Creating an Apache Server on Ubuntu Using Ansible Roles

For this project, you'll need two Ubuntu machines. The first one will be your Ansible controller and the second one will be your target machine for Apache installation. Before starting you should make sure you can connect to your target machine from your controller through Ansible.

You can use the following command to see if everything is working:

```
# ansible all -m ping
172.31.16.233 | SUCCESS => {
"changed": false,
"ping": "pong"
}
```

The 172.31.16.233 is defined in the /etc/ansible/hosts file as:

```
[ubuntu]
ubuntu ansible_host=172.31.16.233
ansible_ssh_private_key_file=/root/key.pem ansible_user=ubuntu
```

Ansible Roles

In your /etc/ansible, there should be a roles folder. Go into the folder and issue the following command:

```
# ansible-galaxy init apache --offline
- apache was created successfully
```

The command should automatically create the following structure:

```
`-- apache
|-- README.md
|-- defaults
|   `-- main.yml
|-- files
|-- handlers
|   `-- main.yml
|-- meta
|   `-- main.yml
|-- tasks
|   `-- main.yml
|-- templates
|-- tests
```

```
| |-- inventory
| `-- test.yml
`-- vars
`-- main.yml
```

Here are the main components we will use in this lesson:

- tasks/main.yml – It is the starting point of the role tasks. You can use the main.yml to point to other task files.
- handlers/main.yml – It contains the handlers.
- files – You can keep your files and resources that you want to deploy here.

The other folders (not used in this tutorial):

- defaults/main.yml – It contains the default variables for the role.
- meta/main.yml – It contains the metadata information for the role.
- templates – It is a folder to place Jinja2 templates.
- test – It can be used for setting up inventory and test cases.
- vars/main.yml — It is used for variable setup.

Let's start with the tasks/main.yml. Paste the following code inside:

```
---
# tasks file for apache
- include_tasks: install.yml
- include_tasks: configure.yml
- include_tasks: service.yml
```

We are dividing the tasks into smaller portions and pointing to other YAML files. So we need to create those files.

install.yml

Inside /etc/ansible/roles/apache/tasks, let's create install.yml with the following code:

```
---
# installing apache2
- name: installing apache2 server
  apt:
    name: apache2
    state: present
```

It is installing apache2 on the Apache server. It's using apt because our target machine is running Ubuntu.

files, configure.yml and handlers/main.yml

Let's set up some files and resources in the /etc/ansible/roles/apache/files/ folder. First, you can get a standard apache2.conf file, make your custom changes and put it in the folder. In our case, we are just going to use default apache conf file:

```
apt install apache2 -y
cp /etc/apache2/apache2.conf /etc/ansible/roles/apache/files/
```

During the run process, ansible will take this apache2.conf file and replace it on the target machine.

Then we are going to create an index.html in the /etc/ansible/roles/apache/files/ folder with the following code.

```
<head>
<title>Ansible Role Demo</title>
</head>
<body>
<h1>
Welcome to Ansible Roles
</h1>
<br/><br/><br/>
<p>
<img src="picture.jpg" width="500" height="500"/>
</p>
</body>
</html>
```

Notice there is an image file in the HTML. We are going to download this image from S3 public hosted bucket file and save it in the /etc/ansible/roles/apache/files/ folder.

Now let's go back to the /etc/ansible/roles/apache/tasks folder and create configure.yml with the following code:

```
---
# Configuring apache2
- name: apache2 configuration file
  copy: src=apache2.conf dest=/etc/apache2/apache2.conf
  notify: restart apache service

- name: create the webpage index.html
  copy: src=index.html dest=/var/www/html/index.html
```

```
- name: copy the image resource
  copy: src=picture.jpg dest=/var/www/html/picture.jpg
```

The above code is coping the resources we saved in the files folder to our target server. We are using the configure.yml to set up our Apache configurations.

Notice the “notify” command. This requires a handler. So we go into the /etc/ansible/roles/apache/handlers/main.yml and enter the following code:

```
---
# resarting server
- name: restart apache service
  service: name=apache2 state=restarted
```

This code is going to restart the Apache server.

Service.yml

Again go back to the /etc/ansible/roles/apache/tasks/ folder create the service.yml file with the following code:

```
---
# tasks file for apache
- name: start apache2 server
  service: name=apache2 state=started
```

This will start the Apache server. We are done with defining the apache role. Our apache folder inside /etc/ansible/roles should look like this now:

```
apache/
|-- README.md
|-- defaults
|   `-- main.yml
|-- files
|   |-- picture.jpg
|   |-- apache2.conf
|   `-- index.html
|-- handlers
|   `-- main.yml
|-- meta
|   `-- main.yml
|-- tasks
|   |-- configure.yml
|   |-- install.yml
|   |-- main.yml
|   `-- service.yml
```

```
|-- templates
|-- tests
|   |-- inventory
|   `-- test.yml
`-- vars
```

Using the Apache role with site.yml

Now in the folder /etc/ansible define the following site.yml:

```
---
- hosts: ubuntu
  become: true
  roles:
    - apache
```

Remember we defined ubuntu inside /etc/ansible/hosts file as

```
[ubuntu]
ubuntu ansible_host=172.31.16.233
ansible_ssh_private_key_file=/root/key.pem ansible_user=ubuntu
```

We can check if our YAML files are well formatted using the following command:

```
# ansible-playbook site.yml --syntax-check
playbook: site.yml
```

Instead of “playbook: site.yml”, you should see warnings if there are any problems.

Now run the following command:

```
# ansible-playbook --become site.yml
```