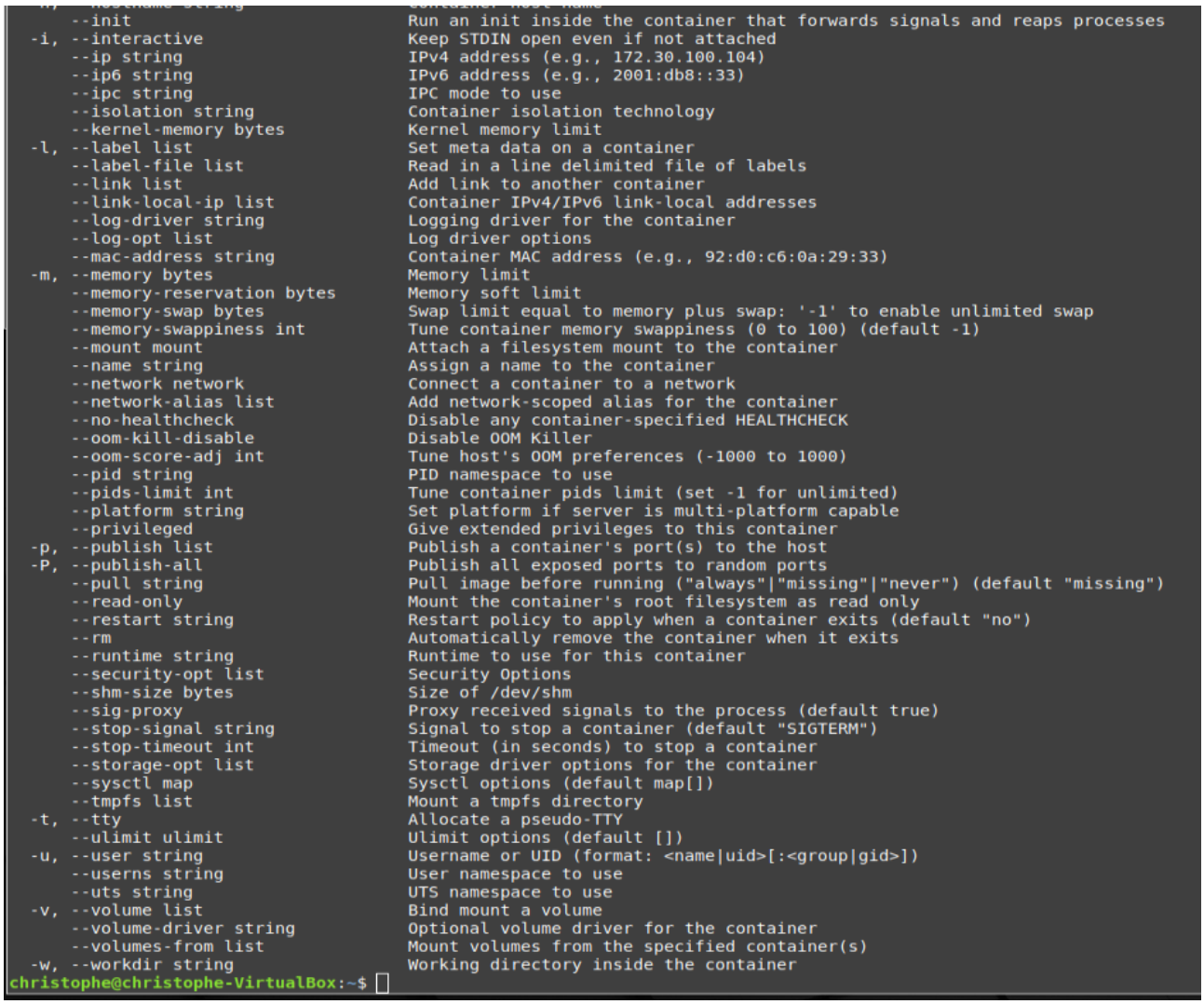
```
christophe@christophe-VirtualBox:~$ docker run debian
christophe@christophe-VirtualBox:~$
```

Le conteneur quitte directement, c'est normal. Il y a besoin d'exploiter des options :

Pour les voir toutes :

```
docker run --help
```

```
christophe@christophe-VirtualBox:~$ docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

Options:
      --add-host list                  Add a custom host-to-IP mapping (host:ip)
  -a, --attach list                    Attach to STDIN, STDOUT or STDERR
      --blkio-weight uint16            Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
      --blkio-weight-device list       Block IO weight (relative device weight) (default [])
      --cap-add list                   Add Linux capabilities
      --cap-drop list                  Drop Linux capabilities
      --cgroup-parent string           Optional parent cgroup for the container
      --cgroupns string                Cgroup namespace to use (host|private)
                                       'host':    Run the container in the Docker host's cgroup namespace
                                       'private': Run the container in its own private cgroup namespace
                                       '':        Use the cgroup namespace as configured by the
                                                  default-cgroupns-mode option on the daemon (default)
      --cidfile string                 Write the container ID to the file
      --cpu-period int                 Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int                  Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int              Limit CPU real-time period in microseconds
      --cpu-rt-runtime int             Limit CPU real-time runtime in microseconds
  -c, --cpu-shares int                 CPU shares (relative weight)
      --cpus decimal                   Number of CPUs
      --cpuset-cpus string             CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string             MEMs in which to allow execution (0-3, 0,1)
  -d, --detach                         Run container in background and print container ID
      --detach-keys string             Override the key sequence for detaching a container
      --device list                    Add a host device to the container
      --device-cgroup-rule list        Add a rule to the cgroup allowed devices list
      --device-read-bps list           Limit read rate (bytes per second) from a device (default [])
      --device-read-iops list          Limit read rate (IO per second) from a device (default [])
      --device-write-bps list          Limit write rate (bytes per second) to a device (default [])
      --device-write-iops list         Limit write rate (IO per second) to a device (default [])
      --disable-content-trust          Skip image verification (default true)
      --dns list                       Set custom DNS servers
      --dns-option list                Set DNS options
      --dns-search list                Set custom DNS search domains
      --domainname string              Container NIS domain name
      --entrypoint string              Overwrite the default ENTRYPOINT of the image
  -e, --env list                       Set environment variables
      --env-file list                  Read in a file of environment variables
      --expose list                    Expose a port or a range of ports
      --gpus gpu-request               GPU devices to add to the container ('all' to pass all GPUs)
      --group-add list                 Add additional groups to join
      --health-cmd string              Command to run to check health
      --health-interval duration       Time between running the check (ms|s|m|h) (default 0s)
      --health-retries int             Consecutive failures needed to report unhealthy
      --health-start-period duration   Start period for the container to initialize before starting health-retries countdown (ms|s|m|h) (default 0s)
      --health-timeout duration        Maximum time to allow one check to run (ms|s|m|h) (default 0s)
      --help                           Print usage
  -h, --hostname string                Container host name
```



Comme vous, pouvez le constater il existe beaucoup d'options, mais rassurez-vous, car vous n'avez pas besoin de tous les connaître, voici pour le moment les options qui nous intéressent pour ce chapitre :

- -t : Allouer un pseudo TTY (terminal virtuel)
- -i : Garder un STDIN ouvert (l'entrée standard plus précisément l'entrée clavier)
- -d : Exécuter le conteneur en arrière-plan et afficher l'ID du conteneur

Lançons maintenant :

```
docker run -ti debian
```

Lancer la mise à jour des packet de debian dans le conteneur :

```
apt-get update
```

```
root@7eeb1ce7a3ea:/# apt-get update
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8182 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [175 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [2592 B]
Fetched 8567 kB in 2s (4792 kB/s)
Reading package lists... Done
root@7eeb1ce7a3ea:/#
```

IX.II.2 Installation de logiciels dans le conteneur

Installons git dans le conteneur :

```
apt-get install -y git
```

```
root@7eeb1ce7a3ea:/# apt-get install -y git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates git-man less libbrotli1 libbsd0 libcbor0 libcurl3-gnutls libedit2 liberror-perl libexpat1 libfido2-1 libgdbm-compat4 libgdbm6 libldap-2.4-2
  libldap-common libmd0 libnghttp2-14 libperl5.32 libpsl5 librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libssh2-1 libx11-6 libx11-data libxau6
  libxcb1 libxdmcp6 libxext6 libxmuu1 netbase openssh-client openssl patch perl perl-modules-5.32 publicsuffix xauth
Suggested packages:
  gettext-base git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn gdbm-l10n sensible-utils
  libsasl2-modules-gssapi-mit | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap libsasl2-modules-otp libsasl2-modules-sql keychain libpam-ssh monkeysphere
  ssh-askpass ed diffutils-doc perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl make libtap-harness-archive-perl
The following NEW packages will be installed:
  ca-certificates git git-man less libbrotli1 libbsd0 libcbor0 libcurl3-gnutls libedit2 liberror-perl libexpat1 libfido2-1 libgdbm-compat4 libgdbm6
  libldap-2.4-2 libldap-common libmd0 libnghttp2-14 libperl5.32 libpsl5 librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libssh2-1 libx11-6 libx11-data
  libxau6 libxcb1 libxdmcp6 libxext6 libxmuu1 netbase openssh-client openssl patch perl perl-modules-5.32 publicsuffix xauth
0 upgraded, 40 newly installed, 0 to remove and 3 not upgraded.
Need to get 20.4 MB of archives.
After this operation, 101 MB of additional disk space will be used.
```

```
Processing triggers for libc-bin (2.31-13+deb11u3) ...
Processing triggers for ca-certificates (20210119) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
root@7eeb1ce7a3ea:/#
```

```
git --version
```

```
root@7eeb1ce7a3ea:/# git --version
git version 2.30.2
root@7eeb1ce7a3ea:/#
```

Attention, si vous sortez du conteneur et relancer l'image de debian, votre installation de git a été fait temporairement.

Entrer la commande :

```
exit
```

```
root@7eeb1ce7a3ea:/# exit
exit
christophe@christophe-VirtualBox:~$
```

```
christophe@christophe-VirtualBox:~$ docker run -ti debian
root@507a3d5f385f:/#
```

```
christophe@christophe-VirtualBox:~$ docker run -ti debian
root@507a3d5f385f:/# git
bash: git: command not found
root@507a3d5f385f:/#
```

IX.II.3 Comment persister les données des conteneurs

Il existe deux façons pour stocker les données d'un conteneur, soit on transforme notre conteneur en image (cependant ça reste une méthode non recommandée, mais je vous montrerai à la fin de ce chapitre comment l'utiliser), soit on utilise le système de volume, nous verrons cette notion après dédiée aux volumes.

IX.III. *Installation de Nginx de docker*

```
docker pull nginx
```

```
christophe@christophe-VirtualBox:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
1efc276f4ff9: Pull complete
baf2da91597d: Pull complete
05396a986fd3: Pull complete
6a17c8e7063d: Pull complete
27e0d286aeab: Pull complete
b1349eea8fc5: Pull complete
Digest: sha256:790711e34858c9b0741edffef6ed3d8199d8faa33f2870dea5db70f16384df79
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
christophe@christophe-VirtualBox:~$
```

Lancement du serveur web sur le port 8080

```
docker run --name monServeurWeb -p 8080:80 nginx
```

```
christophe@christophe-VirtualBox:~$ docker run --name monServeurWeb -p 8080:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/08/11 15:26:30 [notice] 1#1: using the "epoll" event method
2022/08/11 15:26:30 [notice] 1#1: nginx/1.23.1
2022/08/11 15:26:30 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/08/11 15:26:30 [notice] 1#1: OS: Linux 5.4.0-122-generic
2022/08/11 15:26:30 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/08/11 15:26:30 [notice] 1#1: start worker processes
2022/08/11 15:26:30 [notice] 1#1: start worker process 31
2022/08/11 15:26:30 [notice] 1#1: start worker process 32
2022/08/11 15:26:30 [notice] 1#1: start worker process 33
2022/08/11 15:26:30 [notice] 1#1: start worker process 34
```

Dans la commande, il apparait :

```
172.17.0.1 - - [11/Aug/2022:15:28:46 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:103.0) Gecko/20100101 Firefox/103.0" "-"
```

Ce qui serait intéressant c'est de laisser notre conteneur tourner en arrière-plan avec l'option -d

Fait CTRL+C pour demander au serveur nginx de se terminer.

```
^C2022/08/11 15:31:12 [notice] 1#1: signal 2 (SIGINT) received, exiting
2022/08/11 15:31:12 [notice] 31#31: exiting
2022/08/11 15:31:12 [notice] 34#34: exiting
2022/08/11 15:31:12 [notice] 32#32: exiting
2022/08/11 15:31:12 [notice] 32#32: exit
2022/08/11 15:31:12 [notice] 31#31: exit
2022/08/11 15:31:12 [notice] 34#34: exit
2022/08/11 15:31:12 [notice] 33#33: exiting
2022/08/11 15:31:12 [notice] 33#33: exit
2022/08/11 15:31:12 [notice] 1#1: signal 17 (SIGCHLD) received from 32
2022/08/11 15:31:12 [notice] 1#1: worker process 32 exited with code 0
2022/08/11 15:31:12 [notice] 1#1: signal 29 (SIGIO) received
2022/08/11 15:31:12 [notice] 1#1: signal 17 (SIGCHLD) received from 33
2022/08/11 15:31:12 [notice] 1#1: worker process 33 exited with code 0
2022/08/11 15:31:12 [notice] 1#1: signal 29 (SIGIO) received
2022/08/11 15:31:12 [notice] 1#1: signal 17 (SIGCHLD) received from 34
2022/08/11 15:31:12 [notice] 1#1: worker process 34 exited with code 0
2022/08/11 15:31:12 [notice] 1#1: signal 17 (SIGCHLD) received from 31
2022/08/11 15:31:12 [notice] 1#1: worker process 31 exited with code 0
2022/08/11 15:31:12 [notice] 1#1: exit
```

Si on souhaite relancer le conteneur avec le même nom, ce n'est pas possible.
On obtient l'erreur :

```
christophe@christophe-VirtualBox:~$ docker run --name monServeurWeb -d -p 8080:80 nginx
docker: Error response from daemon: Conflict. The container name "/monServeurWeb" is already in use by container "580e3efdb325a0965e5627cac3fa48fb3497724d59bd4ac0
5d233a9b48ef87db". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
christophe@christophe-VirtualBox:~$
```

Pour afficher la liste des conteneurs :

```
docker container ls
```

ou

```
docker ps
```

docker rm <ID_CONTENER>

```
christophe@christophe-VirtualBox:~$ docker rm 580e3efdb325a0965e5627cac3fa48fb3497724d59bd4ac05d233a9b48ef87db
580e3efdb325a0965e5627cac3fa48fb3497724d59bd4ac05d233a9b48ef87db
christophe@christophe-VirtualBox:~$
```

Relance du serveur avec l'option -d

```
christophe@christophe-VirtualBox:~$ docker run --name monServeurWeb -d -p 8080:80 nginx
a33925e349e0af03fec333d72ef49fb1f08b4e6a5d279e60b17aefce83f48e1c
christophe@christophe-VirtualBox:~$
```

Affichage de la liste des containeurs :

```
docker ps
```

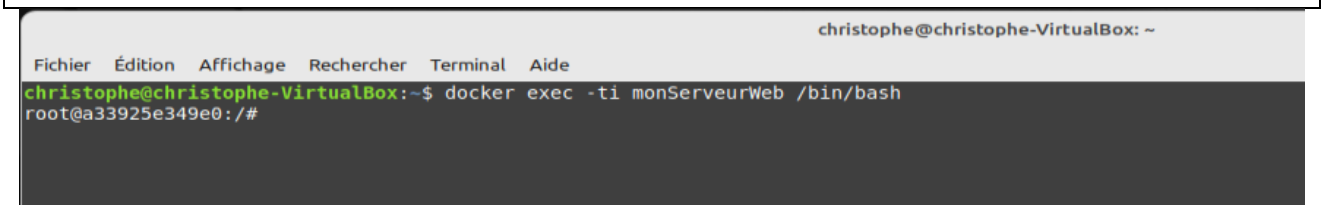
```
christophe@christophe-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                   NAMES
a33925e349e0   nginx     "/docker-entrypoint.…"   37 seconds ago   Up 34 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   monServeurWeb
christophe@christophe-VirtualBox:~$
```

IX.IV. *Exécution d'une commande dans un containeur*

Une fois qu'un conteneur est lancé en arrière-plan, il est possible de continuer à manipuler ce conteneur.

Il existe une commande « docker exec » qui permet de lancer n'importe quelle commande dans un conteneur déjà en cours d'exécution. Nous allons l'utilisée pour récupérer notre interpréteur de commande /bin/bash, ce qui aura pour but de se connecter directement à notre conteneur Nginx.

```
docker exec -ti monServeurWeb /bin/bash
```



Allons dans le bash qui est ouvert, le lieu du fichier index.hml qui sert à nginx de charger la page.

```
cat /etc/nginx/conf.d/default.conf
```

```
root@a33925e349e0:/# cat /etc/nginx/conf.d/default.conf
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}

root@a33925e349e0:/#
```

Affichage du contenu du fichier index.html

```
root@a33925e349e0:/# cat /usr/share/nginx/html/index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@a33925e349e0:/#
```

```
root@a33925e349e0:/# echo "<h1>Wellcome Docker</h1>" > /usr/share/nginx/html/index.html
```

IX.V. *Afficher les logs du containeur*

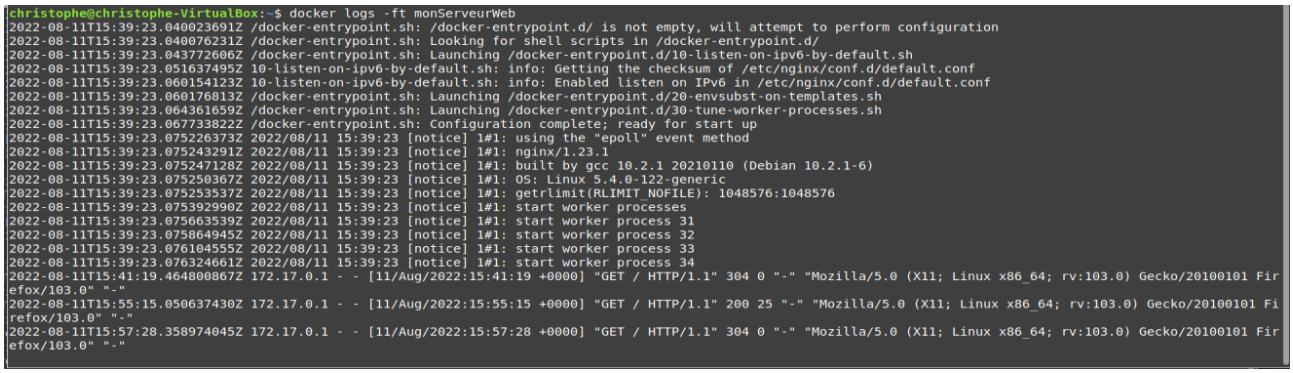
Dès fois, vous aurez besoin de déboguer votre conteneur en regardant les sorties/erreurs d'un conteneur.

Il existe pour cela la commande docker logs qui vient avec deux options très utiles :

- -f : suivre en permanence les logs du conteneur (correspond à tail -f)
- -t : afficher la date et l'heure de réception des logs d'un conteneur

```
docker logs -ft monServeurWeb
```

si vous visitez votre page, alors vous verrez des logs s'afficher successivement sur votre terminal (Ctrl + C pour quitter vos logs)



IX.VI. *Persister les données d'un conteneur (Déprécié)*

Comme promis voici, la commande qui permet de transformer un conteneur en image, afin de stocker nos données

Attention

Je me répète mais c'est important, cette commande n'est pas vraiment recommandée, pour stocker vos données. Il faut pour ça utiliser les volumes, que nous le verrons après.

Voici les étapes que nous allons suivre :

Exécuter notre conteneur basé sur l'image officielle debian
Installer l'outil git
Mettre du texte dans un nouveau fichier
Transformer notre conteneur en image
Relancer notre un nouveau conteneur basé sur cette nouvelle image

```
docker run -ti --name monDebian  debian

apt-get update -y && apt-get install -y git

echo "ceci est un fichier qui contient des donnees de test" > test.txt && cat test.txt
```

(Ctrl + P + Q)

```
christophe@christophe-VirtualBox:~$ docker run -ti --name monDebian debian
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
001c52e26ad5: Already exists
Digest: sha256:82bab30ed448b8e2509aabe21f40f0607d905b7fd0dec72802627a20274eba55
Status: Downloaded newer image for debian:latest
root@6fc1a9f319b3:/#
```

```
root@6fc1a9f319b3:/# echo "ceci est un fichier qui contient des donnees de tests" > test.txt && cat test.txt
ceci est un fichier qui contient des donnees de tests
root@6fc1a9f319b3:/#
```

```
root@6fc1a9f319b3:/# echo "ceci est un fichier qui contient des donnees de tests" > test.txt && cat test.txt
ceci est un fichier qui contient des donnees de tests
root@6fc1a9f319b3:/# christophe@christophe-VirtualBox:~$
```

Maintenant, c'est l'étape ou je vais créer mon image depuis mon nouveau conteneur. Voici son prototype :

```
docker commit <CONTAINER NAME or ID> <NEW IMAGENAME>
```

Ce qui nous donnera :

```
docker commit monDebian debiangit
```

Information

Vous pouvez voir, votre nouvelle image avec la commande docker images

```
root@6fc1a9f319b3:/# christophe@christophe-VirtualBox:~$ docker commit monDebian debiangit
sha256:19d97cde301a51bcebfac4179e1b50e5a854a7ac76050b541e08d1a3687e6bf2
christophe@christophe-VirtualBox:~$
```

Voilà, maintenant, on va lancer notre conteneur basé sur cette nouvelle image :

```
docker run -ti --name debiangit_container  debiangit
```

```
christophe@christophe-VirtualBox:~$ docker run -ti --name debiangit_container debiangit
root@fcf87875f786:/#
```

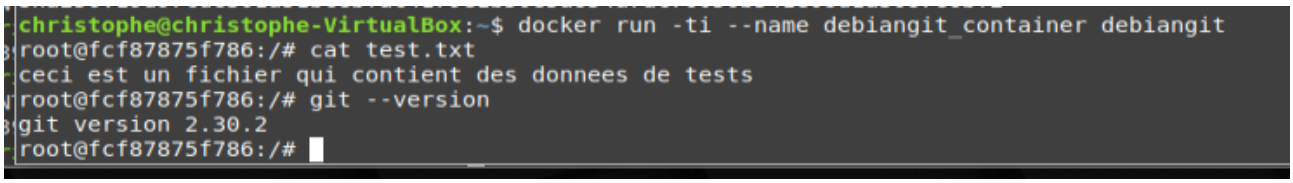
À présent, je vais vérifier si les données ont bien été stockées sur ce nouveau conteneur.

```
cat test.txt
```

Résultat :

```
ceci est un fichier qui contient des donnees de test

git --version
```

Notre outil git et notre fichier sont bien présents dans notre nouveau conteneur 😊.



X. Création d'une image Docker (Debian, Nginx, MariaDB, PHP, Drupal)

X.I. *Objectif*

L'objectif est de créer notre propre image Docker à l'aide du fichier Dockerfile.

Nous allons mettre en place donc : Debian, Nginx, MariaDB, PHP et Drupal en mode non initialisée.

Rappel :

- Debian : OS
- Nginx : serveur web
- MariaDB : moteur de base de données
- PHP : langage de programmation interpréteur
- Drupal : CMS d'un site internet

Rappel : ici c'est juste pour la démonstration. Il conviendrait d'utiliser les VOLUMES pour l'installation de Drupal car les données produites doivent pouvoir rester pérenne ainsi que la base de données.

X.II. *Les instructions Dockerfile*

Voici la liste des instructions Dockerfile les plus communément utilisées :

Mot clef	Description
FROM	Définit l'image de base qui sera utilisée par les instructions suivantes.
LABEL	Ajoute des métadonnées à l'image avec un système de clés-valeurs, permet par exemple d'indiquer à l'utilisateur l'auteur du Dockerfile.
ARG	Variables temporaires qu'on peut utiliser dans un Dockerfile.

ENV	Variables d'environnements utilisables dans votre Dockerfile et conteneur.
RUN	Exécute des commandes Linux ou Windows lors de la création de l'image. Chaque instruction RUN va créer une couche en cache qui sera réutilisée dans le cas de modification ultérieure du Dockerfile.
COPY	Permet de copier des fichiers depuis notre machine locale vers le conteneur Docker.
ADD	Même chose que COPY mais prend en charge des liens ou des archives (si le format est reconnu, alors il sera décompressé à la volée).
ENTRYPOINT	Comme son nom l'indique, c'est le point d'entrée de votre conteneur, en d'autres termes, c'est la commande qui sera toujours exécutée au démarrage du conteneur. Il prend la forme de tableau JSON (ex : CMD ["cmd1","cmd1"]) ou de texte.
CMD	Spécifie les arguments qui seront envoyés au ENTRYPOINT, (on peut aussi l'utiliser pour lancer des commandes par défaut lors du démarrage d'un conteneur). Si il est utilisé pour fournir des arguments par défaut pour l'instruction ENTRYPOINT, alors les instructions CMD et ENTRYPOINT doivent être spécifiées au format de tableau JSON.
WORKDIR	Définit le répertoire de travail qui sera utilisé pour le lancement des commandes CMD et/ou ENTRYPOINT et ça sera aussi le dossier courant lors du démarrage du conteneur.
EXPOSE	Expose un port.
VOLUMES	Crée un point de montage qui permettra de persister les données.
USER	Désigne quel est l'utilisateur qui lancera les prochaines instructions RUN, CMD ou ENTRYPOINT (par défaut c'est l'utilisateur root).

X.III. *Création des sources et du Dockerfile*

Récupérer et extraire le zip image-docker-drupal.zip dans un répertoire sur votre OS.

Voici le contenu du fichier Dockerfile servant à initialiser l'image qui contient tout l'environnement supportant Drupal 9.4.5.

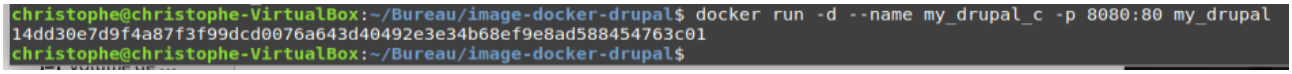
X.IV. *Création de l'image*

```
docker build -t my_drupal .
```

X.V. *Lancement de l'image*

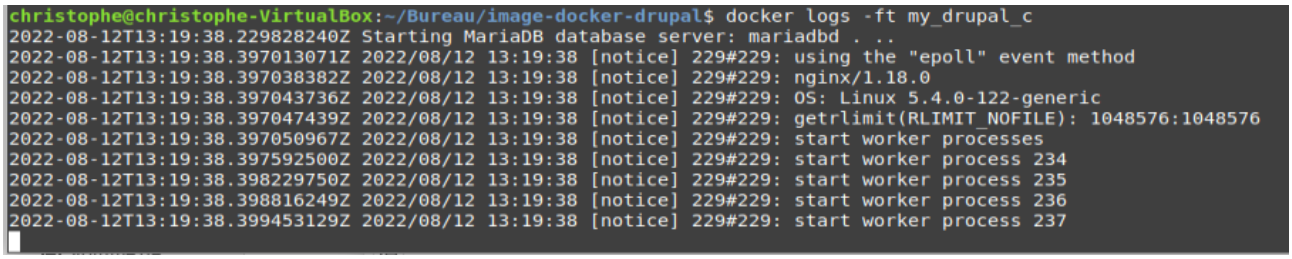
Le lancement de ce docker :

```
docker run -d --name my_drupal_c -p 8080:80  my_drupal
```



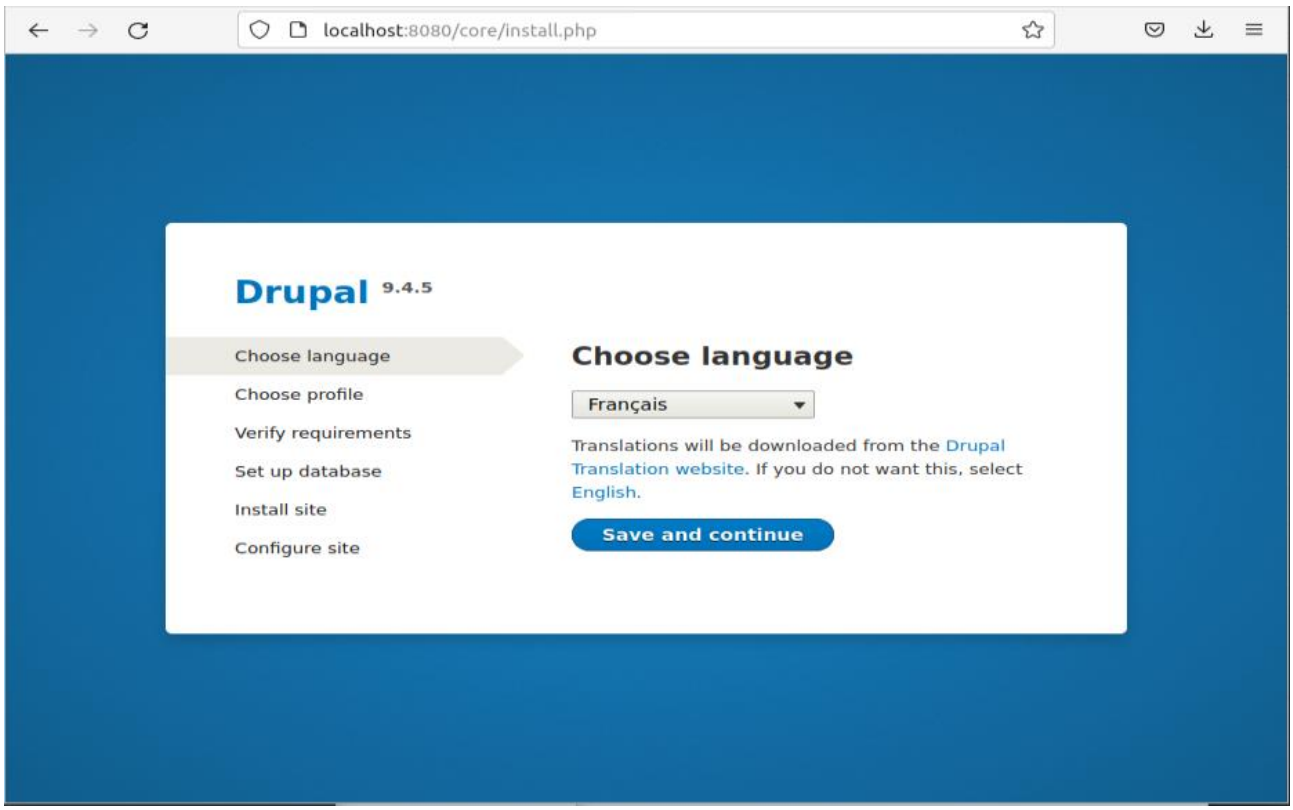
X.VI. *Vérification des logs*

```
docker kogs -ft my_drupal_c
```

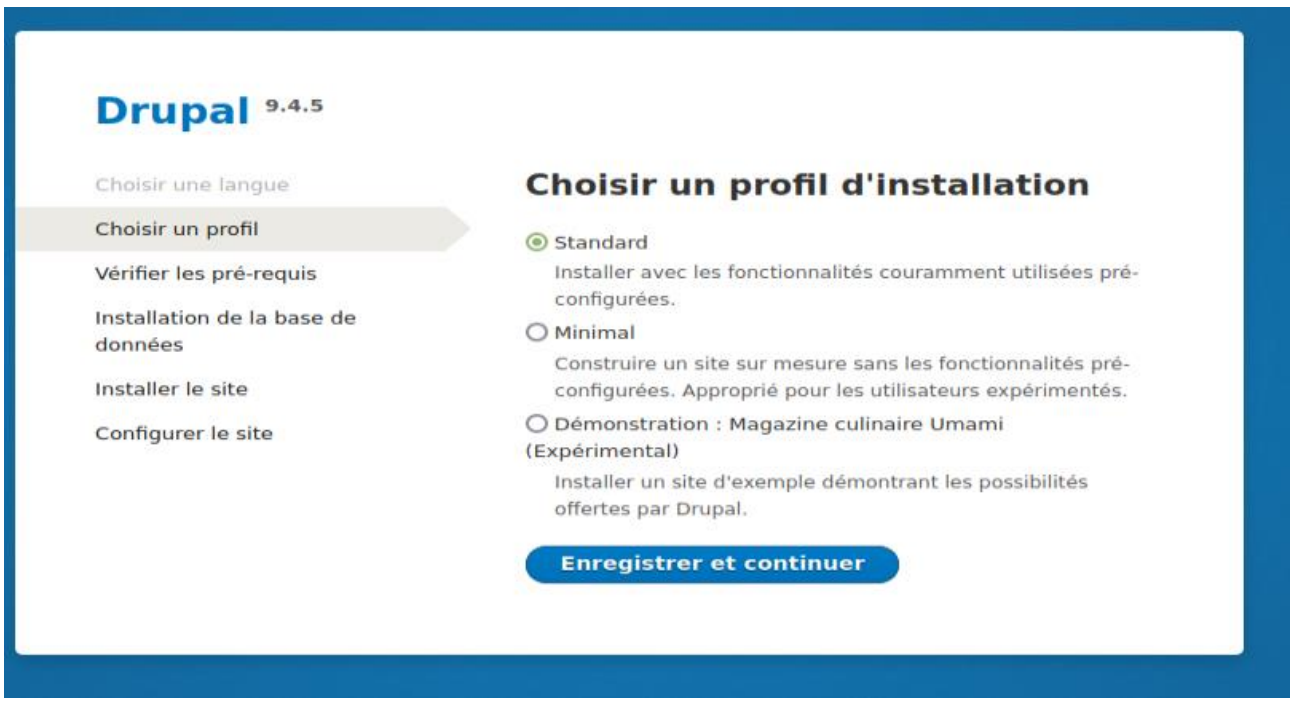


X.VII. *Affichage du site*

Aller dans le navigateur : http://localhost:8080



Cliquer sur Save and Continue.



Nous n'allons pas ici s'attacher à faire une installation From SCRASCH, nous allons donc utiliser le mode de Démonstration.

Dans un vrai site, l'installation de Drupal devrait être réalisée comme des données et non comme une installation de base. En effet, comment gérer sélectivement les mises à jour du site.

De plus, il conviendrait aussi de séparer les fichiers de données public comme les images, son vidéos etc….

Donc cela est fait pour l'exercice et non pour un usage réel de production.

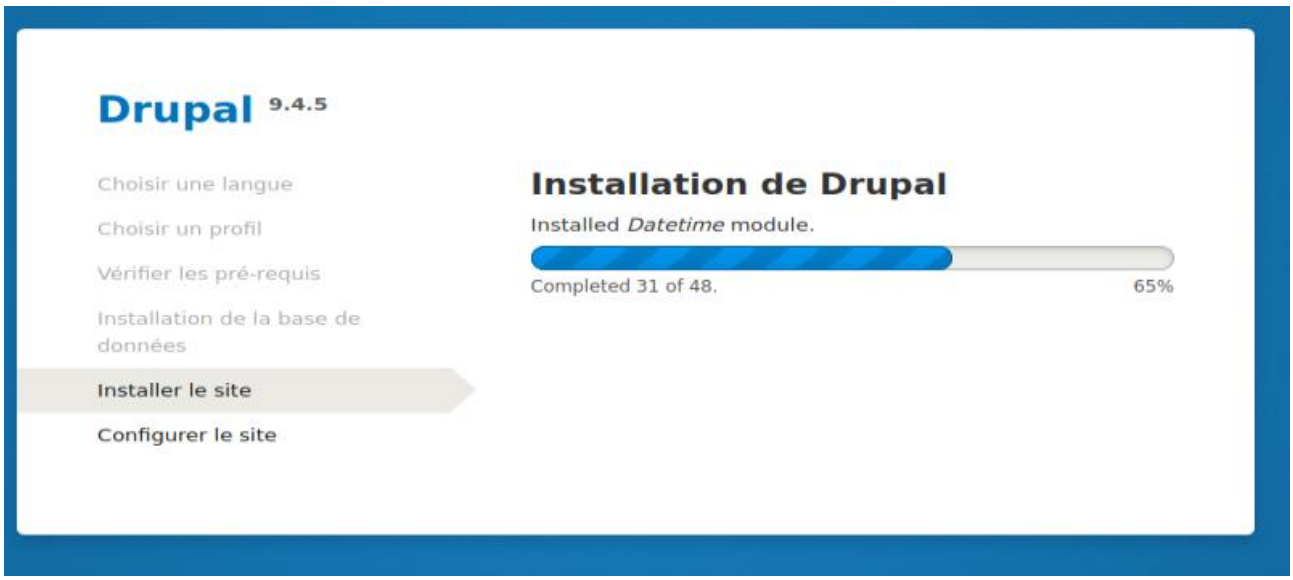
Les informations de connexion initialisées dans la requête SQL :

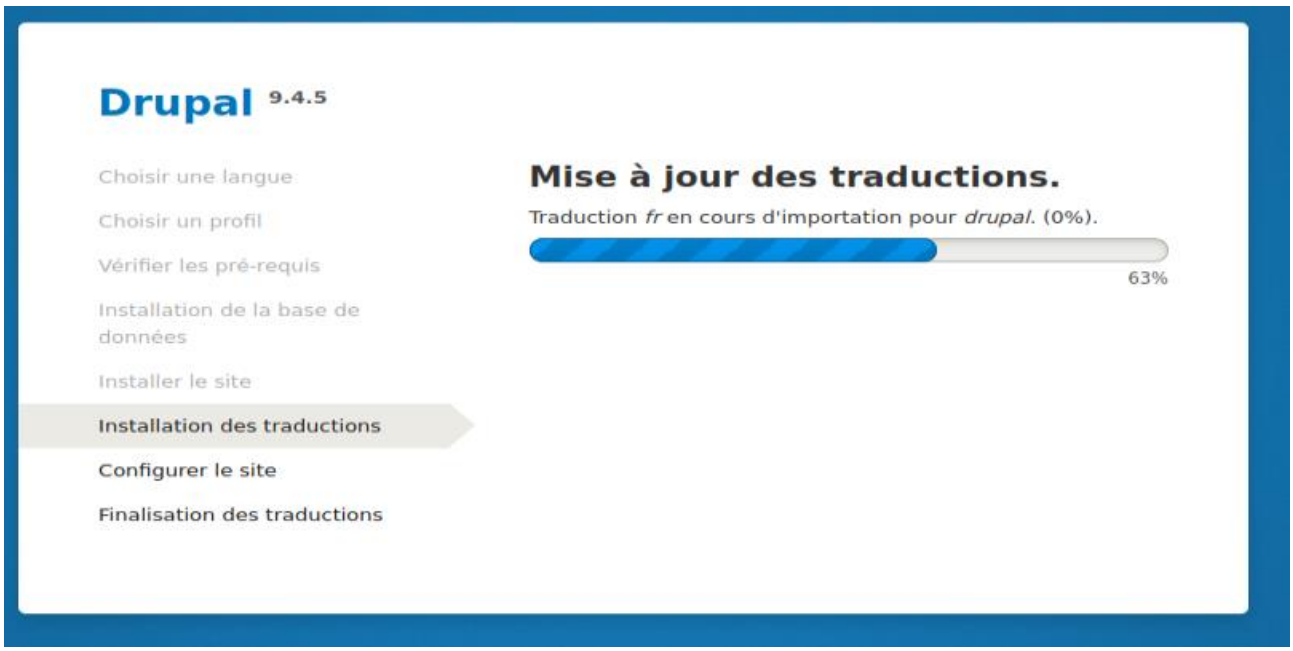
Base de données : drupal_demo

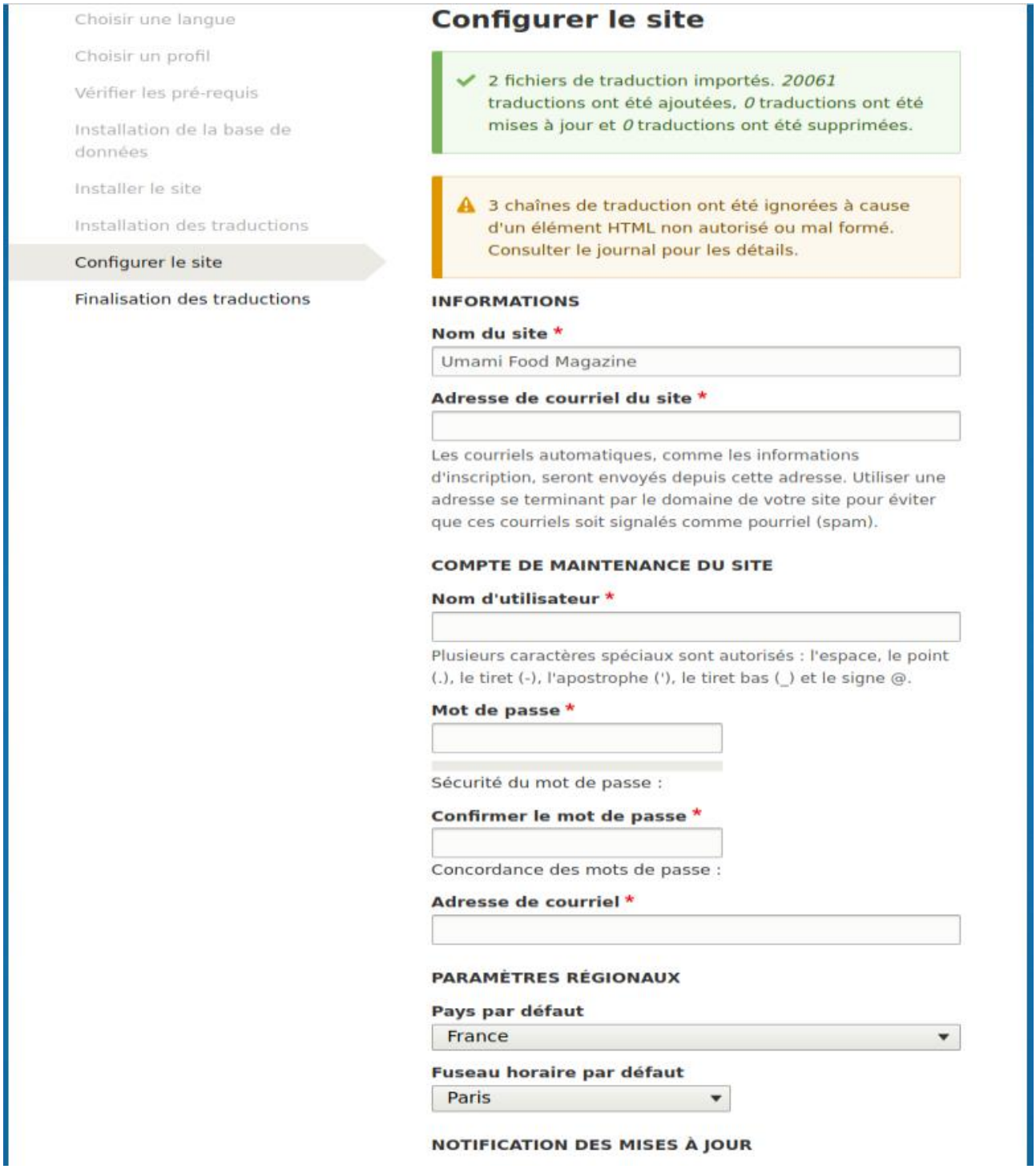
Utilisateur : drupal

Password : drupal12345678!


Drupal 9.4.5
Choisir une langue
Choisir un profil
Vérifier les pré-requis
Installation de la base de données
Installer le site
Configurer le site
Configuration de la base de données
Type de base de données *
MySQL, MariaDB, Percona Server ou équivalent
Nom de la base de données *
drupal_demo
Nom d'utilisateur de la base de données *
drupal
Mot de passe de la base de données
OPTIONS AVANCÉES
Hôte *
localhost
Numéro de port
3306
Préfixe du nom de la table
En cas de partage de cette base de données par plusieurs applications, un préfixe unique - tel que demo_umami_ - évitera les conflits.
Enregistrer et continuer


Drupal 9.4.5
Choisir une langue
Choisir un profil
Vérifier les pré-requis
Installation de la base de données
Installer le site
Configurer le site
Installation de Drupal
Installed Datetime module.
Completed 31 of 48.
65%


Drupal 9.4.5
Choisir une langue
Choisir un profil
Vérifier les pré-requis
Installation de la base de données
Installer le site
Installation des traductions
Configurer le site
Finalisation des traductions
Mise à jour des traductions.
Traduction fr en cours d'importation pour drupal. (0%).
63%

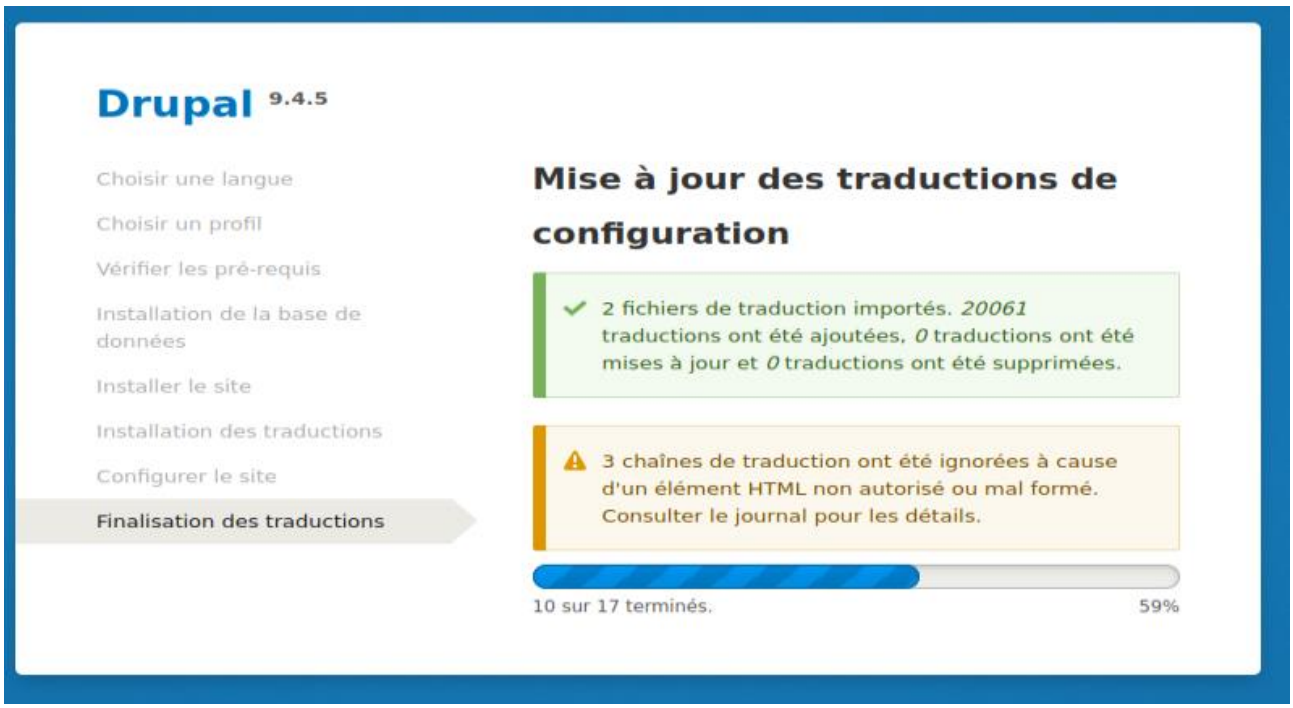


Configurer le site comme il vous convient.

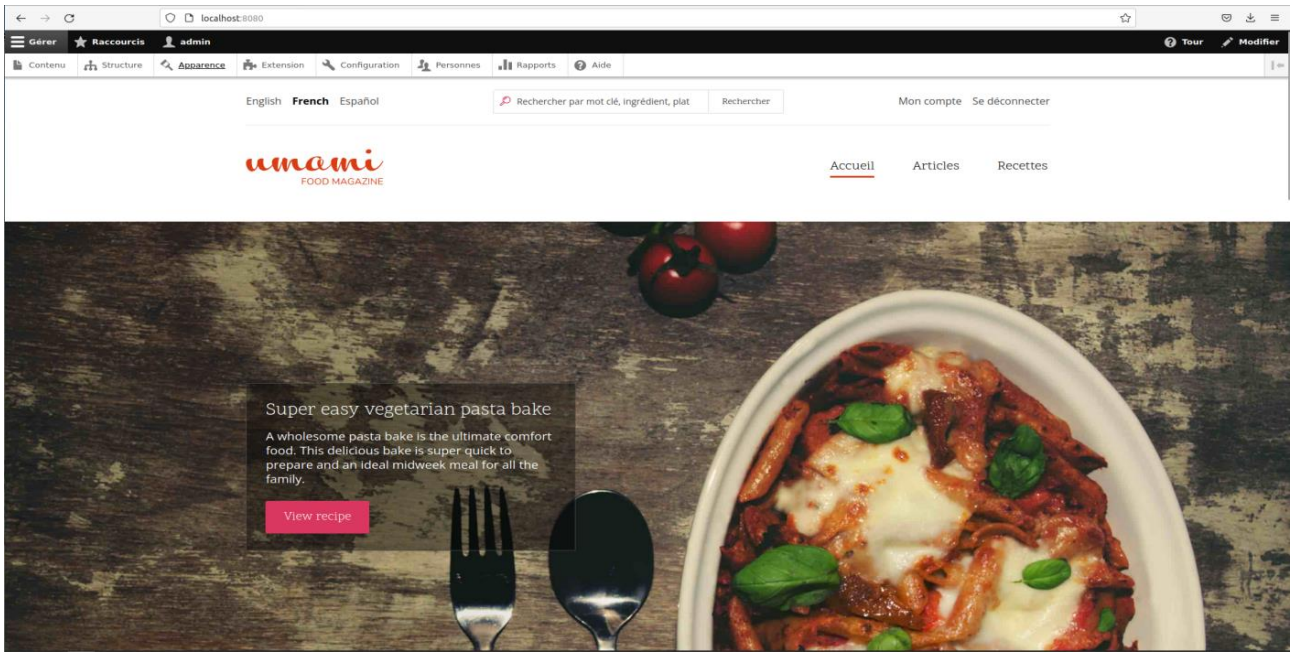
Email : contact@pequignat.eu

Nom d'utilisateur : admin

Mot de passe : XuKBYi6CMW9eJRm



Le site s'affiche :



Nous allons pour éviter de tout recommencer faire une sauvegarde sans passer par les volumes.

```
docker commit my_drupal_c my_drupal_installed
```

pour voir l'image créée :

```
docker image ls
```

```
christophe@christophe-VirtualBox:~$ docker image ls
REPOSITORY            TAG       IMAGE ID       CREATED             SIZE
my_drupal_installed   latest    d41ee38e99bc   6 minutes ago       1.24GB
my_drupal             latest    58aed3c6f10d   42 minutes ago      877MB
<none>                <none>    1c351bc52a64   About an hour ago   124MB
<none>                <none>    4c6bdb38a5c5   4 hours ago         624MB
<none>                <none>    29fe5df3aabd   5 hours ago         621MB
<none>                <none>    e14524f24cf1   6 hours ago         621MB
<none>                <none>    c4f171024d42   6 hours ago         571MB
<none>                <none>    c414ac13fed7   6 hours ago         571MB
<none>                <none>    bee262923c37   6 hours ago         571MB
debiangit             latest    19d97cde301a   22 hours ago        243MB
nginx                 latest    b692a91e4e15   10 days ago         142MB
debian                latest    07d9246c53a6   10 days ago         124MB
```

X.VIII. *Relance de l'image sauvegardée avec les données*

Après un reebot de linux mint par exemple, l'image n'est pas automatiquement relancée.

Il convient donc d'avoir sauvegardé les données. Ouf on avait utilisé la méthode dépréciée :

my_drupal_installed

Pour relancer l'image :

Il faut d'abord supprimer le conteneur my_drupal_c

```
docker rm my_drupal_c
```

```
docker run -d --name my_drupal_c -p 8080:80  my_drupal_installed
```

```
christophe@christophe-VirtualBox:~$ docker rm my_drupal_c
my_drupal_c
christophe@christophe-VirtualBox:~$ docker run -d --name my_drupal_c -p 8080:80 my_drupal_installed
bf58d05576f0783e4e2be36f15d5e798d51649e977e3f9fdae3d90852fcfc50a
christophe@christophe-VirtualBox:~$
```

Cela ne fonctionne pas, car au lancement tout le contexte est reconstruit avec l'exécution des créations des données. Donc cela plante.

```
christophe@christophe-VirtualBox:~$ docker logs -ft my_drupal_c
2022-08-14T09:25:46.302424279Z mkdir: cannot create directory '/run/php': File exists
2022-08-14T09:26:13.904247607Z Starting MariaDB database server: mariadbd . . . . . .
2022-08-14T09:26:14.642364131Z ERROR 1007 (HY000) at line 1: Can't create database 'drupal_demo'; database exists
christophe@christophe-VirtualBox:~$
```

Il aurait fallu protéger la création des données si elles n'existaient pas.

Nous allons donc passer par la suite sur une création plus robuste permettant de séparer les applicatifs des données avec l'usage des volumes.

XI. Exploitation des volumes

Les volumes permettent de sauvegarder les données d'un conteneur.
L'usage comme on l'a vu de sauvegarder l'image à un instant n'est pas recommandé et nécessite certaines précautions.

En effet, les données d'un conteneur sont éphémères.

XI.I. *Explication du fonctionnement du système de fichier dans docker*

Une image Docker, se compose d'un ensemble de layers (calques) en lecture seule. Lorsque vous lancez un conteneur à partir d'une image, Docker ajoute au sommet de cette pile de layeur un nouveau layeur en lecture-écriture. Docker appelle cette combinaison des couches un « Union File Ssytem ».

Voyons comment le moteur gère les modifications des fichiers au sein du conteneur :

- Lors d'une modification de fichier, Docker crée une copie depuis les couches en lecture seule vers le layer en lecture-écriture.
- Lors d'une création de fichier, Docker crée le fichier que sur le layer en lecture-écriture, et ne touche pas au layer en lecture seule.
- Lors d'une suppression de fichier, Docker supprime le fichier que sur le layer en lecture-écriture, et s'il est déjà existant dans le layer en lecture seule alors il le garde.

Les données dans le layer de base sont en lecture seule, elles sont donc protégées et intactes par toutes modifications, seul le layer en lecture-écriture est impacté lors de modifications de données.

Lorsqu'un conteneur est supprimé, le layer en lecture-écriture est supprimé avec. Cela signifie que toutes les modifications apportées après le lancement du conteneur auront disparus avec.

XI.II. *La création des volumes*

XI.II.1 Principes

Afin de pouvoir sauvegarder (persister) les données et également partager des données entre conteneurs, Docker a développé le concept de volumes. Les volumes sont des répertoires (ou des fichiers) qui ne font pas partie du système de fichiers Union mais qui existent sur le système de fichiers hôte.

En outre, les volumes constituent souvent le meilleur choix pour persistance des données pour le layer en lecture-écriture, car un volume n'augmente pas la taille des conteneurs qui l'utilisent et son contenu existe en dehors du cycle de vie d'un conteneur donné.

XI.II.2 Créer et gérer des volumes

Contrairement à un montage lié, vous pouvez créer et gérer des volumes en dehors de la portée de tout conteneur.

Pour créer un volume, nous utiliserons la commande suivante :

```
focker volume create <VOLUMENAME>
```

Soit :

```
docker volume create volume-test
```

Pour lister les volumes :

```
docker volume ls
```

```
christophe@christophe-VirtualBox:~$ docker volume ls
DRIVER    VOLUME NAME
local     volume-test
christophe@christophe-VirtualBox:~$
```

Pour récolter des informations sur un volume, il faut utiliser la commande suivante :

```
docker volume inspect volume-test
```

```
christophe@christophe-VirtualBox:~$ docker volume inspect volume-test
[
    {
        "CreatedAt": "2022-08-14T11:45:23+02:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/volume-test/_data",
        "Name": "volume-test",
        "Options": {},
        "Scope": "local"
    }
]
christophe@christophe-VirtualBox:~$
```

Dans ce résultat, on peut remarquer que toutes les nouvelles données de notre conteneur seront stockées sur le point de montage /var/lib/docker/volumes/volume-test/_data.

Pour supprimer un volume :

```
docker volume rm volume-test
```

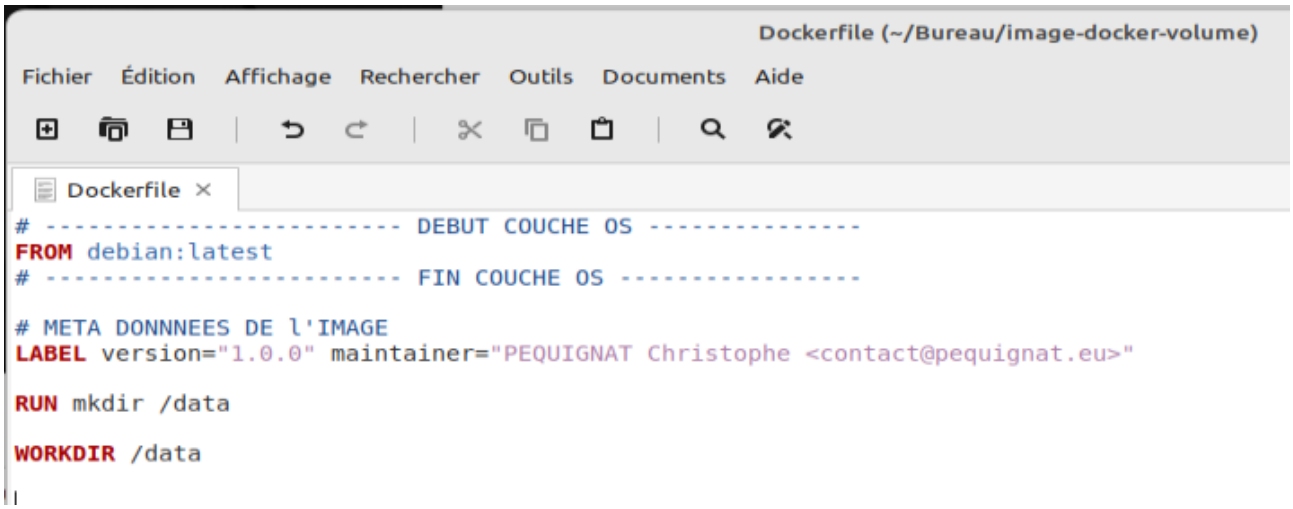
```
christophe@christophe-VirtualBox:~$ docker volume rm volume-test
volume-test
christophe@christophe-VirtualBox:~$
```

XI.III. *Démarrer un conteneur avec un volume*

Si vous démarrez un conteneur avec un volume qui n'existe pas encore, Docker le créera pour vous.

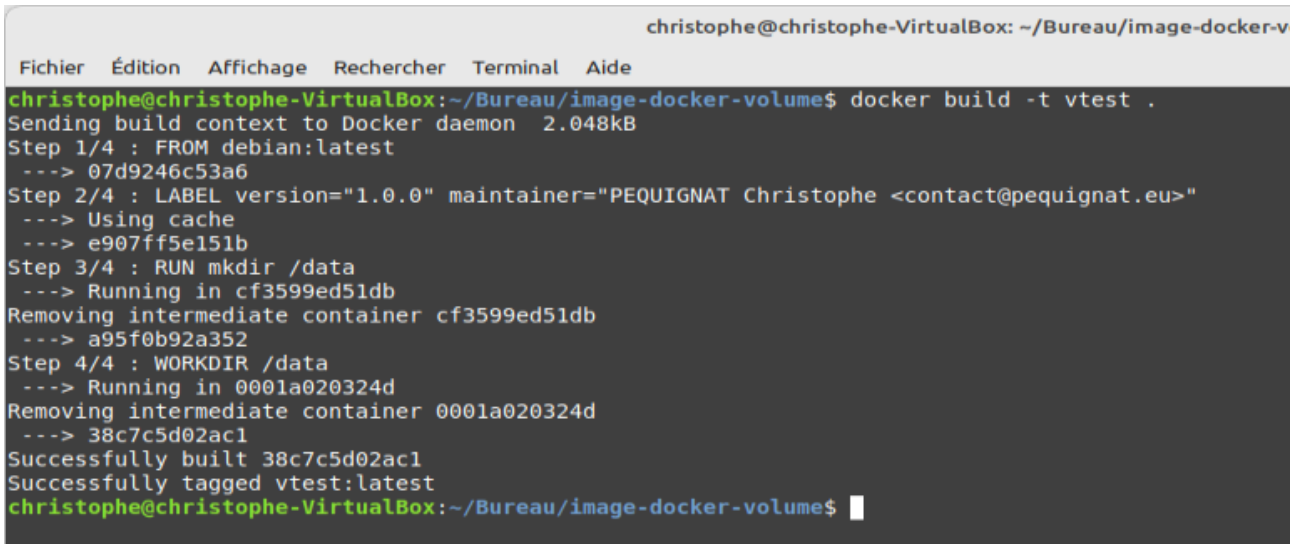
Pour démarrer un conteneur avec un volume, il faut utiliser l'option -v de la commande docker run.

Pour ce chapitre, nous allons créer une petite image pour tester cette option, commencez par créer un Dockerfile avec le contenu suivant :



Ensuite buildez notre image.

```
docker build -t vtest .
```



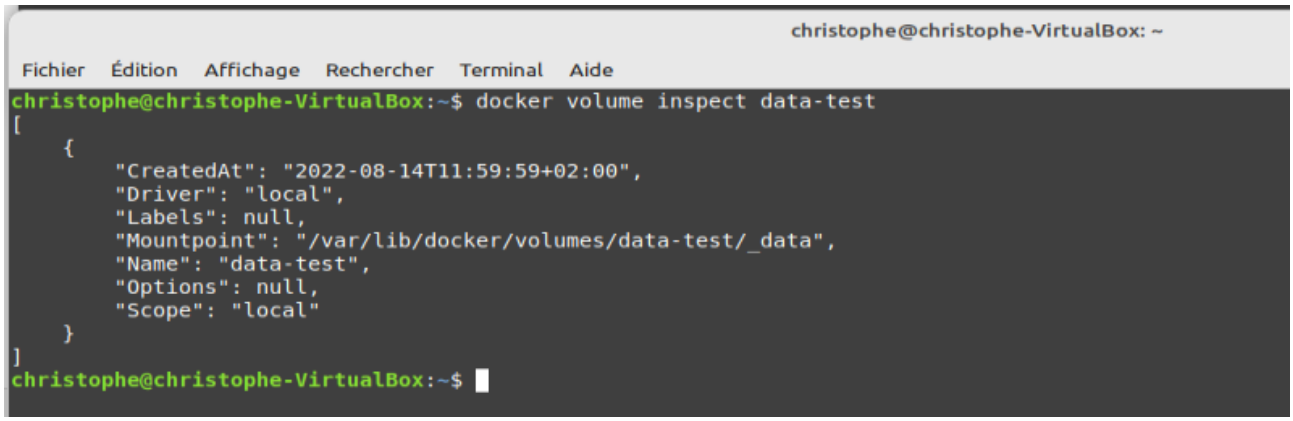
La commande suivante va créer et monter le volume data-test dans le dossier /data du conteneur.

```
docker run -ti --name vtest_c -v data-test:/data vtest
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-volume$ docker run -ti --name vtest_c -v data-test:/data vtest
root@81193c9c657a:/data#
```

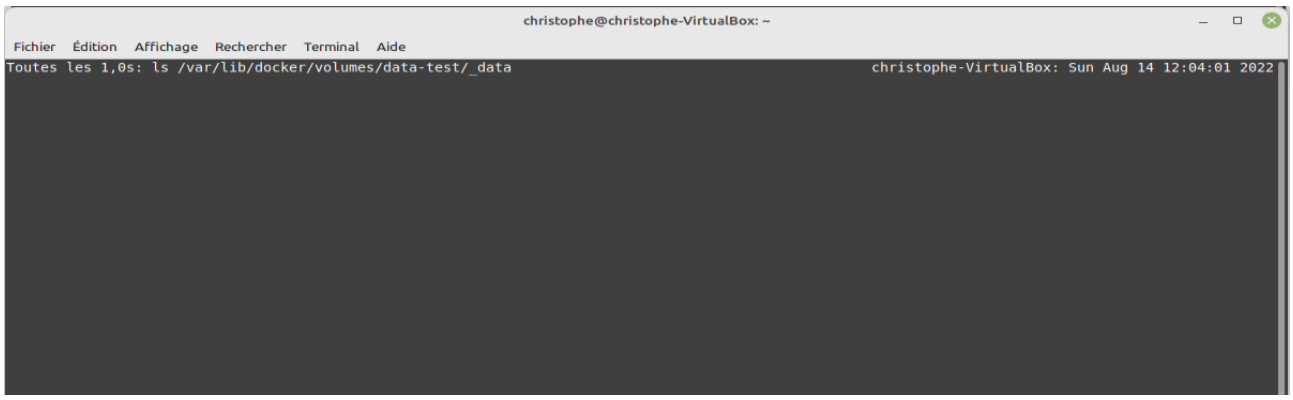
Ouvrez un autre terminal et dans celui-ci inspectez le nouveau volume :

```
docker volume inspect data-test
```



Nous allons essayer de voir en temps réel le contenu de ce volume, pour cela utilisez la commande suivante sur votre nouveau terminal :

```
sudo watch -n 1 ls /var/lib/docker/volumes/data-test/_data
```



Pour le moment le dossier est vide, maintenant retournez vers le terminal avec le shell du conteneur et créez dans le dossier /data un fichier avec le texte suivant :

```
echo "ceci est un test" > test.txt
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-volume$ docker run -ti --name vtest_c -v data-test:/data vtest
root@81193c9c657a:/data# echo "ceci est un test" > test.txt
root@81193c9c657a:/data#
```

Si vous retournez sur le nouveau terminal, vous verrez dessus votre nouveau fichier :

Maintenant, je vais quitter mon conteneur avec la commande exit et le supprimer :

```
docker rm -f vtest_c
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-volume$ docker rm -f vtest_c
vtest_c
christophe@christophe-VirtualBox:~/Bureau/image-docker-volume$
```

Je vais recréer un nouveau conteneur, pour vérifier que les données ont bien été sauvegardées :

```
docker run -ti --name vtest_c -v data-test:/data vtest
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-volume$ docker run -ti --name vtest_c -v data-test:/data vtest
root@75937d55877d:/data#
```

Dans ce même nouveau conteneur, je vérifie le contenu du fichier crée précédemment :

```
cat test.txt
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-volume$ docker run -ti --name vtest_c -v data-test:/data vtest
root@75937d55877d:/data# cat test.txt
ceci est un test
root@75937d55877d:/data#
```

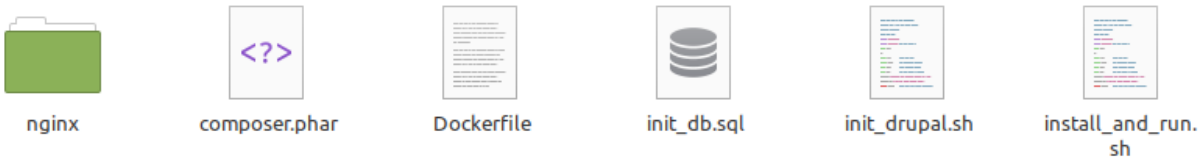
Nos données sont sauvegardées !

XII. Création et lancement avec volumes d'une image Docker Drupal

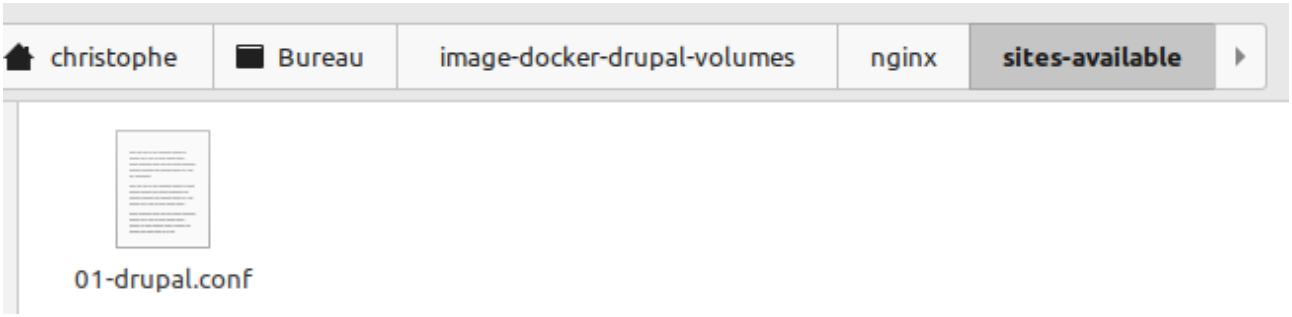
XII.I. *Récupération des sources toutes prêtes*

Prendre le zip « image-docker-drupal-volumes.zip » et désarchiver le quelque part.

Vous trouverez dans ce répertoire extrait :



- Le répertoire nginx contenant la configuration pour la déclaration du site drupal avec la prise en compte de PHP7.4-fpm



- composer.phar : le fichier composer.phar en version 1.10.X récupérer sur le site https://getcomposer.org/download/

- Dockerfile : le fichier de construction de l'image docker que l'on nommera « my_drupal_volumes »

- Init_db.sql : le fichier d'initialisation de la base de données et de son utilisateur pour le site drupal conditionné pour qu'il initialise si ce n'est pas déjà le cas.

- Init_drupal.sh : script shell qui exploite php et composer.phar afin d'initialiser la création du projet drupal

- install_and_run.sh : c'est le lanceur principal qui initialise si besoin l'image docker, construit les volumes si besoin et lance le conteneur avec les volumes déclarés.

XII.II. *Exécution initiale*

Mettre les droits d'exécution avec l'utilisateur courant sur le fichier install_and_run.sh

```
chmod u+x install_and_run.sh
```

Première exécution :

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-drupal-volumes$ ./install_and_run.sh
Creation de l image my_drupal_volumes
Sending build context to Docker daemon  2.005MB
Step 1/22 : FROM debian:latest
 ---> 07d9246c53a6
Step 2/22 : LABEL version="1.0.0" maintainer="PEQUIGNAT Christophe <contact@pequignat.eu>"
 ---> Using cache
 ---> e907ff5e151b
Step 3/22 : ARG APT_FLAGS="-q -y"
 ---> Using cache
 ---> c9e70853c569
Step 4/22 : ARG DRUPALROOT="/var/www/drupal"
 ---> Running in 05944a85fdcb
Removing intermediate container 05944a85fdcb
 ---> 6957054a1b16
Step 5/22 : ARG INITDATA="/init_data"
 ---> Running in 2278c6a82bfe
Removing intermediate container 2278c6a82bfe
 ---> 10ec18a215fc
Step 6/22 : RUN mkdir ${INITDATA}
 ---> Running in 8446f3612c0a
Removing intermediate container 8446f3612c0a
 ---> 21981cbc62ce
Step 7/22 : WORKDIR ${INITDATA}
 ---> Running in 002f4d85e167
Removing intermediate container 002f4d85e167
 ---> 8e4b68b54709
Step 8/22 : COPY composer.phar ${INITDATA}/composer.phar
 ---> 5fbca17704ba
Step 9/22 : COPY init_db.sql ${INITDATA}/init_db.sql
 ---> e55dbe1c04e7
Step 10/22 : COPY init_drupal.sh ${INITDATA}/init_drupal.sh
 ---> b53e11c33111
Step 11/22 : RUN apt-get update -y && apt-get install ${APT_FLAGS} mariadb-server mariadb-client
 ---> Running in f818ce7026ea
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
```

[…]

```
Step 12/22 : RUN apt-get install ${APT_FLAGS} nginx
 ---> Running in a67aaf2a246f
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core geoip-database libbrotli1 libdeflate0
  libexpat1 libfontconfig1 libfreetype6 libgd3 libgeoip1 libicu67 libjbig0
  libjpeg62-turbo libnginx-mod-http-geoip libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream
  libnginx-mod-stream-geoip libpng16-16 libtiff5 libwebp6 libx11-6 libx11-data
  libxau6 libxcb1 libxdmcp6 libxml2 libxpm4 libxslt1.1 nginx-common nginx-core
  sensible-utils ucf
Suggested packages:
  libgd-tools geoip-bin fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core geoip-database libbrotli1 libdeflate0
  libexpat1 libfontconfig1 libfreetype6 libgd3 libgeoip1 libicu67 libjbig0
  libjpeg62-turbo libnginx-mod-http-geoip libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream
  libnginx-mod-stream-geoip libpng16-16 libtiff5 libwebp6 libx11-6 libx11-data
  libxau6 libxcb1 libxdmcp6 libxml2 libxpm4 libxslt1.1 nginx nginx-common
  nginx-core sensible-utils ucf
0 upgraded, 35 newly installed, 0 to remove and 3 not upgraded.
Need to get 19.2 MB of archives.
After this operation, 62.9 MB of additional disk space will be used.
```

[…]

```
Step 13/22 : RUN apt-get install ${APT_FLAGS} php7.4-fpm php7.4-xml php7.4-mysqli php7.4-opcache php7.4-gd php7.4-curl php7.4-json php7.4-mbstring php7.4-zip
 ---> Running in 2ee3db480314
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  bzip2 ca-certificates file libapparmor1 libargon2-1 libcurl4 libldap-2.4-2
  libldap-common libmagic-mgc libmagic1 libnghttp2-14 libonig5 libpsl5
  librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libsodium23
  libssh2-1 libzip4 mailcap media-types mime-support openssl php-common
  php7.4-cli php7.4-common php7.4-readline publicsuffix xz-utils
Suggested packages:
  bzip2-doc libsasl2-modules-gssapi-mit | libsasl2-modules-gssapi-heimdal
  libsasl2-modules-ldap libsasl2-modules-otp libsasl2-modules-sql php-pear
The following NEW packages will be installed:
  bzip2 ca-certificates file libapparmor1 libargon2-1 libcurl4 libldap-2.4-2
  libldap-common libmagic-mgc libmagic1 libnghttp2-14 libonig5 libpsl5
  librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libsodium23
  libssh2-1 libzip4 mailcap media-types mime-support openssl php-common
  php7.4-cli php7.4-common php7.4-curl php7.4-fpm php7.4-gd php7.4-json
  php7.4-mbstring php7.4-mysql php7.4-opcache php7.4-readline php7.4-xml
  php7.4-zip publicsuffix xz-utils
0 upgraded, 39 newly installed, 0 to remove and 3 not upgraded.
Need to get 8597 kB of archives.
After this operation, 34.9 MB of additional disk space will be used.
```

[…]

```
Step 14/22 : RUN apt-get install ${APT_FLAGS} git
 ---> Running in de461c211129
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  git-man less libcbor0 libcurl3-gnutls liberror-perl libfido2-1 libxext6
  libxmuu1 openssh-client patch xauth
Suggested packages:
  gettext-base git-daemon-run | git-daemon-sysvinit git-doc git-el git-email
  git-gui gitk gitweb git-cvs git-mediawiki git-svn keychain libpam-ssh
  monkeysphere ssh-askpass ed diffutils-doc
The following NEW packages will be installed:
  git git-man less libcbor0 libcurl3-gnutls liberror-perl libfido2-1 libxext6
  libxmuu1 openssh-client patch xauth
0 upgraded, 12 newly installed, 0 to remove and 3 not upgraded.
Need to get 9111 kB of archives.
After this operation, 44.2 MB of additional disk space will be used.
```

[…]

```
Step 15/22 : COPY nginx/sites-available/01-drupal.conf /etc/nginx/sites-available/
 ---> 9cce1127480b
Step 16/22 : RUN ls -la /etc/nginx/sites-available/
 ---> Running in 8ea1c866f349
total 16
drwxr-xr-x 1 root root 4096 Aug 14 12:07 .
drwxr-xr-x 1 root root 4096 Aug 14 12:06 ..
-rw-rw-r-- 1 root root 1878 Aug 12 10:59 01-drupal.conf
-rw-r--r-- 1 root root 2412 May 29  2021 default
Removing intermediate container 8ea1c866f349
 ---> 38e649a5e144
Step 17/22 : RUN rm /etc/nginx/sites-enabled/*
 ---> Running in 1bf56aaed091
Removing intermediate container 1bf56aaed091
 ---> 1599bc0ba199
Step 18/22 : RUN ln -s /etc/nginx/sites-available/01-drupal.conf /etc/nginx/sites-enabled/
 ---> Running in 38267e3470a8
Removing intermediate container 38267e3470a8
 ---> 0d117b961699
Step 19/22 : RUN ls -la /etc/nginx/sites-enabled/
 ---> Running in 07c934270c53
total 8
drwxr-xr-x 1 root root 4096 Aug 14 12:07 .
drwxr-xr-x 1 root root 4096 Aug 14 12:06 ..
lrwxrwxrwx 1 root root   41 Aug 14 12:07 01-drupal.conf -> /etc/nginx/sites-available/01-drupal.conf
Removing intermediate container 07c934270c53
 ---> eca3fa49fb94
Step 20/22 : RUN bash ${INITDATA}/init_drupal.sh
 ---> Running in d9bed779a272
Create dir /var/www/drupal
Creating a "drupal/recommended-project" project at "/var/www/drupal"
Warning from https://repo.packagist.org: Support for Composer 1 is deprecated and some packages will not be available. You should upgrade to Composer 2. See https://blog.packagist.com/deprecating-composer-1-support/
Info from https://repo.packagist.org: #StandWithUkraine
Installing drupal/recommended-project (9.4.5)
```

On peut voir que l'on lance le shell init_drupal.sh

```
  Congratulations, you've installed the Drupal codebase
  from the drupal/recommended-project template!


Next steps:
  * Install the site: https://www.drupal.org/docs/8/install
  * Read the user guide: https://www.drupal.org/docs/user_guide/en/index.html
  * Get support: https://www.drupal.org/support
  * Get involved with the Drupal community:
      https://www.drupal.org/getting-involved
  * Remove the plugin that prints this message:
      composer remove drupal/core-project-message
  * Homepage: https://www.drupal.org/project/drupal
  * Support:
    * docs: https://www.drupal.org/docs/user_guide/en/index.html
    * chat: https://www.drupal.org/node/314178
Removing intermediate container d9bed779a272
 ---> a9d334fc4e9a
Step 21/22 : EXPOSE 80
 ---> Running in 7500294148fd
Removing intermediate container 7500294148fd
 ---> 512baad1f388
Step 22/22 : ENTRYPOINT service php7.4-fpm start && service mariadb start && mysql < /init_data/init_db.sql && nginx -g "daemon off;error_log /dev/stdout info;"
 ---> Running in b00051e0fd40
Removing intermediate container b00051e0fd40
 ---> e769b637c0bc
Successfully built e769b637c0bc
Successfully tagged my_drupal_volumes:latest
mariadb_data
my_drupal_volumes_c
5ba8cbbb8c697aa38aafebec1a74ebeed114aaf69878855da2065cb3557b02fb
```

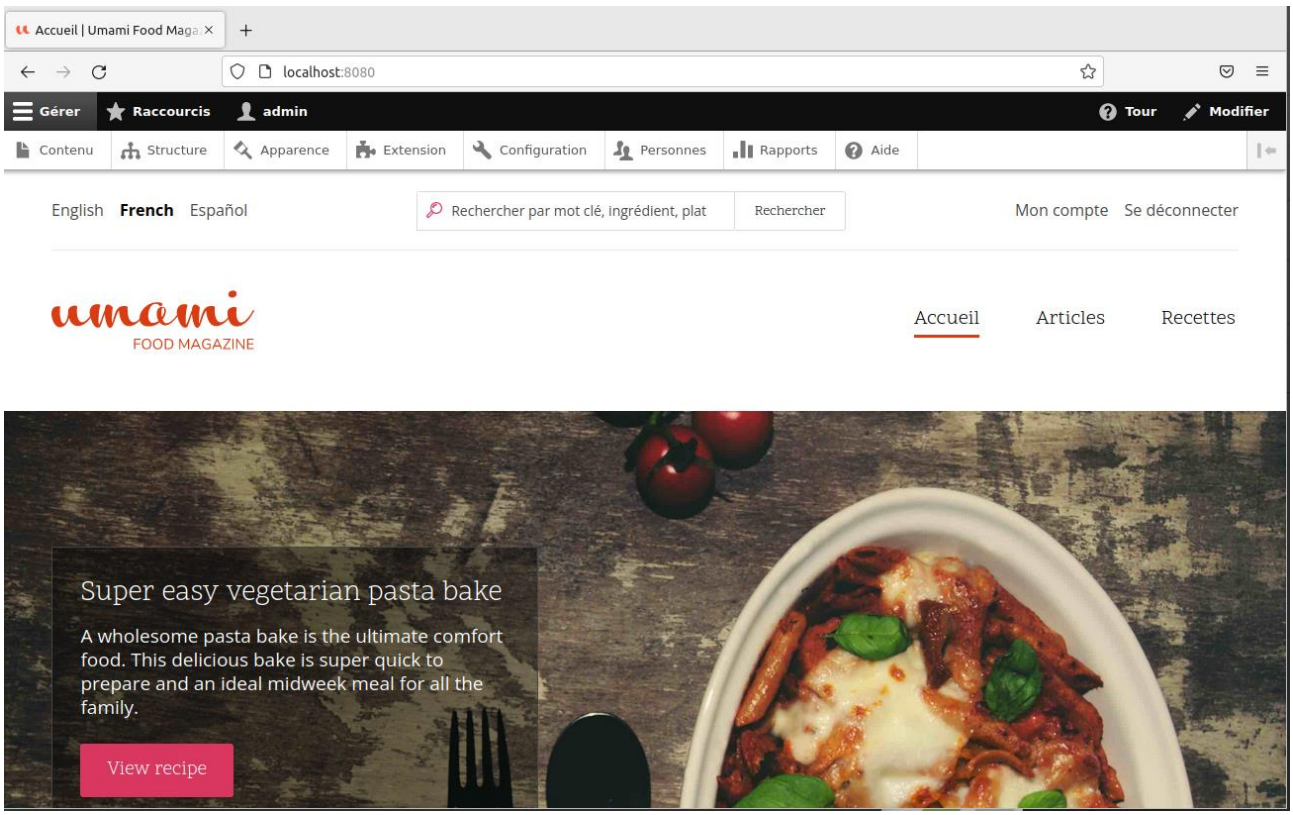
```
Creating and starting container my_drupal_volumes_c
09b5235159510133a60fbf43bb1b240ea017d1fd1f5d487af9b37899320bb78d
2022-08-14T13:29:21.744480076Z Starting MariaDB database server: mariadbd.
2022-08-14T13:29:21.960426392Z 2022/08/14 13:29:21 [notice] 217#217: using the "epoll" event method
2022-08-14T13:29:21.960468502Z 2022/08/14 13:29:21 [notice] 217#217: nginx/1.18.0
2022-08-14T13:29:21.960475607Z 2022/08/14 13:29:21 [notice] 217#217: OS: Linux 5.4.0-122-generic
2022-08-14T13:29:21.960479349Z 2022/08/14 13:29:21 [notice] 217#217: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022-08-14T13:29:21.960700671Z 2022/08/14 13:29:21 [notice] 217#217: start worker processes
2022-08-14T13:29:21.961316723Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 223
2022-08-14T13:29:21.961827531Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 224
2022-08-14T13:29:21.962153482Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 225
2022-08-14T13:29:21.962495236Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 226
```

XII.III. *Configuration du site*

Aller dans le navigateur avec l'URL http://localhost:8080/

Nous nous retrouvons dans le même état qu'au paragraphe § X.VII

Une fois l'installation terminée on arrive sur la page :



XII.IV. *Nouvelle exécution*

Faites un CTRL+C pour quitter l'affichage des logs inclus dans le script de lancement.

Pour être sûr vous pouvez arrêter le container par la commande :

```
docker stop my_drupal_volumes_c
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-drupal-volumes$ docker stop my_drupal_volumes_c
my_drupal_volumes_c
christophe@christophe-VirtualBox:~/Bureau/image-docker-drupal-volumes$
```

Rafraichissez la page du navigateur :

Pour relancer simplement le container sans qu'il soit détruit :

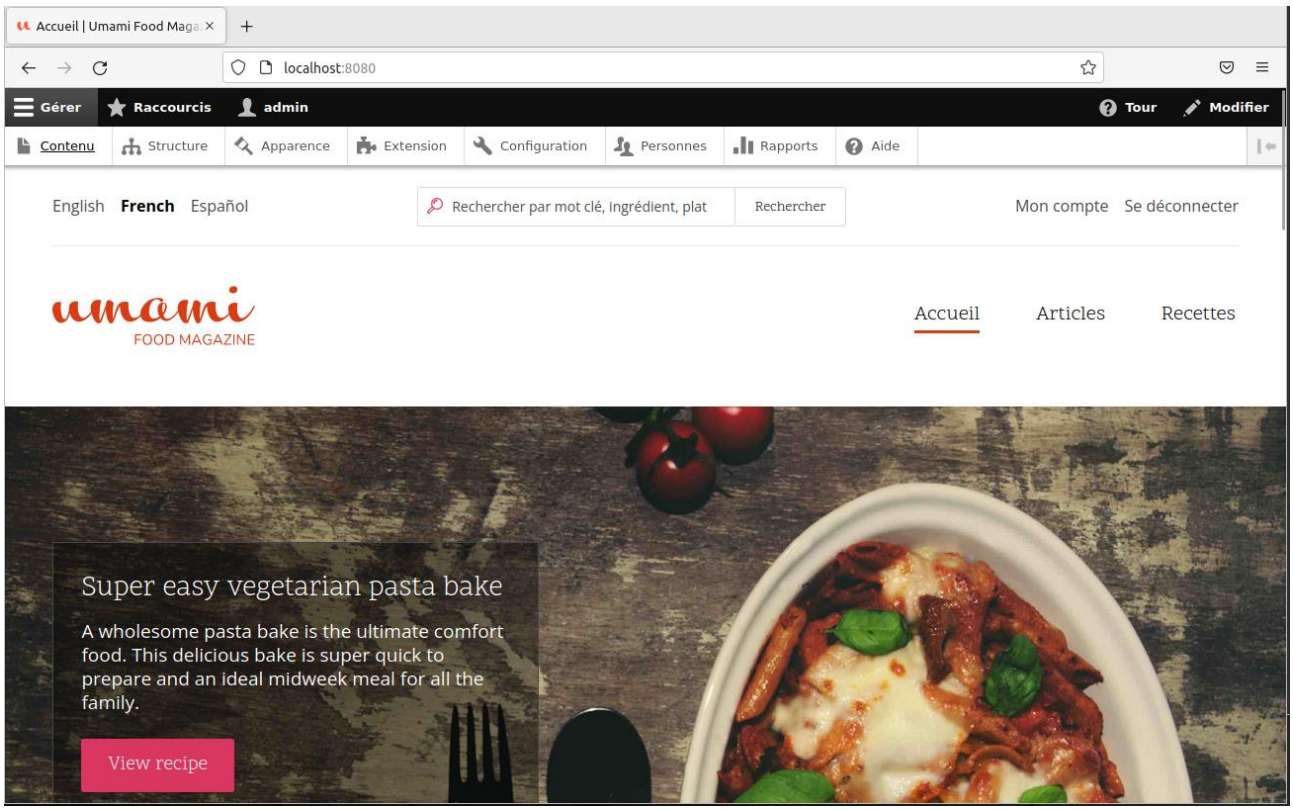
```
docker start my_drupal_volumes_c ;
```

Ou relancer le script :

```
./install_and_run.sh
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-drupal-volumes$ ./install_and_run.sh
Starting container my_drupal_volumes_c
my_drupal_volumes_c
2022-08-14T13:29:21.744480076Z Starting MariaDB database server: mariadbd.
2022-08-14T13:29:21.960426392Z 2022/08/14 13:29:21 [notice] 217#217: using the "epoll" event method
2022-08-14T13:29:21.960468502Z 2022/08/14 13:29:21 [notice] 217#217: nginx/1.18.0
2022-08-14T13:29:21.960475607Z 2022/08/14 13:29:21 [notice] 217#217: OS: Linux 5.4.0-122-generic
2022-08-14T13:29:21.960479349Z 2022/08/14 13:29:21 [notice] 217#217: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022-08-14T13:29:21.960700671Z 2022/08/14 13:29:21 [notice] 217#217: start worker processes
2022-08-14T13:29:21.961316723Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 223
2022-08-14T13:29:21.961827531Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 224
2022-08-14T13:29:21.962153482Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 225
2022-08-14T13:29:21.962495236Z 2022/08/14 13:29:21 [notice] 217#217: start worker process 226
```

Le site conserve ses données !



XII.V. *Consommation des ressources*

Une commande utile pour savoir combien de ressources exploite un conteneur :

```
docker stats <CONTAINER_NAME>
```

```
christophe@christophe-VirtualBox:~/Bureau/image-docker-drupal-volumes$ docker stats my_drupal_volumes_c
```

```
christophe@christophe-VirtualBox: ~/Bureau/image-docker-drupal-volumes
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O         PIDS
09b523515951   my_drupal_volumes_c   0.02%     118.2MiB / 3.839GiB   3.01%     15kB / 57.7kB   7.68MB / 90.1kB   20
```

XIII. Fin du document