ANSIBLE:

- It is a Configuration Management Tool.
- Configuration: Ram, Storage, OS, Software and IP address of device.
- Management: Update, Delete, Add.
- Ansible is simple open-source IT engine which automates application deployment.
- Orchestration, Security and compliance.
- Uses YAML Scripting language which works on KEY-VALUE PAIR
- Ansible GUI is called as Ansible Tower. It was just Drag and Drop.
- Used PYTHON for Back end.

HISTORY:

- Michael Dehhan developed Ansible and the Ansible project began in Feb 2012.
- Ansible was taken over by Red-hat.
- Ansible is Available for RHEL, Debian, CentOS, Oracle Linux.
- Can use this tool whether your servers are in On-prem or in the Cloud.
- It turns your code into Infrastructure i.e. Your computing environment has some of the same attributes as your application.

WHY ANSIBLE?:

While managing the multiple servers its hard to keep their configuration identical. If you have multiple servers which needs to configure the same setup in all. while doing the one to one server their might be a chances to miss some configuration steps in some servers.

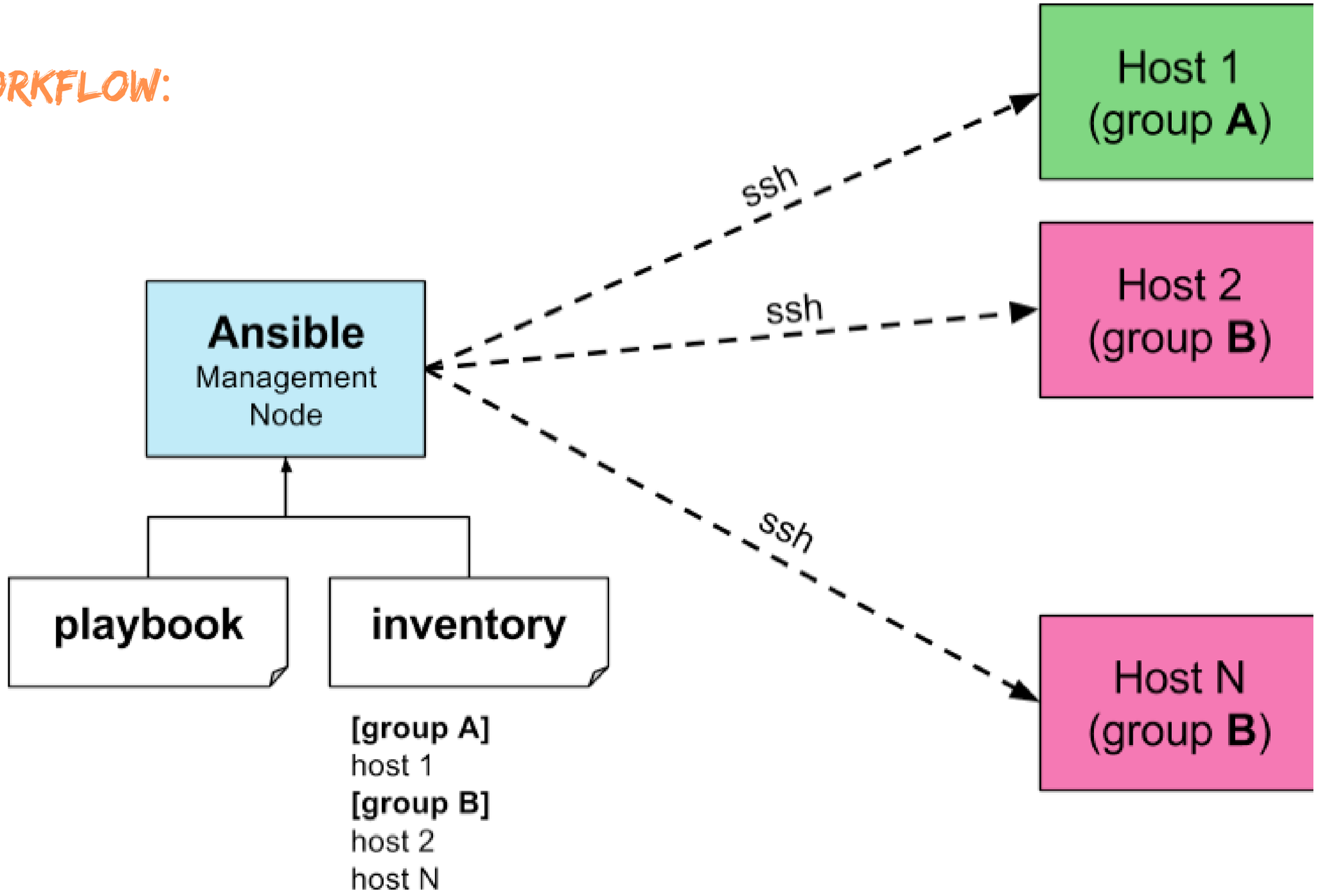
Thats why automation tools come into play! The automation tools like Ansible, Chef, Puppet and SaltStack all are based on a same principle.

DESCRIBE THE DESIRED STATE OF THE SYSTEM

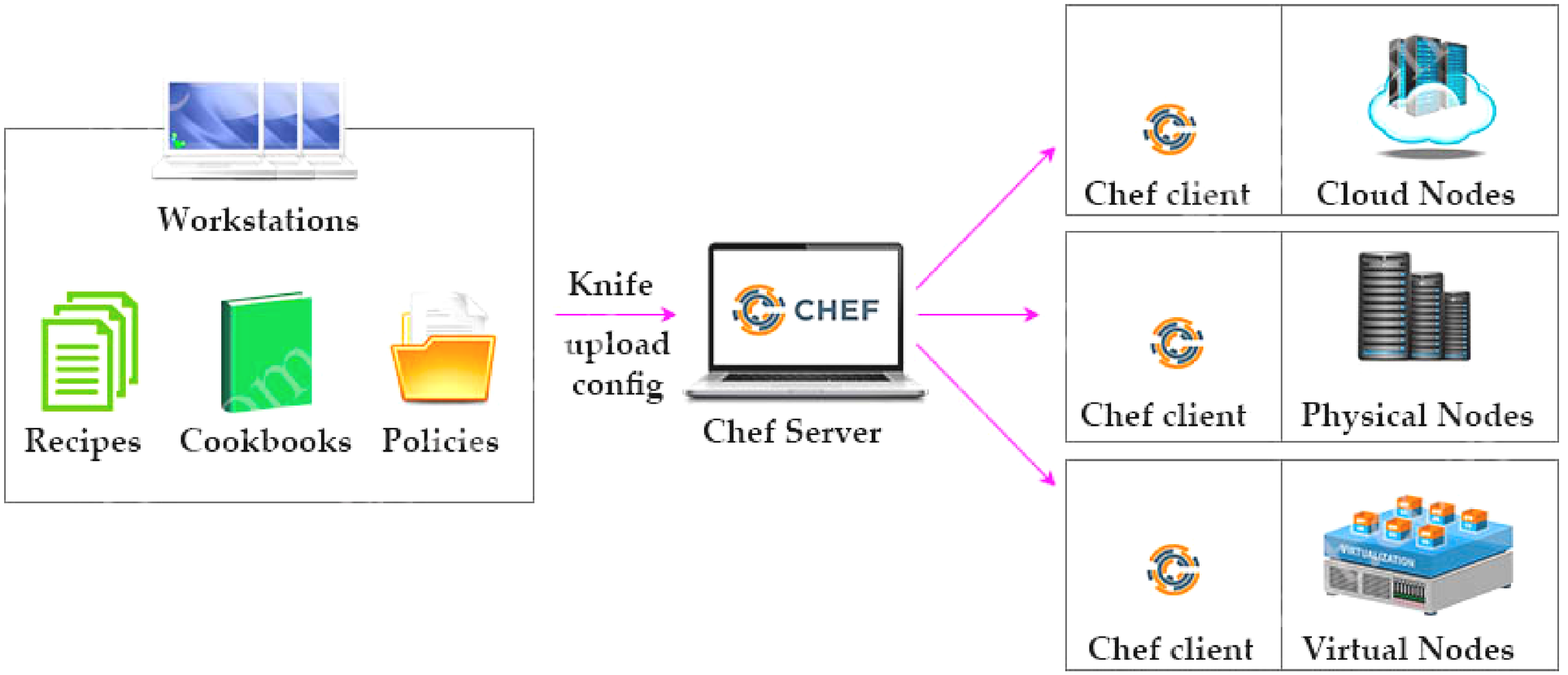
Ansible uses plain SSH. so nothing needs to install on client machines. but other automation tools like Chef/Puppet needs to install agent on client machine when we need to perform a task.

Ansible’s is light weight, relative easy to use and speed of deployment compared to other tools. Ansible handle either via standard SSH commands, or the Paramiko module which provides a Python interface to SSH2.

ANSIBLE WORKFLOW:



CHEF WORKFLOW:



SETUP:

ANSIBLE SERVER:

```
sudo -i
sudo amazon-linux-extras install ansible2 -y
yum install python python-pip python-level openssl -y
vi /etc/ansible/hosts
vi /etc/ansible/ansible.cfg
```

ALL NODES

```
useradd ansible
passwd ansible
visudo
vim /etc/ssh/sshd_config
sudo systemctl restart sshd
sudo systemctl status sshd
su - ansible
hostname -i
```

ANSIBLE SERVER:

ssh-copy-id ansible@localhost

yes & password

exit

ssh-copy-id ansible@privateip

yes & password

exit

HOST PATTREN:

‘all’ patterns refer to all the machines in an inventory.

ansible all–list-hosts ansible <groupname[remo]> --list-hosts

ansible <groupname> [remo][0] --list-hosts

groupname [0] – picks first machine of group

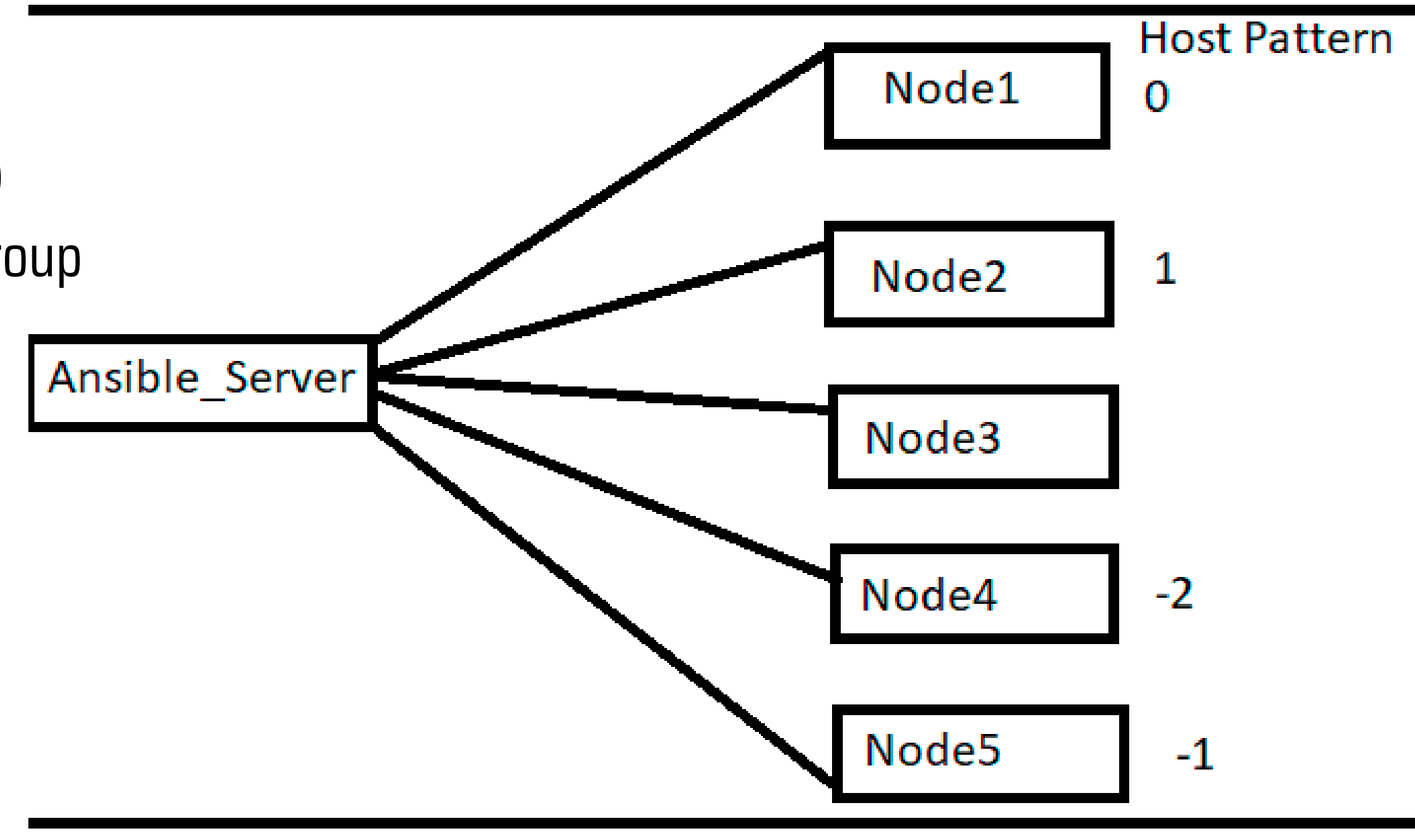
groupname [1] – picks second machine of group

groupname [-1] – picks last machine of group

groupname [1:4] – picks 2,3,4,5 machines in group

groupname [2:5] – picks 3 to 6 machines in the group

ansible all -m ping -v



If we want to push the code from Ansible server to nodes it can be done in 3 ways.

Ad-hoc Commands (Simple Linux) Ad-hoc means temporary & it will over-ride commands.

Modules – A Single Command.

Playbooks – More than one module is called Playbook.

Both module and Playbook is in YAML.

ADHOC COMMANDS:

These commands can be run individually to perform Quick functions.
Not used for configuration management and deployment, bcz the cmds are one time usage.
The ansible ad-hoc cmds uses /usr/bin/ansible/ command line tool to automate single task.
Go to ansible server and switch to ansible server

ansible remo -a “ls” [remo: Group name, -a: argument, ls: command]
ansible remo [0] -a “touch file1”
ansible all -a “touch file2”
ansible remo -a “sudo yum install httpd -y”
ansible remo -ba “yum install httpd -y” (b: become you will become sudo user)
ansible remo -ba “yum remove httpd -y”

AD-HOC COMMANDS:

Write an ansible ad hoc command to check uptime : **ansible all -m command -a "uptime"**

Ansible ad hoc command to check the free Ram memory : **ansible all -a "free -m"**

To check the disk space on all hosts in an inventory file : **ansible all -m shell -a 'df -h'**

To list all the running processes on a specific host : **ansible specific_host -m command -a 'ps aux'**

Check docker is install or not in all the servers : **ansible all -b -m shell -a "docker --verison"**

To check the status of a specific service on all hosts : **ansible all -m service -a 'name=httpd state=started'**

MODULES:

Ansible ships with number of modules (called library modules) that can be executed directly to remote hosts or playbooks.

Your library of modules can reside on any machine, and there are no servers, daemons or database required.

The default location for the inventory file is /etc/ansible/hosts

Go to ansible server and switch to ansible server

ansible remo -b -m yum -a “pkg=httpd state=present” (install: present)

ansible remo -b -m yum -a “pkg=httpd state=latest” (update: latest)

ansible remo -b -m yum -a “pkg=httpd state=absent” (uninstall: absent)

ansible remo -b -m service -a “name=httpd state=started” (start: started)

ansible remo -b -m user -a “name=raj” (to check go to that servers and sudo cat /etc/passwd).

ansible remo -b -m copy -a “src=filename dest=/tmp” (to check go to that server and give ls /tmp).

ansible all -b -m command -a "getent passwd username" (To check user is created or not)

PLAYBOOKS:

Playbooks in ansible are written in YAML language.
It is human readable & serialization language commonly used for configuration files.
You can write codes consists of vars, tasks, handlers, files, templates and roles.
Each playbook is composed of one or more modules in a list.
Playbooks are mainly divided into sections like

TARGET SECTION: Defines host against which playbooks task has to be executed.
VARIABLE SECTION: Defines variables.
TASK SECTION: action you are perfoming.

YAML:

For ansible, Docker, K8S every YAML file starts with a list

Each item in the list is a list ok key-value pairs commonly called Dictionary.

All YAML files have to begin with ”---” and end with “...”

A dictionary is required in a simple key: value form (note: space before value is must)

For example:

```
--- # Customer details
Customer:
    Name: abc
    Age: 21 y
    Salary: 30,000
    Exp: 1 year
```

Extension for playbook file is .yml

BASIC POINTS:

Go to ansible server and login as ansible and create one playbook

Vi target.yml

---# Target Playbook

hosts: remo -- > remo: Groupname

user: ansible -- > ansible: You are ansible user now

become: yes -- > become: become sudo user -- > yes

connection: ssh

gather_facts: yes -- > Gives private IP of the nodes -- > yes

now save that file and execute the playbook by giving the command: ansible-playbook target.yml

Now create one more playbook in ansible server with cmd Vi task.yml

NORMAL PLAYBOOK:

```
---
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
     - name: installing git
       action: yum name=git state=present
```

TO EXECUTE: ansible-playbook playbook.yml
DRY RUN: Check whether the playbook is formatted correctly or not.
ansible-playbook playbook.yml --syntax-check

VARIABLES:

Ansible also provides various ways of setting variables. They are used to store values that can be later used in the playbook.

Variable names in Ansible should start with a letter. The variable can have letter, numbers and underscore. Invalid variable declaration comes when we use dot (.), a hyphen (-), a number or variable separated by mutlple words.

```
--- #VARIABLES
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  vars:
    pkgname: httpd
  tasks:
    - name: installing httpd
      action: yum name='{{pkgname}}' state=present
```

other way of passing arguments is by passing them to the command line while running using the –extra-vars parameter.

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install git
      action: yum name='{{abc}}' state=present
~
```

for single var: ansible-playbook one.yml --extra-vars "abc=git"

for multiple vars: ansible-playbook one.yml --extra-vars "abc=git def=maven"

MULTI VARIABLES:

```
- hosts: all
 become: yes
 tasks:
 - name: abc
 action: yum name={{a}} {{b}} state=present
```

Passing a Varaible file - A Varaible can be defined in a variable file and can be passed to a playbook using the **include**

```
---
  - set_fact: abc=httpd
  - name: install Apache
    yum: name=httpd state=present
```

one.yml

two.yml

```
---
- hosts: dev
  become: yes
  tasks:
    - include: one.yml
    - name: install git
      service: name='{{abc}}' state=restarted
```

HANDLERS:

Handler is same as task but it will run when called by another task. (OR)
and also indicates that it changed something.

```
--- # HANDLER
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
        - name: install httpd server on centos
          action: yum name=httpd state=installed
          notify: restart httpd
  handlers:
          - name: restart httpd
            action: service name=httpd state=restarted
~
```

LOOPS:

Ansible loop includes changing ownership on several files & directories with file module, creating multiple users with user modules and repeating a polling step until result reached.

```
--- # LOOPS
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: add list of users in my nodes
            user: name='{{item}}' state=present
            with_items:
                    - raham
                    - mustafa
                    - shafi
                    - nazeer
```

CONDITIONS:

If we have different scenarios, then we apply conditions according to the scenarios.

WHEN STATEMENT

Sometimes we want to skip a particular command on a particular node.

```
--- # CONDITIONS
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: Install apache server for debian family
            command: apt-get-y install apache2
            when: ansible_os_family== "Debian"
          - name: install apache server for redhat family
            command: yum install httpd -y
            when: ansible_os_family== "RedHat"
```

VAULT:

In ansible we can keep sensitive data like our passwords and keys in encrypted format.
ENCRYPTION TECHNIQUE: AES256 Used by Facebook.
ansible-vault create vault.yml : creating a new encrypted playbook.
ansible-vault edit vault.yml : Edit the encrypted playbook.
ansible-vault rekey vault.yml : To edit the password.
nsible-vault view vault.yml : To view the playbook without decrypt.
ansible-vault encrypt vault.yml : To encrypt the existing playbook.
ansible-vault decrypt vault.yml : To decrypt the encrypted playbook.

ROLES:

We can use two techniques for resulting a set of tasks they are Includes and Roles.

Roles are good for organizing tasks & encapsulating data needed to accomplish the task.

ANSIBLE ROLES: Default, Files, Handlers, Meta, Templates, Tasks, Vars.

We can organize playbooks into directory structure called Roles.

Adding more functionality to the playbooks will make it difficult to maintain in a single file.

mkdir -p playbook/roles/role1/tasks --- > To see o/p use tree command.

Cd playbook & touch master.yml & touch roles/webserver/tasks/main.yml

vi roles/webserver/tasks/main.yml

```
- name: install apache on redhat
  yum: pkg=httpd state=latest
```

```
--- #MASTER  PLAYBOOK
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  roles:
          - webserver
```

TAGS:

If you have a large playbook, it may be useful to run only specific parts of it instead of running the entire playbook. You can do this with Ansible tags. Using tags to execute or skip selected tasks

```
---
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: installing git
      action: yum name=git state=present
      tags: install
    - name: uninstalling git
      action: yum name=git state=absent
      tags: uninstall
```

TO EXECUTE A SINGLE TASK: ansible-playbook abc.yml --tags tagname

TO EXECUTE A MULTIPLE TASK: ansible-playbook abc.yml --tags tagname1,tagname2

TO SKIP A TASK: ansible-playbook abc.yml --skip-tags “uninstall”

GALAXY:

Ansible Galaxy is a free online repository that contains a vast collection of Ansible roles, modules, and playbooks. It is a centralized hub for Ansible automation content that is open-source, community-driven, and maintained by Red Hat, the company behind Ansible.

Ansible Galaxy allows users to search, download, and share Ansible roles, modules, and playbooks with the wider Ansible community.

ansible-galaxy init raham
ansible-galaxy search elasticsearch
ansible-galaxy search elasticsearch --author alikins
ansible-galaxy install alikns.elasticsearch
cd /home/ansible/.ansible/roles/
USER INFO:
Go to the ansible galaxy website and select a username
ansible-galaxy info bonddim.linux
ansible-galaxy collection install bonddim.linux

ADVANTAGES:

Very simple to set up and use.
No special coding skills are necessary to use Ansible's playbooks.
Ansible lets you model even highly complex IT workflows.
You can orchestrate the entire application environment no matter where it's deployed.

DIS ADVANTAGES:

Ansible does not have any notion of state like other automation tools such as Puppet
Ansible does not track dependencies and simply executes sequential tasks and stops when tasks finish, fail, or any error comes.
Ansible has external dependencies to Python modules
Windows interaction requires some scheming

JENKINS SETUP USING PLAYBOOK:

```yaml
---
- hosts: dev[0]
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: getting links from jenkins.io
      get_url:
        url: https://pkg.jenkins.io/redhat-stable/jenkins.repo
        dest: /etc/yum.repos.d/jenkins.repo

    - name: getting 2nd link from jenkins.io
      ansible.builtin.rpm_key:
        state: present
        key: https://pkg.jenkins.io/redhat-stable/jenkins.io.key

    - name: update links
      action: yum name="*" state=latest

    - name: install java
      command: sudo amazon-linux-extras install java-openjdk11 -y

    - name: install jenkins
      action: yum name=jenkins state=present


     - name: import systemd
       ansible.builtin.systemd:
         daemon_reload: yes

     - name: restart jenkins
       ansible.builtin.systemd:
         name: jenkins
         state: restarted
```

PLAYBOOK TO CREATE A FILE/FOLDER:

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: creating a file
      file:
        path: "jenkins.txt"
        state: touch
~
```

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: creating a file
      file:
        path: "folder"
        state: directory
~
~
```

TO ENTER A DATA IN A FILE:

```
---
- hosts: dev
  tasks:
    - name: inserting a data in a file
      copy:
        dest: "devops.txt"
        content: |
          hi this is devops file
          we are inserting the data ij a file
          using ansible playbook
~
```

PLAYBOOK TO CHANGE PERMISSIONS TO A FILE:

```yaml
---
- hosts: dev
  tasks:
    - name: change permissions to a file
      file:
        path: "devops.txt"
        state: touch
        mode: 777
```

PLAYBOOK TO DEPLOY A WEBAPP:

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install httpd
      action: yum name=httpd state=present

    - name: restart httpd
      service: name=httpd state=restarted

    - name: create a file
      file:
        path: "/var/www/html/index.html"
        state: touch

    - name: enter data in a file
      copy:
        dest: "/var/www/html/index.html"
        content: |
          <h1>this is my webapplication, i have deployed using ansible </h1>
~
```

PIP MODULE:

Ansible pip module is used when you need to manage python libraries on the remote servers.

There are two prerequisites if you need to use all the features in the pip module.

- The pip package should already be installed on the remote server.
- Virtualenv package should be installed on the remote server already if you need to manage the packages in the python virtual environment.

```
- hosts: all
  tasks:
  - name: Installing NumPy python library using pip module
    pip:
      name: NumPy
```

RAW MODULE:

RAW module is used when there is more need than Command module or if the command module does not support the operation. This module makes a SSH to the remote machine and run the command. For the Ansible to work we need to have Python available but for this module we don't need a Python to be available

```
---
- hosts: dev
  become: yes
  tasks:
    - name: Install VIM
      raw: yum -y install vim-common
~
```

PLAYBOOK TO GET CODE FROM GIT (PUBLIC REPO):

```
---
- hosts: localhost
  become: yes
  tasks:
    - name: getting code from git
      git:
        repo: "https://github.com/devops0014/pubg.git"
        dest: "/home/mycode"
~
```

PLAYBOOK TO GET CODE FROM GIT (PRIVATE REPO):

```
---
- hosts: localhost
  become: yes

  tasks:
    - name: link
      git:
         repo: 'https://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@github.com/devops0014/ansible.git'
         dest: "/home/mygitcode"
~
~
```

SYNTAX: TOKEN@GITHUB.COM/USERNAME/REPO.GIT

ANSIBLE SETUP MODULES:

ansible_os_family

os name like RedHat, Debian, Ubuntu etc..

ansible_processor_cores

No of CPU cores

ansible_kernel

Based on the kernel version

ansible_devices

connected devices information

ansible_default_ipv4

IP Mac address, Gateway

ansible_architecture

64 Bit or 32 Bit

After executing a playbook, if you want to see the output in json format
ansible -m setup private_ip

if you want to apply a see particular output, you can apply filter.
ansible -m setup -a "filter=ansible_os_family" private_ip
ansible -m setup -a "filter=ansible_devices" private_ip
ansible -m setup -a "filter=ansible_kernel" private_ip

ANSIBLE DEBUG:

it can fix errors during execution instead of editing your playbook.

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - debug:
        msg: "os family for {{ansible_fqdn}} is {{ansible_os_family}}"
~
```

You can see that the task is performing on which OS.

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - debug:
        msg: "ansible_memory_mb  memory is  {{ansible_memory_mb.real}}"

    - debug:
        msg: "ansible_memory_mb  free memory is  {{ansible_memory_mb.real.free}}"

    - debug:
        msg: "ansible_memory_mb used memory is  {{ansible_memory_mb.real.used}}"
~
~
```

Depends upon the memory, we can debug.

Depends upon the ip, we can debug.

```
- debug:
      msg: "ip info of all the devices is  {{ansible_all_ipv4_addresses}}"
```

If you run a command to check the files along with its content in a server, it will not shows us the output.
But we can debug the output.

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: get users
      command: cat /etc/passwd
      register: output

    - debug:
        msg: "users list in the ansible is {{output.stdout}}"
~
```

ANSIBLE IN-OUT OPERATOR:

To check weather GIT is installed or not.

```
---
- hosts: dev
  become: yes
  tasks:
    - name: Check for the GIT
      shell: rpm -qa  | grep git
      register: rpm_status
      ignore_errors: yes

    - name: check if the GIT is installed or not
      debug: msg="git is installed on the client server"
      when: " 'git' in rpm_status.stdout"

    - name: Check if GIT is installed
      debug: msg="GIT is not installed on the client server"
      when: not 'git' in rpm_status.stdout
~
```