docker-run

Lancer une commande dans un nouveau conteneur

Lance un processus dans un nouveau conteneur, avec son propre système de fichier, son propre réseau, et sa propre arborescence isolée. l'image qui démarre le processus peut définir des défaut qui sont liés au processus qui sera lancé dans le conteneur, le réseau à exposer, et plus, mais docker run donne un contrôle final à l'opérateur ou d'administrateur qui a lancé le conteneur depuis l'image. Pour cette raison, docker run a plus d'options que d'autres commande docker. Si l'image n'est pas déja chargée, elle est téléchargée, et toutse les dépendences, depuis le dépot, de la même manière que docker pull.

OPTIONS

-a, –attach=[] Attache à STDIN, STDOUT ou STDERR. En mode foreground.

–add-host=[] Ajoute un mappage host-to-IP personnalisé (host :ip). Ajoute une ligne à /etc/hosts. peut être spécifié plusieurs fois.

–blkio-weight=0 Poid IO entre 10 et 1000

–blkio-weight-device=[] Poid IO au format DEVICE_NAME :WEIGHT

–cpu-shares=0 Partage CPU

–cap-add=[] Ajouter des capabilities Linux

–cap-drop=[] Supprimer de capabilities Linux

–cgroup-parent="" Chemin des cgroups sous lesquels créer le conteneur. Si le chemin n'est pas absolu, le chemin est relatif aux cgroups du processus init.

–cidfile="" Écrit l'ID de conteneur dans le fichier

–cpu-period=0 Limite la période CFS

–cpuset-cpus=CPUSET-CPUS CPU dans lesquels autoriser l'exécution

–cpuset-mems="" Nœuds mémoire autorisés.

–cpu-quota=0 Limite le quota CPU CFS

-d, –detach=true|false Lance de conteneur en tâche de fond et affiche l'ID du conteneur. défaut : false

–detach-keys="" Change la séquence de touche pour détacher le conteneur. Défaut : CTRL-p CTRL-q.

–device=[] Un périphérique hôte pour le conteneur (ex –device=/dev/sdc :/dev/xvdc :run)

–device-read-bps=[] Limite le taux de lecture d'un périphérique (ex –device-read-bps=/dev/sda :1mb)

–device-read-iops Limite le taux de lecture d'un périphérique (ex –device-read-iops=/dev/sda :1000)

–device-write-bps=[] Limite le taux d'écriture d'un périphérique (ex –device-write-bps=/dev/sda :1mb)

–device-write-iops Limite le taux d'écriture d'un périphérique (ex –device-write-iops=/dev/sda :1000)

–dns-search=[] Définis les domaines de recherche DNS.

–dns-opt=[] Définis les options DNS

–dns=[] Définis les serveurs DNS

-e, –env=[] Définis des variables d'environnement.

–entrypoint="" Remplace l'ENTRYPOINT de l'image

–env-file=[] Line un fichier de variables d'environnements

–expose=[] Expose un port, une une plage de ports informant Docker que le conteneur écoute sur les ports spécifiés. Docker utilise cette information pour interconnecter les conteneurs en utilisant des liens et pour définir la redirection de port dans le système hôte.

–group-add=[] Ajoute des groupes additionnels à lancer

-h, –hostname="" nom d'hôte du conteneur

-i, –interactive=true|false Garde STDIN ouvert même s'il n'est pas attaché. défaut : false

–ip="" Définis l'adresse IPv4 de l'interface du conteneur, uniquement avec –net pour les réseaux utilisateurs

–ip6="" Définis l'adresse IPv6 de l'interface du conteneur, uniquement avec –net pour les réseaux utilisateurs

–ipc="" Créé un espace de nom IPC privé pour le conteneur

–isolation="default" spécifique le type de technologie d'isolation utilisée par les conteneurs

-l, –label=[] Définis les métadonnées du conteneur (ex : –label com.example.key=value)

–kernel-memory="" Limit la mémoire kernel. (0 = illimité)

–label-file=[] Lit un fichier de labels

–link=[] Ajoute un lien vers un autre conteneur sous la forme <nom ou id> :alias.

–log-driver="json-file|syslog|journald|gelf|fluentd|awslogs|splunk|none" Pilote de logging pour le conteneur. La commande docker logs ne fonctionne que pour les pilotes json-file et journald.

–log-opt=[] options spécifique au pilote de logging

-m, –memory="" Limite mémoire

–memory-reservation="" Limite soft

–memory-swap="LIMIT" Une valeur limite égale à la mémoire plus le swap. Doit être utilisé avec -m.

–mac-address="" L'adresse mac de conteneur

–name="" Assigne un nom au conteneur au format UUID long, UUID court, ou un nom.

–net="bridge" Définis le mode réseau pour le conteneur (bridge, none, container :<name|id>, host, <network-name>|<network-id>

–net-alias=[] ajoute un alias réseau pour le conteneur

–oom-kill-disable=true|false Active ou non le OOM Killer pour le conteneur

–oom-score-adj="" Ajustement OOM de l'hôte pour le conteneur (-1000 à 1000)

-P, –publish-all=true|false Publie tous les ports exposés aux ports aléatoire dans les interfaces de l'hôte. Défaut : false

-p, –publish=[] Publie le port d'un conteneur, ou une plage de port, à l'hôte (Format : ip :hostPort :containerPort | ip : :containerPort | hostPort :containerPort | containerPort)

–pid=host Définis de mode PID pour le conteneur (host : utilise l'espace de nom PID de l'hôte, non sécurisé)

–uts=host Définis le mode UTS pour le conteneur (host : utilise l'espace de noms UTS dans le conteneur, non sécurisé)

–privileged=true|false Donne des priviléges étendus à ce conteneur. Défaut : false

–read-only=true|false Mounte le système de fichier racine du conteneur en lecture seule.

–restart="no" Stratégie de redémarrage quand le conteneur se termine. (no, on-failure[:max-retry], always, unless-stopped)

-rm=true|false Supprime automatiquement le conteneur quand il se termine. incompatible avec -d. défaut : false

–security-opt=[] Options de sécurité

- **"label :user :USER"** Définis le label utilisateur pour le conteneur
- **"label :role :ROLE"** Définis le label rôle pour le conteneur
- **"label :type :TYPE"** Définis le label type pour le conteneur
- **"label :level :LEVEL"** Définis le label level pour le conteneur
- **"label :disable"** Désactive le confinement de label pour le conteneur

–stop-signal=SIGTERM Signal pour stopper un conteneur

–shm-size="" Taille de /dev/shm. format : <number><unit>

–sig-proxy=true|false signaux proxy reçus au processus (mode non-TTY uniquement). SIGCHLD, SIGSTOP, SIGKILL ne sont pas proxifiés. Défaut : true

–memory-swappiness="" Comportement de la memory swappiness du conteneur (0 à 100)

-t, –tty=true|false Alloue un pseudo-TTY. défaut : false

–tmpfs=[] Créé un montage tmpfs

-u, –user="" Définis le username ou l'UID utilisé et optionnellement le group ou GID pour la commande spécifiée

–ulimit=[] Options ulimit

-v|–volume=[[[HOST-DIR :]CONTAINER-DIR[:OPTIONS]]] Créé un montage. avec -v /HOST-DIR :/CONTAINER-DIR, Docker monte /HOST-DIR dans l'hôte dans /CONTAINER-DIR dans le conteneur. Si HOST-DIR est omis, Docker créé un nouveau volume dans l'hôte. par défaut, les montages sont privés (non-visibles dans l'hôte). Peut être spécifié plusieurs fois.

–volume-driver="" Pilote de volume du conteneur. Ce pilote créé les volumes spécifié soit avec l'instruction VOLUME du dockerfile, ou depuis l'option -v.

–volumes-from=[] Monte les volumes depuis les conteneurs spécifiés.

-w, –workdir="" Répertoire de travail dans le conteneur

Code de sortie

Le code de sortie de docker run donne des information sur la raison de l'erreur du conteneur ou pourquoi il s'est terminé.

125 Si l'erreur vient du service Docker lui-même

126 Si la commande dans le conteneur ne peut être lancée

127 Si la commande dans le conteneur ne peut être trouvée

Autres code Contient le code de sortie de la commande

Exemples

Lancer un conteneur en mode lecture seule :
docker run –read-only –tmpfs /run –tmpfs /tmp -i -t fedora /bin/bash
Exposer les messages de log du conteneur dans les logs de l'hôte :
docker run -v /dev/log :/dev/log -i -t fedora /bin/bash
Attacher un ou plusieurs STDIN, STDOUT, STDERR :
docker run -a stdin -a stdout -i -t fedora /bin/bash

Lier les conteneurs

Cette section décrit comment lier les conteneur sur le réseau par défaut. La fonctionnalité link permet à plusieurs conteneurs de communiquer entre eux. Par exemple, un conteneur dont Dockerfile a exposé le port 80 peut être lancés comme suit :
docker run –name=link-test -d -i -t fedora/httpd
Un second conteneur, dans ce cas nommé linker, peut communiquer avec le conteneur httpd :
docker run -t -i –link=link-test :lt –name=linker fedora /bin/bash
Maintenant le conteneur linker et lié au conteneur link-test avec l'alias lt. Lancer la commande env dans le conteneur linker montre les valiables d'environnement :

```
# env
HOSTNAME=668231cb0978
TERM=xterm
LT_PORT_80_TCP=tcp ://172.17.0.3 :80
LT_PORT_80_TCP_PORT=80
LT_PORT_80_TCP_PROTO=tcp
LT_PORT=tcp ://172.17.0.3 :80
PATH=/usr/local/sbin :/usr/local/bin :/usr/sbin :/usr/bin :/sbin :/bin
PWD=/
LT_NAME=/linker/lt
SHLVL=1
HOME=/
LT_PORT_80_TCP_ADDR=172.17.0.3
_=/usr/bin/env
```

Mapper les ports pour une utilisation externe

Le port exposé d'une application peut être mappée sur un port hôte en utilisant le flag -p. Par exemple, le port 80 peut être mappé au port 8080 :

docker run -p 8080 :80 -d -i -t fedora/httpd

Créer et monter un volume

De nombreuses application nécessitent le partage de données persistantes entre de nombreux conteneurs. Docker permet de créer un Data Volume Container que d'autres conteneurs peuvent monter. Par exemple, créer un conteneur qui contient les répertoires /var/volume1 et /tmp/volume2. L'image doit contenir ces répertoires, donc des instructions mkdir sont nécessaires :

docker run –name=data -v /var/volume1 -v /tmp/volume2 -i -t fedora-data true

docker run –volumes-from=data –name=fedora-container1 -i -t fedora bash

Plusieurs –volumes-from vont monter les volumes de données. Et il est possible de monter les volumes qui vient d'un conteneur DATA dans un autre conteneur :

docker run –volumes-from=fedora-container1 –name=fedora-container2 -i -t fedora bash

Monter des volumes externes

Pour monter un répertoire hôte comme volume conteneur, spécifier le chemin absolue :

docker run -v /var/db :/data1 -i -t fedora bash

En utilisant SELinux, vérifier que l'hôte n'a pas connaissance de la stratégie SELinux du conteneur. Dans l'exemple suivant, SELinux est enforced, le répertoire /var/db n'est pas en écriture pour le conteneur. Au moment d'écrire ce man, la commande suivante doit être lancée pour que la stratégie SELinux soit attachée correctement au répertoire de l'hôte.

chcon -Rt svirt_sandbox_file_t /var/db

Utiliser un labeling de sécurité alternatif

–security-opt permet de changer le schéma de labelisation par défaut pour chaque conteneur. Par exemple, on peut spécifier un niveau NCS/MLS. Spécifier le niveaux dans la commande suivante permet de partager le même contenu entre les conteneurs :

docker run –security-opt label :level :s0 :c100,c200 -i -t fedora bash

Un exemple MLS peut être :

docker run –security-opt label :level :TopSecret -i -t rhel7 bash

Pour désactiver le labeling de sécurité pour ce conteneur :

docker run –security-opt label :disable -i -t fedora bash

Pour resserrer la stratégie de sécurité sur les processus dans un conteneur :

docker run –security-opt label :type :svirt_apache_t -i -t centos bash