TP2-Conteneurisation-2eme partie
Master IAO
Dept. Informatique

1ére partie : Installation de docker-machine :

Cluster : un ou plusieurs manager et worker qui executent docker.

Le but du manager est la gestion du cluster, dispatcher les services sur des conteneurs et peut lui-même contenir des conteneurs. Docker-machine permet de créer des machines virtuelles qui executent docker.

- On execute la commande suivante dans le prompt de linux :

```
$ base=https://github.com/docker/machine/releases/download/v0.16.2 &&
  curl -L $base/docker-machine-$(uname -s)-$(uname -m) >/tmp/docker-machine &&
  sudo mv /tmp/docker-machine /usr/local/bin/docker-machine &&
  chmod +x /usr/local/bin/docker-machine
```

- verifier l'installation de docker

```
$ ./ docker-machine version
```

- après l'installation de docker-machine on va créer une machine virtuelle, pour cela on aura besoin d'un driver et du nom de conteneur qui va héberger la machine.

```
$ ./docker-machine   create  -d      virtualbox      Master.Module.Cloud size memory
```

//----------//

NB: au cas où le driver de virtualbox n'est pas installé sur votre machine, on va retourner une erreur VBoxManage n'est pas installé. Tout d'abord on verifie que VirtualBox est installé par la commande :

```
$ whereis virtualbox
```

-Au cas ou le systeme ne retourne rien, on doit installer le driver de virtualbox :

```
$ sudo apt-get install virtualbox
```

-Apres l'installation on verifie les repertoires ou se trouve virtualbox

```
$ whereis virtualbox
```

//----------//

La machine virtuelle crée sera doté d'une :

-copie local de l'image iso de boot2docker

-adresse ip

-clé SSH (protocole de connexion sécurisé)

-copie des certificats au répertoire de la machine local

Lors de la création de la machine virtuelle, plusieurs variables ont été paramétrés, la commande qui va permettre de récupérer les variables d'environnements de la nouvelle VM à exporter

$ ./docker-machine env Master.Module.Cloud

Pour executer le moteur de la machine virtuelle dans notre shell courant :

$ eval $(./docker-machine env Master.Module.Cloud)

En exécutant cette commande sur votre shell courant, alors n'importe quelle commande Docker que vous exécuterez, sera dorénavant directement prise en compte par votre hôte Docker Master.Module.Cloud et non plus par votre hôte maître, et on peut verifier ceci par la commande : $ ./ Docker-machine active qui va nous afficher la machine vituelle, ou la commande : $ ./ docker-machine ls.

D'apres cette commande on remarque que dans la collonne active on a le symbole *.

Pour verifier ceci on peut afficher n'importe quel variable d'environnements par exemple la variable DOCKER_HOST qui nous afficher l'addresse IP et le port pour se connecter à la machine virtuelle :

$ echo $DOCKER_HOST

Par ailleurs si vous souhaitez vérifier sur quelle hôte Docker se lanceront vos prochaines commandes docker alors soit vous vérifiez si une étoile existe dans la colonne ACTIVE de la commande docker-machine ls. Soit plus simple encore, vous lancez la commande suivante

Afin de lister toutes les machines virtuelles installés, on utilise l'option ls qui nous affiche l'etat de la machine (Active par * et inactive par -) et son etat d'execution (running), aussi la version du moteur de docker et enfin la colonne ERR en cas d'erreur:

$ ./docker-machine ls

Le résultat nous indique distinctement, que nos futurs commandes docker sur le shell courant s'exécuteront directement sur la machine Docker Master.Module.Cloud, ainsi la machine Master.Module

A titre d'exemple on telecharge l'image de httpd :

$ docker run -d -p 8000:80 --name Master.Module.Cloud-httpd htppd

Cependant,ouvrez un nouveau terminal et on exectue la commande $ docker ps

Qu'est ce qu on remarque ??

Qu'on retourne dans notre terminal ou la machine virtuelle est active et on execute la commande on retrouve le conteneur de l'image

$ docker ps

On Remarque qu'on a un container avec le nom Master.Module.Cloud-httpd, dans mon shell de la machine nom Master.Module.Cloud activé.

Une autre manière de vérifier que l'on est sur la machine active, on se connecte par le protocole ssh :

$ ./docker-machine ssh Master.Module.Cloud

Une fois connecte on execute la commande

$ docker ps

Qu'est ce qu'on remarque => on retrouve le meme resultat que celui de la commande en dehors de la machine virtuelle.

Pour vous assurer que le client Docker est automatiquement configuré au début de chaque session de shell**, vous pouvez alors intégrer la commande eval $(docker-machine env vbox-test) votre** fichier ***~/.bash_profile*.**

Quand on veut arreter une machine docker on utilise la commande :

$ ./machine-docker stop

Et si on veut la redemarrer

$ ./machine-docker start

On peut surcharger les ressources allouées automatiquement par défaut à hôte Docker en utilisant les options venant avec le driver virtualbox. Dans cet exemple je vais créer une machine Docker avec 30 Go d'espace disque (20 Go par défaut) et avec 2 Go de ram (1Go par défaut) et

$ ./docker-machine create -d virtualbox \

--virtualbox-disk-size "30000" \

--virtualbox-memory "4000" \

Mobile_Coud_Test

2éme partie :Docker-Swarm

On crée tout d'abord deux machines virtuelles directeur et ouvrier.

$ ./docker-machine create --driver virtualbox directeur

$ ./docker-machine create --driver virtualbox ouvrier

Les deux machines virtuelles crees forment les nœuds de swarm. la première machine virtuelle **directeur** en tant que manager Swarm et **employe** l'autre machine sera un nœud de travail.

On verifie l'installation par

$ ./docker-machine ls

On remarque que chaque machine virtuelle à sa propre addresse ip. Ensuite on active le mode Swarm sur la **machine Docker directeur** de la même manière elle deviendra la leader de notre **Swarm**. Pour ce faire, nous utiliserons la commande **docker-machine ssh**.

La commande qui permet d'activer le mode Swarm sur une machine d'un Swarm

$ ./docker-machine ssh directeur "docker swarm init --advertise-addr 192.168.99.103".

Swarm initialized: current node (seyml9rjc3bo5eozlijjx7jgr) is now a manager.

docker swarm join --token SWMTKN-1-1368nlbw9syzrliv44956cvp9b1g5ivmr2rmu5g238g7q1bro7-1rutjgzngabbv2jx7dlrjnqlr 192.168.99.103:2377

Pour joindre une machine au cluster swarm on execute la commande suivante :

$ ./docker-machine ssh ouvrier "docker swarm join \

--token SWMTKN-1-1368nlbw9syzrliv44956cvp9b1g5ivmr2rmu5g238g7q1bro7-1rutjgzngabbv2jx7dlrjnqlr 192.168.99.103:2377

Pour afficher les differents nœuds de swarm :

$ ./docker-machine ssh directeur "docker node ls"

On remarque bien que le nœud directeur a la propriété de leader.

Nous utilisions des commandes Docker en ssh afin de les exécuter sur le nœud manager. Ce n'est pas vraiment pratique comme solution, en vue de nous faciliter la vie, désormais nous chargerons les variables d'environnements de notre leader Swarm sur notre shell courant , dès lors toutes les prochaines commandes docker lancées depuis le shell courant s'exécuteront directement sur la machine directeur.

On récupére les variables d'environnements de notre nœud **directeur.**

$ ./docker-machine env directeur

On configurer le shell pour lui permettre de communiquer avec le **nœud directeur** :

$ ./eval $(docker-machine env directeur)

Verifier que le nœud directeur est actif.

$ ./ docker-machine active

//----------//

Annexe : Création de la machine docker directeur avec un IP fixe

On creer une machine virtuelle avec 1024 de Ram et 10000 de taille de disque

$: ./docker-machine **create** directeur --driver virtualbox --virtualbox-memory 1024 --virtualbox-disk-size 10000

On demarre la machine directeur :

$ echo "ifconfig eth1 192.168.99.150 netmask 255.255.255.0 broadcast 192.168.99.255 up" | ./docker-machine ssh directeur tee /**var**/lib/boot2docker/bootsync.sh > /dev/null

J'ai assigné l'IP 192.168.99.150 à cette machine virtuelle et pour activer l'adresse on arrete , demarre puis on execute la commande suivante :

$ ./docker-machine regenerate-certs directeur

Pour regenerer les certificats, docker reclamer is vous avez executer la commande $docker-machine env directeur

Arreter et demarrez la machine virtuelle une fois pour toute et executze
$ docker-machine ls,
on verra la nouvelle machine listé avec son IP correct, et aucune erreur ne figurera dans la derniere colonne.