L'option socket pour les accès réseau

Le **socket Docker** est une méthode par laquelle le client Docker communique avec le démon Docker (dockerd). Il est principalement utilisé pour la communication locale sur les systèmes Unix/Linux, bien que Docker supporte également l'accès via TCP. Le socket Docker est un fichier spécial qui sert de point de connexion entre le client Docker et le démon Docker.

Utilisation du socket Docker (`/var/run/docker.sock`)

Le socket Docker par défaut est localisé à l'adresse `/var/run/docker.sock` et est généralement utilisé pour permettre au client Docker d'interagir avec le démon Docker sans nécessiter de communication réseau externe.

1. **Accès au démon Docker via le socket** :
    - Lorsqu'un client Docker exécute une commande, il utilise ce socket pour envoyer des requêtes au démon Docker. Cela permet de contrôler les conteneurs, images, volumes, réseaux, etc.
    - Le démon Docker écoute sur ce socket par défaut pour recevoir des commandes du client Docker.
2. **Exécution des commandes via le socket** :
    - Par exemple, pour lister les conteneurs en cours d'exécution, Docker utilise le socket pour transmettre cette commande au démon :

      ```
      docker ps
      ```

    - Cela envoie la requête via le socket Docker, et le démon répond avec les informations des conteneurs en cours d'exécution.
3. **Communication en utilisant le socket** :
    - Le socket Docker est une interface Unix Domain Socket (UDS) qui est très rapide pour les communications locales entre le client et le démon Docker, car elle évite les surcharges réseau.
    - Le chemin par défaut du socket Docker est `/var/run/docker.sock`.
4. **Accès depuis un conteneur** :
    - Si vous avez un conteneur qui doit interagir avec le démon Docker (par exemple, pour construire des images Docker à partir d'un conteneur), vous pouvez monter ce socket à l'intérieur du conteneur.
    - Exemple :

      ```
      ```

```
docker run -v /var/run/docker.sock:/var/run/docker.sock -it
my-container
```

- Cela permet au conteneur d'exécuter des commandes Docker et d'interagir avec le démon Docker de l'hôte. Cependant, cela donne au conteneur un accès potentiellement risqué au système hôte, et cela doit être utilisé avec prudence.

Sécurisation de l'accès au socket

L'accès au socket Docker doit être géré correctement pour éviter des risques de sécurité. Si un utilisateur non autorisé ou un conteneur malveillant peut accéder au socket Docker, il pourrait prendre le contrôle du démon Docker, et donc, avoir accès à toutes les opérations possibles sur le système hôte.

1. **Contrôle des permissions d'accès** :
   Le fichier de socket `/var/run/docker.sock` est généralement accessible uniquement aux utilisateurs faisant partie du groupe `docker` (par défaut, sur la plupart des distributions Linux). Cela permet de restreindre l'accès au démon Docker aux utilisateurs qui doivent pouvoir interagir avec Docker.

   - **Vérifier les permissions du socket** :

     ```
     ls -l /var/run/docker.sock
     ```

     Exemple de sortie :

     ```
     srw-rw---- 1 root docker 0 Feb 14 10:25 /var/run/docker.sock
     ```

     Cela signifie que le socket appartient à l'utilisateur `root` et au groupe `docker`, et que seuls les membres du groupe `docker` peuvent y accéder.

   - **Ajouter un utilisateur au groupe Docker** :
     Si un utilisateur doit pouvoir exécuter des commandes Docker, vous pouvez l'ajouter au groupe `docker` :

     ```
     sudo usermod -aG docker <username>
     ```

     Après avoir ajouté l'utilisateur, il devra se reconnecter pour que les modifications prennent effet.

2. **Accès restreint pour les conteneurs** :
   En raison des risques de sécurité, donner un accès direct au socket Docker à un conteneur (via le montage de `/var/run/docker.sock`) peut être risqué, car il permet au conteneur d'avoir un

accès complet au démon Docker de l'hôte. Cela peut entraîner des attaques potentielles, comme l'exécution de commandes arbitraires ou l'accès à des informations sensibles.

- **Alternatives sécurisées** :
  - Utilisez des outils comme **Docker API avec authentification TLS** plutôt que de permettre l'accès direct au socket.
  - Configurez un **serveur Docker remote sécurisé avec TLS** pour que les conteneurs n'aient pas à accéder directement au démon Docker via le socket local.

3. **Utilisation de Docker avec des sockets Unix et des utilisateurs dédiés** :
   Il est recommandé de ne permettre l'accès au socket Docker qu'aux utilisateurs qui en ont absolument besoin et de toujours configurer les permissions de manière restrictive. Cela peut être complété par la mise en œuvre d'une surveillance sur les actions effectuées par les utilisateurs ou les conteneurs ayant accès au socket Docker.

4. **Sécurisation de Docker avec des politiques de contrôle d'accès** :
   Vous pouvez configurer Docker pour que des politiques strictes de contrôle d'accès soient appliquées, comme l'usage de `AppArmor`, `SELinux`, ou d'autres mécanismes d'audit pour surveiller et restreindre les actions qui peuvent être effectuées sur Docker. Cela offre une sécurité supplémentaire en cas d'accès non autorisé au socket.

Exemple de configuration sécurisée : Utilisation de TLS pour Docker Remote API

1. **Configurer Docker pour utiliser TLS** :
   Si vous voulez exposer Docker de manière sécurisée sur le réseau, vous pouvez configurer le démon Docker pour écouter via HTTPS, en activant TLS pour chiffrer la communication. Cela permet de restreindre l'accès avec des certificats.

   - Exemple d'activation de TLS dans le fichier `daemon.json` :

     ```
     {
       "host": "tcp://0.0.0.0:2376",
       "tlsverify": true,
       "tlscacert": "/etc/docker/certs/ca.pem",
       "tlscert": "/etc/docker/certs/server-cert.pem",
       "tlskey": "/etc/docker/certs/server-key.pem"
     }
     ```

   - Vous devrez également fournir les certificats appropriés pour que les clients puissent se connecter de manière sécurisée.

2. **Accéder au démon Docker via TLS** :
   Une fois que Docker est configuré pour accepter des connexions sécurisées, vous pouvez vous y connecter en utilisant le client Docker avec les certificats :

   ```
   export DOCKER_HOST="tcp://<docker-server-ip>:2376"
   export DOCKER_TLS_VERIFY="1"
   ```

```
export DOCKER_CERT_PATH="/path/to/certs"
```

Cela garantit que la communication avec Docker est chiffrée et authentifiée, ce qui est particulièrement important dans un environnement de production ou lorsque Docker est exposé à un réseau public.

Résumé

- Le **socket Docker** (`/var/run/docker.sock`) est utilisé pour la communication entre le client Docker et le démon Docker.
- Par défaut, il est accessible uniquement aux utilisateurs faisant partie du groupe `docker`. Il est essentiel de restreindre l'accès au socket pour éviter des risques de sécurité.
- Vous pouvez **sécuriser l'accès au socket** en limitant les permissions ou en utilisant des solutions comme **TLS** pour chiffrer les communications lorsque vous accédez au démon Docker à distance.
- Donner un accès direct au socket Docker aux conteneurs (via le montage du socket) est risqué et doit être évité ou restreint dans un environnement de production.