ansible 특징

- ansible 은 agent 가 필요없다.
- ansible은 복잡한 스크립트를 작성하는 대신 더 높은수준의
플레이를 만들어 호스트 또는 호스트 그룹이 특정상태에 있도록 한다.
- ansible 작업, 플레이, 플레이북은 멱등이어야 한다.
- ansible은 작업을 위해 모듈을 사용한다.
- ansible은 플러그인을 사용한다. 플러그인은 ansible에 추가할수 있는 코드로,
확장하여 새로운 용도와 플랫폼에 맞게 조정할수 있다.

* ansible 설치
control node 에 설치
 # yum -y install ansible
ansible 을 사용할 계정생성 및 sudo 권한 부여
 # useradd devpos
 # passwd devops
 # echo "devops ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/devops
devops 계정으로 login 후 공개키 생성(pass phrase 를 입력안하게 설정)
$ *ssh-keygen* ; 물어보는것들은 전부 엔터키 쳐서 넘어감

---

managed node (node1,node2)
작업을 처리할 계정생성 및 sudo 권한 부여
 # useradd devpos (* managed node 의 계정과 달라도 된다)
 # passwd devops
 # echo "devops ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/devops
---
control node(workstation)
devops 계정으로 생성한 공개키를 managed node 의 devops 계정으로 전송
$ ssh-copy-id devops@node1
$ ssh-copy-id devops@node2

임의의 디렉토리 생성후 ansible 설정 test
$ cd /home/devops
$ mkdir test
$ cd test
test 디렉토리에 ansible 설정파일과 inventory 파일 작성
ansible.cfg 파일내용을 아래처럼 편집기로 작성
[defaults]
inventory = ./inventory
remote_user = devops
ask_pass = false (또는 no)
[privilege_escaltion]
become = true (또는 yes)
become_user = root
become_method = sudo

inventory 파일내용을 아래처럼 편집기로 작성
[webservers] <-- 그룹명
node1
node2
[dbservers]

node3 * host는 /etc/hosts 파일에 등록되어 있는 이름이어야 한다.
node4

* 그룹을 중첩하여 작성할수도 있다.
[webapps:chidren]
dbservers <-- dbservers 와 webservers 는 호스트명이 아니라 그룹명이다.
webservers

인벤토리 test
$ ansible node1 --list-hosts
$ ansible webservers --list-hosts

ansible managed node test
$ ansible all -m ping
- 이 명령의 결과가 성공하면 ansible 환경설정은 ok.

동적인벤토리
- 외부 데이터베이스에서 제공하는 정보를 사용하여 동적으로 생성할수 있다.

ansible 설정파일
/etc/ansible/ansible.cfg - 기본설정파일
~/.ansible.cfg
./ansible.cfg

설정파일을 참조하는 순서는 현재디렉토리의 ansible.cfg 를 가장 먼저 참조
두번째는 $HOME/.ansible.cfg
마지막으로 /etc/ansible/ansible.cfg 참조
* $ANSIBLE_CONFIG 환경변수를 사용할수도 있다.(설정파일보다 환경변수를 가장먼저 참조한다)

ansible 설정파일은 여러개의 섹션으로 구성된다.
섹션은 '[ ]' 내에 정의되어 있다.
일반적으로 자주 사용되는 섹션은 [defaults] 섹션과 [privilege_escalation] 섹션이다.

ansible 설정 예

[defaults]
inventory = ./inventory
remote_user = devops
ask_pass = false

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

ask-pass : 원격사용자로 연결할때 암호요구
become : 관리대상호스트에서 작업에 대한 권한 에스컬레이션을 활성화하거나 비활성화 한다.
become_method = 권한 에스컬레이션 방법, sudo 나 su 등이 있다.
become_user : 관리대상 호스트의 권한에스컬레이션할 계정
become_ask_pass : 관리대상 호스트에서 권한 에스컬레이션에서 암호를 요구할지 여부를 설정

ansible ad-hoc
- 애드혹 명령은 나중에 다시 실행하기 위해 저장할 필요가 없는 하나의 ansible 작업을 신속하게 실행하는 방법, 플레이북 작성없이 명령행에서 직접실행하는 단순한 작업방식
- 간단하고 신속한 작업을 원할때 사용

ex)
$ ansible -m user -a 'name=user100 uid=2000 state=present' servera.example.com

대부분의 모듈은 멱등이다. 이미 올바른 상태라면 아무작업도 하지 않는다

ansible all -m command -a 'hostname'
- command 모듈은 관리대상 호스트의 shell에서 처리되지 않는다. 셸환경변수나 리디렉션, 파이프등 셸에서 기본적으로 제공하는 것들을 사용할수 없다.

yaml syntax

기본자료형
- 리스트 및 딕셔너리(해시)로 표현

```
---
# A list of tasty fruits
- Apple
- Orange
- Strawberry
- Mango
...

# An employee record
martin:
    name: Martin D'vloper
    job: Developer
    skill: Elite

# Employee records
-  martin:
    name: Martin D'vloper
    job: Developer
    skills:
      - python
      - perl
      - pascal
-  cane:
    name: Tabitha Bitumen
    job: Developer
    skills:
      - lisp
      - fortran
      - erlang

---
```

yaml 문법(playbook 작성)

주석처리는 # 으로 한다

```
# playbook test
- hosts: servera.example.com  # servera - minimal installed
```

문자열

```
This playbook just for test
'This playbook just for test'
"This playbook just for test"
```

```
msg : |  <-- 각 라인을 개행(줄바꿈)처리한다
   first line - test message
   second line - test message
   thrid line - test message
```

```
msg: > 각 라인을 개행(줄바꿈)처리하지 않는다.
   first line - test message
   new message and
   new message
```

```
# yaml  list

hosts:
   - server1
   - server2
   - server3
# 아래처럼 한줄로 표기할수도 있다.
hosts: [ server1, server2, server3 ]
```

yaml 딕셔너리

```
yum:
   name: httpd
   state: present
# 아래처럼 한줄포 표기할수도 있다.
yum: {name:httpd, state: present}
```

변수

변수이름 규칙은 대체로 다른언어와 비슷하다.
변수이름은 문자로 시작해야 하고 특수문자는 '_' 만 사용가능하다.

```
옳바른 변수이름
--------
foobar
foo_bar
foo_bar2
```

잘못된 변수이름
----
foo bar
foo-bar
1st_foobar
foo.bar

변수정의가 가능한곳
- inventory 내에서 변수정의
- playbook 에서 vars 블록에서 정의
- palybook 에서 var_files 로 외부파일에서 정의

1. 플레이북에서 변수정의
```
- hosts: all
  vars:
    service: httpd
    service_port: 80
```

2. 외부파일에서 변수정의
```
- hosts: all
  vars_files:
    - vars/service.yml
```

외부파일은 아래처럼 딕셔너리 형태여야 한다.
vars/service.yaml

```
service: httpd
service_port: 80
```

playbook 에서 변수사용
- 정의된 변수를 사용할때는 이중괄호로 묶어서 사용한다.
  - {{service}}
  - {{service_port}}

example)
```
# 변수정의
vars:
  service: httpd

tasks:
 # 변수사용
  - name: install {{httpd}} packages
    yum:
      name: "{{httpd}}"
      state: present
```

* key 에 대한 value 가 변수로 시작하는 경우에는 value 와 변수를 구분하기 위하여
반드시 변수가 포함된 이중괄호를 큰따옴표를 붙여야 한다.

---
3. inventory 내에서 변수정의

host 변수 및 그룹 변수
[web_servers]
server2.example.com service=httpd

[db_servers]
server3.example.com
server4.example.com

[db_servers:vars]
service=mariadb
---
inventory 파일에 변수를 등록하면 inventory 가 복잡해지고 관리하기 좋지 않다.
그래서 group_vars 및 host_vars 디렉토리를 사용하는것이 권장된다.

~/sample/group_vars/web_servers <-- 그룹변수의 디렉토리명은 group_vars 여야하고 그아래 파일명은 반드시
service: httpd   inventory 에 설정되어 있는 그룹명과 일치해야 한다.
~/sample/group_vars/db_servers
service: vsftpd

~/sample/host_vars/server2.example.com <-- 호스트 변수의 디렉토리명은 host_vars 여야하고 그아래 파일명은
service: httpd   반드시 호스트 이름과 일치해야 한다
~/sample/host_vars/server3.example.com
service: vsftpd
---

command line 에서 변수정의

$ ansible-playbook server2.example.com test.yml -e "service=httpd" ; adhoc

배열변수

user1_uid: 1000
homedir: /home/user1
user2_uid: 1001
homedir: /home/user2
user3_uid: 1002
homedir: /var/user3
...
==> 위의 변수들을 아래처럼 배열로 바꿔서 사용할 수 있다.
* python 에서는 아래처럼 정의할수 있다.
users = { 'user1':{'uid':1000,'homedir':'/home/user1'},user2:{'uid':10001,'homedir':'/home/user2'},..}
users.get('user2') ; key 대한 value 출력

```
users:
  user1:
    uid: 1000
    homedir: /home/user1
  user2:
    uid: 1001
```

```
    homedir: /home/user2
  .....
```

변수사용
users.user1.uid ==> 1000
또는 users['user1']['uid']

users.user2.homedir ==> /home/user2
또는 users['user2']['homedir']

register 변수
- task 의 실행결과를 변수에 저장 (주로 디버깅을 위한 목적으로 사용)

example)

```
---
- name : test
  hosts: all
  tasks:
    - name: install package
      yum:
        name: tftp-server
        state: installed
      register: install_result
    - debug: var=install_result
```

* debug 모듈
디버깅을 하기 위한 목적으로 사용하며 메시지나 변수값을 출력해준다
$ ansible-doc dubug 참조

매직변수
- ansible 에 의해 자동으로 설정되는 변수
hostvars
- 관리대상 호스트의 변수를 얻을때 사용
group_names
- 현재 호스트가 속한 그룹리스트
groups
- inventory 내의 전체 그룹 및 모든호스트
inventory_hostname
- inventory 에 실제기록되어 있는 호스트 이름

팩트변수
- ansible 이 managed node(host) 에서 자동으로 검색한 변수
팩트에는 다음과 같은 정보가 포함되어 있다.

호스트이름
커널버전
환경변수
CPU 정보
메모리정보
디스크정보
네트워크정보
운영체제버전

IP 주소
팩트는 managed node 의 상태를 파악하고 해당 상태에 따라서 여러가지 조치를 하기위한 방법

adhoc 명령어로 팩트수집
$ ansible server1 -m setup

특정 팩트변수만 수집하려면
{{ ansible_facts['devices']['xvda']['model'] }}
{{ansible_factc['nodename']}}

이런식으로 사용가능하다.

* 플레이북이 실행될때 팩트를 수집하지 않을려면
- hosts: all
  gather_facts: no

ansible playbook

- 플레이는 순서가 지정된 작업집합
- 인벤토리에서 선택한 호스트만 실행가능
- 1개이상의 플레이가 필요하다
- yaml 형식으로 작성
- 일반적으로 확장자는 yaml 또는 yml 로 한다
- 들여쓰기 규칙은 엄격하게 지켜야 한다.
- 동일한 수준의 계층은 들여쓰기가 같아야 한다.
- 하위 항목은 상위항목보다 들여쓰기가 되어야 한다.
- 빈줄은 얼마든지 사용해도 된다.
- 들여쓰기는 스페이스만 사용해야하고 탭은 사용할수 업다.
    그러나 vim 환경설정으로 탭을 스페이스키처럼 사용할수 있다.
    vim 편집기로 플레이북을 편리하게 사용하고 싶으면 환경설정을
    아래처럼 하면 된다.($HOME/.vimrc)
    autocmd FileType yaml setlocal ai ts=2 sw=w et
- 플레이북의 시작은 '---' 로 시작한다.(생략가능하다)
- 플레이북의 종료는 '...' 이지만 일반적으로 거의 생략한다.
ex) adhoc 명령어
$ ansible servera.example.com -m yum -a 'name=telnet-server state=present'

위에 ad-hoc 명령을 플레이북으로 작성하면

```
---
- name: test playbook
  hosts: servera.example.com
  tasks:
    - name: installed telnet-server
      yum:
        name: telnet-server
        state: present
...
```

--- 생략가능하지만 일반적으로 사용하는것을 권장한다.
-- name 생략가능하지만 사용하는것이 플레이북을 이해하거나 디버깅할때 도움이 된다.

... playbook 의 끝을 나타내는것으로 일반적으로 생략한다.

hosts 는 생략불가
tasks 는 경우에 따라서 생략가능하다. 하나이상의 작업이 등록되어 있는경우는 생략불가
tasks 는 관리대상 호스트에서 실행할 작업을 순서대로 나열하는 플레이
목록에 있는 각 작업은 딕셔너리로 표현하며 키: 값 형태이다.

- name: installed telnet-server
task 아래의 name 은 플레이의 목적을 설명하기 위한것으로 이해하기 쉽게 지정하는게 좋다.
yum: 실행할 ansible 파이썬 모듈이다. 이 모듈의 인수는 딕셔너리 형태이다.
  name: telnet-server
여기서 name 은 설명을 위한것이 아니라 yum 패키지 관리도구로 작업할 패키지
이름이며 파이썬모듈 yum 의 옵션이다.
state : present
state 는 파이썬모듈 yum 옵션이다. value 인 present 는 설치가 되어 있어야 한다는것을
의미한다.

플레이북 실행
ansible-playbook test.yaml

플레이북이 실행이 되면 플레이 및 작업에 설정된 name 이 화면에 출력된다.
맨 아래쪽에 changed 표시는 플레이가 수행되고 난후 managed node 가 변경이 되었는지를 나타낸다.
플레이가 몇번이 반복 실행되더라도 관리대상이 이미 올바른 상태라면 managed node는 변경되지 않는다.
(멱등)

ansible syntax 체크

ansible-playbook --syntax-check test.yaml

ansible dry-run(시뮬레이션 실행)
- 플레이가 실행되지만 managed node 는 실제로 아무런 변화는 없다.
단지 실행되는것처럼 화면에 표시된다.

ansible-playbook -C test.yaml

플레이북 실습 예제

홈디렉토리에 test-playbook 을 생성하고 실습
-> cd ; mkdir test-playbook; cd test-playbook

1. apache 호스트 그룹에는
server.example.com 과 serverb.example.com 두개의 호스트가 등록되어 있다.
2. apache 호스트 그룹에
- 웹서버 설치
- 방화벽 설정(리부팅 하더라도 적용되게 영구설정)
- 서비스 시작(리부팅하는경우에는 자동으로 서비스 시작)

3. 플레이북 실행 완료후 에러가 없으면 웹브라우저로 확인

플레이북 실습 예제 2.

플레이북 실습예제 1이 완료되었으면 아래의 상태가 되도록 플레이북 내용 추가

- 초기페이지를 Hello Apache 가 출력이 되도록 한다.

- 초기페이지는 workstation 의 test-playbook 디렉토리에서 hello apache 내용으로
index.html 파일을 작성해서 사용하도록 한다.

다중 플레이

두개이상의 play 리스트가 포함된 플레이북

```
---
# multi playbook sample

- name: first play
   hosts: servera.example.com
   tasks:
      yum: httpd
      status: present

- name: second play
   hosts: serverb.example.com
   tasks:
      yum: vsftpd
      states: latest
```

제어구문

when

```
tasks:
  - command: /bin/false
    register: result
    ignore_errors: True

  - command: /bin/something
    when: result is failed

  # In older versions of ansible use ``success``, now both are valid but succeeded uses the correct tense.
  - command: /bin/something_else
    when: result is succeeded

  - command: /bin/still/something_else
    when: result is skipped
```

```
tasks:
    - command: echo {{ item }}
```

```
      loop: [ 0, 2, 4, 6, 8, 10 ]
      when: item > 5

-----------------------------------

tasks:
    - name: gather site specific fact data
      action: site_facts
    - command: /usr/bin/thingy
      when: my_custom_fact_just_retrieved_from_the_remote_system == '1234'

-----------------------------------
- hosts: all
  remote_user: root
  vars_files:
    - "vars/common.yml"
    - [ "vars/{{ ansible_facts['os_family'] }}.yml", "vars/os_defaults.yml" ]
  tasks:
  - name: make sure apache is started
    service: name={{ apache }} state=started
-----------------------------------------------

- name: test play
  hosts: all

  tasks:

      - shell: cat /etc/motd
        register: motd_contents

      - shell: echo "motd contains the word hi"
        when: motd_contents.stdout.find('hi') != -1
-------------------------------------------------------

- name: check registered variable for emptiness
  hosts: all

  tasks:

      - name: list contents of directory
        command: ls mydir
        register: contents

      - name: check contents for emptiness
        debug:
          msg: "Directory is empty"
        when: contents.stdout == ""
-------------------------------------------------------
```

loops

loop는 with_list 와 같다.

```
- name: add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  loop:
     - testuser1
     - testuser2


- name: nested loop test
  user:
    name: "{{item.name}}"
    state: present
    groups: "{{item.groups}}"
  with_items:
    - { name: 'user1', groups: 'user1'}
    - { name: 'admin', groups: 'wheel'}
```

nested loop

```
  mysql_user
    name: "{{ item[0] }}"
    priv: "{{ item[1] }}".*:ALL
    append_privs: yes
    password: redhat
  with_netsted
   - [ 'dbuser1', 'dbuser2' ]
   - [ 'testdb','mydb','sampledb']
```

핸들러

- notify 문을 사용하여 명시적으로 호출된 경우에만 실행된다
- 핸들러는 특정작업의 notify문에 나열된 순서가 아니라 핸들러 섹션이 플레이에서 작성된 순서로 항상 실행된다.
- 핸들러는 task 의 모든 작업이 완료된 후에 실행된다.
- notify는 changed 가 있는 경우에만 핸들러에게 알린다.

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    copy:
      src: /var/lib/httpd.conf
      dest: /etc/httpd/httpd.conf
```

```
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

오류처리

플레이북 실행시 위에서 아래로 순차적으로 실행이 되는데실행중 특정 작업이 실패하게 되면 그 아래의 플레이는 더 이상 실행되지 않고 중단된다. 실패한 작업은 무시하고 그 아래의 플레이를 계속 실행하게 하려면 ignore_errors 키워드를 사용하면 된다.

플레이가 실패하게 되면 플레이가 실패하기전에 알림을 받은(notify) 핸들러도 실행되지 않는데
force_handlers 키워드를 사용하면 플레이가 중단되더라도 핸들러는 실행된다.

아래의 소스파일로 ignore_errors / force_handlers test

```
- hosts: myserver2.example.com
    tasks:
    - name: test
      command: /usr/bin/cal
      notify: new debug messages
    - name: pkg installed
      yum:
          name: no_exist_package
          state: present
      - name: debug messages
      debug:
        msg: test debug messages....

  handlers:
    - name: new debug messages
      debug:
        msg: handlers messages......
```

role(역할)구현

- role 컨텐츠를 그룹화하여 다른 사용자와 쉽게 공유할수 있다 (http://galaxy.ansible.com)
- 프로젝트가 크면 role을 사용하면 관리하기 쉽다
- role은 협업작업이 가능하다

```
roles/
└── myproject
    ├── README.md
    ├── defaults
    │   └── main.yml
    ├── files
```

```
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── tasks
    │   └── main.yml
    ├── templates
    ├── tests
    │   ├── inventory
    │   └── test.yml
    └── vars
        └── main.yml
```

defaults: 역할변수의 기본값이 설정되어 있는 디렉토리
files: 역할작업에서 참조하는 정적파일이 있는 디렉토리
handlers: 역할의 핸들러가 정의 되어 있는 디렉토리
meta: 작성자,라이센스,플랫폼 등의 역할종속성을 포함한 역할에 대한 정보가 저장되어 있는 디렉토리
templates: jinja2 템플릿이 있는 디렉토리
tests: 역할을 테스트할대 사용하는 인벤토리가 있는 디렉토리
vars: main.yml 파일에 역할변수의 값을 정의하는 디렉토리

```
- hosts: all
  roles:
    - role1
    - role2
```

role 실습
아래처럼 설정파일과 role 디렉토리를 생성

```
.
├── ansible.cfg
├── inventory
├── roles
│   └── motd
│       ├── defaults
│       │   └── main.yml
│       ├── tasks
│       │   └── main.yaml
│       └── template
│           └── motd.j2
└── use-motd-role.yml
```

```
 cat roles/motd/tasks/main.yaml
---
- name: deliver motd file
  template:
    src: template/motd.j2
    dest: /etc/motd
    owner: root
    group: root
    mode: 0444
```

```
$ cat roles/motd/template/motd.j2
This is the system {{ansible_hostname}}.
```

Todays' date is: {{ansible_date_time.date}}.
Only use this system with permission.
You can ask {{ system_owner }} for access.

* motd.j2 ; jinja2 파일
jinja2 파일은 template 모듈이 처리한다.

```
 cat roles/motd/defaults/main.yml
---
system_owner: user@server2.example.com

cat ./use-motd-role.yml
- name: use motd role playbook
  hosts: myserver2.example.com

  roles:
    - motd
```

$ ansible-playbook use-motd-role.yml