|  |
| --- |
| TỔNG CÔNG TY MẠNG LƯỚI VIETTEL  **TRUNG TÂM KỸ THUẬT TOÀN CẦU**  A picture containing text, tableware, dishware, plate  Description automatically generated  **HƯỚNG DẪN THỰC HIỆN CÀI ĐẶT VÀ CẤU HÌNH TỰ ĐỘNG THIẾT BỊ BẰNG ANSIBLE**  (LƯU HÀNH NỘI BỘ)  ***Hà Nội, tháng 11 năm 2022*** |

# MỤC LỤC

[MỤC LỤC 2](#_Toc135665720)

[MỘT SỐ THUẬT NGỮ 4](#_Toc135665721)

[A. Giới thiệu sơ bộ về Ansible 6](#_Toc135665722)

[B. Sơ lược về ngôn ngữ YAML 7](#_Toc135665723)

[C. Sơ lược về ngôn ngữ template Jinja2 7](#_Toc135665724)

[D. Cài đặt Ansible 10](#_Toc135665725)

[E. Cú pháp thực hiện Ansible 13](#_Toc135665726)

[1. Chạy Ansible từ dòng lệnh với host đơn 13](#_Toc135665727)

[2. Tạo inventory và playbook 13](#_Toc135665728)

[a. Tạo file inventory 13](#_Toc135665729)

[b. Tạo file playbook 15](#_Toc135665730)

[3. Sử dụng Ansible trong quá trình SSH server Linux 16](#_Toc135665731)

[a. Command và shell 16](#_Toc135665732)

[b. Ghi lại hoặc hiển thị kết quả của command/shell ra biến 17](#_Toc135665733)

[c. Leo quyền hoặc chuyển user 18](#_Toc135665734)

[d. Thao tác với user và group 19](#_Toc135665735)

[e. Thao tác với file 21](#_Toc135665736)

[f. Quản lý dịch vụ trên server 23](#_Toc135665737)

[g. Quản lý các gói trên server 24](#_Toc135665738)

[h. Thu thập thông tin host 26](#_Toc135665739)

[i. Cấu hình iptables 27](#_Toc135665740)

[j. Crontab 29](#_Toc135665741)

[4. Sử dụng Ansible thao tác với server Windows 31](#_Toc135665742)

[a. Cấu hình giao thức WinRM 31](#_Toc135665743)

[b. Thực hiện lệnh với module win\_command hoặc win\_shell 31](#_Toc135665744)

[c. Thao tác với file 32](#_Toc135665745)

[d. Cấu hình Windows Firewall 33](#_Toc135665746)

[5. Sử dụng Ansible gọi HTTP request 35](#_Toc135665747)

[6. Một số cú pháp/module dựng sẵn sử dụng trong playbook của Ansible 37](#_Toc135665748)

[a. Khai báo biến 38](#_Toc135665749)

[b. Thực hiện task bỏ qua lỗi nếu có 39](#_Toc135665750)

[c. Thực hiện song song các task trong playbook 39](#_Toc135665751)

[d. Thực hiện một task theo điều kiện xác định 40](#_Toc135665752)

[e. Thực hiện một task theo vòng lặp 41](#_Toc135665753)

[f. Handler 42](#_Toc135665754)

[7. Templating 43](#_Toc135665755)

[a. Template cấu hình 43](#_Toc135665756)

[b. Template log 44](#_Toc135665757)

[8. Gắn thêm module để sử dụng trong Ansible 45](#_Toc135665758)

[9. Tổ chức các kịch bản thành task và role 46](#_Toc135665759)

[F. Tài liệu tham khảo 48](#_Toc135665760)

# MỘT SỐ THUẬT NGỮ

| **Thuật ngữ** | **Ý nghĩa** |
| --- | --- |
| **Ansible** | Công cụ tự động hóa nghiệp vụ CNTT. |
| **BIOS** | Basic Input/Output System, Hệ thống quản lý xuất nhập cơ bản |
| **Bonding** | Gộp nhiều kết nối mạng với nhau thành 1 kết nối, đảm bảo dự phòng kết nối và tăng tốc độ. |
| **Cobbler** | Công cụ quản lý và cấu hình cài đặt hệ điều hành qua PXE. |
| **GNOC** | Quy trình vận hành mạng lưới. |
| **Hardening** | Thuật ngữ nói về việc cấu hình một thiết bị để đảm bảo thiết bị đó hoạt động an toàn trên mạng lưới. |
| **HLD** | High Level Design – Thiết kế mức cao (mô tả logic về hệ thống, vùng mạng kết nối v.v…) |
| **IDRAC** | Integrated Dell Remote Access Controller, giao diện quản trị máy chủ của Dell. |
| **iLO** | Integrated Lights-Out, giao diện quản trị máy chủ HPE. |
| **Jinja2** | Ngôn ngữ biểu diễn template văn bản |
| **LLD** | Low Level Design – Thiết kế mức thấp (mô tả chi tiết về vật lý của các thiết bị trong hệ thống về tọa độ, port đấu nối, tọa độ kết nối chi tiết v.v..) |
| **PXE** | Giao thức cho phép khởi động một phần mềm lấy qua mạng. |
| **RAID** | Redundant Arrays of Inexpensive Disks, Cơ chế gộp nhiều ổ cứng vật lý thành 1 ổ cứng nhằm đảm bảo dự phòng và tăng tốc độ lưu trữ, đọc ghi dữ liệu. |
| **SIRC** | Hệ thống và quản lý giám sát ATTT |
| **SR** | Service Request – Yêu cầu dịch vụ. |
| **UEFI** | Unified Extensible Firmware Interface, Giao diện firmware mở rộng hợp nhất |
| **YAML** | Ain’t Markup Language, Ngôn ngữ kịch bản biểu diễn dữ liệu |

# A. Giới thiệu sơ bộ về Ansible

Ansible là một dự án của Red Hat phục vụ cho công tác quản trị và cấu hình tự động hóa. Trong công tác vận hành và triển khai hệ thống CNTT hiện nay (cả ở mức hạ tầng vật lý lẫn platform, ứng dụng), do nhu cầu ngày càng mở rộng với số lượng hệ thống có thể lên tới hàng trăm, hàng nghìn thiết bị, ứng dụng, VM v.v… nên các phương pháp quản trị và triển khai truyền thống ngày càng không hiệu quả và tiêu tốn thời gian, chi phí lớn.

So với các giải pháp tự động hóa khác, Ansible có một số ưu điểm sau đây:

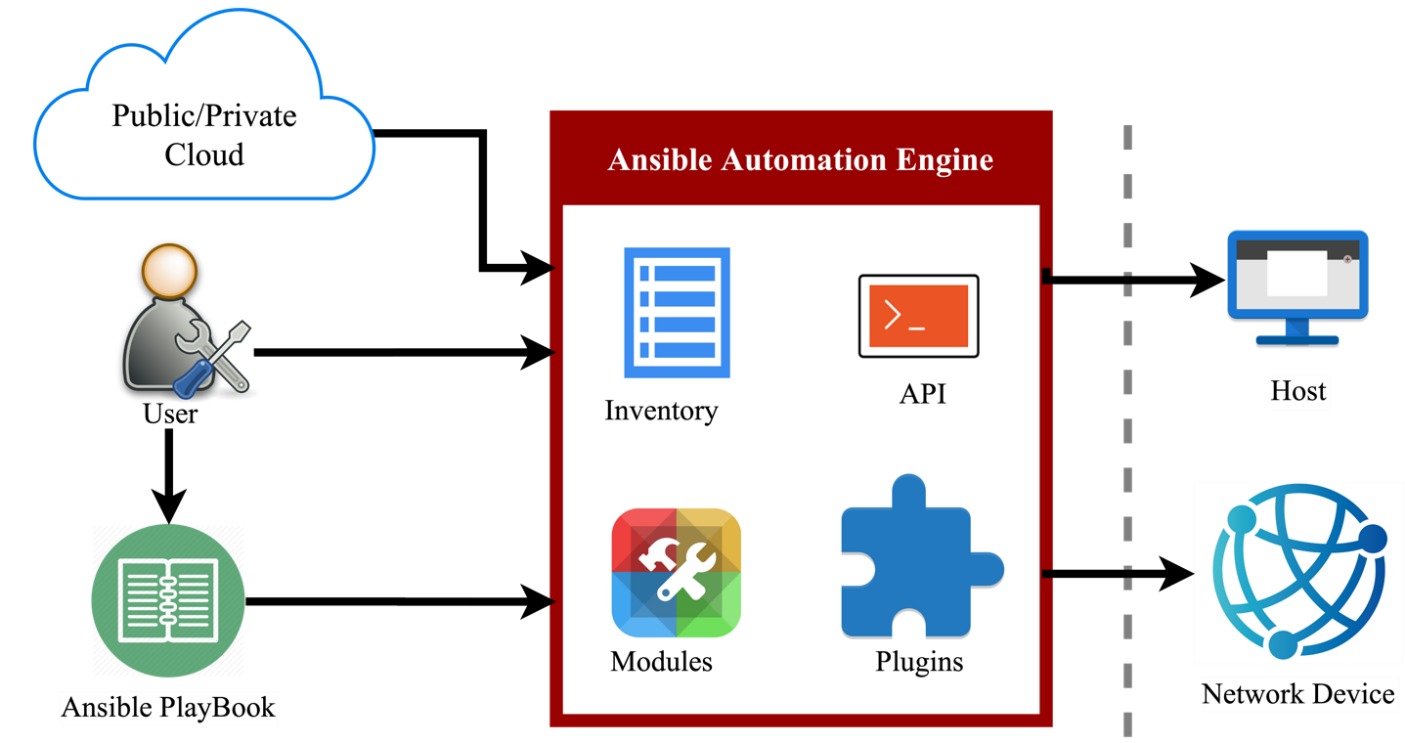
- Không cần phải biết về ngôn ngữ lập trình để có thể sử dụng.

- Ngôn ngữ trình bày và cấu hình của Ansible là đơn giản, tự nhiên và gần gũi với tiếng Anh thông dụng.

- Ansible là công cụ dạng agentless, tức là không cần phải cài bất cứ thành phần gì ở các thiết bị đích để thao tác với Ansible.

- Có sự đóng góp lớn từ cộng đồng và các vendor CNTT lớn.

Luồng hoạt động của Ansible như sau:



Người dùng có thể tương tác trực tiếp với Ansible Automation Engine hoặc tạo ra một file kịch bản Ansible Playbook để tương tác với Ansible Automation Engine qua nó.

Automation Engine đọc Ansible Playbook cũng như danh sách và các thông tin của host trong file Inventoy để sinh ra các câu lệnh và liên tục đẩy qua host hoặc qua các thiết bị mạng

Bên cạnh đó, người dùng có thể từ Public/ Private Cloud, đẩy thao tác bằng cách gọi các API, Ansible sẽ tạo các kênh SSH để thực thi các công việc tương ứng.

Việc sử dụng Ansible hoàn toàn miễn phí. Red Hat cũng có các gói hỗ trợ mất phí dành cho các nhu cầu đặc biệt của doanh nghiệp.

# B. Sơ lược về ngôn ngữ YAML

Ansible được viết bằng ngôn ngữ Python và sử dụng YAML làm ngôn ngữ kịch bản, do đó để làm quen và sử dụng được Ansible tốt nhất thì nên có một số kiến thức cơ bản về YAML. YAML (Ain’t Markup Language) được sử dụng để biểu diễn và đánh dấu dữ liệu (cũng như XML, JSON v.v…) nhưng YAML có cách thức trình bày dễ đọc hơn. Tuy nhiên, nhược điểm của ngôn ngữ này là có cú pháp viết với khoảng trắng, giống như Python, do đó nếu không chú ý về khoảng trắng thì có thể dẫn đến việc biểu diễn sai dữ liệu trong YAML, từ đó có thể ảnh hưởng đến các tiến trình sử dụng file YAML tương ứng. Tốt nhất, phần tử dữ liệu con nên viết thụt vào **2 dấu cách** so với dữ liệu cha.

Với YAML, các chú thích bắt đầu bằng dấu #.

|  |
| --- |
| # yet another comment |

Giá trị một đối tượng được biểu diễn bằng viết trong 1 dòng <key>:<giá trị>

|  |
| --- |
| # Ví dụ YAML  product: Laptop ABC  price: 333  specifications:  os: Windows 10  hdd: 500 GB |

Để biểu diễn 1 mảng, sau dấu ":", các phần tử của mảng biểu diễn trên mỗi dòng, thụt 2 dấu cách so với phần tử cha, bắt đầu bằng dấu "-"

|  |
| --- |
| # Ví dụ YAML  product: Laptop ABC  price: 333  specifications:  os: Windows 10  hdd: 500 GB  ram: # Mảng ram  - branch: XYZ  - speed: XXX |

Nếu 1 dữ liệu văn bản dài, có thể dùng ký hiệu ">" để gộp nó thành 1.

|  |
| --- |
| data: >  Dòng chữ 1  Dòng chữ 2 |

# C. Sơ lược về ngôn ngữ template Jinja2

Ansible (và rộng hơn là thế giới Python) sử dụng Jinja2 làm ngôn ngữ thiết kế template cho các file cấu hình. Một file jinja2 template là 1 file văn bản với các chỉ mục khai báo biến mà khi ứng dụng chạy và sử dụng nó, có thể thay thế các chỉ mục khai báo biến đó bằng giá trị tương ứng để phù hợp với mục đích sử dụng của ứng dụng. Chúng ta sẽ gặp các template Jinja2 như thế này không chỉ ở trong Ansible mà ở nhiều các web framework trên Python khác như Flask, Django v.v...

Có 3 cú pháp cơ bản trong 1 file template Jinja2 là như sau

- Khai báo 1 biến trong văn bản bằng cách đưa tên biến vào trong cặp ngoặc nhọn {{ }}

- Điều hướng và vòng lặp được khai báo trong cụm {% %}

- Comment được khai báo trong {# #}. Khi file template được xử lý thì cụm comment sẽ bị loại bỏ.

Một số ví dụ

Ví dụ 1:

|  |
| --- |
| <html>  <center><h1> The Apache webserver is running on {{ ansible\_hostname }} </h1>  </center>  </html> |

Đây là 1 đoạn script HTML, ở đó có khai báo 1 biến là ansible\_hostname. Đến khi script HTML này được đưa ra xử lý, giá trị tương ứng được truyền vào biến ansible\_hostname sẽ thay thế cho đoạn khai báo biến {{ ansible\_hostname }} nói trên ở server đích.

Ví dụ 2:

|  |
| --- |
| <!DOCTYPE html>  <html>      <head>          <title>If example</title>      </head>      <body>          {% if(name == "Geeksforgeeks") %}              <h3> Welcome </h3>          {% else %}              <h3> Unknown name entered: {{name}} </h3>          {% endif %}      </body>  </html> |

Đây là 1 đoạn script HTML khác, ở đó xét điều kiện giá trị truyền vào của biến name. Nếu giá trị truyền vào cho biến name là Geeksforgeeks, thì title Welcome sẽ được chèn vào sử dụng trong đoạn script HTML kết quả, còn nếu không thì đoạn title Unknown name sẽ được chèn vào script HTML kết quả.

Ví dụ 3:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>For Example</title>  </head>  <body>  <h2> Geeksforgeeks Available Course </h2>  {% for course in courses%}  <h4> {{course}} </h4>  {% endfor %}  </body>  </html> |

Đây là đoạn script HTML ở đó các giá trị của biến course sẽ được in ra ứng với course là giá trị trong 1 biến list courses và được in lặp lại qua tất cả các giá trị trong list courses.

Ví dụ 4

|  |
| --- |
| File base.html  <!DOCTYPE html>  <html lang="en">  <head>  {% block head %}  <link rel="stylesheet" href="style.css" />  <title>{% block title %}{% endblock %} - My Webpage</title>  {% endblock %}  </head>  <body>  <div id="content">{% block content %}{% endblock %}</div>  <div id="footer">  {% block footer %}  &copy; Copyright 2008 by <a href="http://domain.invalid/">you</a>.  {% endblock %}  </div>  </body>  </html> |
| File child.html  {% extends "base.html" %}  {% block title %}Index{% endblock %}  {% block head %}  {{ super() }}  <style type="text/css">  .important { color: #336699; }  </style>  {% endblock %}  {% block content %}  <h1>Index</h1>  <p class="important">  Welcome to my awesome homepage.  </p>  {% endblock %} |

Đây là 1 ví dụ về thừa kế template. Có các block được định nghĩa trong file base.html là block head, block title (nằm trong block head), block content và block footer. File child.html thừa kế lại nội dung của file base.html, và ở đó block title được thay thế giá trị là Index, block head thừa kế lại block head của file base.html đồng thời bổ sung thêm style css, và block content được thay thế bằng giá trị được định nghĩa trong file child.html.

Việc sử dụng jinja2 template rất linh hoạt, người dùng có thể thực hiện “lập trình” template để sao cho đáp ứng với nhu cầu thực tế của mình. Người đọc có thể tham khảo thêm các hướng dẫn liên quan đến dựng file template bằng Jinja2 qua địa chỉ sau: [https://jinja.palletsprojects.com/en/3.1.x/templates/#](https://jinja.palletsprojects.com/en/3.1.x/templates/) (hoặc các phiên bản cao hơn).

# D. Cài đặt Ansible

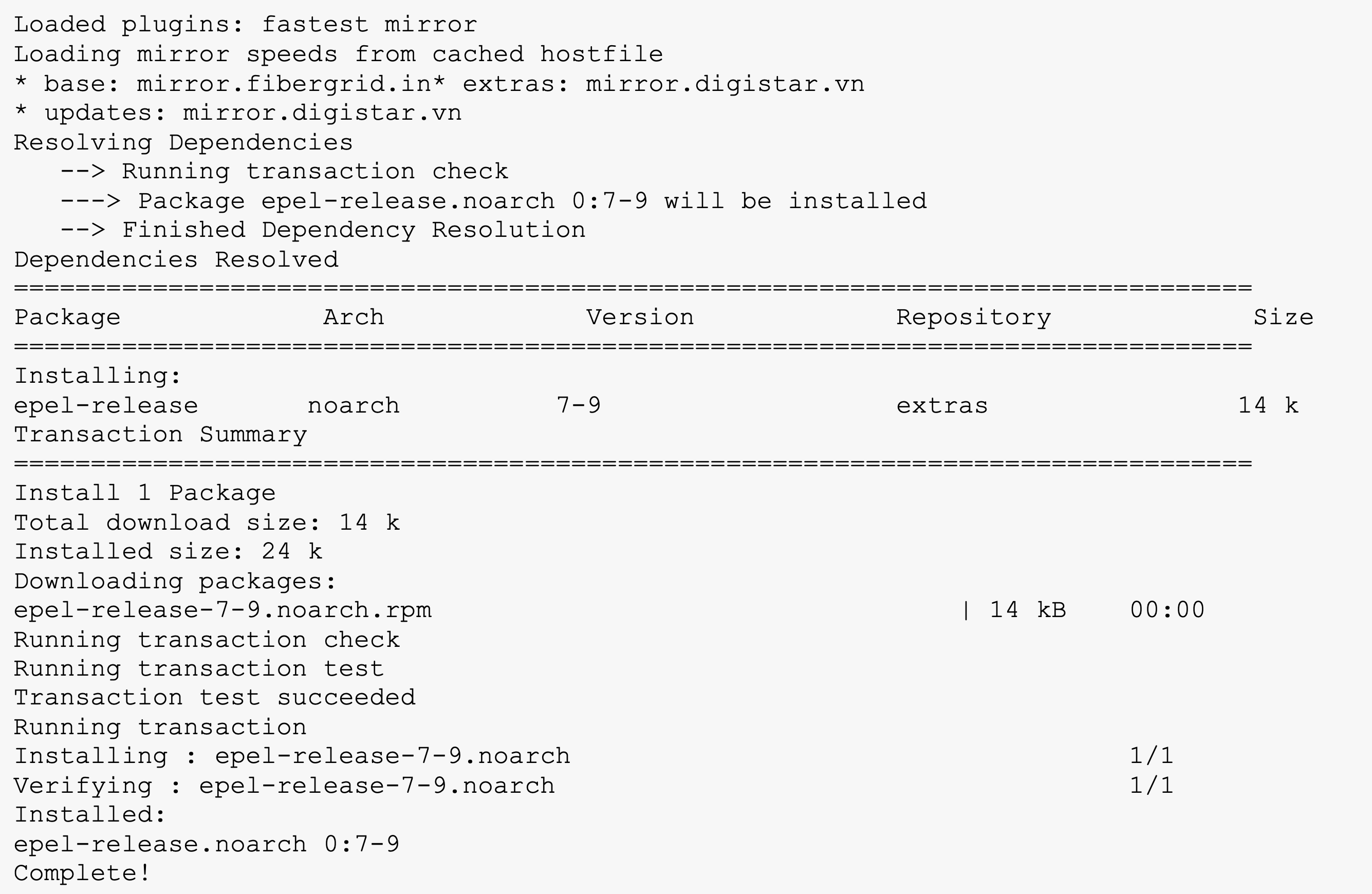
Hướng dẫn sau đây sử dụng cho HĐH họ Red Hat (CentOS/Oracle Linux…)

Bước 1: Cài đặt EPEL Repository

EPEL repository là một repository miễn phí cung cấp rất nhiều gói phần mềm mã nguồn mở (open source software packages) mà không có sẵn trong YUM repository.

Để cài đặt EPEL Repository, sử dụng câu lệnh:

|  |
| --- |
| sudo yum -y install epel-release |



Bước 2: Cập nhật hệ thống

Sử dụng câu lệnh:

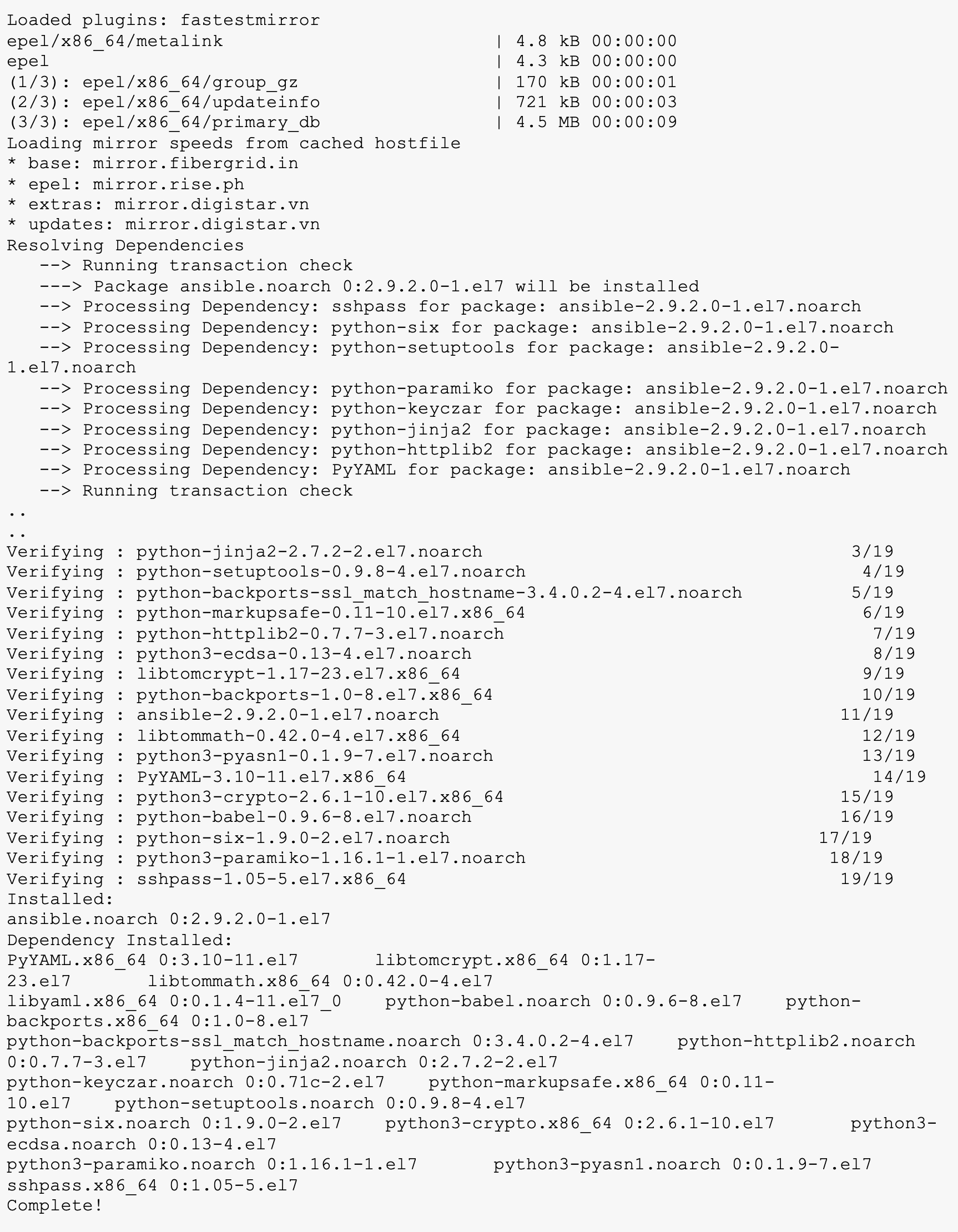
|  |
| --- |
| sudo yum -y update |



Bước 3: Cài đặt Ansible

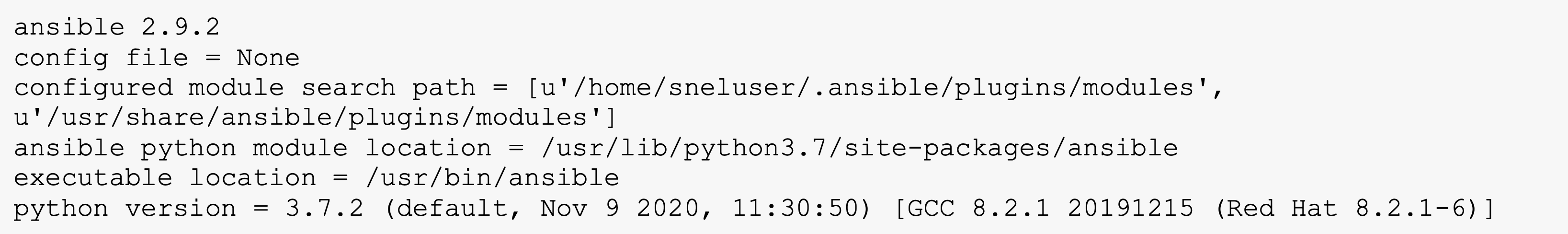
Chạy câu lệnh bên dưới để cài bản Ansible mới nhất:

|  |
| --- |
| sudo yum -y install ansible |



Một cách để kiểm tra Ansible đã được cài đặt thành công chưa là kiểm tra version của Ansible trên máy đó, sử dụng câu lệnh:

|  |
| --- |
| ansible --version |



# E. Cú pháp thực hiện Ansible

## 1. Chạy Ansible từ dòng lệnh với host đơn

Nếu muốn chạy 1 lệnh đơn (ad hoc command) trên dòng lệnh sử dụng ansible, có thể thực hiện theo cú pháp sau:

|  |
| --- |
| ansible **[**pattern**]** -m **[**module**]** -a "[module options]" |

Ví dụ, nếu muốn ping 1 danh sách các server (chú ý, “ping” của Ansible chỉ mang tính chất kiểm tra kết nối như SSH, thư viện python trên host đích chứ không phải ping ICMP IP thiết bị):

|  |
| --- |
| ansible all -i 10.0.0.1,10.0.0.2,10.0.0.3 --module-name ping |

Nếu muốn reboot 1 danh sách các server chạy Linux

|  |
| --- |
| ansible all -i 10.0.0.1,10.0.0.2,10.0.0.3 -a "/sbin/reboot" |

## 2. Tạo inventory và playbook

### a. Tạo file inventory

Để có thể sử dụng ansible cho nhiều thiết bị khác nhau, cần chuẩn bị file inventory để lưu trữ các host cần thiết. Có thể viết file inventory theo 2 cách là INI hoặc YAML. Với kiểu INI, chúng ta có thể viết như sau:

|  |
| --- |
| **[all:vars]**  *ansible\_connection***=**ssh  **[SERVER:vars]**  *ansible\_ssh\_user***=**vt\_admin  *ansible\_ssh\_pass***=**password1  *ansible\_become\_pass***=**password2  **[SERVER]**  *host1 ansible\_host***=**10.0.0.1  *host2 ansible\_host***=**10.0.0.2  **[OTHERHOST]**  10.1.1.1  10.1.1.2  10.1.1.3  *10.1.1.4 mgt\_user***=**admin mgt\_pass=mgtpasswd1 |

Trong file dữ liệu ở trên, chúng ta có các thông tin như sau:

Có các group là all, SERVER và OTHERHOST. Trong đó, group all là group mặc định của inventory. Còn 1 group mặc định khác là ungrouped, là những host không nằm ở group nào khác ngoài all.

Trong group all có 1 biến toàn cục là ansible\_connection, trong group SERVER có 3 biến toàn cục của riêng group là ansible\_ssh\_user, ansible\_ssh\_pass, ansible\_become\_pass. Ngoài ra, mỗi host của group SERVER có 1 biến riêng là ansible\_host. Group OTHERHOST có host 10.1.1.4 là có 2 biến riêng mgt\_user và mgt\_pass.

Với dữ liệu ở trên, ta có thể viết lại file inventory như sau với định dạng YAML (chú ý phần thụt dấu cách trong file YAML như đã khuyến cáo ở mục B).

|  |
| --- |
| **all**:  **vars**:  **ansible\_connection**: ssh  **SERVERS**:  **vars**:  **ansible\_ssh\_user**: vt\_admin  **ansible\_ssh\_pass**: password1  **ansible\_become\_pass**: password2  **host1**:  **ansible\_host**: 10.0.0.1  **host2**:  **ansible\_host**: 10.0.0.2  **OTHERHOST**:  **10.1.1.1**:  **10.1.1.2**:  **10.1.1.3**:  **10.1.1.4**:  **mgt\_user**: admin  **mgt\_pass**: mgtpasswd1 |

Mặc định, khi thực hiện chạy ansible mà không chỉ rõ file inventory cụ thể, ansible sẽ lấy thông tin từ file inventory mặc định nằm trong đường dẫn /etc/ansible/hosts.

Như chúng ta có thể thấy, các file inventory lưu trữ nhiều dữ liệu liên quan đến thiết bị, hoặc password, là những thông tin nhạy cảm có thể bị khai thác. Trong trường hợp cần bảo đảm ATTT tránh lộ lọt thông tin thiết bị hoặc password, tốt nhất nên mã hóa file inventory lại. Ansible cung cấp sẵn công cụ ansible-vault để thực hiện mã hóa và giải mã các file, mặc định chuẩn mã hóa là SHA256.

Để mã hóa file inventory, thực hiện lệnh sau:

|  |
| --- |
| ansible-vault encrypt **<**inventory\_file**>** |

Để giải mã file inventory, thực hiện lệnh sau:

|  |
| --- |
| ansible-vault decrypt **<**inventory\_file**>** |

Nếu muốn xem file inventory, thực hiện lệnh sau:

|  |
| --- |
| ansible-vault view **<**inventory\_file**>** |

Nếu muốn sửa file inventory, thực hiện lệnh sau, lưu ý việc sửa file tuân theo cú pháp của lệnh vi:

|  |
| --- |
| ansible-vault edit **<**inventory\_file**>** |

### b. Tạo file playbook

Để có thể lưu trữ, tái sử dụng lâu dài các câu lệnh hoặc các kịch bản thao tác bằng ansible, người quản trị nên viết các file kịch bản playbook. File Playbook được biểu diễn dưới dạng YAML (cơ bản về YAML có thể xem lại mục B). Trong 1 file playbook có thể có 1 hoặc nhiều “play” là các lệnh thao tác được sắp xếp trong file playbook. Mặc định, Ansible sẽ đọc và thao tác các play trong file playbook theo thứ tự từ trên xuống dưới.

Xét ví dụ bên dưới như sau:

|  |
| --- |
| **-** **name:** Update web servers  **hosts:** webservers  **remote\_user:** root  **tasks:**  **-** **name:** Ensure apache is at the latest version  **ansible.builtin.yum:**  **name:** httpd  **state:** latest  **-** **name:** Write the apache config file  **ansible.builtin.template:**  **src:** /srv/httpd.j2  **dest:** /etc/httpd.conf  **-** **name:** Update db servers  **hosts:** databases  **remote\_user:** root  **tasks:**  **-** **name:** Ensure postgresql is at the latest version  **ansible.builtin.yum:**  **name:** postgresql  **state:** latest  **-** **name:** Ensure that postgresql is started  **ansible.builtin.service:**  **name:** postgresql  **state:** started |

File ansible nói trên có 2 play:

- Play “Update web servers” gồm 2 task là kiểm tra phiên bản apache có phải phiên bản mới nhất hay không, và ghi cấu hình file /etc/httpd.conf từ file /srv/httpd.j2 trên máy thao tác đang cài ansible. Play này sẽ chạy với các host thuộc group webservers và login vào các host đó bằng user root.

- Play “Update db servers” gồm 2 task là kiểm tra phiên bản postgresql có phải phiên bản mới nhất hay không, và kiểm tra postgresql đã chạy hay chưa. Play này sẽ chạy với các host thuộc group databases và login vào các host đó bằng user root.

Để chạy Ansible với inventory và playbook tương ứng, thực hiện lệnh sau

|  |
| --- |
| ansible-playbook **-**i **<**inventory\_file**>** **<**playbook\_file**>** |

Nếu muốn có log chi tiết hơn trong quá trình chạy

|  |
| --- |
| ansible-playbook **-**i **<**inventory\_file**>** **<**playbook\_file**>** **-**v |

Mặc định, ansible sẽ chạy 5 host trong 1 thời điểm, nên nếu muốn chạy nhiều hơn, ví dụ 10 host trong 1 thời điểm thì chạy như sau

|  |
| --- |
| ansible-playbook **-**i **<**inventory\_file**>** **<**playbook\_file**>** -f 10 |

Nếu muốn thao tác với 1 inventory đã được mã hóa bằng ansible\_vault thì thực hiện lệnh sau, chú ý sau đó cần nhập password để giải mã inventory

|  |
| --- |
| ansible-playbook **--**ask-vault-pass **-**i **<**inventory\_file**>** **<**playbook\_file**>** |

## 3. Sử dụng Ansible trong quá trình SSH server Linux

Một trong số những case sử dụng Ansible phổ biến nhất là điều khiển hoặc thao tác hàng loạt server Linux thông qua SSH.

Lưu ý: Mặc định, Ansible có check SSH key của các server, theo đó khi thực hiện SSH server, Ansible sẽ kiểm tra xem trước đó đã add SSH key tương ứng với server chưa. Việc này có thể gây phiền toái với các server mới triển khai, chưa có SSH key. Do đó, thực hiện disable chức năng này trong file /etc/ansible/ansible.cfg hoặc ~/.ansible.cfg

|  |
| --- |
| [defaults]  host\_key\_checking = False |

### a. Command và shell

Xét file kịch bản và inventory sau đây:

|  |
| --- |
| # file playbook  ---  **- name**: Serial number  **hosts**: DEPLOY  **become**: **yes**  **become\_method**: su  **become\_flags**: '-l'  **gather\_facts**: **no**  **tasks**:  **- name**: serial number  **shell**: dmidecode -s system-serial-number  **register**: out  **- debug**: var=out.stdout\_lines |

|  |
| --- |
| # inventory file  **[all:vars]**  *ansible\_connection***=**ssh  **[SERVER:vars]**  *ansible\_ssh\_user***=**vt\_admin  *ansible\_ssh\_pass***=**password1  *ansible\_become\_pass***=**password2  **[SERVER]**  *host1 ansible\_host***=**10.0.0.1  *host2 ansible\_host***=**10.0.0.2  *host3 ansible\_host***=**10.0.0.3 |

File kịch bản ở trên thực hiện thao tác 1 lệnh shell để get serial number của server, bằng cách login vào user vt\_admin, sau đó su sang root để thực hiện lệnh (vì dmidecode chỉ chạy được trên user root).

Điểm đặc biệt của Ansible khiến nhiều người làm việc lâu năm trong lĩnh vực sysadmin Linux yêu thích là có thể sử dụng trực tiếp các lệnh hoặc script Linux trong file playbook, thay vì phải học lại cú pháp mới để thực hiện các công việc quen thuộc trước đó. Ansible cung cấp 2 module để thực hiện thao tác lệnh trên các host Linux là command và shell. Module command là module chạy lệnh mặc định của Ansible, tuy nhiên vì nó không được chạy trực tiếp trên shell của host đích, nên một số thao tác redirect như | < > l & không thực hiện được. Nếu cần thực hiện những thao tác như vậy có thể chuyển sang sử dụng module shell.

Bên cạnh command và shell, Ansible còn có 1 module nữa phục vụ cho thao tác lệnh là raw. Đây là module chạy lệnh ở mức thô nhất trên host (giống như ssh thẳng vào host rồi chạy lệnh, các module command và shell chạy trên thư viện python của host đích, không phải chạy thẳng trên shell của host đích), khuyến cáo chỉ nên sử dụng module này với các thiết bị hoặc hệ điều hành quá đặc thù mà module command và shell không thể thao tác được, còn trong trường hợp mặc định thì không nên sử dụng module raw.

### b. Ghi lại hoặc hiển thị kết quả của command/shell ra biến

Để thực hiện lưu kết quả của 1 command hoặc shell ra biến, trong task tương ứng sử dụng cú pháp register. Ví dụ như sau:

|  |
| --- |
| **tasks**:  **- name**: serial number  **shell**: dmidecode -s system-serial-number  **register**: out |

Kết quả của lệnh dmidecode sẽ được lưu ra 1 biến register tên là out.

Khi chạy lệnh Linux, một biến register sẽ bao gồm các thành phần như sau

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| cmd | Lệnh đã chạy |
| stdout | Kết quả đầu ra tiêu chuẩn của lệnh |
| stderr | Kết quả đầu ra lỗi của lệnh (nếu lệnh có lỗi) |
| stdout\_lines | Cũng như stdout nhưng biểu diễn thành từng dòng (stdout các dấu xuống dòng đều ghi là \n) |
| stderr\_lines | Cũng như stderr nhưng biểu diễn thành từng dòng |
| start | Thời gian bắt đầu chạy lệnh |
| end | Thời gian kết thúc chạy lệnh |
| delta | Thời gian để thực thi lệnh (end-start) |
| rc | Mã lỗi trả về, nếu thành công thì rc = 0 |

Để hiển thị kết quả của biến register ra màn hình, bổ sung thêm 1 task debug như sau:

|  |
| --- |
| **- debug**: var=out.stdout\_lines |

Nếu muốn lưu kết quả được lưu trong biến register ra file, có thể sử dụng module lineinfile để cấu hình như sau

|  |
| --- |
| **- name**: result  **become**: **no**  **lineinfile**:  **dest**: /u01/laboratory/hiepnd8/checklist\_os.txt  **line**: "\n----------\nIP\_OS:{{inventory\_hostname}}\n{{test.stdout}}\n\n"  **insertafter**: EOF  **delegate\_to**: localhost |

### c. Leo quyền hoặc chuyển user

Trong nhiều trường hợp, chúng ta chỉ được phép thực hiện SSH vào server thông qua 1 user được cho phép SSH (cấu hình trong AllowedUsers của SSH server), nhưng lại muốn chạy lệnh dưới quyền của 1 user khác (ví dụ chạy dưới quyền root). Ansible cho phép thực hiện việc chạy lệnh dưới 1 user khác thông qua plugin become, với các tham số như sau:

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| become: yes/no | Có thực hiện chuyển user hay không |
| become\_user | User muốn chuyển sang, mặc định là root |
| become\_method | Cách chuyển user (su/sudo) |
| become\_flags | Flag đi kèm với phương thức chuyển user. Ví dụ leo quyền root với lệnh sudo -i thì sudo là become\_method và -i là become\_flags |

Khi đó trong khai báo host sẽ có các thông số liên quan đến username và password như sau

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| ansible\_ssh\_user | User sử dụng để SSH vào host |
| ansible\_ssh\_pass | Password của user sử dụng để SSH vào host |
| ansible\_become\_pass | Password dùng để become sau khi đã SSH vào host |

Ví dụ:

Sử dụng become để thực hiện chuyển quyền root theo kiểu chạy lệnh “su - root”

|  |
| --- |
| **become**: **yes**  **become\_method**: su  **become\_flags**: '-l' |

Sử dụng become để thực hiện chuyển quyền root theo kiểu chạy lệnh “sudo -i”

|  |
| --- |
| **become**: **yes**  **become\_method**: sudo  **become\_flags**: '-i' |

### d. Thao tác với user và group

Để thao tác với các user và group trên Linux, Ansible có sẵn các module user và group.

Module user

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| name | Tên user muốn thao tác |
| uid | Đặt uid cho user |
| non\_unique | Mặc định là false, nếu để true thì tự động đổi uid của user về 1 giá trị không trùng với các user còn lại |
| update\_password | Mặc định là always, luôn cập nhật password nếu pass mới khác pass cũ  on\_create: Chỉ set password nếu user là mới tạo |
| umask | Cấu hình umask cho user |
| system | Mặc định là false, nếu để true thì user trở thành system account |
| state | Trạng thái của user  Mặc định là present, nghĩa là trả về user đang có hoặc tạo mới nếu user chưa có  Nếu muốn xóa user thì cấu hình state là absent |
| remove | Nếu state=absent, remove=true sẽ xóa home directory của user (giống lệnh userdel --remove). Mặc định là false |
| force | Chỉ dùng khi state=absent, nếu để là true thì giống lệnh userdel --force là xóa cưỡng bức user và home directory |
| create\_home | Mặc định là true, tạo home directory cho user nếu chưa có hoặc tạo mới user. Nếu không muốn tạo thì đặt là false |
| home | Cấu hình home directory |
| group | Primary group của user |
| groups | Danh sách các group khác mà user được add vào. Chú ý, mặc định nếu append=false thì user sẽ bị xóa khỏi tất cả các group khác được add vào mà không nằm trong groups, do đó nếu muốn thêm vào các groups mới mà không xóa khỏi các group cũ thì append phải bằng true |
| append | Nếu để mặc định bằng false thì user sẽ bị xóa khỏi tất cả các group được add vào mà không nằm trong groups, do đó nếu muốn thêm vào các groups mới mà không xóa khỏi các group cũ thì append phải bằng true |
| comment | Thêm comment mô tả cho user |
| expires | Đặt thời gian expire của user |
| password\_expire\_min | Thời gian tối thiểu để đổi password |
| password\_expire\_max | Thời gian tối đa để đổi password, nếu không password bị expire |
| password | Password của user. Chú ý password này là password đã được hash |
| password\_lock | Khóa password của user lại |
| profile | Cấu hình profile cho user |
| shell | Cấu hình shell mặc định của user |
| ... |  |

Module group

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| name | Tên group muốn thao tác |
| gid | Đặt gid cho group |
| non\_unique | Mặc định là false, nếu để true thì tự động đổi gid của group về 1 giá trị không trùng với các group còn lại |
| state | Trạng thái của group  Mặc định là present, nghĩa là trả về group đang có hoặc tạo mới nếu group chưa có  Nếu muốn xóa group thì cấu hình state là absent |
| system | Mặc định là false, nếu để true thì group trở thành system group |
| local | Trong các môi trường quản trị tập trung, nếu đặt local=false thì group sẽ thành group chung cho cả domain, nếu đặt local=true thì group sẽ là group local của server |

Một số ví dụ

Tạo mới user user1 với home user là /u01/user1 và uid=1001, password là 123456a@

|  |
| --- |
| **vars**:  **usr\_passwd**: "123456a@"  **tasks**:  **- name**: add user  **user**:  **name**: user1  **uid**: 1001  **home**: /u01/user01  **password**: "{{ usr\_passwd | password\_hash('sha512') }}" |

Chèn thêm user1 vào các group employee, it

|  |
| --- |
| **tasks**:  **- name**: edit user  **user**:  **name**: user1  **groups**: employee, it  **append**: **true** |

Cấu hình cho user1 password never expire

|  |
| --- |
| **- name**: edit user  **user**:  **name**: user1  **expires**: -1  **password\_expire\_min**: 0  **password\_expire\_max**: 99999 |

Xóa user falseuser1 và thư mục home directory

|  |
| --- |
| **- name**: delete false user  **user**:  **name**: falseuser  **state**: absent  **remove**: **true** |

Tạo group boss

|  |
| --- |
| **- name**: create group  **group**:  **name**: boss  **gid**: 1500  **state**: present |

Xóa group oldies

|  |
| --- |
| **- name**: delete group  **group**:  **name**: oldies  **state**: absent |

### e. Thao tác với file

Có 3 module thường sử dụng để thao tác với file trên Ansible

Module file: Thao tác với các file trên host từ xa

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| path | Đường dẫn file tương ứng |
| state | absent: Xóa file, với thư mục thì xóa đệ quy giống rm -r  directory: Tạo cả cây thư mục giống mkdir -p  file: Không nói gì thêm thì trả lại trạng thái của path  hard: Tạo hoặc thay đổi hardlink  link: Tạo hoặc thay đổi symbolic link  touch: Tạo file |
| src | Dùng cùng với state hard hoặc link, để chỉ file muốn link đến |
| group | Chỉ định group owner của file |
| owner | Chỉ định user owner của file |
| mode | Quyền của file |
| ... |  |

Module copy dùng để upload 1 file từ host ansible lên host từ xa

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| src | Đường dẫn nguồn  Chú ý, nếu muốn upload cả thư mục từ src vào dest thì cuối đường dẫn src không để dấu /, nếu để / thì nó sẽ copy toàn bộ nội dung bên trong thư mục của src vào dest |
| dest | Đường dẫn đích  Nếu src là thư mục thì dest cũng phải là thư mục |
| remote\_src | Nếu để là true, thì là copy file trong nội bộ host từ xa  Nếu để là false (mặc định) thì là upload file từ src ở host ansible lên dest ở host từ xa |
| backup | Backup trước file ở đường dẫn đích |
| group | Chỉ định group owner của file |
| owner | Chỉ định user owner của file |
| mode | Quyền của file |
| directory\_mode | Nếu copy thư mục, thì set quyền thư mục đích |
| ... |  |

Module fetch dùng để download 1 file từ host từ xa về host ansible

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| src | Đường dẫn nguồn, chỉ được phép là file |
| dest | Đường dẫn đích |
| fail\_on\_missing | Nếu src không tồn tại thì báo lỗi |
| validate\_checksum | Checksum file sau khi download, mặc định là true |

Ví dụ:

Set quyền cho 1 file trên server về quyền 644 và owner user là user1, owner group là employee

|  |
| --- |
| **- name**: Change file ownership, group and permissions  **ansible.builtin.file**:  **path**: /u01/file.txt  **owner**: user1  **group**: employee  **mode**: '0644' |

Tạo ra 1 file và set quyền 644 với owner user1:user1

|  |
| --- |
| **- name**: Touch a file, using symbolic modes to set the permissions (equivalent to 0644)  **ansible.builtin.file**:  **path**: /u01/newfile.txt  **state**: touch  **owner**: user1  **group**: user1  **mode**: u=rw,g=r,o=r |

Change owner 1 thư mục và tất cả các file lẫn thư mục con bên trong (giống như chown -R)

|  |
| --- |
| **- name**: Recursively change ownership of a directory  **ansible.builtin.file**:  **path**: /u01/data  **state**: directory  **recurse**: **yes**  **owner**: user1  **group**: user1 |

Upload thư mục Hardening Centos lên thư mục /root/ của server đích và set quyền là 755.

|  |
| --- |
| **- name**: upload hardening files  **copy**:  **src**: /u01/laboratory/autoscripts/Hardening\_Centos7.x  **dest**: /root/  **mode**: 0755 |

Đẩy lệnh copy trên host từ xa thư mục sang thư mục

|  |
| --- |
| **- name**: copy file on remote server  **copy**:  **src**: /u01/data  **dest**: /u02/  **remote\_src**: **true** |

Nếu muốn download 1 file từ host

|  |
| --- |
| **- name**: Store file from remote server  **ansible.builtin.fetch**:  **src**: /u01/log.txt  **dest**: /tmp/ |

### f. Quản lý dịch vụ trên server

Ansible có sẵn các module để phục vụ quản lý các dịch vụ đang hoạt động trên server.

Với các hệ thống Linux còn sử dụng init (như RHEL6 trở về trước), sử dụng module service với các tham số sau

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| enabled: true/false | Cấu hình cho service được chạy lúc boot server không |
| name | Tên của service |
| runlevel | Cấu hình mức runlevel |
| state: started / stopped / restarted / reloaded | Ứng với việc điều khiển service đó start, stop, restart hay reload |
| sleep | Cấu hình khoảng thời gian giữa lúc stop và start trong trường hợp muốn restart service |
| arguments | Các tham số đi kèm khác |

Với các hệ thống Linux sử dụng systemd thì sử dụng module systemd

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| enabled: true/false | Cấu hình cho service được chạy lúc boot server không |
| name | Tên của service |
| state: started / stopped / restarted / reloaded | Ứng với việc điều khiển service đó start, stop, restart hay reload |
| daemon\_reload: true/false | Thực hiện reload lại danh sách các service trên systemd |
| masked: false/true | Mask 1 service |
| scope | Scope thực hiện thao tác tương ứng, gồm các mức system (mặc định), user, global |

### g. Quản lý các gói trên server

Với các hệ điều hành họ RHEL 7 trở về trước, sử dụng module yum để cấu hình quản lý cài đặt/gỡ bỏ các gói hoặc repo

Một số ví dụ như sau

Cài đặt gói httpd mới nhất. Lưu ý, với state là latest thì nếu trong trường hợp gói phiên bản cũ đã có trên hệ thống, yum sẽ thực hiện nâng cấp lên phiên bản mới nhất.

|  |
| --- |
| **- name**: Install the latest version of Apache  **ansible.builtin.yum**:  **name**: httpd  **state**: latest |

Cài đặt gói httpd với phiên bản >=2.4

|  |
| --- |
| **- name**: Install Apache >= 2.4  **ansible.builtin.yum**:  **name**: httpd>=2.4  **state**: present |

Cài nhiều gói khác nhau

|  |
| --- |
| **- name**: Install a list of packages  **ansible.builtin.yum**:  **name**:  - nginx  - postgresql  - postgresql-server  **state**: present |

Gỡ gói apache

|  |
| --- |
| **- name**: Remove the Apache package  **ansible.builtin.yum**:  **name**: httpd  **state**: absent |

Cài gói apache từ một repo khác tên là testing

|  |
| --- |
| **- name**: Install the latest version of Apache from the testing repo  **ansible.builtin.yum**:  **name**: httpd  **enablerepo**: testing  **state**: present |

Cài gói nginx nhưng sử dụng từ 1 URL ngoài.

|  |
| --- |
| **- name**: Install the nginx rpm from a remote repo  **ansible.builtin.yum**:  **name**: http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6.ngx.noarch.rpm  **state**: present |

Cài 1 gói rpm local

|  |
| --- |
| **- name**: Install nginx rpm from a local file  **ansible.builtin.yum**:  **name**: /usr/local/src/nginx-release-centos-6-0.el6.ngx.noarch.rpm  **state**: present |

Nâng cấp tất cả các gói

|  |
| --- |
| **- name**: Upgrade all packages  **ansible.builtin.yum**:  **name**: "\*"  **state**: latest |

Chỉ download gói, không cài

|  |
| --- |
| **- name**: Download the nginx package but do not install it  **ansible.builtin.yum**:  **name**:  - nginx  **state**: latest  **download\_only**: **true** |

Với các hệ điều hành Ubuntu, sử dụng module apt

Một số ví dụ như sau

Cài đặt gói apache

|  |
| --- |
| **- name**: Install apache httpd (state=present is optional)  **ansible.builtin.apt**:  **name**: apache2  **state**: present |

Cài đặt gói với phiên bản xác định

|  |
| --- |
| **- name**: Install the version '1.00' of package "foo"  **ansible.builtin.apt**:  **name**: foo=1.00 |

Cài nhiều gói khác nhau

|  |
| --- |
| **- name**: Install a list of packages  **ansible.builtin.apt**:  **pkg**:  - foo  - foo-tools |

Gỡ gói

|  |
| --- |
| **- name**: Remove "foo" package  **ansible.builtin.apt**:  **name**: foo  **state**: absent |

Cài đặt 1 gói trên local server

|  |
| --- |
| **- name**: Install a .deb package  **ansible.builtin.apt**:  **deb**: /tmp/mypackage.deb |

Nâng cấp tất cả các gói

|  |
| --- |
| **- name**: Update all packages to their latest version  **ansible.builtin.apt**:  **name**: "\*"  **state**: latest |

Nâng cấp version OS

|  |
| --- |
| **- name**: Upgrade the OS (apt-get dist-upgrade)  **ansible.builtin.apt**:  **upgrade**: dist |

### h. Thu thập thông tin host

Xét playbook sau

|  |
| --- |
| ---  **- name**: Misc  **hosts**: TEST  **connection**: ssh  **become**: **no**  **serial**: 1  **gather\_facts**: **yes**  **tasks**:  **- debug**: var=ansible\_facts |

Playbook này rất đơn giản, chỉ thực hiện 1 plugin là gather\_facts và đưa kết quả gather\_facts ra màn hình. Tuy nhiên plugin gather\_facts của Ansible thực hiện thu thập rất nhiều thông tin liên quan đến remote host đang gọi đến, trong số đó có thể kể đến 1 số thông tin đáng chú ý như sau:

|  |  |
| --- | --- |
| **Tham số kết quả** | **Ý nghĩa** |
| all\_ipv4\_addresses | Danh sách các IPv4 được cấu hình |
| all\_ipv6\_addresses | Danh sách các IPv6 được cấu hình |
| architecture | Kiến trúc OS |
| bios\_date | Ngày phát hành phiên bản BIOS |
| bios\_version | Phiên bản BIOS |
| default\_ipv4 | IPv4 được chọn làm default |
| default\_ipv6 | IPv6 được chọn làm default |
| device\_links | Thông tin về các device link đang có trên host |
| devices | Thông tin về tất cả các device vật lý đang có trên host |
| ensxx/bondxx | Thông tin về các interface network, chi tiết đến mức khai báo IP, speed, trạng thái, MAC... |
| interfaces | List danh sách các interface network |
| kernel | Thông tin version kernel |
| memfree\_mb | Dung lượng memory còn trống |
| memory\_mb | Dung lượng chi tiết các memory (RAM, swap...) |
| mounts | Thông tin các mountpoint |
| processor | Thông tin CPU |
| v.v... |  |

Các kết quả từ gather\_facts có thể sử dụng làm đầu vào cho các checklist, hoặc ra các quyết định tương ứng khi sử dụng đến module condition (sẽ được đề cập ở các phần sau).

### i. Cấu hình iptables

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| action | Mặc định là append, nghĩa là để xuống dưới cùng của chain. Tuy nhiên để xuống cuối có thể dẫn đến hiện tượng không ăn rule, nên cấu hình action=insert để rule được chồng lên trên cùng của chain. Ứng với iptables -I hoặc iptables -A |
| chain | Chain muốn sử dụng. Mặc định iptables có các chain INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING, SECMARK, CONNSECMARK |
| state | Mặc định là present, nghĩa là add mới hoặc trả về nếu tồn tại. Nếu muốn xóa, cấu hình state=absent |
| comment | Comment cho rule |
| ip\_version | Sử dụng ipv4 (mặc định) hoặc ipv6 |
| policy | Chính sách của rule, bao gồm ACCEPT, DROP, QUEUE, RETURN |
| destination | Đích, ứng với tham số -d <destination\_address> của iptables. Có thể là IP lẻ, IP dải hoặc hostname |
| destination\_port | Port destination đơn |
| destination\_ports | Danh sách các port destination. Bao gồm các dãy port đơn, hoặc nếu là dải port thì cấu hình là <port1:port2> |
| dst\_range | Cấu hình dải IP destination |
| source | Nguồn, ứng với tham số -s <source\_address> của iptables. Có thể là IP lẻ, IP dải hoặc hostname |
| source\_port | Port source đơn |
| src\_range | Cấu hình dải IP source |
| jump | Target cho rule, ứng với tham số -j của iptables (ACCEPT, DROP, LOG...) |
| log\_level | Cấu hình loglevel, từ 0-7 hoặc theo mô tả của iptables. Chú ý phải có cấu hình jump=log |
| log\_prefix | Prefix cho log. Chú ý phải có cấu hình jump=log |
| rule\_num | Vị trí đặt rule (bằng số) |
| ... | Các cấu hình khác có thể xem thêm tại https://docs.ansible.com/ansible/latest/ collections/ansible/builtin/iptables\_module.html |

Một số ví dụ cấu hình iptables bằng module iptables:

Mở inbound các port HTTP 80, 443

|  |
| --- |
| **- name**: Allow HTTP/HTTPS connections  **ansible.builtin.iptables**:  **chain**: INPUT  **protocol**: tcp  **action**: insert  **destination\_ports**:  - "80"  - "443"  **jump**: ACCEPT |

Mở inbound allow SSH 1 host nào đó

|  |
| --- |
| **- name**: Allow SSH for 1 host  **ansible.builtin.iptables**:  **chain**: INPUT  **protocol**: tcp  **action**: insert  **source**: 10.0.0.1  **destination\_port**: 22  **jump**: ACCEPT  **comment**: Allow SSH for host |

Mở inbound allow FTP passive 1 dải server

|  |
| --- |
| **- name**: Allow FTP passive for a range of host  **ansible.builtin.iptables**:  **chain**: INPUT  **protocol**: tcp  **action**: insert  **src\_range**: 10.0.0.128/27  **destination\_ports**:  - "21"  - "8000:10000"  **jump**: ACCEPT  **comment**: Allow FTP passive for a range of host |

Mở outbound đến HTTP/HTTPS web

|  |
| --- |
| **- name**: Allow connect HTTP/HTTPS to remote web server  **ansible.builtin.iptables**:  **chain**: OUTPUT  **protocol**: tcp  **action**: insert  **destination**: 10.0.0.1  **destination\_ports**:  - "80"  - "443"  **jump**: ACCEPT  **comment**: Allow host to connect to web server |

Mở outbound đến 1 dải server port FTP passive

|  |
| --- |
| **- name**: Allow connect passive FTP to remote FTP server  **ansible.builtin.iptables**:  **chain**: OUTPUT  **protocol**: tcp  **action**: insert  **dst\_range**: 10.0.0.128/27  **destination\_ports**:  - "21"  - "8000:10000"  **jump**: ACCEPT  **comment**: Allow connect passive FTP to remote FTP server |

Lưu lại cấu hình iptables vào file (của RHEL7)

|  |
| --- |
| **- name**: Save current state of the firewall in system file  **community.general.iptables\_state**:  **state**: saved  **path**: /etc/sysconfig/iptables |

### j. Crontab

Các tham số có thể sử dụng với module crontab của Ansible như sau

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| name | Tên của job crontab, khi đưa vào host sẽ có thêm 1 dòng comment Ansible: <crontab name>. Khuyến cáo nên đặt tên job cụ thể. |
| backup: false/true | Backup file crontab trước khi thực hiện. Mặc định là false |
| disabled | Có disable lệnh crontab tương ứng không, mặc định là false |
| user | Sửa trên crontab của user nào. Mặc định là user hiện hành |
| cron\_file | Đường dẫn file crontab của user. Mặc định là /etc/cronttab. Muốn sửa entry này phải có entry user |
| minute | Phút chạy lệnh cron, từ 0-59, \* (mỗi phút 1 lần), \*/x (x phút 1 lần). Mặc định là \* |
| hour | Giờ chạy lệnh cron, từ 0-23, hoặc \* (hàng giờ), \*/x (x giờ 1 lần). Mặc định là \* |
| day | Ngày chạy lệnh cron, từ 1-31, hoặc \* (hàng ngày), \*/x (x ngày 1 lần). Mặc định là \*  Chi tiết xem hướng dẫn lệnh crontab của Linux |
| month | Tháng chạy lệnh cron, từ 1-12, \* (hàng tháng), \*/x (x tháng 1 lần). Mặc định là \* |
| weekday | Chạy ở ngày thứ mấy trong tuần (0 là chủ nhật, 6 là thứ 7, \* là hàng ngày) |
| special\_time | Dạng thức thời gian đặc biệt, bao gồm  annualy, daily (phút đầu tiên của ngày), hourly (phút đầu tiên của giờ), monthly (phút đầu tiên của tháng), reboot (cứ reboot là chạy), weekly (phút đầu tiên của tuần), yearly (phút đầu tiên của năm) |
| job | Lệnh chạy |
| env | Set biến môi trường cho crontab, mặc định là false |
| state | Trạng thái của job crontab tương ứng. Mặc định là present để add job crontab. Nếu muốn xóa job crontab, đặt state là absent. Phải có name của job tương ứng thì state absent mới xóa được |

Một số ví dụ:

Đồng bộ thời gian với NTP server cứ 5 phút 1 lần

|  |
| --- |
| **tasks**:  **- name**: add ntp crontabs  **cron**:  **name**: "sync ntp server 5 minutes"  **minute**: "\*/5"  **job**: "ntpdate -u 192.168.181.50" |

Kiểm tra lại các mount point sau khi reboot server

|  |
| --- |
| **tasks**:  **- name**: add crontabs  **cron**:  **name**: "add mountpoint check after reboot"  **special\_time**: reboot  **job**: "df -h" |

Disable job đồng bộ NTP

|  |
| --- |
| **- name**: add crontabs  **cron**:  **name**: "add ntp crontab"  **backup**: **true**  **minute**: "\*/5"  **job**: "ntpdate -u pool.ntp.org"  **disabled**: **true** |

Xóa 1 job có tên là “old job”

|  |
| --- |
| **- name**: 'Ensure an old job is no longer present. Removes any job that is prefixed by "#Ansible: an old job" from the crontab'  **ansible.builtin.cron**:  **name**: "an old job"  **state**: absent |

Ngoài các module kể trên, Ansible còn có rất nhiều module khác phục vụ cài đặt và cấu hình hệ thống server Linux. Người đọc có thể tìm hiểu thêm các module sau để phục vụ trong quá trình quản trị server hàng ngày:

- Quản lý phân vùng với các module parted (cho phân vùng vật lý), lvg/lvol (cho LVM), <https://www.redhat.com/sysadmin/automating-logical-volume-manager>, <https://docs.ansible.com/ansible/latest/collections/community/general/parted_module.html>

- Cấu hình kernel sysctl với module sysctl, <https://docs.ansible.com/ansible/latest/collections/ansible/posix/sysctl_module.html>

- Cấu hình user limit (open file, process...), <https://docs.ansible.com/ansible/latest/collections/community/general/pam_limits_module.html>

(v.v…)

## 4. Sử dụng Ansible thao tác với server Windows

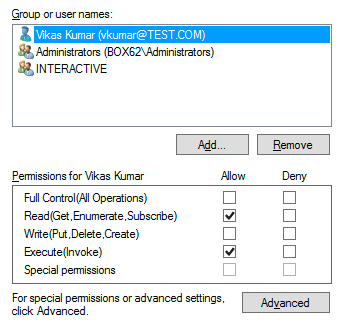
### a. Cấu hình giao thức WinRM

Để có thể thao tác với server Windows server từ xa, có thể sử dụng giao thức WinRM, bắt đầu được đưa vào từ Windows Server 2008. WinRM cho phép người quản trị có thể thao tác từ xa vào server Windows thông qua windows command hoặc powershell. Mặc định WinRM sử dụng port 5985 cho HTTP và 5986 cho HTTPS.

Trong file inventory, khai báo các biến như sau để Ansible thao tác với các server qua WinRM, lưu ý cần thông port 5985 và 5986 của server:

|  |
| --- |
| [all:vars]  ansible\_connection=winrm  ansible\_port=5985  ansible\_winrm\_server\_cert\_validation=ignore  ansible\_winrm\_transport=ntlm  ansible\_user=<user có quyền thao tác với winrm>  ansible\_password=<password của user> |

Nếu không sử dụng được user Administrator của server để thao tác, cần sử dụng các user có quyền thao tác với WinRM để cấu hình trong Ansible. Sử dụng lệnh winrm configSDDL default và cấp các quyền sau cho user tương ứng:



### b. Thực hiện lệnh với module win\_command hoặc win\_shell

Trong file playbook, khai báo các task như sau, ví dụ bên dưới là chạy lệnh netstat cho server Windows:

|  |
| --- |
| ---  - name: windows server playbook  hosts: WINDOWS  tasks:  - name: check netstat  win\_command: netstat -e  register: out  - debug: var=out |

Nếu muốn sử dụng Powershell, có thể sử dụng 1 trong 2 cách sau:

- Dùng win\_command gọi lệnh Powershell

|  |
| --- |
| - name: Run remote PowerShell Script  win\_command: powershell.exe -ExecutionPolicy ByPass -File C:/temp/powershellscript.ps1 |

- Dùng module win\_shell thay cho win\_command nếu muốn sử dụng các thao tác redirect

|  |
| --- |
| - name: Run remote PowerShell script  win\_shell: C:\somescript.ps1 >> c:\somelog.txt |

Cũng giống như SSH server Linux, khi thao tác với server Windows qua module win\_command hoặc win\_shell có thể thu thập các thông tin kết quả trả về qua biến register, các thành phần có trong biến register của 2 module này như sau:

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| cmd | Lệnh đã chạy |
| stdout | Kết quả đầu ra tiêu chuẩn của lệnh |
| stderr | Kết quả đầu ra lỗi của lệnh (nếu lệnh có lỗi) |
| stdout\_lines | Cũng như stdout nhưng biểu diễn thành từng dòng |
| stderr\_lines | Cũng như stderr nhưng biểu diễn thành từng dòng |
| start | Thời gian bắt đầu chạy lệnh |
| end | Thời gian kết thúc chạy lệnh |
| delta | Thời gian để thực thi lệnh (end-start) |
| rc | Mã lỗi trả về, nếu thành công thì rc = 0 |

### c. Thao tác với file

Để phân quyền cho file hoặc thư mục, có thể sử dụng module win\_acl

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| path | Đường dẫn |
| state | Mặc định là thêm (present), nếu muốn xóa thì cấu hình state=absent |
| type | Loại rule, allow hoặc deny |
| user | User hoặc group được áp dụng rule |
| rights | Quyền tương ứng. Ở Windows có các quyền sau: FullControl, Read, Write, Modify, Delete, ListDirectory, ExecuteFile... (chi tiết có thể xem tại https://msdn.microsoft.com/en-us/library/system.security.accesscontrol.filesystemrights.aspx) |

Để thực hiện thao tác tạo/thêm/sửa/xóa 1 file hoặc thư mục, có thể sử dụng module win\_file

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| path | Đường dẫn |
| state | Trạng thái  directory: Tạo thư mục  touch: Tạo file nếu file chưa tồn tại  absent: Xóa file, hoặc xóa cả thư mục và nội dung bên trong |

Để thực hiện copy 1 file từ host Ansible lên host Windows từ xa có thể sử dụng module win\_copy

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| src | Đường dẫn nguồn  Chú ý, nếu muốn upload cả thư mục từ src vào dest thì cuối đường dẫn src không để dấu /, nếu để / thì nó sẽ copy toàn bộ nội dung bên trong thư mục của src vào dest |
| dest | Đường dẫn đích  Nếu src là thư mục thì dest cũng phải là thư mục |
| remote\_src | Nếu để là true, thì là copy file trong nội bộ host từ xa  Nếu để là false (mặc định) thì là upload file từ src ở host ansible lên dest ở host từ xa |
| backup | Backup trước file ở đường dẫn đích |
| ... |  |

Chú ý: Nếu muốn copy 1 thư mục lên server Windows, thì ở cuối đường dẫn trong src không để dấu “/” (nếu để thì copy toàn bộ file bên trong thư mục đó vào dest, chứ không copy thư mục)

Ví dụ:

|  |
| --- |
| - win\_copy:  src: /root/test/resources/powershellscript.ps1  dest: 'C:\temp\' |

Cũng có thể dùng module fetch để download file từ server Windows

|  |
| --- |
| - name: Store file into /tmp/fetched/host.example.com/tmp/somefile  ansible.builtin.fetch:  src: /tmp/somefile  dest: /tmp/fetched |

### d. Cấu hình Windows Firewall

Sử dụng module win\_firewall\_rule cho việc cấu hình firewall rule trên Windows Firewall.

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| state | Trạng thái của rule. Mặc định là present để tạo mới hoặc trả kết quả rule đã tồn tại. Nếu muốn xóa, để giá trị là absent |
| action | Rule được tạo ra để allow (mặc định) kết nối hoặc block kết nối |
| enabled | Mặc định là true để enable rule. Nếu giá trị là false thì rule đó bị disable |
| description | Mô tả về rule |
| name | Tên hiển thị của rule |
| group | Group name |
| direction | in là dành cho cấu hình rule inbound, out là cấu hình dành cho rule outbound |
| localip | Local IP address mà rule muốn áp dụng. Mặc định là any để áp dụng cho tất cả các local IP |
| localport | Local port mà rule muốn áp dụng. Mặc định là any để áp dụng cho tất cả các local port |
| remoteip | Remote IP address mà rule muốn áp dụng. Mặc định là any để áp dụng cho tất cả các remote IP |
| remoteport | Remote port mà rule muốn áp dụng. Mặc định là any để áp dụng cho tất cả các remote port |
| profile | Profile mà rule áp dụng  Ở Windows có 3 profile network là domain, private, public. Nếu để mặc định thì rule này được áp dụng cho cả 3 profile, khuyến cáo nên để mặc định |
| program | Program mà rule áp dụng. Mặc định là any, nghĩa là chương trình nào cũng áp dụng rule tương ứng |
| protocol | Giao thức được áp dụng. Mặc định là any, nghĩa là rule áp dụng cho tất cả các protocol |
| ... |  |

Ví dụ:

Disable rule mặc định mở remote desktop any

|  |
| --- |
| **- name**: disable default remote any rule  **community.windows.win\_firewall\_rule**:  **name**: Remote Desktop - User Mode (TCP-In)  **state**: present  **enabled**: **no**  **- name**: disable default remote any rule  **community.windows.win\_firewall\_rule**:  **name**: Remote Desktop - User Mode (UDP-In)  **state**: present  **enabled**: **no** |

Enable rule ICMP, để đảm bảo ping được host Windows

|  |
| --- |
| **- name**: enable icmp v4 rule  **community.windows.win\_firewall\_rule**:  **name**: File and Printer Sharing (Echo Request - ICMPv4-In)  **state**: present  **enabled**: **yes**  **- name**: enable icmp v6 rule  **community.windows.win\_firewall\_rule**:  **name**: File and Printer Sharing (Echo Request - ICMPv6-In)  **state**: present  **enabled**: **yes** |

Cấu hình allow remote 1 số IP nhất định

|  |
| --- |
| **- name**: allow remote for specific ip  **community.windows.win\_firewall\_rule**:  **name**: Allow Remote Client  **state**: present  **enabled**: **yes**  **action**: allow  **localport**: 3389  **direction**: in  **protocol**: tcp  **localip**:  - 10.0.0.1  - 192.168.100.2 |

Mở outbound cho phép host Windows kết nối đến 1 web server

|  |
| --- |
| **- name**: allow connect to web site  **community.windows.win\_firewall\_rule**:  **name**: Allow Web site  **state**: present  **enabled**: **yes**  **action**: allow  **remoteport**: 80  **direction**: out  **protocol**: tcp  **remoteip**: 10.0.1.100 |

## 5. Sử dụng Ansible gọi HTTP request

Nhiều ứng dụng hiện nay sử dụng giao thức chính để giao tiếp với người dung hoặc các ứng dụng khác là HTTP, bản thân hạ tầng phần cứng của nhiều loại thiết bị như server, storage v.v… cũng cung cấp các API để giúp thao tác với các thiết bị. Ansible cũng cho phép hỗ trợ việc thao tác thông qua HTTP request.

Khi sử dụng Ansible gọi HTTP request, trong file playbook và file inventory, sử dụng connection là local và module uri, lúc này Ansible sẽ gọi connection HTTP tương ứng với URL được khai báo trong mục URI.

|  |
| --- |
| **connection**: local |

Từ bên dưới đây ta định nghĩa việc gọi HTTP request sử dụng 2 biến là ansible\_user cho username và ansible\_password cho password. Như vậy trong file inventory ta có thể cấu hình theo ví dụ như sau:

|  |
| --- |
| [all:vars]  ansible\_connection=local  [ILOSVRS]  10.0.0.1 ansible\_user=<username> ansible\_password=<password> |

Dưới đây là 1 số use case sử dụng Ansible cho HTTP request, sử dụng cho việc gọi ILO API restful của server HPE. Với các server khác hoặc HTTP request khác thì điều chỉnh lại các tham số cho phù hợp.

- Gọi phương thức HTTPS GET, với kết quả được trả về dạng JSON. Cấu hình validate\_certs là false để tạm thời bỏ qua việc xác thực cert HTTPS.

|  |
| --- |
| ---  **- name**: Auto iLO Configuration  **hosts**: ILOSVRS  **connection**: local  **become**: **no**  **gather\_facts**: **no**  **tasks**:  **- name**: get bios info  **uri**:  **url**: "https://{{ inventory\_hostname }}/redfish/v1/Systems/1/bios"  **method**: GET  **validate\_certs**: False  **url\_username**: "{{ ansible\_user }}"  **url\_password**: "{{ ansible\_password }}"  **headers**:  **Accept**: "application/json"  **force\_basic\_auth**: **yes**  **return\_content**: **yes**  **register**: response  **- name**: debug  **debug**:  **var**: response |

- Gọi phương thức HTTPS POST, dữ liệu đầu vào định dạng JSON và kết quả trả về là JSON. Với phương thức PUT hoặc PATCH cũng có thể thực hiện tương tự.

|  |
| --- |
| ---  **- name**: Auto iLO Configuration  **hosts**: ILOSVRS  **connection**: local  **become**: **no**  **gather\_facts**: **no**  **vars**:  **json\_body**: |  **{**  **"Image": "http://10.0.0.2/iso/os.iso"**  **}**  **tasks**:  **- name**: mount iso to host  **uri**:  **url**: "https://{{ inventory\_hostname }}/redfish/v1/Managers/1/VirtualMedia/2/Actions/VirtualMedia.InsertMedia"  **method**: POST  **validate\_certs**: False  **url\_username**: "{{ ansible\_user }}"  **url\_password**: "{{ ansible\_password }}"  **headers**:  **Accept**: "application/json"  **Content-Type**: "application/json"  **Cache-Control**: "no-cache"  **force\_basic\_auth**: **yes**  **return\_content**: **yes**  **body\_format**: json  **body**: "{{ json\_body }}"  **register**: response  **- name**: debug  **debug**:  **var**: response |

Khi register kết quả trả về của API ra biến, biến register bao gồm một số thành phần như sau:

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| allow | Phương thức đã cho thực hiện |
| ansible\_facts | Một số thông tin về các fact tương ứng request đã thực hiện |
| cache\_control | Cache control của request |
| changed | Có change không |
| connection | Trạng thái kết nối sau khi thực hiện request |
| content | Nội dung trả về |
| content\_type | Dạng thức nội dung trả về |
| cookies | Danh mục các cookie |
| date | Thời gian thực hiện |
| failed | Request có bị fail không |
| json | Kết quả trả về dạng json |
| msg | Message trả về |
| redirected | Request có bị redirect không |
| status | Mã lỗi HTTP trả về |
| url | URL đã thực hiện request |

Bên cạnh SSH và HTTP request là các giao thức phổ biến thường được sử dụng với Ansible, Ansible còn hỗ trợ nhiều giao thức khác để thao tác với các thiết bị. Người đọc có thể tham khảo thêm thông tin về các giao thức hỗ trợ trên trang tài liệu của Ansible là docs.ansible.com

## 6. Một số cú pháp/module dựng sẵn sử dụng trong playbook của Ansible

Trong playbook của Ansible, có thể sử dụng một số cú pháp sau đây để giúp điều khiển thực hiện các task theo mong muốn của mình

### a. Khai báo biến

Cũng giống như trong nhiều ngôn ngữ lập trình khác, Ansible cho phép định nghĩa các biến để truyền giá trị cụ thể vào trong playbook tùy theo từng hoàn cảnh mà không cần phải viết lại toàn bộ playbook. Có 2 cách để thực hiện khai báo biến trong 1 playbook Ansible:

Khai báo trực tiếp trong playbook tương ứng

|  |
| --- |
| **vars**:  **ifname**: ens33  **tasks**:  **- name**: check speed of interface  **shell**: ethtool {{ ifname }}  **register**: out  **- debug**: var=out |

Có một số trường hợp, ta muốn nhập giá trị của biến trực tiếp trong quá trình chạy playbook, có thể khai báo vars\_prompt như sau:

|  |
| --- |
| **vars\_prompt**:  **- name**: ifname  **prompt**: Enter interface name  **private**: **no**  **tasks**:  **- name**: check speed of interface  **shell**: ethtool {{ ifname }}  **register**: out  **- debug**: var=out |

Ở ví dụ trên, tên biến được định nghĩa là ifname, khi chạy playbook dấu nhắc dòng lệnh sẽ thông báo “Enter interface name”. Private để no thì các giá trị sẽ hiển thị trên màn hình, còn nếu để yes thì không hiển thị.

Ngoài cách khai báo trực tiếp trong file playbook, ta có thể khai báo biến ở một file khác, rồi trong playbook include đến file đó

|  |
| --- |
| Khai báo trong file ifvars.yml  ---  **ifname**: ens33  Khai báo trong playbook gọi tham chiếu đến file ifvars.yml  **vars\_files**:  /u01/data/ansible/playbook/ifvars.yml  **tasks**:  **- name**: check speed of interface  **shell**: ethtool {{ ifname }}  **register**: out  **- debug**: var=out |

Khi sử dụng với file inventory, các biến được định nghĩa trong file inventory cũng có thể sử dụng sang playbook. Khi soạn inventory, ta nên lưu ý một số biến mặc định sau đây thường đi kèm với inventory:

|  |  |
| --- | --- |
| **Biến** | **Ý nghĩa** |
| ansible\_hostname | Hostname của host. Nếu không tách hostname và IP ra riêng, thì ansible\_hostname cũng là IP của host |
| ansible\_host | IP của host |
| ansible\_port | Port của host |
| ansible\_user | User login (SSH) của host |
| ansible\_ssh\_pass | Password login SSH của host ứng với ansible\_user |
| ansible\_become | Có chuyển user trong lúc chạy không |
| ansible\_become\_method | Phương thức become |
| ansible\_become\_user | User become sang |
| ansible\_become\_pass | Password của user được become sang |
| ansible\_become\_flags | Flag bổ sung thêm để thực hiện become |
| … |  |

### b. Thực hiện task bỏ qua lỗi nếu có

Mặc định, các task trong Ansible được thực hiện theo quy tắc tuần tự từ trên xuống dưới trong playbook, task trước thành công thì task sau mới chạy, nếu trong quá trình thực hiện trên 1 host, có 1 task nào đó bị lỗi thì dừng lại và không thực hiện tiếp các task còn lại với host tương ứng. Trong trường hợp nếu muốn chạy tiếp và bỏ qua các lỗi thì bổ sung thêm cấu hình ignore\_errors như sau:

|  |
| --- |
| **- name**: check ubuntu version  **shell**: lsb\_release -d  **register**: ubuntu\_version  **ignore\_errors**: **yes** |

### c. Thực hiện song song các task trong playbook

Nếu muốn thực hiện các task nào đó song song với nhau (thay vì tuần tự theo mặc định của Ansible), sử dụng 2 cấu hình async và poll trong task như sau:

|  |
| --- |
| **tasks:**  **-** **name:** Simulate long running op (15 sec), wait for up to 45 sec, poll every 5 sec  **ansible.builtin.command:** /bin/sleep 15  **async:** 45  **poll:** 5 |

Nếu muốn hoàn toàn song song thì để poll=0, còn nếu poll>0 thì sẽ mất 1 khoảng thời gian poll kiểm tra kết nối tính bằng giây tương ứng. Thời gian được cấu hình trong async là thời gian tối đa để 1 lệnh được coi là timeout.

### d. Thực hiện một task theo điều kiện xác định

Có thể có một số trường hợp, ta muốn 1 task nào đó chỉ được thực hiện khi đạt một điều kiện nào đó. Có thể sử dụng tham số when như sau

|  |
| --- |
| **- name**: check red hat version  **shell**: cat /etc/redhat-release  **register**: rhel\_version  **ignore\_errors**: **yes**  **- name**: echo when red hat  **shell**: echo "This is Red Hat family"  **register**: out  **when**: rhel\_version is succeeded |

Trong trường hợp trên, task “echo when red hat” thực hiện sau “check red hat version”, dựa vào việc kiểm tra kết quả thực hiện lệnh được lưu vào biến register rhel\_version, nếu kết quả trả về thành công thì task “echo when red hat” mới được thực hiện.

Hoặc có thể điều kiện là 1 xâu nào đó xuất hiện trong kết quả stdout:

|  |
| --- |
| **-** **name:** Register a variable  **ansible.builtin.shell:** cat /etc/motd  **register:** motd\_contents  **-** **name:** Use the variable in conditional statement  **ansible.builtin.shell:** echo "motd contains the word hi"  **when:** motd\_contents.stdout.find('hi') != -1 |

Nếu cấu hình gather\_facts để lấy thông tin về host tương ứng, cũng có thể dựa vào ansible\_fact để quyết định 1 task có thực hiện hay không

|  |
| --- |
| **tasks:**  **-** **name:** Shut down CentOS 6 systems  **ansible.builtin.command:** /sbin/shutdown -t now  **when:**  **-** ansible\_facts['distribution'] == "CentOS"  **-** ansible\_facts['distribution\_major\_version'] == "6" |

Hoặc thực hiện khi có 1 biến nào đó được định nghĩa hoặc không

|  |
| --- |
| **tasks:**  **-** **name:** Run the command if "foo" is defined  **ansible.builtin.shell:** echo "I've got '{{ foo }}' and am not afraid to use it!"  **when:** foo is defined  **-** **name:** Fail if "bar" is undefined  **ansible.builtin.fail:** msg="Bailing out. This play requires 'bar'"  **when:** bar is undefined |

v.v...

Xem thêm các cú pháp sử dụng điều kiện tại đây: https://docs.ansible.com/ansible/latest/playbook\_guide/playbooks\_conditionals.html

### e. Thực hiện một task theo vòng lặp

Trong trường hợp nếu muốn thực hiện lặp lại 1 task theo một điều kiện hoặc 1 tập hợp nào đó, ta có thể sử dụng cấu hình loop như sau

|  |
| --- |
| **tasks**:  **- name**: changing the become user's password and ssh user's password  **user**:  **name**: "{{ item.username }}"  **update\_password**: always  **password**: "{{ item.password | password\_hash('sha512') }}"  **loop**:  **- { username**: "{{ ansible\_become\_user }}", password: "{{ new\_become\_pass }}" }  **- { username**: "{{ ansible\_ssh\_user }}", password: "{{ new\_ssh\_pass }}" } |

Trong playbook trên, task thực hiện đổi password sẽ được thực hiện 2 lần với các username tương ứng được định nghĩa trong file inventory. Đó là ví dụ về 1 loop dựa trên 1 danh sách các item có sẵn.

Có thể, list item đó không cần phải định nghĩa ở trong task, mà được lấy ra từ 1 list được khai báo trước đó

|  |
| --- |
| **-** **name:** Non-optimal yum, slower and may cause issues with interdependencies  **ansible.builtin.yum:**  **name:** "{{ **item** }}"  **state:** present  **loop:** "{{ **list\_of\_packages** }}" |

Cũng có thể không phải item nào trong list loop cũng thỏa mãn, mà phải thỏa mãn 1 điều kiện nào đó mới được tính để đưa vào loop:

|  |
| --- |
| **tasks:**  **-** **name:** Run with items greater than 5  **ansible.builtin.command:** echo {{ item }}  **loop:** **[** **0,** **2,** **4,** **6,** **8,** **10** **]**  **when:** item > 5 |

Trong trường hợp nếu muốn 1 task thực hiện lặp đi lặp lại đến khi có 1 điều kiện được thỏa mãn, có thể sử dụng tham số until như sau:

|  |
| --- |
| **-** **name:** Retry a task until a certain condition is met  **ansible.builtin.shell:** /usr/bin/foo  **register:** result  **until:** result.stdout.find("all systems go") != -1  **retries:** 5  **delay:** 10 |

Trong quá trình loop, có thể bổ sung thêm loop\_control để điều khiển quá trình loop theo mong muốn của mình. Chẳng hạn như dưới đây là điều khiển để trong vòng loop, các lệnh được thực hiện cách nhau 3 giây

|  |
| --- |
| **-** **name:** Create servers, pause 3s before creating next  **community.digitalocean.digital\_ocean:**  **name:** "{{ **item** }}"  **state:** present  **loop:**  **-** server1  **-** server2  **loop\_control:**  **pause:** 3 |

Để theo dõi xem mình đã chạy đến index bao nhiêu (mặc định index bắt đầu từ 0)

|  |
| --- |
| **-** **name:** Count our fruit  **ansible.builtin.debug:**  **msg:** "{{ **item** }}withindex{{ **my\_idx** }}"  **loop:**  **-** apple  **-** banana  **-** pear  **loop\_control:**  **index\_var:** my\_idx |

v.v…

Xem thêm các cú pháp sử dụng vòng lặp tại đây: <https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html>

### f. Handler

Handler là một dạng task đặc biệt được 1 task khác trigger đến sau khi task đó được thực hiện xong. Ví dụ như khi chúng ta update cấu hình, cần restart dịch vụ trên server thì được định nghĩa là “task” update cấu hình “trigger” cho “handler” restart dịch vụ được thực hiện. Việc trigger handler được thực hiện theo trình tự các task có việc trigger tương ứng đó, và cho dù có nhiều task trigger vào handler thì handler đó cũng chỉ chạy 1 lần. Ví dụ bên dưới là sau khi update xong gói DHCP và update xong cấu hình cho DHCP server thì trigger việc restart DHCP server.

|  |
| --- |
| ---  **- hosts**: dhcp  **tasks**:  **- name**: update to latest DHCP  **yum**:  **name**: dhcp  **state**: latest  **notify**: restart dhcp  **- name**: copy the DHCP config  **copy**:  **src**: dhcp/dhcpd.conf  **dest**: /etc/dhcp/dhcpd.conf  **notify**: restart dhcp  **- name**: start DHCP at boot  **service**:  **name**: dhcpd  **state**: started  **enabled**: **yes**  **handlers**:  **- name**: restart dhcp  **service**:  **name**: dhcpd  **state**: restarted |

## 7. Templating

### a. Template cấu hình

Trong nhiều trường hợp, chúng ta không chỉ cần chạy một hoặc nhiều lệnh cho nhiều server, mà còn cần phải thực hiện soạn cấu hình cho nhiều server, điều này thường gặp khi thực hiện cấu hình các dịch vụ chạy trên server như web, DB, process v.v... với các cấu hình thay đổi theo đặc thù của mỗi server. Ansible cho phép thực hiện việc thiết kế ra các template file cấu hình trước, và khi deploy vào các server thì thay thế các variable ứng theo đặc điểm của từng server và hỗ trợ ngôn ngữ biểu diễn template Jinja2 (có thể xem lại một số cú pháp cơ bản của Jinja2 ở mục C).

Giả sử chúng ta có 1 số web server apache cần nạp 1 trang index.html mới có nội dung như sau:

A screenshot of a computer

Description automatically generated with medium confidence

Trong đó thay thế IP 172.16.11.12 bằng IP tương ứng của server đích

Đầu tiên ta soạn 1 file template index.html.j2 như sau

|  |
| --- |
| <html>  <body>  <h1>**Hello world, I am {{ ansible\_host }}** </h1>  </body>  </html> |

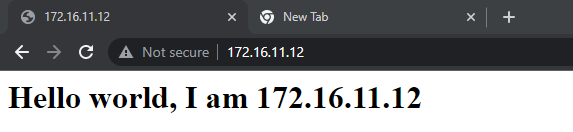
Playbook chúng ta tạo như sau

|  |
| --- |
| ---  **- name**: Misc  **hosts**: TEST  **connection**: ssh  **become**: **yes**  **serial**: 1  **become\_method**: sudo  **become\_flags**: '-i'  **gather\_facts**: **no**  **tasks**:  **- name**: replace template to index file  **template**:  **src**: /u01/data/test\_web/index.html.j2  **dest**: /var/www/html/index.html  **backup**: **true**  **- name**: restart apache  **systemd**:  **name**: apache2  **state**: restarted |

Sử dụng lệnh ansible-playbook để chạy playbook cho các host

|  |
| --- |
| ansible-playbook -i inventory/host\_test playbook/linux\_test\_web.yml |

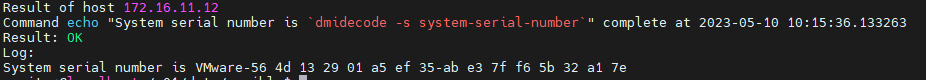
Kết quả chạy playbook thành công, kiểm tra kết quả trên trình duyệt web.



### b. Template log

Ở các ví dụ phía trên, thông thường chúng ta thực hiện ghi log ra 1 biến và hiển thị trực tiếp trên màn hình. Chúng ta cũng đã đề cập sơ qua đến việc sử dụng module lineinfile để ghi log thao tác Ansible ra 1 file, tuy nhiên không chỉ thuần túy ghi log ra file, chúng ta cũng có thể sử dụng 1 template log đầu vào để biểu diễn lại log theo mong muốn của mình.

Giả sử chúng ta cần xuất kết quả ra một file log có định dạng như sau



(với 172.16.11.12 là IP server remote đã chạy lệnh và log là kết quả chạy lệnh tương ứng)

Đầu tiên ta soạn file template log, ví dụ

|  |
| --- |
| Result of host {{ ansible\_host }}  Command {{ output.cmd }} complete at {{ output.end }}  {% if(output.rc == 0) %}  Result: OK  Log:  {% for line in output.stdout\_lines %}  {{ line }}  {% endfor %}  {% else %}  Result: NOK  Log:  {% for line in output.stderr\_lines %}  {{ line }}  {% endfor %}  {% endif %} |

Soạn playbook

|  |
| --- |
| ---  **- name**: Misc  **hosts**: TEST  **connection**: ssh  **become**: **yes**  **serial**: 1  **become\_method**: sudo  **become\_flags**: '-i'  **gather\_facts**: **no**  **tasks**:  **- name**: get serial  **shell**: echo "System serial number is `dmidecode -s system-serial-number`"  **register**: output  **- name**: see the result  **debug**: var=output  **- name**: log the result  **connection**: local  **become**: **no**  **template**:  **src**: /u01/data/ansible/template.log.j2  **dest**: /u01/data/ansible/result.log |

Sử dụng ansible-playbook để chạy playbook

|  |
| --- |
| ansible-playbook -i inventory/host\_test playbook/linux\_test\_log\_something.yml |

Kiểm tra file log kết quả.

## 8. Gắn thêm module để sử dụng trong Ansible

Trong các ví dụ ở trên, chúng ta đã sử dụng các module dựng sẵn tương ứng của Ansible để thực hiện thao tác vào các thiết bị cụ thể. Thông thường, có 2 đường dẫn sau để lưu trữ các module trong Ansible:

- Đường dẫn global dùng cho tất cả các user: /usr/share/ansible/plugins/modules

- Đường dẫn local dành cho user lưu module riêng: ~ (là thư mục home của user)/.ansible/plugins/modules (chú ý thư mục ansible có dấu chấm "." ở phía trước, nghĩa là thư mục ẩn, nếu muốn xem thư mục này cần sử dụng lệnh ls -a).

Có thể kiểm tra lại đường dẫn cụ thể bằng cách check các biến môi trường ANSIBLE\_LIBRARY hoặc cấu hình DEFAULT\_MODULE\_PATH của ansible bằng lệnh sau:

|  |
| --- |
| ansible-config dump |grep DEFAULT\_MODULE\_PATH |

Sau khi download 1 module Ansible và giải nén vào thư mục chứa module tương ứng (global hoặc local), sử dụng các lệnh sau để kiểm tra xem module đã hoạt động được hay chưa. Nếu không có lỗi là có thể sử dụng được module tương ứng.

|  |
| --- |
| ansible localhost -m my\_local\_module |
| ansible-doc -t module my\_local\_module |

Người sử dụng nên tận dụng các modules có sẵn của Ansible để xử lý các thao tác nghiệp vụ hàng ngày trên các HĐH Linux, thay vì lệ thuộc vào command shell. Các modules có sẵn này cho phép người sử dụng có thể tác động an toàn hơn vào các server cũng như giảm được số lệnh phải thực hiện do đã được nhà phát triển và cộng đồng đóng gói và tối ưu trong module.

Người dùng hoàn toàn có thể tự phát triển các module để xử lý nghiệp vụ riêng của mình nếu muốn. Khi đó người dùng cần có kiến thức nhất định về Python và các handle các sự kiện trong Ansible.

## 9. Tổ chức các kịch bản thành task và role

Cho đến lúc này, chúng ta vẫn sử dụng Ansible với 2 kiểu file phổ biến là playbook (để tạo kịch bản tác động) và inventory (để lưu thông tin các host cần tác động). Thông thường, như vậy đã là đủ để thao tác cho nhiều tác vụ. Tuy nhiên, có 1 số tác vụ chúng ta có thể cần thao tác với một yêu cầu cao hơn, dựa trên nền những playbook đã viết nhưng lại chỉ cần tùy biến 1 chút liên quan đến biến, tham số… Nếu vì những lý do này để viết lại 1 playbook khác thì có thể sẽ tốn nhiều công sức. Thay vào đó, chúng ta có thể tổ chức lại các playbook thành 1 role.

Một role phải chứa ít nhất 1 trong 7 thư mục sau để Ansible có thể hiểu là 1 role.

- tasks – chứa danh sách các task chính được thực thi trong role này.

- handlers – chứa các handler, có thể được dùng trong role này hoặc các role khác.

- defaults – chứa các biến được dùng default cho role này

- vars – chứa thông tin các biến dùng trong role, biến trong vars sẽ override biến trong default

- files – chứa các file cần dùng để deploy trong role này, cụ thể như file binary, file cài đặt…

- templates – chứa các file template theo jinja format đuôi \*.j2 (có thể là file config, file systemd…).

- meta – định nghĩa 1 số metadata của role này, như là dependencies

Mỗi một thư mục cần phải chứa 1 file main.yml. Trong đó thì tasks thường là folder quan trọng nhất, thường dùng để chứa những playbook

Để tạo nhanh 1 cây thư mục role, có thể sử dụng lệnh sau:

|  |
| --- |
| ansible-galaxy init <role\_name> |

Giả sử chúng ta cần tạo 1 role để thực hiện cài đặt tự động Apache cho server chạy Centos. Tạo 1 role mới như sau

|  |
| --- |
| ansible-galaxy init test\_apache |

Tất cả các task được để trong thư mục task. Có thể đặt hết tất cả các task cần thực hiện vào file main.yml, tuy nhiên để tổ chức các task cho khoa học và tránh làm rối file main.yml, nên phân chia thành nhiều file nhỏ trong thư mục task như sau

File install.yml

|  |
| --- |
| ---  - name: Install httpd Package  yum: name=httpd state=latest |

File configure.yml

|  |
| --- |
| ---  - name: Copy httpd configuration file  copy: src=files/httpd.conf dest=/etc/httpd/conf/httpd.conf  - name: Copy index.html file  copy: src=files/index.html dest=/var/www/html  notify:  - restart apache |

File service.yml

|  |
| --- |
| ---  - name: Start and Enable httpd service  service: name=httpd state=restarted enabled=yes |

File main.yml include các file cấu hình phía trên

|  |
| --- |
| ---  # tasks file for /etc/ansible/roles/apache  - import\_tasks: install.yml  - import\_tasks: configure.yml  - import\_tasks: service.yml |

Chúng ta thấy trong file configure.yml có 1 handler là restart apache, được gọi sau khi copy xong file index.html. Handler được cấu hình ở file main.yml trong thư mục handlers.

|  |
| --- |
| ---  # handlers file for /etc/ansible/roles/apache  - name: restart apache  service: name=httpd state=restarted |

Trong thư mục meta, sửa file main.yml nhập các thông tin cần thiết. Quan trọng nhất trong phần meta là các thông tin liên quan đến phiên bản ansible tối thiểu (min\_ansible\_version), dependencies, target OS đích sử dụng...

|  |
| --- |
| galaxy\_info:  author: LearnItGuide.net  description: Apache Webserver Role  company: LearnITGuide.net  # If the issue tracker for your role is not on github, uncomment the  # next line and provide a value  # issue\_tracker\_url: http://example.com/issue/tracker  # Some suggested licenses:  # - BSD (default)  # - MIT  # - GPLv2  # - GPLv3  # - Apache  # - CC-BY  license: license (GPLv2, CC-BY, etc)  min\_ansible\_version: 1.2  # If this a Container Enabled role, provide the minimum Ansible Container version.  ... |

Tạo 1 file playbook để gọi role tương ứng

|  |
| --- |
| ---  - hosts: node2  roles:  - apache |

Mặc định, Ansible sẽ tìm role trong các thư mục sau

- /etc/ansible/roles

- <thư mục chứa playbook>/roles

# F. Tài liệu tham khảo

- Tài liệu hỗ trợ chính thức của Ansible: <https://docs.ansible.com/>

- Hướng dẫn của tutorialspoint: <https://www.tutorialspoint.com/ansible/index.htm>