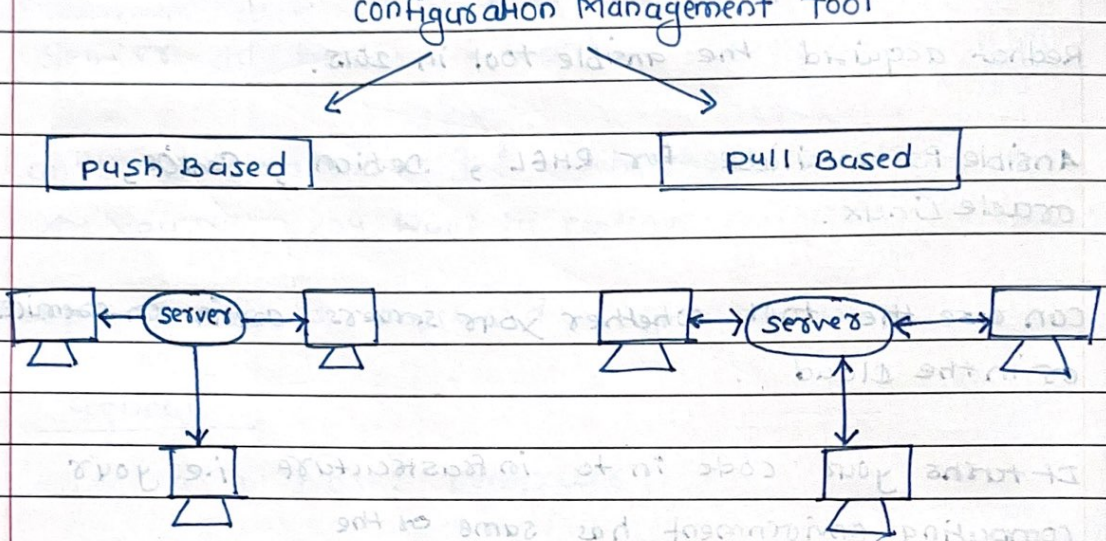
Configuration Management Tool

- Each and every minute detail of your Machine (server, storage) delete, update, create
  configuration Management Tool part - operation part in DevOps.



- push configuration server pushesh configuration to the node.

  ex: Ansible
  Saltstack

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

  ex: Chef
  puppet

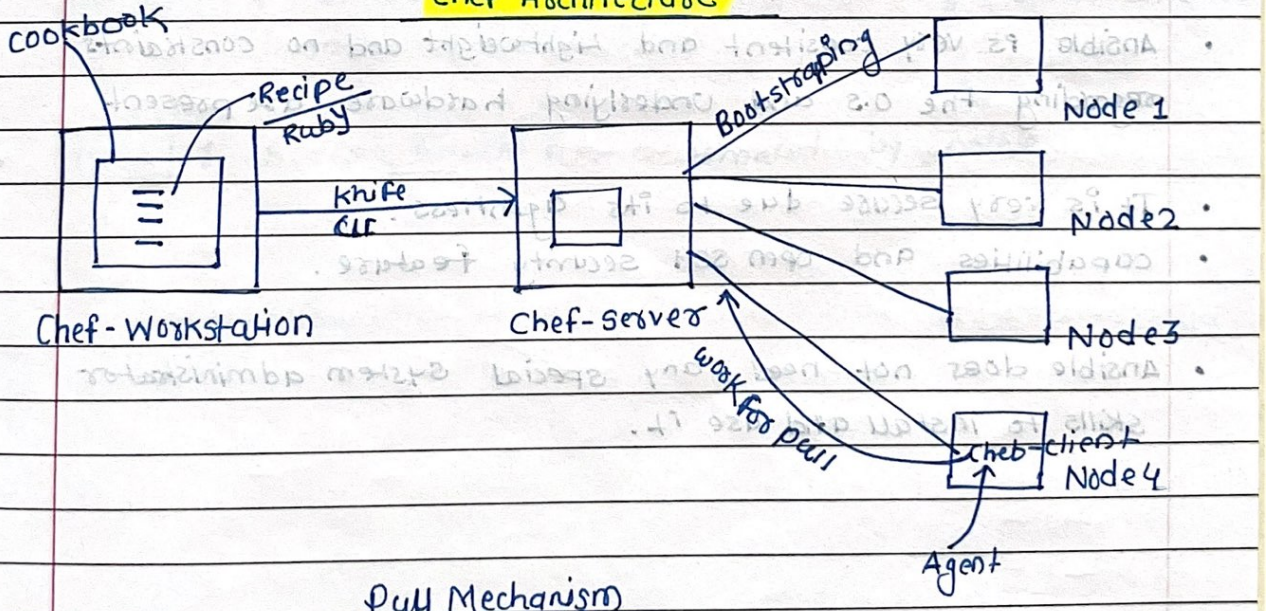
- **Advantage of CM Tool**

  Complete Automation
  Increase Uptime
  Improve performance
  Ensure compliance
  Prevent Errors
  Reduce cost

Page No.
Date

- 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 task 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.

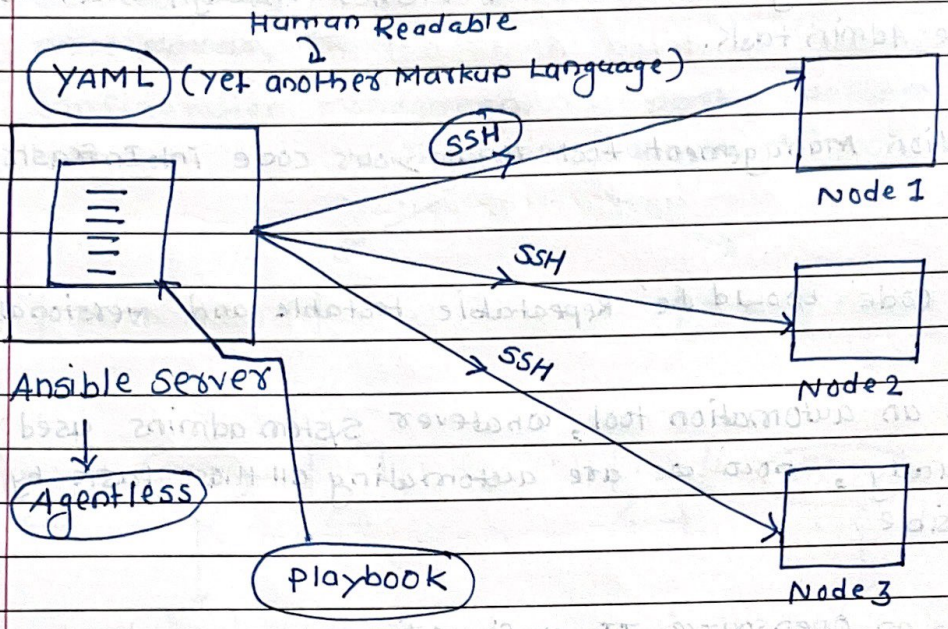
Chef Architecture



Pull Mechanism

Page No.
Date

Ansible Architecture.



Advantages

- Ansible is free to use by everyone.
- Ansible is very consitent 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.

Page No.
Date

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 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.

Page No.
Date

Ansible History

- Michael Dehaan developed Ansible and the ansible 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 in to infrastructure i.e your computing enviornment has same as the as your application.

Page No.
Date

Practicle (1)

step① create EC2 instance like • Ansible server
• Ansible Node-1
• Ansible Node-2

step② When these instances is ready we can assess these 3 instances via putty/MobaXtream.

step③ In Ansible server
↓
~]$ sudo su

step④ ec2-user]# wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
↓
you can download on Ansible epel-release search and find the commonds.

step⑤ ec2-user]# ls
epel-release-latest-7.noarch.rpm
↓

step⑥ ec2-user]# yum install

step⑦ Is this OK [y/d/N]: y
complete

step⑧ ec2-user]# yum update -y
complete

step⑨ ec2-user]# yum install git python python-level python-pip ansible -y

Installing
—|—
—|—
—|—
—|—

Page No.
Date

Step (10) ec2-user] # ansible --version ↵

ansible 2.9.27

Step (11) ec2-user]# vi /etc/ansible/hosts ← we add group in host file means add nodes private ip add. in host/server file. For accessing the nodes via server.

press i

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 ....

remove means uncommented (#) inventory = /etc/ansible/hosts

remove uncommented (#) sudo_user = root

esc

:wq

Page No.
Date

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

Step 14 In Ansible Node1 ↓

N] $ 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-] $ ls
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.

Page No.
Date

step 18 Give sudo rights to ansible user (sudo priveladged) to user's

ec2-user]# visudo

Allow root to run any commands anywhere

root ALL=(ALL) ALL

ansible ALL=(ALL) NOPASSWD: ALL (space)

esc

:wq

step 19 Give sudo priviledged to node1 and (node2)

~]$ sudo su

ec2-user]# visudo

root Allow root to run any commands anywhere

root ALL=(ALL) ALL

ansible ALL=(ALL) NOPASSWD: ALL

esc

:wq.

step 20 sudo priviledged to Node2

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

Page No.
Date

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 user in all three maschine
: Login Ansible server
[root@ip-user] # su - ansible
[ansible@ip-] $ ssh (node1 ip address)
permission denied.
: make a configuration
and do some change as a root user.

Step(25) [ansible@ip-] $ exit
logout
[root@ip-ec2-user] # vi /etc/ssh/sshd_config

uncomment
(#) PermitRootLogin yes
Password Authentication no
commented

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

Page No.
Date

Step 27 Restart service sshd - For change ps done in all user's including server.

- ansible server : service sshd restart
  root user
- 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)
copy from AWS console

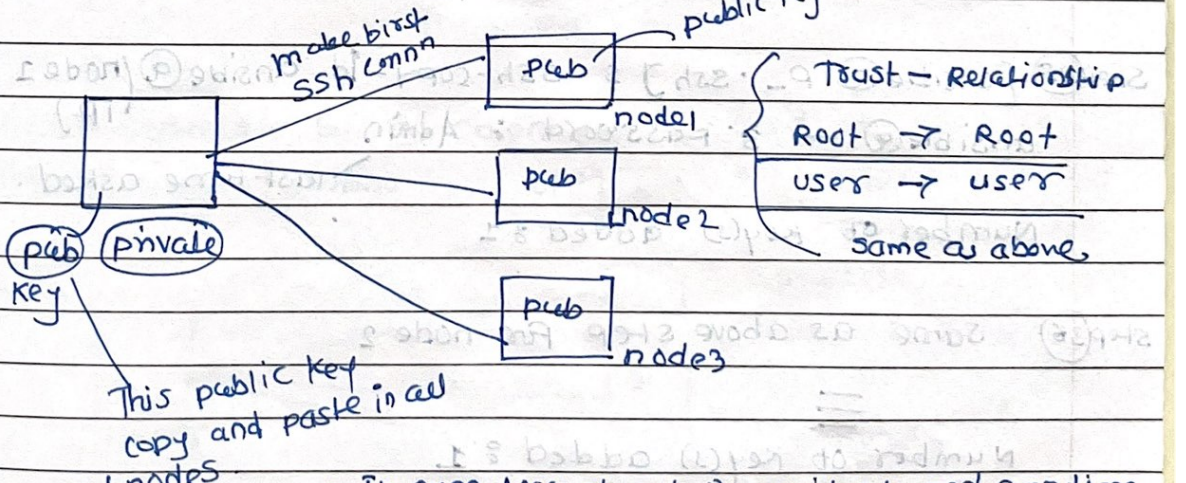
- Make changes or create files and folder in node 1 via Ansible server.
- You made changes in Ansible server you can see in node 1.

Step 29 Follow same in Ansible user for node 2.

[ansible@ip-] # ssh (node2 private-ip address)
copy from AWS console

Make changes. and check in node user.

Step 30 * ssh-keygen concept :



: In case 1000 of node is available and everytime is not possible to login via password that's why ssh-keygen concept is learn :

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

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

step(31) In ansible server

[ansible@ip-~] $ ~~sshkeygen~~ ~~sshkeygen~~ ssh-keygen

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

The key fingerprint is:

} key is genarate

step(32) [ansible@ip-~] $ ls -a

. .. .bash — — — — .ssh

step(33) [ansible@ip-~] $ cd .ssh/

step(34) [ansible@ip-.ssh] $ ls

id_rsa id_rsa.pub know_hosts

step(35) [ansible@ip_.ssh] $ ssh-copy-id ansible@ (node 1 private IP)

ansible@ — 's 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-...ssh] $ cd ..

[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 4 regerss ansible node 1 and 2 without password.

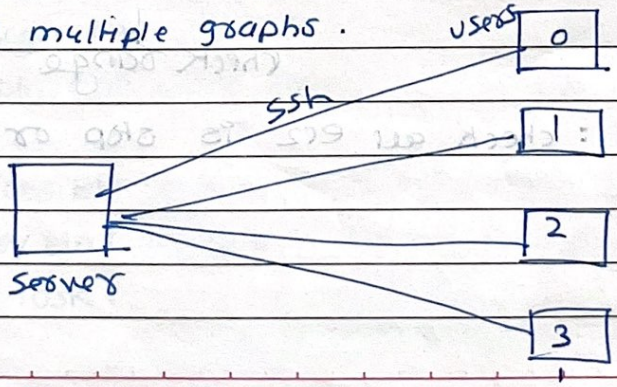
"Host patterns"

"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.
- group name [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 graphs.



* we have to know how many users are available, which user info we have knows, last user check, first user check,

Multiple user are available

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

hosts (2):
ip address node1
ip address node2
} check how many host are available

Step 40 [ansible@ip-~] $ ansible demo --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 (1):
ip address node2.

[2] — not available

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

↓ last node seen.

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

↓ check range

Lab 1 is finished : check all ec2 is stop or not

Page No.
Date

Ad-hoc Commands, Module & playbook.

Ad-hoc commands (simple linux command
- (Temporary)
- No Idempotency (means Repeatation & overide Ad-hoc not know because of that enviornment so Ad-hoc do repeated task and overide same work).

Language YAML

Module - only single command run or do single work.

playbook - more than one module called playbook
- module combination

ex: Single work i.e module

module1 - install httpd
module2 - start httpd
module3 - webserver

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

Ad-hoc commands

- Ad-hoc commands are commands which can be run individually to perform quick 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.

Page No.
Date

Practicle : Ad-hoc command

Start three instances and access these instances via putty/ mobaXtream.

- Ansible server - Machine 1
- Ansible Node 1 - Machine 2
- Ansible Node 2 - Machine 3

In ansible server

~]$ sudo su

[ec2-user]# su - ansible ← go to user

[ansible@ip-~] $ ls

- - - - -

① command

[ansible@ip-~]$ ansible demo -a "ls" (argument, do execute)

: group node information shown here

node 1 :

node 2 :

② [ansible@ip-~]$ ansible all -a "ls"

: all group and group nodes information shown here.

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

: used this command for create file 1 one all node and group.

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

: we can run this command one more then o/p show changed means this is idempotency.

④ [ansible@ip-~]$ ansible demo -a "ls-al"

- all details with hidden file
- all hidden file under nodes will seen here.

⑤ [ansible@ip-~]$ ansible demo -a "sudo yum install httpd -y"

means install httpd server on group demo and there 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 -ba "yum remove httpd -y" (or)

verify s/w remove or not : which httpd.

"Ansible Modules"

- Ansible ships with a number 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

Practicle :
- install = present
- uninstall = absent
- update = latest

(XAML)

Module : write in XAML script.

all run on ansible server — server machine — sudo privileged — module — service module name

- [ansible@ip-~]$ ansible demo -b -m yum -a "pkg=httpd state=present"
  - pkg = s/w name
  - present = install
  - verify httpd install or not : which httpd.
- If above command run more then msg: already run by you
  - (green colour)
- [ansible@ip-~]$ ansible demo -b -m yum -a "pkg=httpd state=latest"
  - ↓ for update pkg.
- [ansible@ip-~]$ ansible demo -b -m yum -a "pkg=httpd state=absent"
  - ↓ for pkg remove in all group (nodes)
- Install once again httpd s/w by run 1st command.
- [ansible@ip-~]$ sudo service httpd status
  - Check service is start or not ←
- [ansible@ip-~]$ ansible demo -b -m service -a "name=httpd state=started"
  - verify : sudo service httpd status.
- [ansible@ip-~]$ ansible demo -b -m user -a "name=xyz"
  - Check in node machine ←

Page No.
Date

- [ansible@ip-~]$ touch copyfromserver
  [———]$ ls
  copied fromserver —
- [ansible@ip-~]$ ansible demo ~~almost~~ [-1] -b -m copy -a
  "src = copied fromserver dest = /tmp"
  ↓ source filename ↓ destination filename
  verify :
- [ansible@ip-~]$ touch file2 → ls ⇒ file2
  [ansible@ip-~]$ ansible demo -b -m copy -a
  "src = file1 dest = /tmp"
  verify : ls /tmp
  in node
- **setup module**

  [ansible@ip~]$ ansible demo -m setup ← all info about all node information.
  [ansible@ip~]$ ansible demo -m setup
  -a "filter = *ipv4*"
  ↓ only ip address related information of node.

Page No.
Date

"Playbook"

- 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 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 Basics (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 "...".

Page No.
Date

- All members of a list lines must begin with same identation level starting with "- "

- for eg:
```
---  # A list of fruits
fruits:
  - Mango
  - Strawberry
  - Banana
  - Grapes
  - Apple
... (end)
```

Syntax:
```
Name: Bhupinder
job: SQL Developer
```
(key → Name, value → Bhupinder)

playbook.yml ← file save like

- A dictionary is represented in a simple key: value (space) form

- 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.

Page No.
Date

Practicle :

Playbook (1)

Step(1) Login EC2 Instance via putty/MobaXtream.

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 ← for group represent
- user : ansible
- become : yes ← for sudo privileged
- connection: ssh ← by default
- 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

Page No.
Date

playbook ②

Step ① Now create one more playbook in ansible server :

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

Step ② --- # Target and Task playbook.
- host: demo
  user: ansible
  become: yes
  connection: ssh
  tasks: ↓ actually this space is here.
  - name: Install httpd on Linux
    action: yum name=httpd state=Installed
            ↳ module name

Step ③ esc : wq !

Step ④ Now, execute this playbook

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

Step ⑤ O/P → PLAY [demo] ***...
TASK [Gathering Facts] ***...
TASK [Install HTTPD on Linux]
PLAY RECAP [...]

Page No.
Date

"variables"

- Ansible uses variable which are debined previously to enable more flexibility in playbooks and roles. They can be used to loop through a set of given values, access various informormation like the host name 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 go to 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 "

Page No.
Date

- A handler is exactly the same as a task, but it will run when called by another task.

or

- 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
```

esc

:wq!

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

Page No.
Date :

"Dryrun"

check wether the playbook is formatted correctly.

:- ansible-playbook handler.yml --check
(dryrun command)

"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
(editor)

```
---  # My loops playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh

  task:
    - name: add a list of users
    user: name= "{{item}}" state=present
    with items:
        → Akash
        → sagar
        → Kabir
        → Rupali
        → Priya
```

esc → :wq!

[ansible@ip]$ ansible-playbook loops.yml

To verify, go inside Node 1

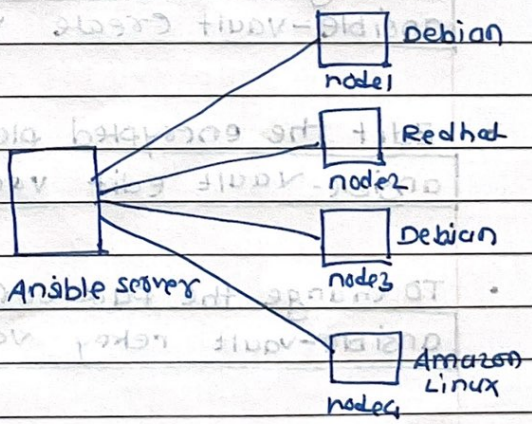
[ ] cat /etc/passwd

"Condition"

- Whenever we have diffferent diffferent scenarios, we put conditions according to the scenario.

When statement

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



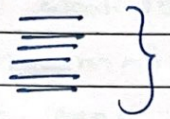
Playbook (6)

```
--- # condition playbook
- host : demo
  user : ansible
  become: yes
  connection: ssh

task:
```

- name: install apache servers on debian family
module → command: apt-get -y install apache2
condition → when: ansible_os_family == "Debian"
- name: install apache for redhat 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`

Page No.
Date

To decrypt an encrypted playbook

```
ansible-vault decrypt target.yml
```

- for creating vault file

```
ansible-vault create target.yml
password
open playbook editor
    playbook
    ===
    ===
```

check vault.yml → vi vault.yml

To see encrypted data

esc → :wq!

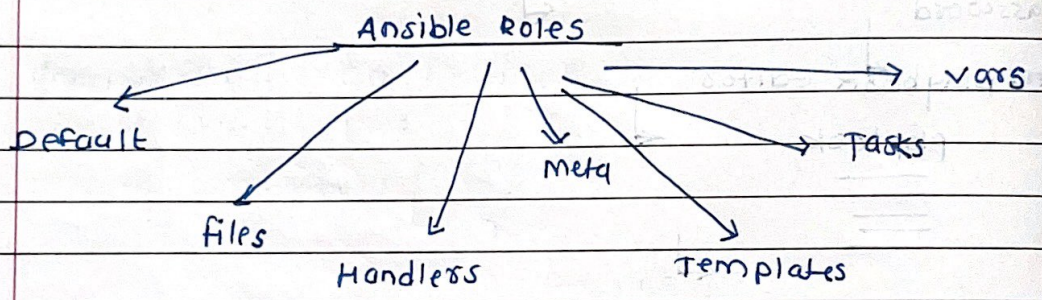
- edit → same as above → edit command (use)
- change password → same as above → (use change password command)
- encrypt existing playbook → use encrypt command.
- decrypt existing playbook → use decrypt command.

Page No.
Date

Roles

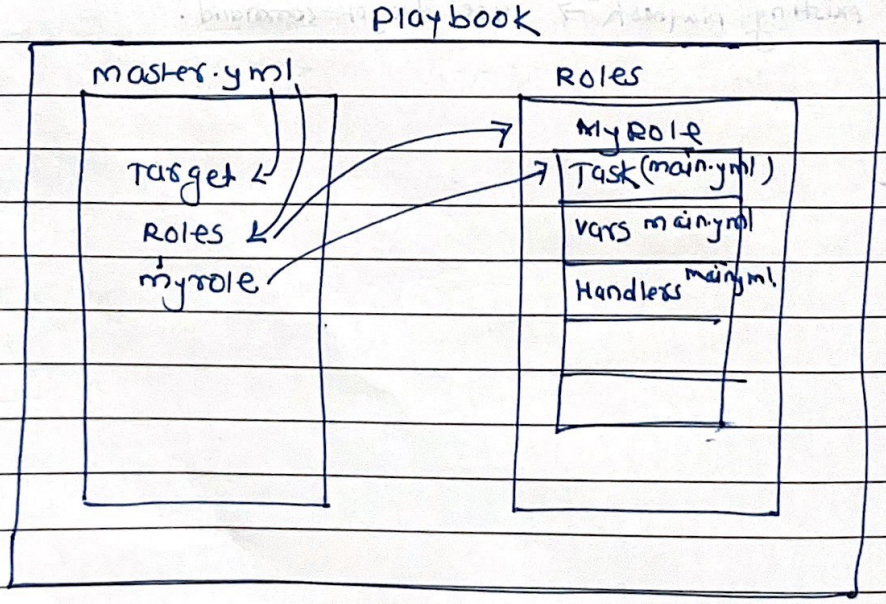
- We can use two techniques for reusing a set of tasks : includes and roles.

- Roles are good for organising tasks and encapsulating data needed to accomplish those tasks.



- We can organise playbooks in to a directory structure called roles.

- Adding more and more functionality to the playbook will make it difficult to maintain in a single file.



Page No.
Date

- Default :- It stores the data about role/application default variable. eg :- 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 transbered 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 eg → Author Name, supported platform, Dependancies if any.

- Task :- It contains all the tasks that is normally in the playbook. eg → 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 default stores variable.

Page No.
Date

Practicle :- 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
└→ task

[ansible@ip] $ touch role/webserver/task/main.yml

[ansible@ip] $ touch master.yml
↳ run tree command for 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 webservers
- host: demo
user: ansible
become: yes
connection: ssh

roles:

- webservers

esc → :wq

exit ↵

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

o/p