

## Configuration Management Tool

Page No. 55

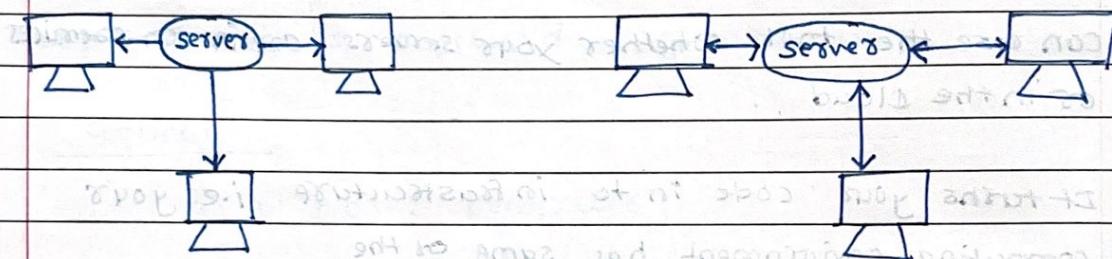
Date

- Each and every minute detail of your Machine (Server, storage) delete, update, create configuration management tool part - operation part in DevOps.

### Configuration Management Tool

Push Based

Pull Based



- push configuration server pushes configuration to the node.

e.g. Ansible

Saltstack

- pull configuration nodes check with the server periodically and fetch the configuration from it.

e.g. Chef

puppet

### Advantage of CM Tool

Complete Automation

Increase Uptime

Improve performance

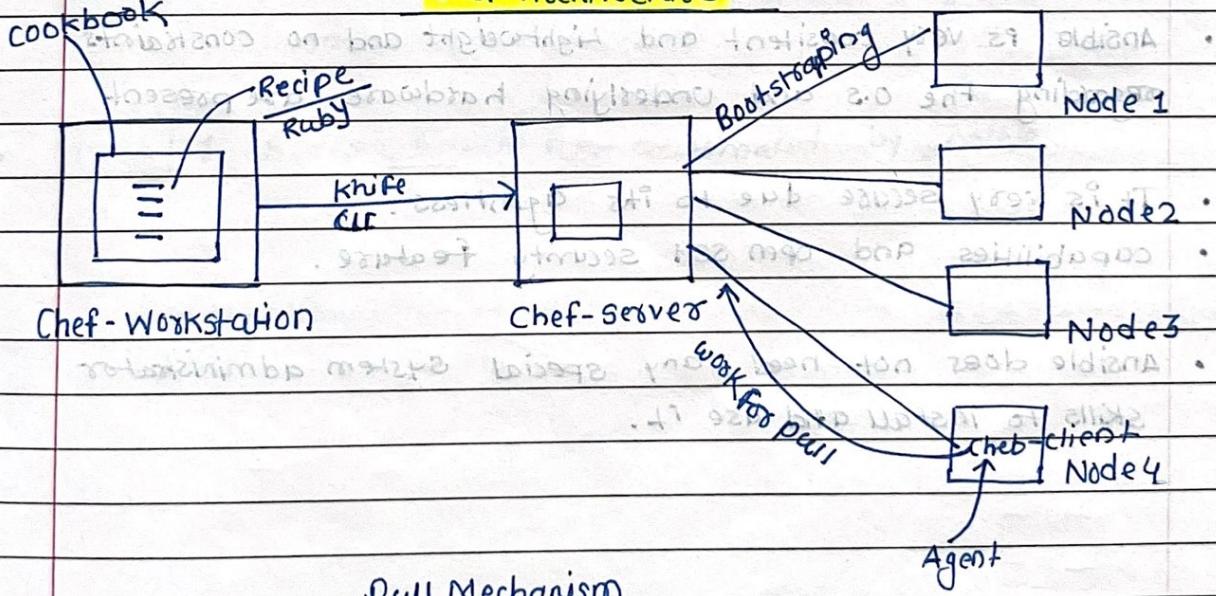
Ensure compliance

Prevent Errors

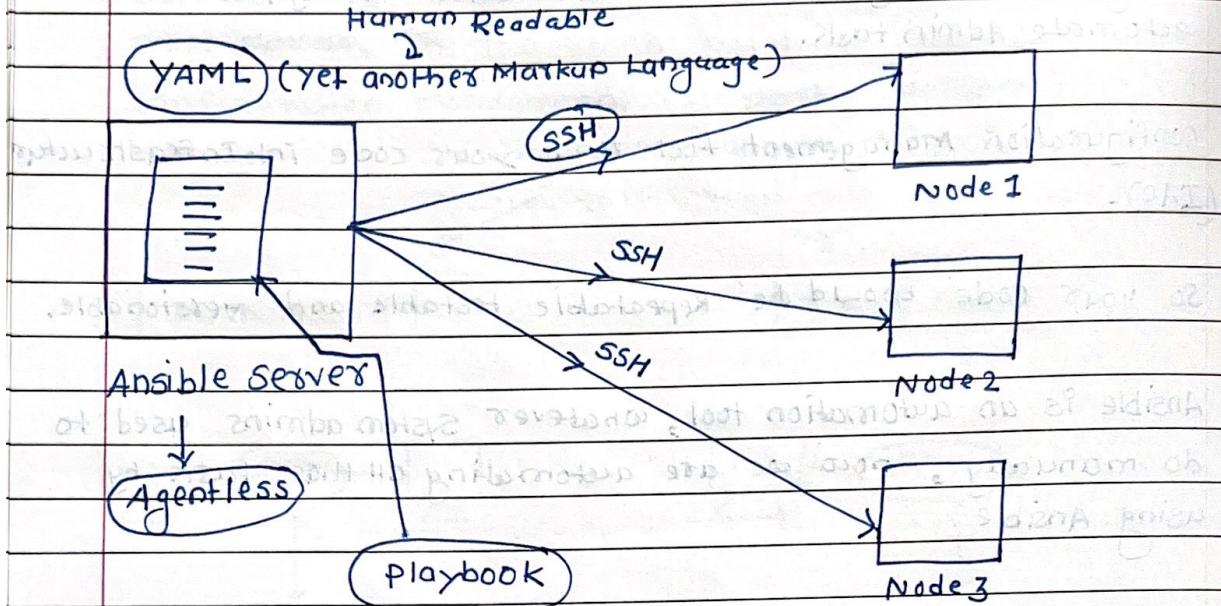
Reduce cost

- Configuration Management - It is a method through which we automate Admin task.
- Configuration Management tool turns your code into Infrastructure (IAC).
- So your code would be Repeatable, Testable and Versionable.
- Ansible is an automation tool, whatever system admins used to do manually, now we are automating all those tasks by using Ansible.
- Ansible is an OpenSource IT-Configuration Management, Deployment and Orchestration Tool.  
It aims to provide large productivity gains to a wide variety of Automation challenges.

### Chef Architecture



## Ansible Architecture.



## Advantages

- Ansible is free to use by everyone.
- Ansible is very consistent and lightweight and no constraints regarding the O.S and underlying hardware are present.
- It is very secure due to its agentless capabilities and open SSH security feature.
- Ansible does not need any special system administrator skills to install and use it.

## Terms used in Ansible

- **Ansible Server** : The machine where ansible is installed and from which all tasks and playbooks will be 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 organising tasks and related files to be later called in a playbook.
- **Fact** : Information fetched from the client system from the global variables with the gather-facts operation.
- **Inventory** : file containing data about the ansible client servers.
- **Play** : Execution of a playbook.
- **Handler** : Task which is called only if a notifier is present.
- **Notifier** : section attribute to a task which calls a handler if the output is changed.
- **playbook** : It consist code in YAML format, which describe tasks to be executed.
- **Host** : Nodes, which are automated by ansible.

## Ansible History

- Michael DeHaan developed Ansible and the project began in February 2012.
- Redhat acquired the ansible tool in 2015.
- Ansible is available for RHEL, Debian, CentOS, Oracle Linux.
- Can use the tool whether your servers are in on-premises or in the cloud.
- It turns your code into infrastructure i.e. your computing environment has same as the as your application.

## Advantages of CM Tool

## Practice (1)

**Step①** Create EC2 instance like Ansible server

Two instances are created : Ansible Node-1

Ansible Node-2

**Step②** When these instances is ready we can assess these 3 instances via putty/Mobaxterm.

**Step③** In Ansible server

```
~]$ sudo su
```

**Step④** [ec2-user] # wget https://dl.fedoraproject.org/pub/epel/

epel-release-latest-7.noarch.rpm

**Step⑤** [ec2-user] # ls

epel-release-latest-7.noarch.rpm

on Ansible epel-release

search and find the

**Step⑥** [ec2-user] # yum install

commands.

**Step⑦** Is this OK [y/n]: y

complete

**Step⑧** [ec2-user] # yum update -y

complete

**Step⑨** [ec2-user] # yum install git python python-level  
python-pip ansible -y

Installing

→ | →

→ | →

→ | →

→ | →

Step 10 ec2-user] # ansible --version  
ansible 2.9.27

Step 11 ec2-user] # vi /etc/ansible/hosts ← we add group in  
press ? ← host file means add  
# - } nodes private ip add. in

# Ex 1 : -----

[demo]

node1:ip (private)

node2:ip (private)

} create group and copy private ip  
of nodes and paste here.

#

esc

:wq

exit

Step 12 ec2-user] # vi /etc/ansible/ansible.cfg

#

#

#

↑ ↑  
press i for editing  
file

# some basic default values

terminal  
means  
uncommented

# inventory = /etc/ansible/hosts

-

-

-

remove  
uncommented

# sudo\_user = root

esc (ctrl + z) → terminal not broaching [oblig]

:wq → file's content written to terminal

Step 13) `ec2-user] # useradd ansible` ← { add user in server, node1, node2 and  
`ec2-user] # passwd ansible` ← { give to passwd for this user,  
 new password : Admin  
 Retype password : Admin  
 passwd : all authentication tokens updated successfully.

Step 14) In Ansible Node1

`[~] $ sudo su`  
`[ec2-user] # adduser ansible`  
`[ec2-user] # passwd ansible`  
 new password : Admin  
 Retype password : Admin  
 passwd : all authentication token updated successfully.

Step 15) same as Ansible Node1 and Ansible-server in Ansible Node2

Step 16) `ec2-user] # su - ansible`

`[ansible@ip -] $ touch file1`

`[ansible@ip -] $ file1`

`[ansible@ip -] $ yum install httpd -y`

msg: you need to be root to perform this command.

Step 17) `ansible@ip -] # sudo yum install httpd -y`

We trust you have received the usual lecture from the local system administrator.

`[sudo] password for ansible :` (enter password)  
 - ansible is not in the sudoers file.

Step 18 Give sudo rights to ansible user (sudo privileged) to users

root ~] # visudo

==

==

partial sudoers

==

Allow root to run any commands anywhere

root ALL=(ALL) ALL

ansible ALL=(ALL) NOpassword & ALL

==

esc

:wq

Step 19 Give sudo privileged to node1 and (node 2)

root ~] # visudo

==

==

Allow root to run any commands anywhere

root ALL=(ALL) ALL

ansible ALL=(ALL) NOpassword & ALL

==

esc

:wq

Step 20 sudo privileged to Node 2

same as node1 and ansible server

Step 21 Now we have to install any software in user's giving sudo permission:

ec2-user# su - ansible

Step 22 [ansible@ip-] \$ yum install httpd -y  
 you need to be root user to perform this action.

Step 23 [ansible@ip-] \$ sudo yum install httpd -y  
 install httpd

Step 24 set up communication (SSH connection establish) :

: Login ansible users in all three machine

: Login Ansible server

[root-user] # su - ansible

[ansible@ip-] \$ ssh (node1 ip address)

permission denied.

: make a configuration  
 and do some change in a root user.

Step 25 [ansible@ip-] \$ exit

logout

[root@ip- ec2-user] # vi /etc/ssh/sshd-config

uncomment

# permitrootlogin yes

# password authentication no  
 committed

Step 26 Same do changes in node1 and node2 as a root user.

**Step 27** Restart service sshd - for change ps done in all user's including servers.

- ansible servers : service sshd restart
- root users

- Node 1 - root - service sshd restart
- Node 2 - root - service sshd restart

**Step 28** [ec2-user] # su - ansible

[ansible@ip - ] # ssh (node1 private ip address)

- Make changes or create files and folder in node1 via Ansible Server.

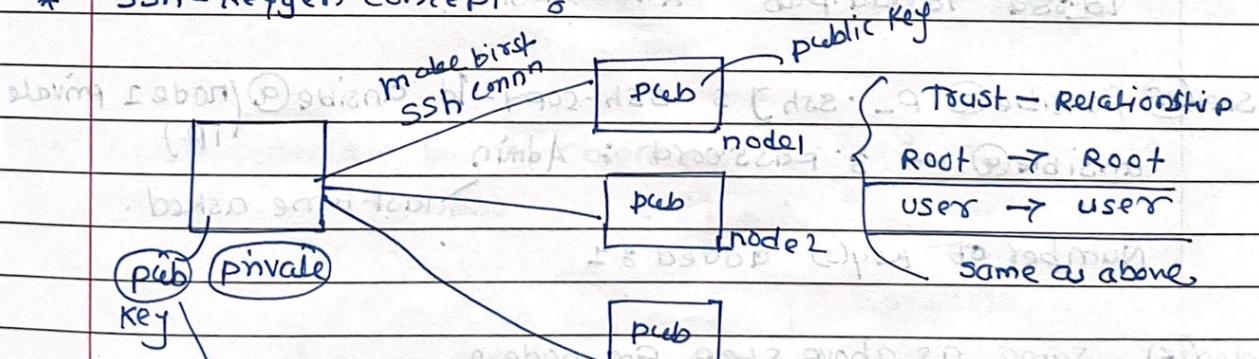
- You made changes in Ansible Server you can see in node1.

**Step 29** Follow same in Ansible User for node 2.

[ansible@ip - ] # ssh (node2 private-ip address)

Make changes. and check in node user.

**Step 30** \* SSH - Keygen concept



This public key copy and paste in all interconect-nodes  
In case 1000 of node is available and everytime  
is not possible to login via password that's why ssh-Keypair concept is learn :-

: login as ansible server, node user 1, node user 2  
as a ansible server.

: Now go to ansible server and create keys Run this command as ansible user.

### Step 31 In ansible server

[ansible@ip-~] \$ ssh-keygen ssh-keygen SSH-Keygen

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

The key fingerprint is:

} key is generated

### Step 32 [ansible@ip-~] \$ ls -a

.. .bashrc ~ - - - - - ssh

### Step 33 [ansible@ip-~] \$ cd ssh/

### Step 34 [ansible@ip-~] \$ ls

id\_rsa id\_rsa.pub known\_hosts

### Step 35 [ansible@ip-~] \$ ssh-copy-id ansible@(node1 private IP)

ansible@ip-~ % password : Admin

last time asked.

Number of key(s) added : 1

### Step 36 Same as above step for node 2

Number of key(s) added : 1

Step 37 [ansible @ ip-172-16-1-10 ~] \$ cd .. & [root@ip-172-16-1-10 ~]

[ansible server] \$ ssh (node1 ip private)  
 \* (we no need to add password right now)

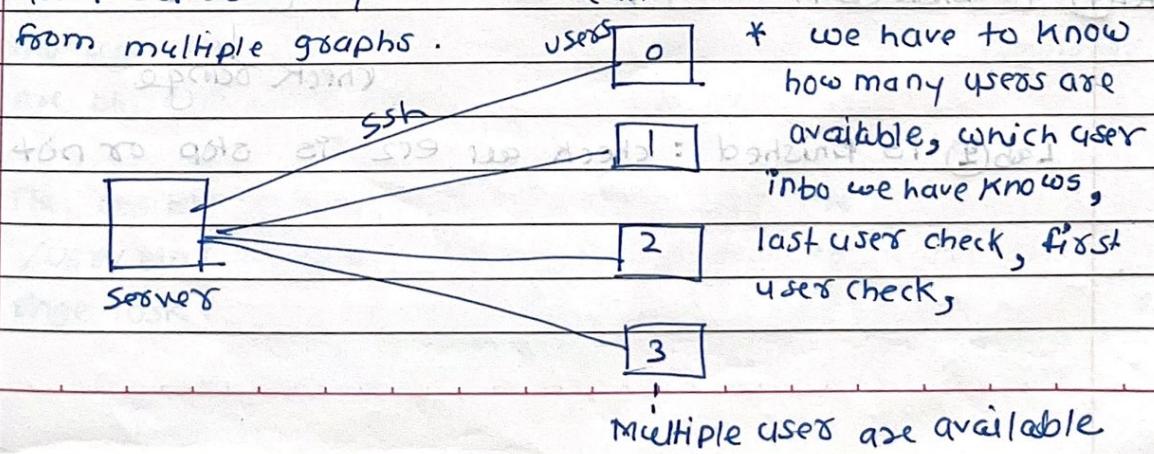
Step 38 [ansible server] \$ ssh (node2 ip private)  
 \* we no need to add password  
 registers ansible node1 and 2 without password.

### Host patterns

"all" pattern refers to all machine in a inventory

- ansible all --list-hosts
- ansible <group name> --list-hosts
- ansible <group name>[0] --list-hosts
- groupname[0] → pick first machine of group.
- group name[1] → pick second machine of group.
- groupname[-1] → picks last machine of group.
- groupname[1:4] → picks first two machine in this group
- groupname[2:5] → picks 3, 4, 5 and 6 machine in the group

- Group separate by a colon can be used to use hosts from multiple groups.



Step 39 [ansible @ ip ~] \$ ansible all --list-hosts

hosts(2) :

ip address node1 } check  
ip address node2 } how many host are available

Step 40 [ansible @ ip ~] \$ ansible demo b1 --list-hosts

hosts(2) :

} In case we have two group then see two groups in this command.

Step 41 [ansible @ ip ~] \$ ansible demo[0] --list-hosts

hosts(1) :

ip address node1 .

Step 42 [ansible @ ip ~] \$ ansible demo[1] --list-hosts

host( ) :

ip address node2 .

[0] smart quote .  
[1] smart quote .  
[2] = not available .

Step 43 [ansible @ ip ~] \$ ansible demo[-1] --list-hosts

↓  
last node seen.

Step 44 [ansible @ ip ~] \$ ansible demo[0:1] --list-hosts

check range

Lab 2 is finished : check all ec2 is stop or not

## ee Ad-hoc Commands, Module & playbook

- Ad-hoc commands (simple linux command)
- | L No idempotency (means Repeation & override)
  - (Temporary) Ad-hoc not know because of that environment so Ad-hoc do repeated task and override same work.

Language

yaml Module - only single command run or do single work.

playbook - more than one module called playbook

L module combination

ex:	Single work i.e module	module1 - install httpd
		module2 - start httpd
		modules - webserver

write YAML script for this module to run all i.e playbook.

## ee Ad-hoc commands

- Ad-hoc commands are commands which can be run individually to perform work operation (function).
- These ad-hoc commands are not used for configuration Management and deployment, because these commands are of one time usage.
- The ansible ad-hoc commands uses the /usr/bin/ansible command line tool to automate a single task.

### Practise § Ad-hoc command

Page No.

Date

Start three instances and access these instances via Putty

mobaXterm 9.0.0000 (www.mobaxterm.com) -

AP-956 Use of Kuhn Procedure to Determine T

Ansible Services - Machine 1

Visible Starts - Magne 1

## Ansible Node1 - Machine 2

Ansible Node2 - Machine 3

## Invisible server

```
ec2-user@ip-172-31-10-10: ~]# sudo su - ansible
```

```
[ansible@ip-~] $ ls
```

## ① command

```
[ansible@ip-~] $ ansible demo -a "ee1s"
```

group node information shown here

node 1 :

node 2 :

② [ansible ② ip -n] \$ ansible all -a "ee1s?"

: all group and group nodes information shown here

③ [ansible@ip-~] \$ ansible all -a "echo touch file1" |

~~1. I would like to thank the author for his/her permission to use this manuscript.~~

one all node and group.

: do ls and verify file1 is created or not on all nodes

specification of root soil biomass ratios

we can run this command one more than one - show

Changed means this is Idempotency.

- (4) [ansible@ip-~] \$ ansible demo -a "ls -al" →  
 • all details : all hidden file under ←  
 with hidden file → nodes will seen here.
- (5) [ansible@ip-~] \$ ansible demo -a "sudo yum install httpd -y"  
 means install httpd server on ← group demo and these nodes.  
 verify s/w installed or not → which httpd.
- (6) for remove s/w on all nodes (group) :  
 [ansible@ip-~] \$ ansible demo -a "sudo yum remove httpd -y"  
 ansible demo -bq "yum remove httpd -y" →  
 verify s/w remove or not ; which httpd.

### Ansible Modules

- Ansible ships with a lot of modules (called 'module library') that can be executed directly on remote hosts or through playbook.
- your library of modules can reside on any machines, and there are no servers, daemons, or database required.
- Where Ansible modules are stored → default location for the inventory file is  
 → /etc/ansible/hosts

Practice 8 → install = present  
 → uninstall = absent  
 → update = latest

Module is written in XML script

→ XML →  
 Ansible uses its own language

all run on ansible  
server → served machine

sudo privileged  
module name  
Date \_\_\_\_\_  
Page No. \_\_\_\_\_

- [ansible@ip-~] \$ ansible demo -b -m yum -a

ee pkg=httpd state=present "

s/w name

install

Verify httpd install or not : which httpd.

- If above command run more then msg: already run by you

- (green colour) -

- (blue colour) -

[ansible@ip-~] \$ ansible demo -b -m yum -a

ee pkg=httpd state=latest "

↓  
for update pkg .

- [ansible@ip-~] \$ ansible demo -b -m yum -a

ee pkg=httpd state=absent "

for pkg remove in group (older)

Install once again httpd s/w by run 1st command . at time

- [ansible@ip-~] \$ sudo service httpd status

Check service is start or not

- [ansible@ip-~] \$ ansible demo -b -m service -a

ee name=httpd state=started "

Verify & sudo service httpd status = Started

- [ansible@ip-~] \$ ansible demo -b -m user -a

ee name=xyz "

Check in node machine

- [ansible @ ip - ~] \$ touch copyfromserver  
[ ] ls → Ansible has created new entries in inventory.  
• copied fromserver ←
- [ansible @ ip - ~] \$ ansible demo -m setup [-l] -b -m copy -a  
“src = copied from server dest = /tmp”  
↓ sourcefilename   ↓ destinationfilename
- [ansible @ ip - ~] \$ touch file1 → ls → file2  
[ansible @ ip - ~] \$ ansible demo -b -m copy -a -f all  
“src = file1 dest = /tmp”  
verify? ls /tmp
- **setup module**

[ansible @ ip ~] \$ ansible demo -m setup  
at been see what information we go tell - nothing ↗ all info about  
[ansible @ ip ~] \$ ansible demo -m setup ↗ all node information.  
-a “filter = \*IPv4\*”  
↳ (Appended question mark by) ↗ only IP address related information of node.

↳ (Answer - part 1) to tell + at tell all in main doc. Every node  
↳ (Answer - part 2) command can be run in shell.

“...” → All commands that have been told in MAX HA  
“...” as All the one by

## Playbook

Page No.

Date

- Playbook in ansible are written in **YAML** format
- It is human readable data serialization language
- It is commonly used for configuration files.
- Playbook is like a file where you write codes.  
consists of vars, tasks, handlers, files templates and roles.
- Each playbook is composed of one or more **modules** in a list module is a collection of configuration files.
- Playbooks are divided into many sections like -

Target section - Defines the host against which playbooks task has to be executed.

Variable section - Define variables

Task section - List of all modules that we need to run, in an order.

\*

## YAML Basic's (Yet Another Markup Language)

- For ansible, nearly every YAML files starts with a list.
- Each item in the list is a list of **Key-value** pairs commonly called a dictionary.
- All YAML files have to begin with **---** and end with **...**.

- All members of a list-line must begin with some indentation level starting with `" "`

for eg: --- # A list of fruits

fruits:

- Mango

- strawberry

- Banana

- Grapes

- Apple

... (end)

writing in key value form

Syntax: Name: Bhupinder

job: SQL Developer

playbook.yml file looks like

- A dictionary is represented in a simple `key: value` -

From

key: value -

- for eg: --- # details of customer

customer:

name: Akash

job: Trainer

skill: Ansible

Exp: 8 years

extension of playbook file is

.yml

Note: - There should be space between : and value.

**Practical 6****Playbook 1**

**Step 1** Login EC2 Instance via Putty/Mobaxterm.

**Step 2** Go to ansible server

~] \$ sudo su

ec2-user ] # su - ansible

[ansible@ip-~] \$ ls

**Step 3** Pb any file here

then remove rm -rf (filename)

Now Create one playbook

**Step 4** ansible@ip-~] \$ vi target.yml

press i and write small

playbook for our understanding

**Step 5** --- # My First Testing YAML playbook

- hosts: demo <sup>for group</sup>

- users: ansible <sup>represent</sup>

- become: yes <sup>for sudo privileged</sup>

- connection: ssh <sup>by default</sup>

- gather-facts: yes

esc

:wq

**Step 6** Now, to execute this playbook

[ansible@ip-~] \$ ansible-playbook target.yml

execution commands.

O/P ---

PLAY

TASK

PLAY RECAP

## playbook(2)

Step(1) Now Create one more playbook in ansible server :

```
[ansible @ ip - ~] $ ansible task.yml
```

-- # Target and Task in playbook separate in different  
- host: demo  
 user: ansible  
 become: yes  
 connection: ssh

tasks:  
 - tasks: Install httpd on Linux  
 action: yum name=httpd state=installed

Step(2) Esc & wq!

Step(3) Now, execute this playbook

```
[ansible @ ip - ~] $ ansible -i playbook task.yml
```

Step(4) O/P → play [demo] \*\* ...

task: TASK [ Gathering facts ] \*\*

TASK [ Install HTTPD on Linux ]

action: PLAY RECAP [ demo ] :

demo : 100% | [ ] = 0m 0s

## "variables"

- Ansible uses Variable which are defined previously to enable more flexibility in playbooks and roles. They can be used to loop through a set of given values, access various information like the hostname of a system and replace certain strings in templates with specific values.
- put variable section above task so that we define it first and use it.

### playbook(3)

NOW goto ansible server and create one playbook,

```
[ansible@ip] $ vi vars.yml
--- # My variable playbook
hosts: demo
user: ansible
become: yes
connection: ssh

vars:
  pkgname: httpd

tasks:
  - name: install httpd server on Linux
    action: yum name = "{{ pkgname }}". state=installed
```

esc

:wq!

NOW execute playbook

```
[ansible@ip] $ ansible-playbook vars.yml
```

## Handlers Section

- A handler is exactly the same as a task, but it 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 also indicates that it changed something.

### Playbook 4

Go to ansible server:

```
[ansible@ip] $ vi handlers.yml
```

# Handler playbook  
- hosts: demo  
 user: ansible  
 become: yes  
 connection: ssh

#### Tasks:

```
- name: Install httpd server  
  action: yum name=httpd state=installed  
  notify: restart HTTPD
```

#### Handlers:

```
- name: restart HTTPD  
  action: service name=httpd state=restarted
```

```
[ansible@ip] $ ansible-playbook handlers.yml
```

**Day 10**

check whether the playbook is formatted correctly.

ansible-playbook handlers.yml --check

**"Loops"**

- Sometimes you want to repeat a task multiple times.
- In computer programming, this is called loops.
- common Ansible loops including changing ownership on several files and/or directories with the file module, and repeating a polling step until certain result is reached.

**playbook (5)**

Now go to ansible server

[ansible@ip] \$ vi loops.yml

dear : oohhooooo

oldman : 3926

298 : sonoooo

343 : dadoooo

343 : momoooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

343 : husbandoooo

343 : wifeoooo

343 : sonoooo

343 : daughteroooo

343 : brotheroooo

343 : sisteroooo

343 : friendoooo

343 : girlfriendoooo

343 : boyfriendoooo

esc → ⌘ wq! save & exit current buffer + quit : know -

[ansible@ip] \$ ansible-playbook loops.yml ; echo -

To verify, go inside Node 1 [root@ip ~]

[ ] cat /etc/passwd

[ ] cat /etc/passwd

### Condition

- Whenever we have different different scenarios, we put condition's according to the scenario.

When statement

something you want to skip a particular command on a particular node.

### playbook (6)

--- # condition playbook

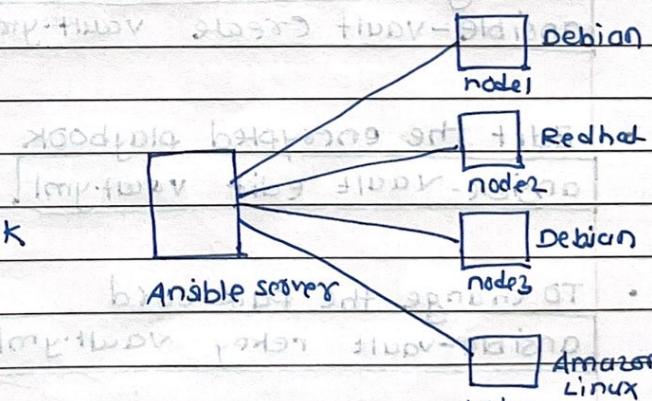
- host: demo

user: ansible

become: yes

connection: ssh

task:



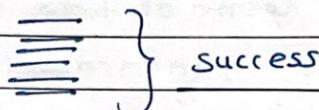
several

```

  - name: install apache on debian family
  module  command: apt-get -y install apache2
  condition ~ when: ansible_os_family == "Debian"
  - name: install apache for fedora family
  module  command: yum -y install httpd
  condition ~ when: ansible_os_family == "RedHat"

```

[ansible @ ip] \$ ansible-playbook condition.yml



## Vault

- Ansible allows keeping sensitive data such as passwords or keys in encrypted files, rather than a plaintext in your playbooks.
- Creating a new encrypted playbook  
ansible-vault create vault.yml
- Edit the encrypted playbook  
ansible-vault edit vault.yml
- To change the password  
ansible-vault rekey vault.yml
- To encrypt an existing playbook  
ansible-vault encrypt target.yml

decrypt

To default an encrypted playbook

ansible-vault decrypt target.yml

- for creating vault file

ansible-vault create target.yml

password

open playbook editor

playbook

play

edit

target

data

check vault.yml → vi vault.yml

to see encrypted data

esc → :wq!

{ edit → same as above → edit command (use)

change password → same as above → (use change password command)

encrypt existing playbook → use encrypt command

decrypt existing playbook → use decrypt command

20108

20108-20109

20109

20109

(target) 20109

20109

20109

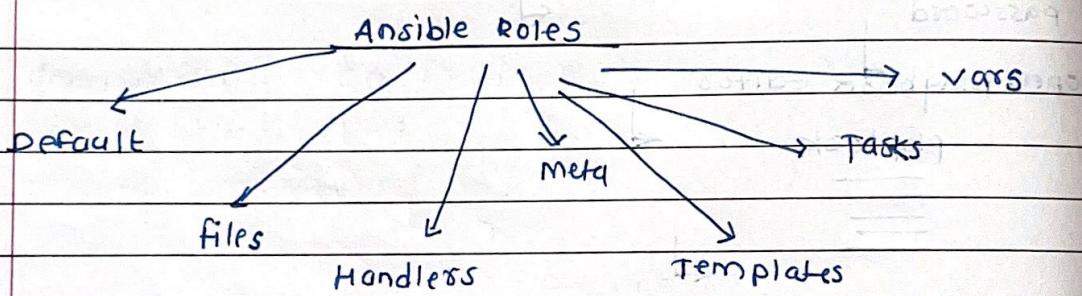
target 20109

20109

20109

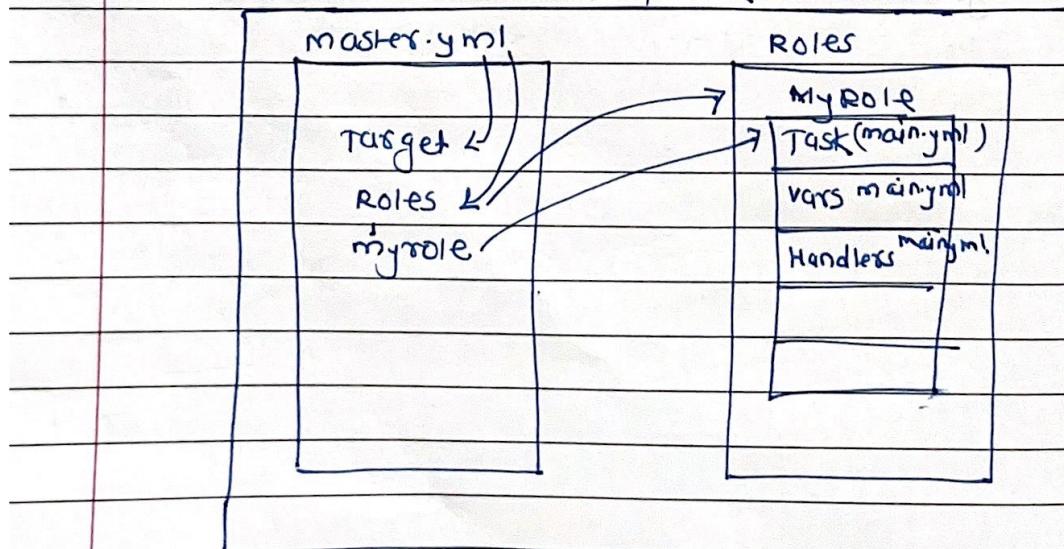
## Roles

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



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

### Playbook



- Default :- It stores the data about role/application default variable. e.g - If you want to run to port 80 or 8080 then Variable needs to define in this path.
- Files :- It contains file 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 contains files that establish roles dependancies e.g Author Name, supported platform, dependencies if any.
- Task :- It contains all the tasks that is normally in the playbook. e.g installing packages and copies files etc.
- vars :- Variable for the role can be specified in this directory and used in your configuration files. Both vars and defaults stores variable.

task name in [ai@client]

variable in vars

variable in defaults for modification

Ansible

Ansible

Ansible

Ansible

## Practical 8 - Roles

[ansible@ip] \$ mkdir -p playbook/roles/webserver/task

[ansible@ip] \$ tree  
eg:  
playbook/roles/webserver/handlers  
o/p → Playbook

└→ roles

└→ webserver

└→ task

[ansible@ip] \$ cd playbook

[ansible@ip] \$ tree

O/P → playbook

└→ roles

└→ webserver

└→ tasks

[ansible@ip] \$ touch roles/webserver/tasks/main.yml

[ansible@ip] \$ touch master.yml

← run tree command to verify

[ansible@ip] \$ vi roles/webserver/tasks/main.yml

— name: install apache on RedHat

yum: pkg=httpd state=latest

esc : wq  
exit

[ansible@ip] \$ vi master.yml

← Playbook master

--- # master playbook for web servers

- host: demo

user: ansible

become: yes

connection: ssh

roles:

~~roles~~

- webserver

esc → g wq  
exit ↴

[ansible@ip - playbook]\$ ansible-playbook master.yml

O/P

|||||