**설치**

**0. Settings**

[Server Spec]  
CPU - Core 2  
Memory - 4096 MB  
HDD - 100 GB  
Network - 외부 통신 설정 필수  
  
\* Ansible은 controller 서버에만 설치 필요 \*  
[Controller Server 요구 사항]  
Python - ver 2.6 or 3.5 이상 설치  
  
[Remote Server 요구 사항]  
ssh로 로그인 가능한 root or sudo 권한 계정 필요  
네트워크상 통신 가능한 상태여야 함

**1. Ansible Install**

**[Rhel/CentOS]**  
yum 활용 위해 Redhat Subscription 활성화 및 EPEL Repository 설치 필요

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | $ yum install -y ansible  $ ansible --version  ansible 2.9.1    config file = /etc/ansible/ansible.cfg    configured module search path = [u'/home/ansible/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']    ansible python module location = /usr/lib/python2.7/site-packages/ansible    executable location = /bin/ansible    python version = 2.7.5 (default, Jun 11 2019, 14:33:56) [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] |  |

\* python 기반으로 개발되었기 때문에 pip 활용한 설치 가능 \*

|  |  |  |
| --- | --- | --- |
| 1 | $ pip install ansible | [cs](http://colorscripter.com/info#e) |

**[Ubuntu]**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | $ apt-add-repository ppa:ansible/ansible  $ apt-get update  $ apt-get install ansible  $ ansible --version | [cs](http://colorscripter.com/info#e) |

**※ 페쇄형 network 환경의 경우, Local Repository 구성 필요 ※**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | $ curl http://[ local\_repository\_server\_ip ]/repository/ansible.repo >> /etc/yum.repos.d/ansible.repo  $ yum clean all  $ yum repolist  $ yum update -y  $ sync  $ reboot  $ yum install -y ansible    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**2. Ansible Tower Install**

Ansible Releases Server에서 Tower Install Program을 제공하고 있으며, Private Network로 구성된 고객도 설치 가능 하도록 Bundle Install 버전을 따로 제공.  
Network 환경에 따라 Ansible Server에 접속하여 필요한 Version의 Install Program을 Download.  
● bundle URL : https://releases.ansible.com/ansible-tower/setup-bundle/  
● 일반 URL : https://releases.ansible.com/ansible-tower/setup/

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | $ wget https://releases.ansible.com/ansible-tower/setup-bundle/ansible-tower-setup-bundle-3.7.3-1.tar.gz  $ tar -zxvf ./ansible-tower-setup-bundle-3.7.3-1.tar.gz  $ cd ./ansible-tower-setup-bundle-3.7.3-1.tar.gz  $ cat ./inventory  [tower]  tower    [database]    [all:vars]  ansible\_become=true    admin\_password='password'    pg\_host=''  pg\_port=''    pg\_database='awxr'  pg\_username='awx'  pg\_password='password'    rabbitmq\_username=tower  rabbitmq\_vhost=tower  rabbitmq\_password='password'  rabbitmq\_port=5672  rabbitmq\_cookie=cookiemonster    rabbitmq\_use\_long\_name=true    $ sh ./setup.sh  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

> 설치 완료 후 https://[ Ansible Tower IP Address ] 로 접속 가능  
> 초기 UserName : admin  
> 초기 Password : ansible tower 설치 디렉토리 경로의 inventory 파일에 지정한 admin\_password 값  
> 최초 접속 시 Tower License 등록 필요  
  
\* Ansible Tower 는 tower 자체 upgrade만 지원하므로, 향후 upgrade 시 ansible engine 별도 upgrade 필요 \*

**운영**

**Ansible 운영 - 1. 초기 설정**

**1. Hosts File Setting**

Ansible 클러스터를 구성하고자 하는 서버 hostnmae과 IP 주소 정보 업데이트

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7 | $ cat /etc/hosts  127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4  ::1         localhost localhost.localdomain localhost6 localhost6.localdomain6    [ ip\_address ] tower  [ ip\_address ] serverA  [ ip\_address ] serverB  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**2. SSH-KEY Setting**

Ansible은 SSH 접속을 기반으로 원격 서버들에게 명령을 전달하기 때문에, Controller 서버와 Remote Server 간 SSH Key가 공유되어야 함.

|  |  |  |
| --- | --- | --- |
| 1  2 | $ ssh-keygen  $ ssh-copy-id [remote-host-user]@[remote-host-ip] | [cs](http://colorscripter.com/info#e) |

**3. Inventory File Setting**

Inventory : 관리 대상이 될 host list를 나열하거나 그룹화한 파일. inventory 파일이 있어야 관리 대상을 정의하고 작업 수행 가능  
vars를 활용하여 대상 그룹 및 host의 전역 변수 정의 가능.  
Ansible Default Inventory 파일 : /etc/ansible/hosts  
default inventory or 별도 inventory 파일 작성 후 활용 가능

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | $ cat /etc/ansible/hosts  # This is the default ansible 'hosts' file.  #  # It should live in /etc/ansible/hosts  #  #   - Comments begin with the '#' character  #   - Blank lines are ignored  #   - Groups of hosts are delimited by [header] elements  #   - You can enter hostnames or ip addresses  #   - A hostname/ip can be a member of multiple groups    # Ex 1: Ungrouped hosts, specify before any group headers.    . . . ( 중 략 ) . . .    $ cat ./inventory  [servers]  serverA  serverB    [servers:vars]  ansible\_connection=ssh  ansible\_user=root  ansible\_ssh\_pass=root    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**4. Ansible Ping Test**

inventory 목록의 server 접속 정상 여부 확인  
  
**\* default inventory 파일 활용 시 \***

|  |  |  |
| --- | --- | --- |
| 1 | $ ansible all -m ping | [cs](http://colorscripter.com/info#e) |

**\* 별도 inventory 파일 활용 시 \***

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9 | $ ansible -i inventory all -m ping  serverA | SUCCESS => {      "changed": false,      "ping": "pong"  }  serverB | SUCCESS => {      "changed": false,      "ping": "pong"  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**Ansible 운영 - 2. Playbook 활용 및 예제**

**1. Playbook**

Ansible은 Playbook을 활용하여 원격 서버들에 대한 설정과 배포를 관리함.  
YAML 포맷으로 표현되며 각각의 playbook은 한 개 이상의 play로 구성됨.  
단독으로 사용되는 것이 아닌 inventory와 playbook의 조합으로 수행.  
  
즉, inventory 파일에서 정의한 대상 서버들이 무엇을 수행할 것인지 playbook을 통해 정의함.  
  
**\* playbook 작성 시 주의 사항 \***  
● YAML 파일 작성 시 들여쓰기는 TAB 키가 아닌 Space Bar 키로 할 것  
● {{ 변수 } }가 있는 곳은 " "로 감쌀 것 > "{{ 변수 }}"  
● 계층 구조상 동일한 수준의 요소들은 들여쓰기 동일하게, 하위 항목은 상위 항목보다 들여 써야 함

**2. 예제**

**1) Ping Test**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9 | $ cat ping\_test.yml  ---  -  name: ping test     hosts: all     gather\_facts: false     tasks:     - name: ping       ping:  ... | [cs](http://colorscripter.com/info#e) |

- name : 작업을 설명하는 식별자(생략 가능)  
- hosts : 작업 대상 hosts(inventory 내 hosts 지정)  
- gather\_facts : 작업 시작 전 host의 자원 정보 facts 수집 여부.  
default로 동작하지만 대상 서버 수가 많을수록 수집 시간이 오래 걸리는 단점 등으로 인해 false로 설정하기도 함.  
- tasks : 작업 목록. 하나의 play는 여러 task로 구성되며 순차적으로 하나씩 실행됨.  
각 task는 하나의 모듈만 실행 가능(모든 task는 name을 가지고 있어야 함)

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | $ ansible-playbook -i inventory ping\_test.yml  PLAY [ping test] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    TASK [ping] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [serverA]  ok: [serverB]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  serverA                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  serverB                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**2) File Module Test**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | $ cat file\_module\_test.yml  ---  - name: touch files test    hosts: all    gather\_facts: false    tasks:    - name: create directory      file:        path: /home/ansible/test        state: directory      - name: touch file      file:        path: /home/ansible/test/test.txt        state: touch  ...  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

> file 모듈 활용하여 지정된 경로에 디렉토리와 파일 생성

**3) Conditional Test**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | $ cat conditional test  ---  - name: test    hosts: all    tasks:    - name: create directory      file:        path: /home/ansible/rhel7        state: directory      - name: touch file      file:        path: /home/ansible/rhel7/test.txt        state: touch      when:      - ansible\_distribution == "RedHat"      - ansible\_distribution\_major\_version == "7"  ...  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

> when 옵션을 활용하여 해당 조건을 만족하는 경우에 task 수행

**Ansible 운영 - 3. Role 활용 및 예제**

**1. Role**

불필요하게 중복되는 소스를 제거하고 자주 사용하는 기능을 재사용하기 위해 Role로 생성 > 호출만 하면 됨  
  
[Role 구조 예시]  
README.md  
defaults/  
main.yml  
files/  
handlers/  
main.yml  
meta/  
main.yml  
tasks/  
main.yml  
templates/  
tests/  
inventory  
test.yml  
vars/  
main.yml  
  
- tasks : 해당 role에서 수행할 task 목록. 해당 role을 수행시키는 메인파일은 main.yml  
- handlers : handler 정의  
- defaults : 기본 변수 설정  
- vars : task 수행 시 사용할 변수 정의 (defaults에서 정의한 변수보다 우선순위가 높음)  
- files : 서버에 복사해야 할 파일이나 스크립트 파일 위치  
- templates : jinja2 template  
- meta : roles에 대한 정보, 호환 버전 등 메타 데이터 정의

**2. Ansible-Role 사용 방법**

|  |  |  |
| --- | --- | --- |
| 1  2  3 | $ mkdir roles  $ cd roles  $ ansible-galaxy init [ Role\_name ] | [cs](http://colorscripter.com/info#e) |

**3. 예제**

**1) OS 버전별 task 수행**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | .  ├── main\_project.yml  ├── roles  │   └── regular\_inspection  │      ├── tasks  │      │   ├── 01-rhel5\_monthly\_regular\_inspection.yml  │      │   ├── 02-rhel6\_monthly\_regular\_inspection.yml  │      │   ├── 03-rhel7\_monthly\_regular\_inspection.yml  │      │   ├── main.yml  │      └── vars  │          └── main.yml  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

> roles 디렉토리 아래 사용하지 않는 폴더는 삭제 가능

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | $ cat ./main.yml  ---  - name: main\_os\_monthly\_regular\_inspection    hosts: all    gather\_facts: false    pre\_tasks:     - setup:         gather\_subset: min    roles:      - role: os-monthly\_regular\_inspection  ...  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

> 실행할 role 지정

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | $ cat ./roles/os-monthly\_regular\_inspection/tasks/main.yml  - name: OS-Check ( RHEL 5 )    include\_tasks: 01-rhel5\_monthly\_regular\_inspection.yml    when:      - ansible\_distribution\_major\_version == "5"    - name: OS-Check ( RHEL 6 )    include\_tasks: 02-rhel6\_monthly\_regular\_inspection.yml    when:      - ansible\_distribution\_major\_version == "6"    - name: OS-Check ( RHEL 7 )    include\_tasks: 03-rhel7\_monthly\_regular\_inspection.yml    when:      - ansible\_distribution\_major\_version == "7" | [cs](http://colorscripter.com/info#e) |

> when 옵션을 활용해 OS 버전별 task 수행

**Ansible - 권한 에스컬레이션**

Ansible은 권한 에스컬레이션 시스템을 사용하여 루트 권한 또는 다른 사용자의 권한으로 작업을 실행할 수 있습니다.

**1. Become 옵션 사용**

play나 task 내에서 become 옵션을 사용하거나 변수 활용, 혹은 명령 줄에서 직접 사용 가능합니다.  
  
become : 특정 사용자로 전환할지 여부. yes 설정 시 활성화 가능.  
become\_user : 원하는 권한을 가진 사용자로 설정. 즉, 현재 로그인되어 있는 사용자가 아닌, 권한을 에스컬레이션할 사용자.  
( default - root )  
become\_method : 특정 사용자 전환 명령. sudo / su / pbrun / doas / dzdo / ksu 중 선택 가능하며 default는 sudo 이다.  
become\_flags : become\_method 명령 실행 시 함께 줄 옵션 설정.  
  
[코드 예시]

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | - name: Ensure the httpd service is running    service:      name: httpd      state: started    become: yes | [cs](http://colorscripter.com/info#e) |

> root 권한이 필요한 작업 수행 시, 권한 에스컬레이션 허용

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | - name: Run a command as the apache user    command: somecommand    become: yes    become\_user: apache | [cs](http://colorscripter.com/info#e) |

> apache 유저로 작업 수행

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6 | - name: Run a command as nobody    command: somecommand    become: yes    become\_method: su    become\_user: nobody    become\_flags: '-s /bin/sh' | [cs](http://colorscripter.com/info#e) |

> su 명령으로 사용자를 변경하여 작업 수행

**2. become 연결 변수**

각 노드, 그룹에 대해 become 관련 연결 옵션 변수를 다르게 설정할 수 있습니다. 이러한 변수는 인벤토리에 정의하거나 일반 변수로 사용할 수 있습니다.  
  
ansible\_become : become 옵션과 동일함. 특정 사용자로 전환할지 여부  
ansible\_become\_method : 특정 사용자 전환 시 사용할 명령  
ansible\_become\_user : 전환할 사용자.  
ansible\_become\_password : 사용자 전환 시 필요한 권한 상승 암호.  
  
[코드 예시 - inventory]

|  |  |  |
| --- | --- | --- |
| 1 | webserver ansible\_user=manager ansible\_become=yes | [cs](http://colorscripter.com/info#e) |

> webserver 라는 이름의 서버에 현재 manager 사용자로 연결되어 있으며, root 사용자로 전환하여 작업 수행하고 싶을 경우

**3. 명령줄 옵션**

--ask-become-pass, -K : 권한 상승 암호 요청.  
--become, -b : become 옵션으로 작업 수행  
--become-method=BECOME\_METHOD : 사용할 권한 에스컬레이션 방법. ( default - sudo )  
--become-user=BECOME\_USER : 작업을 수행할 사용자.

**Ansible - tags 활용**

전체 플레이북을 실행하는 대신 특정 부분만 실행하는 것이 유용할 경우, Ansible tag를 활용하여 작업을 수행할 수 있습니다.

**1. 개별 task에 tag 추가**

개별 task에 하나 이상의 태그를 적용할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | tasks:  - name: Install the servers    yum:      name:      - httpd      - memcached      state: present    tags:    - packages    - webservers    - name: Configure the service    template:      src: templates/src.j2      dest: /etc/foo.conf    tags:    - configuration | [cs](http://colorscripter.com/info#e) |

> 두 개의 작업에 서로 다른 태그를 지정

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | ---  # file: roles/common/tasks/main.yml    - name: Install ntp    yum:      name: ntp      state: present    tags: ntp    - name: Configure ntp    template:      src: ntp.conf.j2      dest: /etc/ntp.conf    notify:    - restart ntpd    tags: ntp    - name: Enable and run ntpd    service:      name: ntpd      state: started      enabled: yes    tags: ntp | [cs](http://colorscripter.com/info#e) |

>둘 이상의 개별 작업에 동일한 태그 적용

**2. include task에 tag 추가**

include\_\* 작업의 태그는 포함된 task 또는 role 내의 모든 작업이 아니라 include 자체에만 적용됩니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7 | - hosts: webservers    tasks:      - name: Include the bar role        include\_role:          name: bar        tags:          - foo | [cs](http://colorscripter.com/info#e) |

> foo 태그는 bar 라는 이름의 role 내부 작업에는 적용되지 않음  
  
\* 이와 관련된 내용은 4번에서 자세히 다룰 예정 \*

**3. play 전체에 tag 추가**

play 내의 모든 task가 동일한 tag를 가져야 하는 경우, play 전체에 tag 지정이 가능합니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | - hosts: all    tags: ntp    tasks:    - name: Install ntp      yum:        name: ntp        state: present      - name: Configure ntp      template:        src: ntp.conf.j2        dest: /etc/ntp.conf      notify:      - restart ntpd      - name: Enable and run ntpd      service:        name: ntpd        state: started        enabled: yes    - hosts: fileservers    tags: filesharing    tasks:    ... | [cs](http://colorscripter.com/info#e) |

**4. role 에 tag 추가**

role 에 tag 를 추가하는 방법은 다음과 같습니다.  
  
1) roles 내에 tag 지정

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | roles:    - role: webserver      vars:        port: 5000      tags: [ web, foo ] | [cs](http://colorscripter.com/info#e) |

2) import\_role 을 활용하여 tag 지정

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | - hosts: webservers    tasks:      - name: Import the foo role        import\_role:          name: foo        tags:          - bar          - baz        - name: Import tasks from foo.yml        import\_tasks: foo.yml        tags: [ web, foo ] | [cs](http://colorscripter.com/info#e) |

3) role 내의 개별 task에 tag 지정  
role 내에서 일부 task만 실행하거나 건너뛰어야 할 때 사용되는 유일한 방법입니다.  
include\_role 을 활용하여 해당 작업에 tag를 지정해야 합니다.  
하지만 Ansible은 include\_role, include\_task 와 같은 동적 작업에서는 tag 상속이 불가능합니다.  
즉, include를 통해 포함된 파일이나 role 내의 task가 아닌 '포함' 자체에만 tag가 적용됩니다.  
  
이를 해결하기 위해 apply 키워드 또는 block 을 활용할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | - name: Apply the db tag to the include and to all tasks in db.yaml    include\_tasks:      file: db.yml      # adds 'db' tag to tasks within db.yml      apply:        tags: db    # adds 'db' tag to this 'include\_tasks' itself    tags: db  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

> apply 키워드 활용

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | - block:     - name: Include tasks from db.yml       include\_tasks: db.yml    tags: db | [cs](http://colorscripter.com/info#e) |

> block 활용

**5. 특수 태그(Special Tags)**

Ansible은 always와 never 라는 특수한 태그를 갖고 있습니다.  
  
- always : 별도로 skip 명령을 실행하지 않는 한, 해당 태그로 지정된 작업을 무조건 수행.  
- never : 별도로 tag 실행 명령을 실행하지 않는 한, 해당 태그로 지정된 작업을 무조건 skip.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | tasks:  - name: Print a message    ansible.builtin.debug:      msg: "Always runs"    tags:    - always    - name: Print a message    ansible.builtin.debug:      msg: "runs when you use tag1"    tags:    - tag1 | [cs](http://colorscripter.com/info#e) |

> always 태그 활용

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | tasks:    - name: Run the rarely-used debug task      debug:       msg: '{{ showmevar }}'      tags: [ never, debug ] | [cs](http://colorscripter.com/info#e) |

> never 태그 활용

**6. tag 실행 관련 command**

--tags all : 모든 작업 실행, tag 무시 (default)  
--tags [tag1,tag2] : 해당 tag가 있는 작업만 실행  
--skip-tags [tag3,tag4] : 해당 tag가 있는 작업을 제외한 모든 작업을 실행  
--tags tagged : 하나 이상의 tag가 있는 작업만 실행  
--tags untagged : tag가 없는 작업만 실행

|  |  |  |
| --- | --- | --- |
| 1 | ansible-playbook example.yml --tags "configuration,packages" | [cs](http://colorscripter.com/info#e) |

> configuration, packages 태그 작업만 실행

|  |  |  |
| --- | --- | --- |
| 1 | ansible-playbook example.yml --skip-tags "packages" | [cs](http://colorscripter.com/info#e) |

> packages 태그가 지정된 작업을 제외한 모든 작업 실행

**7. tag 사용 결과 미리보기**

role이나 playbook 을 실행할 때, 어떤 task에 어떤 tag가 있는지 모르거나 기억하지 못할 수 있습니다.  
이러한 상황에 tag 사용 결과를 미리 확인 할 수 있는 command가 있습니다.  
  
--list-tags : 사용 가능한 tag 목록 생성  
--list-tasks : --tags / --skip-tags 명령 실행 시 수행될 task를 미리 확인 가능

|  |  |  |
| --- | --- | --- |
| 1 | ansible-playbook example.yml --list-tags | [cs](http://colorscripter.com/info#e) |

> 적용 가능한 모든 tag 확인 가능

|  |  |  |
| --- | --- | --- |
| 1 | ansible-playbook example.yml --tags "configuration,packages" --list-tasks | [cs](http://colorscripter.com/info#e) |

> 실행하고자 하는 tag를 어떤 task가 가지고 있는지 불확실할 경우, 해당 tag가 포함된 task 나열 가능  
  
\* 주의 : 이러한 command는 동적으로 포함된 파일 또는 role 내의 tag나 task를 확인할 수 없습니다.

**Ansible - Loop 활용**

동일한 작업을 여러 번 반복하는 경우 Ansible Loop 를 활용하여 작업을 수행할 수 있습니다. Ansible 에서 Loop를 활용하는 키워드는 loop, with\_\* 두가지가 있습니다.

**1. Loop**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | - name: Add several users    user:      name: "{{ item }}"      state: present      groups: "wheel"    loop:       - testuser1       - testuser2 | [cs](http://colorscripter.com/info#e) |

> 간단한 문자열 목록이 반복되는 loop

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | - name: Add several users    user:      name: "{{ item.name }}"      state: present      groups: "{{ item.groups }}"    loop:      - { name: 'testuser1', groups: 'wheel' }      - { name: 'testuser2', groups: 'root' } | [cs](http://colorscripter.com/info#e) |

> loop 내에서 하위 키를 참조하여 해쉬 목록을 반복할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6 | - name: Register loop output as a variable    shell: "echo {{ item }}"    loop:      - "one"      - "two"    register: echo | [cs](http://colorscripter.com/info#e) |

> loop 출력을 변수로 등록할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7 | - name: Give users access to multiple databases    mysql\_user:      name: "{{ item[0] }}"      priv: "{{ item[1] }}.\*:ALL"      append\_privs: yes      password: "foo"    loop: "{{ ['alice', 'bob'] |product(['clientdb', 'employeedb', 'providerdb'])|list }}" | [cs](http://colorscripter.com/info#e) |

>Jinja2 표현식을 사용하여 복잡한 목록을 반복 할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | - name: Create servers    digital\_ocean:      name: "{{ item.name }}"      state: present    loop:      - name: server1        disks: 3gb        ram: 15Gb        network:          nic01: 100Gb          nic02: 10Gb          ...    loop\_control:      label: "{{ item.name }}" | [cs](http://colorscripter.com/info#e) |

> loop\_control 활용 : 복잡한 데이터 구조를 반복 할 때 작업의 콘솔 출력 표시 제한.

**2. with\_\***

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | - name: add several users    user:      name: {{ item }}      state: present      groups: wheel    with\_items:       - testuser1       - testuser2 | [cs](http://colorscripter.com/info#e) |

> with\_items: 기본 loop 지정. loop 내 변수는 {{ item }} 으로 접근 가능.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9 | - name: give users access to multiple databases    mysql\_user:      name: {{ item[0] }}      priv: {{ item[1] }}.\*:ALL      append\_privs: yes      password: foo    with\_nested:      - [ 'alice', 'bob' ]      - [ 'clientdb', 'employeedb', 'providerdb' ] | [cs](http://colorscripter.com/info#e) |

> with\_nested: 중첩 loop 사용

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | ---  - hosts: all    tasks:      # create some test users      - user: name={{ item }} state=present groups=evens        with\_sequence: start=0 end=32 format=testuser%02x        # create a series of directories with even numbers for some reason      - file: dest=/var/stuff/{{ item }} state=directory        with\_sequence: start=4 end=16 stride=2        # a simpler way to use the sequence plugin      # create 4 groups      - group: name=group{{ item }} state=present        with\_sequence: count=4  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

> with\_sequence: 숫자들의 list를 생성(ascending order)해서 loop를 돌릴 때 사용  
stride: 증가 폭을 지정하는 step 옵션  
format: 출력 형식