Activité : utiliser Docker

Pour cette découverte de Docker, vous pouvez réaliser au préalable avoir installé Docker.

Un premier conteneur

Lancement sans option d'un conteneur

Cette commande lance le classique Hello world ! :

```
btssio@ubuntudocker:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:92695bc579f31df7a63da6922075d0666e565ceccad16b59c3374d2cf4e8e50e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

btssio@ubuntudocker:~$
```

Ce qu'il faut retenir de l'exécution de cette commande :

- La première ligne indique que l'image du conteneur '**hello-world:latest**' n'est pas disponible **localement** c'est-à-dire sur le serveur Ubuntu. De plus, comme aucune version du conteneur n'est précisée, c'est la dernière version disponible, la **latest** qui est recherchée.
- L'image recherchée est alors **tirée (pull)** de Docker Hub depuis Internet.
- Une explication des étapes réalisées est indiquée : le client Docker, la session de terminal de votre serveur Ubuntu, contacte le **daemon** Docker du serveur Ubuntu (le service Docker) qui tire l'image du conteneur depuis Docker Hub, **crée un container, exécute** ce qui est prévu dans l'image du conteneur (la production du texte à afficher) et envoie cette sortie texte vers le client Docker votre terminal.
- Le conteneur est ensuite **arrêté automatiquement** car le traitement prévu s'est effectué, l'affichage du texte et rien d'autre.

Afficher les images Docker présentes sur la machine

```
btssio@ubuntudocker:~$ docker images
REPOSITORY    TAG     IMAGE ID       CREATED        SIZE
hello-world   latest  fce289e99eb9   4 months ago   1.84kB
btssio@ubuntudocker:~$
```

Vous pouvez vérifier la présence de l'image qui a été tirée (pull) de Docker Hub et stockée sur votre serveur. Docker gère un **cache** des images sur la machine. Lors du prochain démarrage d'une instance de conteneur sur la même image il n'y aura pas de nouveau téléchargement sauf si elle a été modifiée entre-temps.

INFORMATION
Pour en savoir plus sur une image il suffit de faire une recherche sur Docker Hub. Pour le conteneur hello-world : https://hub.docker.com/_/hello-world

Explication des tags

- Le **TAG** associé à l'image ubuntu est latest et est le marqueur de version de l'image.
- **IMAGE ID** est l'identifiant unique de l'image.
- **CREATED** est la date de création de l'image publiée sur le site Docker Hub.
- **SIZE** est sa taille virtuelle, c'est-à-dire de la couche logicielle téléchargée.

Lister les conteneurs actifs

L'image du conteneur hello-world est présente sur votre serveur mais après le lancement d'un conteneur basé sur cette image, vous avez retrouvé l'invite de commandes de votre serveur. Le conteneur lancé s'est ensuite arrêté automatiquement. Mais existe-il toujours ?

Pour constater que le conteneur lancé est bien arrêté vous pouvez visualiser les conteneurs actifs :

```
btssio@ubuntudocker:~$ docker ps
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
btssio@ubuntudocker:~$
```

Le conteneur hello-world n'est effectivement **plus actif**.

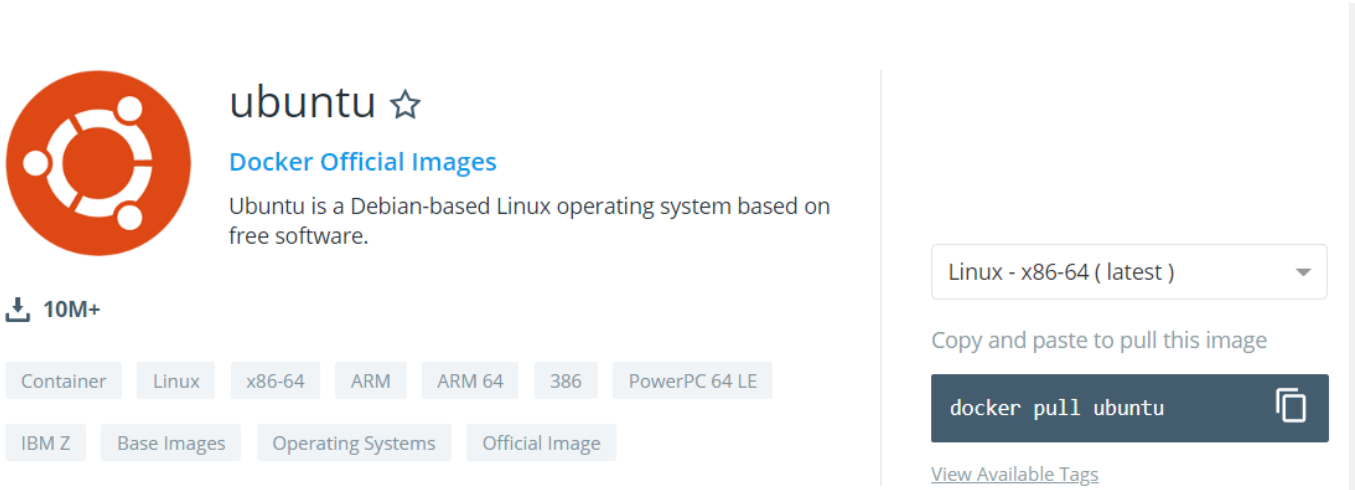
Lister tous les conteneurs y compris ceux qui sont inactifs

Relancez le conteneur hello-world. Vous pouvez à nouveau vérifier que celui n'est pas actif après l'affiche du texte prévu.

Visualisez maintenant tous les conteneurs créés, qu'ils soient actifs ou non :

```
btssio@ubuntudocker:~$ docker ps -a
CONTAINER ID IMAGE       COMMAND  CREATED    STATUS             PORTS NAMES
9bbd425c86f4 hello-world "/hello" 2 seconds  Exited (0) 1 s ago       vibrant_goldberg
ae50ecb3a66d hello-world "/hello" 49 seconds Exited (0) 47 s ago      quirky_elbakyan
btssio@ubuntudocker:~$
```

Les 2 conteneurs basés sur la **même image hello-world, sont bien présents mais arrêtés sans erreur normale (code de sortie 0) comme le montre le champ STATUS : Exited (0). Le premier conteneur s'est arrête il y a 1 seconde, le 2ème il y a 47 secondes. Chaque conteneur : * possède un identifiant de conteneur CONTAINER ID, * et un nom, NAMES, qui a été généré par de démon Docker. La colonne COMMAND nous renseigne sur le processus qui avait été lancé, à savoir un script hello. ===== Comprendre le système de couche logicielle des images Docker ===== ==== Récupérer une image dans sa dernière version ==== <code shell> btssio@ubuntudocker:~$ docker pull ubuntu Using default tag: latest latest: Pulling from library/ubuntu f476d66f5408: Pull complete 8882c27f669e: Pull complete d9af21273955: Pull complete f5029279ec12: Pull complete Digest: sha256:d26d529daa4d8567167181d9d569f2a85da3c5ecaf539cace2c6223355d69981 Status: Downloaded newer image for ubuntu:latest btssio@ubuntudocker:~$ </code> Le TAG est latest par défaut si vous ne précisez pas la version du conteneur voulu. Cette image ubuntu est une image officielle ; elle n'est pas préfixée et vous pouvez avoir des informations sur Docker hub.**



Cette image est basée sur quatre couches logicielles : <code shell> ... f476d66f5408: Pull complete 8882c27f669e: Pull complete

d9af21273955: Pull complete f5029279ec12: Pull complete ... </code>

INFORMATION
Le chargement de ce conteneur **ubuntu** et des couches logicielles nécessaires a lieu **une fois pour toutes**. Tous les lancements de conteneur suivant se baseront sur cette image, y compris - et c'est une des grands avantages de l'architecture par couches - toutes les images qui se trouvent sur Docker Hub et qui ont été construites à partir de cette image ubuntu. Et il existe sur Docker Hub de nombreuses images basées sur Ubuntu qui est une distribution de base fréquemment utilisée pour les images Docker.

Lancez un conteneur Ubuntu. Comme précédemment, le conteneur est ensuite arrêté mais existe bien. La colonne COMMAND indique que le processus qui a été lancé est cette fois-ci un shell Bash. <code shell> btssio@ubuntudocker:~$ docker run ubuntu btssio@ubuntudocker:~$ docker ps -a CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES b5ec671c6d11 ubuntu "/bin/bash" 21 s ago Exited ... laughing_ride ... </code> Vous pouvez prendre connaissance de la configuration du conteneur sur Docker Hub en consultant son fichier **Dockerfile**.

IMPORTANT
Le fichier **Dockerfile** associé à une image Docker, **décrit exactement** comment l'image a été **construite**. Quand vous recherchez sur Docker Hub une image qui doit répondre à votre besoin et qui a été publié par un membre de la communauté, ce fichier vous permet de savoir ce que contient l'image afin de vous permettre de faire un choix éclairé parmi la profusion d'images possibles. Une bonne pratique consiste donc :

- à **choisir une image officielle** si celle-ci répond à votre besoin,
- et, si cela n'est pas le cas, à choisir une image de la communauté en regardant si elle est souvent utilisée (sa **popularité**), sa **documentation** mais aussi son fichier **Dockerfile** pour savoir exactement ce qu'elle contient.

Sur Docker Hub plusieurs Dockerfile sont indiqués pour l'image officielle ubuntu :

Télécharger l'image Docker ubuntu :bionic-20190424 <code shell> btssio@ubuntudocker:~$ docker pull ubuntu:bionic-20190424 bionic-20190424: Pulling from library/ubuntu Digest: sha256:d26d529daa4d8567167181d9d569f2a85da3c5ecaf539cace2c6223355d69981 Status: Downloaded newer image for ubuntu:bionic-20190424 btssio@ubuntudocker:~$ docker images REPOSITORY TAG IMAGE ID CREATED SIZE Ubuntu bionic-20190424 d131e0fa2585 11 days ago 102MB ubuntu latest d131e0fa2585 11 days ago 102MB btssio@ubuntudocker:~$ </code> Il s'agit exactement de la même image comme le montre le champ IMAGE ID. Vous pouvez donc télécharger une version plus ancienne d'Ubuntu si votre projet porte sur une version bien précise de cet OS et faire cohabiter ensuite des conteneurs utilisant des versions différentes de système d'exploitation ou de paquets logiciels. Le champ SIZE indique la taille virtuelle de l'image, c'est à dire la taille de la couche téléchargée qui est plus important que celle du conteneur hello-world car l'image ubuntu ne s'appuie pas sur une couche préexistante. Pour connaître comment cette image a été construite, cliquez sur le lien du Dockerfile : <code shell> FROM scratch ADD ubuntu-bionic-core-cloudimg-amd64-root.tar.gz / ... # overwrite this with 'CMD []' in a dependent Dockerfile CMD ["/bin/bash"] </code> La première ligne - FROM scratch - indique que cette image ne se base pas sur une image préexistante. On part de la version de base d'Ubuntu. La dernière ligne - CMD ["/bin/bash"] - indique le processus qui doit être lancé par le conteneur et qui est ici un shell Bash. Sans rentrer dans le détail du contenu du fichier Dockerfile, les différentes actions décrites dans ce fichier se sont traduites par le téléchargement des quatre couches de l'image ubuntu sur votre serveur Ubuntu.
==== Utiliser les conteneurs ==== Le conteneur ubuntu lancé prévoit bien l'exécution d'un processus shell mais il s'est ensuite automatiquement arrêté. En effet, comme rien n'a été précisé lors du lancement pour que vous puissiez utiliser ce shell, le conteneur a considéré que l'action à faire, lancer un shell et rien d'autre, est terminée. Il s'arrête donc. === Lancer un conteneur en mode interactif === Pour pouvoir réellement interagir et utiliser le conteneur ubuntu grâce à ce processus de shell Bash qui est lancé, il faut : * lancer le conteneur en mode interactif pour lui associer une console TTY : paramètre -i, * ouvrir une console pour que vous puissiez saisir des

commandes : paramètre -t. <code shell> btssio@ubuntudocker:~$ docker run -i -t ubuntu root@394beb25ab78:/# ls bin dev home lib64 mnt proc run srv tmp var boot etc lib media opt root sbin sys usr root@394beb25ab78:/# cd /root root@394beb25ab78:/# touch docker.txt root@394beb25ab78:/# ls docker.txt root@394beb25ab78:/# </code> Le lancement du conteneur est toujours aussi rapide et le résultat de la commande est différent puisque vous avez le **shell du conteneur** avec comme nom **l'identifiant** du nouveau conteneur lancé. Vous êtes identifiés comme **root** sur la VM nommée **394beb25ab78** qui est l'identifiant du nouveau conteneur lancé et qui a servi à attribuer un nom à la VM. La commande **ls** montre l'arborescence de la VM et non celle du serveur Ubuntu. Un fichier a été créé dans le dossier **/root**. Tapez **Exit** pour revenir à la machine hôte en sortant du shell et donc du conteneur qui ne faisant que tourner cet unique processus s'arrête.

IMPORTANT
Toutes les **modifications effectuées** dans le conteneur lancé, comme la création de ce fichier docker.txt, ne modifie pas l'image ubuntu qui a servi à sa création. Les modifications ne sont faites que dans une **nouvelle couche** ajoutée par Docker pour gérer les modifications du système de fichiers afin de n'avoir aucun impact sur la couche inférieure, la couche ubuntu. En lançant un nouveau conteneur, vous consterez que le dossier root ne contient pas de fichier docker.txt.

=== Autre commande pour visualiser les conteneurs actifs === <code shell> btssio@ubuntudocker:~$ docker container ls </code> === Lancer un conteneur en mode détaché === Lors du lancement d'un conteneur, on perd l'accès à la console du serveur Ubuntu. Cette commande permet de lancer le conteneur et de retrouver le shell du serveur sans arrêter le conteneur. btssio@ubuntudocker:~$ docker run -i -t -d ubuntu e70a11313849c4c2ce8beebe72935368b909759ab5eb4e2c2f1bcf715c1a2bd1 btssio@ubuntudocker:~$ docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES e70a11313849 ubuntu "/bin/bash" 3 seconds ago Up 2 s gallant_kilby On peut aussi écrire les paramètres de la façon suivante : btssio@ubuntudocker:~$ docker run -itd ubuntu Se connecter à un conteneur lancé Après le lancement d'un conteneur en mode détaché, il peut être nécessaire de pouvoir accéder à la console du conteneur et donc au shell. Voici deux manières de procéder. S'attacher à la console du conteneur : ATTENTION, en quittant le shell, cela arrête le processus shell et donc le conteneur : btssio@ubuntudocker:~$ docker attach gallant_kilby root@e70a11313849:/# Lancer un processus shell : en quittant la console le conteneur n'est pas arrêté : btssio@ubuntudocker:~$ docker exec -it gallant_kilby root@e70a11313849:/# Obtenir la liste des modifications d'un conteneur par rapport à son image de lancement btssio@ubuntudocker:~$ docker diff 394beb25ab78 C /root A /root/.bash_history A /root/docker.txt btssio@ubuntudocker:~$ Cette commande utilise la syntaxe diff/patch en Linux : • D pour les parties d'arborescence supprimées, • C pour les parties d'arborescence créées : le dossier /root • A pour les ajouts : le fichier .bash_history est créé automatiquement pour stocker l'historique des actions effectuées dans le shell, et le fichier docker.txt que vous avez créé manuellement. Créer une image personnalisée L'image qui a servi à la création d'un conteneur n'est jamais modifiée par ce qui est fait à l'intérieur d'un conteneur. Pour que les modifications puissent faire partie d'une image, il faut créer une nouvelle, dans laquelle une nouvelle couche logicielle sera ajoutée et cette couche supplémentaire contiendra les modifications par rapport à l'image de base. Pour créer une nouvelle image basée sur l'image ubuntu de base mais intégrant les modifications souhaitées comme le fichier docker.txt ou l'installation de paquets logiciels voici comment procéder. Tout d'abord créez un conteneur et personnalisez-le : btssio@ubuntudocker:~$ docker run -i -t ubuntu root@407248dafa24:/# cd /root root@407248dafa24:~# touch docker.txt root@407248dafa24:~# ls docker.txt root@407248dafa24:~# apt-get update Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB] ... root@407248dafa24:~# apt-get install net-tools Reading package lists... Done ... root@407248dafa24:~# ifconfig eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 172.17.0.2 netmask 255.255.0.0 broadcast 172.17.255.255 ether 02:42:ac:11:00:02 txqueuelen 0 (Ethernet) RX packets 2109 bytes 16445383 (16.4 MB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 1570 bytes 110630 (110.6 KB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536 inet 127.0.0.1 netmask 255.0.0.0 loop txqueuelen 1000 (Local Loopback) RX packets 0 bytes 0 (0.0 B) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 0 bytes 0 (0.0 B) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 root@407248dafa24:~# exit btssio@ubuntudocker:~$ En visualisant les conteneurs existants, vous retrouvez celui que vous venez de créer (ID 407248dafa24) avec le nom généré par Docker (festive*sammet) btssio@ubuntudocker:~$ docker ps -a CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 407248dafa24 ubuntu "/bin/bash" 2 m.. Up 2 minutes festive*sammet Puis créez votre nouvelle image en utilisant son ID ou son nom et en lui associant un nouveau nom d'image préfixé par une information qui vous identifie : btssio@ubuntudocker:~$ docker commit festive*sammet techer/ubuntu*nettools sha256:2cd084ad8053b2c7d5747a7b0975006681981ea9f8e8dfea6f20c77bb361ff25 btssio@ubuntudocker:~ $ docker images REPOSITORY TAG IMAGE ID CREATED SIZE techer/ubuntu_nettools latest 2cd084ad8053 7 seconds ago 128MB ubuntu latest d131e0fa2585 11 days ago 102MB ... Une nouvelle image existe avec comme nom techer/ubuntu_nettools et d'une taille supérieure à celle de ubuntu compte tenu du paquet logiciel net Tools installé. La création d'un nouveau conteneur à partir de cette nouvelle image montre qu'elle a été personnalisée : btssio@ubuntudocker:~ $ docker run -it techer/ubuntu_nettools root@88085b7f2c50:/# ls /root docker.txt root@88085b7f2c50:/# Ce premier aperçu des possibilités de Docker est destiné à mieux comprendre la solution Kathara que vous allez ensuite utiliser. Cette solution utilise Docker et des scripts python pour faciliter la création et la gestion des VMs pour maquetter des infrastructures de réseau et de services. 3.D Quelques autres commandes de Docker Supprimer un conteneur docker rm [identifiant ou nom du conteneur] Faire le ménage La commande system et sa sous-commande prune permettant de réaliser le ménage dans les conteneurs arrêtés, les images orphelines et d'autres ressources a priori non utilisées, sans supprimer les images. docker system prune Supprimer une image Les conteneurs qui utilisent cette image doivent au préalable avoir été supprimés. Le paramètre -f force cependant la suppression si cela n'est pas le cas. docker rmi [identifiant ou nom de l'image] Lancer un conteneur avec un nom donné docker run –name=[nom fourni] [image] Lancer un conteneur en mode détaché Lors du lancement d'un conteneur, on perd l'accès à la console du serveur Ubuntu. Cette commande permet de lancer le conteneur et de retrouver le shell du serveur sans arrêter le conteneur. btssio@ubuntudocker:~$ docker run -i -t -d ubuntu e70a11313849c4c2ce8beebe72935368b909759ab5eb4e2c2f1bcf715c1a2bd1 btssio@ubuntudocker:~$ docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES e70a11313849 ubuntu "/bin/bash" 3 seconds ago Up 2 s gallant_kilby Ss connecter à un conteneur lancé Pour accéder ensuite à ce conteneur on utilise la commande attach avec l'identifiant ou le nom du conteneur : btssio@ubuntudocker:~/Documents/lab_exercice$ docker attach e70a11313849 root@e70a11313849:/# Arrêter un conteneur docker stop [conteneur] Démarrer un conteneur arrêté docker start [conteneur]

From:
/ - **Les cours du BTS SIO**

Permanent link:
/doku.php/docker/utiliser?rev=1567457711

Last update: **2019/09/02 22:55**