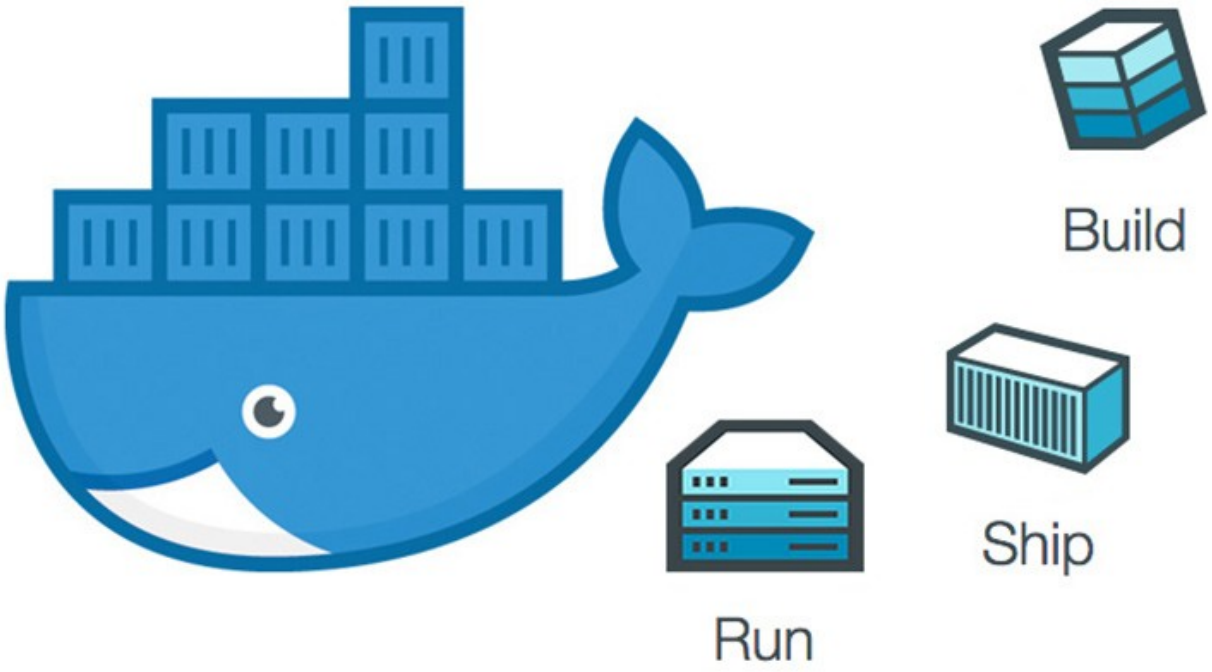
La Théorie



Docker : présentation générale

Docker est un projet open source permettant d'automatiser le déploiement d'applications en tant que conteneurs portables et autonomes exécutables sur le cloud ou localement. Docker est également une entreprise qui développe et diffuse cette technologie, en collaboration avec des fournisseurs de services cloud, Linux et Windows.

Docker : structure

La plateforme open source repose sur trois composants de base : pour faire fonctionner les conteneurs, les utilisateurs ont simplement besoin du moteur Docker et des images Docker spéciales qui peuvent être obtenues à partir du Docker Hub ou créés par eux-mêmes.

Images Docker

Les conteneurs Docker sont basés sur des images. Une image est un modèle en lecture seule avec des instructions pour créer un conteneur Docker. Souvent, une image est basée sur une autre image, avec quelques personnalisations supplémentaires. Par exemple, vous pouvez créer une image basée sur l'image CentOS, mais qui installe le serveur Web Apache et votre application,

ainsi que les détails de configuration nécessaires pour exécuter votre application.
Vous pouvez créer vos propres images ou n'utiliser que celles créées par d'autres et publiées dans un registre. Pour créer votre propre image, vous créez un Dockerfile avec une syntaxe simple pour définir les étapes nécessaires pour créer l'image et l'exécuter. Chaque instruction d'un Dockerfile crée un calque dans l'image. Lorsque vous modifiez le Dockerfile et reconstruisez l'image, seuls les calques qui ont été modifiés sont reconstruits. Cela fait partie de ce qui rend les images si légères, petites et rapides par rapport aux autres technologies de virtualisation.

Le Docker-Hub

Un registre Docker stocke les images Docker. Docker Hub est un registre public que tout le monde peut utiliser, et Docker est configuré pour rechercher des images sur Docker Hub par défaut. Vous pouvez même exécuter votre propre registre privé.
Lorsque vous utilisez les commandes docker pull ou docker run, les images requises sont extraites de votre registre configuré. Lorsque vous utilisez la commande docker push, votre image est poussée vers votre registre configuré.

Le Docker-Engine

Le cœur du projet Docker est le moteur Docker (Docker-Engine). Il s'agit d'une application client-serveur open source.
L'architecture de base du Docker-Engine peut être divisée en trois composants : un daemon avec des fonctions serveur, une interface de programmation (API) basée sur le paradigme de programmation REST (Representational State Transfer) et le terminal du système d'exploitation (Command-Line Interface, CLI) comme interface utilisateur (client).

- **Le daemon Docker**: le Docker-Engine utilise un processus daemon comme serveur. Le daemon docker s'exécute en arrière-plan sur le système hôte et sert à contrôler le moteur Docker de manière centralisée. Dans cette fonction, il crée et gère toutes les images, conteneurs ou réseaux.
- **Le REST-API**[1]: le REST-API spécifie un ensemble d'interfaces qui permettent à d'autres programmes de communiquer et de donner des instructions au daemon Docker. L'un de ces programmes est le terminal du système d'exploitation.
- **Le terminal**: Docker utilise le terminal du système d'exploitation comme programme client. Il interagit avec le

1: une interface de programmation d'application qui fait appel à des requêtes HTTP pour obtenir (GET), placer (PUT), publier (POST) et supprimer (DELETE) des données

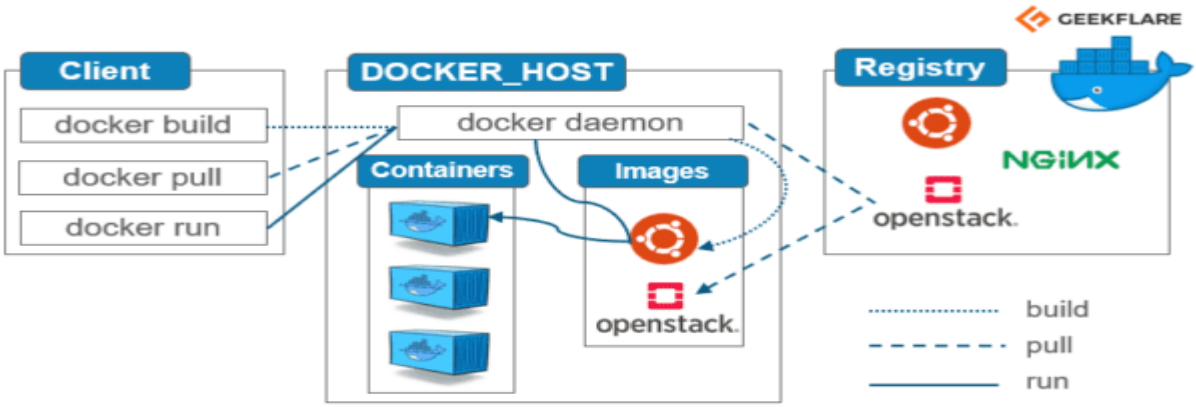
daemon via l'API REST et permet aux utilisateurs de le contrôler grâce aux scripts ou aux entrées utilisateur.

Docker permet aux utilisateurs de démarrer, d'arrêter et de gérer les conteneurs logiciels directement à partir du terminal. Le daemon est adressé à l'aide de la commande Docker et d'instructions comme build (créer), pull (télécharger) ou run (démarrer).

Le client et le serveur peuvent se trouver sur le même système. Alternativement, les utilisateurs ont la possibilité d'accéder à un daemon Docker sur un autre système. Selon le type de connexion à établir, la communication entre le client et le serveur s'effectue via REST-API, les sockets UNIX ou une interface réseau.

Interaction des différents composants du Docker

Le graphique suivant illustre l'interaction des différents composants du Docker en utilisant les commandes docker build, docker pull et docker run:



La commande docker build indique au Docker-Daemon de créer une image. Pour cela, un Dockerfile adapté doit être disponible. Si vous ne voulez pas créer l'image vous-même, mais que vous préférez la charger à partir d'un référentiel dans le Docker-Hub, utilisez la commande docker pull. Si le daemon ne peut pas trouver l'image, il lance automatiquement une extraction du Registry.

Activité n°1: Installation de Docker Engine

Activité de lecture

Extrait 1

Source Digital Guide INOS

Initialement, Docker était exclusivement utilisé sur les distributions Linux, la version actuelle du moteur de conteneur (Container-Engine) est largement indépendante de la plateforme. Des packages d'installation sont disponibles pour Microsoft Windows et macOs ainsi que les services Cloud comme Amazon Web Services (AWS) et Microsoft Azure. Les distributions Linux prises en charge incluent:

- CentOS
- Debian
- Fedora
- Oracle Linux
- Red Hat Enterprise Linux
- Ubuntu
- openSUSE
- SUSE Linux Enterprise

Il y a aussi des distributions de Dockers maintenues par les communautés pour:

- Arch Linux
- CRUX Linux
- Gentoo Linux

Extrait 2

Source Docker docs

Docker Engine has three types of update channels, stable, test, and nightly:

- *The Stable channel* gives you latest releases for general availability.
- *The Test channel* gives pre-releases that are ready for testing before general availability (GA).
- The Nightly channel gives you latest builds of work in progress for the next major release.

Stable

Year-month releases are made from a release branch diverged from the master branch. The branch is created with format <year>.<month>, for example 18.09. The year-month name indicates the earliest possible calendar month to expect the release to be generally available. All further patch releases

are performed from that branch. For example, once v18.09.0 is released, all subsequent patch releases are built from the 18.09 branch.

Test

In preparation for a new year-month release, a branch is created from the master branch with format YY.mm when the milestones desired by Docker for the release have achieved feature-complete. Pre-releases such as betas and release candidates are conducted from their respective release branches. Patch releases and the corresponding pre-releases are performed from within the corresponding release branch.

Nightly

Nightly builds give you the latest builds of work in progress for the next major release. They are created once per day from the master branch with the version format:

```
0.0.0-YYYYmmddHHMMSS-abcdefabcdef
```

where the time is the commit time in UTC and the final suffix is the prefix of the commit hash, for example 0.0.0-20180720214833-f61e0f7.

These builds allow for testing from the latest code on the master branch.

Note: No qualifications or guarantees are made for the nightly builds.

Installation de Docker Engine sur CentOS

Dans ce qui suit, nous allons illustrer le processus d'installation du moteur Docker en utilisant la distribution Linux populaire CentOS comme exemple. Des instructions d'installation détaillées pour les autres plateformes se trouvent dans la documentation de Docker.

Désinstaller les anciennes versions

Les anciennes versions de Docker étaient appelées **docker** ou **docker-engine**. S'ils sont installés, désinstallez-les, ainsi que les dépendances associées.

```
$ sudo yum remove docker \
                                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
```

```
docker-engine
```

NB :

- Ce n’est pas un problème si yum signale qu’aucun de ces packages n’est installé.
- Le contenu de **/var/lib/docker/**, y compris les images, les conteneurs, les volumes et les réseaux, est conservé.
- Le package Docker Engine s'appelle désormais docker-ce.

Installer à partir d'un package

vous pouvez télécharger le fichier **.rpm** de votre version et l’installer manuellement. Vous devez télécharger un nouveau fichier chaque fois que vous souhaitez mettre à niveau Docker Engine.

1. Allez sur **https://download.docker.com/linux/centos/** et choisissez votre version de CentOS. Accédez ensuite à **x86_64/stable/Packages/** et téléchargez le fichier **.rpm** correspondant à la version de Docker que vous souhaitez installer.
2. Installez Docker Engine, en modifiant le chemin ci-dessous par le chemin où vous avez téléchargé le package Docker.

```
$ sudo yum install /path/to/package.rpm
```

Docker est installé mais pas démarré. Le groupe Docker est créé, mais aucun utilisateur n'est ajouté au groupe.

3.Démarrer Docker

```
$ sudo systemctl start docker
```

4.Vérifiez que Docker Engine est correctement installé en exécutant l'image hello-world.

```
$ sudo docker run hello-world
```

Cette commande télécharge une image de test et l'exécute dans un conteneur. Lorsque le conteneur s'exécute, il imprime un message d'information et se ferme.

NB : Mise à niveau du Docker Engine

Pour mettre à niveau Docker Engine, téléchargez le fichier de package plus récent et répétez la procédure d'installation, en utilisant **yum -y upgrade** au lieu de **yum -y install** et en pointant vers le nouveau fichier.

Installer à partir de repository

Avant d'installer Docker Engine pour la première fois sur une nouvelle machine hôte, vous devez configurer le repository Docker. Ensuite, vous pouvez installer et mettre à jour Docker à partir de ce dernier.

1. configurez le repository Docker

```
$ sudo yum install -y yum-utils
$ sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

2.Installer Docker Engine

```
$ sudo yum install docker-ce docker-ce-cli containerd.io
```

Docker est installé mais pas démarré. Le groupe Docker est créé, mais aucun utilisateur n'est ajouté au groupe.

3.Démarrez Docker

```
$ sudo systemctl start docker
```

4.Vérifiez que Docker Engine est correctement installé en exécutant l'image hello-world.

```
$ sudo docker run hello-world
```

Pour installer une version spécifique de Docker Engine, répertoriez les versions disponibles dans le dépôt, puis sélectionnez et installez la version choisie.

1. Répertoriez les versions disponibles dans le dépôt

```
$ yum list docker-ce --showduplicates | sort -r
docker-ce.x86_64  3:18.09.1-3.el7                     docker-ce-stable
docker-ce.x86_64  3:18.09.0-3.el7                     docker-ce-stable
docker-ce.x86_64  18.06.1.ce-3.el7                    docker-ce-stable
docker-ce.x86_64  18.06.0.ce-3.el7                    docker-ce-stable
```

2. Installez la version choisie (le nom de package complet est le nom du package (docker-ce) + la chaîne de version (2ème colonne) commençant au premier deux-points (:), jusqu'au premier tiret, séparé par un tiret (-). Par exemple, docker-ce-18.09.1.

```
$ sudo yum install docker-ce-<VERSION_STRING> docker-ce-cli-<VERSION_STRING>
\ containerd.io
```

Installer à partir d’un script

Docker fournit des scripts pratiques sur get.docker.com et test.docker.com pour installer rapidement et de manière non interactive des versions de pointe et de test de Docker Engine-Community dans des environnements de développement. Le code source des scripts se trouve dans le docker-install repository.

L’exemple ci-dessous utilise le script sur get.docker.com pour installer la dernière version de Docker Engine - Community sur Linux.

```
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh
```

- -f ou –fail: fail silently(no output at all) on server errors
- -L ou --location: location
- -s: --silent

- -S: --show error when used with -s it makes curl show an error message if it fails

Post installation:Gérer Docker en tant qu'utilisateur non root

Si vous souhaitez utiliser Docker en tant qu'utilisateur non root, vous devez maintenant envisager d'ajouter votre utilisateur au groupe «docker» avec quelque chose comme:

```
sudo usermod -aG docker your-user
```

Désinstaller Docker Engine

1. Désinstaller Docker Engine, CLI et Containerd packages

```
$ sudo yum remove docker-ce docker-ce-cli containerd.io
```

2. Les images, conteneurs, volumes ou fichiers de configuration personnalisés sur votre hôte ne sont pas automatiquement supprimés. Pour supprimer toutes les images, conteneurs et volumes:

```
$ sudo rm -rf /var/lib/docker
```

Activité n°2: Premiers Pas

Premiers pas avec les images et les conteneurs

1. Afficher la version de Docker installée

```
$docker version
```

2. Affichez la liste des commandes Docker

```
$docker
```

3. Listez les images présentes sur votre machine.

```
$docker images ou $docker image ls
```

4. Listez les conteneurs actifs sur votre machine

```
$docker ps
```

5. Connectez vous au Docker hub et vérifiez l'existence d'une image nommée alpine(c'est une image Docker minimale basée sur Alpine Linux avec un package index complet et seulement 5 Mo de taille)

6.Exécutez la commande $docker *run alpine:latest* puis exécutez la commande $docker *im*ages? Que se passe t-il si vous lancez un conteneur sans avoir l'image sur votre machine?

L'image sera téléchargée automatiquement à partir de Docker hub.

7. Exécutez les commandes $docker ps et $docker ps -a? Qu'en déduisez-vous?

Le conteneur lancé à partir de l'image alpine ne reste en exécution que quelques secondes. En effet, on lui avait rien demandé.L'état actuel du conteneur est Exit.

8. Lancez un nouveau conteneur nommé M2 à partir de l'image alpine mais veuillez à garder votre conteneur en exécution.

```
$docker run -di --name  M2 alpine:latest
```

9. Vérifiez l'état de votre conteneur.

```
$docker ps
```

10. Exécutez un shell bash interactif sur M2.

```
$docker exec -ti M2 sh
```

11.Exécutez la commande ps -e ? Y-a-t-il un systemd?Pourquoi?

Le conteneur n'intègre pas de noyau, il s'appuie directement sur le noyau de l'ordinateur sur lequel il est déployé.

12. Quittez le shell exécuté sur le conteneur M2.

```
$docker stop M2
```

13. Supprimer M2

```
$docker rm M2
```

14. Supprimez l'image alpine

```
$docker rmi alpine
```

Lancer un conteneur de serveur web Nginx

1. Supprimez tous les conteneurs.

```
$docker rm -f $( docker ps -aq)
```

2. Lancez un conteneur nommé web à partir de la dernière image Nginx sur le Docker hub tout en exposant le port 80 du conteneur vers le port 8080 de la machine locale.

```
$docker run -tid -p 8080:80  --name web  nginx:latest
```

3. Vérifiez l’état de votre conteneur

```
$docker ps
```

à partir d’un navigateur, saisissez dans la barre d’adresse 127.0.0.1:8080

4. Affichez l’intégralité des propriétés de votre conteneur.

```
$docker inspect web
```

5. Identifiez l’adresse IP de votre conteneur.

```
"IPAddress": "172.17.0.2"
```

6. à partir d’un navigateur, saisissez dans la barre d’adresse l’adresse IP de votre conteneur.

Activité n°3: Les volumes Docker

Afin de comprendre ce qu'est un volume Docker, nous devons d'abord préciser le fonctionnement normal du système de fichiers dans Docker.
En effet, une image Docker se compose d'un ensemble de layers (calques) en lecture seule. Lorsque vous lancez un conteneur à partir d'une image, Docker ajoute au sommet de cette pile de layers un nouveau layer en lecture-écriture. Docker appelle cette combinaison de couches un "Union File System".

Comment le moteur Docker gère les modifications de vos fichiers au sein de votre conteneur?

- Lors d'une modification de fichier, Docker crée une copie depuis les couches en lecture seule vers le layer en lecture-écriture.
- Lors d'une création de fichier, Docker crée le fichier que sur le layer en lecture-écriture, et ne touche pas au layer en lecture seule.
- Lors d'une suppression de fichier, Docker supprime le fichier que sur le layer en lecture-écriture, et si il est déjà existant dans le layer en lecture seule alors il le garde.

Les données dans le layer de base sont en lecture seule, elles sont donc protégées et intactes par toutes modifications, seul le layer en lecture-écriture est impacté lors de modifications de données.

Lorsqu'un conteneur est supprimé, le layer en lecture-écriture est supprimé avec. Cela signifie que toutes les modifications apportées après le lancement du conteneur auront disparus avec.

Les volumes dans les conteneurs Docker autorisent les données persistantes et le partage des données de la machine hôte dans un conteneur.

Monter un volume persistant et stocker vos données

Après un redémarrage, aucun conteneur ne se trouve en fonctionnement.

1.Démarrez le conteneur web

```
$docker start web
```

2. Exécutez un shell bash interactif sur le conteneur web.

```
$docker exec -ti web sh
```

3.Installer dans votre conteneur l’éditeur nano

```
#apt-get update
#apt-get install nano
```

4. Éditez la page /usr/share/nginx/html/index.html

```
#nano  /usr/share/nginx/html/index.html
[...]
<body>
<h1>Welcome to Docker World!</h1>
[…]
#exit
```

5. A partir d'un navigateur, saisissez dans la barre d'adresse 127.0.0.1:8080.

6. Arrêtez votre conteneur web

```
$docker stop web
```

7. Démarrez votre conteneur web puis à partir d'un navigateur, saisissez dans la barre d'adresse 127.0.0.1:8080. Que remarquez vous?

```
$docker start web
```

---

8. Supprimez le conteneur web puis le recréez une autre fois.

```
$docker rm  -f web
$docker run -tid -p 8080:80  --name web  nginx:latest
```

9. A partir d'un navigateur, saisissez dans la barre d'adresse 127.0.0.1:8080. Que remarquez vous?

---

10. Supprimez le conteneur web.

```
$docker rm  -f web
```

10. Créez sur la machine locale un répertoire */docker/data/nginx*

```
$sudo mkdir -p /docker/data/nginx
```

11. Lancer un conteneur web1 à partir de l'image nginx avec le volume */docker/data/nginx qui sera partagé avec le répertoire* /usr/share/nginx/html de web1.

```
$docker run -tid -p 8080:80  --name web1  -v /docker/data/nginx:/usr/share/nginx/html nginx:latest
```

12. A partir d'un navigateur, saisissez dans la barre d'adresse 127.0.0.1:8080. Que remarquez vous?

---

13. Dans le répertoire */docker/data/nginx de votre machine locale créez un fichier index.html*

```
$ sudo nano /docker/data/nginx/index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Coucou</title>
  </head>
  <body>
Docker volume
</body>
```

```
</html>
```

14. A partir d'un navigateur, saisissez dans la barre d'adresse 127.0.0.1:8080. Que remarquez vous?

---

15. Supprimez le conteneur web1

```
$docker rm  -f web1
```

16. Lancer un conteneur web2 à partir de l'image nginx avec le volume */docker/data/nginx qui sera partagé avec le répertoire* /usr/share/nginx/html de web2.

```
$docker run -tid -p 8080:80  --name web2  -v /docker/data/nginx:/usr/share/nginx/html nginx:latest
```

17. A partir d'un navigateur, saisissez dans la barre d'adresse 127.0.0.1:8080. Que remarquez vous?

La commande docker volume

1. Exécutez $docker volume et saisissez son rôle, sa syntaxe et le rôle de chacune de ces commandes.

```
$docker   volume
Usage:docker volume COMMAND
Manage volumes
Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls            List volumes
  prune         Remove all unused local volumes
  rm            Remove one or more volumes
Run 'docker volume COMMAND --help' for more information on a command.
```

2. Créez un volume nommé monvolume.

```
$docker   volume  create monvolume
```

3. Vérifiez l'existence de votre volume.

```
$docker   volume  ls | grep monvolume
```

4. Affichez les détails de votre volume et localisez son point de montage.

```
$docker   volume  inspect  monvolume
[
    {
        "CreatedAt": "2020-06-04T15:15:21+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/500000.500000/volumes/monvolume/_data",
```

```
        "Name": "monvolume",
        "Options": null,
        "Scope": "local"
    }
]
```

5. Exécutez la commande ci dessous et en déduire son rôle:

```
$docker run -tid --name web -p 8080:80 --mount
source=monvolume,target=/usr/share/nginx/html nginx:latest
```

---

6. Éditez le fichier index.html qui se trouve dans le point de montage de votre volume.

7 . A partir d'un navigateur, saisissez dans la barre d'adresse 127.0.0.1:8080. Que remarquez vous?

---

8. Essayez de supprimer votre volume avec la commande. Est-ce possible?

```
$docker volume rm monvolume.
$docker volume rm monvolume
Error response from daemon: remove monvolume: volume is in use - [……]
```

9. Que faut-il faire?

```
$docker rm web
$docker volume rm monvolume
```

Activité n°4: Les variables d'environnement

1. Lancez un conteneur nommé testenvvar à partir de la dernière image ubuntu en lui transmettant la variable MYVAR qui vaut 123456.

```
$docker run -tdi --name testenvvar --env MYVAR="123456" ubuntu:latest
```

2. Exécutez un shell bash interactif sur votre conteneur.

```
$docker exec -ti testenvvar sh
```

3. Listez les variables d'environnement et vérifiez que la variable MYVAR est bien définie

```
# env
HOSTNAME=38a05557ebf1
HOME=/root
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
MYVAR=123456
PWD=/
```

3. Quittez le shell exécuté sur votre conteneur et créez un fichier file-var.txt

```
nano file-var.txt
MYPASSWORD="M2cloud"
MYDBPW="M2cloud#"
```

4. Forcez la destruction de conteneur testenvvar

```
$docker rm -f testenvvar
```

5. Lancez un conteneur nommé testenvvar à partir de la dernière image ubuntu en lui transmettant les variables déclarées dans le fichier file-var.txt.

```
$docker run -tdi --name testenvvar  --env-file file-var.txt  ubuntu:latest
```

6. Exécutez un shell bash interactif sur votre conteneur.

```
$docker exec -ti testenvvar sh
```

3. Listez les variables d'environnement et vérifiez que les variables définies dans le fichier file-var.txt sont bien définies

```
# env
HOSTNAME=ebace7b68103
HOME=/root
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/
MYPASSWORD="M2cloud"
MYDBPW="M2cloud#"
```

4. Spécifiez le hostname de votre conteneur?

```
HOSTNAME=ebace7b68103
```

Par défaut, les conteneurs créés avec l'exécution de docker reçoivent un nom d'hôte aléatoire. Vous pouvez donner au conteneur un nom d'hôte différent en passant le drapeau –hostname.

5. Forcer la destruction de conteneur testenvvar

```
$docker rm -f testenvvar
```

6. Lancez un conteneur nommé testenvvar à partir de la dernière image ubuntu en lui attribuant le hostname M2

```
$docker run -tdi --name testenvvar  --hostname M2 ubuntu:latest
```

7. Vérifiez que le hostname a été bien attribué à votre conteneur

```
$docker exec -ti testenvvar sh
#env
HOSTNAME=M2
HOME=/root
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/
```