|  |
| --- |
| 앤서블(Ansible)  기본 매뉴얼  **Version 1.0** |
| 쎄븐정보기술  **최 문환**  CHOI MOONWHAN  moonhwan.choi@seven.co.kr |



|  |  |
| --- | --- |
| **AUTHORIZATION** | 본 매뉴얼은 저작권을 가지므로 무단복제 및 전제를 금하며, 본 내용과 프로그램의 기능 혹은 성능 간에 서로 일치하지 않는 부분이 생길 수 있음을 이해해주시기 바랍니다.  본 매뉴얼에 기술된 내용에 대해 명시적 혹은 묵시적인 보증은 제공하지 않음을 양해하여 주시기 바랍니다 |
| **PREFACE** | 본 매뉴얼은 Ansible대한 정보를 담고 있으며, 운영체제를 다루는 시스템 관리자를 대상으로 하고, 이 시스템 관리자가 다음의 지식을 가지고 있다고 가정합니다. | |
| **necessary knowledge** | ■ 시스템 관리에 대한 기본적 이해  ■ 운영체제를 설치하고 운영하는 데에 필요한 기본적 지식  ■ 서버 및 네트워크 환경에 대한 정보  ■ 가상화 개념에 대한 기본적 이해 | |
|  |  | |

**제/개정 이력**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| 1.0 | 제 정 | 2022/10/05 |
| **개정번호** | **제/개정 페이지 및 내용** | **제/개정 일자** |

목차

[1. 앤서블 이란? 6](#_Toc116636636)

[1-1. 앤서블의 개요 6](#_Toc116636637)

[1-2. 앤서블의 역할 6](#_Toc116636638)

[2. 앤서블 환경 구축 8](#_Toc116636639)

[2-1. 가상머신 9](#_Toc116636640)

[2-2. CentOS 구성 10](#_Toc116636641)

[2-3. 앤서블 설치하기 13](#_Toc116636642)

[2-4. 앤서블 호스트파일 설정 14](#_Toc116636643)

[2-5. Vagrant를 이용한 앤서블 환경 구축 15](#_Toc116636644)

[2-6. Ansible-Server 추가 설정 프로비저닝 23](#_Toc116636645)

[2-7. 베이그런트로 앤서블 노드 추가하기 29](#_Toc116636646)

[2-8. 앤서블 서버와 노드의 통신 34](#_Toc116636647)

[3. 앤서블 기본내용 37](#_Toc116636648)

[3-1. 앤서블의 구성파일들 37](#_Toc116636649)

[3-2. 앤서블 명령어 실행 시 옵션 값 38](#_Toc116636650)

[3-3. 한번의 명령어로 다수의 시스템에 작업하기 39](#_Toc116636651)

[3-3. 앤서블 Play Book 41](#_Toc116636652)

[4. 앤서블 – CentOS 관리하기 45](#_Toc116636653)

[4-1. 앤서블 노드에 nginx설치 및 테스트 그리고 삭제하기 45](#_Toc116636654)

[4-2. Time Zone 변경하기 47](#_Toc116636655)

[4-4. NFS서버와 클라이언트 구성하기 48](#_Toc116636656)

[5. 앤서블 – Ubuntu OS 관리하기 50](#_Toc116636657)

[5-1. 기존 환경에 우분투 OS 추가하기 50](#_Toc116636658)

[5-2. Time Zone 수정하기 53](#_Toc116636659)

[5-3. 우분투에 nginx 추가하기 54](#_Toc116636660)

[6. 앤서블 – Windows 관리하기 (미완성) 55](#_Toc116636661)

[6-1. Vagrant를 이용해서 Windows 노드 추가하기 55](#_Toc116636662)

[7. 앤서블 알아 두면 좋은 팁 59](#_Toc116636663)

[7-1. Debug Module, Debugger 59](#_Toc116636664)

[8. 네트워크 실습을 위한 NX-OSv 랩 구성하기 61](#_Toc116636665)

[8-1. 수동으로 VirtualBox에 NXOSv올리기 62](#_Toc116636666)

[8-2. 앤서블 서버와 NXOSv서버의 연동 71](#_Toc116636667)

[8-3. 네트워크 시스템의 시간 동기화 & 구성파일 백업하기 73](#_Toc116636668)

## 1. 앤서블 이란?



### 1-1. 앤서블의 개요

Ansible (앤서블)은 여러 개의 서버를 효율적으로 관리할 수 있게 해주는 환경 구성 자동화 도구입니다.

2012년에 ‘마이클 데한’ 이라는 개발자가 만들어 소스코드를 공개한 오픈소스 소프트웨어입니다. 2015에 오픈소스 업계의 큰 손인**레드헷**이 인수했습니다.

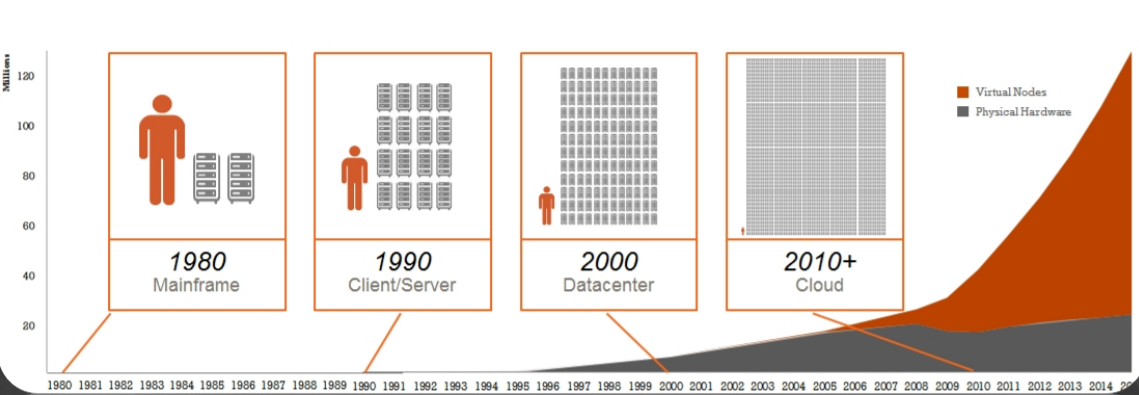
앤서블은 플레이북(PlayBook) 이라는 곳에 실행할 구성을 선언해 놓으면, 필요할 때 마다 자동으로 실행시킬 수 있는 것이 가장 큰 특징입니다.

웹서버의 구성과, DB서버의 구성을 선언해 놓으면 관리자들은 필요할 때마다 그 구성대로 서버의 설정을 배포할 수 있게 해주는 것입니다.

### 1-2. 앤서블의 역할

기존 리눅스에서 동일한 환경을 구성하기 위해 Bash 쉘 스크립트에 패키지의 설치, 설정파일 수정 등을 나열하여 이를 실행하는 것이 일반적이었습니다.

IT의 기술력이 진보함에 따라 인프라 환경도 기존 Data center에서 cloud 환경으로 변화되고 있으며, 한 명의 관리자(Admin)가 관리해야 하는 서버의 숫자가 증가하게 되었습니다.



따라서 클러스터에 존재하는 많은 서버들에 동시에, 동일한 화경을 배포해야 하는 상황이 발생하게 되었고, Bash 쉘 스크립트의 한계점을 갖게 되었습니다.

이를 위해 고안된 Infrastructure as a code 개념입니다.

이것은 환경의 배포와 구성을 규격화된 코드로 정의해 사용하는 것을 의미합니다.

Infrastructure as a code가 가능한 자동화도구를 이용하여 인프라의 상태를 코드로 작성하고 이를 모든 서버에 배포함으로써 특정 환경을 동일하게 유지할 수 있도록 돕게 되었습니다.

이러한 자동화도구 중 가장 대표적인 툴이 바로 앤서블(ANSIBLE)입니다.

## 2. 앤서블 환경 구축

**Ansible-Server Ansible-Nodes**

**브릿지 네트워크**



**인터넷으로**

**연결되는 네트워크**



**HOST PC**



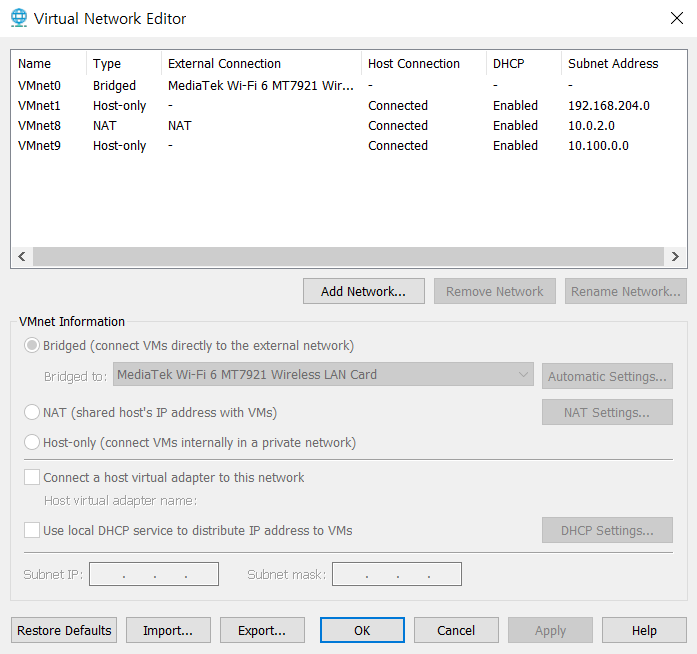
**인터넷**

### 2-1. 가상머신

해당 장에서는 Virtual Box가 아닌 VM Ware로 실습을 진행합니다.

#### 2-1-1. 네트워크 설정

우선 VM ware의 네트워크 설정을 해줍니다.



Change Settings에 들어온 뒤 VMnet0를 Bridge 타입으로 바꾸어 줍니다.

그리고 Bridged to 항목에는 현재 본인이 사용중인 Lan card를 선택하여 줍니다.

#### 2-1-2. VM 생성

텍스트이(가) 표시된 사진

자동 생성된 설명

메모리는 각자의 사양에 맞게 설정해줍니다.

다만 네트워크만 Bridged 타입으로 설정해줍니다.

### 2-2. CentOS 구성

CentOS의 버전은 7버전의 Minimal type의 ISO파일을 이용하여 설치해줍니다.

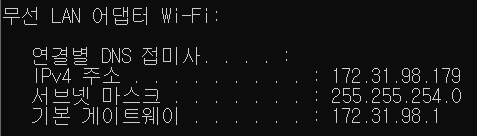


설치는 특이 사항 없습니다.

각 가상머신의 Host Name은 Ansible-Server, Ansible-Node01 ~ 02까지 설정해줍니다.

ip주소의 경우 cmd창에서 ipconfig명령어를 이용하여 나온 본인의 대역에 맞춰서 설정합니다.

그리고 주의할 점은 호스트의 서브넷 대역과 같은 서브넷을 설정해주어야 합니다.



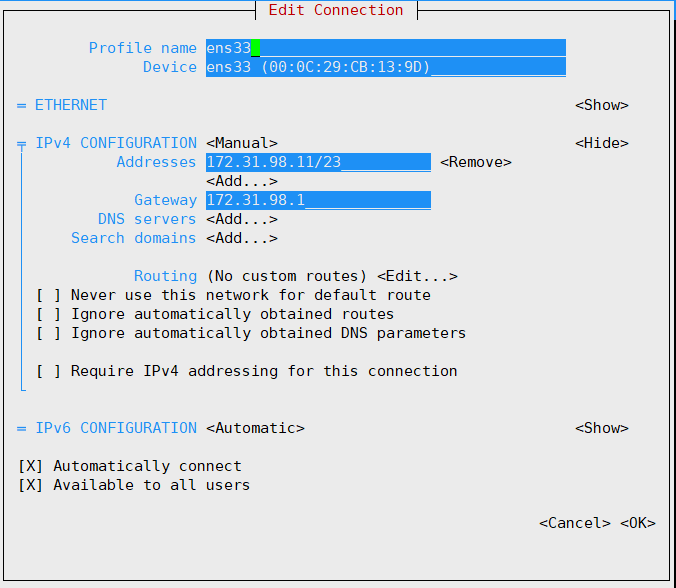
Ansible-Server → 172.31.98.11/23

Ansible-Node01 → 172.31.98.12/23

Ansible-Node02 → 172.31.98.13/23

Gate way → 172.31.98.1

nmtui명령어를 사용하여 위와 같이 ip를 세팅해줍니다.



그리고 ping 명령어로 각 노드 간의 통신을 확인합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

외부와의 통신도 테스트해줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

### 2-3. 앤서블 설치하기

앤서블을 설치하는 방법은 간단합니다.

▶vi /etc/resolv.conf 에 DNS서버 정보 추가하기



▶yum install -y epel-release

기본 패키지에 앤서블이 없기 때문에 추가로 레포지토리를 등록해야 합니다.

▶yum repolist

텍스트이(가) 표시된 사진

자동 생성된 설명

▶yum install -y ansible

설치가 완료되었다면 간단한 명령어로 설치가 제대로 되었는지 확인합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

### 2-4. 앤서블 호스트파일 설정

앤서블을 설치하였다면 앤서블 마스터서버에 각 노드들을 등록해주어야 합니다.

▶vi /etc/ansible/hosts

텍스트이(가) 표시된 사진

자동 생성된 설명

위에서 설정한 각 노드들의 ip주소를 입력해줍니다.

[nginx]의 의미는 밑에 입력한 주소들의 그룹 네임입니다.

나중에 ansible명령어를 실행할 때 사용합니다.

▶ansible all -m ping -k

위의 명령어를 입력하면 최초에는 키 교환이 안 되어있기 때문에 Error 메시지가 출력이 되고 yes를 입력하는 문구가 나옵니다. yes를 입력하고 커맨드가 종료됩니다.

그리고 다시 위의 명령어를 입력하면 SSH비밀번호를 요구합니다. 유저의 비밀번호를 입력합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

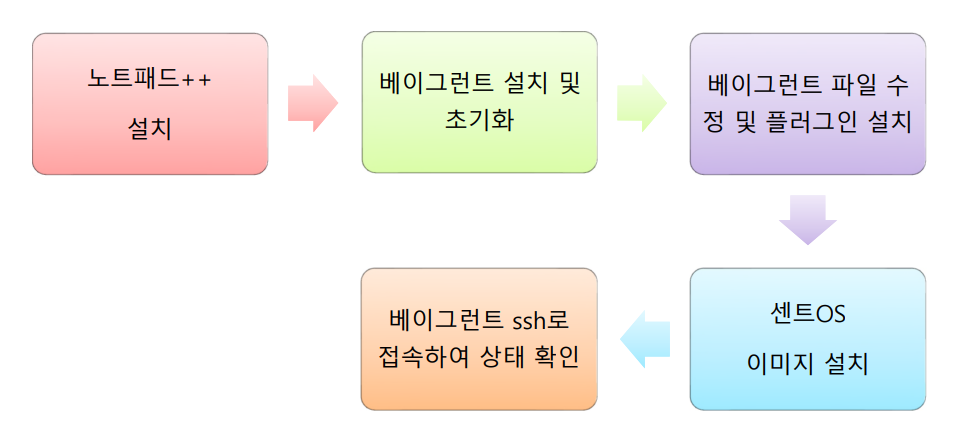
정상적으로 각 노드들과 통신이 되고 있음을 확인할 수 있습니다.

그리고 추가적으로 기호에 맞게 SSH통신 프로그램(Putty, Xshell 등)을 이용하셔도 됩니다.

### 2-5. Vagrant를 이용한 앤서블 환경 구축

이번 목차에서는 베이그런트를 이용해서 앤서블 환경을 구축해봅니다.

**-개요**



전체적인 순서는 위와 같이 진행됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

베이그런트의 기본적인 명령어는 위와 같이 구성됩니다.

**-베이그런트 설치방법**

<https://www.vagrantup.com/>

위의 링크에 접속합니다. 그리고 중간의 다운로드 버튼을 누릅니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

다운로드가 끝나면 설치파일을 실행시켜 설치해줍니다.

그리고 cmd창을 열어 vagrant명령어를 입력해봅니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같은 화면이 나오면 정상적으로 설치가 된 것입니다.

베이그런트 설정 전 디렉토리를 이동합니다. 디렉토리 경로에 혹시 한글 또는 공백이 있을 경우 제대로 실행이 되지 않을 수도 있으므로 디렉토리를 이동시켜 줍니다.

▶cd c:\HashiCory

그리고

▶vagrant init

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 명령어를 실행시키면 Vagrantfile이라는 이름의 샘플 코드가 작성이 됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

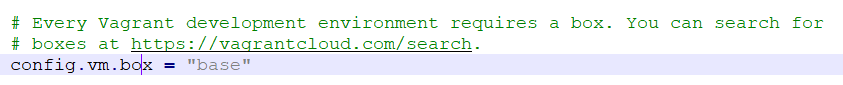
Vagrantfile을 Notepad++로 실행시킵니다.



텍스트이(가) 표시된 사진

자동 생성된 설명

여기서 수정할 부분은 총 2곳입니다.



우선 베이스 OS를 설정해야 합니다. 베이그런트 클라우드 웹사이트에 접속하여 이미지를 검색합시다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같이 여러 이미지들이 나오는데 여기서 우리가 쓸 것은 centos7 버전입니다.



위와 같이 입력합니다.

다음은 공유 디렉토리 설정부분을 수정해야 합니다.

설정하는 것이 다소 번거로우므로 설정하지 않도록 수정합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같이 수정해줍니다.

▶vagrant up

명령어를 실행합니다.

텍스트이(가) 표시된 사진

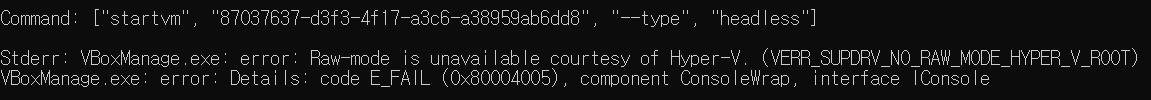
자동 생성된 설명

명령어를 실행하면 제일 먼저 centos7의 이미지 파일을 다운로드 합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

완료되면 위와 같은 메시지가 출력이 됩니다.



혹시라도 위와 같은 에러가 출력이 된다면

1. Hyper-V 기능 끄기

2. Virtual Box 확장 팩 설치 확인 또는 Virtual Box 최신버전으로 업데이트

▶vagrant ssh

텍스트이(가) 표시된 사진

자동 생성된 설명

ssh 명령어로 설치된 가상머신에 접속합니다.

여기 까지가 가장 기본적인 베이그런트를 이용한 가상머신 설치 방법이고

이제는 앤서블 환경을 위한 베이그런트 파일을 작성해봅니다.

**-베이그런트 파일**

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure

# configures the configuration version (we support older styles for

# backwards compatibility). Please don't change it unless you know what

# you're doing.

Vagrant.configure("2") do |config|

config.vm.define:"ansible-server" do |cfg|

cfg.vm.box = "centos/7"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Server(Udemy-Bloter)"

end

cfg.vm.host\_name="ansible-server"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.11"

cfg.vm.network "forwarded\_port", guest: 22, host: 17211, auto\_correct: false, id: "ssh"

cfg.vm.provision "shell", path: "bootstrap.sh"

end

end

기존에 작성했던 베이그런트 파일을 위와 같이 수정합니다.

**-스크립트 파일**

#! /usr/bin/env bash

yum install epel-release -y

yum install ansible -y

그리고 베이그런트 파일에서 사용하는 스크립트 파일 (bootstrap.sh)의 내용은 위와 같습니다.

파일을 모두 수정했다면 실행시켜 봅니다.

▶vagrant up

텍스트이(가) 표시된 사진

자동 생성된 설명

정상적으로 완료되면 위와 같은 메시지가 출력이 됩니다.

▶vagrant ssh

설치가 완료되었다면 ssh 명령어를 이용해 생성한 가상머신에 접속합니다.

▶ip addr

텍스트이(가) 표시된 사진

자동 생성된 설명

설정파일에서 입력한 IP가 제대로 입력되었는지 확인합니다.

▶ansible

그리고 ansible명령어를 사용하여 앤서블이 제대로 설치되었는지 확인해봅니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

### 2-6. Ansible-Server 추가 설정 프로비저닝

이번 장에서는 yaml파일을 조금 더 쉽게 작성 가능하도록 하는 설정을 추가해봅니다.

#### 2-6-1. vim-plug & ansible-vim 설치

**이 방법은 베이그런트를 이용하지 않고 직접 설정하는 방법입니다.**

**베이그런트를 이용하는 방법은 이 뒤에 설명하겠습니다.**

<https://github.com/junegunn/vim-plug> (vim-plug) 접속

텍스트이(가) 표시된 사진

자동 생성된 설명

▶curl -fLo ~/.vim/autoload/plug.vim --create-dirs \

https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

<https://github.com/pearofducks/ansible-vim> (ansible-vim)

텍스트이(가) 표시된 사진

자동 생성된 설명

▶vi ~/.vimrc

call plug#begin()

Plug 'pearofducks/ansible-vim'

call plug#end()

▶yum install -y vim-enhanced

▶yum install -y git

▶vim

텍스트이(가) 표시된 사진

자동 생성된 설명

:PlugInstall 입력

텍스트이(가) 표시된 사진

자동 생성된 설명

vimrc에 입력한 플러그인을 설치하고 완료된다.

▶vi ~/.bashrc

alias vi=’vim’

내용추가

▶su –

설정 적용

#### 2-6-2. 베이그런트를 이용한 프로비저닝 설정

**(해당 내용은 아래의 “3. 앤서블” 기본내용을 본 뒤 참고하시기 바랍니다.)**

**- Vagrantfile 수정하기**

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant\_API\_Version ="2"

Vagrant.configure(Vagrant\_API\_Version) do |config|

config.vm.define:"ansible-server" do |cfg|

cfg.vm.box = "centos/7"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Server(Udemy-Bloter)"

end

cfg.vm.host\_name="ansible-server"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.11"

cfg.vm.network "forwarded\_port", guest: 22, host: 17211, auto\_correct: false, id: "ssh"

cfg.vm.provision "shell", path: "bootstrap.sh"

cfg.vm.provision "file", source: "Ansible\_env\_ready.yml", destination: "Ansible\_env\_ready.yml"

cfg.vm.provision "shell", inline: "ansible-playbook Ansible\_env\_ready.yml"

end

end

새로 추가한 코드의 내용은 다음과 같습니다.

▶Vagrant\_API\_Version ="2"

Vagrant\_API\_Version을 2를 변수로 지정하여 Vagrant Configure의 값으로 넣어준다.

▶cfg.vm.provision "shell", path: "bootstrap.sh"

path를 사용하면 해당 파일을 전송 및 실행한다.

▶cfg.vm.provision "file", source: "Ansible\_env\_ready.yml", destination: "Ansible\_env\_ready.yml"

file을 사용하면 해당 파일을 프로비저닝 하는 서버로 복사한다.

▶cfg.vm.provision "shell", inline: "Ansible-playbook Ansible\_env\_ready.yml"

inline을 사용하면 명령어를 실행한다.

**-bootstrap.sh 파일 수정하기**

#! /usr/bin/env bash

#ansible 설치

yum install epel-release -y

yum install ansible -y

#환경설정 초기 파일 구성 for vagrant Only

mkdir -p /home/vagrant/.vim/autoload /home/vagrant/.vim/bundle

touch /home/vagrant/.vimrc

touch /home/vagrant/.bashrc

.vimrc는 vagrant 사용자의 vim 환경설정 파일이며, .bashrc는 vagrant 사용자의 환경설정 파일입니다.

**-Ansible\_env\_ready.yml 파일 작성**

# Ansible\_env\_ready.yml

---

- name: Setup for the Ansible's Enviroment

hosts: localhost

gather\_facts: no

tasks:

- name: Install vim-enhanced

yum:

name: vim-enhanced

state: present

- name: Install git

yum:

name: git

state: present

- name: Download pathogen.vim

shell: "curl -fLo /home/vagrant/.vim/autoload/pathogen.vim https://tpo.pe/pathogen.vim"

- name: Git clone vim-ansible-yaml

git:

repo: 'https://github.com/chase/vim-ansible-yaml.git'

dest: /home/vagrant/.vim/bundle/vim-ansible-yaml

- name: Configure vimrc

lineinfile:

dest: /home/vagrant/.vimrc

line: "{{ item }}"

with\_items:

- "set number"

- "execute pathogen#infect()"

- "syntax on"

- name: Configure Bashrc

lineinfile:

dest: /home/vagrant/.bashrc

line: "{{ item }}"

with\_items:

- "alias vi='vim'"

- "alias ans='ansible'"

- "alias anp='ansible-playbook'"

이제 모든 파일의 작성 및 수정이 끝났습니다. 이제 수정사항을 적용시켜 봅시다.

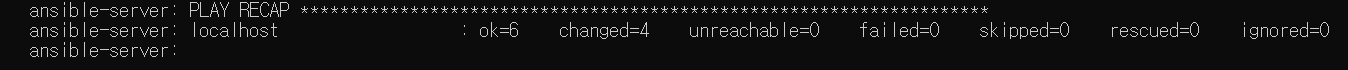
cmd창에서 다음의 명령어를 입력합니다.

▶vagrant provision ansible-server

텍스트이(가) 표시된 사진

자동 생성된 설명

……



provision 명령어를 이용하면 centos를 재가동 하지 않고도 변경된 사항을 적용할 수 있다.

▶vagrant ssh ansible-server

▶ls

텍스트이(가) 표시된 사진

자동 생성된 설명

Vagrantfile에서 설정한 yaml파일이 생성된 것을 볼 수 있다. 그리고 vi 편집기 (vim)로 해당 파일을 열어보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

set number 등등 여러가지 설정들이 적용된 것이 확인 가능하다.

### 2-7. 베이그런트로 앤서블 노드 추가하기

현재 까지는 베이그런트로 서버노드만 추가하였습니다. 이제는 베이그런트를 이용하여 앤서블 클러스터에 묶일 워커 노드들을 추가해 보겠습니다.

총 3가지 파일을 수정 또는 생성해야 합니다.

**-Vagrantfile (수정)**

**-Ansible\_env\_ready.yml (수정)**

**-add\_ssh\_auth.sh (추가)**

우선 Vagrantfile을 수정합니다. 워커노드 2개의 설정과 ssh설정을 위한 스크립트 파일을 불러오는 설정이 추가됩니다.

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant\_API\_Version ="2"

Vagrant.configure(Vagrant\_API\_Version) do |config|

#Ansible-Node01

config.vm.define:"ansible-node01" do |cfg|

cfg.vm.box = "centos/7"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Node01(Udemy-Bloter)"

vb.customize ["modifyvm", :id, "--cpus",1]

vb.customize ["modifyvm", :id, "--memory",512]

end

cfg.vm.host\_name="ansible-node01"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.12"

cfg.vm.network "forwarded\_port", guest: 22, host: 17212, auto\_correct: false, id: "ssh"

end

#Ansible-Node02

config.vm.define:"ansible-node02" do |cfg|

cfg.vm.box = "centos/7"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Node02(Udemy-Bloter)"

vb.customize ["modifyvm", :id, "--cpus",1]

vb.customize ["modifyvm", :id, "--memory",512]

end

cfg.vm.host\_name="ansible-node02"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.13"

cfg.vm.network "forwarded\_port", guest: 22, host: 17213, auto\_correct: false, id: "ssh"

end

#Ansible-Server

config.vm.define:"ansible-server" do |cfg|

cfg.vm.box = "centos/7"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Server(Udemy-Bloter)"

end

cfg.vm.host\_name="ansible-server"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.11"

cfg.vm.network "forwarded\_port", guest: 22, host: 19210, auto\_correct: false, id: "ssh"

cfg.vm.provision "shell", path: "bootstrap.sh"

cfg.vm.provision "file", source: "Ansible\_env\_ready.yml", destination: "Ansible\_env\_ready.yml"

cfg.vm.provision "shell", inline: "ansible-playbook Ansible\_env\_ready.yml"

cfg.vm.provision "shell", path: "add\_ssh\_auth.sh", privileged: false

end

end

다음으로는 ssh관련 설정을 하는 add\_ssh\_auth.sh 파일을 생성합니다.

# add\_ssh\_auth.sh

#! /usr/bin/env bash

#ssh key 생성

sshpass -p vagrant ssh -T -o StrictHostKeyChecking=no vagrant@node01

sshpass -p vagrant ssh -T -o StrictHostKeyChecking=no vagrant@node02

마지막으로 리눅스 호스트 파일에 각 노드들의 정보를 입력하고 앤서블 호스트 파일에 추가한 노드의 정보를 입력하고, ssh를 설치하는 부분을 앤서블 플레이북 파일(Ansible\_env\_ready.yml)에 추가합니다.

# Ansible\_env\_ready.yml

---

- name: Setup for the Ansible's Enviroment

hosts: localhost

gather\_facts: no

tasks:

- name: Add "/etc/hosts"

blockinfile: |

dest=/etc/hosts

content="

172.31.98.11 server

172.31.98.12 node01

172.31.98.13 node02"

- name: Add "/etc/ansible/hosts"

blockinfile: |

dest=/etc/ansible/hosts

content="

[CentOS]

node01

node02"

- name: Install sshpass for Authentication

yum:

name: sshpass

state: present

- name: Install vim-enhanced

yum:

name: vim-enhanced

state: present

- name: Install git

yum:

name: git

state: present

- name: Download pathogen.vim

shell: "curl -fLo /home/vagrant/.vim/autoload/pathogen.vim https://tpo.pe/pathogen.vim"

- name: Git clone vim-ansible-yaml

git:

repo: 'https://github.com/chase/vim-ansible-yaml.git'

dest: /home/vagrant/.vim/bundle/vim-ansible-yaml

- name: Configure vimrc

lineinfile:

dest: /home/vagrant/.vimrc

line: "{{ item }}"

with\_items:

- "set number"

- "execute pathogen#infect()"

- "syntax on"

- name: Configure Bashrc

lineinfile:

dest: /home/vagrant/.bashrc

line: "{{ item }}"

with\_items:

- "alias vi='vim'"

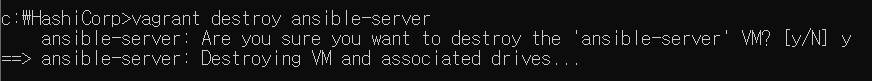
- "alias ans='ansible'"

- "alias anp='ansible-playbook'"

모든 파일의 설정을 마쳤으면 기존에 생성 되어있는 가상머신을 삭제하고 다시 실행시킵니다.

▶vagrant destroy ansible-server

▶y입력



▶vagrant up

텍스트이(가) 표시된 사진

자동 생성된 설명

…

텍스트이(가) 표시된 사진

자동 생성된 설명

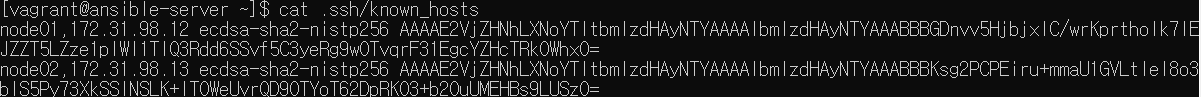
텍스트이(가) 표시된 사진

자동 생성된 설명

파일에 문제가 없다면 정상적으로 3개의 가상머신이 생성됩니다. ansible-server 노드에 접속하여 설정한 내용들이 정상적으로 적용되었는지 확인해 봅니다.

▶vagrant ssh ansible-server

▶cat .ssh/known\_hosts



▶vi /etc/hosts

텍스트이(가) 표시된 사진

자동 생성된 설명

앞서 설정하였던 vim-plug의 내용도 잘 적용되었고 hosts파일에 정상적으로 노드들이 등록되었습니다.

▶cat /etc/ansible/hosts

텍스트이(가) 표시된 사진

자동 생성된 설명

CentOS라는 그룹명으로 node01, node02도 정상적으로 등록되었습니다.

▶vagrant all m -ping -k

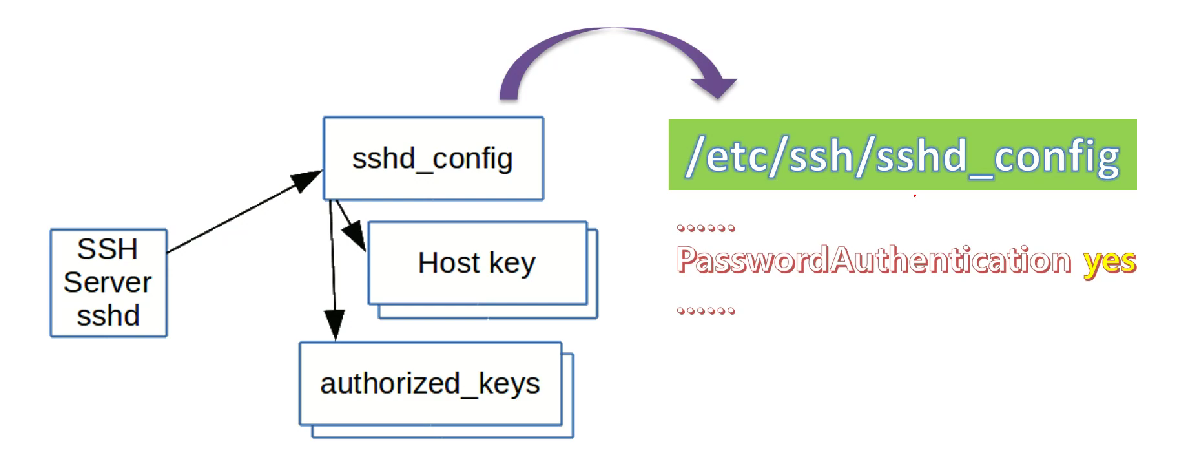
텍스트이(가) 표시된 사진

자동 생성된 설명

하지만 각 노드들 간의 통신이 되지 않습니다.

ssh통신을 할 때 비밀번호를 이용한 통신이 안되게 설정이 되어 있기 때문입니다.

### 2-8. 앤서블 서버와 노드의 통신



외부에서 서버, 노드에 접속하면 sshd\_config에서 설정과 방법 등을 읽어서 적용을 시킵니다.

sshd\_config의 내용중 PasswordAuthentication no -> yes로 내용을 바꾸어 주어야 하는데,

크게 2가지의 방법을 소개하겠습니다.

#### 2-8-1. Ansible Replace Module을 이용하는 방법

# Ansible\_ssh\_conf\_4\_CentOS.yml

---

- name: Ansible\_ssh\_conf\_4\_CentOS

hosts: localhost

gather\_facts: no

tasks:

- name: PasswordAuthentication change from no to yes

replace:

dest: /etc/ssh/sshd\_config

regexp: 'PasswordAuthentication no'

replace: 'PasswordAuthentication yes'

backup: yes

- name: sshd restart to apply "PasswordAuthentication"

service:

name: sshd

state: restarted

위와 같은 내용의 yaml파일을 작성합니다. 그리고 Vagrantfile에 해당 내용을 불러오는 구문을 추가합니다.

cfg.vm.provision "file", source: "Ansible\_ssh\_conf\_4\_CentOS.yml", destination: "Ansible\_ssh\_conf\_4\_CentOS.yml"

cfg.vm.provision "shell", inline: "ansible-playbook Ansible\_ssh\_conf\_4\_CentOS.yml"



#### 2-8-2. Bash Shell을 이용하는 방법

앤서블이 설치되어 있지 않은 노드에서 사용하는 방법입니다.

코드는 다음과 같습니다. 해당 스크립트를 폴더에 추가해줍니다.

# bash\_ssh\_conf\_4\_CentOS.sh

#! /usr/bin/env bash

now=$(date +"%m\_%d\_%Y")

cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config\_$now.backup

sed -i -e 's/PasswordAuthentication no/PasswordAuthentication yes/g' /etc/ssh/sshd\_config

systemctl restart sshd

코드를 잘 작성하였다면 Vagrantfile의 각 노드(node01, node02 둘다) 설정하는 부분의 가장 하단에 작성한 스크립트를 불러오는 라인을 추가해줍니다.

cfg.vm.provision "shell", path: "bash\_ssh\_conf\_4\_CentOS.sh"

텍스트이(가) 표시된 사진

자동 생성된 설명

모든 내용의 수정이 끝났다면

▶ansible all -m ping -k

텍스트이(가) 표시된 사진

자동 생성된 설명

정상적으로 노드 간의 통신이 되는 것을 확인할 수 있습니다.

▶vi /etc/ssh/sshd\_config (서버, 노드 둘 다 확인해 보세요)

Pass 검색



## 3. 앤서블 기본내용

### 3-1. 앤서블의 구성파일들

▶/etc/ansible/ansible.cfg

앤서블의 실행정보를 가지고 있는 기본 환경설정 파일

▶/etc/ansible/hosts

앤서블이 접속하는 호스트들에 대한 정보

3-2. 앤서블 명령어 실행 시 옵션 값

**▶-i (--inventory-file) :** 적용될 호스트들에 대한 파일, 특정 호스트에만 명령을 실행할 때

예를 들어 test라는 파일에 Node01의 IP주소만 적어 놓았다면

ansible all -i ./test -m ping -k를 입력한다면 (all 대신 hosts파일에 설정한 그룹명을 적어도 가능하다)

Node01의 핑 정보만 출력이 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**▶-m (--module-name) :** 모듈을 선택하는 옵션

앤서블의 모듈이란 간단히 기능 정도로 생각 하면 된다.

**▶-k (--ask-key) :** 비밀번호를 물어 보도록 설정

위의 명령어 중 -k 라는 명령어가 보일 텐데 이는 명령어 실행이 비밀번호를 입력하도록 하는 옵션이다. 위 명령어에서 -k옵션을 빼면 오류가 발생하는데 호스트 간의 인증키 교환이 되어 있지 않기 때문이다.

**▶-K (--ask-become-pass) :** 명령어 실행 시 관리자 비밀번호를 입력하게 한다. 관리자의 권한이 필요한 경우 사용한다.

텍스트, 텔레비전, 화면, 시청이(가) 표시된 사진

자동 생성된 설명

**▶--list-hosts :** 특정 명령어 실행 시 영향을 받게 되는 호스트들을 보여준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

### 3-3. 한번의 명령어로 다수의 시스템에 작업하기

다음은 앤서블을 활용하는 가장 기본적인 형태의 예시들입니다.

**- uptime 확인하기**

텍스트이(가) 표시된 사진

자동 생성된 설명

shell 명령어를 사용하기 위해 shell 모듈을 이용했고 변수 값을 입력하기 위해 a 옵션을 이용했습니다. 그리고 ‘ ’ 안에 사용할 명령어를 입력합니다.

사진과 같이 각 호스트들의 uptime을 출력해줍니다.

uptime 외에도 ‘df -h’ 같은 명령어로 디스크 정보를 확인한다든가, ‘free -h’ 명령어로 메모리 정보를 확인할 수도 있습니다.

**- User 생성**

텍스트이(가) 표시된 사진

자동 생성된 설명

-m user 모듈을 이용하여 각 호스트들에게 유저를 생성할 수 있습니다.

**-파일 복사**

텍스트이(가) 표시된 사진

자동 생성된 설명

**-m copy -a “src=대상파일 dest=복사할 위치”**

카피 옵션을 이용하면 특정 파일을 각 호스트들에게 복사할 수 있습니다.

텍스트, 오렌지, 어두운이(가) 표시된 사진

자동 생성된 설명

-

**-서비스 설치**

텍스트이(가) 표시된 사진

자동 생성된 설명

yum 모듈을 이용하면 각 호스트에게 필요한 프로그램을 설치할 수 있습니다. 위의 예시는 아파치 데몬을 설치한 예시입니다.

### 3-3. 앤서블 Play Book

**-앤서블 Playbook이란?**

Playbook의 사전적 의미는 각본, 작전, 계획의 뜻을 가지고 있습니다.

Playbook은 오케스트레이터 언어로 호스트 서버에서 시행할 정책, 시스템 프로세스 단계의 설정 자동화를 위해 사용됩니다.

Playbook은 반복, 재사용, 간단한 구성 관리 및 다중 시스템 배포, 복잡한 애플리케이션 배포 사용에 적합합니다.

**-Playbook의 특징 멱등성(idempotent)**

멱등성이란 연산을 여러 번 적용하더라도 결과가 달라지지 않는 성질입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이와 같이 echo 명령어를 이용하여 내용을 추가하게 되면 똑같은 결과값이 3번 중복되어 입력됩니다.

따라서 멱등성을 위해 아래와 같이 Ansible Playbook을 작성하여 사용하게 됩니다.

vi joon.yml

---

- name: Ansible\_vim

hosts: localhost

tasks:

- name: Add ansible hosts

blockinfile:

path: /etc/ansible/hosts

block: |

[joon]

172.16.0.200

텍스트, 표지판, 닫기이(가) 표시된 사진

자동 생성된 설명

**▶ansible-playbook file-name.yaml**

위의 명령어로 작성한 yaml 코드를 실행하면 위와 같이 내용이 추가가 됩니다. 이 코드를 여러 번 반복해도 똑 같은 내용이 여러 줄 추가가 되지 않습니다.

**- 앤서블 Play Book을 이용하여 각 노드들에 웹서비스 설치 후 가동**

---

- hosts: nginx

remote\_user: root

tasks:

- name: install epel-release

yum: name=epel-release state=latest

- name: install nginx web server

yum: name=nginx state=present

- name: Upload default index.html for web server

copy: src=index.html dest=/usr/share/nginx/html/ mode=0644

- name: Start nginx web server

service: name=nginx state=started

우선 해당내용으로 야믈파일을 하나 작성합니다. 그리고 아래의 명령어로 각 노드들의 방화벽을 해제합니다.

**▶ansible nginx -m shell -a ‘systemctl stop firewalld’ -k**

그리고 인덱스 페이지의 소스코드를 가져와야 하므로 아래의 명령어로 다운로드 받습니다.

**▶curl -o index.html** [**https://www.nginx.com**](https://www.nginx.com)

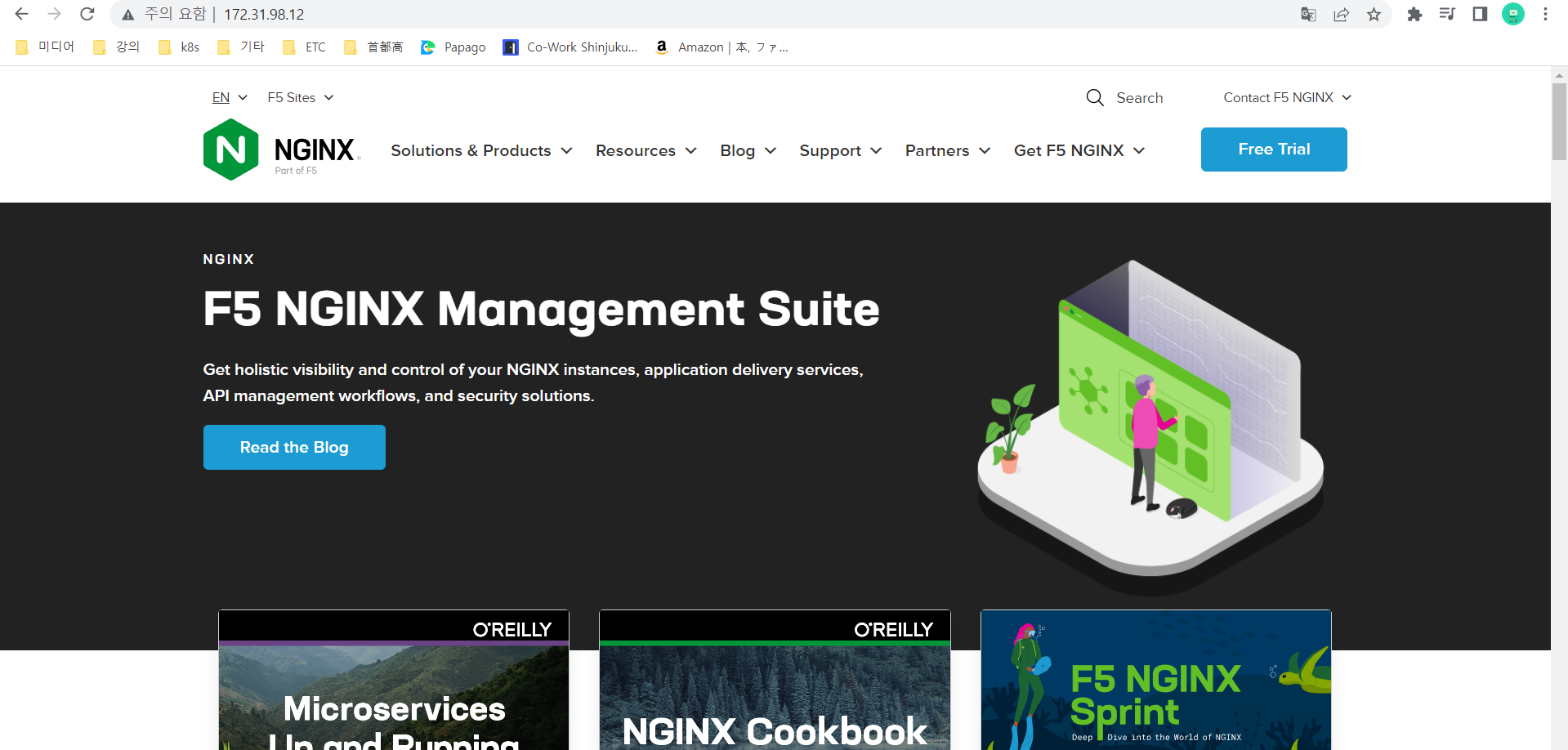
완료가 되었다면 아래의 명령어를 이용해 플레이북 파일을 실행시킵니다.

**▶ansible-playbook nginx.yml -k**

테이블이(가) 표시된 사진

자동 생성된 설명

웹 브라우저를 열고 노드의 아이피 주소를 입력하면



위의 화면과 같이 인덱스 페이지가 출력이 됩니다.

## 4. 앤서블 – CentOS 관리하기

3장에서 다루었던 플레이북 예제와 크게 다르지 않습니다.

약간의 내용을 추가해서 진행하도록 하겠습니다.

### 4-1. 앤서블 노드에 nginx설치 및 테스트 그리고 삭제하기



nginx 배포 과정은 위와 같은 순서로 진행됩니다.

1~3번은 하나의 플레이북으로 묶일 것이고, 삭제는 별도의 플레이북에 작성이 됩니다.

우선 마스터서버에 ssh로 접속을 해줍니다. 그리고 다음의 두 파일을 작성합니다.

**-nginx\_install.yml**

---

- name: install nginx on CentOS

hosts: CentOS

gather\_facts: no

become: yes

tasks:

- name: install epel-release

yum:

name: epel-release

state: latest

- name: install nginx

yum:

name: nginx

state: present

- name: upload default index.html for web server

get\_url:

url: https://www.nginx.com

dest: /usr/share/nginx/html/

mode: 0644

- name: Start nginx web server

service:

name: nginx

state: started

여기서 becom: yes는 root권한을 얻기 위한 코드입니다.

**-nginx\_remove.yml**

nginx를 삭제하는 플레이북 코드입니다.

# nginx\_remove.yml

---

- name: Remove nginx on CentOS

hosts: CentOS

gather\_facts: no

become: yes

tasks:

- name: remove epel-release

yum:

name: epel-release

state: absent

- name: remove nginx web server

yum:

name: nginx

state: absent

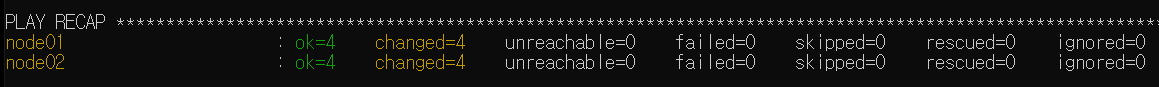
두 파일 모두 작성 완료하였다면, 다음의 명령어로 우선 nginx를 설치합니다.

▶ansible-playbook nginx\_install.yml -k

(ansible-playbook = anp)

텍스트이(가) 표시된 사진

자동 생성된 설명



그리고 노드의 아이피 주소를 인터넷 브라우저에 입력하여 정상적으로 nginx가 설치되었는지 확인합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

▶ansible-playbook nginx\_remove.yml -k

텍스트이(가) 표시된 사진

자동 생성된 설명

삭제용 플레이북도 잘 작동하는 것이 확인됩니다.

### 4-2. Time Zone 변경하기

본 장에서는 간단히 서비스의 인스톨 이외에 기능을 소개하겠습니다.

운영체제의 시간대를 변경하는 것입니다.

간단히 플레이북 파일을 만들어 변경해 보겠습니다.

# timezone.yml

---

- name: Setup linux timezone

hosts: CentOS

gather\_facts: no

become: yes

tasks:

- name: set timezone to Asia/Seoul

timezone:

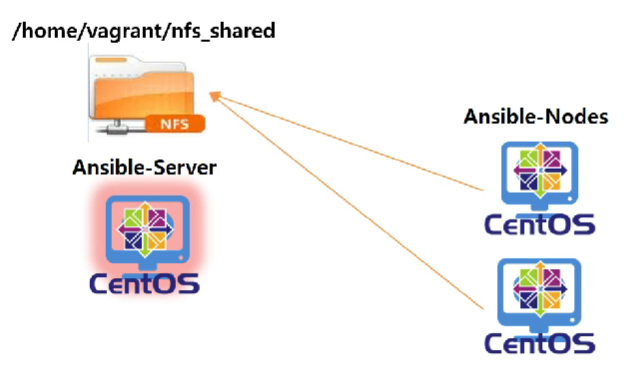
name: Asia/Seoul

텍스트이(가) 표시된 사진

자동 생성된 설명

### 4-4. NFS서버와 클라이언트 구성하기

이번 장에서는 하나의 플레이북으로 앤서블 서버에는 NFS서버를 구축하고 앤서블 노드에는 NFS클라이언트로 설정을 해보겠습니다.



우선 아래와 같은 내용의 플레이북 파일을 생성합니다.

---

- name: Setup for nfs server

hosts: localhost

gather\_facts: no

tasks:

- name: make nfs\_shared directory

file:

path: /home/vagrant/nfs\_shared

state: directory

mode: 0777

- name: configure /etc/exports

become: yes

lineinfile:

path: /etc/exports

line: /home/vagrant/nfs\_shared 172.31.98.0/23(rw,sync)

- name: nfs service start

become: yes

service:

name: nfs

state: restarted

- name: Setup for nfs client

hosts: CentOS

gather\_facts: no

tasks:

- name: make nfs\_client directory

file:

path: /home/vagrant/nfs

state: directory

- name: mount point directory as client

become: yes

mount:

name: /home/vagrant/nfs

src: 172.31.98.11:/home/vagrant/nfs\_shared

fstype: nfs

opts: nfsvers=3

state: mounted

opts: nfsvers=3 는 상대변 nfs버전에 맞게 선언을 해주어야 하기 때문에 추가한 옵션이다.

생성을 완료했다면 플레이북을 실행시켜줍니다.

▶anp nfs.yml -k

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 앤서블 노드에 접속하여 테스트용 파일을 만들고 앤서블 서버에서 조회해 봅니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명



## 5. 앤서블 – Ubuntu OS 관리하기

이번 장에서는 앞서 CentOS를 관리하던 환경에 Ubuntu OS를 추가하여 앞서 해왔던 기능들을 Ubuntu에서 실습해 보겠습니다.

### 5-1. 기존 환경에 우분투 OS 추가하기

우선 Vagrantfile을 수정해야 합니다.

#Ansible-Node03

config.vm.define:"ansible-node03" do |cfg|

cfg.vm.box = "ubuntu/trusty64"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Node03"

vb.customize ["modifyvm", :id, "--cpus",1]

vb.customize ["modifyvm", :id, "--memory",1024]

end

cfg.vm.host\_name="ansible-node03"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.14"

cfg.vm.network "forwarded\_port", guest: 22, host: 17214, auto\_correct: false, id: "ssh"

end

#Ansible-Node04

config.vm.define:"ansible-node04" do |cfg|

cfg.vm.box = "ubuntu/trusty64"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Node04"

vb.customize ["modifyvm", :id, "--cpus",1]

vb.customize ["modifyvm", :id, "--memory",1024]

end

cfg.vm.host\_name="ansible-node04"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.15"

cfg.vm.network "forwarded\_port", guest: 22, host: 17215, auto\_correct: false, id: "ssh"

end

기존 파일에 Node03, Node04의 설정 파일을 추가합니다.

cfg.vm.provision "shell", path: "bash\_ssh\_conf\_4\_CentOS.sh"

이 내용은 우분투의 경우 sshd\_config의 내용을 수정하지 않아도 노드 간의 통신이 가능하므로 코드에서 빼 주도록 합니다.

그리고 Ansible\_env\_ready.yml 파일을 수정합니다.

tasks:

- name: Add "/etc/hosts"

blockinfile: |

dest=/etc/hosts

content="

172.31.98.11 server

172.31.98.12 node01

172.31.98.13 node02

172.31.98.14 node03

172.31.98.15 node04"

- name: Add "/etc/ansible/hosts"

blockinfile: |

dest=/etc/ansible/hosts

content="

[CentOS]

node01

node02

[Ubuntu]

node03

node04"

위의 내용을 추가합니다.

그리고 마지막으로 add\_ssh\_auth.sh 파일을 수정합니다.

# add\_ssh\_auth.sh

#! /usr/bin/env bash

#ssh key 생성

sshpass -p vagrant ssh -T -o StrictHostKeyChecking=no vagrant@node01

sshpass -p vagrant ssh -T -o StrictHostKeyChecking=no vagrant@node02

sshpass -p vagrant ssh -T -o StrictHostKeyChecking=no vagrant@node03

sshpass -p vagrant ssh -T -o StrictHostKeyChecking=no vagrant@node04

수정이 완료되었다면 수정 사항을 프로비저닝 해줍니다.

▶vagrant up

텍스트이(가) 표시된 사진

자동 생성된 설명

node03번부터 추가가 되는 모습을 확인할 수 있다.

▶vagrant provision ansible-server

텍스트이(가) 표시된 사진

자동 생성된 설명

▶ansible all -m ping -k

텍스트이(가) 표시된 사진

자동 생성된 설명

정상적으로 ping이 찍힌다.

### 5-2. Time Zone 수정하기

앞서 CentOS의 Time Zone을 수정하는 것과 동일 합니다. ansible-server의 timezone.yml파일을 아주 조금만 수정해 봅시다.

# vi timezone.yml

---

- name: Setup linux timezone

hosts: CentOS:UbuntuOS

gather\_facts: no

become: yes

tasks:

- name: set timezone to Asia/Seoul

timezone:

name: Asia/Seoul

hosts부분에 본인이 추가한 노드의 그룹명을 추가해 줍니다.

▶anp timezone.yml -k

텍스트이(가) 표시된 사진

자동 생성된 설명

▶ans all -m shell -a “date” -k 텍스트이(가) 표시된 사진

자동 생성된 설명

\* nfs서버의 설정도 타임 존의 경우와 마찬가지로 hosts부분에 본인이 추가한 노드의 그룹명을 추가해 주면 됩니다.

### 5-3. 우분투에 nginx 추가하기

타임 존과 nfs설정과는 다르게 nginx를 추가할 경우에는 기존 파일에서 수정해야 할 부분이 조금 더 많습니다. 그 이유는 CentOS와 우분투의 패키지 매니저 이름이 다르기 때문입니다.

CentOS – yum

Ubuntu - apt

다음의 내용을 nginx\_install.yml 파일에 추가합니다.

- name: install nginx on Ubuntu

hosts: Ubuntu

gather\_facts: no

become: yes

tasks:

- name: install nginx

apt:

pkg: nginx

state: present

update\_cache: yes

- name: upload default index.html for web server

get\_url:

url: https://www.apache.org

dest: /usr/share/nginx/html/

mode: 0644

validate\_certs: no

## 6. 앤서블 – Windows 관리하기 (미완성)

이번 장에서는 현재 구성되어 있는 시스템에 윈도우 운영체제를 사용하는 노드를 한 대 추가하여 관리하는 방법을 알아보겠습니다.

### 6-1. Vagrant를 이용해서 Windows 노드 추가하기

**-Vagrantfile**

#Ansible-Node05

config.vm.define:"ansible-node05" do |cfg|

cfg.vm.box = "sysnet4admin/Windows2016"

cfg.vm.provider:virtualbox do |vb|

vb.name="Ansible-Node05"

vb.customize ["modifyvm", :id, "--cpus",2]

vb.customize ["modifyvm", :id, "--memory",2048]

end

cfg.vm.host\_name="ansible-node05"

cfg.vm.synced\_folder ".", "/vagrant", disabled: true

cfg.vm.network "public\_network", ip: "172.31.98.16"

cfg.vm.network "forwarded\_port", guest: 22, host: 17216, auto\_correct: false, id: "ssh"

cfg.vm.provision "shell", inline: "netsh firewall set opmode disable"

end

node05의 내용을 Vagrantfile에 추가합니다. 테스트 환경을 구성하기 위해서 윈도우 서버의 방화벽을 끄기 위해 cfg.vm.provision "shell", inline: "netsh firewall set opmode disable"를 추가합니다.

**-Ansible\_evn\_ready.yml**

- name: Add "/etc/hosts"

blockinfile: |

dest=/etc/hosts

content="

172.31.98.11 server

172.31.98.12 node01

172.31.98.13 node02

172.31.98.14 node03

172.31.98.15 node04

172.31.98.16 node04"

- name: Add "/etc/ansible/hosts"

blockinfile: |

dest=/etc/ansible/hosts

content="

[CentOS]

node01

node02

[UbuntuOS]

node03

node04

[Win]

node05"

새로 추가한 노드의 호스트 정보를 추가해줍니다.

그리고 모든 파일을 수정했으면

C:\HashiCorp>vagrant up ansible-node05

C:\HashiCorp>vagrant provision ansible-server

위의 명령어를 통해 변경 사항을 적용시킵니다.

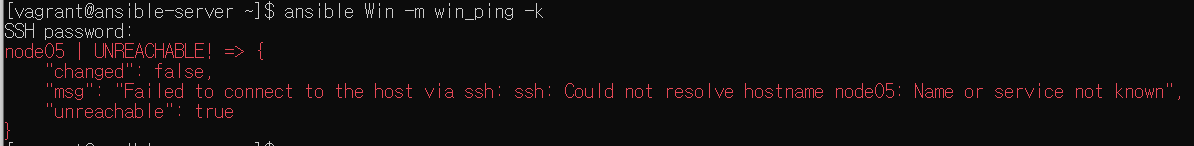
▶vagrant ssh ansible-server

▶ping 172.31.98.16

텍스트, 칠판이(가) 표시된 사진

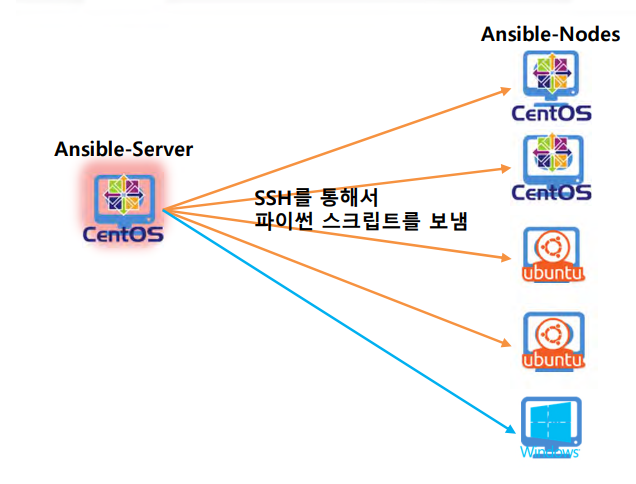
자동 생성된 설명

▶ansible Win -m win\_ping -k



현재 win\_ping모듈을 이용해서 윈도우 노드와 통신을 시도하였지만 실패하였습니다.

실패하는 이유를 설명하기 위해서는 앤서블의 실행 과정을 이해해야 합니다.



위의 그림과 같이 앤서블은 각 노드들에게 SSH를 통해 파이썬 스크립트를 보내서 명령을 수행합니다. 그래서 기본적으로 SSH통신이 되어야 하고 파이썬이 설치되어 있어야 합니다. 하지만 윈도우의 경우 기본 설정으로는 SSH통신, 파이썬이 설치되어 있지 않습니다.

이러한 경우에는 어떻게 해야 할까요?

**-윈도우 노드의 요구사항**

**1. winrm으로 통신이 가능 해야함**

여기서 winrm이란 Microsoft가 구현한 프로토콜로써, 웹 서비스를 사용하여 로컬 및 원격 컴퓨터가 안전하게 통신할 수 있는 방법입니다.

ansible-server에서 ansible 명령어로 모듈을 동작시키면 윈도우와 ssh로 통신하려 하기 때문에 정상적으로 동작하지 않으므로 winrm으로 통신하도록 추가 설정이 필요합니다.

**▶추가 설정 목록**

〮winrm으로 통신해야한다는 조건

〮사용자가 사용가능한 계정이어야 한다는 조건

〮winrm이 통신하도록 열려져 있는 포트 5985로 통신하도록 설정

- name: Add "/etc/ansible/hosts"

blockinfile: |

dest=/etc/ansible/hosts

content="

[CentOS]

node01

node02

[UbuntuOS]

node03

node04

[Win]

node05 ansible\_connection=winrm ansible\_ssh\_user=vagrant ansible\_ssh\_port=5985"

위의 내용처럼 Ansible\_env\_ready.yml파일에 통신할 수 있도록 설정을 추가해줍니다.

**2. 파이썬을 윈도우가 이해하는 언어로 해석해 주어야함**

pywinrm 모듈로 해결할 수 있다.

- name: Install pip

yum:

name: python-pip

state: present

- name: Install pywinrm

pip:

name: pywinrm

state: present

위의 내용을 Ansible\_env\_ready.yml 파일에 추가해줍니다.

**\* 현재 파이썬 버전 관련 이슈로 pywinrm이 설치가 되지 않고 있습니다. 오류 해결 뒤 다시 업로드 하겠습니다.**

## 7. 앤서블 알아 두면 좋은 팁

### 7-1. Debug Module, Debugger

**-Debug란?**

프로그램의 정확성이나 논리적인 오류(버그)를 찾아내는 테스트 과정

지난번 실습에서 이용한 nfs.yml 파일에 디버그 관련 내용을 추가하겠습니다.

- name: Check monitoring status for linux

hosts: CentOS:Ubuntu

strategy: debug

gather\_facts: no

tasks:

- name: linux NFS client

shell: mount | grep nfs\_shared

register: output

- debug:

var: output.stdout\_line

리눅스에서의 shell 모듈을 이용해, mount 명령어를 사용하여 테스트를 진행하는 코드입니다.

shell에 의해 mount 명령어가 실행되고 해당 결과를 저장하는 공간을 설정하고,

저장하는 역할을 register 모듈을 통해 하게 됩니다.

register는 계산 값, 결과 값에 대한 임시보관 상태입니다.

그리고 해당 register를 debug 모듈에서 var에 집어넣습니다. 그러면 debug 모듈에서 해당 값을 출력합니다.

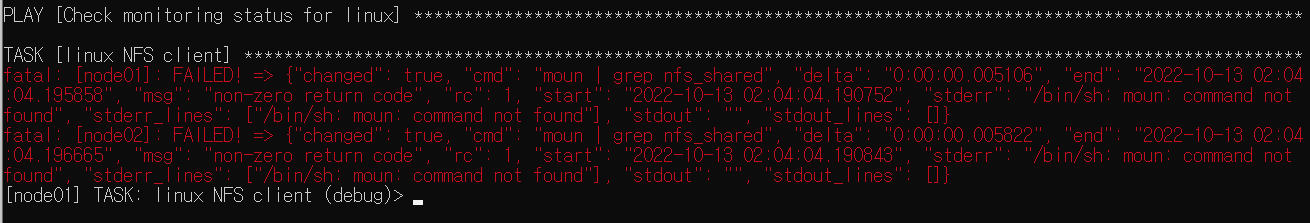
여기서 stdout\_lines는 결과값 중 하나를 말합니다.

또한 "strategy: debug" 가 설정되어 있으면 해당 플레이북에서 오류 발생시 디버거 모드로 전환하게 됩니다.

이제 여기서 디버그 모듈을 테스트해봅니다.

shell: mount | grep nfs\_shared -> shell: moun | grep nfs\_shared

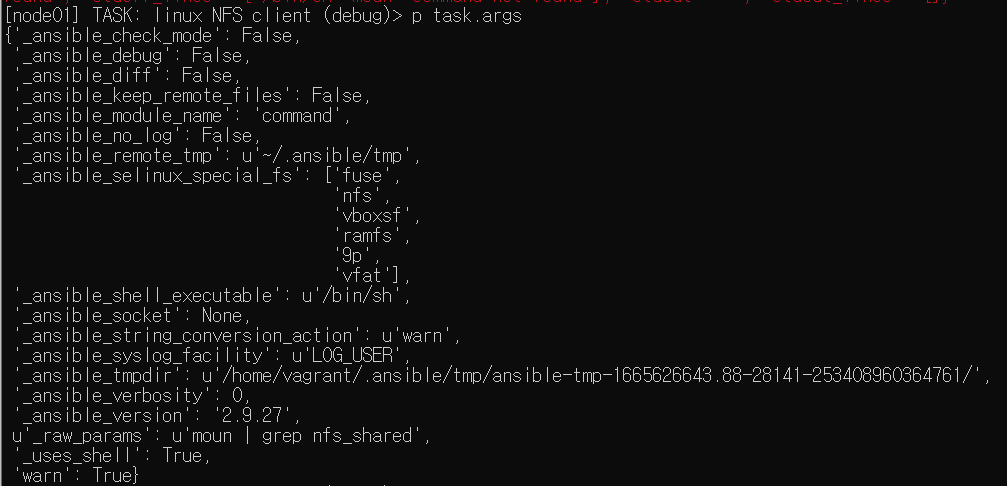
의도적으로 오타를 만든 뒤 위의 nfs.yml 플레이북 파일을 실행시켜 봅니다.



위의 사진과 같이 “/bin/sh: moun: command not fount 와 같은 에러 메시지를 출력하며 디버그 모드로 전환이 됩니다.

이제 디버그 모드에서 다음의 명령어를 실행합니다.

▶p task.args



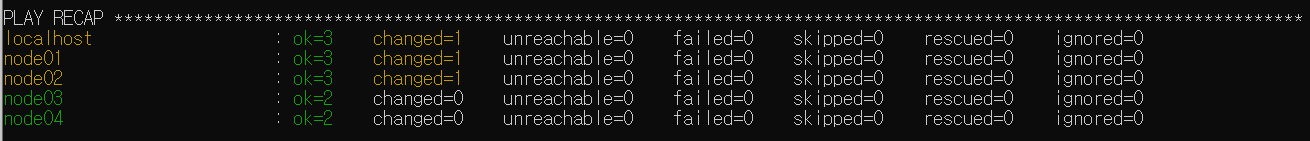
출력 결과를 보면 어느 부분에서 에러가 발생하였는지 알 수 있습니다. 그리고 에러가 발생한 명령어를 수정할 때에는 아래의 명령어를 실행시켜 줍니다.

▶task.args['\_raw\_params']='mount'

그리고 다시 p task.args 명령어를 사용하여 수정한 부분이 정상적으로 바뀌었는지 확인 합니다.

▶redo

그리고 redo 명령어를 사용하면 에러가 발생한 부분의 코드를 재실행 시켜줍니다.



정상적으로 코드가 실행이 됩니다.

## 8. 네트워크 실습을 위한 NX-OSv 랩 구성하기

텍스트이(가) 표시된 사진

자동 생성된 설명

앤서블은 네트워크 또한 관리가 가능합니다. 하지만 실제 네트워크 장비가 없다면 실제로 실습하기가 쉽지 않습니다.

실제 CISCO사의 Nexus스위치 장비에서 사용하는 운영체제인 NX-OS가 있습니다. 이 운영체제를 일반 사용자도 다운로드 받아 테스트해 볼 수 있게 되었습니다.

여기서 사용하게 되는 것이 NX-OSv입니다. NX-OSv를 가상머신에서도 사용할 수 있도록 가상화 한 것입니다.

이번 장에서는 로컬에 NX-OSv를 이용하여 실습 환경을 구성해보겠습니다.

### 8-1. 수동으로 VirtualBox에 NXOSv올리기

설치에 앞서서 OS의 이미지를 구해야 하는데, 이미지는 CISCO사의 홈페이지에서 쉽게 다운로드 받을 수 있습니다.

구글 검색창에 cisco nxosv download 라고 검색을 한 뒤 가장 상단에 나오는 링크로 접속합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

접속을 하게 되면 아래와 같은 화면이 나오는데 최신 버전을 사용해도 상관없지만 편의를 위해 7버전으로 테스트를 진행하겠습니다.

텍스트이(가) 표시된 사진

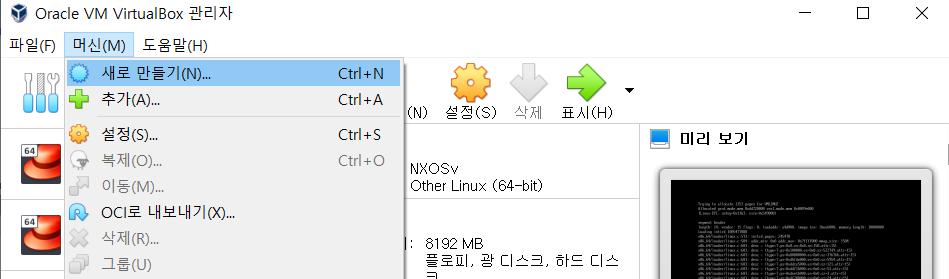
자동 생성된 설명

저희는 vmdk형식의 파일을 다운로드 합니다.

다운로드 클릭 시 로그인 페이지가 나오는데 계정이 없다면 회원가입을 하고 주소를 등록한 뒤 다운로드 진행하시면 됩니다.

다운로드가 완료되었다면 이제 가상머신을 설치해줄 차례입니다.

VirtualBox를 실행시키고 새로운 가상머신을 생성합니다.



텍스트이(가) 표시된 사진

자동 생성된 설명

가상머신 이름은 NXOSv 종류는 Linux/Other Linux(64-bit)로 설정해줍니다.

그리고 메모리는 8GB(8192Mb)로 설정하고 나머지 하드디스크 설정은 디폴트 설정으로 가상머신을 생성합니다.

그리고 가상머신 설정에 들어와 다음의 설정들을 진행하여 줍니다.

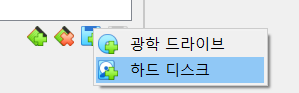
텍스트이(가) 표시된 사진

자동 생성된 설명

EFI사용하기 체크

텍스트이(가) 표시된 사진

자동 생성된 설명



기존 컨트롤러 제거 -> AHCI(SATA)컨트롤러 생성 ->

하드디스크 추가 후 다운로드 받은 vmdk파일 삽입

텍스트이(가) 표시된 사진

자동 생성된 설명

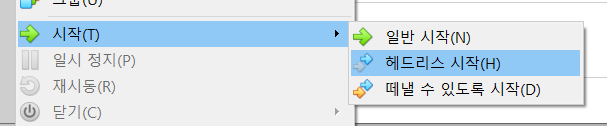
네트워크 설정은 위와 같이 진행해줍니다.

텍스트이(가) 표시된 사진

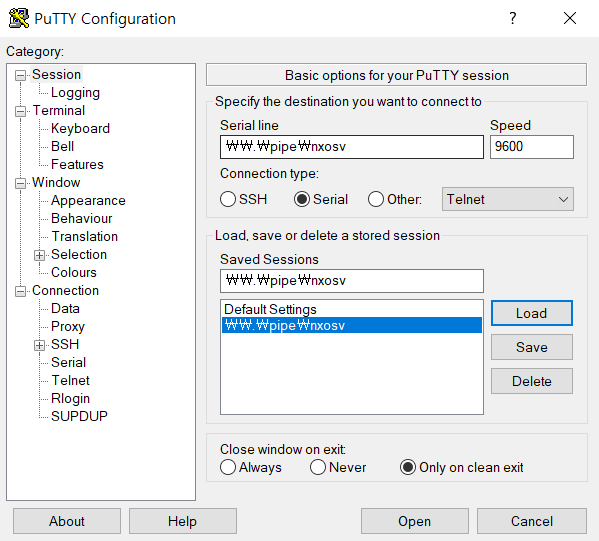
자동 생성된 설명

직렬 포트는 위와 같이 진행해줍니다.

그리고 설정이 완료되었다면



헤드리스 시작으로 가상머신을 시작합니다.



putty는 다음과 같은 설정으로 입력하여 접속한다.

만약 푸티화면에 아무것도 출력이 되지 않는다면 가상머신을 일반시작으로 실행시키고,

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 화면이 나온다면,

**loader > dir**

dir명령어를 입력 후 삽입되어 있는 이미지를 확인한다.

**loader > boot bootflash:nxos.7.xxx (본인이 다운로드 받은 vmdk이미지 버전)**

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 화면이 나온다면 정상적으로 부팅이 된 상태이다. 그리고 이 상태에서 푸티로 접속을 하면

텍스트이(가) 표시된 사진

자동 생성된 설명

이런 형식으로 직렬 통신을 통해 푸티의 화면에 가상머신의 화면이 보이게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

관리자의 비밀번호를 설정하고



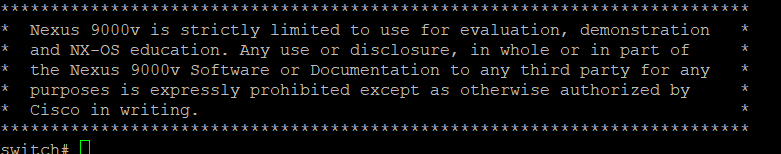
basic configuration dialog의 경우는 사용하지 않음을 선택한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 위와 같이 로그인 창이 출력이 되는데 계정에는 admin을 입력하고 패스워드는

조금 전 본인이 입력한 패스워드를 입력해준다.



로그인 성공 시 위와 같은 화면이 출력이 된다.

텍스트, 텔레비전이(가) 표시된 사진

자동 생성된 설명

**▶conf t**

**▶int mgmt0**

(매니지먼트 포트에 아이피 주소 입력)

텍스트이(가) 표시된 사진

자동 생성된 설명

**▶ip add 본인의 아이피대역.201/서브넷마스크**

**▶no sh**

**▶sh run | in boot**

(현재 부팅 가능한 이미지 확인, 현재 없음)

**▶boot nxos nxos.7.0.3.I7.6.bin 입력**

\*만약 Image Signature verification & bootvar config Failed

에러가 발생한다면 ‘no feature signature-verification’으로 해결합니다.



**▶sh run | in boot**

부트 이미지가 잘 지정이 되었습니다.

조금 전 admin의 비밀번호를 설정할 때 아마 어려운 비밀번호를 입력하라고 강제 받았을 것입니다. 이것을 해제하기 위해

**▶no password strength-check**

**▶username admin password vagrant**

텍스트이(가) 표시된 사진

자동 생성된 설명

vagrant라는 패스워드로 변경이 완료되었습니다.

**▶copy run start**

현재까지 설정한 것들을 복사합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**▶show run int mgmt0**

메니지먼트 포트가 제대로 입력이 되었는지 확인합니다.

이제 cmd 창을 열고 내 컴퓨터에서 가상머신으로 ping이 잘 나가는지 확인합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

여기까지 완료되었다면 정상적으로 VirtualBox에 가상머신이 생성된 것입니다.

### 8-2. 앤서블 서버와 NXOSv서버의 연동

앤서블 서버 구축에 필요한Vagrantfile은 다음의 깃허브 주소에서 다운로드 합니다.

<https://github.com/sysnet4admin/_Lecture_Ansible.adv>

파일을 다운받고 ‘[\_Lecture\_Ansible.adv](https://github.com/sysnet4admin/_Lecture_Ansible.adv)/[ch8](https://github.com/sysnet4admin/_Lecture_Ansible.adv/tree/master/ch8)/**8.2**/’ 경로에 존재하는

Vagrantfile, Ansible\_env\_ready.yml 파일을 본인이 원하는 경로에 가져옵니다.

필요한 경우 파일의 IP주소를 바꾸어 줍니다.

그리고 cmd창을 열어 베이그런트 파일을 실행합니다.

정상적으로 서버가 셋업이 되었다면 ssh를 이용해 서버로 접속합니다.

그리고 nxosv로 핑 테스트를 진행합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음으로 git repository를 ansible-svr4nxosv가상머신으로 클론 합니다.

**▶git clone** [**https://github.com/sysnet4admin/\_Lecture\_Ansible.adv.git**](https://github.com/sysnet4admin/_Lecture_Ansible.adv.git)

텍스트이(가) 표시된 사진

자동 생성된 설명

/home/vagrant/\_Lecture\_Ansible.adv/ch8/8.2 경로에 있는 nxos\_ping\_v2.yml 플레이북 파일을 실행시킵니다. 실행 전 본인 환경에 맞는 ip주소로 내용을 수정합니다.

**▶anp nxos\_ping\_v2.yml -k**

텍스트이(가) 표시된 사진

자동 생성된 설명

정상적으로 ping 테스트가 완료되었습니다.

### 8-3. 네트워크 시스템의 시간 동기화 & 구성파일 백업하기

NSOSv에서 Ansible의 모듈을 검증하기 위해서 네트워크 시스템의 시간을 로컬과 동기화 하고 구성파일을 백업하겠습니다.

NSOSv에서 show clock 명령어를 이용해 시간을 확인해보면,

텍스트, 검은색, 오렌지, 어두운이(가) 표시된 사진

자동 생성된 설명

현재 로컬과는 시간대가 맞지 않는 것이 확인됩니다.

1 ---

2 - name: Setup NXOSv timezone with ntp configuration

3 hosts: NXOSv

4 gather\_facts: no

5

6 tasks:

7 - name: ntp configuration

8 nxos\_config:

9 lines: 'ntp server 172.31.98.10 use-vrf management'

10 - name: timezone to UTC 9 0

11 nxos\_config:

12 lines: 'clock timezone UTC 9 0'

13 save\_when: modified

위의 내용으로 플레이북 파일을 하나 작성 후 실행시켜줍니다.

텍스트, 검은색, 오렌지, 어두운이(가) 표시된 사진

자동 생성된 설명

그 후 다시 네트워크 시스템의 시간을 확인해보면 정상적으로 시간이 바뀌었음을 알 수 있습니다.

그리고 구성파일 백업을 위해 다음의 플레이북 파일을 실행시킵니다.

**/home/vagrant/\_Lecture\_Ansible.adv/ch8/8.3/tftp\_backup\_v2.yml**

그 후 홈 디렉터리로 돌아가 목록을 보면,



**tftp\_shared** 디렉터리가 생성되었습니다. 그리고 해당 디렉터리에

**n9kv-running-config#1**라는 파일이 생성이 되었다면

정상적으로 설정 파일이 백업 된 것입니다.

**■ *참고 자료***

[강의]

★[기초-응용] 다양한 환경을 앤서블(Ansible)로 관리하기 with 베이그런트(Vagrant)

URL: <https://www.inflearn.com/course/ansible-%EC%9D%91%EC%9A%A9/dashboard>

[웹 사이트]

★NXOSv 네트워크 구축 1편

URL: https://youngmind.tistory.com/entry/NXOSv-%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EA%B5%AC%EC%B6%95-1%ED%8E%B8