

Cài đặt: ansible for ubuntu 20.04

apt update -y

apt install software-properties-common -y

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

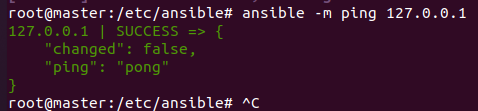
apt install ansible -y

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

[defaults]

host\_key\_checking = false

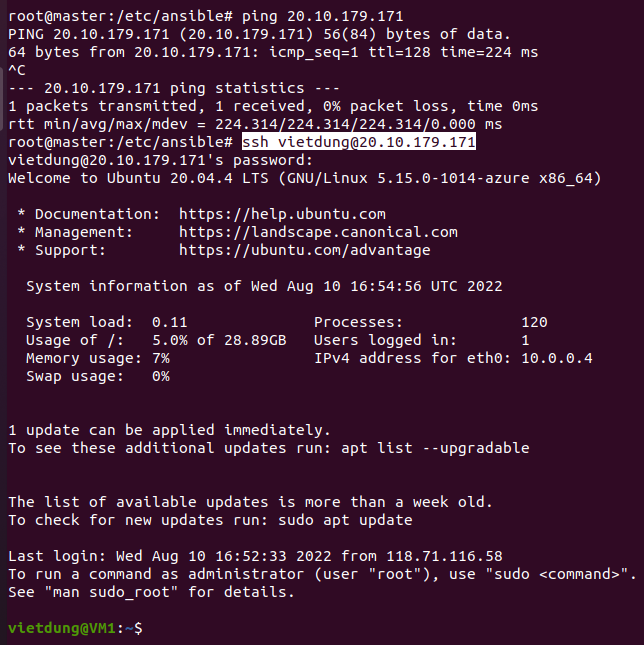
# Xác thực, ssh server

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

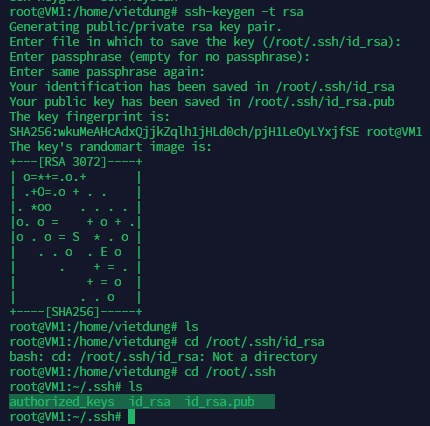


Cách 2: Sử dụng ssh-key (cách 2 đang lỗi)

* 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

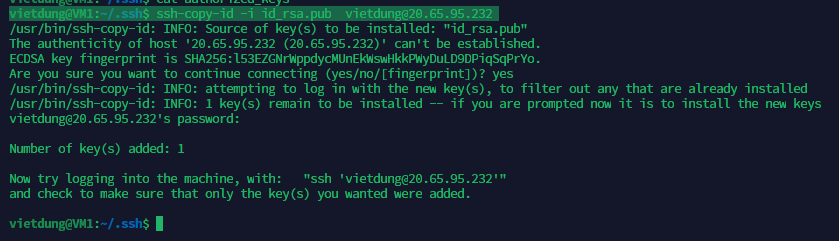
Tạo key từ VM1 và add public key từ VM1 tới server (VM2) cho việc authen

* VM1, gen-key
* Copy publickey tới VM2

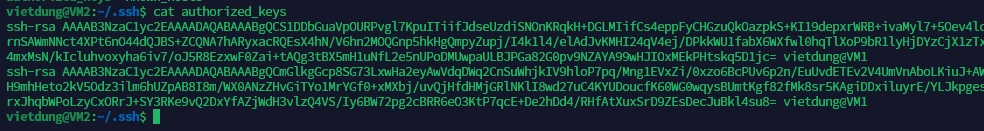
ssh-copy-id -i id\_rsa.pub [vietdung@20.65.95.232](mailto:vietdung@20.65.95.232)

[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

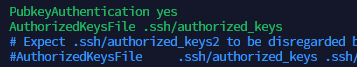
* Ta phải cấu hình server xác thực bằng SSH key

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

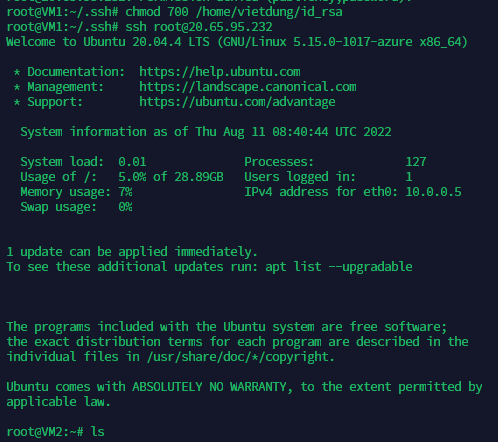
Port 22

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



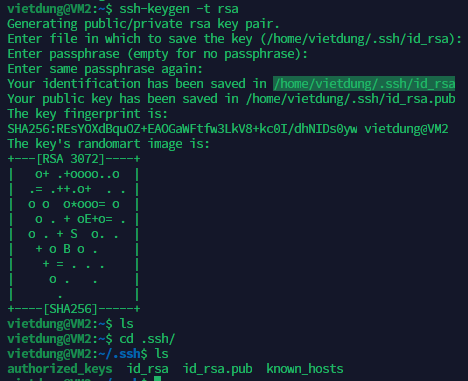
Mới thiết lập cho access vs user root thôi nha

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

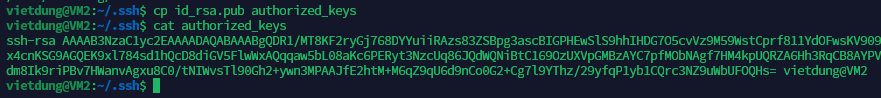
Vẫn vào server với user đang đứng là vietdung

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

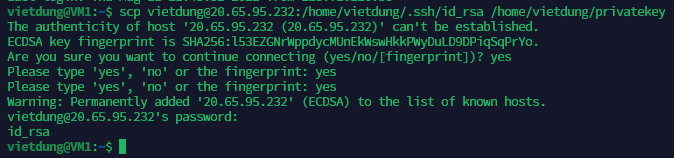
chmod 600 /home/vietdung/.ssh/authorized\_keys

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

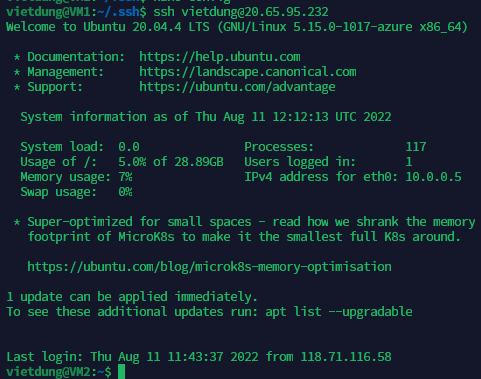
User vietdung

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

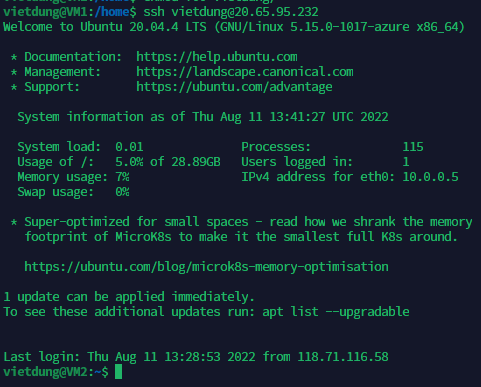
PreferredAuthentications publickey

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

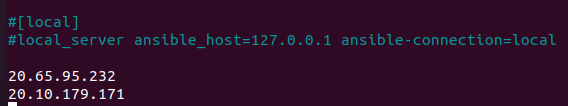
User vietdung

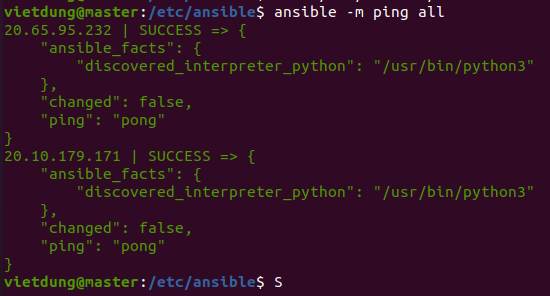
Port 22

PreferredAuthentications publickey

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

nano /etc/ansible/hosts

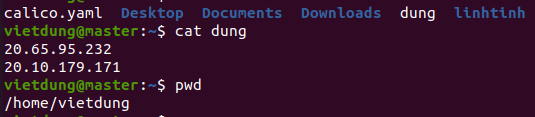


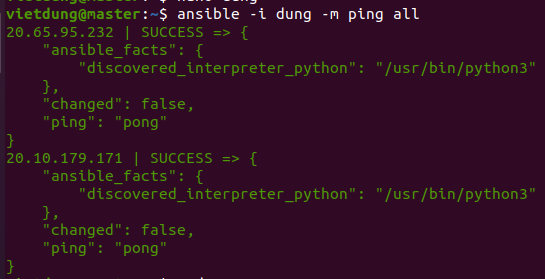


Oke

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

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



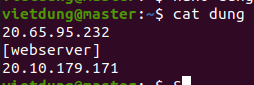


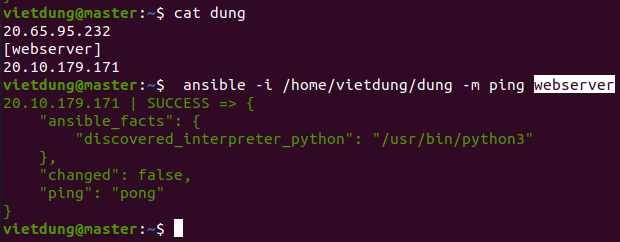
Oke chưa

ansible -i dung -m ping all

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

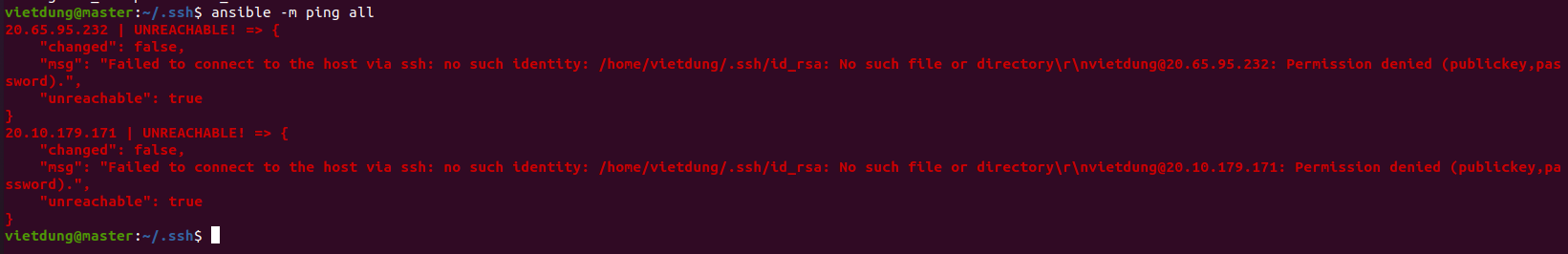
đặt tên cho con **20.10.179.171** là **webserver**





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



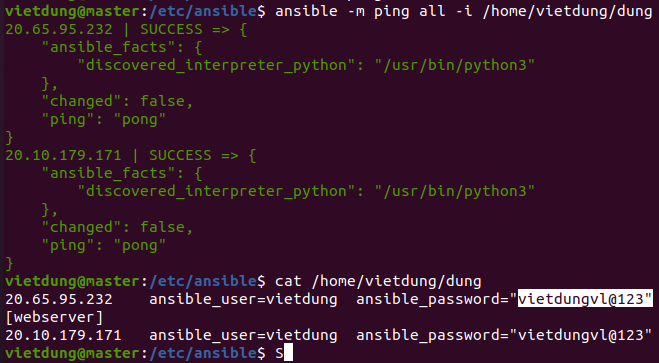
Tuy vậy ansible hỗ trợ 1 kiểu xác thực khác

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

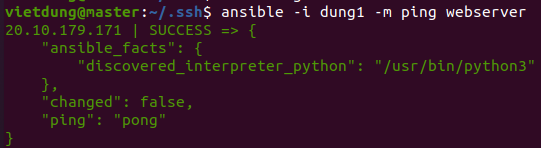
all:

children:

webserver:

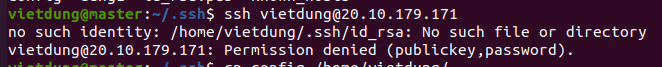
hosts:

20.10.179.171:



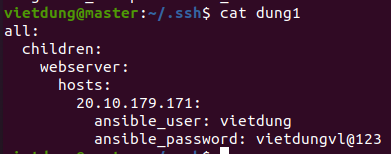
Cái này là với trường hợp đã xác thực với VM qua ssh key

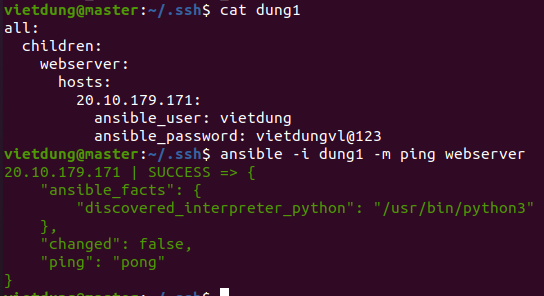
Lưu ý là để ssh tới server mà dùng user/pass thì phải xóa cái file config ở thư mục .ssh/config đi



nano dung1

điền user/pass luôn vào đây



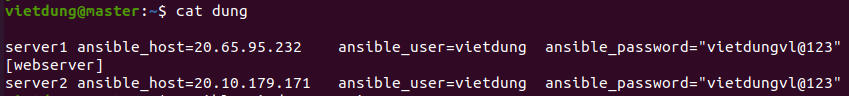


Rồi oke

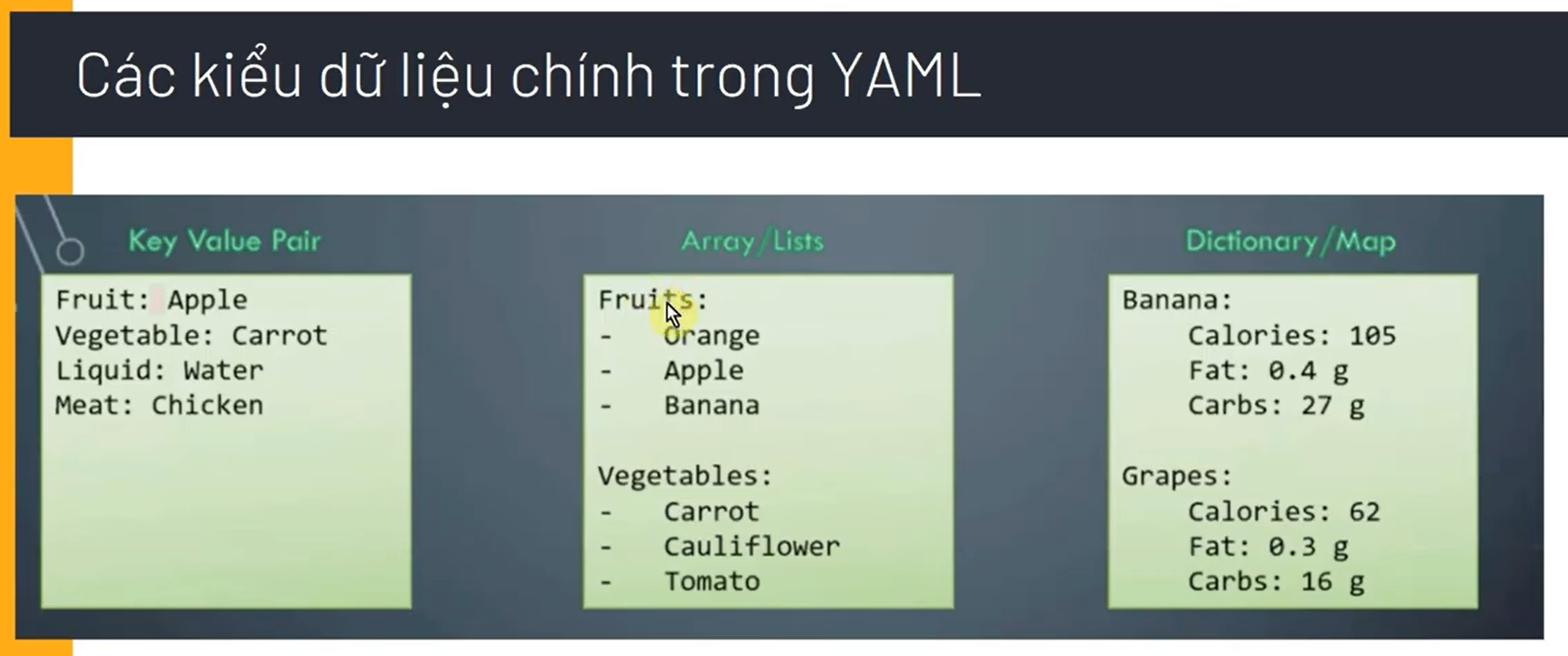
Lưu ý phải là children

Trên là định dạng yaml

* Hoặc ta có thêm định dạng parten như này



# Kiểu dữ liệu



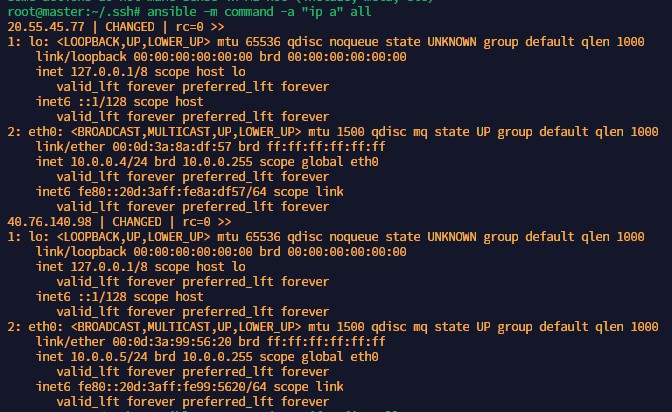
# Playbook, Task, Module

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

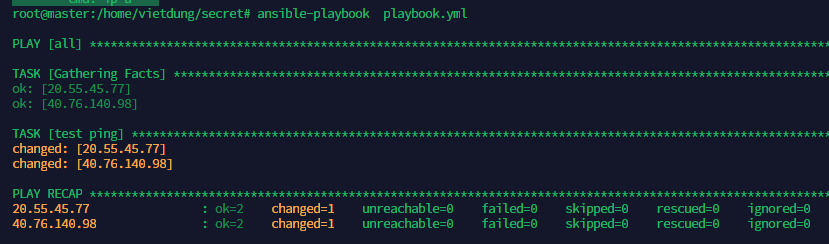
      command:

        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:

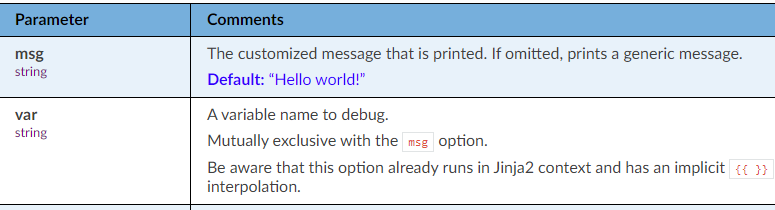
        var: ntp\_server

  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

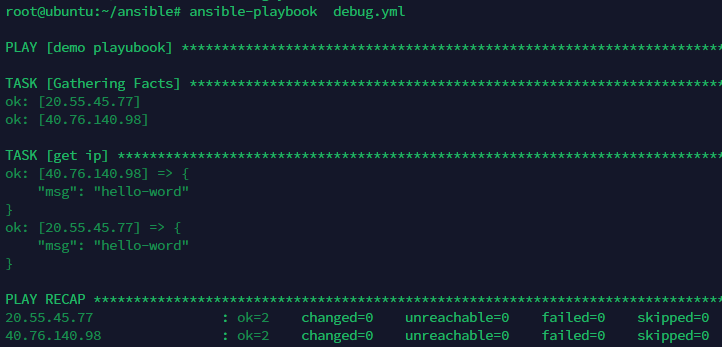
  hosts: all

  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

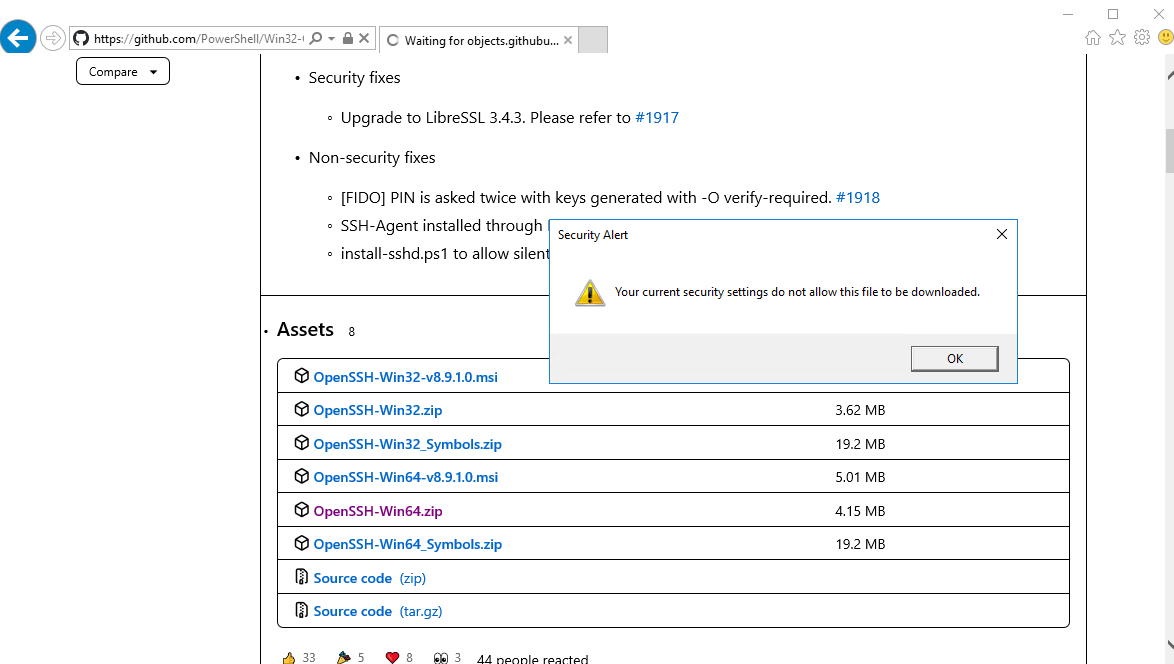
  hosts: all

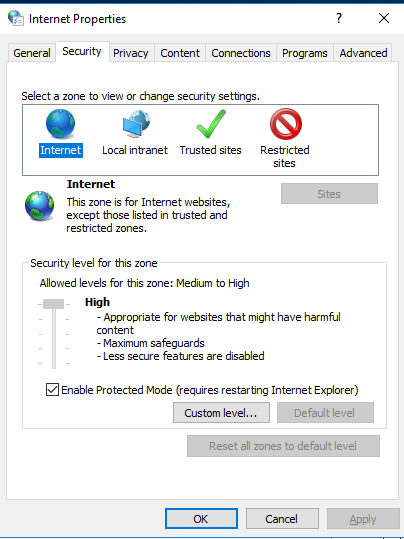
  tasks:

    - name: test ping

      debug:

        var: ansible\_facts





# Template

File playbook

- name: demo

  hosts: all

  become\_user: root

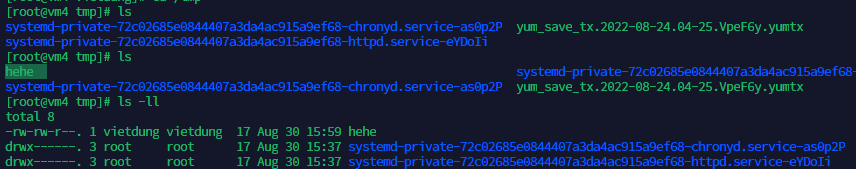
  tasks:

    - 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:

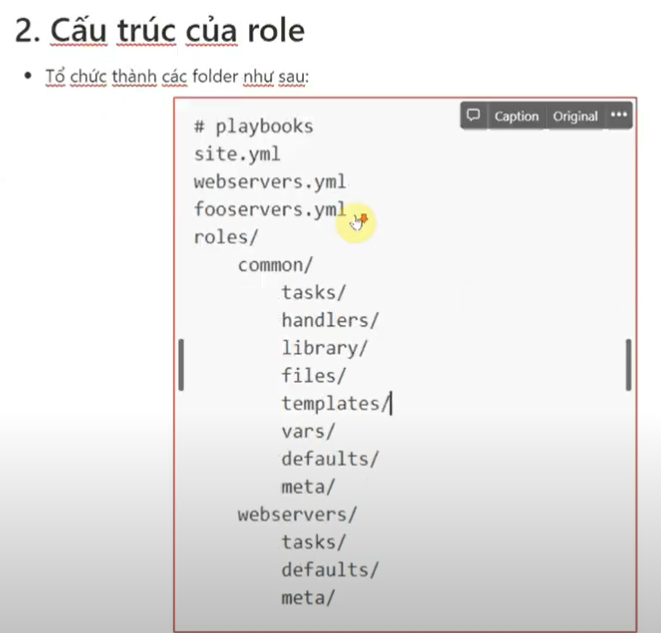
    custome\_variable: demo

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

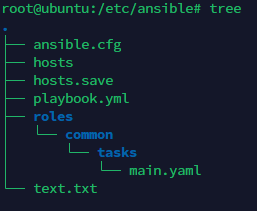
- name: demo test ping

  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

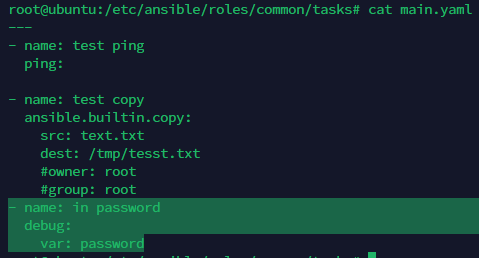
Tạo file folder /vars/main.yaml trong common – role



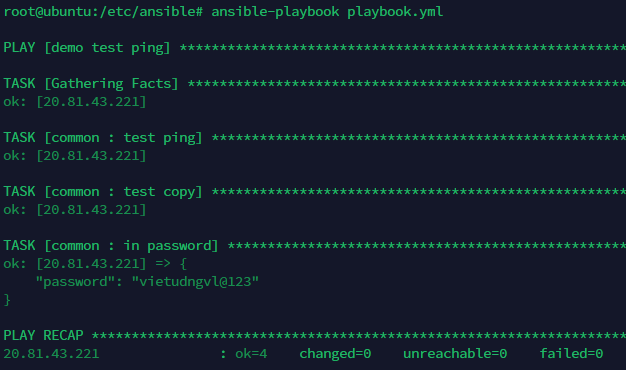
Biến **password** bằng vietdung

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

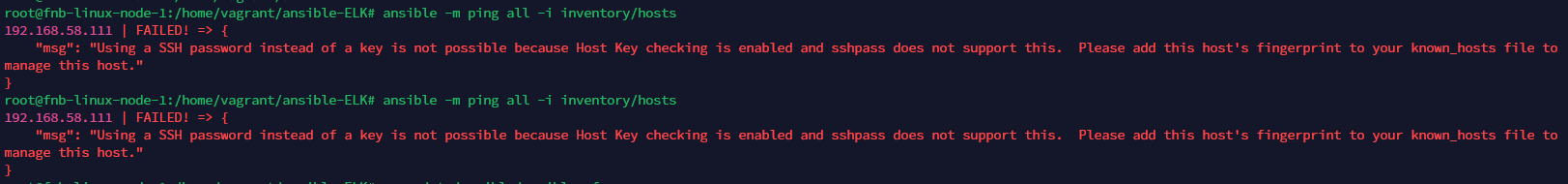
Vào tasks/main.yaml



**Oke chạy**

****

**Lỗi ansible**

****

**"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known\_hosts file to manage this host."**

**Thêm** trongansible/ansible.cfg

[defaults]

host\_key\_checking = false

lưu ý: phải xem cấu hình mặc định của ansible.cfg ở đường dẫn nào

ssh-keyscan -H 192.168.10.11 >> /home/vagrant/.ssh/known\_hosts

# Ansible

**state: present** : đảm bảo trạng thái gói, tiến trình được cài đặt, nếu có n sẽ ko cài nữa

**state: lastest** : có nghĩa là ngoài việc cài đặt, nó sẽ tiếp tục và cập nhật nếu nó không phải là phiên bản mới nhất hiện có.

---

- 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:

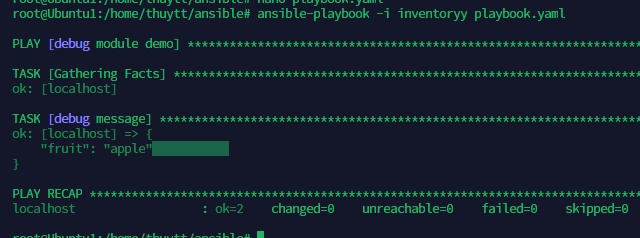
    fruit: "apple"

  tasks:

    - name: debug message

      ansible.builtin.debug:

        var: fruit



3.IN kiểu message + var

---

- name: debug module demo

  hosts: all

  vars:

    fruit: "apple"

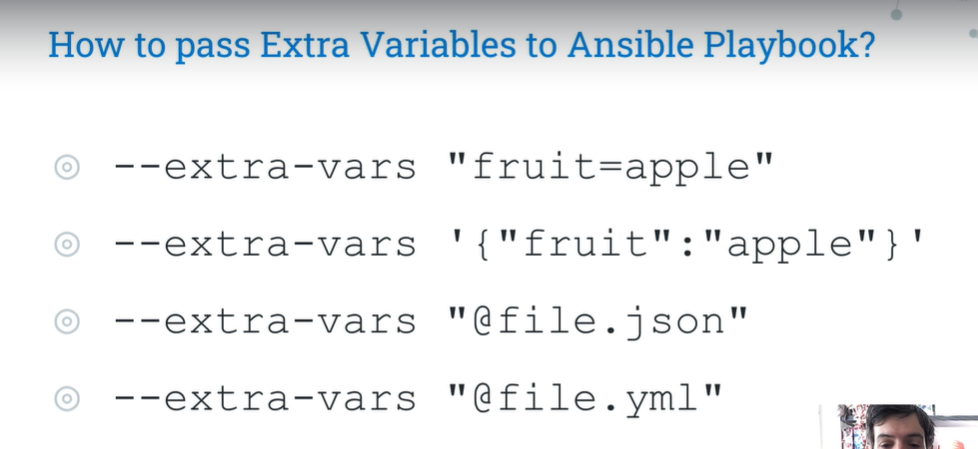
  tasks:

    - 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

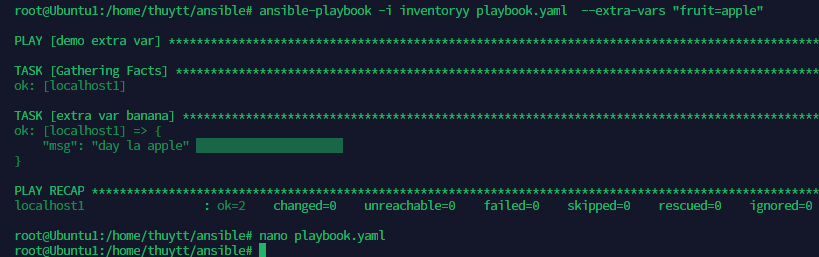
      ansible.builtin.debug:

        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

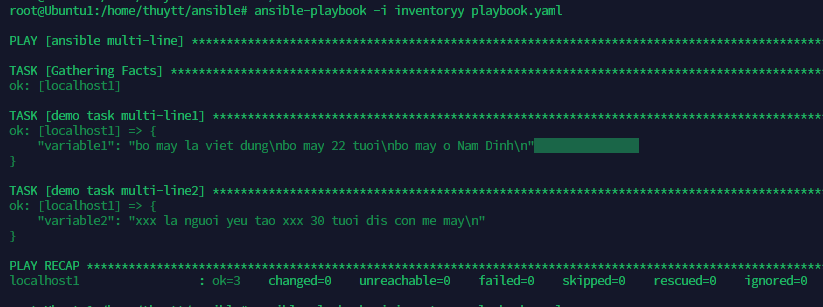
      ansible.builtin.debug:

        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

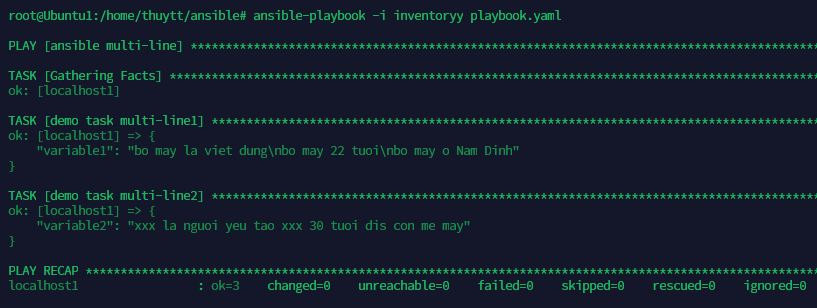
      ansible.builtin.debug:

        var: variable1

    - name: demo task multi-line2

      ansible.builtin.debug:

        var: variable2



Dấu “-” để bỏ \n ở sau, dạng chuẩn

Hoặc

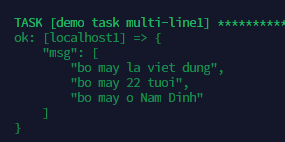
  tasks:

    - 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"

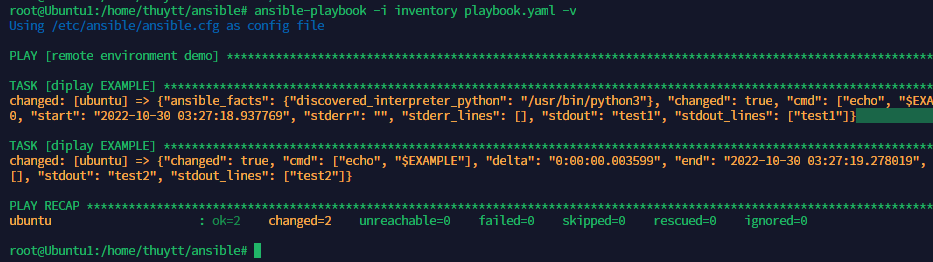
    - name: diplay EXAMPLE

      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:

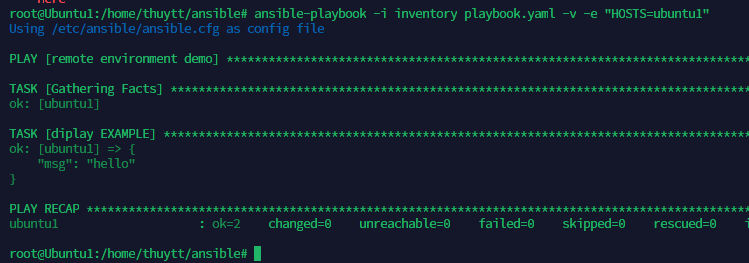
    - name: diplay EXAMPLE

      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 }}"

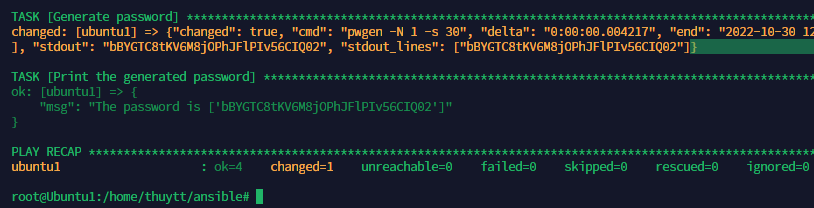


* **cmd** - Lệnh chạy để tạo đầu ra.
* **stdout** - Đầu ra của lệnh.
* **stderr** - Đầu ra lỗi của lệnh.
* **start** - Ngày và giờ khi lệnh bắt đầu được thực thi.
* **end** - Ngày và giờ khi lệnh kết thúc thực thi.
* **delta** - Thời gian chạy lệnh. Đây là sự khác biệt giữa thuộc tính **end** và **start** .
* **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

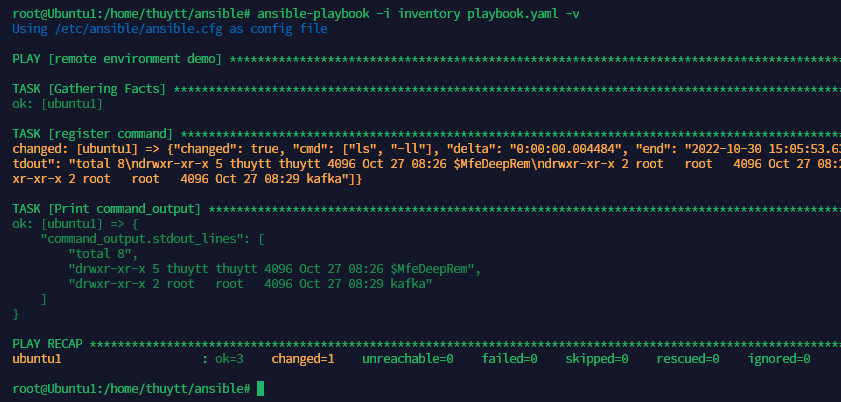
      ansible.builtin.command: 'ls -ll'

      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

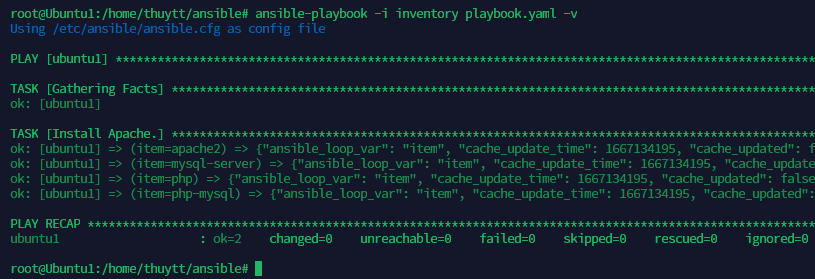
        - php-mysql

        - 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:

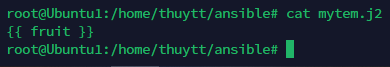
      - name: write to file

        ansible.builtin.template:

          src: "mytem.j2"

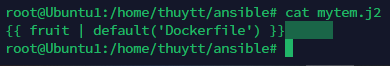
          dest: "{{ output }}"

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

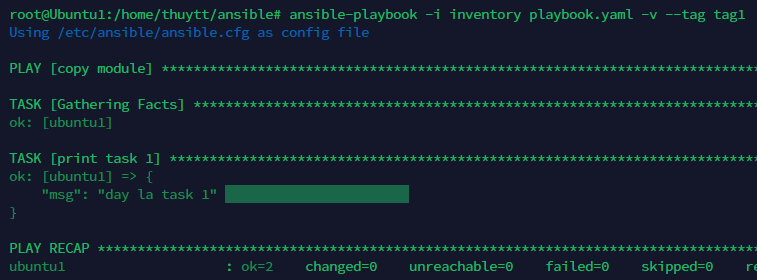
        ansible.builtin.debug:

          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

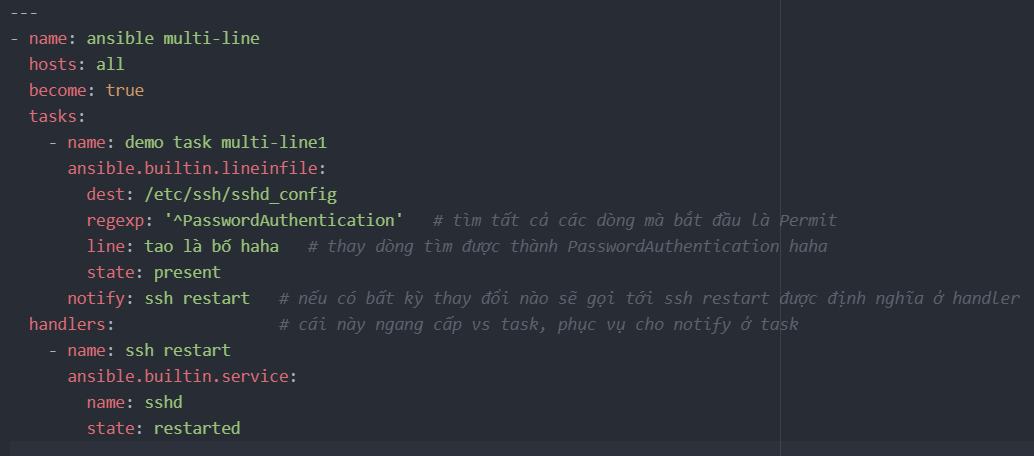
--check, --diff : có thể sử dụng qua command hoặc trong ansible playbook

- --check : là 1 chế độ mô phỏng

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

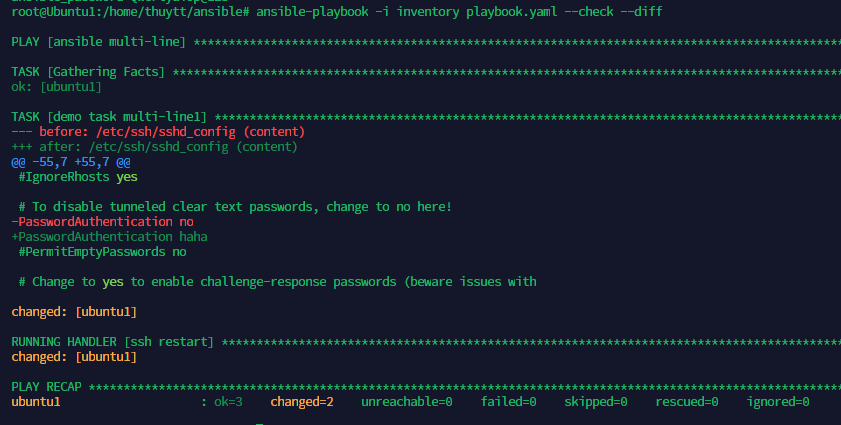
Bài 1: demo thay đổi nội dung file ở target node

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

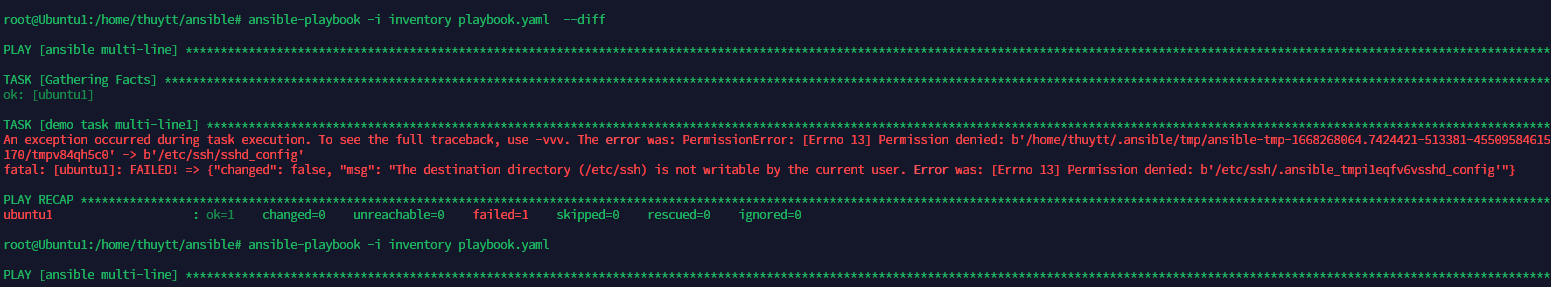


ansible-playbook -i inventory playbook.yaml --check –diff

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

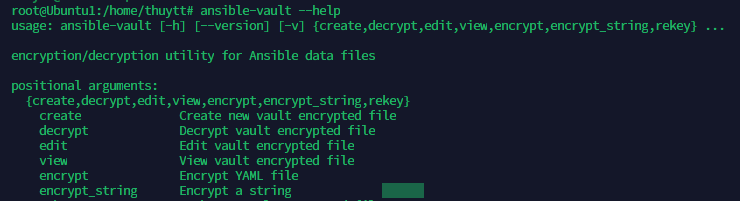
Bay giờ thực thi, bỏ --check đi, --diff sẽ hiện thay đổi, và quá trình dk thực thi

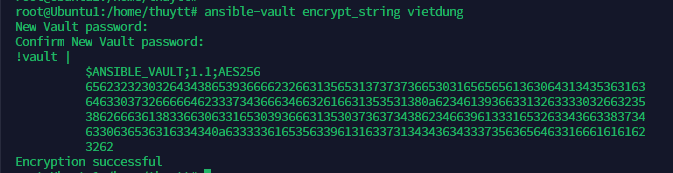
Và thay đổi đã dk thực hiện

## Ansible encryt && decrpt

ansible-vault encrypt +[file]

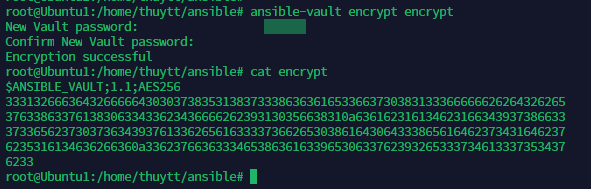
ansible-vault encrypt\_string +[string]



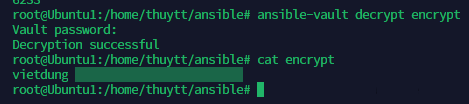


Ansible decrypt

Khi encypt thì n bắt nhập password để encrypt, decrypt



Thì khi decrypt ra ta cũng phải có password này

 nó sẽ over write file

## Ansible blockinfile

---

- name: ansible demo tao file

  hosts: ubuntu2

  become: true

  tasks:

  - name: tao file, noi dung blockfile

    ansible.builtin.blockinfile:

      state: present

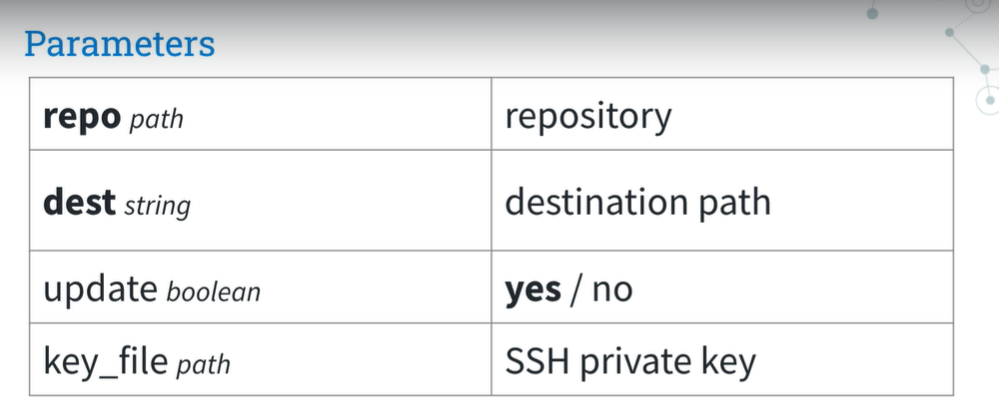
      dest: /etc/hosts

      content:

        vietdung - vietdung123

update 1 content vào 1 file đã tồn tại, các lần sau sẽ ghi đè lên lần trk

## ansible git



Step clone git repo sẽ cần 2 task

* install git
* check clone git

---

- name: check git demo

  hosts: ubuntu2

  become: true # nếu để become ở đây thì cả task sẽ dùng user root

  vars:

    repo: "git@github.com:dungnv84/democi.git"

    dest: "/home/jenkins-ansible"

    sshkey: "~/.ssh/id\_rsa"

  tasks:

    - name: install git  # install git

      ansible.builtin.apt:

        name: git

        update\_cache: true

        state: present

      become: true # become ở đây thì chỉ mỗi task name này sd root

    - name: check git  # clone git repo

      ansible.builtin.git:

        repo: "{{ repo }}"

        dest: "{{ dest }}"

        key\_file: "{{ sshkey }}"

và nó clone git này về ở target host

* check user mà ansible này đang dùng

---

- name: check git demo

  hosts: ubuntu2

  tasks:

    - name: Generate password

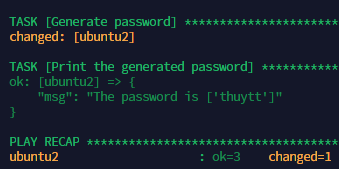
      ansible.builtin.shell: 'whoami'

      register: mypass

    - name: Print the generated password

      ansible.builtin.debug:

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

 đang dùng user **thuytt**

## Ansible copy modules

---

- name: copy module demo

  hosts: all

  become: false

  tasks:

    - name: copy report.txt

      ansible.builtin.copy:

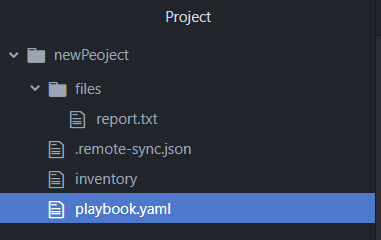
        src: report.txt

        dest: /home/devops/report.txt

        owner: devops

        mode: '0644'

vậy cái src nó sẽ copy file ở đâu, n sẽ lấy ở trong folder /files = playbook.yml của máy ansible



### Copy nhiều file ở 1 folder khác

T sẽ copy ở 1 folder examples

---

- name: copy module demo

  hosts: all

  tasks:

    - name: copy report.txt

      ansible.builtin.copy:

        src: "{{ item }}"

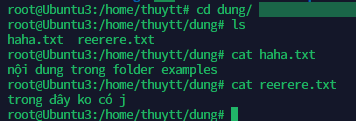
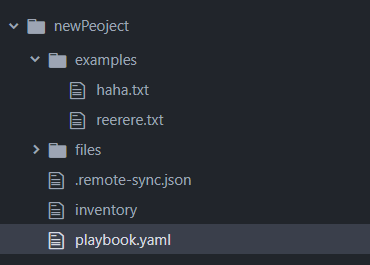
        dest: "/home/thuytt/dung/"

        owner: thuytt

        mode: '0644'

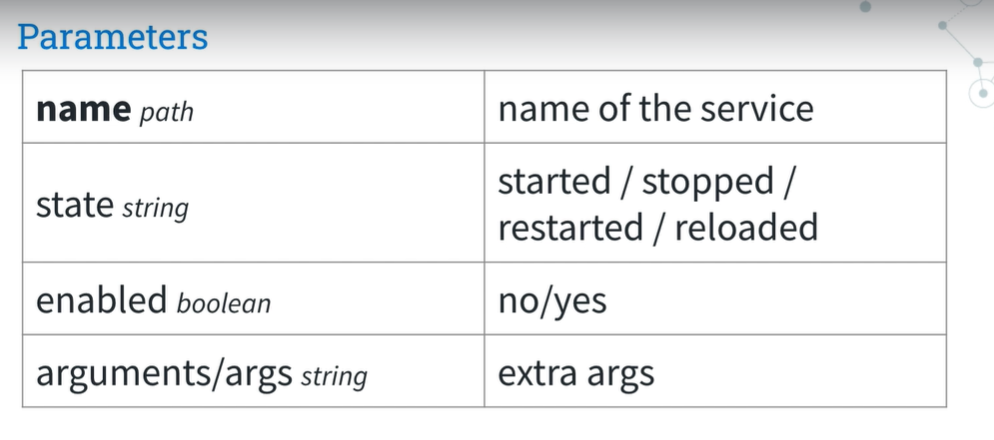
      with\_fileglob: # copy từ 1 folder khác

        - "examples/\*.txt"

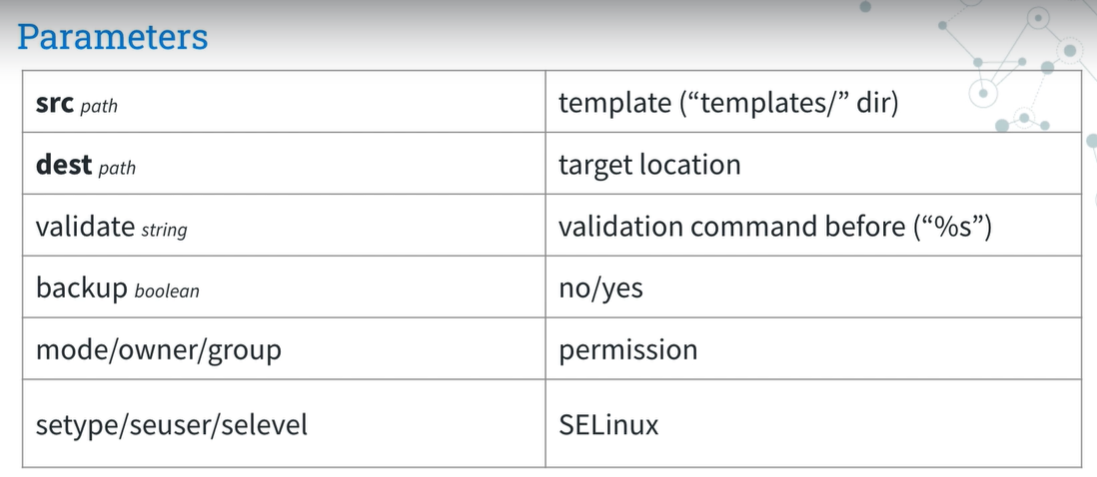


Và máy đích đã có

## Ansible module service



## Ansible template



Phục vụ cho việc update file cấu hình cho service

Ví dụ như file: application.json

---

- name: template module demo

  hosts: all

  become: true

  vars:

    page\_title: "Placeholder"

    page\_description: |

      This is my placeholder page example.

      Multiline is possible ;-)

  tasks:

    - name: install Nginx

      ansible.builtin.apt:

        name: nginx

        state: latest

    - name: apply page template

      ansible.builtin.template:

        src: templates/placeholder.html.j2

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

file **templates/placeholder.html.j2**

<html>

<head>

  <title>{{ page\_title }}</title>

</head>

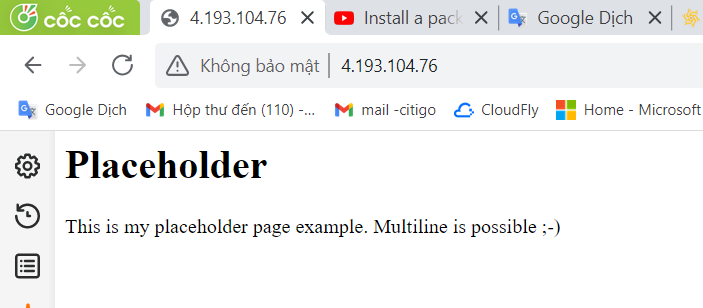
<body>

    <h1>{{ page\_title }}</h1>

    <p>{{ page\_description }}</p> # biến được định nghĩa trong vars

</body>

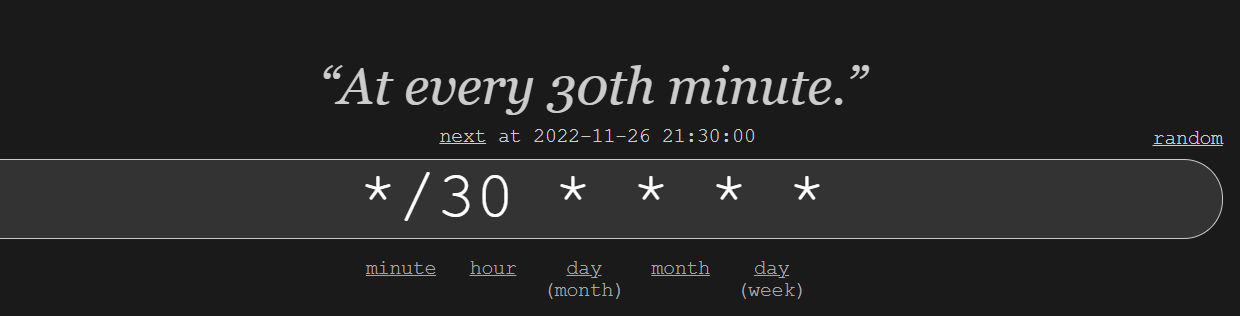
</html>



## Ansible – cronjob

<https://crontab.guru/>

tool để set cronjob



30 phút 1 lần nè

## hostvars[host]['ansible\_host']

---

- name: template module demo

  hosts: all

  become: true

  tasks:

    - name: generate /etc/myhosts file

      ansible.builtin.template:

        src: templates/hosts.j2

        dest: /etc/myhosts

        owner: root

        group: root

        mode: '0644'

# {{ ansible\_managed }}

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4

::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

{% start for print %}

{{ hostvars[host]['ansible\_host'] }}  {{ hostvars[host]['inventory\_hostname'] }} {{ hostvars[host][ 'ansible\_user'] }}

{% endfor %}

{% ... %} và {% ... %} : như là 1 cái print trong file template cho ansible

hostvars là 1 key của inventory

hostvars[host] : truy xuất từng thành phần trong inventory

hostvars[host]['ansible\_host'] : in ra cái ansible\_host của từng cái đó

Nếu ta có 1 inventory

[test1]

test-1 ansible\_host=abc.def.ghi.jkl ansible\_port=1212

[test2]

test2-1 ansible\_host=abc.def.ghi.mno ansible\_port=1212

[test3]

test3-1 ansible\_host=abc.def.ghi.pqr ansible\_port=1212

{{ hostvars['test-1'].ansible\_host }} = abc.def.ghi.jkl

Mà với module template, nếu dest ko có folder, file thì n sẽ tạo ra

## lookup

---

- name: template module demo

  hosts: all

  become: true

  vars:

    content: "{{ lookup('file', 'example.txt') }}"   # tìm nội dung là file có tên là  example.txt

  tasks:

  - name: demo

    ansible.builtin.debug:

      msg: "noi dung trong file la  {{ content }}"

---

- name: environment demo

  hosts: all

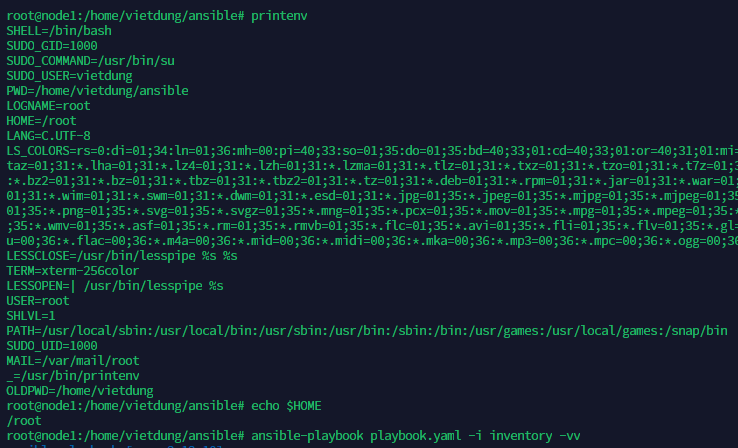
  tasks:

    - name: display HOME

      ansible.builtin.debug:

        msg: "{{ lookup('env', 'HOME') }}"

check env HOME = echo $HOME



File json

---

- name: json read demo

  hosts: localhost

  vars:

    jsondata: "{{ lookup('file', 'example.json') | from\_json }}"

  tasks:

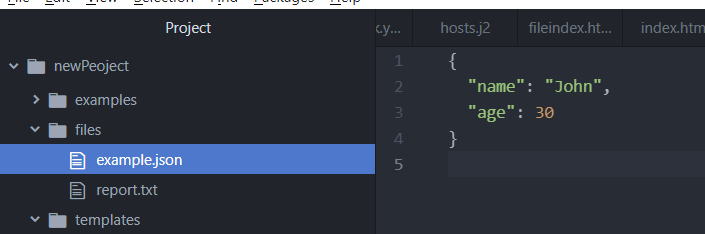
    - name: Print variable

      ansible.builtin.debug:

        var: jsondata

in ra giá trị trong file json

phần from\_json cho thêm vào cho n chi tiết



File json tồn tại trong folder file



## Ansible download file

ansible.builtin.get\_url

download file from HTTP, HTTPS, or FTP to node

download elasticsearch 8.4.3

<https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.4.3-linux-x86_64.tar.gz>

---

- name: download get\_url

  hosts: all

  become: false

  vars:

    myurl: "https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.4.3-linux-x86\_64.tar.gz"

    myurc: "sha256:https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.4.3-linux-x86\_64.tar.gz.sha"

    mydest: /home/vietdung

  tasks:

    - name: downloads

      ansible.builtin.get\_url:

        url: "{{ myurl }}"

        dest: "{{ mydest }}"

        mode: '0644'

        owner: vietdung

        group: vietdung

## With\_items

---

- name: set environment demo

  hosts: all

  gather\_facts: false

  become: true

  vars:

    os\_environment:

      - key: EDITOR

        value: vi

      - key: MY\_ENV\_VARIABLE

        value: ansiblepilot

  tasks:

    - name: customize /etc/environment

      ansible.builtin.lineinfile:

        dest: "/etc/environment"

        state: present

        regexp: "^{{ item.key }}="

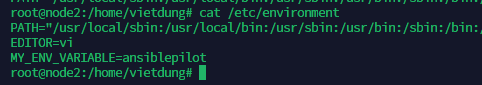
        line: "{{ item.key }}={{ item.value }}"

      with\_items: "{{ os\_environment }}"

with\_items: "{{ os\_environment }}" : tức là các thành phần trong

var: os\_environment sẽ là 1 item của task này

item.key , item.value



## Ansible When

When cho **register**

* is defined
* is succeeded
* is failed
* is skipped

---

- name: remote environment demo

  hosts: elk

  become: True

  tasks:

  - name: Register a variable, ignore errors and continue

    command: echo failed

    register: result

    ignore\_errors: true

  - name: Run only if the task that registered the "result" variable fails

    command: echo failed

    when: result is failed

  - name: Run only if the task that registered the "result" variable succeeds

    command: echo succeeded

    when: result is succeeded

  - name: Run only if the task that registered the "result" variable is skipped

    command: echo defined

    when: result is defined

Biến Vars

* is defined
* is undefined
* is not defined

check thông tin target server

---

- name: remote environment demo

  hosts: elk

  vars:

    epic: 2

  tasks:

  - name: Run only if the task that registered the "result" variable is skipped

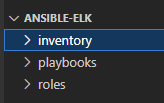
    command: echo defined

    when: epic is undefined

# Ansible ELK

## Role Install apache2

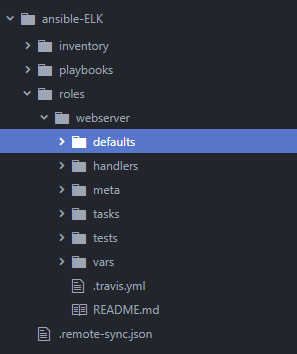
T có 1 dự án cho ansible-ELK, t tạo ra các folder như này trước



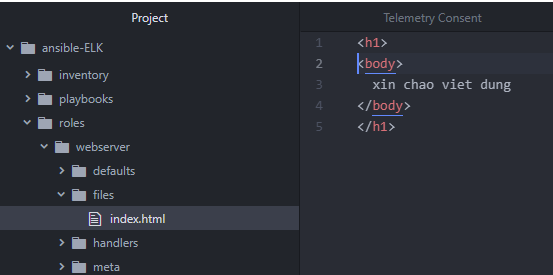
Bgio cd vào roles để tạo 1 role

Dùng ansible-galaxy để có 1 cấu trúc role

ansible-galaxy init webserver

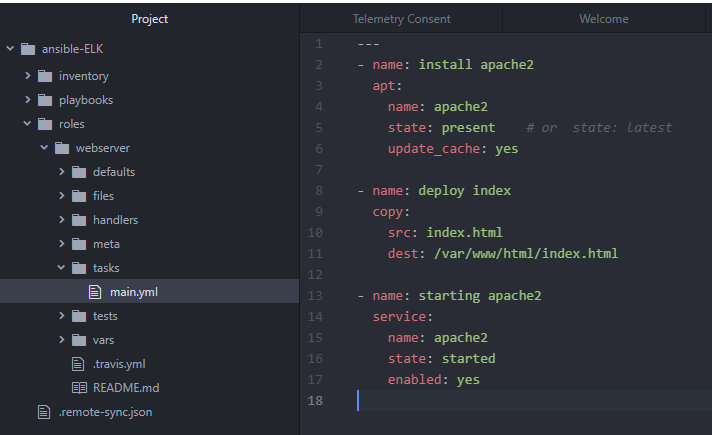


Ta sẽ tạo ra 1 cấu trúc role cho 1 cái chạy webserver (1 project webserver )



B1: Tạo file index.html trong folder files

B2: tạo task cho webserver này



---

- name: install apache2

  apt:

    name: apache2

    state: present    # or  state: latest

    update\_cache: yes

- name: deploy index

  copy:

    src: index.html

    dest: /var/www/html/index.html

- name: starting apache2

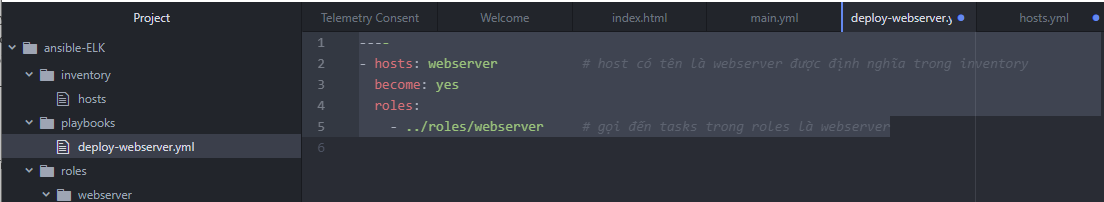
  service:

    name: apache2

    state: started

    enabled: yes

B3: tạo trong playbooks chính để thực thi



---

- hosts: webserver           # host có tên là webserver được định nghĩa trong inventory

  become: yes

  roles:

    - ../roles/webserver     # gọi đến tasks trong roles là webserver

B4: chạy

ansible-playbook -i inventory/hosts.yml playbooks/deploy-webserver.yml

playbooks: có 2 nhiệm vụ

* deploy lên host nào
* chạy task trong role nào

----

- hosts: webserver           # host có tên là webserver được định nghĩa trong inventory

  become: yes

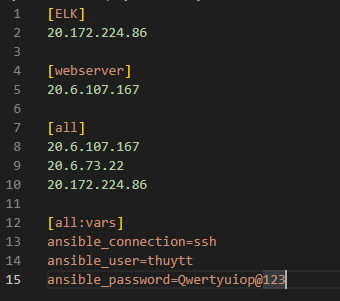
  roles:

    - ../roles/webserver     # gọi đến tasks trong roles là webserver

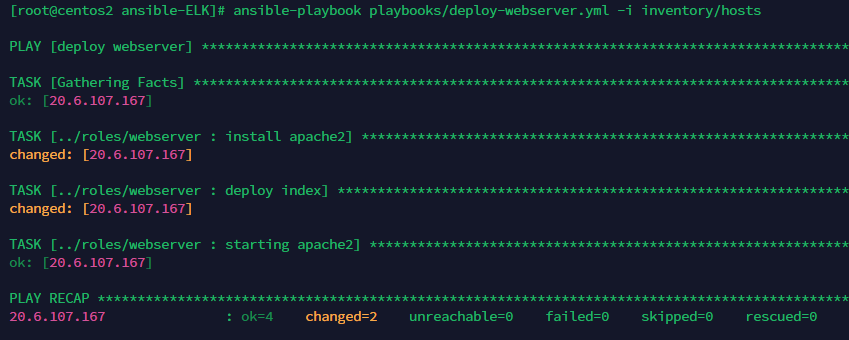
Còn inventory

[webserver]

20.6.107.167



Kqua:





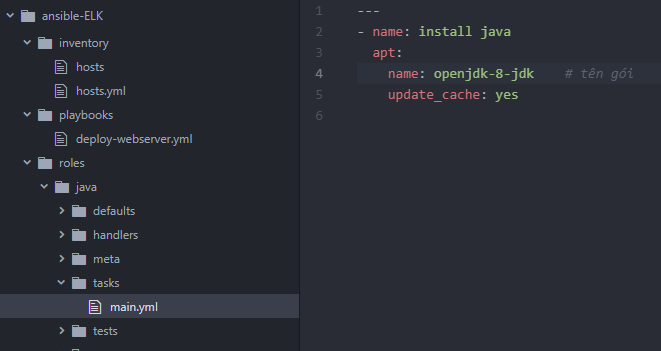
## Role Install Java

Tạo thêm 1 role **java** cho install java

[root@centos2 roles]# ansible-galaxy init java

B1:

Oke giờ vào role của java để tạo tasks



B2: vào playbook để thực hiện việc gọi đến role này

Ta cứ coi role chứa các gói cài, thư viện deploy

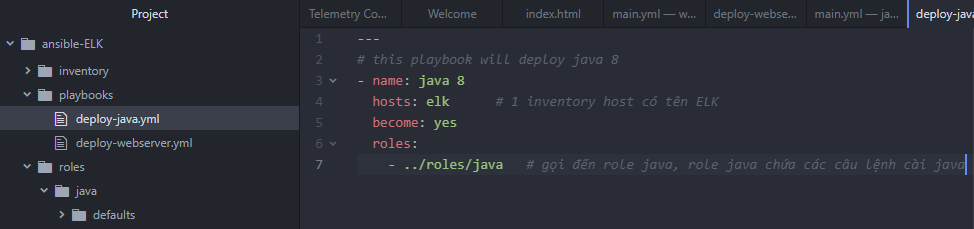
Còn playbook là ta muốn lấy gói cài nào trong role và deploy lên con server nào

1 task để deploy java đã xog

Giờ tạo 1 file playbook để gọi đến task này và deploy lên server nào

* role task: chưa câu lệnh cài thôi

playbooks/deploy-java.yml



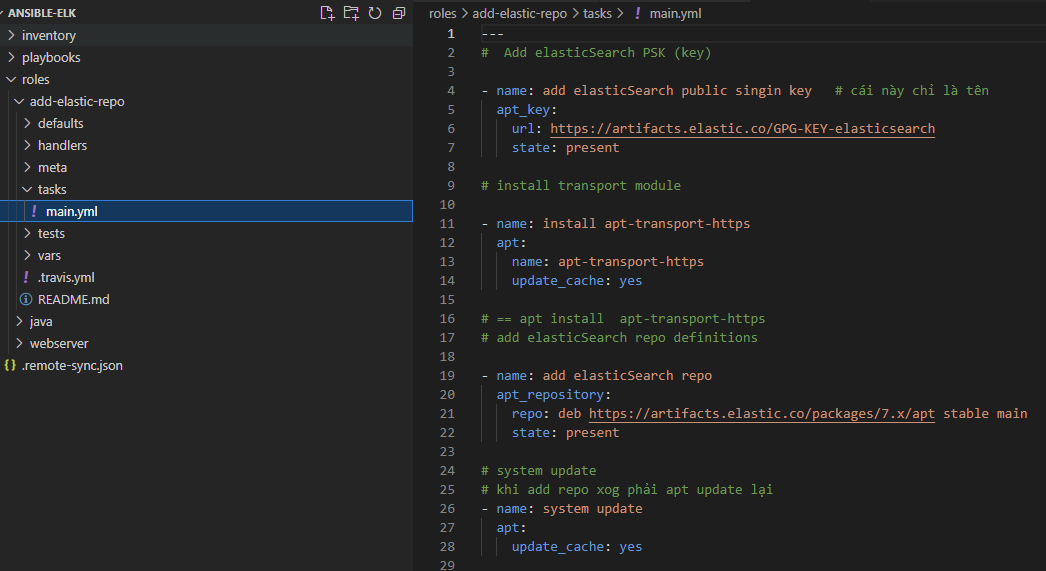
Oke giờ chạy ansible để n deploy java lên server

## Role add repo key for ELK

Role: **add-elastic-repo**

ansible-galaxy init add-elastic-repo

oke lại vào task của Role: **add-elastic-repo** thực hiện nhiệm vụ nào



---

#  Add elasticSearch PSK (key)

- name: add elasticSearch public singin key   # cái này chỉ là tên

  apt\_key:

    url: https://artifacts.elastic.co/GPG-KEY-elasticsearch

    state: present

# install transport module

- name: install apt-transport-https

  apt:

    name: apt-transport-https

    update\_cache: yes

# == apt install  apt-transport-https

# add elasticSearch repo definitions

- name: add elasticSearch repo

  apt\_repository:

    repo: deb https://artifacts.elastic.co/packages/7.x/apt stable main

    state: present

# system update

# khi add repo xog phải apt update lại

- name: system update

  apt:

    update\_cache: yes

Nhưng vấn đề là khi cái url này thay đổi, ta lại phải mất công vào đây để update

Ta sẽ ko làm thế, mà tôi sẽ update biến cho nó

Vào default/main.yml của role này để update biến cho nó



Vào trong file task role

---

#  Add elasticSearch PSK (key)

- name: add elasticSearch public singin key   # cái này chỉ là tên

  apt\_key:

    url: "{{ elastic\_psk\_url }}"      # biến này gọi ở defaults/main.yml

    state: present

# install transport module

- name: install apt-transport-https

  apt:

    name: apt-transport-https

    update\_cache: yes

# == apt install  apt-transport-https

# add elasticSearch repo definitions

- name: add elasticSearch repo

  apt\_repository:

    repo: "{{ elastic\_repo }}"

    state: present

# system update

# khi add repo xog phải apt update lại

- name: system update

  apt:

    update\_cache: yes

**vì vậy, nếu có thay đổi về version, url thì chỉ có giá trị trong biến là thay đổi, còn lại mọi thứ được định nghĩa trong file task này là ko cần thay đổi j**

Rồi oke, bgio tạo playbook cho n thực hiện

## Role install elasticSearch

ansible-galaxy init elasticSearch

copy file /etc/elasticsearch/elasticsearch.yml vào folder **template** của role elasticSearch

Nếu ae nào dùng atom mà download folder về mà ko thấy folder template trong role thì là do trong folder template đó ko có 1 file nào nên atom n ko pull từ server về

Vậy ta chỉ cần cp luôn file /etc/elasticsearch/elasticsearch.yml vào trong template

Cái folder template == file config của role-module đó

Rồi update thông tin file bằng biến

cluster.name: "{{ cluster\_name }}"

node.name: "{{ node\_name }}"

path.data: "{{ path\_data }}"

path.logs: "{{ path\_data }}"

network.host: {{ network\_host }}

http.port: "{{ http\_port }}"

discovery.type: {{ discovery\_type }}

biến được định nghĩa trong folder default

B2: viết tasks/main.yml để install ELK

---

# install elasticSearch

- name: install elasticSearch

  apt:

    name: elasticSearch

    state: present

# copy file config cho elasticsearch

- name: copy file elasticsearch.yml

  templates:

    src: elasticsearch.yml

    dest: /etc/elasticsearch/elasticsearch.yml

# start service elasticSearch

- name: start elasticsearch

  service:

    name: elasticsearch

    state: started

    enable: yes

oke coi như đã xog phần install elasticSearch

B3: update vào file playbook để deploy

File playbook sẽ có luôn 3 nhiệm vụ

* install java
* add repo
* install elastic

playbooks/deploy-elk.yml

---

# this playbook will deploy java 8

- name: java 8

  hosts: elk      # 1 inventory host có tên ELK , tên hosts được install

  become: yes

  roles:

    - ../roles/java   # gọi đến role java, role java chứa các câu lệnh cài java

# add repo for elk

- name: add repo ELK && install elastic

  hosts: elk

  become: yes

  roles:

    - ../roles/add-elastic-repo # role để add repo

    - ../roles/elasticSearch # role để thực hiện các câu lệnh install

Oke **hosts:elk**

Giờ muốn install elk trên host nào, ta chỉ cần update lại cái IP cho hostsname elk trong file inventory là được

Nhanh ko

## Sercurity kibana, elastic



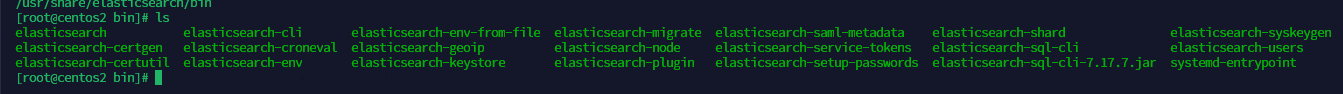
Vào file **/etc/elasticsearch/elasticsearch.yml** thêm dòng

xpack.security.enabled: true

giờ restart lại elasticsearch

* gen password

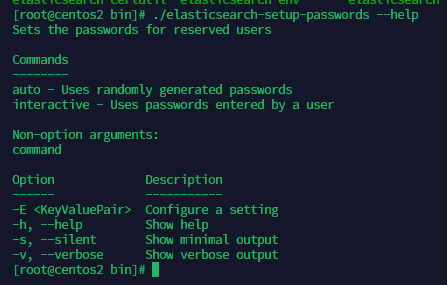
cd /usr/share/elasticsearch/bin



Có 1 số module ở đây, ta quan tâm tới

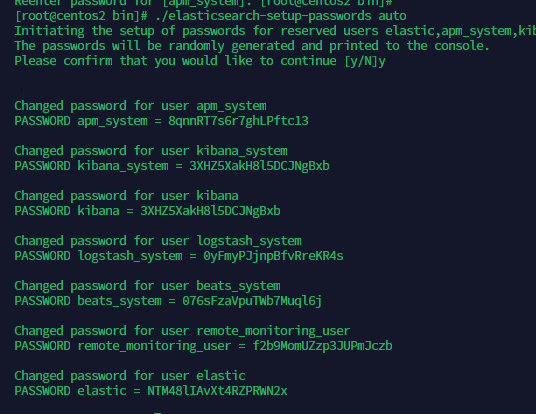
elasticsearch-setup-passwords

có 2 cơ chế để tạo password cho việc credential elasticSearch và kibana



Auto và nhập

Tôi sẽ dùng luôn random



Changed password for user apm\_system

PASSWORD apm\_system = 8qnnRT7s6r7ghLPftc13

Changed password for user kibana\_system

PASSWORD kibana\_system = 3XHZ5XakH8l5DCJNgBxb

Changed password for user kibana

PASSWORD kibana = 3XHZ5XakH8l5DCJNgBxb

Changed password for user logstash\_system

PASSWORD logstash\_system = 0yFmyPJjnpBfvRreKR4s

Changed password for user beats\_system

PASSWORD beats\_system = 076sFzaVpuTWb7Muql6j

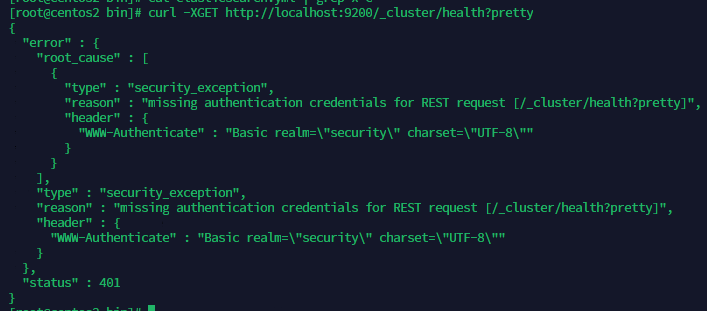
Changed password for user remote\_monitoring\_user

PASSWORD remote\_monitoring\_user = f2b9MomUZzp3JUPmJczb

Changed password for user elastic

PASSWORD elastic = NTM48lIAvXt4RZPRWN2x

* bgio sẽ thử curl đến elasticSearch và n đã yêu cầu authen



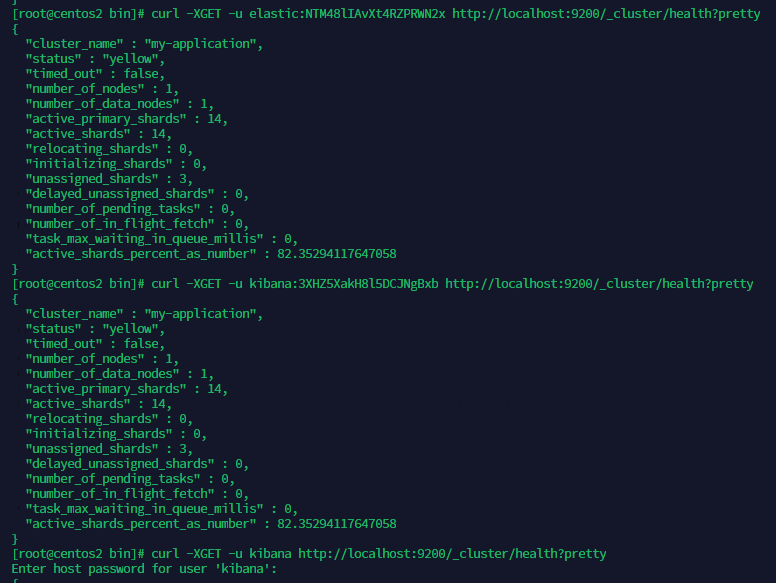
Tôi sẽ dùng 1 số user trên để authen như

* user: elastic

curl -XGET -u elastic:NTM48lIAvXt4RZPRWN2x <http://localhost:9200/_cluster/health?pretty>

* user: kibana

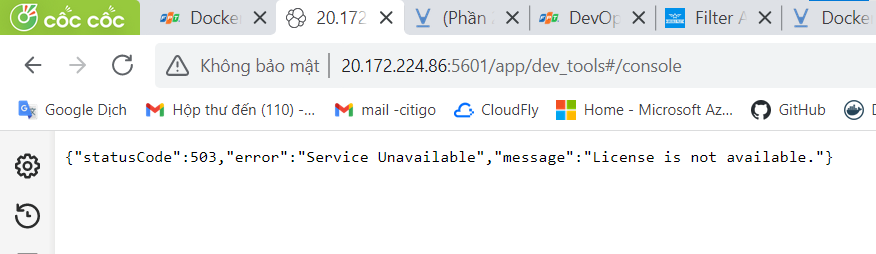
curl -XGET -u kibana:3XHZ5XakH8l5DCJNgBxb <http://localhost:9200/_cluster/health?pretty>



Oke

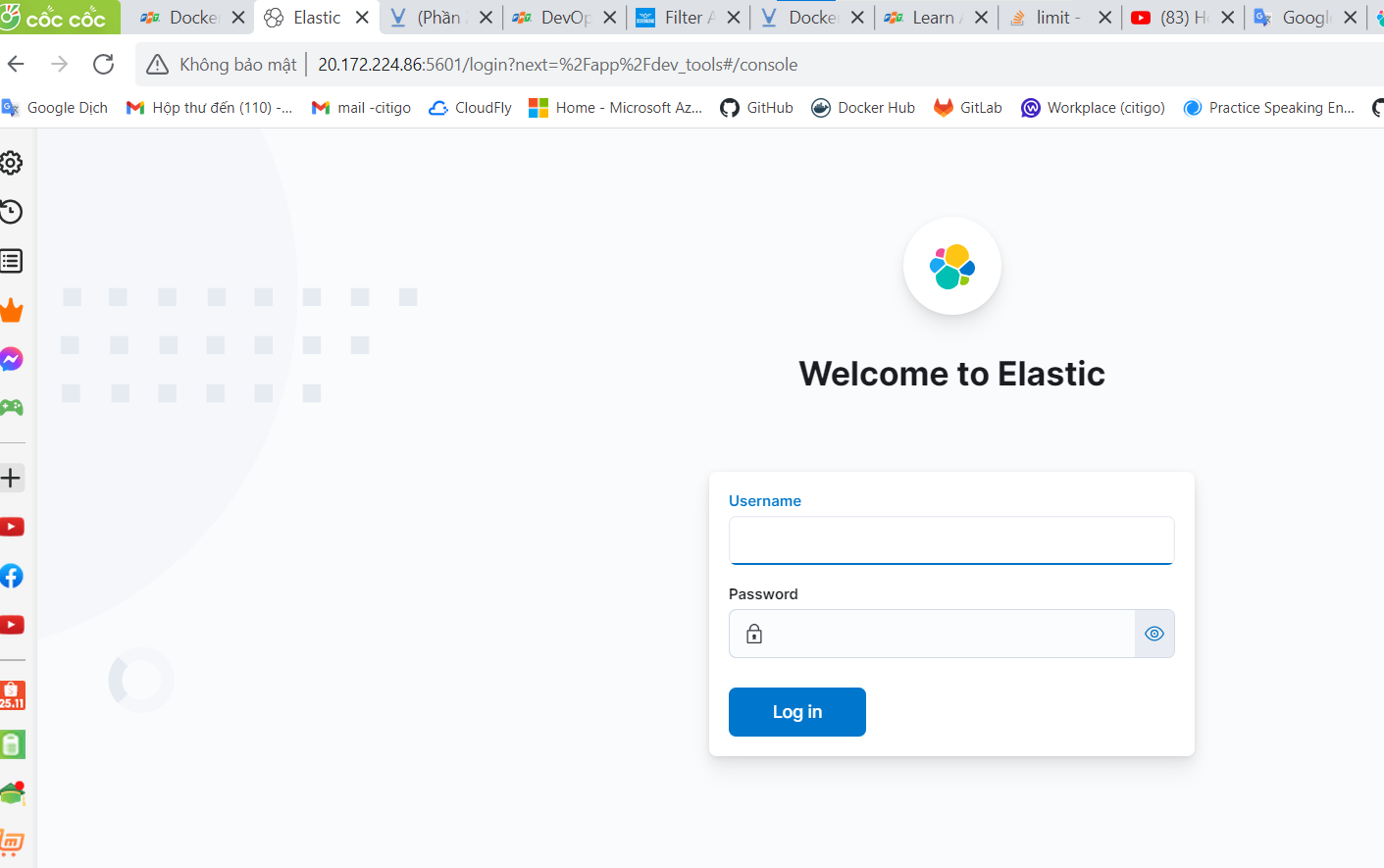
* Giờ config authen cho kibana tới elastic

Hiện tại n chưa được authen



nano /etc/kibana/kibana.yml

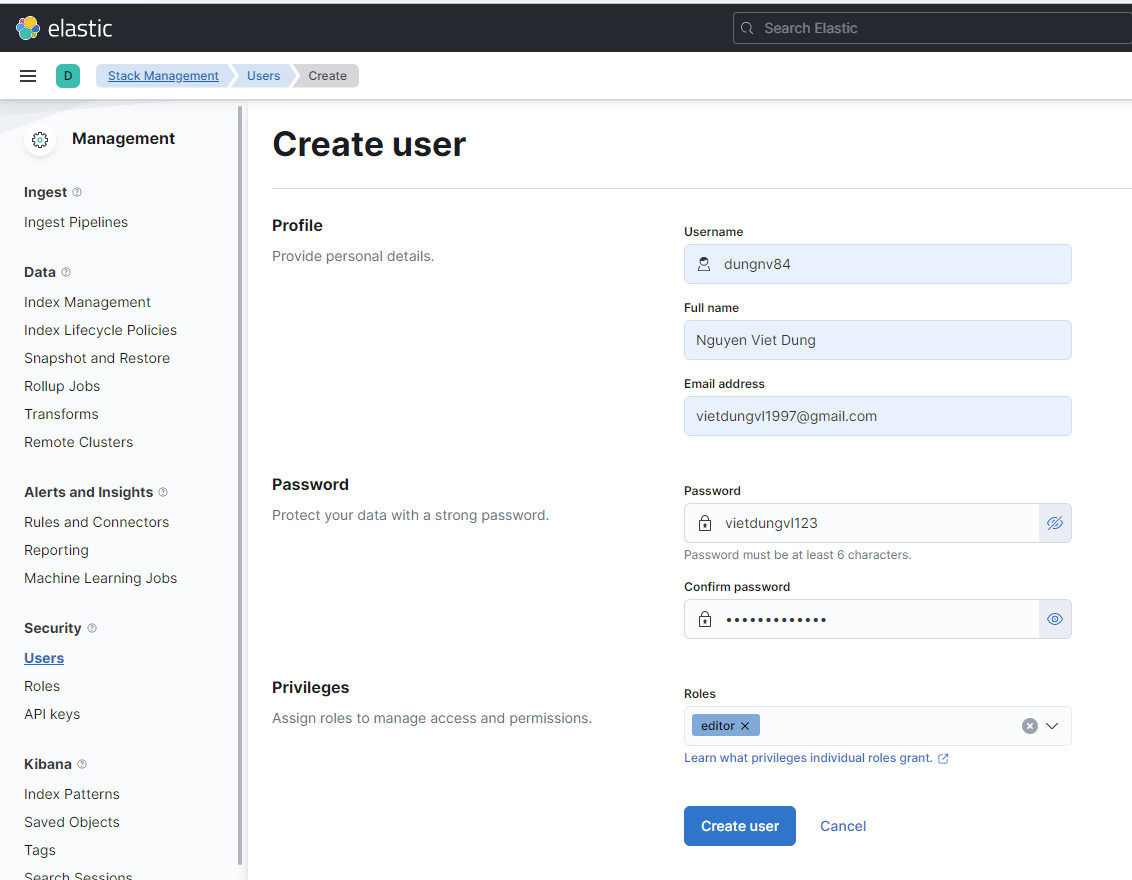




Nhập user/pass của elastic vòa

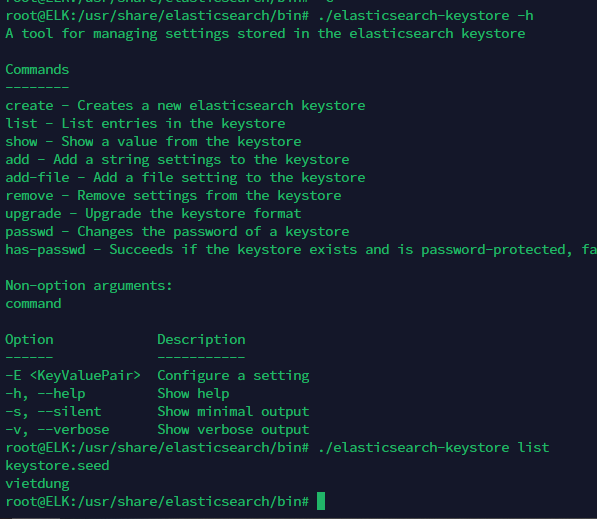
* Tạo user

Manager -> user -> create



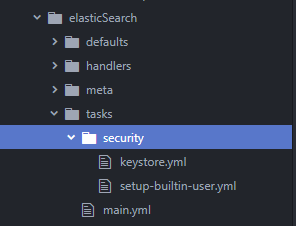
## Setup KeyStore and bootrap password

Tạo 1 kho chứa key



Viết lại role cho elasticseach

* Tệp 1: install elasticsearch và cấu hình elastic
* Tệp 2: check elastic tồn tại ko và thiết lập mật khẩu boottrap
* Tệp 3: tạo mật khẩu và API người dùng
* Tạo 1 thư mục mới trong task là security

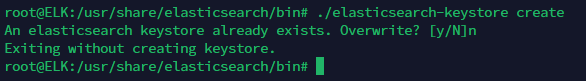


Vậy trong main ta viết rule để gọi đến keystore

# Xpack sercurity configuration  -  task sercurity

- name: including setup xpack security configuration

  include: security/keystore.yml



Chỉ có 1 key store

# Automation ansible for ELK

Để cài ELK ta cần java

* Tạo role cài java
* Tạo role install elasticsearch
  + Install Elasticsearch (file config cho ELK để ở template role elastic)
  + Bật tính năng authen cho elastic (cần file vars để lưu user/pass)
* Viết playbook chạy 2 cái này để deploy ELK

1. Tạo role java oke rồi
2. Install elastichsearch role

Main.yml trong role elk

---

# tasks file for elasticsearch

# Installing Elasticsearch

- name: Installing Elasticsearch

  apt:

   name: elasticsearch

# Replace default elasticsearch.yml

- name: Replace default elasticsearch.yml

  template:

   src: elasticsearch.yml

   dest: /etc/elasticsearch/elasticsearch.yml

# Start Elasticsearch service

- name: Starting Elasticsearch

  service:

   name: elasticsearch

   state: started

   enabled: yes

# Xpack security configuration

# bắt đầu gọi đến task sercurity trong ELK để bật tính năng authen

- name: Including xpack security setup configuration

  include: security/keystore.yml

tạo 1 folder security/keystore.yml cho việc tạo user/pass cho elk lưu trong keystore

nhiệm vụ là tạo 1 keystore để lưu user/password

---

# ----------- Create KeyStore ----------

- name: create elasticsearch keystore

  become: yes

  command: >

   {{ es\_home }}/bin/elasticsearch-keystore create

  args:

   creates: "{{ es\_conf\_dir }}/elasticsearch.keystore"

  environment:

   ES\_PATH\_CONF: "{{ es\_conf\_dir }}"

#  {{ es\_home }}/bin/elasticsearch-keystore = /usr/share/elasticsearch/bin/elasticsearch-keystore

- name: Set elasticsearch keystore permissions

  become: yes

  file: state=file path={{ es\_conf\_dir }}/elasticsearch.keystore owner={{ es\_owner }} group={{ es\_group }} mode={{ es\_mode }}

- name: Check if elasticsearch keystore is setup

  become: yes

  command: >

   {{ es\_home }}/bin/elasticsearch-keystore list

  register: list\_keystore

  changed\_when: False

  environment:

    ES\_PATH\_CONF: " {{ es\_conf\_dir }} "

  check\_mode: no

- name: Create bootstrap password for elastic user

  become: yes

  shell: echo "{{ es\_api\_basic\_auth\_password }}" | {{ es\_home }}/bin/elasticsearch-keystore add -x 'bootstrap.password'

  when:

   - es\_api\_basic\_auth\_username is defined and  list\_keystore is defined and es\_api\_basic\_auth\_username == 'elastic' and 'bootstrap.password' not in list\_keystore.stdout\_lines

  environment:

    ES\_PATH\_CONF: " {{ es\_conf\_dir }} "

  no\_log: true

# ------------including builtin user setup --------------

# lại phi tới file setup-built-in-user.yml

- name: Including builtin user setup

  include: setup-built-in-user.yml

oke lại tạo 1 file **setup-built-in-user.yml**

cập nhật user/pass

---

- name: Restarting Elasticsearch

  service:

   name: elasticsearch

   state: restarted

#------------------------------- Setup Built-in User passwords-----------

- name: Update elastic user password

  uri:

   url: "{{ es\_api\_uri }}/{{ es\_security\_api }}/user/{{es\_api\_basic\_auth\_username}}/\_password"

   method: POST

   body\_format: json

   body: "{ \"password\":\"{{ elastic\_password }}\" }"

   status\_code: 200

   user: "{{es\_api\_basic\_auth\_username}}"

   password: "{{es\_api\_basic\_auth\_password}}"

   force\_basic\_auth: "{{ es\_force\_basic\_auth }}"

   validate\_certs: "{{ es\_validate\_certs }}"

- name: Update Kibana user password

  uri:

   url: "{{ es\_api\_uri }}/{{ es\_security\_api }}/user/{{ kibana\_username }}/\_password"

   method: POST

   body\_format: json

   body: "{ \"password\":\"{{ kibana\_password }}\" }"

   status\_code: 200

   user: "{{es\_api\_basic\_auth\_username}}"

   password: "{{ elastic\_password }}"

   force\_basic\_auth: "{{ es\_force\_basic\_auth }}"

   validate\_certs: "{{ es\_validate\_certs }}"

- name: Update logstash\_system user password

  uri:

   url: "{{ es\_api\_uri }}/{{ es\_security\_api }}/user/{{ logstash\_system\_username }}/\_password"

   method: POST

   body\_format: json

   body: "{ \"password\":\"{{ logstash\_system\_password }}\" }"

   status\_code: 200

   user: "{{es\_api\_basic\_auth\_username}}"

   password: "{{ elastic\_password }}"

   force\_basic\_auth: "{{ es\_force\_basic\_auth }}"

   validate\_certs: "{{ es\_validate\_certs }}"

- name: Update beats\_system user password

  uri:

   url: "{{ es\_api\_uri }}/{{ es\_security\_api }}/user/{{ beats\_system\_username }}/\_password"

   method: POST

   body\_format: json

   body: "{ \"password\":\"{{ beats\_system\_password }}\" }"

   status\_code: 200

   user: "{{es\_api\_basic\_auth\_username}}"

   password: "{{ elastic\_password }}"

   force\_basic\_auth: "{{ es\_force\_basic\_auth }}"

   validate\_certs: "{{ es\_validate\_certs }}"

- name: Update APM system user password

  uri:

   url: "{{ es\_api\_uri }}/{{ es\_security\_api }}/user/{{ apm\_system\_username }}/\_password"

   method: POST

   body\_format: json

   body: "{ \"password\":\"{{ apm\_system\_password }}\" }"

   status\_code: 200

   user: "{{es\_api\_basic\_auth\_username}}"

   password: "{{ elastic\_password }}"

   force\_basic\_auth: "{{ es\_force\_basic\_auth }}"

   validate\_certs: "{{ es\_validate\_certs }}"

- name: Update remote\_monitoring\_user  password

  uri:

   url: "{{ es\_api\_uri }}/{{ es\_security\_api }}/user/{{ remote\_monitoring\_user\_username }}/\_password"

   method: POST

   body\_format: json

   body: "{ \"password\":\"{{ remote\_monitoring\_user\_password }}\" }"

   status\_code: 200

   user: "{{es\_api\_basic\_auth\_username}}"

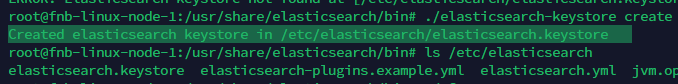
   password: "{{ elastic\_password }}"

   force\_basic\_auth: "{{ es\_force\_basic\_auth }}"

   validate\_certs: "{{ es\_validate\_certs }}"

## Setup KeyStore and bootrap password

Tạo 1 kho chứa key



Chỉ tồn tại duy nhất 1 keystore và khi tạo keystore, n sẽ lưu trong đường dẫn **/etc/elasticsearch/elasticsearch.keystore**

**elastic**

full quyền, cả quyền sercurity

**kibana\_system**

The user Kibana uses to connect and communicate with Elasticsearch.

**logstash\_system**

The user Logstash uses when storing monitoring information in Elasticsearch.

**beats\_system**

The user the Beats use when storing monitoring information in Elasticsearch.

**apm\_system**

The user the APM server uses when storing monitoring information in Elasticsearch.

**remote\_monitoring\_user**

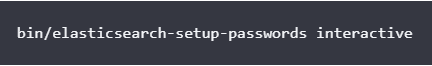
The user Metricbeat uses when collecting and storing monitoring information in Elasticsearch. It has the remote\_monitoring\_agent and remote\_monitoring\_collector built-in roles.

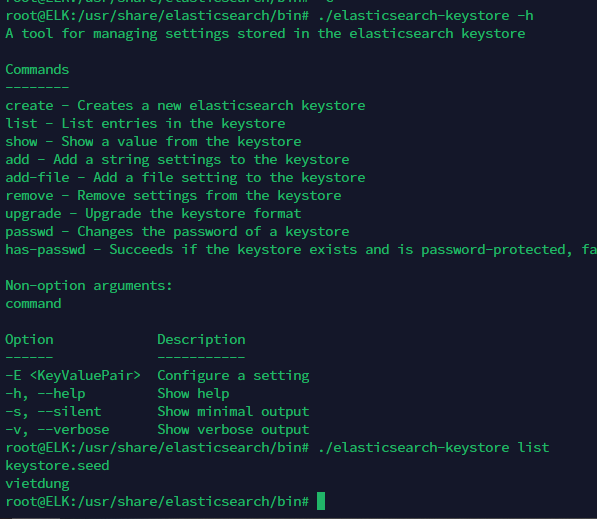
Bootstrap password là gì

Mật khẩu bootstrap là mật khẩu **tạm thời** cho phép bạn chạy các công cụ đặt tất cả mật khẩu người dùng tích hợp.  Bạn không cần biết hoặc thay đổi mật khẩu bootstrap này.

After you [set passwords for the built-in users](https://www.elastic.co/guide/en/elasticsearch/reference/current/built-in-users.html#set-built-in-user-passwords), in particular for the elastic user mật khẩu bootstrap sẽ không còn được sử dụng nữa.

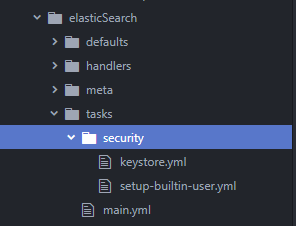
The elasticsearch-setup-passwords tool là phương pháp đơn giản để set the built-in users' passwords cho lần đầu tiên. Nó sử dụng the elastic user’s bootstrap password để chạy các API requests. For example, you can run the command in an "interactive" mode, which prompts you to enter new passwords for the elastic, kibana\_system, logstash\_system, beats\_system, apm\_system, and remote\_monitoring\_user users:





# Viết lại role cho elasticseach

* Tệp 1: install elasticsearch và cấu hình elastic
* Tệp 2: check elastic tồn tại ko và thiết lập mật khẩu boottrap
* Tệp 3: tạo mật khẩu và API người dùng
* Tạo 1 thư mục mới trong task là security

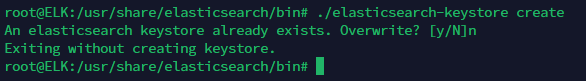


Vậy trong main ta viết rule để gọi đến keystore

# Xpack sercurity configuration  -  task sercurity

- name: including setup xpack security configuration

  include: security/keystore.yml



Chỉ có 1 key store

Bước 1: cài elasticsearch sẽ có 3 phần việc

* 1. Install elasticsearch
* 2. Copy file config của elasticsearch ở template
* 3. Start elasticsearch services

## Install Kibana

Đầu tiên

Bước 1:

* 1. Install kibana
* 2. Copy file config
* 3. Start service

3 cái này viết trong role rồi viết playbook gọi đến role đó

Tạo user/password

POST /\_security/user/jacknich

{

"password" : "l0ng-r4nd0m-p@ssw0rd",

"roles" : [ "monitoring\_user", "remote\_monitoring\_agent" ],

"full\_name" : "Jack Nicholson",

"email" : "jacknich@example.com",

"metadata" : {

"intelligence" : 7

}

}

Change password

POST /\_security/user/jacknich/\_password

{

"password" : "new-test-password"

}

Kia là tên password mới

# Ansible P2

## inventory

T có 1 inventory

[linux]

ubuntu2  ansible\_host=4.193.104.76

[webserver]

web1

web2

web3

[database]

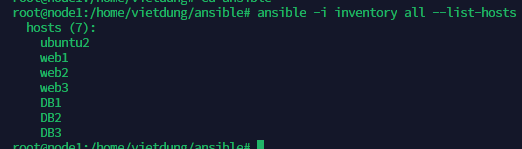
DB1

DB2

DB3

* Lệnh check các host

ansible all -i inventory --list-hosts



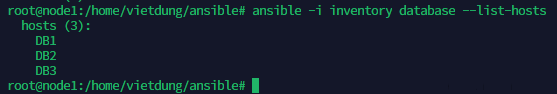
Check các host trong group của inventory

Mặc định tất cả các host đều nằm trong group [ **all** ]

Mặc định group **all** là group to nhất

* Check host trong group database

ansible -i inventory database --list-hosts



## Inventory children

Bây giờ tôi sẽ nhóm cả 3 Group kia vào 1 group bằng cách tạo ra 1 group khác,

Group mẹ cho cả 2

* Cấu trúc sẽ là

[webserver]

web1

web2

web3

[database]

DB1

DB2

DB3

[server:children]             #  group server

database

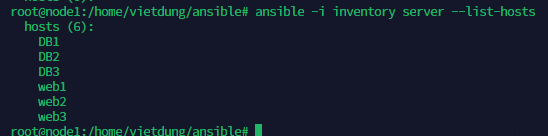
webserver

cấu trúc của group trong group sẽ có thêm children

tức là con của [server:children] sẽ là group con chứ ko phải host

* Các host của 2 group con cũng con cũng là con của group mẹ **server**, hoặc group mẹ **all**

Check các host trong group mẹ server



* Inventory

[database]

DB[1:9]



## inventory vars

[all:vars]

ansible\_connection=ssh

ansible\_user=vietdung

ansible\_password=vietdungvl@123

các var này áp dụng cho tất cả các host nằm trong group all

* Bây giờ có cách ko cần chỉ vị trí file inventory nữa

Ta config đường dẫn file inventory trong file **ansible.cfg**

File **ansible.cfg**  thêm

[defaults]

inventory = ./inventory

oke check

 ko cần chỉ inventory

## Window host cho ansible

Ansible sử dụng SSH và winrm cho linux và window

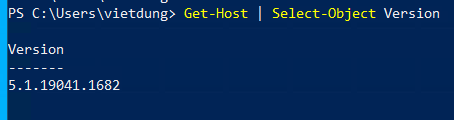
Kể từ Window 10 trở lên ko cần cài winrm vì n đã có sẵn

2 thứ cần verify

* Create user for deploy ansible
* Powershell && .NET
* Winrm (HTTP and HTTPS listen)

Check .NET version

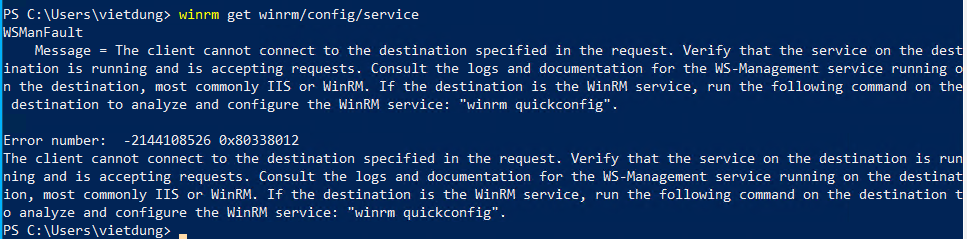
Get-Host | Select-Object Version



Oke version 5.1 > 4.

Check Winrm

winrm get winrm/config/service

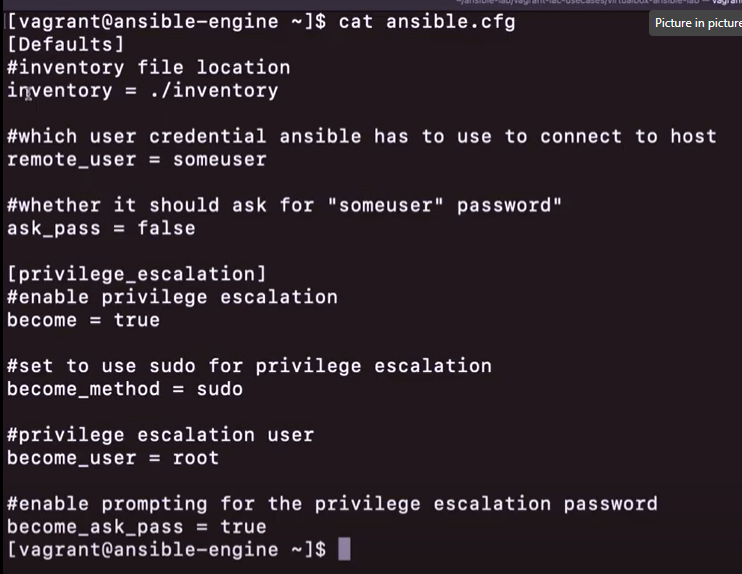


Có lỗi

Set-NetConnectionProfile -NetworkCategory Private

Enable-PSRemoting

winrm quickconfig



Become = true : sử dụng đặc quyền

Become\_method : phương thức sử dụng đặc quyền

Become\_user : root : lên quyền root

19/54

ssh-keyscan -H 192.168.58.111 >> ~/.ssh/known\_hosts

## manager Variables

có nghĩa bạn ko cần thay đổi playbook

Lưu ý khi khai báo biến

* Ko sử dụng khoảng trắng, dấu chấm, số ở đầu, ký tự đặc biệt

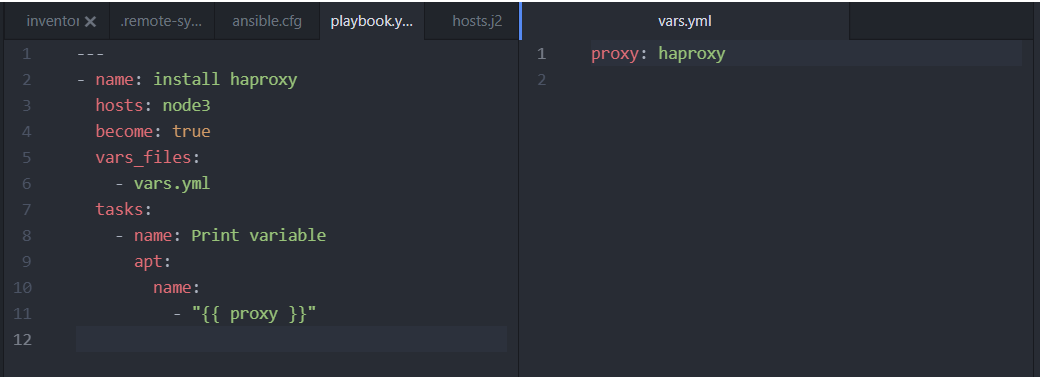


Khai báo biến theo 3 cấp khác nhau

* Global Scope: set trong config ansible hoặc command line
* Play Scope: set trong play
* Host Scope: set trong hosts hoặc groups trong inventory, fact gathering or registered task

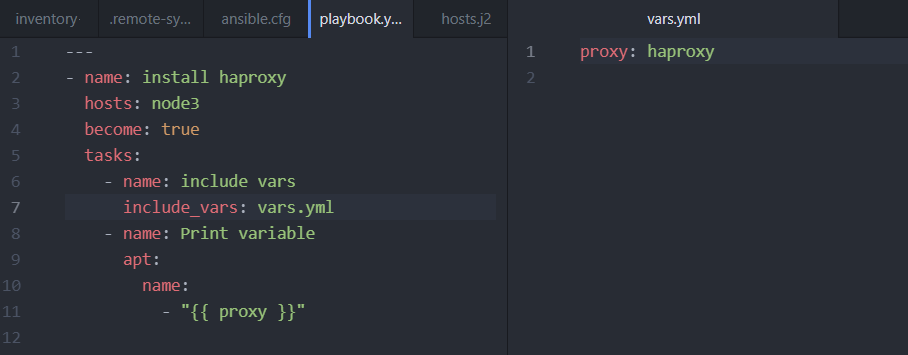
fact gathering: đôi khi để true, n sẽ lấy variable của target host

cách 1:



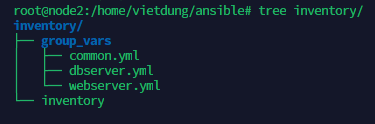
Đường dẫn biến var

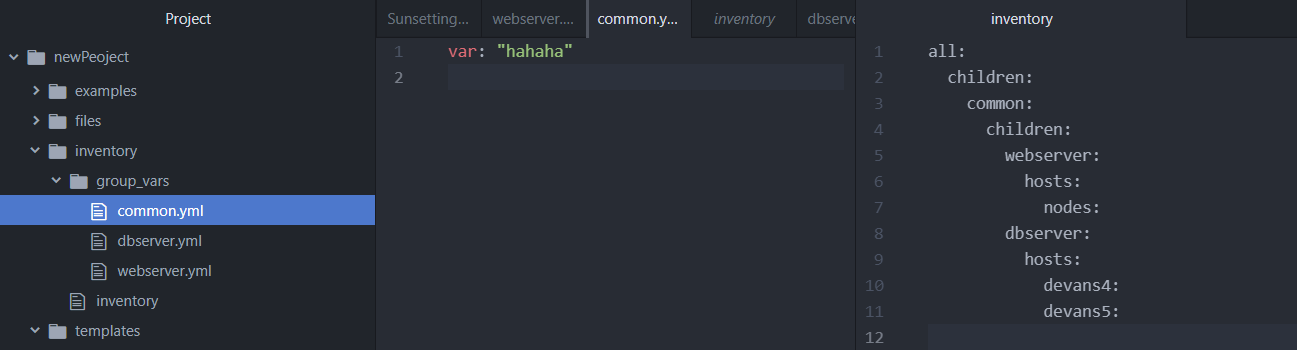
Cách 2:



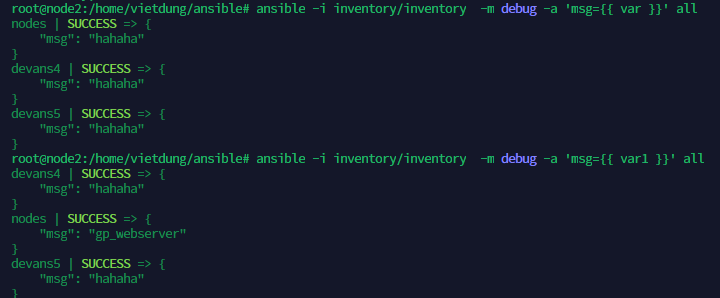


Ta phải theo format như này





Check thử cái variable



Tên trong group var = tên của inventory children

File inventory

all:

  children:

    common:

      children:

        webserver:

          hosts:

            devans[2:3]:

        dbserver:

          hosts:

            devans4:

            devans5:

ta sẽ có các biến trong group\_vars là

* common.yml
* webserver.yml
* dbserver.yml

hoặc ở dạng folder

* common (folder)
* webserver
* dbserver



Command in giá trị của biến {{ vars2 }}, nó sẽ tìm giá trị của biến này ở trong file được định nghĩa trong inventory (con của children là)

File:

* common.yml
* webserver.yml
* dbserver.yml

hoặc ở dạng folder

* common (folder)
* webserver
* dbserver

đây là group\_vars

## Với host\_var

all:

  children:

    common:

      children:

        webserver:

          hosts:

            node1:

        dbserver:

          hosts:

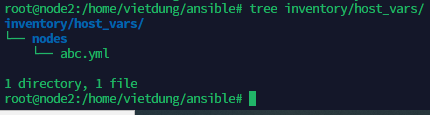
            devans4:

            devans5:

vậy trong folder host\_vars. Ta sẽ chỉ có thể tạo folder có tên là con của hosts

còn group\_vars thì folder, file.yml bằng với tên con của children

* node1
* devans4
* devans5



### Group\_vars password – vault

Tương tự như việc đặt biến trong group\_vars

Đợt trước ta đặt biến trong file host là

[all:vars]

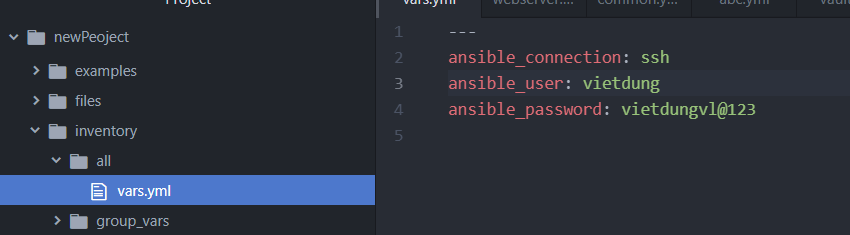
ansible\_connection=ssh

ansible\_user=vietdung

ansible\_password=vietdungvl@123

Vậy bây giờ trong folder inventory, ta lại đặt 1 folder tên all

* tức là biến trong folder all apply cho tất cả server (all) vì all là group to nhất của server



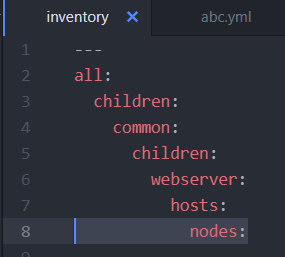
Muốn tạo biến cho 1 tập hợp các server trong group **webserver**

Thì ta lại tạo folder **webserver** , rồi tạo biến trong đó

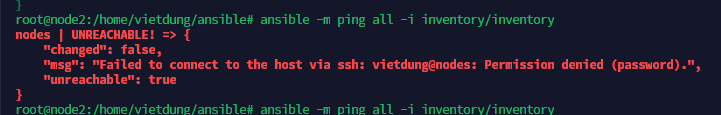


Ak lưu ý là trong file inventory, phần host ta phải để là IP nếu ko có DNS server

* lỗi



Nếu ta để host thế này mà dùng all/vars.yml, nó sẽ báo lỗi



Cần có DNS server

## Playbook

Inventory > playbook < role/tasks

-i : which inventory file ?

-l : limit the play to specific servers

-u : a specific user who use ansible for the ssh command

-k : to pass the password for the ssh connection

-K : to pass the password for elevation of privileges

-C : the check mode or dry run

-D : make a diff before/after ansible run (like git or patch)

--ask-vault : to pass a vaul password when you have secrets in your play

--syntax-check : to check the syntax (indetation)

--vault-password-file : to pass the vault password with a file not in a prompt

# pass cuar file vault

-e : to specify a variable value

-f : the fork number if you parallelize

-t : to specify tags (associate tags for some tasks)

--flush-cache : to clean the facts cache

--list-tasks : list which tasks will be launched

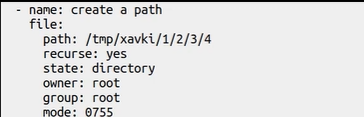
\* recurse : create all the path if necessary (only for directories)

\* state : absent / directory / file / hard / touch

    \* touch : create an empty file

    \* file : check if the file is present

VD:



State: link - tạo 1 symbolic link

Khai báo inventory

---

all:

  children:

    common:

      children:

        webserver:

          hosts:

            nodes         # đã được khai báo trong DNS

hoặc

---

all:

  children:

    common:

      children:

        webserver:

          hosts:

            20.205.209.92:

# Ansible package deb

---

- name: 'Update the apt package index and install packages to allow apt to use a repository over HTTPS'

  apt:

    pkg:

      - ca-certificates

      - curl

      - gnupg

      - lsb-release

    state: present

    update\_cache: true

- name: "Add Docker’s official GPG key:"

  apt\_key:

    url: "https://download.docker.com/linux/ubuntu/gpg"

    keyring: /usr/share/keyrings/docker.gpg

    validate\_certs: no

- name: Setup docker repository

  apt\_repository:

    repo: "deb [signed-by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu focal stable"

    state: present

    validate\_certs: no

- name: 'Install docker packages'

  apt:

    pkg:

      - "docker-ce={{ docker\_version }}"

      - "docker-ce-cli={{ docker\_version }}"

      - containerd.io

      - docker-compose-plugin

    state: present

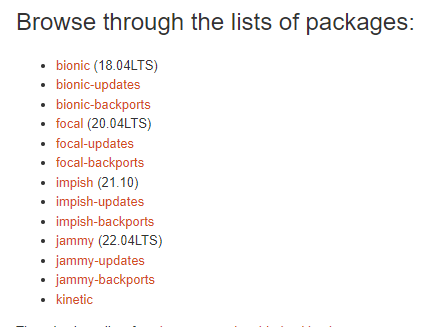
    update\_cache: true

“deb [signed-by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu focal stable"

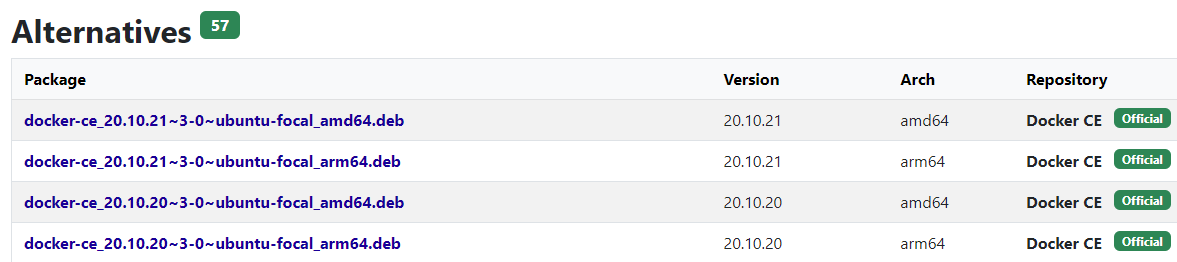
Deb viết tắt của debian

Stable: bản phân phối tiêu chuẩn

Focal: các bản phân phối, version package tương thích với OS version



Focal cho ubuntu 20.04LTS





ca-certificates

tạo sao cần cài ca-cert

vì cài docker-package là install package qua https, down gói về máy

* cần 1 chứng chỉ cho gói và add key cho n

[How to Use Ansible to Install and Set Up Docker on Ubuntu 18.04 | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-use-ansible-to-install-and-set-up-docker-on-ubuntu-18-04)

[Install Docker Engine on Debian | Docker Documentation](https://docs.docker.com/engine/install/debian/)

Command:

sudo add-apt-repository \

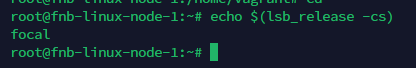
"deb [arch=amd64] https://repos.emqx.io/emqx-ce/deb/ubuntu/ \

$(lsb\_release -cs) \

stable"

= "deb [arch=amd64] https://repos.emqx.io/emqx-ce/deb/ubuntu/ focal stable"

Vì



Focal là cho OS

# apt-get và apt-cache quản lý package trên Ubuntu / Debian

**apt-get** là một công cụ quản lý các package trong Linux Ubuntu và Debian

* Cài đặt các gói phần mềm mới
* Xóa các gói phần mềm hiện có
* Nâng cấp các gói phần mềm hiện có
* Thậm chí được sử dụng để nâng cấp toàn bộ hệ điều hành.

sudo apt-get update

Lệnh **update** được sử dụng để đồng bộ lại các file được chỉ định trong file /etc/apt/sources.list. Lệnh update sẽ tìm nạp các gói từ vị trí của chúng và cập nhật các gói lên phiên bản mới hơn.

### remove

Lệnh remove dùng để xóa package.

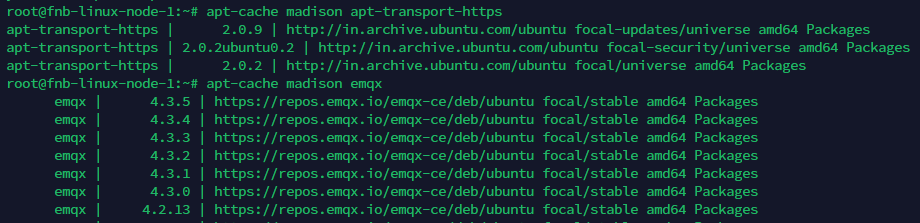
Ví dụ này xóa package nhưng không xóa trong file config.

sudo apt-get remove vsftpd

Lệnh này sẽ xóa package và xóa chúng luôn trong file config.

sudo apt-get purge vsftpd

**apt-cache** được sử dụng để tìm kiếm package đã cài đặt sẵn qua công cụ APT, Nói một cách dễ hiểu, công cụ này được sử dụng để tìm kiếm các gói phần mềm, thu thập thông tin của các gói và tìm kiếm những gói có sẵn trên hệ thống Debian hoặc Ubuntu.



## Ansible.stat : check path file tồn tại

---

- name: Is logstash installed

  stat:

    path: "{{ logstash\_extract\_dir }}/logstash-{{ logstash\_version }}"

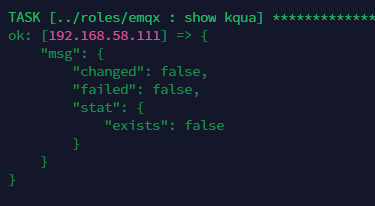
  register: logstash\_dir

- name: show kqua

  debug:

    msg: "{{ logstash\_dir }}"

* Kqua logstash\_dir



* logstash\_dir.stat.exists = false
* có dk when: not logstash\_dir.stat.exists

## Lệnh show có điều kiện



Ví dụ lệnh uname –s in ra chữ Linux có chữ L là chữ hoa

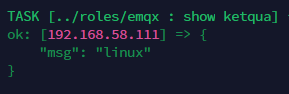
Ta cần biến Linux == linux

- name: show ketqua

  debug:

    msg: "{{ unames.stdout | lower }}"

thêm option lower vào



Task1: cài package <https://www.emqx.com/en/downloads/broker/4.4.1/emqx-4.4.1-linux-amd64.tar.gz>

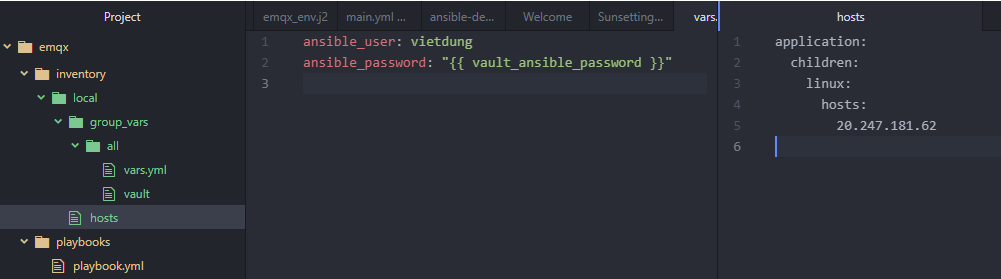
Check amd64 : dpkg --print-architecture

Arm64 : kiến trúc arm64 bit

Amd64 : thực ra n là kiến trúc phát triển của 32bit lên 64 bit, gọi là x86\_64

amd64 == x86\_64 (uname –m)

## ansible – vault



ansible\_password: "{{ vault\_ansible\_password }}"

vault\_ansible\_password: là key

ta sẽ cần 1 file vault là **vault\_ansible\_password: vietdungvl@123**

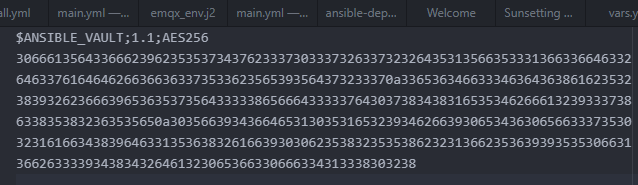
**vietdungvl@123** : value của password

nó cũng set kiểu biến thôi, nhưng vẫn đề là ai để user/password thế này

* tôi cần encrypt luôn nó rồi copy vào vault



Sau khi encrypt



Cái này bằng với **vault\_ansible\_password: vietdungvl@123**

ansible\_password: vietdungvl@123 == ansible\_password:"{{ vault\_ansible\_password }}"

vấn đề là khi encrypt file pass, n bắt nhập pass để encrypt, decrypt file đó

vậy làm thế nào, ta cần tạo 1 secret file lưu pass encrypt

n tương đương vs lệnh

ask\_pass            = false

ask\_vault\_pass      = false

vault\_password\_file = /secrets/vault\_pwd

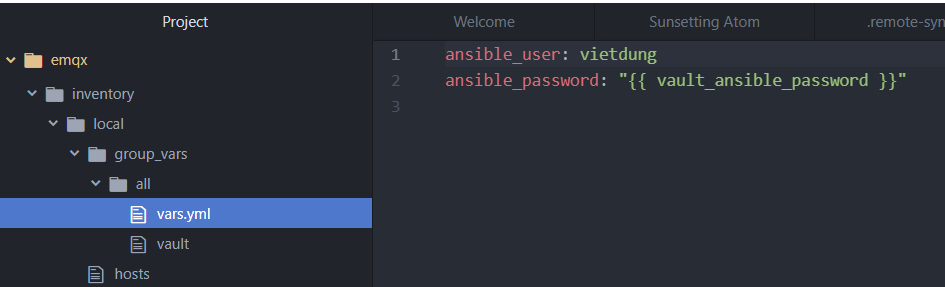
đường dẫn /secrets/vault\_pwd sẽ là vietdung



Password để encrypt vault

Tổng kết lại ta cần

1. Thiết lập ansible user/pass trong group\_vars



1. Tạo vault từ file có content

vault\_ansible\_password: vietdungvl@123

vietdungvl@123: password của target hosst để ssh

password encrypt file: vietdung

1. Tạo secret lưu giá trị encrypt



1. Update config asnible

/etc/ansible/ansible.cfg

ask\_pass            = false

ask\_vault\_pass      = false

vault\_password\_file = /home/vagrant/emqx/secrets/vault\_pwd