Résumé Journalier

Joffrey Hérard

7 avril 2017

1 Docker MEMO

Installation de docker

```
sudo apt-get update
sudo apt-get install curl \
    linux-image-extra-$(uname -r) \
    linux-image-extra-virtual

sudo apt-get install apt-transport-https \
               software-properties-common \
               ca-certificates
curl -fsSL https://yum.dockerproject.org/gpg | sudo apt-key add -
sudo apt-get install software-properties-common
sudo add-apt-repository \
       "deb https://apt.dockerproject.org/repo/ \
       ubuntu-$(lsb_release -cs) \
       main"
sudo apt-get -y install docker-engine
```

Commandes de base

```
docker ps #lister tous les conteneurs en route
docker ps -a #liste tous les conteneurs, mêmes terminés
docker run docker/whalesay cowsay booo #lancer le conteneur walesay avec lancer la commande cowsay e
docker images #lister les conteneurs
docker stop <conteneurs>
docker rm <conteneurs>
docker stop $(docker ps -a -q) #stopper tous les conteneurs
docker inspect -format='{{json .NetworkSettings.Networks}}'  <nom_conteneur>
```

Création de conteneurs perso

```
mkdir mydockerbuilld
nano Dockerfile #Création du dockerfile

#Début du fichier
FROM docker/whalesay:latest #Définition du conteneur sur lequel se baser
RUN apt-get -y update && apt-get install -y fortunes #Commande à faire lors de l'installation
CMD /usr/games/fortune -a | cowsay #Commande à faire lors du lancement
#Fin du fichier

docker build -t docker-whale . #build du conteneur
docker tag 7d9495d03763 maryatdocker/docker-whale:latest #tag pour ajout sur dépôt docker
docker login #login sur docker.com
docker push maryatdocker/docker-whale #push du conteneur
docker rmi -f 7d9495d03763 #suppression d'un conteneur
```

```
docker rmi -f docker-whale #<=====' ou bien

docker run yourusername/docker-whale #lancement du conteneur
```

Fonctionnalités avec le réseau

```
docker network ls #lister les différents réseaux docker
docker run -itd --name=networktest ubuntu #lancer ubuntu avec un réseau bridge = concentrateur d'inter
docker network inspect bridge #inspecter le réseau bridge
docker network disconnect bridge networktest #déconnecter networktest du  réseau bridge
#existe en connecte

docker network create -d bridge my-bridge-network #créer un nouveau réseau nommémy-bridge-network en m
docker run -d --net=my-bridge-network --name db training/postgres #lancer le conteneur directement dan
docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' web
```

2 LXC MEMO

2.1 Réseaux privé

Script UP Script pour up un réseaux prive dans LXC.

```
#!/bin/bash


ip link add name br type bridge

ip link set dev br up

lxc-start -n ct1 -d

lxc-start -n ct2 -d
```

Script DOWN Script pour down un réseaux prive dans LXC .

```
#!/bin/bash

lxc-stop -n ct2

lxc-stop -n ct1

ip link delete dev br
```

CONFIG Fichier de config du conteneur pour un réseaux prive dans LXC.

```
lxc.network.type = veth
lxc.network.flags= up
```

```
lxc.network.link= br
lxc.network.veth.pair= br-ct1
lxc.network.ipv4= 192.168.1.101/24

lxc.rootfs = /usr/share/lxc/config/debian.common.conf


lxc.mount = /var/lib/lxc/ct1/fstab
lxc.utname =ct1
lxc.arch= amd64
lxc.autodev = 1
lxc.ksmg =0
```

2.2 Full bridge

Script UP

```
#!/bin/bash


piddhcp= $(pgrep -f dhcp)
kill -9 $piddhcp
ip link set dev eth0 down
ip addr flush eth0
ip link set dev eth0 up
ip link add name br type bridge
ip link set dev br up
ip link set dev eth0 master br
dhclient br
lxc-start -n ct1 -d
```

Script DOWN

```
#!/bin/bash

piddhcp= $(pgrep -f dhcp)

kill -9 $piddhcp
lxc-stop -n ct1

ip link set dev eth0 nomaster

ip link delete dev br

dhclient eth0
```

CONFIG

```
lxc.network.type = veth
lxc.network.flags= up
```

```
lxc.network.link= br
lxc.network.veth.pair= br-ct1


lxc.rootfs = /var/lib/lxc/ct1/root.fs
lxc.include = /usr/share/lxc/config/debian.common.conf


lxc.mount = /var/lib/lxc/ct1/fstab
lxc.utname =ct1
lxc.arch= amd64
lxc.autodev = 1
lxc.ksmg =0
```

2.3 OpenVSwitch

2.4 Bridge NAT

```
lxc.network.type = veth
lxc.network.flags= up
lxc.network.link= br
lxc.network.veth.pair= br-ct1


lxc.rootfs = /var/lib/lxc/ct1/root.fs
lxc.include = /usr/share/lxc/config/debian.common.conf


lxc.mount = /var/lib/lxc/ct1/fstab
lxc.utname =ct1
lxc.arch= amd64
lxc.autodev = 1
lxc.ksmg =0
```

```
#!/bin/bash

ip link add name br type bridge

ip addr add 192.168.10.1/24

ip link set dev br up

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
lxc-start -n ct2 -d
```

```
#!/bin/bash

lxc-stop -n ct2
```

```
iptables -F
ip link delete dev br
```

2.5 Ligne de commande utiles

```
lxc-create
lxc-start
lxc-info
lxc-ls
lxc-console
lxc-attach
lxc-stop
lxc-destroy


lxc-cgroup -n ctn01 cpuset.spus 0,1

lxc-cgroup -n ctn01 memory.soft_limit_in_bytes 268435456
lxc-cgroup -n ctn01 memory.limit_in_bytes 53687091

fichier de config =

lxc.cgroupe.cpuset.cpus=0,1
lxx.cgroup.memory.soft_limit_in_bytes=268435456
lxc.cgroup.memory.limit_in_bytes=53687091
```

3 KVM MEMO

```
sudo apt-get install qemu-kvm

--enable-kvm === dans qemu


egrep '(vmx|svm)' /proc/cpuinfo
```

4 QEMU MEMO

```
qemu-img         create
                convert
                info
                resize

qemu-img create -f qcow2 mydeb.img 10 G
qemu-img convert -f qcow2 mydeb.img -O raw mydeb.img2
qemu-img info -f qcow2 mydeb.img
qemu-img resize mydeb.img +5G
qemu-img create -f qcow2 -b mydeb.img img1.qcow2
```

```
qemu-system-x86_64
        k = definition du clavier -k fr
        m = quantite de ram -m 512
        hda image disque utilise -hda mydeb.img
        boot gestion du demmarage c, f, n -boot d
        cdrom image iso ou cdrom utilise -cdrom xxx.iso

carte graphgique :

        cirrus
        std
        vmware
        qx1

mode user

qemu-system-x86_64 -k fr -m 512 -hda myDeb.img &

qemu-system-x86_64 -k fr -m 512 -hda myDeb.img -net nic -net user &

qemu-system-x86_64 -k fr -m 512 -hda myDeb.img -netdev user, id=network0 -device e1000, netdev=ne

redirection

qemu-system-x86_64 -k fr -m 512 -redir tcp:5555::80 -hda myDeb.img &
```

5 VitualBox MEMO

6 Config du /etc/network/interfaces de la Proxmox initial

```
############################################################
# network interface settings
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

iface eth1 inet manual

iface eth2 inet manual

iface eth3 inet manual
```

```
auto vmbr0
iface vmbr0 inet static
        address  172.18.10.1
        netmask  255.255.255.0
        bridge_ports eth0.1020
        bridge_stp off
        bridge_fd 0

auto vmbr1
iface vmbr1 inet static
        address  10.22.9.17
        netmask  255.255.255.0
        gateway  10.22.9.254
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
########################################################
```

Références