TD 5

Introduction

Au cours de ce TP nous allons reprendre notre application et la passer à l'échelle. Il existe deux possibilités : soit on loue un serveur avec plus de RAM et de CPU (Scale up) soit on loue plusieurs serveurs qui se répartiront la charge (Scale out).

1)
avantage scale up :
- flexibilité
- évolutivité
- augmenter la capacité
- augmenter la puissance

avantages scale out :
- intégralité du système peut être géré simultanément
- Possibilité d'augmenter les capacités à l'infini sans dégradation des performances
- les Noeuds n'ont pas besoin d'être situés au même endroit

Le stockage scale out devient la norme dans le domaine des systèmes de stockage externes. Toutefois, a long terme il est plus probable que la norme devienne les systèmes capables de s'adapter dans plusieurs dimensions.

Nous allons nous concentrer sur l'option « scale out » en utilisant 3 ordinateurs pour distribuer la charge. Nous utiliserons Vagrant pour lancer des Vms sur notre machine. La configuration de Vagrant est donné par le fichier *vagrantfile.* On utilisera ensuite Ansible pour configurer les Vms .

Partie I : Déployer des Vms avec Vagrant

1)
On commence par installer Vagrant en suivant ce tuto : https://www.vagrantup.com/intro/getting-started/index.html
On installe également VirtualBox.

2)
Dans un nouveau dossier « td5-orain/td5 » on éxécute la commande :

```
orain@orain-TM1701:~/Documents/td5-antho35/td5$ 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.
```

Le fichier de configuration vagrantfile se présente de la manière suivante :

Ouvrir | Vagrantfile ~/Documents/td5-antho35/td5 | Enregistrer

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
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"
```

Ce fichier utilise la synthaxe Ruby.

3)
On peut à présent lancer la machine virtuelle que nous avons initialisé.

```
orain@orain-TM1701:~/Documents/td5-antho35/td5$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'ubuntu/xenial64' could not be found. Attempting to find and in
stall...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'ubuntu/xenial64'
    default: URL: https://vagrantcloud.com/ubuntu/xenial64
==> default: Adding box 'ubuntu/xenial64' (v20190118.0.0) for provider: virtualb
ox
```

Pour s'y connecter on exécute :

```
orain@orain-TM1701:~/Documents/td5-antho35/td5$ 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.
```

4) Pour partager des fichiers entre l'hôte et la machine on modifie le fichier vagrantfile en utilisant la méthode « config.vm.synced_folder ». Le premier paramètre correspond au chemin vers le dossier de la machine hôte et le second correspond au répertoire d'accueil sur la VM.

5)
Pour arrêter notre machine, on sort du SSH et on entre :

```
vagrant@ubuntu-xenial:~$ exit
logout
Connection to 127.0.0.1 closed.
orain@orain-TM1701:~/Documents/td5-antho35/td5$ vagrant halt
==> default: Attempting graceful shutdown of VM...
orain@orain-TM1701:~/Documents/td5-antho35/td5$
```

On a arrété la machine virtuelle.
On peut ensuite la détruire :

```
orain@orain-TM1701:~/Documents/td5-antho35/td5$ vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Destroying VM and associated drives...
```

```
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
```

6)
Ce vagrantfile crée 3 box. Dans la deuxième partie de ce fichier, il y a une boucle pour créer deux machines avec des configurations semblables.

7)
Pour ajouter un sous-réseau virtuel ainsi que des Ips fixes pour nos Vms, il faut ajouter au vagrantfile les lignes suivantes :

```
Vagrant.configure("2") do |config|
  config.vm.network "private_network", ip: "192.168.50.4"
end
```

Puis on relance notre VM : avec « ***vagrant resume*** »

Partie II : Ansible ou l’automatisation de tâches distantes

Une fois l’installation des Vms faite, on va déployer notre galerie à l’aide de Ansible.

1)
On commence par installer Ansible.

```
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo apt-add-repository --yes --update ppa:ansible/ansible
$ sudo apt-get install ansible
```

2) On va commencer par installer Docker sur une unique machine démarrée avec Vagrant.

Tout d’abord on ajoute le code suivant à notre fichier vagrantfile pour utiliser Ansible en tant que provisionneur :

```
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"

  # Disable the new default behavior introduced in Vagrant 1.7, to
  # ensure that all Vagrant machines will use the same SSH key pair.
  # See https://github.com/hashicorp/vagrant/issues/5005
  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"
}
# Profitez-en pour mettre plus de "verbose" :
ansible.verbose = "vvv"
  end
```

On crée un nouveau fichier playbook.yml :

playbook.yml

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

  tasks:
    # Include the Docker installation A
    # - include: install_docker.yml

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

Puis on peut relancer notre VM avec « vagrant up » :

```
changed: [default] => {
    "changed": true,
    "cmd": "echo \"It works!\"",
    "delta": "0:00:00.001901",
    "end": "2019-01-21 16:48:45.164651",
    "invocation": {
        "module_args": {
            "_raw_params": "echo \"It works!\"",
            "_uses_shell": true,
            "argv": null,
            "chdir": null,
            "creates": null,
            "executable": null,
            "removes": null,
            "stdin": null,
            "warn": true
        }
    },
    "rc": 0,
    "start": "2019-01-21 16:48:45.162750",
    "stderr": "",
    "stderr_lines": [],
    "stdout": "It works!",
    "stdout_lines": [
        "It works!"
    ]
}
META: ran handlers
META: ran handlers

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

3) Dans le fichier playbook.yml, on décommente la ligne « - include : install_docker.yml ». On crée un nouveau fichier « install_docker.yml » qui se présente de lamanière suivante :

install_docker.yml Vagrantfile

```
- name: Update all packages to the latest version
  apt:
   upgrade: dist

- name: download key for docker
  shell: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

- name: fingerprint
  apt_key:
   id: 0EBFCD88

- name : add repo
  apt_repository:
   repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable

- name: install docker
  apt:
   name: docker-ce
   state: latest
   update_cache: yes
```

Puis on relance la VM.

4)
Il s’agit maintenant de créer 3 VM (manager, worker-1 et worker-2) sur le même sous-réseau en fixant leur Ips.
En reprenant l’exemple de la question 6) de la partie I on remplit le vagrantfile. Pour cela, on s’assure que les « nodes » soient bien dans le même réseau (ip_manager : 12.0.0.2, ip_worker-1 : 12.0.0.3 et ip_worker-2 : 12.0.0.4). On provisionne avec ansible.

Vagrantfile | manager_playbook.yml | worker_playbo

```
Vagrant.configure("2") do |config|
  # Define box `manager'
  config.vm.define "manager" do |manager|
    manager.vm.hostname = "manager"
    manager.vm.box = "ubuntu/xenial64"
    manager.vm.network "private_network", ip: "12.0.0.2"
    manager.vm.provision "ansible" do |ansible|
      ansible.verbose = "v"
      ansible.playbook = "manager_playbook.yml"
      ansible.extra_vars = {
      ansible_python_interpreter: "/usr/bin/python3",
      manager_ip: "12.0.0.2"
    }
    # Profitez-en pour mettre plus de "verbose" :
    ansible.verbose = "vvv"
    end
  end

  # Define boxes `worker-1' and `worker-2' with a for-loop
  (1..2).each do |i|
    config.vm.define "worker-#{i}" do |worker|
      worker.vm.hostname = "worker-#{i}"
      worker.vm.box = "ubuntu/xenial64"
      worker.vm.network "private_network", ip: "12.0.0.#{i+2}"
      worker.vm.provision "ansible" do |ansible|
        ansible.verbose = "v"
        ansible.playbook = "worker_playbook.yml"
        ansible.extra_vars = {
        ansible_python_interpreter: "/usr/bin/python3",
        manager_ip: "12.0.0.2"
      }
      # Profitez-en pour mettre plus de "verbose" :
      ansible.verbose = "vvv"
      end
    end
  end


end
```

On commence par compléter le « manager_playbook.yml »

```
- hosts: all
  gather_facts: false
   # *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: check swarm is active
       shell: 'docker info | grep "Swarm: active"'
       register: swarm_active
       ignore_errors: true

     - name: swarm init
       when: swarm_active.rc == 1
       shell: docker swarm init  --advertise-addr 12.0.0.2:2377

     - name: get swarm token
       shell: docker swarm join-token -q worker > /vagrant/worker_token

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

L’avant-dernière tâche a pour but de stocker le token dans un fichier partagé afin que les workers le récupèrent et rejoignent le cluster nouvellement créé.

On complète le « worker_playbook.yml » en conséquence :

worker_playbook.yml | Vagrantfile | manager_playbook.yml

```
- hosts: all
  gather_facts: false
   # *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: swarm join
       shell: docker swarm join --token $(cat /vagrant/worker_token) 12.0.0.2:2377

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

Puis on lance la commande « vagrant up ».
On peut maintenant se connecter au manager en SSH :

```
orain@orain-TM1701:~/Documents/td5-antho35/td5/3VM$ vagrant status
Current machine states:

manager                   running (virtualbox)
worker-1                  running (virtualbox)
worker-2                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
orain@orain-TM1701:~/Documents/td5-antho35/td5/3VM$ vagrant ssh manager
```

Puis on lance « docker node ls » et on remarque que 3 VM sont bien présentes et actives dans le swarm :

```
vagrant@manager:~$ sudo docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
dco2m2i5dckkril47c9tarnry *   manager             Ready               Active              Leader              18.09.1
nqzr7n2v4mqgv6gi7212mmfrm     worker-1            Ready               Active                                  18.09.1
tpznbpkc22an4r1ch3fpze0mp     worker-2            Ready               Active                                  18.09.1
```

Cette première partie de TD s’arrête ici, j’ai pu comprendre le fonctionnement d’outils de configuration et de déploiement tels que vagrant et ansible pour gérer des machines virtuelles.