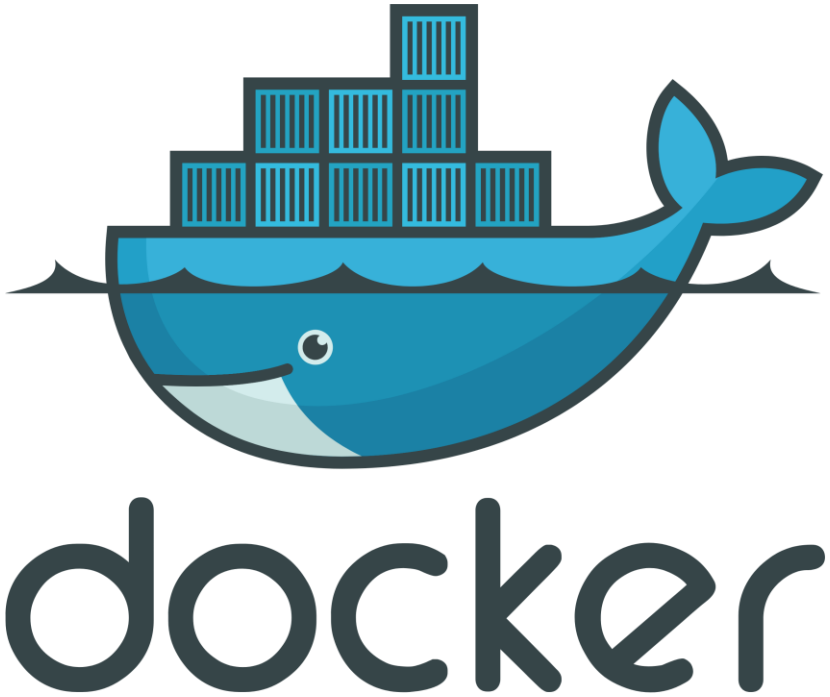


Docker

THOMAS GRZESINSKI

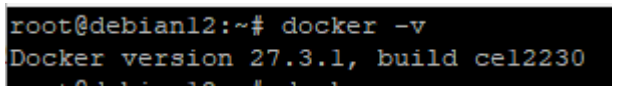
C’est quoi Docker ?

Définition: Docker est une plateforme de développement qui permet de créer, déployer et exécuter des applications dans des conteneurs, offrant ainsi un environnement isolé et portable. Les conteneurs regroupent tout le nécessaire pour faire fonctionner une application, garantissant la cohérence entre différents environnements. Docker facilite la gestion des dépendances et simplifie le déploiement des applications.

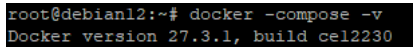
Installation de docker

Lien d'installation pour Docker: https://belginux.com/installer-docker-sous-debian-12/

Vérifier la version de votre docker avec la commande docker –v



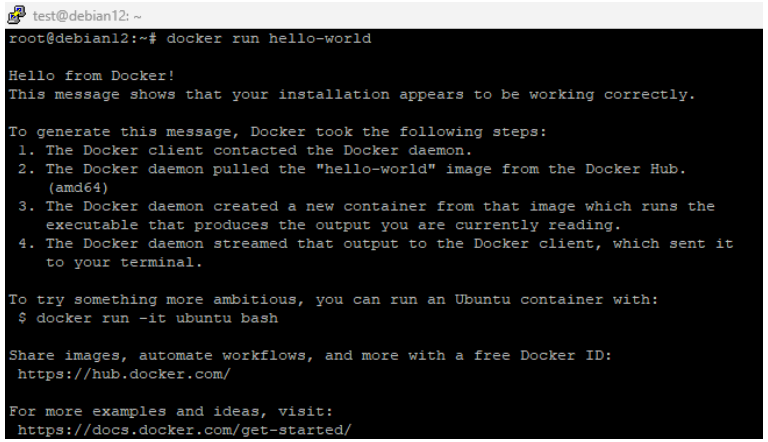
Pour obtenir la version de votre docker compose entrer la commande docker-compose -v



Utilisation de Docker

Pour vérifier si votre Docker fonctionne lancer votre premier conteneur en mettant

docker run hello-world



Si vous avez ceci c’est que votre Docker fonctionne correctement

Utilisation de Docker

Le docker hub contient des milliers d'images vous permettant de lancer votre container adapté à vos besoins

Saisir la commande : docker run -p 8080:80 -d nginx

run : lance le service

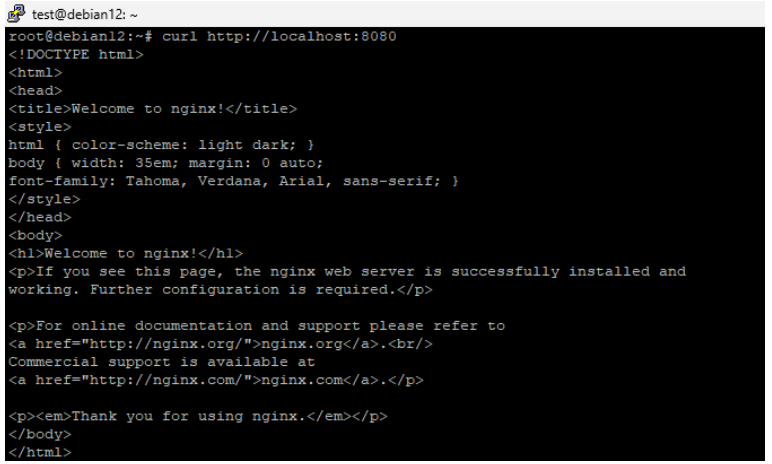
-p 8080:80 : mappe le port 8080 de la machine host vers le port 80 du container

-d : mode détaché

nginx : image

Utilisation de Docker

Accéder à votre page web en utilisant la commande curl http://localhost:8080 et vous pouvez voir que votre conteneur fonctionne donc correctement



La commande docker ps permet permet de lister les conteneurs qui sont en cours d'exécution

```
root@debian12:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                       NAMES
c68ad0da7acd   nginx     "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   crazy_snyder
```

Utilisation de Docker

La commande docker ps –a permet d'afficher tous les conteneurs Docker de votre système, qu'ils soient en cours d'exécution ou arrêtés.

```
root@debian12:~# docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS                      PORTS                                   NAMES
37085306d601   nginx         "/docker-entrypoint.…"   3 minutes ago    Created                                                             amazing_tu
3bf0d964788f   nginx         "/docker-entrypoint.…"   3 minutes ago    Created                                                             intelligent_menin
y
a884f835e3d2   nginx         "/docker-entrypoint.…"   10 minutes ago   Created                                                             affectionate_khor
a
07731fb6db6e   nginx         "/docker-entrypoint.…"   12 minutes ago   Created                                                             inspiring_bohr
c68ad0da7acd   nginx         "/docker-entrypoint.…"   13 minutes ago   Up 13 minutes               0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   crazy_snyder
be9e87db8862   hello-world   "/hello"                 19 minutes ago   Exited (0) 19 minutes ago                                           wonderful_hoover
1d856c521ea7   hello-world   "/hello"                 19 minutes ago   Exited (0) 19 minutes ago                                           loving_greider
```

La commande docker images affiche toutes les images Docker stockées localement sur votre machine

```
root@debian12:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
nginx         latest    3b25b682ea82   3 weeks ago     192MB
hello-world   latest    d2c94e258dcb   18 months ago   13.3kB
```

Utilisation de Docker

Pour arrêter votre container vous devez entrer la commande docker stop suivi de l'ID ou du nom du conteneur exemple

```
root@debian12:~# docker stop crazy_snyder
crazy_snyder
root@debian12:~# curl http://localhost:8080
curl: (7) Failed to connect to localhost port 8080 after 0 ms: Couldn't connect to server
```

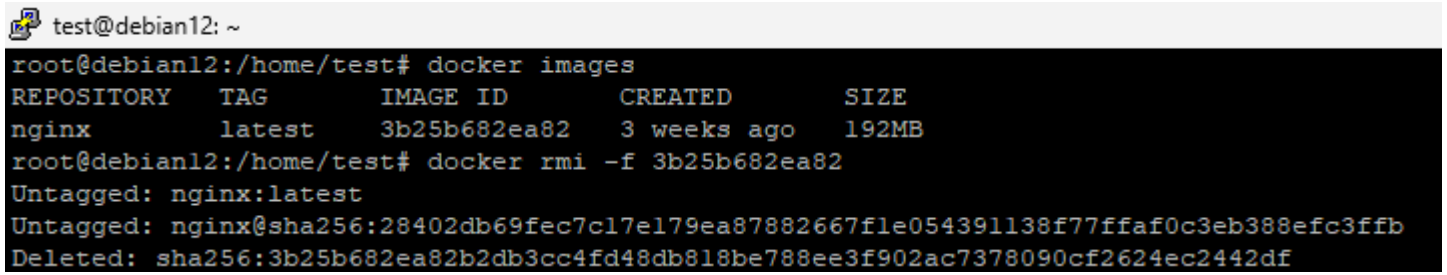
Le container nginx est donc arrêter

Pour supprimer un container vous devez donc entrer la commande docker rm suivis du nom du container

```
root@debian12:~# docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS                     PORTS     NAMES
3bf0d964788f   nginx         "/docker-entrypoint.…"   24 minutes ago  Created                              intelligent_meninsky
a884f835e3d2   nginx         "/docker-entrypoint.…"   30 minutes ago  Created                              affectionate_khorana
07731fb6db6e   nginx         "/docker-entrypoint.…"   32 minutes ago  Created                              inspiring_bohr
c68ad0da7acd   nginx         "/docker-entrypoint.…"   34 minutes ago  Exited (0) 3 minutes ago             crazy_snyder
be9e87db8862   hello-world   "/hello"                 39 minutes ago  Exited (0) 39 minutes ago            wonderful_hoover
1d856c521ea7   hello-world   "/hello"                 40 minutes ago  Exited (0) 40 minutes ago            loving_greider
root@debian12:~# docker rm intelligent_meninsky
intelligent_meninsky
root@debian12:~# docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS                     PORTS     NAMES
a884f835e3d2   nginx         "/docker-entrypoint.…"   31 minutes ago  Created                              affectionate_khorana
07731fb6db6e   nginx         "/docker-entrypoint.…"   33 minutes ago  Created                              inspiring_bohr
c68ad0da7acd   nginx         "/docker-entrypoint.…"   34 minutes ago  Exited (0) 3 minutes ago             crazy_snyder
be9e87db8862   hello-world   "/hello"                 40 minutes ago  Exited (0) 40 minutes ago            wonderful_hoover
1d856c521ea7   hello-world   "/hello"                 40 minutes ago  Exited (0) 40 minutes ago            loving_greider
```

Utilisation de Docker

Pour supprimer une image docker vous devez entrer la commande docker images et ensuite supprimer l'images avec la commande docker rmi –f suivi de l'id ou du nom de l'image



Téléchargement de l’image Apache

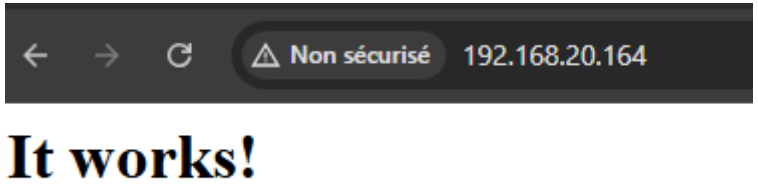
Vous pouvez utiliser apache sous docker

Configuration:

- Pour installer l’image apache vous devez entrer la commande docker pull httpd

- Lancer le conteneur avec la commande docker run –d –name apache 80:80 httpd

- Aller sur internet et taper l’adresse IP de la machine qui contient le docker et vous verrez que votre Apache fonctionne correctement



Téléchargement de l’image Apache

- Sinon si vous voulez le redirigez vers un autre port vous entrez la commande

docker run -d --name apache -p 8081:80 httpd

- Et vous verifiez son fonctionnement avec la commande curl http://localhost:8081

- Vous pouvez observer que votre apache fonctionne correctement

```
root@debian12:/home/test# curl http://localhost:8081
<html><body><h1>It works!</h1></body></html>
```

Création de votre image

On peut nous même créer une image correspondant a nos besoin

Configuration:

- Il vous faut créer un répertoire que nous allons nommer lab0 avec la commande mkdir lab0

```
root@debianl2:~# mkdir lab0
```

- Déplacer vous ensuite dans ce dossier avec la commande cd lab0

- Dans ce dossier créer un fichier que vous allez nommer Dockerfile avec la commande touch Dockerfile

```
root@debianl2:~/lab0# ls
Dockerfile
```

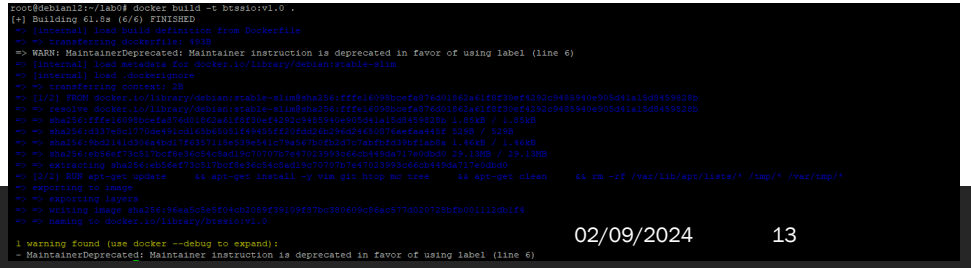
Création de votre image

- Ouvrez le fichier avec la commande nano Dockerfile et placez-y le contenu suivant:

```
# ---------------- DÉBUT COUCHE OS --------------------
FROM debian:stable-slim
# ---------------- FIN COUCHE OS ----------------------

# MÉTADONNÉES DE L'IMAGE
MAINTAINER BTS SIO2 SISR

# ---------------- DÉBUT COUCHE Installation ---------------
RUN apt-get update \
    && apt-get install -y vim git htop mc tree \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
# ---------------- FIN COUCHE Installation -----------------
```

- Sauvegarder le contenu de ce fichier

- Ensuite construisez l'image avec la commande docker build -t btssio:v1.0 .



Création de votre image

- Vérifier la taille de votre image avec la commande docker images

```
root@debian12:~/lab0# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
btssio       v1.0      96ea5c5e5f04   2 minutes ago    285MB
```

- On peut observer que la taille de notre image que nous venons de créer est de 285 MB ,

- Lancer maintenant votre image à l'aide de la commande docker run -tid –name sisr btssio:v1.0

- Et entrer dans le container et vérifier son fonctionnement avec la commande docker exec -ti sisr bash

- Vous pouvez maintenant vérifier chaque application avec les commandes:
  - vim --version
  - Git –version
  - htop --help
  - mc --version
  - tree --version

Création de votre image version légère

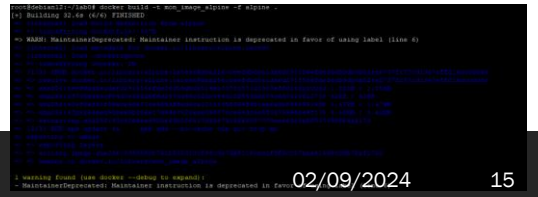
- Vous pouvez faire en sorte de créer de créer la même image mais en beaucoup plus légère avec une base beaucoup plus minimaliste comme Alpine Linux

- Vous devez donc créer un fichier dans votre dossier lab et créer un fichier que nous on va nommer alpine

- Dans ce fichier vous allez donc y mettre ceci

```
# --------------- DÉBUT COUCHE OS --------------------
FROM alpine:latest
# --------------- FIN COUCHE OS ---------------------

# MÉTADONNÉES DE L'IMAGE
MAINTAINER BTS SIO2 SISR

# --------------- DÉBUT COUCHE Installation ---------------
RUN apk update && \
    apk add --no-cache vim git htop mc
# --------------- FIN COUCHE Installation -----------------
```

- Vous construisez ensuite l'image avec la commande docker build -t mon_image_alpine -f alpine .



Création de votre image version lègère

- Exécuter ensuite le conteneur avec la commande docker run -it --name sisr_alpine mon_image_alpine

- Vérifier ensuite la taille de votre image avec la commande docker images

```
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
mon_image_alpine    latest    339559cb761b   40 seconds ago       63.2MB
```

- On peut observer que l'image est beaucoup moins lourde que la précédente

Penser a supprimer chaque images et containers a la fin de chaque Tp

- Stopper tout vos containers avec la commande docker stop $(docker ps -aq)

- Supprimer les tous avec la commande docker rm $(docker ps -aq)

- Supprimer tout les conteneurs arretes ou non avec la comande docker rm -f $(docker ps -aq)

Orchestrer des containers :

Votre équipe de dev vous demande de déployer et tester l'appli nécessitant les fichiers : app.py et requirements.txt . Ce dernier contient les applications qui seront installer via la commande pip. (ici redis et Flask) L'application écrite en python utilise le framework Flask ainsi que la bdd redis ,elle affiche une page avec votre hostname ,un nom provenant d'une variable d'environnement et un compteur du nombre de visites . Nous allons tout d'abord créer une image personnalisée (monimage) comprenant l'applicatif python des développeurs.

Configuration:

- Installer Docker commande avec la commande apt install docker-compose

- Création du répertoire projet `root@debian12:~# mkdir projet`

- Déplacer vous dans ce dossier

Orchestrer des containers :

- Une fois dedans créer les fichiers app.py requirements.txt Dockerfile

```
root@debian12:~/projet# ls -a
.  ..  app.py  Dockerfile  requirements.txt
```

- Dans le fichier requirements.txt spécifier ses dépendances

```
 GNU nano 7.2                                          requirements.txt *
Flask
redis
```

- Dans le fichier app.py mettez le code de l'application Flask

```
 GNU nano 7.2                                                  app.py *
from flask import Flask
import redis
import os

app = Flask(__name__)
r = redis.Redis(host='redis', port=6379)

@app.route('/')
def index():
    r.incr('counter')
    count = r.get('counter').decode('utf-8')
    hostname = os.getenv('HOSTNAME', 'Unknown Host')
    return f'Hostname: {hostname}, Visits: {count}'

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Orchestrer des containers :

- Dans le fichier Dockerfile décrivez l'image Docker

```
 GNU nano 7.2                                                  Dockerfile *
FROM python:2.7-slim

WORKDIR /app

COPY . /app

RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "app.py"]
```

- Créer ensuite le fichier docker-compose.yml

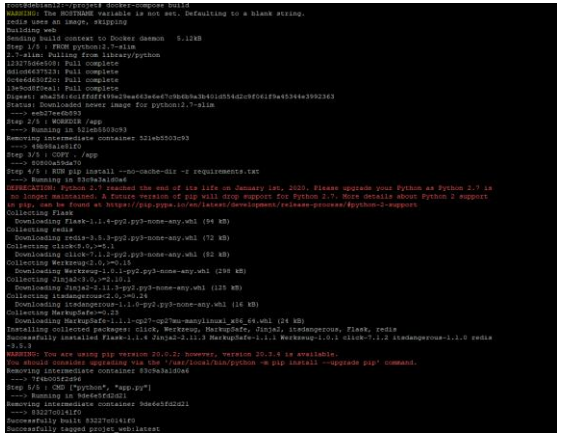
```
touch docker-compose.yml
```

- Mettez dans ce fichier ceci:

```
 GNU nano 7.2                                          docker-compose.yml *
version: '3'

services:
  web:
    build: .
    environment:
      - HOSTNAME=${HOSTNAME}
    ports:
      - "5000:5000"
    depends_on:
      - redis

  redis:
    image: "redis:alpine"
version: '3'

services:
  web:
    build: .
    environment:
      - HOSTNAME=${HOSTNAME}
    ports:
      - "5000:5000"
    depends_on:
      - redis

  redis:
    image: "redis:alpine"
```

Orchestrer des containers :

- Construisez ensuite l’image avec la commande dcker-compose build



Démarrer le conteneur avec la commande docker-compose up