Topics to cover :

Core concepts :

1. Ansible inventory
2. ad-Hoc commands
3. Tasks, plays and playbooks
4. Ansible configure files

Working with common modules

Ansible collection and galaxy

Using variables .

Troubleshooting , conditionals , privilege escalation …

Dynamic inventory for eC2

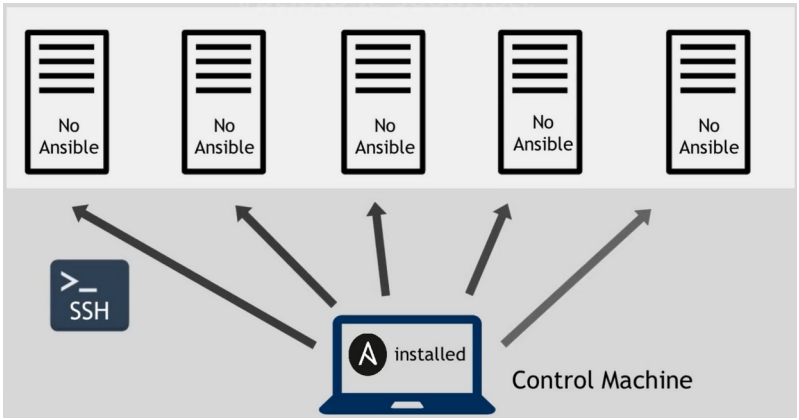
Ansible roles ,
Ansible with docker , kubernetes, terraform , Jenkins pipeline ,AWS . ..

Let’s start !!!

Help to automate tasks ->
Eg: when u multiple servers -> where your distributed application is running u need to deploy your application on all those servers . Upgrade all those eversion on all those servers . Or u want to do repetitive task on all those versions .
Tasks like update , back up , weekly system reboots , creating users assigning permissions to those users .
So u will have to ssh to each server one by one and do the same configuration on all of those servers.
Every time this task need to do again u will need to start from scratch and u will need to do those whole process again and again .
Specially if it has multiple steps u will need to remember those steps each time .

1. U can execute all these tasks to all the machine remote instead on manual .
2. Instead of doing all the tasks manually like scripting ,using command line u all list all the task on the single yaml file .
3. U can reuse the same the same file in different servers
4. Human can make mistake on configuration the complex IT tasks ansible makes it more reliable and less likely for error .

What every It tasks like os update , cloud provisioning u can do all of this using ansible .

Ansible is agent less : 

Usually when u want to use a tool on a machine u need to go to that machine and install a agent for that machine . But for ansible u don’t need to install ansible on the target folder . U can use your own Mahican and control all the servers remotely .

This make it easy as no deployment effort in beginning , no upgrade efforts.

Ansible architecture :

Modules :
Ansible works with modules these are small programming that do the work they are sent from the control machine to the target server .
They do the job like -> install the application , configuration etc. and when they are done get removed .

Modules are very granular => one small specific task , eg: creating or copy a file or installing nginx server , create a cloud instance , install docker container , start docker etc.

```yaml
# Create a jenkins job using basic authentication
- jenkins_job:
    config: "{{ lookup('file', 'templates/test.xml') }}"
    name: test
    password: admin
    url: http://localhost:8080
    user: admin

# Delete a jenkins job using the token
- jenkins_job:
    name: test
    token: asdfasfasfasdfasdfadfasfasdfasdfc
    state: absent
    url: http://localhost:8080
    user: admin
```

And ansible has 100’s of modules that perform these specific task

Module Index

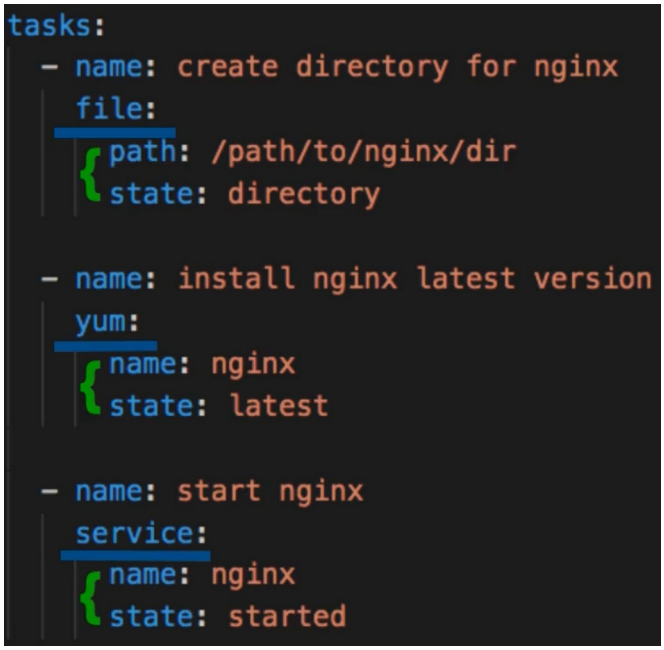
- All modules
- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Net Tools modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules

Ansible uses simple yaml language -> no need to lean another language .
Eg of modules : ->

```yaml
- name: Create a data container
  docker_container:
    name: mydata
    image: busybox
    volumes:
      - /data

- name: Start a container with a command
  docker_container:
    name: sleepy
    image: ubuntu:14.04
    command: ["sleep", "infinity"]

- name: Add container to networks
  docker_container:
    name: sleepy
    networks:
      - name: TestingNet
        ipv4_address: 172.1.1.18
        links:
          - sleeper
      - name: TestingNet2
        ipv4_address: 172.1.10.20
```

How to handle complex application -> group together module in a sequence . That where ansible playbook ->

```yaml
tasks:
  - name: Rename table foo to bar
    postgresql_table:
      table: foo
      rename: bar

  - name: Set owner to someuser
    postgresql_table:
      name: foo
      owner: someuser

  - name: Truncate table foo
    postgresql_table:
      name: foo
      truncate: yes
```

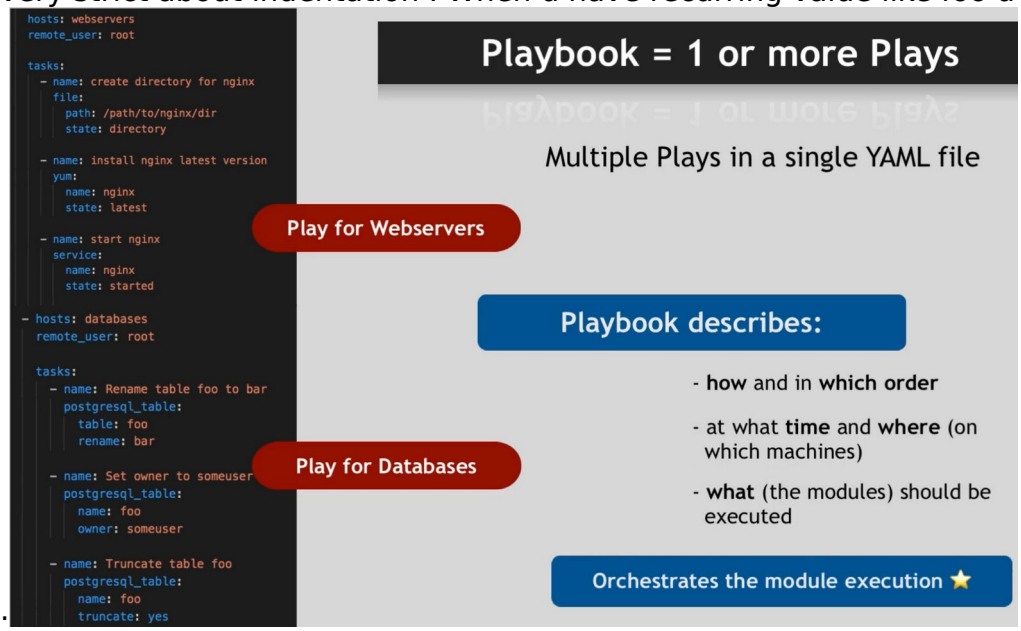

tasks:
- name: create directory for nginx
file:
{path: /path/to/nginx/dir
state: directory
- name: install nginx latest version
yum:
{name: nginx
state: latest
- name: start nginx
service:
{name: nginx
state: started

These are the task that we need to execute -> but where to execute these tasks.

```yaml
- hosts: databases
  remote_user: root

  tasks:
    - name: Rename table foo to bar
      postgresql_table:
        table: foo
        rename: bar

    - name: Set owner to someuser
      postgresql_table:
        name: foo
        owner: someuser

    - name: Truncate table foo
      postgresql_table:
        name: foo
        truncate: yes
```

```yaml
- hosts: databases
  remote_user: root
  vars:
    tablename: foo
    tableowner: someuser

  tasks:
    - name: Rename table {{ tablename }} to bar
      postgresql_table:
        table: {{ tablename }}
        rename: bar

    - name: Set owner to someuser
      postgresql_table:
        name: {{ tablename }}
        owner: someuser

    - name: Truncate table
      postgresql_table:
        name: {{ tablename }}
        truncate: yes
```

Yaml language is very strict about indentation . When u have recurring value like foo u can use variables.



Good practice is to name each of your plays

```yaml
- name: install and start nginx server
  hosts: webservers
  remote_user: root

  tasks:
    - name: create directory for nginx
      file:
        path: /path/to/nginx/dir
        state: directory

    - name: install nginx latest version
      yum:
        name: nginx
        state: latest

    - name: start nginx
      service:
        name: nginx
        state: started
```

```yaml
- name: rename table, set owner and truncate it
- hosts: databases
  remote_user: root
  vars:
    tablename: foo
    tableowner: someuser

  tasks:
    - name: Rename table {{ tablename }} to bar
      postgresql_table:
        table: {{ tablename }}
        rename: bar

    - name: Set owner to someuser
      postgresql_table:
        name: {{ tablename }}
        owner: someuser

    - name: Truncate table
      postgresql_table:
        name: {{ tablename }}
        truncate: yes
```

Where are all the hosts details are listed ->

In the inventory file of ansible -> u can put ip address as well as the host name there . ->

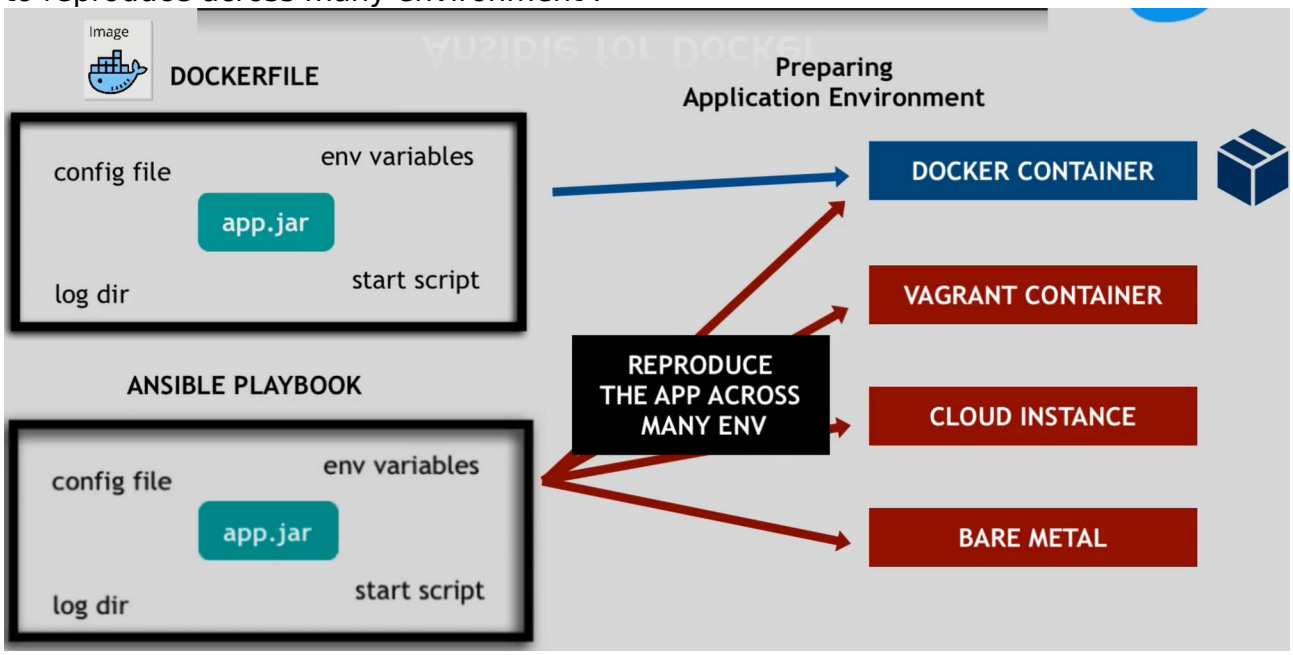
```
[webservers]
web1.myserver.com
web2.myserver.com
```

Ansible for docker _>

Usually when u want to create docker container u want to create a docker file.
Which prepares the application environment -> so in the docker file u have a jar.file , configuration file , Log dir , env variable , start scripts .

All this prepares the environment of the application start up . And with docker file which will create a docker container with application and its prepared environment .

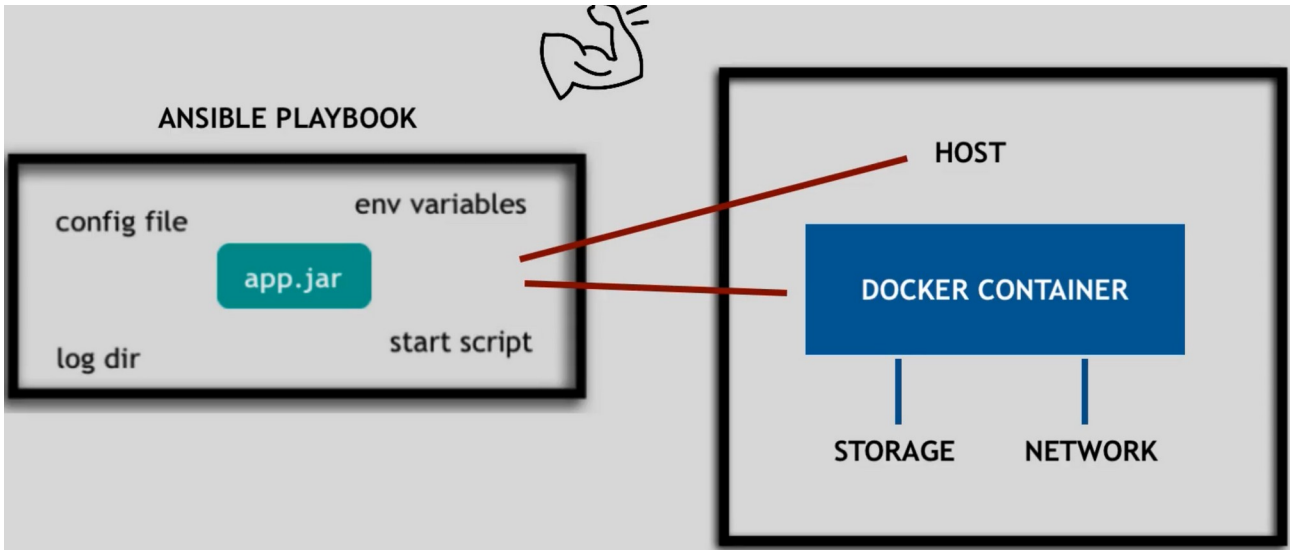
With ansible u can create a alternative to docker file which is more powerful.
With all the same setup in the ansible play book -> u can create a docker container , vagrant container , cloud instance , Bare metal server . Ansible allows your application to reproduce across many environment .



U can not only manage the docker container but also the host that u are running the container .

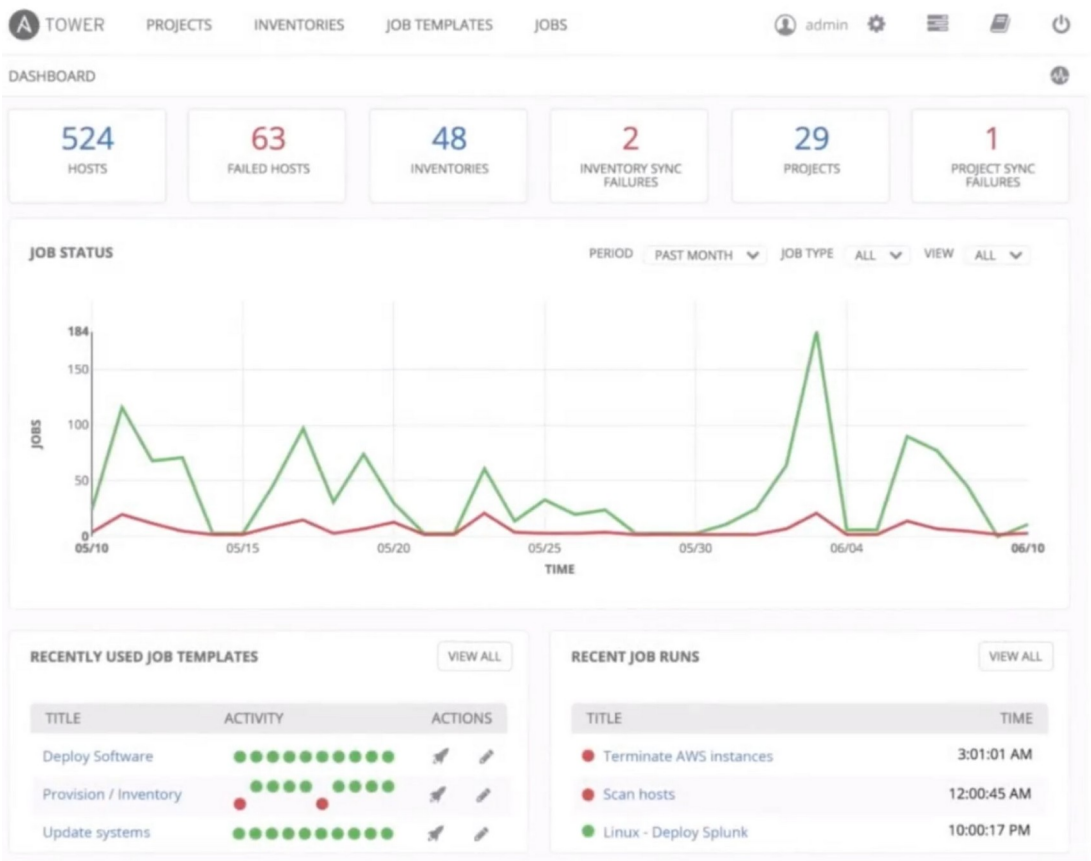
```
[droplet]
134.209.255.142
134.209.235.158

[droplet:vars]
ansible_ssh_private_key_file=~/.ssh/id_rsa
ansible_user=root
~
```

Eg: if the container has the dependency on storage or network of the host . Using ansible u can mange all these which makes it more power full as u can mange the tasks on both on the host and the container level .



Ansible tower :

It is a UI dashboard from red hat ->
: u can centrally store all the automation task over all the team , configure permissions etc. manage inventory and check the status of jobs .



Alternative tools of ansible :

Ansible uses -simple yaml where as puppet or chef uses -Ruby which is more difficult .
Ansible is agentess meaning instead of installing ansible on every system u can manage them remotely where as for other tools like puppet and chef , u need to install , manage ,update on target server .

Installing ansible :

In order to run ansible we need to install in our local machine .
Windows is not supported to use as a control node / where ansible control all the target server.

1. U can install ansible on your local machine
2. U can have a separate remote server where u install ansible .

On Mac installing ansible +> brew install ansible
Ansible is written in python that means we need to install python also .
Knowing python will make it easier to work with ansible for some custom functionality .

Using python package manger u can also install ansible : pip install python .

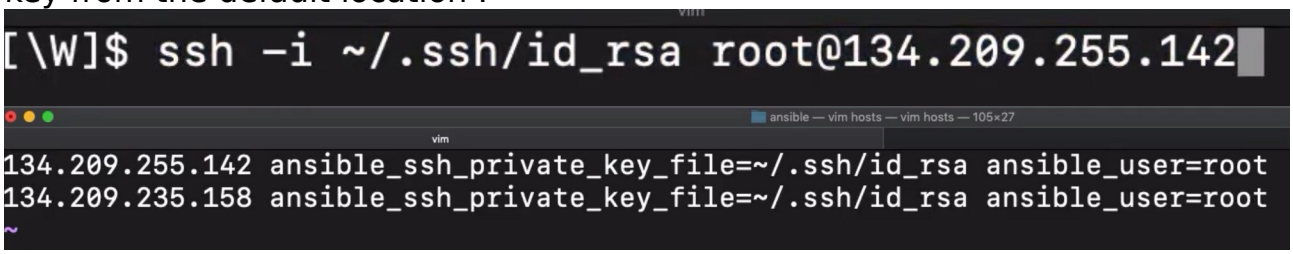
:ansible —version // to check version
: ansible // to check if the ansible is installed or not .
1. Create a 2 linux target servers .

For linux target servers there has to be python installed in order to execute ansible .
As ansible is written in my thong so it needs python interpreter to configure the server.
If it is a window target server when power shell has to be installed in the window server .

2. Login via a ssh@publicIpAddressOfTargetServer
3. In target server : ls /usr/bin/python3 // these is python file installed in the target server .
Python3 is already installed in the target droplet server (digitOcean cloud provider same as AWS etc cloud providers).
Connecting ansible to the remote server . =>
1. First we need to tell ansible what are the target servers it has to manage -> so for that ansible maintain a host file (ansible inventory file) -> this file contains data about the ansible client servers , hosts means the managed server , default location of this file : /etc/ansible/hosts

: vim hosts -> store the ip address of the target sever that ansible will manage .
It also needs the credentials to connect to these servers .
We ssh to target server using private ssh key .
U can use It via a attribute : ansible_ssh_private_key_file = ~/.ssh/id_rsa.
Above is the location of the private key which we can use to connect to the target server .

U can directly log in via ssh root@ipaddressOfRemotetargetServer by default if u will not specify the lpcaiont of private key then it will befault take the private key from the default location .



Ansible ad-hoc commands :

Ad-hoc commands are not used for future uses
A fast way to interact with a desired server .

```
$ ansible [pattern] -m [module] -a "[module options]"
```

Pattern = targeting hosts or groups
-all default group which contains all hosts .

Eg: ansible all -I hosts -m ping

// from this hosts file we want to interact with all the servers on the inventory - m(module) ping the servers . There is a ansible module called ping.

```
[\W]$ ansible all -i hosts -m ping
The authenticity of host '134.209.235.158 (134.209.235.158)' can't be established.
ECDSA key fingerprint is SHA256:mYngU8pL6PBIWPi5g4W1lw0oMakxQpaGxnzGKfXqi7I.
Are you sure you want to continue connecting (yes/no)? 134.209.255.142 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
yes
```

NOTE : now We have connect to those server via

this inventory file (hosts file) =->

Eg: if there are servers present in the multiple cloud platform we need to group these servers together -> how >?

By grouping hosts : -> you can put each host in more then one group
you can create groups that tracks : =>
WHERE = a datacenter/region , eg: east/west
, WHAT = eg: database /webserver etc,
WHEN = what stage , eg: prod , dev , test environment .

```
[droplet]
134.209.255.142 ansible_ssh_private_key_file=~/.ssh/id_rsa ansible_user=root
134.209.235.158 ansible_ssh_private_key_file=~/.ssh/id_rsa ansible_user=root

[aws]
~
```

Here we can see we are login to the specific server using there group name or provide the post name of the server->

```
ansible droplet -i hosts -m ping
```

```
ansible 134.209.235.158 -i hosts -m ping
```

To have a cleaner host file . ==> we can use the variable.

ADD EC2 instance to the inventory :

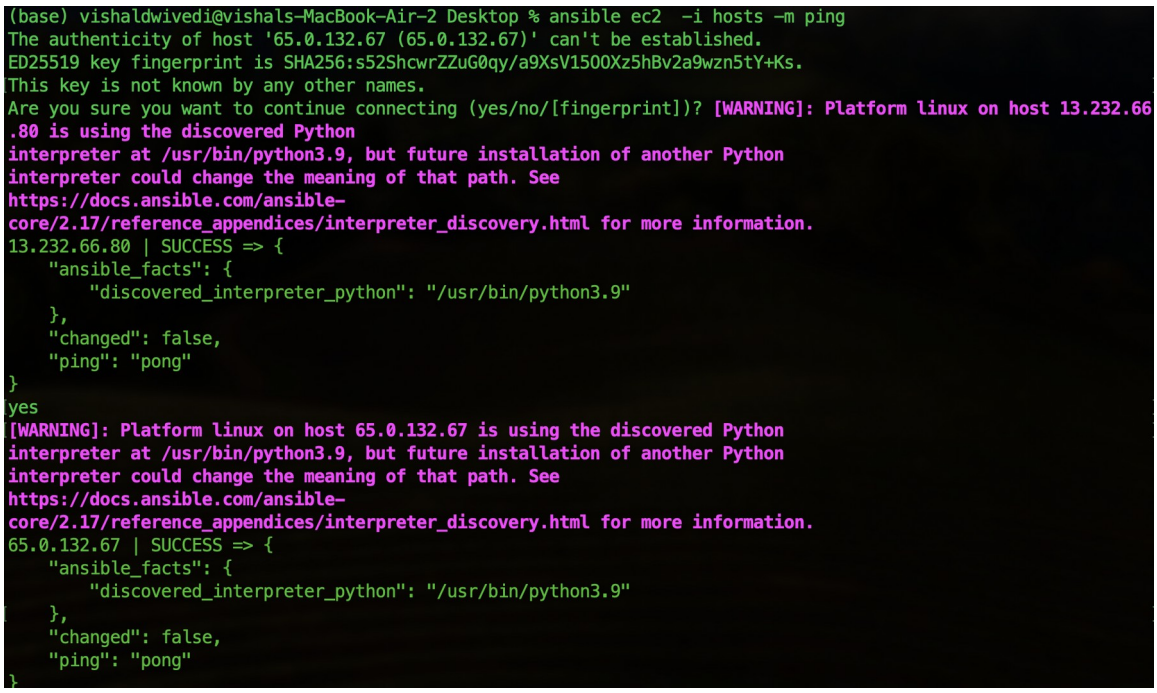
Till now we have created a digital ocean drop let same as aws instance and connect to both of them using ansible we gonna do same using AWS ec2 instances .

1. Create 2 ec2 instances in AWs
2. Now we can configure these two servers in the ansible host file .
3. In the host file in your pc u can also add the DNS name not only the ip address of the instance .
4. chmod 400 admin_key_pair.pem
5. Now we r in one of the server we created , in Linux server ansible need python need to be installed .

sudo dnf install python3 -y

6.

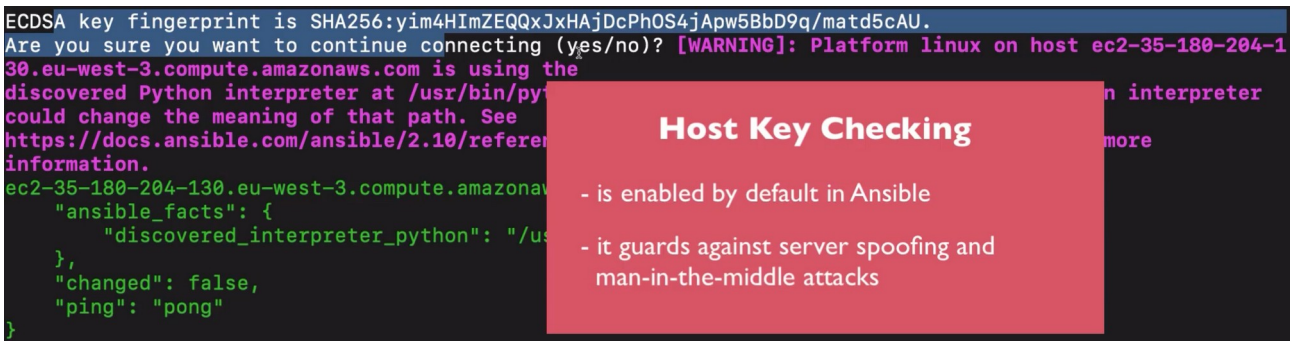
```
[ec2]
ip-172-31-45-27.ap-south-1.compute.internal
ip-172-31-47-189.ap-south-1.compute.internal

[ec2:vars]
ansible_ssh_private_key_file=~/Downloads/admin_key_pair.pem
ansible_user=ec2-user
~
```




```
[ec2]
ec2-15-188-239-5.eu-west-3.compute.amazonaws.com
ec2-35-180-204-130.eu-west-3.compute.amazonaws.com ansible_python_interpreter=/usr/bin/python3

[ec2:vars]
ansible_ssh_private_key_file=~/Downloads/ansible.pem
ansible_user=ec2-user
```

Above we can set the ansible python interpreter to python3 to remove the pink warning .

MANAGING HOSTS KEY CHECKING :



But we want to have this manually .

Authorised key and know hosts :

When we ssh from out machine to the target server both servers need to identify and authenticate each other some way so the server must allow out machine to connect to it as well as our machine out machine should recognise this target server as a valid host of this to happen the server which is ssh ing into the target server must add this server information into file called known hosts and that file is located inside the .ssh folder

: ~/.ssh/known_hosts = it has all the servers that I have connected to when I typed yes and entry has made here .
So my machine has to add the target server inside this know_hosts file . -> command of that is =>:
: ssh-keyscan -H PublicIpAddressOfTargetServer >> ~/.ssh/known_hosts

Now the next step is for the target remote serve to authenticate us for that the target server has to have the public ssh key -> cat ~/.ssh/id_rsa.pub of the connect machine inside it .

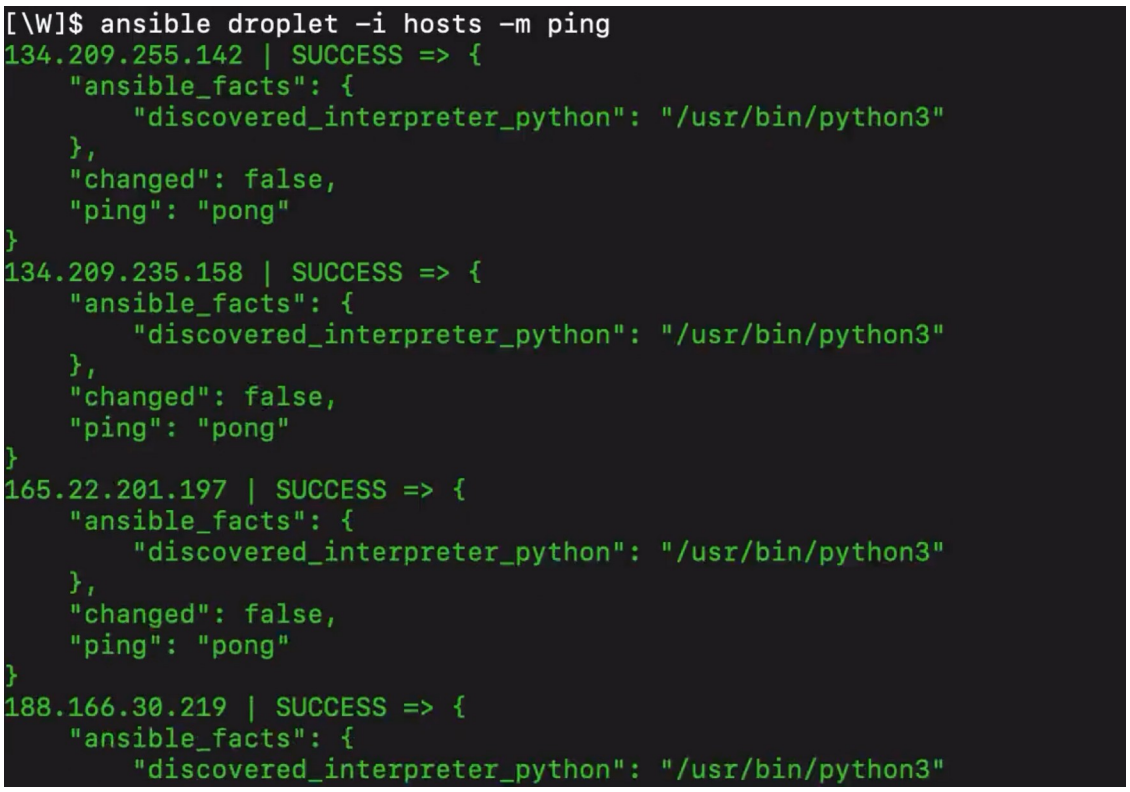
: we can copy our public ssh key to the target using this command :
: ssh-copy-id root@TargetIpAddresss
This will copy the my public key to the : .ssh/authorized_keys of the target server

This u can do with the log time servers . So u will do all this once and u will run ansible command and play books on the target servers .

```
[droplet]
134.209.255.142
134.209.235.158
165.22.201.197
188.166.30.219

[droplet:vars]
ansible_ssh_private_key_file=~/.ssh/id_rsa
ansible_user=root
```

Modules

- Also referred to as "task plugins"
- Ansible executes a module usually on remote server
- And collects return values



Disable the host key checking :

Above was the one way to configuring the ssh key check .

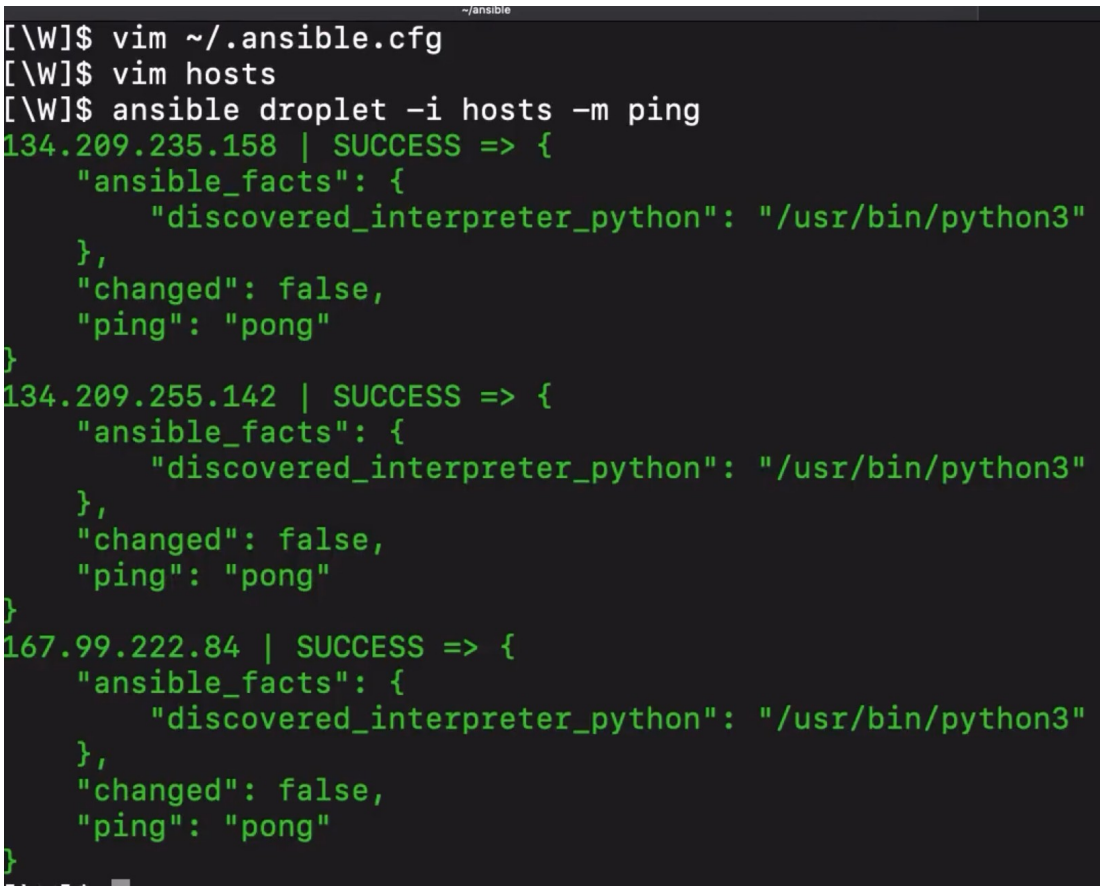
Second way is to disable this ssh key check . This is less secure.

How ever If u have a EPHEMERAL INFRASTRUCTURE - servers are dynamically created and destroyed .
For these type of servers its will be very time waste to run these all commands all the time and set the ssh key check

We gonna disable host key checking in the ansible config file . ==>


```
[defaults]
host_key_checking = False
```

```
vim ~/.ansible.cfg
```



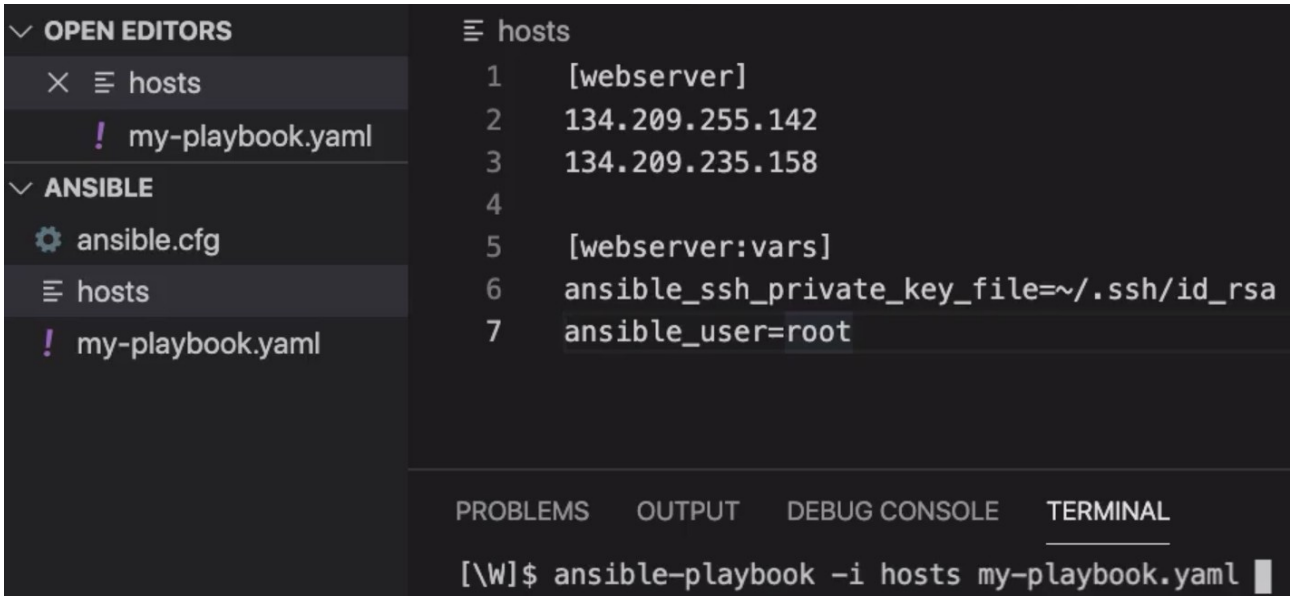
ANSIBLE PALYBOOKS :

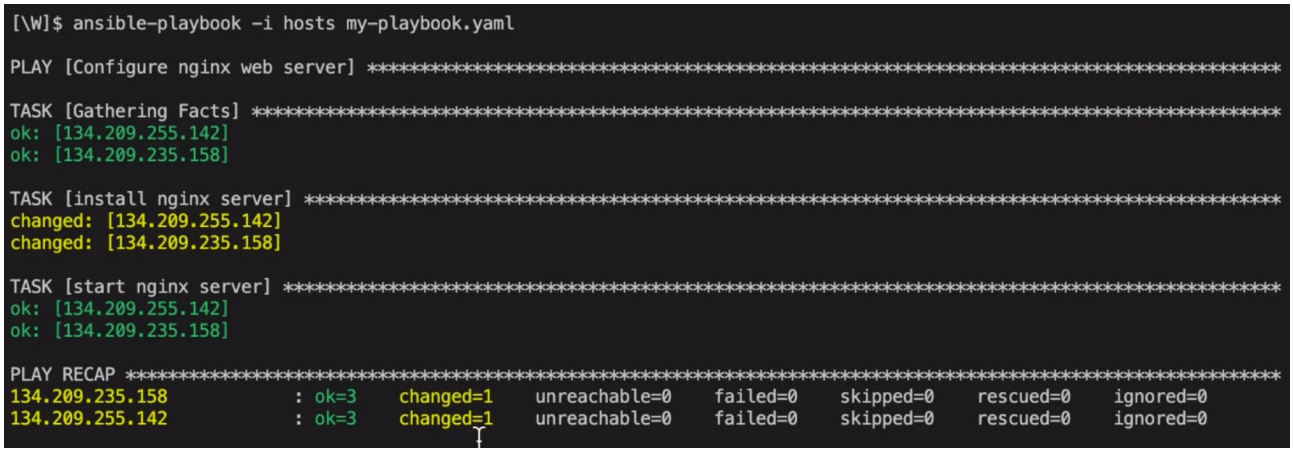
With ansible we just don’t want to ping the servers but also configure them . Since ansible is a infrastructure as a code tool , ansible configure file is treated as a code , create ansible projects ,
And create a git repo for this ansible project same as we did for the terraform project .

1. Create a hosts file
2. Create a my-play-book.yaml file ->

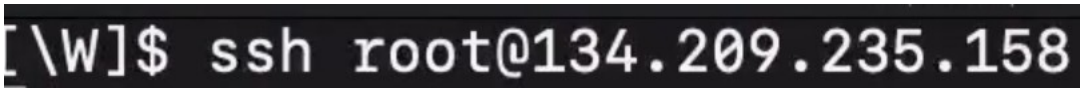
Play-book = ordering list of task , play and tasks run s in order from top to bottom , written in yaml , play book in ansible can have multiple plays , play is a group of task that u want to execute on several servers. Eg: 1 play for all database server , 1 play for all web servers and we will have both of these plays in one play book .

3.



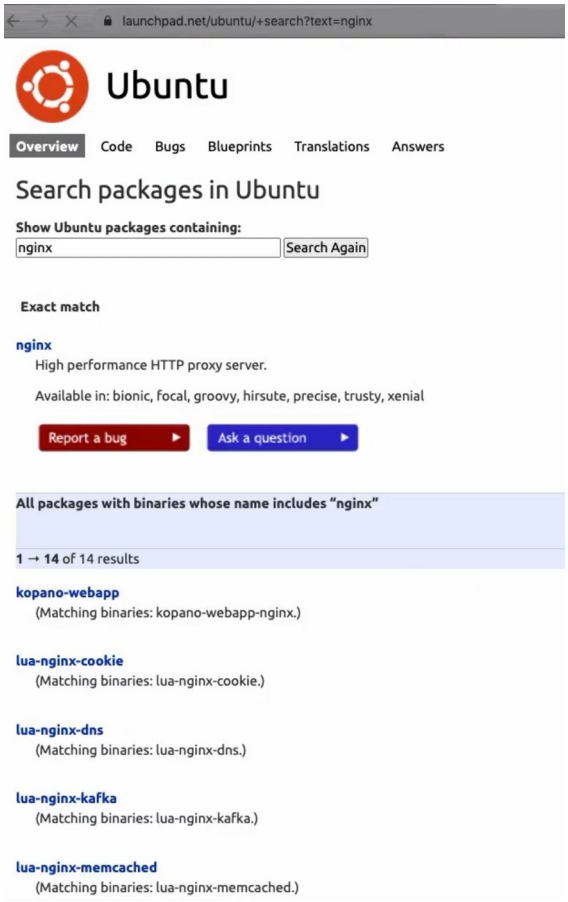


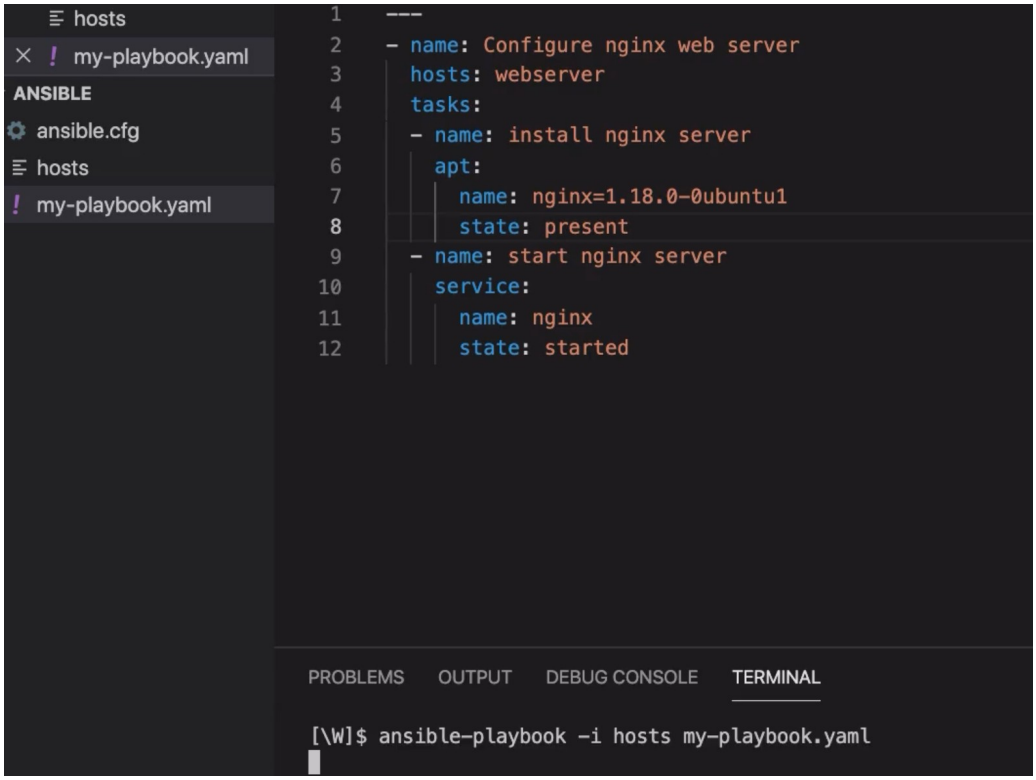
3. Checking if the nginx is installed inner server my login to those server ->



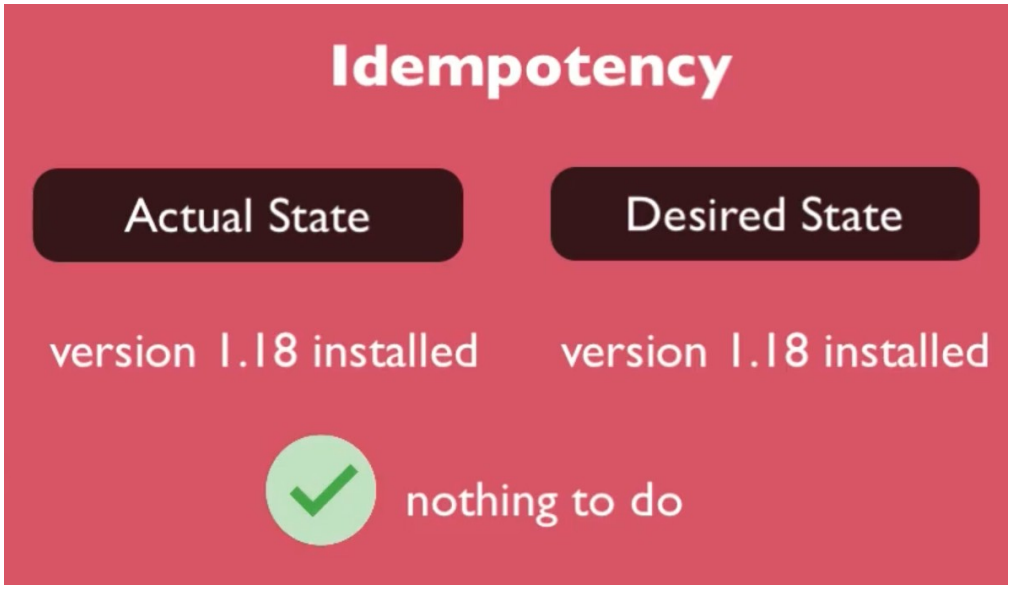
```
root@ubuntu-s-1vcpu-2gb-intel-fra1-01:~# ps aux | grep nginx
root        4844  0.0  0.0  57320  1516 ?        Ss   16:49   0:00 nginx: master process /usr/sbin/nginx
-g daemon on; master_process on;
www-data    4845  0.0  0.2  57876  5356 ?        S    16:49   0:00 nginx: worker process
root       16331  0.0  0.0   8160   736 pts/0    S+   16:53   0:00 grep --color=auto nginx
root@ubuntu-s-1vcpu-2gb-intel-fra1-01:~# nginx -v
nginx version: nginx/1.18.0 (Ubuntu)
root@ubuntu-s-1vcpu-2gb-intel-fra1-01:~#
```

Now how we can install specific version of nginx . —>
U can visit -> here => and see the version u need . ->

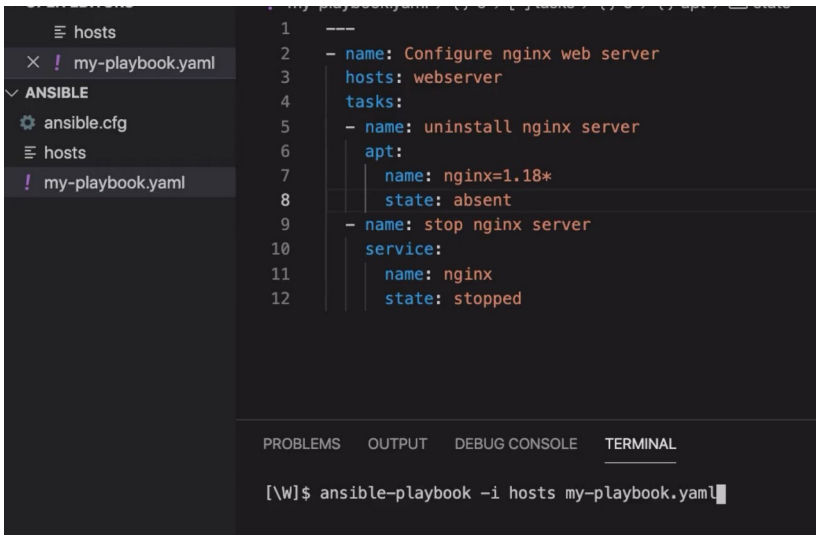



hosts
my-playbook.yaml
ANSIBLE
ansible.cfg
hosts
my-playbook.yaml
1 ---
2 - name: Configure nginx web server
3 hosts: webserver
4 tasks:
5 - name: install nginx server
6 apt:
7 name: nginx=1.18.0-0ubuntu1
8 state: present
9 - name: start nginx server
10 service:
11 name: nginx
12 state: started
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
[\W]$ ansible-playbook -i hosts my-playbook.yaml

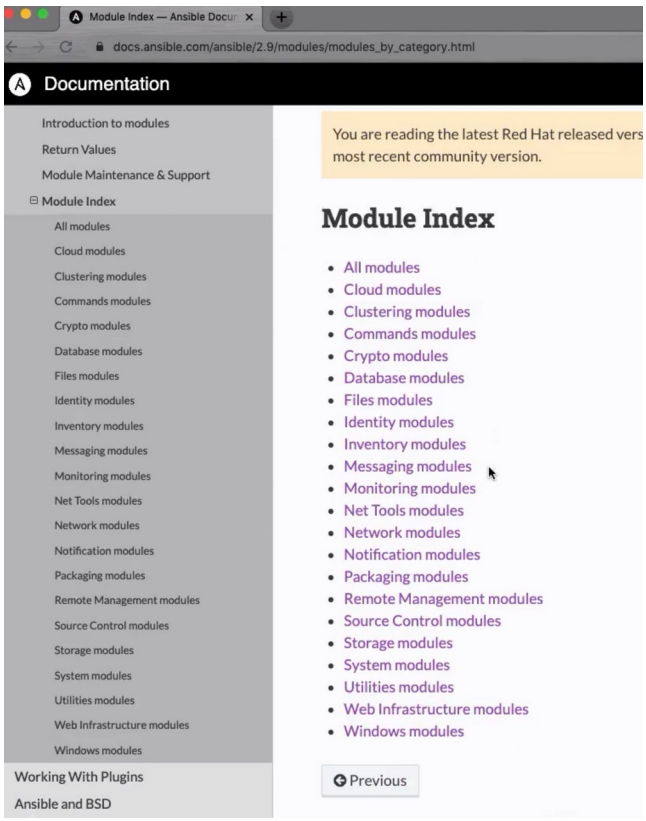
ANSIBLE IDEMPODENCY >



How to uninstall the packages now :



ANSIBLE MODULES :



Visit ansible documentation’s ->

Modules is a reusable piece of code that u can reuse in our play book .

Instead of writing the code your self u can just use existing one

Packing modules is one of the most important module in ansible.

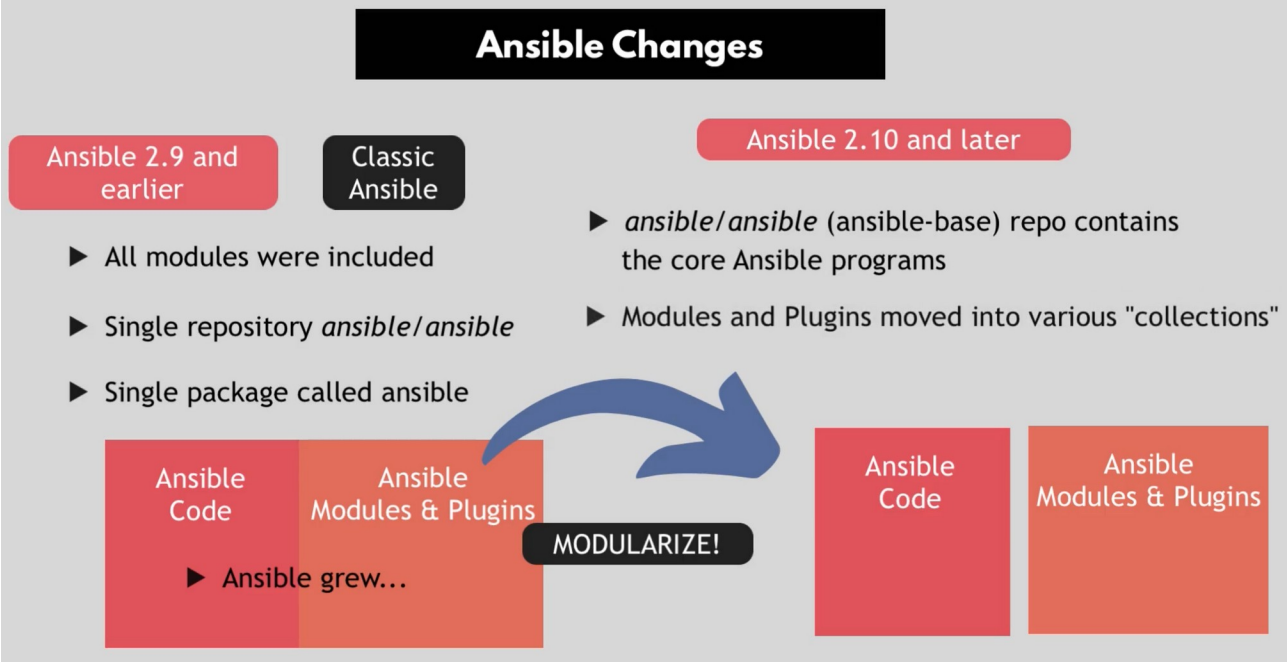
MAJOR CHANGES IN ANSIBLE DISTRIBUTION :

Modules index in the version 2.9 the above list of ansible documentation . But in the latest version in documentation there is not module index there is a collection index.

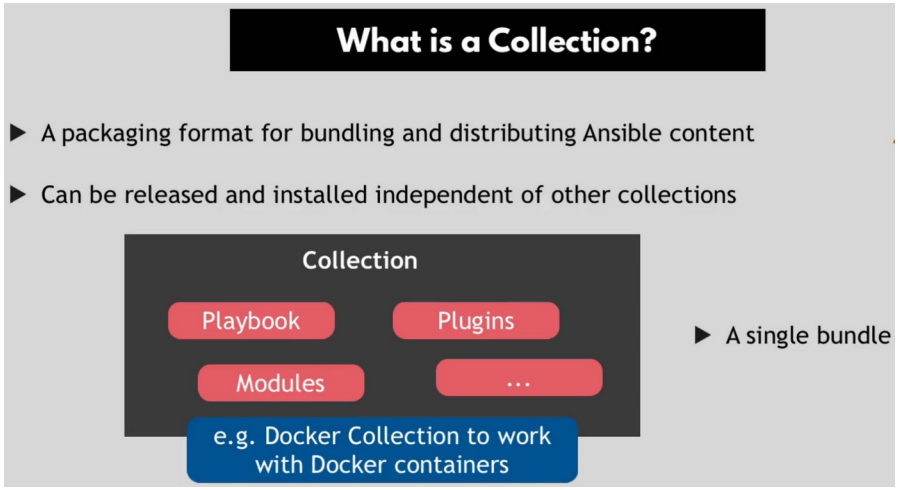
What happened is in the earlier version 2.9 modules are packed in the ansible distribution itself .
For version 2.9 and earlier there was single ansible/ansible repo and single package called ansible.
And when ansible grew in size more contributes grew and new features and modules are added they modularise ansible meaning modules and plugins

moved into various collections from ansible 2.10 and later where ansible/ansible (ansile-base) repo contains the core ansible programs .

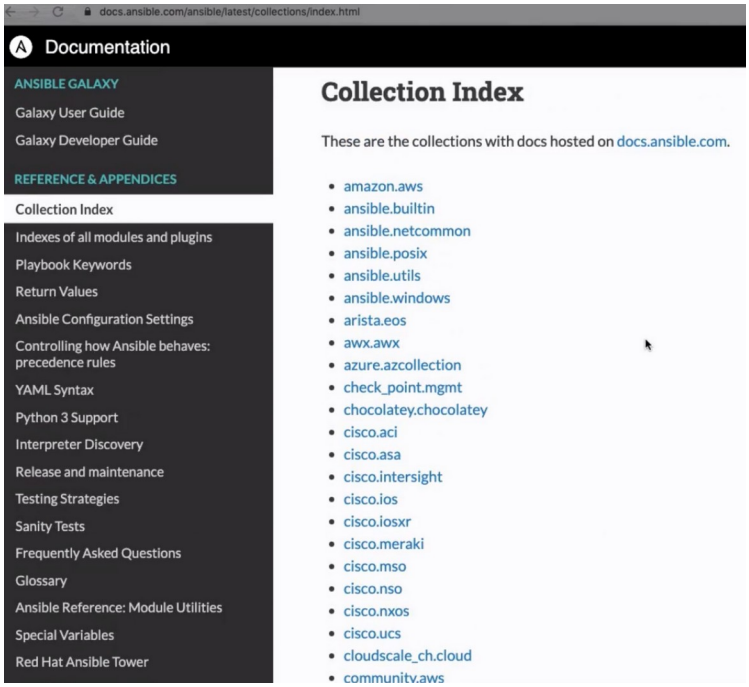
So now ansible base (core) contains all the binary to able to execute ansible and ansible package that contains those modules and plugins that u can a part of ansible in 2.9 thats why when u install ansible u get these two packages install 1. Ansible - base package 2. Ansible package



WHAT IS COLLECTION :
What is the difference btw modules and collection ?
And why we have a collection index in latest ansible version documentation instead of module index .



Now every thing in ansible is part of a collection .



Ansible plugins piece of code that add to ansible functionality or modules.
U can also write your own plugins.

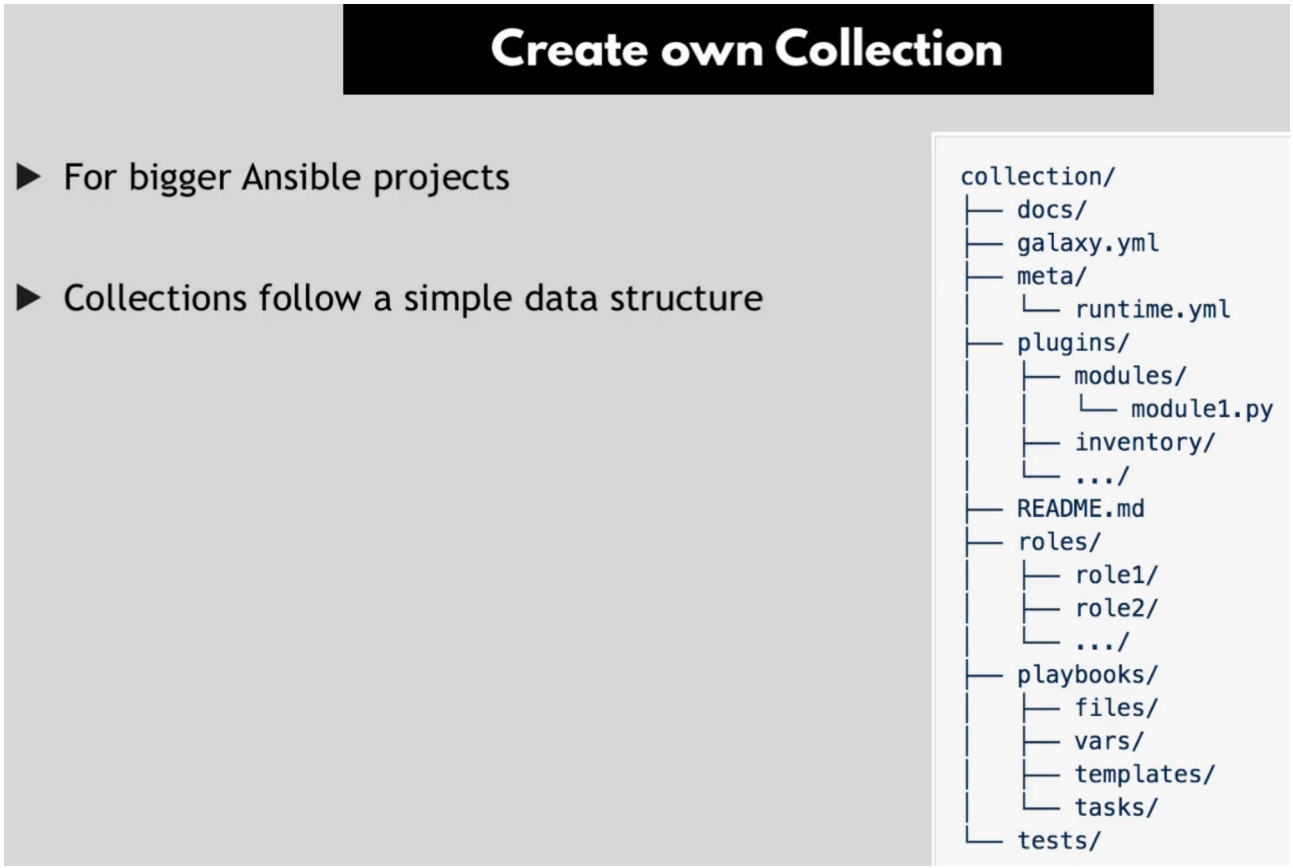
To see the ansible collection list that are available locally in your machine : in terminal use command -> ansible-galaxy collection list

ANSIBLE GALAXY :

Collection group ansible content like modules and plugins are just piece of code so they have to be listed somewhere so that u can download it locally on your machine .
where do collection live ? -> ansible galaxy .

Just like u have repository of npm package , terraform modules etc this is similar thing but for ansible collections .

Also ansible galaxy is a command-line-tool to install individual collections . Eg: : ansible-galaxy install <collection name>



PROJECT 1 .
Automate node app development :

1. Automate node js application on a digital ocean server
2. Create a droplet
3. Write a ansible-playbook that will install node and npm on server .
4. Copy node artifact and unpack and start application .
5. And finally we will verify if our application is successfully running.

Below we are configuring the hosts file in ansible by passing the public ip address of rem>

hosts

```
1   159.89.1.54 ansible_ssh_private_key_file=~/.ssh/id_rsa ansible_user=root
```

Below are the steps to setup all this ->

deploy-node.yaml

```yaml
---
- name: install node and npm
  hosts:
  tasks:
    - name: update apt repo and cache
    apt: update_cache=yes force_apt_get=yes cache_valid_time=3600
    - name: Install nodejs and npm
      apt:
        pkg:
          - nodejs
          - npm
```

```yaml
- name: Deploy nodejs app
  hosts:
  tasks:
  #1) copy tar file location 2) unpack tar file
   # - name: Copy nodejs folder to a server
    # copy:
    #    src: # file in the local machine u want to copy
     #   dest: #/root/newFileName
    - name: Unpack the nodejs file
      unarchive: # unarchine by default takes source from local machine so we dont need the above first step and this  remote_src: yes
        src: # file in the local machine u want to copy
        dest: #/root/
          #tar file will be unpack as a package folder
       #  remote_src: yes
        # this src has to be present in the remote server
        #if we dont specify this it will start looking on the tar file on our local machine .

        # to execute this use command:
        # ansible-playbook -i hosts deploy-node.yaml
        # Step 2 start this application.
        # what we have to do to run this applicaiton
        # install app dependencies listed on package.json we dont have a node modules in aroud unpacked file
```

```yaml
- name: Install dependencies
  npm:
    path: #path of packagejson inside the remot server
- name: start the application
  command: #node pathToTheServerJsFile on remote server
    chdir: # pathToTheServerJsFile on remoteserver
    cmd: node server
  async: 1000 # this will help even if the ansilbe play book execution is completed server should run in background
  poll: 0
```

```yaml
    - name: Ensure app is running
      shell: ps aux | grep node
      register: app_status #storing the message in this variable
    - debug: msg={{app_status.stdout_lines}} # using the varaible and print the mssg
# commands and shell modules are not statefull not idempotent .
```

Note: commands and shell modules are not stateful and not idempotent

EXECUTING tasks with different user .>

Currently we are -> running everything as a root user .

But previously we know that as a security best practices we should not execute this as a root user .
U should be creating the own user for each team member and u should than start the application with non root user the main reason is that root user has all the privileges so when u start the application as a root user and some one hacks into your application they will be able to to much damage as they will have the access of the root user privileges and permissions on that server so if we want to limit that security risk we want a user with way less permissions and use that user instead , so every application will have its own user to do that we need to create a new user in out remote server and

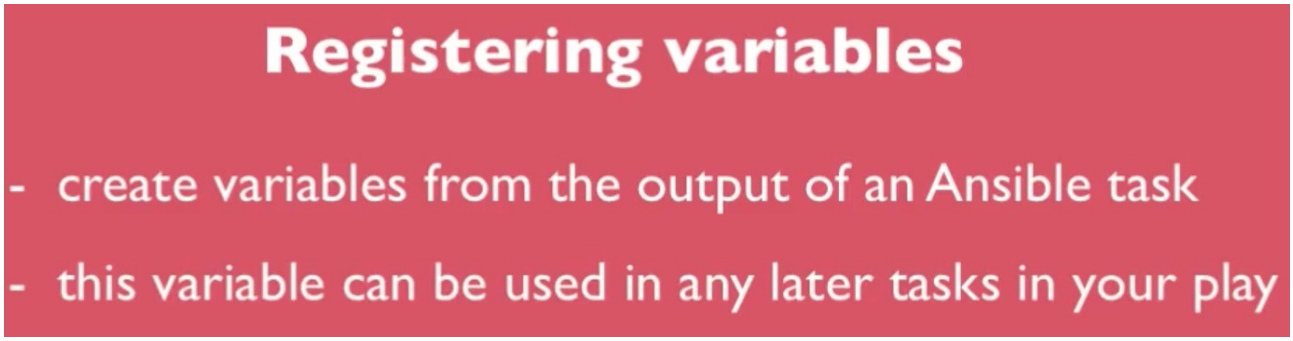
Now we will create a new play in deploy-node.yaml file to create a new linux user inside our remote server

```
- name: create new linux user for node_app
 hosts:
 tasks:
   - name: Create linux user
     user: vishalnodeapp
     comment: vishalAnsible
     group: admin  #below change all the destination , path , chdir attribute to /home/vishalnodeapp

- name: Deploy nodejs app
```

```
hosts:
become: true # allow you to become another user .
become_user: #vishalnodeapp #set the user with desired privilages , default is root .
tasks:
#1) copy tar file location 2) unpack tar file
```

VARIABLE IN ANSIBLE :

RESIGTERED VARIABLE :




```
- name: Ensure app is running
    shell: ps aux | grep node
    register: app_status #storing the message in this variable
  - debug: msg={{app_status.stdout_lines}} # using the varaible and print the mssg
# commands and shell modules are not statefull not idempotent .
```

Eg: 2

```
- name: create new linux user for node_app
 hosts:
 tasks:
  - name: Create linux user
    user: vishalnodeapp
    comment: vishalAnsible
    group: admin  #below change all the destination , path , chdir attribute to /home/vishalnodeapp
    registor: user_creation_result
  - debug: msg ={{user_creation_result.uid}}
```

a