TP Ansible

Boris Teabe, boris.teabe@inp-toulouse.fr

The objective of this lab is to introduce Ansible and automate the deployment of an architecture consisting of multiple Apache2 servers behind a HAProxy load balancer using Ansible.

Connection

You must be connected either to WifiInp if using your personal computer, or to one of the lab machines at N7.

The lab will be conducted on two virtual machines (VMs). Each student will have access to two VMs, which will form their cluster.

Your supervisor will assign two numbers to you (xx and yy), which you will use to connect to your virtual machines. Throughout the lab, these virtual machines will be referred to as your servers. The password is **toto** for both.

```
ssh -p 130xx ubuntu@147.127.121.83

ssh -p 130yy ubuntu@147.127.121.83
```

Ansible Installation

You will need to modify the hosts file (`/etc/hosts`) and designate one server as the master and the other as the slave. First, retrieve the IP addresses of both servers using the command `ip a` to obtain the IP addresses. Then, update the `/etc/hosts` file accordingly.

This is an example of /etc/hosts content.

```
127.0.0.1 localhost
MasterIP master
SlaveIP slave
....
```

Installing the necessary packages for Ansible.

```
sudo bash
apt-get update
apt-get install sshpass
apt-get install python3
apt-get install python3-pip
exit
pip install virtualenv
apt-get install python3-venv
pip install virtualenv
```

Create a Python virtual environment.

```
source ansible/bin/activate
Install ansible
pip install ansible
```

Create a file with the list of machines, `nodes.ini`. Here is an example of the content.

```
MasterIP master
SalveIP slave
```

ssh key exchange.

```
sh-keygen
ssh-copy-id -i /hom/ubuntu/.ssh/id_rsa.pub
ssh-copy-id -i /home/ubuntu/.ssh/id_rsa.pub ubuntu@slave
ssh-copy-id -i /home/ubuntu/.ssh/id_rsa.pub ubuntu@master
```

Check your ansible installation

```
ansible -i node.ini -m ping all  --user ubuntu --ask-pass
```

Ansible palybook

First playbook: Installing Python on the slave node.

Create a directory named `playbooks, then create your first playbook,` installPython.yml`.

Here is the content of the file (read and try to understand. If there is any questions, do not hesitate to ask).

```
- name: My first playbook to install python
  hosts: slave # here give your instance name
  user: ubuntu
  become: yes
  tasks:
    - name: Install Python3
      package:
        name: python3
        state: present
```

Run the playbook.

```
ansible-playbook playbooks/installPython.yml  -i node.ini -kK
```

Install ansible module for docker

```
ansible-galaxy collection install community.docker
```

To install Docker on the slave node using Ansible we will create a playbook named `installDocker.yml`.

Create a file `docker.sh` in the `/home/ubuntu/` directory. The content or this file is:

```
#!/bin/bash
sudo apt install ca-certificates curl gnupg lsb-release -y

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo
"$ID")/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/$(. /etc/os-
release; echo "$ID") $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null

apt-get update

apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin -y
```

Here is the content of `installDocker.yml`

```
- name: Playbook to install docker
  hosts: slave
  user: ubuntu
  become: yes
  tasks:
    - name: Update apt repo and cache
      apt: update_cache=yes force_apt_get=yes cache_valid_time=3600
    - name: Copy Docker installation script update script
      ansible.builtin.copy:
        src: /home/ubuntu/docker.sh
        dest: ~/docker.sh
        mode: 0770
    - name: Run Installation of Docker
      command: ~/docker.sh
```

Run the `intallDocker.yml` playbook and check the output.

Let us start a container on the slave using ansible.

Create a new playbook name `startContainer.yml` with the following content and run it.

```
- name: pull an image
   community.docker.docker_image:
     name: ubuntu
     source: pull
- name: Start a docker container
   community.docker.docker_container:
     name: testdocker
     state: started
     image: ubuntu
     command: /bin/sleep infinity
```

Deploying apache from docker labwork

Now, move back on the docker labwork.

We will build the docker images `apache:v1` form the docker labwork. You need to download all the files from the docker labwork (`index.php` and `haproxy.cfg`).

You also need to recreate `start-apache2.sh` and `Dockerfile` as described in the labwork.

You should copy all these files (`Dockerfile, start-apache2.sh and index.php`) to a new directory: `/home/ubuntu/dockerFile`.

Then we create a playbook to create our image `apache:v1`.

```
- name: Init playbook for apache:v1 image creation
  hosts: slave
  user: ubuntu
  become: yes
  become_method: sudo
  tasks:
    - name: Create a directory on the slave machine if it does not exist
      ansible.builtin.file:
        path: ~/dockerFile
        state: directory
        mode: '700'
    - name: Copy files,
      ansible.builtin.copy:
        src: /home/ubuntu/dockerFile/
        dest: ~/dockerFile/
    - name: Change permission on the start-apache2.sh file
      ansible.builtin.file:
        path: ~/dockerFile/start-apache2.sh
        mode: '+x'
    - name: Build the image
      community.docker.docker_image:
        tag: v1
        name: apache
        source: build
        state: present
        build:
          path: ~/dockerFile/
```

Run the playbook and check the output.

Create a playbook that will start a container from the `apache:v1` image and try to access your `index.php` application as we did in the docker labwork.

For those who can, redo the Docker lab with HAProxy and an Apache server using ansible, and test your deployment.