IIT INSTITUT INTERNATIONAL TECHNOLOGIE Université Nord Américaine privée	Matière : Virtualisation et cloud computing Travaux pratiques	**SE :** Ubuntu 18.04 LTS
	TP 3. Docker	**AU.** 2021-2022 Classe : 3ème GLSI

I. Objectifs

- Installer Docker
- Travailler avec des conteneurs et des images
- Transmettre une image dans un référentiel Docker

II. Mode opératoire

- Un serveur Ubuntu 18.04 LTS
- Un compte sur Docker Hub si vous souhaitez créer vos propres images et les transmettre au Docker Hub

III. Ressources

- Paramètres indiqués par le formateur

IV. Installation de Docker

Commencez par mettre à jour votre liste de paquets existante :

```
elyes@tunisie:~$ sudo apt-get update
```

1. Installation depuis les dépôts officiels :

```
iset@ubuntu:~$ sudo apt install docker.io
```

Vérifiez la version de Docker installée :

```
iset@ubuntu:~$ docker --version
Docker version 19.03.6, build 369ce74a3c
```

2. Installation à partir du référentiel officiel de Docker :

Pour installer la version la plus récente de Docker, nous utiliserons le référentiel officiel de Docker car le package d'installation de Docker disponible dans le référentiel officiel Ubuntu peut ne pas être la version la plus récente.

Pour ce faire, nous allons ajouter une nouvelle source de paquet. Pour garantir la validité des téléchargements, ajoutez la clé GPG de Docker, puis installez le paquet.

Tout d'abord, installez quelques paquets prérequis qui permettent à « apt » d'utiliser les paquets via HTTPS :

```
elyes@tunisie:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Ensuite, ajoutez ensuite la clé GPG du référentiel Docker officiel à votre système :

```
elyes@tunisie:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
```

Maintenant, ajoutez le référentiel Docker aux sources APT :

```
elyes@tunisie:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bion
ic stable"
```

Ensuite, mettez à jour la base de données de paquets avec les paquets Docker du référentiel que vous avez ajouté :

```
elyes@tunisie:~$ sudo apt update
```

Pour s'assurer que vous êtes sur le point d'installer à partir du référentiel Docker au lieu du référentiel par défaut Ubuntu, tapez :

```
elyes@tunisie:~$ apt-cache policy docker-ce
```

Vous verrez une sortie de données comme celle-ci, même si le numéro de version de Docker peut être différent :

```
iset@ubuntu:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:19.03.8~3-0~ubuntu-bionic
  Version table:
     5:19.03.8~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
```

Vous voyez que docker-ce n'est pas encore installé, mais que le candidat à l'installation provient du référentiel Docker pour Ubuntu 18.04 (bionic). Installez alors Docker :

```
elyes@tunisie:~$ sudo apt install docker-ce
```

Vérifiez maintenant la version de Docker installée :

```
iset@ubuntu:~$ docker --version
Docker version 19.03.8, build afacb8b7f0
```

→ Vous voyez que cette version est plus récente que celle installée à partir des dépôts officiels d'Ubuntu.

Vérifiez que Docker est en cours d'exécution :

```
elyes@tunisie:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2019-10-12 11:03:20 PDT; 11s ago
     Docs: https://docs.docker.com
 Main PID: 7999 (dockerd)
    Tasks: 13
   CGroup: /system.slice/docker.service
           └─7999 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Oct 12 11:03:18 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:18.735344270-07:00" level=warning msg="You
Oct 12 11:03:18 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:18.735390852-07:00" level=warning msg="You
Oct 12 11:03:18 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:18.735406671-07:00" level=warning msg="You
Oct 12 11:03:18 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:18.735762917-07:00" level=info msg="Loadin
Oct 12 11:03:19 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:19.299450766-07:00" level=info msg="Defaul
Oct 12 11:03:20 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:19.925846933-07:00" level=info msg="Loadin
Oct 12 11:03:20 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:20.464145942-07:00" level=info msg="Docker
Oct 12 11:03:20 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:20.478556928-07:00" level=info msg="Daemon
Oct 12 11:03:20 ult-tunisie systemd[1]: Started Docker Application Container Engine.
Oct 12 11:03:20 ult-tunisie dockerd[7999]: time="2019-10-12T11:03:20.585582842-07:00" level=info msg="API li
lines 1-19/19 (END)
```

Ajoutez votre nom d'utilisateur au groupe docker pour éviter de taper « sudo » chaque fois que vous exécutez la commande docker car cette commande ne peut être exécutée que par l'utilisateur root ou par un utilisateur du groupe docker, créé automatiquement lors du processus d'installation de Docker :

```
elyes@tunisie:~$ sudo usermod -aG docker ${USER}
```

Pour appliquer la nouvelle appartenance à un groupe et éviter la déconnexion et la reconnexion au serveur, tapez ce qui suit :

```
elyes@tunisie:~$ su - ${USER}
```

Confirmez que votre utilisateur est maintenant ajouté au groupe docker en tapant :

```
elyes@tunisie:~$ id -nG
elyes sudo docker
```

V. Utilisation de la commande Docker :

Utiliser docker consiste à lui transmettre une chaîne d'options et de commandes suivie d'arguments. La syntaxe prend cette forme :

```
Usage:  docker [OPTIONS] COMMAND
```

La commande « docker » permet d'afficher toutes les sous-commandes disponibles :

```
elyes@tunisie:~$ docker
```

```
Management Commands:
  builder     Manage builds
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  engine      Manage the docker engine
  image       Manage images
  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
```

Pour afficher les options disponibles pour une commande spécifique, tapez :

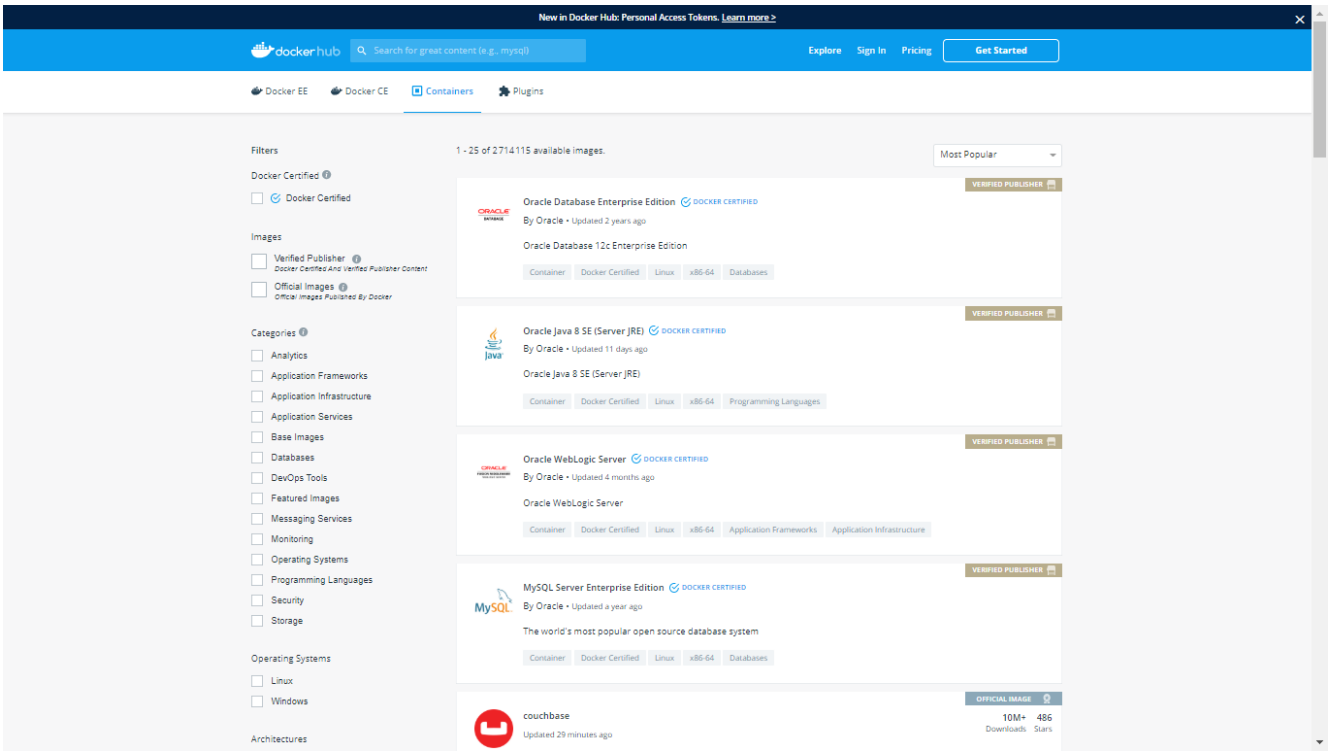
```
docker COMMAND --help
```

Pour afficher des informations sur Docker, tapez :

```
elyes@tunisie:~$ docker info
```

VI. Utilisation des images Docker

Les conteneurs Docker sont construits à partir d'images Docker. Par défaut, Docker extrait ces images de Docker Hub, un registre Docker géré par Docker, la société à l'origine du projet Docker. Accédez à cette adresse pour consulter les images disponibles : https://hub.docker.com/.



Tout le monde peut héberger ses images Docker sur Docker Hub. Ainsi, la plupart des applications et des distributions Linux dont vous aurez besoin auront des images hébergées sur cet espace.

Pour vérifier si vous pouvez accéder aux images et les télécharger à partir de Docker Hub, tapez :

```
elyes@tunisie:~$ docker run hello-world
```

La sortie de données indiquera que Docker fonctionne correctement :

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

Docker a téléchargé l'image à partir de Docker Hub (le référentiel par défaut) car il n'a pas pu trouver initialement l'image hello-world localement. Une fois l'image téléchargée, Docker a créé un conteneur à partir de l'image et de l'application exécutée dans le conteneur. C'est un exemple de conteneur qui s'exécute et se ferme après avoir émis un message de test.

Pour rechercher des images disponibles sur Docker Hub vous pouvez utiliser la commande docker avec la sous-commande search. Par exemple, pour rechercher l'image Ubuntu, tapez :

```
elyes@tunisie:~$ docker search ubuntu
```

Le script analysera Docker Hub et renverra une liste de toutes les images dont le nom correspond à la chaîne de recherche. Dans la colonne OFFICIAL, OK indique une image construite et prise en charge par la société derrière le projet.

```
NAME                                                DESCRIPTION                                     STARS               OFFICIAL
    AUTOMATED
ubuntu                                              Ubuntu is a Debian-based Linux operating sys…   10038               [OK]

dorowu/ubuntu-desktop-lxde-vnc                      Docker image to provide HTML5 VNC interface …   350
    [OK]
rastasheep/ubuntu-sshd                              Dockerized SSH service, built on top of offi…   231
    [OK]
consol/ubuntu-xfce-vnc                              Ubuntu container with "headless" VNC session…   188
    [OK]
ubuntu-upstart                                      Upstart is an event-based replacement for th…   100                 [OK]

ansible/ubuntu14.04-ansible                         Ubuntu 14.04 LTS with ansible                   98
    [OK]
neurodebian                                         NeuroDebian provides neuroscience research s…   59                  [OK]

1and1internet/ubuntu-16-nginx-php-phpmyadmin-mysql-5   ubuntu-16-nginx-php-phpmyadmin-mysql-5       50
    [OK]
ubuntu-debootstrap                                  debootstrap --variant=minbase --components=m…   40                  [OK]

nuagebec/ubuntu                                     Simple always updated Ubuntu docker images w…   24
    [OK]
i386/ubuntu                                         Ubuntu is a Debian-based Linux operating sys…   18
```

La commande pull permet de télécharger sur votre ordinateur l'image que vous souhaitez utiliser. Exécutez la commande suivante pour télécharger l'image officielle Ubuntu sur votre ordinateur :

```
elyes@tunisie:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5667fdb72017: Pull complete
d83811f270d5: Pull complete
ee671aafb583: Pull complete
7fc152dfb3a6: Pull complete
Digest: sha256:b88f8848e9a1a4e4558ba7cfc4acc5879e1d0e7ac06401409062ad2627e6fb58
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

La sous-commande « run » permet d'exécuter un conteneur en utilisant l'image téléchargée. Comme vous l'avez vu avec l'exemple hello-world, si une image n'a pas été téléchargée lorsque docker est exécuté avec la sous-commande run, le client Docker télécharge d'abord l'image, puis exécute un conteneur en l'utilisant.

Pour voir les images téléchargées sur votre ordinateur, tapez :

```
elyes@tunisie:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              2ca708c1c9cc        3 weeks ago         64.2MB
hello-world         latest              fce289e99eb9        9 months ago        1.84kB
```

Les images que vous utilisez pour exécuter des conteneurs peuvent être modifiées et utilisées pour générer de nouvelles images, qui peuvent ensuite être transmises (pushed est le terme technique) vers Docker Hub ou d'autres registres Docker.

VII. Exécuter un conteneur Docker

Les conteneurs ressemblent aux machines virtuelles, mais ils sont plus conviviaux. Exécutez comme exemple un conteneur en utilisant la dernière image d'Ubuntu. La combinaison des options -i et -t vous donne un accès interactif au Shell dans le conteneur :

```
iset@ubuntu:~$ docker run -it --name ubuntu-nginx ubuntu
root@98bd6d05c235:/#
```

Vous travaillez maintenant dans le conteneur. Notez l'ID de conteneur dans l'invite de commande. Dans cet exemple, il s'agit de « 98bd6d05c235 ».

Vous pouvez maintenant exécuter n'importe quelle commande dans le conteneur. Par exemple, mettons à jour la base de données de paquets à l'intérieur du conteneur.

```
# Sudo apt update
```

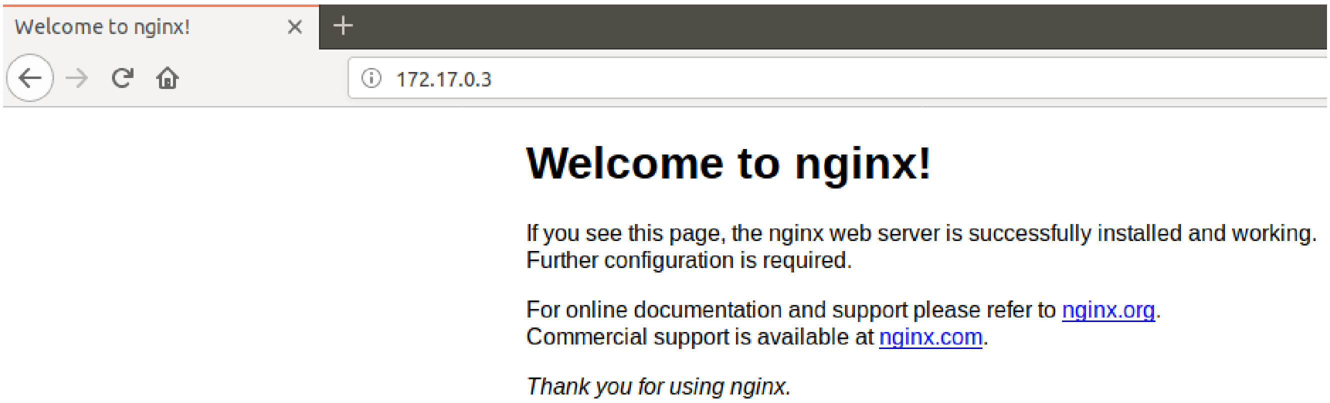
Installez le serveur web nginx :

```
root@98bd6d05c235:/# apt install nginx
```

Notez l'adresse IP du conteneur et démarrer nginx.

```
root@98bd6d05c235:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
root@98bd6d05c235:/# service nginx start
 * Starting nginx nginx                                                       [ OK ]
root@98bd6d05c235:/# ss -anltp
State      Recv-Q      Send-Q            Local Address:Port          Peer Address:Port
LISTEN     0           128                     0.0.0.0:80                 0.0.0.0:*          users:(("nginx",pid=785,fd=6))
LISTEN     0           128                        [::]:80                    [::]:*          users:(("nginx",pid=785,fd=7))
root@98bd6d05c235:/#
```

Mettez cette adresse dans votre navigateur web et vérifier que nginx fonctionne corectement.



Vous pouvez quitter ce conteneur, le démarrer et exécuter le Shell Bash à nouveau sur ce conteneur :

```
root@98bd6d05c235:/# exit
exit
iset@ubuntu:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
98bd6d05c235        ubuntu              "/bin/bash"         10 minutes ago      Exited (0) 8 seconds ago                        ubuntu-ngin
x
dd67641342d7        hello-world         "/hello"            About an hour ago   Exited (0) About an hour ago                    sharp_pike
iset@ubuntu:~$ docker start ubuntu-nginx
ubuntu-nginx
iset@ubuntu:~$ docker exec ubuntu-nginx /bin/bash
iset@ubuntu:~$ docker exec -it ubuntu-nginx /bin/bash
root@98bd6d05c235:/#
```

VIII. Publiez votre première image Docker sur Docker Hub

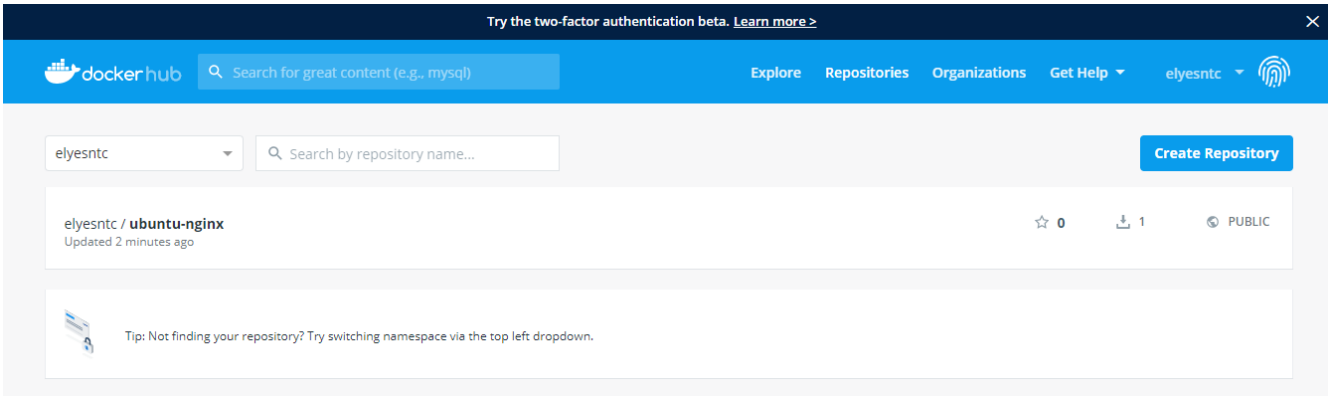
Il peut être utile de valider les modifications ou les paramètres d'un fichier de conteneur dans une nouvelle image. Ensuite vous pouvez déposer cette image

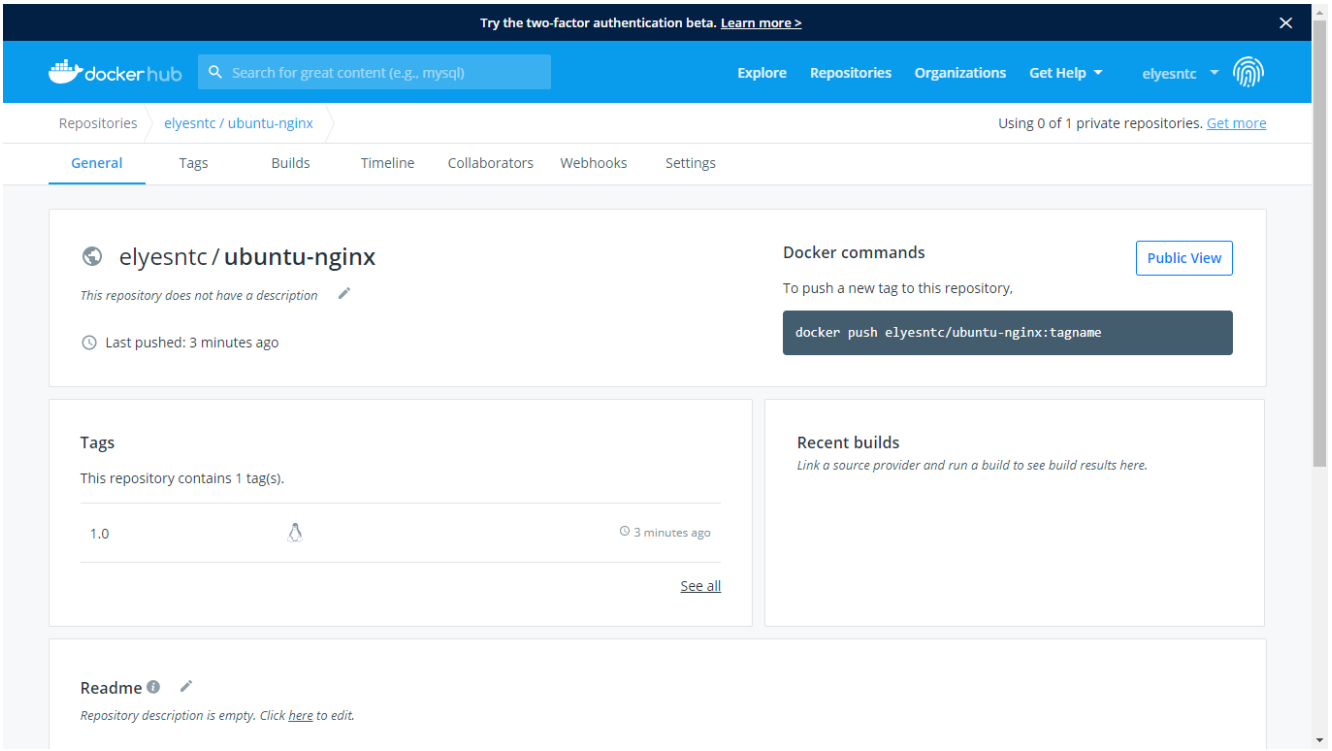
```
root@98bd6d05c235:/# exit
exit
iset@ubuntu:~$ docker commit ubuntu-nginx elyesntc/ubuntu-nginx:1.0
sha256:82a09d0a5b39f71a220f8f87bf4f8961f8c9821bf151d9cdb202503d781c5351
iset@ubuntu:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to creat
e one.
Username: elyesntc
Password:
WARNING! Your password will be stored unencrypted in /home/iset/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
iset@ubuntu:~$
```

Publier l'image : Téléchargez votre image balisée dans le référentiel : Une fois terminé, les résultats de ce téléchargement sont accessibles au public. Si vous vous connectez à Docker Hub, vous y verrez la nouvelle image, avec sa commande pull.

```
iset@ubuntu:~$ docker push elyesntc/ubuntu-nginx:1.0
The push refers to repository [docker.io/elyesntc/ubuntu-nginx]
e3d47dc27759: Pushed
16542a8fc3be: Mounted from elyesntc/ubuntu_nginx
6597da2e2e52: Mounted from elyesntc/ubuntu_nginx
977183d4e999: Mounted from elyesntc/ubuntu_nginx
c8be1b8f4d60: Mounted from elyesntc/ubuntu_nginx
1.0: digest: sha256:2163adb7f02a8855625478ef0a7dc5a5b2ad5d8ce75eb92db4e2166eaaa76a98 size: 1364
iset@ubuntu:~$
```



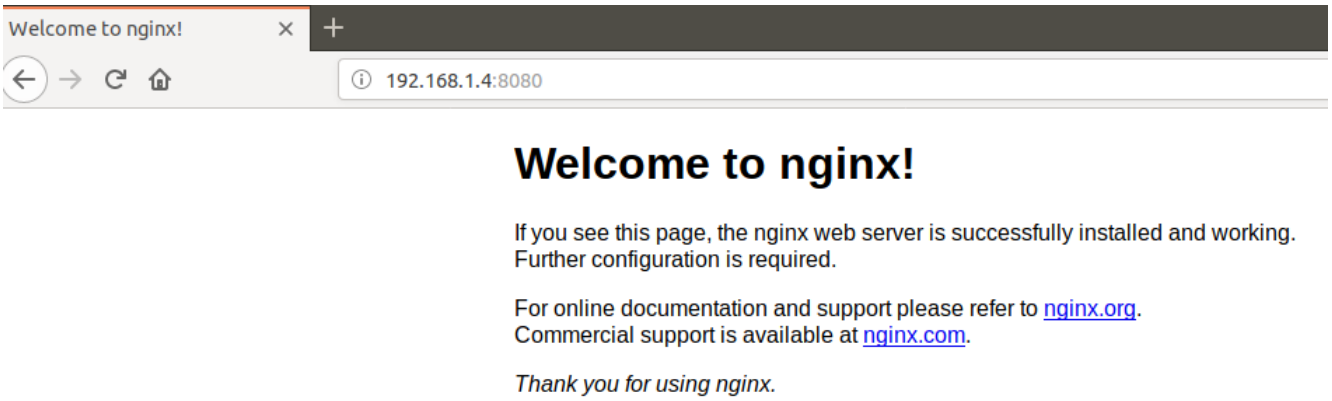


Remarque : Le référentiel Docker Hub contient déjà une image nginx prête, vous pouvez alors l'utiliser directement.

```
iset@ubuntu:~$ docker run -itd --name nginx-test -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
c499e6d256d6: Pull complete
74cda408e262: Pull complete
ffadbd415ab7: Pull complete
Digest: sha256:282530fcb7cd19f3848c7b611043f82ae4be3781cb00105a1d593d7e6286b596
Status: Downloaded newer image for nginx:latest
4133082c2d9db1cb2181b982a8270e9cc1aaba477751f9a328130ad606f32217
```

Repérer l'adresse IP de la machine hôte et tester le mappage de port vers le conteneur.

```
iset@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:c3:36:c0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.4/24 brd 192.168.1.255 scope global dynamic noprefixroute ens33
       valid_lft 75329sec preferred_lft 75329sec
    inet6 fe80::1a9d:94b8:cf03:565f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:dd:cb:e5:0f 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
    inet6 fe80::42:ddff:fecb:e50f/64 scope link
       valid_lft forever preferred_lft forever
9: veth5cba21a@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 62:91:a7:06:7a:95 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::6091:a7ff:fe06:7a95/64 scope link
       valid_lft forever preferred_lft forever
13: veth921450f@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether d2:c0:bd:bc:50:3e brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::d0c0:bdff:febc:503e/64 scope link
       valid_lft forever preferred_lft forever
17: veth588cdbf@if16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether ae:31:73:84:07:0d brd ff:ff:ff:ff:ff:ff link-netnsid 2
    inet6 fe80::ac31:73ff:fe84:70d/64 scope link
       valid_lft forever preferred_lft forever
```



IX. Gestion du réseau sous Docker

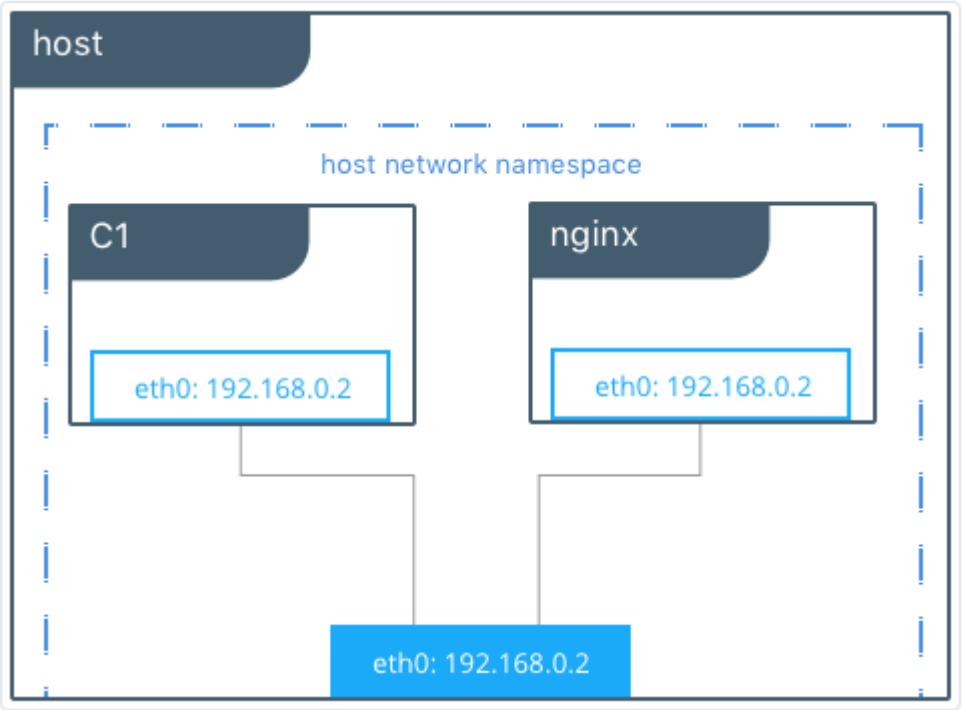
VIII. 1) Les types des réseaux

Lors de l'installation de Docker, trois réseaux sont créés automatiquement. On peut voir ces réseaux avec la commande **docker network ls**.

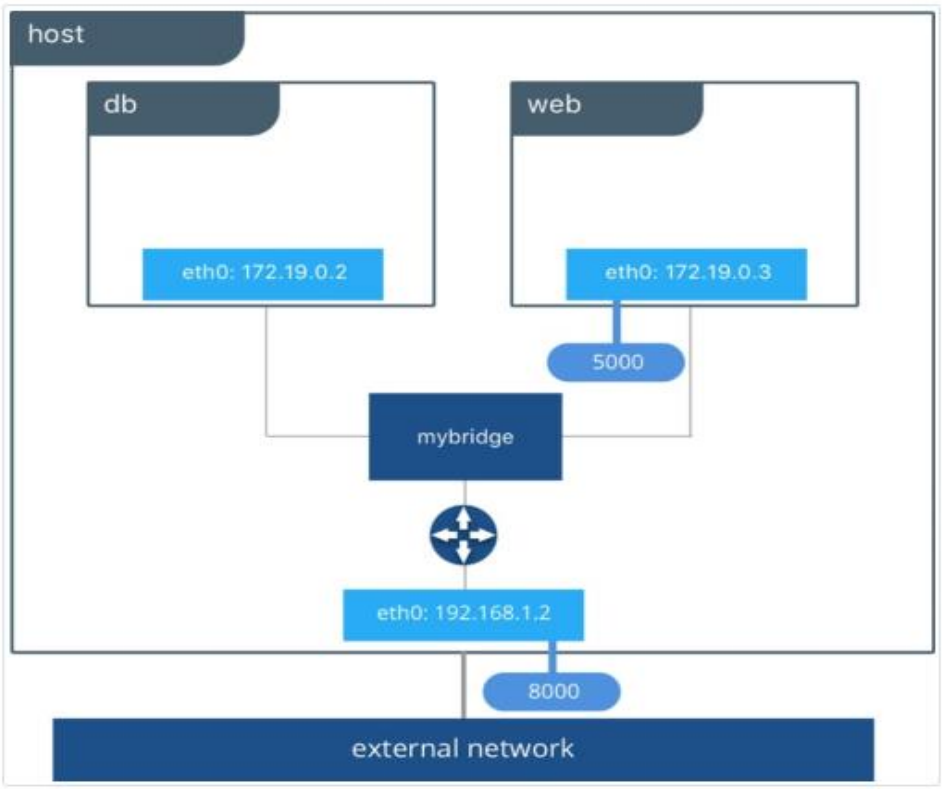
```
elyes@tunisie:~$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ba162b953b30        bridge              bridge              local
2eabbad40474        host                host                local
d2bbbc081f4e        none                null                local
```

Le driver none : C'est le type de réseau idéal, si vous souhaitez interdire toute communication interne et externe avec votre conteneur, car votre conteneur sera dépourvu de toute interface réseau (sauf l'interface loopback).

Le driver host : Ce type de réseau permet aux conteneurs d'utiliser la même interface que l'hôte. Il supprime donc l'isolation réseau entre les conteneurs et seront par défaut accessibles de l'extérieur. De ce fait, il prendra la même IP que votre machine hôte.



Le driver Bridge : Un réseau de type bridge est créé : Le réseau Bridge est présent sur tous les hôtes Docker. Par ailleurs, le réseau bridge est le type de réseau le plus couramment utilisé. Il est limité aux conteneurs d'un hôte unique exécutant le moteur Docker. Les conteneurs qui utilisent ce driver, ne peuvent communiquer qu'entre eux, cependant ils ne sont pas accessibles depuis l'extérieur.



Lors de la création d'un conteneur, si l'on ne spécifie pas un réseau particulier, les conteneurs sont connectés au Bridge docker0.

```
elyes@tunisie:~$ ip addr show docker0
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOW
N group default
    link/ether 02:42:16:17:9a:cd 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
    inet6 fe80::42:16ff:fe17:9acd/64 scope link
       valid_lft forever preferred_lft forever
```

La commande **docker network inspect bridge**, retourne les informations concernant ce réseau :

```
elyes@tunisie:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "ba162b953b3027c25f1f2aa813ba24dbca0d73e44a7cd8508f6c24fe03f5b87f",
        "Created": "2019-10-15T11:56:35.41094278-07:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Créez deux containers avec l'image ubuntu.

```
elyes@tunisie:~$ docker run -itd --name=container1 ubuntu
cfe4abc62c9eada7f606667375568635653aa665f50ffbd75ad5dedcb20955f0
elyes@tunisie:~$ docker run -itd --name=container2 ubuntu
7a760a81794c5115cfe41369d9231d76f8fab7812690176e0d69a4e2c690ce05
```

Et visualisez les informations du réseau avec **docker network inspect bridge** :

```
"Containers": {
    "7a760a81794c5115cfe41369d9231d76f8fab7812690176e0d69a4e2c690ce05": {
        "Name": "container2",
        "EndpointID": "09bb25aba957758ff007314500 7ff8a83d640a873a9725c8780dc967303786db",
        "MacAddress": "02:42:ac:11:00:03",
        "IPv4Address": "172.17.0.3/16",
        "IPv6Address": ""
    },
    "cfe4abc62c9eada7f6066673755686356 53aa665f50ffbd75ad5dedcb20955f0": {
        "Name": "container1",
        "EndpointID": "968f94b415c23c093a9b2732fa8a68915cc53ebf07db847b32514c537d3ece19",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
    }
},
"Options": {
    "com.docker.network.bridge.default_bridge": "true",
    "com.docker.network.bridge.enable_icc": "true",
    "com.docker.network.bridge.enable_ip_masquerade": "true",
    "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
    "com.docker.network.bridge.name": "docker0",
    "com.docker.network.driver.mtu": "1500"
},
"Labels": {}
```


Notez les adresses des deux containers et effectuez un Ping sur chacune d'elle :

```
elyes@tunisie:~$ ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.070 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.100 ms
^C
--- 172.17.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1018ms
rtt min/avg/max/mdev = 0.070/0.085/0.100/0.015 ms
elyes@tunisie:~$ ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.068 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.095 ms
^C
--- 172.17.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1013ms
rtt min/avg/max/mdev = 0.068/0.081/0.095/0.016 ms
```

X. Docker sous Windows

- **Docker Toolbox** est destiné aux anciens systèmes Mac et Windows qui ne répondent pas aux exigences de Docker Desktop pour Mac et Docker Desktop pour Windows. Étant donné que le démon Docker Engine utilise des fonctionnalités de noyau spécifiques à Linux, vous ne pouvez pas exécuter Docker Engine en mode natif sous Windows. Au lieu de cela, vous devez utiliser la commande docker-machine, pour créer et attacher à une petite machine virtuelle Linux sur votre machine. Cette machine virtuelle héberge Docker Engine pour vous sur votre système Windows. Docker Toolbox comprend les outils Docker suivants :

  ✓ Client Docker CLI pour exécuter Docker Engine pour créer des images et des conteneurs

  ✓ Docker Machine pour que vous puissiez exécuter les commandes Docker Engine à partir des terminaux Windows

  ✓ Docker Compose pour exécuter la commande docker-compose

- ✓ Kitematic, l'interface graphique Docker
- ✓ Le shell Docker QuickStart préconfiguré pour un environnement de ligne de commande Docker
- ✓ Oracle VM VirtualBox

- **Docker Desktop** est le meilleur moyen de démarrer avec Docker sous Windows. L'image du conteneur se compose des fichiers du système d'exploitation en mode utilisateur nécessaires pour prendre en charge votre application, votre application, les exécutions ou les dépendances de votre application et tout autre fichier de configuration divers dont votre application a besoin pour fonctionner correctement. Microsoft propose plusieurs images (appelées images de base) que vous pouvez utiliser comme point de départ pour créer votre propre image de conteneur :
  - ✓ Windows - contient l'ensemble complet des API Windows et des services système (moins les rôles de serveur).
  - ✓ Windows Server Core - une image plus petite qui contient un sous-ensemble des API Windows Server, à savoir le framework .NET complet. Il inclut également la plupart des rôles de serveur, bien que malheureusement peu nombreux, pas le serveur de télécopie.
  - ✓ Nano Server - la plus petite image Windows Server, avec prise en charge des API .NET Core et de certains rôles de serveur.
  - ✓ Windows 10 IoT Core - une version de Windows utilisée par les fabricants de matériel pour les petits appareils Internet of Things qui exécutent des processeurs ARM ou x86 / x64.

Bibliographie

https://docs.docker.com/docker-for-windows/
https://www.digitalocean.com/community/tutorials/comment-installer-et-utiliser-docker-sur-ubuntu-18-04-fr
https://blog.alphorm.com/reseau-docker-partie-1-bridge/
https://www.journaldunet.fr/web-tech/developpement/1441133-docker-sur-ubuntu-resoudre-l-erreur-bash-ping-command-not-found/
https://devopssec.fr/article/fonctionnement-manipulation-reseau-docker