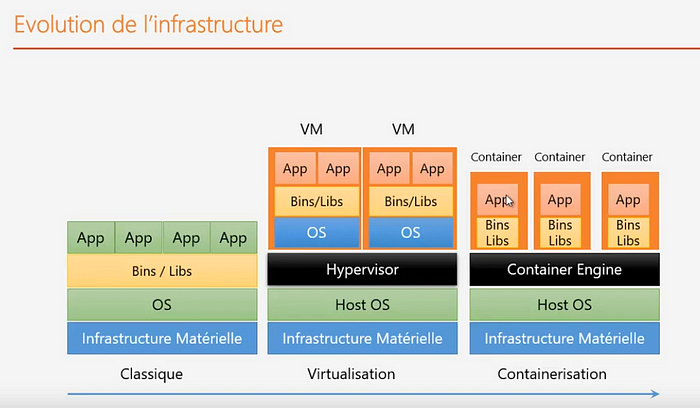
**Tout sur Docker: de la virtualisation au Containerisation. 1/2**

Dans le monde du développement logiciel, le déploiement d’applications constitue souvent un défi de taille. Les développeurs doivent s’assurer que l’application fonctionne correctement sur différents environnements et configurations système, ce qui peut entraîner des problèmes de compatibilité et de stabilité. C’est là qu’intervient Docker, une technologie révolutionnaire qui simplifie et accélère le déploiement d’applications de manière agile et efficace.

L’article explore les fondamentaux de Docker, une plateforme open-source permettant d’emballer des applications et leurs dépendances dans des conteneurs légers et portables. Le concept de conteneurisation est expliqué en détail, mettant en évidence les avantages tels que l’isolation des applications, la facilité de déploiement, et la portabilité des environnements.

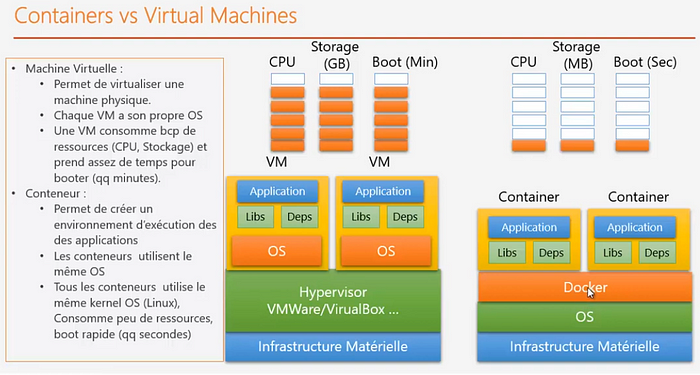
Ensuite on va détailler les principaux composants de Docker, notamment l’image Docker, qui est un package autonome contenant tout ce dont une application a besoin pour s’exécuter, et le registre Docker, qui sert de référentiel centralisé pour stocker et partager des images. L’article met en évidence la simplicité et la flexibilité de Docker, permettant aux développeurs de créer rapidement des environnements de développement cohérents et reproductibles.



Evolution de l’infrastructure

Une infrastructure classique, la virtualisation et la containerisation sont toutes des approches différentes pour gérer et organiser des ressources informatiques. Voici les différences entre ces trois concepts :

1. **Infrastructure classique :**Dans une infrastructure classique, les ressources informatiques (serveurs, réseaux, stockage, etc.) sont gérées de manière physique. Chaque application s’exécute sur un serveur dédié et utilise les ressources de ce serveur. Cela signifie que chaque application a besoin de son propre système d’exploitation et de ses propres dépendances logicielles. Cette approche peut être coûteuse en termes d’investissement matériel, de maintenance et de gestion.
2. **Virtualisation :** La virtualisation consiste à créer des machines virtuelles (VM) en exécutant plusieurs systèmes d’exploitation sur un même serveur physique. Chaque machine virtuelle est isolée et peut exécuter ses propres applications. La virtualisation permet de consolider les ressources matérielles, d’améliorer l’utilisation des serveurs et de faciliter la gestion des applications. Les machines virtuelles peuvent être déployées rapidement, migrées d’un serveur à un autre et redimensionnées en fonction des besoins. Les hyperviseurs (comme VMware, Hyper-V) sont utilisés pour gérer la virtualisation.
3. **Containerisation :** La containerisation est une méthode de virtualisation au niveau du système d’exploitation (OS). Les conteneurs sont des unités logicielles légères qui regroupent une application et toutes ses dépendances logicielles en un seul package. Différents conteneurs peuvent s’exécuter sur un même système d’exploitation, partageant le même noyau du système d’exploitation. Les conteneurs offrent une isolation efficace des ressources tout en permettant une flexibilité et une portabilité élevées des applications. Docker est l’un des outils de containerisation les plus populaires.



En résumé, dans une infrastructure classique, chaque application s’exécute sur un serveur physique distinct. Dans la virtualisation, plusieurs machines virtuelles s’exécutent sur un même serveur physique. Et dans la containerisation, plusieurs conteneurs s’exécutent sur un même système d’exploitation. La virtualisation offre une isolation plus forte entre les applications, tandis que la containerisation offre une flexibilité et une portabilité plus élevées.

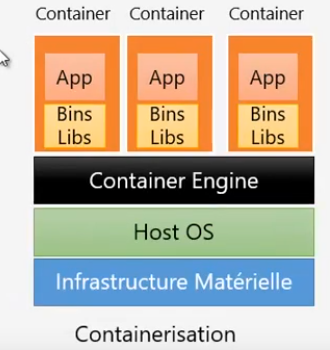
**C’est quoi un container ?**



Un conteneur est une unité logicielle légère et autonome qui regroupe une application et toutes ses dépendances logicielles nécessaires à son exécution. Les conteneurs sont isolés les uns des autres et partagent le même noyau du système d’exploitation hôte. Ils fournissent une méthode de déploiement portable et cohérente des applications, indépendamment de l’environnement sous-jacent.

Les conteneurs utilisent une technologie de virtualisation au niveau du système d’exploitation (OS) pour encapsuler l’application et toutes ses dépendances dans un environnement isolé. Cette encapsulation comprend les bibliothèques, les fichiers binaires, les variables d’environnement, les configurations et tout ce dont l’application a besoin pour s’exécuter correctement.

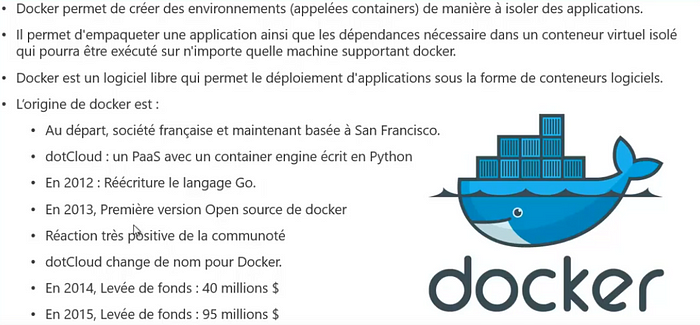
Les conteneurs sont conçus pour être légers et rapides à démarrer, car ils partagent le même noyau du système d’exploitation hôte, ce qui évite la surcharge liée au démarrage d’un système d’exploitation complet pour chaque application.



Les conteneurs sont généralement créés à l’aide d’outils de containerisation tels que Docker. Ces outils permettent de créer, de gérer et de déployer des conteneurs de manière simple et efficace. Ils fournissent également des fonctionnalités avancées telles que la mise en réseau entre conteneurs, la gestion des ressources et la mise à l’échelle horizontale des applications.

En résumé, un conteneur est une unité logicielle autonome qui encapsule une application avec toutes ses dépendances, offrant une portabilité, une isolation et une efficacité élevées dans l’exécution d’applications.

**C’est quoi docker ?**

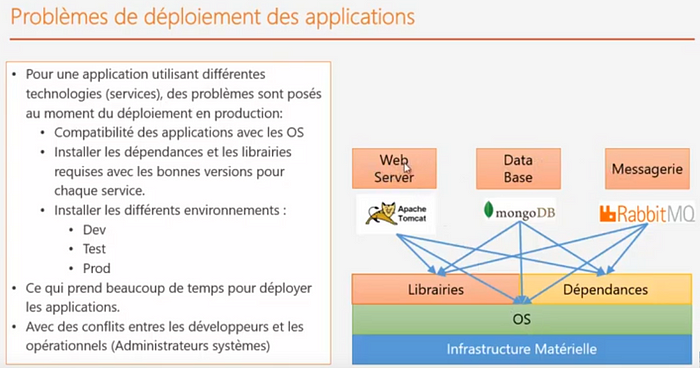


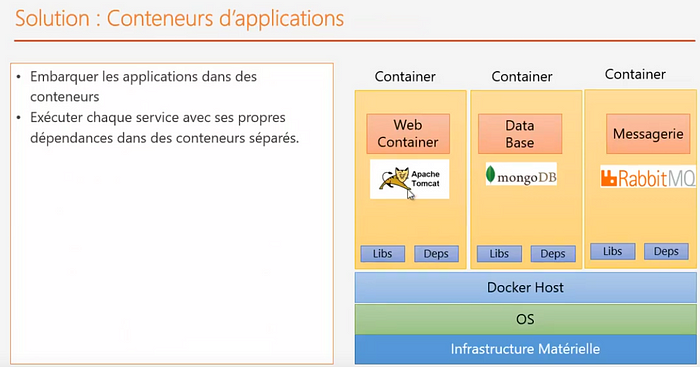
**Pourquoi docker ?**

Voici quelques raisons pour lesquelles Docker est largement adopté :

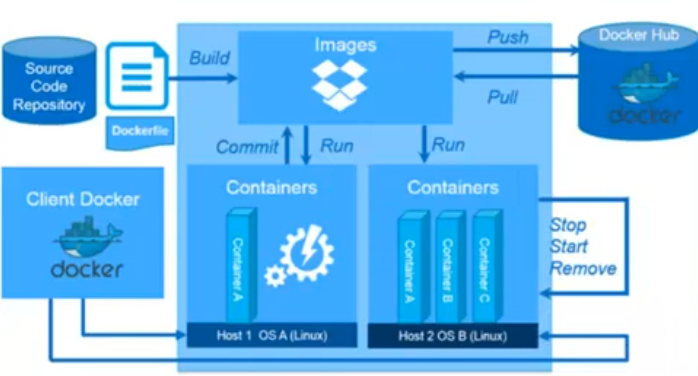
1. Isolation et portabilité : Docker permet d’isoler les applications et leurs dépendances dans des conteneurs autonomes. Cela garantit que chaque application s’exécute dans un environnement cohérent et isolé, sans interférence avec d’autres applications ou avec le système d’exploitation hôte. De plus, les conteneurs Docker sont portables, ce qui signifie qu’ils peuvent être exécutés sur différentes machines, environnements et infrastructures cloud sans nécessiter de modifications significatives.
2. Rapidité et efficacité : Les conteneurs Docker sont légers et rapides à démarrer. Ils partagent le même noyau du système d’exploitation hôte, ce qui réduit la surcharge et permet un démarrage rapide des applications. De plus, Docker utilise des couches d’images, ce qui permet une réutilisation efficace des composants communs entre différentes applications et réduit l’espace de stockage nécessaire.
3. Gestion des dépendances : Docker facilite la gestion des dépendances logicielles. Les images Docker contiennent toutes les bibliothèques, les fichiers binaires et les configurations nécessaires à une application spécifique. Cela élimine les problèmes de compatibilité et de configuration qui peuvent survenir lorsque les applications partagent le même système d’exploitation.
4. Évolutivité et orchestration : Docker s’intègre bien avec des outils d’orchestration tels que Docker Swarm et Kubernetes. Ces outils permettent de gérer facilement et de mettre à l’échelle horizontalement les conteneurs Docker sur plusieurs hôtes, offrant ainsi une haute disponibilité, une résilience et une élasticité des applications.
5. Écosystème et support : Docker dispose d’un vaste écosystème d’outils, de bibliothèques et de services qui facilitent l’intégration et l’utilisation de la containerisation dans divers scénarios. De plus, Docker bénéficie d’une large communauté d’utilisateurs et de développeurs qui fournissent un support, des ressources et des mises à jour régulières.

En résumé, Docker est utilisé pour sa facilité d’utilisation, sa portabilité, son isolation, sa rapidité et son écosystème étendu. Il permet aux développeurs et aux équipes informatiques de créer, de déployer et de gérer des applications de manière efficace, cohérente et évolutive.



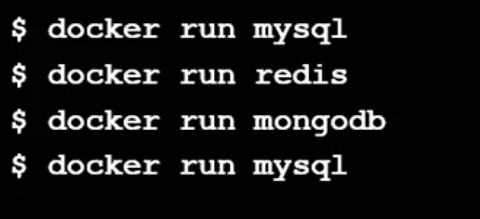


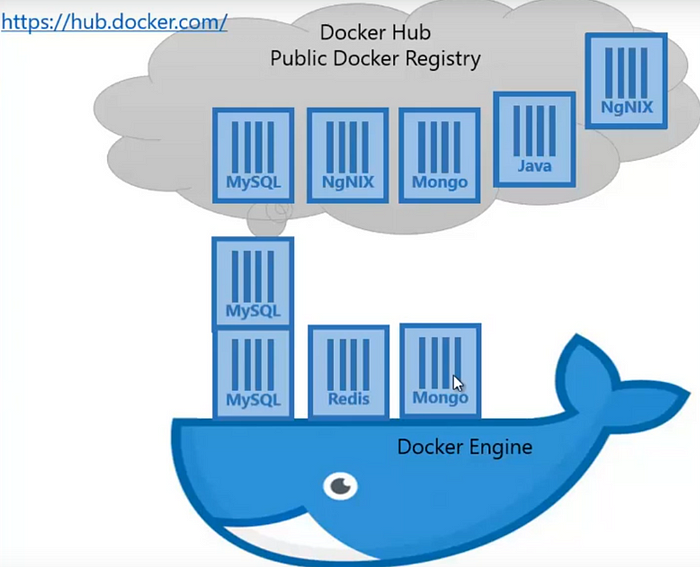
**Architecture globale de docker**



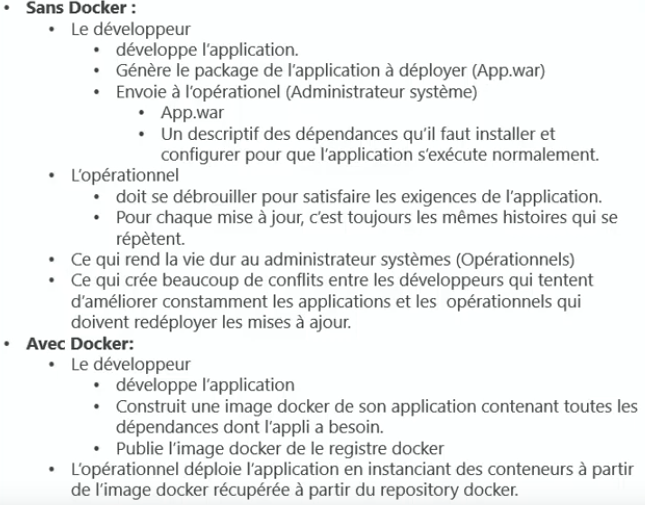


Exemple:





**Docker contribue à instaurer la culture DevOps**





**Les éditions de docker**

Docker propose deux éditions principales : Docker Community Edition (CE) et Docker Enterprise Edition (EE).

1. Docker Community Edition (CE) : Cette édition de Docker est destinée aux développeurs et aux petites équipes. Elle est gratuite et disponible pour différentes plateformes, notamment Windows, macOS et Linux. Docker CE comprend toutes les fonctionnalités essentielles de la containerisation, telles que le moteur Docker, les outils de ligne de commande et les fonctionnalités de gestion des conteneurs. Elle est idéale pour les projets de développement et de test, ainsi que pour les déploiements individuels ou à petite échelle.
2. Docker Enterprise Edition (EE) : Docker Enterprise Edition est une version commerciale de Docker destinée aux grandes entreprises et aux environnements de production. Elle propose des fonctionnalités avancées telles que la sécurité renforcée, la gestion des politiques, l’intégration d’outils d’orchestration (comme Kubernetes) et la prise en charge de la mise à l’échelle horizontale. Docker EE est disponible en différentes éditions, notamment Basic, Standard et Advanced, avec des fonctionnalités et des niveaux de support différenciés. Docker EE offre également une gestion centralisée des conteneurs et une prise en charge à long terme.

Il est important de noter que Docker a annoncé en 2020 un partenariat avec Mirantis, une société spécialisée dans les solutions de cloud et de conteneurisation. Ils ont convenu que Mirantis serait responsable de la prise en charge et de la vente de Docker Enterprise Edition (EE). Par conséquent, les éditions Docker peuvent avoir évolué depuis lors. Je vous recommande de vérifier les informations les plus récentes sur le site officiel de Docker pour obtenir les détails et les mises à jour les plus récentes sur les éditions de Docker.

la on passe à la partie pratique :)

**Installation**

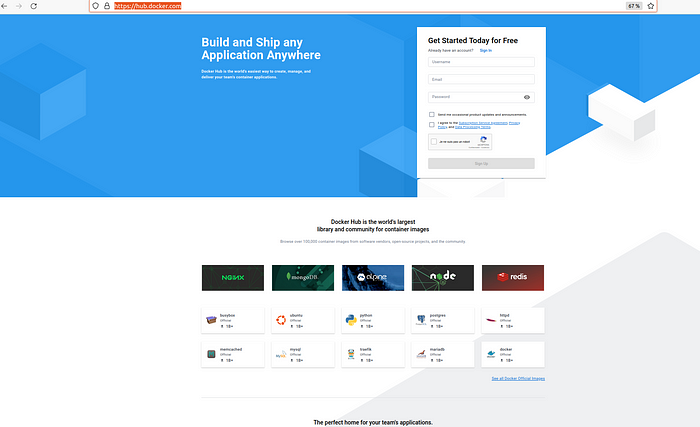


**Vérification de la version docker installée**

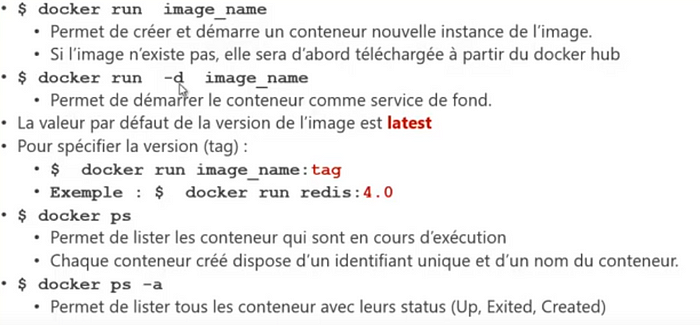
https://miro.medium.com/v2/resize:fit:509/1*ZWs7NueNkRPsxaRRdy9QBQ.png

**Explorer docker hub**

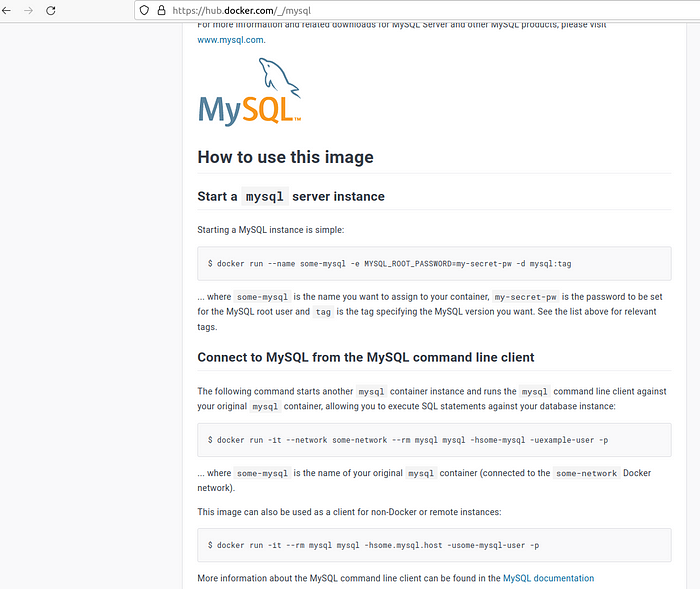
<https://hub.docker.com/>



**Commandes de base docker**



exemple d’une image mysql:



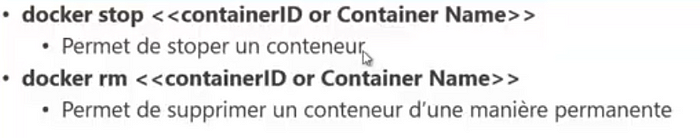
Lister les images docker présentes dans le host local de docker:



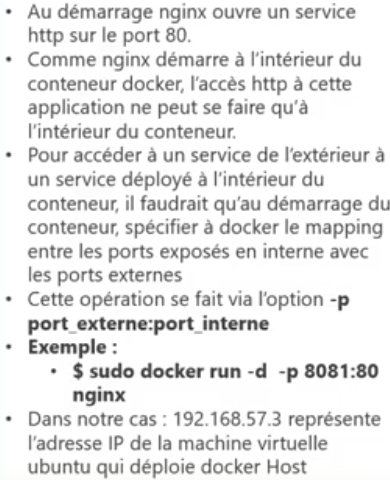
https://miro.medium.com/v2/resize:fit:700/1*zC6ae_itHSl6U3KBxakDhQ.png

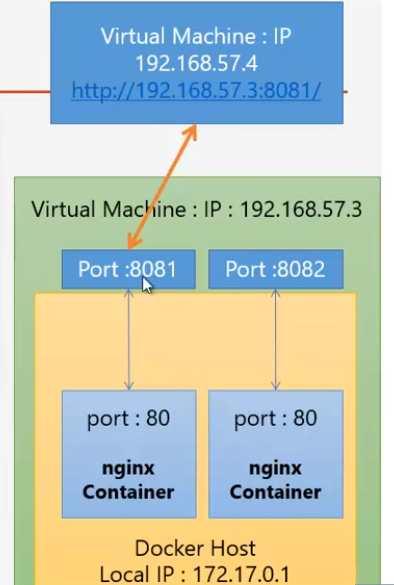
PS: pour supprimer une images il faut supprimer ces conteneurs

et pour supprimer un container il faut l’arrêter d’abord



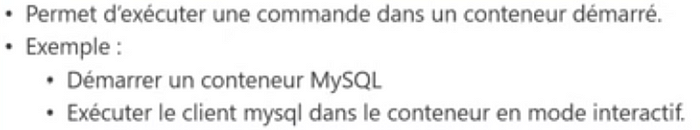
**Mapper les ports pour un accès externe:**







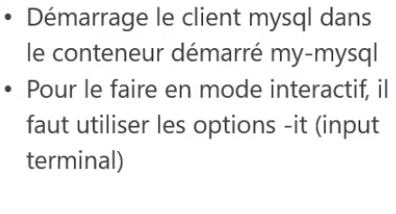
**Exec Command**



exemple:



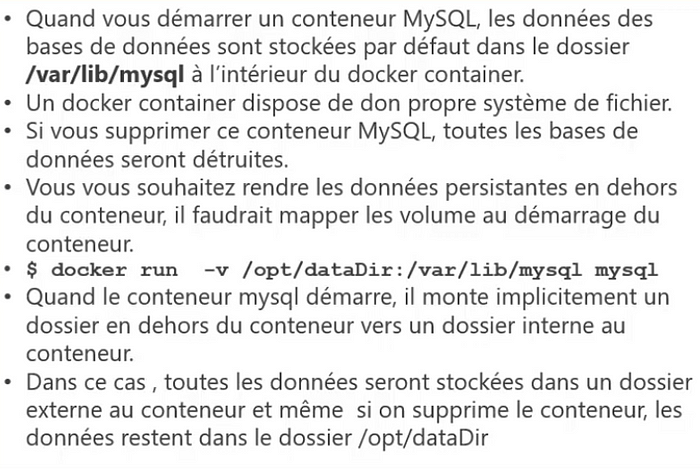
$ docker run --name some-mysql -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql:tag

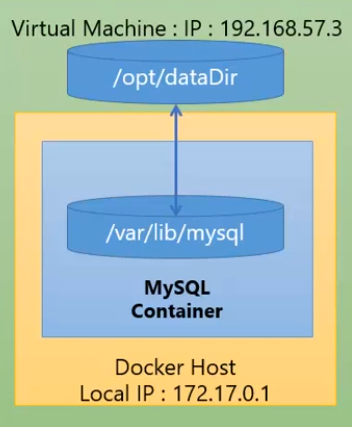


$ docker run -it --network some-network --rm mysql mysql -hsome-mysql -uexample-user -p

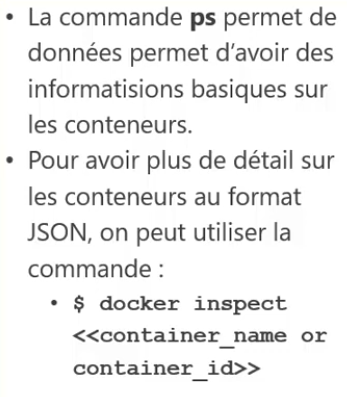


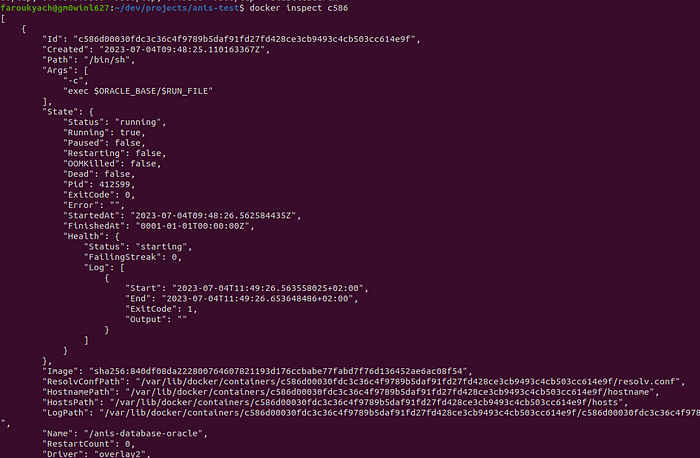
**Volume mapping**



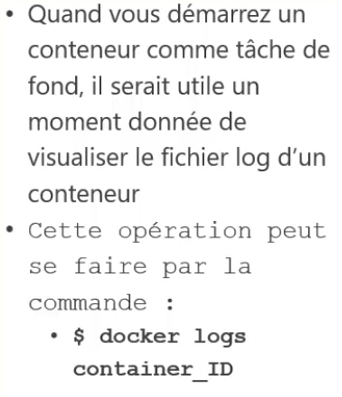


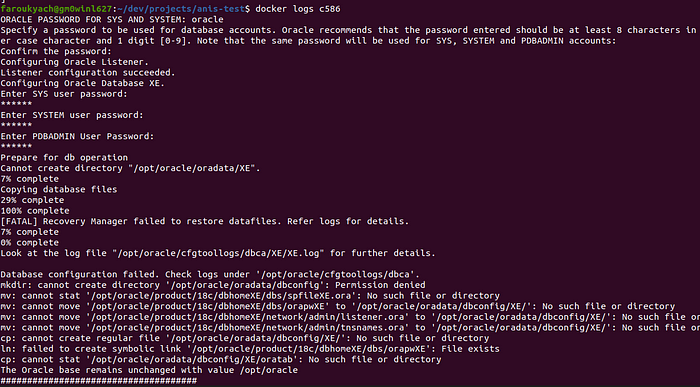
**Inspecter un container**





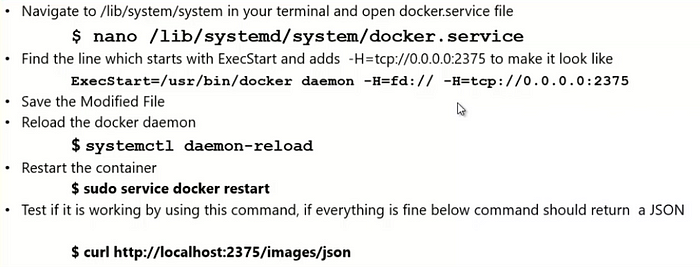
**Container Logs**





**Configurer un Rest Api a partir d’une image docker**

https://miro.medium.com/v2/resize:fit:700/1*MKHyMSohJRq0DRyZWt0pHg.png



pour plus de détail:

<https://docs.docker.com/engine/api/v1.42/>

# Comment créer votre propre image docker avec Dockerfile 2/2

Dans l’article précédent, nous avons abordé les bases de Docker, ce qu’est Docker, son architecture, et les principaux avantages qu’il offre. Nous avons expliqué également les concepts clés tels que les images, les conteneurs et les registres .

Ici, nous guiderons les lecteurs à travers les étapes d’installation de Docker sur différents systèmes d’exploitation, y compris Linux, Windows et macOS. Nous fournirons des instructions détaillées et des commandes spécifiques pour chaque plateforme.

Nous aborderons également la création d’une image docker et les meilleures pratiques pour choisir une image de base appropriée en fonction des besoins de l’application.

# Installer docker client:

## 1. Linux

sudo apt update  
sudo apt install docker.io

Assurez-vous également d’ajouter votre utilisateur au groupe docker pour pouvoir utiliser Docker sans sudo :

sudo usermod -aG docker $USER

Après cette étape, vous devez vous déconnecter et vous reconnecter pour que les modifications prennent effet.

## 2. Windows

Docker Desktop est généralement utilisé pour les systèmes Windows. Vous pouvez télécharger Docker Desktop depuis le site officiel de Docker et suivre les instructions d’installation : <https://www.docker.com/products/docker-desktop>

Assurez-vous que votre version de Windows prend en charge la virtualisation matériel (Hyper-V) car Docker Desktop utilise Hyper-V pour exécuter des conteneurs.

## 3. macOS

Pour macOS, Docker Desktop est également la méthode d’installation recommandée. Vous pouvez télécharger Docker Desktop depuis le site officiel de Docker et l’installer en suivant les instructions : <https://www.docker.com/products/docker-desktop>

Après avoir terminé l’installation, vous pouvez vérifier que Docker est bien installé en exécutant la commande suivante :

docker --version

Cela devrait afficher la version de Docker installée sur votre système. Félicitations, vous avez maintenant le client Docker installé et prêt à être utilisé .

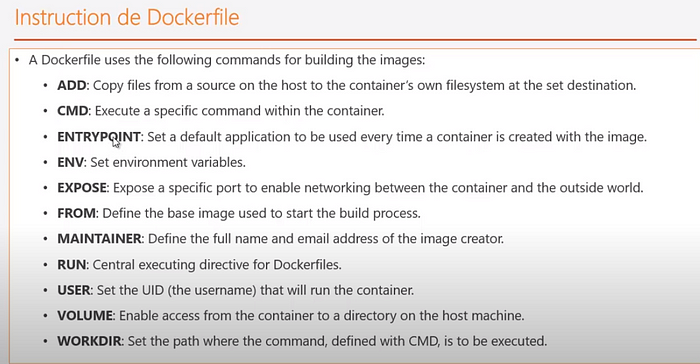
# créer mon propre image docker

Pour créer votre propre image Docker, vous devez suivre les étapes suivantes :

1. **Créez un fichier Dockerfile :** Le Dockerfile est un fichier texte qui définit les instructions pour construire votre image Docker. Il spécifie la configuration de l’image, les dépendances, les commandes à exécuter, etc. Vous pouvez utiliser un éditeur de texte pour créer un fichier Dockerfile. (il faut mettre le fichier dockerfile dans la racine de votre application)

**2. Choisissez une image de base :**Une image de base est une image existante sur Docker Hub ou un registre privé que vous utiliserez comme point de départ pour votre image personnalisée. Sélectionnez une image de base appropriée qui correspond à votre application ou à votre environnement. Par exemple, vous pouvez choisir une image de base Ubuntu, Alpine Linux, etc.

**3. Écrivez les instructions Dockerfile :** Dans le Dockerfile, utilisez les instructions appropriées pour décrire les étapes de construction de votre image. Les instructions courantes incluent FROM, RUN, COPY, ADD, ENV, WORKDIR, etc. Vous pouvez exécuter des commandes, copier des fichiers, définir des variables d'environnement, etc.



dockerfile instructions

**4. Construisez l’image :** Une fois que vous avez terminé d’écrire le Dockerfile, ouvrez un terminal ou une invite de commande, naviguez vers le répertoire contenant le Dockerfile et exécutez la commande docker build. Par exemple :

docker build -t nom\_de\_votre\_image:tag .

Assurez-vous de remplacer nom\_de\_votre\_image par le nom que vous souhaitez donner à votre image et tag par une étiquette spécifique (comme latest, v1.0, etc.). Le point . à la fin de la commande indique que le contexte de construction se trouve dans le répertoire actuel.

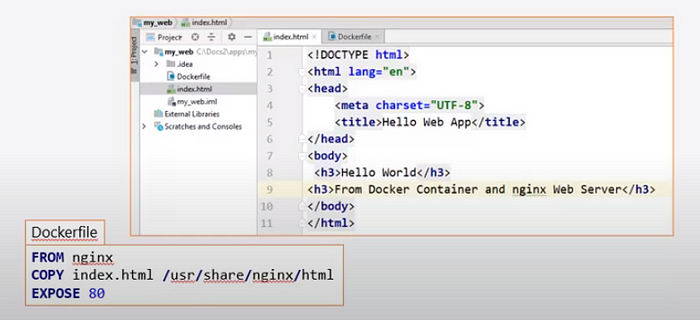
**5. Vérifiez votre nouvelle image :**Une fois que la construction est terminée, vous pouvez vérifier que votre image est bien créée en utilisant la commande docker images.

**6. Exécutez un conteneur avec votre image :**Enfin, pour tester votre nouvelle image, vous pouvez créer et exécuter un conteneur à partir de celle-ci en utilisant la commande docker run. Par exemple :

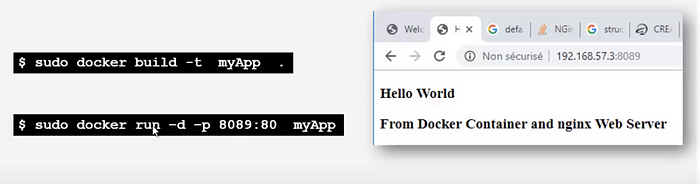
docker run -d --name mon\_conteneur nom\_de\_votre\_image:tag

N’oubliez pas que chaque instruction dans le Dockerfile crée une nouvelle couche dans l’image. Pour minimiser la taille de l’image, essayez de regrouper les commandes et de nettoyer les fichiers temporaires inutiles.

## Voici un exemple:

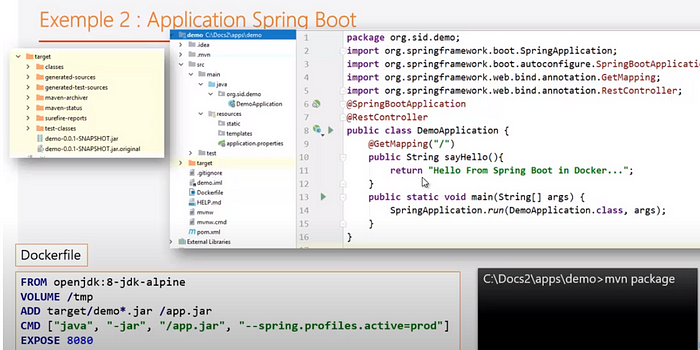


Création d’une image docker avec dockerfile



build & run

## un autre exemple avec une application spring boot



C’est ainsi que vous pouvez créer votre propre image Docker en utilisant un Dockerfile. Une fois que vous êtes satisfait de votre image, vous pouvez la télécharger sur Docker Hub ou un autre registre pour la partager avec d’autres utilisateurs.

# Docker compose

