

Docker **Multi Stage** Builds





- **Images plus légères**
- **Unique fichier de build**

Pourquoi plus légères ?



push, pull, run



container != virtualisation légère

Étapes de fabrication :
Compilation des dépendances
+
Traitements intermédiaires
➔ Image cible

Examples



Simple Docker

```
FROM golang

WORKDIR /go/src/app
COPY . .

RUN go-wrapper download
RUN go-wrapper install

CMD ["go-wrapper", "run"]
```

737MB

Pattern Builder

```
FROM golang

WORKDIR /go/src/ht_docker_multi_stage
COPY . .

RUN go-wrapper download

CMD ["go", "build", "-v"]
```

```
FROM gcr.io/distroless/base

COPY ht_docker_multi_stage /
CMD ["/ht_docker_multi_stage"]
```

Pattern Builder (2)

`build:`

```
docker build -t ht_build_and_package-builder -f Dockerfile.build .  
docker run --rm -v $$PWD:/go/src/ht_docker_multi_stage \  
    ht_build_and_package-builder
```

17.5MB

Multi Stage

```
FROM golang as builder

WORKDIR /go/src/ht_docker_multi_stage
COPY . .

RUN go-wrapper download
RUN go-wrapper install

FROM gcr.io/distroless/base

COPY --from=builder /go/bin/ht_docker_multi_stage /
CMD ["/ht_docker_multi_stage"]
```

17.5MB

NodeJS + Bootstrap

NGinx

```
FROM nginx:1.13-alpine
EXPOSE 80
COPY conf/default.conf /etc/nginx/conf.d/default.conf
COPY build/ /usr/share/nginx/html
```

React build

```
FROM node:8-alpine as dependencies
WORKDIR /usr/src/app
COPY package.json yarn.lock /usr/src/app/
RUN yarn install
```

```
FROM node:8-alpine as builder
WORKDIR /usr/src/app
COPY . .
COPY --from=dependencies /usr/src/app/node_modules /usr/src/app/node_modules
RUN yarn build
```

```
FROM nginx:1.13-alpine
EXPOSE 80
COPY conf/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /usr/src/app/build/ /usr/share/nginx/html
```


Sass build

```
FROM ruby:2-alpine3.6 as sass
RUN apk add --no-cache --virtual build-dep build-base \
    && gem install sass --no-doc \
    && apk del build-dep && rm -rf /var/cache/apk/*
COPY scss /usr/src/app
WORKDIR /usr/src/app
RUN sass bootstrap.scss > App.css
```

#...

```
FROM node:8-alpine as builder
WORKDIR /usr/src/app
COPY . .
COPY --from=dependencies /usr/src/app/node_modules /usr/src/app/node_modules
COPY --from=sass /usr/src/app/App.css /usr/src/app/src/App.css
RUN yarn build
```

Limites

Limites

- Pas de partage de `stage`
- `Lint` pas à jour
- Docker `>= 17.05`

Limites

Pas de partage de `stage`

Comment partager `dependencies` entre container de dev et container intermédiaire de prod ?

- Partage de step au travers du cache docker ?
- Génération des `Dockerfile` ?

Limites

Linter pas à jour

```
$ docker run --rm -it --privileged -v $PWD:/root/ \
  projectatomic/dockerfile-lint dockerfile_lint lint
-----ERRORS-----

Line 1: -> FROM ruby:2-alpine3.6 as sass
ERROR: Invalid parameters for command..
Reference -> https://docs.docker.com/engine/reference/builder/

Line 12: -> FROM node:8-alpine as dependencies
ERROR: Invalid parameters for command..
Reference -> https://docs.docker.com/engine/reference/builder/
```

Limites

Docker **≥ 17.05**

Kubernetes < 1.8  docker ≤ 1.12

Avantages

Avantages

- Toute la construction présente dans un unique fichier
- Suppression d'outils de build externes (type `Makefile`)
- `docker build` + `docker run`, that's all folks

Thanks 🙏

