Prise en compte du cache

Enoncé

1. Modifiez le code du serveur pong de l'exercice précédent. Vous pouvez par exemple ajouter une instruction qui loggue une chaine de caractère.

2. Construisez une nouvelle image en la taguant *pong:1.1*

3. Qu'observez-vous dans la sortie de la commande de build ?

4. Modifiez le *Dockerfile* pour faire en sorte que les dépendances ne soient pas rebuildées si un changement est effectué dans le code. Créez l'image *pong:1.2* à partir de ce nouveau *Dockerfile*.

5. Modifiez une nouvelle fois le code de l'application et créez l'image *pong:1.3*. Observez la prise en compte du cache

Correction

1. Nous modifions ici le serveur *nodejs* que nous avons réalisé dans l'exercice précédent.

Le code suivant est le nouveau contenu du fichier *pong.js*

```
var express = require('express');
var app = express();
app.get('/ping', function(req, res) {
    console.log("received");
    res.setHeader('Content-Type', 'text/plain');
    console.log("pong"); // <-- commentaire ajouté
    res.end("PONG");
});
app.listen(80);
```

2. Nous buildons l'image *pong:1.1* avec la commande suivante

```
$ docker image build -t pong:1.1 .
Sending build context to Docker daemon  4.096kB
Step 1/6 : FROM node:10.15-alpine
```

```
 ---> 288d2f688643
Step 2/6 : COPY . /app/
 ---> a240da42d95d
Step 3/6 : RUN cd /app && npm install
 ---> Running in 36ac860620fa
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN pong@0.0.1 No description
npm WARN pong@0.0.1 No repository field.
npm WARN pong@0.0.1 No license field.

added 48 packages from 36 contributors and audited 121 packages in 2.677s
found 0 vulnerabilities

Removing intermediate container 36ac860620fa
 ---> 2982e15fc190
Step 4/6 : WORKDIR /app
 ---> Running in 1fac7cda598d
Removing intermediate container 1fac7cda598d
 ---> 57aed72b8e57
Step 5/6 : EXPOSE 80
 ---> Running in 25fb2c98ff1f
Removing intermediate container 25fb2c98ff1f
 ---> c4e45d3564ed
Step 6/6 : CMD ["npm", "start"]
 ---> Running in 2d59763522c9
Removing intermediate container 2d59763522c9
 ---> b79378cb0043
Successfully built b79378cb0043
Successfully tagged pong:1.1
```

3. Nous observons que chaque étape du build à été effectuée une nouvelle fois.
   Il serait intéressant de faire en sorte qu'une simple modification du code source ne déclenche pas le build des dépendances.

4. Une bonne pratique, à suivre dans l'écriture d'un *Dockerfile*, est de faire en sorte que les éléments qui sont modifiés le plus souvent (code de l'application par exemple) soient positionnés plus bas que les éléments qui sont modifiés moins fréquemment (liste des dépendances).

On peut alors modifier le Dockerfile de la façon suivante:

```
FROM node:10.15-alpine
COPY package.json /app/package.json
RUN cd /app && npm install
COPY . /app/
WORKDIR /app
EXPOSE 80
CMD ["npm", "start"]
```

L'approche suivie ici est la suivante:

- copie du fichier *package.json* qui contient les dépendances
- build des dépendances
- copie du code source

On rebuild alors une nouvelle fois l'image en lui donnant le tag *pong:1.2*

```
docker image build -t pong:1.2 .
Sending build context to Docker daemon  4.096kB
Step 1/7 : FROM node:10.15-alpine
 ---> 288d2f688643
Step 2/7 : COPY package.json /app/package.json
 ---> 76d962709f8e
Step 3/7 : RUN cd /app && npm install
 ---> Running in 5222e854e356
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN pong@0.0.1 No description
npm WARN pong@0.0.1 No repository field.
npm WARN pong@0.0.1 No license field.

added 48 packages from 36 contributors and audited 121 packages in 3.473s
found 0 vulnerabilities

Removing intermediate container 5222e854e356
 ---> bb0128d009fa
Step 4/7 : COPY . /app/
 ---> 94ee45b832d3
Step 5/7 : WORKDIR /app
 ---> Running in 9a5aa7d959bf
Removing intermediate container 9a5aa7d959bf
 ---> 4844dfe00bb8
Step 6/7 : EXPOSE 80
 ---> Running in 4f15c6985369
Removing intermediate container 4f15c6985369
 ---> 74e4cb35d1a6
Step 7/7 : CMD ["npm", "start"]
 ---> Running in e7b0e8ccb278
Removing intermediate container e7b0e8ccb278
 ---> b2a6c8ccde7e
Successfully built b2a6c8ccde7e
Successfully tagged pong:1.2
```

Un cache est créé pour chaque étape du build.

5. On fait la modification suivante dans le fichier *pong.js*.

```
var express = require('express');
```

```
var app = express();
app.get('/ping', function(req, res) {
    console.log("received");
    res.setHeader('Content-Type', 'text/plain');
    console.log("ping-pong"); // <-- modification du commentaire
    res.end("PONG");
});
app.listen(80);
```

Puis l'on build une nouvelle fois l'image, en la nommant cette fois *pong:1.3*.

```
docker image build -t pong:1.3 .
Sending build context to Docker daemon  4.096kB
Step 1/7 : FROM node:10.15-alpine
 ---> 288d2f688643
Step 2/7 : COPY package.json /app/package.json
 ---> Using cache
 ---> 76d962709f8e
Step 3/7 : RUN cd /app && npm install
 ---> Using cache
 ---> bb0128d009fa
Step 4/7 : COPY . /app/
 ---> 2bc7975fcc81
Step 5/7 : WORKDIR /app
 ---> Running in 9453ad69aaff
Removing intermediate container 9453ad69aaff
 ---> 688b51e2a1f1
Step 6/7 : EXPOSE 80
 ---> Running in 4b165815a81b
Removing intermediate container 4b165815a81b
 ---> 57df5214e6de
Step 7/7 : CMD ["npm", "start"]
 ---> Running in 8097d98ceed0
Removing intermediate container 8097d98ceed0
 ---> 466e5916fe4f
Successfully built 466e5916fe4f
Successfully tagged pong:1.3
```

Nous observons ici que le cache est utilisé jusqu'au step 3 (---> Using cache). Il est ensuite invalidé lors du step 4 car le daemon Docker a détecter le changement de code que nous avons effectué. La prise en compte du cache permet souvent de gagner beaucoup de temps lors de la phase d build.