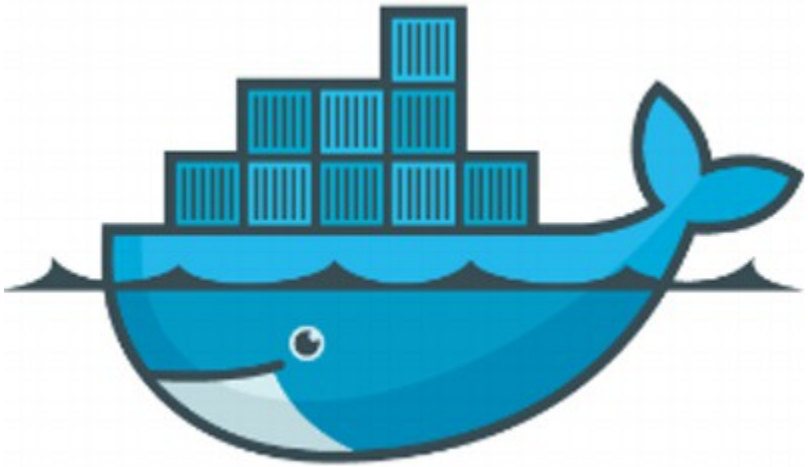


Docker partie 2

Journée ARGOS 11/12/2014

Gaël Beauquin CNRS/DSI

Sommaire

1. Fonctionnalités avancées
2. Gestion des ressources avec Docker
3. Logs sous Docker
4. Docker et la sécurité
5. Optimiser ses Dockerfiles
6. Développements autour de Docker
7. Le futur

Fonctionnalités avancées

- Lier des containers
- Attacher un volume
- Travailler avec le Docker Hub
- Mettre en écoute le daemon sur le réseau

Lier des containers

- Permet de faire communiquer des containers en privé
- Aide à découpler une application en briques cohérentes (application <-> base de donnée)
- Un container lié à un autre récupère des variables d'environnement lui donnant les infos pour se connecter

Attacher un volume

- Permet de partager un répertoire/fichier entre l'hôte et un container
- Pratique pour avoir des données persistantes indépendantes de container
- Possibilité pour un container d'accéder aux volumes d'autres containers

Travailler avec le Docker Hub

- Stockage d'images officielles et personelles
- Fonctionnalité payante : stockage d'images privées
- Possibilité de build automatiques sur GitHub/Bitbucket
- Possibilité de mise en place de webhook pour appeler une URL quand votre image est mise à jour

Docker en réseau

- Par défaut, Docker écoute sur un socket UNIX
- Possibilité de mettre en écoute sur un socket TCP, n'importe qui peut alors accéder à l'API Docker
- Docker peut utiliser TLS et autoriser les certificats signés par une CA donnée.
- On peut utiliser un reverse proxy pour plus de contrôle sur les connections

```
upstream docker {
    server unix:/var/run/docker.sock fail_timeout=0;
}

server {
    listen       4242;
    server_name  docker.dsi.cnrs.fr;

    location / {

        allow 10.0.0.5;
        deny all;

        proxy_pass http://docker;
        proxy_redirect off;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }
```

Gestions des ressources sous Docker

- Limiter la mémoire d'un container :
  - docker run –m=512m monimage
  - Possibilité d'autoriser plus de mémoire que disponible
- Ajuster le CPU attribué à un container
  - docker run –c=2048 monimage
  - La valeur par défaut est 1024, mais ne représente rien de spécial
  - Valeur relatives aux autres containers, ne limite pas en soi un container

Gestions des ressources sous Docker

- Docker utilise les cgroups pour gérer l'allocation de ressources
- Problème : les outils utilisés pour contrôler l'utilisation mémoire/CPU ignorent souvent la notion de cgroup (top, free…)
- Outil en python : cgroup-utils (https://github.com/peo3/cgroup-utils)
- systemd a des commandes pour manipuler les cgroups

Gestion des ressources sous Docker

- Si on veut changer l'allocation RAM ou CPU d'un container en route… pas de commande docker pour ça !
- Pas de commande non plus dans les cgroup-utils
- Commande systemd:

$ sudo systemctl set-property docker-4be96b853089bc6044b29cb873cac460b429cfcbdd0e877c0868eb2a901dbf80.scope CPUShares=512

- La commande systemctl show permet d'afficher la liste des paramètres pour un cgroup donné

Gestion des ressources sous Docker

- Ou alors changer à la main le fichier

/sys/fs/cgroup/cpu/system.slice/docker-adc24c2ecbcf67f4f661e645cb7824e1b5066206c366d13b61cf60c5c876bb0e.scope/cpu.shares

- Possibilité de limiter les I/O disques
- Possibilité de limiter la bande passante disponible en jouant sur iptables
- Pas encore d'implémentation de quota disque

Logs sous Docker

- Typiquement, un container log sur /dev/stdout
- Accès aux logs avec la commande docker logs [container], docker logs –f [container] pour suivre les logs en temps réel (comme tail –f)
- Pratique pour debugger ou vérifier que l’application s’est bien lancée
- Moins pratique pour exploiter les logs de multiples containers !

Logs sous Docker

- 3 méthodes possibles pour industrialiser les logs
  - Configurer un syslog dans chaque container pour envoyer les logs à un serveur distant
  - Définir un volume utilisé par le container pour stocker les logs, et utiliser le syslog de l'host
  - Utiliser un container spécialisé pour ce travail : https://github.com/progrium/logspout

Sécurité sous Docker

- Docker est là pour faciliter le déploiement d'applications, pas pour sécuriser (le demon tourne en root!)
- Exemple d'exploit permettant de sortir d'un container : http://stealth.openwall.net/xSports/shocker.c récupère le /etc/shadow
- Des failles existent, dernière version impérative (actuellement 1.3.2)
- Que faire pour éviter les débordements ?

Sécurité sous Docker

- Considérer le container comme s'il tournait directement sur l'hôte
- Réduire au minimum les exécutions en root dans le container
- Faire très attention à la provenance des images que vous utilisez
- Si possible, utilisez SELinux ou AppArmor pour policer vos containers
- Docker tourne très bien sur une VM !

Sécurité sous Docker

- https://github.com/jpetazzo/docker2docker : Un docker qui lance une VM qui lance un docker !

Sécurité sous Docker

- Les développeurs sont bien conscients de la problématique sécurité
- Depuis la version 1.3, on peut labelliser des containers pour SELinux/AppArmor
- En cours de développement : signature des images sur le docker registry (tech preview dans la 1.3)

Optimiser ses dockerfiles

- Choisir la distribution qui servira de base
  - A l'heure actuelle, debian:wheezy est la plus petite distro avec 85mo
  - Busybox propose un système minimal à 2.4mo (utilisé par exemple par logspout)

- Les packages pour compiler prennent beaucoup de place
- Utilisez un fichier .dockerignore (par exemple pour ignorer .git ou autres fichiers inutiles à l'image elle-même)
- Vous pouvez réutiliser vos images et profiter de l'héritage
  - Création d'une image « webserver »
  - Création de deux images « wsphp4 » et « wsphp5 » qui dépendente de webserver
  - Mettez à jour l'image webserver, quand vous rebuilderez les wsphp ils utiliseront la nouvelle version de webserver.

Optimiser ses dockerfiles

- Chaque ligne d'un dockerfile créé une image intermédiaire ou « layer »
- Possibilité d'utiliser les layers intermédiaires pour « debugger » une image
- Chaque layer est reprise en lecture seule, un fichier supprimé sera en pratique caché
- Construire les commandes pour éviter ce genre de cas et limiter le nombre de layers, mais en gardant le fichier lisible

Développements autour de Docker

- Docker Registry : héberger en local ses propres images (pas très pratique pour l'instant)
- Docker Machine : Manipuler des hôtes Dockers à distance
- Docker Swarm : Gère la clusterisation des containers
- Docker Compose : Permet de définir un ensemble de containers qui forment une application cohérente

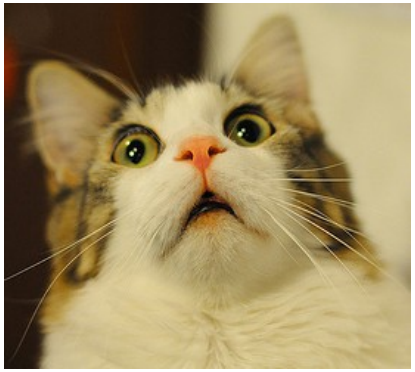
Développements autour de Docker

- Docker fournit une API REST qui permet à n'importe quel programme de s'interfacer avec
- https://docs.docker.com/reference/api/remote_api_client_libraries/
- Différents UI pour Docker et pour Docker Registry
- Wormhole, Ambassador : permettent de lier des containers sur des hôtes différents
- Plugins pour logiciels divers (Jenkins, Nagios, etc)

Le futur

- Docker est très jeune, mais a une importante dynamique derrière
- Supportés par différents clouds : AWS, Google Cloud, Azure…
- Peut-être supplanté par un concurrent (Rocket vient d'être lancé)
- Ces systèmes répondent à une vrai besoin

- Bientôt Docker sous Windows !



Références

- https://goldmann.pl/blog/2014/09/11/resource-management-in-docker/
- http://blog.tutum.co/2014/10/22/how-to-optimize-your-dockerfile/
- https://blog.logentries.com/2014/03/the-state-of-logging-on-docker/
- http://fabiokung.com/2014/03/13/memory-inside-linux-containers/