Docker

Installer Docker CE

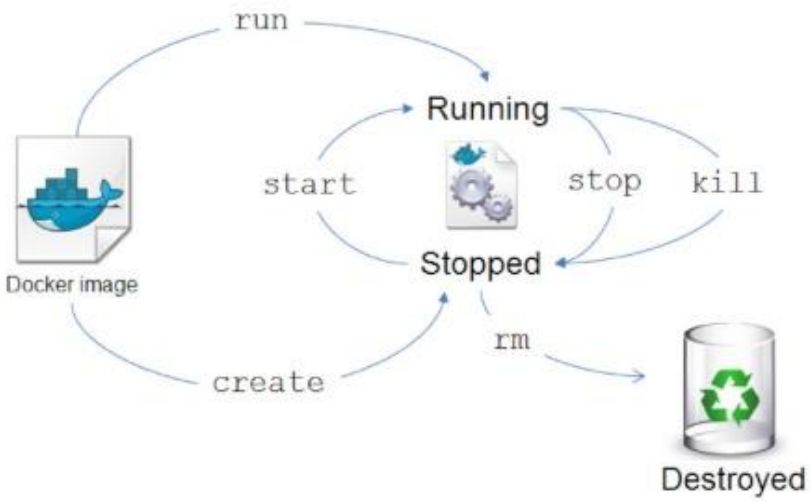
DOCKER propose un programme pour démarrer les conteneurs de façon simple, rapide et gratuit. C'est Docker Community Edition. Pour l'installer sur LINUX, suivez les 3 premières étapes de la doc officielle :
https://doc.ubuntu-fr.org/docker

Pour stopper DOCKER, faite la commande suivante :

DOCKER en place, il faut maintenant pouvoir créer des conteneurs. Pour ça deux options :

- Soit à partir d'une image,
- Soit à partir du Dockerfile.

Cycle de vie d'un conteneur



Le conteneur connait différents états, suivant les commandes.
Un conteneur peut être créé et démarré immédiatement (commande run) ou créé dans l'attente d'être démarré (commande create).

Créer et démarrer un conteneur

Pour ce premier exercice, vous allez créer un conteneur à partir d'une image Nginx. Il s'agit d'un conteneur exécutant le serveur web open source Nginx. Notez que cette image est un official repository du *Docker Hub*, c'est-à-dire une image certifiée par l'entreprise Docker Inc.

Pensez à lancer votre Docker CE, si ce n'est déjà fait :

```
$ sudo service docker start
```

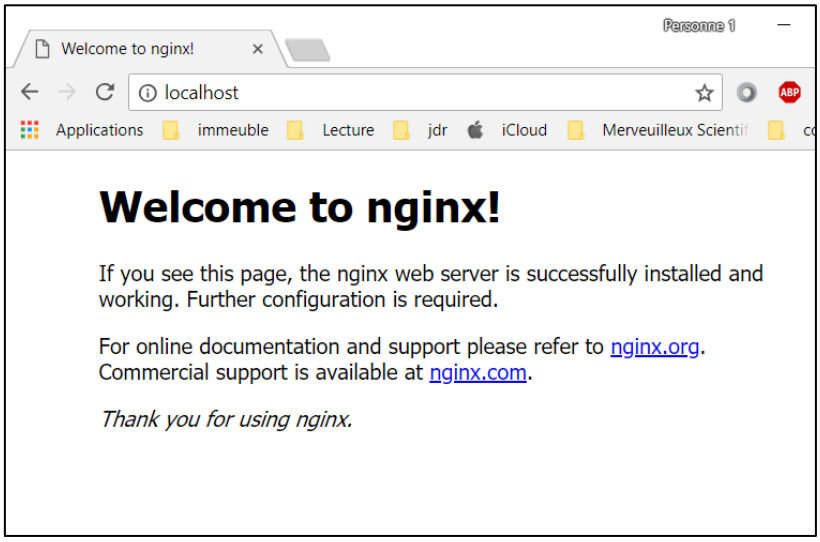
Saisissez la ligne de commande suivante :

```
$ docker run –p 80:80 –d nginx
```

Cette commande spécifie que le port **80** du serveur web, ouvert à l'intérieur du conteneur, doit être exposé sur le port **80** de l'hôte. **-d** est un paramètre pour demander à lancer le conteneur en détaché, c'est-à-dire indépendamment du terminal, pour continuer à l'utiliser. C'est pratique lorsqu'il y a plusieurs conteneurs à lancer. Le paramètre **Nginx** est le nom de l'image qui va être utilisée pour instancier le conteneur.

Si cette commande est exécutée pour la première fois, elle va entraîner le téléchargement de l'image Docker officielle *Nginx* qui va servir d'architecture pour la création de notre conteneur. Cette phase de téléchargement ne sera plus nécessaire par la suite. L'image est désormais stockée dans le registry local de l'hôte.

Pour tester le conteneur, lancez un navigateur et fait appel à l'URL http://localhost pour accéder à la home page de Nginx.



Stopper un conteneur

Pour stopper un conteneur, il est nécessaire de connaitre son identifiant : $ `docker container ls`

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS                    NAMES
da22b4157d53        nginx               "nginx -g 'daemon of…"   40 seconds ago      Up 36 seconds               0.0.0.0:8000->80/tcp     dreamy_babbage
ed46507c01dd        hello-world         "/hello"                 30 hours ago        Exited (0) 30 hours ago                              fervent_saha
```

Cette commande liste les conteneurs en cours d'exécution et retourne certaines informations dont le *container id* et le *name*.

Ensuite, vous avez le choix entre la commande **kill** et la commande **stop**. La première commande va directement tuer le processus principal qui s'exécute dans le conteneur. La seconde va d'abord arrêter le processus et, si ce n'est pas possible, finir par le tuer au bout d'un délais donné. Le stop est donc plus souvent utilisé, avec en paramètre le *container id* du conteneur à arrêter :

```
$ docker stop 7b7fa70a14c2
```

Vous pouvez aussi le faire avec le *name* du docker à arrêter :

```
$ docker stop dreamy_babbage
```

Le conteneur est arrêté et le navigateur ne trouvera plus la page d'accueil à l'URL http://localhost.

Redémarrer un conteneur

Le conteneur stoppé existe toujours ou, plus exactement, l'image complémentée de la couche persistante read/ write propre aux conteneurs est toujours stockée sur le système.

Nous avons vu que la commande **ps** retourne, par défaut, que les conteneurs actifs. Avec le paramètre **–a**, pour *All*, celle-ci va retourner aussi les conteneurs stoppés :

```
$ docker container ls –a
```

Pour redémarrer notre conteneur, il suffit de passer par la commande **start**, avec en paramètre, le *container id* ou du *name* du conteneur stoppé :

```
$ docker start dreamy_babbage
```

Détruire un conteneur

Pour détruire un conteneur, il faut d'abord le stopper. Ensuite la commande **rm** fait le travail :

```
$ docker rm dreamy_babbage
```

Créer un conteneur

Nous avons vu que la commande **run** permet de créer un conteneur et de le démarrer. **create** ne fait que créer le conteneur, il faut alors le démarrer après :

```
$ docker create –p 80:80 --name webserver nginx
$ docker start webserver
```

Le paramètre **--name** permet de forcer le nom du conteneur

Modifier un conteneur à partir du terminal

Pour le moment, nous avons utilisé une image générique prise sur *Docker Hub,* sans la modifier. En pratique, lancer un conteneur sans le modifier n'a aucun intérêt. Nous allons reprendre notre image *Nginx* en modifiant les pages retournées par le serveur web.

```
$ docker run –d –p 80:80 --name webserver nginx
```

Faites attention, nous créons un nouveau conteneur qui va solliciter le port **80**. Il faut donc que vos précédents conteneurs soient bien détruits.

La page d'accueil qui s'affiche par défaut dans le navigateur se trouve (au sein du conteneur) dans le répertoire */usr/share/nginx/html*. Pour faire nos modifications nous allons ouvrir un terminal à l'intérieur du conteneur, grâce à la commande suivante :

```
$ docker exec –t –i webserver /bin/bash
```

Pour confirmer que nous sommes bien à l'intérieur du conteneur, nous allons modifier la home page du serveur web. La commande suivante écrase le contenu du fichier HTML par défaut avec une simple chaîne de caractères « *Je suis dedans* ».

```
# echo “Je suis dedans” > /usr/share/nginx/html/index.html
```

Créer un volume

Les volumes sont le moyen qu'offre Docker pour gérer la persistance des données au sein des conteneurs (et en relation avec leur machine hôte). Comme nous l'avons vu précédemment, la page d'accueil du serveur web Nginx se trouve dans le répertoire */usr/ share/ nginx/ html/ index.html*.

Définissons un volume correspondant à ce chemin à l'aide du paramètre **-v** de la commande **run**, vers un dossier existant sur notre machine (par exemple */webserver_local*).

```
$ docker run –p 80:80 --name webserver –d –v
$HOME/webserver_local:/usr/share/nginx/html nginx
```

La commande **inspect** permet d'obtenir une structure JSON décrivant le conteneur et sa configuration : $ `docker inspect webserver`

Trouvez le document *Mounts* dans la description de votre conteneur. Vous pouvez y lire les informations suivantes :

- **Name** : identifiant unique du document,
- **Source** : le chemin sur le système de fichiers de la machine hôte contenant les données correspondant au volume.
- **Destination** : chemin à l'intérieur du conteneur.
- **RW** : indique que le volume est read-write.

Ajoutez une page *index.html* dans votre dossier local et lancer le navigateur en localhost pour voir cette page s'afficher.

Images Docker

Pour lister les images présentent sur la machine hôte, lancer la commande :

```
$ docker images
```

Avec le paramètre **–a**, pour All, vous allez obtenir une liste plus longue :

```
$ docker images –a
```

Ces images sans nom sont ce que l'on appelle des images intermédiaires. Elles correspondent aux couches dont une image est constituée. Nous verrons plus tard comment ces images intermédiaires sont utilisées lors de la création d'une nouvelle image.

La commande **rmi**, suivi du nom de l'image ou de son identifiant unique, efface cette dernière :

```
$ docker rmi nginx
```

Attention, si un conteneur utilise cette image, vous obtiendrai un message d'erreur. Il faut détruire les conteneurs correspondant avant de supprimer une image.

Il est aussi possible de télécharger une image sans lancer de run sur un conteneur :

```
$ docker pull nginx:1.7
```

Dans cette commande, on spécifie en plus la version à télécharger (**:1.7**). Par défaut, le téléchargement se fait toujours sur la dernière version de l'image.

Vous pouvez vérifier cette version en montant un nouveau conteneur à partir de l'image (commande `docker run –p 8000 :80 ––name webserver2 –d nginx :1.7`, par exemple) et en ouvrant, dans Firefox, une page qui n'existe pas (par exemple : http://localhost:8000/page). L'erreur 404 vous retournera la version de nginx.

Créer une image à partir d'un conteneur

Voyons comme créer une image qui intègre les modifications réalisées sur un conteneur.

A partir de ce conteneur, nous pouvons créer une image en utilisant la commande commit :

```
$ docker commit webserver nginx2
```

‘*nginx2*’ est le nom de la nouvelle image à partir de laquelle nous pouvons maintenant créer de nouveaux conteneurs.