



Today's DevNet Workshop

Ansible 소개 및 활용

박 지 호 프로, CPOC Engineer (Korea SP SE Team)

Feb 21, 2024



Index

- IaC Overview
- Ansible 소개
- Materials for today's workflow
- 요약

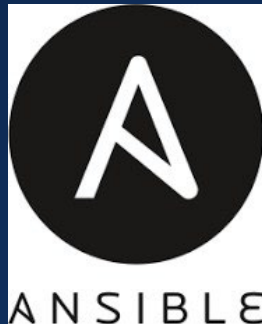


laC Overview



laC (Infrastructure-as-Code) Overview

- 코드를 통해 인프라를 관리
- 구성 편집 및 배포 용이
- 동일 환경에 대한 일관성 보장



laC 의 특징 및 효과

- laC의 특징

- 설정 자동화
- GitHub 내 소스 관리 가능
(인프라 변경 사항 저장, 조회)
- CI / CD Pipeline을 통해 인프라를
자동으로 구성

- laC의 효과

- 오픈소스로 간단하게 모듈 생성
- 에러 방지 및 빠른 속도와 확장성
- 민첩성 / 확장성 / 안정성 에 장점을 가짐

적절한 IaC tool 선택의 필요성

Ansible vs Terraform

➤ 공존 가능성

- Terraform은 VM을 배포한 후, Ansible을 Ad-Hoc task 목적으로 부를 수 있음

➤ 네트워크 프로비저닝 측면에서는...?

- 둘 다 네트워크 인프라 프로비저닝에는 강력함

➤ 다양한 요소를 고려한 신중한 선택이 필요

적절한 IaC tool 선택의 필요성

Triggering playbooks from Terraform

- Provisioner를 사용하여 Terraform을 통해 Ansible playbook 트리거 가능
- 인스턴스 혹은 VM과 독립적으로 실행 가능한 Ansible playbook을 트리거 가능

적절한 IaC tool 선택의 필요성

Triggering playbooks from Terraform

```
resource "vsphere_virtual_machine" "apache-vm" {
  count = 4
  name = "apache-web-server-${count.index + 1}"
  [...]

  provisioner "remote-exec" {
    inline = [
      "touch /home/user/.ssh/authorized_keys",
      "chmod 600 /home/user/.ssh/authorized_keys",
      "echo ${var.ssh-pub-key} >> /home/user/.ssh/authorized_keys"
    ]
  }
  connection {
    type = "ssh"
    user = var.mel_delgado_username
    password = var.mel_delgado_password
    host = "10.200.0.${101 + count.index}"
  }
}

provisioner "local-exec" {
  command = "ansible-playbook -u user -i inventory.yaml apache.yml --vault-password-file ./vault_pass.txt"
}
}
```

remote-exec provisioner,
대상 시스템에 연결하는 방법
지정

local-exec provisioner, local
machine에 Ansible 및
Collections 설치 필요

적절한 IaC tool 선택의 필요성

Triggering playbooks from Terraform

```
resource "null_resource" "attach_esxi_dvs" {  
  depends_on = [  
    module.vmm_domain_vmware["mdr1"]  
  ]  
  
  provisioner "local-exec" {  
    command = "ansible-playbook -i inventory.yaml attach_hosts_vds.yaml"  
    working_dir = "ansible"  
    interpreter = [  
      "/bin/bash", "-c"  
    ]  
    environment = {  
      VMWARE_HOST = "vcsa-mdr1.cisco.com"  
      VMWARE_USER = var.vcenter_username  
      VMWARE_PASSWORD = var.vcenter_password  
    }  
  }  
}
```

Null resource 는 아무것도
프로비저닝하지 않지만
프로비저너에 연결할 수
있음

local-exec provisioner로, local machine은
Ansible과 Collections가 설치되어 있어야 함

생성 및 제거 시간에 대한
프로비저너를 정의하는
옵션

Key Points

- 완벽한 Tool은 없다.
- 각 Tool의 best way를 찾기
- 특정 요소로 접근하여 비교



Ansible 소개

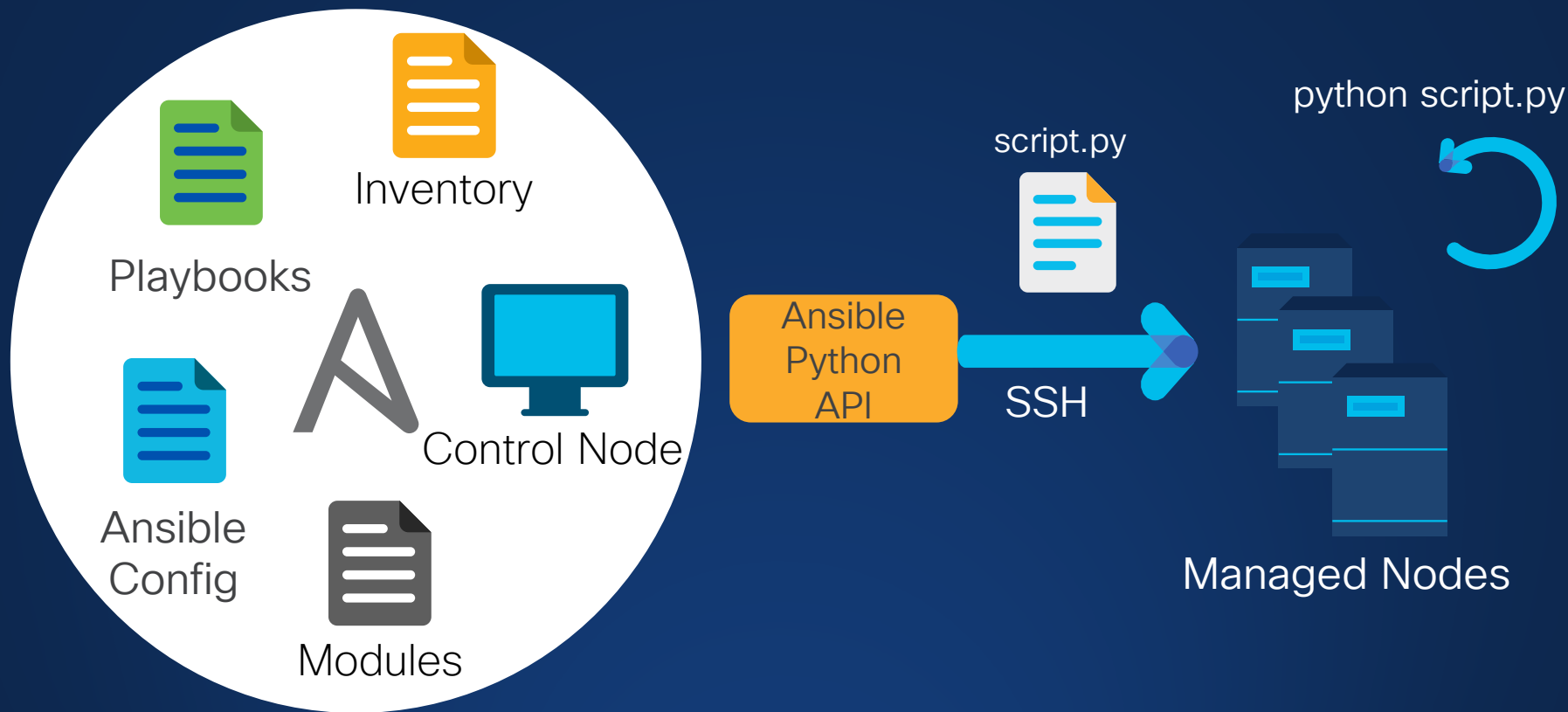


About Ansible

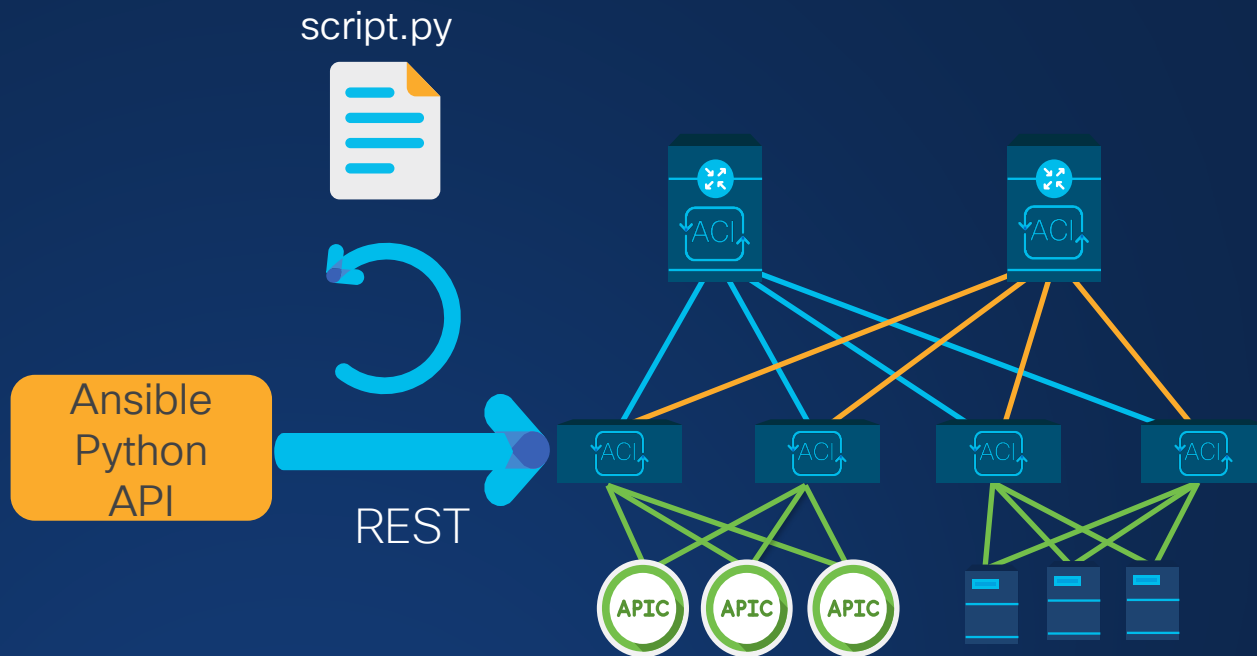


- Configuration, Orchestration tool
- 오픈 소스
- 선언적, 순차적 순서에 따름
- 다양한 시스템 관리
 - VM, 네트워크 장치, 클라우드 인스턴스

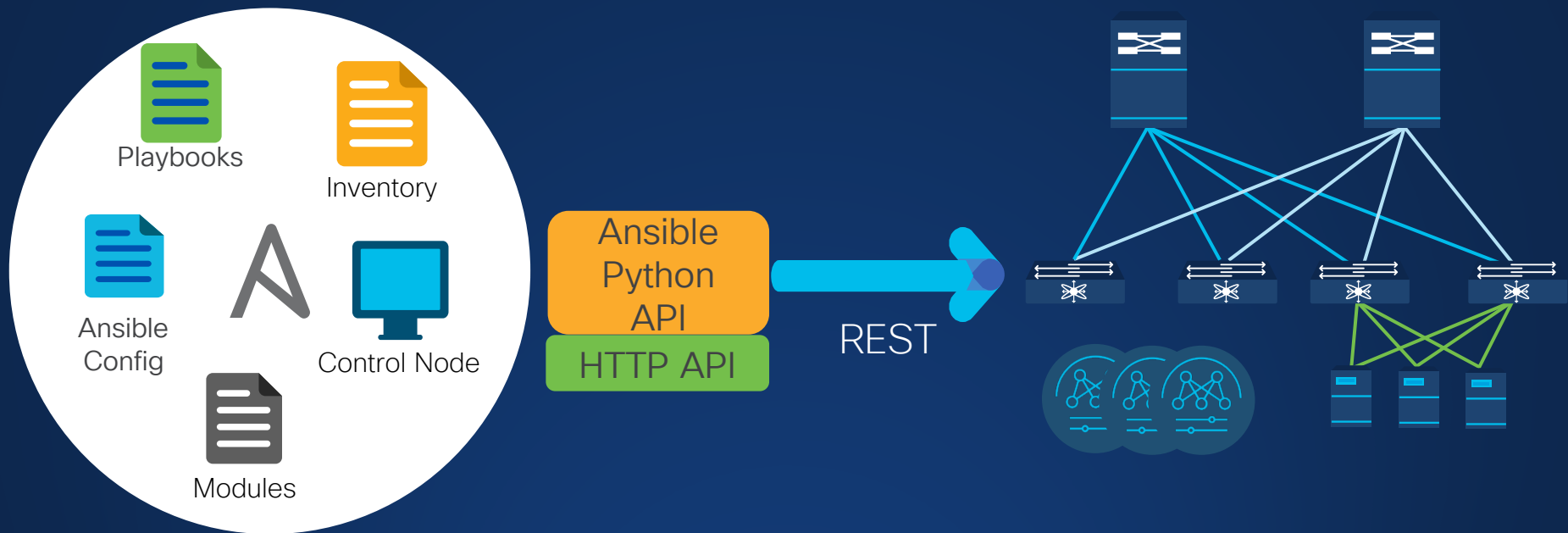
Ansible Architecture



Ansible Architecture



Ansible Architecture



Ansible 구성

- Inventory

- Module

- Playbook

Ansible 설치

➤ Version 체크

```
$ ansible --version
```

➤ Pip를 통한 설치

```
$ py -m pip install --user ansible
```

➤ Upgrade 진행

```
$ py -m pip install --upgrade --user ansible
```

Inventory

- Target Host의 목록을 포함
- INI 또는 YAML 형식
- Inventory item들을 그룹화 시킬 수 있음
- 기본 그룹 - [all] , [ungrouped]
- Inventory item은 variables를 가질 수 있음

Inventory – INI 파일 형식

inventory.ini

[apic]

apic.cisco.com *ansible_host*=10.xx.xxx.xxx *username*=cisco *certificate_name*=labadmin.crt *private_key*=pki/labadmin.key
validate_cert=no

[fabric]

leaf1.cisco.com *ansible_host*=10.xx.xxx.xxx
leaf2.cisco.com *ansible_host*=10.xx.xxx.xxx
leaf3.cisco.com *ansible_host*=10.xx.xxx.xxx
spine1.cisco.com *ansible_host*=10.xx.xxx.xxx

[fabric:vars]

ansible_ssh_user=cisco
ansible_ssh_pass=12345

[sample:group]

apic
fabric

Inventory – YAML 파일 형식

inventory.yml

```
sample: # 그룹 이름
group: # 그룹 내용 시작
  apic: # 'apic' 그룹 정의
    hosts: # 호스트 목록 시작
      apic.cisco.com: # 호스트 이름
        ansible_host: 10.xx.xxx.xxx # 호스트의 IP 주소
        username: 'cisco' # 호스트에 접속할 사용자 이름
        certificate_name: 'labadmin.crt' # 인증서 파일 이름
        private_key: pki/labadmin.key # Private Key 파일의 경로
        validate_certs: no # 인증서 유효성 검사 여부
    fabric: # 'fabric' 그룹 정의
      hosts: # 호스트 목록 시작
        leaf1.cisco.com: # 호스트 이름
        leaf2.cisco.com: # 호스트 이름
        leaf3.cisco.com: # 호스트 이름
        spine1.cisco.com: # 호스트 이름
  vars: # 그룹 변수 정의
    ansible_ssh_user: cisco # SSH 연결에 사용될 사용자 이름
    ansible_ssh_pass: 12345 # SSH 연결에 사용될 비밀번호
```

Module

- Playbook, Ad-Hoc command 내에서 사용할 수 있는 코드의 개별 단위
- Ansible에서 지정, 사용하는 “실행 명령어”
- Ansible이 제공하는 모듈이 아닌 직접 스크립트 사용 -> 먹등성 보장이 어려울 수 있음

Module

A Documentation

Namespace

Collections in the Kubernetes Namespace

Collections in the Lowlydba Namespace

Collections in the Microsoft Namespace

Collections in the Netapp Namespace

Collections in the Netapp_eseries Namespace

Collections in the Netbox Namespace

Collections in the Nginx.io Namespace

Collections in the Openstack Namespace

Collections in the Openvswitch Namespace

Collections in the Ovirt Namespace

Collections in the Purestorage Namespace

Collections in the Sensus Namespace

Collections in the Splunk Namespace

Collections in the T_systems_mms Namespace

Collections in the Telekom_mms Namespace

Collections in the Theforeman Namespace

Collections in the Vmware Namespace

Collections in the Vultr Namespace

Collections in the Vynos Namespace

Collections in the Wti Namespace

Indexes of all modules and plugins

Playbook Keywords

Return Values

Ansible Configuration Settings

Controlling how Ansible behaves:

ANSIBLEFEST

PRODUCTS

COMMUNITY

WEBINARS & TRAINING

BLOG

These are the plugins in the `cisco.aci` collection:

Modules

- [aci_aaa_custom_privilege module](#) – Manage AAA Custom Privileges ([aaa:RbacClassPriv](#))
- [aci_aaa_domain module](#) – Manage AAA domains ([aaa:Domain](#))
- [aci_aaa_role module](#) – Manage AAA roles ([aaa:Role](#))
- [aci_aaa_ssh_auth module](#) – Manage AAA SSH auth ([aaa:SshAuth](#)) objects.
- [aci_aaa_user module](#) – Manage AAA users ([aaa>User](#))
- [aci_aaa_user_certificate module](#) – Manage AAA user certificates ([aaa:UserCert](#))
- [aci_aaa_user_domain module](#) – Manage AAA user domains ([aaa:UserDomain](#))
- [aci_aaa_user_role module](#) – Manage AAA user roles ([aaa:UserRole](#))
- [aci_access_port_block_to_access_port module](#) – Manage port blocks of Fabric interface policy leaf profile interface selectors ([infra:HPortS](#), [infra:PortBlk](#))
- [aci_access_port_to_interface_policy_leaf_profile module](#) – Manage Fabric interface policy leaf profile interface selectors ([infra:HPortS](#), [infra:RsAccBaseGrp](#), [infra:PortBlk](#))
- [aci_access_span_dst_group module](#) – Manage Access SPAN destination groups ([span:DestGrp](#))
- [aci_access_span_filter_group module](#) – Manage Access SPAN filter groups ([span:FilterGrp](#))
- [aci_access_span_filter_group_entry module](#) – Manage Access SPAN filter group entries ([span:FilterEntry](#))
- [aci_access_span_src_group module](#) – Manage Access SPAN source groups ([span:SrcGrp](#))
- [aci_access_span_src_group_src module](#) – Manage Access SPAN sources ([span:Src](#))
- [aci_access_span_src_group_src_path module](#) – Manage Access SPAN source paths ([span:RsSrcToPathEp](#))
- [aci_access_sub_port_block_to_access_port module](#) – Manage sub port blocks of Fabric interface policy leaf profile interface selectors ([infra:HPortS](#), [infra:SubPortBlk](#))
- [aci_aep module](#) – Manage attachable Access Entity Profile (AEP) objects ([infra:AttEntityP](#), [infra:ProvAcc](#))
- [aci_aep_to_domain module](#) – Bind AEPs to Physical or Virtual Domains ([infra:RsDomP](#))
- [aci_aep_to_epg module](#) – Bind EPG to AEP ([infra:RsFuncToEpg](#)).
- [aci_ap module](#) – Manage top level Application Profile (AP) objects ([fv:Ap](#))
- [aci_bd module](#) – Manage Bridge Domains (BD) objects ([fv:BD](#))
- [aci_bd_dhcp_label module](#) – Manage DHCP Labels ([dhcp:Lbl](#))
- [aci_bd_subnet module](#) – Manage Subnets ([fv:Subnet](#))
- [aci_bd_to_l3out module](#) – Bind Bridge Domain to L3 Out ([fv:RsBDToOut](#))
- [aci_bgp_best_path_policy module](#) – Manage BGP Best Path policy ([bgp:BestPathCtrlPol](#))
- [aci_bgp_rr_asn module](#) – Manage BGP Route Reflector ASN.
- [aci_bgp_rr_node module](#) – Manage BGP Route Reflector objects.
- [aci_bgp_timers_policy module](#) – Manage BGP timers policy ([bgp:CtxPol](#))
- [aci_bulk_static_binding_to_epg module](#) – Bind static paths to EPGs ([fv:RsPathAtt](#))
- [aci_cloud_ap module](#) – Manage Cloud Application Profile (AP) ([cloud:App](#))
- [aci_cloud_aws_provider module](#) – Manage Cloud AWS Provider ([cloud:AwsProvider](#))
- [aci_cloud_ibm_module](#) – Manage Cloud IBM CPE Autonomic Edge Profile ([cloud:IbmCpeAeP](#))

 Search this site

Playbook

- 대상 시스템에 대한 일련의 Play, task들을 포함
- 사람이 읽을 수 있는 (Human-Readable) YAML 형식으로 작성됨
- Playbook들은 재사용 및 공유 가능

Playbook 기본 Config의 순서

Playbook.yml

Playbook

```
---  
- name: Playbook sample  
  hosts: apic  
  gather_facts: no  
  
tasks:  
  - name: "Create Tenant"  
    cisco.tenant:  
      host: cisco.com  
      username: admin  
      password: 12345  
      state: present  
      tenant: "tenanty"  
      description: "Tenant created with Ansible"
```

Playbook 시작
Playbook 이름
Host group
Variable 수집
Task list 시작
Task 이름
모듈 선언
로그인
Tenant 생성
Tenant 이름
Description

Playbook command 실행

- Ansible의 Playbook command로 실행

```
$ ansible-playbook -i Inventory.yml playbook.yml
```

The diagram illustrates the command execution for running an Ansible playbook. The command is shown as `$ ansible-playbook -i Inventory.yml playbook.yml`. The file `Inventory.yml` is highlighted with a red box and labeled "Inventory file" with a red callout. The file `playbook.yml` is highlighted with a green box and labeled "Playbook file" with a green callout.

- Inventory 파일이 없다면, default 값인 `/etc/ansible/hosts`로 설정됨

Playbook output command

```
(base) shaphill@SHAPHILL-M-MOXY demo1 % ansible-playbook -i inventory demo.yml
```

```
PLAY [Demo Playbook]
```

```
TASK [Create tenant]
```

```
ok: [https:// network:8041]
```

구성과 일치

```
TASK [Create VRF]
```

```
changed: [https:// network:8041]
```

Configuration 이
원하는 상태에 맞게
구성됨

```
TASK [Create bridge domains]
```

```
changed: [https:// network:8041]
```

```
[...]
```

결과에 대한
요약

```
PLAY RECAP
```

```
https://esc-aci-network:8041 : ok=5  changed=5  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

Key Points

- Ansible Architecture
- Ansible 구성
 - Inventory
 - Module
 - Playbook



Materials for today's workflow —



Material 순서

- Jinja 2
- Ansible Role
- Ansible configuration tip

Jinja 2

- Ansible이 사용하는 템플릿 언어
- Ansible에만 국한되지 않는 Python 라이브러리
- 변수, 루프, 조건 등을 참조하는 데 사용

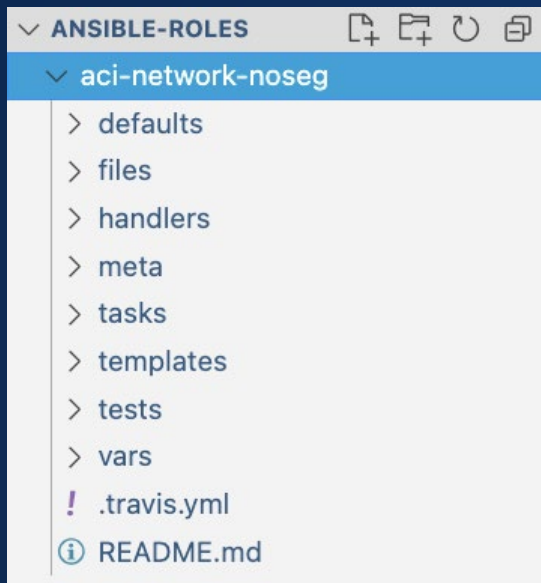


Ansible Role

- Role을 통해 쉽게 재사용 및 공유 가능
- Role은 기본값, 변수, task 등을 조직화된 파일 구조에 저장 가능
- 하나의 컨테이너에 task들을 그룹화 가능
- 호스트 정보가 없다면? -> Playbook 용으로 reserved 됨

Ansible Role

➤ Role 파일의 구조



➤ defaults – 기본 변수

➤ vars – 현재 Role과 관련된 변수

➤ tasks – Role이 실행하는 Ansible의 task

➤ meta – 종속성을 포함한 역할의 metadata

Ansible Role

vars/main.yml

```
---  
  
network_type: l2  
public: no
```

위 변수들은 Role을
사용 시, 정의된 변수에
덮어씀

tasks/main.yml

```
---  
  
- name: Create network  
  
vars:  
  aci_login: &aci_login  
  host: "{{ ansible_host }}"  
  username: "{{ aci_username }}"  
  password: "{{ aci_password | default(omit) }}"  
  [...]  
  
block:  
  - name: Add Bridge Domain  
    cisco.aci.aci_bd:  
      <<: *aci_login  
      tenant: "{{ tenant_name }}"  
      bd: "{{ network_name }}"  
      vrf: "{{ vrf_name }}"  
      l2_unknown_unicast: "{{ 'proxy' if (network_type == 'L3') else 'flood' }}"  
      arp_flooding: yes  
      enable_routing: "{{ 'yes' if (network_type == 'L3') else 'no' }}"  
      state: present  
    [...]
```

해당 변수들은 Inventory 또는
Play에서 가져옴

Ansible Role

Role을 Play level에서 사용하는 시나리오 - 변수 범위에 대한 고려

```
# Playbook 1
```

```
---
```

```
- hosts: apic
  connection: local
  gather_facts: no
  roles:
    - role: aci-network-noseg
      network_name: vlan001
      network_type: L3
      subnet: 192.168.1.1/24
      public: yes
```

```
tasks:
  [...]
```

```
# Playbook 2
```

```
---
```

```
- hosts: apic
  connection: local
  gather_facts: no
  roles:
    - role: aci-network-noseg
      vars:
        network_name: vlan001
        network_type: L3
        subnet: 192.168.1.1/24
        public: yes
```

```
tasks:
  [...]
```

두 시나리오 모두 결과는 같음

Ansible Role

Playbook에 Role을 포함시키는 시나리오 (include_role)

```
# Example playbook
---
- hosts: apic
  connection: local
  gather_facts: no

[...]

tasks:
[...]
- name: Add network
  include_role:
    name: aci-network-noseg
vars:
  network_name: vlan001
  network_type: L3
  subnet: 192.168.1.1/24
  public: yes
```

- Role이 정의된 순서대로 실행
- 변수의 범위는 Role로 지정
- Role 블록 외부의 변수를 참조 불가능

Ansible Role

Playbook에서 Role을 가져오는 시나리오 (import_role)

```
# Example playbook
---
- hosts: apic
  connection: local
  gather_facts: no

[...]

tasks:
[...]
- name: Add network
  import_role:
    name: aci-network-noseg
  vars:
    network_name: vlan001
    network_type: L3
    subnet: 192.168.1.1/24
    public: yes
```

➤ Role이 정의된 순서대로 실행

➤ 변수의 범위는 Role로 지정

➤ Role 블록 외부의 변수를 참조 가능

Ansible configuration tip

기존 Ansible의 task 실행 시 -> SSH 명령을 여러 번 실행 (아래 과정들 수행을 위해)

- 임시 디렉토리 생성
- 모듈 스크립트 전송
- 모듈에 실행 권한 부여 및 모듈 실행 / 삭제

Ansible configuration tip

- 파이프라이닝 설정 활성화
- Module Script의 전송을 처리하지 않아도 됨
- SSH 처리 수가 감소 -> 성능 개선

```
[ssh_connection]  
pipelining = True
```

Ansible configuration tip

Playbook을 이용해 파이프라이닝 대응 처리 과정

```
- name : # Plabook에서 파이프라이닝 대응
  hosts : all
  become : yes
  vars :
    ansible_ssh_pipelining : no
  tasks:
    - name : # ansible_user에 대한 requiretty 설정을 비활성화
      lineinfile : # lineinfile 모듈로 /etc/sudoers 파일 수정
        dest : /etc/sudoers
        regexp : '^Defaults : {{ ansible_user }}\s requiretty' # 특정 패턴 찾기
        line : 'Defaults : {{ ansible_user }} requiretty'
        validate : 'visudo -cf %s'
        backup : yes
```

Ansible configuration tip

Forks로 동시 병렬 배포 수 제어

- Forks : 최대 동시 접속 수를 표시해줌
- 기본 설정 : 5개
- 장비 여러 대에 동시 배포 전, 먼저 조정해주기

요약

- Ansible은 인프라를 자동화하는 Tool이다.
- 또한, 반복 작업을 규모에 맞게 작업하여 시간을 절약한다.
- 다양한 벤더 제품들에 대한 광범위한 Ansible Module 목록을 보유하고 있다.



Thank you

