Ansible

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

1. shell script is homogenous

useradd roboshop --> centos

adduser roboshop --> ubuntu

one shell script can only work one particular distro

2. every time we need to do validations

3. if you have 1000 servers, you need to login and run the script.

4. imperative vs declarative

imperative --> little tough syntax, very strict syntax and follows in sequence

declarative --> easy syntax, no need of sequence, you can write anywhere. whatever you write you will get it, simple syntax

1. ansible can query the server, it can understand what OS it is, based on that it can change the command

2. you no need to write validations

3. ansible can connect any no of servers, no need to login to the server

4. declarative, no need of sequence and easy syntax.

Before Ansible

-----------

puppet

chef

Pull

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

Configuration Server --> chef server

Nodes

-------

1. First nodes will connect to server

2. Pull the configuration

3. RUn the configuration

we need to schedule how frequently nodes should connect to server. like once in 30min

Push

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

Configuration Server --> Ansible server

1. Server will directly push and run the configuration

Push vs Pull

-----------

pull mechansim will have additional agent software

there may not be change in configuration every time, change in configuration is very rare wastage of data.

shell script --> keeping all the commands in a file with variables, conditions, loops, etc

playbook --> keeping all the ansible collections(i.e command in linux) and run it

YAML --> yet another markup language

Installation: yum install ansible

Groups in Inventory:

In Ansible, "group" refers to a logical grouping of hosts. These groups can represent various categories such as roles, environments, or any other organizational structure that makes sense for your infrastructure.

Ex:

web:

hosts:

webserver1:

webserver2:

vars:

http\_port: 80

db:

hosts:

dbserver1:

dbserver2:

vars:

db\_port: 3306

production:

children:

web:

db:

development:

children:

web:

db:

**Adhoc commands:**

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

ansible -i inventory mongodb --become -e ansible\_user=centos -e ansible\_password=DevOps321 -m ansible.builtin.yum -a "name=nginx state=installed"

ansible -i inventory mongodb -b -e ansible\_user=centos -e ansible\_password=DevOps321 -m ansible.builtin.service -a "name=nginx state=started"

**playbooks:** keep all ansible modules/collections in a YAML file and run that file

**Install Nginx Ex:**

- name: install and start nginx

  hosts: mongodb

  become: yes

  tasks:

  - name: install nginx

    ansible.builtin.yum:

      name: nginx

      state: installed

  - name: start nginx

    ansible.builtin.service:

      name: nginx

      state: started

**Variables:**

**Variables form play level Ex:**

- name: printing variables

  hosts: mongodb

  become: yes

  # variables at play level

  vars:

    NAME: kUMARA

    JOB: DevOps

  tasks:

  - name: print hello world

    ansible.builtin.debug:

      msg: "hello world"

  - name: printing variables

    ansible.builtin.debug:

      msg: "my name is {{NAME}}, my job role is {{JOB}}"

**Variables from file Ex:**

**#**variables.yaml

NAME: kUMARA

JOB: DevOps

**#**VarsFormFile.yaml

- name: variables from file

  hosts: mongodb

  vars\_files:

    - variables.yaml

  tasks:

    - name: print my details using vars from file

      ansible.builtin.debug:

        msg: "my name is {{NAME}}, my job is {{JOB}}"

**Variables from Prompt Ex:**

- name: variables from prompt

  hosts: mongodb

  vars\_prompt:

    - name: NAME

      prompt: "what is your name?"

      private: no

    - name: JOB

      prompt: "what is your job role?"

      private: yes

  tasks:

  - name: print values from prompt

    ansible.builtin.debug:

      msg: "my name is {{NAME}}, job is {{JOB}}"

**Variables from Task Level Ex:**

- name: varibles from task level

  hosts: mongodb

  vars:

    NAME: raja

    JOB: DevOps

  tasks:

  - name: printing variables from play level

    ansible.builtin.debug:

      msg: "my name is {{NAME}}, Job is {{JOB}}"

  - name: variables from task level

    vars:

      NAME: Kumara

      JOB: Cloud

    ansible.builtin.debug:

      msg: "my name is {{NAME}}, Job is {{JOB}}"

**Variables from Inventory Ex:**

#invertory

[mongodb]

172.31.29.72

[mongodb:vars]

SERVER\_NAME=mongodb\_server

#vars from inventory:

- name: vars from inventory

  hosts: mongodb

  tasks:

    - name: varible from inventory

      ansible.builtin.debug:

        msg: "server name is {{SERVER\_NAME}}"

**Variables from Command Line Ex:**

- name: vars from commandline

  hosts: localhost

  tasks:

  - name: variable from command prompt

    ansible.builtin.debug:

      msg: "my name is {{NAME}}"

ansible-playbook -i inventory -e ansible\_user=centos -e ansible\_password=DevOps321 -e NAME=Raja 07-Varsfromcommandline.yaml

**Variable precedence:**

1. Command line variables

2. Task Level Variable

3. Variable from files

4. Prompt

5. Play

6. Inventory

7. Roles

**Data Types Ex:**

- name: ansible variable data types

hosts: localhost

vars:

- AGE: 30 #Number

- NAME: "Sivakumar" #String

- isDevOps: true #boolean

- Skills: #list

- DevOps

- AWS

- Docker

- EXPERIENCE: #map

DevOps: 7

AWS: 5

Docker: 4

tasks:

- name: print number variable

ansible.builtin.debug:

msg: "{{AGE}}"

- name: print String variable

ansible.builtin.debug:

msg: "{{NAME}}"

- name: print Boolean variable

ansible.builtin.debug:

msg: "{{isDevOps}}"

- name: print List variable

ansible.builtin.debug:

msg: "{{Skills}}"

- name: print Map variable

ansible.builtin.debug:

msg: "{{EXPERIENCE}}"

**Conditions:**

when expression is true then module/collection will run, otherwise it will not run.

**EX:**

- name: checking user exit or not

  hosts: localhost

  tasks:

  - name: print the user name

    ansible.builtin.command: id Kumara

    register: output

    ignore\_errors: true

  - name: print the output

    ansible.builtin.debug:

      msg: "user name is {{output}}"

  - name: print the user name

    ansible.builtin.command: id Raja

    register: output1

    ignore\_errors: true

  - name: print the output

    ansible.builtin.debug:

      msg: "user name is {{output1}}"

  - name: create user with when cindition

    become: yes

    ansible.builtin.user:

      name: Raja

    when: output1.rc != 0

**Loops:**

**EX1:**

- name: using loops

  hosts: localhost

  tasks:

  - name: printing name using loops

    ansible.builtin.debug:

      msg: "hello {{item}}"

    loop:

      - raja

      - pooja

**EX2:**

- name: installing packages

  hosts: localhost

  become: yes

  tasks:

  - name: install nginx and postmap

    ansible.builtin.yum:

      name: "{{item}}"

      state: installed

#list of string

    loop:

    - nginx

    - postfix

**Ex3:**

- name: map of list

  hosts: localhost

  become: yes

  tasks:

  - name: install and uninstall packages

    ansible.builtin.yum:

      name: "{{item.name}}"

      state: "{{item.state}}"

  #list of map

    loop:

      - {name: 'nginx', state: 'installed'}

      - {name: 'postfix', state: 'absent'}

**Roles:**

Ansible roles are a way to organize and structure your Ansible playbook into reusable units. They provide a convenient method to encapsulate tasks, handlers, variables, and other configuration elements into modular components.

**Template:**

**template** feature in Ansible makes it easy to create files with dynamic content, allowing you to automate the generation of configuration files, scripts, or any other text-based files in your infrastructure.

**Handlers:**

In Ansible, handlers are tasks that are only executed when a task notifies them. They are typically used to trigger actions in response to changes, such as restarting a service or reloading a configuration file.

**Ansible.cfg**

he ansible.cfg file is a configuration file used to customize the behavior of Ansible. It allows you to set various options that affect how Ansible operates, such as default inventory file locations, SSH settings, logging options, and more.

**FEW Ansible Modules:**

ansible.builtin.yum:

ansible.builtin.deburg:

ansible.builtin.user

ansible.builtin.command:

ansible.builtin.shell