Etat de l'art à la mi-projet de semestre Docker and embedded systems - Où comment ne pas cross compiler Docker sur ARM

Gary Marigliano

13 avril 2016

Table des matières

Introduction

1 Contexte

Ce document s'inscrit dans le cadre du projet de semsetre Docker and embedded systems actuellement réalisé par moi-même. Un des buts de ce projet est de cross-compiler Docker à partir de ses sources pour produire un binaire exécutable sur un Odroid XU3 (ARMv7).

Lien : `https://github.com/krypty/docker_and_embedded_systems`

Il est important de noter que la vitesse de développement de Docker est assez halluciante. En effet, sur Github (`https://github.com/docker/docker`) les commits se succèdent à vitesse grand V. Entre chaque version de Docker qui sortent environ tous les mois, il est courant d'avoir plus de 3000 commits qui ont été *pushés*. Tout ceci pour dire qu'à la lecture de ce document, il est quasiment sûr que certaines pistes explorées soient définitivement obsolètes ou au contraire deviennent la voie à suivre du à une mise à jour quelconque.

2 Objectifs

De manière plus précise, ce projet vise à maitriser les parties suivantes :

1. Construction d'un système Linux capable de faire tourner Docker et son *daemon* en utilisant Buildroot. Pour générer le dit système, on dispose d'un *repository* Gitlab herbergé à la Haute Ecole de Fribourg. *TODO* ajouter lien !
2. Cross-compilation de Docker et de son daemon, capable de faire tourner des containers

L'objectif de ce document est d'énumérer les différentes techniques tentées pour (cross-)compiler Docker sur une cible ARM. De cette manière, le lecteur, en cas de reprise du projet ou par simple curiosité, aura une idée des pistes à explorer ou à éviter.

Objectif 1 - Construction d'un système GNU/Linux Docker-ready

Dans cette partie, on verra les ingrédients et pistes à suivre pour conçevoir un système construit à partir de Buildroot capable de faire tourner Docker et son *daemon*.

TODO

— parler du repository de HEIA-FR utilisé pour construire le système
— mettre en annexe le script utilisé pour générer la SD
— expliquer brièvement comment ajouter/modifier les modules du kernel et renvoyer le lecteur au PDF du prof.
— parler du script qui permet de savoir si les pre-requis de Docker sont remplis sur le système actuel et parler de l'obligation d'avoir bash installé.

Objectif 2 - Techniques de compilation essayées

3 La manière officielle

C'est la manière recommandée et qui, un jour, sera celle qu'il faudra employer. Mais aujourd'hui, elle ne permet que de cross compiler un binaire ARM Docker qui n'embarque pas le *deamon*.

3.1 Principe utilisé

Pour compiler Docker de la manière officiellement supportée, on doit utiliser Docker. En effet, le Makefile fourni va lancer un container Docker qui va contenir un système d'exploitation ainsi que tous les pré-requis et dépendances puis lancer la compilation de Docker à l'intérieur de ce container.

3.2 Cheminement général

Sur une machine GNU/Linux

```
git clone https://github.com/docker/docker
cd docker
git checkout v1.10.3 -b tmp_build # vous pouvez remplacez v1.10.3 par la
↪ dernière version (tag) stable
make build
make binary
make cross # pour générer le binaire ARM
```

Le binaire se trouve dans le dossier ./bundle.

3.3 Schéma

3.4 Limitations

Actuellement, il est possible de générer un binaire Docker x64 et ARM mais seule l'architecture x64 intègre le *deamon* nécessaire à la création de containers.

Le binaire ARM est dit CLIENT_ONLY dans le sens où il peut être le client d'un *deamon* Docker remote (instancié sur une autre machine).

Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe

limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations. Il existe certaines limitations.