TP2: Autour de Docker

Ifrim Vasile-Alexandru
M2 SSI

Créer une image Docker minimale pour l'application CMatrix en utilisant un Dockerfile. Les sources du projet CMatrix sont disponibles sur le dépôt GitHub : https://github.com/abishekvashok/cmatrix. Cet exercice vous permettra de pratiquer les concepts de construction d'une image Docker optimisée et légère à partir des sources d'un projet open-source.

1. Cloner le Dépôt CMatrix

— Commencez par cloner le dépôt GitHub du projet CMatrix sur votre machine locale :
— Familiarisez-vous avec le contenu du dépôt, en particulier le fichier README.md et le processus de compilation.

```
$ git clone https://github.com/abishekvashok/cmatrix.git
$ cd cmatrix
```

2. Créer un Dockerfile Minimal

— Votre objectif est de créer une image Docker aussi petite que possible tout en incluant tout ce qui est nécessaire pour compiler et exécuter CMatrix.
— Utilisez une multi-stage build pour séparer l'étape de compilation de l'étape d'exécution afin de réduire la taille de l'image finale.
— Dans la première étape du Dockerfile, utilisez une image de base légère qui contient un compilateur C, comme alpine ou debian-slim, pour compiler CMatrix.
— Dans la deuxième étape, utilisez une image de base encore plus petite, telle que alpine, pour exécuter uniquement le binaire compilé.

The commands of this dockerfile specify how to set up the environment, which dependencies to install, how to compile the application, and what needs to be included in the final image. For our docker image to be as minimal as possible we will also take the following steps:

- minimize layers by packing RUN commands together; this reduces the number of intermediate images (but not the final image)
- don't install recommended dependencies when getting packages
- remove package information and not cache the index locally
- after building, clean binaries and object files
- check dockerfile for any more sugesstions on https://www.fromlatest.io

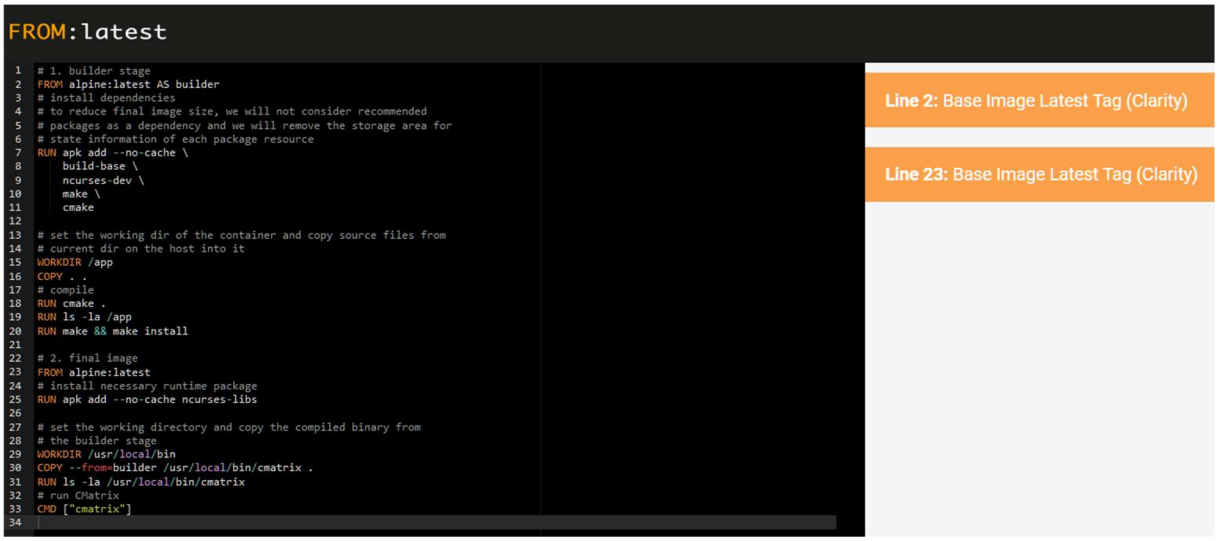
```
# 1. builder stage
FROM alpine:latest AS builder
# install dependencies
# to reduce final image size, we will not consider recommended
# packages as a dependency and we will remove the storage area for
# state information of each package resource
RUN apk add --no-cache \
    build-base \
    ncurses-dev \
    make \
    cmake

# set the working dir of the container and copy source files from
# current dir on the host into it
WORKDIR /app
COPY . .
# compile
```

```
RUN cmake . && make && make install

# 2. final image
FROM alpine:latest
# install necessary runtime package
RUN apk add --no-cache ncurses-libs

# set the working directory and copy the compiled binary from
# the builder stage
WORKDIR /usr/local/bin
COPY --from=builder /usr/local/bin/cmatrix .
# run CMatrix
CMD ["cmatrix"]
```



By correctly applying the suggestions learned, the last indications got from FromLatest are to not use the 'latest' tag: it makes deployments unpredictable

3. Construire l'Image Docker

— Construisez l'image Docker en utilisant la commande suivante:

```
docker build -t cmatrix .
```

— Vérifiez que l'image a été construite et notez sa taille:

```
alex@alex-VMware:~/Reseau/TP2/cmatrix$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
cmatrix      latest    411703d6ffec   3 minutes ago        8.41MB
```

4. Tester l'Image Docker

— Lancez un conteneur basé sur l'image construite pour tester CMatrix.

```
docker run --rm -it cmatrix
```

— Vous devriez voir l'animation de la matrice s'afficher dans le terminal.

Resources:

- hackernoon.com/tips-to-reduce-docker-image-sizes-876095da3b34
- stackoverflow.com/questions/43634109/how-can-i-reduce-the-size-of-docker-images
- stackoverflow.com/questions/49118579/alpine-dockerfile-advantages-of-no-cache-vs-rm-var-cache-apk