Ansible [github: yankils]

→ What & Why?

Radically simple open-source IT automation engine (Automation tool)

Ansible automates:

↳ Configuration Management:-
we can automate/update configurations using Ansible

↳ Provisioning:
creating new resources, vm's, instances, docker containers, ...

↳ Applctn deployment:
deploying apptn. like jar, war, exe. in apptn server.

↳ Orchestration:-
means executing in sequential format. like enable monitoring (we need certain steps sequentially).

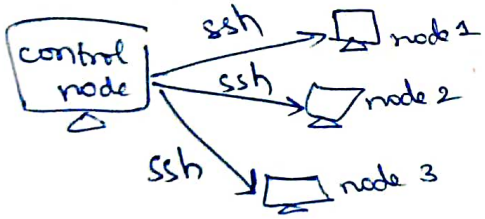
Why?

Simple: Esc2 it is simple & follow yaml format, human readable, no special code skills, tasks executed in sequential order [identation import].

Powerful: we use as Esc2 Configuration, app development, provisioning, orchestration.

Agentless: Use open SSH (no additional agent required), Secure.

→ How Ansible works?



Ansible is installed in control node. It manages so many clients using ssh. To manage clients/nodes it is possible through password less authentication. in nodes.

Components of Ansible

1) Inventory: list of servers info & managed node information. If the node info is not updated in inventory, it cannot manage. So node which are controlled by control node should be updated in inventory.

2) Playbooks: are the actual tasks which we need to execute on the target system. Eg: If we want to create an user on all client nodes, how to create & which user to create, that information is updated in the playbook. & on which server we need to update that info is available in inventory.

3) Modules: usually used by task (activity which we want to do). these modules are shipped with Ansible while installing.. (100 modules we have)

Ansible Terminology

1) Control node:
Any machine with Ansible installed

2) Managed nodes:
The network devices (servers) you manage with Ansible.

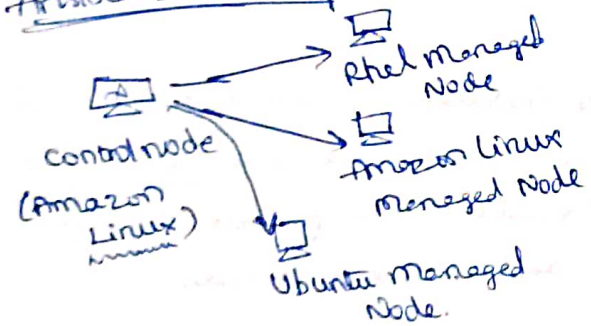
3) Inventory
A list of managed nodes. An inventory file is also sometimes called a "hostfile".

4) Module:
the units of code Ansible executes. Each module has a particular functionality.
[we have predefined modules, we can even create custom module]

5) Tasks:
the units of action in Ansible.

6) Playbooks
ordered lists of tasks.

Ansible Lab setup



→ Setup Ansible Control Node

Prepare Ansible Server:-

1) Setup EC2 instance
2) Setup hostname. (optional)
3) Create ansadmin user
4) Add user to sudoers file (should have root previlages)
5) Generate ssh keys (to copy to client system & enable password less auth)
6) Enable password based login

7) Install Ansible

1) Setup EC2 instance in AWS

Launch 2 EC2 instances. (Amazon linux 64 bits, t2-micro give tags name-Ansible, SG- with TCP-22, HTTP-80, 8080-custom port & launch).

[Use Mobaxterm for SSH]

session → SSH → Remote host :- IP (public IP)

Adv SSH-settings

✓ Use private key :- .pem file key pair associated with instance

specify username → ec2-user, ec2-user, ec2-us, ubuntu

OK (that Amazon Ubuntu)

2) Setup hostname

$ sudo su - root

$ hostname ansible-control-node (for setting up hostname)

$ vi /etc/hostname.

(remove default hostname replace with)

ansible-control-node

:wq

duplicate tab in Mobaxterm — u can see hostname as setup.

sudo su -

clear

3) Now creating a user.

3) # useradd ~~ansible~~ ansadmin

passwd ansadmin

New pd :- ansadmin123

Retyp :- ansadmin123

succesfull.

4) add user to sudoers file

vi visudo

Shift + G to go end of file

under ## Allow root to run

root .....

ansadmin ALL = (ALL) NOPASSWD: ALL

:wq

#sudo su - ansadmin [switching as ansadmin]

5) Generate SSH keys

$ ssh-keygen

enter to give ↵ ↵ ↵

$ ll

total 0

$ cd .ssh [keys are stored here]

$ ls

id_rsa id_rsa.pub [public & private keys]

we will copy this public key into target key. & private key is used to connect with public key.

$ exit

6) Enable password based authentication

clear

vi /etc/ssh/sshd_config

at end

/Password ↵

u can see in middle

#PasswordAuthentication yes

PasswordAuthentication no

So

keep PasswordAuthentication yes

" " . no

correct it

:wq

Now we need to restart the service

service sshd restart.

. restart sshd.service

#

7) Install Ansible on AWS EC2 instance.

Go to github repo firstly install python then Ansible.

yum install python

yum install python-pip

y/es

#clear

pip3 install ansible

successfully installed ansible-2.9.1

to get version

ansible --version.

lets first create one directory (config=NONE) to download config.

mkdir /etc/ansible

cd /etc/ansible

now go to official ansible website & download configuration file.

copy file from github of 491 lines

vi ansible.cfg

paste-file.

:wq

more ansible.cfg

yes

#clear

need to even have host file.

touch (hosts) → is inventory. i.e. node info needs to be there.

ls

→ Setup RHEL as Ansible Managed Node:-

Node - RHEL Managed Node

Setup control node

1) Setup EC2 instance
2) Setup hostname
3) Create ansadmin user
4) Add user to sudoers file
5) Enable pswd based login

{ No need to Generate key pair bcz if is managed node it picks from control node }

1) launch instance — Red hat → t2-micro → tag: Name RHEL managed node → SG: which we created first. → launch. (using existing keypair)

2) SSH → login same as control node.

{ Same steps for Managed node as control node }

For taking IP address to update in control node → # ip addr

hostname: rhel_managed_node

..... copy IP addr

service sshd (restart / reload)

→ Adding RHEL node to Ansible

Now we need to manage this managed node using Ansible. For this we need to update new IP in hosts file on control node i.e. establishing connection blw these 2.

login via SSH both control & managed node in Mobaxterm

In control node server: add RHEL node server IP in hosts (Inventory)

```
# vi hosts
-- add RHEL private IP  172.31.30.140   [To get ur ip adrs in M-N]
:wq
# cat hosts
```

↳ To manage our Ansible we need to login as 'ansadmin' bcz we have created keys over there, those keys we need to copy on to our target system. So that we can enable password less Authentication

```
# sudo su - ansadmin
$ ssh-copy-id 172.31.30.140   [to copy SSH keys to the IP]
enter password:
```

Before this → $ ssh 172.31.30.140 ↵ $ exit

In Managed Node

```
[root@ip-managed-node] # service sshd restart
```

→ i.e. we can communicate with client from control successfully

---

→ Introduction to ad-hoc commands

This command uses /usr/bin/ansible command line tool to automate a single task on 1 or more managed nodes. But this is not re-usable command.

Syntax

```
$ ansible [pattern] -m [module] -a "[module options]"
```

→ Some Ad hoc Ansible commands

Some modules are
- ping
- command
- Stat
- Yum
- User
- setup

} modules which we are using with ad-hoc command

In control node

$ ansible all -m ping ↵

(all servers in inventory) (This module will execute on systems in inventory (hosts) i.e. manage node)

"ping": "pong" → i.e. successful

To test this will add one more server in inventory file which is not exist.

$ sudo vi /etc/ansible/hosts

172.31.30.141

:wq

$ ansible all -m ping

(to check communication with manage node is good or not)

u can see will 140 it is success = pong.

141 " failed

→ Ansible Components:-

↳ Ansible Ad-hoc commands:-

Lets see remaining all modules

$ ansible all -m command -a "uptime" (or "date" or "who")

pattern — module — argument / attribute

u could see this system is up & running from last 20 min....

2 users is ec2-user & ansadmin

2) stat :-

$ ansible all -m stat -a "path = /etc/hosts"

(path = where file exist or not.)

4) yum :- which used to install package

→ This should run under root to

$ ansible all -m yum -a "name = git" -b (become root)

(name = Package name)

to cross check go to manage node

git ↵

u get some commands i.e. git is there in that server.

5) users :- we can create user.

$ ansible all -m user -a "name = johns" -b (for being root)

& In manage node check # cat /etc/passwd

in last u find

If u run same command again 1st it shows in yellow as Change
2nd " " Green as Sucess

Will not create again

6) Setup : gives information of entire manager nodes.

$ ansible all -m setup
(doesnot require attribute)

↳ we get manager-node info

Ansible Inventory

Ansible works against multiple managed nodes or "hosts" in your infrastructure at the same time, using a list or group of lists known as Inventory file

Contains list of hosts (nodes) which are managed by ansible control node

We have default & custom inventory file

→ /etc/ansible/hosts ↳ Use -i option: ansible -i myhosts

→ defined in ansible.cfg file

↳ it is configuration file — if ansible command request some inputs which we have not specified picks from this .cfg file.

In control node

$ cat /etc/ansible/hosts

gives list of hosts

$ ansible all -m ping.

$ pwd

(/home/ansadmin)

$ ls

$ vi hosts

give ip manage node IP

:wq

$ cat hosts

NOW I would like to run ping command on this inventory not on default

$ ansible all -m ping -i hosts

↳ if not specified picks default. [this is defined in ansible.cfg file]

To open $ more /etc/ansible/ansible.cfg to see default

$ sudo vi /etc/ansible/hosts [creating group]

[rhel] — This is 1 group

172.31.30.140

[dummy] — This is 1 group

172.31.30.141

:wq

$ cat /etc/ansible/hosts

↳ We can create any no. of servers in 1 group, no limit.

$ ansible all -m ping -i /etc/ansible/hosts

↳ It pings both servers

Now I want to ping only rhel servers

$ ansible rhel -m ping -i /etc/ansible/hosts

→ Ansible configuration file. — ansible.cfg.

$ ls -l /etc/ansible/ansible.cfg

↳ Whenever we execute ansible command & if we dont provide certain info to execute the command it is going to use some default info from this file

to open file

$ more /etc/ansible/ansible.cfg

Some defaults are inventory, libraries, modules (req for modules), utility,

↓

it request some modules

remote_tmp ⇒ some commands executing on remote system.

local_tmp ⇒ in local tmp

forks=5 ⇒ at a time ansible command can run on these no of system

ask-pass =true/false ⇒ to login in client system weather to ask password or not

come down & /default.

module_name = command

[privilege_escalation]

become = false [while executing dont execute as root]

become_method = sudo [how to become root]

become_ask_pass = false [for becoming root dont ask password]

→ Let's create our own ansible.cfg file.

[ansadmin@ansible-controller ~]

priority of cfg file :

1 ↳ env_var = ANSIBLE_CONFIG
2 ↳ current directory = ansible.cfg
3 ↳ home directory = ~/.ansible.cfg
4 ↳ default = /etc/ansible/ansible.cfg

[$ cat /etc/ansible/ansible.cfg]
$ vi ansible.cfg

[privilege_escalation]
become = True

$ ansible web -m yum -a "name=tree" (to specify like tree or kernel)

no need to specify -b.

→ Ansible Modules

A module is a reusable, standalone script that Ansible runs on a behalf either locally or remotely. through playbooks

modules interact with local machines, an API or a remote system to perform specific tasks like

creating users
Installing packages
Updating configurations
Spining up instances.. etc

modules are the programs that perform the actual work of the task for a play.

To know which modules are installed on our system

$ ansible-doc -l ↵

Ansible playbooks

Playbooks are the set of instructions (plays) that u can send to run a single target or groups of targets (hosts)

→ target :-

Bcz playbooks are providing direction & interactively with modules, Ansible assume u know how to do what u r trying to do & automates it. Thats why Playbooks are like instructions or directions - u telling the automated parts how u want the task configured.

→ tasks:-

If part of playbook needs to start web server, u going to need to know how thats done so you know to use service module & start web server by name.

→ Instructions for writing playbook

⇒ A playbook is a text file written in YAML format, & is normally saved as ".yml"

YAML [Ceelick - Yet another markup lang]
[now - YAML Ain't markup lang]

↳ The playbook begins with a line consisting of three dashes ("---") as a start of document marker.

↳ An item in YAML list starts with single dash followed by a space

↳ hosts & tasks are mandatory items in a playbook
hosts → inventory, tasks → activity

↳ playbook primarily use identation with space characters to indicate structure of its data

↳ modules are used to perform tasks

↳ Comment start with #

Eg: Lets convert one ansible ad-hoc command as script.

↳ command — ansible all -m user -a "name=john" -b

↳

1) first of create_user.yml
2) starts with ---
3) next command start with - host: all
4) become: true
5) tasks:
   - user: name=john

create_user.yml

```
---
- hosts: all
  become: true
  tasks:
  - user: name = john
```

In control node

$ pwd

/home/ansadmin

$ cd /opt

$ ls

aws rh

$ sudo mkdir ansible (creating directory

$ cd ansible/

$ ls -l

$ ~~cd create~~ ansible → it is owned by root

so changing ownership

$ sudo chown ansadmin:ansadmin ansible

$ ls -l

ansadmin ..... ansible

$ clear

$ vi create_user.yml

```
---
-  hosts: all
   become: true
   tasks:
   - user: name = john
```

:wq

$ cat create_uu.yml

$ ansible-playbook create_user.yml [for executing playbook]

command

playbook name

```
Task [Gathering facts]
 ok: 172-31-30-140
Task [user]
 ok [172.31.30.140]
play recap
140.-- OK = 2
```

Lets setup two more managed nodes (Amazon linux & Ubuntu)

To setup managed nodes:-

1) Setup EC2 instance → (Ubuntu)-3
2) Setup hostname [Amazon linux], 2 t2.micro, non-ft-MN, use ansible-SG, same key pair, launch]
3) Create ansadmin user
4) Add user to sudoers file
5) Enable password based login
6) Copy id to control-node → ssh-copy-id 172.31.16.135 and 172.31.27.43

hostname — { amazon-managed-node, ubuntu-managed-node } — user for ubuntu is ubuntu in MobaXterm

In control-node,

$ cd /opt/ansible/

$ ls

create-user.yml

$ vi hosts

groups [webservers] — for getting this # ip addr

172.31.30.140 (rhel inst.)

[appservers] 172.31.16.151 (amazon)

172.31.6.135 (amazon)

[dbservers]

172.31.27.43 (ubuntu)

→ BC2 1 server can be part of multiple groups

To add in ubuntu ansadmin

#useradd -m -d /home/ansadmin ansadmin

ls

ansadmin ubuntu

#sudo su - ansadmin

$ ls

$ pwd

/home/ansadmin

we need → bc2 ssh keys get copied to home directory

→ Running Ansible playbook on rest 2 servers

In control node,

$ cat create_user.yml → from this we created john user on rhel, now this user doesnot exist on amazon & ubuntu → to check #cat /etc/passwd

lets execute on other 2

$ ansible-playbook -i hosts create_user.yml → bcz we want on ansible location inventory

That means, if target system is in desired state then it doesn't change

o/p
play [all] — running playbook on all nodes
task [user]
changed = 1 — 1 change — yellow color
changed = 0 — no change — Green color
tasks [gathering facts]
ok
ok
ok
} gathering info of manager/target nodes

play recap ---
list of servers available, ok = 2 (servers are ok), change (what has changed), unreach, failed ...

lets modify our playbook
$ vi create_user.yml

- name: this playbook is to create user
hosts: all
become: true
tasks: — like docker
- name: creating user john
user: name=john
:wq

(or)
user:
name: john.

$ ansible-playbook -i hosts create_user.yml

o/p
Tasks [user] → Task [creating user john]
Play [all] → Play [this playbook is to create user]

=

Modules

↳ Installing packages - using playbooks using yum module

In control node,
[ansadmin] $ ls
create_user.yml hosts
$ vi install-packages.yml [creating playbook]

- name: this is to install packages
hosts: webservers
become: true
tasks:
- name: install package
yum:
name: git

state: installed

:wq

$ cat install-packages.yml

$cat hosts

[webservers]

2 servers { - shel
- anem

$ansible-playbook -i hosts install_packages.yml

only install in second system as in 1st it is already now.

lets edit playbook

$ vi install-packages.yml

- name: playbook
  hosts: webservers
  #become: true > we are not running as root
  tasks:
  - name: install packages
    yum:
      name: tree
      state: installed

$ ansible-playbook -i hosts install-packages.yml -b

$ cat /etc/ansible/hosts

This is the .conf but [webservers] are not in .cfg

[shel]
172.31.30.140
[dummy]
172.31.30.141

$ ansible-playbook install-packages.yml

It cant run as not match supplied host pattern

File Module – Create/remove a file/directory

→ How to create file/directory on target systems:

[ansadmin@... 7 $ vi create_file.yml

- name: to create file/directory
  hosts: all
  become: true
  tasks:
  - name: creating a file
    file:
      path: /home/ansadmin/dummyfile
      state: touch
:wq

touch for file

$cat create_file.yml
$cat hosts
$ansible all -i hosts --list-hosts

hosts (3):
=

$ ansible-playbook -i hosts create_file.yml (running playbook)
$pwd
$ ls -l
demofile .. (root)

Now, you can see everywhere(targets) demofile.

Lets comment become: true from playbook.

#become: true
path: /home/ansadmin1/dir1
state: directory

$ ansible-playbook -i hosts create_file.yml
$ ls -l
dir1 ... ansadmin1

→ To remove file/directory:

$ vi create_file.yml

state: absent

$ ansible-playbook -i hosts create_file.yml

It get removed.

→ To copy module

Copy from 1 location to other or 1 server to another.

Here, from master to target we are copying.

Lets create one index.html file

$ vi index.html
<h1> ansible playbook to copy file</h1>
$ vi copy_file.yml

- name: to copy file
  hosts: all
  become: true — (not req for copy from req if u want to change ownership then should be on root)
  tasks:
  - name: copy file

```
copy:
  src: /opt/ansible/index.html
  dest: /home/ansadmin
  mode: 0600
  owner: ithn        (to change owner of file.)
```

:wq

$ ansible-playbook -i hosts copy-file.yml → after this u could see on target file that file got created.
→ it creates this file on target
$ ansible-playbook -i hosts copy-file.yml --check
↓ it checks it works fine/not

To check syntax errors

$ ansible-playbook -i hosts copy-file.yml --syntax-check

sec-6 Multitask ansible Playbooks

→ Installing Apache on Rhel

Now we are using existing playbook install-packages.yml

[ansadmin] $ ls

install-packages.yml

$ cat install-packages.yml

---

lets copy this to install-httpd

$ cp install-packages.yml install-httpd.yml

$ vi install-httpd.yml

```
---
name: to install apache
hosts: webservers
become: true
tasks:
- name: install packages
  yum:
    name: httpd
    state: installed
- name: start httpd service
  service:
    name: httpd
    state: started
```

:wq

$ ansible-playbook -i hosts install-httpd.yml

for verifying TLS need to create .cfg

$ grep httpd

Now lets copy DNS(IPv4) of Rhel managed node from console

Search in google, it get opened

we can even uninstall by editing install_httpd.yml
$ cp install_httpd.yml uninstall_httpd.yml
$ vi uninstall_httpd.yml

```
- name: uninstall httpd
  hosts: webservers
  become: true
  tasks:
  - name: Stop httpd service
    service:
      name: httpd
      state: stopped
  - name: uninstall httpd
    yum:
      name: httpd
      state: removed
```

$ ansible-playbook uninstall_httpd.yml --check

→ Install Apache on Ubuntu

in ubuntu server
root@...) # apt install git
apt install apache2

to start service
service apache2 start

in control node
$ vi install_apache2.yml

```
---
- name: apache install on ubuntu
  hosts: db servers
  become: true
  tasks:
  - name: install apache2
    apt:
      name: apache2
      state: present
  - name: start apache2
    service:
      name: apache2
      state: started
```

module

to start service

:wq

install_apache2.yml

$ ansible-playbook install_apache2.yml --check

in ubuntu Mobaxterm

#ps -ef | grep apache2

to check it got installed

Ansible-Ubuntu-Manager node1 DNS — check in google

→ Notify & handlers in a playbook

vi install_httpd.yml

```
---
- name: install httpd
  hosts: webservers
  become: true
  tasks:
  - name: install package
    yum:
      name: httpd
      state: installed
    notify: start httpd service
  handlers:
  - name: start httpd service
    service:
      name: httpd
      state: started
```

install_httpd.yml

(registry o/p in this)
(handlers handles it)

same

$ cat install_httpd.yml
$ vi install_httpd.yml

$ansible-playbook -i hosts install_httpd.yml --check

to check

If the package we are trying to instance is already present then its runs handlers if not they dont get run.

→ Gather facts

How gather facts works:—

[ansible@...] ls

$ cat create_file.yml

:

$ ansible-playbook create_file.yml

Gathering facts - what it does?

Usually it is going to retrive the system information from remote system & manage nodes

→ But for deleting / creating a file we dont require system info. Whenever we are doing some installation on target system, to check what is status of current operating system we require Gather facts

$ ansible all -m setup

if u do this u get info of all servers [which collected by gathering facts]

u can see ansible_os_family: "Debian" → of ubuntu server.

→ as ubuntu is of Debian family

for rehel → "Redhat"

for linux → "Redhat".

let edit file

$ cat create_file.yml

$ vi create_file.yml.

```
- name: creates file or dir
  hosts: all
  become: true
  gather_facts: no → it dont gather facts of system.
  tasks:
  - name: creating a file
    ft:
      Path: /home/ansadmin/dir1
      State: directory.
```

$ ansible-playbook create_file.yml

u can see no gatherfacts & creates a file.

Why to restrict Gather facts:

Assume ur env has 100+ system. If ur collecting each systems info, it takes little longer time to run playbook. Moreover if ur using that info its Good. But if ur not using it, its not necessary

```
$ enable gather facts
$ vi create_fi.yml
    # gather_facts: no
$ ... run playbook
```

Conditions

→ When:-condition:

→ Install Apache using when

$ If u remember we creating 2 playbook for installing apache on ubuntu [install-httpd.yml] & on Rhel [install-apache2.yml]

Not good way so lets use single playbook.

open file

$ cat install-httpd.yml

$ cat install_apache2.yml

lets create a single file

$ vi install-apache-httpd.yml

```
---
- name: playbook to install apache httpd.
  hosts: all
  become: true
  tasks:
  - name: install httpd.
    yum:
      name: httpd
      state: installed
    when: ansible_os_family == "RedHat"
  - name: start apache
    service:
      name: httpd
      state: started
    when: ansible_os_family == "RedHat"
  - name: install apache2
    apt:
      name: apache2
      state: present
    when: ansible_os_family == "Debian"
  - name: start apache2
    service:
      name: apache2
      state: started
    when: os_family == "Debian"

:wq
```

$ clear

} i

} install httpd

} start httpd service

} install httpd

} start httpd service

$ cat install-apache-httpd

$ ansible-playbook -i hosts install-apache-httpd.yml ← --check (for testing)

o/p

task [gathering facts]
ok: 155
ok: 140
ok: 43

task [install httpd packages]
skipping: 43
ok: 135
ok: 140 } redhat

task [start apache]
skipping: 43 ——→ as it is ubuntu
ok: 135
ok: 140 } redhat

task [install apache]
skipping: 140
skipping: 135 } redhat
ok: 43 —— ubuntu

task [start apache]
skipping: 140
skipping: 135 } redhat
ok: 43 —— ubuntu

Before executing this playbook uninstall packages.

$ ansible-playbook -i hosts uninstall-httpd.yml —— on control node

this will done on redhat only, lets go to ubuntu server & uninstall on ubuntu

$ # service apache2 stop
# apt remove apache2
(Y/N) y

On control node refresh ip/dns in chrome, it doesn't work

$ ansible-playbook -i hosts install-apache-httpd.yml ---- here

↳ it will not get ok
it get changed

↳ now again refresh in google it works.

For testing, lets disable gather facts

$vi install_apache_httpd.yml

```
---
- name:
  hosts:
  become:
  gather_facts: no
:wq
```

to rename file name.

$mv install_apache_httpd install_apache_httpd.yml

Now if you run playbook it fails.

$ansible-playbook -i hosts install_apache_httpd.yml --check

Bcz apache_os_family is coming from gather facts, as Gather facts: no, it is failing.

So add it back in edit.

→ Uninstall Apache using when condition

```
[ansadmn ] $ vi uninstall_httpd.yml
---
- name: Uninstall httpd.
  hosts: ~~webservers~~ all
  become: true
  tasks:
  - name: stop httpd service
    service:
      name: httpd
      state: stopped
    when: ansible_os_family == "Redhat"
  - name: uninstall httpd
    yum:
      name: httpd
      state: removed
    when: ansible_os_family == "Redhat"
  - name: stop apache service
```

```
  name: apache2
  state: stopped
when ansible_os_family == "Debian"

- name: uninstall apache
  apt:
    name: apache2
    state: removed absent
  when ansible_os_family == "Debian"
:wq
```

$ ansible-playbook -i host uninstall_http.yml --check

→ Adding copy tasks to Apache playbook

Here lets copy index.html to var/www/html location [Here we are copying our app** to target system]

$ vi index.html

<h1> welcome to Apache tomcat </h1>

$ vi install_apache_httpd.yml

```
at last:
  :
  :
- name: copy index.html
  copy:
    src: /opt/ansible/index.html
    dest: var/www/html
    mode: 0666   (read & write)
:wq
```

} to do we need root privileges or become root

$ ansible-playbook -i hosts install_apache_httpd.yml --check

```
task [copy index.html]
  changed:
    :
    :
```

In --- one manager node ---- all manager nodes

$ cd /var/www/html

no such directory ... it will be in worker

to cross check now after running playbook.

to check content in managed node

$ 'sudo cat index.html.

As index.html is already exist in ubuntu. it get refreshed by new content

Now refresh in google.

→ List & with_items

How to install multiple packages with single task.

[consider]

$ cat install_package.yml

$ vi install_package.yml.

→ list

[with_items is getting deprecated]

```
---
- name: to install packages
  hosts: webservers
  become: true
  tasks:
  - name: install package
    yum:                    — adding in list of packages]
      name: ['git', 'make', 'gcc', 'wget', 'telnet', 'gzip']
      State: installed
```

:wq.

$ ansible-playbook -i host install_package.yml --check.

Before executing lets check whether they are present in managed node

$ rpm -qa | grep wget git, gzip, make, gcc, telnet

$

on ubuntu - how to check

#apt list --installed | grep wget.

#wget.

then

continued

$ ansible-playbook -i host install_package.yml

How to use with_items:

$ vi editor instead of name: [lists.....] → replace with

```
yum:
  name: "{{ item }}"
  State: installed
```

with items:
- git
- make
- gcc
- wget
- telnet
- gzip

:wq

--check → u get deprecated warning.

Ansible Variables

Used ways

1 — define within playbook
2 — passing from external files
3 — passing from hosts inventory
4 — passing while running playbook
5 — using group-vars or hosts_vars & .....

In control node
(enter) sudo su -ans admin
[ans...] $ clear
$cd /opt/ansible
$ ls
$ ls -ltr

$ cat create_user.yml — for creating user job we used this

Now we have many/other user. To overcome we can use variable

1) To define within playbook
$ vi create_user.yml

```
---
- name: to create usr
  hosts: all
  become: true
  vars:
    user: modi
  tasks:
  - name: creating usr
    user:
      name: "{{user}}"
```

key → name, value → {{user}}

:wq

$ ansible-playbook -i hosts create_user.yml

(It has already created.) it get created on all systems

In managed node ↗ to check

$ cat /etc/passwd

2) To define from external files

$ we can create seperate file & pass values

$ vi user.yml

~~create_tom~~ user: tom

$ vi create_usr.yml

```
---
- name: to create user usr
  hosts: all
  become: true
  vars_file:
    - user.yml
  tasks:
  - ' :
    :
    ( same)
```

$ ansible-playbook -i hosts create_usr.yml --check ↵

3) Passing while running playbook

(only)

$ ansible-playbook -i hosts create_user.yml -e "user=sonia"

u can check in managed node this user get created by cd /etc/passwd

⇒ To convert shell commands into a Ansible playbook

In github → yaraskils → ansible → tomcat installation on EC2 instance

$ vi setup-tomcat.yml

(to install java)

```
---
- name: setup tomcat
  hosts: all
  become: true
  tasks:
  - name: install java
    yum:
      name: java
      state: installed
```

} on redhat

when: ansible_os_family == "RedHat"

- name: install java on ubuntu
  apt:
    name: default-jdk
    state: present
  when: ansible_os_family == "Debian"

- name: download tomcat packages
  get_url:
    url: (paste tar.gz link)
    dest: /opt

from tomcat.apache website copy tar.gz link

to untar

- name: untar apache package
  unarchive:
    src: /opt/apache-tomcat-8.5.50.tar.gz
    dest: /opt
    remote_src: yes

to check checksum

- name: add executable permission on startup.sh file
  file:
    path: /opt/apache-tomcat-8.5.50/bin/startup.sh
    mode: 0777

to execute this

- name: to start tomcat services
  shell: nohup ./startup.sh
  args:
    chdir: /opt/apache-tomcat-8.5.50/bin

In manage node

$ cd opt.
ls

Check any tomcat folder is there or not

In control node,

$ ansible-playbook -i hosts setup-tomcat.yml --check

in checks entire work fail as no file is there

→ To check in manage node, the service is running / not

ps -ef | grep tomcat

lets access it from browser,

take (ip) addr from AWS system from console
DNS

DNS/8080 DNS link : 8080

→ Using tags in Playbook

Tags are useful if u want to segregate specific task from ur playbook

$ cat install_apache_httpd.yml

$ vi

after, when — yum
tags: install_apache

when — apt
tags: install_apache

when: start service
tags: start_apache

when: start apt service
tags: start_apache

:wq

* for restrict execution using tags

$ ansible-playbook -i hosts install_apache_httpd.yml --tags "install_apache"

Here, It will only install httpd, but doesn't start the service

→ Error handling in playbook

playbook — sequentially get executed. ఒక వేళ middle to వచ్చిన task fail అయితే next task ని proceed అవ్వదు. అక్కడే ఆగిపోతుంది.

$ vi install_apache_httpd.yml

first lets remove tags -

2 execute playbook

Tho to shel server.

vi /etc/httpd/conf/httpd.conf

testing like on tw

#service httpd restart

It fails as we added which torrent cannot read

from control node, if u run playbook again it fails showing service & copy also doesn't execute

Q) If some tasks are failed in middle, then also u want to execute rest files/tasks.

Ans:- In this case, we can use parameter called "ignore error"

$ vi install-apache-httpd.yml

After starting httpd apach on redhat add a line

when
ignore_errors: yes.

by default no

if above task fails u can proceed further

→ If u run playbook it get failing by ignoring & continuing.

→ In managed node,

remove testing file,

& run playbook in control node, no failures now.

Ansible Vault

→ Ansible vault is the feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plain text in playbooks or roles.

Files are protected with password

↳ Create: to create ansible vault file in encrypted format

↳ view: to view data of encrypted file.

↳ edit: to edit encrypted file.

↳ encrypt: to encrypt an unencrypted file

↳ decrypt: to decrypt an encrypted file.

To use these command we need to have password

↳ --ask-vault-pass: to provide password while running playbook

↳ --vault-password-file: to pass a vault password through a file.

control node
[ansadmin]

To create file in encrypted format

$ ansible-vault create vault-pass.yml

New vault password : abc123

Confirm. : 4

file opens

this is encrypted file1.

$ ls

$ cat vault-pass.yml

12-L1856...

you cant see data as it is encrypted

To see data,

$ ansible-vault view vault-pass.yml

Vault password: abc123

this is file.

To modify file,

$ansible-vault edit vault-pass.yml

this is encrypted file.

:wq

To view,

$ ansible-vault view vault-pass.yml.

To decrypt:

$ansible-vault decrypt vault-pass.yml

Vault password: abc123

decrypted successfully.

$ cat vault-pass.yml

u get data.

$ vi vault-pass.yml

Password: Ansible123

:wq

To encrypt.

$ ansible-vault encrypt vault-pass.yml

New password: abc123

confirm password: abc123

Encrypted successfully

→ Using Ansible Vault with git

(connection-3) Here we are writing an ansible playbook for using Ansible vault.

lets create some private repo in github

github.com → create repo

name: vault

private

Initialize readme

Create

lets clone

$ git clone (url.git

as it is private repo it ask for user & password

To avoid we can give it in link only

$ git clone https://username:pass@github.com/.../vault.git

(myuser) (ansible123)

cloning.....

$ ls

$ cd vault.

Go to github, lets create a file.

vault/index.html

1. <h1> Ansible vault testing </h1>

Commit new file

$ git pull

$ ls

you can see files

$ vi ansible-vault.yml

```
---
- name: ansible playbook to test ansible vault
  hosts: all
  become: true
  tasks:
```

```
- name: clone a repo
  git:
    repo: https://us... ...git
    dest: /opt/ansadmin/test-vault
```

module to create repo

dir for manager node

:wq

$ ansible-playbook -i hosts ansible-vault.yml --check

In manager node,

$ pwd

/home/ansadmin

$ ls

$ cd /opt/ansadmin

$ ls

- u get test-vault

$ cd test-vault

- index.html readme.md

$ cd /opt/ansadmin/test-vault

Now, I dont want to give password in git url. then, we before created vault-pass.yml. lets open this file.

$ ansible-vault view vault-pass.yml

pswd : Abc123

now lets replace pswd from url with this file.

$ clear

$ vi ansible-vault.yml

```
- become: true
  vars_files:
    - vault_pass.yml
  repo: https://username:{{ password }}@github.com/...git
```

Before lets remove this repo on target system

$ cd ..

$ ls

test-vault

$ sudo rm -rf test-vault/

} in all manager nodes

In control node,

Now we are passing our password from encrypted file

$ansible-playbook -i hosts ansible-vault.yml --ask-vault-pass

to open file which is encrypted

password : abc123

In manager node,

$ ls

test vault

lets see by using unencrypted file

$ vi pass.yml

Pabc123

:wq

Before that remove repo from manager node from back page.

$ansible-playbook -i hosts ansible-vault.yml --vault-password-file pass.yml

Ansible Roles

adv: used for creating reusable playbooks, & even other can use this task by roles.

(ansadm - control) $cat install-apache-httpd.yml

If u see long code for playbook, to overcome this long, to shorten it we use roles.

Lets create a role now, — role httpd. to connect this playbook to role

In control node,

$ansible-galaxy init setup-apache

it initiate the role called setup-apache

— role created successfully

$ ls

$ cd setup-apache

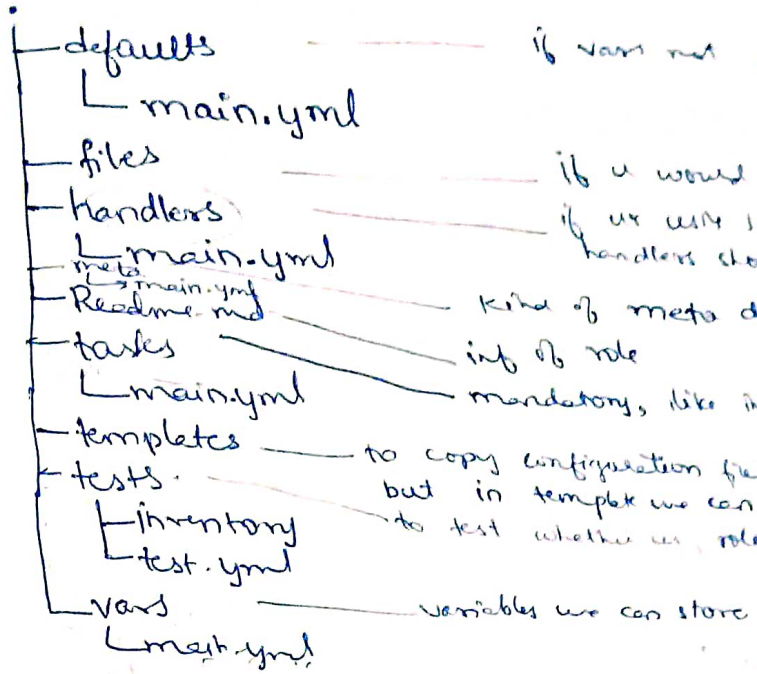
$ls

defaults files handlers meta Readme.md tasks templates tests vars

Lets create tree
sudo
yum install tree

$ clear

$ tree



8 directories, 8 files

tasks: contains main list of tasks to be executed by role.

handlers: contains handlers, which may be used by this role or even anywhere outside this role.

defaults: default vars for roles

vars: other variables for the role

files: contains files which can be deployed via this role

templates: contains templates which can be deployed via this role.

meta: defines some meta data for this role.

Adding task to playbook

[ansadmin - control-node] $ ls

$ cp install_apache_httpd.yml setup-apache.yml (or setup-cache.yml)

$ vi setup-apache.yml

(P.T.O)

```
---
- name: this playbook install httpd
  hosts: all
  become: true
  tasks:
  - name: install package
    yum:
      name: httpd
      state: installed
    when: ansible_os_family == "RedHat"
    notify: start apache
  handlers:
  - name: install apache2
    apt:
      name: apache2
      state: present
    when: ansible_os_family == "Debian"
    notify: start apache2

  - name: copy index.html
    copy:
      src: /opt/ansible/index.html
      dest: /var/www/html
      mode: 0666

    handlers:
    - name: start apache
      service:
        name: httpd
        state: started
      when: ansible_os_family == "RedHat"

    - name: start apache2
      service:
        name: apache2
        state: started
      when: ansible_os_family == "Debian"

    :wq
```

{handlers should be at last.}

```
$ ansible-playbook -i hosts setup-apache.yml --check
```

lets take public DNS of host. in browser should be running in all servers

⇒ lineinfile : module is used to change listen configuration with default of 8080.

for doing this the above handlers add below things: as next task don't work

```
- name: Ensure default apache port is {{ port }}
  lineinfile:
    path: /etc/httpd/conf/httpd.conf
    regexp: '^Listen '
    insertafter: '^#Listen'
    line: Listen {{ port }}
    when: ansible_os_family == "RedHat"
    notify: restart apache
- name: Ensure default apache port is {{ port }} on Ubuntu
  lineinfile:
    path: /etc/apache2/ports.conf
    regexp: '^ Listen'
    insertafter: "# /etc/apache2/sites-enabled/000-default.conf"
    line: Listen
    when: ..... "Debian"
    notify: restart apache2
```

replace with this line

also add vars

```
become: true
vars:
  port: 8082
```

& add in handler at last

```
- name: restart apache
  service:
    name: httpd
    state: restarted

- name: restart apache2
  service:
    name: apache2
    state: restarted
```

{ u can remove when condition from handlers }

→ Convert a playbook into ansible roles.

[ansadm-control-node] $ls.

[2nd-cn] lets open 1 more tab for contro node → duplicate & login as ansadm

```
$sudo su -ansadm
$cd /opt/ansible
$clear
$cd setup_apache
$tree .
```

we will now convert to this way.

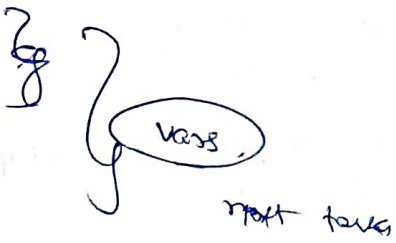
[1st_CN]

```
$more setup-apache.yml
```

see playbook

[2nd_CN]

```
$ vi vars/main.yml
---
#vars file for setup-apache
port: 8082
```




```
$ vi tasks/main.yml
```

Copy all tasks here.


```
$ cp /opt/ansible/index.html files/
$ vi handlers/main.yml
```

copy handler hes


```
$ vi defaults/main.yml
---
#
port: 8080
```

incase not providing variable then take default as 8080

```
:wq
$ ls -l
```

As we didnt use meta, templates, tests

```
$rm -rf meta templates tests
$ls.
$tree .
.
├defaults
│  └main.yml
├files
│  └index.html
├handlers
│  └main.yml
├readme.md
├tasks
│  └main.yml
└vars
   └main.yml
5 directories, 6 files
```

Now have to execute this role.

We need to update setup-apache.yml, before that lets take a backup.

```
$cp setup-apache.yml setup-apache.yml_backup.yml
$vi setup-apache.yml
```

Erase all content & keep only.

```
---
- name: this playbook install httpd
  hosts: all
  become: true
  roles:
    - setup-apache
```

} calling role

→ role name

```
:wr
```

lets cat

```
$cat setup-apache.yml
```

Lets first uninstall in manager node httpd

```
$ ansible-playbook -i hosts uninstall_httpd.yml
$ ls
$
```

in command mode

$ansible-playbook -i hosts setup-apache.yml --check

Check even by not using vars & use defaults

& run above command by editing vars.

Next by passing while running

$ansible-playbook -i hosts setup-apache.yml --extra-vars "port=8989"

now it uses 8989

& then now if u use em vars file also it uses 8989

→ How to check listner on manager nodes??

rhel & clone { # cat /etc/httpd/conf/httpd.conf | grep Listen.

in ubuntu { # cat ports.conf

→ push your playbooks on to Github

[C-N] $ echo "# ansible-for-beginners" >> README.md

$ ls

$ git init

add to staging - $ git add README.

$ git status

commit to repo - $ git commit -m "first commit"

$ git config --global user.email "ammp@gmail.com"

$ git config --global user.name "M P"

$ git remote add origin https:..../.git

$ $git push origin master

username: fx : ammp

pswd : ------

Now see in git hub

done