Ansible

https://byungwoo.oopy.io/c4af6fd2-2dfb-4bb6-909e-1a3803296100

https://heavyrain180828.tistory.com/25

https://watch-n-learn.tistory.com/73

Ansible이란 무엇인가?

Ansible은 파이썬 기반으로 작성되었으며, module과 명령어를 사용하여 여러 서버에 동시에 명령을 실행할 수 있는 자동화 엔진이다. 또한 ansible playbook이라는 스크립트를 통해 IT 어플리케이션 인프라스트럭쳐를 정확하게 묘사할 수 있는 언어이다. Ansible의 주요 목적은 IT 장비를 관리를 자동화 하는 것이다. 원래 무료 오픈소스 프로젝트였으나 RedHat이 인수하였다. 그래서 RHEL에서는 서브스크립션을 구매해야 사용할 수 있따. 하지만 커뮤니티 버전도 있기 때문에 무료로 사용할 수도 있다. 큰 차이는 없으며 기술지원을 받을 수 있느냐 늬 차이이다.

Ansible Playbook이란?

Ansible play는 ansible을 사용한 한 줄의 명령이라고 볼 수 있다. 이러한 play를 모아서 스크립트화한 것이 ansible playbook이다. 즉, 하나 이상의 play를 포함하는 일련의 명령을 모은 스크립트 파일을 playbook 라고 부른다. 이러한 스크립트는 사람이 읽을 수 있는 형식으로 애플리케이션 환경의 모든 측면을 설명/기록 가능하다.
이 Ansible playbook은 Ansible의 문법으로 만들어지며, script form은 yaml을 사용한다. 따라서 읽기/쓰기가 쉽고 일반 텍스트이므로 소스코드처럼 취급하고 git 등의 버전 제어 프로그램을 사용하여 관리할 수 있다.

Ansible의 구성요소

1. 앤서블 본체 : 앤서블 자체
2. 인벤토리 : 관리 대상을 기록해둔 파일. 보통 IP나 hostname 등의 정보를 기록해 둔다.
    a. /etc/ansible/hosts (default), ansible production -m ping
    b. 서버, 네트워크 장비, 서버를 지역별로, 기능별로 나누는 등 대규모의 DC를 운영하는 클라우드 DC와 같은 경우에는 하나의 인벤토리에 모든 서버 정보를 기록하여 관리하느 것이 매우 어렵다. 이 경우 다수의 인벤토리 파일을 작성하여 관리한다.
    ex) zone_seoul.ls, nginx.ls
3. 모듈 : 앤서블에서 실행하는 하나하나의 명령

a. 명령 : yum -y install httpd ⇒ 모듈 : ansible all -m yum -a “name=httpd state=present”

4. Playbok :
    a. 모듈을 이용하여 명령을 전달할 때 두 가지 방법으로 전달이 가능하다.
        i. ad-hoc : ansible all -m yum -a “name=httpd state=present”
        ii. playbook : yum 파일을 고 필요한 모든 내용을 ynl파일에 추가한 뒤 일괄적으로 처리가 가능하다. ⇒ 실무에서는 ad-hoc 보다는 대부분 플레이북을 이용하는 경우가 많다.

CI / CD 환경에서 Jenkins 를 통한 ansible 의 간단한, 명령어 전달이 필요한 경우에는 ad -hoc를 이용하는 경우도 있다.

Ansible 사용이유와 목표

On-premise 환경의 리눅스 서버뿐만 아니라 윈돌우 서버까지 포함하여 명령어를 사용하는 모든 작업을 처리할 수 있다. AWS, Azure, GCP와 같은 클라우드 서버를 대상으로 작업도 가능하다.

Ansible은 시스템을 원하는 상태로 표현하여 필요한 변경만 수행하여 시스템을 원하는 상태로 만드는 것이 Ansible의 목표이다. 각 task에서는 특정 항목이 “특정 상태”에 있는지 확인한다. 예를 들어, 그 상태라고 하는 것은 특정 파일이 있거나, 마운트 되어있거나, 서비스가 중단되어 있는 등의 상태이다. ansible은 이렇게 상태를 확인한 후 시스템이 그 상태가 아니면, task에서 해당 상태로 만들도록 실행한다. 만약 시스템이 이미 해당 상태라면, 아무것도 수행하지 않는다. task가 실패하는 경우 default 설정으로, 실패가 발생한 호스트에 대한 나머지 플레이북은 중단된다. (절차적 언어)

Ansible의 특징 및 장점

1. Agentless
    - 클라이언트를 별도로 실행할 필요없이 SSH 연결이 가능하다면 바로 사용이 가능하다. SSH를 사용한다는 것부터가 많은 검증이 된 오픈소스이므로 애플리케이션으로 인한 보안적인 측면에서도 크게 걱정할 필요가 없다. SSH를 사용함에 따라 Agent 기반의 다른 자동화 오픈소스와 대비하여 속도가 조금 느리다는 단점이 있다.

2. 멱등성

- 같은 작업을 몇번이고 수행하더라도 같은 결과가 얻어지는 성격을 가집니다. 예를 들어 데몬(서비스)을 실행하는 작업을 한 번 수행한 이후에 한 번 더 데몬(서비스)를 실행하는 작업을 수행하게 되면 이미 서비스가 실행되어 있는 상태이기 때문에 실제로 작업을 수행하지 않습니다.

3. 쉬운 문법 (Yaml)
   - 작업 명세서(Ansible-Playbook)을 YAML 파일로 기술하여 가독성이 높습니다.

4. 다양한 플랫폼 지원
   - ansible은 물리, 가상, 클라우드, 컨테이너 환경 모두를 지원하며, OS도 Linux, Unix,Windows 모두 지원한다. 또한 네트워크장치 등 특정한 하드웨어도 지원한다.

Ansible의 사용 예시

구성 관리 집중화	구성파일 관리, 배포 중앙집중화, 애플리케이션 라이프사이클을 중앙에서 관리할 수 있다.
워크플로우 자동화	업무를 관리하고 사용하는데 있어 자동화할 수 있다.
네트워크 자동화	네트워크 설정 작업을 자동화할 수 있다.
애플리케이션 배포	제어 노드에서 하나의 명령으로 호스트 모두에게 한번에 프로그램이나 파일을 전달할 수 있다.
프로비저닝	PXE booting + kickstart + Ansible로 간단하게 대규모 OS설치, 환경구축 작업을 수행할 수 있다.
효과적인 협업 지원	한 코드를 여럿이서 작업하는 개발 등을 효과적으로 할 수 있게 할 수 있다.
애플리케이션 라이프사이클 관리	전체 에플리케이션 라이프사이클을 오케스트레이션할 수 있다.

사용경험

실습환경

- 3대의 CentOS 서버 : 키페어는 없음. 패스워드 인증만 제공한다.
- ansible 의 실체는 ssh 를 이용함 ⇒ ssh -l root 목적지 IP ‘원격지에서 필요한 명령어’
- 패스워드 묻는 것을 없애고 싶다면?
    - key-pair 생성 후, ansible 은 private key, 원격 서버들은 public key를 갖고도록 해 주어야 한다.
    - public key는 원격지의 특정 사용자 홈 디렉토리 아래의 ~/.ssh/authorized_keys 에 등록되어 있어야 한다.
    - 초기에 원격 서버들에 접속하면 원격 서버의 public key 를 저장하도록하므로 일시 중지 상태가 된다. 이를 해결하기 위해서는 사전에 원격 서버들의 public key를 ansible을 운영하는 서버의 특정 계정 .ssh.known_hosts 에 등록해두어야 한다.

가상머신 3개 (centos1, centos2, centos3) 생성

```
#centos1 생성
virt-install --name centos1 --ram 1024 --disk path=./centos
1.qcow2 --vcpus 1 --graphics none --serial pty --console pt
y --import &

#centos2 생성
virt-install --name centos2 --ram 1024 --disk path=./centos
2.qcow2 --vcpus 1 --graphics none --serial pty --console pt
y --import &

#centos3 생성
```

```
virt-install --name centos3 --ram 1024 --disk path=./centos
3.qcow2 --vcpus 1 --graphics none --serial pty --console pt
y --import &

#가상머신 자동시작
virsh autostart centos1
virsh autostart centos2
virsh autostart centos3

#가상머신 IP주소
#centos1 -> 192.168.122.240
#centos2 -> 192.168.122.77
#centos3 -> 192.168.122.176
```

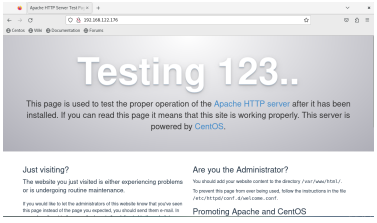
- vi /etc/ansible/hosts

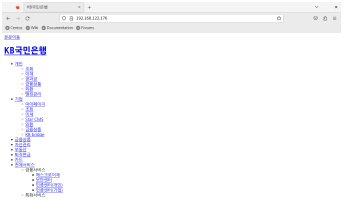
- http 실행 && 방화벽 종료 && 접속이 되는지 확인

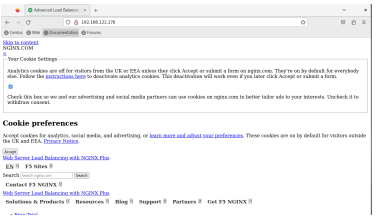
```
# httpd를 설치해야함

#httpd 실행
ansible all -m service -a "name=httpd state=started" -k
#방화벽 종료
ansible all -m service -a "name=firewalld state=stopped" -k
#접속 확인
curl http://192.168.122.240
curl http://192.168.122.77
curl http://192.168.122.176
```

```
#페이지 내용 변경
#kbstar
ansible all -m shell -a 'curl -L http://www.kbstar.com > /v
ar/www/html/index.html' -k
#nginx
ansible all -m shell -a 'curl -L http://www.nginx.com > /va
r/www/html/index.html' -k
```







- root 밑 ansible 디렉토리, 그 밑에 인벤토리 파일 2개(seoul.lst, jeju.lst 추가)

```
mkdir ansible ; cd ansible ; touch seoul.lst ; touch hehu.l
st

#아래 그림은 seoul.lst 내용
```


```
ansible swarm-worker -i seoul.lst -m ping -k
#SSH password: (test123)
```

Quiz. 현재 우리는 ad-hoc 방식으로 원격지 서버에 모듈을 이용한 패키지 설치 등을 진행하고 있다. 하지만 매번 패스워드를 입력해 주어야 한다. 이 경우 gitlab-runner, jenkins 등의 ci / cd 도구는 패스워드 입력을 할 수 없으므로 파이프라인이 정상적으로 동작하지 않게 된

다. 이를 해결하기 위해서는 -k 옵션 없이 실행될 수 있어야 한다. 이는 키 페어를 이용하여 해결 가능하다.

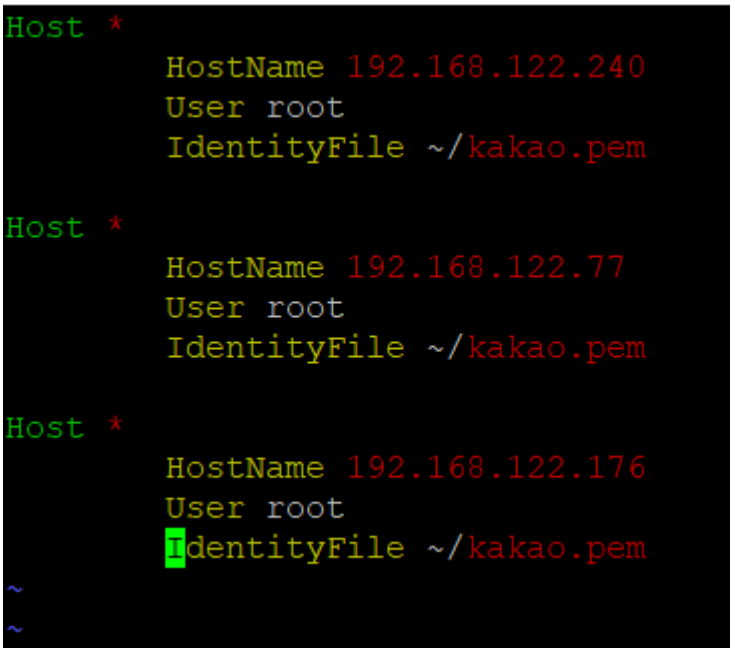
copy를 이용하여 로컬에서 키페어를 만든 뒤, 퍼블릭 키를 원격지에 넣고 아래의 명령 실행이 가능하도록 해보세요.

```
ssh-keygen -q -N "" -f kakao.pem

ansible all -m shell -a “mkdir .ssh ; chmod 700 .ssh ; touch ~/.ssh/authorized_keys ; chmod 644 ~/.ssh/authorized_keys” -k

ansible all -m copy -a "src=kakao.pem.pub dest=~/.ssh/authorized_keys" -k

vi .ssh/config
```



ansible

```
PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys
# but this is overridden so installations will only check .ssh/authorized_ke
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication yes
```

centos?

```
PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorize
# but this is overridden so installations will only check .ssh/authori
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no
```