□□ Dockerizer et publier

Dans ce petit exemple j'ai voulu dockerizé une image de Debian avec `openjdk-18-jdk`. Généralement quand on dockerize une quelque chose c'est très équivalent à l'installer en *bare* sur un serveur donc connaitre Linux est un prérequis pour pouvoir faire ceci. Mais il y a quelques difficultés en plus quand on le crée pour Docker.

- Il n'y a pas de système de init, donc pas de systemd. Par exemple pour utiliser php-fpm et nginx, il faut manuellement les installer, les configurer et les démarrer dans le entrypoint. Alors que normalement on a pas besoin de ce soucier de ça
- Il est bien de pouvoir connaitre alpine Linux car c'est une distribution light très souvent utilisée pour créer des images Docker pour des raisons d'optimisation. C'est différent nottament car beaucoup de fichiers binaires Linux ne sont pas compatibles avec cette distribution et que les configurations de logiciels commun tel que php-fpm ou nginx peuvent être différentes sur celle ci
- On peut optimiser encore plus mais cela nécessite d'avoir de très bonnes connaissances du système que l'on utilise pour savoir ce qui peut être supprimé

□□ Tester des choses dans un pod

On sais que l'on veut partir d'une image debian. Mais on ne sais peut'être pas les instructions exacte pour y installer openjdk-18-jdk. Donc on peut créer un pod Debian et l'utiliser comme environement de test.

```
docker run --rm -it docker.io/library/debian
```

Maintenant, on a une distribution Debian pour y faire nos tests.

□□ Créer le dockerfile

Pour cela on va d'abord créer un dossier vide pour notre projet

```
mkdir docker-openjdk
cd docker-openjdk
```

Maintenant on peut créer un fichier `Dockerfile` et y ajouter ceci:

```
# On part d'une image de debian
FROM debian

# On ajoute le répertoire SID de debian, met a jour et installe le paquet openjdk-18-jdk
RUN echo "deb http://deb.debian.org/debian sid main contrib non-free" >> /etc/apt/sources.list
RUN apt-get update
# On fait également attention a ce que aucune commande ne nécessite une intéraction de l'utilisateur avec "-y"
RUN apt-get install -y openjdk-18-jdk

# On indique la commande principale qui va être lançée quand on crée un pod avec l'image
CMD [ "/bin/bash" ]
```

Une fois cela fait on peut build l'image, on va mettre comme nom `docker.io/<username>/debian-openjdk-18` car c'est l'addresse vers laquelle on va publier l'image plus tard. Et on va aussi utiliser `--no-cache` dans les cas où le build dépend de resources externes car sinon Docker ne les reprendera pas.

```
sudo docker build . --no-cache -t docker.io/<username>/debian-openjdk-18
```

□□ Publier dans un registery

Maintenant, on peut aller sur Docker Hub, créer un nouveau repo. Puis ensuite on va se connecter via docker et publier le tout

```
sudo docker login docker.io
sudo docker push docker.io/<username>/debian-openjdk-18
```

Maintenant l'image a bien été créée à partir de nos instructions et a été publiée sur le registery de Docker Hub. On peut maintenant tester si l'image fonctionne correctement en lançant un nouveau pod.

```
docker run --name debian-openjdk-18-test -it docker.io/<username>/debian-openjdk-18
```

Nous pouvons maintenant par exemple l'utiliser dans un GitLab-CI ou encore un docker-compose par exemple.

```
image: <username>/debian-openjdk-18
```

Revision #3
Created 26 April 2023 19:09:35 by SnowCode
Updated 4 May 2023 06:34:32 by SnowCode