ANSIBLE

**Steps for installing ansible**

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

python3 get-pip.py --user

python3 -m pip -V

python3 -m pip install --user ansible

python3 -m pip install --user ansible-core

python3 -m pip install --upgrade --user ansible

or

sudo dnf install ansible

ssh-keyscan -H 172.31.19.27 >> ~/.ssh/known\_hosts

make these 2 files manually inside /etc/ansible

-ansible.cfg

-hosts

cat ansible.cfg

[defaults]

inventory = /etc/ansible/hosts

interpreter\_python = /usr/bin/python3

[all:vars]

ansible\_python\_interpreter=/usr/bin/python3.9

cat hosts

[webservers]

ansi2 ansible\_host=ip-172-31-80-195 ansible\_ssh\_user=ec2-user ansible\_ssh\_private\_key\_file=/home/ec2-user/.ssh/id\_rsa ansible\_ssh\_common\_args='-o StrictHostKeyChecking=no'

[local]

localhost ansible\_connection=local

-------------

If we want to change ip/hostname to some other name then use below command

sudo hostnamectl set-hostname ansijen

/bin/bash

Some sample playbooks

1. **For pinging and displaying a msg**

- name: First basic playbook

hosts: all

tasks:

- name: Test connectivity

ping:

- name: "print output"

debug: msg="Alright!"

1. **for installing and starting service on both local and remote server**

- name: Install and start the service

hosts: all

become: yes

tasks:

- name: Installing nginx

yum:

name: nginx

state: present

- name: Starting the nginx

service:

name: nginx

state: started

enabled: true

1. **for copying files to dest**

---

- name: copy files to remote

hosts: all

tasks:

- name: copying files

copy:

src: /home/ec2-user/nik.txt

dest: /home/ec2-user/

owner: nikita

group: nikita

mode: 0777 or ugo=rwx

backup: true

1. **file and directory creation/deletion**

---

- name: file module

hosts: all

tasks:

- name: file creation

file:

path: /home/ec2-user/new.txt

state: touch or absent(for removing)

- name: creating a directory

file:

path: /home/ec2-user/myfolder

state: directory or absent(for removing)

1. **for changing permission**

---

- name: perm change

hosts: all

tasks:

- name: changing

file:

path: /home/ec2-user/nik.txt

mode: u=rw,g=r,o=r

1. **script running on remote**

make a test script on remote machines

eg: test.sh

#!/bin/bash

echo "Hey Girl :)"

touch testfile.txt

now make playbook in local machine

---

- name: Run a script

hosts: all

tasks:

- name: run script

shell: /home/ec2-user/test.sh >> test.log (below steps are req when we need to make logs in diff location)

args:

chdir: /home/ec2-user/myfolder/

creates: test.log

1. **to schedule a cron in remote machines**

install cron in server using below command

sudo yum install cronie

sudo systemctl start crond

---

- name: Cron setup

hosts: all

tasks:

- name: add cron job

cron:

name: Run test script

minute: 15

hour: 00

day: "\*"

month: "\*"

weekday: "\*"

user: ec2-user

job: /home/ec2-user/test.sh (make sure that this script is present in remote servers)

1. **To remove cron from all the remote servers**

---

- name: To remove a cron

hosts: all

tasks:

- name: remove

cron:

name: Run test script (this name should be same which we kept for adding)

state: absent

user: ec2-user

1. **To comment the crons**

---

- name: Cron setup

hosts: all

tasks:

- name: add cron jov

cron:

name: Run test script

minute: 15

hour: 00

day: "\*"

month: "\*"

weekday: "\*"

user: ec2-user

job: /home/ec2-user/test.sh

disabled: yes

1. **For creating user and group in remote**

---

- name: User management

hosts: all

become: yes

tasks:

- name: user creation

user:

name: nick

comment: for devops team

shell: /bin/bash

groups: QA, nick

1. **To remove user and all data related to it**

---

- name: User management

hosts: all

become: yes

tasks:

- name: user creation

user:

name: nick

comment: for devops team

shell: /bin/bash

state: absent

remove: yes

1. **To change password of user**

- name: Set pass

hosts: all

become: yes

tasks:

- name: Password

user:

name: nick

update\_password: always

password: "{{'abc123' | password\_hash('sha512')}}" (this is for encrypting the password)

1. **To kill a process on servers**

---

- name: Kill a process

hosts: all

become: yes

tasks:

- name: find a process & kill it

ignore\_errors: yes

shell: "pgrep nginx | xargs kill"

1. **To kill and start process on servers**

---

- name: Kill a process

hosts: all

become: yes

tasks:

- name: find a process & kill it

ignore\_errors: yes

shell: "pgrep nginx | xargs kill"

- name: start the service

service:

name: nginx

state: started

1. **To download a file from internet to all the machines**

---

- name: Download files

hosts: all

become: yes

tasks:

- name: download

get\_url:

url: https://www.python.org/ftp/python/3.13.2/python-3.13.2-amd64.exe

dest: /home/ec2-user/

1. **To enable firewall service or port**

sudo yum remove -y firewalld ( to install firewall)

sudo yum install -y firewalld

sudo systemctl enable --now firewalld

---

- name: Firewall changes

hosts: all

become: yes

tasks:

- name: enable a service in firewalld

firewalld:

port: 80/tcp

permanent: true

state: enabled

- name: reload the firewalld

service:

name: firewalld

state: reloaded

NOTE: There are some adhoc commands which we can use without making playbook

Eg: ansible all -m command -a "free -h"

If we want to run only a specific task and not all the tasks mentioned in playbook then we use tags

This we need to define in playbook

Eg:

- name: Install and start the service

  hosts: all

  become: yes

  tasks:

  - name: Installing nginx

    yum:

     name: nginx

     state: present

    tags: i-nginx

  - name: Starting the nginx

    service:

      name: nginx

      state: started

      enabled: true

    tags: ss-nginx

now on terminal run

ansible-playbook app.yml -t i-nginx

**VARIABLES** in ansible

Eg:

- name: Install and start the service

  hosts: all

  become: yes

vars:

- app: nginx

  tasks:

  - name: Installing nginx

    yum:

     name: “{{ app }}”

     state: present

    tags: i-nginx

  - name: Starting the nginx

    service:

      name: “{{ app }}”

state: started

      enabled: true

    tags: ss-nginx

**HANDLERS** in ansible

**Handlers** in Ansible are special tasks that only run **when notified** by another task. They are commonly used for actions like **restarting services** after configuration changes.

If we want to run 2nd task only when 1st task is done

Eg:

- name: Firewall changes

  hosts: all

  become: yes

  tasks:

  - name: enable a service in firewalld

    firewalld:

      port: 80/tcp

      permanent: true

      state: enabled

    notify:

      - reload the firewalld

  handlers:

  - name: reload the firewalld

    service:

      name: firewalld

      state: reloaded

**CONDITION**

**Conditions** in Ansible allow you to run tasks **only when** certain criteria are met. This is done using the when statement.

**Eg: Check if file exist**

- name: Delete a file only if it exists

  file:

    path: /tmp/testfile

    state: absent

  when: ansible\_facts['os\_family'] == "Debian" and ansible\_facts['distribution\_version'] == "10"

This will **delete /tmp/testfile only on Debian 10**.

**LOOP**

Loops in Ansible allow you to **repeat** a task multiple times with different values, reducing redundancy.

Eg:

- name: user add

  hosts: all

  become: yes

  tasks:

  - name: user creation

    user:

      name: "{{ item }}"

      comment: for devops team

      shell: /bin/bash

    loop:

     -  Ram

     -  Shyam

     -  Poo