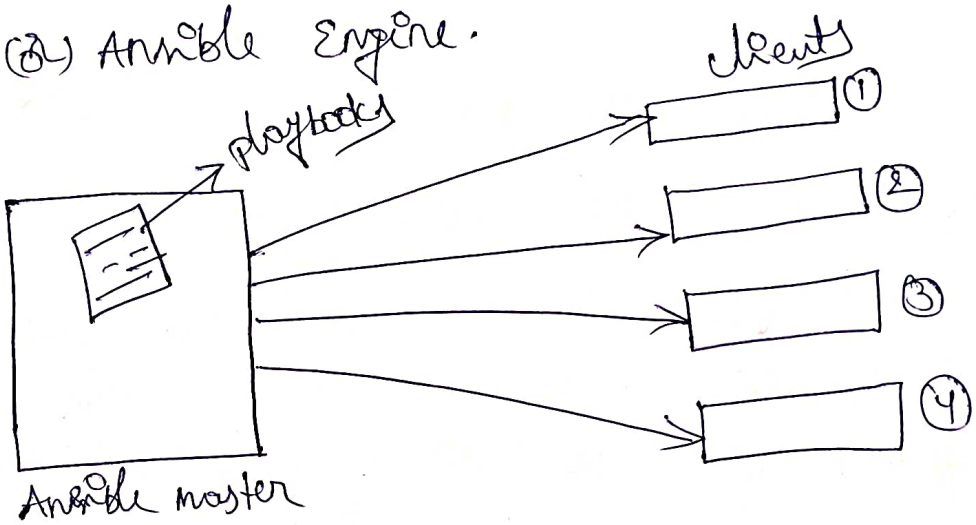
Configuration managment tools → Ansible, chef, PUPPET & Salt.

Ansible:- It is an open source IT automation engine that automates Provisioning Configuration managment, application deployment, orchestration & many other IT Process.

where the ansible is to installed that server we called as Ansible master (or) Ansible Engine.



Ansible is agentless server that means you not able install all packages in destination servers (clients). But cheff, Puppet & Salt are agent servers that means you able to install all Packages in destination servers.

Installation:-

Create two instances & Give tag names master & client. & Connected to a Putty.

Step(1):- ~~Sudo~~ Add the user in master & client server.

↳ sudo adduser master
↓
Give password & fullname (Y)

↳ sudo adduser client
↓
Give Password & fullname (Y)

Step②:- Config in ssh folder in master & client.

↳ sudo vi /etc/ssh/sshd_config

↓

PasswordAuthentication "yes"

↓

:wq!

↓

sudo service sshd restart

↳ sudo vi /etc/ssh/sshd_config

↓

PasswordAuthentication "yes"

↓

:wq!

↓

sudo service sshd restart.

Step③:- user can be a superuser in both master & client.

↳ sudo visudo

↓

user Privilege Specification

root ALL=(ALL:ALL) ALL

master ALL=(ALL:ALL) NOPASSWD: ALL

↓

Cntrl+x
Shift Y } Enter

↳ sudo visudo

↓

user Privilege Specification

root ALL=(ALL:ALL) ALL

client ALL=(ALL:ALL) NOPASSWD:ALL

↓

Cntrl+x
Shift Y } Enter.

Step④:- you need to create keys in new users (master & client) not be in normal users (ec2-user, ubuntu).

→ switching to new user.

↳ su master

↓

only in "master" not in home & ubuntu after "su master".

↓

generating key } ssh-keygen

↓

ls -la (shows Private & Publickeys)

↳ su client

↓

only in "client" not in home/ubuntu after "su client".

↓

~~ssh-keygen~~

↓

~~ls -la~~

↳ if you want to login client server from master through Password Authentication then easily.

from master } ↳ ssh client@IPv4 (Public or Pvt).

↓

Enter Password

↓

you are in client server

↓

Exit.

But we need Password by Authentication so follow,

Step ⑤ :- In client server (su client).

↳ mkdir .ssh

↓

cd .ssh

↓

sudo vi authorized_keys

↓

Paste the Public key of master ssh-keygen

↓

:wq!

for copying key from master to client :-

↳ ssh-copy-id client@IPv4 (Pvt or Public)

So now login to client server through PasswordlessAuthentication.

↳ from master } ↳ ssh client@IPv4 (Public or Pvt).

↓

you are in client server without entering the Password.

↓

ls -al shows all ~~keys~~ fully of client server.

↓

Exit (now you are in master).

Step⑥:- Install ansible Packages Search on google "how to install ansible on ubuntu". only in master Server.

↓

repository adding } → sudo apt-add-repository ppa:ansible/ansible.

↓

sudo apt update

↓

sudo apt install ansible

[ansible --version
python --version]

whatever new Packages installed on Server those all files Comes under in 'etc' directory

↓

cd /etc/ansible

↓

ls

[ansible.cfg, hosts, roles]

only group information will store in "hosts" file.
this host file location path is there in ansible.cfg file. we can change the location path of host file from etc to opt anywhere. but you need to Put location in ansible.cfg file.
& remove Comment[#] of inventory Paths in ansible.cfg file.
forks

Step⑦:- Create a group in hosts file

↓

vi hosts

↓

At last you can write

[myclient]

54.173.54.137 ansible_user=client

↓

client IP

↓

:wq!

Step⑧:- check client response from master server.

↳ ansible -m ping myclient

↓

response was coming (obviously Pong).

[Ping=Pong].

NOTE:- If you stop & start the instance the PublicIP's changes. So at the time you keep client servers IP's in hosts file group of master server.

↳ ansible client -m ping for checking.

But in realtime we are using Elastic IP's.

**:- only you can execute Adhoc commands in "su master" & client "su client".

**:- whatever Packages installing on server those all files goes to /etc directory.

Ad hoc Commands:- Ad hoc Commands are Commands which Can be run individually to Perform quick functions. you Can Control the destination Servers from master Server by using Ad hoc Commands.

Commands:-

① ansible client -m ping → test the client Server

② ansible client -m ping -O → test the client Server in one line.

③ ansible client -a "ls -l /opt" → checking the data in client Server

It means Every Server application data being in /opt directory.

④ ansible client -a "cat /etc/passwd" → listing users in client Servers.

⑤ ansible client -a "touch /opt/vasu" --become → 'vasu' file creating in client Server at in /opt directory.

(or)

~~⑥~~

ansible client -a "sudo touch /opt/vasu" → without using become you can use sudo. but it Sometimes not working

⑥ ansible client -m shell -a "hostname" → Hostnames shows in client server.

⑦ ansible client -m shell -a "df -h" → Shows the Server diskspace in human rlc.

⑧ ansible client -m copy -a "src=/opt/devops dest=/opt/" --become → copying the 'devops' file from master to client Server.

(9) ansible client -a "git --version" → checking the git version in client server.

(10) ansible client -m apt -a "name=maven state=Present" --become → Installing maven in client server.

(11) ansible client -a "mvn --version" → checking the maven version in client server.

(12) ansible client -m service -a 'name=apache2 state=stopped' --become
started "
restarted "
latest "
→ stop, start & restarted the apache2.

(13) ansible client -m apt -a "name=maven state=absent" --become → uninstalling maven in client server.

(14) To know about memory →

i) ansible client -a "free -h"

ii) ansible client -m file -a "path=/temp/testfile state=touch"

iii) ansible client -a "df -h"

~~(15)~~

NOTE:- Shell modules & Command modules are non idem potent. that means we can execute in any no. of times either it's may be installing package.. etc anything (nothing to do).

Configuration managment tools :-

- Ansible } Agent less (Server, Destination) } Push based server
- Chef
- Puppet
- Saltstack } Agent required in destination Server } Pull based Server

CI-CD tools :-

- Jenkins
- Bamboo
- Travis CI

IAC tools :-

- Terraform
- cloudformation

Containerization tools :-

- Docker
- Rocket
- CoreOS

Container orchestration tools :-

- Docker Swarm
- kubernetes
- openshift
- Messos

monitoring tools :-

- Prometheus
- Grafana
- Nagios
- cloudwatch

cloud Providers :-

- AWS
- Azure
- GCP
- IBM cloud
- oracle

Artifact stores :-

- nexus
- JFROG
- ARtifactory

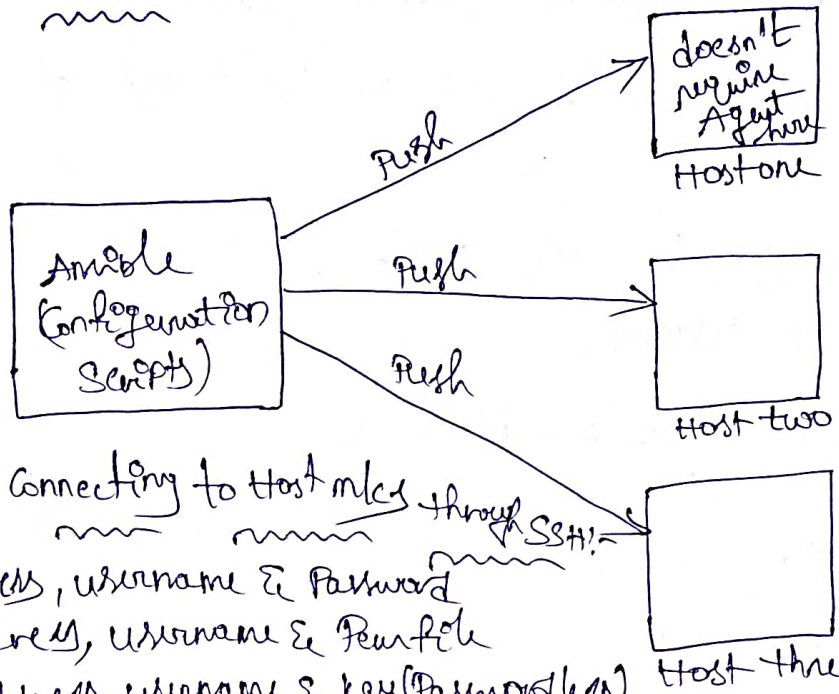
SCM tools :-

- Github
- Gitlab
- Bitbucket

Build tools :-

- maven
- Gradle
- Ant

Ansible work!-



Configuration can anything :-

↳ user & Group managment
↳ file managment
↳ managing Packages (Softwares)
↳ Deploying APPs

Ansible Connecting to Host m/cs through SSH!-

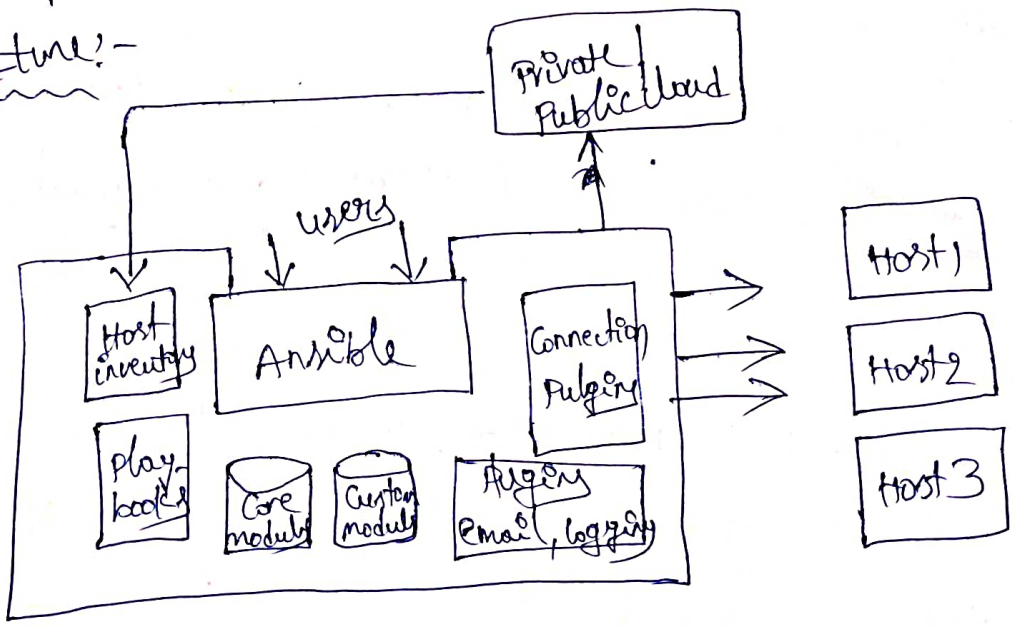
→ IPAddress, username & Password
→ IPAddress, username & Pemfile
→ IPAddress, username & key (Password less) (SSH)

↳ Using Ansible we can create & install tomcat server & deploying apps in tomcat

↳ using Ansible we can Create & install Docker & ~~det~~ managing Containers in Docker

↳ But this ansible is doesn't a master slave architecture ~~First~~ that means doesn't manging the Server Just it's Configuration done in destination Servers.

Architecture!-



Host Inventory :- Host machine details consists like IP address, username, password .. etc . there are two types of Host inventories.

1) Static Host inventory :- It's static file in which we will list group host servers.

default location: /etc/ansible/hosts

2) Dynamic inventory :- It's script (python, shell script) which will fetch host details dynamically from external sources (Aws, Database .. etc).

whenever change in infrastructure if I am using static inventory we need to update Host file manually. but in dynamic inventory automatically fetch the details from cloud.

Playbook :- It is Configuration script (yml) Contains no. of plays each play contains tasks. those tasks will be executed in a hosts.

Connection plugins :-

Ansible is connected to the host machines for performing tasks with the help of Connection plugins [SSH, winRM]
↓ ↓
linux windows.

modules :-

whenever executed the playbooks, the playbooks executes with the help of modules. In Ansible we have a thousands of modules. Each module will having some python script so based on this script modules playbooks can be executing.
there are two types,
↳ Core module (default modules all) ↳ custom module (for write your own module based on Python script)

NOTE:- If your Host Servers in Private Subnet then playbooks not directly downloads the Packages from internet so you need download Packages on Ansible Server (Public Subnet) & then Copy to Private Subnet through playbook.yml file.

Configuration & Installation Ansible:-

↳ Create one user (Ansible) & Set Password

↳ Enable PasswordAuthentication in /etc/ssh/sshd_config &

```
PasswordAuthentication yes
# Password Authentication no
```

Restart the Service

↳ Give Sudo access to ansible user

```
{ansible ALL=(ALL) NOPASSWD: ALL}
```

↳ Switch to 'ansible' user & Install 'Ansible Package' in this user.

→ /etc/Ansible

↓

ls (Hosts, Ansible.cfg)

→ In this 'Hosts' file, we need to Configure Host Server details, Username & Passwords & Pemkey paths.

↳ keep it host_key checking as false,

vi /etc/hostansible/ansible.cfg

↓

host_key_checking = false

:wq!

Search :/host_key :Set nu

↳ Generate 'ssh-keygen' in Ansible user & this Public key passed to Host m/cs in for Password less Authentication.

↓ when I am login to host machine user then its no need to enter password.

mkdir .ssh

↓

cd .ssh

↓

sudo vi authorized_keys

↓

"Paste Public key"

↓

:wq!

(or) for Connecting through 'Pem-file', then GOTO Hostfile,

↓

vi /etc/ansible/hosts

↓

[appserver]

172.31.40.50 ansible_user=ec2-user ansible_ssh_Private_key_file=~/DevOpPem

↓ username

Create file & Give Path in ansible server

↓

chmod 700 DevOpPem

↳ checking whether Connectivity working or not.

↓

ansible all -m ping

↓

Connectivity happens.

NOTE:- Better to create user (ansible) in host machines manually or else you can create through playbook yml file & execute.

→ create user & Passwd
→ enable Passwd Authentication
→ restart sshd
→ Give visudo Permissions.

3 ways to Connecting Host m/cs from Ansible using :-

i) if you have ec2-user & any .Pem key in host machines.
then in 'host' file enter (in Ansible m/c).

↓

vi /etc/ansible/hosts

↓

172.31.40.50 ansible_user=ec2-user ansible_ssh_Private_key_file=~/DevOPPem

[Host m/c details]

↓

:wq!

↓↓

ansible all -m Ping → for checking whether pings work or not

ii) if you have username(ansible) & Password in host machines.
then in 'host' file enter (in Ansible m/c),

↓

vi /etc/ansible/hosts

↓

#
172.31.40.50 ansible_user=ansible ansible_Password=DevOPS@2020

[Host m/c user details]

↓

:wq!

↓↓

ansible all -m Ping → for checking Pings work or not

[here install Some ssh Pakcage if throws Some error.

iii) If have a username(ansible) & Password in host machine, but Im not enter those details in "host" file, for that create a "ssh-keygen" keys & copy to host m/c user.

↓

In Ansible m/c } ssh-keygen

↓

ls (Public, Prt, known-hosts)

↓

then Public key copy to host m/c user(ansible).

↓

ssh-copy-id ansible@172.31.34.199

↓ host m/c user ↓ Host m/c IP

↓

Copied to host m/c user.

---

vi /etc/ansible/hosts

↓

#
#
172.31.34.199

↓

:wq!

↓

ansible all -m ping → for checking ping work or not.

Configuration can be done in three ways on Host m/cs :-

→ Ansible Commands

↳ ansible <Groupname/Hostname> -b -m <modulename> -a <arguments>

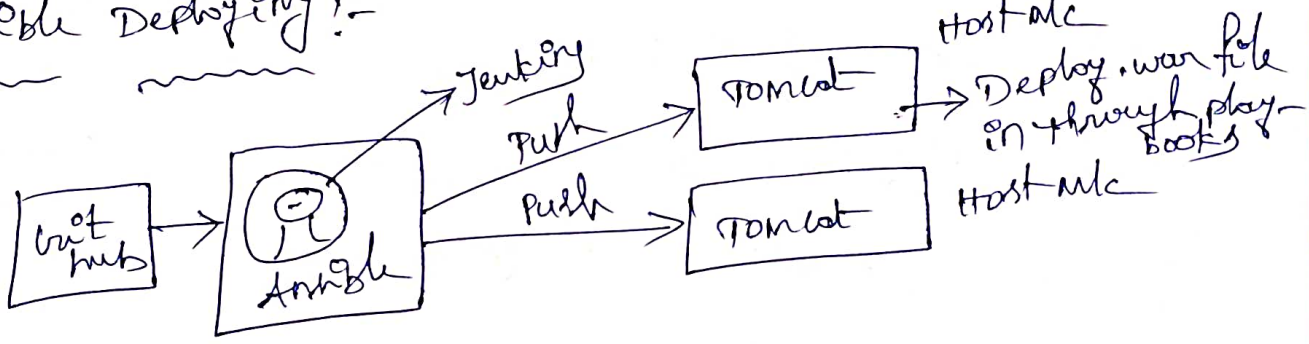
-b → Become a root user (sudo access) to perform the configuration.

-m → module

there are ~~two~~ thousands modules in this. modules are yum, apt, aws modules, Docker, k8's, copy ... etc.

↳ ansible-doc -l → for showing all modules default.

↳ Ansible Deploying :-



→ Ansible playbooks

→ Ansible Roles

Shell modules :- The shell modules in Ansible is used to execute all shell Commands against the target host.

Command modules :-

Task1 :- i) Install httpd server in host m/c
ii) Copy index.html to in host m/c
iii) start httpd servile.

↳ through Adhoc commands :- for installing through commands you need to enter 3 commands,

i) ansible all -b -m yum -a "name=httpd state=Present" → for installing httpd Software

ii) ansible all -b -m copy -a "src=index.html dest: /var/www/html/index.html" → for copy the file to host m/c

iii) ansible all -b -m servile -a "name=httpd state=started" → for start the httpd servile

↳ through playbook :- Instead of entering the 3 commands, you need to create one playbook yaml file & execute it.

```
---
-host: all
 become: true
 tasks:
  -name: Install httpd
   yum:
    name: httpd
    state: Present
  -name: Copy index.html
   copy:
    src: index.html
    dest: /var/www/html/index.html
  -name: start httpd servile
   servile:
    name: httpd
    state: started
```

↓

for run the playbook → ansible-playbook <playbook.yml>

Results :-

1) Gathering facts
2) Install httpd
3) Copy index.html
4) Start httpd servile
5) Play Recap

Adhoc Commands :-

1) if your host inventory file in another path then executing the playbook,

↳ ansible-playbook -i <inventoryfilepath> <playbookname.yml>

↳ ansible-playbook <playbookname.yml>

2) for syntax checking of playbook.yml file & indentitation also,

↳ ansible-playbook <playbookname.yml> --syntax-check

3) Dryrun command means it is not going to perform the Configuration Just its validation what's going to be Performs,

↳ ansible-playbook <playbookname.yml> --check

4) if you run your playbook in 'verbose' mode ansible will display additional logs,

↳ ansible-playbook <playbookname.yml> -v → for little bit info
↳ " " " -vv → for more info
↳ " " " -vvv → for more more info.

5) tasks are run step wise in playbook file,

↳ ansible-playbook <playbookname.yml> --step
(yes/no/continues)

gather facts :- while executing the playbook, then initially its Gathering facts means host machine details which os ...etc information. if you don't want gather the facts then keep in playbook yml file "gather_facts: no(or)false".

↳ ansible all -m setup
→ for knowing more facts about target m/c.

How we can skip or execute specific tasks :-

playbookName.yml :-

```
- hosts: appserver
  become: true
  gather_facts: no
  tasks:
    - name: Install httpd
      yum:
        name: httpd
        state: Present
      tags:
        - install
    - name: copy index.html
      copy:
        src: index.html
        dest: /var/www/html/index.html
      tags:
        - copy
```

Ansible tags :-

It's great feature which can help you execute respective tasks from the playbook. By default all tasks from the playbooks are executed but with tags we can control this behaviour & execute the tasks with matching tags.

↳ ansible-playbook <playbookname.yml> --list-tags → for showing list of tags which are present in playbook file.

↳ ansible-playbook <playbookname.yml> --tags "<tagname>,<tagname>" → which tags tasks are performing here while executing.

↳ ansible-playbook <playbookname.yml> --skip-tags "<tagname><tagname>"
→ except the tags tasks, all remaining tags task executes

Handlers :- Tasks are executed by default from top to bottom. Handlers also special kind of tasks will not be executed by default. Some task has to notify the handler.

If the task is changed then only handler will be executed at the end of play.

force handler :- force handlers you can set the property force_handlers: true in playbook which will run the handler task even though you have task failures.

```
---
-hosts: all
 become: true
 gather_facts: no
 tasks:
  - name: Install http
    yum:
      name: httpd
      state: Present
    tags:
      - install
  -name: copy index.html
   copy:
     src: index.html
     dest: /var/www/html/index.html
   tags:
     - copy
   notify:
     - start httpd servie
   handlers:
     -name: start httpd servie
      servie:
        name: httpd
        state: started
      tags:
        - install
        - start
```

```
---
-hosts: all
 become: true
 gather_facts: no
 vars:
  name: mithey-tech
```

→ variable refering variable name this matches in index.html name.

```
template:
  src: index.html
  dest: /var/www/html/index.html
```

→ used for replace the with some values

Content:
welcome to {{name}}
variable

notify to handler
handler task

this handler final task will executed only when above tasks to be changed.

template:- if you want to replace with some names only it's possible on template with the help of variables.

variables:- variable refers in many ways.

1) Runtime variable
   ↳ ansible-playbook <playbookname.yml> --extra-args "name=Mithun"
2) playbook variables
3) Groupvars
4) Host vars
5) Role vars.

Results in URL through Runtime variable:-

Any welcome to mithun

Results in URL through playbook variable:-

Any welcome to mithun tech

1) Runtime variable:- this variable can be executed only when through the command playbook execution.

↳ ansible-playbook <playbookName.yml> --extra-args "name=Mithun"

Here Mithun is placed instead of 'name' variable. these variables you can use in anywhere either in playbook or index.html file.. etc.

2) playbook variable:- this variable only pass in playbook yaml file not in command while execution of playbook.

3) Group variables:- Group level we can define variables these variables will be available to the servers which are part of that group.

for this you must create a Groupname yaml file under the group folder

→ /etc/ansible/group_vars/appServers.yml.

→ /etc/ansible/group_vars/DPServers.yml

→ /etc/ansible/group_vary/all.yml

Groupname vi /etc/ansible/hosts

[appServers]
170.10.20.11
[DPServers]
179.11.20.19

→ In this variables only reflecting the appServers host m/c playbook.yml task

→ In this variable only reflects into the DPservers host m/c playbook

→ In this variable reflects both appServer & DPServer hosts m/c.

Host variables:- here you can define variable in host level or system level. Create you must file with IP & hostnames of hostfile. here define the variables,

↳ vi /etc/ansible/host_vars/170.10.20.11.yml →

(or) 'hostname' of hostfile

↳ vi /etc/ansible/host_vars/179.11.20.19.yml

Preferences for taking variables:-

1) Run time
2) playBook
3) Host variable
4) Group var (specific group)
5) Group var (all)

Conditions:- Ansible supports conditional evalutions before executing specific task on the target hosts.

Playbookname.yml:-

```
- hosts: all
  become: true
  vars:
    name: Mithun
  tasks:
   - name: Install HTTP
     yum:
       name: httpd
       state: Present
     tags:
       - Install
     when: ansible_os_family == "RedHat"
```

Condition here i.e httpd only install only RedHat os. if it Host file not RedHat then task will be "Skipping".

↳ for checking internal gather facts we have a command,

→ ansible all -m setup

Loops:- Sometimes you want to repeat a tasks multiple times. In Computer Programing, this is called loop.

↳ For installing wget, git & zip software you need write 3 tasks in playbook. Instead of 3 tasks you can write single tasks with the help of loop.

Playbookname.yml:- (Installing softwares):-

```
- hosts: all
  tasks:
  - name: Install softwares
    yum:
      name: "{{item}}"
      state: Present
    with_items:
      - git
      - wget
      - zip
```

referring to here

loop

```
- name: Install softwares
  yum:
    name: ['wget', 'git', 'zip']
    state: Present
```

this is only for 'yum' module because with_items is deprecated for yum module.

playbookname.yml (Create username & Password):-

```
- hosts: all
  tasks:
  - name: Create users
    user:
      name: "{{item.username}}"
      Password: "{{item.Pwd}}"
      Create_home: "{{item.homeDir}}"
      state: Present
    with_items:
      - {'username': 'balaji', 'Pwd': 'DevOps@2020', 'homeDir': "yes"}
      - {'username': 'vasu', 'Pwd': 'DevOps@2021', 'homeDir': "no"}
```

referring to here

Requirment:-

Create an Ansible Role that will add/remove a user & give the Privileges by reading pattren from metafile & metafile pattren is given below,

ashwani: Present: super
nagendra: Present: normal
raju: absent: super

NOTE:-

1) Present/absent is state of the user whether it'll be available on the system & not.
2) super denotes a user will be able to use sudo Privileges
3) normal will be not able to use sudo Privileges.

```
-hosts: all
 tasks:
  -name: Create users
    user:
      name: "{{item.username}}"
      state: "{{item.state}}"
    with_items:
    -{'username':'ashwani', 'state': 'Present', 'sudoaccess': "super"}
    -{'username': 'nagendra', 'state': 'present', 'sudoaccess': "normal"}
    -{'username': 'raju', 'state':'absent', 'sudoaccess': "super"}
    -name: Setup Access for Ansible user
      Copy:
        dest: "/etc/sudoers.d/{{item.username}}"
        Content: "{{item.username}} ALL=(ALL) NOPASSWD: ALL"
        validate: /usr/sbin/visudo -cf %s
      with_items:
        - { user details
        -
        -
      when: item.sudoaccess == "super"
```

user details

Condition here only executes super users.

for creating user playbook if you have users.conf file,

vi usermanagement/files/users.conf

↓

ashwani: Present: super
nagendra: Present: normal
raju: absent: super
0 1 2

```
- hosts: all
  tasks:
  - name: Create user
    user:
      name: "{{item.split(':').0}}"
      state: "{{item.split(':').1}}"
    loop: "{{lookup('file','users.conf').splitlines()}}"
  - name: Setup sudo access for Ansible user
    Copy:
      dest: "/etc/sudoers.d/{{item.split(':').0}}"
      Content: "{{item.split(':').0}} ALL=(ALL) NOPASSWD: ALL"
      validate: /usr/sbin/visudo -cf %s
    loop: "{{lookup('file','users.conf').splitlines()}}"
    when: item.split(':').2 == "super" and item.split(':').1 == "Present"
```

if you want install Jenkins or uninstall Jenkins also through the playbook, Go through the 'Ansible-playbook' in Github.

Ansible vault :- Ansible vault is a feature of ansible that allows you to keep Sensitive data such as passwords or keys in encrypted files.

Commands :-

1) ansible-vault create <filename> → for creating the file & it will be encrypting format

2) ansible-vault encrypt <filename> → if you have a file then by using this command encrypting the file. encrypted files we are not able to see the content by using cat command.

3) ansible-vault view <filename> → for viewing the encrypting file with the help of password

4) ansible-vault edit <filename> → if you have a encrypted file then directly you can edit with the help of this command. because manually we not able to edit the (vi editor) file or if you want to edit that file then 1st you can decrypt the file & then edit the file Again you can encrypt the file.

5) ansible-vault decrypt <filename> → for decrypting the file.

6) ansible-vault rekey <filename> → if you want to change the password you can change with the help of this command.

vi /etc/ansible/hosts

↓

#
#
172.10.20.30 ansible_user=ansible ansible_password={{system_pass}}

↓ :wq!

this variable you can refere in all.yml file [etc/ansible/group_vars/all.yml]. but this all.yml file will be encrypted stage. So if trying to pings the host m/c not able to connect you need enter Password while executing.

↳ ansible all -m ping --ask-vault-pass

↓

" enter here vault password "

↓

pings succesfully.

(a)

here human intervention is required for typing the vault password

without human intervention to typing password, you create a another way.

↳ ansible all -m ping --vault-password-file=n/vaultpass

you can create file with the help of this name. Enter the password in this file. then while executing of this command Password will takes from that file.

Ansible-TOWER:- It is the enterprise Product not a opensource Product. this is entirly Gui Product. with the help of GUI you can Create, execute the playbook... etc

Ansible Role:- It's set of tasks, handlers, variables, files, inventory ... etc to configure specific requirements, organised in a ~~Profit~~ Predefined structure. Easy to understand, maintain & share Compared to ansible playbook.

for creating Ansible Role Command,

↳ ansible-galaxy init <RoleName>
httpd or tomcat

↓

yum install tree -y [for shows files & folder in tree structure.

↓

tree httpd

↓

```
httpd/
├── defaults
│   └── main.yml      } → variables reference.
├── files             } → only files reference & copying the files
├── handlers          } → handlers task reference.
│   └── main.yml
├── meta              } → here show meta info. like which os,
│   └── main.yml      } Author .. etc
├── README.md
├── tasks             } → tasks define here.
│   └── main.yml      }
├── templates         } → templates like replacing the name with values with the help of variables.
├── test              } → host list details
│   ├── inventory     } → Executing the playbook only locally
│   └── test.yml      }
└── vars              } → variables define here (1st Preference)
    └── main.yml      }
```

if you want to run locally then use that local inventory file for localhost,

↳ ansible-playbook -i httpd/test1/test.yml
↳ means under the test1 inventory file takes

But I want to run this Role in host m/c so here I am writing seperate file,

→ ansible-playbook -i http/test1/inventory httpd/test/test.yml
(or) vi site.yml
path
Custom inventory

↓

```
- hosts: appServers
  become: true
  roles:
  - httpd  ⟶ Give ansible Role here
```

↓

:wq!

↳ ansible-playbook site.yml --ask-vault-pass

↓ Give Inventory path, if not give default inventory takes /etc/ansible/hosts.

"Enter Password"

↓

Executing & Install httpd in host m/c.

order of Playbook execution:-

↳ first ansible runs the tasks from the roles section followed by the tasks from the tasks section.

```
---
- name: order of Taskexecution
  hosts: node1
  tasks:
    - name: Installing httpd service
      yum:
        name: httpd
        state: Present
      become: true

  roles:
    - role: role_date
```

2nd execution

→ 1st Execution

Better to write 1st role section & later task section.

↳ suppose we want to run some tasks before the roles & after the task section. this can be achieved also,

Pre_tasks: → task section that run before the roles

Post_tasks: → " " " " after the tasks.

order execution when using Pre & Post tasks

```
---
- name: order of Taskexecution
  hosts: node1
  Pre_tasks:
    - name: executing Pre_task
      debug:
        msg: 'Greeting from Pre_task'

  roles:
    - role: role_date

  tasks:
    - name: Installing httpd
      yum:
        name: httpd
        state: Present
      become: true

  Post_tasks:
    - name: executing Post_task
      debug:
        msg: 'Greeting from post_task'.
```

1→ Instead of defining the roles inside the 'roles' section, you can define the roles inside the 'task' section by using

1. include_role: → add the role dynamically

2. import_role: → add the role statistically

```
---
- name: Demo for Include_role
  hosts: node1
  tasks:
   - name: first task
     debug:
       msg: 'execution task'
   - name: Include_role
     include_role:
       name: role_date
   - name: sum of two numbers
     debug:
       msg: 'the total is {{ 80 + 80 }}'
```

```
- name: import_role
  import_role:
    name: role_date.
```

↓

successfully executes.

But if something did wrong in role_date file the playbook is 'Aborted' in include_role & the playbook is gets 'stoped' in import_role.

Ansible Tower:-

① → log in tower after installation done. then 1st Create a organisation.

→ Goto 'organisation' → ⊕ → Name:- Ansible_Tower_Demo
Description:- NA
Galaxy Creds:- (Ansible-Galaxy)
↓
Save
↓
here Goto organisation & showing here.

② Create user & add Permission in organisation.

→ Goto 'user' → ⊕ → first & last name:- Peter
organisation:- Ansible_Tower_Demo
Email:-
Username:- Peter
Password:-
User type:- normal
↓
Save

Goto Particular user 'Peter' & Add Permission.

name	Type	Role
Ansible_Tower_Demo	organisation	Member

③ Creating team & Add existing user.

→ Goto 'team' → ⊕ → Select 'org' here → Give team name 'Dev-team' → Save

④ Create Static Inventory.

1st → Goto 'Inventories' → ⊕ → Inventory →
Name:- Prod_Inventory
org:- Ansible_Tower_Demo
↓
Save

2nd → Goto created Inventory (Prod_Inventory) → under the Inventory Goto 'Groups' → Give, name :- Prod-servers
Description :- Prod-Servers
↓
Save

here showing 'Prod-Server' group under the Inventory.

3rd → Again Goto Created Inventory (Prod-Inventory) → under the 'Hosts' → Give Hostname: ansiblenode1
Description :-
↓
Save

finally shows :- under Hosts,

Hosts	Description	Related Groups
ansiblenode1		Prod-servers

(5) Create new user in target m/c.
- sudo useradd vamsi
↓
sudo Passwd vamsi
↓
Save

(6) create m/c credentials to access the inventory hosts.
→ Goto 'Credentials' → name :- node1_Creds
Description :- node1_Creds
Org :- Ansible_Power_Demo
Credential type :- 'machine' type
↓
add username & Passwd (vamsi, vamsi)
& Previous Metasploit
↓
Prevlge_escalation method :- Sudo
↓ Save

It's add in 'credential Section'.

⑦ Create git Credentials to access Github from Tower,

→ Goto 'Credentials' → ⊕ → name:- Git_Creds

Description:- Git_Creds

org:- Ansible_Tower_Demo

Credentialtype:- 'Source Control' type

↓

give access of Github related sections

↓

Username: 'Github username'

Password: Token of Github

↓

Save

(It's appears under the 'Credentials' section).

⑧ Create Project on tower for VM playbook.

→ Goto terminal → mkdir /var/lib/Projects/ansible_Project

↓

cd /var/lib/Projects/ansible_Project

↓

vi Demo.yml

Again Goto 'GUI of tower' → Goto 'Project' →

⊕ → name:- Demo_Project_Tower

Description:- ———

org:- Ansible_Tower_Demo

SCM type:- Manual { Manual ✓ Git Mercurial

Project Base Path:- /var/lib/Projects

Playbook directory:- ansible_Projects

↓

Save

this Project appears in 'Project' section.

9) Create Project on tower for Github playbook:

Goto 'Project' → ⊕ →

Name:- Demo_Project_Github

Description:-

Org :- Ansible_Tower_Demo

SCM type :- Git

SCM URL :- ———

SCM Credentials :- 'Previously you added'

SCM Branch :- (In this branch our .yml file we have, sample.yml)

↓

Save

this appears in 'Project' section

10) Create & launch Job template for the Projects hosted on VM:-

Goto 'Template' → ⊕ → Job template →

name :- Job Template-Project in Tower VM

Description:-

Job type :- Run { Run ✓ Check

Inventory :- Prod_Inventory

Project :- Demo_Project_Tower

playbook :- Demo.yml

Credentials :- Node1_Creds

Enable_Privilege_escalation ☑

↓

Save

Goto that 'Template' & click Button.

↓

See execution results

(11) Create & launch Job template for GitHub Projects:-

→ Goto 'Template' → ⊕ → Job template →

name :- JobTemplate-GitHubProject

Description:-

Job type :- RUN

Inventory :- Prod_Inventory

Project:- Demo_Project_GitHub

playbook:- sample.yml

Credentials - node1_creds

Enable Privilegeescalation :- ☑

↓

Save

Under the template, click the button & execution will start.