Chapter 2 : ansible

==

Before devops as a system admin:

- Hard for single person to do configuration on large number of servers.
- If there are 500 servers then you have to hire higher staff to manage the system administration
- Almost all tasks were manually based

After devops (all work can managed by devops engg.) :

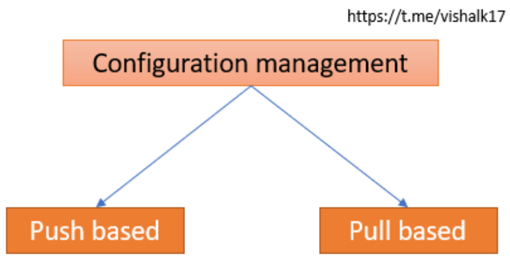
- Single server/person can handle number of servers
- Involves automation
- Infrastructure as a code
- Number of configuration management handle with the code

What is configuration management?

Configuration management is a process for maintaining computer systems, servers, and software in a desired, consistent state.

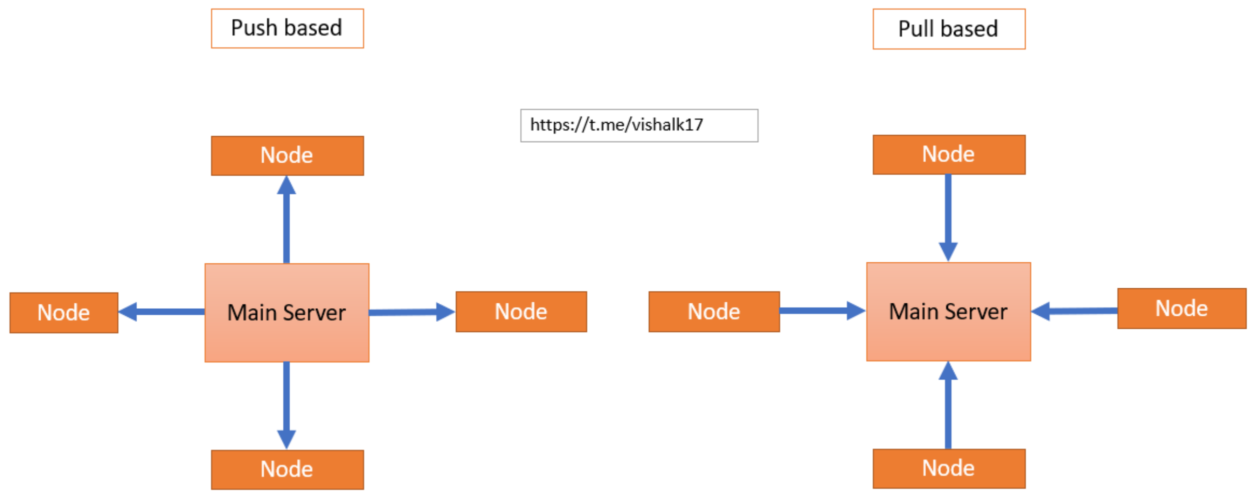
- Classify and manage systems by groups and subgroups.
- Centrally modify base configurations.
- Roll out new settings to all applicable systems.
- Automate system identification, patches, and updates
- Identify outdated, poor performing, and noncompliant configurations.
- Prioritize actions.
- Access and apply prescriptive remediation.

Types of configuration management?



Chapter 2 : ansible

==



Push based

- Centralized server pushes configs on the nodes.
- Server contacts nodes and send updates as needed.
- Since main server manages all things thus has overall control over nodes.
- Easier to use example ansible.

Pull based

- Well Suited but difficult to manage.
- Nodes runs an agent that ask central server if/when it has updates to run.
- Agent required to be install on nodes
- Example chef, puppet

Ansible:

Ansible is an open source IT Configuration Management, Deployment & Orchestration tool. It aims to provide large productivity gains to a wide variety of automation challenges. This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.

Why Do We Need Ansible?

Well before I tell you what is Ansible, it is of utmost importance to understand the problems that were faced before Ansible.

Chapter 2 : ansible

===

Let us take a little flashback to the beginning of networked computing when deploying and managing servers reliably and efficiently has been a challenge. Previously, system administrators managed servers by hand, installing software, changing configurations, and administering services on individual servers.
As data centers grew, and hosted applications became more complex, administrators realized they couldn't scale their manual systems management as fast as the applications they were enabling. It also hampered the velocity of the work of the developers since the development team was agile and releasing software frequently, but IT operations were spending more time configuring the systems. That's why server provisioning and configuration management tools came to flourish.

Consider the tedious routine of administering a server fleet. We always need to keep updating, pushing changes, copying files on them etc. These tasks make things very complicated and time consuming.

But let me tell you that there is a solution to the above stated problem. The solution is – ***Ansible.***

Advantages :

- Ansible is free for everyone
- Very light weight.
- Very secure
- It directly communicates with node using ssh.
- Agentless
- Push mechanism
- Doesn't need any special system admin with skills to install and use it
- Ansible use YAML (yet another markup language

Disadvantages:

- Ansible is new therefore less support and less doc. available
- Cannot achieve full automation

Chapter 2 : ansible

===

Terms in Ansible:

- Ansible server : Then machine where ansible is installed and from which all task and playbook will run
- Module : Basically, a module is a command or set of similar commands meant to be executed on the client side
- Task: A task is a section that consist of a single procedure to be completed.
- Role: A way of organizing tasks and related files to be later called in a playbook
- Inventory: File containg data about the ansible client servers
- Play : execution of playbook.
- Handler: Task which is called only if a notifier is present
- Notifier: Section attributed to a task which calls a handler if the output is changed.
- Playbooks: It consist code in YAML format, which describes task to be executed.
- Nodes : which are automated by Ansible.

Lab Practical:

Common things you have to do on ansible server and on its nodes:

Step 1 :

```
### add user with password start ###
```

Step 2:

```
### provide ansible as a user root privilege start ####
```

Step 3:

```
##                  edit /etc/ansible/ansible.cfg  start ###
#                   uncomment inventory line to enable hosts file
#                   uncomment sudo_user line
```

Step 4:

```
##                  edit /etc/ssh/sshd_config start ###
#
#                   uncomment PermitRootLogin yes
#                   uncomment PasswordAuthentication yes
```

Chapter 2 : ansible

==

```
#                    comment PasswordAuthentication no

#                    restart sshd service
```

I have made a script let me share with you : script link :
https://github.com/vishalk17/devops/blob/main/ansible/general_setup.sh

```
#!/bin/bash

# ****** allow further script only if root user executing the script start*******
if [[ $EUID == 0 ]]
then

        ## install require pkges ##
        amazon-linux-extras install ansible2 -y
        yum install git -y
        sleep 4

        ### add user with password start ###
        # -m : The user's home directory will be created if it does not exist.
        # -p EncryptedPasswordHere : The encrypted password, as returned by crypt().
        # username : Add this user to the Linux system,
        #
        #  useradd -m username
        #
        #  change/set password of ansible user
        #  echo "newuser:newpassword" | chpasswd
        #
        useradd -m ansible
        echo "ansible:ansible" | chpasswd
        #
        ### add user with password End ###

        ### provide ansible as a user root privilege start ####
        #
        echo 'ansible ALL=(ALL)       NOPASSWD: ALL' >> /etc/sudoers
        #
        ### provide ansible as a user root privilege end ####

        ##          edit /etc/ansible/ansible.cfg  start ###
        #           uncomment inventory line to enable hosts file
        #           uncomment sudo_user line
        #           replace lines by sed command
        #           sed -i "s%old_line%new_line%g" /path/to/the/file
        #
        sed -i "s%#inventory      = /etc/ansible/hosts%inventory      = /etc/ansible/hosts%g" /etc/ansible/ansible.cfg
        sed -i "s%#sudo_user      = root%sudo_user      = root%g" /etc/ansible/ansible.cfg
        #
        ##          edit /etc/ansible/ansible.cfg  End ###

        ##          edit /etc/ssh/sshd_config start ###
```

Chapter 2 : ansible

==

```
		#
		#		uncomment PermitRootLogin yes
		#		uncomment PasswordAuthentication yes
		#		comment PasswordAuthentication no
		#		restart sshd service
		#
		sed -i "s%#PermitRootLogin yes%PermitRootLogin yes%g" /etc/ssh/sshd_config
		sed -i "s%#PasswordAuthentication yes%PasswordAuthentication yes%g" /etc/ssh/sshd_config
		sed -i "s%PasswordAuthentication no%#PasswordAuthentication no%g" /etc/ssh/sshd_config

		service sshd restart
		#
		##		edit /etc/ssh/sshd_config end ###
		#
else
				echo " you are normal user, only root user can excute this script $0"
fi
#
# ****** allow further script only if root user executing the script End*******
```

Execute above script on all nodes and server. Make sure you are root user.

```
sudo su -
wget https://github.com/vishalk17/devops/raw/main/ansible/general_setup.sh
bash general_setup.sh
```

Further more you will required to add nodes in ansible server and setup ssh keys for nodes.

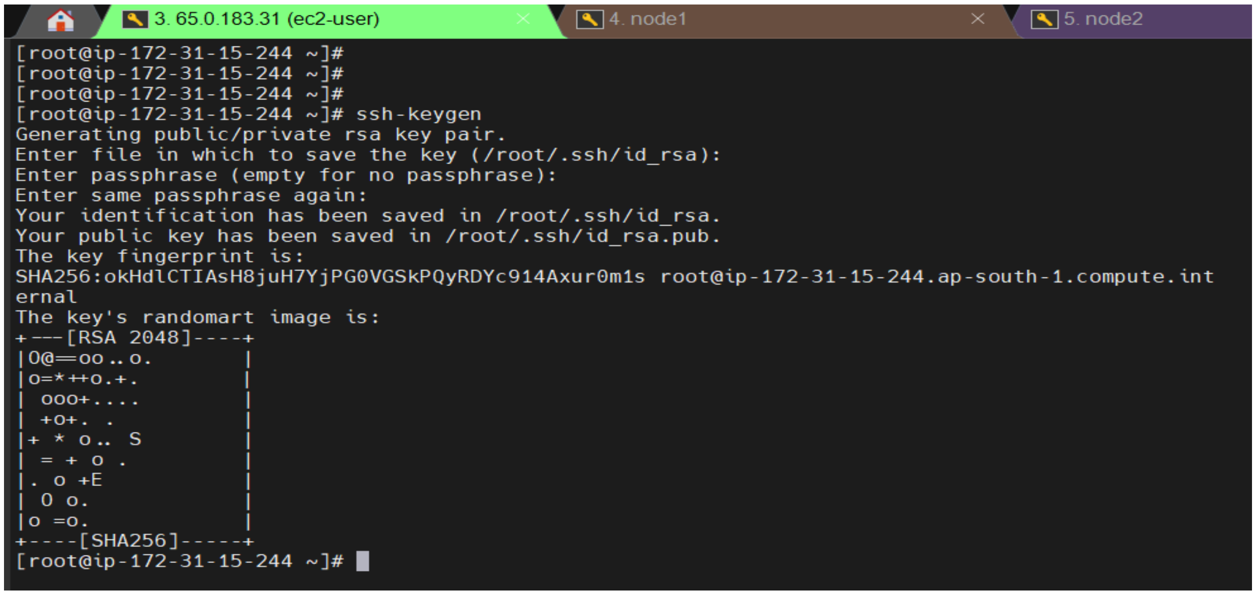
Step 5:

As Ansible works on the agentless architecture of using SSH to communicate with its nodes, set-up the ssh keys. Basically, we have one ansible server and multiple nodes. We control the nodes with our ansible server and hence we create a public ssh-key on the ansible server and copy it in the all node machines. Execute the following command on the ansible server:

```
ssh-keygen
```

Chapter 2 : ansible

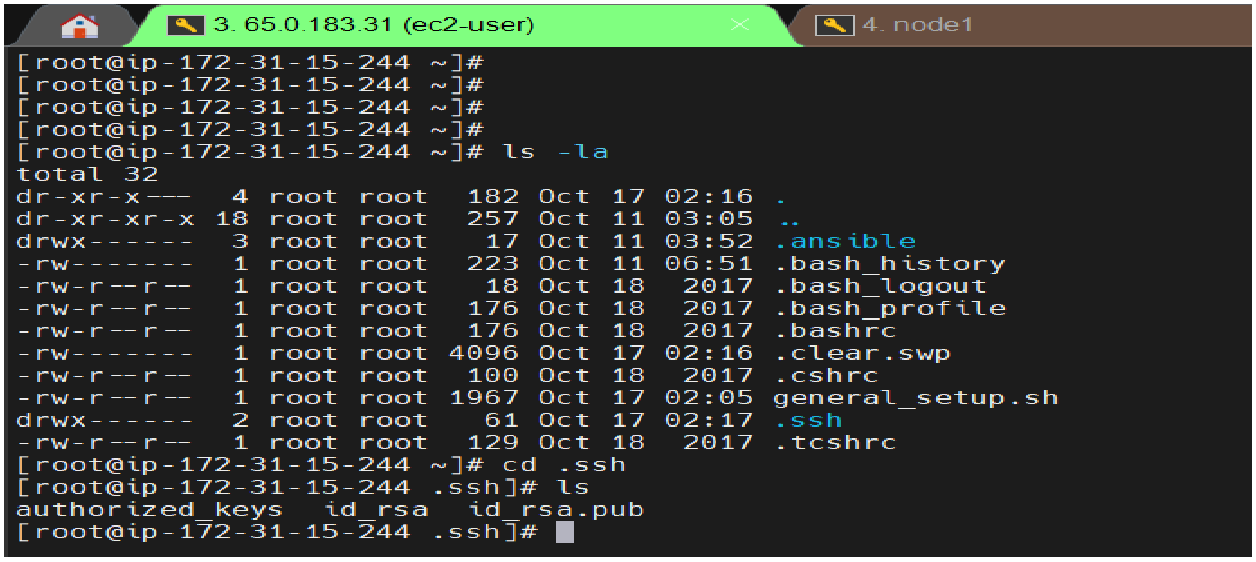
==



3. 65.0.183.31 (ec2-user)   4. node1   5. node2

```
[root@ip-172-31-15-244 ~]#
[root@ip-172-31-15-244 ~]#
[root@ip-172-31-15-244 ~]#
[root@ip-172-31-15-244 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:okHdlCTIAsH8juH7YjPG0VGSkPQyRDYc914Axur0m1s root@ip-172-31-15-244.ap-south-1.compute.int
ernal
The key's randomart image is:
+---[RSA 2048]----+
|O@==oo..o.       |
|o=*++o.+.        |
| ooo+....        |
| +o+. .          |
|+ * o.. S        |
| = + o .         |
|. o +E           |
| O o.            |
|o =o.            |
+----[SHA256]-----+
[root@ip-172-31-15-244 ~]#
```

You'll be prompted to enter the filename where you'd like to save your key and also prompt you for creating a password for accessing the generated key which is optional. By default, the public key gets saved in .ssh/id_rsa.pub file and the private key gets saved in .ssh/id_rsa.



3. 65.0.183.31 (ec2-user)   4. node1

```
[root@ip-172-31-15-244 ~]#
[root@ip-172-31-15-244 ~]#
[root@ip-172-31-15-244 ~]#
[root@ip-172-31-15-244 ~]#
[root@ip-172-31-15-244 ~]# ls -la
total 32
dr-xr-x---   4 root root  182 Oct 17 02:16 .
dr-xr-xr-x  18 root root  257 Oct 11 03:05 ..
drwx------   3 root root   17 Oct 11 03:52 .ansible
-rw-------   1 root root  223 Oct 11 06:51 .bash_history
-rw-r--r--   1 root root   18 Oct 18  2017 .bash_logout
-rw-r--r--   1 root root  176 Oct 18  2017 .bash_profile
-rw-r--r--   1 root root  176 Oct 18  2017 .bashrc
-rw-------   1 root root 4096 Oct 17 02:16 .clear.swp
-rw-r--r--   1 root root  100 Oct 18  2017 .cshrc
-rw-r--r--   1 root root 1967 Oct 17 02:05 general_setup.sh
drwx------   2 root root   61 Oct 17 02:17 .ssh
-rw-r--r--   1 root root  129 Oct 18  2017 .tcshrc
[root@ip-172-31-15-244 ~]# cd .ssh
[root@ip-172-31-15-244 .ssh]# ls
authorized_keys  id_rsa  id_rsa.pub
[root@ip-172-31-15-244 .ssh]#
```

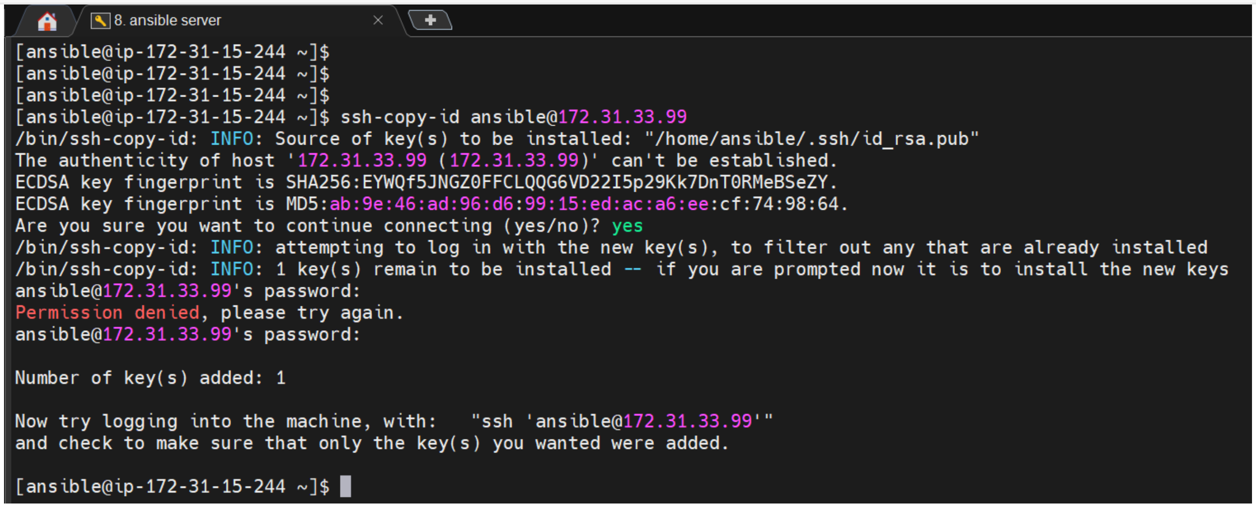
Step 6:

After the key is generated, the next task is to copy public key of Ansible server to its nodes. Use the command below:

Chapter 2 : ansible

==

```
ssh-copy-id user-name@<ip address of your node machine>

example

ssh-copy-id ansible@172.31.43.137
```



```
[ansible@ip-172-31-15-244 ~]$
[ansible@ip-172-31-15-244 ~]$
[ansible@ip-172-31-15-244 ~]$
[ansible@ip-172-31-15-244 ~]$ ssh-copy-id ansible@172.31.33.99
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
The authenticity of host '172.31.33.99 (172.31.33.99)' can't be established.
ECDSA key fingerprint is SHA256:EYWQf5JNGZ0FFCLQQG6VD22I5p29Kk7DnT0RMeBSeZY.
ECDSA key fingerprint is MD5:ab:9e:46:ad:96:d6:99:15:ed:ac:a6:ee:cf:74:98:64.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@172.31.33.99's password:
Permission denied, please try again.
ansible@172.31.33.99's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansible@172.31.33.99'"
and check to make sure that only the key(s) you wanted were added.

[ansible@ip-172-31-15-244 ~]$
```

Ansible commands: (ansible has host pattern)

ansible all –list-host .... "All" refers to all nodes in inventory file.

ansible <group-name> --list-hosts .... " list all nodes under the group-name specified

ansible <group-name>[0] --list-hosts .. list first node in group (forward order)

ansible <group-name>[-1] --list-hosts .. list last node in group (reverse order)

....

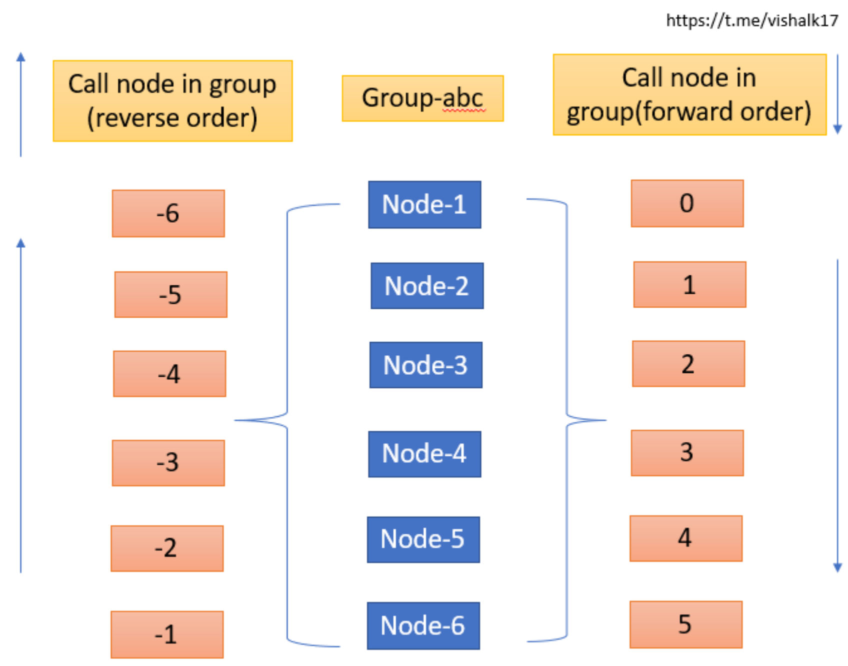
<group-name>[0:1] ... picks 1st and 2nd node

<group-name>[2:5] ... picks 3,4,5,6 nodes

<group-name-1>[0:2]:<group-name-2>[4:7] ...picks nodes 1,2,3 in group-name-1 & pick nodes 5,6,7,8, in group-name-2

Chapter 2 : ansible

==



1. Ad-hoc commands (temporary)

- These are commands can be run individually to perform quick functions.
- There is no Idempotency (can use same command for multiple times)
- There commands are not used for configuration mgt and deployment because this Commands are of one time usage.
- The ansible ad-hoc commands uses the /usr/bin/ansible command line tool to Automate a single task
- Can use linux command as it is

```
ansible dev -a “ls”
ansible all -a “touch vish”
ansible dev -a "sudo yum install httpd -y"
ansible dev -ba "yum remove httpd -y "                    … -b = become root user
```

2. Module

- Ansible ships with number of modules that can be executed directly on remote host or through playbook.
- Your library of modules can be reside on any machine and none other separate server or database required.
- Idempotency present

Chapter 2 : ansible

==

Pkg module:

Install = present

Uninstall / remove = absent

Update = latest

```
ansible dev -b -m yum -a "pkg=httpd state=present"

where,

-b = sudo privilege
-m = module
-a = execute whatever in inverted comma’s
Yum = yum module
dev = group name mentioned in /etc/ansible/hosts

ansible dev -b -m yum -a "pkg=httpd state=absent"
ansible dev -b -m yum -a "pkg=httpd state=latest"
```

service module:

```
ansible dev -b -m service -a "name=httpd state=started"
```

create user

```
ansible dev -m user -ba "name=vishalk17"
```

copy

```
ansible dev -m copy -ba "src=file1.txt dest=/mnt/"
```

Chapter 2 : ansible

===

setup

```
ansible dev -m setup
ansible dev -m setup -a “filter=*ipv4*
```

whenever we execute any ansible command setup first check status of that thing then further whatever module will going to execute.

3. Playbook

- Written in yaml format
- It is like a file where ,we write a codes consist of vars,task, handlers,files, templates and roles
- Each playbook is composed of one or more “module” in a list module is a collection of configuration files.
- Playbook divided in many sections like,

target section- defines host against which playbook task has to be executed

Variable section- defines variables

Task Section- List of all modules that we need to run in order

Handlers: Handlers are just like regular tasks in an Ansible playbook, but are only run if the Task contains a **notify** directive and also indicates that it changed something. For example, if a config file is changed, then the task referencing the config file may notify a service restart handler.

Let me give you an example of a playbook which will start the Apache httpd server program:

==

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

I hope the example will relate you to all the description of the playbook components that I have mentioned above. If it is still not clear to you, don’t worry all your doubts will be clear in the later part of this blog.

This is all about playbooks. The playbooks which will be written by you. But Ansible provides you with a wide range of modules as well, which you can use.

YAML (Yet Another Markup Language):

- for ansible, nearly every yaml file start with list
- Each item in the list is a list of key-value pairs commonly called a dictionary.
- All yaml files have to begin with “- - -” & end with “. . .” [not mandatory]
- Extension to playbook must has to be .yaml for better identation.
- All members of a list lines must begin with some indentation level starting with “ – “
- Dictionary is represented in a simple key:{space}value form

  For Example:

```
--- # Detail of customer
 customer:
    name: vishal
     job: founder & ceo of vish group of business
     status: businessman
```

Note : There should be space between : and value

Chapter 2 : ansible

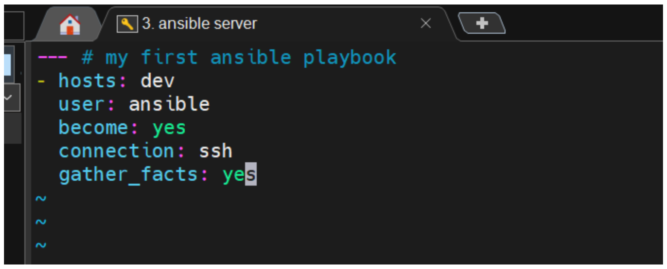
==

LAB :

Create playbook :

1.

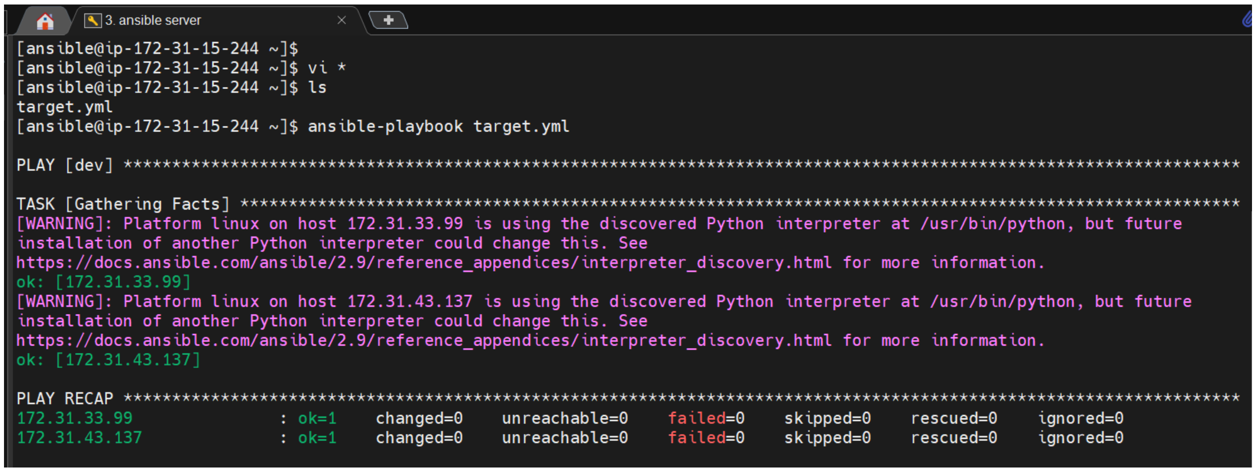
```
vi target.yml                                    (edit and save following)
```



```
--- # my first ansible playbook
- hosts: dev
  user: ansible
  become: yes
  connection: ssh
  gather_facts: yes
~
~
~
```

Execting playbook:

```
ansible-playbook target.yml
```

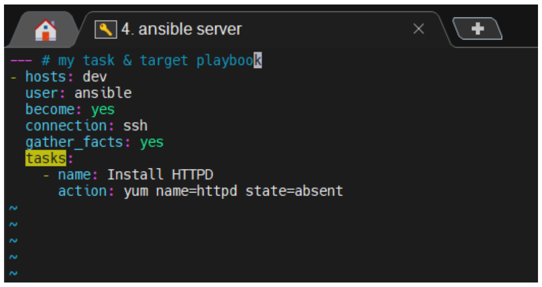


```
[ansible@ip-172-31-15-244 ~]$
[ansible@ip-172-31-15-244 ~]$ vi *
[ansible@ip-172-31-15-244 ~]$ ls
target.yml
[ansible@ip-172-31-15-244 ~]$ ansible-playbook target.yml

PLAY [dev] *********************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
[WARNING]: Platform linux on host 172.31.33.99 is using the discovered Python interpreter at /usr/bin/python, but future
installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.33.99]
[WARNING]: Platform linux on host 172.31.43.137 is using the discovered Python interpreter at /usr/bin/python, but future
installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.43.137]

PLAY RECAP *********************************************************************************************************************
172.31.33.99               : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.43.137              : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Chapter 2 : ansible

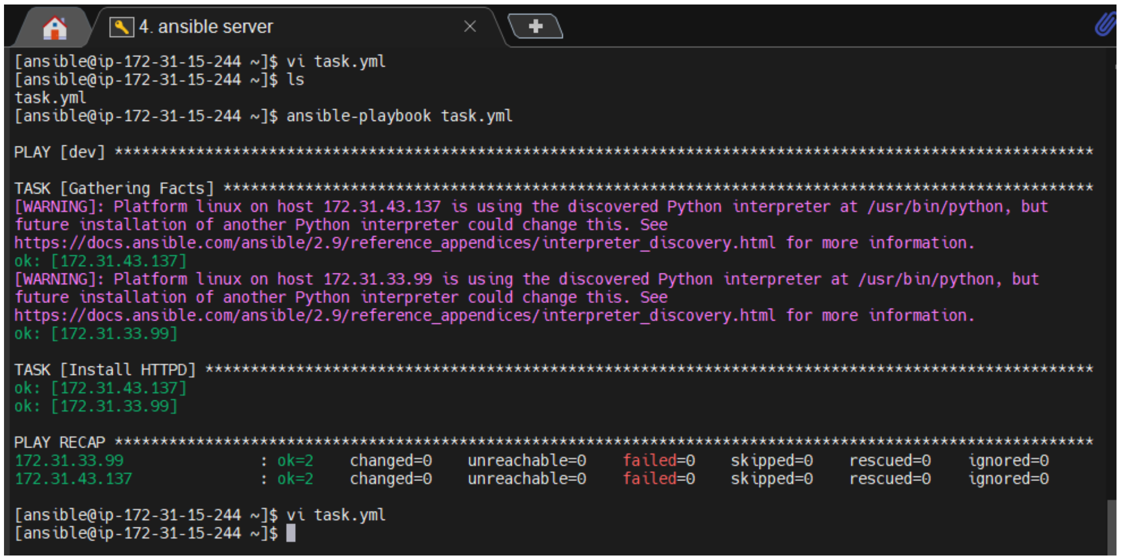
==

2. task and target



```
--- # my task & target playbook
- hosts: dev
  user: ansible
  become: yes
  connection: ssh
  gather_facts: yes
  tasks:
    - name: Install HTTPD
      action: yum name=httpd state=absent
~
~
~
~
~
```

Execute



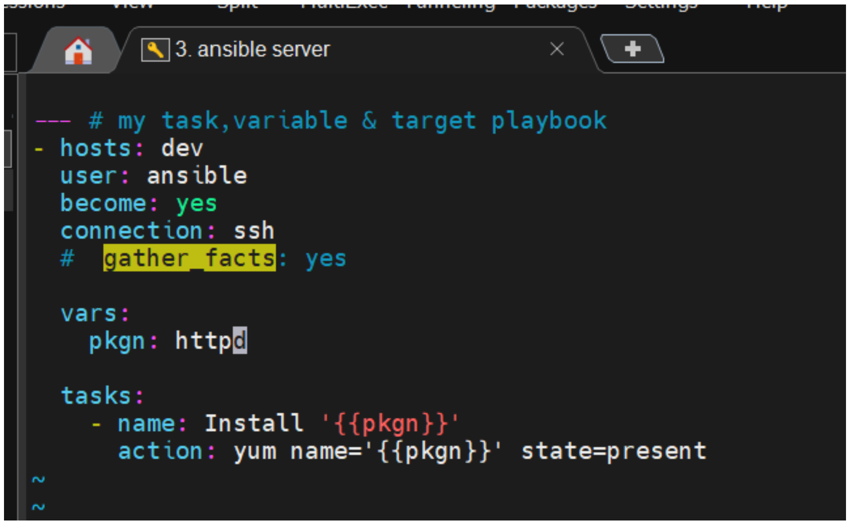
```
[ansible@ip-172-31-15-244 ~]$ vi task.yml
[ansible@ip-172-31-15-244 ~]$ ls
task.yml
[ansible@ip-172-31-15-244 ~]$ ansible-playbook task.yml

PLAY [dev] **********************************************************************************************

TASK [Gathering Facts] **********************************************************************************
[WARNING]: Platform linux on host 172.31.43.137 is using the discovered Python interpreter at /usr/bin/python, but
future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.43.137]
[WARNING]: Platform linux on host 172.31.33.99 is using the discovered Python interpreter at /usr/bin/python, but
future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.33.99]

TASK [Install HTTPD] ************************************************************************************
ok: [172.31.43.137]
ok: [172.31.33.99]

PLAY RECAP **********************************************************************************************
172.31.33.99               : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.43.137              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@ip-172-31-15-244 ~]$ vi task.yml
[ansible@ip-172-31-15-244 ~]$
```

Chapter 2 : ansible

==

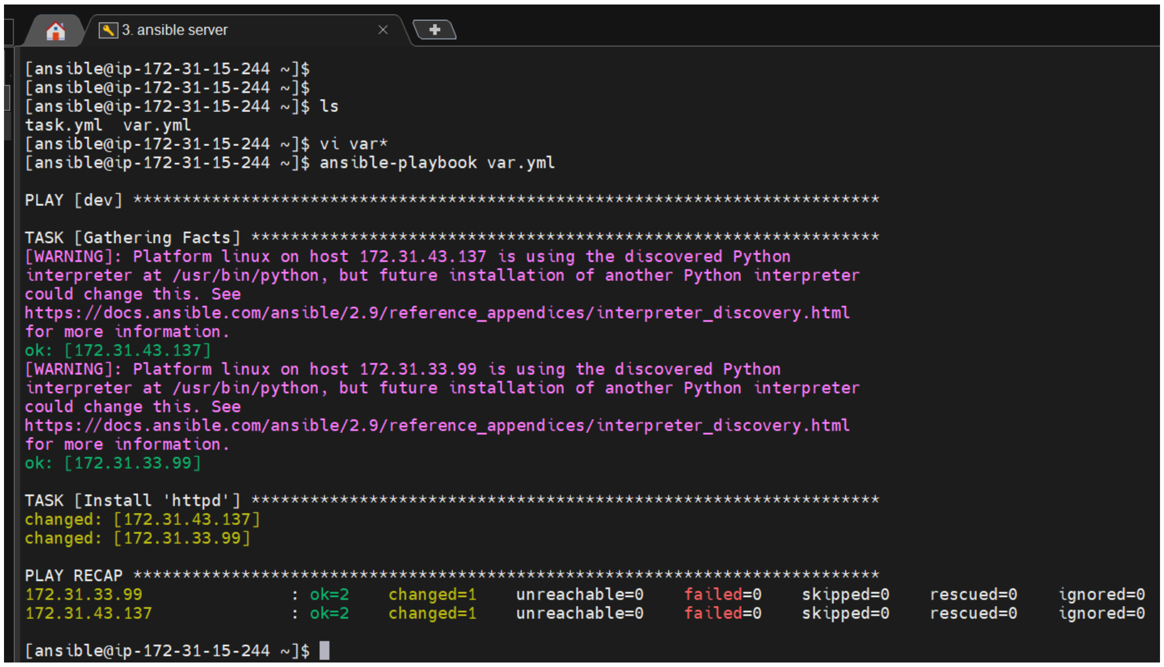
- **Variables**

- Ansible uses variables which are defined previously to enable more flexibity in playbooka & roles.They can be used to loop through a set of given values, access various information like the host name of a system and replace certain strings in templates with specific values.
- Put variable section above the task so that we define it first and use it later.



```
--- # my task,variable & target playbook
- hosts: dev
  user: ansible
  become: yes
  connection: ssh
  #  gather_facts: yes

  vars:
    pkgn: httpd

  tasks:
    - name: Install '{{pkgn}}'
      action: yum name='{{pkgn}}' state=present
~
~
```

After execution



```
[ansible@ip-172-31-15-244 ~]$
[ansible@ip-172-31-15-244 ~]$
[ansible@ip-172-31-15-244 ~]$ ls
task.yml  var.yml
[ansible@ip-172-31-15-244 ~]$ vi var*
[ansible@ip-172-31-15-244 ~]$ ansible-playbook var.yml

PLAY [dev] ***************************************************************************

TASK [Gathering Facts] ***************************************************************
[WARNING]: Platform linux on host 172.31.43.137 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html
for more information.
ok: [172.31.43.137]
[WARNING]: Platform linux on host 172.31.33.99 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python interpreter
could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html
for more information.
ok: [172.31.33.99]

TASK [Install 'httpd'] ***************************************************************
changed: [172.31.43.137]
changed: [172.31.33.99]

PLAY RECAP ***************************************************************************
172.31.33.99               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.43.137              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@ip-172-31-15-244 ~]$
```

Chapter 2 : ansible

==

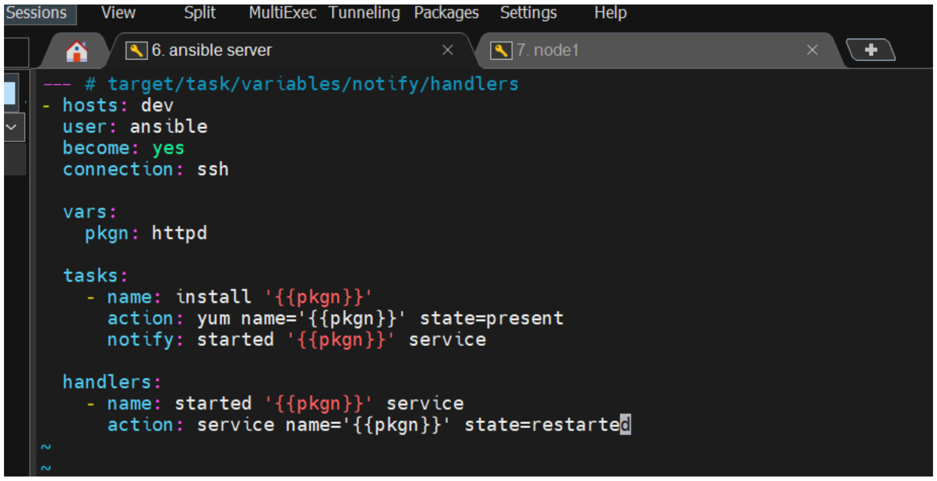
- **Handlers**

- It is exactly the same as a task but will run when called by another task.
- Handlers are just like regular tasks in an ansible playbook, but are only run if the task contains a notify directive and also indicates that it changed something.

- **Dry run**

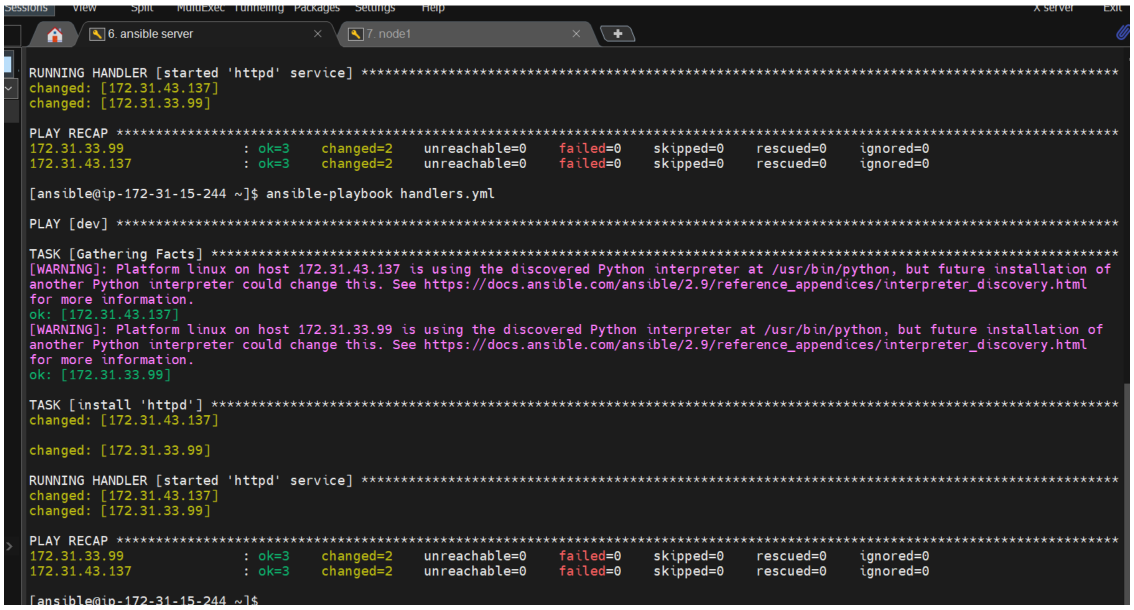
- Check whether the playbook is formatted correctly.

ansible-playbook handlers.yml –check



```
--- # target/task/variables/notify/handlers
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  vars:
    pkgn: httpd

  tasks:
    - name: install '{{pkgn}}'
      action: yum name='{{pkgn}}' state=present
      notify: started '{{pkgn}}' service

  handlers:
    - name: started '{{pkgn}}' service
      action: service name='{{pkgn}}' state=restarted
~
~
```

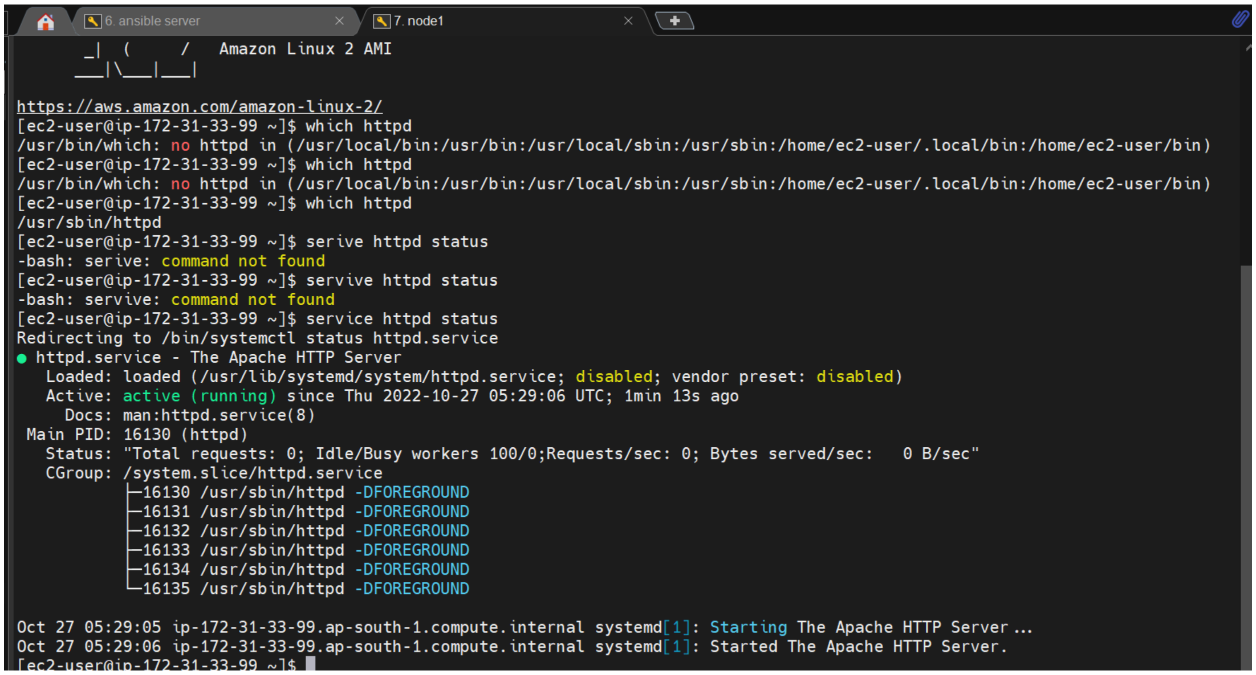
After execution



```
RUNNING HANDLER [started 'httpd' service] ****************************************************************************
changed: [172.31.43.137]
changed: [172.31.33.99]

PLAY RECAP ***********************************************************************************************************
172.31.33.99               : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.43.137              : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@ip-172-31-15-244 ~]$ ansible-playbook handlers.yml

PLAY [dev] ***********************************************************************************************************

TASK [Gathering Facts] ***********************************************************************************************
[WARNING]: Platform linux on host 172.31.43.137 is using the discovered Python interpreter at /usr/bin/python, but future installation of
another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html
for more information.
ok: [172.31.43.137]
[WARNING]: Platform linux on host 172.31.33.99 is using the discovered Python interpreter at /usr/bin/python, but future installation of
another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html
for more information.
ok: [172.31.33.99]

TASK [install 'httpd'] ***********************************************************************************************
changed: [172.31.43.137]

changed: [172.31.33.99]

RUNNING HANDLER [started 'httpd' service] ****************************************************************************
changed: [172.31.43.137]
changed: [172.31.33.99]

PLAY RECAP ***********************************************************************************************************
172.31.33.99               : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.43.137              : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@ip-172-31-15-244 ~]$
```

Chapter 2 : ansible

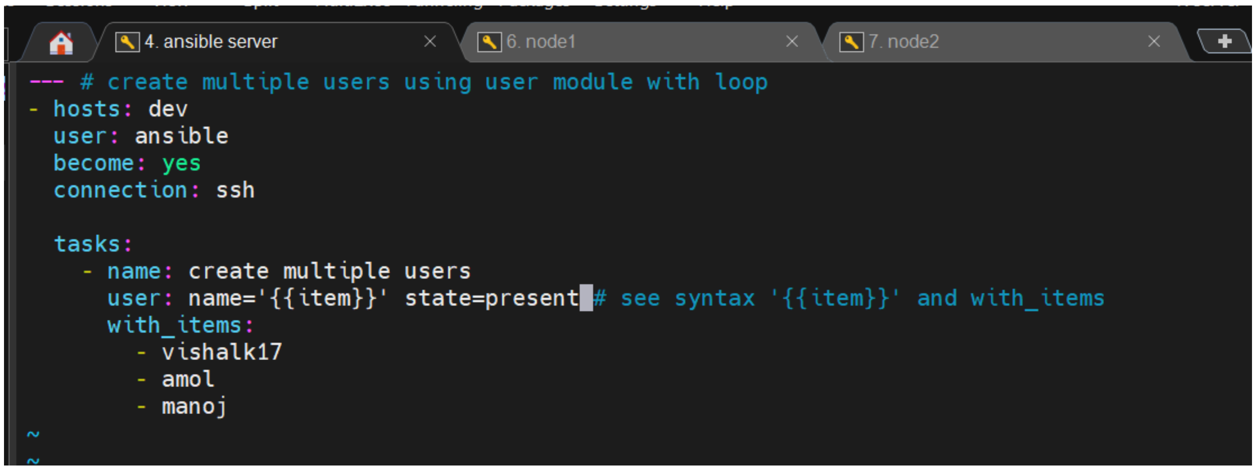
==



```
      _|  (     /   Amazon Linux 2 AMI
     ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-33-99 ~]$ which httpd
/usr/bin/which: no httpd in (/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/ec2-user/.local/bin:/home/ec2-user/bin)
[ec2-user@ip-172-31-33-99 ~]$ which httpd
/usr/bin/which: no httpd in (/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/ec2-user/.local/bin:/home/ec2-user/bin)
[ec2-user@ip-172-31-33-99 ~]$ which httpd
/usr/sbin/httpd
[ec2-user@ip-172-31-33-99 ~]$ serive httpd status
-bash: serive: command not found
[ec2-user@ip-172-31-33-99 ~]$ servive httpd status
-bash: servive: command not found
[ec2-user@ip-172-31-33-99 ~]$ service httpd status
Redirecting to /bin/systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2022-10-27 05:29:06 UTC; 1min 13s ago
     Docs: man:httpd.service(8)
 Main PID: 16130 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─16130 /usr/sbin/httpd -DFOREGROUND
           ├─16131 /usr/sbin/httpd -DFOREGROUND
           ├─16132 /usr/sbin/httpd -DFOREGROUND
           ├─16133 /usr/sbin/httpd -DFOREGROUND
           ├─16134 /usr/sbin/httpd -DFOREGROUND
           └─16135 /usr/sbin/httpd -DFOREGROUND

Oct 27 05:29:05 ip-172-31-33-99.ap-south-1.compute.internal systemd[1]: Starting The Apache HTTP Server...
Oct 27 05:29:06 ip-172-31-33-99.ap-south-1.compute.internal systemd[1]: Started The Apache HTTP Server.
[ec2-user@ip-172-31-33-99 ~]$
```

- **Loops:**

- Sometimes you want to repeat a task multiple times in computer programming this is called loops. Common ansible loops include changing ownership on server files and or directories with the file module , creating multiple users with the user module and repeating a polling step until certain result is reached.

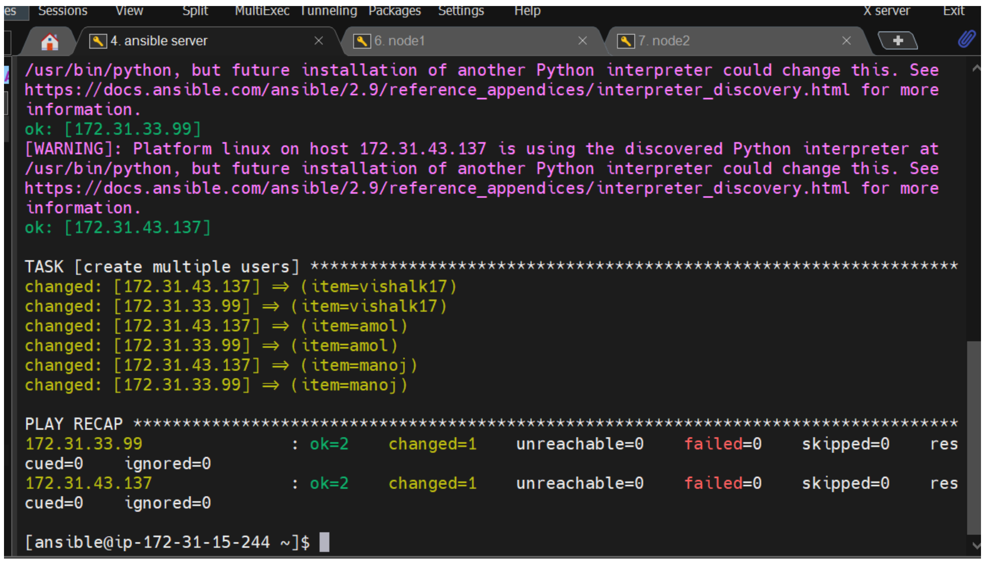


```
--- # create multiple users using user module with loop
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: create multiple users
      user: name='{{item}}' state=present # see syntax '{{item}}' and with_items
      with_items:
        - vishalk17
        - amol
        - manoj
~
~
```

After execution

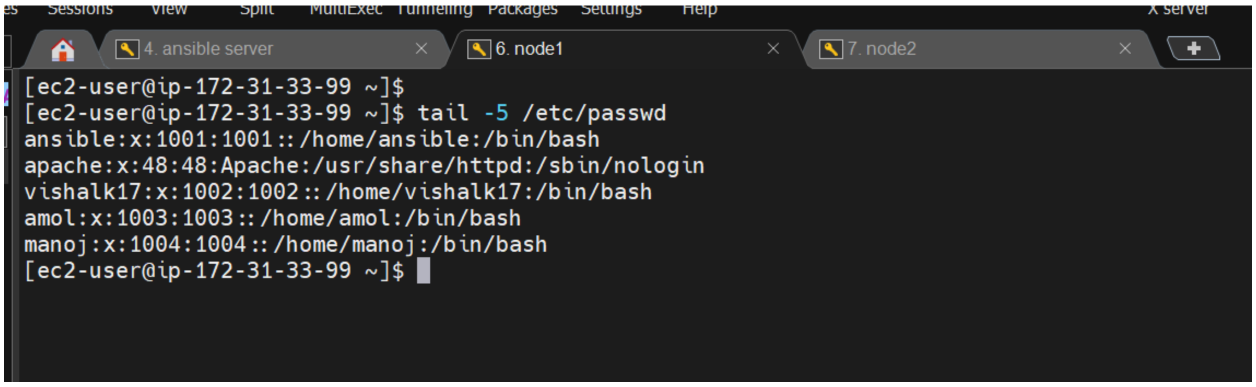
Chapter 2 : ansible

==

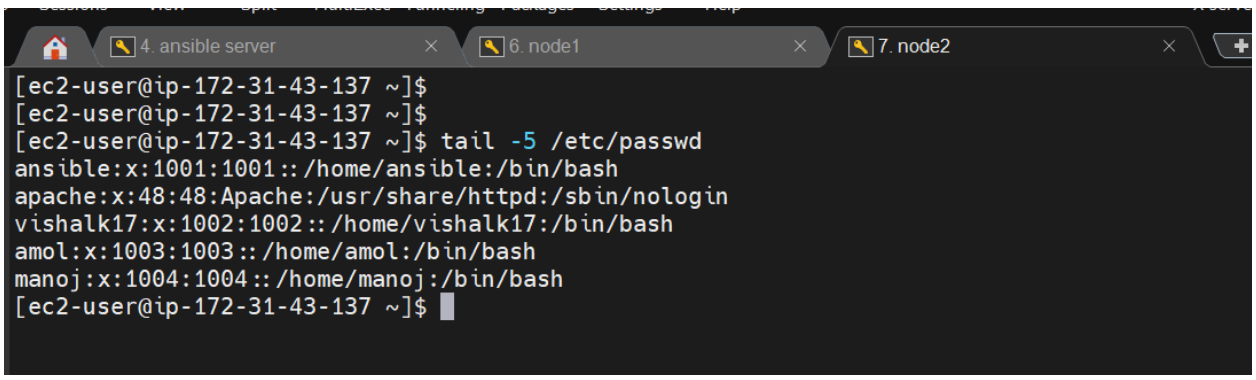


```
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [172.31.33.99]
[WARNING]: Platform linux on host 172.31.43.137 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [172.31.43.137]

TASK [create multiple users] ****************************************************************
changed: [172.31.43.137] => (item=vishalk17)
changed: [172.31.33.99] => (item=vishalk17)
changed: [172.31.43.137] => (item=amol)
changed: [172.31.33.99] => (item=amol)
changed: [172.31.43.137] => (item=manoj)
changed: [172.31.33.99] => (item=manoj)

PLAY RECAP **********************************************************************************
172.31.33.99               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
172.31.43.137              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0

[ansible@ip-172-31-15-244 ~]$
```

Lets verify on nodes whether users are created or not



```
[ec2-user@ip-172-31-33-99 ~]$
[ec2-user@ip-172-31-33-99 ~]$ tail -5 /etc/passwd
ansible:x:1001:1001::/home/ansible:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
vishalk17:x:1002:1002::/home/vishalk17:/bin/bash
amol:x:1003:1003::/home/amol:/bin/bash
manoj:x:1004:1004::/home/manoj:/bin/bash
[ec2-user@ip-172-31-33-99 ~]$
```



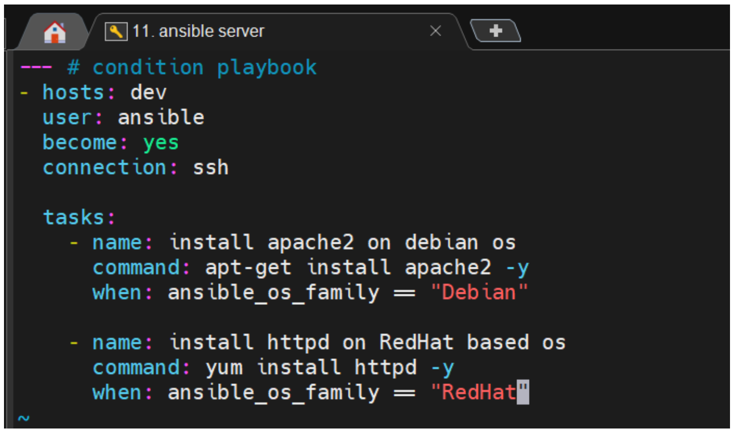
```
[ec2-user@ip-172-31-43-137 ~]$
[ec2-user@ip-172-31-43-137 ~]$
[ec2-user@ip-172-31-43-137 ~]$ tail -5 /etc/passwd
ansible:x:1001:1001::/home/ansible:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
vishalk17:x:1002:1002::/home/vishalk17:/bin/bash
amol:x:1003:1003::/home/amol:/bin/bash
manoj:x:1004:1004::/home/manoj:/bin/bash
[ec2-user@ip-172-31-43-137 ~]$
```

Chapter 2 : ansible

==

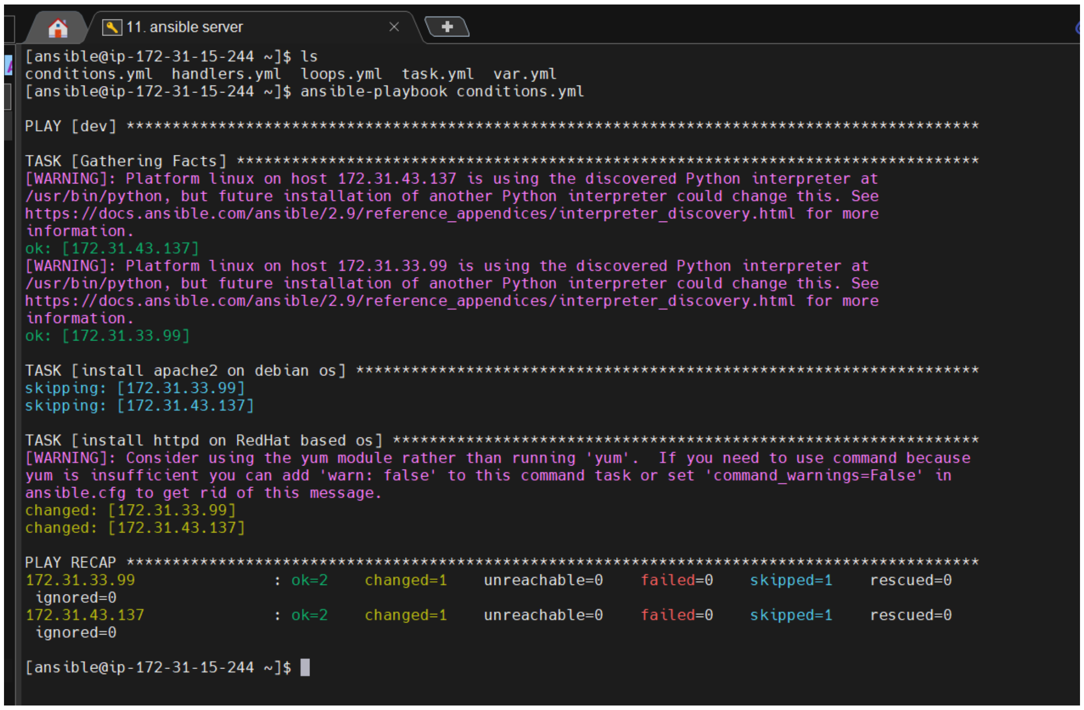
- **Conditions:**

- In a playbook, you may want to execute different tasks, or have different goals, depending on the value of a fact (data about the remote system), a variable, or the result of a previous task. You may want the value of some variables to depend on the value of other variables. Or you may want to create additional groups of hosts based on whether the hosts match other criteria. You can do all of these things with conditionals.



```
--- # condition playbook
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install apache2 on debian os
      command: apt-get install apache2 -y
      when: ansible_os_family == "Debian"

    - name: install httpd on RedHat based os
      command: yum install httpd -y
      when: ansible_os_family == "RedHat"
~
```

After execution



```
[ansible@ip-172-31-15-244 ~]$ ls
conditions.yml  handlers.yml  loops.yml  task.yml  var.yml
[ansible@ip-172-31-15-244 ~]$ ansible-playbook conditions.yml

PLAY [dev] *******************************************************************************************

TASK [Gathering Facts] *******************************************************************************
[WARNING]: Platform linux on host 172.31.43.137 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [172.31.43.137]
[WARNING]: Platform linux on host 172.31.33.99 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [172.31.33.99]

TASK [install apache2 on debian os] ******************************************************************
skipping: [172.31.33.99]
skipping: [172.31.43.137]

TASK [install httpd on RedHat based os] **************************************************************
[WARNING]: Consider using the yum module rather than running 'yum'.  If you need to use command because
yum is insufficient you can add 'warn: false' to this command task or set 'command_warnings=False' in
ansible.cfg to get rid of this message.
changed: [172.31.33.99]
changed: [172.31.43.137]

PLAY RECAP *******************************************************************************************
172.31.33.99               : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0
 ignored=0
172.31.43.137              : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0
 ignored=0

[ansible@ip-172-31-15-244 ~]$ 
```

My both machine’s os based on redhat . thus Debian task had been skipped

Chapter 2 : ansible

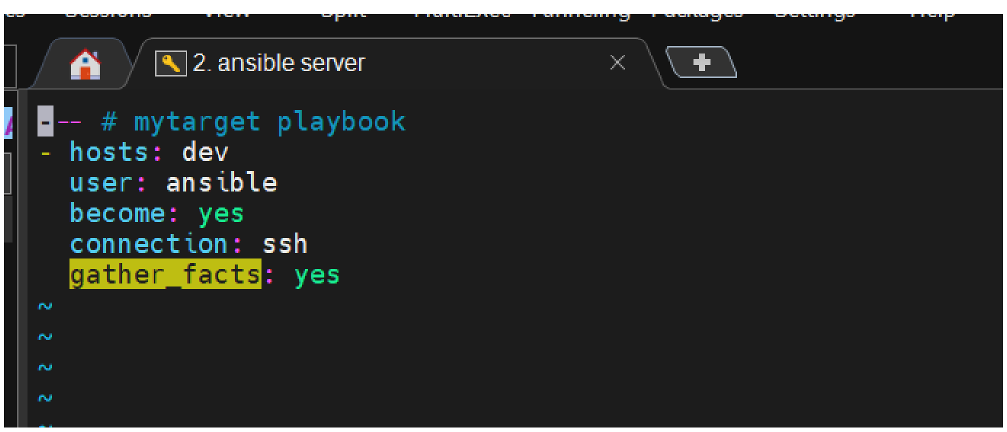
==

- **Vault :**

- Ansible allows keeping sensitive data such as passwords or keys in encrypted files,rather than a plaintext in your playbooks.
- Encryption based on AES256 algorithem.

ansible-vault create xyz.yml	…… creating a new encrypted playbook
ansible-vault edit xyz.yml	…… edit the encrypted playbook
ansible-vault rekey xyz.yml	…… to change the password of playbook
ansible-vault encrypt xyz.yml	…… to encrypt the existing playbook
ansible-vault decrypt xyz.yml	…… to decrypt an encrypted playbook

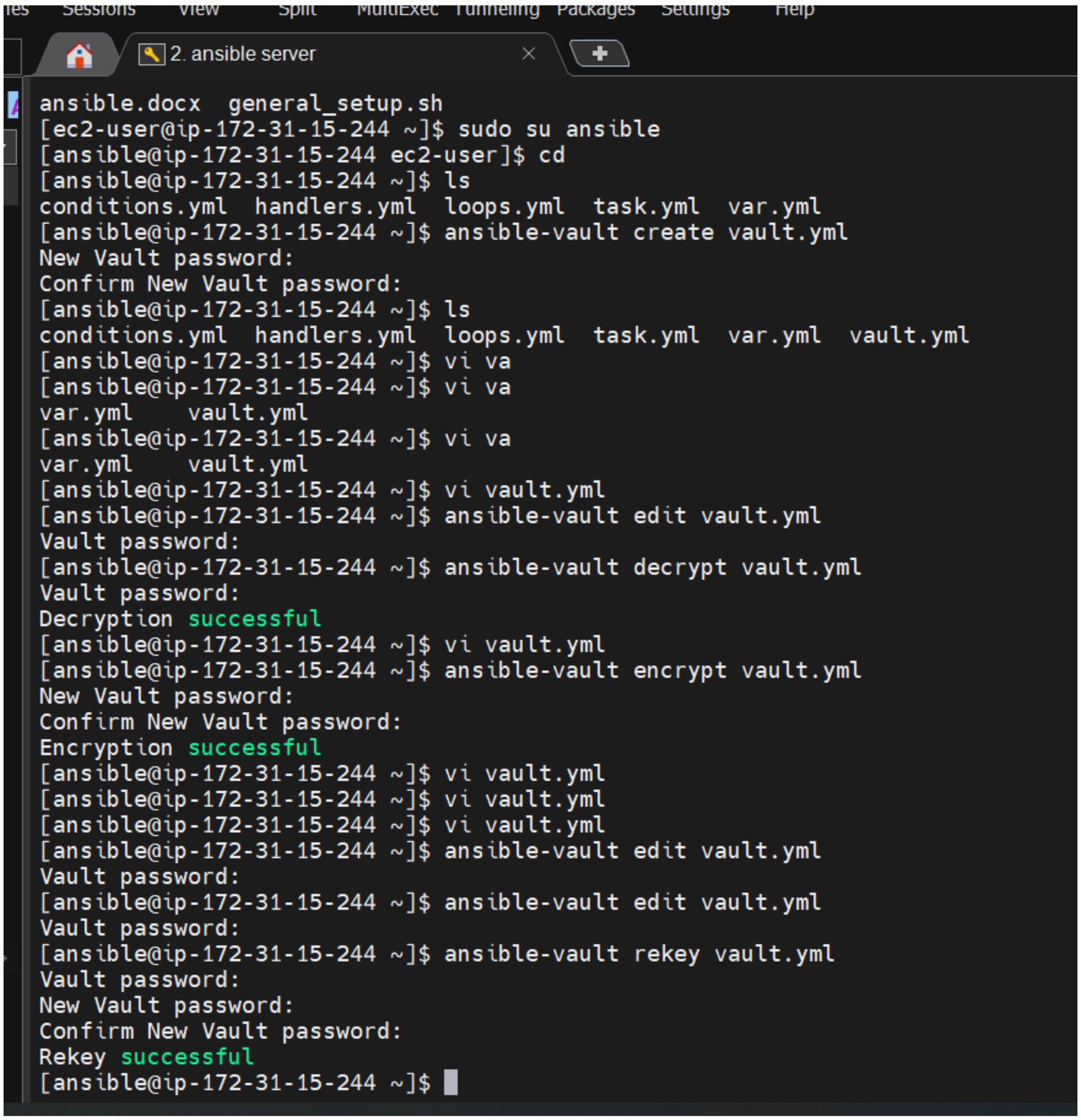
vault.yml



```
--- # mytarget playbook
- hosts: dev
  user: ansible
  become: yes
  connection: ssh
  gather_facts: yes
~
~
~
~
```

Chapter 2 : ansible

==

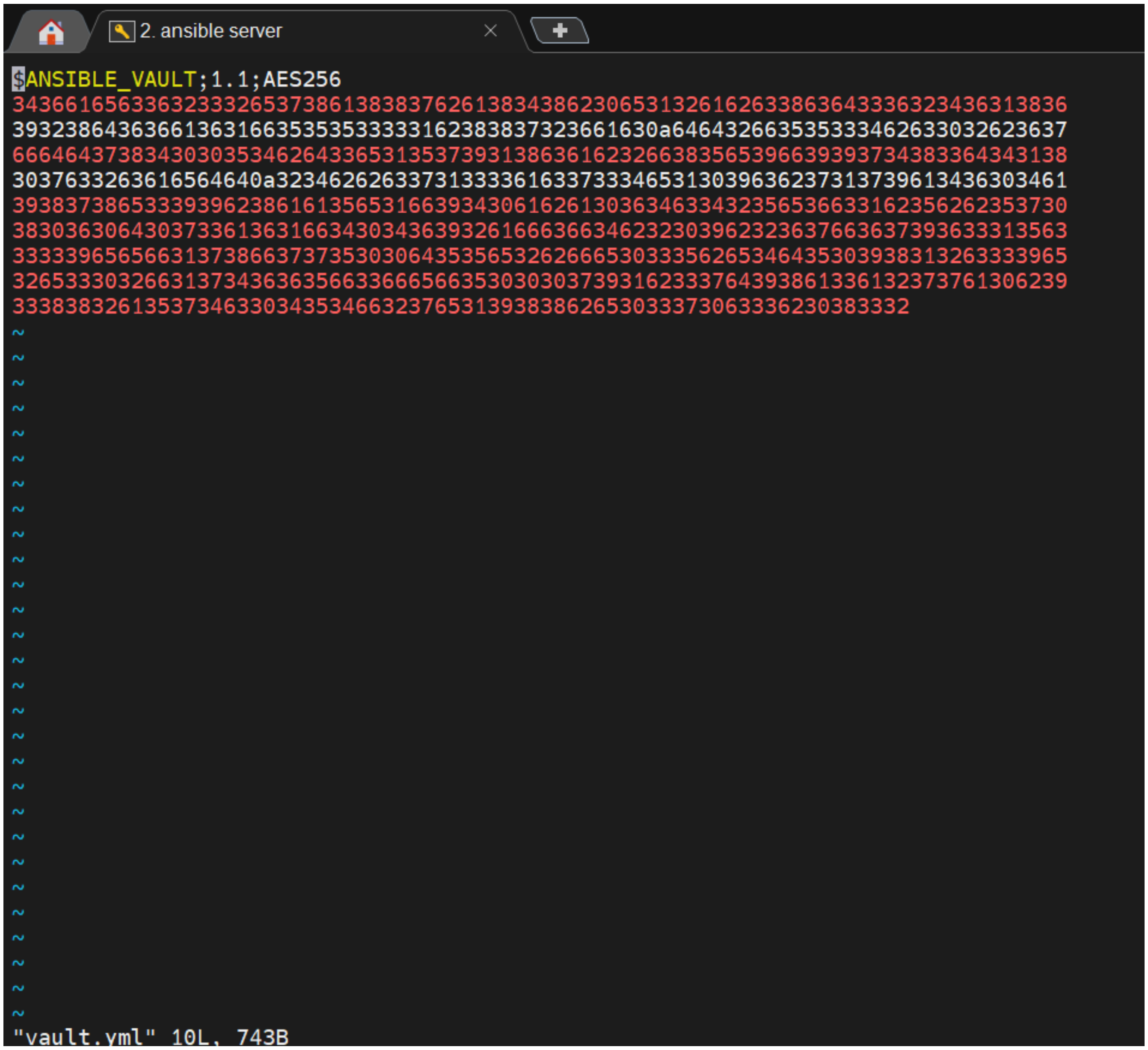


```
ansible.docx  general_setup.sh
[ec2-user@ip-172-31-15-244 ~]$ sudo su ansible
[ansible@ip-172-31-15-244 ec2-user]$ cd
[ansible@ip-172-31-15-244 ~]$ ls
conditions.yml  handlers.yml  loops.yml  task.yml  var.yml
[ansible@ip-172-31-15-244 ~]$ ansible-vault create vault.yml
New Vault password:
Confirm New Vault password:
[ansible@ip-172-31-15-244 ~]$ ls
conditions.yml  handlers.yml  loops.yml  task.yml  var.yml  vault.yml
[ansible@ip-172-31-15-244 ~]$ vi va
[ansible@ip-172-31-15-244 ~]$ vi va
var.yml    vault.yml
[ansible@ip-172-31-15-244 ~]$ vi va
var.yml    vault.yml
[ansible@ip-172-31-15-244 ~]$ vi vault.yml
[ansible@ip-172-31-15-244 ~]$ ansible-vault edit vault.yml
Vault password:
[ansible@ip-172-31-15-244 ~]$ ansible-vault decrypt vault.yml
Vault password:
Decryption successful
[ansible@ip-172-31-15-244 ~]$ vi vault.yml
[ansible@ip-172-31-15-244 ~]$ ansible-vault encrypt vault.yml
New Vault password:
Confirm New Vault password:
Encryption successful
[ansible@ip-172-31-15-244 ~]$ vi vault.yml
[ansible@ip-172-31-15-244 ~]$ vi vault.yml
[ansible@ip-172-31-15-244 ~]$ vi vault.yml
[ansible@ip-172-31-15-244 ~]$ ansible-vault edit vault.yml
Vault password:
[ansible@ip-172-31-15-244 ~]$ ansible-vault edit vault.yml
Vault password:
[ansible@ip-172-31-15-244 ~]$ ansible-vault rekey vault.yml
Vault password:
New Vault password:
Confirm New Vault password:
Rekey successful
[ansible@ip-172-31-15-244 ~]$
```

Chapter 2 : ansible

==

This is how encryption look like when you try to open encrypted playbook



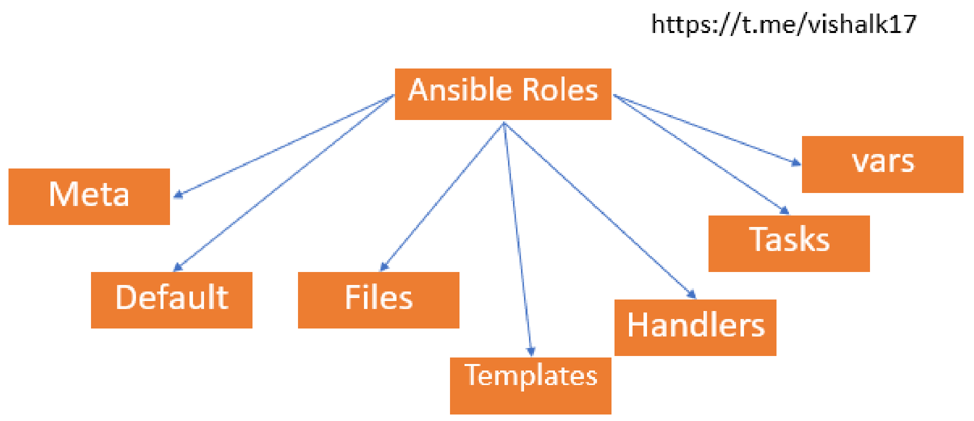
```
$ANSIBLE_VAULT;1.1;AES256
34366165633632333265373861383837626138343862306531326162633863643336323436313836
39323864363661363166353535333331623838373236616 30a6464326635353334626330326236 37
66646437383430303534626433653135373931386361623266383565396639393734383364343138
30376332636165646 40a3234626263373133336163373334653130396362373137396134363 03461
39383738653339396238616135653166393430616261303634633432356536633162356262353730
38303630643037336136316634303436393261666366346232303962323637663637393633313563
33333965656631373866373735303064353565326266653033356265346435303938313263333965
32653330326631373436363566336665663530303037393162333764393861336132373761306239
33383832613537346330343534663237653139383862653033373063336230383332
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"vault.yml" 10L, 743B
```

Chapter 2 : ansible

==

Roles:

- We can use two techniques for running a set of tasks:- includes and roles
- Roles are good for organizing tasks and encapsulating data needed to accomplish those tasks



- We can organize playbook into a directory structure called roles
- Adding more and more functionality to the playbooks will make it difficult to maintain in a single file.

Default : It stores the data about role/application default variables eg. If you want to run to port 80 or 8080 then variables needs to define in this path.

Files: it contains files need to be transferred to the remote vm (static files)

Handlers: They are triggers or task we can segregate all the handlers required in playbook

Meta : This directory contain files that establish roles dependencies eg. Author name, supported platform, dependencies if any.

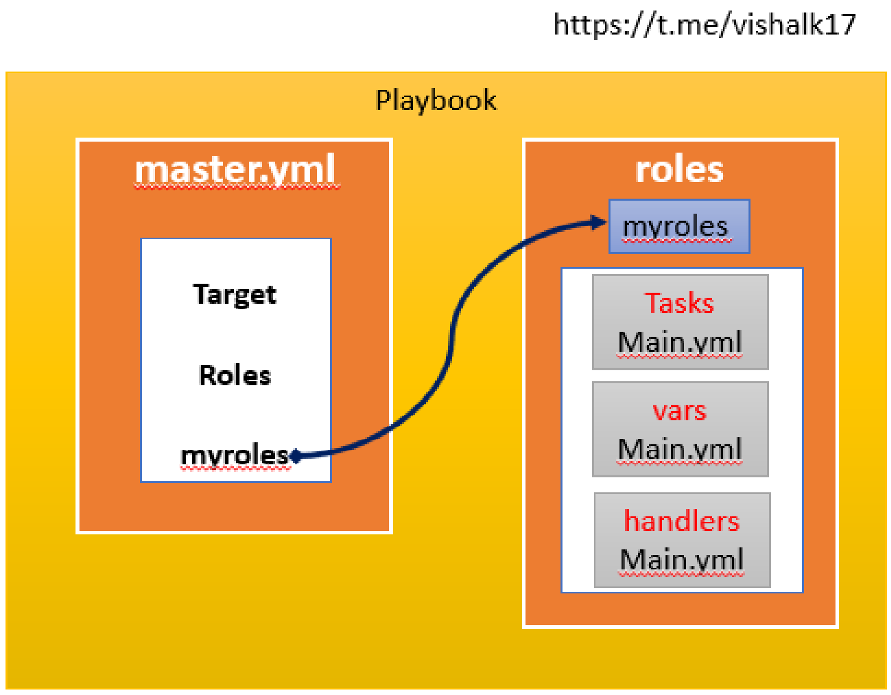
Tasks: It contains all the tasks that is normally in the playbook eg. Install packages and copies files etc.

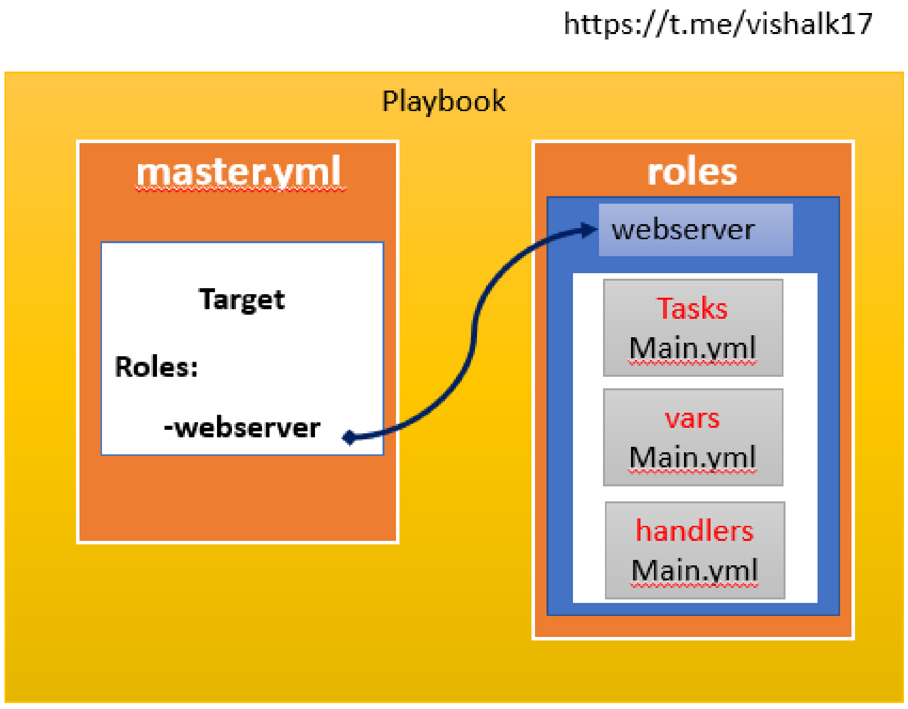
Vars: variables for the role can be specified in this directory and used in your configuration files both vars and default stores variables.

Notes by vishalk17 (t.me/vishalk17) Devops

Chapter 2 : ansible

==

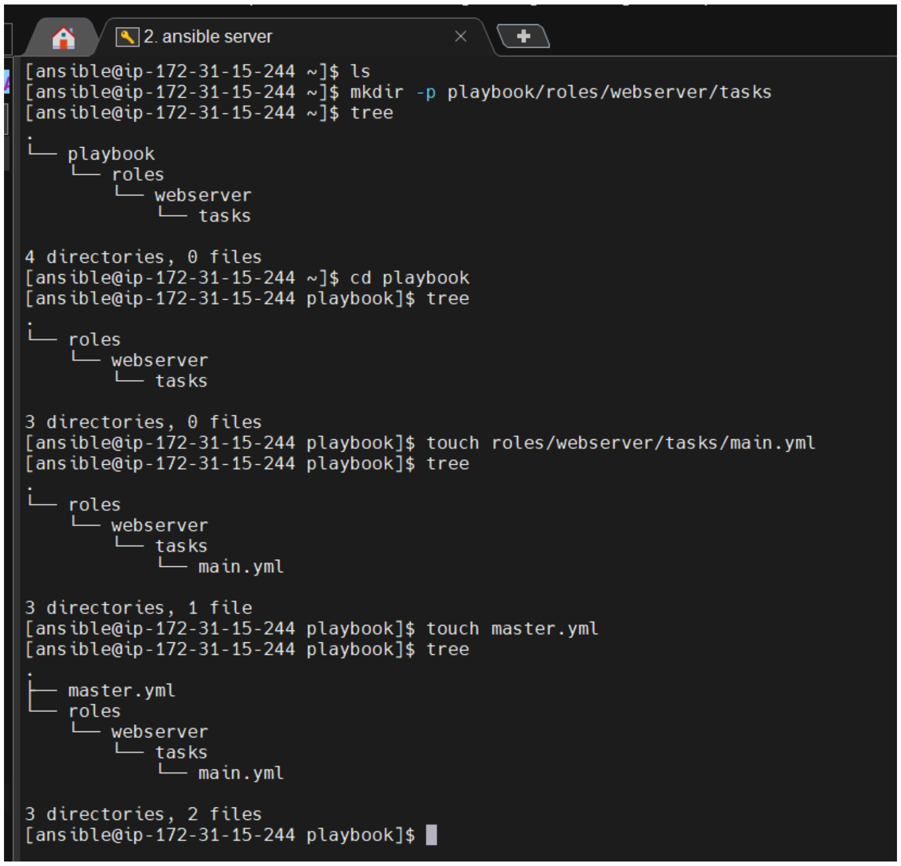




Chapter 2 : ansible

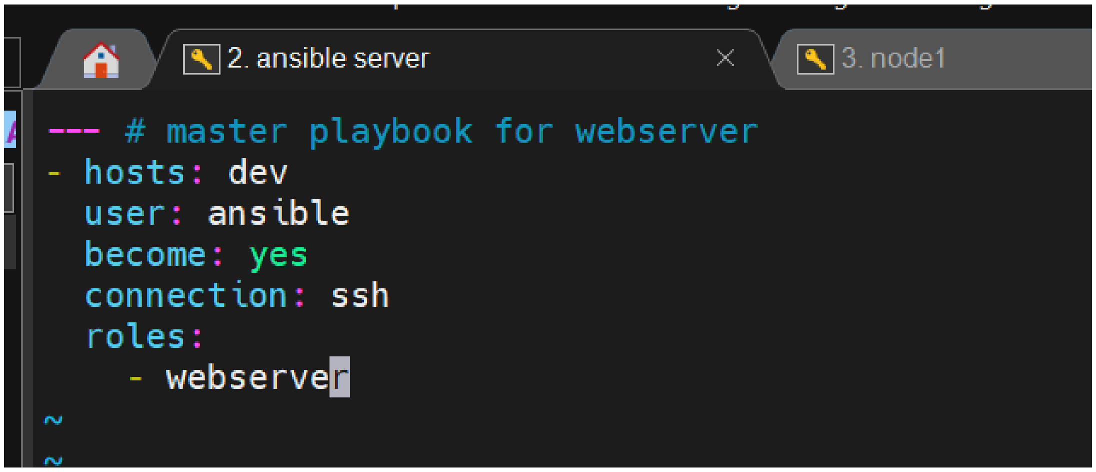
==

Tree structure



```
[ansible@ip-172-31-15-244 ~]$ ls
[ansible@ip-172-31-15-244 ~]$ mkdir -p playbook/roles/webserver/tasks
[ansible@ip-172-31-15-244 ~]$ tree
.
└── playbook
    └── roles
        └── webserver
            └── tasks

4 directories, 0 files
[ansible@ip-172-31-15-244 ~]$ cd playbook
[ansible@ip-172-31-15-244 playbook]$ tree
.
└── roles
    └── webserver
        └── tasks

3 directories, 0 files
[ansible@ip-172-31-15-244 playbook]$ touch roles/webserver/tasks/main.yml
[ansible@ip-172-31-15-244 playbook]$ tree
.
└── roles
    └── webserver
        └── tasks
            └── main.yml

3 directories, 1 file
[ansible@ip-172-31-15-244 playbook]$ touch master.yml
[ansible@ip-172-31-15-244 playbook]$ tree
.
├── master.yml
└── roles
    └── webserver
        └── tasks
            └── main.yml

3 directories, 2 files
[ansible@ip-172-31-15-244 playbook]$
```

vi playbook/master.yml

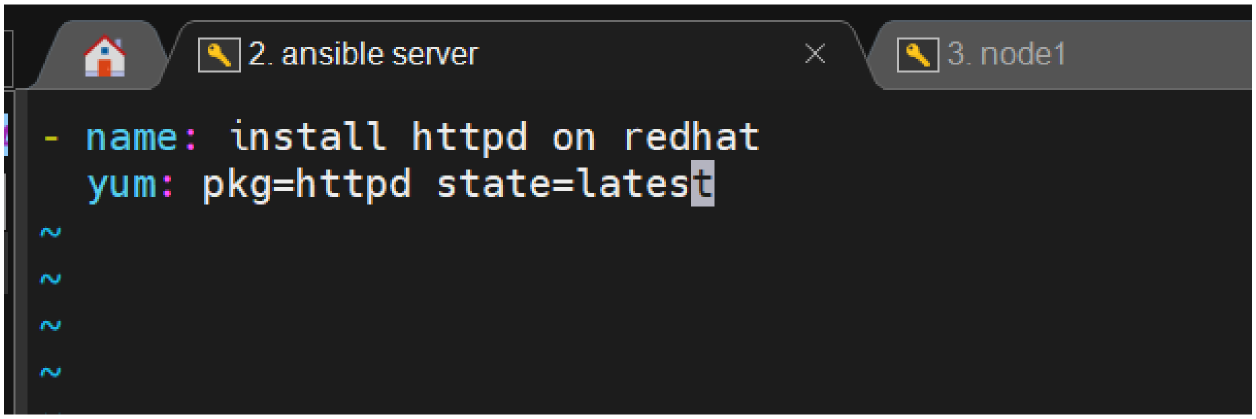


```
--- # master playbook for webserver
- hosts: dev
  user: ansible
  become: yes
  connection: ssh
  roles:
    - webserver
~
~
```

Chapter 2 : ansible

===

vi playbook/roles/webserver/tasks/main.yml

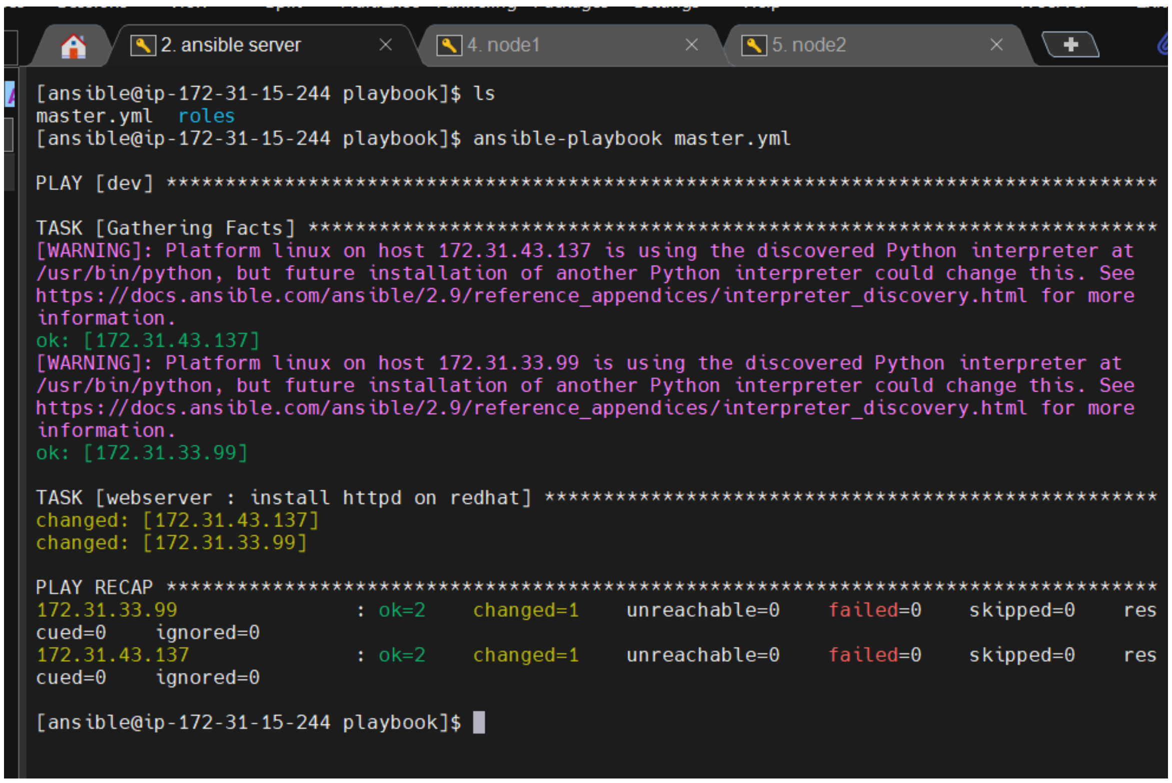


```
- name: install httpd on redhat
  yum: pkg=httpd state=latest
~
~
~
~
```

Output

cd playbook

ansible-playbook master.yml



```
[ansible@ip-172-31-15-244 playbook]$ ls
master.yml  roles
[ansible@ip-172-31-15-244 playbook]$ ansible-playbook master.yml

PLAY [dev] ***************************************************************************

TASK [Gathering Facts] ***************************************************************
[WARNING]: Platform linux on host 172.31.43.137 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [172.31.43.137]
[WARNING]: Platform linux on host 172.31.33.99 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [172.31.33.99]

TASK [webserver : install httpd on redhat] *******************************************
changed: [172.31.43.137]
changed: [172.31.33.99]

PLAY RECAP ***************************************************************************
172.31.33.99               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0
172.31.43.137              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    res
cued=0    ignored=0

[ansible@ip-172-31-15-244 playbook]$
```