7 30 1일차

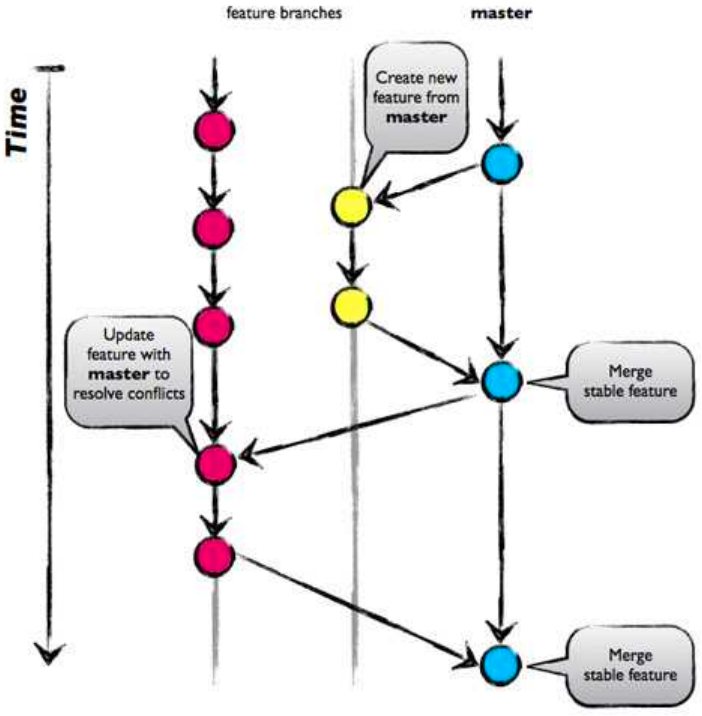
클라우드 요구 기술

- 클라우드 사용 경험
- 리눅스 쉘 스크립트 작성 가능
- 파이썬 스크립트 작성 가능

과 등등 필요하다..

cicd 파이프라인 구성
현업에서도 cicd는 굉장히 중요한 요소
마이크로 서비스 아키텍처 구현 패턴들 중요

깃 전략



국내에선 해당 깃 플로우 전략을 많이 사용

해외에선 복잡해짐을 겪었기에 메인 브랜치만을 사용하려 노력 중 브랜치 merge를 지속적으로 해주면서

데브옵스

개발적인 능력과 운영적인 능력을 모두 알아 개발자들과 운영자들 사이를 이어주는 직무

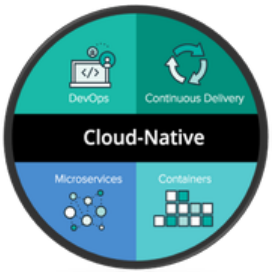
- 자동화된 스크립트 코드 작성 테스트

chaos Engineering
시스템이 격동의 예측치 못한 상황을 견딜 수 있도록 신뢰성을 쌓기 위해 운영 중인 소프트웨어 시스템에 실험을 하는 규율

이런 예상치 못한 상황에 빠졌을 때 해결하기 위한 자동화 코드를 작성하는 것 또한 데브옵스의 업무

이런 상황에 빠지지 않게 끔 하는 것도 데브옵스의 업무

클라우드 네이티브



- 컨테이너
  컨테이너 기반이다.

- cicd

- 마이크로 서비스
  마이크로 서비스 아키텍처 패턴으로 개발

- 데브옵스
  데브옵스가 cicd 구현

1. Ansible 소개

수년 전까지 대부분의 시스템 관리 및 인프라 관리는 그래픽이나 명령줄 인터페이스를 통해 수행된 수동 작업을 통해 구성되어져 왔습니다. 시스템 관리자는 표준 작업을 수행하기 위해 체크리스트를 만들고 관리 가이드 및 임기응변이나 축척된 경험들을 통해 시스템을 관리해 왔습니다.

하지만 이런 방식은 다양한 문제가 발생합니다 시스템 관리자가 단계를 건너뛰거나 작업을 할 때 실수하기가 쉽습니다 또한 각 서버를 수동으로 관리하기 때문에 구성이 동일해야 되는 여러 서버에서 차이가 생기기 쉽습니다 관리자가 현재 실행 중인 것과 똑같은 새 서버를 설정하려는 경우 설치된 모든 패키지를 검토하고 구성, 버전 및 설정을 문서화하는 데 많은 시간을 소비해야 합니다. 이로 인해서 유지 관리가 더 어려워지고 환경이 오류나 불안정이 생길 수 있습니다.

그래서 최근 시스템들은 자동화 방식을 많이 채택하고 있습니다 수동으로 시스템을 관리하거나 인프라를 관리할 때 발생하는 여러 문제들을 피할 수 있도록 도와줍니다 시스템 관리자는 자동화를 통해 모든 시스템을 신속하고 올바르게 배포하고 구성할 수 있습니다

Ansible이란

앤서블은 오픈 소스 자동화 플랫폼입니다 강력한 자동화 작업을 관리하고 다양한 워크플로우 및 환경에 맞게 조정할 수 있습니다 그리고 처음 사용하는 사용자의 경우에도 생산성을 높이기 위하여 매우 빠르게 활용할 수 있습니다. 다음은 앤서블의 세 가지 주요 특징입니다

- 앤서블은 강력합니다
  구성 관리 워크플로우 자동화, 네트워크 자동화용 어플리케이션을 배포할 수 있습니다 그래서 전체 어플리케이션의 라이프 사이클을 오케스트레이션 할 수 있습니다
- 앤서블은 에이전트가 필요 없습니다
  일반적으로 앤서블은 OpenSSH 또는 WinRM을 사용하여 관리하는 호스팅 연결하고 모듈이라는 소형 프로그램을 내보내서 해당 호스트에 연결합니다. 이러한 프로그램은 시스템을 원하는 특정 상태로 만드는데 사용이 되고 작업을 완료하면 제거됩니다 에이전트나 추가 사용자 지정 보안 인프라가 없기 때문에 ansible은 훨씬 효율적이고 안전합니다
- 앤서블은 간단합니다
  앤서블 코드는 사람이 읽을 수 있는 자동화를 제공합니다 즉 사람이 읽기 쉽고 이해하고 변경할 수 있는 자동화 도구입니다 작성하는 데는 특별한 코딩 기술이 필요하지 않습니다 그래서 처음 앤서블을 접하는 사용자들도 금방 배워서 사용할 수 있습니다

코드로 인프라 관리하기

앤서블과 같은 자동화 도구를 사용을 하면 코드로 인프라를 관리할 수 있습니다. 기계가 읽을 수 있는 자동화 언어를 사용을 해서 필요한 it 인프라 상태를 정의하고 설명할 수 있습니다. 이상적으로는 이 자동화 언어는 현재 상태를 쉽게 이해하고 변경할 수 있도록 사람이 읽기에도 쉬워야 합니다. 그런 다음이 코드를 인프라에 적용하여 실제로 해당 상태임을 확인합니다.

자동화 도구를 사용할 때 간단한 텍스트 파일로 표현하면 버전 제어 시스템에서 관리하기가 쉽습니다 이때는 모든 변경 사항을 버전 제어 시스템에서 체크인 할 수 있으므로 변경 기록이 지속적으로 유지된다는 장점이 있습니다 이전에 사용한 정상적인 구성으로 되돌리려는 경우 해당 버전을 체크아웃하여 인프라의 적용할 수 있습니다

또한 자동화 도구를 사용해서 DevOps 패턴을 구현하는데 도움이 되는 기반을 구축할 수 있습니다 개발자는 자동화 언어로 원하는 구성을 정의할 수가 있고, 운영자는 이러한 변경 사항을 더욱 쉽게 검토해서 피드백을 제공하고 해당 자동화를 사용하여 시스템이 개발자가 기대하는 상태에 있음을 재현해서 확인할 수 있습니다

작업 자동화 및 코드로 인프라를 관리하는 방법을 사용해서 서버에서 수동으로 작업 되는 수를 줄이면 일관성을 유지할 수 있습니다. 궁극적으로 자동화를 통해 it 인프라의 변화를 이끌어 **인적 오류**를 완화할 수 있습니다.

최근에 json → yaml 사람이 읽을 수 있는 코드 제공

Infrastructure As Code →IAC

XaC → JacC

GitOps

kubectl → api-server→etcd→Controller Manager → api-server→kubelet

kubectl 명령어 치면 api-server를 통해 etcd에 저장 컨트롤러 매니저가 상태 감시 이상 상태 발견시 pod 재생성 명령 api-server를 통해 kubelet에 전달

Ansible 아키텍처

앤서블을 구성하는 노드는 제어 노드와 관리 호스트라는 두 가지 유형으로 구성됩니다 앤서블 패키지는 제어 노드에 설치를 하고 저장되어 있는 코드를 실행합니다. 관리 호스트는 인벤토리라고 부르는 파일에 나열되어 있으며 시스템을 그룹으로 구성해서 보다 쉽게 일괄 관리할 수 있습니다. 인벤토리는 텍스트 파일에 정적으로 정의를 하거나 스크립트를 실행해서 외부 소스에서 그룹 및 호스트 정보를 가져와 동적으로 정의할 수 있습니다

관리 호스트 목록은 인벤토리라고 부르는 파일로 정의 하고 실제 수행해야 하는 작업 목록은 플레이북이라는 파일에 정의 할 수 있습니다. 그래서 어디에 어떤 작업을 수행 할 지를 파일로 관리합니다. 플레이북 파일에는 다양한 작업을 정의한 플레이를 작성하는데 쉽게 코드를 작성하고 읽기 쉽도록 YAML 형식을 사용합니다.

하나 이상의 플레이를 포함하는 파일을 플레이북이라고 하며 플레이 내부에는 여러 작업들을 정의 합니다. 그리고 각 작업은 특정 인수와 함께 작은 코드 조각인 모듈을 실행합니다(Python, PowerShell 또는 일부 기타 언어로 작성됨). 본질적으로 각 모듈은 툴킷의 툴입니다. Ansible은 다양한 자동화 작업을 수행할 수 있는 **수백 개의 유용한 모듈**과 함께 제공됩니다. 시스템 파일에 대한 작업, 소프트웨어 설치 또는 API 호출을 수행할 수 있습니다. 작업에 사용하는 경우 모듈은 일반적으로 시스템의 특정 측면이 특정 상태에 있는지 확인합니다. 예를 들어 특정 모듈을 사용하는 작업에서 파일이 존재하고 특정 권한과 콘텐츠가 있는지 확인할 수 있습니다. 다른 모듈을 사용하는 작업에서는 특정 파일 시스템이 마운트되어 있는지 확인할 수 있습니다. 시스템이 해당 상태에 있지 않으면 작업에서 해당 상태로 만들거나 아무 작업도 수행하지 않습니다.

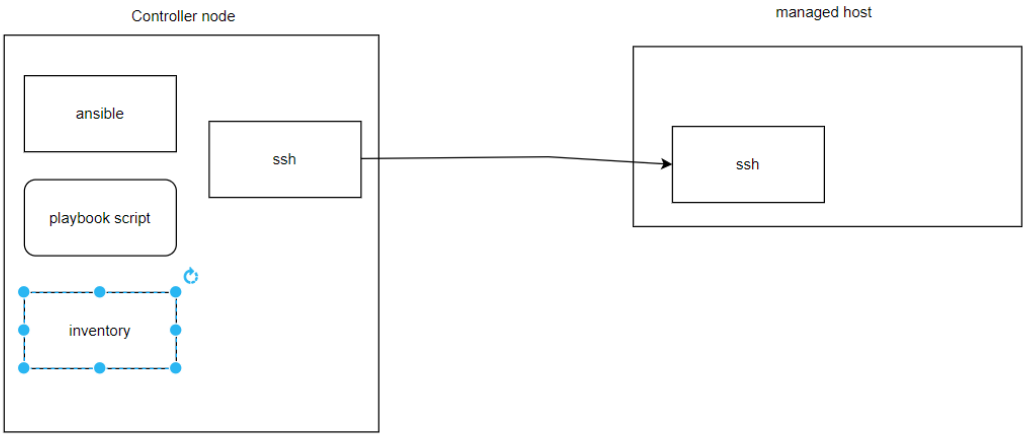
작업이 실패하는 경우 Ansible의 기본 동작은 실패한 호스트에 대해 플레이북의 나머지 부분을 중단하고 나머지 호스트로 작업을 계속하는 것입니다. 작업, 플레이 및 플레이북은 멱등(Idempotence)이 되도록 설계되어 있습니다. 즉, 동일한 호스트에서 플레이북을 안전하게 반복 실행할 수 있습니다. 시스템이 올바른 상태에 있을 때 플레이북을 실행하면 어떠한 변경사항도 발생하지 않습니다.

하지만 작업 자체가 실패가 아닌 잘못되었다면 브레이킹을 해서 빠져나온다. 더 수행 x

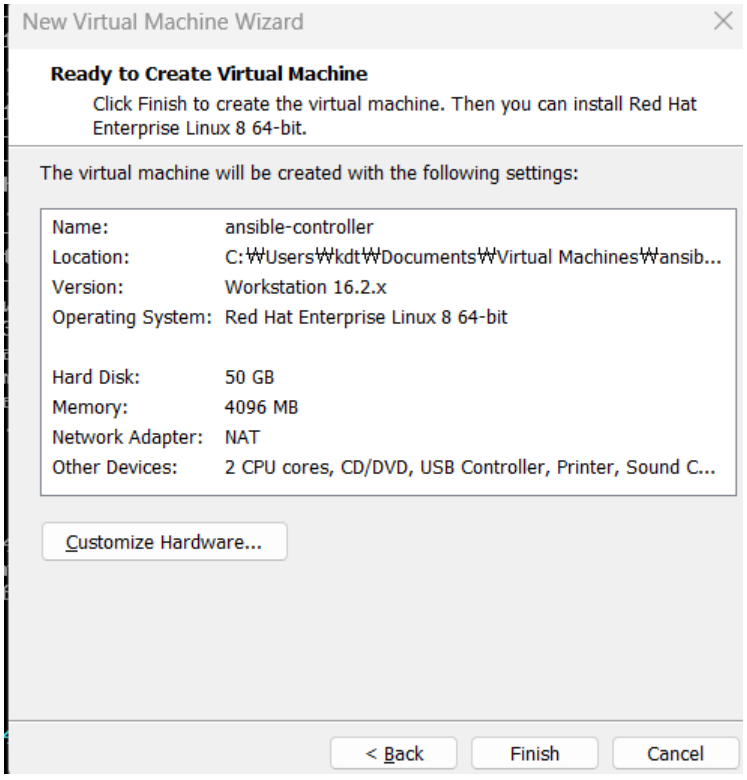
멱등성 : 여러번 시도를 해도 똑같은 성질을 얻음 상태 변화 x

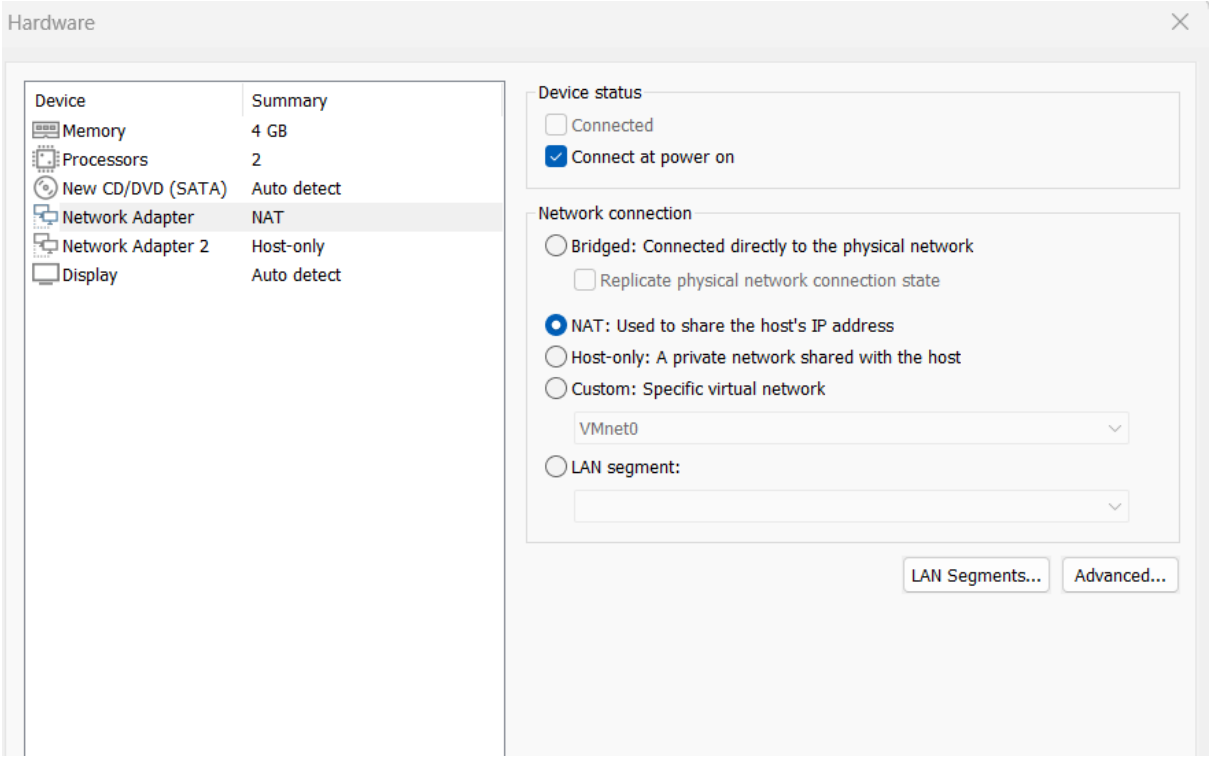
ansible은 대부분이 멱등으로 되어있다.

ex) yaml 파일로 pod 생성 반복할 때 똑같은 개수만 있는 것


Controller node
managed host
ansible
ssh
playbook script
inventory
ssh

실습 환경 구축


New Virtual Machine Wizard
Ready to Create Virtual Machine
Click Finish to create the virtual machine. Then you can install Red Hat Enterprise Linux 8 64-bit.
The virtual machine will be created with the following settings:
Name: ansible-controller
Location: C:₩Users₩kdt₩Documents₩Virtual Machines₩ansib...
Version: Workstation 16.2.x
Operating System: Red Hat Enterprise Linux 8 64-bit
Hard Disk: 50 GB
Memory: 4096 MB
Network Adapter: NAT
Other Devices: 2 CPU cores, CD/DVD, USB Controller, Printer, Sound C...
Customize Hardware...
< Back
Finish
Cancel



core : 2

memory : 4096

disk : 50gb

network : nat + host-only(local 내에 격리)

External / Internal Network

host only subnet ip 172.16.0.0 으로 할당

iso image는 rocky 8.10

root password 간단하게 root로

```
tcp6       0      0 :::22                   :::*                    LISTEN      938/sshd
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:da:9f:fc brd ff:ff:ff:ff:ff:ff
    altname enp3s0
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:da:9f:06 brd ff:ff:ff:ff:ff:ff
    altname enp11s0
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# _
```

- lo

- ens160

  nat

- ens190

  host-only

- 사용자가 수동으로 ip 할당

```
[root@localhost ~]# nmcli con mod ens192 \
> ipv4.address 172.160.0.200/24 \
> ipv4.method manual
[root@localhost ~]# ip a
```

```
[root@localhost ~]# nmcli con up ens160
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/1)
[root@localhost ~]#
```

- ip 할당 확인

```
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:da:9f:fc brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 192.168.150.131/24 brd 192.168.150.255 scope global dynamic noprefixroute ens160
       valid_lft 1781sec preferred_lft 1781sec
    inet6 fe80::20c:29ff:feda:9ffc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:da:9f:06 brd ff:ff:ff:ff:ff:ff
    altname enp11s0
    inet 172.160.0.200/24 brd 172.160.0.255 scope global noprefixroute ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feda:9f06/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

💡 160으로 잘못 설정하여 16으로 수정 후 다시 진행하였음

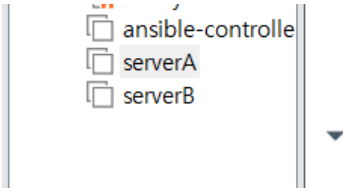
Ansible 설치

```
yum install epel-release
yum install ansible
ansible --version
```

```
[root@localhost ~]# ansible --version
ansible [core 2.16.3]
  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/python3.12/site-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Jul  2 2024, 20:57:30) [GCC 8.5.0 20210514 (Red Hat 8.5.0-22)] (/usr/bin/python3.12)
  jinja version = 3.1.2
  libyaml = True
[root@localhost ~]# ansible -m ping localhost
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

1. 네트워크 연결
2. ssh 키 전달
3. 인벤토리 설정

서버 복제



복제할 경우 unique한 값들이 겹칠 경우 충돌 발생

- serverA

  ip 172.16.0.201 할당
- serverB

  ip 172.16.0.202 할당

네트워크 연결

```
Last login: Tue Jul 30 01:33:26 2024
[root@localhost ~]# ping 172.16.0.201
PING 172.16.0.201 (172.16.0.201) 56(84) bytes of data.
64 bytes from 172.16.0.201: icmp_seq=1 ttl=64 time=2.29 ms
64 bytes from 172.16.0.201: icmp_seq=2 ttl=64 time=1.59 ms
^C
--- 172.16.0.201 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.590/1.938/2.286/0.348 ms
[root@localhost ~]# ping 172.16.0.202
PING 172.16.0.202 (172.16.0.202) 56(84) bytes of data.
64 bytes from 172.16.0.202: icmp_seq=1 ttl=64 time=1.50 ms
64 bytes from 172.16.0.202: icmp_seq=2 ttl=64 time=1.98 ms
64 bytes from 172.16.0.202: icmp_seq=3 ttl=64 time=0.872 ms
^C
```

ping 보냈을 시 연결된 것을 확인 가능

도메인 네임 서비스 우선순위

다음과 같은 순서로 찾아간다.

1. cache
2. /etc/hosts
3. /etc/resolve.conf

ip가 아닌 dns로 접근하기 위해 다음과 같이 파일 수정

- /etc/hosts 수정

```
[root@controller ~]# vi /etc/hosts
[root@controller ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6


172.16.0.201 servera
172.16.0.202 serverb
```

- ansible 결과 확인

```
[root@controller ~]# ansible -i inventory -m ping servera
The authenticity of host 'servera (172.16.0.201)' can't be established.
ECDSA key fingerprint is SHA256:inST6CLyHs8i9/pbY/DPP4E0d3zkxSXTqDin39KVh3U.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
servera | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

```
[root@controller ~]# ansible -i inventory -m ping serverb
The authenticity of host 'serverb (172.16.0.202)' can't be established.
ECDSA key fingerprint is SHA256:inST6CLyHs8i9/pbY/DPP4E0d3zkxSXTqDin39KVh3U.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
serverb | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

ssh 키 전달

servera, serverb 접근 시 비밀번호 활용이 아닌 ssh 키로 접근을 하여 자동화를 구축한다.

- servera,serverb에 ansible-user 생성

```
[root@controller ~]# useradd ansible-user
[root@controller ~]# ssh servera 'useradd ansible-user'
[root@controller ~]# ssh serverb 'useradd ansible-user'
```

- sudo 명령 가능하게끔 파일 수정

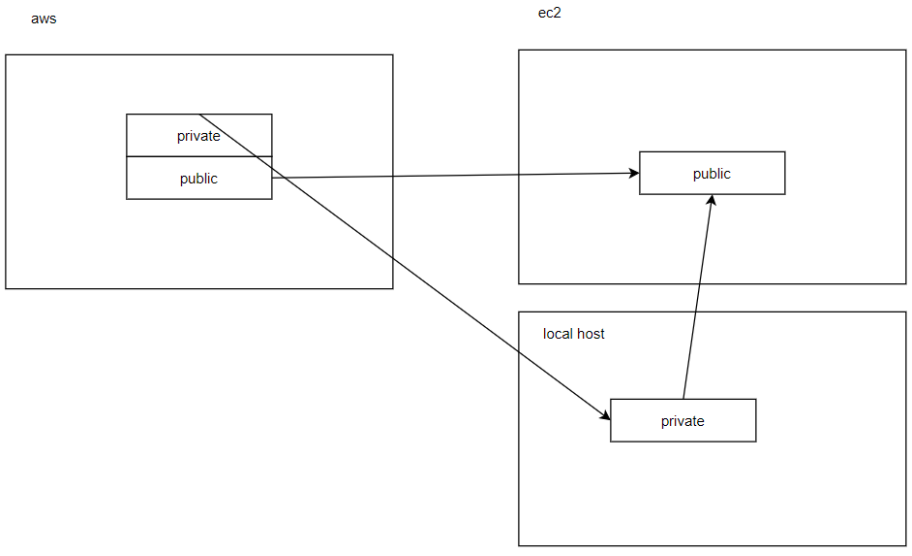
```
[root@controller ~]# vi /etc/sudoers
```

```
##
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
ansible-user    ALL=(ALL)       ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL

## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
```

- ansible이 sudo 권한 사용 가능한지 확인

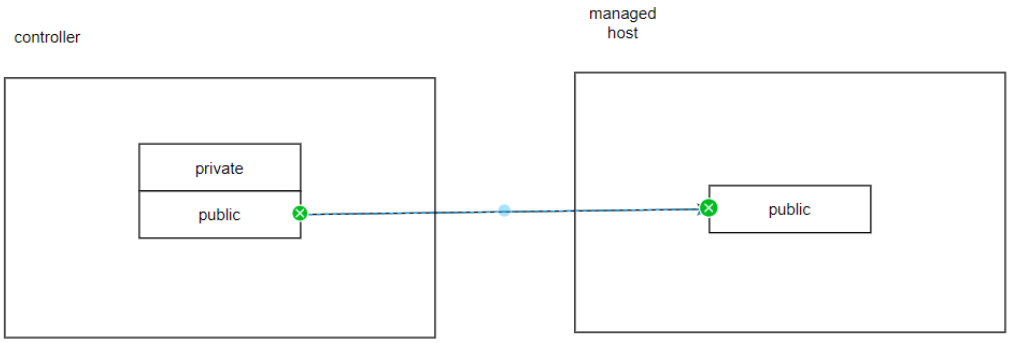
```
[root@controller ~]# su - ansible-user
마지막 로그인 : 화  7월  30 02:45:12 EDT 2024 일시 pts/0
[ansible-user@controller ~]$ sudo yum install httpd

로컬 시스템 관리자에게 일반적인 지침를 받았으리라 믿습니다.
보통 세가지로 요약합니다:

    #1) 타인의 사생활을 존중하십시오.
    #2) 입력하기 전에 한 번 더 생각하십시오.
    #3) 막강한 힘에는 상당한 책임이 뒤따릅니다.

[sudo] ansible-user의 암호 :
마지막 메타자료 만료확인 (0:12:13 이전 ): 2024년  07월  30일  (화 ) 오전  02시  34분  29초 .
종속성이 해결되었습니다.
================================================================================================================================
 꾸러미                          구조                버전                                          저장소
================================================================================================================================
설치 중 :
 httpd                          x86_64              2.4.37-65.module+el8.10.0+1840+b070a976.1     appstream
종속 꾸러미 설치 중 :
 apr                            x86_64              1.6.3-12.el8                                  appstream
 apr-util                       x86_64              1.6.1-9.el8                                   appstream
 httpd-filesystem               noarch              2.4.37-65.module+el8.10.0+1840+b070a976.1     appstream
 httpd-tools                    x86_64              2.4.37-65.module+el8.10.0+1840+b070a976.1     appstream
 mod_http2                      x86_64              1.15.7-10.module+el8.10.0+1830+22f0c9e0       appstream
 rocky-logos-httpd              noarch              86.3-1.el8                                    baseos
취약한 종속 꾸러미 설치 중 :
 apr-util-bdb                   x86_64              1.6.1-9.el8                                   appstream
 apr-util-openssl               x86_64              1.6.1-9.el8                                   appstream
```

- aws 키 활용 구조



aws에선 pem key 만들고 local에서 비밀키로 공개키를 가지고 있는 ec2에 접근한다.

- ansible 키 구조



ansible에선 내 공개키로 ssh를 통해 server a, b로 접근 server a b도 공개키 가지고 있다.

이렇게 하면 비밀번호 없이 자동화 가능

2. 플레이북 실행

Ansible을 사용하기 위해서 작성된 코드를 가진 파일이 필요합니다. 크게 3가지 유형의 파일이 있습니다. 다음은 유형 별 파일의 종류 입니다.

- 구성파일
- 인벤토리 파일
- 플레이북 파일

사용자는 세 가지 유형의 파일을 저장할 작업 디렉토리를 따로 생성하고 파일이나 혹은 하위 디렉토리로 위치시킵니다. 이제부터 각 파일에 대한 기본적인 내용을 알아보겠습니다.

구성파일

루트로 ssh 접근하는건 보안적으로 취약하다.

에스컬레이션: 일반 사용자가 루트로 권한 상승

사용자는 Ansible을 사용하기 전에 연결 설정이나 권한 설정 등을 변경할 수 있습니다. Ansible의 동작방법을 설정하는 파일을 구성 파일이라고 부르며 이 파일은 기본적으로 ***/etc/ansible/ansible.cfg*** 에 위치해 있습니다. 하지만 보통 일반적

인 경우 사용자가 프로젝트 별로 관리하기 위한 작업 디렉토리를 따로 생성하여 구성 파일을 위치 시켜 관리합니다.

Ansible 구성 파일은 각 섹션에 키-값 쌍으로 정의된 설정이 포함된 여러 개의 섹션으로 구성되며 섹션 제목은 대괄호로 묶여 있습니다. 다음은 가장 일반적으로 관리하는 섹션과 내용입니다.

```
[defaults]
inventory = ./inventory
remote_user = user # 이 계정으로 ssh 접근할 수 있게끔 통일 관리하기 편하게 ansible용 유저
ask_pass = false # 완전한 자동화를 위해 password 물어보지 않게끔

[privilege_escalation]
become = true
become_method = sudo # sudo로 전환
become_user = root # 루트 계정으로 상승 ansible 대부분의 작업이 루트 권한이 필요
become_ask_pass = false
```

- 구성파일 작성 및 반영 확인

```
[ansible-user@controller ansible]$ cat ansible.cfg
[defaults]
inventory = ./inventory
remote_user = ansible-user
ask_pass = false

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

[ansible-user@controller ansible]$ ansible --version
ansible [core 2.16.3]
  config file = /home/ansible-user/ansible/ansible.cfg
  configured module search path = ['/home/ansible-user/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.12/site-packages/ansible
  ansible collection location = /home/ansible-user/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Jul  2 2024, 20:57:30) [GCC 8.5.0 20210514 (Red Hat 8.5.0-22)] (/usr/bin/python3.12)
  jinja version = 3.1.2
  libyaml = True
[ansible-user@controller ansible]$ exit
logout
[root@controller ~]# ansible --version
ansible [core 2.16.3]
  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/python3.12/site-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Jul  2 2024, 20:57:30) [GCC 8.5.0 20210514 (Red Hat 8.5.0-22)] (/usr/bin/python3.12)
  jinja version = 3.1.2
  libyaml = True
```

ansible-user의 구성 파일과 root의 구성 파일의 위치가 다름을 확인 가능 즉 변경이 됐다.

💡 ansible의 대부분의 작업은 root 권한이 필요한데 root로 ssh 접근하는 것은 보안 취약 따라서 ansible 용 유저로 접근한 뒤 권한 부여

다음은 각 행에 대한 설명입니다.

- [defaults] : 섹션이름. 기본적인 내용을 설정
  - inventory : 인벤토리 파일의 경로를 지정
  - remote_user : 관리 호스트에 연결할 때 사용할 사용자 이름 지정. 지정하지 않았을 경우 현재 사용자의 이름이 지정
  - ask_pass : 관리호스트에 ssh 연결할 때 암호를 묻는 메세지 표시 여부를 지정.
- [privilege_escalation] : 권한 상승을 위한 사용자 전환 설정
  - become : 연결 후 관리 호스트에서 자동으로 사용자를 전환할지의 여부를 지정합니다 (일반적으로 root 로 전환).
  - become_method : 사용자 전환 방식을 지정. (일반적으로 기본값 sudo 를 사용, su 는 옵션).
  - become_user : 관리 호스트에서 전환할 사용자를 지정(일반적으로 기본값인 root).
  - become_ask_pass : become_method 매개 변수에 대한 암호를 묻는 메시지 표시여부를 지정.

사용자는 구성파일에 설정된 내용을 확인할 수 있도록 ***ansible-config view*** 명령어를 지원합니다.

연결 설정

Ansible은 제어 노드에서 관리 호스트로 연결하기 위해 ssh를 설정합니다. 이것은 defaults 섹션에 정의 할 수 있습니다.

ssh 로 연결할 때 사용자명을 별도로 구성되어 있지 않으면 Ansible 명령을 실행하는 로컬 사용자와 같은 사용자 이름을 사용하여 관리 호스트에 연결합니다.

다른 원격 사용자를 지정하려면 remote_user 매개 변수를 해당 사용자 이름으로 설정합니다. Ansible을 실행하는 로컬 사용자에게 개인 SSH 키가 있거나 관리 호스트에서 원격 사용자로 인증할 수 있도록 하는 키가 구성된 경우 Ansible은 자동으로 로그인됩니다.

인증을 위한 키를 구성하기 위해서 ***ssh-keygen*** 명령어를 실행하여 키페어를 생성할 수 있습니다. 또한 키페어 중 공개키를 배포 하기 위해 ***ssh-copy-id*** 명령어를 지원합니다. 제어노드에서 다음처럼 명령어를 입력하여 키를 생성하고 배포합니다.

```
ssh-keygen -N '' -f ~/.ssh/id_rsa
ssh-copy-id web1.test.com
```


passwd 인증 자체를 점차 줄여나가는 실정 보안에 취약하기에 키 기반으로 해야 한다.

권한 상승

보안 및 감사로 인해 Ansible을 원격 호스트에 권한이 없는 일반 사용자로 연결한 후 권한을 에스컬레이션하여 관리 액세스 권한을 가진 root 사용자로 전환할 수 있습니다. 이것은 Ansible 구성 파일의***[privilege_escalation]*** 섹션에 설정할 수 있습니다.

기본적으로 권한 에스컬레이션을 활성화하려면 구성 파일에 ***become = true*** 매개 변수를 설정합니다. 권한이 기본적으로 설정되어 있더라도 나중에 다양한 방법으로 이를 재정의할 수 있습니다. ***become_method*** 매개 변수는 권한을 에스컬레이션하는 방법을 지정합니다. 기본값은 sudo 를 사용하며 ***become_user*** 매개 변수를 통해 어떤 사용자로 에스컬레이션

할지 지정할 수 있습니다. 기본값은 ***root*** 입니다. 선택한 ***become_method*** 메커니즘에서 권한을 에스컬레이션하기 위해 사용자가 암호를 입력해야 하는 경우, 구성 파일에 ***become_ask_pass = true*** 매개 변수를 설정하면 됩니다.

관리 호스트에는 ssh로 연결한 사용자에게 sudo를 사용할 수 있는 권한이 설정 되어 있어야 합니다. /etc/sudoers 파일에 작성하거나 혹은 /etc/sudoers.d/디렉토리에 추가 파일을 생성하여 설정할 수 있습니다. 다음은 sudo 권한을 설정하는 내용입니다

```
ansible_user ALL=(ALL) NOPASSWD:ALL
```

인벤토리 파일

인벤토리 파일에는 앤서블에서 관리할 호스트 목록을 정의합니다. 특정 호스트는 그룹에도 할당하여 집합적으로 관리할 수도 있습니다. 그룹은 하위 그룹을 포함할 수 있으며, 호스트는 여러 그룹의 멤버가 될 수 있습니다. 그리고 인벤토리는 호스트 및 그룹에 적용되는 변수를 설정할 수 있습니다. 변수에 대한 내용은 뒤에서 자세히 다뤄보겠습니다.

호스트 인벤토리는 두 가지 방법으로 정의할 수 있습니다. 텍스트 파일을 사용해서 정적 호스트 인벤토리를 정의할 수 있습니다. 그리고 외부 정보 프로바이더(동적)를 사용하여 필요에 따라 정의하려면 Ansible 플러그인을 사용하면 됩니다.

정적 인벤토리

정적 인벤토리 파일은 텍스트 파일이며 앤서블이 관리할 호스트를 목록으로 지정합니다. 이 파일은 일반적인 INI스타일 형식 또는 YAML을 포함한 다양한 형식을 사용하여 작성할 수 있습니다. 다음처럼 호스트 명 혹은 IP주소를 한줄에 입력합니다.

```
web1.test.com
web2.test.com
db1.test.com
db2.test.com
192.168.0.100
```

그리고 인벤토리 파일에는 호스트 그룹을 지정할 수도 있습니다. 호스트 그룹을 사용하면 여러 호스트들을 일괄적으로 처리할 수 있어서 훨씬 효과적입니다. 그룹은 대괄호[ ]를 사용하여 이름을 지정한 뒤 다음 행부터 한 줄씩 그룹의 구성원 호스트들을 나열할 수 있습니다. 다음은 webserver와 dbserver그룹 예시입니다.

그룹으로 지정되지 않을 호스트들은 모두 위에다 작성해야 한다. 띄어쓰기는 무시된다.

```
192.168.0.100

[web-servers]
web1.test.com
web2.test.com

[db-servers]
db1.test.com
db2.test.com
```

호스트는 여러 개의 그룹에 있을 수 있습니다. 실제로 호스트를 여러 그룹으로 구성하면 호스트의 역할, 실제 위치, 프로덕션 여부 등에 따라 다양한 방식으로 구성할 수 있으므로 이 방법을 사용하는 것이 좋습니다. 그러면 특성, 용도 또는 위치에

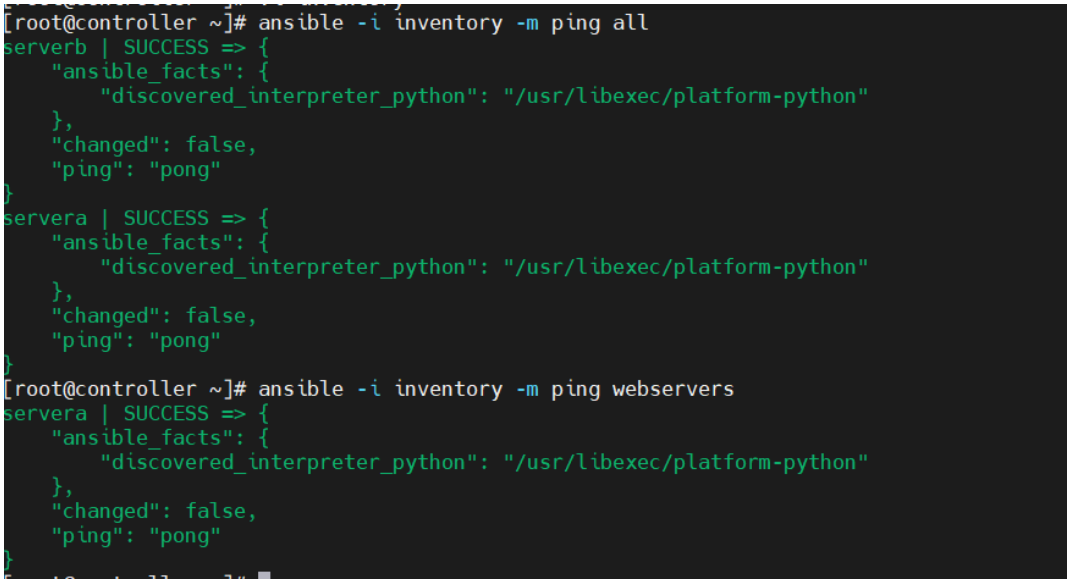
따라 특정 호스트 집합에 Ansible 플레이를 쉽게 적용할 수 있습니다. 또한 편의상 두 개의 호스트 그룹을 자동으로 구성하고 있습니다.

- all : 모든 호스트 목록을 포함하는 그룹
- ungrouped : 인벤토리에서 그룹에 속하지 않는 모든 호스트 목록

inventory 설정 및 all ungrouped 테스트

- inventory

```
serverb

[webservers]
servera
```



중첩 그룹

Ansible 호스트 인벤토리에 호스트 그룹을 여러 개 포함할 수 있습니다. 이 작업은 호스트 그룹 이름 생성 시:children 접미사를 추가하면 됩니다. 다음은 중첩 그룹의 예시입니다.

그룹을 포함하는 그룹

```
[dev]
web1.test.com
web2.test.com

[prod]
```

```
web3.test.com
web4.test.com

[web-servers:children]
dev
prod
```

범위 지정

호스트 이름 및 IP주소를 작성할 때 범위를 지정하여 인벤토리 파일을 간단히 작성 할 수 있습니다. 관리 호스트 개수가 많아 짐에 따라 행의 수가 늘어나 가독성이 떨어지는 상황을 방지 하고 좀 더 가독성을 높일 수 있습니다. 범위는 숫자와 영문에 해당하는 값을 처음 값과 마지막 값을 입력하여 표현합니다.

```
[처음값:마지막값]
```

다음은 범위를 표현하는 예시입니다.

- 192.168.[4:7].[0:255] → 192.168.4.0 ~ 192.168.7.255 범위의 모든 IP 주소
- server[01:20].example.com → server01.example.com 부터 server20.example.com 까지 모든 호스트
- [a:e].dns.example.com → a.dns.example.com 부터 e.dns.example.com 까지의 호스트

인벤토리 파일 위치 재정의

인벤토리 파일의 기본 위치는 ***/etc/ansible/hosts*** 입니다. 하지만 거의 대부분 작업 디렉토리를 따로 생성하여 플레이북 파일과 함께 위치시켜 관리합니다. ansible 명령어의 -i 인자는 인벤토리 파일의 기본위치가 아닌 사용자가 원하는 파일로 대체하는 옵션입니다.

```
ansible -i ./inventory
```

사용자는 앤서블 구성파일의 defaults 섹션에서 인벤토리 파일의 위치를 정의 할 수도 있습니다.

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

동적 인벤토리

벤더에서 제공을 해준다. 제공해주는 스크립트 파일을 사용하면 된다.

플레이북 파일

Ansible은 특정한 작업을 수행하기 위해 플레이북 파일에 작업 내용을 미리 정의 해 놓습니다. 이때 플레이북 파일에는 단일 혹은 여러 개의 플레이가 작성 될 수 있습니다. 단일 플레이에는 작업 대상이 되는 관리 호스트 정보와 작업 내용을 담고 있는 정보가 필요합니다. 작업(tasks)은 특정 작업을 수행하기 위해 모듈 목록을 구성한 필드 이며 여기에 작성된 내용은 순서대로 실행됩니다. 다음은 플레이북 파일의 플레이 예시입니다.

```
---
- name: test play
  hosts: web1.test.com
  tasks:
    - name: create new user
```

```
      user:
        name: user01
        uid: 4000
        state: present
```

- hosts:

  작업 대상

- tasks:

  작업 내용

- JSON 작성 예시

ex1 (key와 value)

{

“data1” :{

“data2” : “value2”

}

}

ex2 (value만)

{

“data1” : [

“value2”

]

}

json은 이렇게 작성하기 번거롭고 괄호 하나 놓치면 전체 오류 따라서 보통 yaml 사용

들여쓰기 띄어쓰기 굉장히 중요 레벨 수준을 잘 맞춰줘야 한다.

kubectl처럼 yaml 날리면 바로 만들어진다.

플레이북은 YAML 포맷으로 작성된 텍스트 파일이며, 일반적으로 확장명 .yml 을 사용하여 저장됩니다. 플레이북은 데이터의 구조를 표현하기 위해 공백 문자를 사용하여 들여쓰기합니다. YAML에서는 들여쓰기에 필요한 공백 수에 대해 두 가지 기본 규칙이 적용됩니다.

- 계층 구조상 동일한 수준의 데이터 요소는 공백이 동일해야 합니다.

- 하위 목록은 상위 목록 보다 들여 써야 합니다.

중요한 점은 탭 문자는 오류가 발생하기 때문에 사용하는 것을 권장하지 않고 스페이스바로 공백을 입력합니다. 또는 편집기의 yaml문법 지원 플러그인을 설치해서 탭키를 자동으로 두 칸의 공백으로 입력되도록 설정할 수 있습니다.

플레이북은 문서의 시작을 나타내는 **시작 마커인 세 개의 하이픈(---)** 기호로 줄을 시작하며 종료를 나타내는 **종료 마커인 세 개의 도트(...) 를 입력할 수 있지만 보통 생략**할 수 있습니다.

플레이북은 시작마커와 종료 마커 사이에 플레이들 목록을 구성합니다. 이때 목록(list)은 하나의 대시 기호로 공백으로 시작합니다.

```
---
- play1
- play2
- play3
...
```

플레이 내부에서는 키와 값 쌍으로 작성된 데이터들을 입력합니다. 동일한 수준의 세가지 키인 name, hosts, tasks 를 지정하고 각 항목 별로 값을 입력합니다. 다음은 예시에서 사용된 name 항목의 값입니다.

```
name: test play
```

name 키는 플레이의 목적 및 목표를 나타내는 정보이며 임의의 문자열을 입력할 수 있습니다. name 항목은 필수 항목은 아니지만 **가독성을 높여주기 때문에 사용하는 것을 권장**합니다.

다음은 실행할 호스트를 선택하기 위한 hosts 항목의 예시입니다.

```
hosts: web1.test.com

# web*를 사용해서 web이라는 이름을 가진 모든 목록 사용 가능
```

hosts에 입력된 값은 **인벤토리에 포함되어 있는 목록**을 선택할 수 있습니다. 이때 호스트나 그룹 혹은 호스트 패턴을 값으로 입력할 수 있습니다.

마지막으로 tasks 항목에서는 **실제로 수행할 작업 목록**을 구성합니다. 다음은 새로운 사용자를 생성하기 위한 tasks 예시입니다.

```
  tasks:
    - name: create new user
      user:
        name: user01
        uid: 4000
        state: present
```

tasks 하위에는 대시 기호를 입력하여 단일 항목을 지정하고 있는데 이때 name과 user라는 두 가지 키를 입력하고 있습니다. **name 항목은 마찬가지로 목적을 설명하는 정보이며 필수 항목은 아닙니다**. 그리고 user라는 항목은 모듈이라고 부르며 하위 수준으로 입력된 인자를 통해 원하는 작업 내용을 구성할 수 있습니다. 예시에서는 user라는 항목의 하위에 name과 uid 그리고 state 정보를 입력했습니다. **즉 위 작업은 uid가 4000인 user01 사용자가 현존하는지 확인해서 사용자가 없다면 새로 생성하고, 있다면 그대로 두는 작업**입니다.

모듈은 사용자가 원하는 작업 내용을 수행할 수 있도록 미리 설계 되어져 있으며 키워드만 입력하면 그 즉시 모듈을 실행할 수 있습니다. 또한 작업 목록에는 단일 모듈 혹은 여러 개의 모듈이 정의 될 수 있습니다. 다음은 여러 개의 모듈이 구성된 작업 예시입니다.

```
tasks:
  - name: web server start
    service:
      name: httpd
      state: started
  - name: ntp server start
    service:
      name: chronyd
      state: started
  - name: mail server start
    service:
      name: postfix
      state: started
```

<aside>
💡 Ansible 공식 페이지에는 다양한 모듈들의 정보가 있습니다. 다음은 링크는 공식 페이지의 모듈 인덱스 페이지입니다.

https://docs.ansible.com/ansible/2.8/modules/modules_by_category.html

</aside>

플레이북 실행

ansible-playbook는 제어노드에서 플레이북 파일의 내용을 읽어 플레이를 실행하는 명령어 입니다. 예를 들어 site.yaml 파일을 작성하고 이를 플레이북으로 실행하고자 한다면 다음처럼 입력할 수 있습니다.

```
# site.yaml
- name: sample playbook
  hosts: node01
  tasks:
  - name: sample task
    yum:
      name: httpd
      state: latest

 # 실제 환경 yaml
 - name: sample playbook
  hosts: servera
  tasks:
  - name: sample task
    yum:
      name: httpd
      state: latest
```

state 종류

- absent

  있으면 지워라

- present

  없으면 만들어라

- latest

  최신버전으로 만들어라

```
ansible-playbook site.yaml
```

- 에러 발생

```
## Allows people in group wheel to run all commands
wheel    ALL=(ALL)       ALL

## Same thing without a password
%wheel  ALL=(ALL)       NOPASSWD: ALL
```

ssh servera로 a 서버 접속 후 /etc/sudoers 아래 파일 이렇게 수정한다.

nopasswd 옵션을 활성화 한다.

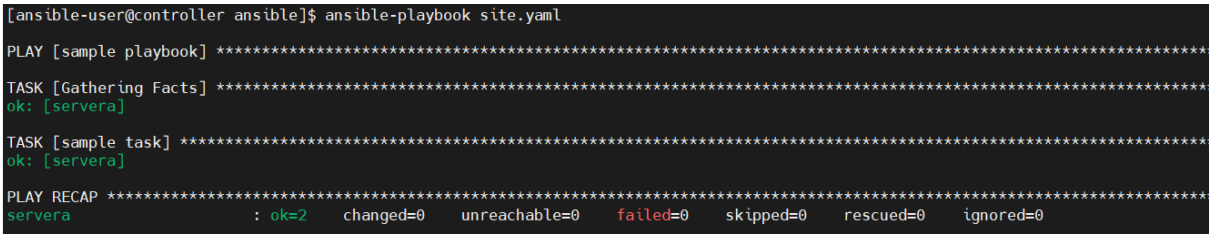
```
[ansible-user@controller ansible]$ ansible-playbook site.yaml

PLAY [sample playbook] *********************************************************************************************************************************

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

TASK [sample task] *************************************************************************************************************************************
changed: [servera]

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

플레이북을 실행하면 실행 중인 플레이와 작업을 표시하는 출력이 생성됩니다. 출력에는 또한 실행한 각 작업의 결과가 보고됩니다. 위 명령어의 결과는 다음과 같이 출력 됩니다.

```
PLAY [sample playbook] *********************************************************

TASK [Gathering Facts] *********************************************************
ok: [node01]

TASK [sample task] *************************************************************
changed: [node01]

PLAY RECAP *********************************************************************
node01 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=
0
```

일반적으로 Ansible 플레이북의 작업은 멱등이며, 플레이북을 여러 번 실행하는 것이 안전합니다. 관리 호스트가 이미 올바른 상태인 경우 변경되지 않습니다. 예를 들면, 이전 예제의 플레이북이 다시 실행하면 다음과 같은 내용이 출력됩니다.

```
PLAY [sample playbook] ***************************************************

TASK [Gathering Facts] ***************************************************
ok: [node01]

TASK [sample task] *******************************************************
ok: [node01]

PLAY RECAP ***************************************************************
node01   : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescue
d=0
```



두 개의 출력에서 다른 점은 처음 출력은 changed=1 로 보고 되었지만 두 번째 출력은 changed=0 으로 보고 되었습니다. 즉 두 번째 작업에서는 이미 구성 되었으니 따로 작업을 처리 하지 않고 그대로 두었다는 의미가 됩니다.

상세한 작업의 경과를 출력하기 위해서는 -v 옵션을 사용할 수 있습니다. v문자의 개수에 따라 수준을 조절할 수 있습니다.

- v : 간단한 작업 결과를 표시
- vv : 작업 결과와 구성 내용을 표시
- vvv : 관리 호스트에 연결하기 위한 추가 정보도 표시
- vvvv : 스크립트를 사용했다면 관련 내용도 함께 표시

앤서블은 플레이북을 실행하기 전에 ***--syntax-check*** 옵션으로 해당 구문의 유효성을 미리 검사하는 기능을 지원합니다. 구문에 문제가 있을 경우 오류를 보고 합니다. 또한 비슷한 기능으로 ***--check*** 옵션을 통해 실제 작업을 처리하지 않지만 작업에 대한 결과가 어떻게 보고되는지를 확인할 수 있는 기능을 지원합니다.

다중 플레이

단일 플레이북은 여러 개의 플레이를 정의 할 수 있습니다. 플레이는 순서가 지정된 작업 목록과 작업 대상이 지정되어 있습니다. 이때 다양한 작업 대상을 지정해서 각각의 대상에게 맞는 작업 목록들을 구성할 수 있습니다.

이러한 기능은 여러 호스트에서 다양한 작업을 수행해야 하는 복잡한 배포를 오케스트레이션할 때 매우 유용합니다. 하나의 호스트 집합에 대해 하나의 플레이를 실행하고, 해당 플레이가 완료되면 다른 호스트 집합에 대해 다른 플레이를 실행하는 플레이북을 작성할 수 있습니다. 다음은 DB서버와 WEB 서버의 서비스를 함께 구성하는 플레이북 예시입니다.

```
- name: first play
  hosts: dbservers
  tasks:
    - name: first task
      service:
```

```
        name: mariadb
        enabled: true
- name: secodn play
  hosts: webservers
  tasks:
    - name: first task
      service:
        name: httpd
        enabled: true
```

플레이에서 설정할 수 있는 항목은 hosts 및 tasks 이외에도 서버에 연결할 사용자를 재구성하거나 권한 상승을 조정할 수 있습니다. 다음은 플레이에 설정하는 원격 사용자 설정과 권한 상승의 예시입니다.

```
- name: sample play
  hosts: demo-host
  remote_user: ansible-user
  become: true
  tasks:
    - name: sample task
      service:
        name: httpd
        enabled: true
```

YAML 문법

yaml은 사람이 읽고 쓰기 쉽도록 개발되었습니다. 처음 사용하는 사용자도 불편함 없이 금방 익혀서 사용할 수 있도록 문법적인 제약이 강하지 않지만 몇 가지 규칙을 알고 있어야 합니다. 먼저 yaml에서 주석을 사용할 수 있습니다. 주석은 해시 기호(#) 를 제일 왼쪽에 입력합니다. 중간에 해시 기호가 나올 경우 해시 앞에 공백을 두어야 합니다.

```
# this is comment
data # also this is comment
```

yaml은 dictionary 와 list 데이터 형식을 사용합니다. 키-값 페어 데이터는 dictionary 데이터 형으로 인라인 형식의 중괄호를 사용합니다.

```
name: testservice
serviceport: 80

{name: testservice, serviceport: 80}
```

대시 기호를 사용한 list 데이터는 대괄호를 사용하여 인라인 형식으로 입력할 수 있습니다.

```
hosts:
  - servera
  - serverb
  - serverc

hosts: [servera, serverb, serverc]
```

또한 플레이북 내에서 속기 형식을 입력할 수 있습니다. 동일한 수준의 데이터를 같은 행에 입력하는 방식이라서 들여쓰기가 많을수록 번거로운 작업을 피할 수가 있지만, 익숙하지 않은 사용자는 가독성이 떨어질 수 있습니다.

```
tasks:
  - name: httpd service start
    service: name:httpd enabled=true state=started
```