

# MSCYBER#13 - Utilisation des conteneurs Docker

---

En tant qu'administrateur infrastructure et sécurité, vous devez vous familiariser avec la technologie des conteneurs pour partager vos connaissances avec les collaborateurs de la DSI afin de permettre le passage à cette technologie dans l'entreprise.

## Authors

Roblot Jean-Philippe - [jroblot.simplon@proton.me](mailto:jroblot.simplon@proton.me)

## Version

31/01/2024 - V1R0

## Releases

Docker 25.0.1   Ubuntu Server LTS 22.04   OpenSSH 10.0.11

Powered by <https://shields.io>

## Prérequis

### Machine serveur

- VPS accessible via SSH sur l'ip 192.168.1.201 sur le port TCP/22 et le compte (simplon/simplon)
- Le VPS dispose d'une pré-installation de l'outil DOCKER
- Vous devez modifier la configuration pour passer sur le port 666, interdire l'utilisation de mots de passe pour la connexion SSH et utiliser uniquement des clés SSH-RSA

### Machine client

- Laptop Windows 10 Pro Version 22h2
- Nom : GDO-PC-PF1M1RXE
- CPU : 6 core
- RAM 32Go
- SSD : 512Go
- Putty - OpenSSH

## Configuration du serveur

- Générer une paire de clés côté client : `ssh-keygen -t rsa`
- Copier la clé publique sur le serveur

```
cat ~/.ssh/id_rsa.pub | ssh user@server "mkdir -p ~/.ssh && cat >>
~/.ssh/authorized_keys"
```

- Modifier la configuration d' OpenSSH sur le serveur

```
Port 666
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

- Se reconnecter au serveur en spécifiant notre id à la première connexion

```
ssh simplon@192.168.1.201 -i id_rsa
```

## 1. Exploration des conteneurs

---

1. Vérifier l'installation et la version de docker disponible sur la plateforme

Basiquement :

```
docker -v

# Sortie
Docker version 25.0.1, build 29cf629
```

De façon détaillée :

```
docker info # Donne les info détaillée de Docker
```

Cette commande permet de répondre aux questions 2 et 3 également

2. Lister les conteneurs éventuellement en cours d'exécution ou à l'arrêt

```
#Via docker info
Server:
Containers: 1
Running: 0
Paused: 0
Stopped: 1

# Lister les containers en cours d'execution
Docker ps

# Lister tous les containers
docker ps --all
```

3. Lister les images de conteneurs déjà disponibles dans l'environnement

```
# Via docker info
Images: 1

# Plus spécifiquement
docker images
```

4. Où se trouve le registre d'images par défaut de la configuration ?

Sur un système Linux, l'emplacement par défaut est `/var/lib/docker/`. A titre information, voici les chemin sur Windows et MacOS :

```
C:\ProgramData\Docker\Desktop

~/Library/Containers/com.docker.docker/Data/vms/0/
```

5. Quel est le répertoire de travail par défaut de docker dans l'environnement ?

Paramétrable dans le Dockerfile via l'instruction `WORKDIR`, la valeur prise par défaut est `/`

6. Sur quel port Docker publie sont API de commande à distance ? Le port 2375

```
# via docker info
WARNING: API is accessible on http://0.0.0.0:2375 without encryption.
Access to the remote API is equivalent to root access on the host.
Refer
to the 'Docker daemon attack surface' section in the documentation for
more information: https://docs.docker.com/go/attack-surface/
```

### 7. Quels composants docker permet-il de manipuler ?

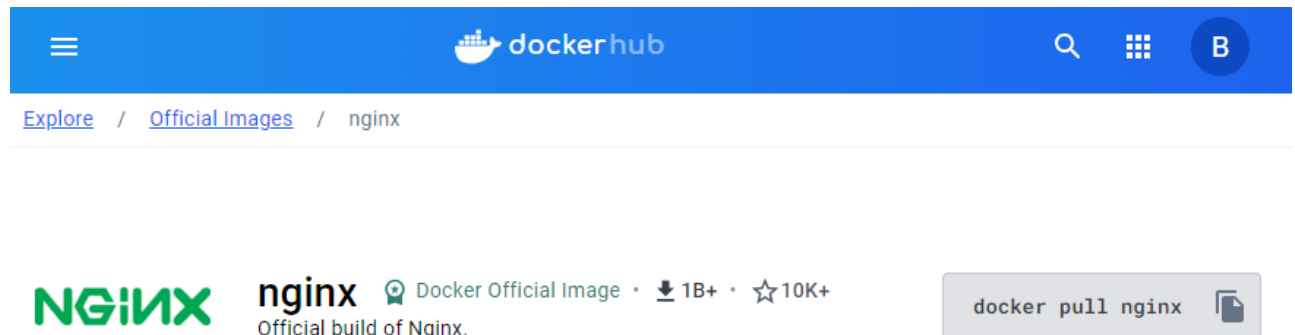
Nous pouvons manipuler des images, des conteneurs, du réseau et le stockage.

Ressource pour l'architecture Docker : [Architecture of Docker](#)

### 8. A quoi sert le Docker Hub ? Trouver l'image NGINX officiel sur le site en question

C'est un registre officiel permettant de gérer et partager des images open source ou propriétaire : [Docker Hub](#).

Image Nginx officielle :

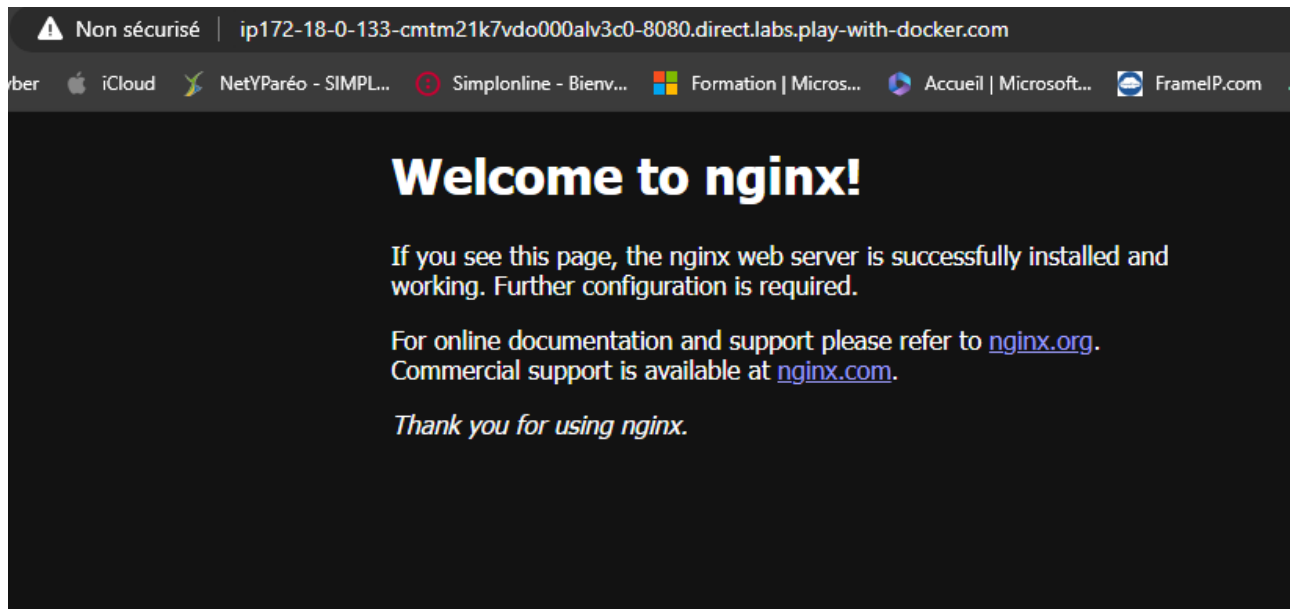


### 9. Lancer un premier conteneur en utilisant l'image de base NGINX et en exposant le site par défaut sur le port TCP/8080

```
# Rechercher l'image officielle via le terminal
$ docker search nginx
NAME                                DESCRIPTION
STARS    OFFICIAL    AUTOMATED
nginx                                         Official build of Nginx.
19555    [OK]

# Télécharger l'image
$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx

# Lancer le conteneur Nginx en ouvrant le port 8080
docker run -p 8080:80 -d nginx # le paramètre -d permet de lancer le
conteneur en tâche de fond et de garder la main sur notre terminal
```



10. Avez-vous pensé à utiliser le mode "détaché" pour lancer votre conteneur ? C'est utile ... ou pas ? (cf paramètre `-d` ci dessus). Si on ne lance pas en mode détaché, le conteneur se lance en premier plan et nous n'avons plus la main sur notre Docker.

11. Ouvrir une console terminal interactive sur le conteneur NGINX pour inspecter son contenu  
`sudo docker inspect nginx | less`

12. Arrêter votre conteneur NGINX

```
# Identifier notre conteneur via son id ou nom
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
PORTS         NAMES
b8fab17e39ee   nginx     "/docker-entrpoint...."  7 minutes ago  Up 7 minutes
0.0.0.0:8080->80/tcp    peaceful_payne

# Arrêter le conteneur
$ docker stop b8fab17e39ee
```

13. Afficher la liste des conteneurs terminés

```
$ docker ps -a --filter "status=exited"
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
PORTS         NAMES
b8fab17e39ee   nginx     "/docker-entrpoint...."  25 minutes ago  Exited (0) 17
minutes ago    peaceful_payne
```

14. Supprimer manuellement votre conteneur et vérifier la liste des conteneurs disponibles

```
$ docker rm b8fab17e39ee
```

```
$ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|-------|---------|---------|--------|-------|-------|
|--------------|-------|---------|---------|--------|-------|-------|

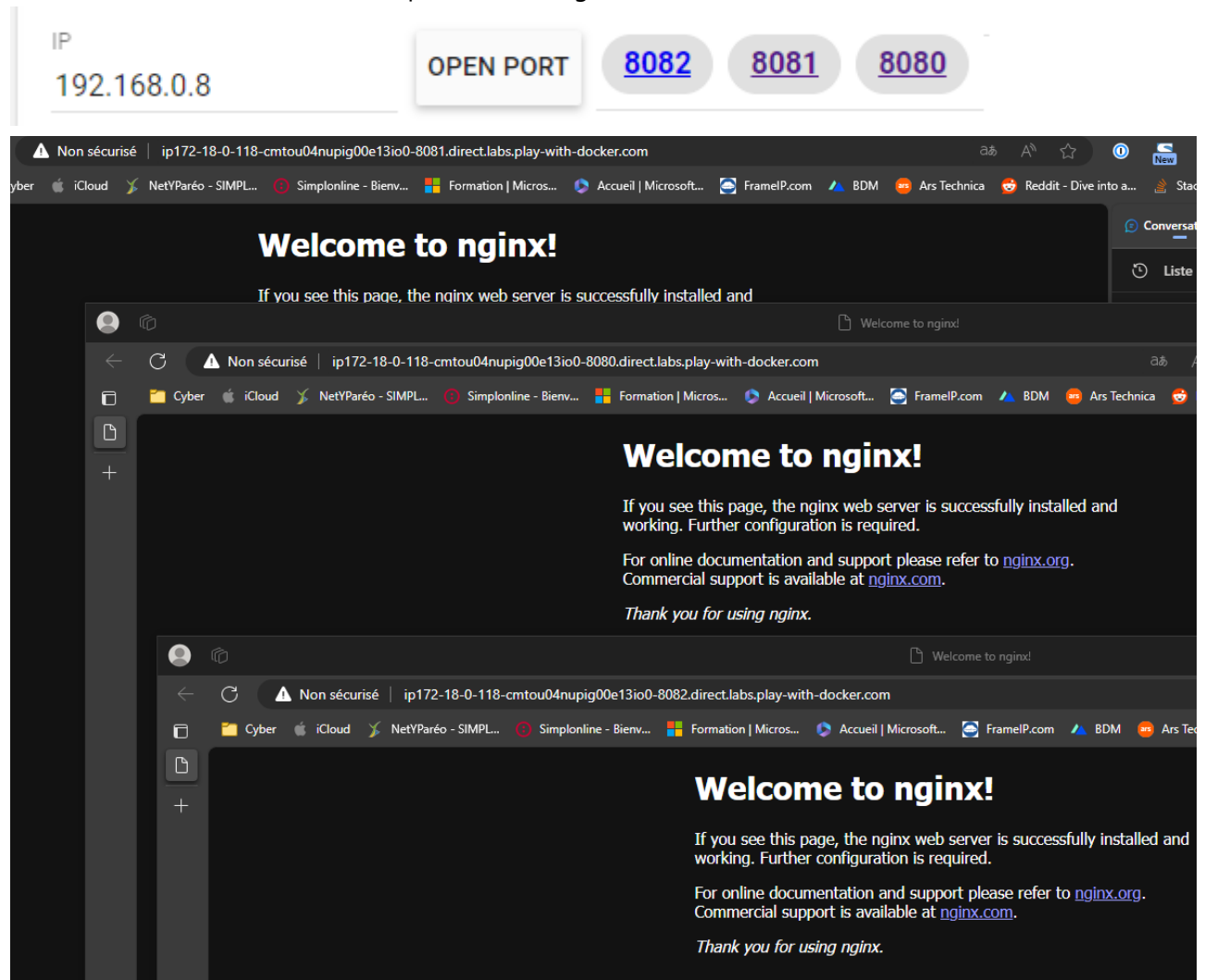
15. Lancer trois conteneurs NGINX en exposant chaque conteneur sur un port différent : 8080, 8081 et 8082

```
$ docker run -p 8080:80 -d nginx
```

```
$ docker run -p 8081:80 -d nginx
```

```
$ docker run -p 8082:80 -d nginx
```

16. Vérifier l'accès à vos conteneurs depuis votre navigateur



17. Arrêter tous les conteneurs NGINX

```
$ docker stop $(docker ps -q)
```

```
c163ac49bb57
```

```
6be1fd1f4638
```

```
63f7aa943292
```

18. Supprimer manuellement tous les conteneurs terminés en une seule commande

```
$ docker rm $(docker ps -a -q -f status=exited)
```

19. Lister les images disponibles sur votre VM

20. Inspecter le contenu (les couches) de l'image NGNIX disponible sur votre VM

21. Quelle est la différence entre une image et un conteneur ?

Une image Docker est un modèle en lecture seule, utilisé pour créer des conteneurs Docker. Elle est composée de plusieurs couches empaquetant toutes les installations, dépendances, bibliothèques, processus et codes d'application nécessaires pour un environnement de conteneur pleinement opérationnel.

22. A quoi sert un fichier Dockerfile ? Chaque conteneur Docker débute avec un " Dockerfile ". Il s'agit d'un fichier texte rédigé dans une syntaxe compréhensible, comportant les instructions de création d'une image Docker.

Un Dockerfile précise le système d'exploitation sur lequel sera basé le conteneur, et les langages, variables environnementales, emplacements de fichiers, ports réseaux et autres composants requis.

## 2. Ma première application

Suivre les instructions suivantes pour mettre en ligne votre première application web avec des conteneurs.

- Ouvrir une instance dans le PLAYGROUND DOCKER

Créer un répertoire de travail

```
$ mkdir my-webapp
$ cd my-webapp
```

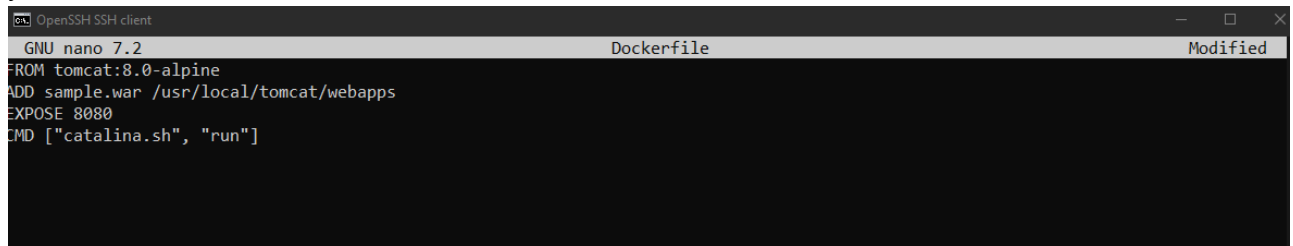
- Télécharger une application web Java \$ wget https://tomcat.apache.org/tomcat-8.0-doc/appdev/sample/sample.war

```
[node1] (local) root@192.168.0.8 ~
$ mkdir my-webapp
[node1] (local) root@192.168.0.8 ~
$ cd my-webapp
[node1] (local) root@192.168.0.8 ~/my-webapp
$ wget https://tomcat.apache.org/tomcat-8.0-doc/appdev/sample/sample.war
Connecting to tomcat.apache.org (151.101.2.132:443)
saving to 'sample.war'
sample.war 100% |*****| 4606 0:00:00 ETA
'sample.war' saved
```

- Créer & éditer le contenu du fichier Dockerfile

```
$ touch Dockerfile
$ nano Dockerfile
FROM tomcat:8.0-alpine
ADD sample.war /usr/local/tomcat/webapps
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

y

A screenshot of a terminal window titled 'OpenSSH SSH client'. Inside the terminal, the GNU nano 7.2 editor is open, editing a file named 'Dockerfile'. The file content is as follows:

```
FROM tomcat:8.0-alpine
ADD sample.war /usr/local/tomcat/webapps
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

The terminal window has standard window controls (minimize, maximize, close) in the top right corner.

- Créer une image locale de votre application

```
$ docker build -t my-webapp:v1
```

- Exécuter un conteneur à partir de votre image

```
$ docker run -p 80:8080 my-webapp:v1
```

- Vérifier en cliquant sur le lien « OPEN PORT - 80 » dans le DOCKER PLAYGROUND Ajouter le chemin `"/sample/"` dans l'url de la page pour voir le HELLO WORLD ... OU PAS !?



The image shows two screenshots of a web browser. The top screenshot is the Apache Tomcat 8.0.53 homepage. It features a navigation bar with links like Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists. A green banner congratulates the user for installing Tomcat. Below this, there's a 'Recommended Reading' section with links to 'Security Considerations HOW-TO', 'Manager Application HOW-TO', and 'Clustering/Session Replication HOW-TO'. A 'Developer Quick Start' section provides links to 'Tomcat Setup', 'First Web Application', 'Realms & AAA', 'JDBC DataSources', 'Examples', 'Servlet Specifications', and 'Tomcat Versions'. The main content area is divided into three columns: 'Managing Tomcat' (with links to Release Notes, Changelog, Migration Guide, and Security Notices), 'Documentation' (with links to Tomcat 8.0 Documentation, Tomcat 8.0 Configuration, Tomcat Wiki, and various configuration files), and 'Getting Help' (with links to FAQ and Mailing Lists, including tomcat-announce, tomcat-users, taglibs-user, and tomcat-dev). The bottom section contains 'Other Downloads' (Tomcat Connectors, Tomcat Native, Taglibs, Deployer), 'Other Documentation' (Tomcat Connectors mod\_jk Documentation, Tomcat Native, Deployer), 'Get Involved' (Overview, SVN Repositories, Mailing Lists, Wiki), 'Miscellaneous' (Contact, Legal, Sponsorship, Thanks), and 'Apache Software Foundation' (Who We Are, Heritage, Apache Home, Resources).

The bottom screenshot shows a sample 'Hello, World' application page. It features the Tomcat logo and a title 'Sample "Hello, World" Application'. The text explains that this is the home page for a sample application used to illustrate the source directory organization of a web application. It provides two links to prove that they work: 'To a JSP page.' and 'To a servlet.'

### 3. Travailler avec plusieurs conteneurs

- Ouvrir une instance dans le PLAYGROUND DOCKER Récupérer les images

```
$ docker image pull redis:alpine
$ docker image pull russmckendrick/moby-counter
```

- Créer le réseau & Analyser les différences l'hôte

```
$ ip a | tee /tmp/netinf.old
$ docker network create net-moby
$ ip a | tee /tmp/netint.new
$ diff /tmp/netinf.old /tmp/netint.new
```

La commande **tee** lit à partir de l'entrée standard et écrit à la fois dans le fichier spécifié et dans la sortie standard

```
[node1] (local) root@192.168.0.8 ~
$ ip a | tee /tmp/netinf.old
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    link/ether 02:42:83:39:c6:6f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
217697: eth0@if217698: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 6a:7b:f6:d5:d0:38 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.8/23 scope global eth0
        valid_lft forever preferred_lft forever
217701: eth1@if217702: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:12:00:70 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.112/16 scope global eth1
        valid_lft forever preferred_lft forever
```

```
[node1] (local) root@192.168.0.8 ~
$ ip a | tee /tmp/netint.new
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    link/ether 02:42:83:39:c6:6f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
3: br-198d02b3e063: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    link/ether 02:42:ec:d6:d0:09 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-198d02b3e063
        valid_lft forever preferred_lft forever
217697: eth0@if217698: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 6a:7b:f6:d5:d0:38 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.8/23 scope global eth0
        valid_lft forever preferred_lft forever
217701: eth1@if217702: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:12:00:70 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.112/16 scope global eth1
        valid_lft forever preferred_lft forever
```

```
[node1] (local) root@192.168.0.8 ~
$ diff /tmp/netinf.old /tmp/netint.new
--- /tmp/netinf.old
+++ /tmp/netint.new
@@ -6,6 +6,10 @@
     link/ether 02:42:83:39:c6:6f brd ff:ff:ff:ff:ff:ff
     inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
         valid_lft forever preferred_lft forever
+3: br-198d02b3e063: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
+  link/ether 02:42:ec:d6:d0:09 brd ff:ff:ff:ff:ff:ff
+  inet 172.19.0.1/16 brd 172.19.255.255 scope global br-198d02b3e063
+  valid_lft forever preferred_lft forever
 217697: eth0@if217698: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 6a:7b:f6:d5:d0:38 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.8/23 scope global eth0
```

- Lancer les conteneurs et les connecter au réseau

```
$ docker run -d --name redis --network net-moby redis:alpine
$ docker run -d --name moby01 --network net-moby -p 80:80
russmckendrick/moby-counter
```

- Compléter pour inspecter la configuration réseau `$ docker network ?`  
`docker network inspect net-moby`
- Compléter pour afficher et analyser `/etc/hosts`

```
$ cat /etc/hosts
127.0.0.1      localhost # IP du réseau local de nos conteneur

fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.0.8    node1 # IP de notre instance Docker
```

- Compléter pour afficher et analyser `/etc/resolv.conf`

```
$ cat /etc/resolv.conf
search 51ur3jppi0eupdptvsj42kdvgc.bx.internal.cloudapp.net
nameserver 127.0.0.11 #La commande nous permet de récupérer l'ip du DNS
options ndots:0
```

- Compléter pour tester la configuration de chaque conteneur `$ docker exec ?`

```
$ docker exec -it redis cat /etc/hosts
127.0.0.1      localhost
::1           localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.19.0.2     a9f548e3a542 # IP du conteneur 'redis'
```

- Vérifier la configuration, interroger le serveur DNS du réseau

```
$ docker exec moby01 nslookup redis 127.0.0.11

# Sortie
```

```
Server:      127.0.0.11
Address 1: 127.0.0.11

Name:        redis
Address 1: 172.19.0.2 redis.net-moby
```

## 4. Utiliser le stockage avec un conteneur

- Lancer un conteneur en partageant un dossier hôte

```
$ docker run -it -v /tmp/data:/data ubuntu /bin/bash
```

```
#Sortie
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
29202e855b20: Pull complete
Digest:
sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd060a74
Status: Downloaded newer image for ubuntu:latest
root@a68bc6eb1270:/#
```

- Créer un fichier depuis le terminal du conteneur

```
root@containerID:/# cd /data
root@containerID:/# touch testfile
root@containerID:/# exit
```

- Vérifier la création du fichier dans le dossier hôte

```
$ ls -hal /tmp/data
```

```
$ ls -hal /tmp/data
ls: -hal: No such file or directory
/tmp/data:
testfile
```

**Question** = Que se passe-t-il si un fichier partagé est modifié simultanément par un processus du conteneur et un processus de l'hôte ?

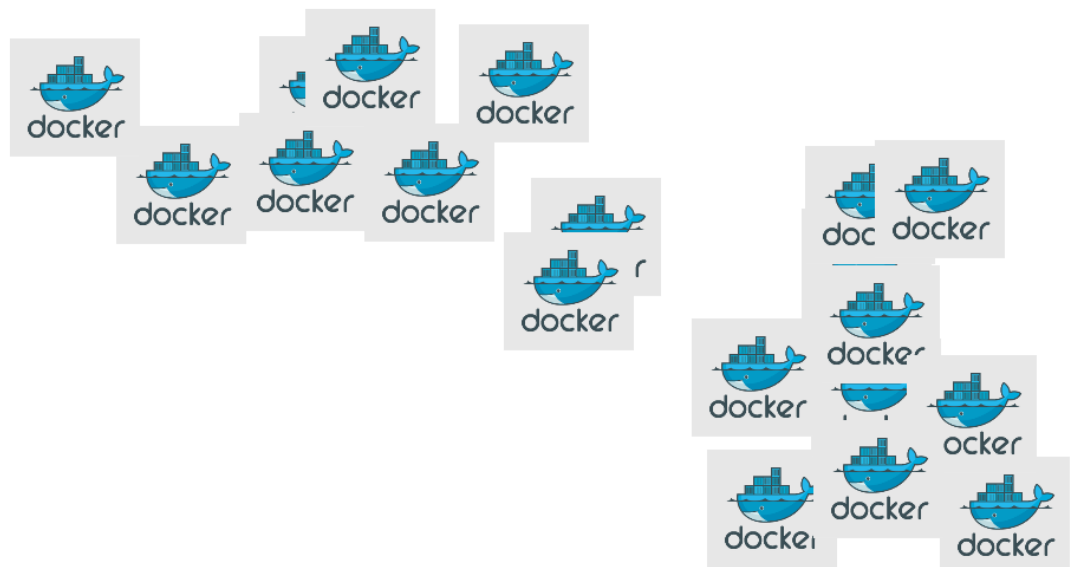
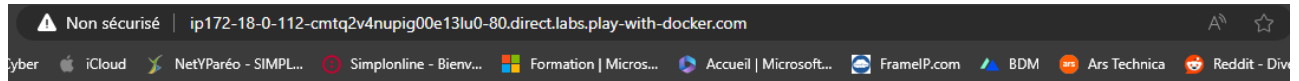
*Dans Docker, si un fichier partagé est modifié simultanément par un processus du conteneur et un processus de l'hôte, cela peut entraîner une condition de concurrence.*

*En général, le système de fichiers ne garantit pas quelles modifications seront préservées.*

- Démarrer une nouvelle application MOBY-COUNTER

```
$ docker network create net-moby
$ docker run -d --name redis --network net-moby redis:alpine
$ docker run -d --name moby01 --network net-moby -p 80:80
  russmckendrick/moby-counter
```

- Visiter l'application moby-counter et créer un beau dessin



- Supprimer le conteneur redis  
`$ docker stop redis && docker rm redis`
- Visiter l'application ainsi déconnecter du backend  
*L'application ne fonctionne plus*
- Créer un conteneur redis en attachant le volume « caché »  
`$ docker container run -d --name redis -v <volume_id>:/data --network net-moby redis:alpine`

```
$ docker volume ls -q
76b62abc64803215d77df8a01db77f4f998801eee990163ab7e5c15a816bf5d2
[node1] (local) root@192.168.0.8 ~
```

```
$ docker container run -d --name redis -v  
76b62abc64803215d77df8a01db77f4f998801eee990163ab7e5c15a816bf5d2:/data --  
network net-moby redis:alpine  
20ea5a20dcffec9053a5e6f9c1c4bc122460558c79ba6841a9170806a29a07d0
```

- Visiter l'application et retrouver votre beau dessin ... ou pas => et bah si !
- Compléter la commande et afficher le contenu du volume

\$ docker exec ?

```
$ docker exec redis ls /data  
dump.rdb
```