FACULTE DES SCIENCES DE TUNIS

Mini projet LINUX

Installation d'un conteneur Docker sur un OS Ubuntu

Réalisé par : Sami Majouli

SRT M2 en ligne

Sommaire

I. Aperçu global du projet

II. La technologie de Docker /conteneur

1. Docker
2. Les conteneurs
3. Avantages du docker

III. Environnements

1. Environnement matériel
2. Environnement logiciel

IV. Réalisation

1. Configuration de référentiel Docker
2. Installation de moteur docker

V. Réalisation d'un Serveur HTTP Python dans un Conteneur Docker

1. Création de l'Application Python et configuration du Conteneur Docker
2. Construction et Exécution du Conteneur Docker
3. Test de l'Application dans le Navigateur

Conclusion

BIBLIOGRAPHIE

I- Aperçu global du projet

Le but de ce projet consiste à implanter un conteneur Docker sur un système d'exploitation Ubuntu. Le déroulement du projet suit les étapes ci-dessous :

- ✓ Installation d'une image ISO Ubuntu sur VMWare Workstation
- ✓ Installation d'un conteneur Docker compatible avec la version Ubuntu installée
- ✓ Enfin, le projet se conclut par la phase de test du bon fonctionnement.
- ✓ Création de l'Application Python et configuration du Conteneur Docker
- ✓ Test de l'Application dans le Navigateur

II- La technologie de Docker /conteneur

1- Docker

Qu'est-ce que Docker ? Docker, lancé en 2013, est une plateforme de conteneurs qui a grandement contribué à populariser la conteneurisation. Cette technologie permet la création aisée de conteneurs et d'applications basées sur ces derniers. Bien qu'il existe d'autres solutions, Docker demeure la plus largement utilisée, en grande partie grâce à sa facilité de déploiement et d'utilisation par rapport à ses concurrents.

Docker est une solution open source, sécurisée et économique, bénéficiant de contributions de nombreux individus et entreprises. Une vaste communauté a développé un écosystème étendu de produits, services et ressources en soutien à ce projet.

Initialement conçu pour Linux, Docker prend également en charge les conteneurs sur Windows ou Mac en utilisant une "layer" de virtualisation Linux entre le système d'exploitation Windows/macOS et l'environnement d'exécution Docker. Cette approche permet l'exécution de conteneurs Windows natifs sur des environnements de conteneurs Windows ou Linux.

Quels sont les différents éléments de Docker ? La plateforme Docker repose sur diverses technologies et composants, dont voici les principaux :

1- **Docker Engine :** Il s'agit de l'application installée sur la machine hôte pour créer, exécuter et gérer des conteneurs Docker. Le Docker Engine, aussi appelé moteur Docker, agit comme une technologie client-serveur, facilitant la création et l'exécution des conteneurs. On distingue le Docker Engine Enterprise, avec des fonctionnalités de gestion avancées, et le Docker Engine Community, la version originale proposée en open source.

2- **Docker Daemon** : Le Docker Daemon gère les requêtes API pour administrer les aspects de l'installation, tels que les images, les conteneurs et les volumes de stockage.

3- **Docker Client** : Il constitue l'interface principale permettant la communication avec le système Docker. Le client Docker reçoit les commandes via l'interface en ligne de commande et les transmet au Docker Daemon.
4- **Dockerfile** : Chaque conteneur Docker commence par un "Dockerfile", un fichier texte contenant des instructions compréhensibles pour la création d'une image Docker. Le Dockerfile précise le système d'exploitation, les langages, les variables d'environnement, les emplacements de fichiers, les ports réseau, et d'autres composants nécessaires au conteneur.
5- **Les images Docker** : Ces modèles en lecture seule sont utilisés pour créer des conteneurs Docker. Composées de plusieurs couches, les images regroupent installations, dépendances, bibliothèques, processus et codes d'application nécessaires pour créer un environnement de conteneur pleinement opérationnel. Après la rédaction du Dockerfile, l'utilitaire "build" est utilisé pour créer une image, représentant un fichier portable indiquant les composants logiciels que le conteneur exécutera et comment.

2- Les conteneurs

Les conteneurs sont des unités d'exécution légères et autonomes créées à partir d'images Docker. Ces images, composées de plusieurs couches, encapsulent l'ensemble des éléments nécessaires à l'exécution d'une application, tels que le code, les bibliothèques, les dépendances, les variables d'environnement et les fichiers de configuration. Les conteneurs offrent un environnement isolé et cohérent pour le déploiement d'applications, indépendamment de l'infrastructure sous-jacente.

L'utilisation de conteneurs présente plusieurs avantages. Tout d'abord, ils garantissent une portabilité élevée, permettant à une application de s'exécuter de manière cohérente sur différents environnements, qu'il s'agisse d'un ordinateur local, d'un serveur en centre de données ou dans le cloud. De plus, les conteneurs sont rapides à démarrer, facilitant ainsi le déploiement et l'évolutivité des applications.

La technologie des conteneurs repose sur le concept de virtualisation au niveau du système d'exploitation. Contrairement aux machines virtuelles, les conteneurs partagent le même noyau du système d'exploitation hôte, ce qui les rend plus légers et plus efficaces en termes de ressources. Cette approche permet d'optimiser l'utilisation des ressources système tout en assurant une isolation entre les applications.

En résumé, les conteneurs offrent une solution efficace pour l'emballage, la distribution et l'exécution d'applications, simplifiant ainsi le processus de développement, de déploiement et de gestion des logiciels.

3- Avantages du docker

Docker présente plusieurs avantages significatifs, ce qui explique en grande partie son adoption généralisée dans le domaine du développement logiciel et du déploiement d'applications. Voici quelques-uns des principaux avantages de Docker :

1- Portabilité : Les conteneurs Docker garantissent la portabilité des applications. Une fois empaquetée dans un conteneur, une application peut être exécutée de manière cohérente sur n'importe quel environnement prenant en charge Docker, qu'il s'agisse d'un ordinateur local, d'un serveur en centre de données ou dans le cloud.
2- Isolation : Les conteneurs fournissent une isolation légère entre les applications. Chaque conteneur fonctionne de manière indépendante des autres, assurant ainsi qu'une application et ses dépendances sont encapsulées et ne perturbent pas les autres applications sur le même système.
3- Efficacité des ressources : Contrairement aux machines virtuelles, les conteneurs partagent le même noyau du système d'exploitation hôte. Cela les rend plus légers et plus efficaces en termes de ressources, permettant une utilisation plus efficace des capacités matérielles.
4- Rapidité de déploiement : Les conteneurs peuvent être démarrés en quelques secondes, offrant ainsi une rapidité de déploiement considérable. Cela facilite l'évolutivité des applications et la mise en œuvre de stratégies de déploiement continu.
5- Gestion simplifiée : Docker automatise le processus de déploiement, de gestion et de mise à l'échelle des applications. Les images Docker et les fichiers de configuration (Dockerfile) permettent de décrire de manière détaillée l'environnement d'exécution de l'application, simplifiant ainsi les opérations.
6- Écosystème robuste : Docker bénéficie d'une vaste communauté d'utilisateurs et de contributeurs. Le Docker Hub offre un registre public où les utilisateurs peuvent partager et découvrir des images Docker. Cette communauté active contribue au développement d'un écosystème riche en outils, en extensions et en ressources.
7- Développement et test facilités : Les développeurs peuvent créer des environnements de développement identiques à ceux de la production en utilisant des conteneurs. Cela élimine les problèmes liés aux différences d'environnement entre le développement et la production.
8- Évolutivité horizontale : Les applications basées sur Docker peuvent facilement être mises à l'échelle horizontalement, en ajoutant simplement de nouveaux conteneurs au sein d'un cluster, facilitant ainsi la gestion de la charge.
9- Sécurité : Les conteneurs Docker fournissent une couche de sécurité supplémentaire en isolant les applications. De plus, Docker intègre des fonctionnalités de sécurité, telles que la possibilité de définir des politiques de contrôle d'accès et des options de gestion des droits d'utilisateur.

En somme, Docker offre une solution complète pour la gestion des applications, améliorant l'efficacité du développement, du déploiement et de la gestion des logiciels.

III- Environnements

1- Environnement matériel

- Un ordinateur portable Dell inspiron 3580 :
  - Processeur : Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
  - Mémoire installé (RAM): 12,0 GB
  - Système d'exploitation: Windows 10 Home
- On installe une machine virtuelle Linux :
  - Processeur :Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
  - Mémoire installé (RAM) : 5044 Go
  - Système d'exploitation : Ubuntu 18.04

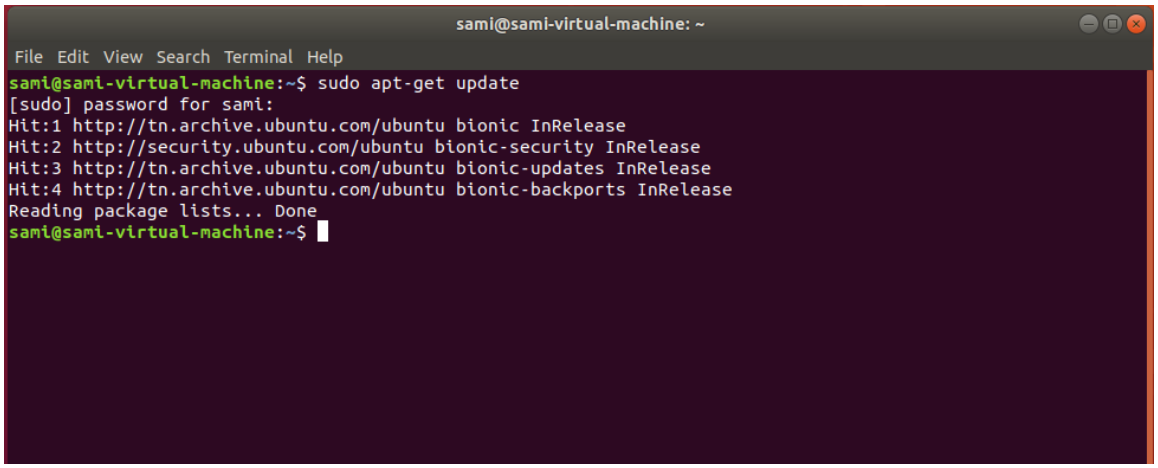
2- Environnement logiciel

VMware® Workstation 14 Pro version : 14.1.1 build-7528167

IV- Réalisation

1- Configuration de référentiel Docker

Mise à jour Ubuntu :



Installation des packages pour permettre à apt d'utiliser un référentiel via HTTPS avec la commande :

$ sudo apt-get install \

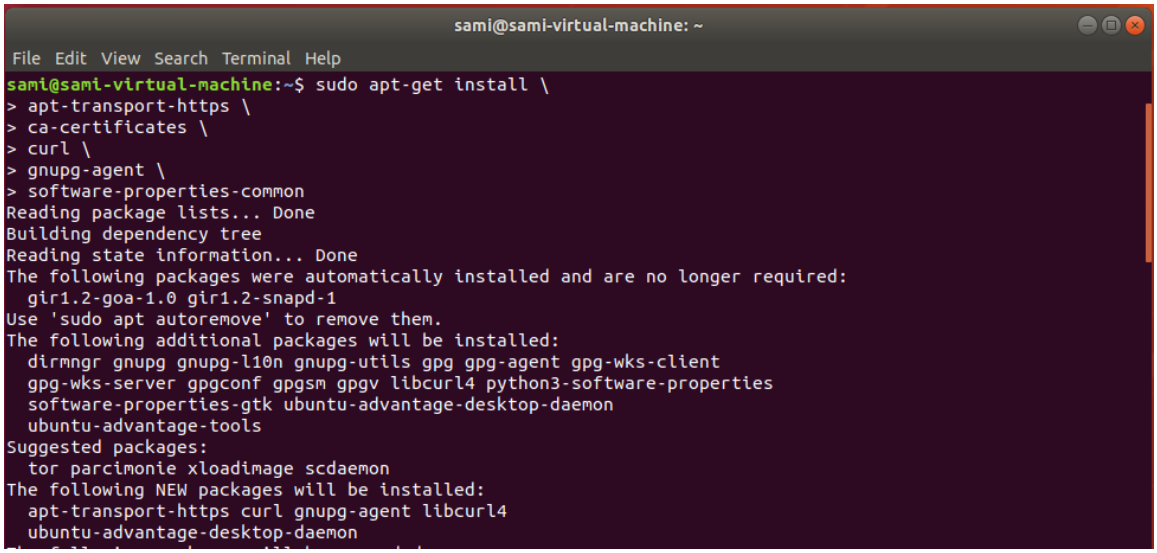
apt-transport-https \

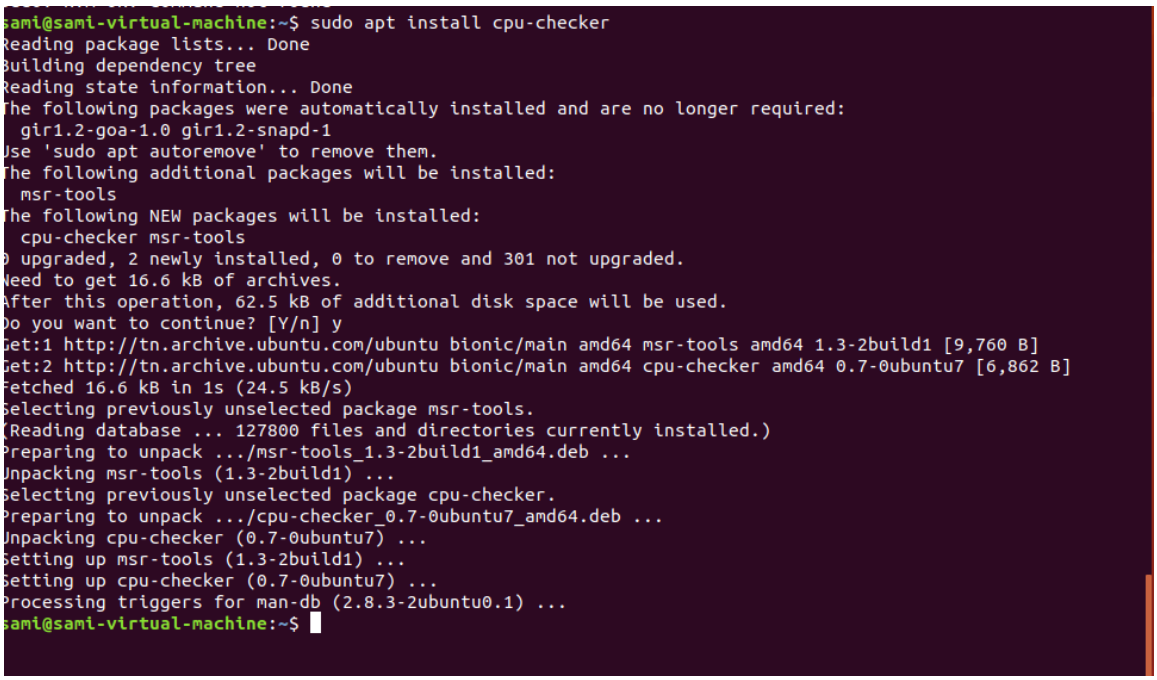
ca-certificates \

curl \

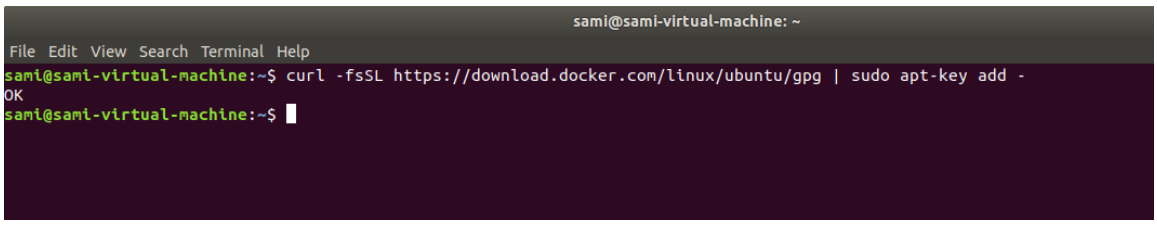
gnupg-agent \

software-properties-common





Puis ajouter la clé GPG du site de Docker avec la commande : $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -



Ensuite, nous ajoutons le dépôt pour configurer le référentiel stable :

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
sami@sami-virtual-machine:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Hit:1 http://tn.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://tn.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://tn.archive.ubuntu.com/ubuntu bionic-backports InRelease
Get:5 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]
Get:6 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [39.0 kB]
Fetched 103 kB in 1s (73.8 kB/s)
Reading package lists... Done
sami@sami-virtual-machine:~$
```

2- Installation de moteur docker

Mettre à jour l'index APT avec la commande : $ sudo apt-get upadate

```
sami@sami-virtual-machine:~$ sudo apt update
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://tn.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 https://download.docker.com/linux/ubuntu bionic InRelease
Hit:4 http://tn.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://tn.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
301 packages can be upgraded. Run 'apt list --upgradable' to see them.
sami@sami-virtual-machine:~$ apt list --upgradable
Listing... Done
apparmor/bionic-updates,bionic-security 2.12-4ubuntu5.3 amd64 [upgradable from: 2
apport/bionic-updates,bionic-updates,bionic-security,bionic-security 2.20.9-0ubun
apport-gtk/bionic-updates,bionic-updates,bionic-security,bionic-security 2.20.9-0
```

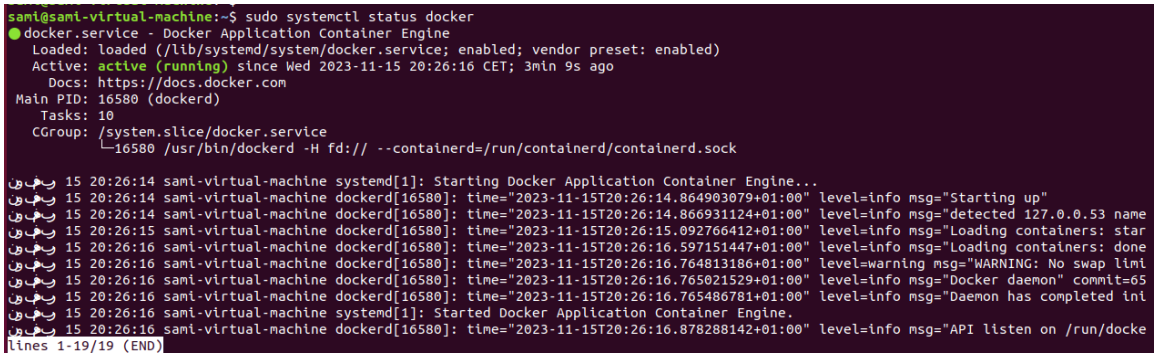
Installer la dernière version de Docker Engine et de container avec la commande :

$ sudo apt install docker-ce

```
sami@sami-virtual-machine:~$
sami@sami-virtual-machine:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 gir1.2-snapd-1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin git
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc git-el git-emai
  git-mediawiki git-svn
Recommended packages:
  slirp4netns
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-
  pigz
0 upgraded, 10 newly installed, 0 to remove and 301 not upgraded.
Need to get 115 MB of archives.
After this operation, 435 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://tn.archive.ubuntu.com/ubuntu bionic/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 https://download.docker.com/linux/ubuntu bionic/stable amd64 containerd.io amd64 1.6.21-1 [28.3
Get:3 http://tn.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all 0.17025-1 [22.8 kB]
Get:4 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git-man all 1:2.17.1-1ubuntu0.18 [
Get:5 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git amd64 1:2.17.1-1ubuntu0.18 [3,
Get:6 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-buildx-plugin amd64 0.10.5-1
```

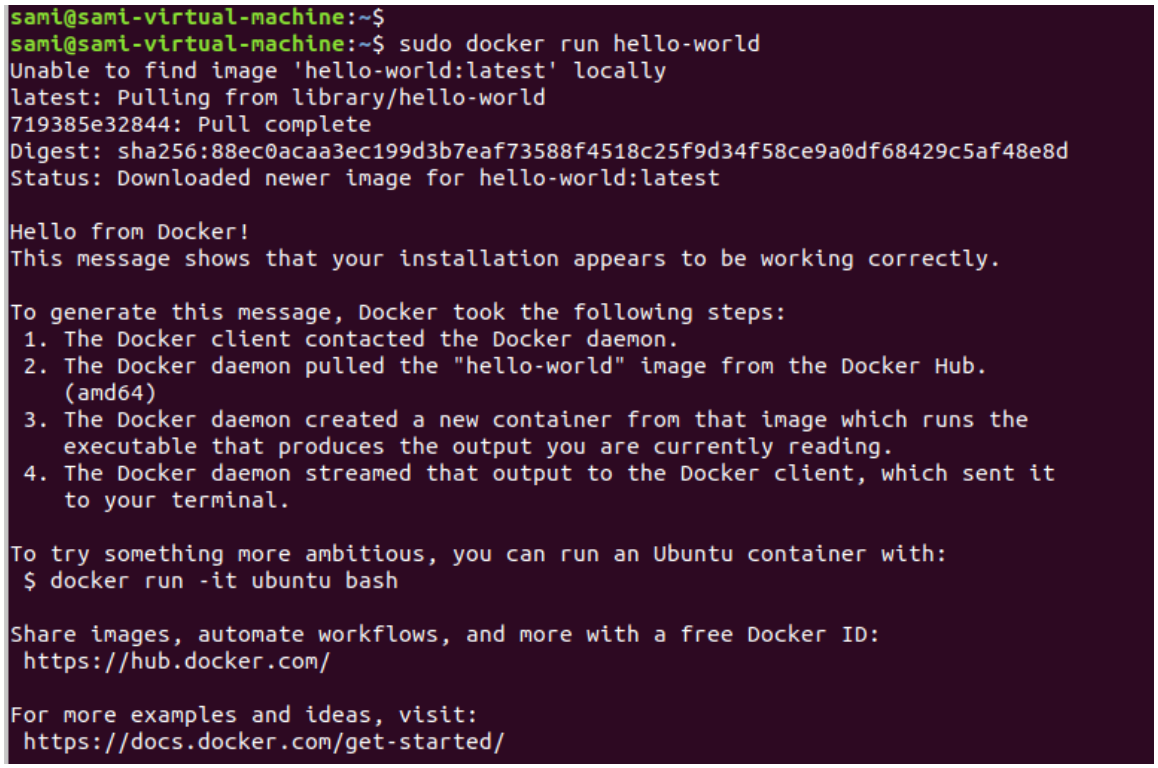
Une fois l'installation terminée, on vérifie l'état du service avec la commande :

$ sudo systemctl status docker



Une fois que Docker est installé, il suffit d'utiliser l'image de test pour vérifier que tout fonctionne comme prévu. Faites-le avec la commande suivante :

$ sudo docker run hello-world



L'image "hello-world" s'est bien exécuté et cela montre que tout fonctionne correctement

Maintenant, si nous voulons rechercher des images disponibles, il suffit d'utiliser la commande suivante :

$ sudo docker search debian

```
sami@sami-virtual-machine:~$ sudo docker search debian
NAME                              DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
ubuntu                            Ubuntu is a Debian-based Linux operating sys…   16588     [OK]
debian                            Debian is a Linux distribution that's compos…   4847      [OK]
neurodebian                       NeuroDebian provides neuroscience research s…   105       [OK]
bitnami/debian-base-buildpack     Debian base compilation image                   2                    [OK]
kasmweb/debian-bullseye-desktop   Debian Bullseye desktop for Kasm Workspaces     6
kasmweb/debian-bookworm-desktop   Debian Bookworm desktop for Kasm Workspaces     2
mirantis/debian-build-ubuntu-xenial                                               0
rancher/debianconsole                                                             1
datadog/debian-i386                                                               0
mirantis/debian-build-ubuntu-trusty                                               0
osrf/debian_arm64                 Debian arm64 Base Images                        1
dokken/debian-10                  Debian 10 image for use with kitchen-dokken     0
dokken/debian-9                   EOL: Debian 9 image for kitchen-dokken          0
pihole/debian-base                Debian docker with a custom pam build           5
dokken/debian-11                  Debian 11 image for use with kitchen-dokken     1
ustclug/debian                    Official Debian Image with USTC Mirror          2
dokken/debian-8                   EOL: Debian 8 image for kitchen-dokken          1
jitesoft/debian                   Debian base image.                              0
dockette/debian                   My Debian Sid | Jessie | Wheezy Base Images     2                    [OK]
dokken/debian-12                  Debian 12 image for use with Test Kitchen's …   0
corpusops/debian-bare             https://github.com/corpusops/docker-images/     0
corpusops/debian                  debian corpusops baseimage                      0
pihole/debian-debootstrap         fork of multiarch/debian-debootstrap            0
homebrew/debian7                  Homebrew maintainer image for building patch…   0
osrf/debian_armhf                 Debian Armhf Base Images                        1
sami@sami-virtual-machine:~$
```

Nous pouvons lister les images dans le système avec cette commande :

$ sudo docker images

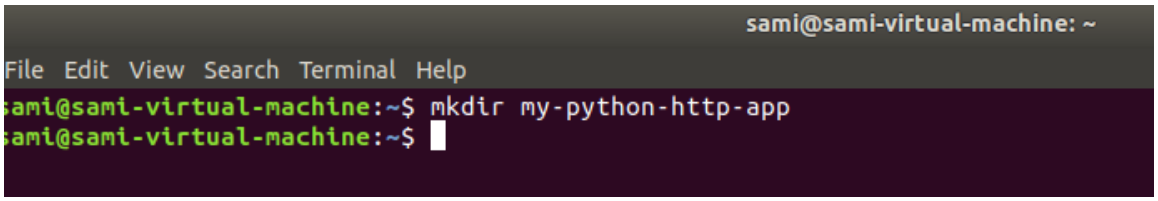
```
sami@sami-virtual-machine:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
hello-world   latest    9c7a54a9a43c   6 months ago    13.3kB
sami@sami-virtual-machine:~$
```

VI. Réalisation d'un Serveur HTTP Python dans un Conteneur Docker

1. Création de l'Application Python et Configuration du Conteneur Docker

- **Création d'un nouveau dossier pour l'application :**

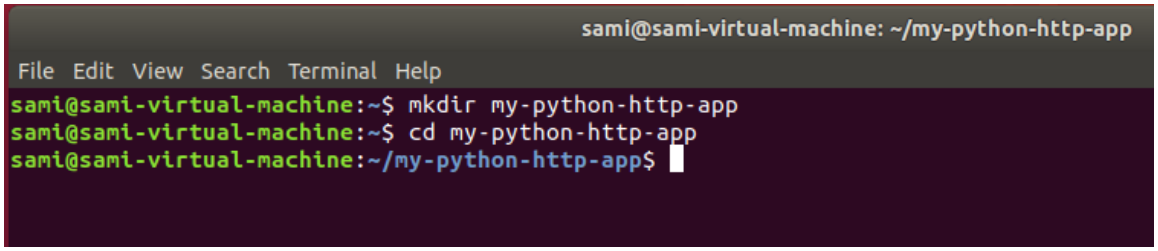
mkdir my-python-http-app



J'ai créé un nouveau dossier appelé "my-python-http-app"

J'accède à ce dossier via la commande :

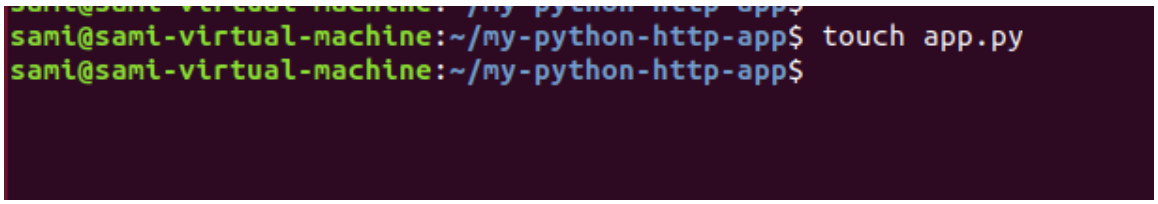
cd my-python-http-app



- **Création du script Python (app.py) :**

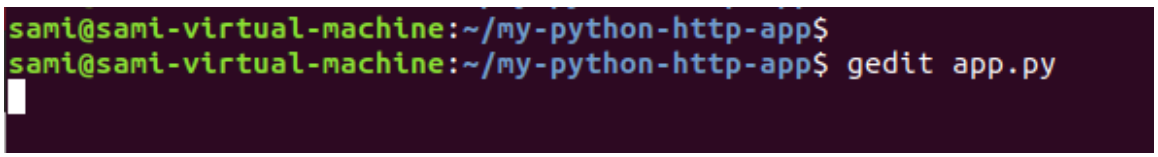
Je crée tout d'abord un fichier avec le nom : « app.py»

touch app.py

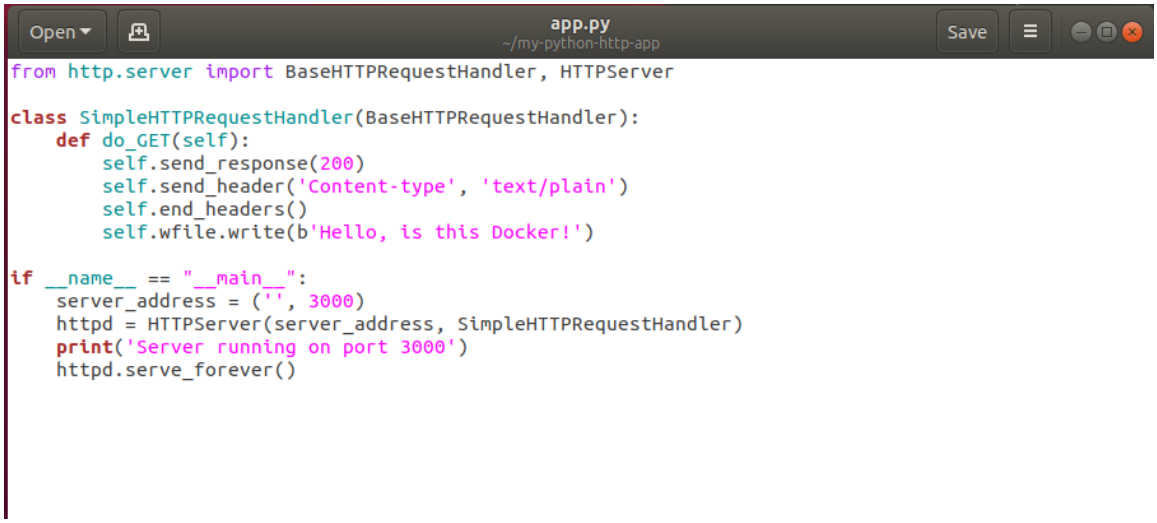


J'accéde au fichier :

gedit app.py

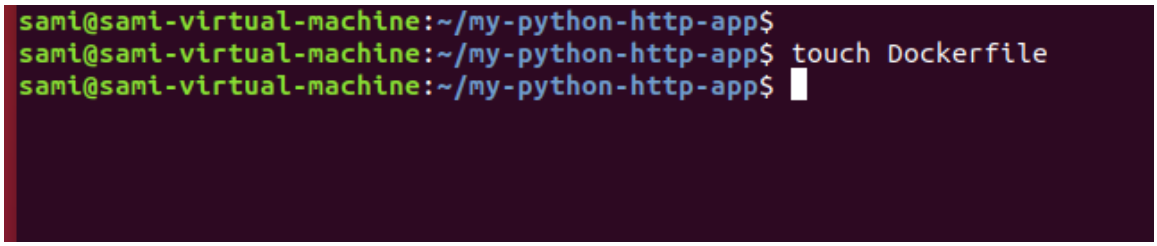


Je met le code python dans le fichier et j'enregistre



- **Créez un fichier Dockerfile :**

touch Dockerfile



J'accède au fichier Dockerfile

gedit Dockerfile

Open *Dockerfile ~/my-python-http-app Save

```
# Utilisez une image Python
FROM python:3.9

# Définissez le répertoire de travail
WORKDIR /app

# Copiez les fichiers de votre application
COPY . .

# Exposez le port sur lequel l'application écoute
EXPOSE 3000

# Commande pour démarrer votre application
CMD ["python", "app.py"]
```

2. Configuration et Exécution du Conteneur Docker

➢ Construction de l'image Docker :

Il faut s'assurer qu'on est bien dans le répertoire contenant le Dockerfile.

Cette commande construit une image Docker avec le tag "my-python-http-app:1.0".

sudo docker build -t my-python-http-app:1.0 .

```
sami@sami-virtual-machine:~/my-python-http-app$
sami@sami-virtual-machine:~/my-python-http-app$ sudo docker build -t my-python-http-app:1.0 .
[sudo] password for sami:
[+] Building 33.4s (4/7)
 => [internal] load build definition from Dockerfile                                                0.1s
 => => transferring dockerfile: 327B                                                                0.0s
 => [internal] load .dockerignore                                                                   0.1s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/python:3.9                                       5.7s
 => [1/3] FROM docker.io/library/python:3.9@sha256:30678bb79d9eeaf98ec0ce83cdcd4d6f5301484ef86873a711e69df2ca77e8a  27.4s
 => => resolve docker.io/library/python:3.9@sha256:30678bb79d9eeaf98ec0ce83cdcd4d6f5301484ef86873a711e69df2ca77e8ac  0.1s
 => => sha256:30678bb79d9eeaf98ec0ce83cdcd4d6f5301484ef86873a711e69df2ca77e8ac 1.86kB / 1.86kB       0.0s
 => => sha256:62072a293549fb69041a44936a3ea5cebcbc7195cc532e509fe940175b2f5430 2.01kB / 2.01kB       0.0s
```

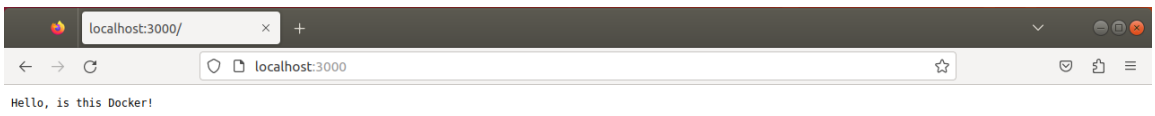
➢ Exécution du conteneur Docker :

sudo docker run -p 3000:3000 my-python-http-app:1.0

```
sami@sami-virtual-machine:~/my-python-http-app$
sami@sami-virtual-machine:~/my-python-http-app$ sudo docker run -p 3000:3000 my-python-http-app:1.0
[sudo] password for sami:
```

3. Test de l'Application dans le Navigateur

Le conteneur Docker a démarré l'application Python, et le serveur HTTP est accessible à l'adresse : « **http://localhost:3000** » dans le navigateur.



Voilà, une application Python vient d'être crée, cette application permet de créer un serveur HTTP basique, cette application s'est exécutée à l'intérieur d'un conteneur Docker.

Conclusion

En conclusion de ce mini-projet, j'ai réussi à créer une application Python simple qui crée un serveur HTTP basique. En utilisant Docker, j'ai encapsulé cette application dans un conteneur, permettant ainsi une isolation et une portabilité optimales.

Ce mini-projet m'a servi de point de départ pour comprendre les concepts fondamentaux de Docker et comment les intégrer dans le cycle de développement des applications, contribuant ainsi à une gestion plus efficace des dépendances, de l'isolation, et du déploiement.

BIBLIOGRAPHIE

- https://www.hostinger.fr/tutoriels/installer-docker-sur-ubuntu
- https://docs.docker.com/engine/install/ubuntu/
- https://www.codeflow.site/fr/article/how-to-install-and-use-docker-on-ubuntu-18-04
- https://doc.ubuntu-fr.org/docker