**Now We are going to discuss about ROLES :**

Ansible Roles :

Roles in ansible build on the idea of including files and combining them to form clean, reusable abstractions – they allow you to focus more on the big picture and only dive down into the details when needed.

We will start with understanding of includes so that roles make more sense, but our ultimate goal should be understanding roles – roles are great and you should use them every time you write playbooks.

Now we were discussed playbooks and modules, in playbooks if you have more tasks to write the playbook filesize wil get increased in size wise and length wise that is why we are going to use roles.

Till now we were created palybook files and playbook modules under Playbooks folder

cd Playbooks

Using Roles in Playbooks:

Lecture: Roles – The Directory structure and Tempaltes

As we talk about and use roles, lets demonstrate the major areas and functions one at a time.

Now we are going to create one more folder called Roles and in that we are going create roles there.

mkdir Roles

Now inside the Roles folder am going to create few directories according to my hosts file setup.

cat /etc/ansible/hosts

[test@ip-172-31-1-169 ~]$ cat /etc/ansible/hosts

[local]

localhost

[redhat]

172.31.1.209

172.31.13.71

[appserver]

172.31.10.81

[ubuntu]

172.31.31.163

[test@ip-172-31-1-169 ~]$

now am going to create above directories

cd Roles

cd roles

mkdir redhat

mkdir appserver

mkdir ubuntu

[test@ip-172-31-1-169 roles]$ ls -lrt

total 0

drwxrwxr-x. 2 test test 6 Jun 17 21:54 redhat

drwxrwxr-x. 2 test test 6 Jun 17 21:54 appserver

drwxrwxr-x. 2 test test 6 Jun 17 21:54 ubuntu

[test@ip-172-31-1-169 roles]$

I want to see the tree structure of my Roles folder.

cd

[test@ip-172-31-1-169 ~]$ tree Roles

Roles

├── appserver

├── redhat

└── ubuntu

3 directories, 0 files

[test@ip-172-31-1-169 ~]$

If you are not able to see above you should install tree first using below command

sudo yum install tree

Now am going to inside appserver folder

cd /Roles/appserver

[test@ip-172-31-1-169 ~]$ cd Roles

[test@ip-172-31-1-169 roles]$ ls -lrt

total 0

drwxrwxr-x. 2 test test 6 Jun 17 21:54 redhat

drwxrwxr-x. 2 test test 6 Jun 17 21:54 appserver

drwxrwxr-x. 2 test test 6 Jun 17 21:54 ubuntu

[test@ip-172-31-1-169 Roles]$ cd appserver

[test@ip-172-31-1-169 appserver]$ mkdir files

[test@ip-172-31-1-169 appserver]$ mkdir templates

[test@ip-172-31-1-169 appserver]$ mkdir handlers

[test@ip-172-31-1-169 appserver]$ mkdir defaults

[test@ip-172-31-1-169 appserver]$ mkdir meta

[test@ip-172-31-1-169 appserver]$ mkdir tasks

[test@ip-172-31-1-169 appserver]$ mkdir vars

[test@ip-172-31-1-169 appserver]$ cd ..

[test@ip-172-31-1-169 Roles]$ cd ..

[test@ip-172-31-1-169 ~]$ tree Roles

Roles

roles

├── appserver

│   ├── defaults

│   ├── files

│   ├── handlers

│   ├── meta

│   ├── tasks

│   ├── templates

│   └── vars

├── redhat

└── ubuntu

10 directories, 0 files

[test@ip-172-31-1-169 ~]$

Same thing we have to repeat for redhat and ubuntu directories .

Now you can see like below

[test@ip-172-31-1-169 ~]$ tree Roles

Roles

└── roles

├── appserver

│   ├── defaults

│   ├── files

│   ├── handlers

│   ├── meta

│   ├── tasks

│   ├── templates

│   └── vars

├── redhat

│   ├── defaults

│   ├── files

│   ├── handlers

│   ├── meta

│   ├── tasks

│   ├── templates

│   └── vars

└── ubuntu

├── defaults

├── files

├── handlers

├── meta

├── tasks

├── templates

└── vars

25 directories, 0 files

[test@ip-172-31-1-169 ~]$

The hosts groups are keep on adding here. And here there are some directory structure they are :

deafaults : Where wil have default mechanisms and the files which are going to run bydefault at boot time.

Files : Files directory where we will keep all the files

Handlers : Which we will keep all the handlers or events to handle

meta : it wil keep the meta data

tasks : it wil have our regullar tasks such as, we were writing commands such as called command module ping module yum module shell module so every taks will be written in task directory

templates : Like apache templates

vars : static variables and we can redirect all the variables in this directory.

Remember these 7 directories wil be available incase of creating ansible roles and should be there as and our ansible infrastructure increases.

So why we are going with these 7 directories and what is the benefit of these 7 directories in ansible roles we will discuss about these things one by one.

So before that am going to check my ansible is claen and working fine or not

cd

tree Roles

ansible all -m ping

**Using Roles in Playbooks:**

* Role based tasks
* Task Order – Pre and Post Tasks
* Roles – Conditional Execution
* Roles – Variable Substitution
* Roles – Handlers

**Role based tasks**

Now we are going create a task under appserver directory.

[test@ip-172-31-1-169 ~]$ cd Roles

[test@ip-172-31-1-169 Roles]$ tree appserver

[test@ip-172-31-1-169 Roles]$ cd roles

[test@ip-172-31-1-169 roles]$ cd appserver

[test@ip-172-31-1-169 appserver]$ ls -lrt

total 0

drwxrwxr-x. 2 test test 6 Jun 17 22:07 files

drwxrwxr-x. 2 test test 6 Jun 17 22:07 templates

drwxrwxr-x. 2 test test 6 Jun 17 22:08 handlers

drwxrwxr-x. 2 test test 6 Jun 17 22:08 defaults

drwxrwxr-x. 2 test test 6 Jun 17 22:08 meta

drwxrwxr-x. 2 test test 6 Jun 17 22:08 tasks

drwxrwxr-x. 2 test test 6 Jun 17 22:08 vars

[test@ip-172-31-1-169 appserver]$ cd tasks

[test@ip-172-31-1-169 tasks]$ ls -lrt

total 0

[test@ip-172-31-1-169 tasks]$

Under tasks directory we are going to create one yml file thats called main.yml

vim main.yml

- name: Install apache web services

yum: name=httpd state=installed

notify: RESTART HTTPD

under handlers directory we are going to create one yml file thats calles main.yml

vim main.yml

- name: RESTART HTTPD

service: name=httpd state=started

under vars directory we are going to create one yml file thats called main.yml

vim main.yml

copyright: apache installation

Now we are going to roles folder and creating main yml file called appserver.yml file

vim appserver.yml

---

- hosts: appserver

user: test

sudo: yes

gather\_facts: yes

connection: ssh

roles:

- appserver

:wq

Now we have create main playbook yml file. So that we are able to run playbook now

ansible-playbook appserver.yml – - check

ansible-playbook appserver.yml

[test@ip-172-31-1-169 roles]$ ansible-playbook appserver.yml --check

PLAY [appserver] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [172.31.10.81]

TASK [appserver : Install apache web services] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [172.31.10.81]

RUNNING HANDLER [appserver : RESTART HTTPD] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [172.31.10.81]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

172.31.10.81 : ok=3 changed=2 unreachable=0 failed=0

[test@ip-172-31-1-169 roles]$ ansible-playbook appserver.yml

PLAY [appserver] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [172.31.10.81]

TASK [appserver : Install apache web services] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [172.31.10.81]

RUNNING HANDLER [appserver : RESTART HTTPD] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [172.31.10.81]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

172.31.10.81 : ok=3 changed=2 unreachable=0 failed=0

Now we can go to our all nodes and we can check services started or not

sudo systemctl status httpd

[test@ip-172-31-10-81 ~]$ sudo systemctl status httpd

● httpd.service - The Apache HTTP Server

Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)

Active: active (running) since Sat 2017-06-17 23:17:58 EDT; 12min ago

Docs: man:httpd(8)

man:apachectl(8)

Main PID: 7902 (httpd)

Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"

CGroup: /system.slice/httpd.service

├─7902 /usr/sbin/httpd -DFOREGROUND

├─7903 /usr/sbin/httpd -DFOREGROUND

├─7904 /usr/sbin/httpd -DFOREGROUND

├─7905 /usr/sbin/httpd -DFOREGROUND

├─7906 /usr/sbin/httpd -DFOREGROUND

└─7907 /usr/sbin/httpd -DFOREGROUND

Jun 17 23:17:58 ip-172-31-10-81.ap-south-1.compute.internal systemd[1]: Starting The Apache HTTP Server...

Jun 17 23:17:58 ip-172-31-10-81.ap-south-1.compute.internal systemd[1]: Started The Apache HTTP Server.

[test@ip-172-31-10-81 ~]$

If you want to remove httpd on node and you can re run palybook file state=started or state=restarted in handlers section.

Sudo yum remove httpd --- on node system

again re run same playbook file in master system.

ansible-playbook appserver.yml

Now we have done role based tasks and now we are moving forward.

[test@ip-172-31-1-169 ~]$ cd Roles

[test@ip-172-31-1-169 Roles]$ tree roles

roles

├── appserver

│   ├── defaults

│   ├── files

│   ├── handlers

│   │   └── main.yml

│   ├── meta

│   ├── tasks

│   │   └── main.yml

│   ├── templates

│   └── vars

│   └── main.yml

├── appserver.yml

├── redhat

│   ├── defaults

│   ├── files

│   ├── handlers

│   ├── meta

│   ├── tasks

│   ├── templates

│   └── vars

└── ubuntu

├── defaults

├── files

├── handlers

├── meta

├── tasks

├── templates

└── vars

24 directories, 4 files

[test@ip-172-31-1-169 Roles]$

this is our tree structure and main playbook files will be outside of roles.

**Task Order – Pre and Post Tasks :**

Now we are going to do pre taks post taks examples

[test@ip-172-31-1-169 ~]$ cd Roles

[test@ip-172-31-1-169 Roles]$ cd roles

[test@ip-172-31-1-169 roles]$ vim appserver.yml

---

- hosts: appserver

user: test

sudo: yes

gather\_facts: yes

connection: ssh

pre\_tasks:

- name: how long the role taks took to execute

raw: date > /home/test/start.log

roles:

- appserver

post\_tasks:

- name: when did the role task completed

raw: date > /home/test/end.log

:wq

ansible-playbook appserver.yml –check

ansible-playbook appserver.yml

after this execution I can go and verify on all nodes in /home/test/ location for start.log and end.log files.

[test@ip-172-31-10-81 ~]$ ls -lrt

total 76

-

-rw-r--r--. 1 root root 29 Jun 18 00:20 start.log

-rw-r--r--. 1 root root 29 Jun 18 00:20 end.log

[test@ip-172-31-10-81 ~]$

[test@ip-172-31-10-81 ~]$ cat start.log

Sun Jun 18 00:20:45 EDT 2017

[test@ip-172-31-10-81 ~]$ cat end.log

Sun Jun 18 00:20:46 EDT 2017

[test@ip-172-31-10-81 ~]$

**Roles – Conditional Execution :**

Now we are going to discuss on Conditional execution with roles. This example am going to uninstall httpd on all mentioned notes so that this will be good for this example. So now am going to remove httpd on all nodes which are mentioned in playbook.

Go to all nodes

sudo yum remove httpd

if you want to clean it just type below command

sudo yum clean all

on ubuntu node :

sudo apt-get remove apache2

Now come to master server.

cd Roles

cd roles

cd appserver

cd tasks

vim main.yml

- name: Install apache web server -redhat

yum: name=httpd state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_errors: yes

- name: Install apache web server -ubuntu

apt: name=httpd state=latest

when: “ansible\_os\_family == 'Debian'”

ignore\_errors: yes

:wq

for this in appserver.yml playbook we have to set hosts as all then we can check this functionality in our environment.

If you want to what is ansible\_os\_family, it is picking system information like from set up module, if you want to check it using setup module you can do it

ansible all -s -m setup -a “filter=ansible\_os\_family”

you will get all nodes details about operating system.

Now we are going to run our main playbook

cd roles

ansible-playbook appserver.yml

Here we can identify the skipping tasks when it is running on ubuntu tasks all redhat tasks can be skipped when it is running on redhat tasks all ubuntu tasks can be skipped that we can identify in output of the playbook.

**Roles – Variable Substitution :**

now we are going to work with variable substitution. For the same we have to go to appserver role and in that task section need to open main.yml file

cd Roles

cd roles

cd appserver

cd tasks

Am going to take this main.yml as backup file

cp -pr main.yml main.yml\_backup

why because evrytime you are going run master playbook it is going to run only main.yml file

vim main.yml

- name: Install apache web server -redhat

yum: name={{ redhat\_apache }} state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_errors: yes

- name: Install apache web server -ubuntu

apt: name={{ debian\_apache }} state=installed

when: “ansible\_os\_family == 'Debian'”

ignore\_errors: yes

:wq

now I have defined some variables in tasks section main.yml file so those for variables we have to assign some values in variables section main.yml file

cd ..

cd vars

vim main.yml

copyright: apache license -----this is already there not required now but no problem here

redhat\_apache: httpd

debian\_apache: apache2

:wq!

Now we have to execute the main job that is called appserver.yml playbook.

cd ..

cd ..

now you are in roles directory you can run now.

Here everything we are working under folder and under in role this can be like project.

Before this palybook execution if you want to remove apache from all nodes again you can remove just folloe the above removal steps for apache using yum and apt-get modules.

ansible-playbook appserver.yml

**Now we are going to discuss on Handlers Sections :**

again we have to go to first tasks section and main.yml file

cd appserver

cd tasks

again am going to take backup if something goes wrong I can make revert working changes to back.

cp -pr main.yml main.yml\_Variables\_taks

vim main.yml

- name: Install apache web server -redhat

yum: name={{ redhat\_apache }} state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_erros: yes

notify: RESTARTHTTPD

- name: Install apache web server -ubuntu

apt: name={{ debian\_apache }} state=installed

when: “ansible\_os\_family == 'Debian'”

ignore\_erros: yes

notify: RESTARTAPACHE2

:wq

cd ..

cd handlers

here am taking backup for main.yml file

cp -pr main.yml main.yml\_backup

vim main.yml

- name: RESTARTHTTPD

service: name=httpd state=restarted

- name: RESTARTAPACHE2

service: name=apache2 state=restarted

:wq

rememeber earlier we were working only one os with playbooks and modules now we are able to work with mutli os using roles.

cd ..

cd ..

am in roles directory now

if you want to remove again apache in all nodes you just remove for clean installation

ansible-playbook appserver.yml

after this output if you want to check in all nodes go to each node and type below command to check

systemctl status httpd ---> for red hat nodes

service apache2 status ---> for ubuntu node

so everythins is working fine as expected. Now am going to remove from all nodes for next examples.

Since we are discussing handlers section am going to automate little bit extra thing now using declared variables. No need to pass again httpd and apache2 in handlers section why because we have already declared in vars section main.yml file.

go to handlers section main.yml file

cd handlers

vim main.yml

- name: RESTARTHTTPD

service: name={{ redhat\_apache }} state=restarted

- name: RESTARTAPACHE2

service: name={{ debian\_apache }} state=restarted

:wq

now again am going to roles directory and am going to run the playbook again same like before it should be executed.

Till now we have seen Role based task task order – pre and post tasks and conditional execution tasks and variable substitution and finally Handlers and one more how to pass variables into handlers section.

Now we are going to discuss on Waiting For Events in roles section :

It means, as a name indicates the roles are going to wait until some event is been completed as we did it in module section where we install apache server but we just gave a module called wait for that shows a some other event should get complete before this task should get complete to continue furthur.

So what we are going to do for waiting for sections in roles, so lets start with same concept what we did now,

cd Roles

cd roles

cd appserver

cd tasks

cp -pr main.yml main.yml\_handlers

vim main.yml

we are going to remove notify why because we are going to do manual start with help of waitfor module and we are going to take same example which was used in module section so that easy to understand.

- name: Install apache web server -redhat

yum: name={{ redhat\_apache }} state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_errors: yes

- name: wait for service listning to port 80

wait\_for:

port: 80

state: started

- name: Install the ftp package

yum: name=ftp state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_errors: yes

:wq

so basic tthing we have to remember here in the program we are waiting for module used to open port number 80 manually so the playbook really wait for the port should be opend by ourself manually if there are services already opend means this playbook is not going work this wil get stuck when you reach wait\_for module in playbook.

For this playbok execution better to stop httpd and apache2 on all nodes for understanding this concept clearly.

Go to each node and check for the httpd

systemctl status httpd ---> for red hat nodes

service apache2 status ---> for ubuntu node

Then if it active we have to remove for this example like below

sudo yum remove httpd -y ----> for red hat nodes

sudo apt-get remove apache2 ----> for ubuntu nodes

cd ..

cd ..

so we are in roles directory .

ansible-playbook appserver.yml

when the playbook waiting wait for module so that time we have to go each node hosts file and we have to start port 80 manually with below commands

sudo systemctl start httpd → Red hat nodes

sudo service apache2 start → Ubuntu node

for the first time we are going to start the httpd for Redhat manually so all nodes having port 80 is active now. If you re run the playbook again now it will not wait for the port number 80 which already started manullay when previous run.

So if you want to check wait for module again you should rmove first then you can test it.

ansible-playbook appserver.yml

Now we are going to discusss about Executing a task until module in roles :

so here we are going to execute a task something pops up, this lecture is all about executing a task until pop you know something like services comes up. We use to execute until some manual intervention happened like above example open the port.

But now the ansible infrastructure or ansible software is going to detect itself is something like this is going to happen some trigger is going to happen or something like error messages are going to come then it is going to perform task automatically. No need manual intervention.

So for that example have been defined in ansible called Executing a task until,

So there is difference between waiting for the events and Executing a task untiil modules, this task executing a task until is going to be detected by ansible but this task waiting for events is actually designed to do it manually.Okay

again same we have to go to tasks directory we have work as per this module in roles.

cd appserver

cd tasks

cp -pr main.yml main.yml\_WaitFor

vim main.yml

- name: Install apache web server -redhat

yum: name={{ redhat\_apache }} state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_erros: yes

- shell: systemctl status httpd

register: result

until: result.stdout.find(“active(running)”)!= -1

retries: 5

delay: 5

- debug: var=result

- name: Install the ftp package

yum: name=ftp state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_erros: yes

:wq

Above playbook,

shelll is a module it is detecting httpd is active or not.

register is a module it is going to register the result of above shell module.

Until is a module it is checking registered result with some condition to pass or fail.

Retries is going to retry the task for these many times

delay is the time for every try it is going to wait for these many seconds.

Debug is the module and it is diplaying the debug results in one by one when it is trying to checking.

Now go to all redhat nodes and check httpd is active or not if it is active it will continue without errors if it is not active it will try 5 times with 5 seconds time period inbetween try.

So do with 2 scenarios one httpd is on and another one httpd is off on redhat nodes for example.

sudo systemctl status httpd

sudo systemctl start httpd

sudo systemctl stop httpd

execute playbook first all are in active and try to stop httpd on all red hat nodes and try to execute when it is trying to checking in playbook if you are not happy with 5 times retry and 5 seconds delay you just put 10 retries and 10 seconds delay. That time you will get enough time to restart on all nodes so that you will not get any errors.

cd roles

ansible-playbook appserver.yml

here we are able to all retries because of debug module in playbook.

**Now we are going to discuss about passing a variable module in roles :**

just try to connect all nodes once to just check

ansible all -m ping

everything should be fine

just check what is the tree strtucure of our roles directory.

cd Roles

tree roles

we can the files which are created by us in tree structure.

So now again am going to roles directory and we are going to change main appserver.yml playbook. Till now we have done all changes into tasks directory but now in main playbook file.

cd roles

cp -pr appserver.yml dynamic.yml

Here am going to pass variable hosts nodes selection like group names we have to pass input when it is going to be running.

vim dynamic.yml

---

- hosts: '{{ hosts }}' ----------------- changed only this line

user: test

sudo: yes

gather\_facts: yes

connection: ssh

pre\_tasks:

- name: how long the role taks took to execute

raw: date > /home/test/start.log

roles:

- appserver

post\_tasks:

- name: when did the role task completed

raw: date > /home/test/end.log

:wq

Now am going to execute this dynamic.yml file with user input using below command,

same directory we have to be

cd roles

ansible-playbook dynamic.yml - -extra-vars “hosts=redhat”

2 nodes are going to run executing before task which was defined for Executing a task until.

ansible-playbook dynamic.yml - -extra-vars “hosts=appserver”

1 node is going to run executing before task which was defined for Executing a task until.

If you want to stop the httpd on appserver node and test the playbook once again it will retry the specified times with specified seconds delay like 5 ,5 or 10, 10.

**Now we are going to use TAGS in roles :**

Now am going o create tag for each task to identify the tasks when needs to be run on playbook,

so go to tasks directory under appserver role

cd appserver

cd tasks

cp -pr main.yml main.yml\_executinguntil

vim main.yml

- name: Install apache web server -redhat

yum: name={{ redhat\_apache }} state=installed

when: “ansible\_os\_family == 'RedHat'”

ignore\_erros: yes

notify: RESTARTHTTPD

tags: redhat --------- added only this line here

- name: Install apache web server -ubuntu

apt: name={{ debian\_apache }} state=installed

when: “ansible\_os\_family == 'Debian'”

ignore\_erros: yes

notify: RESTARTAPACHE2

tags: ubuntu --------- added only this line here

:wq

cd

cd Roles

cd roles

ansible-palybook - -tags=redhat appserver.yml

ansible-palybook - -tags=ubuntu appserver.yml

opposite way also we can pass tags like below

ansible-palybook - -skip-tags=ubuntu appserver.yml --- it is going to run on redhat tag

Now we have 2 tasks in our example tasks playbook but we have created 2 tags for 2 tasks and while running main appserver play we have passed only tag. So that our playbook is going run only one task and it is going to skip another task as per your input.

We are done with using tags now.

**We are going to discuss about Breaking a playbook into a Role :**

so far what ever we did that is breaking a playbook into a role.

If I just type cd

cd Roles

tree roles

you can see all the yml files so here each yml file connect with all created or available directories like conneting with handlers and connecting with vars and connecting takss with in a role. We are splitting this full playbook into a tree structure like above tree and we have one master yml file which is taking what to do what not to do. So this is what we have done already.

But if you go to Playbooks,

cd ..

cd Playbooks

ls -lrt

Here you can see all playbooks but there is not directory tree structure and it is a huge files here and we dont know how to maintain all these files and it is very tds.

But when you come to Roles section

cd

tree

see here the structure of tree directory

cd Playbooks

tree

Here everything in single file structure

When you come to roles we can how many tasks to want to add we can add and we can specify this role designed for dev environment and this role is for test environment finally this role is designed for production environment with assosiated tasks and assosiated handlers and associated vars.

Eventually if you go to google and tyoe github ansible openstack examples modules.

you see the modules here

<https://github.com/openstack-ansible/ansible-role-docker>

here you click on roles folder meta data tasks vars everything provided in a structured manner

So finally always better and great and easy to maintain and easy to understand to tasks in roles than playbooks or palybooks with modules.

So We are done with Roles concept now and we have covered alomost all topics in ansible if you need extra things go to ansible documentation and there you can find out all the things are available in ansible to undestanding of extra about roles.

Thanks Siva