Configuration Management:

This is the process of remotely configuring servers present at different geographical locations from one point of control

Advantages of Configuration Management Tools:

1. Provisioning of Servers
2. Replica Environments
3. Handling Snowflake servers
4. Idempotence

Provisioning of Servers: Performing activities like installing software, starting, stopping, and restarting services, installing patches and updates, copying necessary configuration files etc.
All these activities can be performed from one point of control

Creating Replica Environments: During disaster recovery the entire infra which is compromised can easily be recreated with configuration management tools

Handling Snowflake Servers: After a period, each server present in a data centre starts behaving like a snowflake. Ie it can run on slightly different hardware and software configurations. Configuration Management tools can capture all this information into simple setup files which can be later used for setting up a similar environment.

Idempotent: Configuration Management tools are used to bring the servers to a specific state which is called as a desired state. If the remote servers are already in desired state, configuration management tools will not reconfigure that server.

Popular Configuration Management Tools:

1. Ansible
2. Chef
3. Puppet
4. SaltStack

What is Ansible?

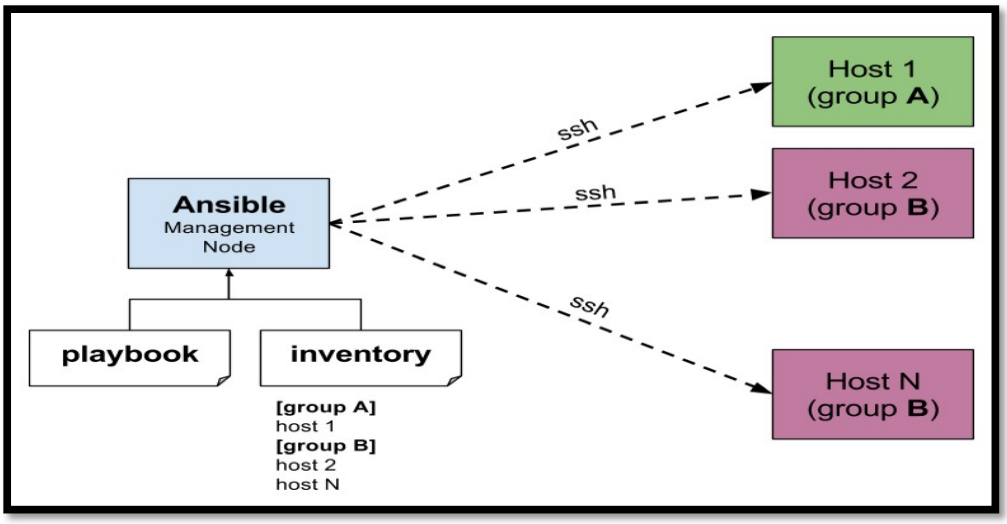
Ansible is an open source, command-line IT automation software application written in Python. It can configure systems, deploy software, and orchestrate advanced workflows to support application deployment, system updates, and more.

Ansible's main strengths are simplicity and ease of use. It also has a strong focus on security and reliability, featuring minimal moving parts. It uses OpenSSH for transport (with other transports and pull modes as alternatives) and uses a human-readable language that is designed for getting started quickly without a lot of training.

Why Ansible?

Ansible delivers simple IT automation that ends repetitive tasks and frees up DevOps teams for more strategic work.

How it works?



Ansible Controller Node performs configuration management on the Target Nodes.
The server where ansible is installed is termed as the Controller Node. The target nodes are termed as Managed Nodes or Managed Hosts.

Ansible is called agentless because it doesn't have any client software installed on the managed hosts. It will perform remote configuration through passwordless SSH. This model is called as the push model. Ansible uses different modules in python for performing remote configurations.

Ansible can configure remote servers in 3 ways

1. Ad hoc Commands
2. Ansible Playbooks
3. Ansible Roles

Ad Hoc Commands: Ansible commands to perform configuration management. Only one module at a time.

Playbooks and Roles: are YAML files which can work on multiple ansible modules

Creating a 3 Node Ansible Cluster: (1 Controller and 2 managed nodes)

1. Create 3 Ubuntu 20.04 instances
2. Update the apt repository on all the servers
3. Change the hostnames of the servers
4. Enable passwordless ssh connectivity between controller node and managed nodes
5. Generate ssh keys on the controller and copy the keys to managed nodes

For Detailed instructions :
https://gist.github.com/nocturnaldevops/a76d8452000ef077aae4093a70adad51

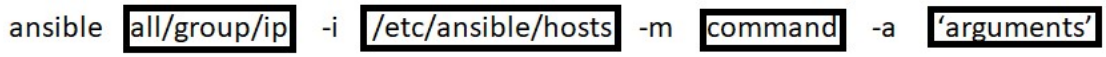
There are different modules in ansible to perform various activities.

1. Command Module – to run basic Linux commands. This module is the default module
2. Shell Module – To execute shell scripts and complex commands
3. Ping – To check the remote managed nodes are reachable or not
4. Copy – for copying files from the controller to managed nodes
5. Fetch – To fetch the files from managed nodes to the controller
6. File – To create files/directories in the managed nodes
7. User – To create users or modify the existing users configurations in the managed nodes
8. Apt - To install/ uninstall/update the packages on ubuntu Debian based machines
9. Yum – To install packages on centos, oracle linux, fedora etc.,
10. Service – To start, stop and restart services on managed nodes
11. Git – To perform version controlling on managed nodes
12. Uri – To check if a remote url is reachable from managed nodes
13. Debug – for displaying output of any module. The output comes in json format
14. Stat – For capturing information about files and folders present in managed nodes
15. Include – Used to call child playbooks in parent playbooks
16. Ec2 – Used for creating ec2 instances on aws
17. Docker_container – used for docker container management
18. Docker_image - to run commands related to docker images
19. Docker_login – To login to hub.docker.com from remote servers

List of ansible modules: https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html

Ad-Hoc Commands Use Cases

Structure of an Ad-Hoc Command



Run an ad hoc command to get a list of all the files from /home/ubuntu on all the managed nodes

- ansible all -i /etc/ansible/hosts -m command -a 'ls /home/ubuntu'

The above command can be simplified to

- ansible all -m command -a 'ls /home/ubuntu' ##DEFAULT INV: /etc/ansible/hosts##

Furthermore, simplified to

- ansible all -a 'ls' ###DEFAULT MODULE: COMMAND###

Get a list of last 5 users present in the managed nodes

- ansible all -a 'cat /etc/passwd | tail -5' # The command fails
- ansible all -m shell -a 'cat /etc/passwd | tail -5' #Shell module to perform complex cmds

****shell module and command modules are not idempotent**

Using Copy module, copy a file from controller to all the managed nodes

- ansible all -m copy -a 'src="sourcefilefromcontroller" dest="managednodesdestn" '
- ansible all -m copy -a 'src=/home/ubuntu/file1.txt dest=/tmp'
- ansible all -m copy -a 'src=/home/ubuntu/dir1 dest=/home/ubuntu/' #entire directory

Copy the file and change the permissions at the destination

- ansible all -m copy -a 'src=/home/ubuntu/file1.txt dest=/tmp owner=root group=root mode=700' -b

Privilege Escalation:
Ansible uses existing privilege escalation systems to execute tasks with root privileges or with another user's permissions. Because this feature allows you to 'become' another user, different from the user that logged into the machine (remote user), we call it become.

Push the content into a file at the destination
ansible all -m copy -a 'content="Hello world" dest=/tmp/file1.txt'

Copy the file on managed node1 to another location on the same managed node through ansible controller
ansible <mnip> -m copy -a 'src=/home/ubuntu/file1.txt dest=/tmp/file1.txt remote_src=yes'

Copy the files from managed node to the controller
ansible <mnip> -m fetch -a 'src=/home/ubuntu/file1.txt dest=/tmp/file1.txt'

Create a user account in all the managed nodes
ansible all -m user -a 'name=username, password=password, uid=uid, comment="comment", home=/home/username, shell="shelltype' -b

Install Elinks package in all the managed nodes
anisble all -m apt -a 'name=elinks state=present update_cache=yes' -b

Uninstall java on all managed nodes
ansible all -m user -a 'name=default-jdk state=absent' -b

Start Jenkins as a service in all managed nodes
ansible all -m service -a 'name=jenkins state=started' -b

Stop tomcat service in all the managed nodes
ansible all -m service -a 'name=tomcat9 state=stopped' -b

Disadvantages of Ad-Hoc commands.

The disadvantages of Ad-hoc command is that we can execute only one module at a time. Given a Configuration which has lot of steps to perform becomes tedious with adhoc commands. Adhoc commands are not meant for performing complex configuration management activities.
The solution to go with for complex configuration management activities is to go with **Playbooks.**

What is a Playbook?
A playbook is a combination of multiple plays. Plays contain tasks that has to be performed. These plays are designed to work on single host/ group of hosts or all the hosts present in the Default inventory file. Playbooks are created using YAML scripting. The main advantages of Playbooks over adhoc commands is Reusability.

Structure of a Playbook:

```
---

- name: Description of the playbook
  hosts: all/group/ip
  tasks:
    - name: Install Tree
      apt:
        name: tree
        state: present
        update_cache: yes
…
```

Ansible Playbooks for multiple configurations updated to
https://github.com/nocturnaldevops/ansible

Grouping in Ansible:
The servers added in the inventory can be grouped together based on their functionality with the concept of grouping in Ansible.
To group servers, target the set of servers and add them under a group name
Ex:
[devgroup]
172.17.0.2
172.17.0.5
172.17.0.10
[qagroup]
172.17.0.9
172.17.1.10
172.17.0.7

The servers under the group name devgroup belongs to the devgroup and the servers under qagroup are QAServers.