casos de uso

vagrant con Tomcat (7/8) y JDK (7/8)

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "peru/ubuntu-18.04-desktop-amd64"
  config.vm.box_version = "20190401.01"
  config.vm.provision "file", source: "tomcat.service", destination:
"/tmp/tomcat.service"

  config.vm.provider :virtualbox do |v|
    v.name = "altrankas-tomcat7-JDK8"
    v.linked_clone = true
    v.memory = 2048
    v.cpus = 2
  end

  config.vm.provision "shell", path: "install.sh"

  config.vm.provision "shell", path: "addInsecureKey.sh"

end
```

tomcat.service

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=REPLACE_PATH_HERE/jre
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
```

```
WantedBy=multi-user.target
```

install.sh

```
#!/bin/bash

#
http://apache.uvigo.es/tomcat/tomcat-7/v7.0.93/bin/apache-tomcat-7.0.93.tar.gz
TOMCAT_TGZ="apache-tomcat-7.0.93.tar.gz"
TOMCAT_URL_DOWNLOAD="http://node0.a11.altran.es:9090/${TOMCAT_TGZ}"
TOMCAT_PATH="/opt/tomcat"
JAVA_PATH="/usr/lib/jvm/java-8-oracle"
JAVA_INSTALLER="oracle-java8-installer"
DEBIAN_FRONTEND=noninteractive

sudo add-apt-repository -y ppa:webupd8team/java
sudo apt-get -y update

echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 select true"
| sudo debconf-set-selections
sudo apt-get install -y curl ${JAVA_INSTALLER} > /dev/null 2>&1

sudo echo "JAVA_HOME=\"${JAVA_PATH}/bin/\"" >> /etc/environment

sudo groupadd tomcat
sudo useradd -s /bin/false/ -g tomcat -d ${TOMCAT_PATH} tomcat
sudo mkdir -p ${TOMCAT_PATH}
sudo curl ${TOMCAT_URL_DOWNLOAD} --output /tmp/${TOMCAT_TGZ}
sudo tar xzvf /tmp/${TOMCAT_TGZ} -C ${TOMCAT_PATH} --strip-components=1
sudo rm /tmp/${TOMCAT_TGZ}
sudo chgrp -R tomcat ${TOMCAT_PATH}
sudo chmod -R g+r ${TOMCAT_PATH}/conf
sudo chmod g+x ${TOMCAT_PATH}/conf
sudo chown -R tomcat ${TOMCAT_PATH}/webapps/ ${TOMCAT_PATH}/work/
${TOMCAT_PATH}/temp/ ${TOMCAT_PATH}/logs/

sudo mv /tmp/tomcat.service /etc/systemd/system/tomcat.service
sudo sed -i "s|REPLACE_PATH_HERE|${JAVA_PATH}|g"
/etc/systemd/system/tomcat.service
sudo systemctl daemon-reload
sudo systemctl enable tomcat
sudo sed -i '/<\/tomcat-users>/ i\  <user username="admin" password="admin"
roles="admin,manager-gui,admin-gui"/>' ${TOMCAT_PATH}/conf/tomcat-users.xml
sudo systemctl start tomcat
sudo echo "CATALINA_HOME=\"${TOMCAT_PATH}\"" >> /etc/environment
```

addInsecurekey.sh

```
#!/bin/bash
su - vagrant
echo "ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7gHzIw+
niNltGEFHzD8+v1I2YJ6oXevct1YeS0o9HZyN1Q9qgCgzUFtdOKLv6IedplqoPkcmF0aYet2PkEDo3
MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx4O7PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilF
```

```
R61RGL+GPXQ2MWZWFYbAGjyiYJnAmCP3NOTd0jMZEnDkbUvxhMmBYSdETk1rRgm+R4LOzFUGaHqHDL
KLX+FIPKcF96hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure
public key
" >> .ssh/authorized_keys
```

despligue

- crear una box en Vagrant : `vagrant package <nombre_maquina> –output fichero.box`
- Vagrantfile

3 nodos vagrant + docker

la idea es desplegar un cluster swarm

```
Vagrant.configure("2") do |config|
   config.vm.define "nodo1" do |nodo1|
     nodo1.vm.box = "debian/jessie64"
     nodo1.vm.hostname = "nodo1"
#     nodo1.vm.network "public_network", bridge: "eth0"
     nodo1.vm.network "private_network", ip: "10.0.100.101"
     nodo1.vm.provision "shell", path: "install-docker.sh"
     nodo1.vm.provider "virtualbox" do |vb|
       vb.name = "debian-swarm-nodo1"
       vb.memory = 1024
       vb.cpus = 1
       vb.linked_clone = true
     end
   end
   config.vm.define "nodo2" do |nodo2|
     nodo2.vm.box = "debian/jessie64"
     nodo2.vm.hostname = "nodo2"
     nodo2.vm.network "private_network", ip: "10.0.100.102"
     nodo2.vm.provider "virtualbox" do |vb|
       vb.name = "debian-swarm-nodo2"
       vb.memory = 1024
       vb.cpus = 1
       vb.linked_clone = true
     end
   end
   config.vm.define "nodo3" do |nodo3|
     nodo3.vm.box = "debian/jessie64"
     nodo3.vm.hostname = "nodo3"
     nodo3.vm.network "private_network", ip: "10.0.100.103"
     nodo3.vm.provider "virtualbox" do |vb|
       vb.name = "debian-swarm-nodo3"
       vb.memory = 1024
```

```
      vb.cpus = 1
      vb.linked_clone = true
    end
  end
 end
```

```
#!/bin/bash
sudo apt-get remove -y docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install -y \
   apt-transport-https \
   ca-certificates \
   curl \
   gnupg2 \
   software-properties-common \
   vim
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/debian \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get install -y docker-ce
sudo usermod -aG docker vagrant
#newgrp docker
#docker run hello-world
```

vagrant + docker

```
Vagrant.configure(2) do |config|
  config.vm.box = "debian/jessie64"
  config.vm.hostname = "debivan"
  config.vm.network "forwarded_port", guest: 80, host: 1080
  config.vm.network  "public_network", bridge: "enp0s31f6"
  config.vm.synced_folder "./data","/vagrant"
#  config.vm.provision "shell", path: "install.sh"
  config.vm.provision "docker" do |d|
    d.pull_images "nginx"
    d.run "nginx",
      args: "-v '/vagrant:/usr/share/nginx/html' -p '80:80'"
  end
  config.vm.provider "virtualbox" do |vb|
    vb.name = "debivan-vm"
    vb.memory = "512"
    vb.cpus = 2
    vb.linked_clone = true
  end
end
```

configuraciones simples

```
Vagrant.configure(2) do |config|
  config.vm.box = "debian/jessie64"
  config.vm.hostname = "prueba1"
  config.vm.provision "shell", path: "actualizar.sh"

  config.vm.network :private_network, ip: "192.168.100.10" # only host
  config.vm.network :public_network, ip: "192.168.100.20"
  config.vm.network :public_network,:bridge=>"eth0"
  config.vm.network :forwarded_port, guest: 80, host: 8080
  config.vm.synced_folder "./www","/www"

  config.vm.provider "virtualbox" do |vb|
    vb.name = "VM-Prueba1"
    vb.memory = 1024
    vb.cpus = 3
    vb.linked_clone = true
    vb.gui = true
  end
end
```

```
  Vagrant.configure("2") do |config|
     config.vm.define "web" do |nodo1|
       nodo1.vm.box = "debian/jessie64"
       nodo1.vm.hostname = "web"
       nodo1.vm.network "public_network", bridge: "eth0"
       nodo1.vm.network "private_network", ip: "10.0.100.101"
     end
     config.vm.define "db" do |nodo2|
       nodo2.vm.box = "debian/jessie64"
       nodo2.vm.hostname = "db"
       nodo2.vm.network "private_network", ip: "10.0.100.102"
     end
   end
```

master (Dani)

- la configuración de las VM está en fichero separado

```
API_VERSION = '2'
VMS_FILE_PATH = 'vagrant/vms.yml'

hosts = YAML.load_file(VMS_FILE_PATH)

Vagrant.configure(API_VERSION) do |config|
  hosts.each do |host|
```

```
    config.vm.define host['name'] do |node|
      node.vm.box = host['box']
      node.vm.hostname = host['hostname']
      node.vm.network :private_network, ip: host['ip']

      host['sync']&.each { |sync| config.vm.synced_folder sync['host'],
sync['guest'], type: sync['type'] }
      host['ports']&.each { |port| node.vm.network :forwarded_port, guest:
port['guest'], host: port['host'] }

      if host['ansible_local']
        host['ansible_local']['playbooks']&.each { |pb|
          config.vm.provision :ansible_local do |ansible|
             ansible.limit = pb['limit']
             ansible.become = pb['become']
             ansible.playbook = pb['playbook']
             ansible.inventory_path = pb['inventory']
             ansible.version = host['ansible_local']['config']['version']
             ansible.install_mode = host['ansible_local']['config']['mode']
          end
        }
      end

      config.vm.provider :virtualbox do |v|
        v.linked_clone = true
        v.name = host['name']
        v.cpus = host['cpus']
        v.memory = host['memory']
      end
    end
  end
end
```

```
- name: openshift-master
  box: centos/7
  hostname: 10.0.0.10
  memory: 2048
  cpus: 2
  ip: 10.0.0.10
  ports:
    - host: 8443
      guest: 8443
  sync:
    - host: .
      guest: /vagrant
      type: nfs

- name: openshift-node1
  box: centos/7
  hostname: 10.0.0.11
  memory: 2048
  cpus: 2
  ip: 10.0.0.11
  ports:
```

```
    - host: 8081
      guest: 80
    - host: 8481
      guest: 443
  sync:
    - host: .
      guest: /vagrant
      type: nfs

- name: openshift-node2
  box: centos/7
  hostname: 10.0.0.12
  memory: 2048
  cpus: 2
  ip: 10.0.0.12
  ports:
    - host: 8082
      guest: 80
    - host: 8482
      guest: 443
  sync:
    - host: .
      guest: /vagrant
      type: nfs

- name: bastion
  box: centos/7
  memory: 512
  cpus: 1
  ip: 10.0.0.2
  sync:
    - host: .
      guest: /vagrant
      type: nfs
  ansible_local:
    config:
      mode: pip
      version: 2.4.1.0
    playbooks:
      - limit: "all"
        become: true
        playbook: ./ansible/prerequisites.yml
        inventory: ./ansible/inventory/hosts
      - limit: "all"
        become: true
        playbook: ./openshift-ansible/playbooks/byo/config.yml
        inventory: ./ansible/inventory/hosts
```

master

```
VAGRANTFILE_API_VERSION = '2'
DEFAULT_PROVIDER_MEMORY = '256'
```

```
DEFAULT_PROVIDER_CPUS = '2'
DEFAUTL_LINKED_CLONE = true

def do_ports(vm, host)
  if host.has_key?('ports')
    host['ports'].each do |port|
      vm.network :forwarded_port, guest: port['guest'], host: port['host']
    end
  end
end

def do_provision(vm,host)
  if host.has_key?('shell')
    vm.provision "shell",path: host['shell']
  end
end

def do_syncfolder(vm,host)
  if host.has_key?('syncfolder')
    host['syncfolder'].each do |syncf|
      vm.synced_folder syncf['host'], syncf['guest']
    end
  end
end


def do_provider(vm,host)
  memory = DEFAULT_PROVIDER_MEMORY;
  cpus = DEFAULT_PROVIDER_CPUS;

  chars = { :memory => DEFAULT_PROVIDER_MEMORY, :cpus => DEFAULT_PROVIDER_CPUS,
:linked_clone => DEFAULT_LINKED_CLONE }

  if host.has_key?('provider')
    host['provider'].each do |singlechar|
      #memory = host['memory'] ? host['memory'] : DEFAULT_PROVIDER_MEMORY;
      #memory = singlechar['memory'] ? singlechar['memory'] :
DEFAULT_PROVIDER_MEMORY;
      #cpus = singlechar['cpus'] ? singlechar['cpus'] : DEFAULT_PROVIDER_CPUS;
      chars[:memory] = singlechar['memory'] ? singlechar['memory'] :
DEFAULT_PROVIDER_MEMORY;
      chars[:cpus] = singlechar['cpus'] ? singlechar['cpus'] :
DEFAULT_PROVIDER_CPUS;
    end # singlechar
  end

  vm.provider :virtualbox do |v|
    v.name = host['name']
    v.memory = chars[:memory]
    v.cpus = chars[:cpus]
    v.linked_clone = chars[:linked_clone]
  end

  # caracteristicas.each_with_index do |v,k|
  #   #puts "#{k}"
  #   vm.provider.k = v[k]
```

```
  # end

end


hosts = YAML.load_file('Vagranthosts.yml')

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  hosts.each do |host|
    config.vm.define host['name'] do |node|
      node.vm.box = host['box']
      node.vm.hostname = host['name']
      node.vm.network :private_network, ip: host['ip']

      do_ports(node.vm,host)
      do_provision(node.vm,host)
      do_syncfolder(node.vm,host)
      do_provider(node.vm,host)

    end # node
  end # host
end  # config
```

```
- name: c7-gitlab
  ip: 192.168.56.130
  box: centos/7
  shell: c7-gitlab.sh
  ports:
    - host: 8080
      guest: 80
  syncfolder:
    - host: './data'
      guest: '/vagrant'
  provider:
    - cpus: 2
      memory: 512
```