# Demo 1

## Lancez votre premier conteneur en local

Nous allons prendre en main Docker, avant de commencer parlons de docker Hub

### Docker Hub

Docker hub est une plate-forme qui comparables a Github mais pour le partage des images et donc qui nous permettra de récupérer et de partager des images et Bonne nouvelle ! ça sera inclus dans notre demo !

### Démarrez votre premier conteneur !

Pour demarer votre premier conteneur vous devez utilisez la commande **docker run hello-world**  avec cette commande on lance un conteneur avec une image hello-world disponible en local,

Dans notre cas, le conteneur a démarré, puis affiché du contenu, et il a fini par s'arrêter. Si vous souhaitez que votre conteneur reste allumé jusqu’à l'arrêt du service qu'il contient, vous devez ajouter l’argument **--detach (-d)** .  Il vous permet de continuer à utiliser la console pendant que votre conteneur tourne sur un autre processus .

C’est basique mais comme on dit “ **un p’tit pas pour l’humanite mais un grand pas pour l’univers !”**

### Arrêtez votre conteneur Docker

Vous avez créé un conteneur avec l'option **--detach** , vous aurez donc sûrement besoin de l'arrêter ! Pour cela, faites appel à la commande **docker stop ID\_RETOURNÉ\_LORS\_DU\_DOCKER\_RUN**  .

Maintenant que votre conteneur Docker a été arrêté, vous pouvez le supprimer avec la commande **docker rm ID\_RETOURNÉ\_LORS\_DU\_DOCKER\_RUN**

## Récupérez une image du Docker Hub

Vous pouvez aussi avoir besoin de récupérer des images sur le Docker Hub sans pour autant lancer de conteneur. Pour cela, vous avez besoin de lancer la commande suivante : **docker pull hello-world**

En lançant cette commande, vous téléchargez une image directement depuis le Docker Hub, et vous la stockez en local sur votre ordinateur.

## Affichez l'ensemble des conteneurs existants

Quand vous créez des conteneurs avec l’argument --detach  , vous pouvez avoir besoin de savoir si les conteneurs sont toujours actifs ; pour cela, vous devez utiliser la commande**docker ps**

Vous pouvez aussi voir l'ensemble des images présentes en local sur votre ordinateur, avec la commande **docker images -a**

## Comment nettoyer mon système

Après avoir fait de nombreux tests sur votre ordinateur, vous pouvez avoir besoin de faire un peu de ménage. Pour cela, vous pouvez supprimer l'ensemble des ressources manuelles dans Docker.

Ou vous pouvez laisser faire Docker pour qu'il fasse lui-même le ménage. Voici la commande que vous devez utiliser pour faire le ménage :  **docker system prune**

Celle-ci va supprimer les données suivantes :

1. l'ensemble des conteneurs Docker qui ne sont pas en status running ;
2. l'ensemble des réseaux créés par Docker qui ne sont pas utilisés par au moins un conteneur ;
3. l'ensemble des images Docker non utilisées ;
4. l'ensemble des caches utilisés pour la création d'images Docker.

Vous savez maintenant télécharger, démarrer et arrêter des conteneurs ; vous êtes aussi capable de parler de Docker Hub et vous savez pourquoi on l'utilise.

## Créez votre premier Dockerfile

Vous savez utiliser l'interface de commande de Docker et récupérer des images depuis le Docker Hub. Mais comment créer votre propre image ?

nous allons créer ensemble une image Docker, dans laquelle nous allons installer Node.js, ainsi que les différentes dépendances de notre projet.

La première chose que vous devez faire est de créer un fichier nommé "Dockerfile", puis de définir dans celui-ci l'image que vous allez utiliser comme base, grâce à l'instruction**FROM** qui télécharge une image de base pour notre conteneur , utilisons l’image alpine et sa version 3.14

L’image alpine et version 3.14 : **FROM alpine:3.14**

Une fois que cette commande est effectue on doit indiquer a notre conteneur une commande a effectuer

CMD[ “echo”, “Hello world “]

Donc on a la commande **echo** qui va imprimer sur la sortie standards **Hello World**

Ensuite il faut construire notre image avec la commande **docker build -t mon\_image .**

L’option -t pour renommer l’image et . pour indiquer que on construit l’image dans le dossier de reference. On peut voir notre image avec la commande **docker images**  et lancer notre conteneur avec la commande **docker run mon\_images**

## Utilisez des images grâce au partage sur le Docker Hub

Vous avez maintenant créé votre propre image via un Dockerfile, et vous souhaitez partager cette image avec vos collègues. Pour cela, il existe deux solutions :

soit vous partagez votre fichier Dockerfile à chacun de vos collègues, et vous leur demandez de créer eux-mêmes leur propre image avec un docker build  ;

soit vous envoyez votre image sur votre propre ****registry****.

### Créez votre image sur le Docker Hub

Vous allez créer votre image Docker sur le Docker Hub, pour la rendre accessible de façon publique à l'ensemble de la communauté.

Pour cela, vous devez vous rendre sur le site [https://hub.docker.com/](https://hub.docker.com/" \t "https://openclassrooms.com/fr/courses/2035766-optimisez-votre-deploiement-en-creant-des-conteneurs-avec-docker/_blank) et vous connecter.

Puis, cliquez sur le lien **Create Repository.** Vous arrivez alors sur une page où vous devez saisir le nom de votre image, ainsi qu'une description.

Nous allons maintenant la publier sur le Docker Hub. Pour cela, nous devons utiliser plusieurs commandes.

Voici la première commande que vous allez utiliser :

➜ **docker tag nom\_image:latest YOUR\_USERNAME/nom\_repo:latest**

**Ou** si votre conteneur n’a pas de nom

➜ **docker tag id\_du\_conteneur username/nom\_repo:latest**

Vous pouvez maintenant exécuter la dernière commande nécessaire pour envoyer votre image vers le Docker Hub. Pour cela, vous allez exécuter la commande **docker push YOUR\_USERNAME/nom\_repo:latest.**

vous retournez maintenant sur la page Docker Hub de votre image, vous pourrez voir qu'il existe une première version de celle-ci.