

Fabrication d'une image Docker

La base

La plus immédiate manière de construire une image de conteneur est d'écrire un fichier `Dockerfile`.

[Référence](#)

Il existe d'autres formats spécialisés :

- [Earthly](#), organisation similaire aux Makefile
- [Mopy](#), pour des projets Python
- [Gockerfile](#), pour des projets Go

DONT NOUS NE PARLERONS PLUS

Un `Dockerfile` est une recette de cuisine, qui permet d'indiquer à `docker build` les instructions pour fabriquer une image.

Créez un répertoire `mon-phpinfo`, quelque part, puis un fichier `Dockerfile` avec le contenu suivant :

```
FROM php:8.2-apache
RUN echo '<?php phpinfo();' > /var/www/html/index.php
```

Fabriquez l'image :

```
cd mon-phpinfo
docker build --tag phpinfo .
docker run --detach --publish 80:80 phpinfo
```

`http://localhost/`

⚠ Attention au TLS !

Le navigateur risque d'essayer d'afficher

`https://localhost/`

au lieu de

`http://localhost/` !

Assurez-vous de forcer le protocole non sécurisé

Image de base

```
mkdir -p projets/ma-base # adaptez selon votre organisation
cd projets/ma-base
code Dockerfile # code ou un vrai éditeur de texte
```

```
# syntax=docker/dockerfile:1.17.1
FROM alpine:3.22.1
```

```
docker build --tag mon-alpine:1 .
```

Choix d'une distribution Linux comme base

- Alpine
- Debian GNU/Linux
- Ubuntu
- aucune (from `scratch`)

Alpine

Alpine a été conçue pour être une base de conteneur.

Avantages :

- faible empreinte disque

- faible occupation mémoire
 - faible surface d'attaque
 - immutabilité par défaut
-

Inconvénients :

- la bibliothèque C moins compatible (`musl libc`)
 - gestion de paquets particulière (`apk`)
 - utilise `busybox` par défaut
(version limitée du shell et des utilitaires de base)
-

Debian GNU/Linux

Une vénérable distribution, loin d'être la plus antique, elle reste une référence parmi les administrateurs systèmes.

Avantages :

- large support
 - stable
 - second plus important système de gestion de paquets
 - shell `bash` par défaut
-

Inconvénients :

- un peu plus gros qu'Alpine
-

Ubuntu

Distribution basée sur Debian, prend les avantages et inconvénients

Avantages :

- support commercial
 - supporte des versions logiciel plus récentes que Debian
-

Algorithme de choix

1. ai-je besoin d'une distribution ?
 2. la distribution m'est-elle connue ?
 3. les prérequis désirés sont-ils disponibles ?
 4. quid de la compatibilité ?
-

En vrai...

On choisit Alpine ou Debian.

L'expérience et la pratique vous donnerons les clés pour choisir de manière plus fine. En première intention, restez simple, choisissez une distribution qui vous sera familière !

L'usage

La manière de fabriquer une image dépend en partie de son usage

1. développement
2. test
3. production

Mais au début, on fera simple.

Une application front

Une application qui affiche la webcam

<https://gitlab.com/miwoteam/cheese>

Environnement de développement

```
FROM nginx:1.29.1-alpine3.22
```

```
ADD --chmod=0444 \
https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.21/lodash.js \
s \
```

```
/usr/share/nginx/html  
COPY index.html index.css index.js /usr/share/nginx/html/
```

Warning

`ADD` est dangereux car il peut chercher des ressources sur l'Internet.
Préférez-lui `COPY`

Environnement de déploiement

```
FROM nginx:1.29.1-alpine3.22  
  
ENV  
LODASH_MIN_URL=https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4  
.17.21/lodash.min.js  
ADD --chmod=0444 \  
    $LODASH_MIN_URL \  
    /usr/share/nginx/html  
RUN touch /usr/share/nginx/html/favicon.ico  
COPY index.html index.css index.js /usr/share/nginx/html/
```

Build context

Le contexte de fabrication est l'ensemble des fichiers contenu dans un répertoire.

Classiquement, c'est le répertoire qui contient le `Dockerfile`.

Tous les fichiers contenus dans ce répertoire sont transmis au daemon Docker pour construire l'image.

Chaque instruction du Dockerfile représente une couche de l'image.

.dockerignore

Sauf à utiliser un `.dockerignore` pour limiter les fichiers transmis.

```
# Le client Docker lit ces fichiers
Dockerfile
.dockerignore

# L'historique peut être immense
.git

# Contient souvent des secrets
.env
```

Votre IDE favori sait générer ce fichier pour votre type de projet. N'hésitez pas à utiliser cette fonctionnalité.

Multistage

Il est possible d'avoir plusieurs recettes dans notre `Dockerfile` :

```
# syntax=docker/dockerfile:1.17.1
FROM nginx:1.29.1-alpine3.22 AS dev
# Suite des instructions du premier étage

FROM nginx:1.29.1-alpine3.22 AS live
# Suite des instructions du second étage
```

On construit en indiquant la cible :

```
docker build --target dev --tag cheese:dev .
```

Par défaut, c'est le dernier `FROM` qui initie le dernier stage.

Aller plus loin

- [Documentation officielle](#)