Configuration du daemon Docker

Dans cet article nous allons parler des options qui sont disponibles pour la configuration du daemon Docker.

Lancement du daemon Docker

Une fois que Docker est installé, le daemon Docker *dockerd* est géré par un système d'init tel que *systemd* ou *systemV*. Ce dernier est cependant moins utilisé, de nombreuses distributions utilisant maintenant *systemd*.

Note: il est bien sur possible de lancer le daemon en ligne de commande. Cette approche n'est pas recommandée dans un environnement de production mais s'avère très utile en développement, elle permet notamment de tester des configurations différentes.

Les options de configuration du daemon Docker

De nombreuses options peuvent être spécifiées lors du lancement du daemon Docker, la liste complète est disponible à l'adresse suivante:
https://docs.docker.com/engine/reference/commandline/dockerd/

Celles-ci permettent par exemple de configurer:

- le driver utilisé pour la gestion des logs (nous y reviendrons dans un chapitre dédié)
- le network
- la sécurisation de la communication entre le client et le daemon
- …

Les exemples ci-dessous illustrent l'utilisation de plusieurs options dans le cas d'un lancement en ligne de commande et d'un lancement depuis systemd.

Lancement en ligne de commande

L'exemple suivant montre le lancement du daemon Docker en ligne de commande.

Plusieurs options sont spécifiées ici:

- -D indique que le mode debug est activé

- --tls=true indique que les communications avec le daemon sont faites de manière sécurisée
- --tlscert et --tlskey précisent l'emplacement des clés de chiffrement
- -H indique la socket sur laquel le daemon écoute (et auquel un client devra s'adresser), il s'agit dans le cas présent d'une socket tcp

```
$ dockerd -D --tls=true --tlscert=/var/docker/server.pem --
tlskey=/var/docker/serverkey.pem -H tcp://192.168.99.100:2376
```

Dans ce cas de figure, tout client docker pourra accéder au daemon à partir du moment ou il a accès à l'IP 192.168.99.100 sur le port 2376. La communication client / serveur sera encryptée avec TLS, mais il n'y aura pas de vérification de l'identité du client.

Lancement avec systemd

L'exemple suivante montre le lancement du docker daemon par le process d'init *systemd*

> Le fichier de configuration de dockerd, généralement situé dans l'arborescence de /etc/systemd/system/docker.service.d, à un contenu similaire (aux options près) à celui ci-dessous.

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2376 -H unix:///var/run/docker.sock
--storage-driver aufs --tlsverify --tlscacert /etc/docker/ca.pem --tlscert
/etc/docker/server.pem --tlskey /etc/docker/server-key.pem --label
provider=packet
Environment=
```

Différentes options sont spécifiées ici:

- *-H* indique la socket sur laquelle le daemon écoute. Cette option est utilisée 2 fois, la première permet une connexion *tcp* et la seconde une connexion en local via la socket unix */var/run/docker.sock*
- *--tlscacert*, *--tlscert*, *--tlsckey* sont utilisées pour spécifier que la communication est sécurisée et définir les chemins vers les clés de chiffrement et le certificat de l'authorité de certification
- *--tlsverify* permet de vérifier l'authenticité du client docker en s'assurant que le certificat utilisé par le client à bien été signé par l'authorité de certification du daemon
- *--storage-driver* indique le driver utilisé pour le stockage des images. Il existe plusieurs

drivers, nous en parlerons dans un prochain chapitre
- *--label* permet d'ajouter un label au daemon Docker, le format est key=value. Le label ajouté ici référence le cloud provider sur lequel la machine virtuelle a été instanciée

Le fichier de configuration /etc/docker/daemon.json

Par défaut, le fichier de configuration */etc/docker/daemon.json* permet de spécifier des options qui seront prises en compte lors du lancement du daemon. C'est un fichier utilisable quelque soit la distribution Linux utilisée.

L'exemple ci-dessous montre le contenu de ce fichier dans le cadre d'un daemon Docker déployé sur PWD (play-with-docker.com)

```
$ cat /etc/docker/daemon.json
{
    "experimental": true,
    "debug": true,
    "log-level": "info",
    "insecure-registries": ["127.0.0.1"],
    "hosts": ["unix:///var/run/docker.sock", "tcp://0.0.0.0:2375"],
    "tls": false,
    "tlscacert": "",
    "tlscert": "",
    "tlskey": ""
}
```

En plus des options que nous avons vues dans les exemples précédents, d'autres sont également spécifiées ici:

- *experimental* permet d'utiliser les version *edge* de Docker CE et ainsi de profiter des dernières nouveautés
- *debug* (équivalent au flag *-D* utilisé en ligne de commande) permet de lancer le daemon en mode debug
- *log-level* permet de spécifier le niveau de log
- *insecure-registries* permet de spécifier une liste de *registries* (librairies d'images Docker) auquel le daemon peut accéder de façon non sécurisée (nous y reviendrons dans le chapitre Registry)
- *hosts* définit les sockets exposées, celles-ci permettent à des clients docker de communiquer avec le daemon. La socket unix */var/run/docker.sock* permet une communication en local, la socket tcp *0.0.0.0:2375* permet une communication via le réseau
- les options liées à la sécurisation via TLS des communications client / serveur (*tls*,

tlscacert, *tlscert*, *tlskey*) ne sont pas utilisées ici, le daemon est accessible en tcp sur le port 2375 de façon non sécurisée

En résumé

Nous avons vu ici différentes façon de configurer un daemon Docker et quelques exemples d'options. De nombreuses options sont disponibles, nous reviendrons sur certaines d'entre elles dans la suite du cours.