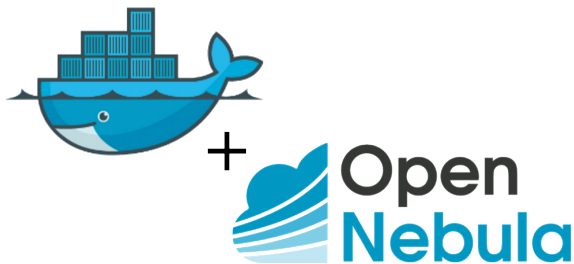


DOCKER-1 Docker Machine OpenNebula Driver



Continguts

- 1 Instal·lar docker-machine
- 2 Instal·lar el driver per a OpenNebula de docker-machine
 - 2.1 Preparació Sistema
 - 2.2 Instal·lació del driver
- 3 Instal·lar docker-engine
- 4 Com utilitzar el driver d'OpenNebula amb docker-machine:
 - 4.1 Crear el nostre Docker Engine
 - 4.2 Enllaçar la shell local amb la de la maquina virtual
- 5 Exemple d'ús

Per a poder crear màquines Docker Engine a l'OpenNebula podem fer servir el driver de docker-machine, un cop instal·lat podrem crear i instanciar els nostres contenidors de Docker.

Requeriments:

- **Docker Engine:** Ens permet crear els contenidors, gestionar-los i instanciar-los
- **Docker Machine:** Ens permet crear i gestionar les màquines virtuals
- **Driver Docker Machine Opennebula:** Ens permet utilitzar el docker-machine al cloud de l'OpenNebula

Instal·lar docker-machine

Com a usuari **privilegiat**, ens descarreguem el binari executem:

```
curl -L https://github.com/docker/machine/releases/download/v0.6.0/docker-machine-`uname -s`-`uname -m`  
> /usr/local/bin/docker-machine && \  
chmod +x /usr/local/bin/docker-machine
```

Un cop instal·lat, per comprovar que podeu executar docker-machine, executeu des del vostre usuari:

```
docker-machine version
```

Si us diguéis que no troba el binari, executeu:

```
export PATH=$PATH:/usr/local/bin
```

Podeu trobar més informació a: <https://docs.docker.com/machine/install-machine/>

Instal·lar el driver per a OpenNebula de docker-machine

Abans d'instal·lar el driver, hem de preparar l'equip per a l'instal·lació.

Preparació Sistema

Paquet GO

```
wget https://storage.googleapis.com/golang/go1.6.linux-amd64.tar.gz
tar -C /usr/local -xvzf go1.6.linux-amd64.tar.gz
export GOROOT=/usr/local/go
export PATH=$PATH:$GOROOT/bin
```

En creem el directori work en el nostre espai de treball, per exemple al \$HOME:

```
mkdir $HOME/work
```

Indiquem les variables d'entorn necessàries:

```
export GOPATH=$HOME/work
export PATH=$PATH:$GOPATH/bin
```

Paquet GIT, BZR

```
sudo apt-get install git bzr
```

Paquet GODEP

```
go get github.com/tools/godep
```

Instal·lació del driver

```
go get github.com/OpenNebula/docker-machine-opennebula
cd $GOPATH/src/github.com/OpenNebula/docker-machine-opennebula
make build
make install
```

ONE_AUTH i ONE_XMLRPC

Ens creem un fitxer on indicarem el nostre usuari i password de l'OpenNebula

```
mkdir -p $HOME/.one
echo usuari:password > $HOME/.one/one_auth
```

Creem les variables d'entorn necessàries:

```
export ONE_AUTH=$HOME/.one/one_auth
export ONE_XMLRPC=http://iaas.csuc.cat:2633/RPC2
```

Podeu trobar més informació a: <https://github.com/OpenNebula/docker-machine-opennebula>

Instal·lar docker-engine

<https://docs.docker.com/engine/installation/linux/>

Com utilitzar el driver d'OpenNebula amb docker-machine:

Crear el nostre Docker Engine

Existeix una imatge a l'OpenNebula d'un sistema operatiu on ja està instal·lat el docker i on podem llançar contenidors, es diu *boot2docker*. També existeix una versió d'Ubuntu.

Si no està disponible, des del Marketplace de l'OpenNebula en la descarreguem.

Un cop tenim descarregada la imatge, creem la màquina virtual:

```
docker-machine create --driver opennebula --opennebula-image-id [num_imatge_boot2docker] --opennebula-network-id [num_xarxa] --opennebula-b2d-size [mida_en_MB] [nom_màquina_virtual]
```

Més opcions:

- `--opennebula-cpu`: CPU value for the VM
- `--opennebula-vcpu`: VCPUs for the VM
- `--opennebula-memory`: Size of memory for VM in MB
- `--opennebula-template-id`: Template ID to use
- `--opennebula-template-name`: Template to use
- `--opennebula-network-id`: Network ID to connect the machine to
- `--opennebula-network-name`: Network to connect the machine to
- `--opennebula-network-owner`: User ID of the Network to connect the machine to
- `--opennebula-image-id`: Image ID to use as the OS
- `--opennebula-image-name`: Image to use as the OS
- `--opennebula-image-owner`: Owner of the image to use as the OS
- `--opennebula-dev-prefix`: Dev prefix to use for the images: 'vd', 'sd', 'hd', etc...

- `--opennebula-disk-resize`: Size of disk for VM in MB
- `--opennebula-b2d-size`: Size of the Volatile disk in MB (only for b2d)
- `--opennebula-ssh-user`: Set the name of the SSH user
- `--opennebula-disable-vnc`: VNC is enabled by default. Disable it with this flag

Enllaçar la shell local amb la de la màquina virtual

Ara el que farem serà connectar la nostra shell local a la de la màquina virtual perquè cada cop que executem docker, aquest dongui ordres al docker instal·lat a la màquina virtual.

Obtenim les variables d'entorn de la màquina virtual.

```
docker-machine env [nom_màquina_virtual]
```

Linkem la shell.

```
eval "$(docker-machine env [nom_màquina_virtual])"
```

Ara ja podem utilitzar docker normalment.

Variables d'entorn

Per mantenir les noves variables d'entorn de manera persistent, afegiu-les al vostre `.profile` (`$HOME/.profile`)

Exemple d'ús

Actualment disposeu de dues imatges, ja precreades amb el Docker Engine instal·lat que podeu fer servir:

- `boot2docker`
- `Docker-Machine-Ubuntu-14.04`

Crear un contenidor amb nginx en una màquina Docker amb el driver de Docker Machine al OpenNebula:

- **Crear una màquina al OpenNebula amb el Docker Engine i un disc volàtil de 10 GB amb la comanda:**

```
docker-machine create --driver opennebula --opennebula-network-id $NETWORK_ID --opennebula-image-name boot2docker --opennebula-image-owner oneadmin --opennebula-b2d-size 10240 mydockerengine
```

❗ On \$NETWORK_ID, serà l'ID de la xarxa on crearem la màquina amb Docker Engine.

- Podem comprovar que tot a funcionat bé amb la comanda:

```
docker-machine ls
```

```
MacBookAir:~ miguelangel$ docker-machine ls
NAME            ACTIVE  URL                STATE    URL                SWARM    DOCKER    ERRORS
default         -       virtualbox         Stopped
mydockerengine  *       opennebula         Running  tcp://192.168.1.100:2376
MacBookAir:~ miguelangel$ docker-machine env mydockerengine
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.1.100:2376"
export DOCKER_CERT_PATH="/Users/miguelangel/.docker/machine/machines/mydockerengine"
export DOCKER_MACHINE_NAME="mydockerengine"
# Run this command to configure your shell:
# eval $(docker-machine env mydockerengine)
MacBookAir:~ miguelangel$ eval $(docker-machine env mydockerengine)
MacBookAir:~ miguelangel$
```

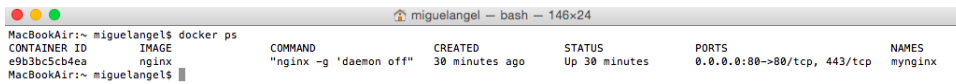
- Les següents comandes ens mostra les variables d'entorn necessaries per accedir a la shell:

```
docker-machine env mydockerengine
eval $(docker-machine env mydockerengine)
```

- Ara podem començar a fer servir docker sobre la màquina Docker Engine que acabem de crear:

```
docker pull nginx
docker run --name mynginx -d -p 80:80 nginx
```

- **Finalment, comprovem que podem accedir al servidor web que acabem de crear:**



A terminal window titled 'miguelangel — bash — 146x24' showing the command 'docker ps' and its output. The output lists the 'mynginx' container with details on its image, command, creation time, status, ports, and name.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e9b3bc5cb4ea	nginx	'nginx -g 'daemon off''	30 minutes ago	Up 30 minutes	0.0.0.0:80->80/tcp, 443/tcp	mynginx

