Devops

Un mélange parfait entre Développement et Administration Système

DOCKER : Commandes basiques Docker

Obtenir les informations de versions de Docker

Vérification de la version de docker

```
docker version
```

Vérification de la version de docker compose

```
docker-compose version
```

Obtenir les informations plus globale de Docker

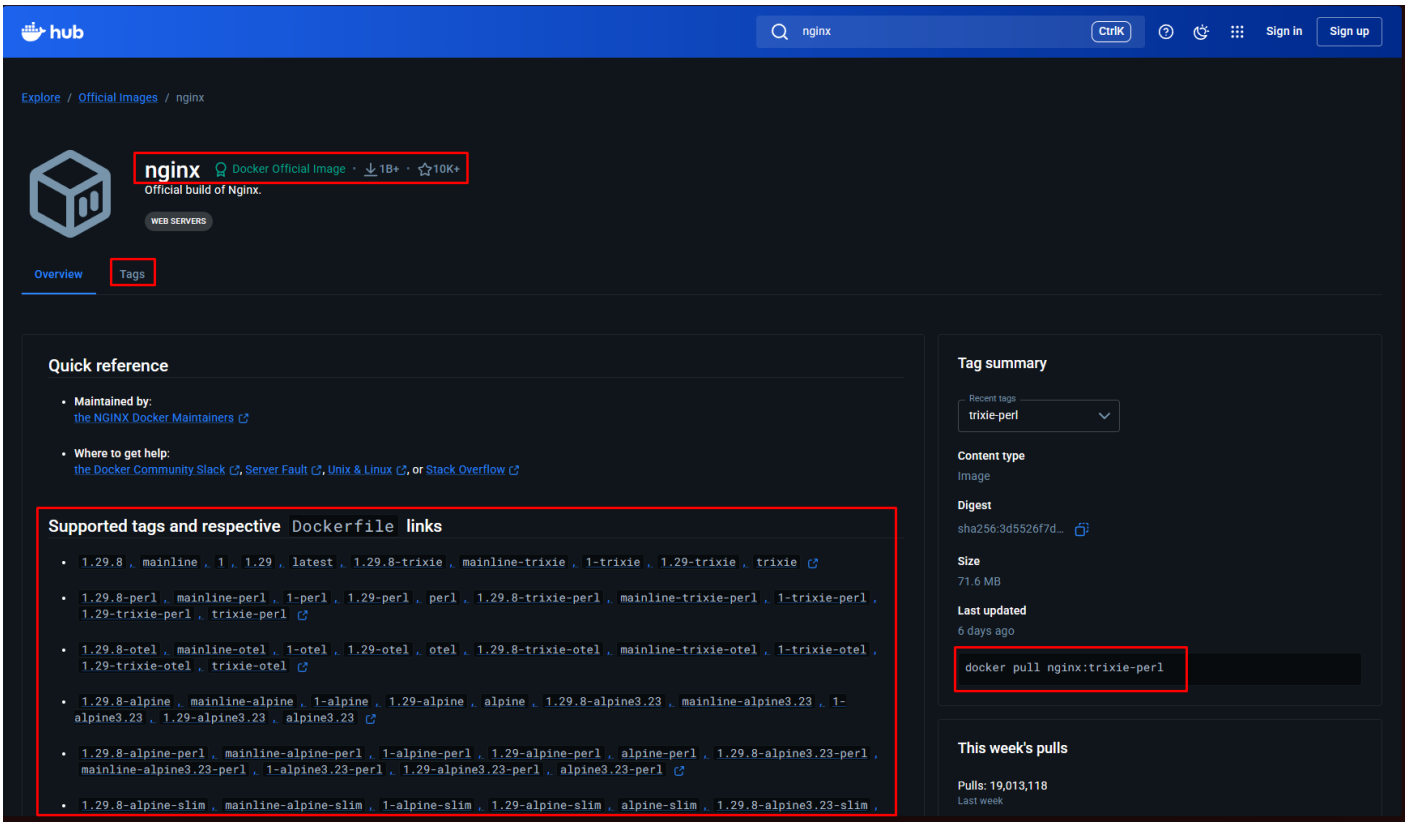
```
docker info
```

Utilisation d'un conteneur

Bibliothèque de conteneur

Le moyen le plus courant pour obtenir des images est avec le Docker Hub

Une fois l'image choisie, il faut le nom de l'image et la version que l'on souhaite télécharger :



Le titre permet d'indiquer le nom de l'image et potentiellement d'autres informations (Image officielle, nombre de téléchargements, nombre de favoris).

Les Tags indiquent l'ensemble des versions disponible au téléchargements.

Sur cette page nginx nous indique les dernières versions disponibles.

Téléchargement d'une image

Pour terminer a droite on retrouve la commande pour télécharger une image :

```
docker pull NOM_IMAGE_VERSION
```

Lancement d'un conteneur

Lancer un conteneur basique, ici depuis l'image nginx avec le port 80.

Le --publish permet de mettre en relation le port du conteneur avec le port de la machine hôte.

```
docker container run --publish 80:80 nginx
```

Ou plus simplement

```
docker container run -p 80:80 nginx
```

Lancer un conteneur mais en arrière plan dans cette configuration il aura un nom aléatoire

```
docker container run --detach --publish 80:80 nginx
```

Le --detach s'écrit aussi -d

```
docker container run -d -p 80:80 nginx
```

Le --name permet de choisir le nom du conteneur qui sera crée

```
docker container run --detach --name nginx_web --publish 80:80 nginx
```

De la même manière que les commandes précédentes il est possible de simplifier l'argument

```
docker container run -d --name nginx_web -p 80:80 nginx
```

Dans le cas d'une petite infrastructure renommer les conteneur n'est pas obligatoire mais on peut vite s'y perdre.

Affichage des conteneurs

Il est possible d'afficher l'ensemble des conteneurs qui ont été crées mais pas encore supprimés.

```
docker container ls -a
```

Pour afficher les conteneurs actifs sur la machine hôte c'est avec la commande suivante

```
docker ps
```

Stopper un conteneur

Pour stopper un conteneur il faut son nom ou son ID

```
docker container stop 'NOM/ID'
```

Idem avec

```
docker stop 'NOM/ID'
```

Supprimer un conteneur

Pour supprimer le ou les conteneur qui sont arrêtés mais toujours présent en fonds
Il est possible d'en supprimer plusieurs en ajoutant tous les noms a la suite les un des autres.

```
docker container rm 'NOM/ID'
```

Idem avec

```
docker rm 'NOM/ID'
```

Il n'est normalement pas possible de supprimer un conteneur en cours d'exécution sauf en le forçant avec l'ajout suivant

```
docker rm -f 'NOM/ID'
```

Affichage de logs

Pour regarder les logs

```
docker container logs 'NOM/ID'
```

Idem avec

```
docker logs 'NOM/ID'
```

Afficher les processus

Voir les processus en cours dans un container docker

```
docker container top 'NOM/ID'
```

Idem avec

```
docker top 'NOM/ID'
```

Informations sur les conteneur

La commande inspect permet d'obtenir un grand nombre d'informations sur la configuration et le fonctionnement du conteneur.

```
docker container inspect 'NOM/ID'
```

Idem avec

```
docker inspect 'NOM/ID'
```

Il est possible de formater le résultats pour n'avoir que certains résultats.
En modifiant la valeur entre crochet pour correspondre a la recherche souhaitée.

```
docker container inspect --format '{{.NetworkSettings.IPAddress}}' 'NOM/ID'
```

Il faudra faire attention a suivre le cheminement que l'on peut retrouver dans la commande inspect. Autre par exemple State.Status

image.png

Utilisation des ressources d'un conteneur

La commande stats permet d'afficher les informations d'utilisation des ressources d'un conteneur.

```
docker container stats 'NOM/ID'
```

Idem avec

```
docker stats 'NOM/ID'
```

Intervenir dans un conteneur

Lancer un conteneur et intervenir a l'intérieur

Pour lancer un conteneur la commande sera identique mais il y aura quelques options en plus pour accéder a un terminal

```
docker container run -it --name webserver -p 80:80 nginx bash
```

-t permet de simuler un terminal
-i permet de de garder le terminal ouvert et recevoir l'output des commandes
bash correspond a la commande qui est lancée au lancement du terminal, il permet de créer un terminal

On entre directement dans la console du conteneur
pour quitter le container il faut entrer `exit` mais cette commande éteint aussi le container

La raison pour laquelle le container s'éteint c'est que la commande pour faire fonctionner le container a été définit avec bash donc si l'on met fin au bash on met fin au conteneur par contre si on execute une autre commande puis avec la syntaxe ci-dessous que l'on execute une commande bash supplémentaire le conteneur restera allumé.

Lancer une commande dans un conteneur

Pour lancer une commande dans un conteneur qui tourne déja

```
docker container exec -it 'NOM/ID' 'commande'
```

ou

```
docker exec -it 'NOM/ID' 'commande'
```

la commande pour accéder a un terminal reste bash :

```
docker exec -it 'NOM/ID' bash
```

Etapes de création d'un conteneur

Dans un premier temps docker regarde sur son cache local si l'image est déja présente si elle ne l'est pas il la télécharge depuis DockerHub ou un autre repository s'il est configuré, comme exemple de repository personnalisé on peut citer Harbor.

Il téléchargera forcément la dernière version (:latest) de l'image si aucune version ne lui est précisé.
Il crée ensuite un conteneur basé sur l'image indiquée dans la commande.

Il attribue ensuite une IP virtuelle au container sur le réseau privé interne au serveur docker.
Avec l'option --publish il va ouvrir le port définit et le rediriger vers le port définit du conteneur.

Il va ensuite démarrer le conteneur avec la commande (CMD), qui est définit dans le Dockerfile lors de la création de l'image (normalement a la fin).

Exemple d'utilisation

La commande suivante permet de créer un conteneur avec l'application de base de donnée MySQL, le lancement du conteneur est simple et rapide mais le mot de passe 'root' sera généré

aléatoirement.

```
docker container run -d --name sqlserv --env MYSQL_RANDOM_ROOT_PASSWORD=yes -p 3306:3306
 mysql
```

Pour récupérer le mot de passe c'est plutôt simple il est noté de façon claire dans les logs du conteneur.

```
[Note] [Entrypoint]: GENERATED ROOT PASSWORD: 7Lh6eUrE7cRKJDQVPjEEodtEeSNNF3Hh
```

Il est important de le conserver et de supprimer cette ligne des logs.