ANSIBLE HANDS-ON

Ansible hands-on documentation has been divided into 3 segments.

A. Creating Ansible Playbook
B. Creating Ansible Roles
C. Using Ansible Roles in Playbook

Prerequisites:

1. Ansible needs to be installed in master.
2. Connection between Master and Host needs to be set through ssh. For more information refer to the Ansible Installation Documentation.

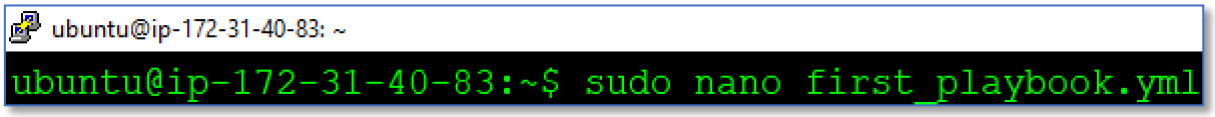
A- Creating Ansible Playbook

This playbook consists of two plays with following tasks:

- Play 1: Execute a command in host1, Execute a script in host1
- Play 2: Execute a script in host2, Install nginx in host2

Step 1: Create the .yml file.

```
sudo nano <playbookname>
```



Step 2: Add the following content in the .yml file.

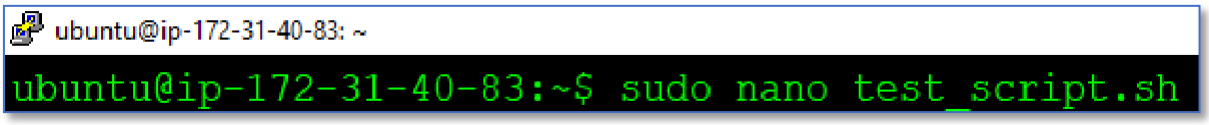
```
---
-hosts: host1 sudo: yes name: Play 1 tasks:
-name: Execute command 'Date'
-command: date
-name: Execute script on server script: test_script.sh
-hosts: host2 name: Play 2 sudo: yes tasks:
-name: Execute script on server script: test_script.sh
-name: Install nginx
-apt: name=nginx state=latest
```

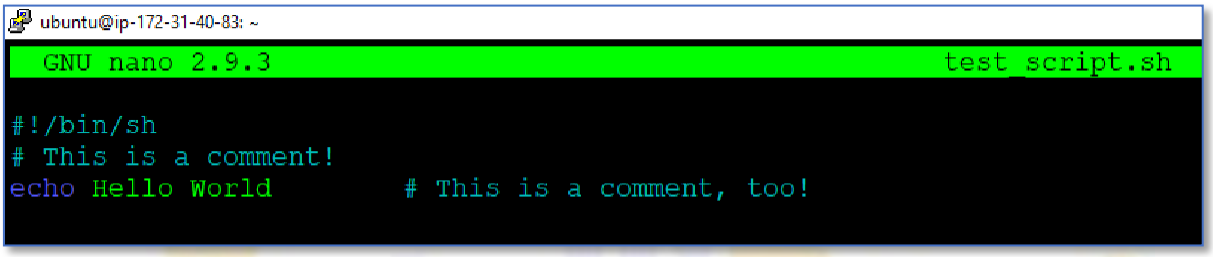
```
---
 - hosts: host1
   become: true
   name: play1
   tasks:
     - name: Execute command 'Date'
       command: date
     - name: execute script on server
       script: test_script.sh


 - hosts: host2
   become: true
   name: play2
   tasks:
     - name: execute script on server
       script: test_script.sh
     - name: ensure nginx is at the latest version
       apt: name=nginx update_cache=yes state=latest
```

Step 3: Now to be able to perform "Execute script on server" task we need to have the **.sh** file (unix/linux *shell* executables *files*) in master machine. Create test.sh file as shown.

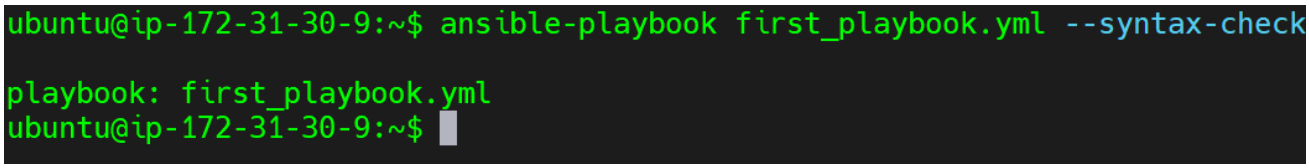
```
sudo nano <file_name>
```





Step 4: Before executing the playbook that we just created we need to have to check for syntax errors.

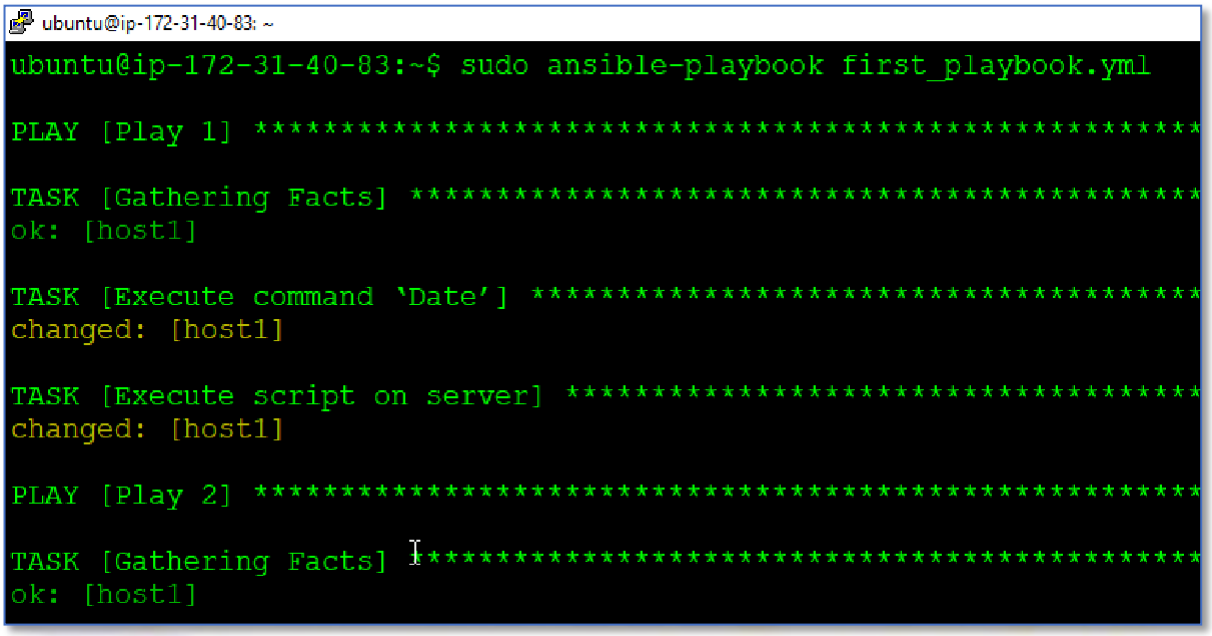
```
ansible-playbook <playbook> --syntax-check
```



This means our playbook is syntax error free. Let us move ahead and execute the playbook.

Step 5: To execute the playbook use the following command.

```
sudo ansible-playbook <playbook>
```



Great! We have successfully created our very first Ansible playbook.

Remember that using playbook we can run the same command repeatedly, but if everything was configured on the first run, then all subsequent runs make no changes.

B- Creating Ansible Roles

Step 1: Ansible roles should be written inside "*/etc/ansible/roles/*". Use the following command to create one Ansible role.

```
sudo ansible-galaxy init <role name>
```

```
ubuntu@ip-172-31-30-9:/etc/ansible/roles$ sudo ansible-galaxy init apache
- Role apache was created successfully
ubuntu@ip-172-31-30-9:/etc/ansible/roles$
```

Step 2: Install tree package using sudo apt install tree. Use tree command to view structure of the role

```
sudo apt install tree
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles
ubuntu@ip-172-31-40-83:/etc/ansible/roles$ sudo apt install tree
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 154 not upgraded.
```

Now let us see the structure of the role that we just created using the following command.

```
tree <role name>
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles
ubuntu@ip-172-31-40-83:/etc/ansible/roles$ tree apache
apache
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Now we are ready to create the tasks that our roles are supposed to perform.

Step 3: Go inside *task* folder inside *apache* directory. Edit *main.yml* using the following command. Make changes as shown. Save and then exit.

```
sudo nano main.yml
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/tasks$ sudo nano main.yml
```

Now we will divide the tasks to be performed into three categories. Install, configure and services. We will create three different .yml files to reduce the complexity. Include those separate task files in the main.yml file as shown.

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

Remember that order of the list in yml file matters. So here install.yml gets executed first, then configure.yml and then service.yml.

Step 4: Now inside *task* folder, create *install.yml* and add the installation tasks to be performed as shown below.

We will install the latest version of apache2 in the remote machine with the help of apt module as shown below.

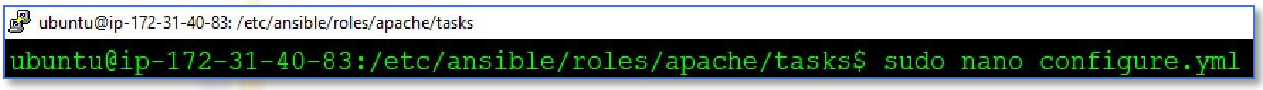
```
sudo nano install.yml
```

```
---
  - name: install apache2
    apt: name=apache2 update_cache=yes state=latest
    become: true
```

```
 GNU nano 4.8                                                install.yml
---
  - name: install apache2
    apt: name=apache2 update_cache=yes state=latest
    become: true
```

Step 5: Then create *configure.yml* and add the required configurations that need to be performed on remote machine as shown below.

```
sudo nano configure.yml
```




```
---
#configure apache2.conf and send copy.html file
  - name: apache2.conf file
    copy: src=apache2.conf dest=/etc/apache2/
    become: true
    notify:
      - restart apache2 service

  - name: send copy.html file
    copy: src=copy.html dest=/home/ubuntu/
    become: true
```

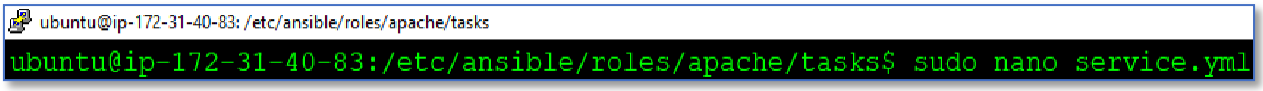
We will configure apache2.conf file in the remote machine and also, we will restart the apache2 service. Then we will send one file from */etc/ansible/roles/apache/files* folder to the remote machine. The destination path has been set to /home/ubuntu/ as shown.

```
 GNU nano 4.8                                                configure.yml
---
#configure apache2.conf and send copy.html file
  - name: apache2.conf file
    copy: src=apache2.conf dest=/etc/apache2/
    become: true
    notify:
      - restart apache2 service

  - name: send copy.html file
    copy: src=copy.html dest=/home/ubuntu/
    become: true
```

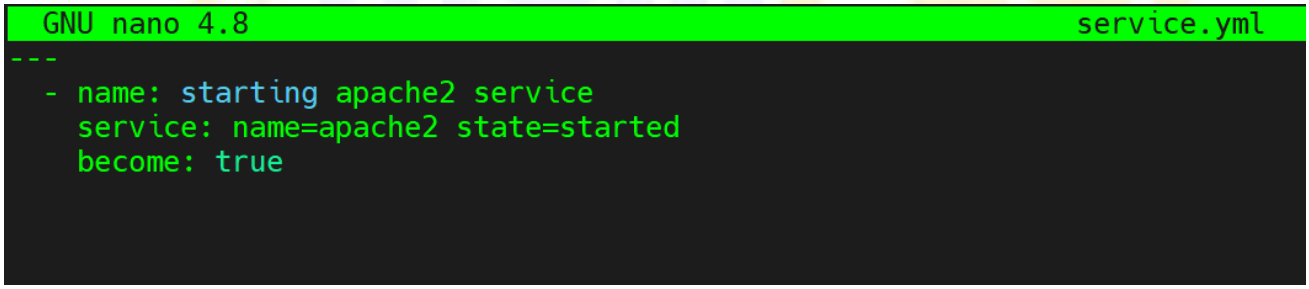
Step 6: Again, inside *task* folder, create *service.yml* and add the required configurations that need to be performed on remote machine as shown below.

```
sudo nano service.yml
```



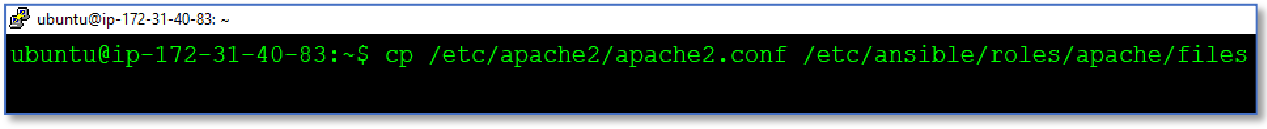
We will configure apache2.conf file in the remote machine

```
---
  - name: starting apache2 service
    service: name=apache2 state=started
    become: true
```



Step 7: Now go inside files. Store the files that needs to be pushed to the remote machine. Copy the apache2.conf file from */etc/apache2* directory to */etc/ansible/roles/apache/files* and create the html file.

```
cp /etc/apache2/apache2.conf /etc/ansible/roles/apache/files
```



Create one html file as well. My dummy html file looks like this.

```
<html>
 <title> Some File </title>
 <body> <h1> Copy This File> </h1>
</body>
</html>
```

ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/files

```
  GNU nano 2.9.3                                              copy.html

<html>
  <title> Some File </title>
  <body> <h1> Copy This File> </h1> </body>
</html>
```

Check whether our files are ready or not by using the following command.

```
ls
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/files
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/files$ ls
apache2.conf  copy.html
```

Step 8: Go inside handlers and add the action that needs to be performed after *notify* from *configure.yml* is executed. Use the following two commands.

```
cd /etc/ansible/roles/apache/handlers/

sudo nano main.yml
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/handlers
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/handlers$ sudo nano main.yml
```

Add the following content inside handlers file.

```
---
#handlers file for apache
  - name: restart apache2 service
    service: name=apache2 state=restarted
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/handlers
  GNU nano 2.9.3                                                main.yml

---
# handlers file for apache
  - name: restart apache2 service
    service: name=apache2 state=restarted
```

Remember that notify name and handler name should match.

```
  GNU nano 4.8                                          configure.yml
---
#configure apache2.conf and send copy.html file
  - name: apache2.conf file
    copy: src=apache2.conf dest=/etc/apache2/
    become: true
    notify:
      - restart apache2 service

  - name: send copy.html file
    copy: src=copy.html dest=/home/ubuntu/
    become: true
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/handlers
  GNU nano 2.9.3                                          main.yml

---
# handlers file for apache
  - name: restart apache2 service
    service: name=apache2 state=restarted
```

Step 9: Go inside meta and add information related to the role.

```
cd /etc/ansible/roles/apache/handlers/
sudo nano main.yml
```

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/meta
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/meta$ sudo nano main.yml
```

Add author information, role descriptions, company information etc. as shown below.

```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/meta
  GNU nano 2.9.3                                          main.yml

galaxy_info:
  author: Intellipaat
  description: Simple apache role
  company: Intellipaat

  # If the issue tracker for your role is not on github, uncomment the
  # next line and provide a value
  # issue tracker url: http://example.com/issue/tracker
```

Step 10: Go to the */etc/ansible/* and create one top level *.yml* file where we can add hosts and roles to be executed. Execute apache role on the hosts that is under the group name servers, added in the inventory file */etc/ansible/hosts*

```
cd /etc/ansible/
sudo nano site.yml
```

For more than one hosts following commands can be used.

```
---
 - hosts: host1
   roles:
     - apache
```

```
  GNU nano 4.8                                                site.yml
---
  - hosts: host1
    roles:
      - apache
```

Step 11: Before we execute our top level yml file we will check for syntax errors put our configuration in there as shown below.

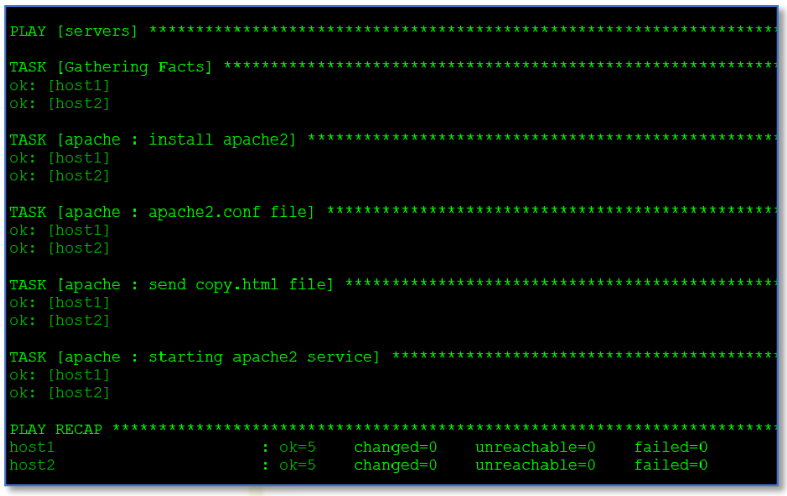
```
ansible-playbook <filename.yml> --syntax- check
```

Step 12: Execute the top level *.yml* file

```
ansible-playbook <filename.yml>
```

```
ubuntu@ip-172-31-40-83: /etc/ansible
ubuntu@ip-172-31-40-83:/etc/ansible$ ansible-playbook site.yml
```

The output looks like this.



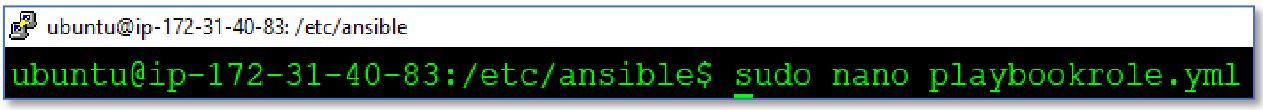
Congratulations! You have successfully created Ansible Role.

Now let us see how to use this Ansible role that we've just created along with other tasks in a Ansible Playbook.

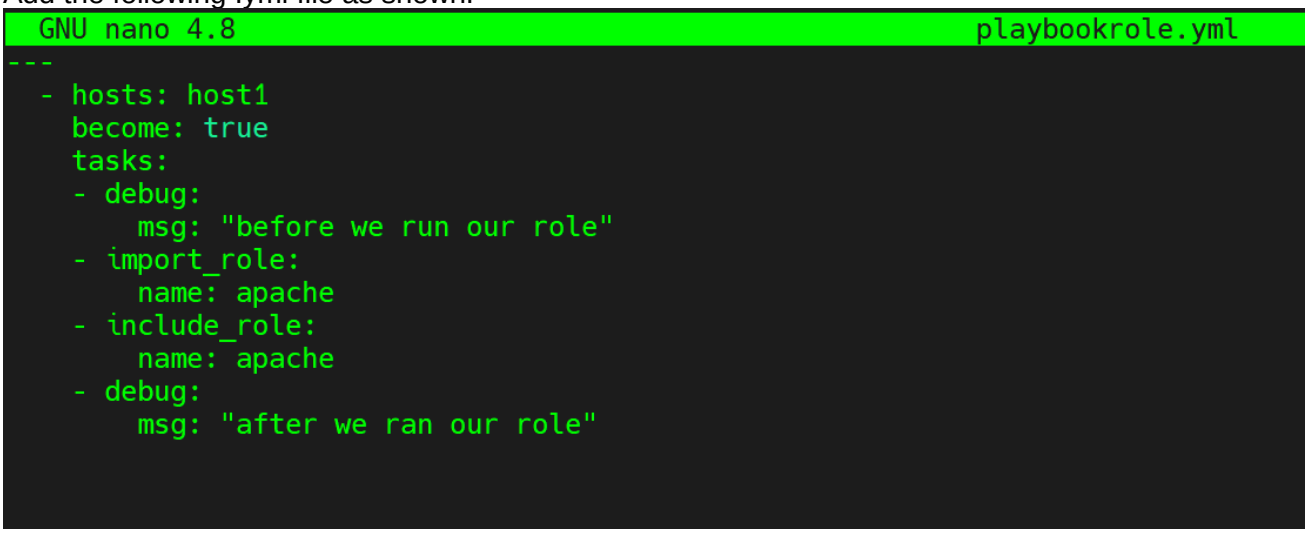
C- Using Ansible Roles in Playbook

Step 1: To use ansible roles along with other tasks in playbook use *import_role* and *include_role.* Create one playbook called to execute on the remote machines along with two *debug* tasks before and after *apache role*.

```
sudo nano <playbook name>
```



Add the following .yml file as shown.



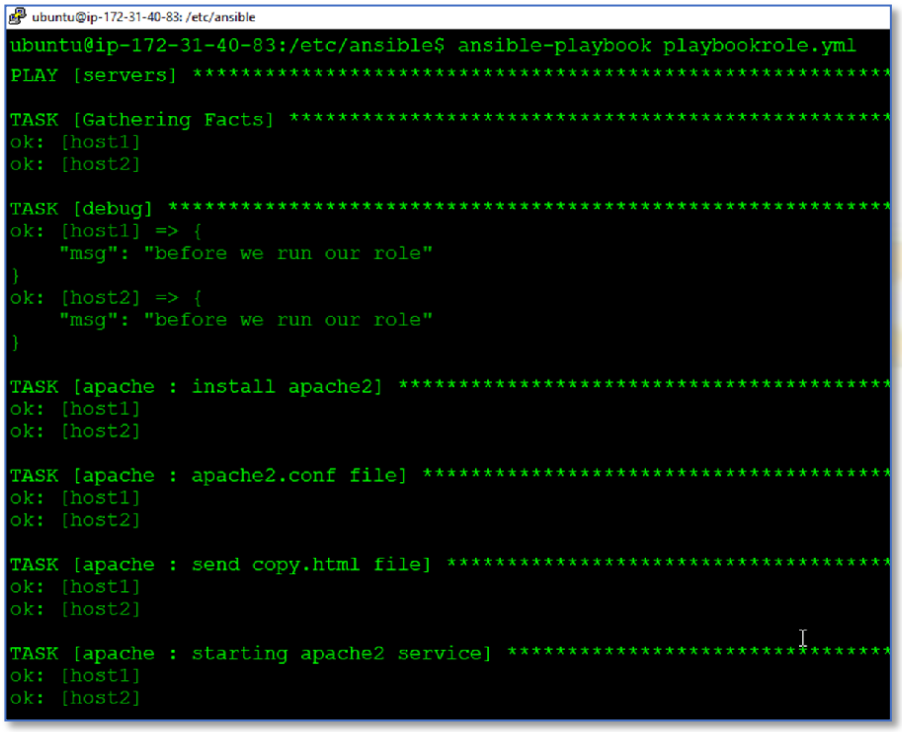
```
---
  - hosts: host1
    become: true
    tasks:
    - debug:
        msg: "before we run our role"
    - import_role:
        name: apache
    - include_role:
        name: apache
    - debug:
        msg: "after we ran our role"
```

Step 2: Check for syntax error and execute the playbook with roles.

```
ansible-playbook <playbookname> --syntax-check
```

Step 3: Check for syntax error and execute the playbook with roles.

```
ansible-playbook <playbookname>
```



Congratulations! You have successfully integrated Ansible roles with Ansible playbook.