

DEVOPS I : TOUT SAVOIR DE DOCKER EN QUELQUES HEURES

Comment trouver et créer ses propres
images Docker ?

OBJECTIFS DE LA SECTION

Objectifs

- Qu'est-ce qu'une image ?
- Apprendre à utiliser DockerHub correctement
- Analyser le cache d'une image Docker
- Taguer ses images et les publier sur DockerHub
- Les éléments de base d'un fichier Dockerfile
- Lancer ses propres images Docker
- Exercice sur la création d'image Docker

QU'EST-CE QU'UNE IMAGE ?

La définition d'une image

- Une image correspond aux **fichiers binaires** et aux **dépendances**, ainsi qu'aux **métadonnées** concernant la façon dont il faudra l'exécuter.
- Définition officielle : "Une image est une collection ordonnées de modifications du filesystem root et les paramètres d'exécution correspondants pour l'exécuter à l'intérieur d'un container. »

Mais une image n'est pas...

- Il n'y a pas de système d'exploitation complet, pas de kernel (noyau) et pas de modules kernel (notamment les drivers).
- L'image peut être aussi petite qu'un simple fichier (par exemple lorsqu'on a une app en Go), mais aussi assez grande lorsqu'il s'agit d'une distribution Ubuntu avec APT, apache et PHP installés.

Les spécifications officielles d'une image

- On peut retrouver cet article (en anglais) qui reprend les spécifications officielles d'une image Docker :
- <https://docs.docker.com/registry/spec/manifest-v2-2/>

UTILISER CORRECTEMENT DOCKER HUB

Qu'est-ce que Docker Hub ?

- Docker Hub est un dépôt sur le Cloud totalement géré par Docker.
- On peut y publier et utiliser des images (officielles ou non) et partager son travail avec la communauté (un peu comme le fait github et gitlab pour le code)
- Il y existe des dépôts privés, utilisable seulement par vous de sorte que les autres utilisateurs ne puissent récupérer vos images personnalisées.

S'inscrire et se connecter à Docker Hub

- <https://hub.docker.com/>

Trouver les images officielles

- Si on effectue une recherche pour une image mysql, on peut voir qu'il y a plus de 19000 résultats.
- Pour distinguer les bonnes images, on va d'abord distinguer les images officielles des autres :

The screenshot shows the Docker Hub search results for the query 'mysql'. The interface includes a left sidebar with filters, a main search results area, and a right sidebar with sorting options. The 'mysql' image is the top result, showing it is an 'Official Image' with over 10 million downloads and 9.1K stars. The 'mariadb' image is the second result, also an 'Official Image' with over 10 million downloads and 3.2K stars. An orange arrow points to the 'Official Images' filter in the left sidebar, and an orange circle highlights the 'Official Image' badge on the 'mysql' image card.

Filters

Docker Certified ⓘ

☐ Docker Certified

Images

☐ Verified Publisher ⓘ
Docker Certified And Verified Publisher Content

☐ Official Images ⓘ
Official Images Published By Docker

Categories ⓘ

☐ Analytics

☐ Application Frameworks

☐ Application Infrastructure

☐ Application Services

☐ Base Images

1 - 25 of 18 586 results for **mysql**. [Clear search](#)

mysql

Updated 31 minutes ago

MySQL is a widely used, open-source relational database management system (RDBMS).

Container Linux x86-64 Databases

Official Image ⓘ

10M+ 9.1K
Downloads Stars

mariadb

Updated 31 minutes ago

MariaDB is a community-developed fork of MySQL intended to remain free under the GNU GPL.

Container Linux ARM 64 386 PowerPC 64 LE x86-64 Databases

Official Image ⓘ

10M+ 3.2K
Downloads Stars

Trouver les images officielles (2)

- Un autre point important, c'est que les images officielles n'ont pas de « / » à la différence des autres images :

The screenshot displays three Docker images on the Docker Hub interface. The first image is 'percona' by 'percona', marked as an 'OFFICIAL IMAGE'. The second is 'MySQL Server Enterprise Edition' by 'Oracle', marked as a 'VERIFIED PUBLISHER' and 'DOCKER CERTIFIED'. The third is 'mysql/mysql-server' by 'mysql', which is not an official image. Red arrows point to the image names 'percona', 'MySQL Server Enterprise Edition', and 'mysql/mysql-server'. The arrows for 'percona' and 'MySQL Server Enterprise Edition' point to names without slashes, while the arrow for 'mysql/mysql-server' points to a name with a slash, illustrating the difference in naming for official vs. non-official images.

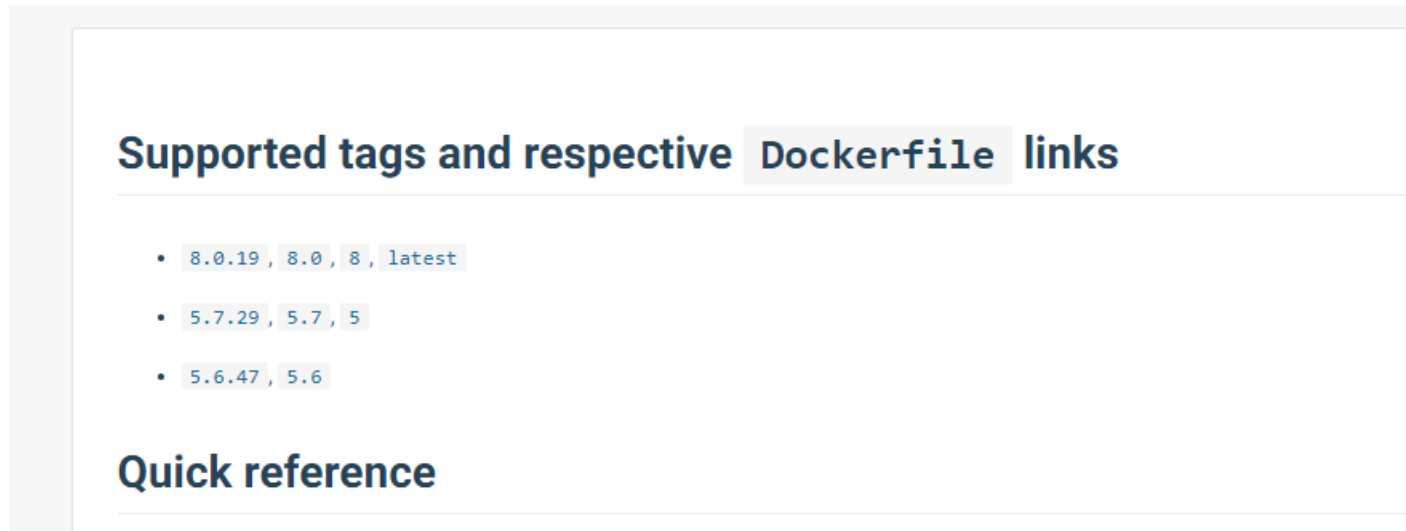
Image Name	Author	Official / Verified	Downloads	Stars
percona	percona	Official Image	10M+	469
MySQL Server Enterprise Edition	Oracle	Verified Publisher, Docker Certified	-	-
mysql/mysql-server	mysql	No	10M+	673

Comprendre la page repository

- On y a accès à beaucoup de documentations, quelles options on peut utiliser, quelles sont les variables d'environnement, le port par défaut, etc...
- https://hub.docker.com/_/mysql?tab=description
- On peut également y voir les différentes versions disponibles
- https://hub.docker.com/_/mysql?tab=description

Les tags

- Dans les tags supportés, on peut voir qu'il y a plusieurs manières d'appeler une version de l'image.
- Par exemple ci-dessous, que l'on utilise la version **8.0.19**, **8.0**, **8** ou **latest**, on obtiendra la même image.



The screenshot shows the Docker Hub page for the MySQL image. It features a section titled "Supported tags and respective Dockerfile links" which lists three groups of tags: 8.0.19, 8.0, 8, latest; 5.7.29, 5.7, 5; and 5.6.47, 5.6. Below this is a "Quick reference" section.

Supported tags	respective Dockerfile links
8.0.19, 8.0, 8, latest	
5.7.29, 5.7, 5	
5.6.47, 5.6	

Quick reference

- Si on fait un « `docker pull mysql:8.0.19` » ou `mysql:latest` cela revient au même

Liste des images officielles

- <https://github.com/docker-library/official-images/tree/master/library>

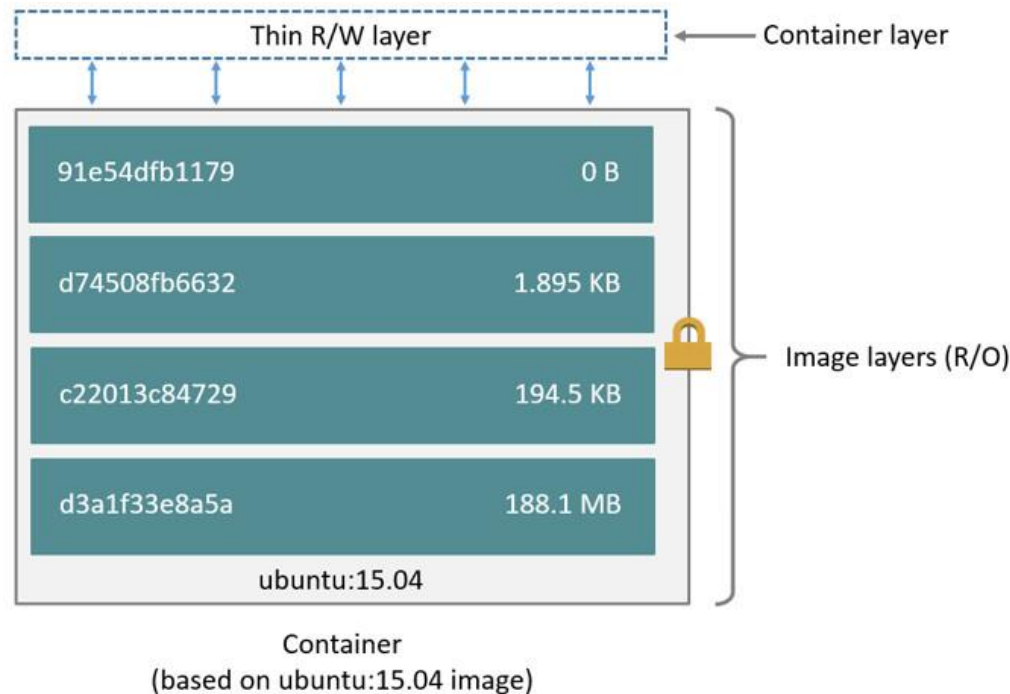
ANALYSER LE CACHE D'UNE IMAGE DOCKER



La commande « docker image history »

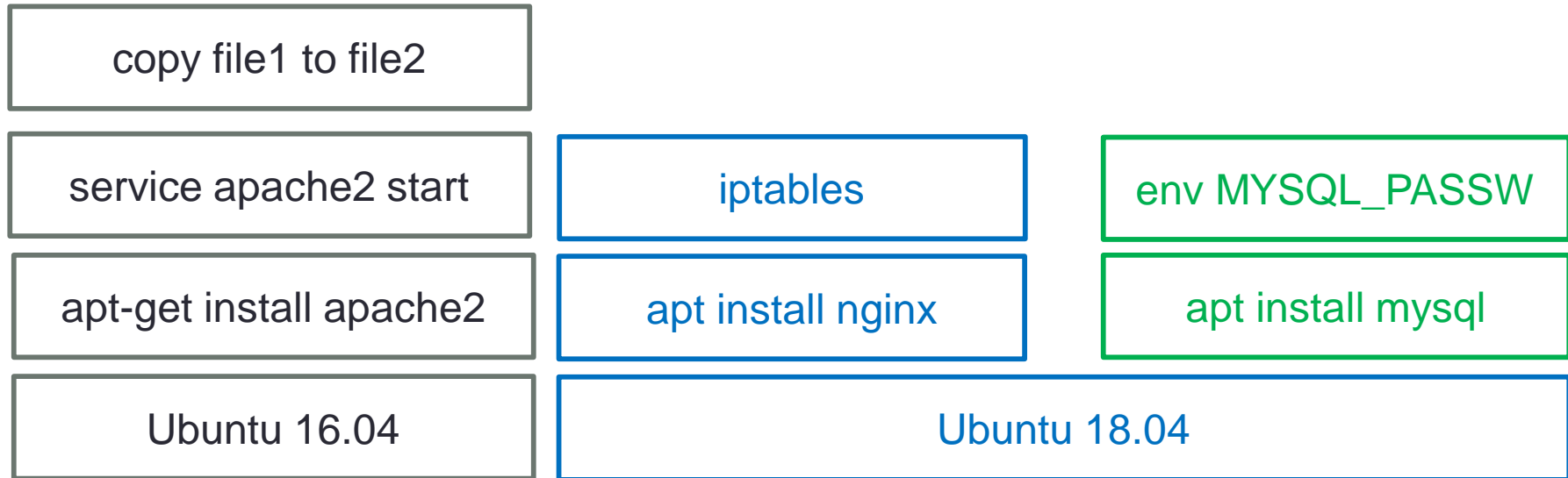
- **docker image history <NAME/ID>**
 - Montre les différentes couches de modifications qui ont été appliquées à l'image Docker
 - Toutes les images démarrent au tout début avec une couche vide appelée « scratch », et tous les changements qui arrivent ensuite sur le système de fichiers de cette image est une autre couche.
 - Tous les changements n'influent pas sur la taille de l'image, comme pour les métadonnées (comme les commandes)
- https://docs.docker.com/engine/reference/commandline/image_history/

Les couches d'une image Docker



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

L'intérêt du cache



- Les couches qui ont déjà été téléchargées n'ont pas besoin de l'être à nouveau pour une nouvelle image

La commande « docker image inspect »

- **docker image inspect <NAME/ID>**
 - Retourne en format JSON les métadonnées correspondantes à l'image inspectée
 - Ports exposés
 - ID de l'image
 - Variables d'environnement
 - Commande lancée au démarrage du conteneur
- https://docs.docker.com/engine/reference/commandline/image_inspect/

TAGUER SES IMAGES ET LES POUSSER DANS DOCKER HUB

La commande « docker image tag »

- `docker image tag <SOURCE_IMAGE>:TAG <TARGET_IMAGE>:TAG`
 - Le TAG correspond à un pointeur vers un commit spécifique sur une image
 - Le TAG « latest » correspond au tag par défaut
- https://docs.docker.com/engine/reference/commandline/image_tag/

La commande « docker image push »

- **docker image push <IMAGE_NAME>**
 - Upload les modifications des couches de l'image vers Docker Hub par défaut (on peut changer le dépôt destination)
 - Attention à s'authentifier avec la commande « docker login » avant de push des images vers Docker Hub
- https://docs.docker.com/engine/reference/commandline/image_push/

La commande « docker login »

- **docker login**
 - Permet de s'authentifier auprès de Docker Hub (ou un autre dépôt)
 - On peut observer le fichier créé à partir de cette commande en faisant un « **cat .docker/config.json** »
- <https://docs.docker.com/engine/reference/commandline/login/>

La commande « docker logout »

- **docker logout**

- Permet de supprimer le fichier d'authentification créé à partir de la commande « docker login »
- Important d'utiliser cette commande lorsqu'on se situe sur des machines dont on ne maîtrise pas parfaitement la sécurité, ou qui est utilisée par plusieurs utilisateurs

- <https://docs.docker.com/engine/reference/commandline/logout/>

LES ÉLÉMENTS BASIQUES DU FICHIER DOCKERFILE

Qu'est-ce qu'un Dockerfile ?

- Les dockerfiles sont des fichiers textes décrivant les différentes étapes de création d'un conteneur totalement personnalisé.
- Il existe des paramètres **obligatoires** et des paramètres **optionnels**.
- Vous pouvez trouver un exemple de DockerFile dans :
 - *udemy-docker/Dockerfile/Exemple 1/Dockerfile*
- <https://docs.docker.com/engine/reference/builder/>

Les différents paramètres – FROM

- 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**

Les différents paramètres - ENV

- 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"**

Les différents paramètres - RUN

- 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`

Les différents paramètres - EXPOSE

- 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**

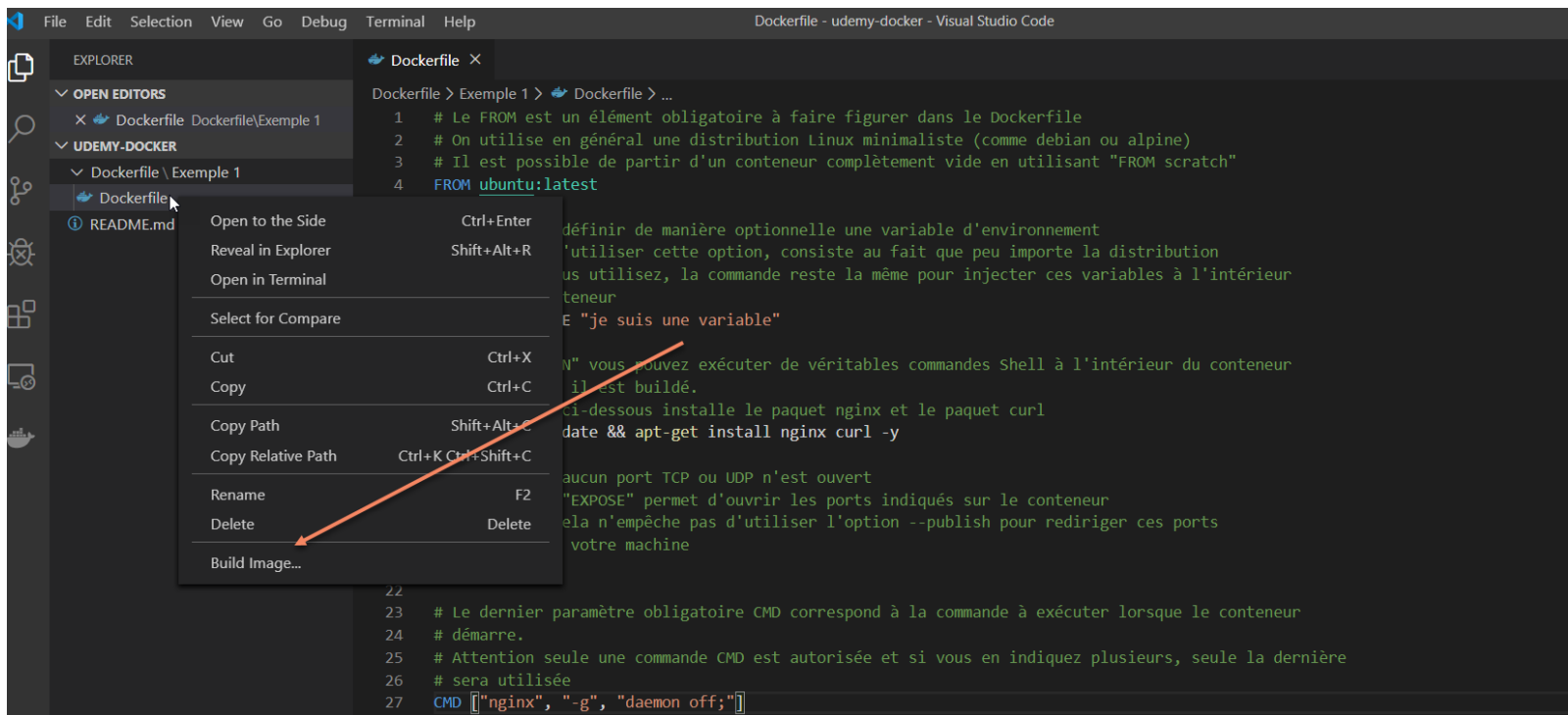
Les différents paramètres - CMD

- Le dernier paramètre obligatoire CMD correspond à la commande à exécuter lorsque le conteneur démarre.
- Attention seule une commande CMD est autorisée et si vous en indiquez plusieurs, seule la dernière sera utilisée
- On sépare les différents éléments de la commande comme dans l'exemple ci-dessous:
- `CMD ["nginx", "-g", "daemon off;"]`

LANCER SES PROPRES IMAGES DOCKER

Builder automatiquement avec Visual Studio

- Il est possible de builder automatiquement ses images Docker grâce à Visual Studio Code :



La commande « docker build »

- **docker image build -t <IMAGE_NAME>:<TAG> .**
 - Vous devez vous trouver dans le répertoire contenant le fichier Dockerfile pour exécuter la commande
 - L'option **-t** indique que l'image va être taguée
 - Attention à ne pas oublier le point à la fin de la commande
- https://docs.docker.com/engine/reference/commandline/image_build/

Petite remarque

- En cas de modification des derniers éléments du Dockerfile, lorsqu'on rebuild l'image, seules les dernières couches sont recalculées et réexécutées.

EXERCICE 1

Instructions

Créer sa première image

- Notre but va être de créer un conteneur basé sur l'image « **ubuntu:16.04** » capable d'exécuter la commande « **curl** » et sur lequel la page html située à : « **udemy-docker/Dockerfile/Exercice 1/sample.html** » tourne sur un processus nginx.

Créer sa première image

- Commencez par indiquer la partie « FROM »
- Spécifier grâce au « RUN » les différentes commandes à exécuter pour installer curl et nginx
- Exposez ensuite le port 80
- Indiquez le répertoire **/var/www/html** comme « WORKDIR »
- Copiez ensuite à l'intérieur de ce répertoire le fichier **sample.html** en le renommant **index.html**
- Utilisez le CMD suivant : **CMD ["nginx", "-g", "daemon off;"]**

Créer sa première image

- Construisez ensuite votre image en l'appelant **mywebserver** et en utilisant le tag **latest**.
- Lancez l'image et vérifiez que vous obtenez la bonne page en lançant <http://localhost> depuis votre navigateur.
- Rebuildez l'image en modifiant le tag pour lui permettre d'être poussé sur votre propre dépôt Docker Hub (indiquez votre nom de compte puis un / et le nom de l'image).
- Effacez l'image de votre cache, puis relancez un conteneur en lui spécifiant votre image stockée sur DockerHub.

EXERCICE 1

Correction

Les commandes à utiliser

- Pour centos :
 - `docker container run --publish 80:80 -it --name webserver_centos centos:8`
- Pour apache :
 - `docker container run --detach --publish 8080:80 -name apache_server httpd`
- Pour mysql :
 - `docker container run --detach --publish 3306:3306 --name mysql_server --env MYSQL_RANDOM_ROOT_PASSWORD=yes mysql`
 - `docker container logs mysql_server`