**ANSIBLE:**

**1)Ansible playbooks**

Ansible playbook is a **script** which contains one or more tasks to **perform some actions** on the **remote and local servers** with the help of **ansible modules** .

1. **EXAMPLE PLAY BOOK :**

provided hosts list is empty, only localhost is available. Note that the implicit localhost

does not match 'all' - name: create a directory

**file:**

path: /etc/ec2-user

state: directory

mode: "0755"

- name: Copy file with owner and permissions

**ansible.builtin.copy:**

src: /home/ec2-user/testing

dest: /home/ec2-user/dir1

- name: Compress Directory contents

become: yes

**archive:**

path: /home/ec2-user/dir1/

dest: /home/ec2-user/compressed.zip

format: zip

remove: yes

- name: Unarchive a file that is already on the remote machine

**ansible.builtin.unarchive:**

src: /home/ec2-user/compressed.zip

dest: /home/ec2-user/dir3

remote\_src: yes

**CREATE A FOLDER:**

- name: Create a directory if it does not exist

ansible.builtin.file:

path: /home/Ubuntu/test

state: directory

- name: Create a directory if it does not exist

ansible.builtin.file:

path: /home/ubuntu/platinum

state: directory

mode: '0755'

**REMOVE A FILE:**

- name: remove a directory if it does not exist

ansible.builtin.file:

path: /etc/some\_directory

state: absent

mode: '0755'

- name: Remove file (delete file)

ansible.builtin.file:

path: /home/ubuntu/gone.txt

state: absent

**ZIP THE FILE:**

- name: Create a zip archive of /path/to/foo

community.general.archive:

path: /path/to/foo

format: zip

- name: Create a zip archive

community.general.archive:

path: /etc/ansible/test.zip

format: zip

**CREATE UNZIP FILE:**

- name: Unarchive a file that needs to be downloaded (added in 2.0)

ansible.builtin.unarchive:

src: https://example.com/example.zip //zip file is getting downloaded from the internet

dest: /usr/local/bin

remote\_src: yes

- name: Unarchive a file that is already on the remote machine

ansible.builtin.unarchive:

src: /home/ubuntu/water.zip

dest: /home/ubuntu/cool

remote\_src: yes

**COPY FILE FROM ONE PLACE TO ANOTHER PLACE:**

- name: Copy file with owner and permissions

ansible.builtin.copy:

src: /home/ubuntu/stack.txt

dest: /home/ubuntu/final.txt

mode: "777"

- name: Copy file with owner and permissions

ansible.builtin.copy:

src: /srv/myfiles/foo.conf

dest: /etc/foo.conf

owner: foo

group: foo

mode: '0644'

---

- name: playbook to do deployment

hosts: localhost

become: yes

tasks:

- name: delete a directory

**shell: “<linux commands>”**

**2. TYPES OF ANSIBLE MODULES ?**

1. ansible.builtin.copy
2. archive - zip the files
3. ansible.builtin.unarchive - unzip the files
4. file - manage files, folders and permissions
5. get\_url - to download from internet or any url , for ex-nexus (downloading artifacts from nexus)
6. shell – linux commands can be used in the shell module
7. command – windows commands can be used in the command module
8. raw - linux commands can be used in the shell module & windows commands
9. win\_file – manage files and permissions
10. win\_zip – zip the files
11. win\_get\_url – to download from nexus
12. win\_unzip - unzip the files
13. yum – install packages on amazon linux machine,redhat machine (ex- packages – apache, tomcat, java,Jenkins etc)]
14. apt-get - install packages on ubuntu machine
15. service – to start and stop the services
16. ansible.builtin.command
17. ansible.builtin.debug:

3. **ANSIBLE PLAYBOOK RUNNING COMMAND ?**

Ansible –-playbook (playbook name).yaml/yml.

7. **Ansible host file location**

hosts file – it is called as inventory

**/etc/ansible/hosts**

Hosts file – contains the information of localhost and remote hosts details (for ex- slave machine’s details)

4. **Ansible Vault**

To store some of the important info or other things(ex-ssh key , credentials)

encrypt

**ansible-vault encrypt <yaml filename ex-abc.yaml >**

**ansible-vault encrypt ansvault.yaml**

DECRYPT:

**Ansible -vault decrypt <yaml filename ex-abc.yaml >**

5. **ansible playbook running command with extra variables**

---

- name: extra variable demo

hosts: localhost

vars:

fruit: "banana"

tasks:

- name: print message

ansible.builtin.debug:

msg: fruit is "{{ fruit }}"

ansible-playbook extrafruit.yaml --extra-vars "node=localhost"

**Multi extra variable -> Ansible –playbook <playbookname.yaml> --extra vars “var1=value var2=value”**

**---**

**- name: extra variable demo**

**hosts: "{{ node }}"**

**tasks:**

**- name: print message**

**ansible.builtin.debug:**

**msg: " fruit is banana"**

ansible-playbook extra.yaml --extra-vars "node=localhost"

**ONE extra variable -> Ansible –playbook <playbookname.yaml> --extra vars “var1=value”**

6. **master slave connectivity – steps and understanding**

master slave method in jenkins

Install Java on master node.

Install Jenkins on master node.

Install java on slave node.

Create a user and ssh keys on slave node. - SSH-KEYGEN

Setting up the remote server - ssh-copy-id (ne slave ip addr5ess)

copy the public key on master node

Join slave node to master.

Test the setup.

8) **different ansible play book commands**

1) with\_items – install multiple packages with the same task ex-https.https,java,etc

2) any\_errors\_fatal – if any task on the playbook fails then abort the entire playbook

3) tags – which is used to call specific tasks in the playbook which has been tagged with some name

4) Idempotent – check whether the given tasks has been performed or not , if not then only it will do it

5) handlers -

9) **creating new file and making it as inventory**

1) Create a file as inventory (as like as hosts(inventory) file) -🡪 etc/ansible/hosts

Converting the normal file as inventory file ex-**abc**

1. **ansible –playbook -i <inventory\_filename> <playbook\_name>**

inventory file – contains all the machine details

10) **handlers concept**

- name: Install Apache on RHEL server

hosts: webserver

tasks:

- name: Install the latest version of Apache

dnf:

name: httpd

state: latest

notify:

- Start Apache

handlers:

- name: Start Apache

service:

name: httpd

state: started

1. call the handler only when the task is successful
2. if not successful then the handlers can’t be called