Docker – Création d’une image

Archi Open source

Dockerfile

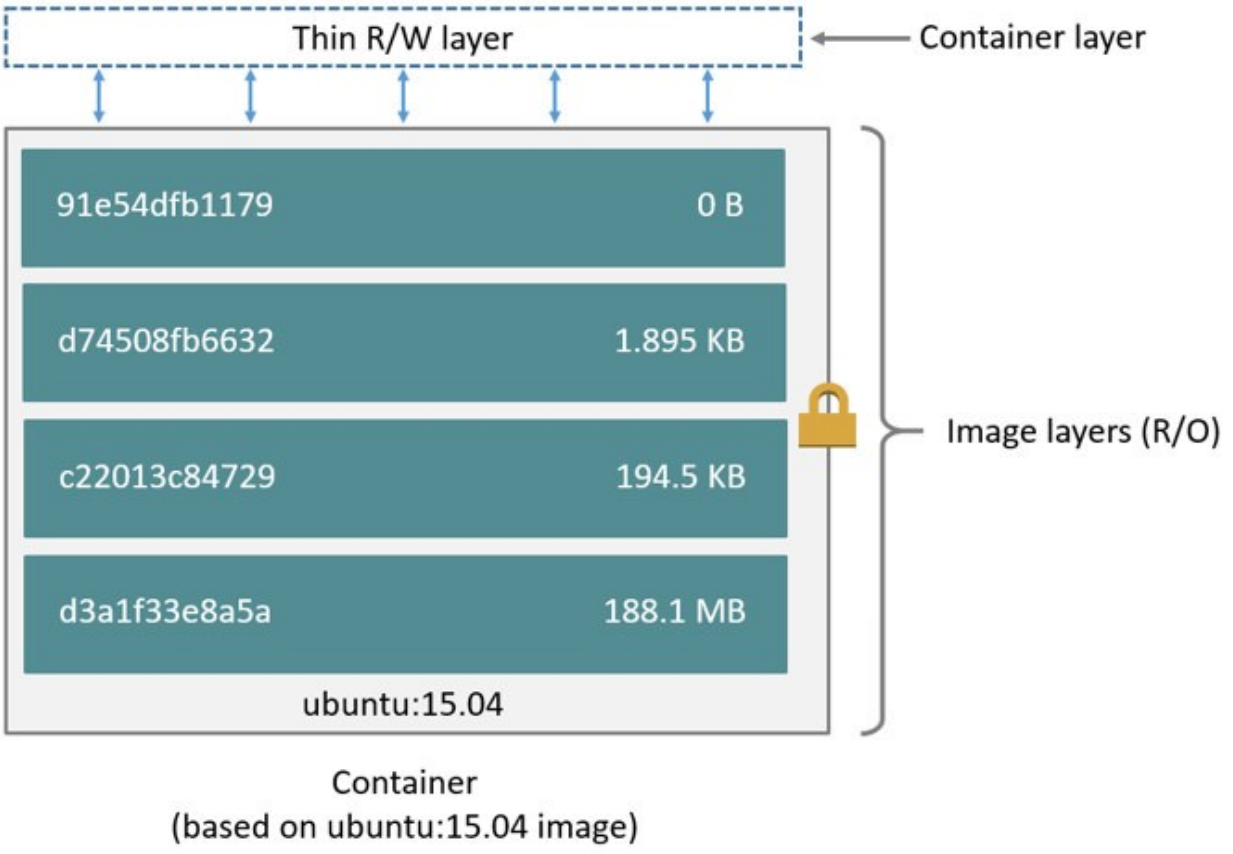
- Permet de créer sa propre image Docker
- Fichier permettant de donner les instructions de création de l’image

FROM ubuntu:15.04	→ Image de base
COPY . /app	→ Copie du répertoire courant
RUN make /app	→ Exécution d’une commande
CMD python /app/app.py	→ Commande s’exécutant au démarrage du conteneur

Images - Layers

- Une image est constituée d’une série de layers
- Un layer représente une instruction dans l’image docker
- Chaque layer est en lecture seule **sauf** le dernier layer
- Chaque commande du Dockerfile précédent crée un layer

Images - Layers



Images - Layers

- Le dernier layer est créé lors de la création d’un conteneur
  - Enregistre tous les changements fait durant l’exécution du conteneur
  - Ce dernier layer **n’affecte donc pas l’image** de base

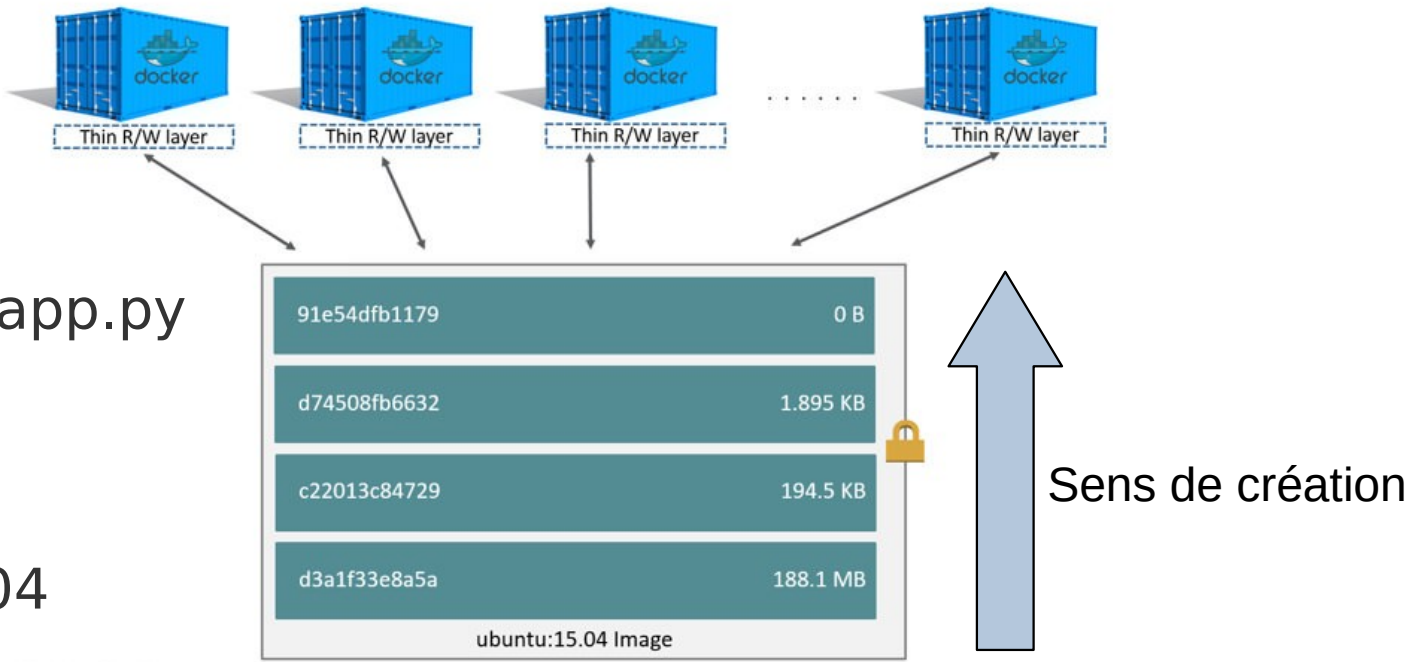
Images - Layers

CMD python /app/app.py

RUN make /app

COPY . /app

FROM ubuntu:15.04



Images - Layers

- Seul Run, Copy et Add créent des layers ayant une répercution sur la taille
- Optimisations dans la création d’une image
- Réutilisation des layers précédents si inchangés
- Structure app

```
├── requirements.txt
└── src
    └── server.py
```

Images - Layers

V1 :

```
FROM python:3.8

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY src/ .

CMD [ "python", "./server.py" ]
```

V2 :

```
FROM python:3.8

COPY src/ .

COPY requirements.txt .

RUN pip install -r requirements.txt

CMD [ "python", "./server.py" ]
```

Images - Layers

- Va directement exécuter le code « COPY »
- Si les « requirements » ne changent pas, pas besoin de les réinstaller

- => Permet d’aller plus vite lors de la création du conteneur

Images - Layers

- Faire les images les plus petites possible
- Avec le moins de layer possible
- => Créer des conteneurs éphémères
  - Peut être facilement stoppé, détruit, rebuidé, remplacé
  - Avec un minimum de configuration
  - Faire des conteneurs Stateless

Images – Build context

- Le build context est le répertoire courant lors de la création de l’image
- Permet d’éviter de créer des contexte trop lourd
  - Exclure les dépendances
  - Exclure les fichiers « parasites »
- => Créer un .dockerignore (même principe que le .gitignore)
  - Tous les paths se trouvant dans ce fichier seront exclu du build context

Docker – Multi-Stage build

- Que faire si on doit compiler son code C ?
- Inclure GCC dans le conteneur ?

  Solution => Le Multi-Stage build
- Permet de créer un conteneur intermédiaire pour la compilation
- Copie du résultat dans le conteneur « final »

Docker – Multi-Stage Build

```
FROM golang:1.11-alpine AS build   #Image de « build »
… (Copie projet)
RUN dep ensure -vendor-only        #Install dépendances
COPY . /go/src/project/
RUN go build -o /bin/project
FROM scratch                       # Creation de l'image finale
COPY --from=build /bin/project /bin/project
CMD ["--help"]
```

- L'image finale **n'a qu'un layer** (copy)

Découpler une application

- Un conteneur doit faire tourner un seul service
- Pour une application web nous avons généralement 3 conteneurs
  - Serveur Web
  - Base de donnée
  - Cache (Redis)
- Pour faire communiquer les conteneurs entre eux on utilisera des networks

Dockerfile

- FROM : Image de base
- LABEL : Donner une petite note à une image
- RUN : Exécute une commande
- CMD : Exécuter une commande par défaut au lancement du conteneur
- EXPOSE : Donne une **information** sur les ports que l’image écoute
- ENV : Définit une variable d’environement

Dockerfile

- ADD : Transfert des fichiers vers l’image. Soit URL soit dossier local
- COPY : Transfert des fichiers vers l’image depuis un dossier local
- ENTRYPOINT : Idem que CMD à part qu’on ne sait pas le bypass. Utile lorsqu’on veut utiliser un conteneur comme un exécutable
- VOLUME : Crée un point de montage. Accessible via /var/lib/docker/volumes (dépends des OS…)

Dockerfile

- USER : Définit le UID et GID pour les commandes RUN, CMD, ENTRYPOINT
- WORKDIR : Change le dossier de travail courant (un peut commet un CD)
- ...

Exercice

- Voir sur Moodle