Ansible-AWX를 이용하여 WP구축 및 Galera 데이터베이스 클러스터 구성

목표

Ansidle에 전체적인 이해와 AWX로 Wordpress 를 구현하기 위해서 AWX설치와 각 역할의 구성들을

어떻게 Playbook으로 구성하고 실행에대한 이해

---

1. Ansible 개요

Ansible 이란?

`ref.` <https://docs.ansible.com/>

`Ansible` 은 에플리케이션을 원격 노드에 배포하고 반복적으로 서버를 프로비저닝하는데 사용되는 오픈소스 도구

마스터-클라이언트 구조로 사용 할 수도있지만 푸시 모델 설정을 사용하여 다중 계층 응용프로그램 및 응용프로그램 아티팩트를 푸쉬하기 위한 공통 프레임워크를 제공함

`Ansible` 의 주요목표는 **단순성과 사용 편의성**이다. 또한 보안과 안정성에 중점을 두고 있으며 부품이동의 최소화, 전달을 위한 OpenSSH이 사용, 프로그램에 익숙하지 않은 사람까지도 이해하기 쉽도록 설된 언어를 포함한다.

`Ansible` 은 에이전트 없이 시스템을 관리한다. 원격 데몬을 업그레이드하는 방법이나 데몬이 제거돼 시스템을 관리 할 수 없는 문제에 대한 질문은 없다. OpenSSH는 가장 많이 검토된 오픈 소스구성 요소중 하나이므로 보안노출이 크게 줄어든다.

Ansible은 분산돼있으며 기존 OS 자격 증명을 사용하여 원격 컴퓨터에대한 엑세스를 제어한다. Kuberos,LDAP, 중앙 집중식 인증관리 시스템과 쉽게 연결가능하다.

`Ansible`은 매년 약 3-4회 Ansible의 새로운 주요버전을 release한다. 핵심 응용 프로그램은 언어 설계 및 설정의 단순성을 평가하면서 보수적으로 발전중이다. 하지만 새로운 모듈, 플로그인 개발에 대한 커뮤니티는 활발하게 움직이며 각 release에 많은 새로운 모듈에 추가한다.

Ansible의 장점

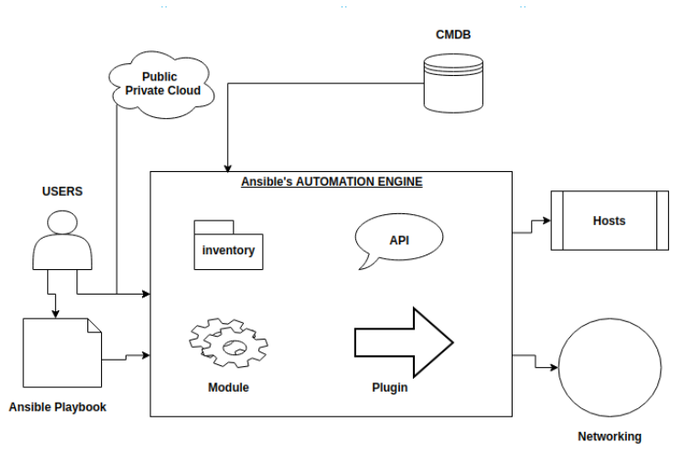
- `SSH` 기반이므로 원격 노드에 에이전트를 설치할 필요가 없다.
- `Yaml` 언어를 사용하기 때문에 쉽게 배울수 있다.
- 플레이북 구조는 간단하고 명확하게 구조화되어 있다.
- 변수 기능을 사용하여 같은 작업에 대해서 다른 구성으로 쉽게 구성 할 수 있다.
- 다른 도구보다 훨씬 간소화 된 코드 기반이다.

Ansible의 단점

- 다른 프로그래밍 언어를 기반으로 하는 도구보다 덜 강력하다.
- DSL(Domain Specific Language)을 통해 로직을 수행한다. DSL은 학습할 때까지 문서를 자주 확인하는것을 의미한다.
- 변수 등록은 기본적인 기능조차도 요구되기 때문에 더 쉬운 작업을 더 복잡하게 만들 수 있다.

- 플레이 내 변수의 값을 확인하기가 어렵다.

Ansible Architecture



`Ansible` 은 클라우드 프로비저닝, configuration 관리, 어플리케이션 배포, 인프라-서비스 오케스트레이션 및 여러 기타 IT 요구사항을 자동화 하는 IT자동화 엔진

`Ansible` 은 처음부터 Multi-tier 배포를 위해 설계되었으므로 한 번에 하나의 시스템을 관리하는 것이 아니라 모든 시스템이 상호 관련되는 방식을 설명함으로써 IT인프라를 모델링한다.

에이전트를 사용하지 않고, 추가적인 커스텀 보안 인프라또한 사용하지 않으므로 배포가 매우 쉽다. 가장 중요한 것은 간단한 언어인 `YAML` 을 사용해 자동화 작업을 설명 할 수 있다.

Controll Node

`Ansible` 이 설치된 모든 시스템

어떤 제어 노드에서든 `/usr/bin/ansible` 또는 `usr/bin/ansible-playbook` 호출함으로써 command 와 playbook 을 실행 할 수 있다. 랩탑, 데스크탑, 서버는 모두 python 만 설치되어있다면 모두 컨트롤 노드가 될 수 있다. 단, Window 시스템을 제어노드로 둘 수는 없고 제어노드를 가지는것은 가능하다.

Managed nodes

`Ansible` 로 관리하는 네트워크장치

관리되는 노드는 호스트라고도 한다. 관리하는 노드에는 `Ansible` 이 설치되어있지 않다.

Inventory

관리되는 노드의 목록

Ansible은 관리되는 모든 기기를 원하는 그룹으로 묶는파일을 가지고 관리하는 기기를 나타낸다. 새 시스템을 추가하기 위해 추가적인 SSL 서명 서버가 필요하지 않으므로 NTP 또는 DNS문제로 인해 특정 시스템이 연결되지 않는 번거로움이 없다.

인벤토리 파일은 때때로 호스트 파일이라고 한다. 인벤토리는 관리되는 노드에 대한 각각의 IP주소와 같은 정보를 지정 할 수 있다. 또한 인벤토리는 관리 노드를 구성하여 쉽게 확장 가능하도록 그룹을 만들고 중첩 시킬 수 있다.

인벤토리의 기본위치는 `/etc/ansible/hosts` 이고 `-i <path` 옵션을 통해 command line에서 다른 인벤토리 파일을 지정 할 수 있다.

일반적으로 인벤토리 파일의 구성

```
[webservers]
www.node1.com
www.node2.com
```

```
[dbservers]
db.node1.com
db.node2.com
```

인벤토리 호스트가 나열되면 변수는 간단한 텍스트 파일로써 이에 할당 될 수 있다. 동적 인벤토리를 사용해 EC2, Rackspace, OpenStack과 같은 데이터소스에서 인벤토리를 가져온다.

Modules

`Ansible` 코드의 실행 단위

`Ansible`은 노드에 연결하고 `Ansible modules`라는 스크립트를 푸시함으로써 움직인다. 대부분의 모듈은 원하는 시스템의 상태를 설명하는 파라미터를 승인한다. 기본적으로 SSH를 통해 `Ansible`은 이러한 모듈을 실행하고, 완료되면 제거한다. 모듈 라이브러니는 모든 시스템에 상주 할 수 있으며, 서버 daemon, DB가 필요하지 않다.

자신만의 모듈을 작성 할 수 있지만 반드시 그럴 필요가 있는가에 대해 우선 고민해야한다. 일반적으로 자주 사용하는 터미널 프로그램, 텍스트 편집기 및 버전 제어 시스템을 사용해 컨텐츠 변경사항을 추척한다. JSON을 반환할 수 있는 모든 언어(Ruby, Python, bash, etc)로 특수 모듈을 작성해야 할 것이다.

각 모듈은 특정 유형의 데이터베이스에서 사용자를 관리하는것부터 특정 유형의 네트워크 장치에서 VLAN 인터페이스 관리까지 특정한 용도로 사용된다. 작업으로 단일 모듈을 호출하거나, 플레이북에서 여러 다른 모듈을 호출 할 수 있다.

Plugins

`Plugin`은 Ansible의 핵심 기능을 강화한다. 모듈이 대상 시스템에서 별도의 프로세스로 실행되는 반면, 플러그인은 `/usr/bin/ansible/` 프로세스 내의 제어노드에서 실행된다. 플러그인은 데이터 변환, output로깅, 인벤토리 연결 등 Ansible의 핵심기능에 대한 옵션과 확장을 제공한다.

Ansible에서 여러 편리한 플러그인이 포함되어 있으며 쉽게 작성 할 수 있다. 인벤토리 플러그인을 작성하여 JSON을 반환하는 모든 데이터소스에 연결 할 수 있다. 플러그인은 Python으로 작성해야 한다.

Tasks

`Ansible` 의 작업단위

`ad-hoc` 명령을 사용하여 단일 작업을 한 번 실행가능

Playbooks

반복해서 실행하고자 해당 작업을 실행 순서대로 저장해 놓은 정렬된 작업리스트

플레이 북은 한 번에 몇개의 기기를 처리해야하는지에 대한 세부적인 제어를 통해 여러 인프라 토폴로지의 다양한 조각들을 정교하게 조정 할 수 있다.

`Ansible`은 잘 조저된 단순한 오케스트레이션 접근 방식을 가지고있다. 자동화 코드가 앞으로 수년 동안 당신에게 완벽히 이치에 맞아야 하며 특별한 구문이나 특징에 대해 기억 할 것이 거의 없을 것이라고 믿기때문

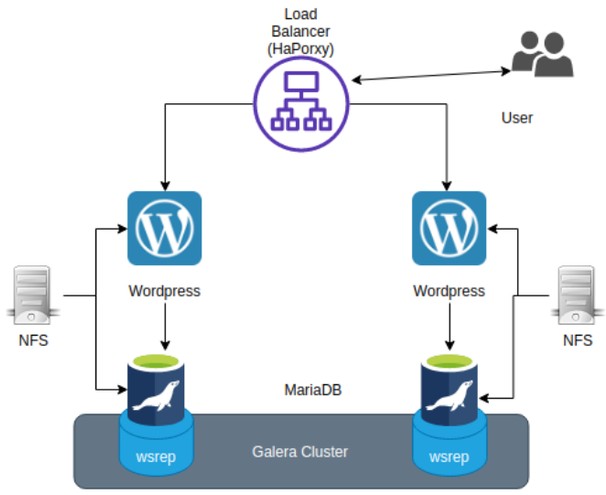
플레이북의 기본구성

```
- hosts: webservers
  become: yes
  remote_users: student
  tasks:
  - yum: httpd
 ...
```

---

2. 실습 시스템 구성

실습 서버 구성도



노드 시스템 버전 및 IP구성

Name	Version	Ip	Gw	DNS
Ubuntu / LoadBalancer	18.04	192.168.122.55/24	192.168.122.1	8.8.8.8, 8.8.4.4
Centos	7.8			
Wordpress	5.4.2	[wp1] 192.168.122.51/24, 192.168.123.51/24 [wp2] 192.168.122.52/24, 192.168.123.52/24	192.168.122.1 192.168.123.1	8.8.8.8, 8.8.4.4
MariaDB	10.4	[db1] 192.168.123.101/24 [db2] 192.168.123.102/24	192.168.123.1	
apache	2.4.6			

전체적인 역할 구성도

이번 구성은 role역할로 한번의 playbook 실행으로 모든 노드에 설치 할 수 있도록 구성하려고한다.

`Ansible` 에서는 롤역할을 한번에 만들어주는 명령어가 있다.

`ansible-galaxy init [하나의 플레이북에 해당하는 폴더] --init-path [플레이북을 하나로 묶을 폴더]`

명령어로 만들 수 있다.

```
roles
|
├── apache ## wp1,wp2에 http와 php를 설치해서 wp구축환경을 구성
|   ├── files
|   |   └── hello.txt
|   ├── README.md
|   ├── tasks
|   |   └── main.yml
|   ├── templates
|   |   └── apache.conf.j2
|   └── vars
|       └── main.yml
|
├── common ## 모든 노드에 필요한 epel-release와 semanage 설치
|   ├── README.md
|   └── tasks
|       └── main.yml
|
|  ## 데이터베이스 galera j2파일 때문에 역할을 나누게됨
|  ## 좋은 변수구성법을 알게되면 db1,2를 합쳐서 구성 해볼 예정
|
├── galera_db1 ## db1서버에 galera maria 구성
|   ├── README.md
|   ├── tasks
|   |   └── main.yml
|   ├── templates
|   |   └── galera.cnf.j2
|   └── vars
|       └── main.yml
|
├── galera_db2 ## db1서버에 galera maria 구성
|   ├── README.md
|   ├── tasks
```

```
|   |   └── main.yml
|   ├── templates
|   |   └── galera_db2.cnf.j2
|   └── vars
|       └── main.yml
|
├── haproxy  ## 외부에서 통신을 받고 wp1,wp2로 부하분산을 하기위해 구성
|   ├── handlers
|   |   └── main.yml
|   ├── README.md
|   ├── tasks
|   |   └── main.yml
|   ├── templates
|   |   ├── centos
|   |   |   └── haproxy.cfg.j2
|   |   └── ubuntu
|   |       └── haproxy.cfg.j2
|   └── vars
|       └── main.yml
|
├── nfs ## 데이터베이스에 Wordpress에 입력되는 데이터를 받을 수 도록 구성
|   ├── handlers
|   |   └── main.yml
|   ├── README.md
|   ├── tasks
|   |   └── main.yml
|   ├── templates
|   |   ├── exports.j2
|   |   └── wp-config.php.j2
|   └── vars
|       └── main.yml
└── site.yml
|
```

```
/roles/site.yml

 ## site.yml 구성

- hosts: wp ## 모든 노드들이 들어감
  become: yes
  roles:
    - common

- hosts: wp-db  ## 데이터베이스 서버들
  become: yes
  roles:
    - nfs

- hosts: db1
  become: yes
  roles:
    - galera_db1

- hosts: db2
  become: yes
  roles:
    - galera_db2

- hosts: wp-web  ## wp1, wp2 노드
  become: yes
  roles:
    - apache

- hosts: wp-lb ## ha 노드
  become: yes
  roles:
    - haproxy
```

이런식으로 `roles` 폴더에 플레이북을 6개를 구성했고

`site.yaml` 에서는 각 인벤토리 그룹에 역할에 맞는 플레이북으로 구성 될 수 있도록 설정해주어야 한다.

`host`에는 각 노드들을 적어주고 `roles` 에는 플레이북폴더를 적어주면된다.

---

3. AWX 설치

먼저 Awx를 설치하기 위해서는 docker가 필요하고 기본적인 시스템사양이 있다.

AWX 요구사항

Aa 시스템 명	Tags
Ansible	2.4
Docker	
Docker-compose / python 모듈	
GNU make	
Git	1.8.4

Aa 시스템 명	Tags
Node	6.x LTS
NPM	3.x LTS

설치법

```
## 먼저 최신 Ansible 버전을 사용하기 위해서 epel 저장소를 추가한다

## epel-release 설치

sudo yum -y install epel-release.noarch

## 저장소 등록

sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

## 도커 명령어 패키지 설치

sudo yum -y install docker-ce docker-ce-cli containerd.io

## 엔서블 및 소프트웨어 설치

sudo yum -y install ansible make git nodejs npm python-pip

## 파이썬 설치 후 파이썬 업데이트

sudo yum -y update python

## 파이썬으로 도커 설치

sudo pip install docker docker-compose
```

설치를 순서대로 해주고 다 완료가 되었으면

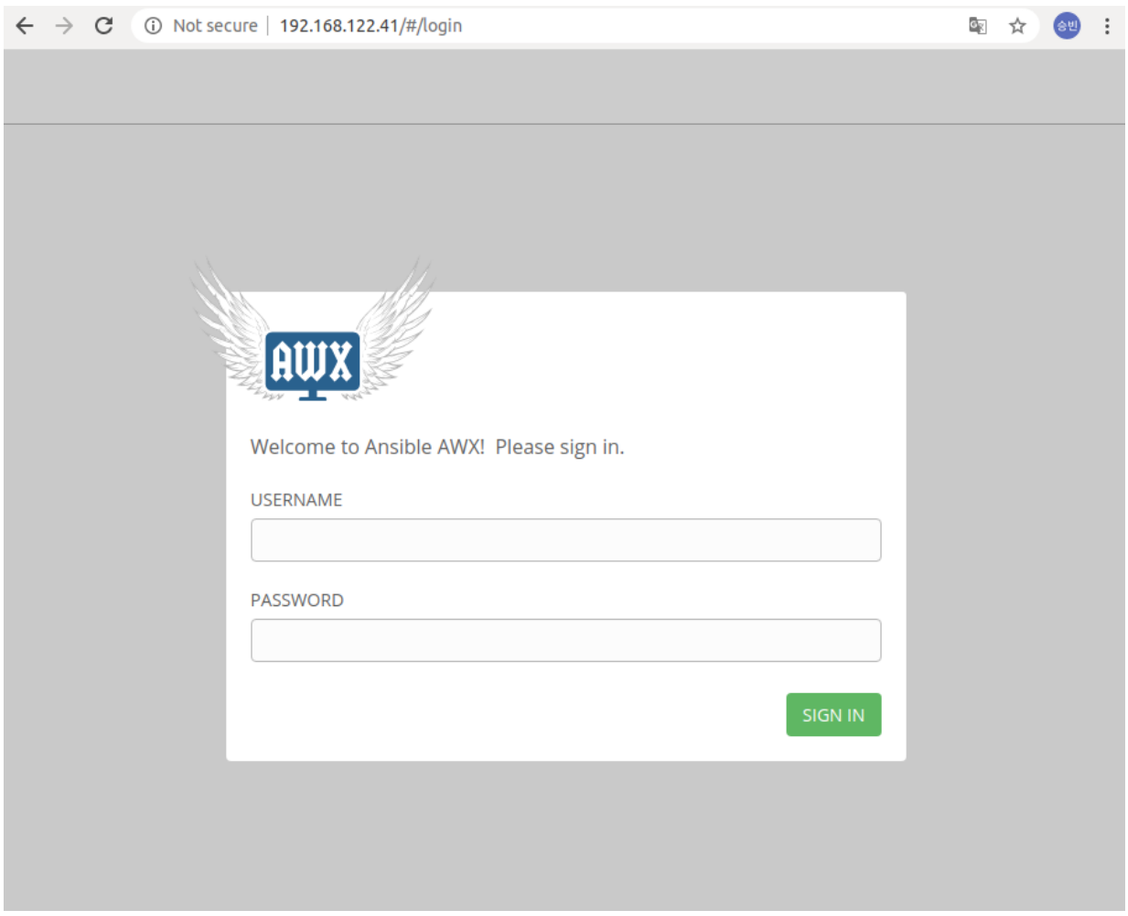
```
sudo systemctl enable docker
sudo systemctl start docker
```

도커 서비스를 재시작 및 `enable`해준다.
그 다음 해야할 일은 `github`에서 awx 설치패키지를 복사해 와야한다.

```
git clone https://github.com/ansible/awx.git
```

복사가 끝나면 `installer` 폴더속에 `inventory` 에 변수를 설정해야하는데 우리가 해야할 일은
140번째 라인에 있는 `project_data_dir=/var/lib/awx/projects` 에 주석을 풀어주는 일이다.
AWX 에서 프로젝트를 구성하기 위해서 필요하다.

변수를 다 설정했으면 `ansible-playbook -i inventory install.yaml` 로 설치를 해준다.



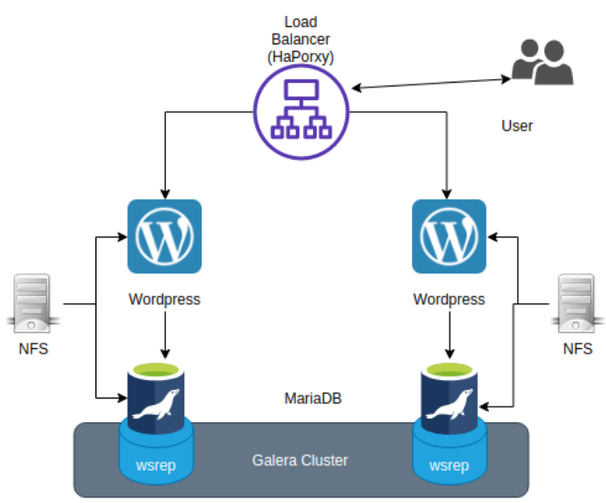
설치가 다되고 컨트롤러 노드에 ip로 접속하면 해당 페이지가 나온다.

`inventory` 변수에서 계정과 비밀번호를 설정하지 않으면 기본적으로

계정은 `admin` 과 비밀번호는 `password` 로 설정되어있다.

---

4. 역할 구성



NFS 및 Galera Cluster 데이터 베이스 구성

데이터 베이스를 구성할때에는 먼저 nfs를 구성하고 그다음 mariadb를 설치하려고 한다.

```
/roles/nfs

## 데이터베이스에 nfs구성하기
```

```
## nfs를 사용 할 수 있도록 패키지 설치
- name: Install nfs-utils
  yum:
    name: nfs-utils
    state: latest

## nfs로 공유할 디렉토리 생성 권한은 0775로 구성
- name: Create a directory for nfs exports
  file:
    path: "{{ nfs['exports']['directory'] }}"
    state: directory
    mode: '0775'

## block과 when으로 사용할수있는 디바이스가 있는경우에만 실행
## 만약에 사용할 수 있는 디바이스가 있다면 LVM으로 5G생성
- block:
  - name: Create a new primary partition for LVM
    parted:
      device: "{{ nfs['block']['device'] }}"
      number: 1
      flags: [ lvm ]
      state: present
      part_start: 5GiB
  - name: Create a filesystem
    filesystem:
      fstype: "{{ nfs['block']['fs_type'] }}"
      dev: "{{ nfs['block']['device'] }}1"
  - name: mount /dev/vdb1 on /wordpress
    mount:
      path: "{{ nfs['exports']['directory'] }}"
      src: "{{ nfs['block']['device'] }}1"
      fstype: "{{ nfs['block']['fs_type'] }}"
      state: mounted
  when: nfs['block']['device'] is defined

## nfs로 공유할 IP대역 설정
- name: Create exports to webserver
  template:
    src: exports.j2
    dest: /etc/exports
  notify:
  - Re-export all directories

## 워드프레스 아카이브파일 다운주소 등록 및 다운
- name: Set wordpress url
  set_fact:
    wp_url: "https://ko.wordpress.org/wordpress-{{ wordpress['source']['version'] }}-{{ wordpress['source']['language'] }}.tar.gz"
    wp_filename: "wordpress-{{ wordpress['source']['version'] }}-{{ wordpress['source']['language'] }}.tar.gz"
- name: Download wordpress sources
  get_url:
    url: "{{ wp_url }}"
    dest: "/tmp/{{ wp_filename }}"

## 다운받은 워드프레스 아카이브파일을 공유디렉토리에 압축풀기
## 공유디렉토리에 워드프레스 디렉토리가 있으면 아파치 서버와 연결만 되면 바로 실행 할 수 있다.
- name: Unarchive wordpress archive
  unarchive:
    src: "/tmp/{{ wp_filename }}"
    dest: "{{ nfs['exports']['directory'] }}"
    remote_src: yes
    owner: root
    group: root

## 워드프레스 설정파일 복사하기
- name: Copy wp-config.php
  template:
    src: wp-config.php.j2
    dest: "{{ nfs['exports']['directory'] }}/wordpress/wp-config.php"

## nfs 설정이 끝났으니 서비스 시작
- name: Start nfs service
  service:
    name: nfs
    enabled: true
    state: started

## nfs 관련 방화벽 허용
- name: Allow port for nfs, rpc-bind, mountd
  firewalld:
    service: "{{ item }}"
    permanent: yes
    state: enabled
    immediate: yes
  with_items: "{{ firewall_nfs_lists }}"
```

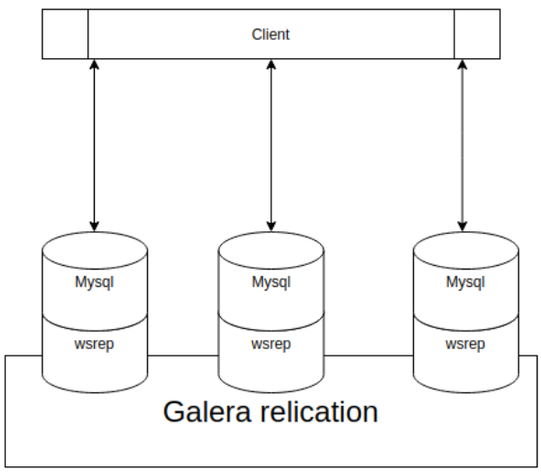
Group vars 폴더안에있는 변수파일

task파일에 변수를 선언하면 제일먼저 같은 역할 폴더안에 vars폴더에 main.yml에서 찾고 선언된 변수가 없으면 groupvars에서 변수를 찾아서 적용시켜준다.

```
/group_vars/nfs.yaml

## nfs에 관련된것들을 미리 변수로 선언해놓은 파일

nfs:
  exports:
    directory: /wordpress
    subnet: 192.168.123.0/24
    options: rw,sync,no_root_squash
  block:
```

```
    #device: /dev/vdb
    fs_type: ext4
```

Galera mysql Cluster 구조



Galera replication 은 mysql(mariadb) 클러스터에 데이터들을 실시간으로 동기방식의 복제를 시켜주는 구조이다.

각각의 노드가 있고 아무 노드에 업데이트가 발생하면 모든 노드에 데이터를 복사를 하고 완료가 되면 업데이트 데이터를 저장한다.

wsrep 이 이러한 galera replication 에 노드들에 업데이트가 있을때마다 나머지 노드들에게 복사를 하고 업데이트 데이터를 저장하는 역할을 하는 모듈이다.

galera cluster는 mysql (mariadb)이 버전이 다르더라도 같은 클러스터로 구성 할 수 있다.

Galera cluster 구성

galera mariadb구성 yaml파일과 j2파일이다.

데이터 베이스 그룹을 wp-db로 구성하였고

galera는 mariadb 패키지와 같이 설치된다.

galera conf 파일 구성은 `bind-address={{ node1_ip }}` 에 실행하는 노드의 ip를 넣어주고 wsrep 모듈의 위치는 `/usr/lib64/galera-4/libgalera_smm.so` 명시를 해준다.

그 후에 galera 클러스터에 이름과 클러스터들의 ip주소들을

`wsrep_cluster_address="gcomm://{{ node1_ip }},{{ node2_ip }}"` 로 클러스터를 구성하려고하는 노드들의 ip 주소를 다 적어주면된다.

```
/roles/galera_db[1.2]

## Galera Cluster 구성 task 파일

## Mariadb 10.4 버전 repo 등록 및 설치
## Galera는 Mariadb 패키지에 포함

- name: Add yum_repository for mariadb
  yum_repository:
    name: MariaDB
    baseurl: http://mirror.yongbok.net/mariadb/yum/10.4/centos7-amd64
    gpgkey: http://mirror.yongbok.net/mariadb/yum/RPM-GPG-KEY-MariaDB
    gpgcheck: 1
    description: MariaDB
- name:
  yum:
    name: MariaDB
    state: present

## Mariadb 서비스 실행
- name: start mariadb service
  service:
    name: mariadb
    state: started
```

```
### Galera.cfg 진저파일 구성


## setup 모듈로 db노드에 팩트값으로 변수구성
- setup:
  delegate_to: db1
- set_fact:
    db1_ip: "{{ ansible_eth1.ipv4.address }}"
    hostname: "{{ ansible_hostname }}"
- setup:
  delegate_to: db2
- set_fact:
    db2_ip: "{{ ansible_eth1.ipv4.address }}"

## 템플릿에 진저파일을 이용해서 galera 구성후 배치
- template:
    src: galera.cnf.j2
    dest: /etc/my.cnf.d/galera.cnf

## galera 사용을 위한 포트 허용을 반복문변수를 이용해서 적용

- name: Allow port for mountd
  firewalld:
    port: "{{ item }}"
    permanent: yes
    state: enabled
    immediate: yes
  with_items: "{{firewall_port}}"

## mysql 모듈 방화벽을 설정하기 위한 패키지 설치
- name:
  yum:
    name: policycoreutils-python
    state: installed
- name:
  seport:
    ports: "{{ item }}"
    proto: tcp
    setype: mysqld_port_t
    state: present
  with_items: "{{ mysql_port }}"
- name:
  seport:
    ports: 4567
    proto: udp
    setype: mysqld_port_t
    state: present

## 데이터베이스 서비스 중단
- name: stop mariadb service
  service:
    name: mariadb
    state: stopped

## 데이터베이스 포트모듈 permissive
- name: mysqld_t Permissive
  selinux_permissive:
    name: mysqld_t
    permissive: true

## Galera 클러스터 생성 및 서비스 시작
- name:
  command: galera_new_cluster
```

```
## /roles/templeate/galera.cfg.j2 파일

## Galera 구성 진저(j2)파일을 통하여 변수값으로 구성파일 작성하기

[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2

## bind-address에 db1에 setup 모듈로 받은 ip 팩트값을 넣어줌
bind-address={{ db1_ip }}

[galera]
wsrep_on=1
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so

## wsrep_cluster_name 은 구성할 클러스터 이름
wsrep_cluster_name="wp_wsrep_cluster"

## 구성중인 클러스터에 노드들에 모든 ip를 넣어줘야함
wsrep_cluster_address="gcomm://{{ db1_ip }},{{ db2_ip }}"

## wsrep_node_name={{ hostname }} 에는 db1에 호스트네임을 팩트값으로 넣어줌
wsrep_node_name={{ hostname }}

## wsrep_node_address={{ db1_ip }} 에도 마찬가지로 ip 팩트값으로 넣어줌
wsrep_node_address={{ db1_ip }}

## wsrep에 동기화 방식
wsrep_sst_method=rsync
```

```
## /roles/vars/main.yaml

firewall_port:
- 3306/tcp
- 4444/tcp
- 4567/tcp
- 4568/tcp
- 4567/udp
mysql_port:
- 4567
- 4568
- 4444
```

데이터 베이스 구성이 끝났다.

아파치 구성

아파치 구성은 먼저 아파치패키지를 설치하고 nfs 공유디렉토리에 있는 워드프레스를 공유받아서 워드프레스를 실행시키는 단계이다.

```
/roles/apache

## nfs 공유 디렉토리를 공유 받을 수 있는 명령어가 있는 패키지 설치
- name: Install nfs-utils
  yum:
    name: nfs-utils
    state: latest

## 데이터베이스 팩트값으로 변수선언
- name: Delegate collecting facts for mariadb
  setup:
  delegate_to: db2
- name: Set facts for mariadb private ip
  set_fact:
    db_private_ip: "{{ ansible_eth1.ipv4.address }}"

## 설정한 데이터베이스 ip변수값으로 nfs 공유디렉토리 설정하고 아파치폴더에 마운트적용
- name: Mount nfs share
  mount:
    path: /var/www/html
    src: "{{ db_private_ip }}:{{ nfs['exports']['directory'] }}"
    fstype: nfs
    state: mounted

## 아파치 설치
- name: Install httpd
  yum:
    name: httpd
    state: latest

## 아파치 설정파일 적용 및 notify 알림을 handler에게 전달
- name: Copy configuration
  template:
    src: apache.conf.j2
    dest: /etc/httpd/conf/httpd.conf

## 아파치 방화벽 포트 허용
- name: Open http port
  firewalld:
    port: "{{ apache['port'] }}/tcp"
    permanent: yes
    state: enabled
    immediate: yes

## seboolean 아파치 리스트 반복문을 이용해서 허용
- name: Active seboolean for httpd
  seboolean:
    name: "{{ item }}"
    state: yes
    persistent: yes
  with_items: "{{ sebool_httpd_lists }}"

## php 7.4 버전 패키지를 위한 remi-release 설치 적용
- name: Install remi-release-7 for php74
  yum:
    name: "{{ php['repo']['pkg'] }}"
    state: latest

## 워드프레스를 위한 기본 php 와 php-mysql 설치
- name: Install php and php-mysql
  yum:
    name: php,php-mysql
    enablerepo: remi-php74
    state: latest

## 아파치 설정파일 적용을 완료했으니 서비스 시작
- name: Start httpd
  service:
    name: httpd
    state: started
    enabled: true
```

```
## /roles/apache/vars/main.yml
```

```
## seboolean 반복문으로 허용하기 위한 변수 리스트
sebool_httpd_lists:
  - httpd_can_network_connect ## 아파치가 네트워크에 연결할 수 있도록
  - httpd_can_network_connect_db ## 아파치가 데이터베이스에 연결할 수 있도록
  - httpd_use_nfs  ## 아파치가 nfs를 사용할 수 있도록
```

```
## /roles/apache/template/apache.cfg.j2

## 아파치에 설정파일

## 팩트값으로 변수를 지정해서 ip를 유동적으로 지정 할 수 있게 함
Listen {{ ansible_eth1.ipv4.address }}:{{ apache['port'] }}

Include conf.modules.d/*.conf

User apache
Group apache

ServerAdmin root@localhost

<Directory />
    AllowOverride none
    Require all denied
</Directory>

DocumentRoot "/var/www/html"

<Directory "/var/www">
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>

<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>

<Files ".ht*">
    Require all denied
</Files>

ErrorLog "logs/error_log"

LogLevel warn

<IfModule log_config_module>
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      # You need to enable mod_logio.c to use %I and %O
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio
    </IfModule>
    CustomLog "logs/access_log" combined
</IfModule>

<IfModule alias_module>
    ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
</IfModule>

<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>

<IfModule mime_module>
    TypesConfig /etc/mime.types
    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz
    AddType text/html .shtml
    AddOutputFilter INCLUDES .shtml
</IfModule>

AddDefaultCharset UTF-8

<IfModule mime_magic_module>
    MIMEMagicFile conf/magic
</IfModule>

EnableSendfile on

IncludeOptional conf.d/*.conf
```

Load Balancer 구성

진저파일(j2)를 이용해서 노드에 새로운 구성(cfg)파일을 구성 할 수 있다.

```
/roles/haproxy/task/main.yml

## Loadbalancer(haproxy) 구성

- name: Deploy to Ubuntu ## 우분투 OS에서만 설치를 할 수 있도록 block과 when으로 구성하였다.
  block:
  - name: Install haproxy
    apt:
      name: haproxy
      state: latest
      update_cache: yes

## haproxy 노드에 set_fact모듈을 통해서 ip를 변수로 받고
   j2파일에 적용시켜 설정파일을 수정하여 적용시키는 단계

  - name: Set facts for haproxy public ip
    set_fact:
      haproxy_public_ip: "{{ ansible_ens3.ipv4.address }}"
  - name: Delegate collecting facts for wordpress1
    setup:
    delegate_to: wp1
  - name: Set facts for wordpress1 private ip
    set_fact:
      wordpress1_private_ip: "{{ ansible_eth1.ipv4.address }}"
  - name: Delegate collecting facts for wordpress2
    setup:
    delegate_to: wp2
  - name: Set facts for wordpress2 private ip
    set_fact:
      wordpress2_private_ip: "{{ ansible_eth1.ipv4.address }}"
  - name: Copy haproxy configuration
    template:
      src: ubuntu/haproxy.cfg.j2
      dest: /etc/haproxy/haproxy.cfg
  when: ansible_distribution == "Ubuntu"

## 설정을 마치고 노드에 haproxy 서비스를 실행함

- name: Start haproxy service
  service:
    name: haproxy
    enabled: true
    state: started
```

lb 노드에 haproxy를 설치하고 `haproxy.cfg.j2` 이름의 진저파일에 선언한 변수들에 맞게 적용이 된다.

```
/roles/haproxy/vars/main.yml

## LoadBalancer 변수파일

haproxy: ## Ubuntu 18.04 에서 Haproxy 구성
  frontend:
    port: 80
  backend:
    name: wp
    balance_type: roundrobin
    web1:
      port: 80
    web2:
      port: 80
```

```
/roles/haproxy/template/haproxy.cfg.j2

## 템플릿 진저파일 구성도

global
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    stats socket /var/lib/haproxy/stats
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
```

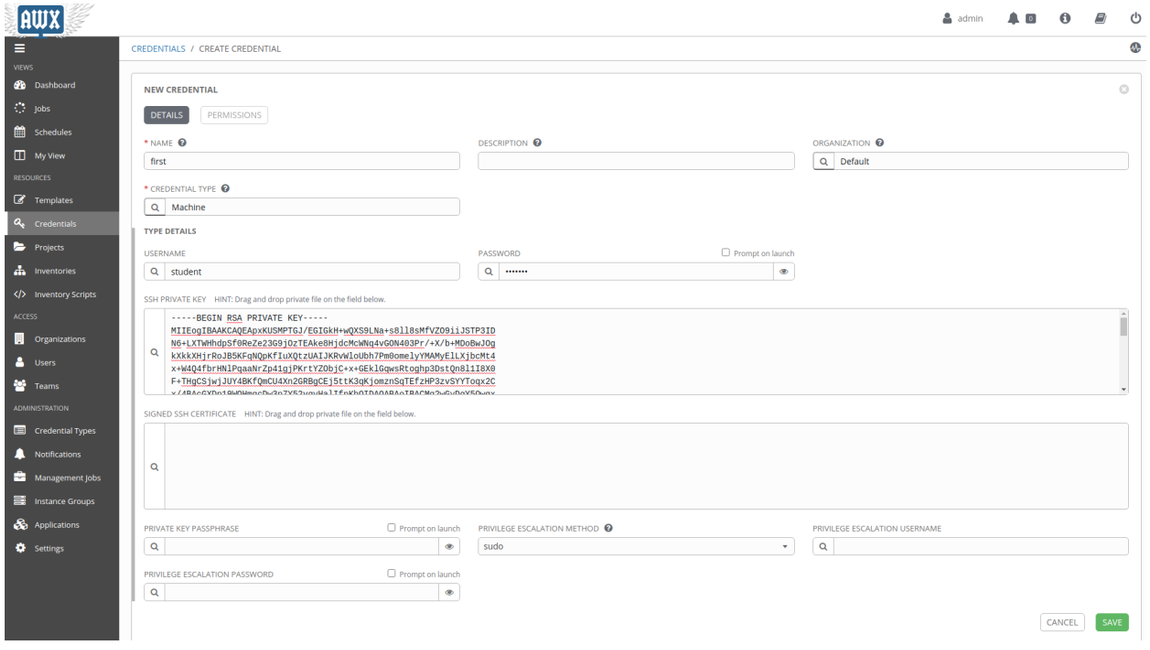
```
## 변수에 선언된 frontend 포트를 받아와서 새로운 구성파일에 적용시켜준다.

frontend  main {{ haproxy_public_ip }}:{{ haproxy['frontend']['port'] }}
    acl url_static       path_beg       -i /static /images /javascript /stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js

    #use_backend static          if url_static
    default_backend             {{ haproxy['backend']['name'] }}

## 변수에 선언된 backend 포트를 받아와서 새로운 구성파일에 적용시켜준다.

backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

backend {{ haproxy['backend']['name'] }}
    balance     {{ haproxy['backend']['balance_type'] }}
    server  web1 {{ wordpress1_ip }}:{{ haproxy['backend']['web1']['port'] }} check
    server  web2 {{ wordpress2_ip }}:{{ haproxy['backend']['web2']['port'] }} check


## Handler 파일 구성

## task가 실행될때 notify가 있으면 Handler가 받아서 제일 마지막에 실행된다.

##- name: Restart haproxy service
##  service:
##    name: haproxy
##   state: restarted
```

---

5. AWX Playbook

credentials 구성

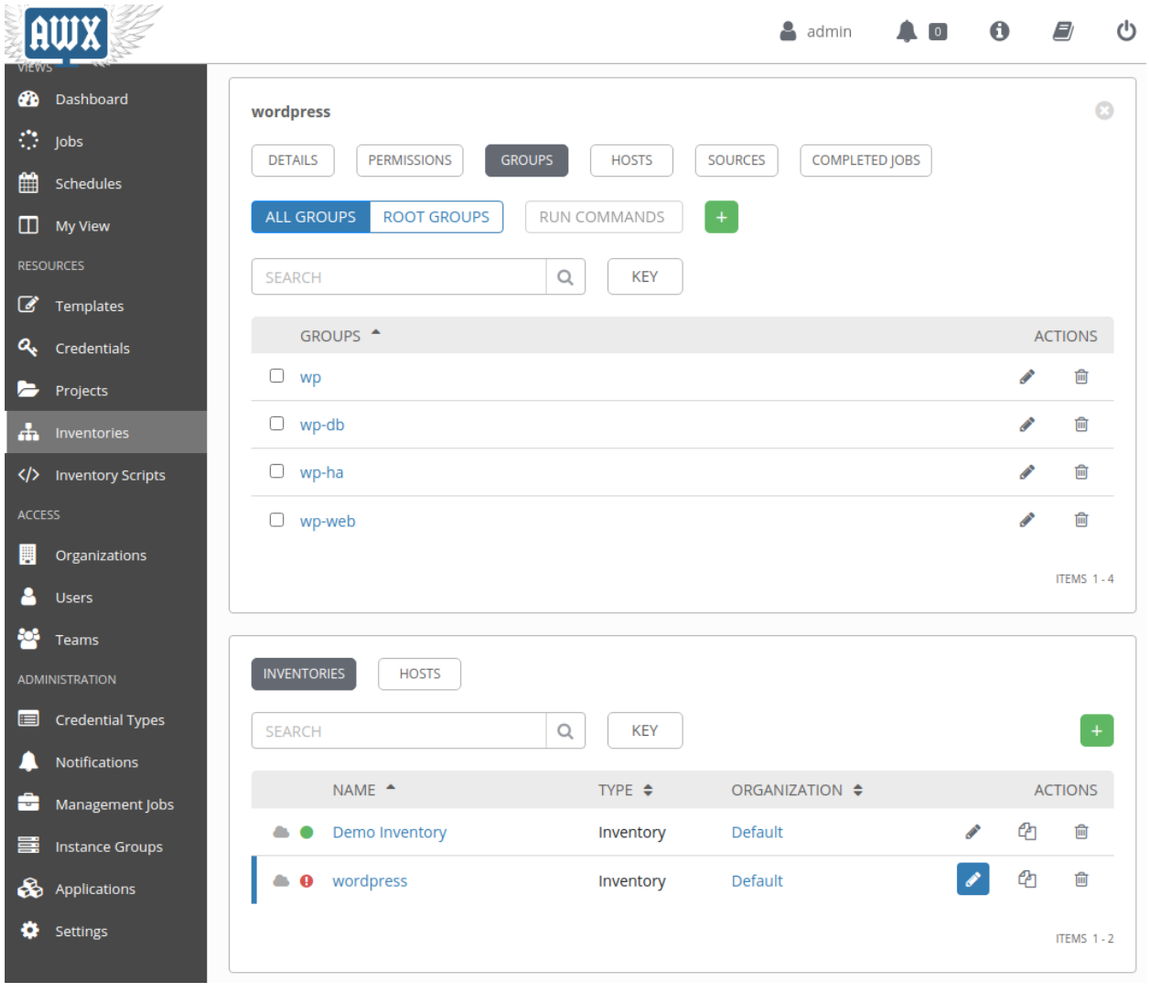
credentials 는 권한을 담당한다.
우리가 노드에서 시스템에 관련되어있거나 패키지를 설치할때 root의 권한이 필요하고 ssh 접속을 할때에도 이러한 권한들이 필요한데 credentials 에 컨트롤러에 개인키를 등록하고 인증권한을 sudo 로 선언하면 AWX에서 sudo권한을 사용할 수 있기때문에 개인키와 `PRIVILEGE ESCALATION METHOD` 에
sudo를 정해주어야한다.

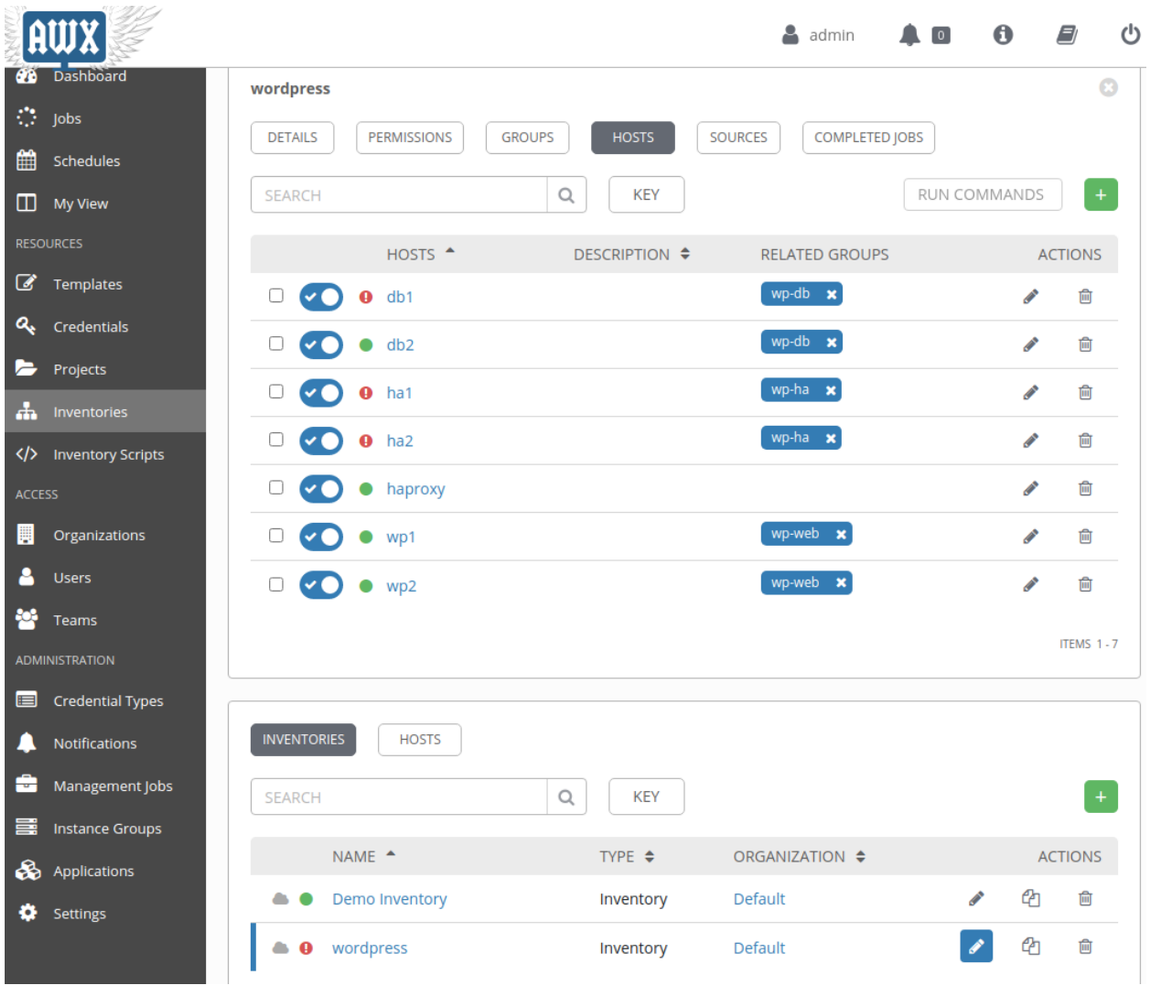


Inventory 구성

인벤토리에 그룹은 혼란을 막기 위해서 `site.yaml` 에 그룹과 같게 만들어준다.
제일 상위 그룹에 wp를 만들고 그안에 각기능의 그룹을 만든다.

그 후에 각 노드에 맞는 이름에 호스트를 만들어서 기능의 그룹에 할당한다.



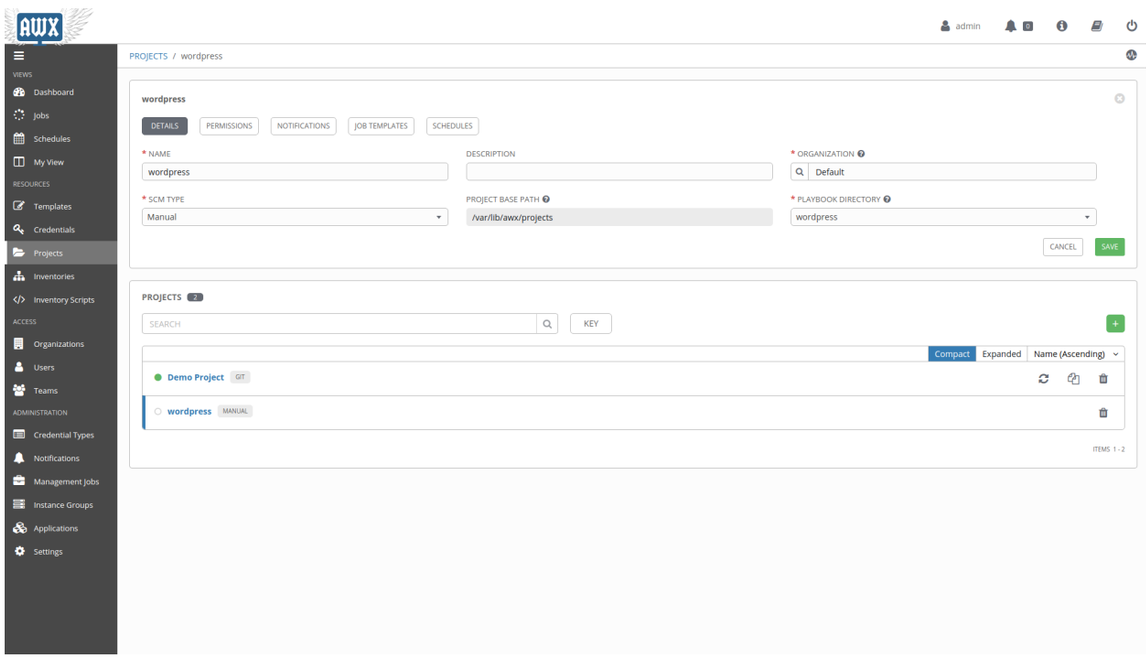


project 생성 및 구성

먼저 프로젝트를 만들기위해서는 컨트롤러 노드에 Cli 환경에서 `/var/lib/awx/project` 에 먼저 프로젝트 디렉토리를 생성해야한다.

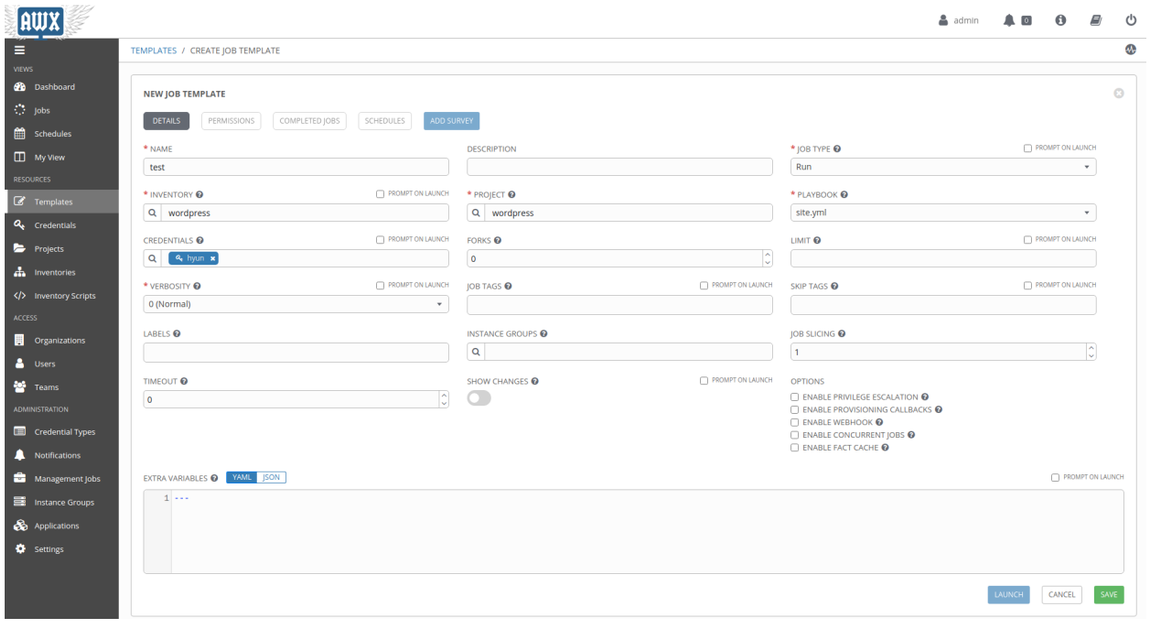
그 후에 project에서 `scm type` 은 메뉴얼로 정해주고 `PLAYBOOK DIRECTORY` 에는 방금 생성한 폴더를 정해주면 완성된다.

```
## 컨트롤러 노드에서 AWX 프로젝트폴더 만드는법

sudo mkdir /var/lib/awx/project/wordpress
```



Template 구성

ansible의 플레이북이라고 할 수 있는 역할이 AWX에서는 Template이다. 생성 할때 지금까지 구성했던 인벤토리, 프로젝트, 인증 을 넣어주고 템플릿을 실행할 수 있는 yaml파일(site.yaml)을 등록해주면 된다.



6. 실행결과

지금까지 구성한 roles역할을 실행

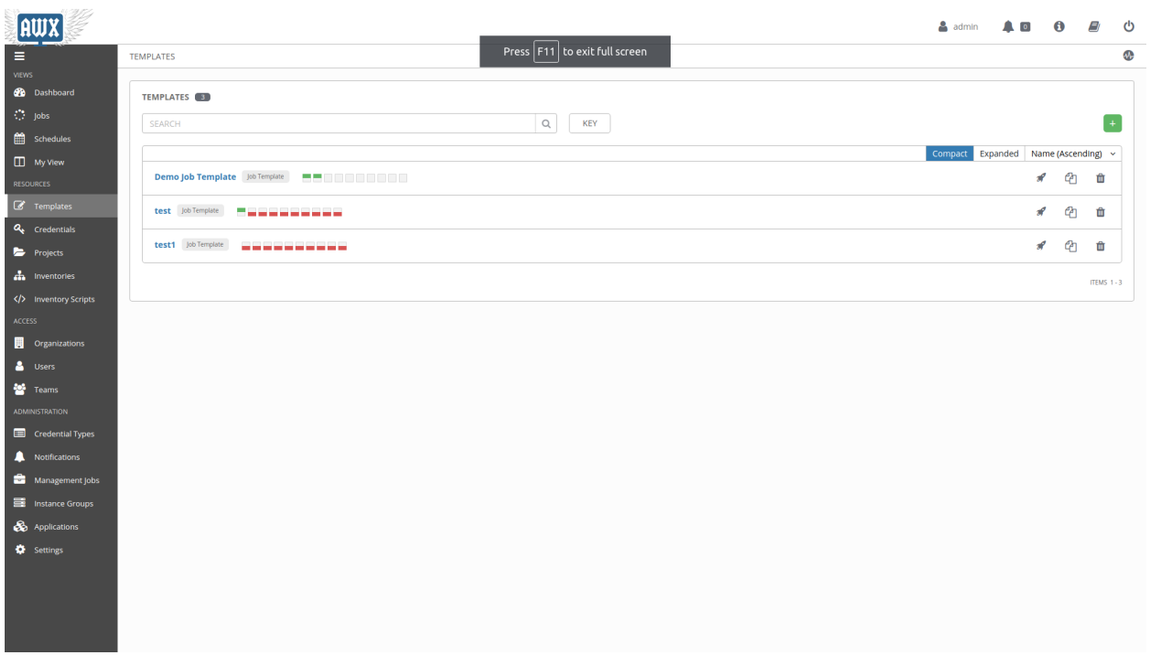
AWX로 구성한 roles역할구성을 site.yml로 실행

Template 실행

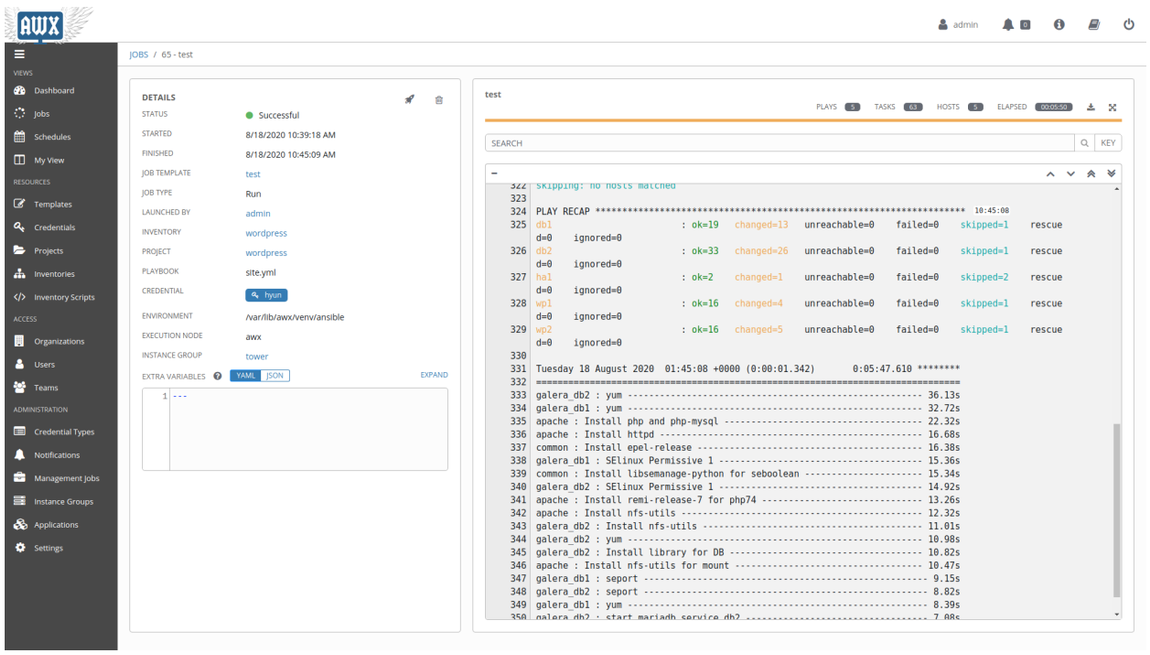
AWX 템플릿에서 로켓모양을 클릭하면 해당 템플릿이 실행된다.

실행창이 뜨고 site.yml에 있는 순서대로 실행된다.

템플릿에 로켓모양을 통해서 실행시켜준다.



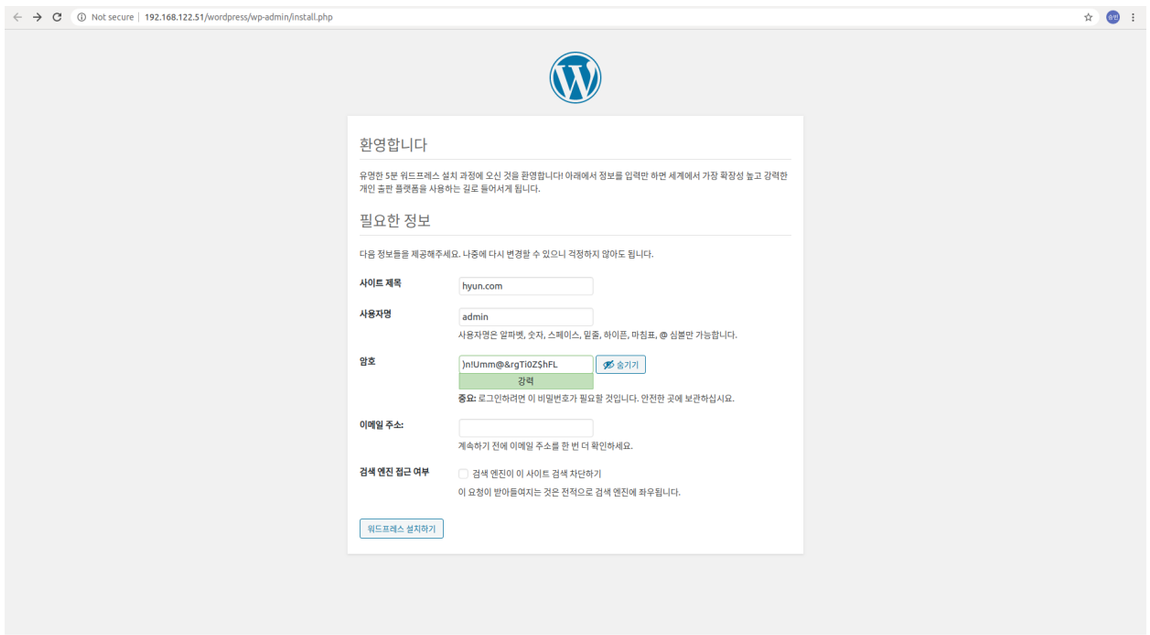
실행되는순간 다른페이지로 이동하게 되고 그안에서 실시간으로 playbook이 실행되는것을 확인 할 수 있다.

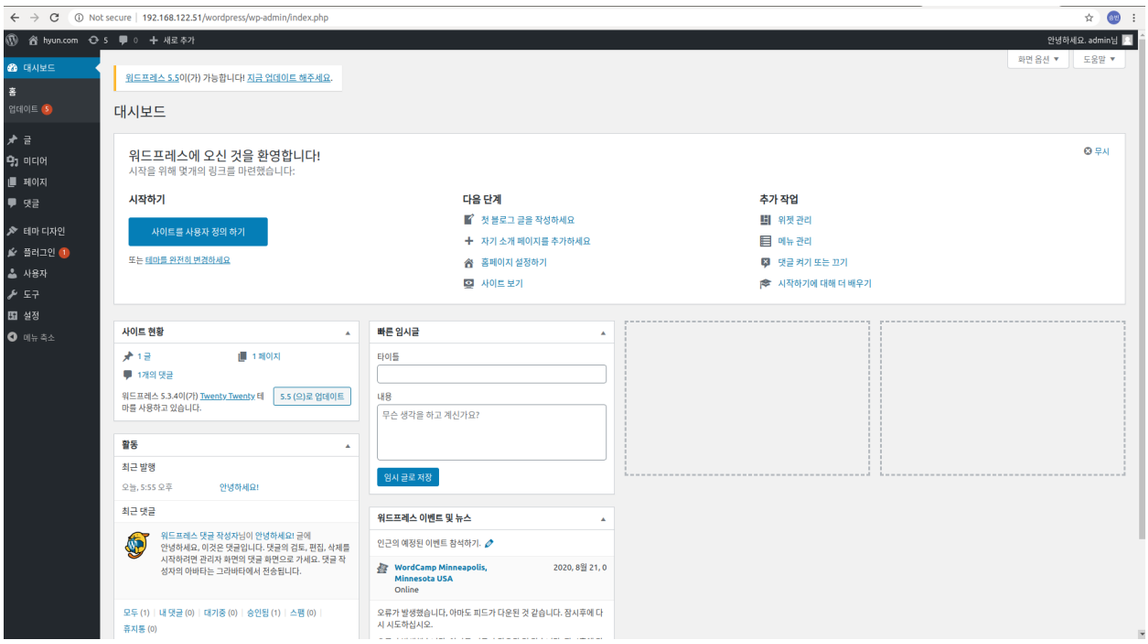


실행이 종료가 되면 `PLAY RECAP`으로 템플릿으로 호스트들에게 어떠한 결과가 생겼는지 알수있게 나온다.

워드프레스 작동확인

실행을 마치고 wp1 IP주소 (LoadBalancer는 작동을 안하는관계로)로 접속하고 워드프레스에 작동을 확인했고 사이트를 만들고 접속해서 마무리를 지었다.





Galera Cluster 작동확인

각 데이터베이스에 원격접속하여 Galera 클러스터가 잘 작동하는지 확인

```
[root@db1 student]# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.4.14-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statemen
t.

MariaDB [(none)]> create database wordpress;
Query OK, 1 row affected (0.016 sec)

MariaDB [(none)]>
```

```
[root@db2 student]# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 10.4.14-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
| wordpress          |
+--------------------+
5 rows in set (0.001 sec)

MariaDB [(none)]>
```

db1에서 새로운 데이터베이스를 생성 직후에 바로 db2에서 확인하면 생성이 되어있는것을 확인 할 수 있다.

결론

이번 Ansible-AWX 실습을 통해서 Ansible에 자동화에 편리함을 느끼고 오류에대한 정확한 피드백으로 오류수정을 하면서 왜 Ansible을 많이 사용하는지 알게되었다. 그리고 다양한 모듈과 그 모듈에 맞는 DOC를 찾아보면서 grep을 조금 더 친숙하게 사용하게 된것에 만족한다.

데이터베이스에 클러스터에 대한 생각을 못했는데 Galera replication 방식에 클러스터를 공부하면서 `rsync`에 작동방식에 대해서 알게되었고 동기화가 되면 데이터보존과 연동이 편리하고 좀 더 안전하게 데이터를 지킬 수 는것이구나 느꼈고 앞으로는 시스템을 다양한 관점으로 분석하여 그 안에 숨어있는 원리와 선배님들에 지혜를 배우면서 더욱 성장해야겠다는 생각을 했다.