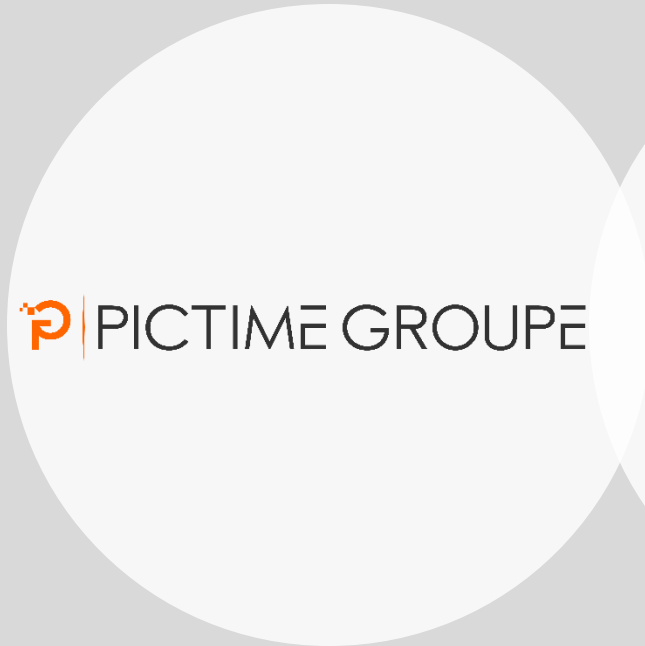


Conteneurisation

Dockerfile

2 Octobre 2018

Le Dockerfile

Le "code source" du conteneur

Sa syntaxe est presque aussi riche que la commande "run"

```
# se baser sur l'image officielle centos dans sa dernière version
FROM centos

# definir une variable d'environnement
ENV MY_ENV_VAR=MY_ENV_VALUE

# lancer une commande shell (ici update)
RUN yum -y upgrade
# Comme toutes les commandes de ce fichier, elle ne sera exécutée qu'a la création de l'image
# pas à chaque instanciation

# copier le contenu du répertoire scripts (Situé au même niveau que le dockerfile) à la racine
COPY scripts/ /

# donner les droits +x sur tous les scripts shell à la racine
RUN chmod +x /*.sh

# definir "/run.sh" comme script de lancement du conteneur
ENTRYPOINT ["/run.sh"]
```

```
$ # construire le conteneur en le nommant mongroupe/monconteneur:latest
$ docker build -t mongroupe/monconteneur:latest .
```

Exercice

Construire un premier conteneur

Les Layers

Exemple de build

```
$ docker build -t build4layers .
Sending build context to Docker daemon  2.048kB
Step 1/5 : FROM nginx:latest
---> 2073e0bcb60e
Step 2/5 : RUN apt-get update
---> Using cache
---> 91675d758968
Step 3/5 : RUN apt-get install -y curl
---> Using cache
---> 7ccf6e6f305e
Step 4/5 : RUN rm -rf /var/lib/apt/lists/*
---> Using cache
---> aee7f848a717
Step 5/5 : RUN rm -rf /etc/apt/sources.list.d/nginx.list
---> Using cache
---> 67976c3b5b09
Successfully built 67976c3b5b09
Successfully tagged build4layers:latest
```

```
$ docker history build4layers
IMAGE               CREATED             CREATED BY                                      SIZE
67976c3b5b09        5 minutes ago       /bin/sh -c rm -rf /etc/apt/sources.list.d/ng…   0B
aee7f848a717        5 minutes ago       /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
7ccf6e6f305e        5 minutes ago       /bin/sh -c apt-get install -y curl              7.86MB
91675d758968        6 minutes ago       /bin/sh -c apt-get update                       17.4MB
2073e0bcb60e        3 weeks ago         /bin/sh -c #(nop)  CMD ["nginx" "-g" "daemon…   0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  STOPSIGNAL SIGTERM           0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>           3 weeks ago         /bin/sh -c ln -sf /dev/stdout /var/log/nginx…   22B
<missing>           3 weeks ago         /bin/sh -c set -x     && addgroup --system -…   57.5MB
<missing>           3 weeks ago         /bin/sh -c #(nop)  ENV PKG_RELEASE=1~buster     0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  ENV NJS_VERSION=0.3.8        0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  ENV NGINX_VERSION=1.17.8     0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  LABEL maintainer=NGINX Do…   0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           3 weeks ago         /bin/sh -c #(nop) ADD file:ba0c39345ccc4a882…   69.2MB
```

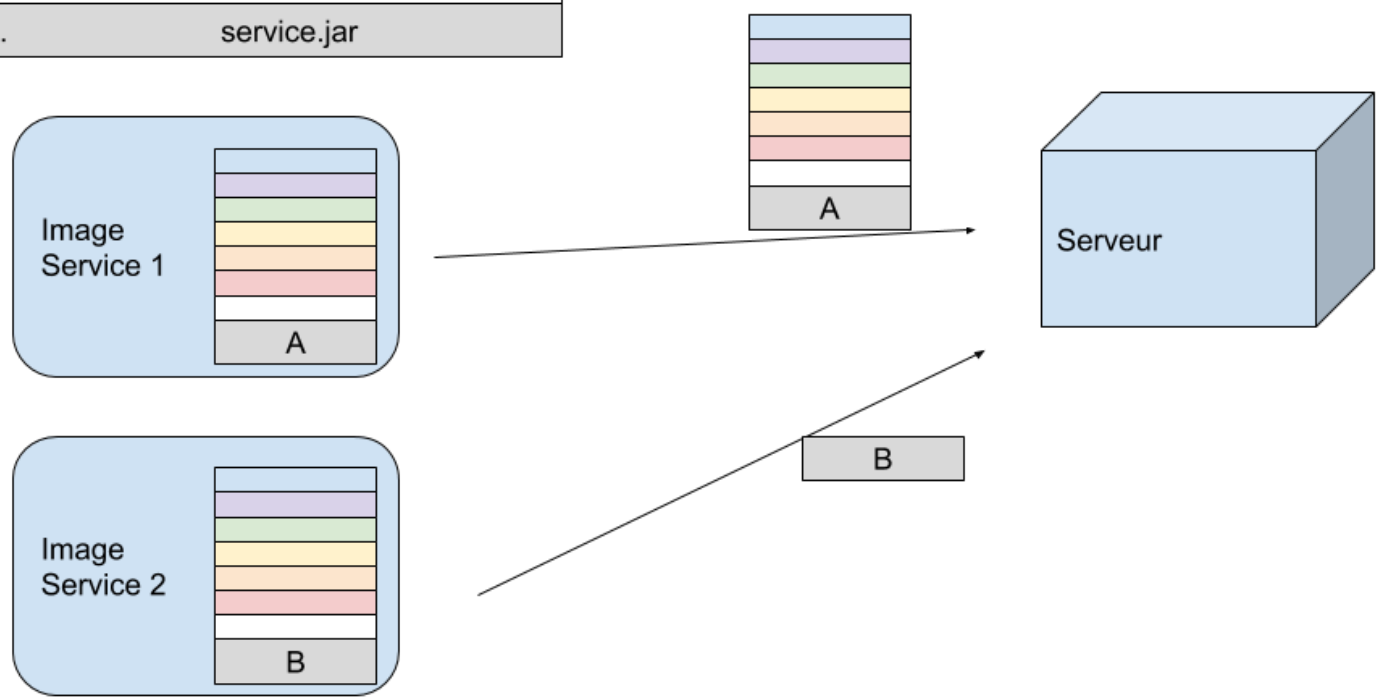
```
FROM nginx:latest

RUN apt-get update
RUN apt-get install -y curl
RUN rm -rf /var/lib/apt/lists/*
RUN rm -rf /etc/apt/sources.list.d/nginx.list
```

Intérêt :

FROM	alpine ?
RUN ...	installation de java ?
RUN ...	installation d'un outil ?
EXPOSE ...	8080 ?
VOLUME ...	logback.xml ?
ADD ...	script de lancement ?
ENTRYPOINT ...	java -jar ... ?
ADD ...	service.jar

1. Accélérer les builds (en réutilisant les layers)
2. Limiter la bande passante consommée



Build Multistage

Principes

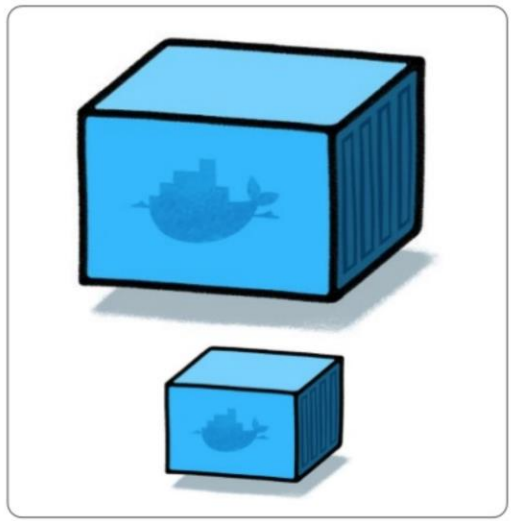
Découper la construction d'une image permet :

- De simplifier chaque partie
- De gagner en espace et en nombre de layers
- De simplifier le CI/CD
- Avoir des images adaptées aux environnements

1
2

```
# Dockerfile
# build stage
FROM buildbase as build
...
...
...

# production ready stage
FROM runbase
...
COPY --from=build
/artifact /app
```



Exemple :

```
FROM golang:1.7.3 AS builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go    .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app .
CMD ["./app"]
```

Conteneur multi-process

Supervisord

En temps normal :

- Un conteneur = un processus sur le Pid 1
- Si le processus de Pid 1 tombe : le conteneur s'arrête
- Si on éteint le conteneur, docker envoi le sigint/sigterm au processus de Pid 1

Dans un conteneur multi-process :

- Le Pid 1 est pris par le "process controller"
- Le contrôleur s'occupe de surveiller la présence des processus "métiers"
- Si un processus métier tombe, le contrôleur le relance ou se coupe (selon la configuration)
- Si le contrôleur se coupe, le conteneur s'arrête
- Si on éteint le conteneur, docker envoi le sigint/sigterm au contrôleur qui le fait suivre aux processus métiers