ANSIBLE

Configuration Management

Configuration and management is an automated method for maintaining computer systems and software in a known, consistent state.

The process of standardizing and administering resource configurations and entire IT infrastructure in an automated way is Configuration Management

There are two types of configuration management tool

1. Pull based configuration management
2. Push based configuration management

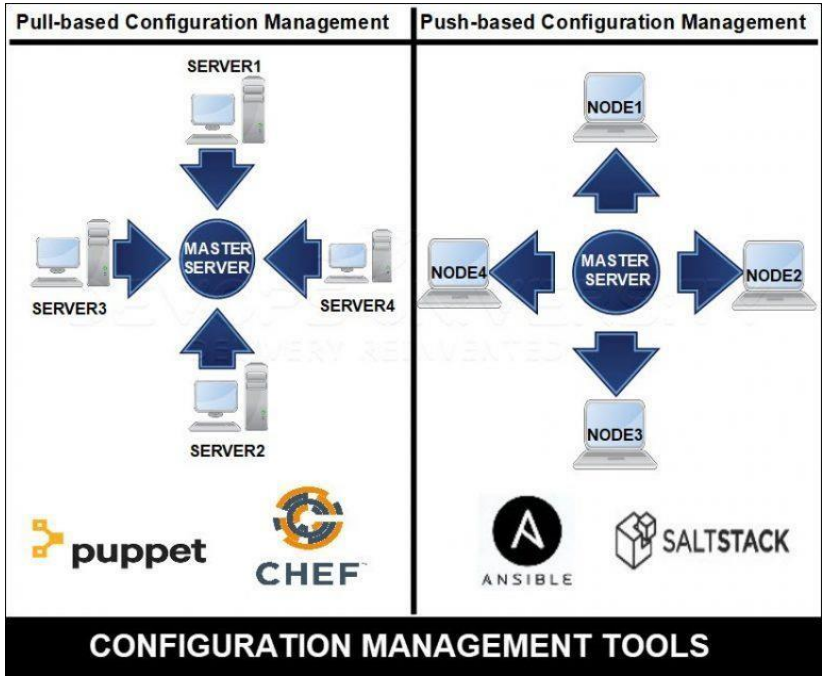
Pull Model

- The server nodes run an agent daemon that periodically checks from the master node when there are any updates to be pulled and applied.

- A daemon needs to be installed on all machines and a setup of the central authority is required.

Push Model

- Here, it is the central server or the master node which takes the responsibility to contact the server nodes to send updates as and when they occur.
- Whenever a change is made to the infrastructure (code), each node is informed of the update and they run the changes



CONFIGURATION MANAGEMENT TOOLS

Terraform

SALTSTACK

puppet

CHEF

ANSIBLE

docker

Use of Configuration Management

- **Automation** of configuration policies and management for prompt remediation.
- **Scalability** with tools capable of on boarding an extensive infrastructure over time.
- **Flexibility** with varying tool capabilities and seamless integration into existing solutions
- **Cloud**compatibility to meet the needs of multiple environments in a hybrid infrastructure
- **Protection** against future risks with a system devoted to configuration policies
- **Visibility** provided by the logging and analysis of transactions related to CM

Difference between Chef, Puppet, Ansible, SaltStack

Ansible	Puppet	Saltstack	Chef
Streamlined provisioning	Orchestration	Automation for CloudOps	Infrastructure automation
Configuration management	Automated provisioning	Automation for ITOps	Cloud automation
App deployment	Role-based access control	Continuous code integration and deployment	Compliance and security management
Automated workflow for Continuous Delivery	Visualization and reporting	DevOps toolchain workflow automation with support for Puppet, Chef, Docker, Jenkins, and Git.	Automated workflow for Continuous Delivery
Security and Compliance policy integration	Configuration automation	Application monitoring and auto-healing	Chef-Server using RabbitMQ, AMQP protocol.
Simplified orchestration	Code and node management	Orchestration	Automation for DevOps workflow

WHAT IS ANSIBLE?

- **Ansible** is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.
- Ansible is easy to deploy because it does not use any agents or custom security infrastructure.
- Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. **YAML** (It's a humanreadable data serialization language & is commonly used for configuration files, but could be used in many

applications where data is being stored)which is very easy for humans to understand, read and write.

History of Ansible

- **Michael DeHaan** developed Ansible, and the Ansible project began in **February 2012**.
- The creator of **Cobbler** and **Func** is also the controller of the **Fedora Unified** network.
- **RedHat** acquired the Ansible tool in 2015.
- Ansible is included as part of the **Fedora distribution** of the Linux.
- Ansible is also available for **RedHat Enterprise Linux, Debian, CentOS, Oracle Linux,** and **Scientific Linux** via Extra Packages for

Enterprise Linux **(EPEL)** and **Ubuntu** as well as for other operating systems.

WHY USE ANSIBLE?

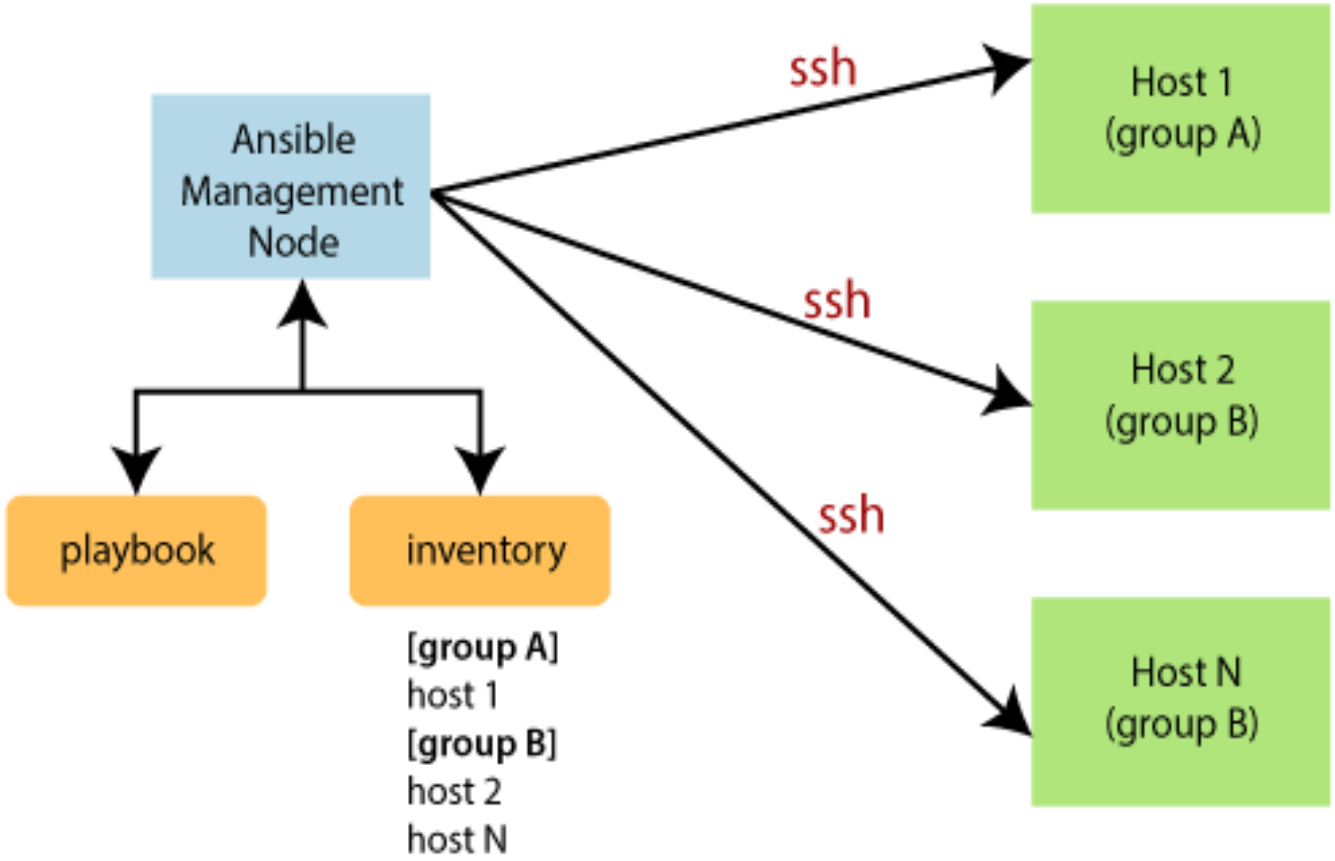
- Ansible is free to use by everyone.
- Ansible is very consistent and lightweight, and no constraints regarding the operating system or underlying hardware are present.
- It is very secure due to its agentless capabilities and open **SSH** security features.
- Ansible does not need any special system administrator skills to install and use it.
- Ansible has a smooth learning curve determined by the comprehensive documentation and easy to learn structure and configuration.

- Its modularity regarding **plugins, inventories, modules,** and **playbooks** make Ansible perfect companion orchestrate large environments.

Ansible workflow

- 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.

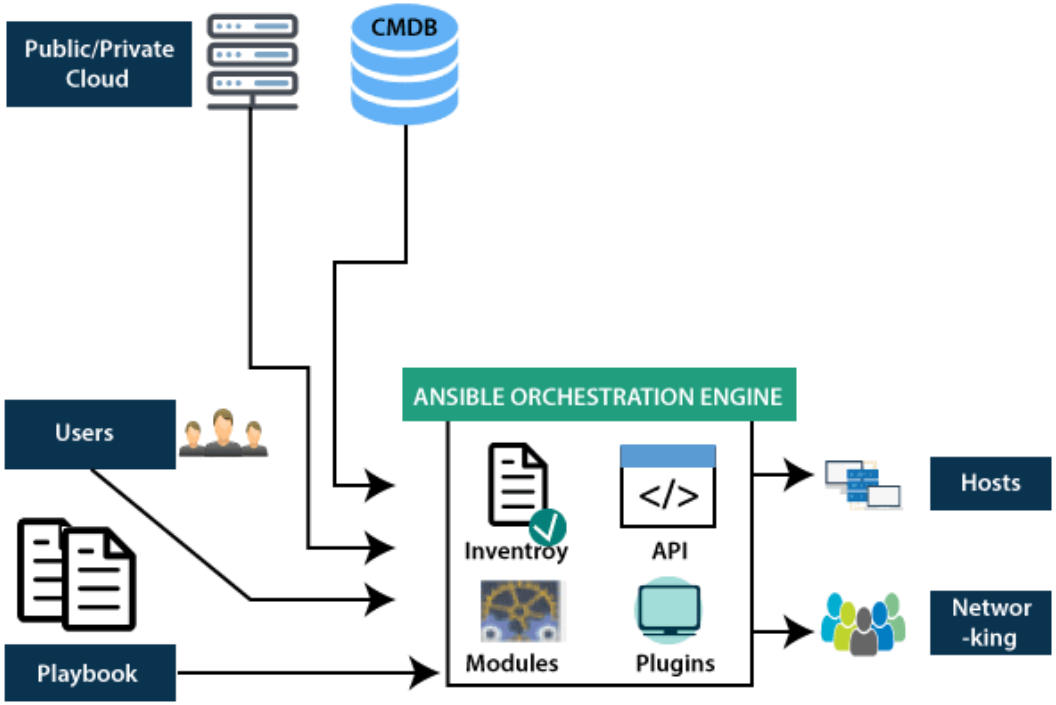
Ansible workflow



- 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 host's machine and install the software.
- Ansible removes the modules once those are installed so expertly.
- 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 Architecture

- The Ansible orchestration engine interacts with a user who is writing the Ansible playbook to execute the Ansible orchestration and interact along with the services of private or public cloud and configuration management database.


Public/Private Cloud
CMDB
Users
ANSIBLE ORCHESTRATION ENGINE
Inventroy
API
Modules
Plugins
Playbook
Hosts
Networ -king

- **Inventory:** Inventory is lists of nodes or hosts having their IP addresses, databases, servers, etc. which are need to be managed.
- **Adhoc:** An Ansible ad hoc command uses the /usr/bin/ansible command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable. So why learn about ad hoc commands first? ad hoc commands demonstrate the simplicity and power of Ansible.

- **Modules:** Ansible connected the nodes and spread out the Ansible modules programs. Ansible executes the modules and removed after finished. These modules can reside on any machine; no database or servers are required here. You can work with the chose text editor or a terminal or version control system to keep track of the changes in the content.
- **Plugins:** Plugins is a piece of code that expends the core functionality of Ansible. There are many useful plugins, and you also can write your own.

- **Playbook:** Playbooks consist of your written code, and they are written in YAML format, which describes the tasks and executes through the Ansible. Also, you can launch the tasks synchronously and asynchronously with playbooks.

Playbook structure

- 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.

- The function of a play is to map a set of instructions defined against a particular host.
- YAML is a strict typed language; so, extra care needs to be taken while writing the YAML files. There are different YAML editors but we will prefer to use a simple editor like notepad++. Just open notepad++ and copy and paste the below yaml and change the language to YAML (Language → YAML).
- A YAML starts with --- (3 hyphens)

Create a Playbook

```
---
  name: install and configure DB
  hosts: testServer become: yes

  vars:
    oracle_db_port_value : 1521

  tasks:
  -name: Install the Oracle DB
    yum: <code to install the DB>

  -name: Ensure the installed service is enabled and running
  service:
    name: <your service name>
```

HOW TO INSTALL AND SETUP ANSIBLE IN EC2 INSTANCE

STEP 1. First create the 3 ec2 linux instance and name one of the instance as master and other two instance as node1 and node2.

STEP 2. Then now connect all the instance to the MOBAXTERM for easy use

STEP 3. Now run the following commands to install the ansible in all the instances (i.e., master and node).

-> sudo su

-> wget https://dl.fedoraproject.org/pub/epel/epel-releaselatest-7.noarch.rpm

-> ls

epel-release-latest-7.noarch.rpm

-> yum install epel-release-latest-7.noarch.rpm -y

-> yum install git python python-level python-pip openssl ansible -y

Check the version of ansible by typing the command ansible –version

STEP 4. After installing the ansible in all three instances, now go to master and edit hosts file and ansible.cfg file. -> vi /etc/ansible/hosts

vi /etc/ansible/ansible.cfg

STEP 5. Now go to sshd_config and edit it in all the instance (i.e., master and nodes)

Vi /etc/ssh/sshd config

STEP 6. Now add user to all the instances (i.e., master and nodes) by entering the commands

STEP 6. Now type visudo and do the changes to that file in all the instances

STEP 7. Now start the sshd servie in all the instance by entering the command -> service sshd restart

After restarting the sshd service , switch from root user to the created username i.e., admin

-> su – admin

After switching enter the command to connect to the different nodes from master -> ssh privateIP_of_node

Then give the password of the user, master will get connected to the respective node

HOW TO CONNECT FROM MASTER TO DIFFERENT NODES WITHOUT ENTERING PASSWORD

STEP 1. First go to master instance and then switch to the user by entering command

-> su – admin

After that enter the command

-> ssh-keygen

Then press enter for three times

Then enter ls –a to check whether hidden .ssh directory has been generated or not, after that change directory to .ssh by entering command

-> cd .ssh

STEP 2. Now enter the command to establish a relationship between master and node, then enter the password

-> ssh-copy-id username@privateIP

STEP 3. Now try to connect to the nodes from master using the ssh.

Commands

adhoc command

$ ansible all -m yum -a “name=httpd state=present”

Here it will install httpd in all host systems.

$ ansible all -m apt -a “name=httpd state=present”

It will install httpd in all host systems.

To Ensure a service is started on all servers:

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

It will start httpd in all hosts that you defined in web group.

To restart the service:

$ ansible web -m service -a “name=httpd state=restarted”

To stop the service:

$ ansible web -m service -a “name=httpd state=stopped”

Ansible ad hoc command to create a file

$ ansible all -m file -a “path=/project/devops/abcd.txt state=touch”

Here it will create an empty file called abcd.txt in all remote servers.

To create a files and directories in ansible we use file module.

MODULE COMMAND

$ ansible <group_name> -b -a yum "pkg=httpd state=latest"