by elroy | Published 18 May, 2023

Biết cách sử dụng ansible, không muốn nhàn cũng khó. Nếu bạn đã trải qua quãng thời gian cài đặt máy chủ, triển khai, thiết lập hệ thống thủ công bằng cách cài đặt bằng tay từng ứng dụng một như nào là Docker, Nginx, Database,… hay là bạn muốn triển khai website thì lại phải tốn một đống thời gian copy file nọ, tải xuống file kia thì quả là một công việc **nhàm chán**, **tốn cực kỳ nhiều thời gian**, và hơn hết là với những dự án phức tạp việc bạn triển khai thủ công như vậy thì khi muốn mở rộng hệ thống, hoặc tạo hệ thống tương tự có khi bạn cũng không thể nào mà nhớ nổi chi tiết chính xác các bước. Vì vậy, bạn sử dụng ansible cấu hình thành những file và cho hệ thống tự động triển khai thì bạn nghĩ xem bạn sẽ tiết kiệm được rất nhiều thời gian triển khai cũng như, có thể đem file cấu hình đó đi chạy cho dự án khác vô tư. Có thể bạn đã hiểu được phần nào sử dụng ansible để làm gì rồi chúng ta cùng vào chi tiết.

Nội dung

Ansible là gì?

Ansible là một bộ công cụ mã nguồn mở và bộ công cụ này bao gồm việc cung cấp phần mềm, quản lý cấu hình và triển khai ứng dụng (khái niệm tại Ansible – wikipedia)

Hay nói một cách đơn giản Ansible là một công cụ tự động hóa các công việc triển khai hạ tầng, ứng dụng, website, cấu hình mạng,… trên cả môi trường On-premise và Cloud, giúp chúng ta có thể cấu hình hạ tầng thay vì những thao tác thủ công thì chỉ cần sử dụng ansible cấu hình những file và chạy vài câu lệnh là xong.

Sử dụng Ansible cho những mục đích gì (cụ thể)

- **Quản lý cấu hình hệ thống**: ansible cho phép tự động hóa quá trình cài đặt và cấu hình hệ thống trên nhiều máy chủ (Ví dụ: bạn có 1 máy chủ tạm gọi là A và bạn sẽ kiểm soát những máy chủ khác, bạn chỉ cần cấu hình ở 1 máy chủ đó và không cần phải truy cập các máy chủ khác)
- **Triển khai ứng dụng**: ansible có thể tự động hóa quá trình triển khai ứng dụng trên các máy chủ mục tiêu (Ví dụ: bạn muốn deploy một website lên trên server khác thì cũng chỉ cần cấu hình ở server A là cũng sẽ tự động hóa quá trình deploy)
- **Quản lý dịch vụ**: ansible cho phép bạn quản lý các dịch vụ hệ thống, bao gồm cả việc khởi động, dừng và khắc phục sự cố. (Ví dụ: bạn có hàng loạt các server mà bạn cần update package, dừng các service đang chạy thì bạn không thể nào mà vào hàng chục cái thủ công như vậy được bạn chỉ cần cấu hình và chạy 1 câu lệnh sử dụng ansible sẽ giúp bạn làm việc đó trên các server khác)
- **Tự động hóa công việc hàng ngày**: ansible giúp giảm công sức và thời gian bằng cách tự động hóa các tác vụ hàng ngày (Ví dụ: bạn có thể sử dụng Ansible để sao lưu và phục hồi dữ liệu, quản lý người dùng và quyền truy cập, kiểm tra các hệ thống và báo cáo lỗi,…)
- Còn rất nhiều bạn sẽ tìm hiểu và liệt kê nhé…

Kiến thức cơ bản phải biết để sử dụng Ansible

Đây là phần kiến thức đã đúc kết nên bạn hãy nắm bắt để có thể sử dụng ansible, bạn không cần phải hình dung ra nó phức tạp hay sử dụng nó như thế nào cả đơn giản chỉ là hiểu “à thì ra nó dùng để làm cái này” vậy là được.

Lý thuyết cơ bản

- **hosts** (đối với Ubuntu mặc định khi cài đặt ansible sẽ nằm ở **/etc/ansible/hosts**): đây là file cấu hình những máy chủ mà bạn muốn điều khiển (bạn có nhớ không chúng ta chỉ đứng ở 1 máy chủ và điều khiển những máy chủ khác và danh sách IP, domain đó nằm ở trong đây).
- **Groups**: đây là phần tổ chức các server trong file hosts (hiểu đơn giản là bạn có nhiều máy chủ bạn cấu hình thành các group để ví dụ group webserver cài đặt docker, nginx thì sẽ tường minh và thống nhất)
- **Inventory**: 2 định dạng file **.ini** và **.yaml** bản chất là giống như file hosts để lưu những thông tin máy chủ tuy nhiên trong thực tế không phải lúc nào chúng ta cũng sử dụng một file hosts mặc định mà chúng ta có thể tạo những thư mục cho

từng dự án riêng, hoặc cấu hình ứng dụng riêng vậy nên chúng ta cũng sẽ tiến hành tạo file để lưu các host riêng.

- **Playbook**: để sử dụng ansible thì đây là một file có định dạng **.yml** hoặc **.yaml** mà trong đó khai báo các thành phần để chạy công việc tự động (ví dụ có 1 file docker-setup.yml thì file này được cấu hình để cài đặt docker và từ với các cấu hình trong đó có thể cài đặt trên tất cả các máy chủ chỉ định hoặc chạy tác vụ như deploy code, bật tắt service,…)
- **Task**: đây là một bước thực hiện cụ thể trong một playbook. Nó đại diện cho một công việc nhỏ, chẳng hạn như cài đặt một gói phần mềm, khởi động một dịch vụ hoặc tạo một tệp tin. Các task trong playbook được thực hiện tuần tự.
- **Module**: đây là các thành phần thực thi được sử dụng trong các **task** của **playbook**. Ansible cung cấp nhiều module tích hợp sẵn cho các tác vụ thông thường, bao gồm quản lý gói phần mềm, cấu hình hệ thống, triển khai ứng dụng, quản lý mạng và nhiều hơn nữa (bạn có thể hiểu đơn giản là thay vì cài đặt loằng ngoằng một đống câu lệnh thì chỉ cần sử dụng module của ansible sẽ giúp bạn tự động việc đó, hay đơn giản hơn nữa để bạn hình dung bạn cần giải phép tính a x b thì thay vì tính tay thì trong máy tính đã tích hợp phép nhân và bấm máy tính là ra)
- **Var**: Ansible cho phép bạn định nghĩa biến để sử dụng trong playbook (đơn giản là bạn cấu hình các biến để bạn có thể sử dụng lại được các cấu hình, truyền biến, bảo mật dữ liệu,… bạn có một biến server chẳng hạn thì bạn truyền vào đó server nào cũng được là do bạn cấu hình ở file chứa các biến hoặc trong playbook)
- **Roles**: Roles là một cách tổ chức và tái sử dụng code trong Ansible. Chúng cho phép bạn nhóm các task, biến và file cấu hình vào một thư mục đặc biệt và tái sử dụng chúng trong nhiều playbook khác nhau. Roles giúp tăng tính module và tái sử dụng code (ví dụ bạn có những role cài đặt như docker, nginx, postgresql,… thì khi sử dụng ansible playbook bạn sẽ gọi đến các role đó vậy là tương ứng bạn có thể tái sử dụng lại mà không cần viết lại cấu hình cài đặt nhiều lần nữa)
- **Handlers** được sử dụng để thực hiện các tác vụ hoặc hành động chỉ khi có sự thay đổi trong cấu hình của máy chủ mục tiêu. Handlers thường được sử dụng để khởi động lại dịch vụ hoặc thực hiện các tác vụ tương tự sau khi cấu hình đã thay đổi (ví dụ bạn cài đặt các công cụ và yêu cầu nó khởi động lại sau khi cài đặt xong chẳng hạn)

Túm gọn lại này, bạn có một file **inventory** (hosts) để lưu trữ những thông tin máy chủ đã được chia thành các **groups** mà bạn muốn quản lý, tác động lên và bạn có các file **playbook** để cấu hình các **task** trong đó trong các task bạn sử dụng các **module** có sẵn của Asible để cấu hình nhanh các ứng dụng và tổ chức các **roles** để tái sử dụng lại các công việc tránh lặp đi lặp lại và sử dụng **handlers** để tạo các công việc khi thay đổi cấu hình, vậy là chúng ta đã đi qua phần lý thuyết cơ bản nhất để sử dụng Ansible.

Mô hình hoạt động

Bạn sẽ sử dụng một server tạm gọi là devops admin’s server từ server này sẽ tiến hành ssh để truy cập đến những server được chỉ định và chạy những cấu hình đã được thiết lập.

Kiến thức thêm xíu

- **Dynamic inventory**: một cách tự động tạo ra danh sách các tài nguyên (hosts, servers, mạng, thiết bị,…) mà hệ thống quản lý cấu hình và triển khai sẽ tương tác và quản lý.
    - Thay vì cung cấp một danh sách tĩnh và cố định các tài nguyên, dynamic inventory cho phép bạn tạo danh sách này dựa trên các nguồn dữ liệu động như hạ tầng đám mây, môi trường ảo hóa, các máy chủ vật lý, hoặc cơ sở dữ liệu. Điều này trong sử dụng ansible giúp tự động hóa quá trình quản lý và triển khai hệ thống, giảm bớt công việc thủ công và tiết kiệm thời gian.

- **Template**: là một khái niệm và công cụ được sử dụng để tạo các file cấu hình động. Một template là một file có thể chứa các biến và biểu thức sử dụng Ansible Jinja2 để tạo ra các giá trị động trong quá trình triển khai (ví dụ như những file cấu hình bạn có thể tạo dưới dạng .j2 và sẽ copy các nội dung file đó qua bên server chỉ định để thiết lập cấu hình cho một phần nào đó được xác định)
- **Ansible Galaxy**: một kho lưu trữ trực tuyến của các roles và các bản mẫu (playbooks) được sử dụng trong Ansible. Nó cung cấp một cách tiện lợi để tìm kiếm, chia sẻ và tái sử dụng các phần mềm cấu hình và các tài nguyên được phát triển bởi cộng đồng Ansible (bạn có thể hiểu đơn giản là ví dụ bạn muốn cài đặt docker thì bạn sẽ tải role docker trên kho lưu trữ này và sẵn dùng tuy nhiên thì mình khuyên bạn nên tự thiết lập)
- **Ansible vault**: một công cụ trong Ansible được sử dụng để mã hóa và bảo vệ các dữ liệu nhạy cảm, chẳng hạn như mật khẩu, khóa SSH, thông tin xác thực, hoặc các thông tin quan trọng khác trong quá trình triển khai (ví dụ bạn muốn mã hóa 1 file thì bạn sẽ sử dụng nó tăng tính bảo mật)

- **Ansible Tower**: hiện đã được đổi tên thành Red Hat Ansible Automation Platform – một giao diện người dùng đồ họa và nền tảng quản lý Ansible. Tìm hiểu cách triển khai, quản lý và giám sát các quy trình tự động hóa Ansible với Ansible Tower.

Đây cũng là phần kiến thức cần thiết để bạn triển khai hiệu quả hơn khi sử dụng Ansible nhưng trong bài viết này mình sẽ không quá tập trung vào nó mà cung cấp để bạn có thể tìm hiểu hoặc trong những bài viết sau mình sẽ làm cụ thể hơn.

Chi tiết bạn có thể xem tại trang chủ: Ansible

Các sử dụng Ansible (Step by step)

Để hiểu thực tế cách sử dụng Ansible thì chúng ta sẽ vào một đầu bài cụ thể cơ bản sau: triển khai ứng dụng web api chạy bằng docker và cấu hình load balancing bằng nginx. Mình sẽ dùng những thành phần kể trên để bạn nắm được ngay những kiến thức đó, chúng ta cùng bắt đầu.

Các bước thực hiện

Step 0: Chuẩn bị tài nguyên

Step 1: Cài đặt ansible

Step 2: Sử dụng ansible cài đặt, cấu hình docker và run container

Step 3: Sử dụng ansible cài đặt nginx, cấu hình load balancing

Step 0: Chuẩn bị tài nguyên

3 server tương ứng như sau:

- Server thứ nhất dùng để quản lý các cấu hình sử dụng Ansible tự động cho server thứ hai và server thứ ba tức là bạn chỉ cần thao tác trên server này thôi có IP là **10.32.3.130**
- Server thứ hai dùng để chạy các container web api có IP là **10.32.3.131**
- Server thứ ba dùng để chạy nginx cấu hình load balancing có IP là **10.32.3.132**

Nếu bạn thực hành trên thiết bị của bạn thì bạn có thể xem bài viết Cách dùng Vagrant tạo máy ảo để tạo nhanh 3 server chỉ với 1 câu lệnh. Vậy với những máy chủ trên là những tài nguyên cơ bản ban đầu.

Images tham khảo đều được chạy trên server thứ hai IP là **10.32.3.131**:

```
$ docker pull elroydevops/accountingservice-1

$ docker pull elroydevops/accountingservice-2

$ docker pull elroydevops/accountingservice-3
```

Đây chỉ là 3 image mình build ra từ dự án sample NetCore tương ứng với data trả ra của từng image sẽ khác nhau nên không có gì phức tạp cả bạn có thể sử dụng luôn hoặc image của cá nhân bạn, dự án bạn để làm

***Lưu ý:** dự án nào thì khi build thành image thì cũng vậy mình chỉ cần làm việc với image đó cho nên là với vai trò DevOps Engineer thì kể cả bạn không biết code Pro cũng được mà bạn chỉ cần biết cách triển khai hiểu đơn giản là biết đóng các dự án thành Image và đem đi sử dụng.*

Step 1: Cài đặt Ansible và sử dụng ansible

Như mình đã nói các server mình sử dụng là Ubuntu, nên mình cũng khuyên bạn nên sử dụng Ubuntu để làm.

Cài đặt và cấu hình sử dụng ansible

`$ ssh root@10.32.3.130` ssh đến server quản lý

`$ mkdir -p /home/setup; cd /home/setup; vim install_ansible.sh` tạo thư mục setup, tiến hành truy cập đến thư mục đó và tạo file script cài đặt trong đó nội dung file như sau:

```
#!/bin/bash

# Install required packages
sudo apt update
sudo apt install -y software-properties-common
sudo apt-add-repository --yes --update ppa:ansible/ansible
sudo apt install -y ansible

# Install Ansible via pip
sudo apt install -y python3-pip
sudo pip3 install ansible
```

`$ chmod +x install_ansible.sh` thêm quyền thực thi cho file

`$ ./install_ansible.sh` chạy file cài đặt Ansible

Vậy chúng ta đã cài đặt thành công ansible siêu đơn giản và tiếp theo là sử dụng ansible

`$ mkdir -p /home/project; cd /home/project` tạo thư mục project để cấu hình dự án và tiến hành truy cập đến thư mục đó

Cấu hình inventory

`$ vim inventory.ini` tạo file và nội dung như sau:

```
[all_hosts] # Định nghĩa nhóm (group) máy chủ có tên là all_hosts
target1 ansible_ssh_host=10.32.3.131 # xác định máy chủ "target1" với địa chỉ IP là "10.32.3.131"
target2 ansible_ssh_host=10.32.3.132 # xác định máy chủ "target2" với địa chỉ IP là "10.32.3.132"

[webservers] # Định nghĩa nhóm máy chủ được chọn làm web servers
target1

[loadbalancingservers] # Định nghĩa nhóm máy chủ được chọn làm load balancing servers
target2
```

Tiếp theo để có thể từ máy chủ thứ nhất thực hiện các tác vụ lên máy chủ thứ hai và ba chúng ta sẽ sử dụng ssh

`$ ssh-keygen -t rsa` tạo keygen

`$ ssh-copy-id root@10.32.3.131` copy keygen của tài khoản server thứ nhất sang server thứ hai và nhập password server thứ hai

`$ ssh-copy-id root@10.32.3.132` copy keygen của tài khoản server thứ nhất sang server thứ ba và nhập password server thứ ba

Vậy là bây giờ bạn có thể thực hiện các tác vụ từ server thứ nhất rồi.

Roles trong sử dụng ansible

Chúng ta sẽ sử dụng ansible roles để tổ chức các cấu hình cài đặt, mình sẽ nói qua luôn phần roles một cách đơn giản luôn, bạn có thể hiểu đơn giản là mỗi một ứng dụng, tool,… đều sẽ tạo thành 1 role vậy tương ứng và cấu trúc trong roles sẽ có thể là:

- Thư mục "**tasks**": Thư mục này chứa các tệp tin YAML mà định nghĩa danh sách các tác vụ mà Ansible sẽ thực hiện trên các máy chủ mục tiêu. Các tác vụ này được liệt kê theo thứ tự và sẽ được thực thi một cách tuần tự.
- Thư mục "**handlers**": Thư mục này chứa các tệp tin YAML định nghĩa các handlers, các tác vụ được kích hoạt khi có sự thay đổi hoặc thông báo từ các tác vụ khác. Handlers thường được sử dụng để khởi động lại dịch vụ hoặc thực hiện các hành động khác khi cấu hình thay đổi.
- Thư mục "**library**": Thư mục này chứa các module mở rộng hoặc script tùy chỉnh có thể được sử dụng trong các tác vụ của role. Các module mở rộng có thể cung cấp chức năng bổ sung mà Ansible không hỗ trợ sẵn.
- Thư mục "**files**": Thư mục này chứa các tệp tin tĩnh (không phải tệp tin template) mà cần được sao chép từ máy điều khiển Ansible đến máy chủ mục tiêu. Các tệp tin này có thể là các script, file cấu hình, hoặc tài nguyên khác cần thiết cho role.
- Thư mục "**templates**": Thư mục này chứa các tệp tin template Jinja2. Các tệp tin template cho phép tạo ra các tệp tin động bằng cách kết hợp dữ liệu và biến từ Ansible với cú pháp Jinja2.
- Thư mục "**vars**": Thư mục này chứa các tệp tin YAML chứa các biến cục bộ cho role. Các biến này có thể được sử dụng trong các tác vụ và templates trong role.
- Thư mục "**defaults**": Thư mục này chứa các tệp tin YAML định nghĩa các giá trị mặc định cho role. Các giá trị mặc định có thể được ghi đè bởi biến trong inventory hoặc khi triển khai role.
- Tệp tin "**meta**": Tệp tin này chứa thông tin mô tả về role và các phụ thuộc của nó. Nó có thể chứa các định nghĩa về các role cần được chạy trước hoặc sau role hiện tại, các phụ thuộc gói và phiên bản Ansible yêu cầu, tác giả, mô tả và các thông tin khác.

Chúng ta sẽ bắt đầu sử dụng ansible roles để cấu hình cài đặt, vì vậy sau chúng ta cũng sẽ dễ dàng hơn để sử dụng lại cũng như tường minh hơn.

`$ pwd` kiểm tra xem chúng ta đang ở đâu nếu ở "**/home/project**" là chính xác

`$ mkdir roles` tạo thư mục roles

Step 2: Cài đặt docker sử dụng Ansible

`$ ansible-galaxy init roles/docker` tạo một role là docker để cài đặt docker

`$ vim roles/docker/tasks/main.yml` mở file cài đặt các tasks với nội dung sau:

```
---
# tasks file for docker
- name: Install required packages
  apt:
    name: "{{ packages }}"
    state: present
  vars:
    packages:
      - apt-transport-https
      - ca-certificates
      - curl
      - gnupg-agent
      - software-properties-common

- name: Add Docker GPG key
  apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
    state: present

- name: Add Docker repository
  apt_repository:
    repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable
    state: present

- name: Install Docker
  apt:
    name: docker-ce
    state: present

- name: Add user to docker group
  user:
    name: "root"
    groups: docker
    append: yes

- name: Download Docker Compose
  get_url:
    url: https://github.com/docker/compose/releases/download/1.29.2/docker-compose-Linux-x86_64
    dest: /usr/local/bin/docker-compose
    mode: 'a+x'

- name: Access Permission
  command: chmod +x /usr/local/bin/docker-compose
```

`$ vim roles/docker/handlers/main.yml` mở file cài đặt handlers với nội dung sau:

```
---
# handlers file for docker
- name: Restart Docker
  service:
    name: docker
    state: restarted
```

`$ vim webservers-config.yml` tạo file playbook cài đặt cho webserver với nội dung:

```
---
- name: Web servers setup
  hosts: webservers
  become: true
  roles:
    - docker
```

`$ ansible-playbook -i inventory.ini webservers-config.yml` tiến hành sử dụng ansible playbook để cài đặt docker trên server thứ hai

Tiến hành ssh sang server thứ 2 với IP **10.32.3.131** kiểm tra kết quả thành công:

Cài đặt chạy các container docker sử dụng Ansible

`$ ansible-galaxy init roles/run-project` tương tự tạo cấu trúc role run-project trống

`$ vim roles/run-project/tasks/main.yml` mở file cài đặt các tasks với nội dung sau:

```
- name: Pull Docker images
  docker_image:
    name: "{{ item }}"
    source: pull
  loop:
    - elroydevops/accountingservice-1
    - elroydevops/accountingservice-2
    - elroydevops/accountingservice-3

- name: Run containers
  docker_container:
    name: "{{ item.name }}"
    image: "{{ item.image }}"
    state: started
    ports:
      - "{{ item.ports }}"
  loop:
    - { name: "container-1", image: "elroydevops/accountingservice-1", ports: "8081:80" }
    - { name: "container-2", image: "elroydevops/accountingservice-2", ports: "8082:80" }
    - { name: "container-3", image: "elroydevops/accountingservice-3", ports: "8083:80" }
```

Được kết quả bạn có thể thấy không hề bị chạy lại các cài đặt trước ví dụ như docker đã thiết lập

Tiến hành kiểm tra xem 3 container đã chạy chưa bằng lệnh sau và thấy 3 container đã chạy:

Tiếp theo chúng ta sẽ tiến hành sử dụng ansible cấu hình nginx làm load balancing tại server thứ ba có IP là 10.32.3.132

Step 3: Cài đặt nginx sử dụng Ansible

`$ ansible-galaxy init roles/nginx` tương tự tạo cấu trúc role nginx trống

`$ vim roles/nginx/tasks/main.yml` mở file cài đặt các tasks với nội dung sau:

```
- name: Install nginx
  yum:
    name: nginx
    state: present

- name: Allow HTTP traffic
  ufw:
    rule: allow
    port: 80
    proto: tcp

- name: Allow HTTPS traffic
  ufw:
    rule: allow
    port: 443
    proto: tcp

- name: Reload UFW
  command: ufw reload
```

`$ vim loadbalancingservers-config.yml` tạo playbook sử dụng ansible chạy load balancing với nội dung sau:

```
---
- name: Deploy Nginx Load Balancer
  hosts: loadbalancingservers
  become: true
  roles:
    - nginx
```

`$ ansible-playbook -i inventory.ini loadbalancingservers-config.yml` tiến hành sử dụng ansible playbook để cài đặt nginx trên server thứ ba và được kết quả

Kiểm tra trên browers thấy thành công

cấu hình Load balancing bằng nginx sử dụng ansible

Tiếp theo chúng ta tiến hành sử dụng ansible cấu hình load balancing tương tự như vậy chúng ta cũng cần một role mới

`$ ansible-galaxy init roles/loadbalancing` tương tự tạo cấu trúc sử dụng ansible role loadbalancing trống

`$ vim roles/loadbalancing/default/main.yml` mở file default với nội dung sau:

```
---
nginx_conf_path: /etc/nginx/conf.d/loadbalancing.conf

# List of backend servers
backend_servers:
  - name: server1
    address: 10.32.3.131
    port: 8081
  - name: server2
    address: 10.32.3.131
    port: 8082
  - name: server3
    address: 10.32.3.131
    port: 8083

load_balancing_method: round-robin

load_balancer_port: 80
```

`$ vim roles/loadbalancing/task/main.yml` mở file trong task với nội dung sau:

```
---
# tasks file for roles/loadbalancing
- name: Configure load balancing
  template:
    src: loadbalancing.conf.j2
    dest: "{{ nginx_conf_path }}"
    owner: root
    group: root
    mode: '0644'
  notify: Reload nginx
```

`$ vim roles/loadbalancing/templates/loadbalancing.conf.j2` tạo file template với nội dung sau:

```
upstream backend {
    {{ load_balancing_method }};
    {% for server in backend_servers %}
    server {{ server.address }}:{{ server.port }};
    {% endfor %}
}

server {
    listen {{ load_balancer_port }};
    location / {
        proxy_pass http://backend;
    }
}
```

`$ vim roles/loadbalancing/handlers/main.yml` mở file cài đặt các handlers với nội dung sau:

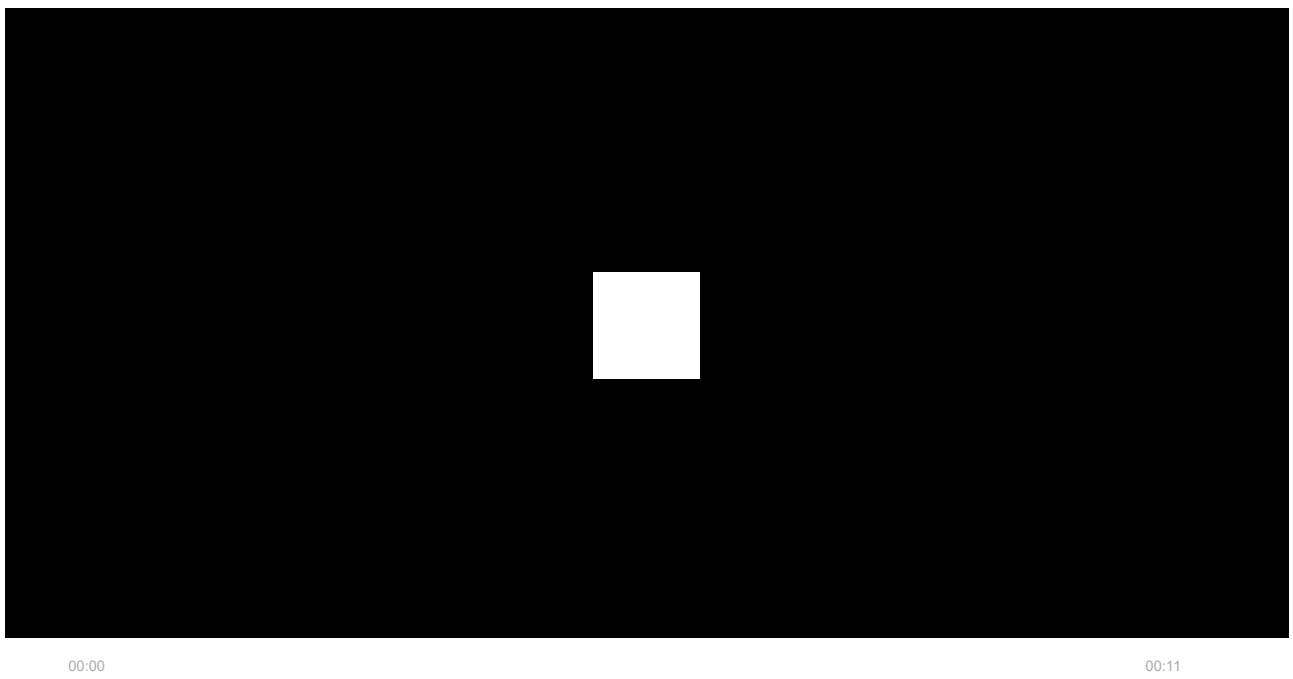
```
- name: Reload nginx
  service:
    name: nginx
    state: reloaded
```

`$ vim loadbalancingservers-config.yml` sửa lại sử dụng ansible playbook chạy load balancing với nội dung sau:

```
---
- name: Deploy Nginx Load Balancer
  hosts: loadbalancingservers
  become: true
  roles:
    - nginx
    - loadbalancing
```

`$ ansible-playbook -i inventory.ini loadbalancingservers-config.yml` tiến hành chạy playbook để thiết lập loadbalancing trên server thứ ba

Tại đây 3 container của chúng ta đang chạy trên server thứ nhất là 10.32.3.131 tương ứng các port 8081 8082 8083 và url api là/GetAccountingDatas nên sẽ truy cập từ Ip cấu hình load balacing là **10.32.3.132/GetAccountingDatas**



Ansible là một công cụ rất mạnh mẽ để giúp bạn tự động hóa các hạ tầng, cấu hình, triển khai,… vì vậy biết sử dụng ansible những công việc bạn chỉ cần cấu hình 1 lần và có thể sử dụng lại rất nhiều lần tiết kiệm được rất nhiều thời gian của bạn.

Vậy là trong bài viết này mình đã hướng dẫn xong cách sử dụng Ansible trong seri devops basic sau mình sẽ làm thêm nhiều case thực tế sử dụng ansible để các bạn được sự mạnh mẽ của nó. Nếu bạn muốn tìm hiểu về DevOps thì có thể theo dõi website của mình, mình luôn hướng tới sự chi tiết, đơn giản, Mình sẽ làm nhiều hướng dẫn hơn về các kỹ năng các tool khác để giúp bạn tối ưu được những tác vụ cũng như tiết kiệm những chi phí khi phát triển phần mềm. Thanks.

Mọi thắc mắc bạn có thể liên hệ:

Email: elroydevops@gmail.com

YOU MAY ALSO LIKE

Published 10 June, 2023

Published 28 October, 2022

Leave a comment

Your email address will not be published. Required fields are marked *

*COMMENT

*NAME

*EMAIL

WEBSITE

SAVE MY NAME, EMAIL, AND WEBSITE IN THIS BROWSER FOR THE NEXT TIME I COMMENT.

POST COMMENT