# Solution M5: Jenkins

## Steps

First, we will start with the **Vagrantfile** which will spin up two machines

# -\*- 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: "192.168.99.101"

    docker.vm.network "forwarded\_port", guest: 80, host: 8082, auto\_correct: true

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

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

      ansible.become = true

      ansible.install\_mode = :default

      ansible.playbook = "playbook-docker.yml"

      ansible.galaxy\_role\_file = "requirements-docker.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

  config.vm.define "jenkins" do |jenkins|

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

    jenkins.vm.hostname = "jenkins.dob.lab"

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

    jenkins.vm.network "forwarded\_port", guest: 8080, host: 8080, auto\_correct: true

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

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

      ansible.become = true

      ansible.install\_mode = :default

      ansible.playbook = "playbook-jenkins.yml"

      ansible.galaxy\_role\_file = "requirements-jenkins.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

We are using local **Ansible** provisioning

For the **docker** host, the **vagrant/docker/requirements-docker.yml** file contains just two roles

# from galaxy

- src: geerlingguy.docker

- src: geerlingguy.java

And the **vagrant/docker/playbook-docker.yml** file contains all provisioning instructions

---

- hosts: all

  roles:

    - geerlingguy.java

    - geerlingguy.docker

  tasks:

    - name: Add jenkins host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.100 jenkins.dob.lab jenkins'

    - name: Add docker host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.101 docker.dob.lab docker'

    - name: Add vagrant user to docker group

      user:

        name: vagrant

        groups: docker

        append: yes

    - name: Install git

      dnf:

        name: git

        state: present

    - name: Enable HTTP communication

      firewalld:

        service: http

        permanent: true

        state: enabled

        immediate: true

The **jenkins** host is provisioned in the same manner

First, the **vagrant/jenkins/requirements-jenkins.yml** file states the required roles

# from galaxy

- src: geerlingguy.jenkins

- src: geerlingguy.java

Then the actual **vagrant/jenkins/playbook-jenkins.yml** playbook does the whole job

---

- hosts: all

  vars:

    my\_jenkins\_plugins:

      build-pipeline-plugin:

        enabled: yes

      ssh:

        enabled: yes

      ssh-slaves:

        enabled: yes

      git:

        enabled: yes

      github:

        enabled: yes

      log-parser:

        enabled: yes

      copyartifact:

        enabled: yes

      template-project:

        enabled: yes

  roles:

    - geerlingguy.java

    - geerlingguy.jenkins

  tasks:

    - name: Add jenkins host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.100 jenkins.dob.lab jenkins'

    - name: Add dob-db host

      lineinfile:

        path: /etc/hosts

        line: '192.168.99.101 docker.dob.lab docker'

    - name: Open port 8080/tcp

      firewalld:

        port: 8080/tcp

        permanent: true

        state: enabled

        immediate: true

    - name: Install plugins

      jenkins\_plugin:

        name: "{{ item.key }}"

        timeout: 90

        url\_username: admin

        url: http://localhost:8080

        url\_password: admin

      register: my\_jenkins\_plugin\_installed

      with\_dict: "{{ my\_jenkins\_plugins }}"

    - name: Initiate the fact

      set\_fact:

        jenkins\_restart\_required: no

    - name: Check if restart is required

      set\_fact:

        jenkins\_restart\_required: yes

      when: item.changed

      with\_items: "{{ my\_jenkins\_plugin\_installed.results }}"

    - name: Restart Jenkins if required

      service:

        name: jenkins

        state: restarted

      when: jenkins\_restart\_required

    - name: Wait for Jenkins to start up

      uri:

        url: http://localhost:8080

        status\_code: 200

        timeout: 5

      register: jenkins\_service\_status

      # Keep trying for 5 mins in 5 sec intervals

      retries: 60

      delay: 5

      until: >

          'status' in jenkins\_service\_status and

          jenkins\_service\_status['status'] == 200

      when: jenkins\_restart\_required

    - name: Reset the fact

      set\_fact:

        jenkins\_restart\_required: no

      when: jenkins\_restart\_required

    - name: Plugin enabling

      jenkins\_plugin:

        name: "{{ item.key }}"

        timeout: 90

        url\_username: admin

        url: http://localhost:8080

        url\_password: admin

        state: "{{ 'enabled' if item.value['enabled'] else 'disabled'}}"

      when: >

        'enabled' in item.value

      with\_dict: "{{ my\_jenkins\_plugins }}"

    - name: Add Jenkins credentials

      command: /vagrant/jenkins-add-credentials.sh

    - name: Add Jenkins slave node

      command: /vagrant/jenkins-add-node.sh http://localhost:8080/ docker.dob.lab vagrant

    - name: Create Jenkins job

      command: /vagrant/jenkins-create-job.sh

There are three **bash** shell scripts initially located in the **vagrant/jenkins** folder

They are used to add credentials for the **vagrant** user, register and provision the **Jenkins Agent** and to create the build job

As usual, the whole infrastructure id brought up with one simple command

**vagrant up**