

Docker : Révision



Objectifs

- Retour
- Les images
- Docker run
- Dockerfile
- Les couches images de docker
- Partage d'image

Docker – RUN

- `docker run [-rm] [autres options] <nom_image> [arguments]`
 - Télécharge l'image si elle n'existe pas en local
 - Crée un conteneur à partir de l'image et l'exécute

Définition d'une image

- Une image correspond aux **fichiers binaires** et aux **dépendance**, ainsi qu'aux **métadonnées** concernant la façon dont il faudra l'exécuter.

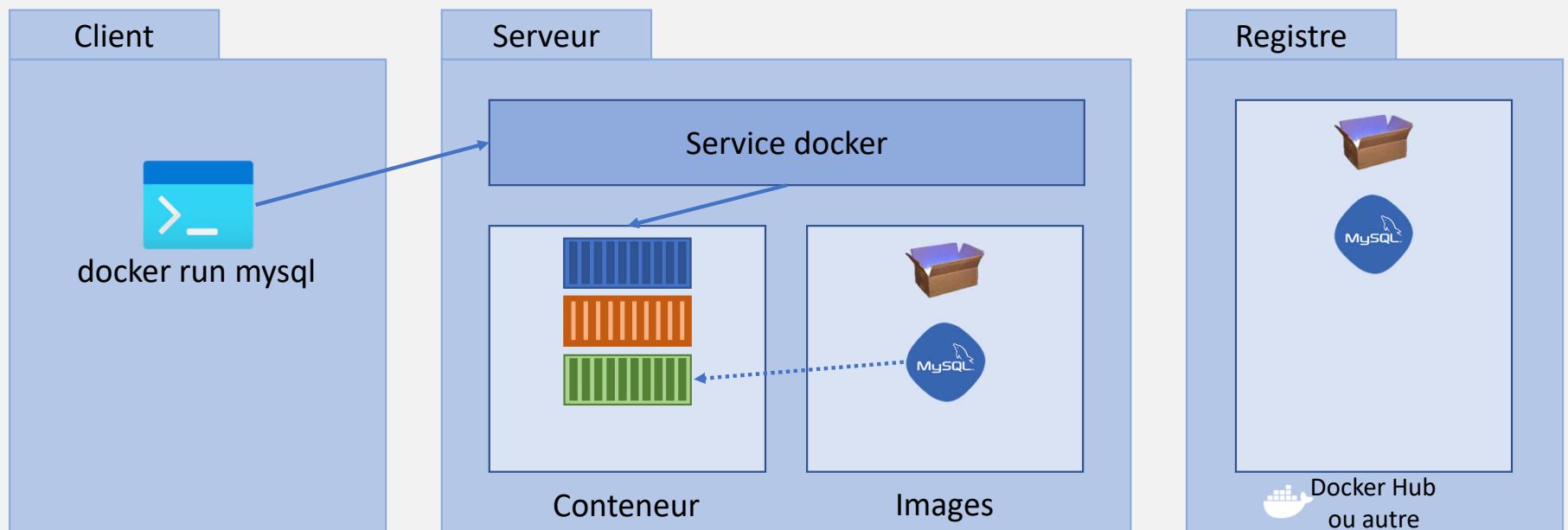
Utilisations

- Déploiement :
 - Léger : ne contient que l'application et ses dépendances
 - Autonome : contient toutes les dépendances
 - Sécurité : isolement du système de fichiers et du réseau
- Développement :
 - Accéder rapidement à un service sans l'installer sur sa machine et risquer de la déstabiliser
 - Dans un environnement complexe, par exemple plusieurs applications qui communiquent ensemble, on peut les déployer et se concentrer sur l'application que l'on développe sans passer beaucoup de temps à installer des pré-requis

Que se passe-t-il avec la cmd run ?

- Lorsqu'on utilise la commande **docker container run** docker va d'abord dans son cache local
- S'il ne la trouve pas, il va faire une recherche sur le Docker Hub par défaut (dépôt que l'on peut modifier)
- Télécharge la dernière version de l'image (sauf si on lui précise la version à télécharger) et la stocker dans son cache.
- Création d'un nouveau conteneur basé sur l'image spécifiée et se prépare à l'exécuter.

Docker – run



Dockerfile

- Les images sont créées à partir du fichier « Dockerfile »

```
FROM scratch  
  
COPY premier-programme /  
ENTRYPOINT ["premier-programme"]
```

- Le fichier premier-programme a été préalablement créé :

```
jpduches@Bilbo:~/docker/cpp$ ls  
premier-programme.cpp  
jpduches@Bilbo:~/docker/cpp$ cat premier-programme.cpp  
#include <iostream>  
  
int main(int argc, char** argv) {  
    std::cout << "Bonjour à tous" << std::endl;  
}  
jpduches@Bilbo:~/docker/cpp$ g++ ./premier-programme.cpp -o ./premier-programme -static  
jpduches@Bilbo:~/docker/cpp$ ls  
premier-programme  premier-programme.cpp  
jpduches@Bilbo:~/docker/cpp$
```

Documentation : <https://docs.docker.com/engine/reference/builder/>

Dockerfile : build

- L'image est créée à partir du Dockerfile
- Utilisez la commande `docker build --tag <nom_image>:<version> .`

```
> docker build --tag premier-programme:latest .  
Sending build context to Docker daemon 51.71kB  
Step 1/3 : FROM scratch  
--->  
Step 2/3 : COPY premier-programme /  
---> ef6a20c69526  
Step 3/3 : ENTRYPOINT ["/premier-programme"]  
---> Running in 137f676d2d6c  
Removing intermediate container 137f676d2d6c  
---> e0e83c6eaf36  
Successfully built e0e83c6eaf36  
Successfully tagged premier-programme:latest
```

- Test :

```
> docker run --rm premier-programme  
Bonjour à tous !
```

Dockerfile

- Les images sont créées à partir du fichier « Dockerfile »

```
> docker image history premier-programme
```

IMAGE BY	CREATED	CREATED SIZE	COMMENT	
e0e83c6eaf36	14 minutes ago	/bin/sh -c #(nop)	ENTRYPOINT ["/premier-pro...	0B
ef6a20c69526	14 minutes ago	/bin/sh -c #(nop)	COPY file:d502050d334ff3dd...	23kB

Dockerfile

- On peut aussi partir d'une image existante :

```
FROM busybox
```

```
ENTRYPOINT ["echo"]
```

```
CMD ["Bonjour à tous"]
```

```
> docker build --tag echo:latest .
Sending build context to Docker daemon 2.048kB
Step 1/3 : FROM busybox
----> edabd795951a
Step 2/3 : ENTRYPOINT ["echo"]
----> Running in e8da44107e81
Removing intermediate container e8da44107e81
----> 4b9b29f503bd
Step 3/3 : CMD ["Bonjour à tous"]
----> Running in 04545cc53647
Removing intermediate container 04545cc53647
----> 5c368523eb4a
Successfully built 5c368523eb4a
Successfully tagged echo:latest
```

```
> docker image history echo
```

IMAGE	CREATED	CREATED BY		SIZE	COMMENT
5c368523eb4a	About a minute ago	/bin/sh -c #(nop)	CMD ["Bonjour à tous"]	0B	
4b9b29f503bd	About a minute ago	/bin/sh -c #(nop)	ENTRYPOINT ["echo"]	0B	
edabd795951a	2 days ago	/bin/sh -c #(nop)	CMD ["sh"]	0B	
<missing>	2 days ago	/bin/sh -c #(nop)	ADD file:4e5169fa630e0afed...	1.22MB	

Dockerfile

```
FROM busybox  
  
ENTRYPOINT ["echo"]  
CMD ["Bonjour à tous"]
```

```
> docker run --rm echo  
Bonjour à tous  
> docker run --rm echo "Ici un nouvel argument !"  
Ici un nouvel argument !
```

Exemple de fichier Dockerfile

```
# Le FROM est un élément obligatoire à faire figurer dans le Dockerfile
# On utilise en général une distribution Linux minimaliste (comme debian ou alpine)
# Il est possible de partir d'un conteneur complètement vide en utilisant « FROM scratch »
FROM ubuntu:latest

# Vous pouvez définir de manière optionnelle une variable d'environnement
# L'avantage d'utiliser cette option, consiste au fait que peu importe la distribution
# Linux que vous utilisez, la commande reste la même pour injecter ces variables à l'intérieur de votre conteneur
ENV MA_VARIABLE "je suis une variable"

# Grâce au "RUN" vous pouvez exécuter de véritables commandes Shell à l'intérieur du conteneur au moment où il est buildé.
# La commande ci-dessous installe le paquet nginx et le paquet curl
RUN apt-get update && apt-get install nginx curl -y

# Par défaut, aucun port TCP ou UDP n'est ouvert. La commande "EXPOSE" permet d'ouvrir les ports indiqués sur le conteneur
# Attention, cela n'empêche pas d'utiliser l'option --publish pour rediriger ces ports vers ceux de votre machine
EXPOSE 80 443

# Le dernier paramètre obligatoire CMD correspond à la commande à exécuter lorsque le conteneur démarre.
# Il peut s'agir d'un exécutable, si vous omettez l'exécutable, vous devez spécifier une instruction ENTRYPOINT
# Attention seule une commande CMD est autorisée et si vous en indiquez plusieurs, seule la dernière sera utilisée
CMD ["nginx", "-g", "daemon off;"]
```

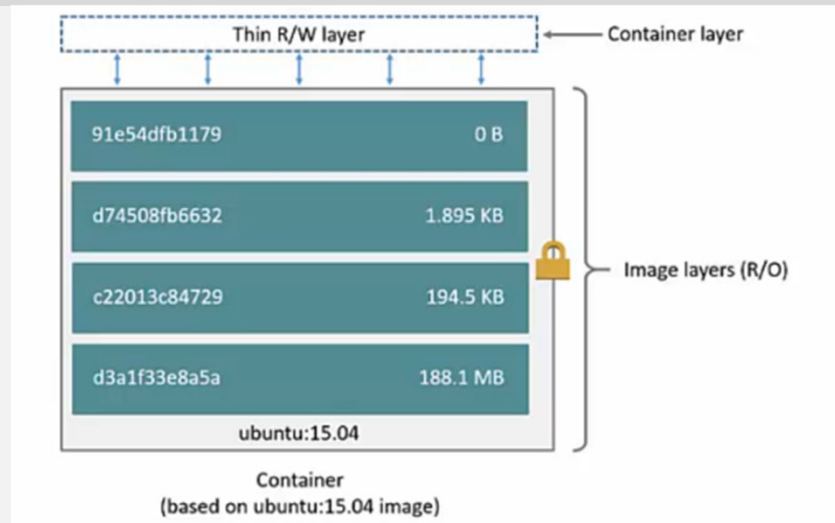
Docker image inspect <image>

- Retourne en format JSON les métadonnées correspondes à l'image inspectée : Port exposés, ID de l'image, Variable d'environnement, Cmd.

```
jpduches@Bilbo:~/docker$ docker image inspect mysql:latest
[
  {
    "Id": "sha256:c60d96bd2b771a8e3cae776e02e55ae914a6641139d963defeb3c93388f61707",
    "RepoTags": [
      "mysql:latest"
    ],
    "RepoDigests": [
      "mysql@sha256:8b928a5117cf5c2238c7a09cd28c2e801ac98f91c3f8203a8938ae51f14700fd"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2021-07-22T09:46:29.474252274Z",
    "Container": "f67c60fa88bbda745eaf4ad5adec3968b9248ce6babe7f0797d8e26972964e59",
    "ContainerConfig": {
      "Hostname": "f67c60fa88bb",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "ExposedPorts": {
        "3306/tcp": {},
        "33060/tcp": {}
      },
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
```

```
    "StdinOnce": false,
    "Env": [
      "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
      "GOSU_VERSION=1.12",
      "MYSQL_MAJOR=8.0",
      "MYSQL_VERSION=8.0.26-1debian10"
    ],
    "Cmd": [
      "/bin/sh",
      "-c",
      "#(nop) ",
      "CMD [\\\"mysqld\\\"]"
    ],
    "Image": "sha256:a717583fdcec69e6c839d2647da972980b6dcce80cad9bcdce9c760c10e222ba",
    "Volumes": {
      "/var/lib/mysql": {}
    },
    "WorkingDir": "",
    "Entrypoint": [
      "docker-entrypoint.sh"
    ],
    "OnBuild": null,
    "Labels": {}
  },
  {
    "DockerVersion": "20.10.7",
    "Author": "",
    "Config": {
      "Hostname": "",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
```

Les couches d'une image Docker



- Chaque couche possède sa propre signature SHA unique
- Ce peut être l'installation d'un paquet avec apt install, ou le paramétrage d'une variable d'environnement.

Docker image history

- Montre les différentes couches de modifications qui ont été appliquées à l'image Docker
- Toutes les images démarrent au tout début avec une couche vide appelée « scratch », et tous les changements qui arrivent ensuite sur le système de fichiers de cette image sont une autre couche.

```
jpduches@Bilbo:~/docker$ docker image history premier-programme
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
ac6affa7c467   6 minutes ago   ENTRYPOINT ["/premier-programme"] 0B        buildkit.dockerfile.v0
<missing>      6 minutes ago   COPY premier-programme / # buildkit 2.37MB    buildkit.dockerfile.v0
jpduches@Bilbo:~/docker$
```

```
<missing>      13 days ago    /bin/sh -c #(nop) ADD file:45f5dfa135c848a34... 69.3MB
jpduches@Bilbo:~/docker$ docker image history mysql:latest
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
c60d96bd2b77   13 days ago     /bin/sh -c #(nop) CMD ["mysqld"]    0B
<missing>      13 days ago     /bin/sh -c #(nop) EXPOSE 3306 33060 0B
<missing>      13 days ago     /bin/sh -c #(nop) ENTRYPOINT ["docker-entryp... 0B
<missing>      13 days ago     /bin/sh -c ln -s usr/local/bin/docker-entryp... 34B
<missing>      13 days ago     /bin/sh -c #(nop) COPY file:345a22fe55d3e678... 14.5kB
<missing>      13 days ago     /bin/sh -c #(nop) COPY dir:2e040acc386ebd23b... 1.12kB
<missing>      13 days ago     /bin/sh -c #(nop) VOLUME [/var/lib/mysql] 0B
<missing>      13 days ago     /bin/sh -c { echo mysql-community-server m... 378MB
<missing>      13 days ago     /bin/sh -c echo 'deb http://repo.mysql.com/a... 55B
<missing>      13 days ago     /bin/sh -c #(nop) ENV MYSQL_VERSION=8.0.26-... 0B
<missing>      13 days ago     /bin/sh -c #(nop) ENV MYSQL_MAJOR=8.0      0B
<missing>      13 days ago     /bin/sh -c set -ex; key='A4A9406876FCBD3C45... 1.84kB
<missing>      13 days ago     /bin/sh -c apt-get update && apt-get install... 52.2MB
<missing>      13 days ago     /bin/sh -c mkdir /docker-entrypoint-initdb.d 0B
<missing>      13 days ago     /bin/sh -c set -eux; savedAptMark="$(apt-ma... 4.17MB
<missing>      13 days ago     /bin/sh -c #(nop) ENV GOSU_VERSION=1.12    0B
<missing>      13 days ago     /bin/sh -c apt-get update && apt-get install... 9.34MB
<missing>      13 days ago     /bin/sh -c groupadd -r mysql && useradd -r -... 329kB
<missing>      13 days ago     /bin/sh -c #(nop) CMD ["bash"]            0B
<missing>      13 days ago     /bin/sh -c #(nop) ADD file:45f5dfa135c848a34... 69.3MB
jpduches@Bilbo:~/docker$
```


Partager ses images

- Il faut tagger l'image avec son id

```
jpduches@Bilbo:~$ docker image push premier-programme
Using default tag: latest
The push refers to repository [docker.io/library/premier-programme]
ed56007c31df: Preparing
denied: requested access to the resource is denied
jpduches@Bilbo:~$ docker image tag premier-programme:latest jpduches/premier-programme
jpduches@Bilbo:~$ docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
jpduches/premier-programme  latest     ac6affa7c467  About an hour ago  2.37MB
premier-programme        latest     ac6affa7c467  About an hour ago  2.37MB
bitnami/laravel        latest     664eed9f099f  18 hours ago    592MB
nginx                  latest     08b152afcfae  13 days ago     133MB
mysql                  latest     c60d96bd2b77  13 days ago     514MB
alpine/git             latest     b8f176fa3f0d  2 months ago    25.1MB
jpduches@Bilbo:~$ docker image push jpduches/premier-programme
Using default tag: latest
The push refers to repository [docker.io/jpduches/premier-programme]
ed56007c31df: Pushed
latest: digest: sha256:263fba4af006f9f34d0e1ad544924f6294ea650ccf90e2516c7f9197ed5ebae6 size: 526
jpduches@Bilbo:~$
```

Attention, vous devez être connecté : docker login et après faire un logout quand vous avez terminé