# Solution M4: Ansible

This is one possible and fully automated solution of the tasks included in the homework

You must adjust some values like IP addresses, image (or box) names, repository names, credentials, etc. to match your use case

## Task 1

We will follow different approach this time

We will ask Vagrant to use a local Ansible provisioner

Our **Vagrantfile** will look like this

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure(2) do |config|

  config.ssh.insert\_key = false

  config.vm.define "docker" do |docker|

    docker.vm.box = "shekeriev/centos-8-minimal"

    docker.vm.hostname = "docker.dob.lab"

    docker.vm.network "private\_network", ip: "192.168.99.100"

    docker.vm.network "forwarded\_port", guest: 80, host: 8000

    docker.vm.provision "shell", inline: <<EOS

echo "\* Add EPEL repository ..."

dnf install -y epel-release

echo "\* Install Python3 ..."

dnf install -y python3

echo "\* Install Python docker module ..."

pip3 install docker

EOS

    docker.vm.provision "ansible\_local" do |ansible|

      ansible.become = true

      ansible.install\_mode = :default

      ansible.playbook = "playbook.yml"

      ansible.galaxy\_role\_file = "requirements.yml"

      ansible.galaxy\_roles\_path = "/etc/ansible/roles"

      ansible.galaxy\_command = "sudo ansible-galaxy install --role-file=%{role\_file} --roles-path=%{roles\_path} --force"

    end

  end

end

The provisioning is done in two steps

First, we use the **shell** provisioner to add some prerequisites, mostly the **Python** related packages which are required for the **docker\_container** module that will be used in the **Ansible** playbook later

Next, we set the **ansible\_local** provisioner

It will install **Ansible** locally on the machine using the package system

Next, it will install a role based on the **requirements.yml** file which has the following content

# from galaxy

- src: geerlingguy.docker

And finally, it will launch the **playbook.yml** file which has the following content

---

- hosts: all

  roles:

    - geerlingguy.docker

  tasks:

    - name: Add vagrant user to docker group

      user:

        name: 'vagrant'

        groups: docker

        append: yes

    - name: Spin-up one NGINX container

      docker\_container:

        name: con-nginx

        image: nginx

        state: started

        ports:

          - "80:80"

Finally, we will end up with a container with **NGINX** in it, which we can reach on http://localhost:8000

## Task 2

For this part of the homework we will need two machines

Knowing that we can use local **Ansible** provisioning in the **Vagrantfile** we will utilize it this time again

Our **Vagrantfile** will look like this

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure(2) do |config|

  config.ssh.insert\_key = false

  config.vm.define "web" do |web|

    web.vm.box = "shekeriev/centos-8-minimal"

    web.vm.hostname = "dob-web.dob.lab"

    web.vm.network "private\_network", ip: "192.168.99.100"

    web.vm.network "forwarded\_port", guest: 80, host: 8000

    web.vm.synced\_folder "web/", "/vagrant"

    web.vm.provision "ansible\_local" do |ansible|

      ansible.become = true

      ansible.install\_mode = :default

      ansible.playbook = "playbook.yml"

    end

  end

  config.vm.define "db" do |db|

    db.vm.box = "shekeriev/ubuntu-20-04-server"

    db.vm.hostname = "dob-db.dob.lab"

    db.vm.network "private\_network", ip: "192.168.99.101"

    db.vm.synced\_folder "db/", "/vagrant"

    db.vm.provision "shell", inline: <<EOS

echo "\* Disable auto-update timers and service if present ..."

systemctl disable --now apt-daily-upgrade.timer &> /dev/null || true

systemctl disable --now apt-daily.timer &> /dev/null || true

systemctl disable --now unattended-upgrades.service &> /dev/null || true

sed -i s/1/0/g /etc/apt/apt.conf.d/20auto-upgrades || true

EOS

    db.vm.provision "ansible\_local" do |ansible|

      ansible.become = true

      ansible.install\_mode = :default

      ansible.playbook = "playbook.yml"

      ansible.galaxy\_role\_file = "requirements.yml"

      ansible.galaxy\_roles\_path = "/etc/ansible/roles"

      ansible.galaxy\_command = "sudo ansible-galaxy install --role-file=%{role\_file} --roles-path=%{roles\_path} --force"

    end

  end

end

Both machines are provisioned with local **Ansible** provider

Additionally, the Ubuntu machine uses a inline shell provider to disable the auto-update functionallity

This time, there is no need of installing upfront additional packages using another provisioner

Let us create a **web/playbook.yml** file with the following content

---

- hosts: all

  tasks:

    - name: Add dob-web host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.100 dob-web.dob.lab dob-web'

    - name: Add dob-db host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.101 dob-db.dob.lab dob-db'

    - name: Install Apache web server and PHP

      dnf:

        name:

          - httpd

          - php

          - php-mysqlnd

        state: present

    - name: Disable SELinux

      selinux:

        state: disabled

    - name: Copy site files

      copy: src=site/ dest=/var/www/html/

    - name: Open HTTP port in the firewall

      firewalld:

        service: http

        permanent: true

        state: enabled

        immediate: true

    - name: Start Apache web server

      service:

        name: httpd

        state: started

        enabled: yes

Here we did not use any pre-defined roles, we installed packages manually instead

Next, we must take care of the **db** host

For the **db** host we decided to use a role, so we created **db/requirements.yml** file

# from galaxy

- src: geerlingguy.mysql

And then the playbook file in **db/playbook.yml** with the following content

---

- hosts: all

  vars\_files:

    - vars/main.yml

  roles:

    - geerlingguy.mysql

  tasks:

    - name: Add dob-web host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.100 dob-web.dob.lab dob-web'

    - name: Add dob-db host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.101 dob-db.dob.lab dob-db'

    - name: Import DB

      shell: mysql -u root --password=root < db\_setup.sql

    - name: Allow all access to tcp port 3306

      ufw:

        rule: allow

        port: 3306

        proto: tcp

Note, that there is a link to a variables file. This is needed in or der to fine-tune the role to match the requirements and conditions for Ubuntu. Its content is:

mysql\_bind\_address: '0.0.0.0'

mysql\_packages:

  - mariadb-client

  - mariadb-server

Now, that we have all the required files, we can run the environment with

**vagrant up**

After a while, our environment will be ready, and our application will be accessible at [**http://localhost:8000**](http://localhost:8000)on the host