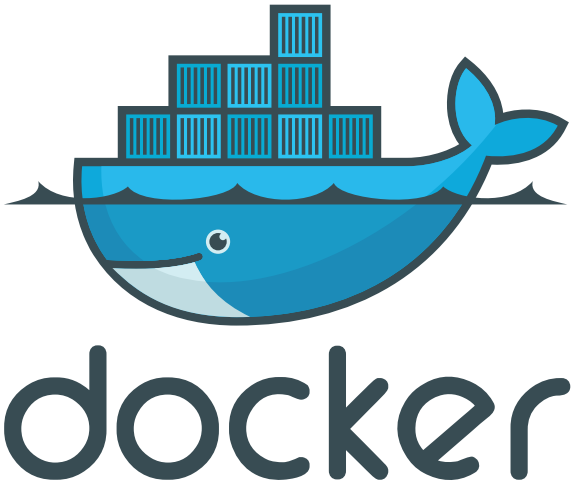

MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Projet de semestre Docker and embedded systems

Auteur :
Gary Marigliano

Encadrant :
Jean-Roland Schüler

Contact :
gary.marigliano@master.hes-so.ch

Mandant :
Haute École d'ingénierie et d'architecture de Fribourg

Version 0.0.1
3 mai 2016

Historique

Version	Date	Auteur(s)	Modifications
0.0.1	02.05.16	Gary Marigliano	Création du document

1. Résumé du document

TODO

Table des matières

1 Résumé du document

2 Introduction **2**
2.1 Contexte . . . 2
2.2 Objectifs . . . 2

3 Présentation de Docker **3**
3.1 Introduction . . . 3
3.2 Containers vs machines virtuelles . . . 3
3.3 Système de fichiers en couche . . . 3
3.4 Dockerfile . . . 3
3.5 Contraintes liées au monde de l'embarqué . . . 3

4 Matériel utilisé et mise en place de la cible **4**
4.1 La carte ODROID-XU3 Lite . . . 4
4.2 Installation . . . 4

5 Objectif 3 - TODO **5**
5.1 Situation actuelle . . . 5
5.2 Structure de la suite du document . . . 5

6 Configuration du système d'exploitation hôte **6**
6.1 Passage en revue du benchmark de sécurité : CIS Docker 1.11.0 Benchmark . . . 6
6.1.1 Ne pas utiliser AUFS . . . 6
6.1.2 User namespace . . . 6
6.1.3 Interdiction de la communication réseau entre containers . . . 6
6.2 Séparation des données Docker dans une partition chiffrée . . . 6
6.3 TODO . . . 6

7 Création et utilisation des images Docker **7**

8 Utilisation des containers **8**

9 Déroulement du projet **9**
9.1 Planning initial . . . 9
9.2 Planning final . . . 9

10 Proposition d'améliorations vis à vis du travail précédent **10**

Appendices **12**

A Installation de Archlinux ARM sur ODROID-XU3 Lite **13**
A.1 Micro SD Card Creation . . . 13
A.2 eMMC Module Creation . . . 14

2. Introduction

2.1 Contexte

Ce document est le rapport de fin de projet de semestre Docker and embedded systems. 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 Lite (ARMv7). De plus, une partie concernant la sécurité de Docker est également traitée.

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

Ce projet de semestre s'inscrit dans une certaine continuité avec les projets de semestre et de bachelor de M. Loic Bassang [**?**]. Plusieurs pistes intéressantes avaient en effet été mentionnées dans ces projets là notamment une partie concernant la sécurité et Docker. Ainsi, ce rapport fera parfois des parallèles avec ces documents.

Il est important de noter que la vitesse de développement de Docker est assez hallucinante. 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.

2.2 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 hébergé à la Haute École d'ingénierie et d'architecture de Fribourg
2. Cross compilation de Docker et de son *daemon*, capable de faire tourner des containers
3. Comprendre, analyser et évaluer l'aspect sécurité de Docker dans le cadre d'une utilisation avec une carte embarquée

Les deux premiers points ont été traités dans un précédent rapport appelé "État de l'art à la mi-projet de semestre Docker and embedded systems - Ou comment ne pas cross compiler Docker sur ARM".

Ce document se concentre, dès lors, sur le dernier point ainsi que sur le déroulement global du projet.

3. Présentation de Docker

Remarque : Si le lecteur a déjà lu le rapport "État de l'art à la mi-projet de semestre Docker and embedded systems - Ou comment ne pas cross compiler Docker sur ARM", il ne lui est pas nécessaire de relire ce chapitre sachant qu'il s'agit du même contenu.

3.1 Introduction

TODO

3.2 Containers vs machines virtuelles

TODO

3.3 Système de fichiers en couche

TODO dire voir rapport Loic + ajouter en bibtex

3.4 Dockerfile

TODO

3.5 Contraintes liées au monde de l'embarqué

TODO

4. Matériel utilisé et mise en place de la cible

Ce chapitre présente le matériel utilisé dans le projet ainsi que son installation et sa configuration de base.

Afin de réaliser ce projet, une carte embarquée ODROID-XU3 Lite a été mise à disposition afin d'y faire tourner Docker.

4.1 La carte ODROID-XU3 Lite

Cette carte possède les caractéristiques suivantes [**?**] :

- Samsung Exynos5422 Cortex™-A15 1.8Ghz quad core and Cortex™-A7 quad core CPUs
- Mali-T628 MP6(OpenGL ES 3.0/2.0/1.1 and OpenCL 1.1 Full profile)
- 2Gbyte LPDDR3 RAM at 933MHz (14.9GB/s memory bandwidth) PoP stacked
- eMMC5.0 HS400 Flash Storage
- USB 3.0 Host x 1, USB 3.0 OTG x 1, USB 2.0 Host x 4
- HDMI 1.4a for display

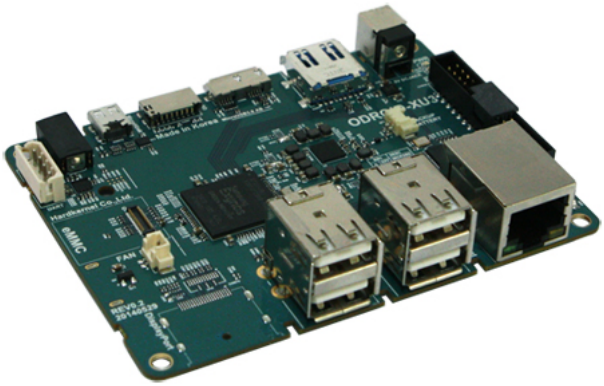

FIGURE 4.1 – ODROID-XU3 Lite

4.2 Installation

Initialement, il était prévu de générer un système d'exploitation minimal qui aurait été capable de faire tourner Docker et des containers. Malheureusement, il n'a pas été possible de cross compiler Docker *et son daemon* afin de lancer des containers sur ce système minimal. Plus d'informations sont disponibles dans le rapport État de l'art à la mi-projet de semestre Docker and embedded systems - Ou comment ne pas cross compiler Docker sur ARM.

Ainsi, il a été décidé, de la même manière que pour le travail de bachelor précédent, d'utiliser une distribution GNU/Linux proposant Docker dans ses packages. Le choix s'est donc porté sur **Archlinux ARM** [**?**].

Sur la page wiki de la distribution (`https://archlinuxarm.org/platforms/armv7/samsung/odroid-xu3`), on peut suivre un guide de génération de la carte SD qui contient le système d'exploitation. Ce guide est disponible à l'appendice A.

5. Objectif 3 - TODO

5.1 Situation actuelle

TODO : rappeler les objectifs, en particulier l'objectif courant (le 3)...

TODO : dire qu'on part d'une distribution Archlinux ARM et qu'on utilise un Odroid XU3 car pas réussi à cross compiler Docker. Tout comme cela avait été fait pour le travail de Bachelor précédent.

5.2 Structure de la suite du document

Pour ce projet, il a été décidé d'étudier la question de la sécurité avec Docker avec une approche en couches. A peu à la manière du modèle OSI [1] en réseau, chaque couche représente un ensemble de fonctionnalités qui, dans le cas de ce projet, doit faire l'objet d'une évaluation de la sécurité.

L'étude de la sécurité de Docker a donc été séparée avec les couches arbitraires suivantes :

- Compilation et installation de Docker : en particulier les options de compilation
- Configuration du système d'exploitation hôte : configuration du kernel, configuration des options de lancement de Docker, etc.
- Création et utilisation des images Docker : Bonnes pratiques et contraintes liées au monde de l'embarqué
- Utilisation des containers

Remarque : Chacune de ces couches fait l'objet d'un chapitre dans ce rapport excepté le premier point : Compilation et installation de Docker. En effet, celui-ci n'est pas traité car, comme annoncé précédemment, la cross compilation de Docker sur un système ARM n'a pas aboutie. L'effort est alors concentré sur les autres points.

1. Modèle OSI : `https://fr.wikipedia.org/wiki/Mod%C3%A8le_OSI`

6. Configuration du système d'exploitation hôte

TODO TODO TODO TODO TODO TODO TODO TODO TODO Dans ce chapitre, on présente diverses bonnes pratiques et configurations dans le but de sécuriser Docker et/ou le système l'hébergeant.

Parmi ces techniques, on peut citer :

— TODOchaptertitle
— TODO

6.1 Passage en revue du benchmark de sécurité : CIS Docker 1.11.0 Benchmark

TODO décrire ce que c'est ce bench, énumerer les points testés et en explorer un certain nombre

6.1.1 Ne pas utiliser AUFS

TODO

6.1.2 User namespace

TODO

6.1.3 Interdiction de la communication réseau entre containers

TODO

6.2 Séparation des données Docker dans une partition chiffrée

TODO

6.3 TODO

TODO

7. Création et utilisation des images Docker

TODO

8. Utilisation des containers

TODO

9. Déroulement du projet

9.1 Planning initial

TODO

9.2 Planning final

TODO

10. Proposition d'améliorations vis à vis du travail précédent

TODO : passer en revue et critique positivement le travail de Bachelor précédent. Dire que ce n'est pas une critique négative mais apporter un avis supplémentaire et plus récent (Docker évoluant beaucoup)

Bibliographie

[1] Center for Internet Security. CIS Docker 1.11.0 Benchmark, avril 2016. `https://benchmarks.cisecurity.org/tools2/docker/cis_docker_1.11.0_benchmark_v1.0.0.pdf`.

[2] Adrian Mouat. *Using Docker*. O'Reilly Media, 2005.

[3] Prakhar Srivastav. Docker for beginners, avril 2016. `http://prakhar.me/docker-curriculum/`.

Appendices

A. Installation de Archlinux ARM sur ODROID-XU3 Lite

Remarque : Ce guide requiert l'utilisation d'un ordinateur sous GNU/Linux.

Source : `https://archlinuxarm.org/platforms/armv7/samsung/odroid-xu3`

A.1 Micro SD Card Creation

Replace sdX in the following instructions with the device name for the SD card as it appears on your computer.

1. Zero the beginning of the SD card :

```
dd if=/dev/zero of=/dev/sdX bs=1M count=8
```

2. Start fdisk to partition the SD card :

```
fdisk /dev/sdX
```

3. At the fdisk prompt, create the new partitions :
   a. Type o. This will clear out any partitions on the drive.
   b. Type p to list partitions. There should be no partitions left.
   c. Type n, then p for primary, 1 for the first partition on the drive, and enter twice to accept the default starting and ending sectors.
   d. Write the partition table and exit by typing w.
4. Create and mount the ext4 filesystem :

```
mkfs.ext4 /dev/sdX1
mkdir root
mount /dev/sdX1 root
```

5. Download and extract the root filesystem (as root, not via sudo) :

```
wget
 ↪ http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-xu3-latest.tar.gz
bsdtar -xpf ArchLinuxARM-odroid-xu3-latest.tar.gz -C root
```

6. Flash the bootloader files :

```
cd root/boot
sh sd_fusing.sh /dev/sdX
cd ../..
```

7. (Optional) Set the MAC address for the onboard ethernet controller :
   a. Open the file root/boot/boot.ini with a text editor.
   b. Change the MAC address being set by the setenv macaddr command to the desired address.

c. Save and close the file.

8. Unmount the partition :
   umount root
9. Set the boot switches on the ODROID-XU3 board to boot from SD :
   a. With the board oriented so you can read the ODROID-XU3 on the silkscreen, locate the two tiny switches to the left of the ethernet jack.
   b. The first switch (left) should be in the off position, which is down.
   c. The second switch (right) should be in the on position, which is up.
10. Insert the micro SD card into the XU3, connect ethernet, and apply 5V power.
11. Use the serial console (with a null-modem adapter if needed) or SSH to the IP address given to the board by your router.
    — Login as the default user alarm with the password alarm.
    — The default root password is root.

A.2 eMMC Module Creation

1. Attach the eMMC module to the micro SD adapter, and plug that into your computer.
2. Follow the above steps to install Arch Linux ARM, and boot the board with the eMMC still attached to micro SD adapter, plugged into the SD slot in the board.
3. Re-flash the bootloader to the protected boot area of the eMMC module :

```
cd /boot
./sd_fusing.sh /dev/mmcblk0
```

4. Power off the board :

```
poweroff
```

5. Remove the micro SD adapter, and detach the eMMC module.
6. Set the boot switches on the ODROID-XU3 board to boot from eMMC :
   a. With the board oriented so you can read the ODROID-XU3 on the silkscreen, locate the two tiny switches to the left of the ethernet jack.
   b. The first switch (left) should be in the on position, which is up.
   c. The second switch (right) should be in the on position, which is up.
7. Connect the eMMC module to the XU3, ensuring you hear a click when doing so, connect ethernet, and apply 5V power.
8. Use the serial console (with a null-modem adapter if needed) or SSH to the IP address given to the board by your router.
   — Login as the default user alarm with the password alarm.
   — The default root password is root.