**ANSIBLE**

**Ansible is an open source IT Configuration Management, Deployment & Orchestration tool.**

1. To automate the deployment process.

2. It uses **PUSH method** ---> Master/server contact Nodes (Server ---> Nodes)

3. Agent Less

3. Credentials are required for login.

4. Playbooks, Playbooks

5. YAML File script

6. **Python has to be installed on Nodes for working with nodes**.

7. Python on node and server should be of same version

8. **By default Python gets installed when we install Ansible on server**.

9. **Ansible Control Server must be a Linux machine.**

Ansible Terms:

**Controller Machine**: The machine where Ansible is installed, responsible for running the provisioning on the servers you are managing.

**/usr/bin/ansible** or **/usr/bin/ansible-playbook**

Inventory: An initialization file that contains information about the list of managed nodes. An inventory file is also sometimes called a “hostfile”.

Static and Dynamic Inventories in Ansible

In Ansible, there are two types of inventory files: Static and Dynamic.

**Static Inventory:**

Static inventory is default inventory and is defined in the **/etc/ansible/ansible.cfg** file. Default file can be changed inside the ansible.cfg file. If you want

to use the custom file as inventory input can specify it using ” -i /path/to/file ” with Ansible command line.

Ex: ansible webserver -m setup

**Dynamic Inventory:**

Keeping your inventory always up-to-date become a little bit problematic. In such case Dynamic inventory comes into picture, generally are scripts (Python/Shell) for dynamic environments (for example cloud environments) With Ansible, as aforementioned, can use “-i” to specify the custom inventory file.

Reduces human error, as information is collected by scripts.

Very less manual efforts for managing the inventories.

Ex: **ansible -i webserver -m setup**

**Playbook:** The entry point for Ansible provisioning, where the automation is defined through tasks using YAML format. Each playbook is an aggregation of one or more plays in it. Playbooks are structured using Plays. There can be more than one play inside a playbook.

A YAML starts with --- (3 hyphens)

YAML Tags:

**name**: name of the Ansible playbook

**hosts**: lists of hosts or host group against which we want to run the task

**vars**: define the variables which you can use in your playbook.

**task**: A block that defines a single procedure to be executed, e.g. Install a package. You can execute a single task once with an ad-hoc command.

Each task internally links to a piece of code called a **module**.

**Creating Reusable Playbooks**

In Ansible, there are three ways to do this: includes, imports, and roles.

Includes and imports: (added in Ansible version 2.4) allow users to break up large playbooks into smaller files, which can be used across multiple parent playbooks or even multiple times within the same Playbook.

You can then use import\_tasks or include\_tasks to execute the tasks in a file in the main task list.

Roles: allow more than just tasks to be packaged together and can include variables, handlers, or even modules and other plugins. Unlike includes and imports, roles can also be uploaded and shared via Ansible Galaxy

**Ad-Hoc command:** The Ad-Hoc command is the one-liner ansible command that performs one task on the target host.

Format--> ansible <host group> <-m module> <Arg to the Module>

(OR)

$ ansible [pattern] -m [module] -a "[module options]"

Ex: ansible webservers -m yum -a "name=acme state=present"

ansible webservers -m service -a "name=httpd state=started"

**Module**

**Module**: Ansible modules are reusable, standalone scripts that can be used by the Ansible API, or by the ansible or ansible-playbook programs.

Few Modules

Module 1: **Package management**: these modules can install, upgrade, downgrade, remove, and list packages.

Ex: APT, YUM, package, apk

Module 2: **Service**: The service module enables you to start, stop, and reload installed packages.

Ex: state: started

service:

name: foo

pattern: /usr/bin/foo

state: started

Module 3: **Copy**: The copy module copies a file from the local or remote machine to a location on the remote machine.

Ex: copy:

src: /mine/ntp.conf

dest: /etc/ntp.conf

Module 4: **Debug**: The debug module prints statements during execution and can be useful for debugging variables or expressions without having to halt the playbook.

debug:

var: hostvars[inventory\_hostname]

verbosity: 4

Module 5: **File**: The file module manages the file and its properties. It sets attributes of files, symlinks, or directories.

It also removes files, symlinks, or directories.

Ex: archive,file,find

file: path: /etc/some\_directory

state: directory

mode: '0755'

Module 6: **Git**: The git module manages git checkouts of repositories to deploy files or software.

Module 7: **Command**: One of the most basic but useful modules, the command module takes the command name followed by a list of space-delimited arguments.

Ex: command: cat /etc/motd

Module 8: **Cloud**: Ex: s3\_bucket, iam\_role, ec2

Module 9: **Windows**: Ex: win\_command, win\_copy,

**Role**: A pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of a provisioning.

Each directory must contain a main.yml file, By default, in ***/etc/ansible/roles***. The relevant content of role file:

**tasks** - contains the main list of tasks to be executed by the role.

**handlers** - contains handlers, which may be used by this role or even anywhere outside this role.

**defaults** - default variables for the role (see Using Variables for more information).

**vars** - other variables for the role (see Using Variables for more information).

**files** - contains files which can be deployed via this role.

**templates** - contains templates which can be deployed via this role.

**meta** - defines some meta data for this role. See below for more details.

**Play:** A provisioning executed from start to finish is called a play. In simple words, execution of a playbook is called a play.

**Facts:** Global variables containing information about the system, like network interfaces or operating system.

Ansible Facts:The task of collecting this remote system information is called as "Gathering Facts" by ansible and the details collected are generally known as facts or variables.

Handlers: Used to trigger service status changes, like restarting or stopping a service.

**Template:** A template in Ansible is a file which contains all your configuration parameters, but the dynamic values are given as variables.

During the playbook execution, the variables will be replaced with the relevant values.

The template files will usually have the .j2 extension, which denotes the Jinja2 templating engine used.

The variables in a template file will be denoted by the double curly braces, ‘{{ variables }}’.

**Ansible has three main files that you need to consider:**

**Host/inventory file**: Contains the entry of the nodes that need to be managed

**Ansible.cfg file**: Located by default at ***/etc/ansible/ansible.cfg***, it has the necessary privilege escalation options and the location of the inventory file

**Main file**: A playbook that has modules that perform various tasks on a host listed in an inventory or host file

**Ansible Galaxy**

Ansible Galaxy is a free site for finding, downloading, rating, and reviewing all kinds of community developed Ansible roles and can be a great way to get a jumpstart on your automation projects.

The Galaxy client allows you to download roles from Ansible Galaxy, and also provides an excellent default framework for creating your own roles.

Ansible will look in several places for the config file:

ansible.cfg in the current directory where you ran ansible-playbook ***/etc/ansible/ansible.cfg***

**Ansible Vault:**

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles.

$ ansible-vault create foo.yml

$ ansible-vault decrypt foo.yml

$ ansible-vault view foo.yml

$ ansible-playbook --ask-vault-pass site.yml

**Ansible Tower:**

Ansible Tower is a web-based solution that makes Ansible even more easy to use for IT teams of all kinds. It’s designed to be the hub for all of your automation tasks.

Tower allows you to control access to who can access what, even allowing sharing of SSH credentials. Inventory can be graphically managed or synced with a wide variety of cloud sources

**Ansible Master And Node Configurations:**

Enable Authentication applicable only for using AWS VM instance\*\*\*

$ vi /etc/ssh/sshd\_config

Check for PasswordAuthentication whether yes or no.

**PasswordAuthentication - yes**

**How to run a playbook as root user?**

become\_user: root

become: true

ansible\_become\_user

$ansible-playbook main.yaml -i hosts --user=git -k --become-user=root --ask-become-pass --become-method=su

**How to get detailed logs and output while executing playbook command**

$ ansible-playbook -v playbook.yaml

If you pass the -vv flag to ansible-playbook on the command line, you'll see the stdout and stderr -v =verbose

\* for more verbosity add a v.

**What is Ansible.cfg file**

Certain settings in Ansible are adjustable via a configuration file (ansible.cfg). he latest ansible.cfg file should be present in /etc/ansible,

**Which module is used to Copy a file from node to master**

The COPY module copies a file from the local or remote machine to a location on the remote machine.

Use the fetch module to copy files from remote locations to the local box.

Ansible Fetch is to pull the files from the remote server to the control machine.

Ansible Synchronize is used to copy the files between remote servers (or) target hosts.