DEVOPS I : TOUT SAVOIR DE DOCKER EN QUELQUES HEURES

La persistance des données au sein des conteneurs

JORDAN ASSOULINE

OBJECTIFS DE LA SECTION

Objectifs

- Quelques explications sur la durée de vie d'un conteneur
- Que sont les Volumes dans Docker, et comment les utiliser ?
- Utiliser le Bind Mounting pour modifier la configuration des services en live sur un conteneur
- Exercice sur l'utilisation des Volumes pour mettre à jour le service Jenkins
- Exercice sur l'utilisation du Bind Mounting pour modifier le conteneur d'un site web en direct

DURÉE DE VIE D'UN CONTENEUR

JORDAN ASSOULINE

La notion d'immuabilité

- Les conteneurs sont conçus pour être la plupart du temps **immuable** et **éphémère** (non changeant, et temporaire).

- **L'infrastructure immuable**: on ne fait que redéployer les conteneurs, ceux-ci ne changent jamais.
  - Si des modifications sont nécessaires, on redéploye de nouveaux conteneurs basés sur des images différentes.

- Bénéfices : **fiabilité** et **consistance**, tout en permettant la reproduction de tous les changements

Et si on a besoin de conserver les données ?

- On parle de **persistance** des données lorsqu'on peut conserver celles-ci même si le conteneur a été supprimé ou recréé.

- Très utile surtout dans les conteneurs avec l'utilisation de bases de données.

- On peut utiliser deux méthodes:
  - **Volumes**: création d'une localisation particulière située en dehors du système de fichiers du conteneur (UFS)
  - **Bind Mounts**: création d'un lien virtuel entre un chemin absolu au niveau du conteneur, et un chemin absolu au niveau de la machine hôte

Quand utiliser plutôt les volumes ?

- Partager des données **entre plusieurs conteneurs** en cours d'exécution

- Lorsque l'hôte Docker n'est pas paramétré pour avoir un répertoire donnée ou un fichier de structure (permet de **découpler la configuration de l'hôte avec le conteneur**)

- Lorsqu'on souhaite stocker les données du conteneur sur un **hôte distant** ou sur le **Cloud**

- Lorsqu'on souhaite **restaurer**, **sauvegarder** ou **migrer** les données depuis un hôte Docker vers un autre hôte.

Quand utiliser plutôt les bind mounts ?

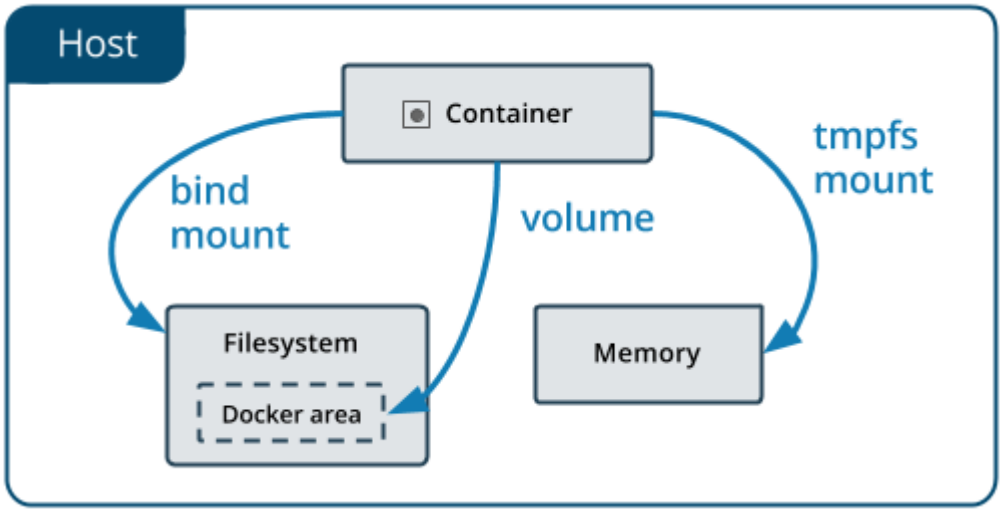
- Partager des **fichiers de configuration** de l'hôte Docker vers les conteneurs (par exemple pour la résolution DNS, le fichier /etc/resolv.conf est automatiquement monté depuis l'hôte vers les conteneurs)

- Partager du **code source** ou des éléments générés (**artefacts**) entre un environnement de développement sur l'hôte Docker et un conteneur.

- Lorsque le fichier, ou les dossiers structurels de l'hôte Docker ont une réelle **stabilité** et une **consistance** garanties

Documentation complémentaire

- On peut retrouver cet article (en anglais) qui reprend les détails concernant la persistance des données dans Docker:
- https://docs.docker.com/storage/



JORDAN ASSOULINE

LES VOLUMES

Les volumes dans votre Dockerfile

- Le **VOLUME** est un élément optionnel à faire figurer dans le Dockerfile si on veut utiliser cette fonctionnalité.

- **`VOLUME /var/lib/mysql`**

- Lorsqu'un nouveau conteneur est démarré, un volume va être créé et assigné au répertoire indiqué ci-dessus.

- Cela signifie que tous les fichiers qu'on mettra dans ce répertoire, à l'intérieur du conteneur, seront stockés sur l'hôte Docker.

Suppression des volumes

- Les volumes, ont besoin d'être supprimés **manuellement**.
- On ne peut pas supprimer les volumes directement en supprimant le conteneur.
- C'est normal, car les données stockées dans ces volumes sont considérées comme étant sensibles et importantes.

La commande « docker volume prune »

- **`docker volume prune`**
  - Permet de supprimer tous les volumes qui ne sont plus utilisés sur l'hôte Docker

- https://docs.docker.com/engine/reference/commandline/volume_prune/

La commande « docker image inspect »

- **`docker image inspect mysql`**
  - Dans la partie « Config » puis « Volumes », on peut récupérer l'information concernant les volumes qui sont créés lors du démarrage d'un nouveau conteneur.

```
"Volumes": {
                "/var/lib/mysql": {}
            },
```

JORDAN ASSOULINE

La commande « docker container inspect »

- **docker container inspect mysql**
  - Après avoir démarré un container mysql avec « docker container run –d –e MYSQL_ALLOW_EMPTY_PASSWORD=True mysql » on peut trouver l'information du type :


```
"Mounts": [
            {
                "Type": "volume",
                "Name": "15792e64f31e2979cb7c6ec02124e7d939111965da585192ca4f2121bc20aef5",
                "Source": "/var/lib/docker/volumes/15792e64f31e2979cb7c6ec02124e7d939111965da585192ca4f2121bc20aef5/_data",
                "Destination": "/var/lib/mysql",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
            }
        ],
```


JORDAN ASSOULINE

La commande « docker volume ls »

- **`docker volume ls`**

  - Permet de donner la liste des volumes actuels sur l'hôte Docker

  - https://docs.docker.com/engine/reference/commandline/volume_ls/

La commande « docker volume inspect »

- **`docker volume inspect <VOLUME_NAME>`**

  - Donner des informations détaillées sur un ou plusieurs volumes
  - *Attention, sur Windows, il faut aller dans la machine virtuelle elle-même pour accéder aux volumes stockés localement*

  - https://docs.docker.com/engine/reference/commandline/volume_inspect/

La commande « docker container run »

- **`docker container run -d --name mysql -e MYSQL_ALLOW_EMPTY_PASSWORD=True -v mysql-db:/var/lib/mysql sql`**

  - L'option **–v** permet de spécifier le volume à créer. On commence par le nom du volume ici **mysql-db**, puis après les « **:** », on spécifie le chemin du volume à l'intérieur du conteneur, ici **/var/lib/mysql**.

  - https://docs.docker.com/engine/reference/commandline/run/

La commande « docker volume create »

- **`docker volume create [OPTIONS] [NAME]`**

  - Permet de créer un volume docker.
  - Très utile pour pouvoir lui spécifier des options particulières, comme les options de driver grâce à l'option **-o**.

  - https://docs.docker.com/engine/reference/commandline/volume_create/

BIND MOUTING

JORDAN ASSOULINE

Les Bind Mounting dans Docker

- Le **BIND MOUNTING** permet d'effectuer une association entre des fichiers ou des répertoires de la machine hôte, avec ceux d'un ou de plusieurs conteneurs.

- Les fichiers et répertoires de l'hôte en Bind Mounting, écrasent ceux qui existent à l'intérieur d'un conteneur

- Ne peuvent être paramétrés que lors de l'utilisation de la commande **docker container run** et non dans le fichier Dockerfile comme les volumes

La syntaxe du Bind Mounting

- Commande pour monter des volumes:
  - … **`run -v mysql-db:/var/lib/mysql`**

- Commande pour faire du Bind Mounting:
  - … **`run -v /home/jassouline/mysql:/var/lib/mysql`**
  - Au lieu de spécifier juste un nom, on a besoin de spécifier le chemin absolu du fichier avant le « : »

  :

  - … **`run -v //c/Users/Jordan/mysql:/var/lib/mysql`**
  - Voici la notation pour un système Windows

JORDAN ASSOULINE

EXERCICE 1

Instructions

Instructions

- Le but de cet exercice sera de mettre à jour un service que l'on utilise, tout en maintenant la configuration grâce à l'utilisation des volumes

Instructions

- Nous allons utiliser l'image « jenkins » qui est un service permettant de faire de l'intégration continue.

- Par défaut, Jenkins utilise dans son Dockerfile:
  - **`VOLUME /var/jenkins_home`**

- Il nous est indiqué que pour utiliser l'image Docker de Jenkins, une commande suffit :
  - `docker run -p 8080:8080 -p 50000:50000 jenkins`

Instructions

- Vous allez donc démarrer un conteneur jenkins ayant pour nom « **jenkins1** » dans sa version « **2.32.1** » et en paramétrant le nom du volume comme étant : « **jenkins_volume** »

- Rendez-vous ensuite sur la page d'accueil de Jenkins, faîtes une installation classique des plugins et créer un simple Job comme indiqué dans la vidéo

- Arrêtez le conteneur jenkins1, puis recréer un deuxième conteneur jenkins2 avec des paramètres identiques, mais cette fois en version « **latest** »

- Vérifiez que vous obtenez toujours les jobs que vous avez créé

EXERCICE 1

Correction

Correction

- Commençons par démarrer ce conteneur en utilisant l'option permettant de nommer le volume : « jenkins_volume »
  - `docker run --name jenkins1 -p 8080:8080 -p 50000:50000 -v jenkins_volume:/var/jenkins_home -d jenkins:2.32.1`
  - `docker container stop jenkins1`
  - `docker run --name jenkins2 -p 8080:8080 -p 50000:50000 -v jenkins_volume:/var/jenkins_home -d jenkins:latest`

JORDAN ASSOULINE

EXERCICE 2

Instructions

Instructions

- Le but de cet exercice sera de modifier en temps réel un site web sans avoir à créer un nouveau conteneur.

JORDAN ASSOULINE

Instructions

- Une page html se trouve dans : « docker-volumes/Exercice 2/index.html »

- Nous allons utiliser l'image httpd, que vous pouvez trouver sur dockerhub.

- Jeter un œil à la section « **How to use this image** » pour savoir quel est le répertoire de l'image que nous allons binder avec le répertoire où se trouve actuellement la page index.html située dans l'Exercice 2.

Instructions

- Le conteneur devra avoir le nom « **my_apache** » et utiliser l'image « **httpd:2.4** »

- Pour rappel, grâce à l'option –v vous allez devoir binder le répertoire dans lequel vous vous situé (attention à mettre le chemin complet entre **"** pour que les espaces soient bien interprétés)

- Vérifier ensuite que le site internet s'affiche bien, puis modifier la page d'accueil depuis votre répertoire actuel, et vérifier que la page a bien été modifiée en direct en actualisant votre navigateur

EXERCICE 2

Correction

JORDAN ASSOULINE

Correction

- `cd C:\Users\jorda\OneDrive\Bureau\Code\udemy-docker\docker-volumes\Exercice 2`

- `docker container run --name my_apache -p 8080:80 -v C:\Users\jorda\OneDrive\Bureau\Code\udemy-docker\docker-volumes\Exercice 2:/usr/local/apache2/htdocs/ httpd:2.4`