Variables d'environnement : `DOCKER_HOST`

Qu'est-ce que `DOCKER_HOST` ?

`DOCKER_HOST` est une variable d'environnement utilisée par le client Docker (`docker`) pour déterminer l'adresse à laquelle il doit se connecter au démon Docker (`dockerd`). Cette variable est essentielle lorsque vous devez exécuter des commandes Docker sur un démon Docker distant ou personnalisé.

- **Valeur par défaut** : Par défaut, le client Docker se connecte au démon Docker local via le socket Unix `/var/run/docker.sock` sur les systèmes Linux. Ce socket est la méthode de communication par défaut lorsque vous exécutez Docker localement sur une machine.
- **Configuration distante** : Si vous souhaitez exécuter des commandes Docker sur un serveur distant, vous devez configurer `DOCKER_HOST` pour pointer vers l'adresse du démon Docker distant (par exemple via TCP ou via un socket Unix distant).

En définissant `DOCKER_HOST`, vous permettez au client Docker de communiquer avec un démon spécifique, même si celui-ci ne se trouve pas sur la machine locale.

Exemple de `DOCKER_HOST` :

- Pour se connecter au démon Docker via HTTP/TCP sur un serveur distant :

```
export DOCKER_HOST="tcp://192.168.1.100:2375"
```

- Pour se connecter au démon Docker via un socket Unix distant :

```
export DOCKER_HOST="unix:///var/run/docker.sock"
```

Comment configurer `DOCKER_HOST` ?

La variable `DOCKER_HOST` peut être configurée de plusieurs façons, selon votre environnement et la façon dont vous voulez interagir avec le démon Docker.

1. **Configuration temporaire via la ligne de commande**

   Pour configurer `DOCKER_HOST` temporairement, vous pouvez le faire directement dans votre terminal en utilisant la commande `export` (sur Linux/macOS) ou `set` (sur Windows) :

   - **Sur Linux/macOS** :

     ```
     export DOCKER_HOST="tcp://192.168.1.100:2375"
     ```

   Cette configuration est temporaire et ne persistera que pour la session en cours. Si vous fermez le terminal ou redémarrez, cette configuration sera perdue.

2. **Configuration permanente dans un fichier de profil**

   Pour rendre la configuration permanente, vous pouvez ajouter la commande `export` dans votre fichier de profil shell. Selon le shell que vous utilisez, vous pouvez modifier l'un de ces fichiers :

   - **Bash** : Ajoutez l'export dans `~/.bashrc` ou `~/.bash_profile`.
   - **Zsh** : Ajoutez l'export dans `~/.zshrc`.
   - **Fish** : Ajoutez l'export dans `~/.config/fish/config.fish`.

   Exemple pour Bash :

   ```
   echo 'export DOCKER_HOST="tcp://192.168.1.100:2375"' >> ~/.bashrc
   source ~/.bashrc
   ```

3. **Utilisation de `DOCKER_HOST` dans des scripts**

   Si vous exécutez des scripts automatisés et que vous devez vous connecter à un démon Docker distant, vous pouvez configurer `DOCKER_HOST` dans le script lui-même avant d'exécuter des commandes Docker.

   Exemple de script Shell :

   ```
   #!/bin/bash
   export DOCKER_HOST="tcp://192.168.1.100:2375"
   docker ps
   ```

4. **Utilisation de `DOCKER_HOST` avec Docker Compose**

   Lorsque vous utilisez Docker Compose, vous pouvez définir la variable d'environnement `DOCKER_HOST` avant d'exécuter des commandes Docker Compose pour qu'il interagisse avec le démon distant.

   Exemple :

   ```
   export DOCKER_HOST="tcp://192.168.1.100:2375"
   docker-compose up
   ```

5. **Configurer `DOCKER_HOST` pour un autre utilisateur (utilisation globale)**

   Si vous voulez que tous les utilisateurs sur votre machine aient accès à la même configuration `DOCKER_HOST`, vous pouvez définir la variable dans les fichiers système ou de profil global, comme `/etc/profile` (sur les systèmes Unix/Linux).

   Exemple :

```
sudo echo 'export DOCKER_HOST="tcp://192.168.1.100:2375"' >>
/etc/profile
source /etc/profile
```

Autres considérations

1. **Sécurisation de l'accès distant avec `DOCKER_HOST`** :
   - Lorsque vous configurez `DOCKER_HOST` pour un démon distant accessible via TCP, il est important de sécuriser l'accès, car la connexion peut être vulnérable si elle n'est pas chiffrée.
   - Utilisez des options comme `TLS` (Transport Layer Security) pour sécuriser les communications entre le client Docker et le démon Docker.
   - Exemple d'activation de TLS sur le démon Docker :

     ```
     DOCKER_HOST="tcp://192.168.1.100:2376"
     DOCKER_TLS_VERIFY=1
     DOCKER_CERT_PATH="/path/to/certs"
     ```

2. **Problèmes courants** :
   - **Erreur de connexion** : Si `DOCKER_HOST` est mal configuré, vous recevrez une erreur comme `Cannot connect to the Docker daemon`. Assurez-vous que l'adresse et le port sont corrects.
   - **Problèmes de permissions** : Lorsque vous utilisez des sockets Unix, vous devez avoir les permissions appropriées pour accéder à `/var/run/docker.sock`. Sinon, vous devrez exécuter les commandes Docker en tant que superutilisateur (`sudo`).

Résumé

- `DOCKER_HOST` est une variable d'environnement qui détermine à quel démon Docker se connecter. Par défaut, il pointe vers le démon local via le socket Unix, mais vous pouvez le configurer pour pointer vers un démon distant en utilisant des protocoles comme TCP.
- Vous pouvez configurer `DOCKER_HOST` temporairement dans le terminal, de manière permanente dans un fichier de profil, ou directement dans des scripts pour automatiser des processus.
- Il est crucial de sécuriser les connexions distantes avec des mécanismes comme TLS lorsque vous travaillez avec des démons Docker accessibles sur un réseau.