

8/1 3일차

내용 리마인드

리눅스 스크립트는 실행형 따라서 역등성 구현하려면 if문으로 사용자가 구성을 해야 함
리눅스 스크립트는 리눅스에 한정적

ansible을 이용하면 위와 같은 과정 x 또 ssh를 사용하여 접근 python 이용하여 리눅스 말고도 가능

변수



위치에 따른 우선순위가 다양하다. 이를 이용하여 재지정 가능하다. 대부분의 경우 scope이 좁은 경우가 우선 순위가 높다

변수: 변수 암호화, 인벤토리 변수, 플레이북 변수

- group_vars, host_vars 디렉토리

디렉토리 아래에 그룹 이름 호스트 이름으로 파일을 만들면 ansible-playbook이 돌 때 찾아다가 변수로 정의

- ansible-vault

파일 암호화하는 데 이용한다.

- 팩트 변수, 매직 변수

조건문에서 많이 이용

제어문

제어문 : 반복문, 조건문, 후속 작업, 실패 처리

- 반복문

loop 를 이용해 사용 list 데이터를 받아서 item이라는 변수에 항목들을 넣어서 작업

변수는 종괄호 두 개로 묶어준다.

- 후속 작업

notify로 handler에게 보고 task가 이미 원하는 대로 구성되어 있으면 보고 x handler는 동작을 하지 않는다. 멍등성

자동화 도구

모놀리식 아키텍처 시대와 비교해서 굉장히 복잡해졌다. 따라서 이런 자동화 도구로 script 파일 짤 능력 키우고 협업이 굉장히 중요하다.

기술만큼 중요한 것도 협업 능력

파일 관리

파일들을 수동으로 관리하는 것이 아닌 자동으로 관리하기 위해서 어떤 툴과 내용이 있는지 확인한다

파일관리 일반 모듈

Linux 파일 관리와 관련하여 일반적으로 사용되는 대부분의 모듈은 `ansible.builtin` 컬렉션에 `ansible-core` 와 함께 제공되며 파일의 권한 및 기타 속성의 생성, 복사, 편집, 수정과 같은 작업을 수행합니다. 다음 표는 자주 사용되는 파일 관리 모듈의 목록을 제공합니다.

모듈	설명
<code>lineinfile</code>	특정행이 파일에 있는지 확인하거나 역참조 정규식을 사용하여 기존 행의 내용을 변경.
<code>blockinfile</code>	마커 선으로 둘러싸인 여러 줄의 텍스트 블록을 삽입 및 업데이트 또는 제거
<code>copy</code>	특정 파일을 관리호스트의 특정 위치로 복사. <code>file</code> 모듈과 유사한 속성을 지님
<code>fetch</code>	<code>copy</code> 와 반대로 동작. 관리호스트의 파일을 제어노드로 가져와 저장.
<code>file</code>	권한, 소유권, SELinux 컨텍스트, 일반 파일의 타임 스탬프, 심볼릭 링크, 하드 링크 및 디렉터리와 같은 속성을 설정. 일반 파일, 심볼릭 링크, 하드 링크 및 디렉터리를 생성하거나 제거.
<code>stat</code>	linux <code>stat</code> 명령어와 유사. 파일의 상태 정보를 조회

관리 호스트에서 새로운 파일을 만들거나 파일 속성을 수정하려면 ***file*** 모듈을 사용합니다. 다음은 `selinux` 컨텍스트를 설정하는 예시입니다.

```
- name: set context
  file:
    path: /path/to/samba_file
    setype: samba_share_t
```

- RBAC : Role Base Access Control
역할 기반으로 제공
- ABAC: Attribute Base Access Control
`selinux`는 속성을 제어해서 제공

그리고 나서 파일을 확인 하면 다음처럼 확인할 수 있습니다.

```
$ ls -Z samba_file
-rw-r--r--.  owner group unconfined_u:object_r: samba_share_t :s0 samba_file
```

파일을 제거할 때도 마찬가지로 file 모듈을 사용합니다. state 속성을 absent 로 지정하면 해당 파일이 존재할 경우 삭제하는 작업을 수행합니다.

```
- name: delete file if it's present
  file:
    dest: /path/to/file
    state: absent
```

그리고 **file** 모듈의 속성을 더해서 복사하거나 편집하는 기능을 제공하는 모듈들이 있습니다. 먼저 **copy** 모듈은 작업 디렉토리의 파일을 관리 호스트로 복사합니다. 이때 selinux 타입이나 권한 설정을 변경할 수 있습니다. 간단히 사용하는 예시는 다음과 같습니다.

```
- name: copy file
  copy:
    src: file
    dest: /path/to/file
```

copy와는 반대로 관리 호스트의 파일을 제어노드로 복사해올때는 **fetch** 모듈을 사용합니다.

```
- name: fetch from host
  fetch:
    src: "/home/{{ user }}/.ssh/id_rsa.pub"
    dest: "files/keys/{{ user }}.pub"
```

stat 모듈은 Linux stat 명령과 유사하게 파일의 팩트를 검색합니다. 매개 변수는 파일 속성을 검색하고 파일의 체크섬을 확인하는 등의 기능을 제공합니다.

```
- name: verify checksum
  stat:
    path: /path/to/file
    checksum_algorithm: md5
    register: result

- debug:
  msg: "Checksum is {{ result.stat.checksum }}"
  #var: result # result 안에 들어 있는 /path/to/file 내용 확인 가능
```

파일의 내용을 확인하고 수정하도록 지원하는 모듈은 **lineinfile**과 **blockinfile**이 있습니다. 단일 행을 수정하거나 혹은 여러 줄에 걸쳐 수정하도록 지원하는 모듈입니다. 각 모듈의 예시는 다음과 같습니다.

```
- name: lineinfile sample
  lineinfile:
    path: /path/to/file
```

```
line: 'Add this line to the file'
state: present
```

```
- name: blockinfile sample
  blockinfile:
    path: /path/to/file
    block: |
      First line of block
      Second line of block
    state: present
```

템플릿 파일

lineinfile과 **blockinfile**은 파일의 내용을 확인하고 수정하는 작업을 수행합니다. 이보다 좀 더 큰 범위로 템플릿을 정의하고 원하는 값을 자동으로 가져와 변수로 지정하여 치환한 뒤 파일의 내용을 구성하도록 지원하는 템플릿 파일이 있습니다.

Jinja2 템플릿 문법은 파일이 배포될 때 표현식으로 구성하거나 변수 및 팩트를 사용하여 자동으로 사용자 지정되는 템플릿 구성 파일을 작성할 수 있습니다. 변수 및 논리 표현식은 태그 사이 또는 구분 기호 사이에 입력합니다. Jinja2 템플릿을 평가할 때는 `{{ EXPR }}` 표현식이 해당 표현식이나 변수의 결과로 바뀝니다. 코드를 반복하거나 테스트를 수행하는 특수 제어 구조에 `{% EXPR %}`을 사용할 수도 있습니다. `{# COMMENT #}`구문은 주석을 묶는 데 사용할 수 있습니다.

```
{# /etc/hosts line #}
{{ ansible_facts['default_ipv4']['address'] }}    {{ ansible_facts['hostname'] }}
```

Jinja2 템플릿은 데이터, 변수 및 식의 여러 요소로 구성됩니다. 변수와 식은 Jinja2 템플릿이 렌더링되면 동적으로 값이 작성됩니다. 템플릿에서 사용된 변수는 플레이북의 `vars` 섹션에서 지정할 수 있습니다. 또한 관리호스트의 팩트를 템플릿에서 변수로 사용할 수 있습니다. 템플릿 파일은 대부분 플레이북을 위한 프로젝트의 `templates` 디렉터리에 보관되고 일반적으로 `.j2` 파일 확장자가 할당되어 파일이 Jinja2 템플릿 파일임을 나타내고 있습니다. 다음은 `sshd` 설정 템플릿 파일의 예시입니다.

```
# {{ ansible_managed }}
# DO NOT MAKE LOCAL MODIFICATIONS TO THIS FILE BECAUSE THEY WILL BE LOST
Port {{ ssh_port }}
ListenAddress {{ ansible_facts['default_ipv4']['address'] }}

HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

SyslogFacility AUTHPRIV

PermitRootLogin {{ root_allowed }}
AllowGroups {{ groups_allowed }}

AuthorizedKeysFile /etc/.rht_authorized_keys .ssh/authorized_keys

PasswordAuthentication {{ passwords_allowed }}
```

템플릿 파일을 통해 관리 호스트에 구성파일을 배포 하기 위해서 **template** 모듈을 사용합니다. **copy** 모듈과 마찬가지로 src와 dest 속성을 지정합니다.

```
tasks:
  - template:
      src: /tmp/j2-template.j2
      dest: /tmp/dest-config-file.txt
```

실습

- 파일 타입 변경 파일 권한 변경 실습1

```
# file.yaml
- name: Set Context
  hosts: seoul
  tasks:
    - name: Touch a file and set permissions
      file:
        path: /etc/samba_file
        owner: ansible-user
        mode: '0640'
        state: touch

    - name: Set SELinux context
      file:
        path: /etc/samba_file
        setype: samba_share_t
```

- 확인

```
ansible seoul -a 'ls -Z /etc/samba_file'
```

```
[ansible-user@controller ansible]$ ansible-playbook file/file.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure, it should be a dictionary, got:
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:20: Section [korea:childrun] has
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [Set Context] *****
TASK [Gathering Facts] *****
ok: [servera]

TASK [Touch a file and set permissions] *****
changed: [servera]

TASK [Set SELinux context] *****
changed: [servera]

PLAY RECAP *****
servera                : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible-user@controller ansible]$ ansible seoul -a 'ls -Z /etc/samba_file'
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure, it should be a dictionary, got:
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:20: Section [korea:childrun] has
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
servera | CHANGED | rc=0 >>
unconfined_u:object_r:samba_share_t:s0 /etc/samba_file
```

- 파일 삭제 실습 2

```
# delete-file.yaml
- name: Set Context
  hosts: seoul
  tasks:
    - name: delete file if it's present
      file:
        dest: /etc/samba_file
        state: absent
```

- 확인

```
ansible seoul -a 'ls -Z /etc/samba_file'
```

```
[ansible-user@controller ansible]$ ansible-playbook file/delete-file.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [Set Context] *****

TASK [Gathering Facts] *****
ok: [servera]

TASK [delete file if it's present] *****
changed: [servera]

PLAY RECAP *****
servera : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

[ansible-user@controller ansible]$ ansible seoul -a 'ls -Z /etc/samba_file'
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
servera | FAILED | rc=2 >>
ls: cannot access '/etc/samba_file': No such file or directorynon-zero return code
```

- 파일 복사 실습3

```
# copy.yaml
- name: Copy File
  hosts: seoul
  tasks:
    - name: copy file
      copy:
        src: /home/ansible-user/ansible/file/file.yaml
        dest: /etc/file.yaml
```

- 결과

```
ansible seoul -a 'ls /etc/file.yaml'
```

```
[ansible-user@controller ansible]$ ansible-playbook file/copy.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has i
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/an
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [Copy File] *****

TASK [Gathering Facts] *****
ok: [servera]

TASK [copy file] *****
changed: [servera]

PLAY RECAP *****
servera : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0

[ansible-user@controller ansible]$ /home/ansible-user/ansible/file/file.yaml^C
[ansible-user@controller ansible]$ ansible seoul -a 'ls /etc/file.yaml'
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has i
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/an
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
servera | CHANGED | rc=0 >>
/etc/file.yaml
```

- fetch 실습 4

```
# fetch.yaml
- name: Fetch File
  hosts: seoul
  vars:
    - user: ansible-user
  tasks:
    - name: fetch from host
      fetch:
        src: "/home/{{ user }}/.ssh/authorized_keys"
        dest: "files/keys/{{ user }}.pub"
```

- 결과

```
[ansible-user@controller ansible]$ ansible-playbook file/fetch.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure, it should be a dictionary, got:
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:20: Section [korea] has unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[DEPRECATION WARNING]: Specifying a list of dictionaries for vars is deprecated in favor of specifying a dictionary. This feature will be removed in version 2.18. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.

PLAY [Fetch File] *****

TASK [Gathering Facts] *****
ok: [servera]

TASK [fetch from host] *****
changed: [servera]

PLAY RECAP *****
servera                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
[ansible-user@controller ansible]$ cat file/files/keys/ansible-user.pub/servera/home/ansible-user/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCKf0rXorgSjWZMOMUq/+xjWhZ/bEsR0Rpr84JI7141URaqfXgVzUglUnoXXkLDLpJ/ATN5hz549gwXNY+vtRtTYT9zo1Bsb/42PkGEaObUbMiM0wmR/L90Ck
sm2Gaud/4US0EqKS10lejcn7JUSaSuilLYKuWMBBG8RyC/bc+GcJSXvP/Vpsu ibRfRiYMuXSkHCyry6I/7et6tWbqwup09w2b1B9m1RnV09uekFq9mLmIL40DzyijuxmLdFc91VhpkEBZ/JYjLR5620fpxASv
x8B5v096JtUxI2lucqkm4sv9Kw0c5+b6CWvZX4me/ZKzAjnZkVvPVR+4Bq77a/COF5r2L4MFsU07R12z ffAgV1/p0n3WEKX+Hj6FJ38Yq07XhHhVhXYM id0q6PcIn5LPjUG45+v5sWwXHTuBRX2LIhpvacxY
yrzkI7Ho42ZWUjh3Khm2Wra/uLRZy+crhKo6qj8C0/hgI/SgNU9ELvQoR1mIQa/QHxx+B0v20E= ansible-user@controller
[ansible-user@controller ansible]$
```

• stat 실험 5

```
# stat.yaml
- name: stat module
  hosts: seoul
  tasks:
    - name: check stat module
      stat:
        path: /var/log/secure
        checksum_algorithm: md5
        register: result

    - name: check result
      debug:
        var: result
```

• 결과

```
[ansible-user@controller ansible]$ ansible-playbook file/stat.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure, it should be a dictionary, got:
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:20: Section [korea:childrun] has unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [stat module] *****

TASK [Gathering Facts] *****
ok: [servera]

TASK [check stat module] *****
ok: [servera]

TASK [check result] *****
ok: [servera] => {
  "result": {
    "changed": false,
    "failed": false,
    "stat": {
      "exists": false
    }
  }
}

PLAY RECAP *****
servera                : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```


- lineinfile 실습 6

```
# lineinfile.yaml
- name: Add a new line before the matching pattern/line.
  hosts: servera.example.com
  tasks:
    - name: Add a new line
      lineinfile:
        path: /etc/hosts
        line: '10.178.0.7 windows.example.com'
```

- 결과

```
ansible-user@controller ansible]$ ansible-playbook file/lineinfile.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure, it should be a list of dicts.
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:20: Section not found: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [Add a new line before the matching pattern/line.] *****
TASK [Gathering Facts] *****
ok: [servera]

TASK [Add a new line] *****
changed: [servera]

PLAY RECAP *****
servera : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

```
hosts
[root@servera etc]# cat hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
[root@servera etc]# cat hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
10.178.0.7 windows.example.com
```

- blockinfile 실습7

```
# blockinfile.yaml
- name: block test
  hosts: seoul
  tasks:
    - name: block test
      blockinfile:
        path: /home/ansible-user/index.html
        block: |
          Welcome to the Hello world.
          Show your ability in Hello world.
        state: present
        create: yes # 기존에 index.html이 없기에
```

- 결과

```
[ansible-user@controller ansible]$ ansible-playbook file/blockinfile.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure, it s
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:20: Se
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [block test] *****

TASK [Gathering Facts] *****
ok: [servera]

TASK [block test] *****
changed: [servera]

PLAY RECAP *****
servera                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
[root@servera ansible-user]# ls
index.html
[root@servera ansible-user]# cat index.html
# BEGIN ANSIBLE MANAGED BLOCK
Welcome to the Hello world.
Show your ability in Hello world.
# END ANSIBLE MANAGED BLOCK
```

- 템플릿파일 실습 8

```
# config.j2
ServerName {{ ansible_fqdn }}
HostName {{ ansible_hostname }}
ListenAddress {{ ansible_facts['default_ipv4']['address'] }}
```

```
# jinja.yaml
- name: Template Test
  hosts: all
  tasks:
    - name: Create configuration file from template
      ansible.builtin.template:
        src: /home/ansible-user/ansible/file/config.j2
        dest: /tmp/dest-config-file.txt
```

- 결과

```
ansible all -a 'cat /tmp/dest-config-file.txt'
```

```
[ansible-user@controller ansible]$ ansible-playbook file/jinja.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [Template Test] *****

TASK [Gathering Facts] *****
ok: [serverb]
ok: [servera]

TASK [Create configuration file from template] *****
changed: [servera]
changed: [serverb]

PLAY RECAP *****
servera      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
serverb      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
[ansible-user@controller ansible]$ ansible all -a 'cat /tmp/dest-config-file.txt'
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML invent
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansibl
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
servera | CHANGED | rc=0 >>
# config.j2
ServerName servera
HostName servera
ListenAddress 192.168.150.132
serverb | CHANGED | rc=0 >>
# config.j2
ServerName serverb
HostName serverb
ListenAddress 192.168.150.133
```

역할

FHS

Filesystem Hierachy Standard

/bin → 실행 파일 /usr/bin

/home → 홈 디렉토리

/etc →

Ansible을 사용하여 복잡하거나 많은 작업을 수행해야 하는 플레이북을 작성한다면 관리하기가 힘들어 질 수 있습니다. 이 때 사용자는 여러 개의 플레이북 파일로 분리하여 통합 플레이북에 결합시켜 실행할 수 있습니다. 그래서 좀 더 가독성을 높이고 관리가 쉽게 구성할 수 있습니다.

가져오기와 포함하기

사용자는 크게 두 가지 방법으로 분리된 파일을 결합할 수 있습니다. **import** 혹은 **include** 키워드를 사용하여 파일을 가져오거나 포함시킬 수 있습니다. 가져오기(import)와 포함하기(include)는 의미는 비슷하지만 기술적인 차이가 있습니다. 가져오기는 정적으로 실행하는 반면 포함하기는 동적으로 처리합니다. 간단히 보면, 정적인 방법은 플레이를 실행하는 순간 변수를 읽어오는 변수가 끝날 때 까지 그대로이지만 동적인 방법은 플레이가 실행되며 변경되는 부분을 바로 반영하여 후속 작업에 영향을 미치게 되는 것입니다.



따라서 include playbook을 사용하면 예기치 않게 실패하는 경우가 생길 수도 있다.

import_playbook 혹은 **include_playbook** 키워드는 분리된 파일을 결합하는 키워드입니다. 결합하는 파일은 온전한 플레이북 파일이기 때문에 **import_playbook/include_playbook** 기능은 플레이북의 최상위 수준에서만 사용할 수 있으며 플레이 내에서는 사용할 수 없습니다. 여러 개의 플레이북을 결합하는 경우 순서대로 실행합니다. 다음은 두 개의 추가 플레이북을 결합하는 통합 플레이북 예시입니다.

```
name: database play
  include_playbook: db.yaml
- name: webserver play
  import_playbook: web.yaml
```

import_tasks 혹은 **include_tasks**는 플레이 하위에 작성하는 작업 목록만 분리한 파일을 결합하는 키워드입니다. 마찬가지로 정적/동적 결합을 지원하며 플레이 레벨이 아니라 하위 레벨이기 때문에 분리된 파일에는 **hosts** 속성은 정의되어 있지 않습니다.

```
--
- name: web server installation hosts: webserver tasks:
  - import_tasks: webserver
```

가져오기와 포함하기를 사용할 때 주의 할 점은 변수를 사용하는 반복문 혹은 조건문을 사용하는 방식이 서로 다름을 이해하는 것입니다. 예를 들어 **when** 조건을 사용할 때 정적인 import를 사용하면 분리된 파일의 모든 작업에 조건을 적용하여 판단하지만 동적인 include 를 사용하면 결합할지에 대한 조건으로 판단하여 결합되고 나면 파일에 있는 모든 작업이 그대로 수행됩니다. 간단히 보면 분리된 파일의 작업을 모두 실행하거나 혹은 **when** 조건을 적용하여 실행하거나 두 가지로 나뉘게 됩니다.

분리된 파일에 정의한 변수는 결합한 통합 플레이북에 변수값을 입력할 수 있습니다.

```
# task.yaml
- name: install {{ package }}
  yum:
    name: "{{ package }}"
    state: present
- name: start {{ service }}
  service:
    name: "{{ service }}"
    enabled: true
    state: started
```

```
tasks:
  - import_tasks: task.yaml
    vars:
      package: httpd
      service: httpd
```

역할 (Role)

포함하거나 가져온 다수의 파일과 다양한 상황을 관리하는 작업 및 핸들러로 인해 플레이가 길고 복잡할 수 있습니다. 이를 작성한 사람이 만들어 놓은 코드 파일을 공유한다면 다른 사용자는 의미를 파악하거나 논리적인 흐름을 놓쳐 문제가 발생할 여지가 매우 높습니다.

Ansible 역할을 사용하면 일반적인 Ansible 코드를 더 쉽게 재사용할 수 있습니다. 인프라를 프로비저닝하거나 애플리케이션을 배포하는 데 필요한 모든 작업, 변수, 파일, 템플릿, 기타 리소스를 **표준화된 디렉터리 구조로 패키징**할 수 있습니다.

역할로 구성된 디렉터리를 복사하여 다른 프로젝트로 복사한 다음 플레이 내에서 해당 역할을 호출하면 훨씬 편하게 공유하고 재사용할 수 있습니다. 잘 작성된 역할은 플레이북에서 변수를 사용해 사이트별 호스트 이름, IP 주소, 사용자 이름, 비밀번호 또는 기타 로컬의 특정 세부 정보를 설정하여 역할의 동작을 수정할 수 있습니다. 예를 들어 데이터베이스 서버를 배포하는 역할은 호스트 이름, 데이터베이스 관리 사용자 및 암호와 설치에 맞게 사용자 지정된 기타 매개 변수를 설정하는 변수를 지원하도록 작성할 수 있습니다. 그래서 개발 및 QA 그리고 프로덕션에서 같은 역할을 사용하지만 서로 다른 변수 집합을 통해 현재 환경에 맞도록 구성하는 것입니다. 또한 역할을 호출하는 플레이에 해당 변수를 설정하지 않을 경우 기본 값을 설정하도록 할 수 있습니다.

Ansible 역할에는 다음과 같은 장점이 있습니다.

- 다른 사용자와 쉽게 공유
- 웹 서버, 데이터베이스 서버 또는 Git 리포지토리와 같은 시스템 유형의 필수 요소를 정의
- 대규모 프로젝트를 쉽게 관리
- 다른 사용자와 동시에 개발(비선형 구조)

자신의 역할을 작성, 사용, 재사용 및 공유하는 것 외에도 다른 소스에서 역할을 가져올 수 있습니다. **Ansible Galaxy** 웹 사이트(galaxy.ansible.com/)에서 다른 사용자가 만들어 놓은 역할을 찾고 다운로드하여 검토 한 뒤 재사용할 수 있습니다.

디렉터리 구조

역할은 표준화 된 구조의 디렉토리를 사용합니다. 최상위 디렉토리는 역할 이름으로 생성하며 하위 디렉토리 이름은 각각의 역할에 맞게 정해진 이름을 사용합니다. 다음은 하위 디렉토리 이름과 그 역할에 대한 표입니다.

디렉토리	역할
defaults	main.yml 파일에 기본 변수와 값을 정의. 우선순위가 낮음
files	작업에서 복사하려는 정적인 파일 저장
handlers	main.yml파일에 핸들러 정의
meta	main.yml파일에 작성자 라이선스 플랫폼 및 종속성 정보 작성
tasks	main.yml에 작업 내용 정의
templates	템플릿 문법으로 작성된 Jinja2 파일 저장
tests	역할을 테스트 하는데 사용할 수 있는 인벤토리와 test.yml 구성
vars	main.yml 파일에 변수값 정의. 우선순위 높음

역할에서 변수는 vars/main.yml 혹은 defaults/main.yml로 정의 합니다. defaults 디렉토리는 우선순위가 낮아 기본값을 설정하여 나중에 재정의 할 수 있도록 활용하며 vars 디렉토리는 우선순위가 높아 인벤토리 변수로는 재정의가 안됩니다.

역할 생성

역할은 roles 디렉토리의 하위에 역할 이름으로 디렉토리를 구성합니다. 여기서 roles 디렉토리의 위치는 기본적으로 /usr/share/ansible/roles 혹은 /etc/ansible/roles 등으로 지정 되어 있지만 ansible.cfg 파일에서 defaults 섹션의 roles_path 속성으로 재정의 할 수 있습니다.

```
[defaults]
...(생략)...
roles_path = /usr/local/roles
```

사용자는 해당 디렉토리 하위에 역할의 이름으로 역할의 최상위 디렉토리를 구성하고 표준화된 하위 디렉토리를 구성할 수 있습니다. 예를 들어 작업 디렉토리 하위에 roles 디렉토리를 지정하고 wordpress 라는 역할을 구성한다면 다음과 같은 디렉토리 구조를 가지게 됩니다.

```
.
├─ ansible.cfg
├─ inventory
├─ roles
│   └─ wordpress
│       ├── defaults
│       │   └─ main.yml
│       ├── files
│       ├── handlers
│       │   └─ main.yml
│       ├── meta
│       │   └─ main.yml
│       ├── README.md
│       ├── tasks
│       │   └─ main.yml
│       ├── templates
│       ├── tests
│       │   ├── inventory
│       │   └─ test.yml
│       └─ vars
│           └─ main.yml
└─ site.yaml
```

디렉토리 구조는 사용자가 직접 **mkdir** 명령어를 통해 생성하는 대신 **ansible-galaxy** 명령어를 통해 기본 구조 스켈레톤 (skeleton)을 자동으로 생성할 수 있습니다. 다음 명령어는 roles 디렉토리에 새로운 역할을 위한 디렉토리 구조를 생성하는 명령어입니다.

```
ansible-galaxy init wordpress
```

역할 이름을 입력해서 최상위 디렉토리를 생성하고 그 하위에 표준화된 디렉토리를 생성한 뒤 비어있는 파일 main.yml 등을 자동으로 구성합니다. 사용자는 자동으로 만들어진 디렉토리와 파일을 통해 역할구성을 보다 간편하게 시작할 수 있습니다.

외부 소스에서 역할 구성

사용자가 처음부터 수동으로 만드는 대신 커뮤니티에 다른 사용자가 만든 역할을 가져와 재사용 할 수 있습니다. git 레포지토리 혹은 Ansible Galaxy와 같은 오픈소스 커뮤니티에서 많은 역할을 검색하고 다운로드 하여 로컬환경에 맞도록 재지정 후 사용할 수 있습니다.

ansible-galaxy 명령어는 역할에 대한 정보를 검색하거나 다운로드 및 설치 그리고 제거 등을 지원합니다. 설치 및 관리하고자 하는 역할에 대한 정보는 **roles/requirements.yml** 파일에 입력합니다. 예를 들어 github에 사용자가 공유한 역할 디렉토리가 저장되어 있는 경우 다음과 같이 작성합니다.

```
# roles/requirements.yml
- src: https://github.com/USERNAME/wordpress.role
  scm: git
  version: "1.5.0"
```

git에 저장된 역할에서 version 정보는 브랜치 이름, 태그 및 git commit 해시 정보 등으로 입력할 수 있습니다. 만약 버전이 지정되지 않으면 항상 최신 브랜치에서 커밋을 사용합니다.

역할을 설치하기 위해서 다음과 같이 **ansible-galaxy role install** 명령어를 사용합니다.

```
ansible-galaxy role install -r roles/requirements.yml -p roles
```

- r 옵션으로 역할에 대한 정보가 입력되어 있는 파일을 지정하고, -p 옵션으로 설치하려는 대상 디렉토리의 위치를 지정할 수 있습니다. -p 옵션이 생략되었을때 ansible.cfg 파일의 defaults 섹션에는 해당 위치를 지정하는 기본 roles 디렉토리 경로를 입력할 수 있는 **roles_path** 속성을 지원합니다.

로컬에 설치한 역할은 **ansible-galaxy list** 명령어로 확인할 수 있습니다.

```
ansible-galaxy list
```

또한 삭제 할 때 는 **remove** 하위 명령어를 사용합니다.

```
ansible-galaxy remove wordpress
```

모듈로 역할 사용하기

플레이북에서 역할을 사용하기 위해 호출할 때는 두가지 방법이 있습니다.

tasks의 모듈로 정적 혹은 동적으로 추가

플레이에서 roles 키워드를 사용

먼저 import_role 혹은 include_role을 사용하는 방법이 있습니다. 두가지의 차이는 정적/동적인 방법이며 마찬가지로 조건문 및 반복문을 사용하는 방식이 다릅니다.

include_role의 경우 조건과 반복을 전체로 적용하지만 import_role은 역할 내부의 각 작업별로 조건과 반복을 설정할 수 있습니다. 다음은 역할을 가져오기와 포함하기의 예시입니다.

```
- hosts: import role
  tasks:
    - import_role:
        name: myrole1
    - include_role:
        name: myrole2
```

해당 역할의 변수와 정적/동적(jinja2) 파일 그리고 핸들러와 작업목록을 가져와 추가하여 플레이북에서 실행합니다. 그리고 역할에 대한 변수는 플레이북에서 정의 할 수 있습니다.

```
tasks:
  - import_role:
      name: role1
  vars:
    var1: value1
    var2: value2
```

플레이에서 roles 사용하기

roles 키워드는 플레이 수준에서 역할을 호출합니다. 하위수준인 작업 목록에서 모듈을 사용하는 이전 방법과는 달리 간편하게 호출할 수 있습니다.

```
- name: sample play
  hosts: all
  roles:
    - role: role1
    - role: role2
```

만약 tasks 항목과 같이 작성된다면 roles는 항상 tasks 항목 이전에 실행됩니다. 다음처럼 roles 키워드가 플레이 북에서 후 순위로 나열되어 있다고 하더라도 실행은 항상 tasks 이전에 실행됩니다.

```
- name: sample play
  hosts: all
  tasks:
    - debug:
        msg: "Run after roles"
  roles:
    - role: role1
    - role: role2
```

각 역할마다 사용할 변수는 항목별로 지정할 수 있습니다.

```
- name: sample play
  hosts: all
  roles:
    - role: role1
      var1: value1
```



```

    var2: value2

- role: role2

```

들여쓰기가 귀찮거나 내용을 축소할 경우 속기를 사용할 수 있지만 권장하지는 않습니다. 다음은 위의 예시를 속기로 작성한 예시입니다.

```

- name: sample play
  hosts: all
  roles:
    - { role: role1, var1: value1, var2: value2 }
    - role: role2

```

역할과 함께 많이 사용하는 작업 순서를 조절하는 키워드로 **pre_tasks**와 **post_tasks**가 있습니다. **pre_tasks**는 roles 보다 우선 실행됩니다. 그리고 알림을 받은 핸들러 역시 먼저 실행됩니다. **post_tasks**는 모든 tasks가 끝난 뒤 핸들러 까지 실행을 마치고 나면 실행합니다. 다음은 실행순서를 표현한 간단한 예시입니다.

```

- hosts: all
  pre_tasks:
    - debug:
        msg: pre_tasks
        notify: myhandler
        changed_when: true
  roles:
    - role: role1
  tasks:
    - debug:
        msg: tasks
        notify: myhandler
        changed_when: true
  post_tasks:
    - debug:
        msg: post_tasks
        notify: myhandler
        changed_when: true
  handlers:
    - name: myhandler
      debug:
        msg: handler

```

여기서 핸들러는 **pre_tasks , roles/tasks, post_tasks** 가 실행이 끝난 시점에 총 세 번 실행 됩니다

컬렉션

Ansible은 역할에 필요한 모듈을 필요한 경우 직접 개발해서 사용하는 것을 허용합니다. 그래서 ISV에서 직접 개발한 모듈과 플러그인 및 역할을 함께 배포하기 위해 Ansible은 컬렉션(Collection)을 사용합니다. 그래서 Ansible core 모듈과 별도로 모듈 및 플러그인들을 개발할 수 있습니다. 예를 들면 다음과 같은 컬렉션들이 있습니다.

- redhat.insight : RHEL 용 모듈 및 역할 제공
- cisco.ios : 시스코 장비를 지원하는 모듈과 플러그인 제공
- community.crypto : SSL/TLS 인증서를 생성 관리하는 모듈을 제공

컬렉션을 사용하기 위해서 **ansible-galaxy collection** 하위 명령어를 지원합니다. 다음은 컬렉션을 설치하는 예시입니다..

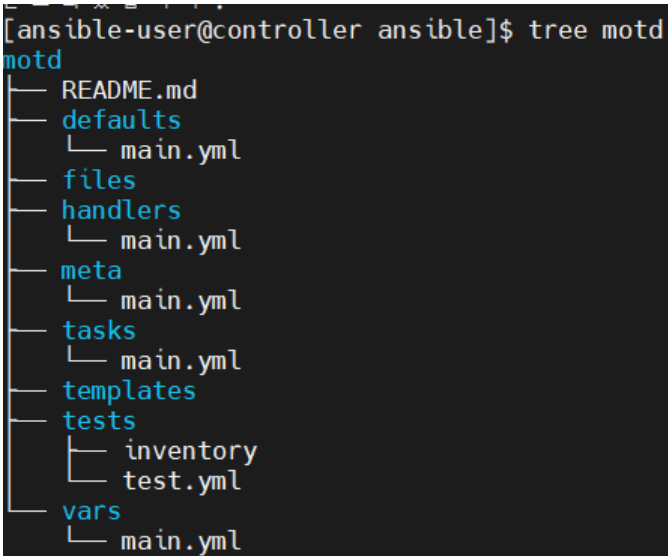
```
ansible-galaxy collection install community.crypto \
-r collections/ requirements.yml \
-p ./collections
```

역할 실습

1. 실습 1

- 역할 생성

```
ansible-galaxy init motd
sudo yum install tree -y
tree motd
```



```
[ansible-user@controller ansible]$ tree motd
motd
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
├── vars
│   └── main.yml
```

- Task yaml 작성 - motd

```
# motd/tasks/main.yml
---
- name: copy motd file
  template:
    src: templates/motd.j2
    dest: /etc/motd
    owner: root
```

```
group: root
mode: '0444'
```

```
# motd/templates/motd.j2
Welcome to {{ ansible_hostname }}
This file was created on {{ ansible_date_time.date }}
Go away if you have no business being here
Contact {{ system_manager }} if anything is wrong
```

```
# motd/defaults/main.yml
system_manager: admin@example.com
```

- **motd role - motd**

```
# motd-role.yml
---
- name: use motd role
  hosts: seoul
  remote_user: ansible-user
  become: true
  roles:
    - motd
```

- **playbook 실행 및 확인 - motd**

```
ansible-playbook motd-role.yml

ansible seoul -m command -a 'cat /etc/motd'
```

```

[ansible-user@controller ansible]$ ansible-playbook motd-role.yaml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inve
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansi
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [use motd role] *****

TASK [Gathering Facts] *****
ok: [servera]

TASK [motd : copy motd file] *****
changed: [servera]

PLAY RECAP *****
servera                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    re

[ansible-user@controller ansible]$ ansible seoul -m command -a 'cat /etc/motd'
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inve
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansi
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
servera | CHANGED | rc=0 >>
# motd/templates/motd.j2
Welcome to servera
This file was created on 2024-07-30
Go away if you have no business being here
Contact dlwpdnr213@naver.com if anything is wrong
[ansible-user@controller ansible]$

```

2. 실습2

• Roles 생성 – vhost

```

mkdir roles ; cd roles
ansible-galaxy init vhost-ansible
rm -rf vhost-ansible/{defaults,vars,tests}

```

cmd1 ; cmd2 : 앞 cmd 이후에 실행

cmd1 && cmd2 : 앞 명령어 정상적으로 끝나고 이후에 실행

cmd1 || cmd2 : 앞 명령어 정상적으로 안 끝나면 실행

cmd1 | cmd2: 앞 명령어의 결과를 cmd2로 넘긴다 파이프라인

• Role 구성

```

# vhost-ansible/tasks/main.yml
---
- name: Install httpd package
  yum:

```

```

    name: httpd
    state: latest

- name: Create directory
  file:
    path: /var/www/vhosts/{{ ansible_hostname }}
    state: directory

- name: Started and enabled httpd service
  service:
    name: httpd
    state: started
    enabled: true

- name: Install vhost file
  template:
    src: vhost.conf.j2
    dest: /etc/httpd/conf.d/vhost.conf
    owner: root
    group: root
    mode: '0644'
  notify:
    - restart httpd

```

```

# vhost-ansible/handlers/main.yml
---
- name: restart httpd
  service:
    name: httpd
    state: restarted

```

```

# vhost-ansible/templates/vhost.conf.j2

# {{ ansible_managed }}
ServerAdmin admin@{{ ansible_fqdn }}
ServerName {{ ansible_fqdn }}
ErrorLog logs/{{ ansible_hostname }}-error.log
CustomLog logs/{{ ansible_hostname }}-common.log common
DocumentRoot /var/www/vhosts/{{ ansible_hostname }}/
Options +Indexes +FollowSymlinks +Includes

```

```

mkdir -pv files/html
echo 'Hello World index file' > files/html/index.html

```

- vhost 역할로 플레이 실행

```
# vhost-role.yml
---
- name: Use vhost-ansible role playbook
  hosts: localhost
  become: true
  pre_tasks:
    - name: pre_tasks message
      debug:
        msg: 'Check web server configuration.'
  roles:
    - vhost-ansible
  post_tasks:
    - name: Check HTML content installed
      copy:
        src: files/html/
        dest: "/var/www/vhosts/{{ ansible_hostname }}"
    - name: post_tasks message
      debug:
        msg: 'Web server is created.'
```

• 확인

```
[ansible-user@controller ansible]$ ansible-playbook vhost-role.yml
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with yaml plugin: YAML inventory has invalid structure,
<class 'ansible.parsing.yaml.objects.AnsibleUnicode'>
[WARNING]: * Failed to parse /home/ansible-user/ansible/inventory with ini plugin: /home/ansible-user/ansible/inventory:20
unknown type: childrun
[WARNING]: Unable to parse /home/ansible-user/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available

PLAY [Use vhost-ansible role playbook] *****

TASK [Gathering Facts] *****
ok: [servera]

TASK [pre_tasks message] *****
ok: [servera] => {
  "msg": "Check web server configuration."
}

TASK [vhost-ansible : Install httpd package] *****
ok: [servera]

TASK [vhost-ansible : Create directory] *****
ok: [servera]

TASK [vhost-ansible : Started and enabled httpd service] *****
ok: [servera]

TASK [vhost-ansible : Install vhost file] *****
ok: [servera]

TASK [Check HTML content installed] *****
changed: [servera]

TASK [post_tasks message] *****
ok: [servera] => {
  "msg": "Web server is created."
}

PLAY RECAP *****
servera                : ok=8    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
cat /etc/httpd/conf.d/vhost.conf
cat /var/www/vhosts/servera/index.html
```

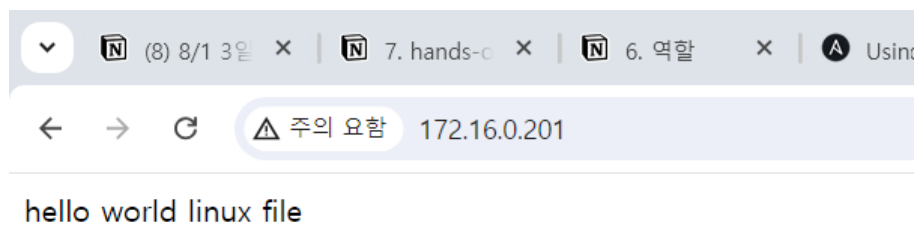
```
[root@servera tmp]# cat /etc/httpd/conf.d/vhost.conf
# vhost-ansible/templates/vhost.conf.j2

# Ansible managed
ServerAdmin admin@servera
ServerName servera
ErrorLog logs/servera-error.log
CustomLog logs/servera-common.log common
DocumentRoot /var/www/vhosts/servera/
Options +Indexes +FollowSymLinks +Includes
```

```
[root@servera tmp]# cat /var/www/vhosts/servera/index.html
hello world linux file
```

```
curl servera
```

```
[ansible-user@controller ansible]$ curl servera
hello world linux file
```



앤서블 사용처

컨테이너 기술은 어렵고 인력이 많이 필요하고 수준 높은 엔지니어가 필요하기에 꼭 필수적인 것은 아니다. 컨테이너 필요 없이 구현 가능하다면 앤서블도 좋은 선택지

또한 네트워크 장비같은 경우 컨테이너로 띄울 수는 없다. 따라서 이런 경우에는 앤서블을 사용해야 한다.

커널

command line → keyboard → terminal → shell → kernal

앤서블로 k8s 환경 구성

```
- name: kubernetes install play
hosts: all
vars:
  controlplane_IP: "172.16.0.201"
  TOKEN: "kdigit.1234567890hellow"
  kube_config_dir: /etc/kubernetes
tasks:
- name: download docker
  get_url:
    url: https://download.docker.com/linux/centos/docker-ce.repo
    dest: /etc/yum.repos.d/docker-ce.repo
- name: kernel module enable
  modprobe:
    name: "{{ item }}"
    state: present
    persistent: present
  loop:
    - br_netfilter
    - overlay
- name: sysctl configuration
  sysctl:
    name: "{{ item }}"
    value: "1"
    reload: yes
    sysctl_file: /etc/sysctl.d/k8s.conf
    state: present
    sysctl_set: yes
  loop:
    - net.bridge.bridge-nf-call-ip6tables
    - net.bridge.bridge-nf-call-iptables
    - net.ipv4.ip_forward
- name: permissive selinux
  ansible.posix.selinux:
    policy: targeted
    state: permissive
- name: firewalld disable
  service:
    name: firewalld
    state: stopped
```



```

    enabled: false
- name: remote swap
  shell: swapoff -a
  when: ansible_swaptotal_mb > 0 # fstab
- name: remove swap from fstab
  mount:
    name: none
    fstype: swap
    state: absent
- name: kubernetes repo
  yum_repository:
    name: k8s_repo
    description: k8s_repo
    baseurl: https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
    enabled: yes
    gpgcheck: yes
    repo_gpgcheck: yes
    gpgkey:
      - https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key

- name: download containerd and kubeadm
  yum:
    name: "{{ item }}"
    state: latest
  loop:
    - containerd
    - kubeadm
    - kubelet
- name: copy container config file
  copy:
    src: config.toml
    dest: /etc/containerd/config.toml

- name: enable service kubelet and cotnainerd
  service:
    name: "{{ item }}"
    state: started
    enabled: true
  loop:
    - containerd
    - kubelet

- name: copy /etc/hosts
  copy:
    src: /etc/hosts
    dest: /etc/hosts

- name: Check if kubelet.conf exists
  stat:
    path: "{{ kube_config_dir }}/kubelet.conf"

```

```

    get_attributes: no
    get_checksum: no
    get_mime: no
    register: kubelet_conf

- name: init cluster
  shell: |
    kubeadm init --apiserver-advertise-address={{ controlplane_IP }} \
                --pod-network-cidr=192.168.0.0/16 \
                --token={{ TOKEN }}
  when: inventory_hostname in groups['controlplanes'] and (not kubelet_conf.stat
  ignore_errors: yes

- name: join cluster
  shell: |
    kubeadm join --token={{ TOKEN }} \
                --discovery-token-unsafe-skip-ca-verification \
                {{ controlplane_IP }}:6443
  when: inventory_hostname in groups['worker_nodes'] and (not kubelet_conf.stat

```



ansible-doc 명령어로 모르는 모듈 사용법을 알 수 있다.

• 결과

```

TASK [download containerd and kubeadm] *****
ok: [servera] => (item=containerd)
ok: [servera] => (item=kubeadm)
changed: [servera] => (item=kubelet)
changed: [serverb] => (item=containerd)
changed: [serverb] => (item=kubeadm)
changed: [serverb] => (item=kubelet)

TASK [copy container config file] *****
ok: [servera]
changed: [serverb]

TASK [enable service kubelet and cotnainerd] *****
ok: [servera] => (item=containerd)
changed: [serverb] => (item=containerd)
changed: [servera] => (item=kubelet)
changed: [serverb] => (item=kubelet)

TASK [copy /etc/hosts] *****
changed: [servera]
changed: [serverb]

TASK [Check if kubelet.conf exists] *****
ok: [servera]
ok: [serverb]

TASK [init cluster] *****
skipping: [serverb]
changed: [servera]

TASK [join cluster] *****
skipping: [servera]
changed: [serverb]

PLAY RECAP *****
servera      : ok=14   changed=4   unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
serverb      : ok=15   changed=13  unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

```

```
[root@servera /]# kubelet --version
Kubernetes v1.30.3
```