Ansible Labs

Variables Lab

Step 1 : Create directory var-labs and change into the directory
# mkdir var-labs
# cd var-labs

First, create the playbook playbook.yml and define the following variables in the vars section: web_pkg, which defines the name of the package to install for the web server; web_service for the name of the web service to manage; Add the python_pkg variable to install a required package for the uri module; and rule, which defines the service to open.

```
- name: Install Apache and start the service
  hosts: all
  vars:
    web_pkg: httpd
    web_service: httpd
    python_pkg: python-httplib2
    rule: http
```

Step 2 : Create the tasks block and create the first task which uses the yum module to install the required packages.

```
tasks:
  - name: Install the required packages
    yum:
       name:
          - "{{ web_pkg }}"
          - "{{ python_pkg }}"
      state: latest
```

Step 3 : Create task to start and enabled the httpd service.

```
 - name: Start and enable the {{ firewall_service }} service
   service:
     name: "{{ firewall_service }}"
     enabled: true
     state: started
```

Step 4 : Add a task that will create some content in /var/www/html/index.html.

```
- name: Create web content to be served
  copy:
     content: "Example web content"
     dest: /var/www/html/index.html
```

Step 5 : Add a task that will use the firewalld module to add a rule for the web service.

```
- name: Open the port for {{ rule }}
  firewalld:
    service: "{{ rule }}"
    permanent: true
    immediate: true
    state: enabled
```

Step 6 : Save the playbook. The playbook can be run using the ansible-playbook command. Watch the output as Ansible starts by installing the packages, starting and enabling the services, and ensuring the web server is reachable

```
# ansible-playbook playbook.yml
```

Step 7 : Check the output by doing curl ipofclient.

Note: The code reference is available on http://github.com/vsaini44/ansible_class/var_lab1.yml

Inclusion Lab

Step 1 : **Create directory inc-labs and change into the directory**
# mkdir inc-labs
# cd inc-labs

One task file, one variable file, and one playbook will be created for this exercise. The variable file defines, in YAML format, a variable used by the playbook. The task file defines the required tasks and includes variables that will be passed later on as arguments.

Create a directory called tasks and change into that directory.
# mkdir tasks && cd tasks

Step 2 : In the tasks directory, create the environment.yml task file. Define the two tasks that install and start the web server; use the package variable for the package name, service for the service name, and svc_state for the service state.

Tasks # vim environment.yml

```
  - name: Install the {{ package }} package
    yum:
      name: "{{ package }}"
      state: latest
  - name: Start the {{ service }} service
    service:
      name: "{{ service }}"
      state: "{{ svc_state }}"
```

Step 3 : Change back into the main project directory. Create a directory named vars and change into that directory

tasks# cd ..
# mkdir vars && cd vars

Step 4 : . In the vars directory, create the variables.yml variables file. The file defines the firewall_pkg variable in YAML format. The file should read as follows:

```
---
firewall_pkg: firewalld
```

Step 5 : Change back to the top-level project directory for the playbook.
Vars# cd ..

Step 6 : Create and edit the main playbook, named playbook.yml. The playbook imports the tasks as well as the variables; and it installs the firewalld service and configures it.
Start by adding the webserver host group. Define a rule variable with a value of http

```
---
- hosts: webserver
  vars:
    rule: http
```

Step 7 : Continue editing the playbook.yml file. Define the first task, which uses the include_vars module to import extra variables in the playbook. The variables are used by other tasks in the playbook. Include the variables.yml variable file created previously.

```
tasks:
  - name: Include the variables from the YAML file
    include_vars: vars/variables.yml
```

Step 8 : Define the second task which uses the include module to include the base environment.yml playbook. Because the three defined variables are used in the base playbook, but are not defined, include a vars block. Set three variables in the vars section: package set as httpd, service set as httpd, and svc_state set as started.

```
 - name: Include the environment file and set the variables
   include: tasks/environment.yml
   vars:
     package: httpd
     service: httpd
     svc_state: started
```

Step 9 : Create three more tasks: one that installs the firewalld package, one that starts the firewalld service, and one that adds a rule for the HTTP service. Use the variables that were defined previously.

```
- name: Install the firewall
  yum:
    name: "{{ firewall_pkg }}"
    state: latest
```

```
 - name: Start the firewall
   service:
     name: firewalld
     state: started
     enabled: true
```

```
 - name: Open the port for {{ rule }}
   firewalld:
     service: "{{ rule }}"
     immediate: true
     permanent: true
     state: enabled
```

Step 10 : Finally, add a task that creates the index.html file for the web server using the copy module. Create the file with the Ansible ansible_fqdn fact, which returns the fully qualified domain name. Also include a time stamp in the file using an Ansible fact. The task should read as follows:

```
- name: Create index.html
  copy:
    content: "{{ ansible_fqdn }} has been customized using Ansible on the
{ ansible_date_time.date }}\n"
   dest: /var/www/html/index.html
```

Step 11: Run the Playbook and check the curl output now

```
# ansible-playbook incursion.yml
```

Note: The code reference is available on http://github.com/vsaini44/ansible_class/incursion.yml