UNIVERSITÉ DE RENNES 1

ESIR ECOLE SUPERIEURE D'INGENIEURS DE RENNES

23/01/2019

TD 5

> *« J'atteste que ce travail est original, qu'il indique de façon appropriée tous les emprunts, et qu'il fait référence de façon appropriée à chaque source utilisée »*

Léo Guilpain – ESIR3 - IoT

Table des matières

Introduction

Le but de ce TP est de faire en sorte que notre application passe l'échelle. Nous allons étudier la solution « scale-out »

Question 01

Scale - Out

La solution du « scale-out » consiste à augmenter le nombre de machine afin de répartir la charge correctement. En augmentant le nombre de machine cela nous permet de garder en fonctionnement notre système même si une des machines ne fonctionne plus.

De plus, en termes de flexibilité (possibilité d'ajout de nouvelles machines en cas d'augmentation de la demande) et d'administration (diminution du temps de configuration, de migration, etc), cela représente un avantage.

Je me suis servi de ce site : https://www.lemagit.fr/conseil/Quels-sont-les-avantages-du-stockage-NAS-Scale-out

Scale – up

La méthode scale-up consiste à posséder un serveur très puissant qui est capable de gérer toutes les modifications et toutes les nouvelles configurations. Cette méthode permet de remplacer un serveur par un autre tout en ne prenant pas plus de place.

De plus cette solution est moins chère que le scale-out. Enfin, comme dit plus haut pour le scale-out il faut moins de machines, cela implique donc une consommation inférieure.

Je me suis servi de ces sites :

- https://www.lemagit.fr/conseil/Scale-up-ou-scale-out-le-meilleur-choix-pour-votre-datacenter
- https://www.oreilly.com/library/view/cloud-architecture-patterns/9781449357979/ch01.html

Partie 1 : Déployer des VMs avec Vagrant

Question 01 & 02

```
leo@leo:~/ESIR3/Cloud/TP5/td5-LeoGui/td5/Vagrant$ vagrant init ubuntu/xenial64
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

Figure 1 : vagrant init

Le fichier “VagrantFile” a bien été créé. Dans le répertoire « Vagrant ».

Question 03

Pour lancer sa machine virtuelle, il suffit d'exécuter la commande : « vagrant up ». Pour vérifier, il suffit de taper « virtualbox » pour pouvoir voir les machines virtuelles en fonctionnement. Comme on peut le voir ci-dessous, notre machine virtuelle a bien été lancée.

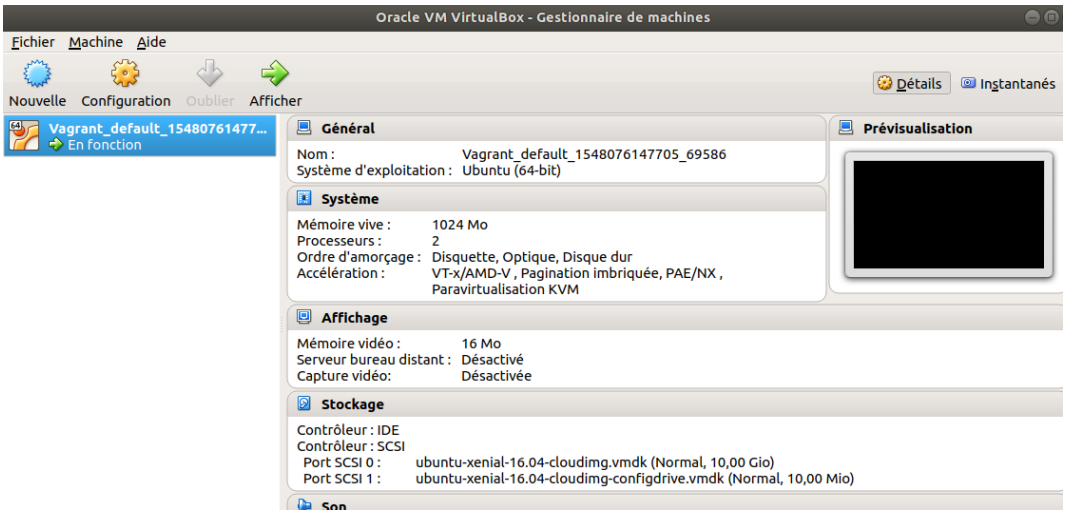

Figure 2 : virtualbox

On se connecte ensuite à la machine virtuelle via la commande “vagrant ssh”

```
leo@leo:~/ESIR3/Cloud/TP5/td5-LeoGui/td5/Vagrant$ vagrant ssh
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-141-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


vagrant@ubuntu-xenial:~$
```

Figure 3 : vagrant ssh

Question 04

Par défaut, tous les fichiers qui se trouve dans le dossier du VagrantFile sont partagés. On vérifie ceci grâce à la commande “ls /vagrant”.

```
vagrant@ubuntu-xenial:~$ ls /vagrant
ubuntu-xenial-16.04-cloudimg-console.log  Vagrantfile
```

Figure 4 : ls / vagrant

Question 05

Plusieurs solutions s’offrent à nous quant à l’arrêt de la machine virtuelle préalablement lancée :

- **Suspendre** : “vagrant suspend”
- **Arrêter** : “ vagrant halt”
- **Détruire** : “vagrant destroy”

```
leo@leo:~/ESIR3/Cloud/TP5/td5-LeoGui/td5/Vagrant$ vagrant halt
==> default: Attempting graceful shutdown of VM...
leo@leo:~/ESIR3/Cloud/TP5/td5-LeoGui/td5/Vagrant$ vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Destroying VM and associated drives...
```

Figure 5 : Test des commandes d'arrêt

Question 06

```
Vagrant.configure(2) do |config|
        # Define box `m0'
        config.vm.define "m0" do |m0|
                m0.vm.hostname = "m0"
                m0.vm.box = "ubuntu/xenial64"
        end

        # Define boxes `m1' and `m2' with a for-loop
        (1..2).each do |i|
                config.vm.define "m#{i}" do |machine|
                        machine.vm.hostname = "m#{i}"
                        machine.vm.box = "ubuntu/xenial64"
                end
        end
end
```

Figure 6 : Vagrantfile

Ici dans ce Vagrantfile, on définit 3 box. La première portera le nom de « m0 ». Tandis que les deux autres seront nommées « m1 » et « m2 ». Les machines virtuelles tourneront sous Ubuntu.

Question 07

On utilise un réseau privé. Il faut donc ajouter la commande suivante à notre Vagrantfile :

« config.vm.network "private_network", ip: "192.168.50.4" »

Partie 2 : Ansible ou l'automatisation de tâches distantes

Question 01 et 02

Comme indiqué dans le TP, on créé un fichier « playbook.yml » que l'on remplit comme ceci :

```
- hosts: all
# *Become* root
  become: true
# Speeds up the script
  gather_facts: false

  tasks:
  # Include the Docker installation Ansible play file (empty for now)
  # - include: install_docker.yml

    - name: Print some dummy thing
      shell: echo "It works!"
```

Figure 7 : "playbook.yml"

Ensuite, il suffit de modifier le Vagrantfile comme ci-dessous :

```
Vagrant.require_version ">= 1.7.0"


Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/xenial64"

  config.ssh.insert_key = false

  config.vm.provision "ansible" do |ansible|
   ansible.verbose = "v"
   ansible.playbook = "playbook.yml"
  end
```

Figure 8 : Vagrantfile

Une fois que tout ceci a été mis en place, il faut lancer la machine virtuelle. On exécute donc un « vagrant up ». Comme indiqué, nous obtenons l'erreur python.

```
fatal: [default]: FAILED! => {"changed": false, "module_stderr": "Shared connect
ion to 127.0.0.1 closed.\r\n", "module_stdout": "/bin/sh: 1: /usr/bin/python: no
t found\r\n", "msg": "MODULE FAILURE", "rc": 127}
```

Figure 9 : Erreur python

On rajoute donc la ligne suivante dans le Vagrantfile pour forcer vagrant à utiliser python3 :

```
Vagrant.configure("2") do |config|
  # The most common configuration options are docum
  # For a complete reference, please see the online
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/xenial64"

  config.ssh.insert_key = false

  config.vm.provision "ansible" do |ansible|
   ansible.verbose = "v"
   ansible.playbook = "playbook.yml"
   ansible.extra_vars = {
  ansible_python_interpreter: "/usr/bin/python3"
}
  end
```

Figure 10 : Vagrantfile python3

```
changed: [default] => {"changed": true, "cmd": "echo \"It works!\"", "delta": "0:00:00.001637", "end": "2019-01-
21 22:16:12.902474", "rc": 0, "start": "2019-01-21 22:16:12.900837", "stderr": "", "stderr_lines": [], "stdout":
 "It works!", "stdout_lines": ["It works!"]}

PLAY RECAP *********************************************************************
default                    : ok=1    changed=1    unreachable=0    failed=0
```

Figure 11 : vagrant up

On voit bien en sortie la valeur "It works"comme indiqué dans le « playbook.yml » donc cela a bien fonctionné.

Question 03

On veut maintenant installer docker sur la machine virtuelle. On va donc créer le fichier « install_docker.yml ». Pour ce faire nous allons suivre ce qui a été fait sur ce lien : https://andrewaadland.me/2018/10/14/installing-newer-versions-of-docker-in-ubuntu-18-04/?fbclid=IwAR3HTCmEHPAhf9r9CtmFzRPvN2vJjpnxf0DCT3NnwwcBa3qyzqwGMnEc9ag

```yaml
- name: "APT - Add Docker GPG key"
  apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
    state: present

- name: "APT - Add Docker repository"
  apt_repository:
    repo: "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
    state: present
    filename: docker

- name: "APT - install misc packages"
  apt:
    name: "{{ item }}"
    update_cache: yes
  with_items:
    - "aptitude"
    - "apt-transport-https"
    - "ca-certificates"
    - "curl"
    - "software-properties-common"

- name: "APT - install 'docker-ce'"
  apt:
    name: "docker-ce"
    update_cache: yes

```

Figure 12 : "install_docker.yml"

Puis on l'inclut dans le fichier « playbook.yml ».

```yaml
- hosts: all
# *Become* root
  become: true
# Speeds up the script
  gather_facts: false

  tasks:
  # Include the Docker installation Ansible play file (empty for now)
    - include: install_docker.yml

    - name: Print some dummy thing
      shell: echo "It works!"

```

Figure 13 : Include docker in playbook

Pour vérifier la bonne installation de docker sur la machine virtuelle, il suffit de se connecter en ssh à cette dernière et d'effectuer la commande "docker -v". Si la version de docker ne s'affiche pas, alors docker n'est pas installer sur la machine.

Comme on peut le voir, docker a bien été installé sur la machine virtuelle. Son numéro de version est : 18.09.1

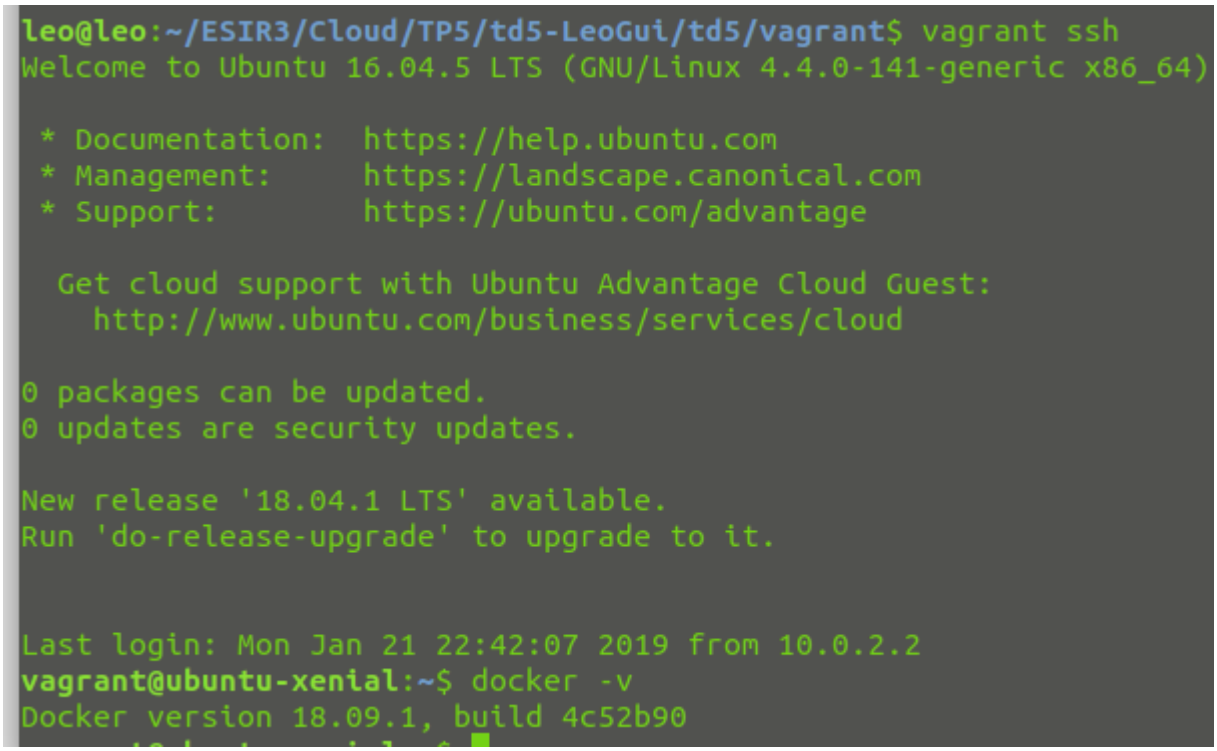

Figure 14 : docker version

Question 04

On remplit le vagrant file comme ceci. Ensuite, on ajoute 3 nouveaux fichiers : “worker1_playbook.yml”, “worker2_playbook.yml” et “manager_playbook.yml”.

```
Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/xenial64"

  config.ssh.insert_key = false

# Define box `worker1'
  config.vm.define "worker1" do |worker1|
    worker1.vm.hostname = "worker1"
    worker1.vm.box = "ubuntu/xenial64"
    worker1.vm.provision "ansible" do |ansible|
      ansible.verbose = "v"
      ansible.playbook = "worker1_playbook.yml"
      ansible.extra_vars = {
        ansible_python_interpreter: "/usr/bin/python3",
        manager_ip: "192.168.42.17"
      }
    end
    worker1.vm.network :private_network,ip: "192.168.42.15"
  end

 # Define box `worker2'
 config.vm.define "worker2" do |worker2|
   worker2.vm.hostname = "worker2"
   worker2.vm.box = "ubuntu/xenial64"
   worker2.vm.provision "ansible" do |ansible|
     ansible.verbose = "v"
     ansible.playbook = "worker2_playbook.yml"
     ansible.extra_vars = {
       ansible_python_interpreter: "/usr/bin/python3",
       manager_ip: "192.168.42.17"
     }
   end
  worker2.vm.network :private_network,ip: "192.168.42.16"
  end

  # Define box `manager'
  config.vm.define "manager" do |manager|
    manager.vm.hostname = "manager"
    manager.vm.box = "ubuntu/xenial64"
    manager.vm.provision "ansible" do |ansible|
      ansible.verbose = "v"
      ansible.playbook = "manager_playbook.yml"
      ansible.extra_vars = {ansible_python_interpreter: "/usr/bin/python3"}
    end
    manager.vm.network :private_network,ip: "192.168.42.17"
  end
end
```

Figure 15 : Vagrantfile

Comme on peut le voir dans le Vagrantfile, la ligne « manager_ip » a été rajoutée. Cela permet de faire passer la variable qui correspond à l'adresse IP du manager. Les « workers » connaissent ainsi l'adresse IP du manager.

Les fichiers playbook sont configurés comme l'ancien fichier "playbook.yml". C'est à dire que l'on inclut "install_docker.yml".

On peut voir sur les screens ci-dessous que docker est bien installé sur les 3 machines virtuelles.

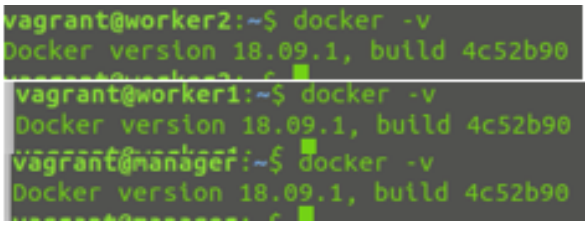

Figure 16 : docker -v

Lorsque l'on tape la commande "virtualbox", on peut voir que les machines sont correctement lancées.

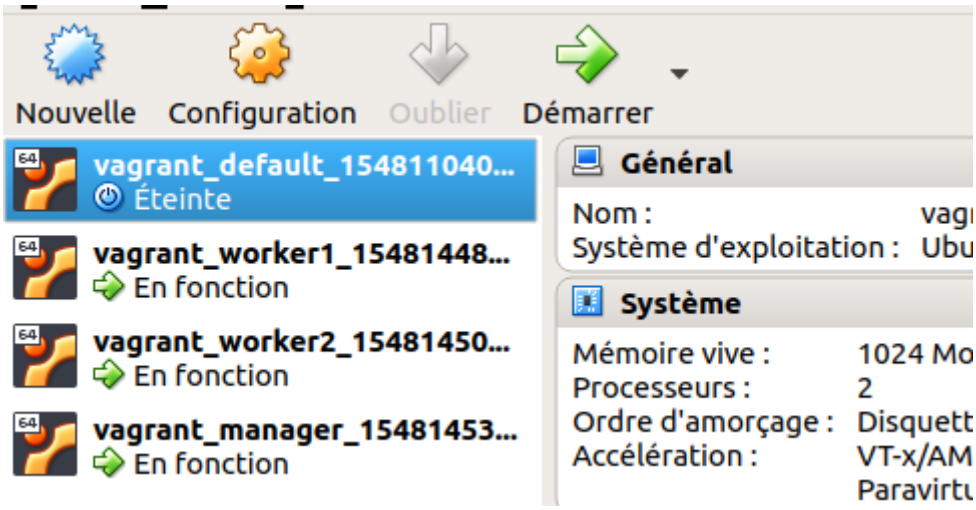

Figure 17 : VirtualBox

Conclusion

Dans ce TP sur la mise à l'échelle, nous avons pu en apprendre un peu plus sur le « scale out » notamment en utilisant Vagrant et Ansible. Nous avons pu créer plusieurs machines virtuelles possédant Docker. Malheureusement, des problèmes d'installations, notamment un passage de Ubuntu 17.10 à 18.04 pour le bon fonctionnement du TP m'ont empêché d'aller plus loin.