Limiter la taille de vos containeurs docker

Utiliser des une images plus petite offre des avantages tels que des temps d’importation et de téléchargement plus courts. Cela peut permettre de raccourcir les temps d’exécution de vos CI ou du démarrage de vos pods Kubernetes. Donc voici quelques conseils pour y parvenir :

1. Éviter d’installer des outils inutiles

Pour protéger vos applications contre le piratage, n’installez pas d’outil inutile. Par exemple n’allez pas installer netcat qui permet de créer un tunnel TCP, même si certains de ces outils pourraient être utiles pour déboguer vos applications.

2. Utiliser l’option -no-install-recommends si possible

Pour éviter les packages recommandés qui sont inutiles et pour n’utiliser que les dépendances principales, ajoutez l' option –no-install-recommends à la commande d’installation. Pour alpine linux, apk possède une option –virtuel permettant de détruire ces packages dés qu’ils ne sont plus nécessaires.

3. Limiter le nombre de layers

Évitez les layers inutiles. Chaque instruction RUN dans un Dockerfile ajoute une nouvelle couche à votre image. La combinaison de plusieurs commandes RUN en une seule à l’aide de l’option && aide à réduire le nombre de couches.

4. Utilisez .dockerignore

Docker possède l’equivalent du .gitignore, c’est à dire le fichier .dockerignore. Cela permet d’exclure les fichiers qui ne sont pas utiles pour votre image réduisant ainsi sa taille.

```
pattern:
{ term }
term:
'*' matches any sequence of non-Separator characters
'?' matches any single non-Separator character
'[' [ '^' ] { character-range } ']'
character class (must be non-empty)
c matches character c (c != '*', '?', '\\', '[')
'\\' c matches character c

character-range:
c matches character c (c != '\\', '-', ']')
'\\' c matches character c
lo '-' hi matches character c for lo &lt;= c &lt;= hi

additions:
'**' matches any number of directories (including zero)
'!' lines starting with ! (exclamation mark) can be used to make exceptions to exclusions
'#' lines starting with this character are ignored: use it for comments
```

Exemple:

```
# ignore all markdown files (md) beside all README*.md other than README-secret.md
*.md
!README*.md
README-secret.md
```

5. Utiliser les images officielles alpine

Le principal avantage en utilisant des images à base de distribution Alpine est leur taille. Regardez le tableau suivant :

DISTRIBUTION /	VERSION /	TAILLE /
Debian	bullseye-slim	80MB
OracleLinux	8	246MB
Ubuntu	21.04	80 MB
Alpine	3.14	5.6MB

Autre avantages:

- l'installation des packages alpine est bien plus rapide.
- compte tenu de la taille mini peu de paquets installés et donc plus sécurisé car contenant moins de failles possibles

6. Utiliser le plus souvent possible le multi stage

Le multi-stage est une fonctionnalité intéressante introduite avec la version 17.05 de Docker. Il est possible de décrire plusieurs étapes dans un même fichier Dockerfile, Chaque stage commençant par une commande FROM. Je vous conseille de nommer chaque stage en ajoutant AS , cela permet d'y faire référence dans un autre stage. Il est possible de copier une partie du système de fichier d'un stage avec la commande COPY en utilisant le paramètre –from=. La première application possible est de construire l'image en trois stages :

- Le premier décris la partie commune des deux autres
- Le second permet de construire votre application
- Le troisième permet de ne copier que le résultat de la construction du second. Par exemple pour les images python utilisant pip, qui consomme énormément de place lors de la compilation des packages, il est possible de garder le résultat de la compilation en utilisant un environnement virtuel.

Un exemple : Création d'un image Ansible :

Non optimisée

```
FROM alpine:3.10.2

RUN apk update && apk upgrade
RUN apk add --no-cache python3 openssl
RUN apk add --no-cache --virtual .build-deps python3-dev gcc ca-certificates libffi-dev openssl-dev build-base
RUN pip3 install --no-cache-dir --upgrade pip ansible
RUN apk del .build-deps

docker images
REPOSITORY                    TAG             IMAGE ID         CREATED           SIZE
ansible                       0.1             5e43ab54b9a0     56 seconds ago    393MB
```

Optimisée

```
FROM alpine:3.10.2 as base
RUN apk update && apk --no-cache upgrade && apk add --no-cache python3 openssl

FROM base as builder
RUN apk add --no-cache --virtual .build-deps python3-dev gcc ca-certificates libffi-dev openssl-dev build-base
RUN python3 -m venv /opt/venv
 Make sure we use the virtualenv:
ENV PATH="/opt/venv/bin:$PATH"
RUN pip3 install --no-cache-dir ansible

FROM base

COPY --from=builder /opt/venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"

docker images

REPOSITORY                    TAG             IMAGE ID         CREATED           SIZE
ansible                       0.2             304de2e8b235     51 seconds ago    165MB
```

On voit que l’image ne fait plus que 165MB contre 393MB pour la version non optimisée. Ce principe peut être appliqué à plein de cas possible. Par exemple je l’utilise pour générer ce blog en copiant que le code html généré par hugo dans une image nginx-alpine. Résultat l’image ne fait que 9.3 MB

7. Analyser vos images avec Dive

Dive est l’outil d’analyse de container.

Pour installer Dive il suffit sur la page release du projet et de télécharger le package correspondant à votre Distribution.

Pour analyser une image il suffit de taper :

```
dive <nom de votre image>
```

Il est possible de lancer le build depuis Dive

```
dive build -t <votre tag> .
```

dive
Image not found or type unknown

C’est la touche [CTRL] qui permet de changer de fonctionnalité. Par exemple pour sortir [CTRL]+[C]

Mots clés :

docker, devops, tutorials, kubernetes,

Autres Articles

- Installer ansible (python3) sur windows avec CygWin
- Ansible l'outil de gestion de configuration
- Démarrer avec les containers Docker

- Automatiser le déploiement de votre blog Hugo avec wercker

Revision #2
Created 13 October 2021 20:19:16 by garfi
Updated 13 October 2021 20:19:48 by garfi