**Ansible**

Ansible is open source, a configuration Management Tool and Deployment Tool, maintained by ***Redhat***

**Configuration Management:**

This is process of configuring remote servers from one point of control

Configuration is some tasks, which we want to execute on the server, it can be

* Creating users/Groups
* Installing software or running software
* Creating / updating / copying files
* Start / stop services etc
* Changing permissions/Ownership of files/directory
* Creating and deleting files and folders

It is not replacement tool, the main components of Ansible are playbooks, configuration management, deployment

Ansible uses the playbooks to deploy, manage, build, test and configure anything from full server environments to custom complied source code for applications.

Ansible was written in **Python**

**Open-source configuration tools:**

* Ansible
* Chef
* Puppet
* Saltstack

**Need of Ansible**

If we want to configure multiple system(i.e 100+) with same configuration without anisible we have to configure manually each machine, in this process their may chance of error prone i.e human error and time consumption.

**Advantages**

1. **Provisioning of servers**

The applications that should be installed on server can be done very quickly from a **single centralized location**.

1. **Disaster recovery**

In case of natural disaster, there is a possibility that a complete data center can be lost. Configuration management tools can capture the information about these servers and store them in **simple files called as setup files**. These setup files can be used for re configuring a similar data center at another location.

1. **Idempotent**

Configuration management tools are used to bring the server to a particular state, called as desired state. If a server already in the desired state, configuration management **tools will not reconfigure that server**.

**Note:** Configuration management tools cannot be used for installing OS from the scratch. They can be used only for managing the applications on top of the OS.

**Ansible how different is from Terraform?**

* **Terraform:** Infrastructure as a code
* By using terraform we can create the server once we have a server, we can use Ansible to configure those servers based on our requirement
* Ansible does not maintain state of server, by using Ansible also we can create the server but it is not better Option.

**Ansible Features**

* It is a open source *configuration management tool*, created using Python.
* Main machine in which ansible is installed is called as controller.
* Remote severs that Ansible configures, are called as managed nodes.
* Ansible configure machines in an agent-less manner using SSH
* Ansible uses agent less policy for configures remote server’s i.e Ansible is installed only on 1 machine, and we do not require any client side software to be installed on the remote servers.
* Built it on top of Python and hence provide a lot of Python’s functionality
* YAML – Based Playbooks
* Uses SSH for secure connections
* Follows Push based architecture for sending configurations
* Ansible performs configuration management through password less SSH.

**Two types of configuration tools**

1.Push based

2.Pull based

**Push based**

Master Server (Installed Ansible) have some configuration scripts, then Main server only will connect to different host servers for copying files or install software’s etc

In other words, Center server pushes the configuration information on target servers

Ansible is push based

**Pull based**

Here, target server only pull to **center server**

**Tools like** Puppet and chef and pull based.

This servers are pull based configuration management tools

Agents on the server periodically checks for the configuration information from central server (Master)

**Note:**

**We cannot install ansible in window machine**

**We can configure windows machine using Ansible**

Ansible is not master slave architecture, host machine can be anything for

example installed Jenkins, installed Docker, installed Kubernetes.

**Why Ansible?**

* Predefined module/functionality
* No need to write everything from scratch
* Ansible scripts basically YAML based
* Not required to learn scripting language, because everything will be **YAML**
* It will save the time
* Ansible Modules are idempotent

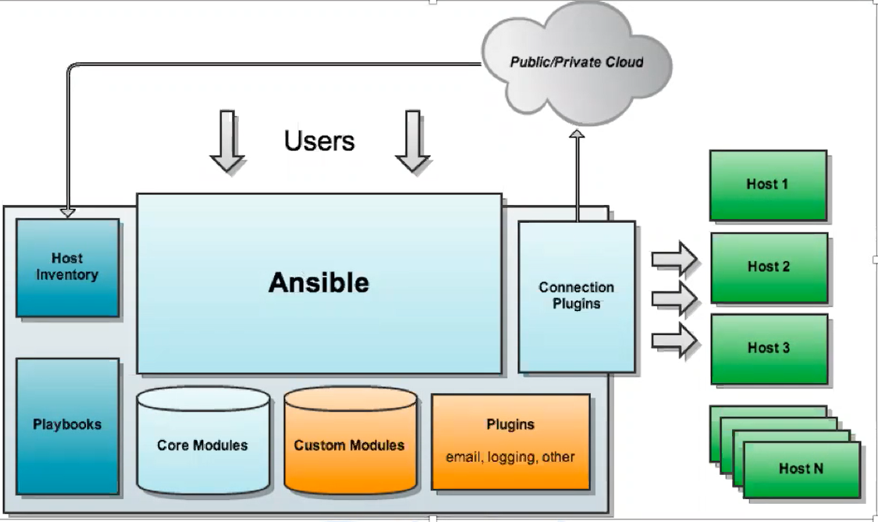
Idempotent: If configuration already done for that server, it will not change anything

* Lot of predefined functionality available in Ansible
* Agent less and push based
* Manage the machine using SSH protocol

**What can Ansible do?**

* Configuration management tool
* Application Deployment
* Continuous Delivery

**Ansible Architecture**



**Ansible Modules**

1.Host Inventory

2.Playbooks

3.Core Modules

4.Custom Modules

5.Plugins (email,logging,other)

6. Connection Plugins

**Host Inventory file**

Ansible’s inventory hosts file is used to list and group your servers Which we want to configure using ansible

Its default location is

* **etc/ansible/hosts**

**I**

See the content of the host file

# cat /etc/ansible/hosts (default inventory file path)

# 192.168.122.1 This is one of the node IP

**In Inventory file we can mentioned IP address or hostname also**

**Note:** We can create our own inventory for that we need to mentioned in below path The ansible\* command will use a different host inventory file when they are used with the

**--inventory PATHNAME option, -I PATHNAME for shot**

Some important points in inventory file

* Comments begin with the ‘#’ character
* Blank lines are ignored
* Groups of hosts are delimited by [header] element
* You can enter hostnames or IP multiple groups
* A hostname/IP can be a member of multiple groups
* Ungrouped hosts are specifying before any group headers, like below

**Host Inventory**

It will maintain the details for HOST machine (server) to configure, it can be list and group of servers (Host)

1. Static Inventory

2. Dynamic Inventory

**Static Inventory**

It is file in which we can list host(server) details like (host name, password details etc and we can group the servers

Default location of host inventory

* /etc/ansible/hosts

**Dynamic Inventory**

If any changes in infrastructure like remove ***server add some servers or terminate server*** We need to update the inventory file based on the changes

It is script (like python, shell scripts) which will fetch host details dynamically from external source like cloud Provider etc

No need to update the details manually

**When we can go for Static and Dynamic Inventory?**

If our requirement is not dynamic (like not creating server / deleting servers / terminating / modifying server very frequently then we can go for **static inventory** or else we can use dynamic inventory

**Connection Plugin**

To connect windows or Linux it will use some connection plugin

For Linux 🡪 SSH

For window 🡪 WINRM(windows remote machine)

**Playbooks**

Playbook is a script which we want to execute(configuration) YAML based scripts

It contains task which we want to perform on the servers

**Sample script**

host: all 🡪 all the machine going to perform below task

name: create a user

user:

name: rajesh

createHome: yes

password: abc123

**Module**

Modules are the programs that perform the actual work of the tasks of a play

**Core Module**

Ansible comes with lot of predefined modules we will use ansible module to write task There are 400 core modules

**Tasks**

The goal of play is to map a group of hosts to some well-defined roles, represented by things ansible calls tasks. At a basic level, a task is nothing more than a call to an ansible module

**Custom Module**

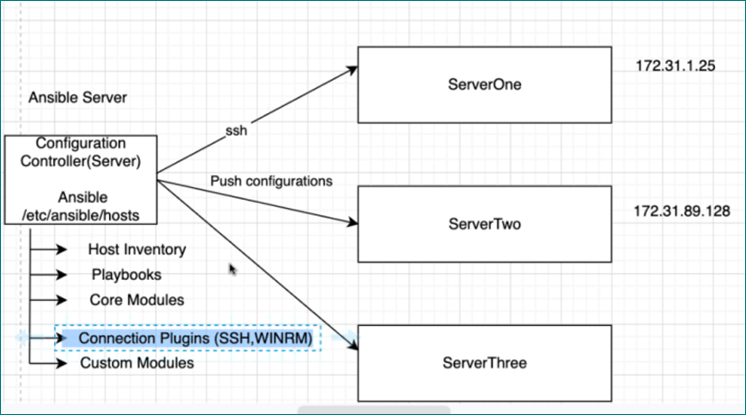
If expected module which is not available in the core module, in this case we will create our own module and will publish same will called as Custom Module

**Playbook:**

Ansible playbooks are written using the YAML it’s Markup Language (YAML) language.

We can create a file with .yml or yaml extension

* YAML files optionally begin with a three dash (---)
* Next immediate line starts with single dash (-). Name is optional
* hosts expect value like all or group
* Do you want to become a root on target server uses **become**
* What action do you want to perform? specify under tasks



**Syntax of adhoc command**

# ansible all/host\_name/ip\_address -i path\_of\_inventory\_file -m module\_name -a arguments

**Default Inventory host location:**

**Default Location: /etc/ansible/hosts**

To open Default inventory file:

**# sudo vim /etc/ansible/hosts**

If you want you ansible host file in another location, then we need to set this environment variable

**export ANSIBLE\_HOSTS=/home/ansible/custom\_inventory\_path**

(OR)

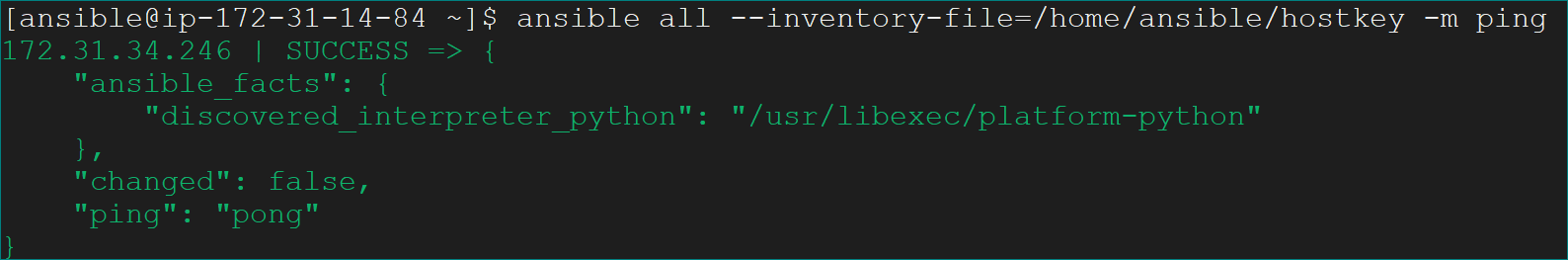
We can specify the ansible host while running command with the --inventory –file or (-i) flag:

**ansible all --inventory-file=/home/ ansible/custom\_inventory\_path -m ping**

**ansible all –i /home/ansible/custom\_inventory\_path –m ping**

**Note: ‘all’ is not part of controller(i.e ansible machine)**

**ansible locahost -a “df –h”**



**Note:**

If we don’t mentioned any file/module name default will be inventory file/command module

We can define the IP Address of locahost(i.e controller machine) ip address in host file

**ansible all -m free**

**ansible all -m ‘free’**

**Command Module**

This module is used for executing basic Linux commands on managed nodes

The default module is module. When using command module we need not use -m option

* **ansible all -a free**
* ansible all -m command -a 'free' or ansible all -m command -a free
* ansible all -m command -a df –h

**User Module**

Command to create user and set home directory, user id, default working shell etc

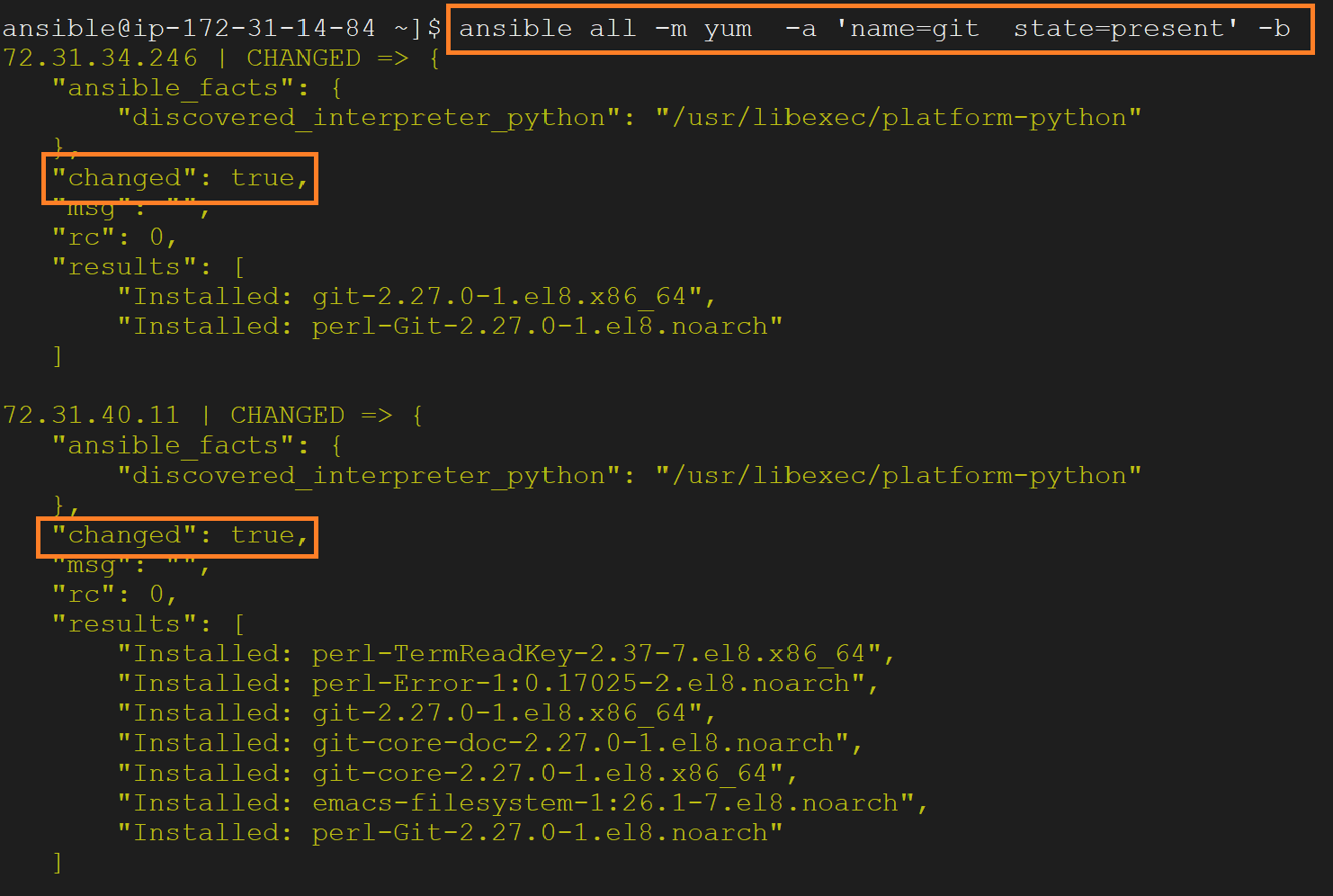
**ansible all -m user -a 'name=minu password=mani uid=133 comment=userinfo home=/home/ansible/ shell=/bin/bash' -b**

**-b become for higher privileges on managed nodes**

**Yum Module(install software into managed nodes)**

ansible all -m yum -a ‘name=git state=present’ –b

ansible all -m **apt**  -a ‘name=git state=present’ –b (in UBUNTU Operating System)



Note:

**“Changed : true which means expected command successfully executed in managed nodes**

**We get "changed": false**

**That means git is already installed on it. The command has no effect in the nodes**

**File Module**

**ansible all -m file -a 'name=/home/ansible/file1.txt state=touch' -b**

state=touch is to create files

state=directory is to create directory

state=absent is for deleting file/directory

**Copy Module**

Modules are the programs that perform the actual work of the tasks of a play

Core modules are the modules that come bundled with Ansible

This is used for copying the files from controller into managed nodes.

We know in the file **/etc/passwd** we have all the information about users

Now I want to copy the file into all nodes

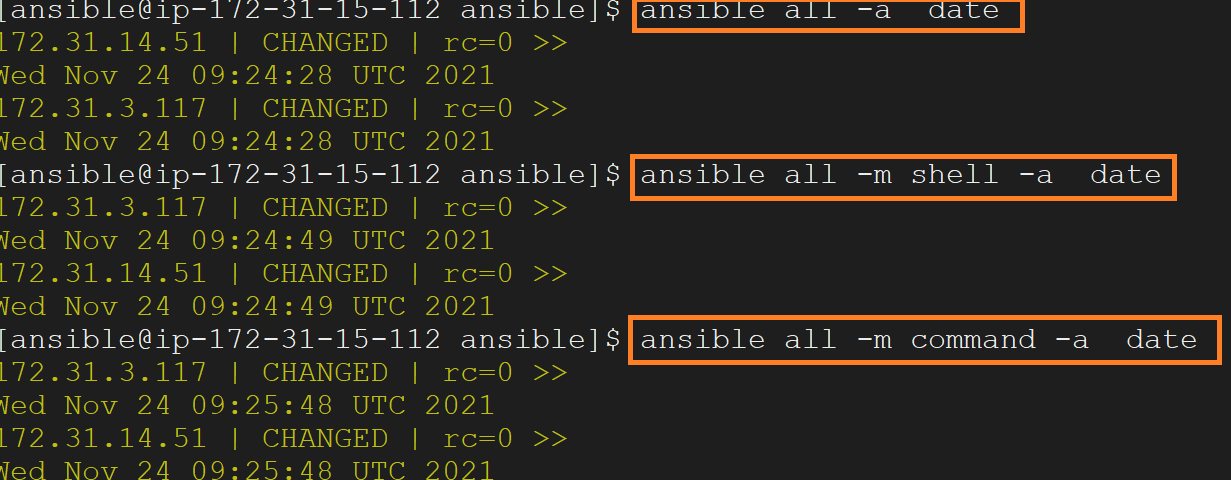
**ansible all -m copy -a 'src=/etc/passwd dest=/tmp'**

The following ad hoc command with copy module copies the file from Src location on the local control machine to the specified location on the remote server

**ansible testserver -m copy -a "src=~/Downloads/index.html dest=/var/www/html owner=apache group=apache mode=0644"**

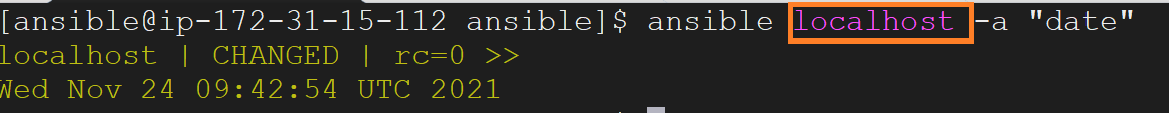
**Shell Module**

The `shell' module takes the command name followed by a list of space-delimited arguments. Either a free form command or `cmd' parameter is required, see the examples. It is almost exactly like the [command] module but runs the command through a shell (`/bin/sh') on the remote node. For Windows targets, use the [win\_shell] module instead.

****

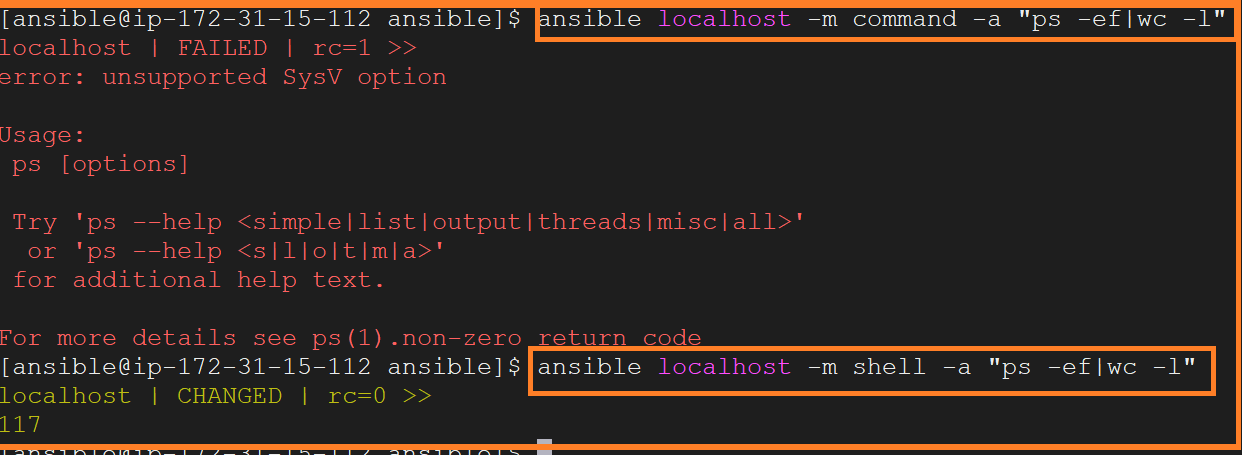
Default module will be **command** module only

Using **localhost**



**Difference between command and shell module**

* Command module will not support regular expressions and operators, variable etc..
* And it will support only system commands
* Shell module will support all regular expression and operators
* This module will be non-idempotent, it will execute ‘n’ number of times



**Yum Module**

Ansible has a specific module for managing the Yum packages. You can install, remove, upgrade or downgrade versions and many more by using this module.

Whether to install (`present' or `installed', `latest'), or

remove (`absent' or `removed') a package.

`present' and `installed' will simply ensure that a desired

package is installed.

`latest' will update the specified package if it's not of the

latest available version.

`absent' and `removed' will remove the specified package.

Default is `None', however in effect the default action is

`present' unless the `autoremove' option is enabled for this

module, then `absent' is inferred.

(Choices: absent, installed, latest, present,

removed)[Default: (null)]

Both the "present" and "installed" parameters have the same behavior.

Installing Httpd service in host machines

**#installing service from controller machine**

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

**#verify service status from controller machine**

**ansible all -b -a "service httpd status**"

**#Uninstall service**

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

**ansible -b all -m yum -a "name=httpd state=removed"**

**Service Module**

Used for starting / stoping / restarting services on managed nodes.

**Start the httpd service using service module**

**#starting the httpd service from controller machine**

**ansible -b all -m service -a "name=httpd state=started enabled=yes"**

**#Verify service status from controller machine**

**ansible all -b -a "service httpd status"**

**#stop service from controller machine**

**ansible -b all -m service -a "name=httpd state=stopped"**

**state values are “must be one of: reloaded, restarted, started, stopped”**

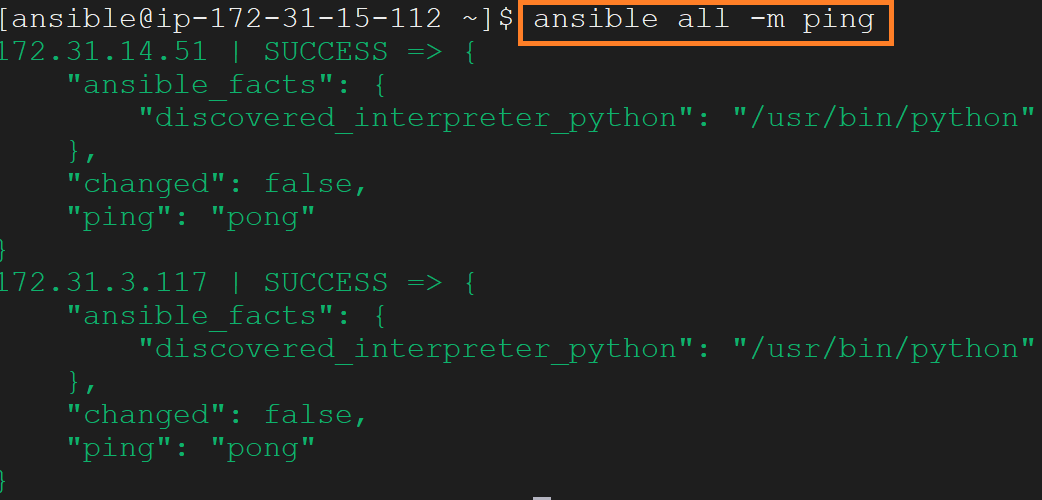
**ping Modue**

This module is used to check if the remote server is pingable or not.

**ansible <all/group/ip-address> -m ping**

ansible 🡪 Every ansible commands start with ansible

all 🡪 all managed nodes which is updated in host file



**User Module**

This module is used for user management like create user, setting password, assign home directory etc

**#Creating users from controller machine**

**ansible all -m user -a "name=testuser" –b**

**#Verify user created or not in host machine**

**ansible all -a "cat /etc/passwd"| grep testuser**

**Cron Module**

**# Run the job every 15 minutes**

$ ansible multi -s -m cron -a "name='daily-cron-all-servers' minute=\*/15

job='/path/to/minute-script.sh'"

**# Run the job every four hours**

$ ansible multi -s -m cron -a "name='daily-cron-all-servers' hour=4

job='/path/to/hour-script.sh'"

**# Enabling a Job to run at system reboot**

$ ansible multi -s -m cron -a "name='daily-cron-all-servers' special\_time=reboot

job='/path/to/startup-script.sh'"

**# Scheduling a Daily job**

$ ansible multi -s -m cron -a "name='daily-cron-all-servers' special\_time=daily

job='/path/to/daily-script.sh'"

**# Scheduling a Weekly job**

$ ansible multi -s -m cron -a "name='daily-cron-all-servers' special\_time=weekly

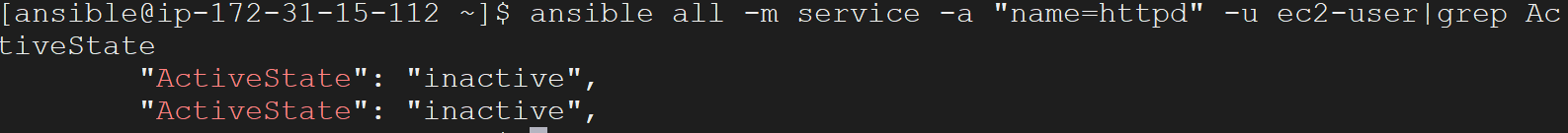
job='/path/to/daily-script.sh'"

**Ansible ad hoc command to check the service status**

In this example, we are going to see how to check the status of service using ansible ad hoc command

For example, let’s suppose we want to check the status and more information about the httpd service the following ad hoc command with ansible service module would help

**ansible all -m service -a "name=httpd" -u ec2-user|grep ActiveState**



-u – remote SSH user

**ansible ad hoc command to download a file from URL**

To download a file from URL in ansible ad hoc. You can either invoke linux commands like CURL or WGET but the preferred way is to use the get\_url module of Ansible.

This is how you can use get\_url module in Ansible ad hoc to download a file in remote system

**ansible all -m get\_url -a "url=https://nodejs.org/dist/v14.17.4/node-v14.17.4-linux-x64.tar.xz dest=/tmp mode=0755"**

**File Module**

Ansible file module is used to creating and deleting the file or multiple files in the remote server. You can also create and delete the directories and change the permissions of the data

#**Deleting file from remote location**

**ansible all -m file -a "path=/tmp/node-v14.17.4-linux-x64.tar.xz state=absent"**

**#creating file with all parameters**

**ansible all -m file -a " state=touch path=/tmp/test.txt mode=777 owner=testuser group=testuser" –b**

**#creating directory**

**ansible all -m file -a "path=/file/newfolder state=directory" –b**

**LineInFile:**

This is module will add the content to the file

**#add content to the file**

**ansible all -m lineinfile -a 'dest=/tmp/test.txt line="This is lineinfiles"' -b**

**Group Module:**

We can create and remove group using group module

**#create group**

**ansible all -m group -a "name=devops state=present gid=2002" –b**

**#add user to the group**

**ansible all -m user -a "name=jagan group=devops state=present" -b**

**#delete group**

**ansible all -m group -a "name=devops state=absent" –b**

**GroupModulePlay.yaml**

---

- hosts: appServers

become: true

tasks:

- name: Creating group management

group:

name: production

state: present

gid: 2021

- name: creating user management

user:

name: suresh

group: production

state: present

...

**Yum Module**

**ansible all -m yum -a "name=java download\_only=yes update\_cache=yes" -b**

**YumModuelPlay.yaml**

**#This play book will start the httpd and start the service**

**---**

**- hosts: appServers**

**become: true**

**tasks:**

**- name: Install httpd**

**yum:**

**name: httpd**

**state: latest**

**update\_cache: yes**

**- name: start httpd**

**service:**

**name: httpd**

**#state: started**

**enabled: yes**

**- name: copy html file to host machines**

**copy:**

**src: index.html**

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

**...**

**package Module:**

This modules manages packages on a target without specifying a package manager module (like [ansible.builtin.yum](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/yum_module.html" \l "ansible-collections-ansible-builtin-yum-module), [ansible.builtin.apt](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html" \l "ansible-collections-ansible-builtin-apt-module), …). It is convenient to use in an heterogeneous environment of machines without having to create a specific task for each package manager.

**Note:**

**If we use package module, Irrespective of operating system software will be installed in host machine**

**ansible appServers -m package -a "name=httpd " –b**

**ansible appServers -m package -a "name=httpd state=absent" -b**

**Fetch Module:**

* This is opposite of copy module
* I want to get the file ( server.xml) from node to controller
* Fetch module is used to copy files from managed nodes to controller.
* Command to copy ***tomcat-server.xml*** file from all managed nodes into **/tmp** folder on the controller.

**fail\_on\_missing:**

When set to 'yes', the task will fail if the remote file cannot be read for any reason. Prior to Ansible-2.5, setting this would only fail if the source file was missing.

The default was changed to "yes" in Ansible-2.5.

**ansible appServers -m fetch -a "src=/home/ec2-user/file3.txt dest=/tmp/var/ fail\_on\_missing=no" –b**

**ansible all -m fetch -a 'src=/etc/tomcat8/server.xml dest=/tmp'**

**FetchModulePlay.yml**

**---**

**- hosts: appServers**

**tasks:**

**- name: Copy files from host to ansible machine**

**fetch:**

**src: /home/ec2-user/file4.txt # host machine source info**

**dest: /tmp/ # controller machine destination details**

**fail\_on\_missing: no**

**...**

**Git Modules**

This is used to perform git version controlling on the managed nodes.

**ansible all -m git -a 'repo=https://github.com/sunildevops77/repo1.git dest=/tmp/mygit' -b**

The above command will download the files in all managed nodes.

**GitModulePlay.yml**

***#Example read-write git checkout from github***

**---**

**- hosts: appServers**

**tasks:**

**- name: This is used to perform git version controlling on the managed nodes.**

**git:**

**repo: https://github.com/Venki22/sampletest.git**

**dest: /tmp/mygit**

**clone: no # if ‘no’ do not clone the repository if it does not exist locally**

**update: no**

**...**

**Note:**

Ansible command to clone remote git repository into all managed nodes

**Replace Module**

* This is used for replacing a specific string with other string
* Ansible command to change the port number of tomcat from 8080 to 9090
* This module will replace all instances of a pattern within a file. It is up to the user to

maintain idempotence by ensuring that the same pattern would never match any replacements made.

**ansible localhost -m replace -a "regexp=8080 replace=9090 path=/etc/tomcat8/server.xml backup=yes"**

**ReplaceModulePlay.yml**

---

- name: File is ReplaceModulePlay.yml

hosts: appServers

tasks:

- name: This is used for replacing a specific string with other string

replace:

regexp: 8080

replace: 9090

path: /home/ansible/file22

backup: yes

...

**URI Module**

I want to check ‘facebook’is reachable for not in all managed nodes.

urI module is used to check if the url is reachable or not.

Command to check if facebook.com is reachable on all managed nodes.

**ansible appServers -m uri -a "url=http://facebook.com" user=admin password=Oct2021$ body\_format=json method=POST status\_code=200"**

**Play books:**

* Adhoc commands are capable of working only on one module and one set of arguments.
* When we want to perform complex configuration management activities,
* adhoc commands will be difficult to manage.

In such scenarios, we use play books.

* Play book is combination of plays.
* Each play is designed to do some activity on the managed nodes.
* These plays are created to work on single host or a group of hosts or all the hosts.
* The main advantage of play books is reusability.
* Play books are created using yaml files.
* Playbook start with dashes**(---)** and end with **(…)**
* Each play has first hosts, variable and tasks

**#Test Connection with managed nodes**

**---**

**- name: Test Connection**

**hosts: appServers**

**tasks:**

**- name: ManagedNodes connection Test**

**remote\_user: "{{name}}"**

**ping:**

**- name: verify connection**

**debug:**

**msg: "host machines {{name}} connected successfully"**

**...**

**#To run**

**ansible-playbook testConnection\_ping.yml --extra-vars " name=ansible"**

**UserManagement.yml**

- name: Create user and copy password file

hosts: all

tasks:

- name: User creation

user:

name: kiran

password: sunilsunil

uid: 6779

home: /home/kiran

- name: Copy password into users home dir

copy:

src: /etc/passwd

dest: /home/kiran

...

**#Playbook Options:**

**$ ansible-playbook -help**

$ ansible-playbook <Playbook-name.yml> -b

$ ansible-playbook <Playbook-name.yml> --check

$ ansible-playbook <Playbook-name.yml> --syntax-check

$ ansible-playbook <Playbook-name.yml> -v

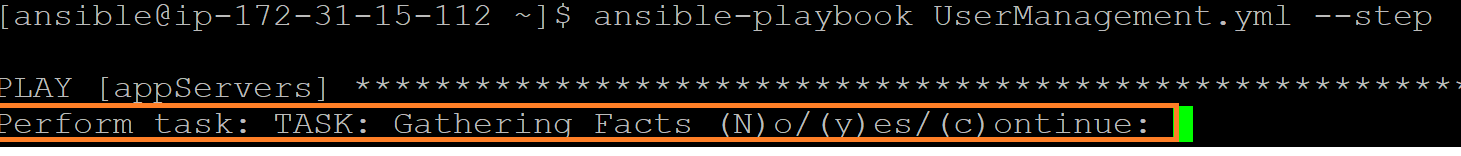
$ ansible-playbook <Playbook-name.yml> -vv

$ ansible-playbook <Playbook-name.yml> -vvv

$ ansible-playbook <Playbook-name.yml> --list-hosts

$ ansible-playbook <Playbook-name.yml> --step

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



$ ansible-playbook <Playbook-name.yml> --list-tags

$ ansible-playbook <Playbook-name.yml> --version

$ ansible-playbook <Playbook-name.yml> --list-tasks

$ ansible-playbook <Playbook-name.yml> --inventory <custom-path>

$ ansible-playbook <Playbook-name.yml> -i <custom-path>

**Note:** If any error while running playbook, use -v,-vv or -vvv option to debug the playbook

**Variable:**

There are different ways in which you can define variables in Ansible.The simplest way is by using vars section of a playbook.The below example defines a variable name called package and it is using in a task called **Install a Package**

**We have different types of variables**

* Runtime vars
* Play scope variable
* Group vars
* Host vars
* Role vars

**#File name is InstallUnstallPackage\_vars.yml**

---

- hosts: appServers

Become:”{{become}}”

tasks:

- name: Install and Unstall

package:

name: "{{name}}"

state: "{{state}}"

#update\_cache: "{{update\_cache}}"

...

**#To run the playbook by passing values to the variables**

**ansible-playbook InstallSoftware\_vars.yml --extra-vars "name=git state=latest update\_cache=yes" -b**

**ansible-playbook InstallSoftware\_vars.yml --extra-vars "name=git state=absent" -b**

**Play Scope Variable:**

Playscope variables are definined within the playbook and thet can effect only in one single play.

These variables are definied at level of individual plays and they can effect only one play.

**#Playscope variables are defined within the playbook and thet can effect only in one single play.**

---

- name: playscope varialbe

hosts: appServers

**become: true**

**vars:**

a: git

b: absent

#c: yes

tasks:

- name: Install git

package:

name: "{{a}}"

state: "{{b}}"

# update\_cache: "{{c}}"

...

**#To run**

ansible-playbook playscoptVariablePlay.yml

**#We can run by using extra vars from command line**

ansible-playbook playscoptVariablePlay.yml --extra-vars "**a=tree b=present"** -b

The above command will install tree because **global scope variables have higher priority**