Lab Work Task. Web Server Provisioning

Review

Developing custom modules and filters.Learning by doing.

Task

On Host Node (Control Machine):

1. Create folder ~/cm/ansible/day-3. All working files are supposed to be placed right there.
2. Develop custom filter to select an url to download mongodb depends on OS name and S/W version from https://www.mongodb.org/dl/linux/
   Requirements:
   - Write a playbook (name: **mongodb.yml**) to prove that this module works
   - At least 9 versions of MongoDB for 3 different Linux distributives (list with links)
   - Filter should process a list of urls and takes 3 options: os_family (discovered by ansible, variable, produced by setup module), os release number and mongodb_version (set in play vars)

```
- hosts: localhost
  connection: local
  vars:
    rel_version: 3.0.14
    mongo_src:
    - mongodb-linux-x86_64-ubuntu1204-3.4.2
    - mongodb-linux-x86_64-ubuntu1404-3.4.2
    - mongodb-linux-x86_64-ubuntu1204-3.2.12
    - mongodb-linux-x86_64-rhel62-3.5.4
    - mongodb-linux-x86_64-rhel70-3.5.4
    - mongodb-linux-x86_64-rhel55-3.0.14
    - mongodb-linux-x86_64-rhel64-3.0.14
    - mongodb-linux-x86_64-rhel70-3.0.14
    - mongodb-linux-x86_64-debian81-3.5.3
    - mongodb-linux-x86_64-debian71-3.4.2
    - mongodb-linux-x86_64-debian71-3.4.1
  tasks:
  - debug: msg={{ mongo_src | get_mongo_src( ansible_distribution,
ansible_distribution_major_version, rel_version ) }}
```

```
from __future__ import (absolute_import, division, print_function)
__metaclass__ = type

from ansible import errors


def get_mongo_src(arg, dist, dist_v, rel_v):
    if dist == "CentOS":
        dist = "rhel"
    for item in arg:
        data = str(item).split("-")
        if data[4] == str(rel_v) and str(dist)+str(dist_v) in
data[3]:
            return item
    return "Dependency not found!"


class FilterModule(object):
    def filters(self):
        return {'get_mongo_src': get_mongo_src}
```

3. Develop custom module to manage VirtualBox:
   Arguments:
   - path to vagrantfile
   - state: started, stopped, destroyed
   Return values:
   - state: running, stopped, not created
   - ip address, port
   - path to ssh key file
   - username to connect to VM
   - os_name
   - RAM size
   Errors:
   - file doesn’t exists
   - failed on creation
   - etc

```
#!/bin/bash

source $1

VGPATH=$path
STATE=$state

if [ -z "$VGPATH" ]; then
    printf '{"failed": true, "msg": "missing required arguments: path
to vagrant file."}'
    printf "\n"
    exit 1
fi
if [ -z "$STATE" ]; then
    printf '{"failed": true, "msg": "missing "$STATE" required
arguments: state."}'
    printf "\n"
    exit 1
fi

if [ ! -f $VGPATH/Vagrantfile ]; then
   printf '{"failed": true, "msg": "No file Vagrantfile found."}'
   printf "\n"
   exit 1
fi


cd $VGPATH
BOX_STATE=$(vagrant global-status | grep $VGPATH | awk '{print $4}')

function getting_variables
{
    RES_BOX_STATE=$(vagrant global-status | grep $VGPATH | awk '{print
$4}')
    RES_BOX_PORT=$(vagrant ssh-config | grep Port | awk '{print $2}'
2>/dev/null)
    RES_BOX_IPADDR=$(vagrant ssh -c "hostname -I | cut -d' ' -f2"
2>/dev/null)
    RES_BOX_SSH_PATH=$(vagrant ssh-config | grep IdentityFile | awk
'{print $2}' 2>/dev/null)
    RES_BOX_USR=$(vagrant ssh-config | grep -w "User" | awk '{print
$2}' 2>/dev/null)
    RES_BOX_OS=$(vagrant ssh -c "cat /etc/redhat-release" 2>/dev/null)
    RES_BOX_MEM=$(vagrant ssh -c "cat /proc/meminfo | grep MemTotal |
awk '{print \$2 \$3}'" 2>/dev/null)

    printf '{"changed": true, "failed": false, "msg": "Well Done!",
"RES_BOX_STATE": "%s", "RES_BOX_PORT": "%s", "RES_BOX_IPADDR": "%s",
"RES_BOX_SSH_PATH": "%s", "RES_BOX_USR": "%s", "RES_BOX_OS": "%s",
"RES_BOX_MEM": "%s" }' "$RES_BOX_STATE" "$RES_BOX_PORT"
"$RES_BOX_IPADDR" "$RES_BOX_SSH_PATH" "$RES_BOX_USR" "$RES_BOX_OS"
"$RES_BOX_MEM"
   exit 0

}

function start_machine
{
    if [ "$BOX_STATE" == "running" ]; then
        getting_variables
    else
        cd $VGPATH; vagrant up
        getting_variables
    fi
    printf '{"failed": false, "msg": "Continue."}'
}

function stop_machine
{
    if [ "$BOX_STATE" == "" ]; then
        printf '{"failed": false, "msg": "Machine not created."}'
        printf "\n"
    elif [ "$BOX_STATE" == "poweroff" ]; then
        printf '{"failed": false, "msg": "Machine currently stopped."}'
        printf "\n"
    elif [ "$BOX_STATE" == "running" ]; then
        cd $VGPATH; vagrant halt --force
        printf '{"failed": false, "msg": "Machine stopped."}'
        printf "\n"
    else
        printf '{"failed": true, "msg": "Something goes wrong. Cant get
machine status"}'
        printf "\n"
        exit 1
    fi
}

function destroy_machine
{
    if [ "$BOX_STATE" == "" ]; then
        printf '{"failed": false, "msg": "Machine currently stopped."}'
        printf "\n"
    else
        cd $VGPATH; vagrant destroy --force
    fi
}

case $STATE in
    started)
        start_machine
    ;;
    stopped)
        stop_machine
    ;;
    destroyed)
        destroy_machine
    ;;
    *)
        printf '{"failed": true, "msg": "invalid state selected
{started | stopped | destroyed}"}'
        printf "\n"
        exit 1
    ;;
esac
```

4. Create a playbook (name: **stack.yml**) to provision Tomcat stack (nginx + tomcat) on VirtualBox VM
   Requirements:
   - 2 Plays: provision VM, roll out Tomcat stack (using roles from previous lab work)
   - 2nd play should work with dynamically composed Inventory (connection settings to VM), http://docs.ansible.com/ansible/add_host_module.html

```
5. # STARTING VM
   - name: Local
     hosts: localhost

     tasks:
     - name: vagrant started
       runbox:
         path: /home/student/cm/ansible/day-3
         state: started
       register: contents

     - debug: msg={{contents}}

     - add_host:
          name: vagrant
          ansible_port: "{{contents.RES_BOX_PORT}}"
          ansible_host: 127.0.0.1
          ansible_connection: ssh
          ansible_user: "{{contents.RES_BOX_USR}}"
          ansible_ssh_private_key_file: "{{contents.RES_BOX_SSH_PATH}}"
       when: contents.RES_BOX_STATE == 'running'
   # INSTALLATION PLAY
   - name: Installation
     hosts: vagrant
     vars_files:
     - variables.yml
     roles:
     - java
     - nginx
     - tomcat
   #VERIFICATION PLAY
   - name: Verification
     vars_files:
     - variables.yml
     hosts: vagrant
     roles:
     - java_test
     - nginx_test
     - tomcat_test
```

6. Verification Procedure: playbook will be checked by instructor’sCI system as follows:
    6.1 Connect to student’s host by ssh (username “student”) with own ssh key.
    6.2 Go into the folder mentioned in point 1
    6.3 Destroy: vagrant destroy
    6.4 Execute VM provisioning: ansible-playbook stack.yml -i localhost, -c local -vv
    6.5 If previous steps are done successfully, instructor will check report (pdf-file)
7. Feedback: report issues/problems you had during the development of playbook and time spent for development.