**ANSIBLE**

**CONFIGURATION MANAGEMENT:**

Configuration management is a set of processes and tools used to systematically manage changes to software, hardware, or any other system throughout its lifecycle. It involves tracking and controlling the configuration of items within a system, ensuring consistency and integrity, and providing the ability to trace and audit changes.

It is a method through which we automate admin tasks. Configuration management tool turns your code into infrastructure. So your code would be testable, repeatable and version able.

Infrastructure refers to the composite of: Software, Network, Storage and Process.

**ANSIBLE:**

1. Ansible is one among the DevOps configuration management tools which is famous for its simplicity.
2. It is an open source software developed by Michael DeHaan and its ownership is on RedHat
3. Ansible is an open source IT Configuration Management, Deployment & Orchestration tool.
4. This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.
5. Ansible is an automation tool that provides a way to define infrastructure as code.
6. Infrastructure as code (IaC) simply means that managing infrastructure by writing code rather than using manual processes.
7. The best part is that you don’t even need to know the commands used to accomplish a particular task.
8. You just need to specify what state you want the system to be in and Ansible will take care of it.
9. The main components of Ansible are playbooks, configuration management and deployment.
10. Ansible uses playbooks to automate deploy, manage, build, test and configure anything
11. Ansible is developed using Python Programming language.

**ANSIBLE FEATURES:**

* Ansible manages machines in an agent-less manner using SSH
* Built on top of Python and hence provides a lot of Python's functionality
* YAML based playbooks
* Uses SSH for secure connections
* Follows push based architecture for sending configuration related notifications

**PUSH BASED VS PULL BASED:**

* Tools like Puppet and Chef are pull based
* Agents on the server periodically checks for the configuration information from central server (Master)
* Ansible is push based
* Central server pushes the configuration information on target servers.

**WHAT ANSIBLE CAN DO?**

1. **Provision and configure infrastructure:** Ansible can provision and configure various types of infrastructure, including servers, network devices, storage systems, and cloud resources. It allows you to define desired infrastructure states in simple YAML files and then automatically deploys and configures the necessary resources.
2. **Automate tasks and workflows:** With Ansible, you can automate a wide range of tasks and workflows, such as software installation, configuration file management, service orchestration, and system updates. It provides a vast collection of pre-built modules that can be used to perform specific actions on remote systems.
3. **Enforce desired system configurations:** Ansible ensures that systems are consistently configured according to predefined specifications. It allows you to define configuration files, templates, and policies, and then applies them across multiple systems, ensuring consistency and reducing configuration drift.
4. **Manage and deploy applications:** Ansible can handle application deployment and management tasks. It supports various application deployment strategies and can integrate with container orchestration tools like Docker and Kubernetes, enabling streamlined application deployment across different environments.
5. **Simplify multi-node orchestration:** Ansible is designed for managing large-scale infrastructures with multiple nodes. It can execute tasks on multiple systems simultaneously, facilitating efficient orchestration and reducing the time required for performing tasks across multiple machines.

**HOW ANSIBLE WORKS?**

* Ansible works by connecting to your nodes and pushing out a small program called Ansible modules to them.
* Then Ansible executed these modules and removed them after finished. The library of modules can reside on any machine, and there are no daemons, servers, or databases required.
* The Management Node is the controlling node that controls the entire execution of the playbook.
* The inventory file provides the list of hosts where the Ansible modules need to be run.
* The Management Node makes an SSH connection and executes the small modules on the hosts machine and install the software.
* It connects to the host machine executes the instructions, and if it is successfully installed, then remove that code in which one was copied on the host machine.

**Ansible basically consists of three components**

Ansible requires the following components in order to automate Network Infrastructure.

1) Controlling Node

2) Managed Nodes

3) Ansible Playbook

**Controlling Nodes** are usually Linux Servers that are used to access the switches/routers and other Network Devices. These Network Devices are referred to as the Managed Nodes.

**Managed Nodes: (Host Machines)**

Managed Nodes are stored in the hosts file for Ansible automation.

**Ansible Playbook:**

Ansible Playbooks are expressed in YAML format and serve as the repository for the various tasks that will be executed on the Managed Nodes (hosts).

Playbooks are a collection of tasks that will be run on one or more hosts.

**Host Inventory file:**

* Ansible's inventory hosts file is used to list and group your servers.
* Its default locaton is /etc/ansible/hosts
* ***Note: In inventory file we can mention IP address or Hostnames also.***

**INVENTORY FILE IMPORTANT POINTS:**

1. Comments begins with '#' character
2. Blank lines are ignore.
3. Groups of hosts are delimited by '[header]' elements
4. You can enter hostnames or IP-addresses
5. A hostname/IP can be a member of multiple groups
6. Ungrouped hosts are specifying before any group headers like below

Ansible inventory hosts file is used to list and group your servers. Its default location is **/etc/ansible/hosts**

**SAMPLE INVENTORY FILE:**

#Blank lines are ignore

#Ungrouped hosts are specifiying before any group headers like below

192.168.122.1

192.168.122.2

192.168.122.3

[webservers]

192.168.122.1

#192.168.122.2

192.168.122.3

[dbserver]

192.168.122.1

192.168.122.2

Raghuit-db1.com

Raghuit-db2.com

**ANSIBLE SETUP:**

Create 3 Amazon Linux Instances of type t2.micro (Free tier eligible)

1 - Control Node

2 - Managed Nodes

**EXECUTE BELOW COMMANDS IN ALL NODES:**

**# Create one new user**

$ sudo useradd ansible

$ sudo passwd Ansible

**# Provide sudoer permissions (like root user)**

$ sudo visudo

ansible ALL=(ALL) NOPASSWD: ALL

**# Activate Password Authentication YES**

$ vi /etc/ssh/sshd\_config

**PasswordAuthentication yes (Might be default value NO, change to YES)**

**# Restart sshd service**

$ sudo systemctl restart sshd

**# Switch to Ansible user**

$ sudo su - ansible

**\*\* EXECUTE BELOW COMMANDS ONLY AT MASTER NODE \*\***

**# Generate a new KEY-PAIR using SSH**

$ ssh-keygen

**# Copy Public Key (Authorized Keys) into Managed Nodes**

$ ssh-copy-id ansible@<ManagedNode-Private-IP>

Ex : $ ssh-copy-id ansible@172.31.8.95

**make sure port 22 is open for SSH (Anywhere)**

**# Install Python and PIP[Package Installers for Python] software for Ansible Runtime**

$ sudo yum install python3 -y

$ python3 --version

$ sudo yum install python3-pip –y

**# Install Ansible**

$ pip3 install ansible --user

$ ansible --version

**# create ansible root directory**

$ sudo mkdir /etc/ansible

**# create ansible configuration and inventory files**

$ sudo vi /etc/ansible/ansible.cfg

**Go to:**

<https://raw.githubusercontent.com/ansible/ansible/devel/examples/ansible.cfg>

**(or)**

[**https://github.com/ansible/ansible/tree/stable-2.9/examples**](https://github.com/ansible/ansible/tree/stable-2.9/examples)

**Uncomment lines**

inventory = /etc/ansible/hosts

sudo\_user = ansible

:wq (save and quit)

$ sudo vi /etc/ansible/hosts

**https://raw.githubusercontent.com/ansible/ansible/devel/examples/hosts**

[webservers]

65.0.205.235

[dbservers]

13.234.239.70

:wq (save and quit)

4) Test Ansible and after installation successful

$ ansible all --list-hosts

$ ansible webservers --list-hosts

$ ansible dbservers --list-hosts

$ ansible webservers[0] --list-hosts

$ ansible webservers[1] --list-hosts

**ANSIBLE AD-HOC COMMANDS**

Switch to ansible user and run ansible ad-hoc commands

$ sudo su ansible

**To run any ansible command we will follow below syntax:**

# ansible [ all / groupName / HostName / IP ] -m <<Module Name>> -a <<args>>

Note: Here -m is the module name and -a is the arguments to module.

**Example:**

**# ping all managed nodes listed in host inventory file**

$ ansible all -m ping

**#ping only webservers listed in host inventory file**

$ ansible webservers -m ping

**#ping only dbservers listed in host inventory file**

$ ansible dbservers -m ping

**# it will display date from all host machines.**

$ ansible all -m shell -a date

**# It will display uptime from all host machines.**

$ ansible all -m shell -a uptime

There are two default groups, **all and ungrouped**. all contains every host. Ungrouped contains all hosts that don’t have another group

**# It will display the all the modules available in Ansible.**

$ ansible-doc -l

**# To display particular module information**

$ ansible-doc <moduleName>

**# To display shell module information**

$ ansible-doc shell

**# it will display details of copy module**

$ ansible-doc -l | grep "copy"

**#It will display more information about yum module**

$ ansible-doc yum

**PING MODULE:**

**# It will ping all the servers which you have mentioned in inventory file (/etc/ansible/hosts)**

$ ansible all -m ping

# It will display the output in single line.

$ ansible all -m ping -o

**SHELL MODULE:**

**# To execute all shell commands**

# Date of all machines

$ ansible all -m shell -a 'date'

**# Release of all the machines**

$ ansible all -m shell -a 'cat /etc/\*release'

**# Check the service status on all the machines**

$ ansible all -b -m shell -a 'service sshd status'

**# Here it will check the disk space use for all the nodes which are from db servers group**

$ ansible dbservers -b -m shell -a "df -h"

**# Here it will check the disk space use for all the nodes which are from webservers group**

$ ansible webservers -b -m shell -a "free -m"

**# Here it will display date from from webservers group**

$ ansible webservers -b -m shell -a "date"

**YUM MODULE:**

**# It will install vim package in all node machine which you have mentioned in host inventory file.**

$ ansible all -b -m yum -a "name=vim"

**# Check git version in all machines**

$ ansible all -m shell -a "git --version"

**# to install git client in all node machines**

$ ansible all -m shell -b -a "yum install git -y"

**# To installl git only in webserver nodes**

$ ansible webservers -m shell -b -a "yum install git -y"

# To install webserver only in particular machine

$ ansible 172.1921.1.0 -m shell -b -a "yum install git -y"

$ ansible all -m yum -b -a "name=git state=present"

$ ansible all -m yum -b -a "name=git state=latest"

$ ansible all -m yum -b -a "name=git state=absent"

**present : install**

**latest : update to latest**

**absent : un-install**

**# to install any software in ubuntu server then we should use apt package manager**

$ ansible all -m apt -a "name="git state="present"

# To install httpd package in all node machines

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

Note: Here state=latest, is not a mandatory, it is by default.

**# To update httpd package in all node machines.**

$ ansible all -b -m yum -a "name=httpd state=latest"

**# To remove httpd package in all node machines.**

$ ansible all -b -m yum -a "name=httpd state=absent"

$ ansible all -m copy -a "src="index.html dest=/var/www/html/index.html"

**# start httpd service**

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

$ ansible all -b -m shell -a "service httpd start"

Note: For privilege escalations we can use -b option (sudoer permissions)

**YAML (Yet Another Mark-up Language) | YAMLAinant mark-up language**

* We can make use of this language to store data and configuration in a human-readable format.
* YAML files will have .yml as an extension
* Official Website: https://yaml.org/

**Sample YML File Data**

Fruit: Apple

Vegetable: Carrot

Liquid: Water

Meet: Chicken

**Array/List**

Fruits:

- Orange

- Apple

- Banana

- Guava

Vegetables:

- Carrot

- Cauliflower

- Tomoto

**Here - dash indicate the element of any array.**

name: Raghu

age: 38

phno: 123456

email: javabyraghu@gmail.com

hobbies:

- cricket

- dance

- singing

**# person data in yml**

person:

id: 101

name: Raju

email: raju@gmail.com

address:

city: Hyd

state: TG

country: India

job:

companyName: IBM

role: Tech Lead

pkg: 25 LPA

hobbies:

- cricket

- chess

- singing

- dance

**# using --- hypens to seperate the data**

---

person:

id: 101

name: Raju

email: raju@gmail.com

address:

city: Hyd

state: TG

country: India

job:

companyName: IBM

role: Tech Lead

pkg: 25 LPA

hobbies:

- cricket

- chess

- singing

- dance

---

movie:

name: Bahubali

hero: Prabhas

heroine: Anushka

villian: Rana

director: SS Rajamouli

budget: 100cr

...

**PLAYBOOKS**

1. Playbook is a single YAML file, containing one or more plays in a list.
2. Plays are ordered sets of tasks to execute against host servers from your inventory file.
3. Play defines a set of activities (tasks) to run on managed nodes.
4. Task is an action to be performed on the managed node

Examples are:

a) Execute a command

b) Run a shell script

c) Install a package

d) Shutdown / Restart the hosts

Note: Playbooks YML / YAML starts with the three hyphens ( --- ) and ends with three dots (…)

**Playbook contains the following sections:**

1) Every playbook starts with 3 hyphens (---)

2) Host section: Defines the target machines on which the playbook should run. This is based on the Ansible host inventory file.

3) Variable section: This is optional and can declare all the variables needed in the playbook. We will look at some examples as well.

4) Tasks section: This section lists out all the tasks that should be executed on the target machine. It specifies the use of Modules. Every task has a name which is a small description of what the task will do and will be listed while the playbook is run.

**Playbook to Ping All Host Nodes**

---

- hosts: all

gather\_facts: no

remote\_user: anisble

tasks:

- name : Ping

ping:

remote\_user: ansible

...

**hosts**: The tasks will be executing in specified group of servers.

**name**: which is the task name that will appear in your terminal when you run the playbook.

**remote\_user**: This parameter was formerly called just user. It was renamed in Ansible 1.4 to make it more distinguishable from the user module (used to create users on remote systems).

Note : Remote users can also be defined per task.

**# Run the playbook Using below command**

$ ansible-playbook <<Playbbok file name>>

**# It will run the playbook.yml playbook in verbose**

$ ansible-playbook playbook.yml -v

$ ansible-playbook playbook.yml -vv

$ ansible-playbook playbook.yml -vvv

**# It will provide help on ansible\_playbook command**

$ ansible-playbook --help

**# It will check the syntax of a playbook**

$ ansible-playbook playbook.yml --syntax-check

**# It will do in dry run.**

$ ansible-playbook playbook.yml --check

**# It will display the which hosts would be effected by a playbook before run**

$ ansible-playbook playbook.yml --list-hosts

**# It execute one-step-at-a-time, confirm each task before running with (N)o/(y)es/(c)ontinue**

$ ansible-playbook playbook.yml --step

**INSTALL HTTPD + COPY index.html + START SERVICE**

* Create index.html file in the location where our playbook is exist.
* Create yml file with below content.

---

- hosts: all

become: true

tasks:

- name: Install Httpd

yum:

name: httpd

state: present

- name: Copy index.html

copy:

src: index.html

dest: /var/www/html/index.html

- name: Start Httpd Server

service:

name: httpd

state: started

...

**Execute the playbook yml using ansible-playbook command**