Ansible

What is Ansible?

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It is designed to simplify the process of managing large-scale IT environments, making it easier to deploy and manage applications and infrastructure consistently.

Create Environment to Practice Ansible

- **Create two ec2 instance ansible server and node server**
- **To install ansible in ansible server**

  Userdata script:

  #!/bin/bash

  sudo su

  sudo apt update

  sudo apt install software-properties-common

  sudo add-apt-repository --yes --update ppa:ansible/ansible

  sudo apt install ansible -y

- **Connect to the ec2 instances**
- **Generate a key for passwordless ssh on both servers**

  ssh-keygen

  cd ~/.ssh/

  Copy public key of ansible server and paste in node server

  Ansible server:

  cat <public-key> #copy key

  node server:

  sudo vi authorized_keys # paste key in the file and save

- **Configure Ansible server**

  cd /etc/ansible
  ls
  sudo vi hosts
  [webserver] #node group name
  <node-ip>

add node ip in hosts file and save

sudo vi ansible.cfg
uncomment the inventory and sudo user from defaults and save

Run Playbooks

What is playbook?

In Ansible, a playbook is a file that defines a set of tasks to be executed on a group of hosts. Playbooks are written in YAML (Yet Another Markup Language) and serve as the central mechanism by which Ansible configurations, deployments, and orchestration are defined. They allow you to specify automation instructions in a human-readable format.

- **Playbook create**

  sudo mkdir Playbooks
  sudo vi apache_install

  yaml

```
- name: Ensure web server is installed
  hosts: demo
  become: yes
  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: present
      when: ansible_os_family == "Debian"

    - name: Ensure Apache is running
      service:
        name: apache2
        state: started
```

```
        enabled: yes
```

- **Run a playbook**

  ansible-playbook Playbook/apache_install

copy files

make sure the file is present at path

```
---
- name: Copy file and create user in dev group
  hosts: webservers
  become: yes  # Ensure the task runs with sudo privileges
  tasks:
    - name: Copy file to remote host
      copy:
        src: /home/ubuntu/myfile.txt
        dest: /home/ubuntu/myfile.txt
        owner: appu
        group: dev
        mode: '0644'
        backup: true
```

create and delete file and permissions

Make sure the user is created in node

```
---
- name: create file in remote server
  hosts: webservers
  become: yes
  tasks:
```

```
    - name: create file
      file:
        path: /home/ubuntu/new_file.txt
        state: absent
        owner: appu
        group: appu
        mode: u=rwx,g=r,o=r

  - name: create directory
      file:
        path: /home/ubuntu/playbookfiles
        state: directory
```

cron-job

create test.sh file in node

test.sh

```
#!/bin/bash
echo “hello buddy”
touch  testfile
```

give execute permissions

sudo chmod u+x test.sh

sudo vi Playbooks/cron_job.yaml

```
---
- name: cron jobs
  hosts: webservers
  tasks:
    - name: cron jobs
      cron:
        name: cron job for remote server
```

minute: 30

hour: 18

day: 15

month: "*"

weekday: "*"

job: /home/ubuntu/test.sh

Download files

Make sure the user is created in node

- name: Downlaod files
  hosts: all

  tasks:
  - name: Download file
    get_url:
      url: https://www.python.org/ftp/python/3.12.2/Python-3.12.2.tar.xz
      dest: /tmp/script/
      owner: appu
      group: appu
      mode: 0777

Ansible Ad-hoc commands

Ansible ad hoc commands are one-time command-line instructions used to perform tasks on remote hosts without the need to write and run a playbook. They are useful for quick tasks, debugging, and testing. These commands are executed directly from the command line and allow you to perform tasks such as installing packages, managing files, and restarting services on remote hosts.

Syntax:

ansible <host-pattern> -m <module> -a "<module arguments>"

-u <username> -b

- -m : module name
- -a : arguments
- -u : username
- -b : runs the command as sudo user
- If you need to give password for sudo user **--ask-become-pass**

e.g: ansible webservers -m apt -a "name=apache2 state=present" --become --ask-become-pass

Commands:

Ping

ansible webservers -m ping

```
ubuntu@ip-172-31-29-135:/etc/ansible$ ansible webservers -m ping
172.31.29.100 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
```

ls

ansible webservers -a "ls"

```
ubuntu@ip-172-31-29-135:/etc/ansible$ ansible webservers -a "ls"
The authenticity of host '172.31.29.100 (172.31.29.100)' can't be established.
ED25519 key fingerprint is SHA256:GjcU7tWFqLCVceoIcPjeYuvi3aWaY1SsL414MIOe5kg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
172.31.29.100 | CHANGED | rc=0 >>
file1
file2
```

df –h

ansible webservers -m command -a "df -h"

```
ubuntu@ip-172-31-29-135:/etc/ansible$ ansible webservers -m command -a "df -h"
172.31.29.100 | CHANGED | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        19G  1.8G   17G  10% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  876K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M   76M  744M  10% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   12K   96M   1% /run/user/1000
```

Copy

ansible webservers -m copy -a "src=/home/ubuntu/testfile.txt dest=/home/ubuntu"

```
ubuntu@ip-172-31-29-135:/etc/ansible$ touch /home/ubuntu/testfile.txt
ubuntu@ip-172-31-29-135:/etc/ansible$ echo "hello buddy" > /home/ubuntu/testfile.txt
ubuntu@ip-172-31-29-135:/etc/ansible$ ansible webservers -m copy -a "src=/home/ubuntu/testfile.txt dest=/home/ubuntu"
172.31.29.100 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "checksum": "8acde34d5b3caefd0d47674cf682054d7a4980e0",
    "dest": "/home/ubuntu/testfile.txt",
    "gid": 1000,
    "group": "ubuntu",
    "md5sum": "9435e2e820cfcd9bc51ab4588dea3805",
    "mode": "0664",
    "owner": "ubuntu",
    "size": 12,
    "src": "/home/ubuntu/.ansible/tmp/ansible-tmp-1716647370.5215979-3122-91354947145763/source",
```

Shell script

Make sure script is present in node server

ansible webservers -m shell -a "/home/ubuntu/test.sh"

```
ubuntu@ip-172-31-29-135:/etc/ansible$ ansible webservers -m shell -a "/home/ubuntu/test.sh"
172.31.29.100 | CHANGED | rc=0 >>
"hello buddy"
```

Variables

What is variables?

In Ansible, variables are a fundamental feature that allows you to parameterize your playbooks, roles, and tasks. They enable you to define dynamic values that can be reused throughout your automation scripts, making them more flexible, readable, and easier to maintain. Variables can be defined in multiple places and have different scopes, providing powerful ways to manage complex configurations.

Create a Playbook

```
---
- name: Install and configure Apache on Ubuntu
  hosts: webservers
  become: yes
  vars:
    apache_package: apache2
    apache_service: apache2

  tasks:
    - name: Ensure Apache package is installed
      apt:
        name: "{{ apache_package }}"
        state: present
        update_cache: yes

    - name: Ensure Apache service is started and enabled
      service:
        name: "{{ apache_service }}"
        state: started
        enabled: yes
```

Handlers

In Ansible, handlers are special tasks that are triggered by other tasks using the notify directive. Handlers are typically used to perform actions that should only occur if a change was made by a task, such as restarting a service after a configuration file has been modified.

Create a playbook

Make sure index.html file is present at path given

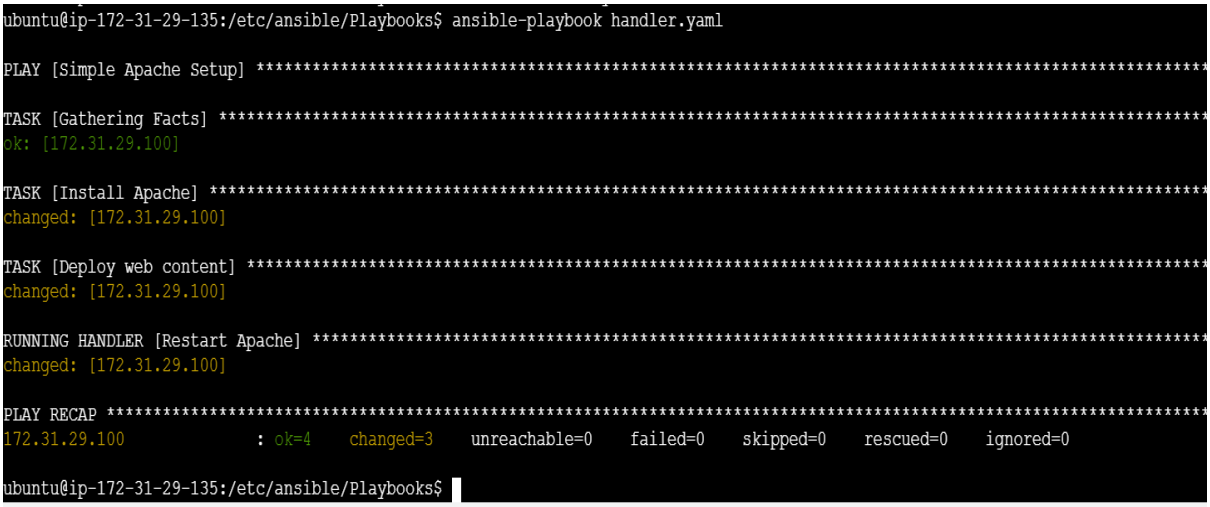
```
---
- name: Simple Apache Setup
  hosts: webservers
  become: yes

  vars:
    apache_package: apache2
    apache_service: apache2
    web_content: /var/www/html/index.html
    web_content_src: /home/ubuntu/index.html

  tasks:
    - name: Install Apache
      apt:
        name: "{{ apache_package }}"
        state: present

    - name: Deploy web content
      copy:
        src: "{{ web_content_src }}"
        dest: "{{ web_content }}"
      notify: Restart Apache
```

```
handlers:
  - name: Restart Apache
    service:
      name: "{{ apache_service }}"
      state: restarted
```



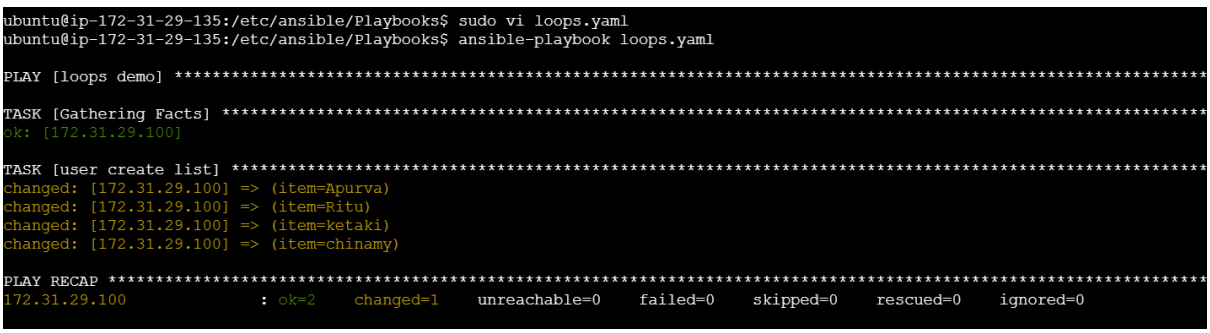
Loops

What is loops?

In Ansible, loops allow you to perform repetitive tasks multiple times with different items. This is particularly useful when you need to apply the same task to a collection of items, such as installing multiple packages, creating multiple users, or processing multiple files. Loops enhance the flexibility and efficiency of your playbooks by reducing redundancy and promoting code reuse.

Create a playbook

```
---
- name: loops demo
  hosts: webservers
  become: yes
  tasks:
    - name: user create list
      user:
        name: "{{item}}"
        state: present
      with_items:
        - Apurva
        - Ritu
        - ketaki
        - chinamy
```



Conditions

What is conditions?

In Ansible, conditions are used to control the execution of tasks based on specific criteria. These conditions allow you to run or skip tasks depending on the state of the system, the values of variables, or other factors. The most common way to apply conditions in Ansible is by using the ‘when’ clause.

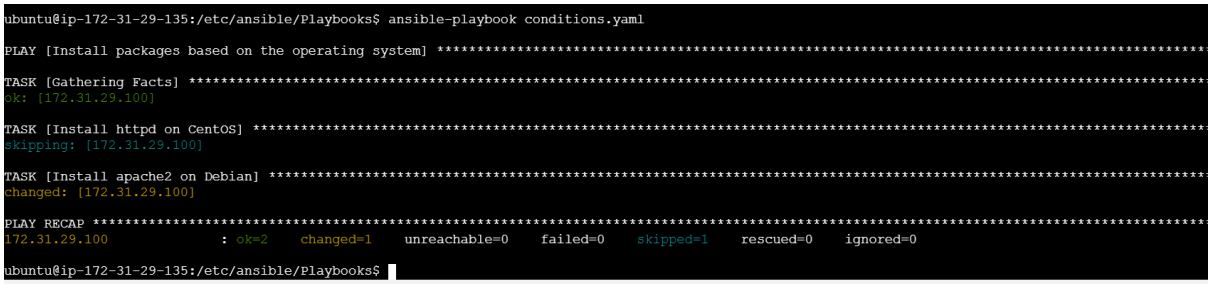
Create a playbook

```
---
- name: Install packages based on the operating system
  hosts: all
  become: yes

  tasks:
    - name: Install httpd on CentOS
      yum:
        name: httpd
        state: present
      when: ansible_facts['os_family'] == "RedHat"

    - name: Install apache2 on Debian
      apt:
        name: apache2
        state: present
      when: ansible_facts['os_family'] == "Debian"
```

- The first task uses the yum module to install httpd (Apache HTTP Server) but only if the host's OS family is RedHat (when: ansible_facts['os_family'] == "RedHat").
- The second task uses the apt module to install apache2 (Apache HTTP Server) but only if the host's OS family is Debian (when: ansible_facts['os_family'] == "Debian").



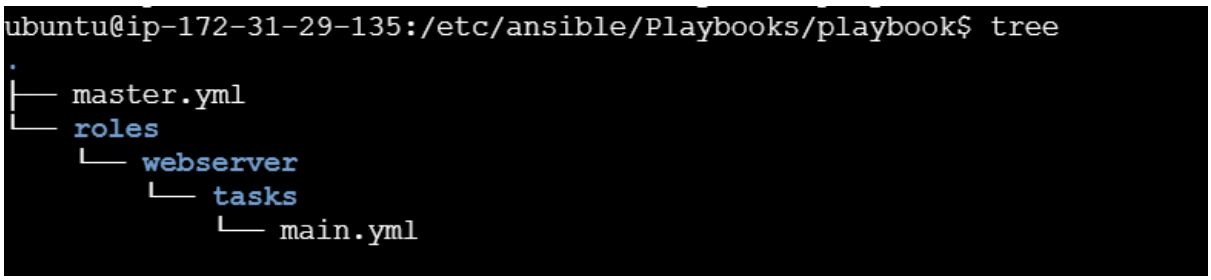
Roles

What is Roles?

In Ansible, roles are a way to organize and structure your playbooks and reusable automation content. Roles allow you to break down your playbooks into smaller, reusable components, making them easier to manage, share, and reuse across multiple projects.

Example:

Create a folder tree like this



Open main.yml

Sudo vi roles/webserver/tasks/main.yml

```
---
- name: apache install in ubuntu
  apt: pkg=apache2 state=latest
```

sudo vi master.yml

```
---
- name: apache installation
  hosts: webservers
  become: yes
  roles:
    - webserver
```

Run a playbook

ansible-playbook master.yml