

Cài đặt: ansible for ubuntu 20.04

apt update -y

apt install software-properties-common -y

apt-add-repository --yes --update ppa:ansible/ansible

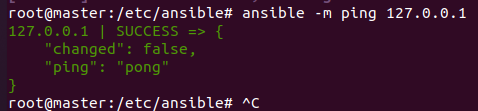
apt install ansible -y

nano /etc/ansible/host

thêm: ansible quản lý domain local

[local]

local\_server ansible\_host=127.0.0.1 ansible-connection=local



Oke tạm thời đã cài đặt thành công

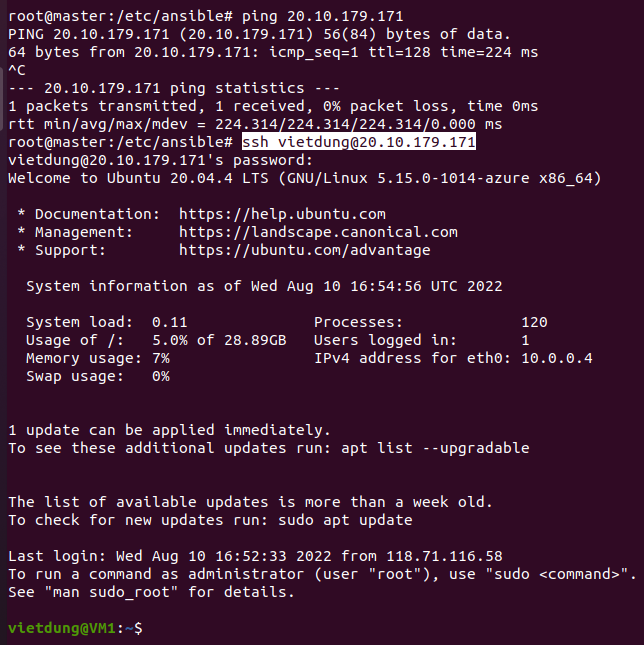
# Xác thực, ssh server

Ta có 1 con VM tạo từ azure có IP public là 20.10.179.171

Ping là đang thông

Cách 1: từ máy local ssh tới VM bằng user/pass

ssh $user@IP

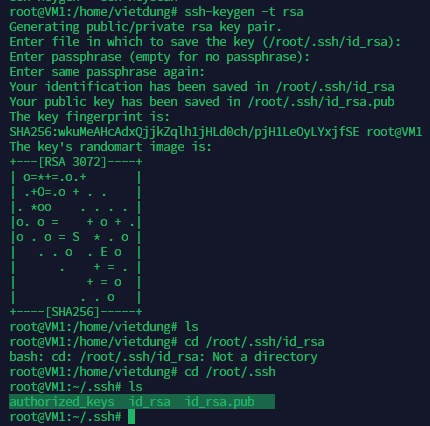


Cách 2: Sử dụng ssh-key (cách 2 đang lỗi)

* Có passphare
* Không có passphare

Vào con VM để gen key trước

ssh-keygen -t rsa



Bgio lại có 2 cách để copy key này sang con local

Vì ko thể copy từ con VM tới con local được

Nên ta tạo 1 VM2 coi như làm server

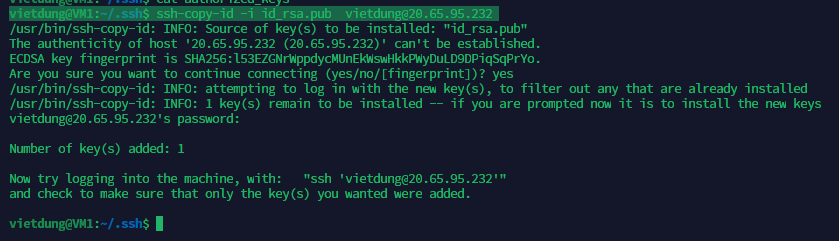
Tạo key từ VM1 và add public key từ VM1 tới server (VM2) cho việc authen

* VM1, gen-key
* Copy publickey tới VM2

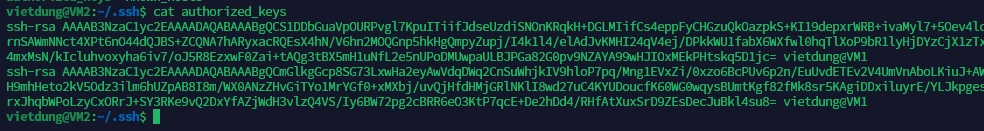
ssh-copy-id -i id\_rsa.pub [vietdung@20.65.95.232](mailto:vietdung@20.65.95.232)

[vietdung@20.65.95.232](mailto:vietdung@20.65.95.232) : là user/IP của server

ý nghĩa là muốn khai báo public key của VM1 tới server (VM2)



Oke đã được add tới VM2



Oke trên server (VM2) đã có public key của client VM1

Giờ vào VM1 để ssh tới server



Đù vẫn bắt nhập pass ak

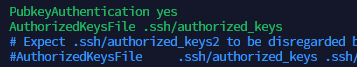
* Ta phải cấu hình server xác thực bằng SSH key

nano /etc/ssh/sshd\_config

thiết lập 2 dòng này

PubkeyAuthentication yes

AuthorizedKeysFile .ssh/authorized\_keys



## Cách 3

Vào con server genkey

ssh-genkey -t rsa

nano /etc/ssh/sshd\_config

thêm dòng

PubkeyAuthentication yes

AuthorizedKeysFile .ssh/authorized\_keys

systemctl restart sshd

cp id\_rsa.pub /root/.ssh/authorized\_keys

* /root/.ssh/authorized\_keys : lưu ý là cái này thiết lập cho client access server qua tk root

chmod 600 /root/.ssh/authorized\_keys

chmod 700 /root/.ssh

chmod 700 /root/

* Lưu ý dòng:

AuthorizedKeysFile .ssh/authorized\_keys

Nơi lưu file xác thực (nơi lưu file public key), trong đó mỗi tài khoản sẽ có 1 đường dẫn của file này

Vd: TK root thì n trong thư mục

/root/.ssh/authorized\_keys

* Tk vietdung

/home/vietdung/.ssh/authorized\_keys

Tức là khi ssh đến server qua user vietdung thì n sử dụng /home/vietdung /.ssh/authoriz còn đến server qua tk root thì n sẽ dùng cái /root/.ssh

File lưu Publickey: .ssh/authorized\_keys

---: root /root/.ssh/authorized\_keys

---: abc /home/abc/.ssh/authorized\_keys

* Bây giờ, cái private key vẫn đang nằm trên server, mà cái private key này thì cần cho client authen tới server

Copy file private key ở server tới client

Có 2 cách copy:

1. Đứng từ phía client

scp root@20.65.95.232:/root/.ssh/id\_rsa /home/vietdung/

[root@20.65.95.232:/root/.ssh/id\_rsa](mailto:root@20.65.95.232:/root/.ssh/id_rsa): cái này là user root vs IP, folder cảu server

lưu ý là lệnh này sẽ ko dùng được vì thằng client sẽ ko có quyền thực hiện trên folder root của server

cho nên cách này chỉ dùng cho access user thường của server

1. Đứng từ phía server

scp id\_rsa [vietdung@20.10.179.171:/home/vietdung](mailto:vietdung@20.10.179.171:/home/vietdung/key)

[vietdung@20.10.179.171:/home/vietdung](mailto:vietdung@20.10.179.171:/home/vietdung) : user IP và folder của client

còn id\_rsa là của server

oke đã cp về dk



Nếu client đứng ở user root thì sẽ sử dụng file /root/.ssh/config cho việc xác thưc ssh key

Còn file /root/.ssh/authorized\_keys sẽ là nơi lưu public key ở server cho client access đến user root của server

Tương tự client đứng ở user vietdung thì n sẽ sử dụng file /home/vietdung/.ssh/config cho việc xác thưc ssh tới server

Còn file /home/vietdung/.ssh/authorized\_keys sẽ là nơi lưu public key ở server của client cho client access đến user vietdung của server

Đang đứng ở user root của client

Nano /root/.ssh/config

Host 20.65.95.232

User root

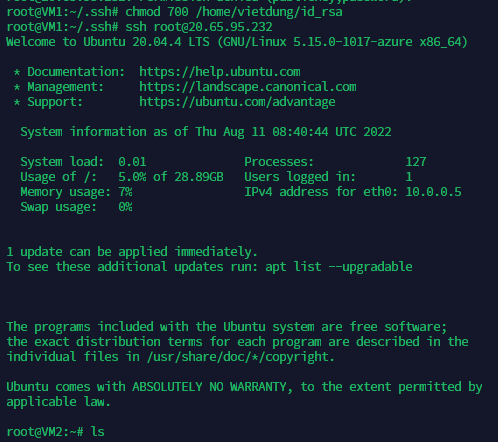
Port 22

PreferredAuthentications publickey

IdentityFile "/home/vietdung/id\_rsa"

IP kia là IP server, chọc đến user root của server

Private key cho việc xác thực ở /home/vietdung/id\_rsa



Oke đã vào dkVM2 server r mà ko cần user/pass



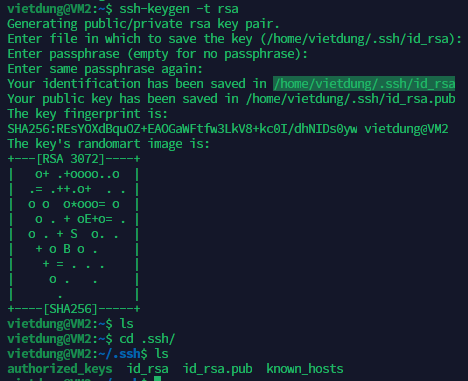
Mới thiết lập cho access vs user root thôi nha

* Cách 4: tạo key từ user vietdung và access tới server bằng user vietdung

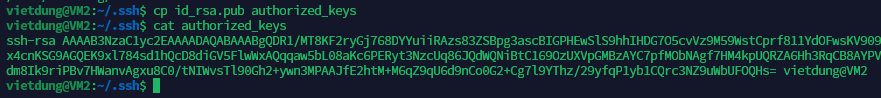
Vẫn vào server với user đang đứng là vietdung

ssh-genkey –t rsa

thì lúc này user: vietdung sinh ra key ở /home/vietdung/.ssh



cp id\_rsa.pub /home/vietdung/.ssh/authorized\_keys



Ko cần config sshd\_config nữa

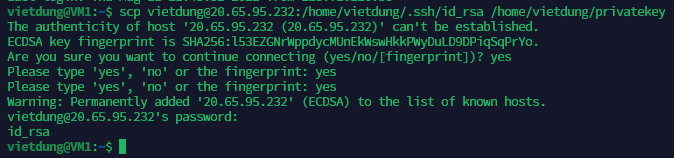
chmod 600 /home/vietdung/.ssh/authorized\_keys

chmod 700 /home/vietdung/.ssh/

chmod 700 /home/vietdung

về máy client copy, lần này là copy được, trước đó vs cách 3 là ko copy ở folder root của server dk

scp vietdung@20.65.95.232:/home/vietdung/.ssh/id\_rsa /home/vietdung/privatekey



[vietdung@20.65.95.232:/home/vietdung/.ssh/id\_rsa](mailto:vietdung@20.65.95.232:/home/vietdung/.ssh/id_rsa) : này là của server

nano /home/vietdung/.ssh/config

Host 20.65.95.232

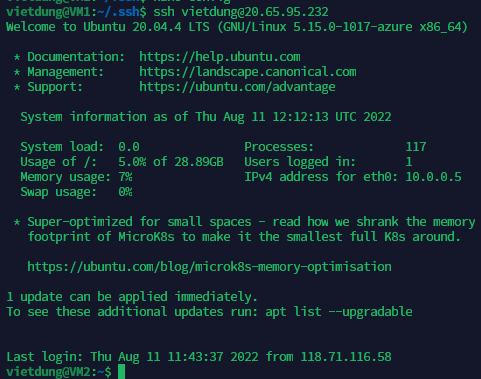
User vietdung

Port 22

PreferredAuthentications publickey

IdentityFile "/home/vietdung/privatekey"

/home/vietdung/privatekey : nơi lưu privatekey



Oke đã được, access dk vào VM2 r

## Cách 5: gen key từ client rồi push publickey lên server

Bgio vào client genkey

VM1 - client

ssh-keygen -t rsa

cp publickey tới vietdung/authen của client hoặc root/authen nếu muốn access user root

nano /home/vietdung/.ssh/config

Host 20.65.95.232

User vietdung

Port 22

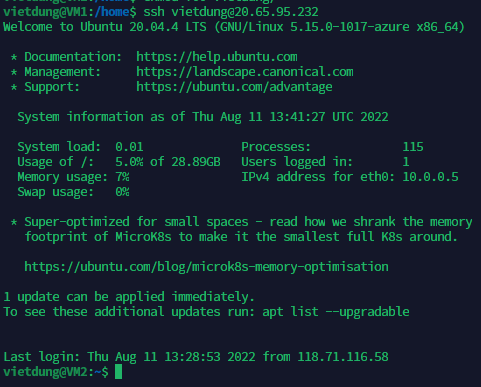
PreferredAuthentications publickey

IdentityFile "/home/vietdung/.ssh/id\_rsa"

chmod 700 .ssh/

chmod 700 vietdung/

VM2 # chmod 600 authorized\_keys



Oke đã SSH đk

# Thiết lập ansible từ ubuntu tới VM1 – VM2



Ban đầu cũng thiết lập ssh từ ừ ubuntu tới 2 con VM

Ubuntu # ssh-keygen -t rsa

vietdung@master:~$ chmod 700 .ssh/

vietdung@master:/home$ chmod 700 vietdung/

scp id\_rsa.pub [vietdung@20.65.95.232:/home/vietdung/.ssh/authorized\_keys](mailto:vietdung@20.65.95.232:/home/vietdung/.ssh/authorized_keys)

lệnh scp này lưu ý ta ko nên dùng, vì n sẽ ghi đè lên nội dung của file cũ

ta nên copy nội dung rồi add vào file authorized\_keys

lưu ý ta sẽ dùng 1 key để xác thực cho cả 2 user, tức là push file publickey tới 2 server

vietdung@master:~/.ssh$ nano config

Host 20.65.95.232

User vietdung

Port 22

PreferredAuthentications publickey

IdentityFile "/home/vietdung/.ssh/id\_rsa"

Host 20.10.179.171

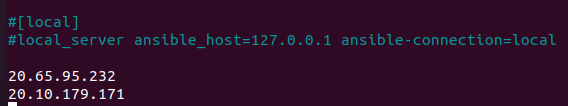
User vietdung

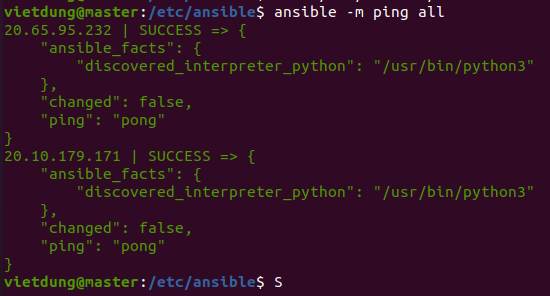
Port 22

PreferredAuthentications publickey

IdentityFile "/home/vietdung/.ssh/id\_rsa"

nano /etc/ansible/hosts

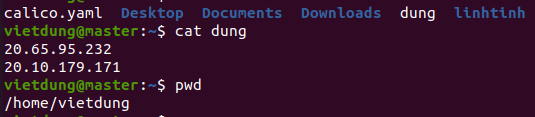


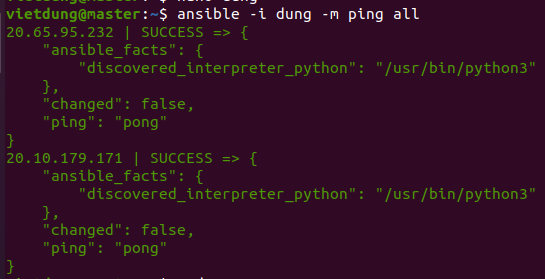


Oke

Nếu ta ko muốn sử dụng file mặc định /home/vietdung/ansible

Ta tạo 1 file bất kỳ chưa IP của máy là được



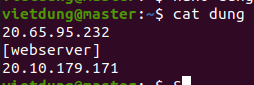


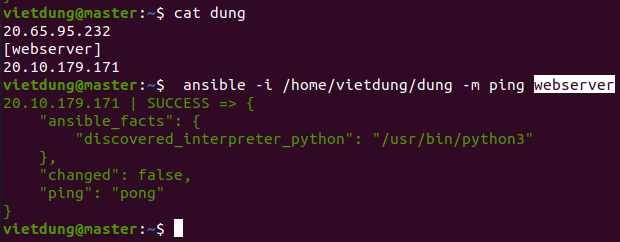
Oke chưa

ansible -i dung -m ping all

ansible -i /home/vietdung/dung -m ping all

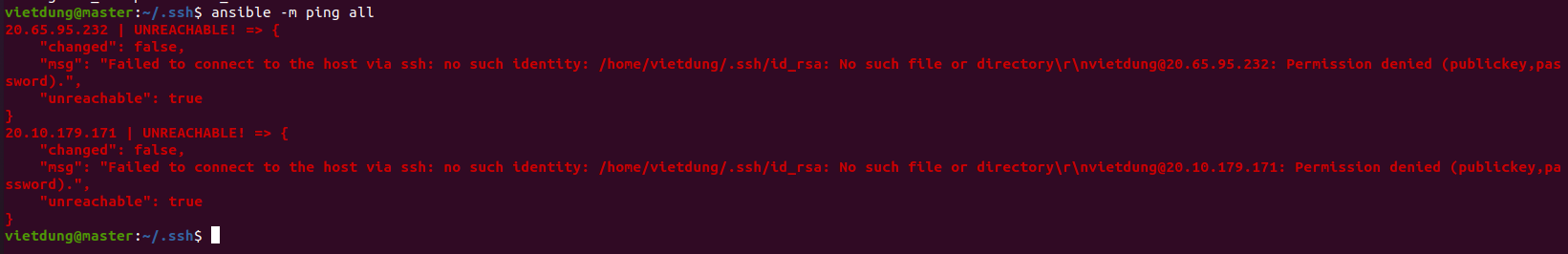
đặt tên cho con **20.10.179.171** là **webserver**





Bgiờ xóa cái private key trên ubuntu, tức là sẽ ko thể ssh qua ssh key tới VM1 VM2 nữa

Ansible ping ko được, n yêu cầu phải ssh tới server trước



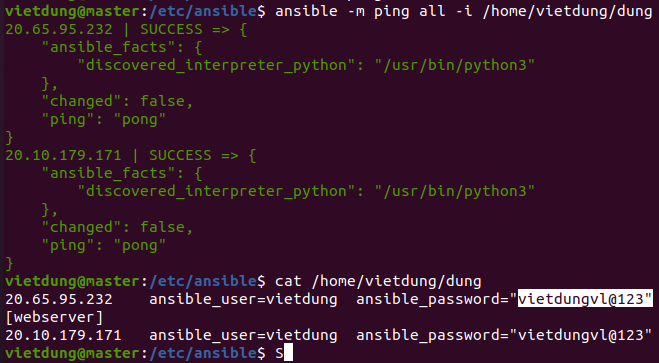
Tuy vậy ansible hỗ trợ 1 kiểu xác thực khác

nano /home/vietdung/dung

20.65.95.232 ansible\_user=vietdung ansible\_password="vietdungvl@123"

[webserver]

20.10.179.171 ansible\_user=vietdung ansible\_password="vietdungvl@123"



Oke thế là thiết lập kiểu này thì ko cần ssh key, và

nano dung1

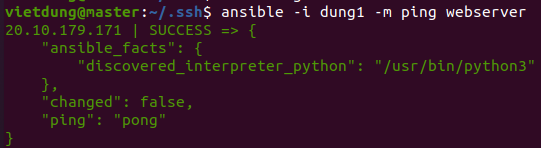
all:

children:

webserver:

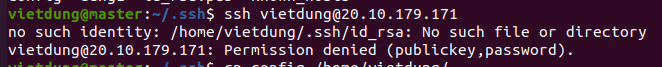
hosts:

20.10.179.171:



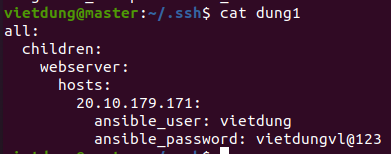
Cái này là với trường hợp đã xác thực với VM qua ssh key

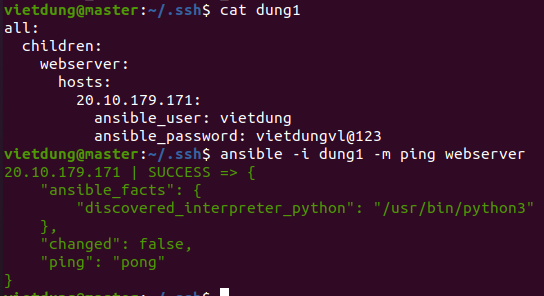
Lưu ý là để ssh tới server mà dùng user/pass thì phải xóa cái file config ở thư mục .ssh/config đi



nano dung1

điền user/pass luôn vào đây



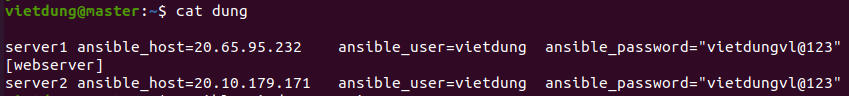


Rồi oke

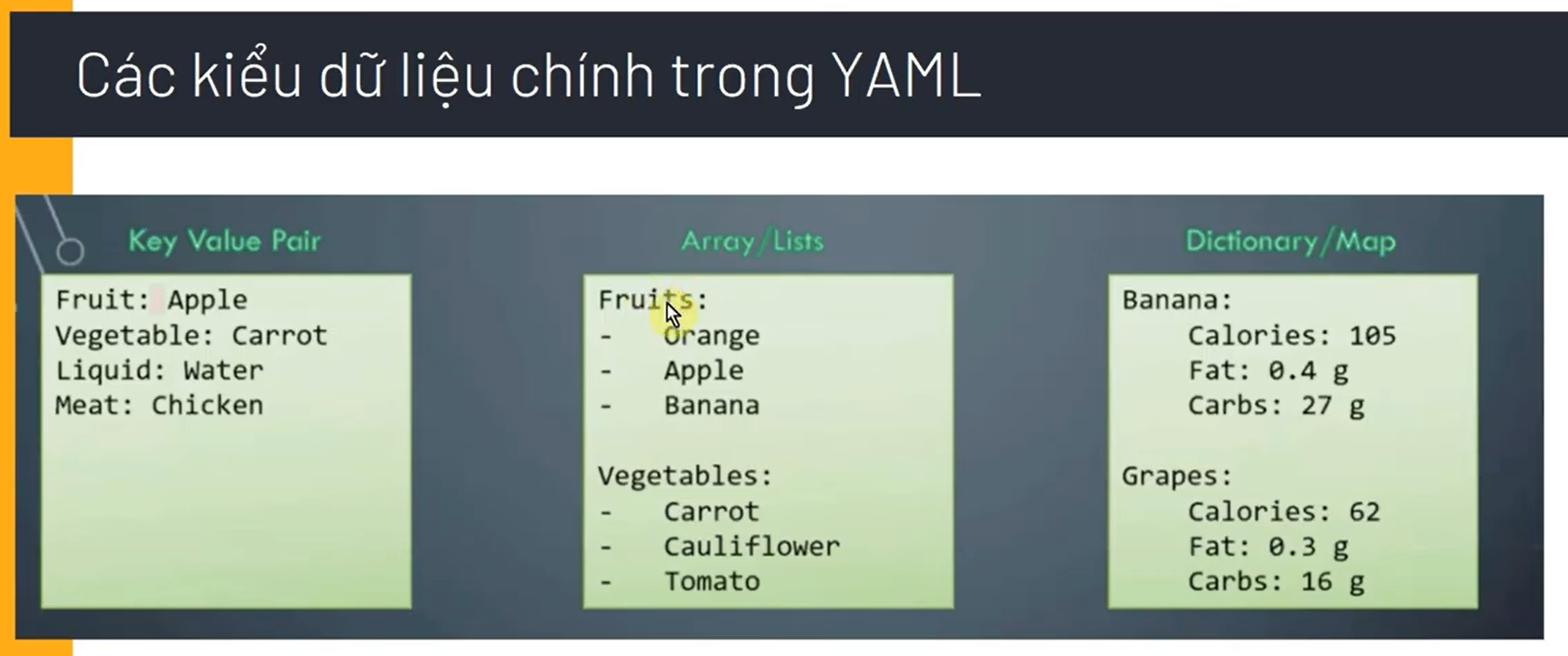
Lưu ý phải là children

Trên là định dạng yaml

* Hoặc ta có thêm định dạng parten như này



# Kiểu dữ liệu



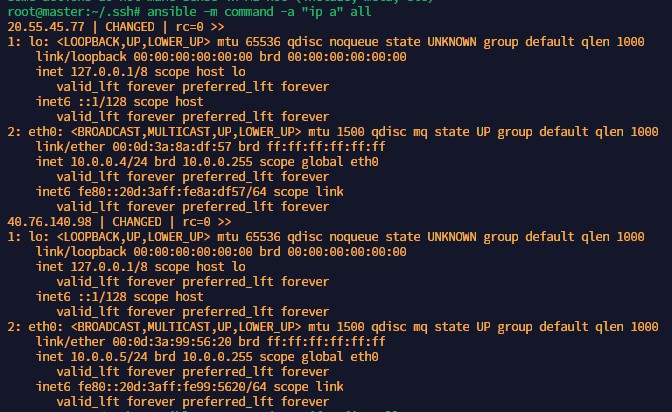
# Playbook, Task, Module

ansible -i hosts -m ping all

ansible -i hosts -m command -a “ip a” all

-m : module - module là command

-a : argument của module



N hỗ trợ việc mình làm 1 việc trên nhiều máy 1 lúc

Nhưng bgio có 1 yêu cầu phức tạp là

ansible -i hosts -m command -a “apt install –y ; systemctl restart nginx; ….” All

* Phức tạp

## Tạo playbook

nano playbook.yml

---

- hosts: all

  tasks:

    - name: test ping

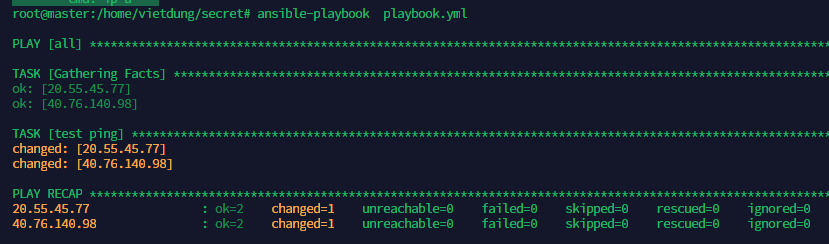
      command:

        cmd: ip a

chia thành từng tasks có name là test ping

ansible-playbook playbook.yml

or ansible-playbook playbook.yml -i dung (với dung là file cấu hình)



VD2:

---

- hosts: all

  tasks:

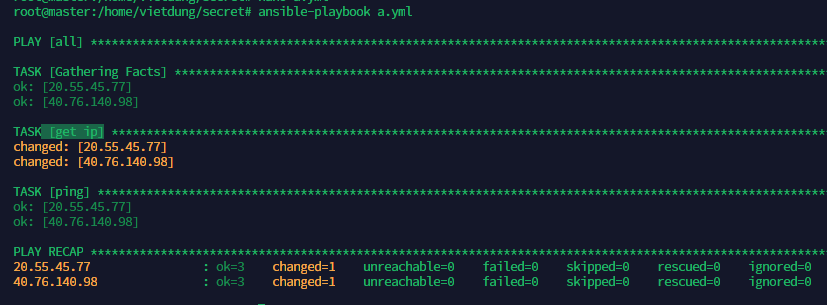
    - name: get ip

      command:

        cmd: ip a

    - name: ping

      ping:



N chạy cũng phân task rõ ràng

Search module ansible

nano /etc/ansible/hosts

server1 ansible\_host=20.55.45.77  ansible\_user=vietdung  ansible\_password="vietdungvl@123"

40.76.140.98  ansible\_user=vietdung  ansible\_password="vietdungvl@123"

[all:vars]

  ntp\_server=192.168.1.1

nano playbook.yaml

---

- name: Demo play

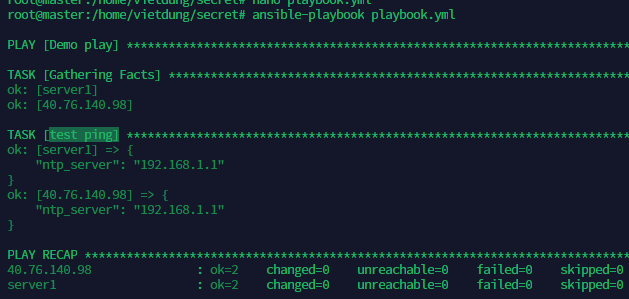
  hosts: all

  tasks:

    - name: test ping

      debug:

        var: ntp\_server



Cách x:

Nano playbook.yml

---

- name: Demo play

  hosts: all

  tasks:

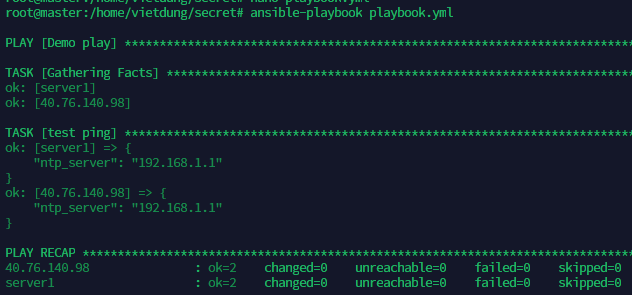
    - name: test ping

      debug:

        var: ntp\_server

  vars:

    ntp\_server: 192.168.1.1



Debug: để in

* In kiểu msg
* In ra giá trị của biến var

VD tiếp



Tạo 1 file /var/file.yml



Nano playbook.yml

---

- name: Demo play

  hosts: all

  tasks:

    - name: test ping

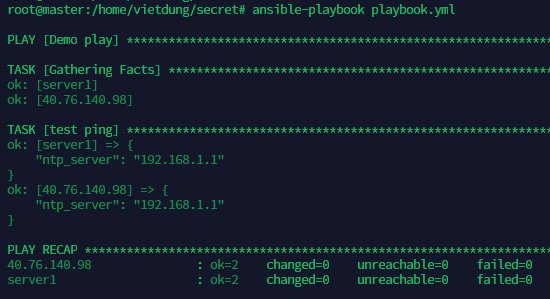
      debug:

        var: ntp\_server

  vars\_files:

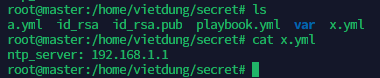
    -  ./var/file.yml

Đường dẫn của vars\_files tính từ cái đường dẫn cảu playbook.yml



Oke

Bây giờ tạo file x.yml ngang hàng với playbook.yml



Nano playbook.yml

---

- name: Demo play

  hosts: all

  tasks:

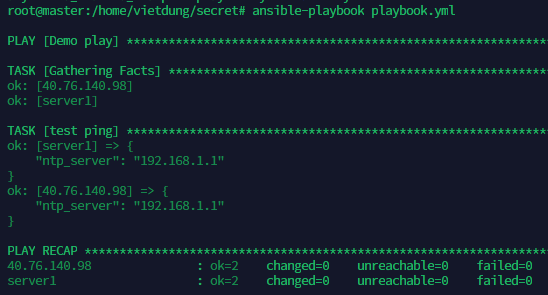
    - name: test ping

      debug:

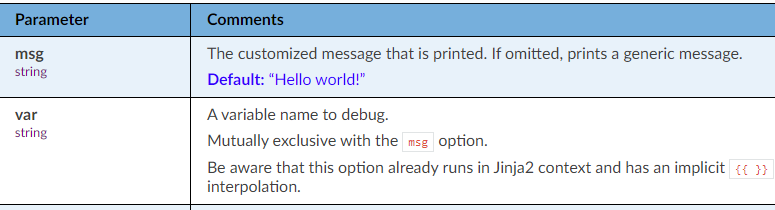
        var: ntp\_server

  vars\_files:

    -  ./x.yml



# variable – debug



Trong code, debug rất quan trọng, để show error, n giống print

Debug cũng là 1 module,

VD: nano debug.yml

---

- name: demo playubook

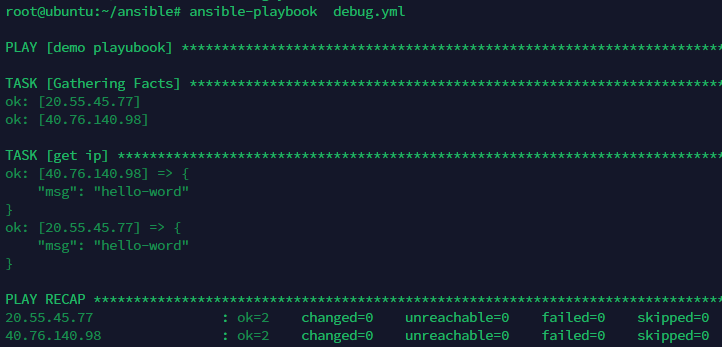
  hosts: all

  tasks:

    - name: get ip

      debug:

        msg: "hello-word"



Oke đã in ra **hello world**

## Show biến – var

---

- name: demo playubook

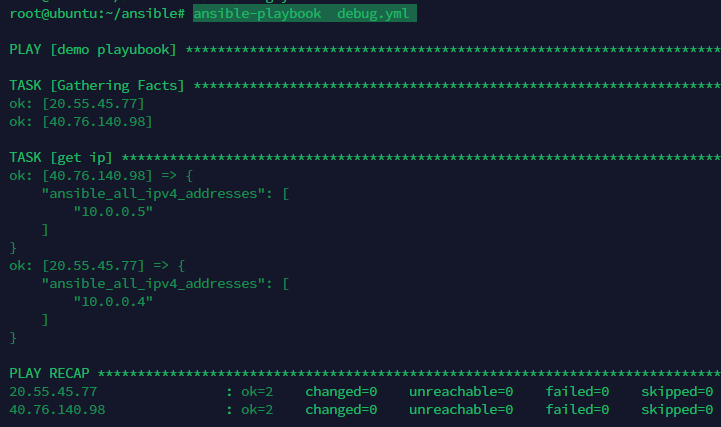
  hosts: all

  tasks:

    - name: get ip

      debug:

        var: ansible\_all\_ipv4\_addresses



Đã show ra IP

Tạo file biến

nano /etc/ansible/hosts

server1 ansible\_host=20.55.45.77  ansible\_user=vietdung  ansible\_password="vietdungvl@123"

40.76.140.98  ansible\_user=vietdung  ansible\_password="vietdungvl@123"

[all:vars]

  ntp\_server=192.168.1.1

nano playbook.yaml

---

- name: Demo play

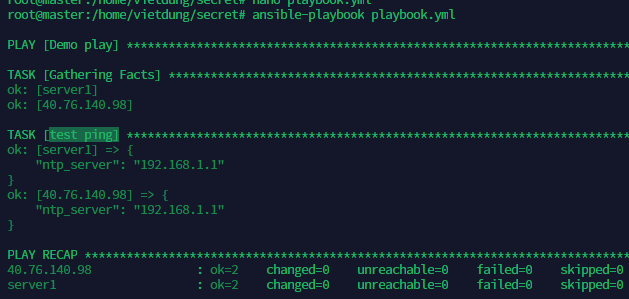
  hosts: all

  tasks:

    - name: test ping

      debug:

        var: ntp\_server



Cách x:

Nano playbook.yml

---

- name: Demo play

  hosts: all

  tasks:

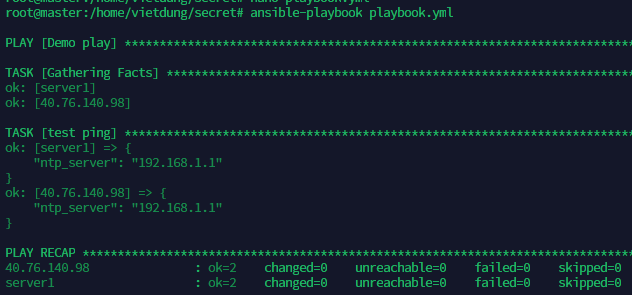
    - name: test ping

      debug:

        var: ntp\_server

  vars:

    ntp\_server: 192.168.1.1



Debug: để in

* In kiểu msg
* In ra giá trị của biến var

VD tiếp



Tạo 1 file /var/file.yml



Nano playbook.yml

---

- name: Demo play

  hosts: all

  tasks:

    - name: test ping

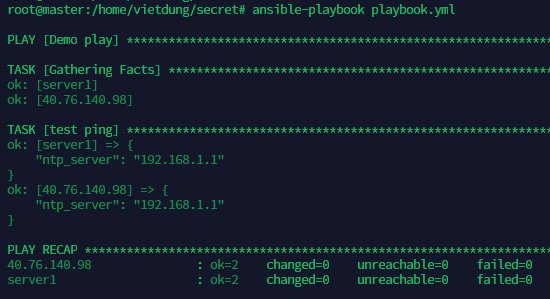
      debug:

        var: ntp\_server

  vars\_files:

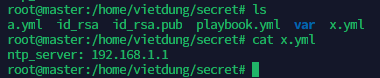
    -  ./var/file.yml

Đường dẫn của vars\_files tính từ cái đường dẫn cảu playbook.yml



Oke

Bây giờ tạo file x.yml ngang hàng với playbook.yml



Nano playbook.yml

---

- name: Demo play

  hosts: all

  tasks:

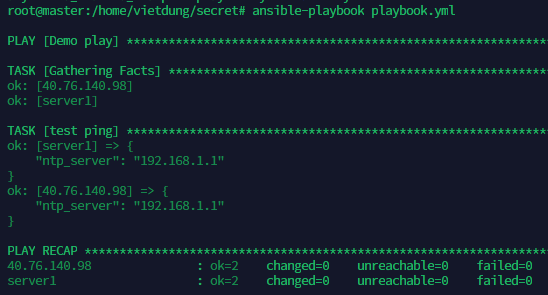
    - name: test ping

      debug:

        var: ntp\_server

  vars\_files:

    -  ./x.yml



## ansible\_facts

- name: demo playbook

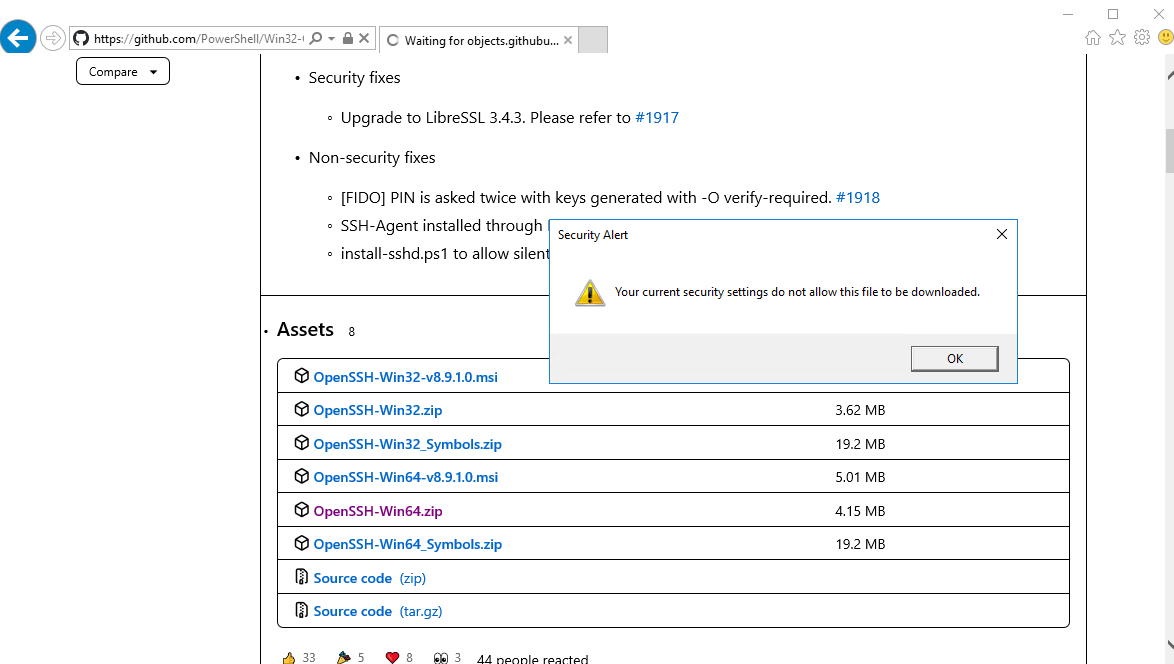
  hosts: all

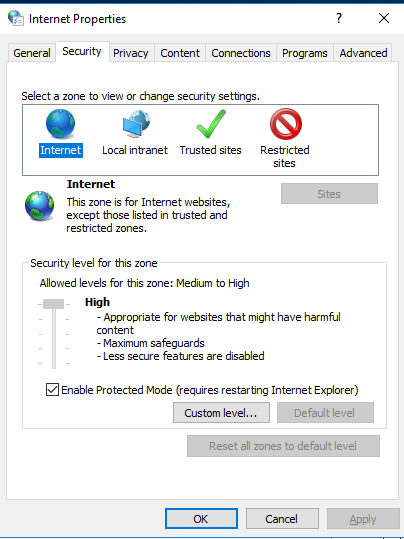
  tasks:

    - name: test ping

      debug:

        var: ansible\_facts





# Template

File playbook

- name: demo

  hosts: all

  become\_user: root

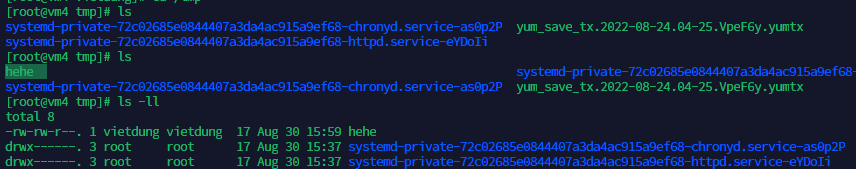
  tasks:

    - template:

        src: ./foo.j2

        dest: /tmp/hehe

copy từ file playbook **foo.j2** tới folder **/tmp/hehe** của con host

****

Đã copy được n sang host mới

Tạo biến trong file /foo.j2 để copy đến từng server, n mang 1 giá trị khác

## Biến

{{ custome\_variable }}

Biến có dạng 2 dấu ngoặc

* Nội dung trong file foo.j2 là biến **{{ custome\_variable }}**
* Copy file **foo.j2** này tới host khác
* Nhưng giá trị trong file sẽ là giá trị của biến

.

Playbook

- name: demo

  hosts: all

  become\_user: a

  tasks:

    - template:

        src: ./foo.j2

        dest: /tmp/hehe

  vars:

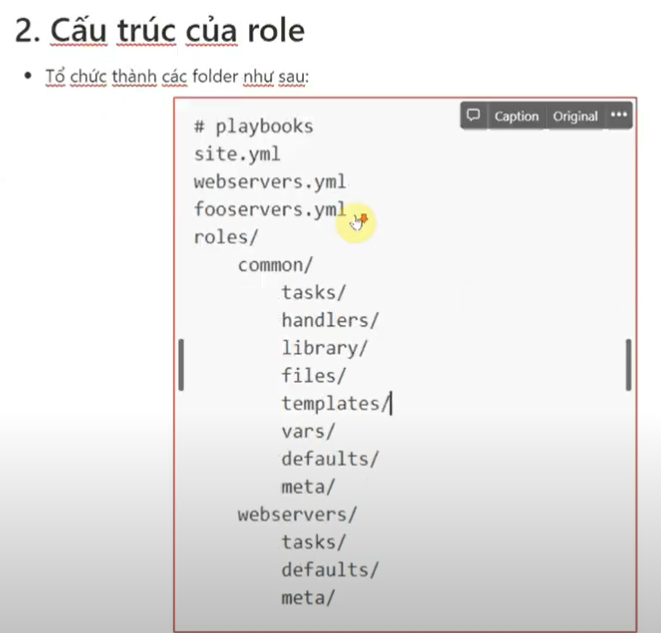
    custome\_variable: demo

chuyền biến kia có giá trị là demo

ta có thể thay giá trị của biến là 1 đường dẫn của file

còn trong file đã được xác định đó là biến, ko phải giá trị

# Role



Trong role quy ước tên 1 số thư mục như mục

* Tasks (bắt buộc phải có)
* File
* Vars
* template
* …..

VD.

Tạo 1 tasks trong role, định nghĩa cho task sẽ là file main.yaml



Task này chỉ có nhiệm vụ ping

Tạo playbook để gọi đến role task đó

Playbook.yml

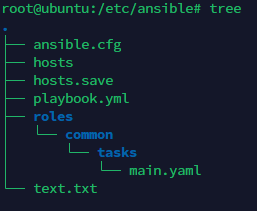
- name: demo test ping

  hosts: all

  roles:

    - common

VD2: tạo thêm task copy



nano /etc/ansible/roles/common/tasks/main.yaml

---

- name: test ping

  ping:

- name: test copy

  ansible.builtin.copy:

    src: text.txt

    dest: /tmp/tesst.txt

    #owner: root

    #group: root

Playbook

- name: demo test ping

  hosts: all

  roles:

    - common

File src tính từ file playbook để thực thi

Vào con host dk copy

Oke đã có



## Tạo biến trong role

Nó sẽ quy định 1 số cái

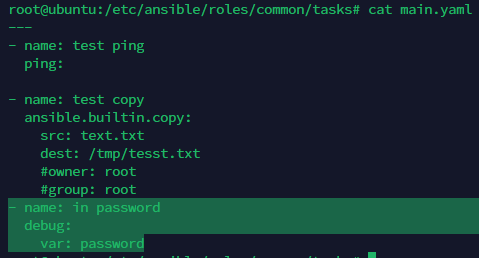
Tạo file folder /vars/main.yaml trong common – role



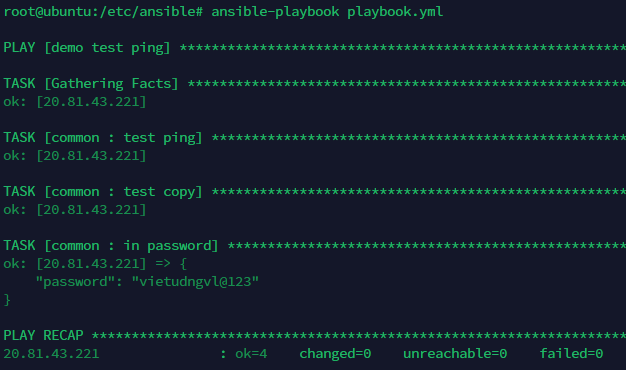
Biến **password** bằng vietdung

Nhưng nếu muốn in ra ta phải tạo n trong file **task – role**

Vào tasks/main.yaml



**Oke chạy**

****

# Ansible

---

- name: example playbook

  hosts: all                  # target host cho cái name

  vars:

    myvar: "example text"     # biến môi trường kiểu string

    mybool: false

    mycities:

      - New York

      - Paris

  tasks:                      # cái này là quan trọng nhất để thực thi task

    - name: print var

      ansible.builtin.debug:  # module để in

        msg: "value {{ myvar }}"

      notify: reload

    - name: condition

      ansible.builtin.debug:

        msg: "example condition"

      when: mybool

    - name: print cities

      ansible.builtin.debug:

        var: item             # mặc định ở vòng loop cái này luôn là item

      loop: "{{ mycities }}"

  handlers:                   # cái này chưa biết

    - name: reload

      ansible.builtin.debug:

        msg: "example handler"

## Module1: module in

* Tạo file inventory

nano inventory

localhost ansible\_connection=local

* 3 kiểu in cho module debug

1.In kiểu message

---

- name: debug module demo

  hosts: all

  vars:

    fruit: "apple"

  tasks:

    - name: debug message

      ansible.builtin.debug:

        msg: "đây là tao"

2.In kiểu variable

---

- name: debug module demo

  hosts: all

  vars:

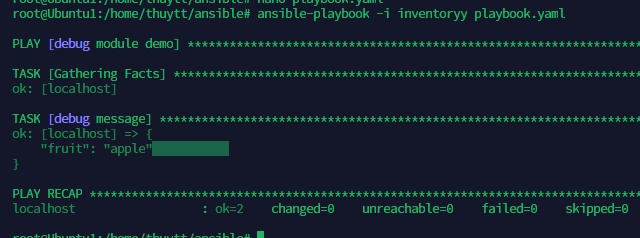
    fruit: "apple"

  tasks:

    - name: debug message

      ansible.builtin.debug:

        var: fruit



3.IN kiểu message + var

---

- name: debug module demo

  hosts: all

  vars:

    fruit: "apple"

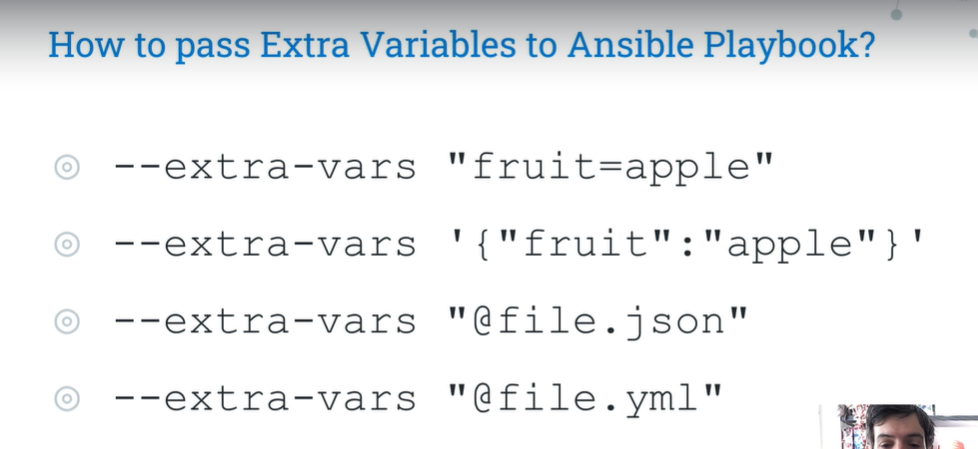
  tasks:

    - name: debug message

      ansible.builtin.debug:

        msg: "our fruit is {{ fruit }}"

## Extra Variables



Var này chỉ có giá trị trong dòng lệnh terminal để truyền vào playbook

---

- name: demo extra var

  hosts: all

  vars:

    fruit: "banana"

  tasks:

    - name: extra var banana

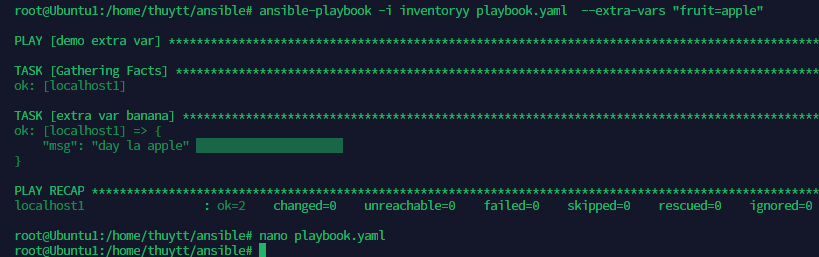
      ansible.builtin.debug:

        msg: "day la {{ fruit }}"

t có 1 playbook này, nhưng t truyền 1 variable khác vào giá trị var

C1:

ansible-playbook -i inventoryy playbook.yaml --extra-vars "fruit=apple"



Kqua đã mang giá trị truyền vào

def = define

C2:

ansible-playbook -i inventoryy playbook.yaml --extra-vars '{"fruit": "apple"}'

## Variable nhiều dòng

---

- name: ansible multi-line

  hosts: all

  vars:

    variable1: |

      bo may la viet dung

      bo may 22 tuoi

      bo may o Nam Dinh

    variable2: >

      xxx la nguoi yeu tao

      xxx 30 tuoi

      dis con me may

  tasks:

    - name: demo task multi-line1

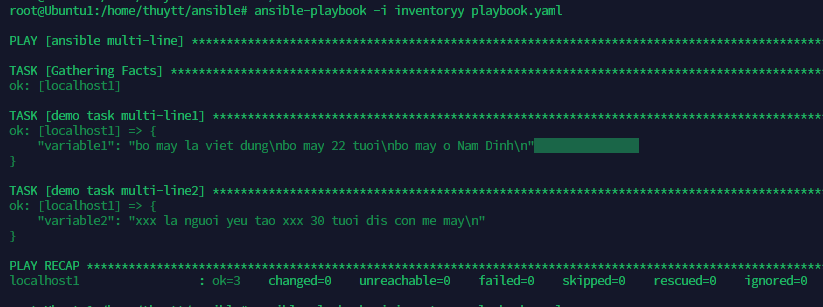
      ansible.builtin.debug:

        var: variable1

    - name: demo task multi-line2

      ansible.builtin.debug:

        var: variable2



    variable1: |

      bo may la viet dung

      bo may 22 tuoi

      bo may o Nam Dinh

output:

"variable1": "bo may la viet dung\nbo may 22 tuoi\nbo may o Nam Dinh\n"

    variable2: >

      xxx la nguoi yeu tao

      xxx 30 tuoi

      dis con me may

output:

"variable2": "xxx la nguoi yeu tao xxx 30 tuoi dis con me may\n"

C2:

---

- name: ansible multi-line

  hosts: all

  vars:

    variable1: |-

      bo may la viet dung

      bo may 22 tuoi

      bo may o Nam Dinh

    variable2: >-

      xxx la nguoi yeu tao

      xxx 30 tuoi

      dis con me may

  tasks:

    - name: demo task multi-line1

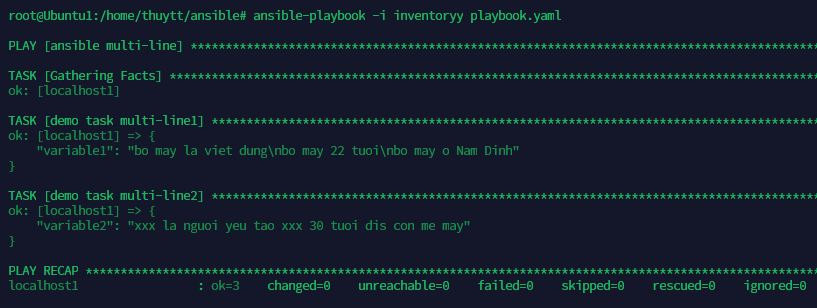
      ansible.builtin.debug:

        var: variable1

    - name: demo task multi-line2

      ansible.builtin.debug:

        var: variable2



Dấu “-” để bỏ \n ở sau, dạng chuẩn

Hoặc

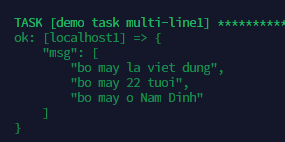
  tasks:

    - name: demo task multi-line1

      ansible.builtin.debug:

        msg: "{{ variable1.split('\n') }}"

split : tách \n (cái này áp dụng cho dạng “|” )



## Inventory\_hostname && ansible\_hostname

File inventory trỏ đến con server khác

ubuntu  ansible\_host=20.6.107.99

[all:vars]

ansible\_connection=ssh

ansible\_user=thuytt

ansible\_password=Qwertyuiop@123

file playbook.yml

---

- name: ansible multi-line

  hosts: all

  gather\_facts: true

  tasks:

    - name: demo task multi-line1

      ansible.builtin.debug:

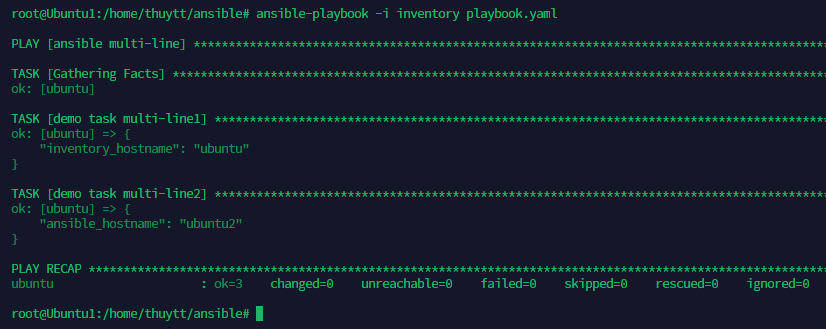
        var: inventory\_hostname

    - name: demo task multi-line2

      ansible.builtin.debug:

        var: ansible\_hostname

mặc định cái var inventory\_hostname và ansible\_hostname là thông tin mặc định của thằng target host định nghĩa trong inventory



Nhầm, ansible\_hostname là hostname thằng server

Inventory\_hostname: tên định nghĩa trong file inventory

gather\_facts: true – mặc định là true, n thu thập thông tin target server

## environment

---

- name: remote environment demo

  hosts: all

  gather\_facts: false

  environment:

    EXAMPLE: test1

  tasks:

    - name: diplay EXAMPLE

      ansible.builtin.command: "echo $EXAMPLE"

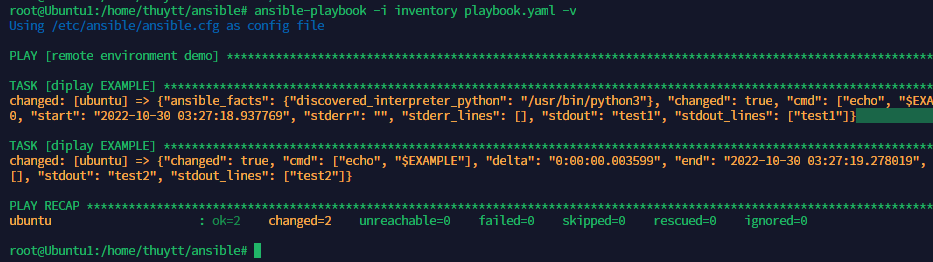
    - name: diplay EXAMPLE

      ansible.builtin.command: "echo $EXAMPLE"

      environment:

        EXAMPLE: test2

Giá trị enviroment trong task mạnh hơn bên ngoài



Thêm trường –v

ansible-playbook -i inventory playbook.yaml –v

* Limit

[linux]

ubuntu1  ansible\_host=20.6.107.99

ubuntu2  ansible\_host=20.6.107.167

[all:vars]

ansible\_connection=ssh

ansible\_user=thuytt

ansible\_password=Qwertyuiop@123

* Chỉ deploy trên ubuntu1

ansible-playbook -i inventory playbook.yaml -v --litmit=ubuntu1

## truyền enviroment

---

- name: remote environment demo

  hosts: "{{ HOSTS }}"

  gather\_facts: true

  tasks:

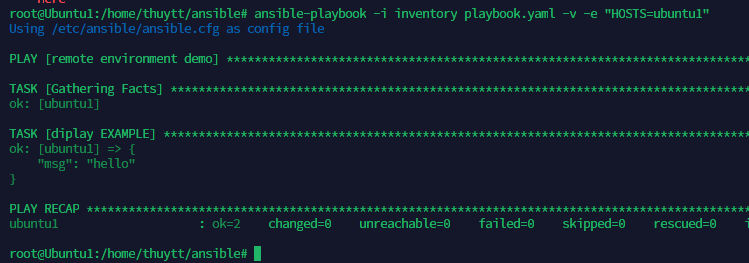
    - name: diplay EXAMPLE

      ansible.builtin.debug:

        msg: "hello"

truyền hosts khi chạy ansible

ansible-playbook -i inventory playbook.yaml –v -e "HOSTS=ubuntu1"



## Ansible Register Module

Nếu bạn muốn lưu trữ kết quả đầu ra trong một biến và sử dụng nó sau này, thì bạn có thể sử dụng **Ansible Register Module**

Mà đơn giản n là exec command tới target host

Bgio tôi sẽ dùng cái này để tạo user/password

---

- name: remote environment demo

  hosts: ubuntu1

  user: ansible

  become: True

  tasks:

    - name: Ensure pwgen is installed

      apt:

         name: pwgen            # đảm bảo là pwgen là được install

         state: present         # Nếu chưa có n sẽ install

         update\_cache: True

    - name: Generate password

      ansible.builtin.shell: 'pwgen -N 1 -s 30'

      register: mypass

    - name: Print the generated password

      ansible.builtin.debug:

        msg: "The password is {{ mypass.stdout\_lines }}"

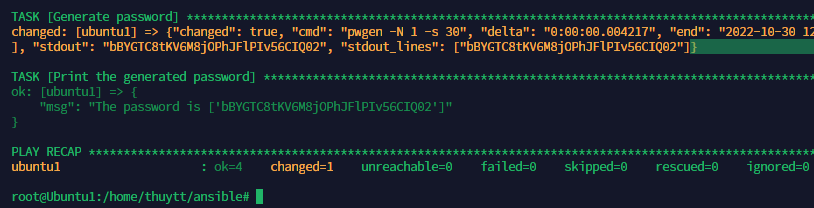


* **cmd** - Lệnh chạy để tạo đầu ra.
* **stdout** - Đầu ra của lệnh.
* **stderr** - Đầu ra lỗi của lệnh.
* **start** - Ngày và giờ khi lệnh bắt đầu được thực thi.
* **end** - Ngày và giờ khi lệnh kết thúc thực thi.
* **delta** - Thời gian chạy lệnh. Đây là sự khác biệt giữa thuộc tính **end** và **start** .
* **stdout\_lines** - Một mảng chứa mỗi dòng đầu ra của lệnh. Tương tự như **stdout** , nhưng **stdout** phân tách các dòng bằng ký tự dòng mới (\ n) thay vì mảng.

thêm cái này là tách được password ra.

Thấy dòng stdout vs stdout\_lines kia ko

msg: "The password is {{ mypass.stdout\_lines }}"



### ansible.builtin.command

---

- name: remote environment demo

  hosts: ubuntu1

  tasks:

    - name: register command

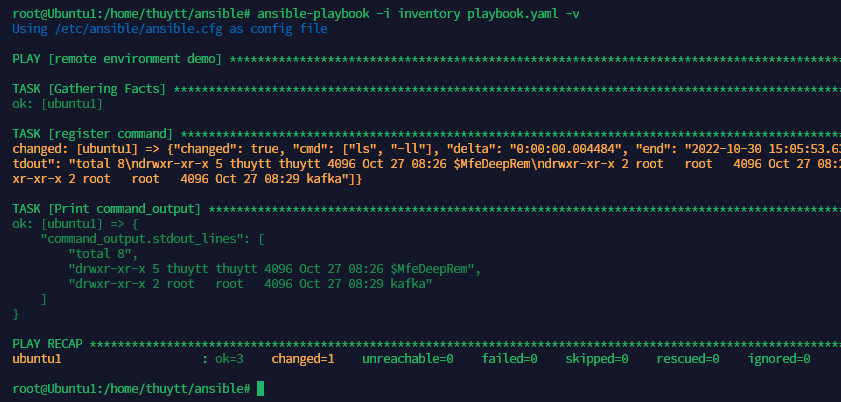
      ansible.builtin.command: 'ls -ll'

      register: command\_output        # gán biến command\_output mang giá trị của lệnh ls -ll

    - name: Print command\_output

      ansible.builtin.debug:

        var: command\_output.stdout\_lines



### ansible.builtin.shell

---

- name: remote environment demo

  hosts: ubuntu1

  tasks:

    - name: register command

      ansible.builtin.command: 'ls -ll'

      register: command\_output        # gán biến command\_output mang giá trị của lệnh ls -ll

    - name: Print command\_output

      ansible.builtin.debug:

        var: command\_output.stdout\_lines

cách hoạt động cũng tương tự như command,

vì 1 số trường hợp hệ điều hành ko hỗ trợ shell thì còn command

# install service

---

- hosts: ubuntu1

  become: yes

  tasks:

    - name: Install Apache.

      apt:

        name: "{{ item }}"

        state: present

      loop:

        - apache2

        - mysql-server

        - php

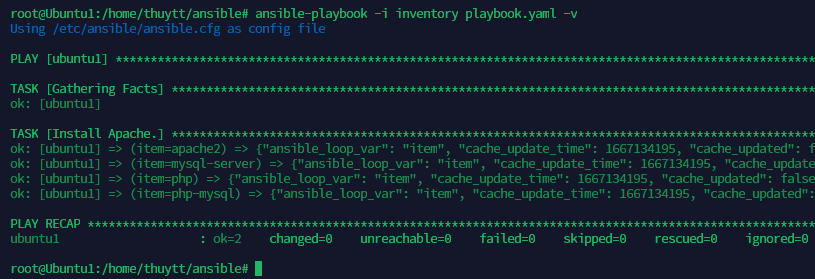
        - php-mysql

        - ansible

become và become\_user

Theo tôi hiểu, nó **become\_user** là một cái gì đó tương tự như **su <username>,** và **become** có nghĩa là một cái gì đó giống như sudo su hoặc "thực hiện tất cả các lệnh với tư cách là người dùng sudo"

có cái này mới install được, ko nó sẽ báo lỗi ko có quyền



Install xog, chạy lại thì n báo ko có j thay đổi lên nó ko install nữa

# Ansible tiếp

## Ansible copy

Kiểu 1: truyền biến được định nghĩa bên trong playbook

---

  - name: copy module

    hosts: ubuntu1

    vars:

      fruit: "bannaa"

      output: "output.txt"

    tasks:

      - name: write to file

        ansible.builtin.copy:

          content: "{{ fruit }}"

          dest: "{{ output }}"

copy giá trị của biến **fruit** vào biến **output : dùng**  ansible.builtin.copy

Kiểu 2: dùng template, customez variable

---

  - name: copy module

    hosts: ubuntu1

    vars:

      fruit: "apple"

      output: "output.txt"

    tasks:

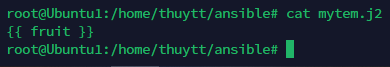
      - name: write to file

        ansible.builtin.template:

          src: "mytem.j2"

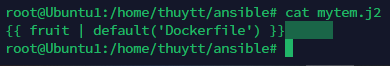
          dest: "{{ output }}"

file  **mytem.j2**



File này để gọi biến

Hoặc



Mặc định giá trị trong biến fruit được định nghĩa bên trong playbook,

Nếu fruit = null, thì lúc này fruit sẽ được gán fruit = Dockerffile

default('Dockerfile'): Dockerfile là value của fruit nếu fruit = rỗng trong playbook

## đánh tags thực hiện

---

  - name: copy module

    hosts: ubuntu1

    tasks:

      - name: print task 1

        ansible.builtin.debug:

          msg: "day la task 1"

        tags: tag1

      - name: print task 2

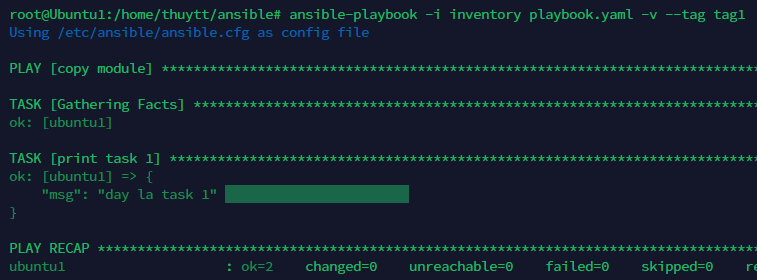
        ansible.builtin.debug:

          msg: "day la task 2"

        tags: tag2

ansible-playbook -i inventory playbook.yaml -v --tag tag1

chỉ thực thi command được đánh tag1



--tag untagged : ko có tag nào được thực hiện

--tag all : thực hiện tất cả các tag

## Check && Diff: kiểm tra thay đổi

Đôi khi bạn cần xác nhận thay đổi trên bất kỳ target node

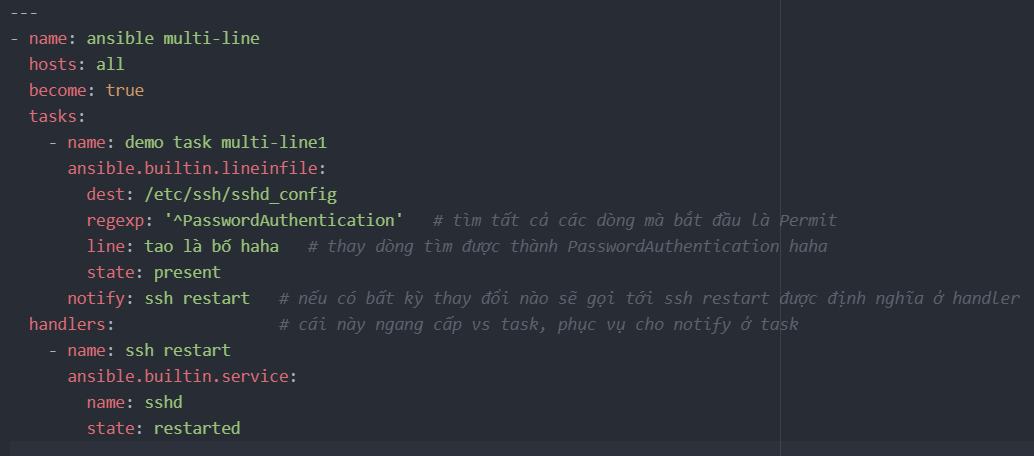
--check, --diff : có thể sử dụng qua command hoặc trong ansible playbook

- --check : là 1 chế độ mô phỏng

Thực hiện ansible playbook mà ko cần bất kỳ hành động nào trên máy mục tiêu

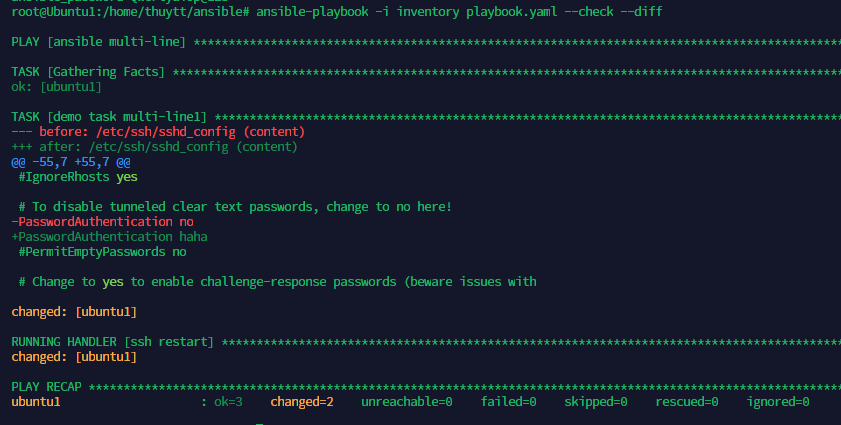
Bài 1: demo thay đổi nội dung file ở target node

ansible.builtin.lineinfile: thay đổi nội dung dòng trong file



ansible-playbook -i inventory playbook.yaml --check –diff

chạy câu lệnh này để kiểm tra xem trước là ta sẽ thay đổi những cái j,



Đây vậy là nó đã thay đổi nội dung trong file kia



Nếu n báo lỗi permission là do m chưa set quyền root cho nó

Thêm become: true

Bay giờ thực thi, bỏ --check đi, --diff sẽ hiện thay đổi, và quá trình dk thực thi

Và thay đổi đã dk thực hiện