

Anssible 이란?

여러 서버·네트워크 장비의 설정, 배포, 운영 작업을 코드 베이스 자동화하는 오픈소스 IT 자동화 IaC 도구

playbook 이라는 yaml 파일의 명령을 inventory 의 호스트들에게 적용 시킨다.

ssh 인증서 방식으로 연결해 명령어 실행하는 동작처럼 커넥팅할 수 있다.

Linux에서는 SSH, Windows에서는 WinRM, 네트워크 장비에서는 SSH/API 등을 이용한다.

대규모 IDC 운용에 사용된다.

INVENTORY

대상 호스트를 그룹화해서 관리 및 사용한다.

경로 : /etc/ansible/ 안에 inventory 또는 hosts 를 만들어주면 된다. 이름은 필요한 대로 만들면 된다.

vim inventory

Plain text

[web]

192.168.10.[11:12]

[db]

www.db.com

[cisco]

192.168.20.1

192.168.20.2

[group : children]

db

cisco

그룹 단위로 호스트를 지정하거나 그룹 대상으로 명령어를 실행할 수 있다.

● 뽀모도로

```
hosts: web
ansible web -m ping
```

Plain text

PLAYBOOK

대상 파일과 실제 실행할 파일을 분리시킨다.

호스트의 Task를 정의하는 yaml 파일로, Task 자체가 ansible 에 라이브러리로 제공되고 있다.

예: 종료시켜줘 -> 각 시스템별 종료 명령어가 전부 라이브러리에 제공되어서 해당하는 명령어가 자동 실행.

```
- name: Configure Web Servers
hosts: web
become: yes
```

Plain text

```
tasks:
```

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

```
- name: Start nginx
  service:
    name: nginx
    state: started
    enabled: yes
```

Agentless

관리 대상에 별도의 Agent 를 설치하지 않는다.

Control node

Ansible을 설치하고 실제 자동화 명령을 실행하는 관리 주체

- 설치 | Inventory 보관 | Playbook 작성/실행 | SSH Key 관리 | managed node 에 명령 전달

Managed node

Ansible에게 관리받는 대상 서버나 네트워크 장비.

일반적인 Linux 서버라면 별도 Ansible Agent를 설치할 필요가 없다.

- 웹 | DB | DNS | Cisco | 기타 관리 대상

Idempotency 멱등성

같은 작업을 여러 번 반복 실행해도 최종 상태가 동일하게 유지되는 성질

ansible 은 "동작" 을 지시하는 게 아닌 "상태" 를 세팅하는 방식이라고 생각하면 된다.

원하는 상태가 이미 완성되어 있다면 불필요한 변경을 다시 수행하지 않는다.

```
[root@localhost .ssh]# ssh ansible@172.16.21.201 "mkdir kong"
ansible@172.16.21.201's password:
[root@localhost .ssh]# ssh ansible@172.16.21.201
ansible@172.16.21.201's password:
Last login: Mon Sep  7 11:43:23 2026 from 172.16.21.200
[ansible@localhost ~]$ ls
kong
```

Plain text

```
[root@localhost .ssh]# ssh ansible@172.16.21.201 "mkdir kong"
ansible@172.16.21.201's password:
mkdir: 'kong' 디렉토리를 만들 수 없습니다: 파일이 있습니다
```

개념이해

단일 호스트 원격 제어 : 쉬움

멀티 호스트 원격 제어 : FOR 쓰면 쉬움 + 멀티스레드

클라우드(IDC) 원격 제어 : 서로 같거나 다른 네트워크에 존재, 호스트 많음.

여러대의 호스트를 동시 제어할 수 있는 프로그램 필요 -> ansible

파이썬으로 제작됨.

설치

1. 제어노드 설치

- epel 등록 필요 : `dnf -y install epel-release`
- 앤서블 설치 : `dnf -y install ansible bash-completion vim-enhanced tar`
- 파이썬 설치 필요

tar : 서버에서는 배포시 tar 사용하기 때문에 tar 를 같이 설치할 것.

sshpass : 인증서 자동 로그인 작업

설치 확인 : `ansible --version`

```
[root@localhost .ssh]# ansible --version
```

Plain text

```
ansible [core 2.14.18]
```

```
config file = /etc/ansible/ansible.cfg
```

```
configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
```

```
ansible python module location = /usr/lib/python3.9/site-packages/ansible
```

```
ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
```

```
executable location = /usr/bin/ansible
```

```
python version = 3.9.25 (main, Aug 6 2026, 00:00:00) [GCC 11.5.0 20240719 (Red Hat 11.5.0-14)]
```

```
(/usr/bin/python3)
```

```
jinja version = 3.1.2
```

```
libyaml = True
```

네트워크 제어 시, ansible 의 cisco.ios 모듈을 사용해야 하는데 cisco 구형 12.4 버전은 최신 ansible 과 호환되지 않음.

ansible 2.12 버전까지만 호환성을 가진다.

가장 호환성이 좋은 버전은 2.9.27

하위버전 사용 필요시 :

ansible 삭제 -> python3 pip 설치 -> pip3 install ansible==2.9.27

2. 관리 노드

- ssh 설치, 연결 설정, 계정 설정

실습

사용 기초 : list graph cfg

```
# 인벤토리 옵션 완전 생략시 내 실행위치PWD 하위 인벤토리 실행 -> 없으면 /etc/ansible 에 있는 인벤토리 사용
# config 도 마찬가지다. PWD 에 있는 설정부터 사용
ansible -i ./inventory --list-host [그룹명]
ansible -i ./inventory --list-host all # 모든 그룹
ansible -i ./inventory --list-host ungrouped # 그룹 명칭이 아닌 기본명칭
```

ansible 에서 - 사용시 warning 발생.

```
[root@localhost ansible]# ansible-inventory -i ./inventory --graph all
@all:
  |--@ungrouped:
  |--@ubuntu:
  | |--172.16.0.202
  | |--172.16.0.201
  | |--172.16.0.200
  |--@rocky:
  | |--172.16.0.200
  | |--172.16.0.201
  |--@group:
  | |--@kkk:
  | | |--172.16.0.200
  | |--@domain:
  | | |--www.han.com
  | | |--wwa.han.com
```

Plain text

cfg 파일 생성

```
[defaults]
inventory = ./inventory
```

Plain text

사용기초 : 원격 명령어 실행

```
ansible kkk -i ./inventory -m ansible.builtin.command -a "mkdir koing" -u ansible
```

Plain text

```
ansible.cfg 안에
[default]
ask_pass = true # 비밀번호 입력 이 필요하다
```

```
ansible kkk -m ansible.builtin.command -a "mkdir koing" -u ansible
```

```
ansible.cfg 안에
[default]
remote_user = ansible # 추가
```

```
ansible kkk -m ansible.builtin.command -a "mkdir koing"
```

동작상태 확인

rc 를 보면 된다. rc=1 이면 실패 rc=0 이면 성공

Plain text

```
172.16.0.201 | FAILED | rc=1 >>
mkdir: 'koing' 디렉토리를 만들 수 없습니다: 파일이 있습니다non-zero return code
[root@localhost ansible]# ansible kkk -m ansible.builtin.command -a "rm -rf koing"
SSH password:
172.16.0.201 | CHANGED | rc=0 >>
```

권한 상승시키기

Bash / Shell

```
ansible aaa -m command -a "poweroff"
```

```
172.16.21.201 | FAILED | rc=1 >>
Call to PowerOff failed: Access deniednon-zero return code
172.16.21.202 | FAILED | rc=1 >>
Call to PowerOff failed: Interactive authentication required.non-zero return code
```

실패 원인 : 일반사용자인 ansible 에게는 권한이 없음

```
vim ansible.cfg
```

```
[privilege_escalation]
become = true # -k : ssh 사용하겠다. -K become 을 사용하겠다. 커맨드창에서 사용할 땐 -k -K 로 쓴
become_method = sudo #
become_user = root
become_ask_pass = true # sudo passwd 창 입력
```

```
[root@localhost ansible]# ansible aaa -m command -a "poweroff"
SSH password:
BECOME password[defaults to SSH password]:
```

```
172.16.21.202 | FAILED! => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
```

```

    },
    "changed": false,
    "module_stderr": "Shared connection to 172.16.21.202 closed.\r\n",
    "module_stdout": "\r\nansible is not in the sudoers file.\r\n",
    "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error",
    "rc": 1
}
172.16.21.201 | FAILED! => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "module_stderr": "Shared connection to 172.16.21.201 closed.\r\n",
  "module_stdout": "\r\n로컬 시스템 관리자에게 일반적인 지침을 받았으리라 믿습니다.\r\n보통 세가지로
  "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error",
  "rc": 1
}

```

일반 사용자 계정이 sudo 를 사용하더라도 sudoers 파일 안에 허용된 명령만 사용할 수 있다.
sudoers 에 ansible 계정이 poweroff 명령어를 사용할 수 있도록 허용해야 한다.

sudoers 관리

~(TODO)~

sudo visudo

```

99 ## Allow root to run any commands anywhere
100 root    ALL=(ALL)    ALL

```

Plain text

루트가 모든 명령어를 사용할 수 있는 이유 : ALL=(ALL)

```

99 ## Allow root to run any commands anywhere
100 root    ALL=(ALL)    ALL
101 ALL     ALL=(ALL) NOPASSWD: ALL

```

Plain text

```

[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = false

```

```
[root@localhost ansible]# ansible aaa -m command -a "poweroff"
```

SSH password:

```
172.16.21.202 | FAILED | rc=-1 >>
```

Missing sudo password

```
172.16.21.201 | FAILED | rc=-1 >>
```

Failed to connect to the host via ssh: System is going down. Unprivileged users are not permitted to log in anymore. For technical details, see pam_nologin(8).

Connection closed by 172.16.21.201 port 22

101 처럼 절대하지말것 일단 모든 계정에거 모든 권한을 다 주고 become passwd 도 쓰지 않겠다.

지금 Ansible 을 쓰고 있지만 root 권한을 사용하는 상태까지 와버렸다.

sudoers 를 안전하게 설정하는 방법을 "나중에" 알아보자.

모듈

ansible-doc command

Plain text

```
# 'cmd' is module parameter
```

```
- name: Run command if /path/to/database does not exist (with 'cmd' parameter)
```

```
ansible.builtin.command:
```

```
  cmd: /usr/bin/make_database.sh db_user db_name
```

```
  creates: /path/to/database
```

```
#플레이북으로 옮긴다
```

```
vim mk.yaml
```

```
--- #이거 꼭써야됨
```

```
- name: 1234 # 플레이북 이름 내맘
```

```
  hosts: aaa # 대상 인벤토리 그룹
```

```
  tasks:
```

```
    - name: mkdir 1234 # 첫번째 태스크
```

```
      ansible.builtin.shell: # 사용 모듈
```

```
        cmd: mkdir 1234 && ls # 모듈 인자
```

```
        register: result # 다 예약어다. 레지스터레 리설트 저장
```

```
- name: print #두번째 태스크
```

```
  debug:
```

```
var: result
```

Plain text

```
[root@localhost ansible]# vim mk.yaml
```

```
[root@localhost ansible]# ansible-playbook mk.yaml
```

```
SSH password:
```

```
PLAY [1234] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [172.16.21.201]
```

```
ok: [172.16.21.202]
```

```
TASK [mkdir 1234] *****
```

```
changed: [172.16.21.202]
```

```
changed: [172.16.21.201]
```

```
TASK [print] *****
```

```
ok: [172.16.21.201] => {
```

```
  "result": {
```

```
    "changed": true,
```

```
    "cmd": "mkdir 1234 && ls",
```

```
    "delta": "0:00:00.004002",
```

```
    "end": "2026-09-07 15:01:02.918153",
```

```
    "failed": false,
```

```
    "msg": "",
```

```
    "rc": 0,
```

```
    "start": "2026-09-07 15:01:02.914151",
```

```
    "stderr": "",
```

```
    "stderr_lines": [],
```

```
    "stdout": "1234\nkong",
```

```
    "stdout_lines": [
```

```
      "1234",
```

```
      "kong"
```

```
    ]
```

```
  }
```

```
}
```

```
ok: [172.16.21.202] => {
```

```
  "result": {
```

```
    "changed": true,
```

```
    "cmd": "mkdir 1234 && ls",
```

```
    "delta": "0:00:00.004691",
```

```
    "end": "2026-09-07 06:01:02.879215",
```

```
    "failed": false,
```

```
    "msg": "",
```

```
"rc": 0,
"start": "2026-09-07 06:01:02.874524",
"stderr": "",
"stderr_lines": [],
"stdout": "1234",
"stdout_lines": [
    "1234"
]
}
```

PLAY RECAP *****

```
172.16.21.201      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.21.202      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

```
"stdout_lines": [
    "1234",
    "kong"
]
는 LS의 결과값.
```

Plain text

df.yaml

```
---
- name: df
  hosts: aaa
  tasks:
    - name: disk free
      command:
        cmd: df
      register: result

    - name: print
      debug:
        var: result
```

Plain text

PLAY [1234] *****

TASK [Gathering Facts] *****

```
PLAY [df] ***** # 전체 플레이북 이름

TASK [Gathering Facts] ***** # 시작할때 호스트 찾는 태
스크
ok: [172.16.21.201]
ok: [172.16.21.202]

TASK [disk free] ***** # 태스크 1
changed: [172.16.21.202]
changed: [172.16.21.201]

TASK [print] ***** # 태스크 2
PLAY RECAP ***** # ???
172.16.21.201      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

SSH 인증서 생성 플레이북

1. 키 생성

```
---
- name: keygen 2048
  hosts: localhost
  tasks:
    - name: ssh keygen
      user:
        name: root
        generate_ssh_key: yes
        ssh_key_bits: 2048
        ssh_key_file: /root/.ssh/id_rsa
        force: yes
```

Plain text

2. 등록

```
- name: copy pub key to server
  hosts: aaa
  tasks:
    - name: Set auth_key taken from file
      authorized_key:
        user: ansible
        state: present # 설치된 상태로 만들어 달라. 프로그램 이름 알려주고 state 를present 를 해 주면 설치한다.
```

Plain text

상태.

```
key: "{{ lookup('file', '/root/.ssh/id_rsa.pub') }}"
```

3. 테스트

```
[root@localhost ansible]# ssh ansible@172.16.21.201
```

Plain text

```
Last login: Mon Sep 7 15:36:20 2026 from 172.16.21.200
```

```
[ansible@localhost ~]$ exit
```

후에 ansible.cfg 에서 become 을 false 로 바꿔주면 -> 인증서로 로그인 이 된다.

결론

Playbook 만 잘 만들면 된다!

하지만 줄바꿈, 띄어쓰기 하나만 틀려도 작동하지 않는다.

제목

텍스트를 입력하거나 /를 입력하여 명령을 입력하세요.