System

■ vagrant

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

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

        config.vm.define "ansible-node01" do |cfg|
    cfg.vm.box = "centos/7"
    cfg.vm.provider "virtualbox" do |vb|
      vb.name = "Ansible-Node01"
    end
    cfg.vm.host_name = "ansible-node01"
    cfg.vm.network "private_network", ip: "192.168.2.26"
    cfg.vm.network "forwarded_port", guest:22, host: 60011, auto_correct: true, id: "ssh"
    cfg.vm.network "forwarded_port", guest:80, host: 60081
    cfg.vm.synced_folder "../data", "/vagrant", disabled: true
    cfg.vm.provision "shell", path: "bash_ssh_conf_4_CentOS.sh"
  end

        config.vm.define "ansible-node02" do |cfg|
    cfg.vm.box = "centos/7"
    cfg.vm.provider "virtualbox" do |vb|
      vb.name = "Ansible-Node02"
    end
    cfg.vm.host_name = "ansible-node02"
    cfg.vm.network "private_network", ip: "192.168.2.27"
    cfg.vm.network "forwarded_port", guest:22, host: 60012, auto_correct: true, id: "ssh"
    cfg.vm.network "forwarded_port", guest:80, host: 60082
    cfg.vm.synced_folder "../data", "/vagrant", disabled: true
    cfg.vm.provision "shell", path: "bash_ssh_conf_4_CentOS.sh"
  end

        config.vm.define "ansible-node03" do |cfg|
    cfg.vm.box = "centos/7"
    cfg.vm.provider "virtualbox" do |vb|
      vb.name = "Ansible-Node03"
    end
```

```
    cfg.vm.host_name = "ansible-node03"
    cfg.vm.network "private_network", ip: "192.168.2.28"
    cfg.vm.network "forwarded_port", guest:22, host: 60013, auto_correct: true, id: "ssh"
    cfg.vm.network "forwarded_port", guest:80, host: 60083
    cfg.vm.synced_folder "../data", "/vagrant", disabled: true
    cfg.vm.provision "shell", path: "bash_ssh_conf_4_CentOS.sh"
  end

      config.vm.define "ansible-server" do |cfg|
    cfg.vm.box = "centos/7"
    cfg.vm.provider "virtualbox" do |vb|
      vb.name = "Ansible-Server"
    end
    cfg.vm.host_name = "ansible-server"
    cfg.vm.network "private_network", ip: "192.168.2.25"
    cfg.vm.network "forwarded_port", guest:22, host: 60010, auto_correct: true, id: "ssh"
    cfg.vm.synced_folder "../data", "/vagrant", disabled: true
  end


end
```

vagrant : 가상머신 관리 프로그램
가상화 되어 있는 서버를 관리하려고 하니 온프레미스 형식으로는 여러대를 설치하는데 오래 걸림
provider : virtualbox, vmware, AWS 등 가상의 머신을 제공할수 있는 곳을 의미
vagrant box : vagrant를 사용할 때의 운영체제(운영체제를 미리 만들어 놓았음)
vagrant up : virtualbox에서 사용할 수 있는 vagrant box를 가지고 와서 다운 받아서 환경에 맞게 설정함.

ansible-server(IaC)
"프로그래밍형 인프라"라고도 하는 IaC (Infrastructure as Code)는 인프라 구성을 마치 소프트웨어를 프로그래밍하는 것처럼 처리하는 방식이다.
인프라를 관리하기에 수작업을 하기에는 양이 방대하기 때문에 프로그래밍형 인프라를 사용함.
JavaScript, Python, Ansible을 사용하여서 자동화 코드를 이용하여 관리를 함(네트워크든 시스템이든 구분 없음)
Nexus OS를 장착한 Cisco 장비 Ansible로 관리를 함.

IaC는 문법에 맞게 값을 넣어주면 됨.
서비스 세팅 등 사용할 수 있음

DevOps??
Develop Operations
생산성을 높이기 위해서 협업하는 것을 의미
시스템관리자 DevOps
Linux환경에서 개발한 App을 서버에 구축할 수 있고 관리 할 수 있는 사람(서버 관리자 시스템 운영자 의미)
프로그래머는 서버를 잘 모르고 서버는 프로그램을 잘 모름.
프로그래머에게는 코드가 편함.
프로그래머들도 서버에 대한 이해도가 없어도 서버를 만들 수 있음.
프로그래머가 요구하는 것을 서버에 맞게 만들어 놓고 붙이면 됨.
인프라에 유연성 반복성 확장성을 부여함

운영자 한 명이 동일한 코드를 사용하여서 1대 또는 1,000대의 시스템을 구축하고 관리 할 수 있음.

DevOps(IaC)의 장점
비용 절감
배포 속도 향상
오류 감소
인프라 일관성 향상
구성 변동 제거

iac 툴
Chef
Puppet
Red Hat Ansible Automation Platform (링크 : 앤서블을 주목하는 이유)
Saltstack
Terraform
AWS CloudFormation

Vagrant통해서 Ansible-Server를 만들고 Ansible을 설치하여서 관리를 함.
인프라를 접속할 때는 ssh접속을 사용
telnet 접속 보안에 취약해서 사용하지 않음

C:₩HashiCorp>vagrant ssh ansible-server
[vargrant@ansible-server~]$ sudo yum install epel-release –y 입력
[vagrant@ansible-server ~]$ sudo yum install ansible –y 입력
ansible python 기반으로 설치가 됨.
우분투와 Cent OS에서 설치할 때 방법은 같으나 각각 약간의 차이가 있음.

SSH접속 할 때 ECDSA 방식으로 암호화를 시킴.

[vagrant@ansible-server ~]$vagrant ssh@ip 192.168.().()
vagrant 입력 yes하고 빠져나옴
[vagrant@ansible-server ~]$ cd~/.ssh
[vagrant@ansible-server ~]$ ls
authorise_key known_host
처음에는 없음 [vagrant@ansible-server ~]$vagrant ssh@ip 192.168.().()을 입력하면 known_host 생성됨
[vagrant@ansible-server ~]$ cat known_hosts 입력

[vagrant@ansible-server ~]$ ansible all -m ping
192.168.2.27 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
    "unreachable": true
}
192.168.2.26 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
    "unreachable": true
}
192.168.2.28 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
    "unreachable": true
에러뜸

[vagrant@ansible-server ~]$ ansible all -m ping -k
SSH password:vagrant 입력
192.168.2.27 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
192.168.2.28 | SUCCESS => {
    "ansible_facts": {

```
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
192.168.2.26 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

초록색 글자로 띄워져야 함.

```
                                    -m ping 'ping'
ansible-server      ---------------------------------------------->      node01~03
      개인키        <----------------------------------------------      공용키
known_hosts                          'pong'                          authorized_keys
```

SSH 접속할 때 두 가지의 방법
패스워드, 키 인증
authorizied_key에서 ssh_keygen해서 키생성
개인키를 공용키로 만들어서 패스워드 없이 접속을 함.
known_hosts와 authorized_keys가 있어야 패스워드로 접속을 함.
yes를 입력하면 known_hosts 파일을 생성함.

[SSH Key 인증 방식]
ssh-keygen 개인키 공용키

bash_ssh_conf_4_CentOS.sh 파일 → SSH 접속을 관리하는 파일

```
#! /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
```

sed : 파일의 문자열 변환

ansible all –m ping
ansible CentOS –m ping
tail /etc/ansible/hosts

<SSH 키 생성>
[vagrant@ansible-server .ssh]$cd
[vagrant@ansible-server ~]$vi host_inven.lst
192.168.2.26
192.168.2.27

[vagrant@ansible-server ~]$ ansible –i host_inven.lst all –m ping –k
SSH password:vagrant
192.168.2.26 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
192.168.2.27 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[vagrant@ansible-server ~]$ ansible -i host_inven.lst 192.168.2.26 -m ping -k
SSH password:vagrant
192.168.2.26 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}

<ShellScripts 명령어 한 번에 실행하기>
[vagrant@ansible-server ~]$ ls
host_inven.lst
[vagrant@ansible-server ~]$ pwd
/home/vagrant

```
[vagrant@ansible-server ~]$ ansible all -m shell -a 'ls' -k
SSH password:
192.168.2.28 | CHANGED | rc=0 >>

192.168.2.27 | CHANGED | rc=0 >>

192.168.2.26 | CHANGED | rc=0 >>

[vagrant@ansible-server ~]$ ansible all -m shell -a 'pwd' -k
SSH password:
192.168.2.27 | CHANGED | rc=0 >>
/home/vagrant
192.168.2.28 | CHANGED | rc=0 >>
/home/vagrant
192.168.2.26 | CHANGED | rc=0 >>
/home/vagrant

[vagrant@ansible-server ~]$ ansible all -m shell -a 'touch test.txt' -k
SSH password:
[WARNING]: Consider using the file module with state=touch rather than running 'touch'.  If you need to use command
because file is insufficient you can add 'warn: false' to this command task or set 'command_warnings=False' in
ansible.cfg to get rid of this message.
192.168.2.28 | CHANGED | rc=0 >>

192.168.2.26 | CHANGED | rc=0 >>

192.168.2.27 | CHANGED | rc=0 >>

[vagrant@ansible-server ~]$ ansible all -m shell -a 'ls' -k
SSH password:vagrant
192.168.2.28 | CHANGED | rc=0 >>
test.txt
192.168.2.26 | CHANGED | rc=0 >>
test.txt
192.168.2.27 | CHANGED | rc=0 >>
test.txt

[vagrant@ansible-server ~]$ ansible all -m shell -a 'uptime' -k
SSH password:vagrant
```

```
192.168.2.26 | CHANGED | rc=0 >>
 02:35:01 up  1:54,  1 user,  load average: 0.04, 0.04, 0.05
192.168.2.27 | CHANGED | rc=0 >>
 02:35:01 up  1:53,  1 user,  load average: 0.00, 0.01, 0.03
192.168.2.28 | CHANGED | rc=0 >>
 02:35:01 up  1:53,  1 user,  load average: 0.00, 0.01, 0.04

[vagrant@ansible-server ~]$ ansible all -m shell -a 'df -h' -k
SSH password:vagrant
192.168.2.28 | CHANGED | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        237M     0  237M   0% /dev
tmpfs           244M     0  244M   0% /dev/shm
tmpfs           244M  4.5M  240M   2% /run
tmpfs           244M     0  244M   0% /sys/fs/cgroup
/dev/sda1        40G  3.0G   38G   8% /
tmpfs            49M     0   49M   0% /run/user/1000
192.168.2.26 | CHANGED | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        237M     0  237M   0% /dev
tmpfs           244M     0  244M   0% /dev/shm
tmpfs           244M  4.5M  240M   2% /run
tmpfs           244M     0  244M   0% /sys/fs/cgroup
/dev/sda1        40G  3.0G   38G   8% /
tmpfs            49M     0   49M   0% /run/user/1000
192.168.2.27 | CHANGED | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        237M     0  237M   0% /dev
tmpfs           244M     0  244M   0% /dev/shm
tmpfs           244M  4.5M  240M   2% /run
tmpfs           244M     0  244M   0% /sys/fs/cgroup
/dev/sda1        40G  3.0G   38G   8% /
tmpfs            49M     0   49M   0% /run/user/1000
```

ping : 동작확인 모듈
shell : 사용하는 명령어 보내는 모듈

<모듈을 이용한 계정 생성>

```
[vagrant@ansible-server ~]$ su root
Password:vagrant
[root@ansible-server vagrant]# ansible all -m user -a "name=busanit" -k
```

SSH password:
192.168.2.27 | FAILED! => {
"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
192.168.2.26 | FAILED! => {
"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
192.168.2.28 | FAILED! => {
"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
에러뜸
[root@ansible-server vagrant]# ls ~/.ssh
루트의 개인키 생성해주어야 함

ansible all –m ping해서 yes 3번입력한 후 ls ~/.ssh
known_hosts가 생성되어 있음

[root@ansible-server vagrant]# ansible all -m user -a "name=busanit" -k
SSH password:vagrant
192.168.2.27 | CHANGED => {
"ansible_facts": {
"discovered_interpreter_python": "/usr/bin/python"
},
"changed": true,
"comment": "",
"create_home": true,
"group": 1001,
"home": "/home/busanit",
"name": "busanit",
"shell": "/bin/bash",
"state": "present",
"system": false,
"uid": 1001
}
192.168.2.26 | CHANGED => {

```
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1001,
    "home": "/home/busanit",
    "name": "busanit",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1001
}
192.168.2.28 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1001,
    "home": "/home/busanit",
    "name": "busanit",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1001
}
```


정상적으로 됨.

<계정 생성 확인>

```
[root@ansible-server vagrant]# ansible all -m shell -a 'tail -3 /etc/passwd' -k
SSH password:vagrant
192.168.2.27 | CHANGED | rc=0 >>
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
busanit:x:1001:1001::/home/busanit:/bin/bash
192.168.2.28 | CHANGED | rc=0 >>
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
```

busanit:x:1001:1001::/home/busanit:/bin/bash
192.168.2.26 | CHANGED | rc=0 >>
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
busanit:x:1001:1001::/home/busanit:/bin/bash

<계정 삭제>

```
[root@ansible-server vagrant]# ansible all -m user -a "name=busanit state=absent" -k
SSH password:vagrant
192.168.2.27 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "force": false,
    "name": "busanit",
    "remove": false,
    "state": "absent"
}
192.168.2.26 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "force": false,
    "name": "busanit",
    "remove": false,
    "state": "absent"
}
192.168.2.28 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "force": false,
    "name": "busanit",
    "remove": false,
    "state": "absent"
}
[root@ansible-server vagrant]# ansible all -m shell -a 'tail -3 /etc/passwd' -k
SSH password:vagrant
```

192.168.2.28 | CHANGED | rc=0 >>
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
192.168.2.27 | CHANGED | rc=0 >>
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
192.168.2.26 | CHANGED | rc=0 >>
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash

user모듈 사용자 생성 삭제시 사용

<httpd 설치>
[root@ansible-server vagrant]# ansible all -m yum -a 'name=httpd state=present' -k
SSH password:vagrant

[root@ansible-server ~]# vi index.html
[root@ansible-server ~]# ansible all -m copy -a 'src=index.html dest=/var/www/html/index.html' -k
SSH password:vagrant

<http 서비스 시작>
[root@ansible-server ~]# ansible all -m service -a 'name=httpd state=started' -k
SSH password:vagrant

<방화벽 중지하기>
[root@ansible-server ~]# ansible all -m shell -a 'systemctl stop firewalld' -k
SSH password:vagrant

포트를 통해서 연결해 주는 것 Port Forwarding

<http 제거>
[root@ansible-server ~]# ansible all -m yum -a 'name=httpd state=absent' -k
SSH password:vagrant

<모듈 사용 형식>
[계정명@호스트명]# ansible 호스트 –m 모듈명 –a '명령어' -k(-k는 패스워드 인증일 때 필요)
일반 계정이면 Root권한으로 변환해야함.(root로 접속해야 함)

<YAML문법을 이용해서 모듈 동작하기 – nginx 설치 및 삭제>
ansible을 YAML 문법으로 만들어 놓음

<설치>
nginx_install.yml 생성 아래 내용 추가

```
---
- name: Install nginx on CentOS
  hosts: CentOS
  gather_facts: no(결과 보여줌)
  become: yes(루트권한)

  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
      get_url: url=https://www.nginx.com dest=/usr/share/nginx/html/ mode=0644
    - name: start nginx web server
      service: name=nginx state=started
```

```
[root@ansible-server ~]# ansible-playbook nginx_install.yml -k
SSH password:vagrant
```

ansible-playbook : YAML 파일을 실행시키는 명령어

<제거>

```
[root@ansible-server ~]# vim nginx_remove.yml
```

아래 내용 입력

```
---
- 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
```

```
[root@ansible-server ~]# ansible-playbook nginx_remove.yml -k
SSH password:vagrant
```

<YAML 문법>
2칸 들여쓰기
Key:Value 형식으로 정의
배열 – 로 표시
주석은 #으로 표시합니다.
참/거짓은 true, false외에 yes, no를 지원합니다.
숫자 : 정수 또는 실수를 따옴표(") 없이 사용하면 숫자로 인식합니다.
여러 줄을 표현하는 방법입니다.

"|" 지시어는 마지막 줄바꿈이 포함

YAML
JSON
newlines_sample: |
        number one line
        second line
        last line

"|-" 지시어는 마지막 줄바꿈을 제외

YAML
JSON
newlines_sample: |-
        number one line

        second line

        last line

">" 지시어는 중간에 들어간 빈줄을 제외

YAML
JSON
newlines_sample: >
        number one line

        second line

        last line

주의사항

띄어쓰기
key와 value사이에는 반드시 빈칸이 필요합니다.

error (not key-value, string) → key:value
ok → key: value

문자열 따옴표
→대부분의 문자열을 따옴표 없이 사용할 수 있지만 :가 들어간 경우는 반드시 따옴표가 필요합니다.

error → windows_drive: c:
ok → windows_drive: "c:", windows_drive: 'c:'

참고
json2yaml
JSON에 익숙하신 분들을 위한 YAML 변환 사이트입니다.
https://www.json2yaml.com/(opens new window)

yamllint
YAML 문법을 체크하고 어떻게 해석하는지 결과를 알려줍니다.

<똑같이 한 번에 하는 방법>

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

N = 3

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

# CentOS
(1..N).each do |i|
    config.vm.define "ansible-node0#{i}" do |cfg|
      cfg.vm.box = "centos/7"
      cfg.vm.provider "virtualbox" do |vb|
      vb.name = "Ansible-Node#{i}"
      end
      cfg.vm.host_name = "ansible-node#{i}"
      cfg.vm.network "private_network", ip: "192.168.2.1#{i}"
      cfg.vm.network "forwarded_port", guest:22, host: "6001#{i}", auto_correct: true, id: "ssh"
      cfg.vm.network "forwarded_port", guest:80, host: "6008#{i}"
      cfg.vm.synced_folder "../data", "/vagrant", disabled: true
```

```
        cfg.vm.provision "shell", path: "bash_ssh_conf_4_CentOS.sh"
        end
  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"
      end
      cfg.vm.host_name = "ansible-server"
      cfg.vm.network "private_network", ip: "192.168.2.10"
      cfg.vm.network "forwarded_port", guest:22, host: 60010, auto_correct: true, id: "ssh"
      cfg.vm.synced_folder "../data", "/vagrant", disabled: true
      cfg.vm.provision "shell", inline: "yum install epel-release -y"
      cfg.vm.provision "shell", inline: "yum install ansible -y"
      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
```

ansible_env_ready.yml 만들기

```
---
- name: Setup for the Ansible's Environment
  hosts: localhost
  gather_facts: no

  tasks:
    - name: Add "/etc/ansible/hosts"
      blockinfile:
        path: /etc/ansible/hosts
        block: |
          [CentOS]
          192.168.56.11
          192.168.56.12
          192.168.56.13

    - name: Install sshpass for Authentication
      yum:
        name: sshpass
```

```
    state: present

- name: Create vim env's directories & files
  shell: "{{ item }}"
  with_items:
    - "mkdir -p /home/vagrant/.vim/autoload /home/vagrant/.vim/bundle"
    - "touch /home/vagrant/.vimrc"
    - "touch /home/vagrant/.bashrc"

- 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:
    path: /home/vagrant/.vimrc
    line: "{{ item }}"
  with_items:
    - "set number"
    - "execute pathogen#infect()"
    - "syntax on"

- name: Configure Bashrc
  lineinfile:
    path: /home/vagrant/.bashrc
    line: "{{ item }}"
```

```
        with_items:
          - "alias ans='ansible'"
          - "alias anp='ansible-playbook'"

C:₩HashiCorp> vagrant provision
→ provision 스크립트를 추가시킴

[vagrant@ansible-server ~]$ tail /etc/ansible/hosts
# leading 0s:

## db-[99:101]-node.example.com

# BEGIN ANSIBLE MANAGED BLOCK
[CentOS]
192.168.2.11
192.168.2.12
192.168.2.13
# END ANSIBLE MANAGED BLOCK

block : 여러 줄
block: | 표시 여러줄 들어감, > 표시 한줄에 들어감

shell: "{{ item }}"
item : 변수
{{}}쓰는 문법 → Jinja2 문법

curl = url copy

[vagrant@ansible-server ~]$ ls -al
total 24
drwx------. 4 vagrant vagrant  150 Apr 11 05:28 .
drwxr-xr-x. 3 root    root      21 Apr 30  2020 ..
-rw-rw-r--. 1 vagrant vagrant 1650 Apr 11 05:29 ansible_env_ready.yml
-rw-------. 1 vagrant vagrant    5 Apr 11 05:28 .bash_history
-rw-r--r--. 1 vagrant vagrant   18 Apr  1  2020 .bash_logout
-rw-r--r--. 1 vagrant vagrant  193 Apr  1  2020 .bash_profile
-rw-r--r--. 1 vagrant vagrant  280 Apr 11 05:29 .bashrc
drwx------. 2 vagrant vagrant   29 Apr 11 05:25 .ssh
drwxr-xr-x. 4 root    root      36 Apr 11 05:27 .vim
-rw-r--r--. 1 root    root      47 Apr 11 05:29 .vimrc
[vagrant@ansible-server ~]$ vim .vimrc
```

```
[vagrant@ansible-server ~]$
[vagrant@ansible-server ~]$ vim .vimrc
[vagrant@ansible-server ~]$
[vagrant@ansible-server ~]$ vim ansible_env_ready.yml
[vagrant@ansible-server ~]$ echo @SHELL
@SHELL
[vagrant@ansible-server ~]$ echo %SHELL
%SHELL
[vagrant@ansible-server ~]$ echo $SHELL
/bin/bash
[vagrant@ansible-server ~]$ ll
total 4
-rw-rw-r--. 1 vagrant vagrant 1650 Apr 11 05:29 ansible_env_ready.yml
```

ssh pass??
sshpass란 다른 컴퓨터에 바로 ssh 연결을 할 수 있고, 연결된 컴퓨터에서 명령어를 실행할 수 있는 기능을 말합니다. 즉, 다른 컴퓨터에 ssh연결을 한 뒤, 명령어까지 실행할 수 있는 기능입니다.

sshpass -p 'ssh접속비밀번호' ssh 'ssh접속계정'@'ssh접속주소' '명령어'

<시간보기>

```
[vagrant@ansible-server ~]$ timedatectl
      Local time: Mon 2022-04-11 06:46:01 UTC
  Universal time: Mon 2022-04-11 06:46:01 UTC
        RTC time: Mon 2022-04-11 06:45:59
       Time zone: UTC (UTC, +0000)
     NTP enabled: yes
NTP synchronized: yes
 RTC in local TZ: no
      DST active: n/a
```

<타임존 보기>

```
[vagrant@ansible-server ~]$ ansible all -m shell -a "timedatectl | grep 'Time zone'" -k
SSH password:vagrant
192.168.2.13 | CHANGED | rc=0 >>
       Time zone: UTC (UTC, +0000)
192.168.2.12 | CHANGED | rc=0 >>
       Time zone: UTC (UTC, +0000)
192.168.2.11 | CHANGED | rc=0 >>
       Time zone: UTC (UTC, +0000)
```

<KST로 변경>

```
[vagrant@ansible-server ~]$ timedatectl set-timezone "Asia/Seoul"
==== AUTHENTICATING FOR org.freedesktop.timedate1.set-timezone ===
Authentication is required to set the system timezone.
Authenticating as: root
Password:vagrant
==== AUTHENTICATION COMPLETE ===
[vagrant@ansible-server ~]$ timedatectl
timedatectl
      Local time: Mon 2022-04-11 15:52:14 KST
  Universal time: Mon 2022-04-11 06:52:14 UTC
        RTC time: Mon 2022-04-11 06:52:13
       Time zone: Asia/Seoul (KST, +0900)
     NTP enabled: yes
NTP synchronized: yes
 RTC in local TZ: no
      DST active: n/a
```

vim timezone.yml 생성후 아래 내용 입력

```
1 ---
2 - name: Setup CentOS timezone
3   hosts: CentOS
4   gather_facts: no
5   become: yes
6
7   tasks:
8     - name: set timezone to Asia/Seoul
9       timedatectl: name=Asia/Seoul
```

```
[vagrant@ansible-server ~]$ anp timezone.yml -k
SSH password:vagrant

[vagrant@ansible-server ~]$ ansible all -m shell -a "timedatectl | grep 'Time zone'" -k
SSH password:vagrant
```

timezone이 서울로 되어 있음.