Introduction et bonnes pratiques de la conteneurisation ANF Mathrice CIRM 2022

Laurent Facq, Romain Théron

21 Novembre 2022

mathrice

Concepts

Il existe plusieurs façons d'exploiter de manière efficiente un matériel informatique pour lancer plusieurs services (processus) :

- Façon naïve : sur un même OS, sur même matériel
- Façon étanche : sur plusieurs OS, sur un même matériel (virtualisation)
- Façon séparée : isoler au niveau OS les applications sur un même OS (chroot ou conteneurs)
  - sur un OS natif
  - sur un OS virtualisé

Nous allons ici nous pencher sur l'approche séparée avec des conteneurs. C'est une réponse de plus en plus présente car elle est plus efficiente en terme de ressources et apporte maintenant des garanties en terme de sécurité.

mathrice

Concepts

Un conteneur permet de faire tourner une application et ses dépendances de façon isolée

- chroot :
  - change la racine d'un processus
  - isolation uniquement du filesystem, on peut toujours voir les processus externes etc. . .
- namespaces
  - différent types : pid, réseau, mount, hostname etc. . .
  - sécurité bien plus poussée qu'avec chroot

mathrice

la sécurité dans les conteneurs (cgroups (v2), user namespaces - L. Facq

Immutabilité

- Un conteneur n'est jamais modifié
- On le supprime et on en lance un nouveau à la place, avec la configuration voulue, ou pour faire une mise à jour
- Une modification faite sur le conteneur qui tourne n'existera plus lors de son prochain lancement
- On peut revenir à la version précédente en cas de souci
- On utilise donc des volumes séparés pour les données persistantes (base de données, fichiers html, php etc. . .)

mathrice

Reproductibilité

- Un conteneur Docker sera toujours identique (si on utilise la même version de l'image) quel que soit le système sur lequel il tourne (mêmes versions de programmes, librairies)
- Cela peut être utile également pour développer, car ça permet de s'assurer que tout le monde travaille bien avec le même environnement
- Ou bien pour la recherche où la reproductibilité est importante

mathrice

OCI

L'Open Container Initiative est une structure qui définit des standards concernant les conteneurs.

- Image : format d'une image (différentes couches, liste de ces couches, paramètres d'exécution)
- Runtime : comment lancer une image extraite sur le disque
- Distribution : une API pour standardiser la distribution du contenu

mathrice

Runtime

Le runtime sert à créer et faire fonctionner le conteneur. Il en existe plusieurs types :

- bas niveau : servent à créer et lancer le conteneur (ex runc, crun, containerd)
- haut niveau : utilisent les runtimes de bas niveau et ajoutent des fonctionnalités par dessus comme la gestion des images, l'orchestration, la personnalisation du conteneur... (ex docker, podman)

Principes Images

- Une image est un ensemble de couches reliées par un système de fichiers (aufs, overlayfs, zfs etc. . .)
- Chaque instruction dans un Dockerfile crée une nouvelle couche
- Chaque couche a un identifiant unique (le hash de l'archive des fichiers contenus dans la couche)
- L'image de base (ubuntu, alpine, scratch etc. . .) est la première couche
- Les suivantes contiennent la différence entre la couche inférieure et elles-mêmes (layer)
- Les couches sont partagées entre plusieurs images si elles sont identiques et donc stockées une seule fois dans le cache

mathrice

exemple Dockerfile

```
# image de base
FROM debian:11

# l'instruction RUN permet d'executer une commande dans l'image de base
# ici l'installation de nginx
RUN apt update && apt install -y nginx

# on copie un fichier depuis le reertoire courant dans l'image
COPY index.html /var/www/html/

# Le port qui sera expose par l'application (indication)
EXPOSE 80/tcp

# la commande qui sera lancee quand on instancie le conteneur
# le processus doit rester au premier plan sinon le conteneur s'arrete
CMD ["nginx", "-g", "daemon off;"]
```

Documentation de référence :
https://docs.docker.com/engine/reference/builder/

Bonnes pratiques 1/2

Par défaut les programmes dans le conteneur tournent en tant que root. Pour changer ça on peut utiliser la directive USER. Du coup il n'est plus possible d'écouter sur un port inférieur à 1024.

```
EXPOSE 8000/tcp
USER 1001
CMD ["nginx", "-g", "daemon off;"]
```

Attention certains programmes sont prévus pour être lancés en tant que root, donc il faudra ajuster des permissions/changer des paramètres. Cf le TP de demain :-)

mathrice

Bonne pratiques 2/2

- Ne pas installer plus de paquets que nécessaire (mais certains peuvent être utiles pour débugger)
- Minimiser le nombre de couches, par exemple en utilisant && dans les instructions RUN
- Ne pas hésiter à mettre des instructions RUN sur plusieurs lignes pour plus de lisibilité (en mettant un backslash à la fin de la ligne)
- Utiliser les multi-stage builds : On compile le programme dans une première image, puis on copie le résultat dans une image plus légère (souvent utilisé avec go et rust qui compilent statiquement). `https://docs.docker.com/develop/develop-images/dockerfile-best-practices/`

mathrice

Construction d'une image

- On se place dans le répertoire qui contient le Dockerfile et le fichier index.html
- `docker build -t nginx_anf:v1 .`

- l'option -t permet de donner un nom et un tag à notre image
- le point indique qu'on construit l'image depuis le répertoire courant

Registres

- Un registre est une bibliothèque contenant de multiples images au format OCI
- On peut donc télécharger (pull) une image depuis un registre pour pouvoir ensuite instancier un conteneur
- Il y a un registre local à la machine qui est utilisé par défaut quand on construit une image
- Il existe plusieurs registres publics, il est également possible d'en monter un
- On peut pousser (push) ses propres images dans un registre public ou privé

mathrice

Réseau

Par défaut docker propose trois types de réseau :

- bridge (par défaut) : les conteneurs dans un réseau de type bridge peuvent communiquer entre eux et vers l'extérieur, mais ne sont pas accessibles en dehors de l'hôte
- host : le conteneur utilise le réseau de l'hôte
- none : le conteneur n'a pas accès au réseau

Le réseau de type bridge est celui qui est généralement utilisé.

mathrice

Réseau - Bridge

- Il est possible de créer plusieurs réseaux de type bridge pour isoler des groupes de conteneurs
- Pour rendre un conteneur accessible depuis l'extérieur, on fait du mapping de port. Un port de l'hôte est redirigé vers le conteneur voulu. Par ex : mon conteneur écoute sur le port 80 et je redirige le port 8080 de l'hôte vers ce port
- Docker utilise iptables pour permettre aux conteneurs de se connecter à l'extérieur et pour faire le mapping des ports

mathrice

Fin

Questions ?