

# Examen Docker

## Rappel du travail demandé

### Serveur WEB

Créer un serveur web qui affiche le fichier index.html

Le fichier index.html est produit par un autre conteneur qui insère dans le fichier la date toutes les 60 mn

Fournir un dossier explicatif avec :

- les fichiers sources
- les traces des sorties standard

## Compte rendu étapes par étapes

**N.B.:** toutes les instructions ont été fait par un user docker ayant des droits administration pour des raisons de bonnes pratiques.

## I – CREATION DU REPERTOIRE DE TRAVAIL

Création du repertoire de travail à l'aide de la commande **mkdir**

```
root@debian:/home/user# mkdir ubuntu
```

On se positionne dans le repertoire de travail avec la commande **cd**

```
root@debian:/home/user/ubuntu# cd ..  
root@debian:/home/user# cd ubuntu/
```

## II – CREATION DU SCRIPT

On souhaite qu'un conteneur produise et génère le fichier index.html. Pour cela nous allons créer un script qui s'appellera **entrypoint.sh**

```
root@debian:/home/user/ubuntu# vim entrypoint.sh
```

La commande **vim entrypoint.sh** va créer le fichier et l'ouvrir. On pourra écrire notre code.



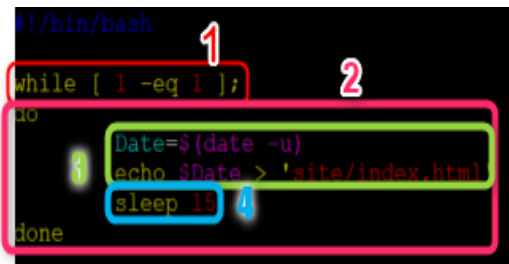
SCREENPRESSO.COM

Créer et partager vos captures d'écran avec Screenpresso (gratuit)

Le script devra indiquer la date toutes les heures dans un fichier index.html.

Pour cela nous allons:

- 1- Créer une boucle infini **while [ 1 -eq 1 ]**; => Comme 1=1 sera toujours vrai, il va exécuter la tâche 2 de façon illimitée
- 2- **Do ... done** comportera les instructions de la boucle
- 3- Déclarer une variable **Date** qui va prendre la valeur de **date -u** qui est une fonction qui affiche l'heure. **Echo \$Date > 'site/index.html'** va générer la date dans le fichier **index.html**
- 4- **Sleep 15** est le temps de pause avant de commencer une nouvelle boucle.



```
#!/bin/bash
while [ 1 -eq 1 ];
do
    Date=$(date -u)
    echo $Date > 'site/index.html'
    sleep 15
done
```

Voici le script final avec le sleep de 3600 secondes équivalent à **1 heure**.

Executable File	13 lines (7 sloc)	115 Bytes
-----------------	-------------------	-----------

```
1  #!/bin/bash
2
3  while [ 1 -eq 1 ];
4  do
5      Date=$(date -u)
6      echo $Date > 'site/index.html'
7      sleep 3600
8  done
9
```

Avec la commande **chmod +x entryptpoint.sh**, on donne des droits d'exécution sur ce fichier.

```
root@debian:/home/user/ubuntu# chmod +x entryptpoint.sh
```

### III – CREATION DU DOCKERFILE

Création du Dockerfile avec **vim Dockerfile**

```
root@debian:/home/user/ubuntu# vim Dockerfile
```

Le contenu du Dockerfile:

- 1- On fera appel à la **dernière image de ubuntu**
- 2- On **copie** le script **entrypoint.sh** local sur le **conteneur ubuntu**
- 3- On initialise le **entrypoint.sh** comme étant le **script à exécuter au démarrage** du conteneur ubuntu
- 4- On aurait pu rajouter le repertoire de travail si on n'avait pas été dessus avec **WORKDIR**

#!/\ Erreur sur la capture remplacer “exec” par “sh”

### IV – CREATION DU DOCKER-COMPOSE.YML

N.B.: On suppose que docker-compose a bien été installé sur le poste

Faire **vim docker-compose.yml** pour créer et insérer du contenu

```
root@debian:/home/user/ubuntu# vim docker-compose.yml
```

Dans ce fichier nous allons mettre:

- 1- Une rubrique services dans lequel il y aura les services et leurs utilisations
- 2- Deux services ubuntu et un serveur web nginx
- 3- Déclaration d'un volume commun aux deux services
- 4- Les utilisations de chaque service:
  - a. Pour **Ubuntu**, “**build: .**” va **construire** en s'appuyant sur le **Dockerfile** créé précédemment, lui indique le nom du conteneur et le **volume** sur lequel il va pointer
  - b. Pour le service **web**, mettre l'**image nginx:1.9.12**, le **nom** du conteneur, indiquer les **ports** sur lequel on veut pointer ainsi que le **volume** partage dans lequel le **fichier index.html** devra être récupéré
  - c. La **declaration** du volume “**vol\_share**” qui ne l'initialisera s'il n'a pas été créé

```

version: "3.8"

services:
  1 ubuntu: 2
    build: .
    container_name: product_site
    volumes:
      - vol_share:/site
  3 web: 4
    image: nginx:1.9.12
    container_name: srv_web
    ports:
      - 8080:80
    volumes:
      - vol_share:/usr/share/nginx/html

volumes:
  vol_share:

```

Maintenant que nous avons fini les configurations, on peut exécuter la commande **docker-compose up -d** qui va permettre de lancer toutes les instructions que l'on a indiquées dans le `docker-compose.yml`

```

root@debian:/home/user/ubuntu# docker-compose up -d
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Creating network "ubuntu_default" with the default driver
Building ubuntu
Sending build context to Docker daemon 4.096kB

Step 1/3 : FROM alpine:latest
latest: Pulling from library/alpine
4c0d98bf9879: Pulling fs layer
4c0d98bf9879: Verifying Checksum
4c0d98bf9879: Download complete
4c0d98bf9879: Pull complete
Digest: sha256:08d6ca16c60fe7490c03d10dc339d9fd8ea67c6466dea8d558526b1330a85930
Status: Downloaded newer image for alpine:latest
--> e50c909a8df2
Step 2/3 : COPY /entrypoint.sh /entrypoint.sh
--> 600016d94a3f
Step 3/3 : ENTRYPOINT [ "sh", "/entrypoint.sh" ]
--> Running in 6daalad3034f
Removing intermediate container 6daalad3034f
--> 46c54163096d
Successfully built 46c54163096d
Successfully tagged ubuntu_ubuntu:latest
WARNING: Image for service ubuntu was built because it did not already exist. To rebuild this image you must use 'docker-compose build' or 'docker-compose up --build'.
Creating product_site ... done
Creating srv_web ... done

```

Avec **docker-compose ps**, on visualise les services lancés par **Compose**.

```

root@debian:/home/user/ubuntu# docker-compose ps
      Name                Command             State              Ports
-----
product_site    sh /entrypoint.sh   Up
srv_web         nginx -g daemon of  Up    443/tcp, 0.0.0.0:8080->80/tcp

```

La commande `docker ps` nous **montrera les conteneurs** qui sont en train de **s'exécuter**.

```

root@debian:/home/user/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
9dlc141baae2   ubuntu_ubuntu  "sh /entrypoint.sh"     About a minute Up           About a minute
210db7c13fdb   nginx:1.9.12  "nginx -g 'daemon of..." About a minute Up           About a minute 443/tcp, 0.0.0.0:8080->80/tcp

```

La commande précédente est utilisée pour récupérer l'id du conteneur nginx. En faisant un **"docker inspect + ID\_Container"**

```
root@debian:/home/user/ubuntu# docker inspect 210db7c13fdb
```

Ce dont on a besoin dans l'**inspect** est l'**adresse ip** du serveur web et celui-ci est **192.168.112.2**

```
210db7c13fdb
{
  "NetworkID": "feabe09823b41aa8e0bccc08647e9323c0c97234ac3f30d58ef22d94ea374f6",
  "EndpointID": "73c3e42elf2746753482e2cfaf6e891e0c9732bd827369d0fb6027c1de76d948",
  "Gateway": "192.168.112.1",
  "IPAddress": "192.168.112.2",
  "IPPrefixLen": 20,
  "IPv6Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "MacAddress": "02:42:c0:a8:70:02",
  "DriverOpts": null
}
```

Pour voir le contenu du site et verifier que tout fonctionne bien.

Il y a deux possibilités:

- 1- Faire un **curl 0.0.0.0:8080**
- 2- Faire un **curl 192.168.112.2:8080**

```
root@debian:/home/user/ubuntu# curl 192.168.112.2
Tue Feb 9 22:00:48 UTC 2021
root@debian:/home/user/ubuntu# curl 192.168.112.2
Tue Feb 9 22:00:48 UTC 2021
root@debian:/home/user/ubuntu# curl 192.168.112.2
Tue Feb 9 22:00:58 UTC 2021
```