ANSIBLE

Ansible is an opensource IT configuration management, deployment and orchestration tool.

It aims to provide large productivity gains to a wide variety of automation challenges.

Ansible History:

- Michael Dehaan developed ansible and the ansible project began in February 2012.
- Redhat acquired the ansible tool in 2015.
- Ansible is available for RHEL, Debian, cent OS and oracle Linux.
- You can use this tool whether your servers are in on-premises or in cloud.
- It turns your code into infrastructure i.e. your computing environment has some of the same attributes as your application.

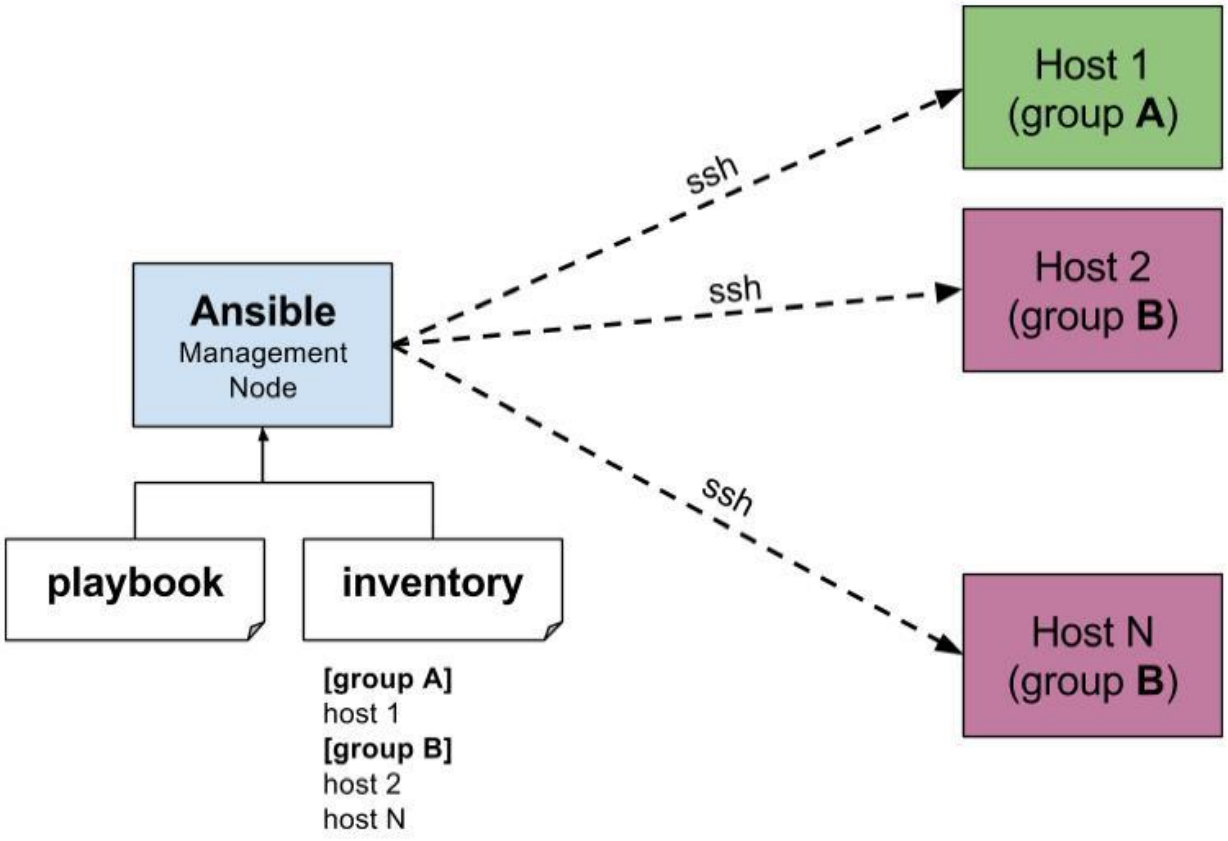
Advantages:

- Ansible is free to use by everyone.
- Ansible is very consistent and light weight and no constrains regarding the OS or underlying hardware are present.
- It is very secure due to its agentless capabilities and open SSH security features.
- Ansible doesn't need any special system administrator skills to install and use it.
- It is push mechanism.

Disadvantages:

- Insufficient user interface, though ansible tower is GUI, but it is still in development stage.
- Cannot achieve full automation by ansible.
- New to the market, therefore limited support and document is available.

Ansible Architecture



Terms used in Ansible:

a. Ansible Server: the machine where ansible is installed and from which all tasks and Playbooks will be run.
b. Module: basically, a module is a command or set of similar commands meant to be executed on the client side.
c. Task: a task is section that consist of a single procedure to be completed.
d. Role: a way of organizing tasks and related files to be later called playbook.
e. Fact: information fetched from the client form the global variables with the gather facts operation.
f. Inventory: file containing data about the ansible client servers.
g. Play: execution of playbook.
h. Handler: task which is called only if notifier is present.
i. Notifier: section attributed to a task which calls a handler if the output is changed.
j. Playbooks: it consists code in YAML format which describes tasks to be executed.
k. Host: nodes which are automated by ansible.

Go to AWS account- create 3 EC2 instances in same AZ.

Take access of all machines via putty.

Now go inside ansible server and download ansible package

wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm

Now do #ls

yum install epel-release-latest-9.noarch.rpm

#yum update -y

Now we have to install all the packages one by one

sudo yum install ansible sudo yum install python

sudo yum install git

Let's rename Master server and Node 1 and Node2,their names

[ec2-user@ip172-31-91-98] $ sudo hostnamectl set-hostname ansible-master

[ec2-user@ip172-31-80-251] $ sudo hostnamectl set-hostname ansible-Node1

[ec2-user@ip172-31-82-233] $ sudo hostnamectl set-hostname ansible- Node2

[ec2-user@ip172-31-91-98] $ hostname

ansible-master ansible- Node1 ansible-

Node2

Now go to host file inside ansible server and paste private ip of node1 and node2

vi /etc/ansible/hosts

Now this host file is only working after updating ansible.cfg file

vi /etc/ansible/ansible.cfg

[defaults]

inventory = /etc/ansible/hosts sudo-user

= root

Now create one user, in all the three instances

adduser ansible

Now set password for this user

passwd ansible

Now switch as ansible user

su – ansible

This ansible user don't have sudo priviledges right now. If you want to give sudo ppriviledge to ansible user # visudo

Now go inside this file

Root ALL= (ALL) ALL

(ansible ALL= (ALL) NOPASSWD: ALL) :wq!

Now do this thing in other nodes also.

Now go to ansible server and try to install httpd package as an ansible user.

sudo you install httpd -y

Now establish connection between server and node, go to ansible server

$ ssh 172.31.41.240 o/p-

permission denied

now we have to do some changes in sshd-config file, go to ansible server

vi /etc/ssh/sshd-config

Do some changes and saved the file.

Do this work in node1 and node2 also.

Now verify in ansible server

su -ansible

ssh 172.31.41.240

Now it asks for passwd, enter the password after that you will be inside node1.

Now go to ansible server and create keys.

Run this command as ansible user.

ssh-keygen

ls -a

o/p- .ssh #

cd .ssh/ ls

o/p- id_rsa id_rsa_pub

now I need to copy public key in both the nodes. (ansible@private_ip of node)

ssh-copy-id ansible(username)@172.31.41.240

Ask for password

ssh-copy-id ansible@172.31.41.228

-Ask for password

Now verify, go to ansible server #

cd ..

ssh 172.31.41.240

Now you will enter into node1.

Host Patterns:

vi /etc/ansible/hosts

“all” pattern refers to all the machines in an inventory

Ansible all –list-host

Ansible <group name> --list-hosts

Ansible <groupname>[0] –list-hosts

Groupname[0]- picks first machine of the group

Groupname[1]- picks second machine of the group

Groupname[-1]- picks last machine of the group

Groupname[0:1]- picks first two machines of the group

Groupname[2:5]- picks 3,4,5 and 6 machines of the group

Group separated by colon (:) can be used to use hosts from multiple groups. Groupname1:groupname2

Ad-hoc Commands:

- Ad-hoc commands are commands which can be run individually to perform quick functions.
- These ad-hoc commands are not used for configuration management and deployment, because these commands are of one-time usage.
- The ansible ad-hoc commands uses the /usr/bin/ansible command line tool to automate a single task.

Go to ansible server

$ ansible demo -a “ls”

$ ansible demo[0] -a “touch filez”

$ ansible all -a “touchfile4”

$ ansible demo -a “ls-al”

$ ansible demo -a “sudo yum install httpd -y” or

$ ansible demo -ba “yum install httpd -y”

$ ansible demo -ba “yum remove httpd -y”

Ansible Modules:

- Ansible ships with a number of modules (called module library) that can be executed directly on remote hosts or through “playbooks”.
- Your library of modules can reside on any machine and there are no servers, daemons or databases required.

Q. where ansible modules are stored?

The default location of the inventory file is /etc/ansible/hosts.

$ ansible demo -b -m yum -a “pkg=httpd state=present”

$ ansible demo -b -m yum -a “pkg=httpd state=latest”

$ ansible demo -b -m yum -a “pkg=httpd state=absent”

$ ansible demo -b -m service -a “name=httpd state=started”

$ ansible demo -b -m user -a “name=raj”

$ ansible demo -b -m copy -a “src=file4 dest=/tmp”

$ ansible demo -m setup

$ ansible demo -m setup -a “filter= *ipv4* ”

Playbook:

- Playbooks in ansible are written in YAML format.
- It is human readable data serialization language and is commonly used for configuration files.
- Playbook is like a file where you write codes consists of variables, tasks, handlers, files, templates and roles.
- Each playbook is composed of one or more ‘modules’ in a list. Module is a collections of configuration files.
- Playbooks are divided into many sectors like
  a. Target section: defines the host against which playbooks task has to be executed. b. Variable: define variables
  c. Task section: list of modules that we need to run in an order.

YAML (Yet Another Markup Language):

- For ansible nearly every YAML files starts with a list.
- Each item in the list is a list of key-value pairs commonly called as a directory.

- All YAML files have to begins with "---" and ends with "…".
- All members of a list lines must begin with same indentation level starting with "-".

For e.g:

--- # a list of fruits Fruits:

- Mango
- Strawberry
- Banana
- Grapes
- Apple

…

A directory is represented in a sample key : value form

--- # details of customer Customer:

Name: Ashok

Job: ASE

Skills: devops

Exp: 12 years

Extension for playbook files is .yml.

Note: there should be space between : and value.

Go to ansible server.

Now create one playbook.

vi target.yml

--- # target playbook

-hosts: demo

-user: ansible

-become: yes

-connection: ssh

-gather_facts: yes

Esc- :wq!

$ ansible-playbook target.yml

Variables:

- Ansible uses variables which are defined previously to enable more flexibility in playbooks and roles. They can be used to loop through a set of given values, access various

information like the host name of a system and replace certain strings in templates with specific values.

- Put variable section above tasks so that we define it first and use it later.

Now go to ansible server and create one playbook.

$ vi vars.yml

--- # my variable playbook

-host: demo

- user: ansible

-become: yes

-connection: ssh

Vars:

Pkgname: httpd Tasks:

-name: install httpd server

-action: yum name= "{{packagename}}" state=install Esc

- :wq!

Now execute playbook $ ansible-playbook vars.yml

Handlers Section:

A handler is exactly the same as a task, but it will run when called by another task.

Or

Handlers are just like regular tasks in an ansible playbook, but are only run if the task contains a 'notify' directive and also indicates that it changed something.

DRY-RUN:

Check whether the playbook is formatted correctly or not.

Anible-playbook handlers.yml –check

Go to ansible server

$ vi handler.yml

---# handlers playbook

-hosts: demo

-user: ansible

-become: yes -

connection: ssh

Tasks:

Name: install httpd server

Action: yum name=httpd state=installed

Notify: restart HTTPD

Handlers:

Name: restart HTTPD

Action: service name=httpd state=restarted

Esc- :wq!

Now execute this playbook

$ ansible-playbook handlers.yml

Loops:

Sometimes you want to repeat a task multiple time. In computer programming this is called as loops. Common ansible loops include changing ownership on several files and/or directories with the file module, creating multiple users with the user module and repeating a polling step until certain result is reached.

Now go to ansible server.

$ vi loops.yml

---# my loops playbook

-host: demo

-user: ansible

-become: yes -

connection: ssh

Tasks:

-name: add a list of users

-user: name= "{{item}}" state=present With

items:

-Ashok

-Anupam

-Kritika

-Sunny

-Summy

Esc- :wq!

$ansible-playbook loops.yml

To verify go inside node1

$cat /etc/passwd

Conditions:

Whenever we have different different scenarios we put conditions to the scenario.

We put conditions in ansible by “when” statement.

---# condition palybook

-hosts: demo

-user: ansible

-become: yes

-connection: ssh Tasks:

-name: install apache on Debian

Command: apt-get -y install apache2

When: ansible_os_family == “Debian”

-name: install apache for redhat

Command: yum -y install httpd

When: ansible_os_family ==

“Redhat”

Vault:

Ansible allows keeping sensitive data such as passwords or key in encrypted files, rather that a plaintext in your playbooks.

Creating a new encrypted playbook:

$ ansible-vault create vault.yml

Edit the encrypted playbook:

$ ansible-vault edit vault.yml

To change the password:

$ ansible-vault rekey vault.yml

To encrypt on existing playbook:

$ ansible-vault encrypt target.yml

To decrypt an encrypted playbook:

$ ansible-vault decrypt target.yml

Roles:

- We can use two techniques for reusing a set of tasks : includes and role.
- Roles are good for organizing tasks and encapsulating data needed to accomplish those tasks.
- We can organize playbook into a directory structure called roles.
- Adding more and more functionality to the playbooks will make it difficult to maintain in a single file.

Ansible Roles:

a. Default: it stores the data about role/ application. Default variables e.g: if you want to run to port 80 or 8080 then variables need to define in this path.
b. Files: it contains files need to be transferred to the remote VM (static files).
c. Handles: they are triggers or task. We can segregate all the handlers required in playbook.
d. Meta: this directory contains files that establish roles dependencies. E.g: author name, supported platform, dependencies if any.
e. Templates:
f. Tasks: it contains all the tasks that is normally in the playbook. E.g: installing packages and copies files etc.
g. Vars: variables for the role can be specified in this directory used in your configuration files. Both vars and default stores variables.

$ mkdir -p playbooks/roles/webserver/tasks

$ tree -p playbook/roles/webserever/handler

$cd playbook/

$ tree

$ touch roles/webserver/tasks/main.yml

$ touch master.yml

$ vi roles/webserver/tasks/main.yml

Inside main.yml

-name: install apache -yum:

pkg=httpd state=latest Esc-

:wq!

$vi master.yml

-host: all

-user: ansible -

become: yes -

connection: ssh

Roles:

-webserver

$ansible-playbook master.yml