**Ansible:**

Agentless and uses SSH to communicate.

Uses Python because it is widely used compared to Ruby.

Ansible

Ansible Playbook Yaml files

Inventory file is used to communicate to target machines.

roles

Anisible manages manages machines in an agentless manner with SSH/WINRM/REST.

Inventory is an INI file which contains information about servers you are managing. Host file.

Ansible Playbook the entry point for ansiible provisioning where automation is defined through tasks using YAML format.

Install Ansible in ansible host using below commands:

**sudo apt-get install software-properties-common && \**

**sudo apt-add-repository ppa:ansible/ansible && \**

**sudo apt-get update && \**

**sudo apt-get install ansible**

We can get multiple ansible commands using

**ls -lrt /usr/bin**

**-rwxr-xr-x 1 root root 12493 Nov 14 01:16 ansible-connection**

**-rwxr-xr-x 1 root root 5932 Nov 14 01:16 ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-vault -> ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-pull -> ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-playbook -> ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-inventory -> ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-galaxy -> ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-doc -> ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-console -> ansible**

**lrwxrwxrwx 1 root root 7 Nov 14 01:16 ansible-config -> ansible**

But widely we use ansible,ansible-playbook,ansible-vault,ansible-galaxy.

In order to establish connection with target nodes we need to edit the **hosts** file located in **/etc/ansible** directory.

Remove everything from hosts **>hosts** and edit the hosts file **vi hosts** andadd the target machines private ip in hosts file.

Now got to **ansible.cfg** file in **/etc/ansible** directory and uncomment to disable SSH key host checking.

# uncomment this to disable SSH key host checking

**host\_key\_checking = False**

Also enable log rotation by uncommenting below.

# logging is off by default unless this path is defined

# if so defined, consider logrotate

**log\_path = /var/log/ansible.log**

Ansible has so many modules. You can get different modules from below URL.

<https://docs.ansible.com/ansible/latest/modules/modules_by_category.html>

If want to check whether communication is established or not.

**ansible -i hosts all -m ping**

Where:

-i is to specify inventory file which is hosts

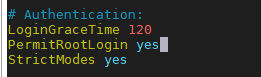
all is for all nodes specified in hosts file

-m is module name which is ping

But these will be unreachable because we have not provided authentication.

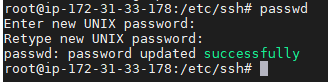
Now in ansible host use **ssh-keygen** command to generate public and private keys and take the public key from **id\_rsa.pub** file and paste it in **authorized\_keys** of target nodes.

For username and password approach go to ansible target into **/etc/ssh** directory and edit **sshd\_config** and change the **PermitRootLogin** from **without-password** to **yes**. Also change **PasswordAuthentication** from **no** to **yes**.

Once done **service ssh restart**

Later give new password.



Now go to ansible host and give below command for password authentication.

**ansible -i hosts dev -m ping -u root -k**

Give password and it will ping.

If we have hundreds of target nodes then we can use **authorized\_keys** module.

Now if we give **ansible -i hosts all -m ping** we will get connection success.

To install any package like tree package:

**ansible -i hosts all -m apt -a “name=tree state=latest”**

here it is -m module is apt and -a attribute name is tree and state is latest. It will install tree package in all hosts linked to ansible host.

If we execute another time it will not do anything. So it will exhibit idempotent behavior.

If want to install package in local machine or ansible host then update the host file with

**localhost ansible\_connection=local**

We are mentioning **ansible\_connection=local** because it should not use default ssh communication to communicate.

So in order to perform multiple tasks like package installation, files and service startup, user creation and group creation for configuring node we will be using ansible-playbooks in yaml format. If use CLI to do these tasks then we have to execute the ansible commands multiple times to overcome this we will use playbooks.

All modules are kept in YAML files (playbooks)

We can also group the target nodes adding the group name. Dev and local are group names. We can add any number of nodes under the group.

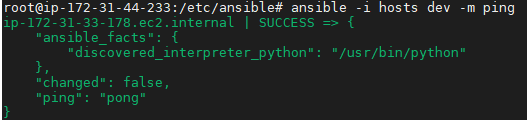
**[dev]**

**ip-172-31-33-178.ec2.internal**

**[local]**

**localhost ansible\_connection=local**

Now if ping only dev it will execute only nodes under dev group.



Please find below YAML file and sample playbook.

---

**- hosts: 'dev'**

**vars:**

**package\_name: nginx**

**tasks:**

**- name: add repo**

**apt\_repository:**

**repo: ppa:nginx/stable**

**- name: install nginx packagename**

**apt:**

**name: "{{ package\_name }}"**

**state: latest**

**- name: copy server.xml file**

**copy:**

**src: server.xml**

**dest: "/etc/nginx/server.xml"**

**notify: restart the service**

**- name: start the service**

**service:**

**name: "{{ package\_name }}"**

**state: started**

**handlers:**

**- name: restart the service**

**service:**

**name: "{{ package\_name }}"**

**state: restarted**

In Ansible all executions happen sequentially. So we can use handler in such a way that when any changes or modification in server.xml file service should restart. So we use meta parameters like notify to do it. Also in vars section we have declared local variable package\_name: nginx. So wherever nginx we can replace with "{{ package\_name }}".

If repository is not there for a particular host then it will not pick the package from repository. To overcome this we have to add repo as well.

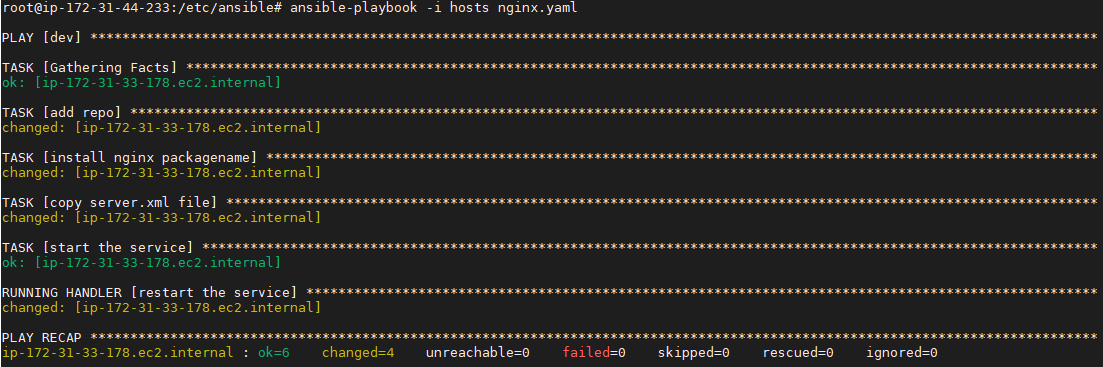
Now add above yaml code in nginx.yml file.

Ansible cannot take yaml files are input but only parameters. Ansible-playbook can take yaml files as input.

Now execute ansible-playbook using below command.

**ansible-playbook -i hosts nginx.yaml**

Below is the output:



If green no change and if yellow changed, red failed.

If you want to go into more deeper. Like debugging mode.

**ansible-playbook -i hosts nginx.yaml -vvv** upto 5 v’s can be given.

If ansible target node is rhel and ansible controller is ubuntu. then we have to use when statement like below.

**- name: install nginx packagename**

**apt:**

**name: "{{ package\_name }}"**

**state: latest**

**when ansible\_osfamily == “Debian”**

**- name: install nginx packagename**

**yum:**

**name: "{{ package\_name }}"**

**state: latest**

**when ansible\_osfamily == “redhat”**

Note: Ansible host cannot be windows machine but target nodes can be in any machine either windows or unix.

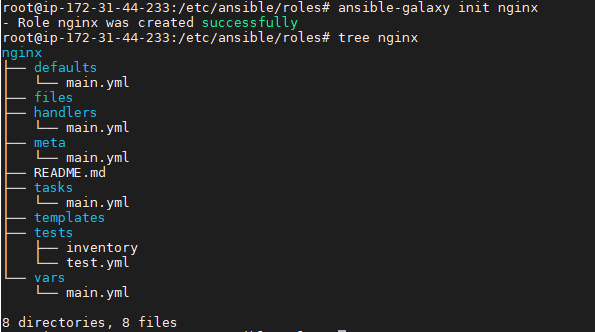
**Role**: a pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of a provisioning

If we have reusable code then we can declare them in roles and these roles can be called in our yaml file.

So in order to create roles we use ansible-galaxy command. So go to roles folder and issue below command.

**Ansible Galaxy** is a repository for Ansible Roles that are available to drop directly into your Playbooks to streamline your automation projects.

**ansible-galaxy init nginx**

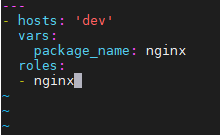


Now copy nginx.yaml which we created before to site.yaml in /etc/ansible directory.

**cp nginx.yaml site.yaml**

**vi site.yaml**

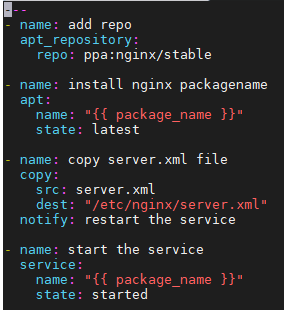
One edited call the roles as below. We can call any number of roles. Just append as - jenkins, - tomcat etc along with - nginx.



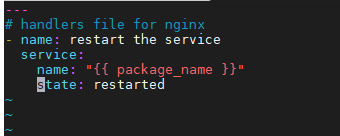
Now got to **/etc/ansible/roles/nginx/tasks/main.yml** file. Edit main.yml file. now copy only tasks in the file.

Refer below github repository for example.

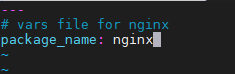
<https://github.com/alokvk1234/ansible-jenkins>



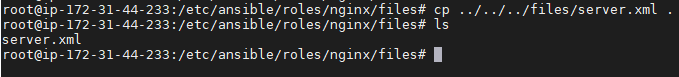
Now edit /etc/ansible/roles/nginx/handlers/main.yml file.



package we can declare in etc/ansible/roles/nginx/vars/main.yml file.

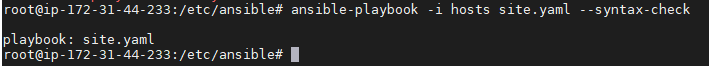


Files we have to keep in etc/ansible/roles/nginx/files directory.



Once done we can perform syntax check by giving below command. If any errors it will show up.

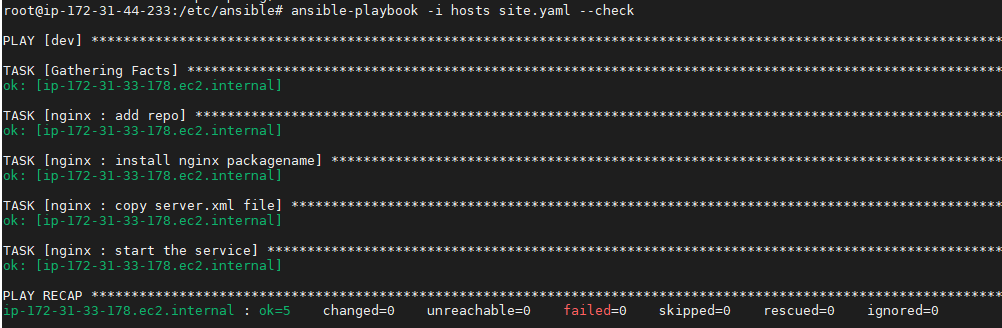
**ansible-playbook -i hosts site.yaml --syntax-check**

****

Before executing playbook we can check it whether it executes correctly in target machine by giving --check or -C.

**ansible-playbook -i hosts site.yaml --check**

So it executes perfectly as per below screen shot.

****

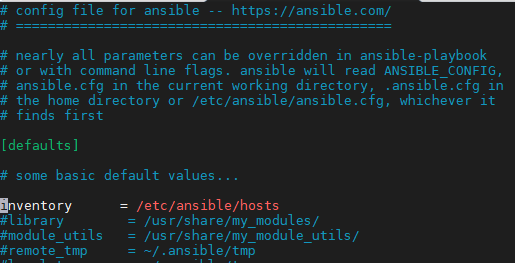
Now you can run the playbook.

**ansible-playbook -i hosts site.yaml**

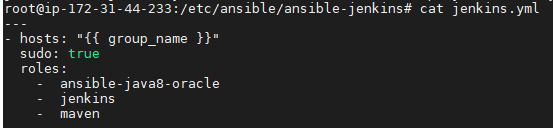
Note: No need to declare the inventory file everytime in command. You can got to ansible.cfg file in /etc/ansible directory and uncomment the defaults inventory. Once done we can give command directly like below

**ansible all -m apt -a “name=git state=latest”** noneed to give **-i hosts** it is picked up by default

**ansible-playbook site.yaml** noneed to give **-i hosts** it is picked up by default

****

In yaml file if you’re giving hosts as some variable like below and in inventory file if group\_name is webserver like below.





Then execute ansible-playbook as below

**ansible-playbook -i jenkins.ini jenkins.yml -e “group\_name=webserver”**

Note: Multiple modules can be executed at a time based on certain conditions then we can use block statements. In order to handle exceptions

**Blocks**

Blocks allow for logical grouping of tasks and in play error handling. Most of what you can apply to a single task (with the exception of loops) can be applied at the block level, which also makes it much easier to set data or directives common to the tasks. This does not mean the directive affects the block itself, but is inherited by the tasks enclosed by a block. i.e. a when will be applied to the tasks, not the block itself.

tasks**:**

**-** name**:** Install, configure, and start Apache

block**:**

**-** name**:** install httpd and memcached

yum**:**

name**:**

**-** httpd

**-** memcached

state**:** present

**-** name**:** apply the foo config template

template**:**

src**:** templates/src.j2

dest**:** /etc/foo.conf

**-** name**:** start service bar and enable it

service**:**

name**:** bar

state**:** started

enabled**:** True

when**:** ansible\_facts['distribution'] == 'CentOS'

become**:** true

become\_user**:** root

ignore\_errors**:** yes

In the example above, each of the 3 tasks will be executed after appending the *when* condition from the block and evaluating it in the task’s context. Also they inherit the privilege escalation directives enabling “become to root” for all the enclosed tasks. Finally, ignore\_errors: yes will continue executing the playbook even if some of the tasks fail.

**Blocks error handling**

Blocks also introduce the ability to handle errors in a way similar to exceptions in most programming languages. Blocks only deal with ‘failed’ status of a task. A bad task definition or an unreachable host are not ‘rescuable’ errors.

**Playbook -conditionals**

Conditionals can be added to Playbook to check some condition and then decide whether task should get executed or not.-name: Shutdown Debian Based Systemscommand: /sbin/shutdown -t nowwhen: ansible\_os\_family== "Debian"This task will check if target machine’s OS family is of Debian or not.If it is Debian then it will shutdown the machine

**Playbook –Templates**

Templates are typically used to set up configuration files, allowing for the use of variables and other features intended to make these files more versatile and reusable.

**Playbook –Handlers**

•Handlers are used to trigger a state change in a service, such as a restart or a stop.

•Handlers are typically defined as an array in a handlers section of the playbook.

**handlers:**

**-name: restart apache**

**service: name=apache2 state=restarted**

**-name: other handler**

**service: name=other state=restarted**

**Use it as this:**

**-name: Change default Apache virtual host**

**template: src=vhost.tpldest=/etc/apache2/sites-available/000-default.conf**

**notify: restart apache**

Note: Static variables will be declared in vars directory.

Types of variables:

1. Static

Global are facters Facts: global variables containing information about the system, like network interfaces or operating system

1. default
2. group\_vars
3. hosts\_vars
4. extra\_vars

extra\_vars🡪hosts\_vars🡪group\_vars🡪default

Extra\_vars has to be defined dynamically during runtime unlike group and hosts vars are explicitly defined in playbook yaml file.

Extra\_vars cannot be taken directly but should be passed as parameter during runtime.If it is single key value pair can be passed as below:

**ansible-playbook -i hosts site.yml -e “group\_name=webserver”** So here it will override group\_vars and hosts\_vars and directly it will pick defined value in runtime.

If you have hundreds of parameters then create a yaml file and pass it like below.

**ansible-playbook -i hosts site.yml -e @abc.yaml**

**Ansible Vault:**

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. These vault files can then be distributed or placed in source control. Ansible Vault can encrypt any structured data file used by Ansible.

To enable this feature, a command line tool - [**ansible-vault**](https://docs.ansible.com/ansible/latest/cli/ansible-vault.html#ansible-vault) - is used to edit files, and a command line flag ([**--ask-vault-pass**](https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#cmdoption-ansible-playbook-ask-vault-pass)**,**[**--vault-password-file**](https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#cmdoption-ansible-playbook-vault-password-file)**or**[**--vault-id**](https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#cmdoption-ansible-playbook-vault-id)) is used. Alternately, you may specify the location of a password file or command Ansible to always prompt for the password in your ansible.cfg file. These options require no command line flag usage.

Ansible AWX:

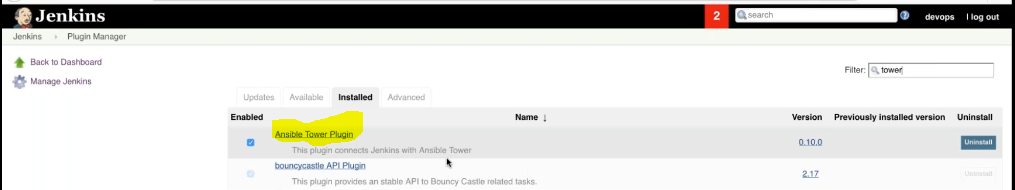
Refer below URL to setup AWX:

<https://computingforgeeks.com/how-to-install-ansible-awx-on-ubuntu-linux/>

Ansible-playbook can be run through jenkins. But build history can be 10 or 15 maximum.

But tower can make up all the history but it is only for ansible-playbook commands not for anyother but jenkins can support multiple platforms.

Jenkins can also run AWX tower jobs uning Ansible tower plugin in jenkins. Means jenkins can run even ansible tower jobs also.



We no need to create jenkins agent and install awx tower instead we have separate plugin using which we can directly integrate ansible awx tower and run the tower jobs.

Jenkins will invoke the template job in tower using rest api just like a webhook.

Jenkins 🡪ansible-playbook through ansible plugin /agent must be ansible host

tower 🡪ansible-playbook

**Best recommended way of CI/CD pipeline is as below:**

Jenkins🡪tower🡪ansible-playbook through ansible tower plugin