**ANSIBLE**

**What is configuration management?**

**Configuration**: It means each and every minor detail of your machine or specification of your machine.

E.g. Ram, hard disk, how many users and groups are there, what kind of packages your running, how many services are there?

**Management**: add, edit, delete etc.

Adding user, installing package means managing the configuration of your machine.

There are two ways by which we can manage our configuration

1) Manual: means system admin do the task

2) Automated: means using tool like Ansible, Chef and Puppet we do task.

Devops engineer manage configuration of your machine using above tools in automated manner.

Ansible is a configuration management or administration tool.

Configuration management tool turn your code into infrastructure,

So your code would be Transferable, testable and version able.

Ansible works on push mechanism whereas chef works on pull mechanism.

**Architecture of Ansible**

In my company 100 machines are there we call them as node or host.

A requirement is to create some 100 files in each server.

To do so,

Create EC2 instance and install ansible package. Python is pre-requisite to install ansible package.

Create one file in ansible server and write code in YAML script to do your required task, this file we called "**Playbook**"

Connect ansible server with nodes through SSH.

Server push playbook in nodes, before playbook run Setup module go in note and check the existing configuration of nodes and if file is already there playbook won't get executed or file is not there playbook get executed.

Not repeating the same task again and again is call **idempotency**.

**Advantages of Ansible**

1) Open source and free to use

2) Lightweight required very less resources

3) Secure

4) Agent less

5) Push mechanism

**Disadvantages of Ansible**

1) New to the market so less documentation available.

2) Not 100% automation need to push playbook.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­­­­­­­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Lab**

1) First create 3 Linux EC2 instance with **SSH** and **HTTP** port. 1 as ansible server and 2 as host

2) Download epel repository in ansible server

**#wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm**

3) Install epel package in ansible server: **yum install epel..... -y**

4) Update ansible server: **yum update -y.**

5) Install packages in ansible server: **yum install git paython python-devel python-pip openssl ansible -y.**

To see ansible package install or not

**#which ansible**

To see ansible version

#**ansible --version**

ansible 2.9.27

Create group of host machine IP in below file also called as inventory file

**#vim /etc/ansible/hosts**

[webserver]

Node 1 private IP

Node 2 private IP

“[] You can create any number of groups”

“If server and host both are in same region then give private IP otherwise public IP”

Uncomment inventory and sudo\_user (ansible blocking sudo privilege’s to root user) in below file

**#vim /etc/ansible/ansible.cfg**

“If you forget to uncomment then configuration file won't recognized host file”

Create one user in server as well as host and assign password

**#useradd ansadmin**

**#passwd ansadmin**

Add ansadmin user in **/etc/sudoers**

**#visudo** OR **#visudo /etc/sudoers**

ansadmin ALL=(ALL) NOPASSWD: ALL

WE HAVE CREATED USER AND PROVIDED ROOT PREVELIGES TO USER IN SERVER AS WELL AS HOST.

Now simply switch user using **su - ansadmin** and use **sudo** word before executing any command.

Edit configuration file

**#vim /etc/ssh/sshd\_config**

Uncomment **PermitRootLogin** **yes** and **PasswordAuthentication yes**

Reload or restart service

**#systemctl reload sshd.service**

Now connect to node from ansible server using following command

**#ssh <Private IP of host>**

It will ask password.

To avoid providing password again and again build trust relationship.

Root to root or user to user

Not between root to user.

Being ansadmin user generate key

**#ssh-keygen**

Being present in ansible server copy key to the host using following command

**#ssh-copy-id <host username>@<host private ip>**

Host pattern

To see all nodes Ip

**#ansible all --list-host**

To see particular grp nodes Ip

**#ansible <group name> --list-host**

To see particular Ip from grp nodes up

**#ansible <grp>[number] --list-host**

Number = 0, 1, 2 to -1(last IP in group) -2(second last IP in group)

<Grp>[1:4] = 1 to 4 host IP in group

To see multiple group nodes Up

**#ansible <grp1>:<grp2> --list-host**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**We can push code in 3 ways**

1) Ad-hoc command

2) Module

3) Playbook

1) Ad-hoc command is simply Linux commands used for single task.

There is no idempotency in Ad-hoc command.

Ad-hoc commands are commands which can be run individually to perform quick functions.

These ad-hoc commands are not used to for confirmation management and deployment because this command is of one time user.

ls in single group

**#ansible webserver -a "ls"**

-a = action/perform action inside " "

**#ansible all -a "ls -al"**

-a = action/perform action inside " "

Create file in single host of group

**#ansible webserver[0] -a "touch file1"**

-a = action/perform action inside " "

Create file in all group

**#ansible all -a "touch file1"**

-a = action/perform action inside " "

To install package

**#ansible all -a "sudo yum install httpd -y"**

OR

**#ansible all -ab "yum install httpd -y"**

-a = action/perform action inside " "

-b = become root

If you have to ignore warning messages then pass below entry in ansible configuration file

**#sudo vim /etc/ansible/ansible.cfg**

**command\_warnings=False**

**interpreter\_python=auto\_silent**

2) Module is written in YAML used to run single command

There is idempotency in module command

ls in single group

**#ansible webserver -m command -a "ls"**

**#ansible all -m command -a "ls -al"**

Create file in single host of group

**#ansible webserver[0] -m command -a "touch file1"**

Create file in all group

**#ansible all -m command -a "touch file1"**

Here command is module it is default module where u mention or not.

To install package

**#ansible all -b -m yum -a "pkg=httpd state=present"**

-a = action/perform action inside " "

-b = become root

-m = module

**Install=present**

**Uninstall=absent**

**Update=Latest**

**#ansible all -b -m yum -a "pkg=httpd"**

**#ansible all -m yum -a "name=httpd state=present " -b**

To start service

**#ansible all -b -m service -a "name=httpd state=started"**

To create user

**#ansible all -b -m user -a "name=u1" –b**

**OR**

**#ansible all –m ansible.builtin.user –a “name=rahul” -b**

To copy

**#ansible all -b -m copy -a "src=/file dest=/tmp"**

INSTEAD OF ALL YOU CAN ADD GROUP NAME AS WE DID IN ABOVE COMMANDS

3) Playbook is written in YAML used to run multiple command/module.

Playbooks in ansible are written in YAML format.

It is human readable data serialization language.

It is commonly used for configuration files.

Playbook is like files where you write a code consist of target, variable, tasks, handler, files, templates and roles.

Playbooks are divided into many sections like -

Target section: Define the host against which Playbook task has to executed (group, node)

Variable section: Define variable

Task section: list of all modules that we need to run, in an order.

YAML

For ansible nearly every YAML file start with a list.

Each item in the list is list of key-value pairs commonly called dictionary.

Eg. Key = name & value = Rahul

Key = job & value = engineer

All YAML file have to be begin with "---" and end with "...". End is not mandatory.

All member of a least line must begin with same indentation level starting with "-"

Eg.

--- # A list of fruit s

Fruits:

- mango

- apple

- banana

...

Dictionary is represented in simple key value form

Eg.

--- # A list of customer

Customer:

name: rahul

job: engine

exp: 5years

...

Extension of playbook is .yml and there should be space between: and value.

**Lab**

#vim target.yml

--- # Target Playbook

- host: webserver

  user: ansible

  become: yes

  connection: ssh

  gather\_facts: yes

...

Here host in my group, I want perform task being ansible user, become means give sudo privilege, connect between ansible server and host is SSH, get facts and display.

To execute playbook

#ansible-playbook target.yml

Create playbook in ansible server to install httpd package

#vim target.yml

--- # Target Playbook

- host: webserver

  user: ansible

  become: yes

  connection: ssh

  tasks:

  - name: install https on Linux

    actions: yum name=httpd state=installed/present

...

Create playbook in ansible server to install httpd package with variable

#vim var.yml

--- # Variable Playbook

- host: webserver

  user: ansible

  become: yes

  connection: ssh

vars:

      pkgname: httpd

tasks:

      - name: install https on Linux

      actions: yum name='{{pkgname}}' state=installed/present

**Handler section**

A handler is exactly the same as a task but it will run when called by previos (dependent on previous task)

Create playbook in ansible server to install httpd package and start https service with help of handler

#vim handler.yml

--- # Handler Playbook

- host: webserver

  user: ansible

  become: yes

  connection: ssh

  tasks:

      - name: install https on Linux

      actions: yum name=httpd state=installed/present

      notify: restart HTTPD

  handler:

      - name: restart HTTPD

        action: service name=httpd state=restarted

DryRun

Check whether the Playbook is formatted correctly and it is working fine or not in server.

#ansible-playbook handler.yml --check

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Loops**

Repeating the task again and again then we go for loops concept.

Create playbook in ansible server to create multiple user

#vim loops.yml

--- # loop Playbook

- host: webserver

  user: ansible

  become: yes

  connection: ssh

  tasks:

      - name: add list of user

        user: name = "'{{item}}' state=present

        with items:

          - u1

          - u2

          - u3

**Condition**

When we have different scenarios, we put conditions according to the scenario.

When is used for condition.

Create playbook in ansible server with condition

#vim condition.yml

-- # Condition Playbook

- host: webserver

  user: ansible

  become: yes

  connection: ssh

  tasks:

      - name: install apache2 on debian

        command: apt-get -y install apache2

        when: ansible\_os\_familt=="Debian"

      - name: install https on Linux

        command: yum install httpd -y

        when: ansible\_os\_familt=="RedHat"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Vault**

It is used to encrypt you playbook

Create a new encrypted playbook

#ansible-vault create vault.yml

Now it will ask to assign password

To view encrypted playbook

#ansible-vault view vault.yml

Edit an encrypted playbook

#ansible-vault edit vault.yml

Now it will ask password to enter

Change password of encrypted playbook

#ansible-vault rekey vault.yml

To encrypt an existing playbook

#ansible-vault encrypt target.yml

To decrypt an existing playbook

#ansible-vault decrypt target.yml

**Roles**

Roles are good for organizing task

We can organize Playbook into a directory structure called roles.

Types of roles:

Default

File

Handler

Meta

Template

Tasks

Vare

**Lab 1**

#mkdir -p playbook/roles/webserver/tasks

#cd playbook

#touch roles/webserver/tasks/main.yml

#touch master.yml

#vim roles/webserver/tasks/main.yml

-name: install HTTPS

 yum: pkg=httpd state=latest

esc ; wq!

#vim master.yml

- host: webserver

  user: ansible

  become: yes

  connection: ssh

  roles

     - webserver

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Lab 2**

To create role directory structure first create one directory

#mkdir playbooks

#cd playbooks

Now create roles directory structure

#ansible-galaxy init <dirname>

Eg. #ansible-galaxy init role

If you do tree you will see so many section with main.yml file ( vars, tasks, files, handlers etc.)

Now create task in tasks main.yml file

Handler in handlers main.yml file

Put file in files main.yml file

Now created one master.yml file in playbooks directory

- host: webserver

  user: ansible

  become: yes

  connection: ssh

  roles

     - role

Now u can run master playbook and your master playbook will run role one by one