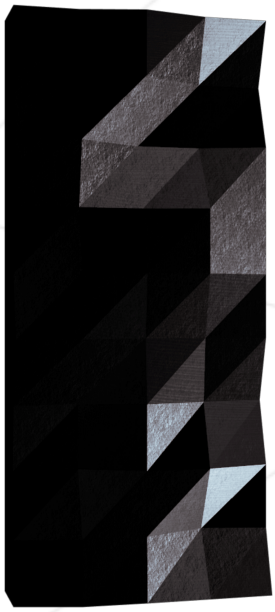

Docker On Mac

Sur Linux c'est trop simple

Jean-Jacques MOIROUX jmoiroux@student.42.fr
42 staff pedago@42.fr

Résumé: Ce projet a pour but de vous faire découvrir les bases de Docker. Docker a de multiples utilisations, ne vous limitez pas à l'utilisation de Docker conseillée dans divers tutoriaux sur le web. À la fin de ce sujet, vous saurez écrire des Dockerfiles.

Table des matières

Chapitre I

Préambule

Docker from Wikipedia (http ://fr.wikipedia.org/wiki/Docker)

Il existe différents noms communs pour désigner la profession qui exerce cette activité selon les régions du monde.

Le mot débardeur attesté dès le xvie siècle en français classique1 est probablement formé du préfixe dé et de la racine bard (dont l'étymologie reste controversée) : bard désignant depuis l'ancien français (xiie siècle) une « sorte de brancard à claire-voie utilisé pour le transport de fardeaux »1. Littéralement, le débardeur est l'ouvrier portuaire qui procède au débardage (ou débarde), c'est-à-dire qui décharge à quai toutes sortes de marchandises.

Le mot docker, contraction de l'anglais dockworker (littéralement, « ouvrier du quai »), est attesté pour la première fois en français à la fin du xixe siècle et s'est peu à peu répandu dans la majorité des ports du monde au cours du xxe siècle avec la mondialisation des échanges et la suprématie économique du monde anglo-saxon. Toutefois, le terme débardeur, moins usité de nos jours en Europe, s'est maintenu plus largement au Québec2.

Débardeurs au port de Cap-Haïtien : le transport avec peu de moyens mécaniques est encore pratiqué dans les pays en développement.

Dockers chargement de la cargaison en sac - MV Rothstein, Port-Soudan en 1960 À Marseille entre autres, le terme de portefaix était utilisé pour désigner la corporation qui œuvrait sur le Vieux Port pendant la marine à voile qui n'avait pas seulement un rôle de déchargement ou de chargement de la marchandise mais aussi une spécialisation dans connaissance de la qualité du produit manipulé, les portefaix représentaient aussi les intérêt du négociant "Maitre Portefaix" et avaient aussi le rôle d'acheminer la marchandises dans les magasins3..

Le terme aconier, parfois écrit acconier (ou l'anglicisme stevedore), recouvre des notions proches mais néanmoins différentes d'un point de vue juridique. De fait, l'aconier est l'entrepreneur dont le métier consiste à préparer matériellement et juridiquement les opérations de réception, de déplacement et d'entreposage de marchandises transportées par voie maritime et est donc l'employeur des débardeurs.

Chapitre II

Sujet

II.1 Consignes

- Le sujet sera corrigé par des humains.
- Les containers lors de la soutenance seront hébergés sur votre machine en local.
- Interdiction de faire une soutenance avec un container sur une machine distante, relire la règle juste au-dessus au cas où vous n'auriez pas compris cette règle-ci.
- Il est fortement conseillé pour chaque mission de fournir un script .sh pour lancer le container, le correcteur POURRA vérifier ce script de lancement.
- Vous pouvez utiliser Ubuntu, Debian, fedora, CentOS, ArchLinux ou autres du moment que l'arborescence de répertoire suivante est respectée : `/home/login (/home/user)`. Vous trouverez des images officielles sur Docker Hub.

II.2 Mission 0 - Installer Docker sur Mac

- Rendre vos fichiers pour soutenance dans mission00 (s'il y en a).
- Installer Docker sur votre session Mac. Il est fort probable que vous ayez besoin d'utiliser **Brew** et **Boot2docker**.
- Dans le cas où Brew ne fonctionne pas sur votre session, au lieu de perdre du temps à vous plaindre auprès de la pédago, demandez à votre voisin de droite ou de gauche de copier son répertoire **-/.brew** et de vous le transmettre en totalité.
- Il est possible d'installer Docker en utilisant des VM, mais dans ce cas-ci Docker perd tout son intérêt.
- Montrez à votre correcteur qu'un **docker** fonctionne.

Exemple :

```
$> docker info
Containers: 0
Images: 0
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 0
Execution Driver: native-0.2
Kernel Version: 3.16.0-30-generic
Operating System: Ubuntu 14.10
CPUs: 2
Total Memory: 3.633 GiB
Name: ubuntu
ID: PDDP:L64U:RQBT:G5ET:TEAC:XIJK:7JMR:ZVJW:NWU3:JMMD:YV36:WRUM
WARNING: No swap limit support
$>
```

II.3 Mission 1 - Hello World

- Rendre vos fichiers pour soutenance dans mission01 : le script du lancement du container (non obligatoire).
- Afficher "**Hello World**" dans votre navigateur à l'url suivant

```
IPContainer:xxxx ou localhost:xxxx #(si vous trouvez comment faire)
```

où [xxxx] est le port fourni par Docker pour accéder au container hébergeant le serveur web.

- À la soutenance, le correcteur lancera le container `hello/world:v42 ([REPOSITORY]:[TAG])` en mode interactif. Le prompt du container devra afficher

```
login@42born2code:/#
```

Attention : Ne pas utilisez d'image pré-construite qui fait le Hello World tout en un. Il faut bien entendu utiliser une image officielles Linux, vous avez le choix de la distribution.

II.4 Mission 2 - SSH

Rendre vos fichiers pour la soutenance dans le dossier `mission02`. Vous devez rendre le `Dockerfile` et les fichers ajoutés à l'exécution du build de l'image (autant que besoin pour l'accomplissement de la mission).

Au build du container, `OpenSSH` sera configuré automatiquement sur le port 42 (interne au container). Votre clé SSH `PUBLIQUE` (ne donner jamais votre clé `PRIVEE`) sera préconfigurée dans le répertoire **home/login/.ssh/authorized_keys** du container pour accepter les connexions provenant de votre clé privée ; ainsi vous pourrez vous connecter au container.

À la soutenance, votre correcteur devra build votre `Dockerfile` et lancer votre container en mode `daemon` et se connecter directemenr dans le container avec la commande suivante :

```
ssh login@IPContainer -p [yyyy] #ou
ssh login@localhost -p [yyyy] #(si vous trouvez comment faire)
```

où [yyyy] est le port fourni par Docker pour accéder au container (docker ps).

`Pro tip: dans la vrais vie, un container sécurisé ne permet pas de connexion SSH.`

II.5 Mission 3 - Volumes on MacOS X

Rendre vos fichiers pour soutenance dans `mission03`.

Trouver un moyen de partager un répertoire de la machine physique avec un répertoire à l'intérieur du container (montage `Volume`). On est au courant, c'est moche, mais vous aurez juste à montrer lors de la soutenance qu'un

```
touch filetest
```

dans le répertoire partagé est bien synchronisé avec le répertoire interne du container.

Attention: Ici on ne vous demande pas un container/volume "-volume" comme il est possible de faire de base. Vous devrez donc utiliser l'option "-v répertoirePhysique:répertoireContainer" au lancement du container.

Chapitre III

Bonus

Pour le rendu des bonus d'environnement complet :

- Chaque bonus sera nommé `bonus01`, `bonus02` .... `bonus[xx]`.
- Dans chacun de ces répertoires il y aura un `Dockerfile` et les fichiers joints lors de l'exécution du build (une création d'environnement ne devrait pas dépasser les 500Ko par bonus, `Dockerfile` compris).
- Pourquoi ne pas générer des `Dockerfiles` pour des environnements multiples ? Par exemple :
  - environnement de développement pour `RubyOnRails`, servi par `Nginx` et `PostgresSQL` ;
  - environnement `Django`, avec du `NoSQL` par exemple ;
  - environnement `PHP` ;
  - environnement `Java` prêt à être déployé.

  Ce sont des idées, vous devrez convaincre votre correcteur que votre `Dockerfile` peut faire gagner des demi-journées d'installation voir des semaines pour la mise en production d'environnements `Linux`.
- D'autres exemples :
  - Externaliser (monter en `Volume)` les logs `Linux /var/log` (comme cela votre image `Docker ne grossira pas)` ;
  - Externaliser une base de données (pour la persistance des données par exemple) ;
  - Lancer un container à l'intérieur d'un container (Légende urbaine) ;
  - Déplacer la `VM Boot2docker` sur une Clé USB pour alléger votre session.
- Expliquer à votre correcteur comment vous y êtes arrivé lors de la soutenance ! :)

Chapitre IV

Conclusion

Si vous terminez ce projet avec le dernier bonus proposé, vous aurez peut-être remarqué que vous disposez à présent d'un environnement Linux en ROOT avec dossier partagé et un processus qui consomme 1% de votre CPU. Pour la mémoire, c'est la même idée, tout vos problèmes de VM sont réglés. Docker n'est pas une VM. Si ça marche en local dans un container, ce même container marchera sur le serveur de production à distance. Il y a une multitude d'options à découvrir sur www.docker.com, ne vous arrêtez pas à ce sujet. Docker sous Linux n'a pas de sur-couche Boot2docker, vous aurez un accès direct à Docker. Voir aussi LXC d'où découle l'idée de Docker.

Quelques docs : Documentation officielle

Bon jeu!!!