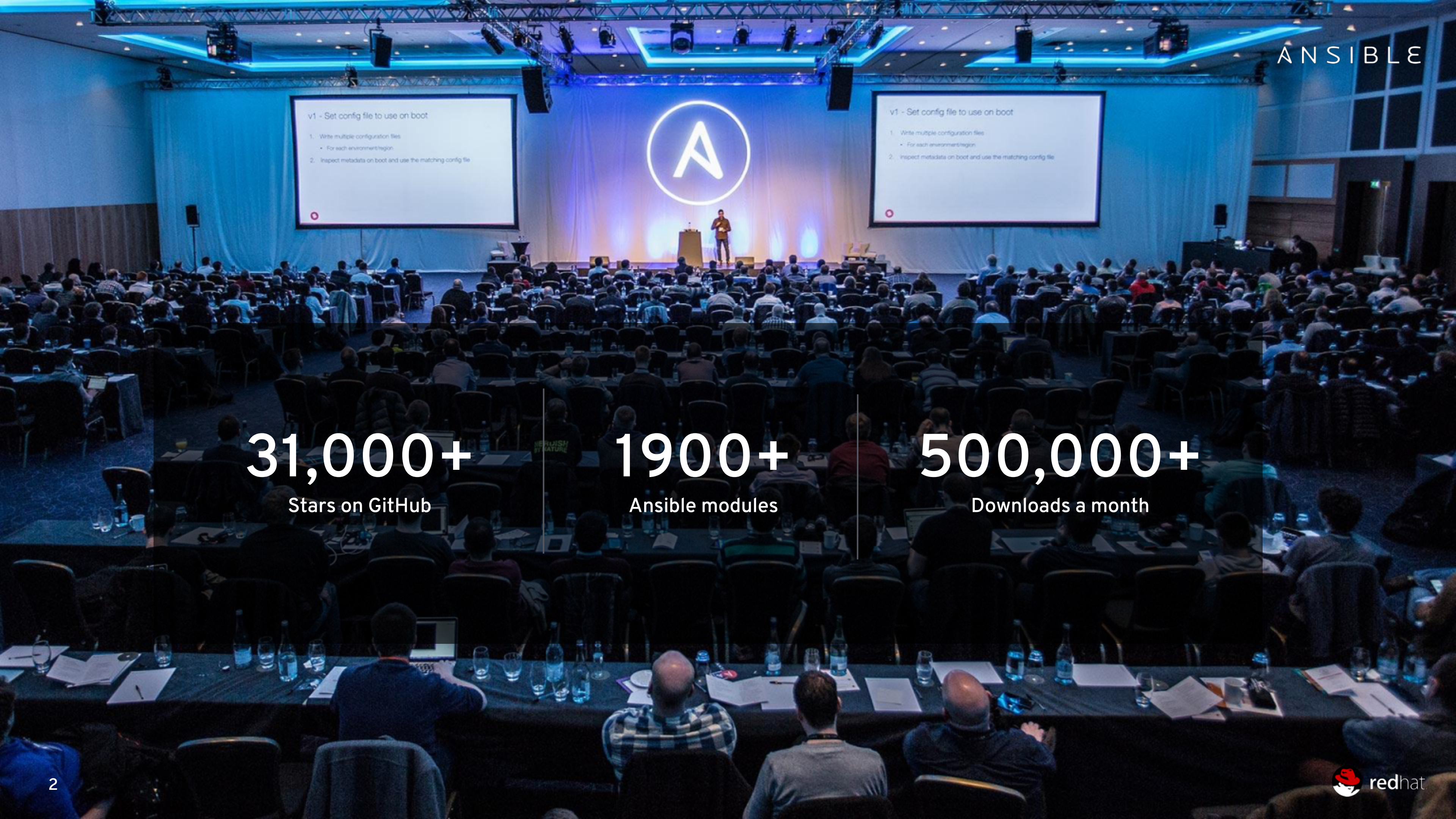
ANSIBLE

# IAC on OpenStack (feat. ansible)

김용기 부장
Sr. Solution Architect
Red Hat

redhat


v1 - Set config file to use on boot
1. Write multiple configuration files
• For each environment/region
2. Inspect metadata on boot and use the matching config file

**31,000+**
Stars on GitHub

**1900+**
Ansible modules

**500,000+**
Downloads a month

# WHY ANSIBLE?

## SIMPLE

읽기 쉽고

코딩을 아주 잘 할 필요없이

순서대로 실행

모든 팀에 유용

Get productive quickly

## POWERFUL

애플리케이션 배포

설정 관리

워크플로우 오케스트레이션

네트워크 자동화

Orchestrate the app lifecycle

## AGENTLESS

에이전트 없이

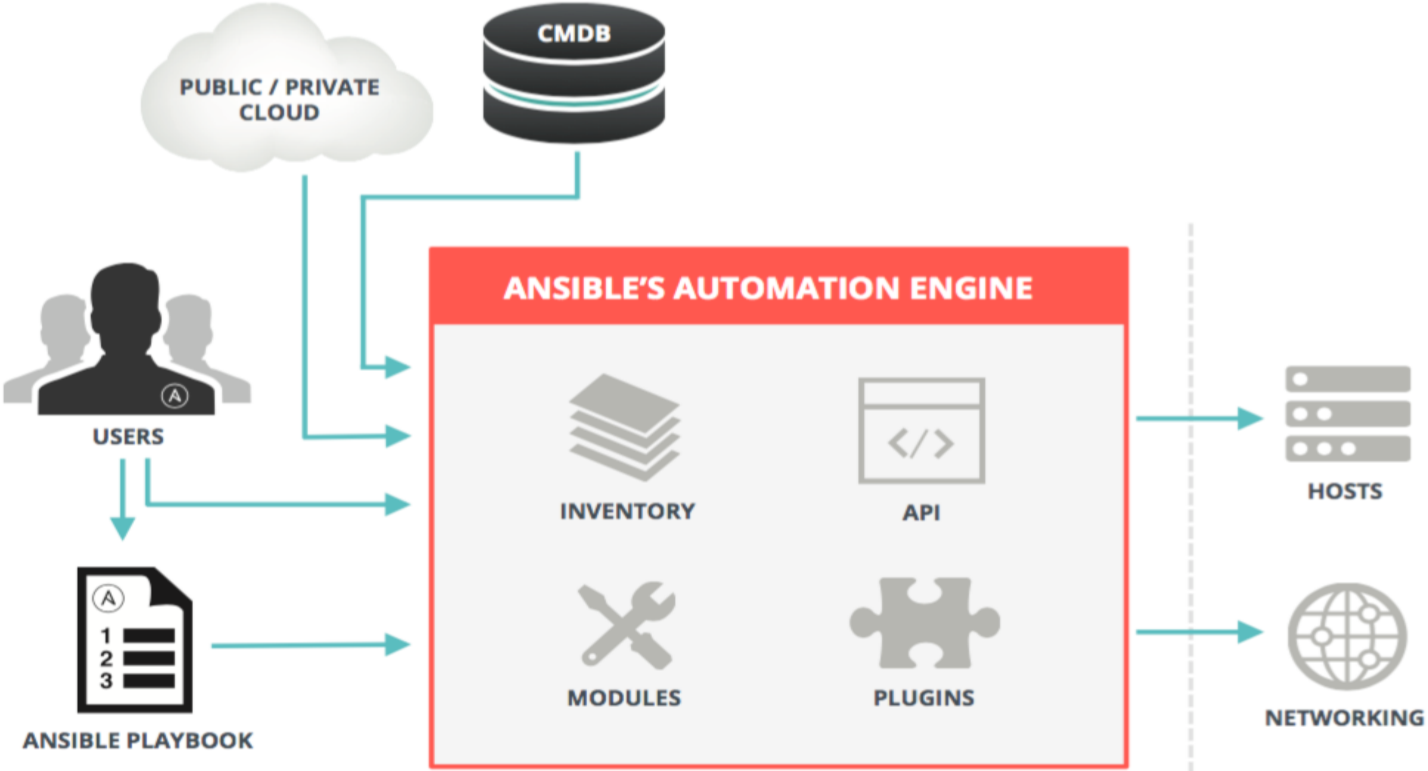
OpenSSH & WinRM 사용

보안 강화

즉시 사용 가능

More efficient & more secure

# Ansible 아키텍처



# PLAYBOOK EXAMPLE

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

  tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest
```

```
      state: started
```

declarative, 선언형 방식

# ANSIBLE SHIPS WITH OVER 1250 MODULES

## CLOUD

- AWS
- Azure
- CenturyLink
- CloudScale
- Digital Ocean
- Docker
- Google
- Linode
- OpenStack
- Rackspace
- And more...

## VIRT AND CONTAINER

- Docker
- VMware
- RHEV
- OpenStack
- OpenShift
- Atomic
- CloudStack
- And more...

## WINDOWS

- ACLs
- Files
- Commands
- Packages
- IIS
- Regedits
- Shell
- Shares
- Services
- DSC
- Users
- Domains
- And more...

## NETWORK

- Arista
- A10
- Cumulus
- Big Switch
- Cisco
- Cumulus
- Dell
- F5
- Juniper
- Palo Alto
- OpenSwitch
- And more...

## NOTIFY

- HipChat
- IRC
- Jabber
- Email
- RocketChat
- Sendgrid
- Slack
- Twilio
- And more...

# WHAT CAN I DO WITH ANSIBLE?

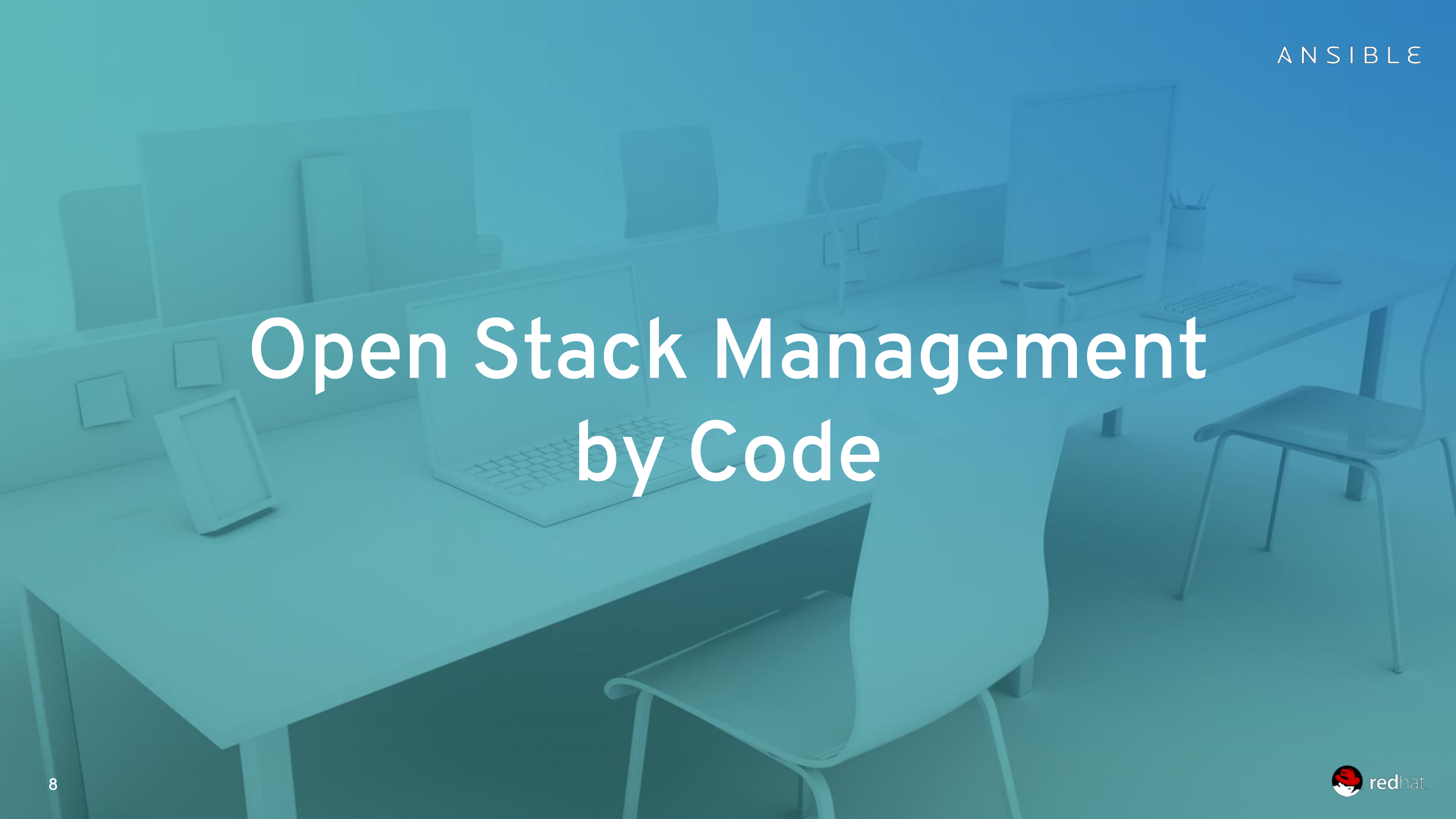
Automate the deployment and management of your entire IT footprint.

Do this...

- Orchestration
- Configuration Management
- Application Deployment
- Provisioning
- Continuous Delivery
- Security and Compliance

On these...

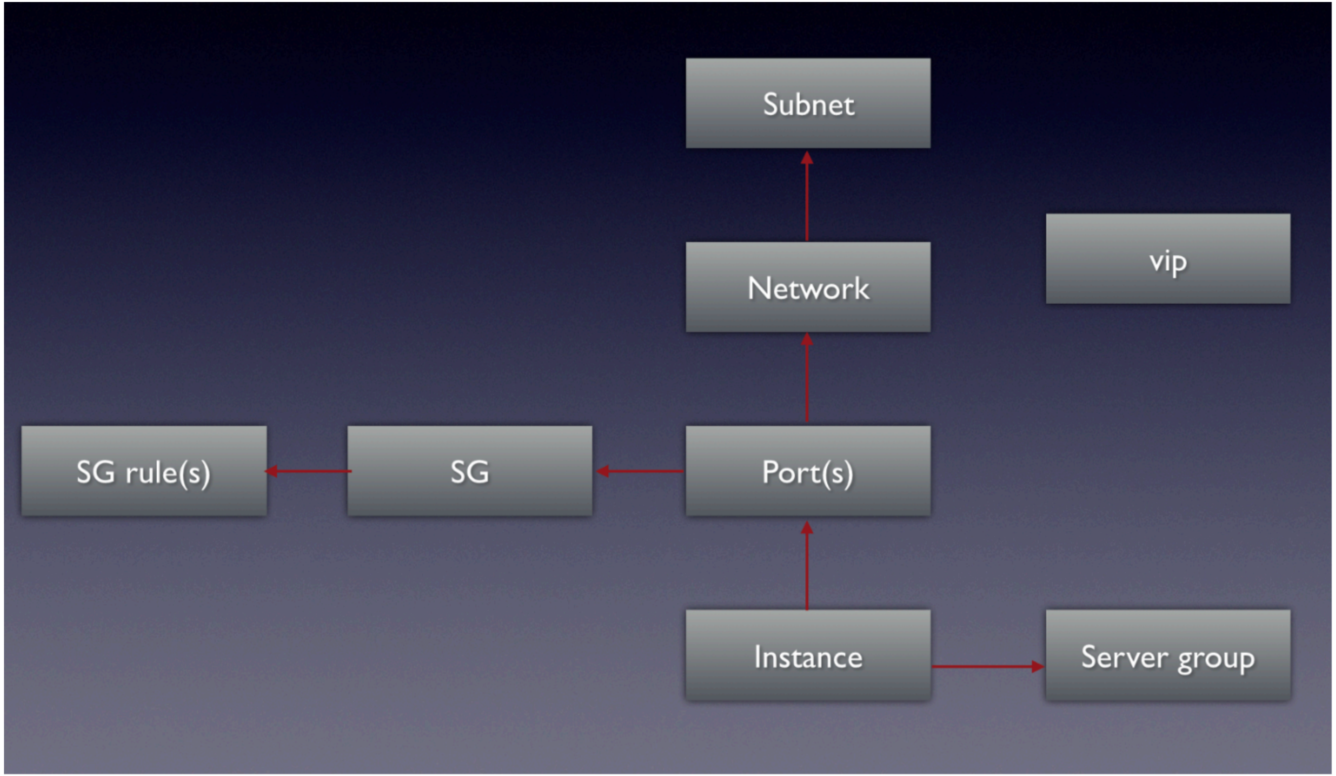
- Firewalls
- Load Balancers
- Applications
- Containers
- Clouds
- Servers
- Infrastructure
- Storage
- Network Devices
- And more...



# Open Stack Management by Code

# 기본 인프라 생성

## 인프라 생성도



https://medium.com/@michalmedvecky/managing-your-openstack-infrastructure-with-hashicorp-terraform-8c93ade214b4

# 기본 인프라 설정

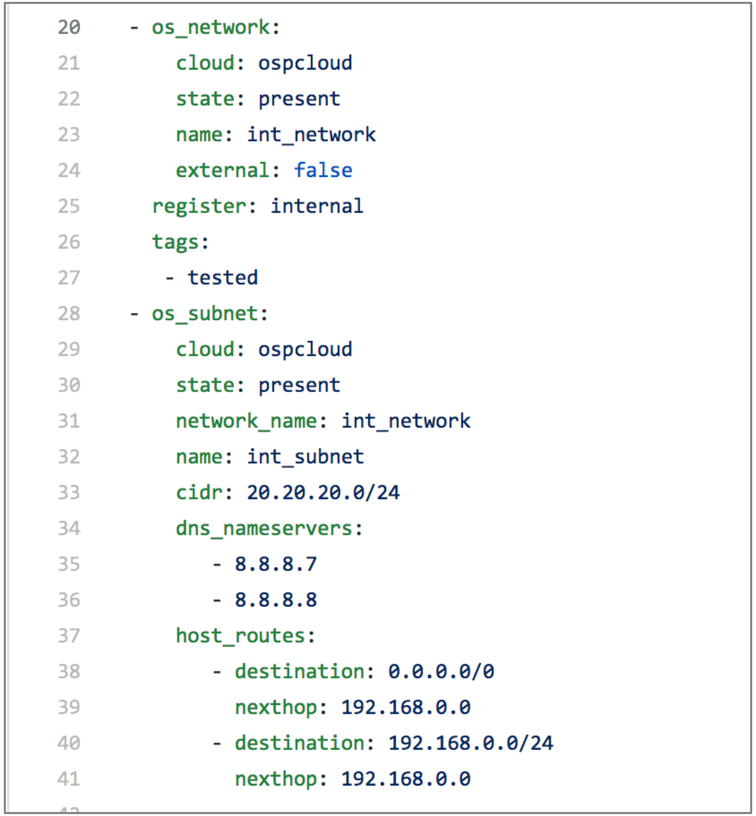
## Network/Subnet/Router

### 매뉴얼 작업 경우

- 테넌트별 별도 생성
- 입력값이 부정확할 때 통신 에러 발생

### 작업 자동화 경우

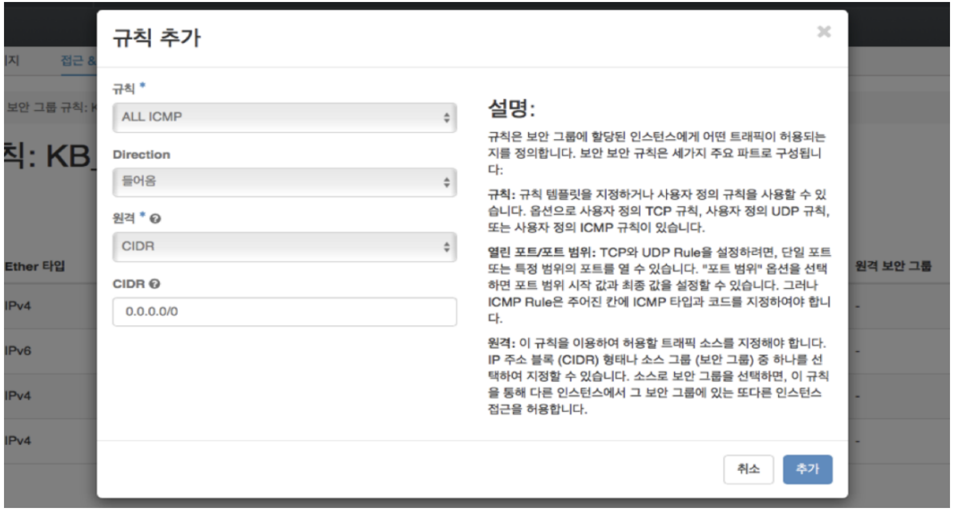
- 코드를 통해 기존 설정 확인
- 신속한 신규 네트워크 생성



```
- os_network:
    cloud: ospcloud
    state: present
    name: int_network
    external: false
  register: internal
  tags:
   - tested
- os_subnet:
    cloud: ospcloud
    state: present
    network_name: int_network
    name: int_subnet
    cidr: 20.20.20.0/24
    dns_nameservers:
       - 8.8.8.7
       - 8.8.8.8
    host_routes:
       - destination: 0.0.0.0/0
         nexthop: 192.168.0.0
       - destination: 192.168.0.0/24
         nexthop: 192.168.0.0
```

# Security Group

| | 매뉴얼 작업 경우 | 작업 자동화 경우 |
|---|---|---|
| Security Group | • 기존 보안 그룹을 복사하여 생성 불가<br>• 새로운 SG마다 신규로 규칙 입력 필요 | 코드를 복사/편집하여 SG 생성 |



## 규칙 추가

규칙 *: ALL ICMP

Direction: 들어옴

원격 *: CIDR

CIDR: 0.0.0.0/0

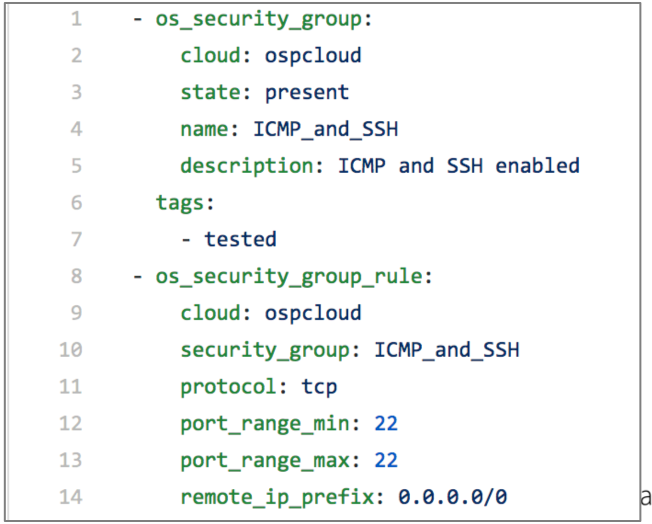
**설명:**

규칙은 보안 그룹에 할당된 인스턴스에게 어떤 트래픽이 허용되는지를 정의합니다. 보안 보안 규칙은 세가지 주요 파트로 구성됩니다:

**규칙:** 규칙 템플릿을 지정하거나 사용자 정의 규칙을 사용할 수 있습니다. 옵션으로 사용자 정의 TCP 규칙, 사용자 정의 UDP 규칙, 또는 사용자 정의 ICMP 규칙이 있습니다.

**열린 포트/포트 범위:** TCP와 UDP Rule을 설정하려면, 단일 포트 또는 특정 범위의 포트를 열 수 있습니다. "포트 범위" 옵션을 선택하면 포트 범위 시작 값과 최종 값을 설정할 수 있습니다. 그러나 ICMP Rule은 주어진 칸에 ICMP 타입과 코드를 지정하여야 합니다.

**원격:** 이 규칙을 이용하여 허용할 트래픽 소스를 지정해야 합니다. IP 주소 블록 (CIDR) 형태나 소스 그룹 (보안 그룹) 중 하나를 선택하여 지정할 수 있습니다. 소스로 보안 그룹을 선택하면, 이 규칙을 통해 다른 인스턴스에서 그 보안 그룹에 있는 또다른 인스턴스 접근을 허용합니다.

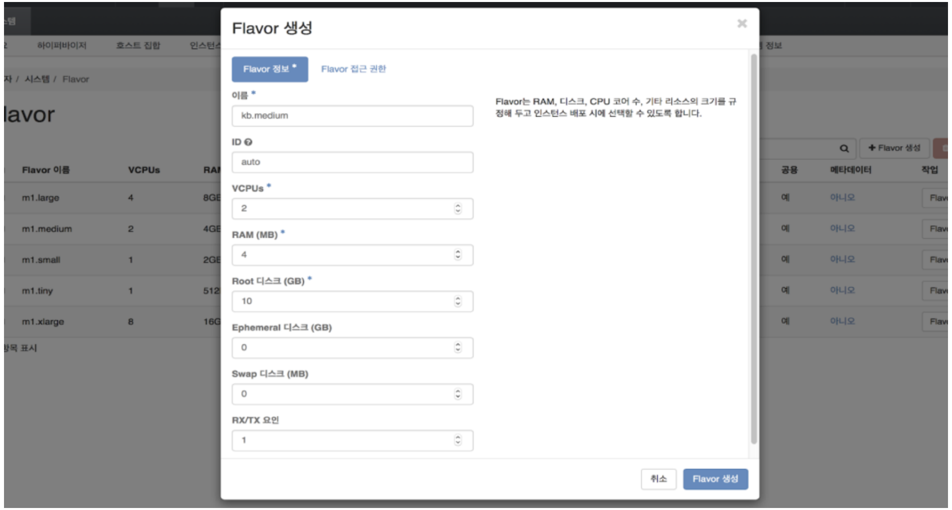


```
- os_security_group:
    cloud: ospcloud
    state: present
    name: ICMP_and_SSH
    description: ICMP and SSH enabled
  tags:
    - tested
- os_security_group_rule:
    cloud: ospcloud
    security_group: ICMP_and_SSH
    protocol: tcp
    port_range_min: 22
    port_range_max: 22
    remote_ip_prefix: 0.0.0.0/0
```

# 기본 인프라 설정

## Flavor

| | 매뉴얼 작업 경우 | 작업 자동화 경우 |
|---|---|---|
| Flavor | • 기존 flavor 편집 불가<br>• 편집 필요시, 기존 스펙을 확인하고 재생성 필요 | 코드를 복사/편집하여 생성 |



하이퍼바이저 호스트 집합 인스턴스 정보

자 / 시스템 / Flavor

lavor

| Flavor 이름 | VCPUs | RA | 공용 | 메타데이터 | 작업 |
|---|---|---|---|---|---|
| m1.large | 4 | 8GE | 예 | 아니오 | Flav |
| m1.medium | 2 | 4GE | 예 | 아니오 | Flav |
| m1.small | 1 | 2GE | 예 | 아니오 | Flav |
| m1.tiny | 1 | 512 | 예 | 아니오 | Flav |
| m1.xlarge | 8 | 16G | 예 | 아니오 | Flav |

항목 표시

Flavor 생성

Flavor 정보 * | Flavor 접근 권한

이름 *: kb.medium

ID: auto

VCPUs *: 2

RAM (MB) *: 4

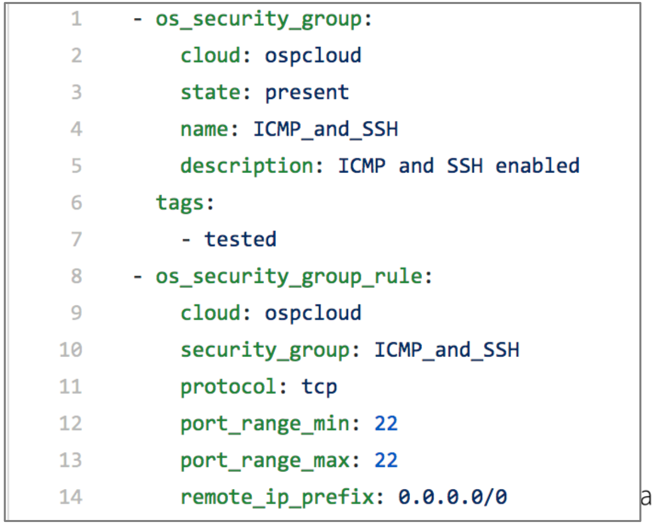
Root 디스크 (GB) *: 10

Ephemeral 디스크 (GB): 0

Swap 디스크 (MB): 0

RX/TX 요인: 1

Flavor는 RAM, 디스크, CPU 코어 수, 기타 리소스의 크기를 규정해 두고 인스턴스 배포 시에 선택할 수 있도록 합니다.



```
- os_security_group:
    cloud: ospcloud
    state: present
    name: ICMP_and_SSH
    description: ICMP and SSH enabled
  tags:
    - tested
- os_security_group_rule:
    cloud: ospcloud
    security_group: ICMP_and_SSH
    protocol: tcp
    port_range_min: 22
    port_range_max: 22
    remote_ip_prefix: 0.0.0.0/0
```

## 인스턴스 배포

인스턴스 생성 코드 예제

- Heat 사용시
- Terraform 사용시
- Ansible 사용시

# 인스턴스 배포

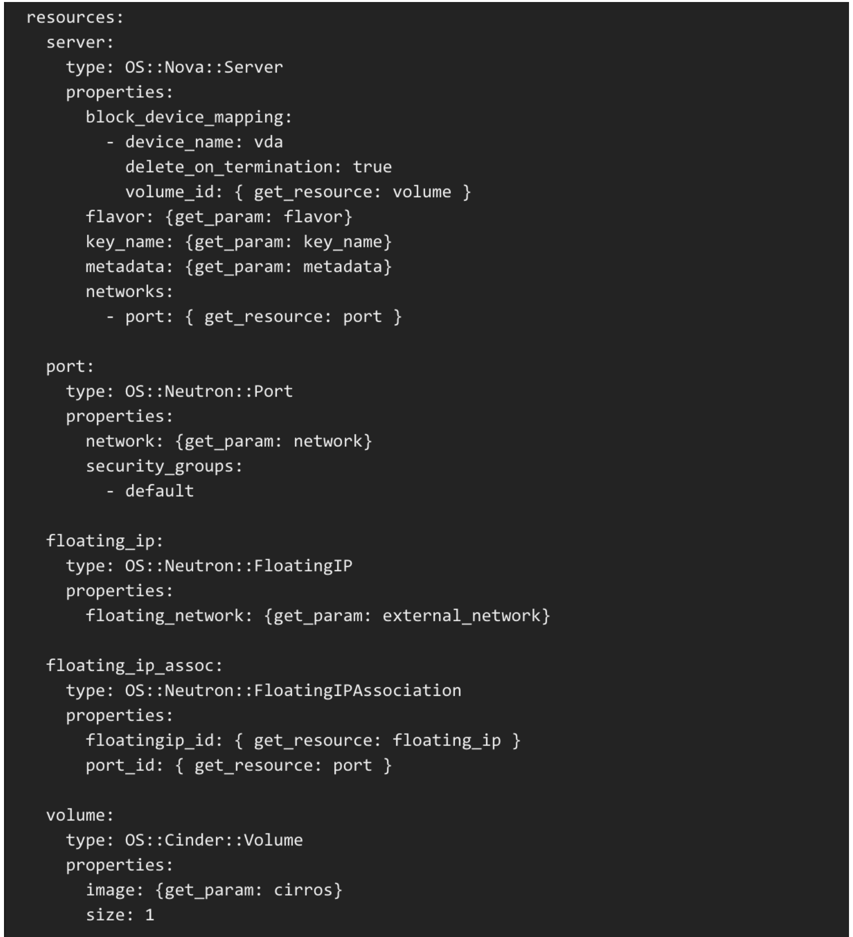
ANSIBLE

## Heat 코드

- YAML 형식
- stack 을 통한 서비스 구동
- ceilometer와 연동하여 auto scale 가능
- 필요파일:

  - template.yaml

  - environment.yaml



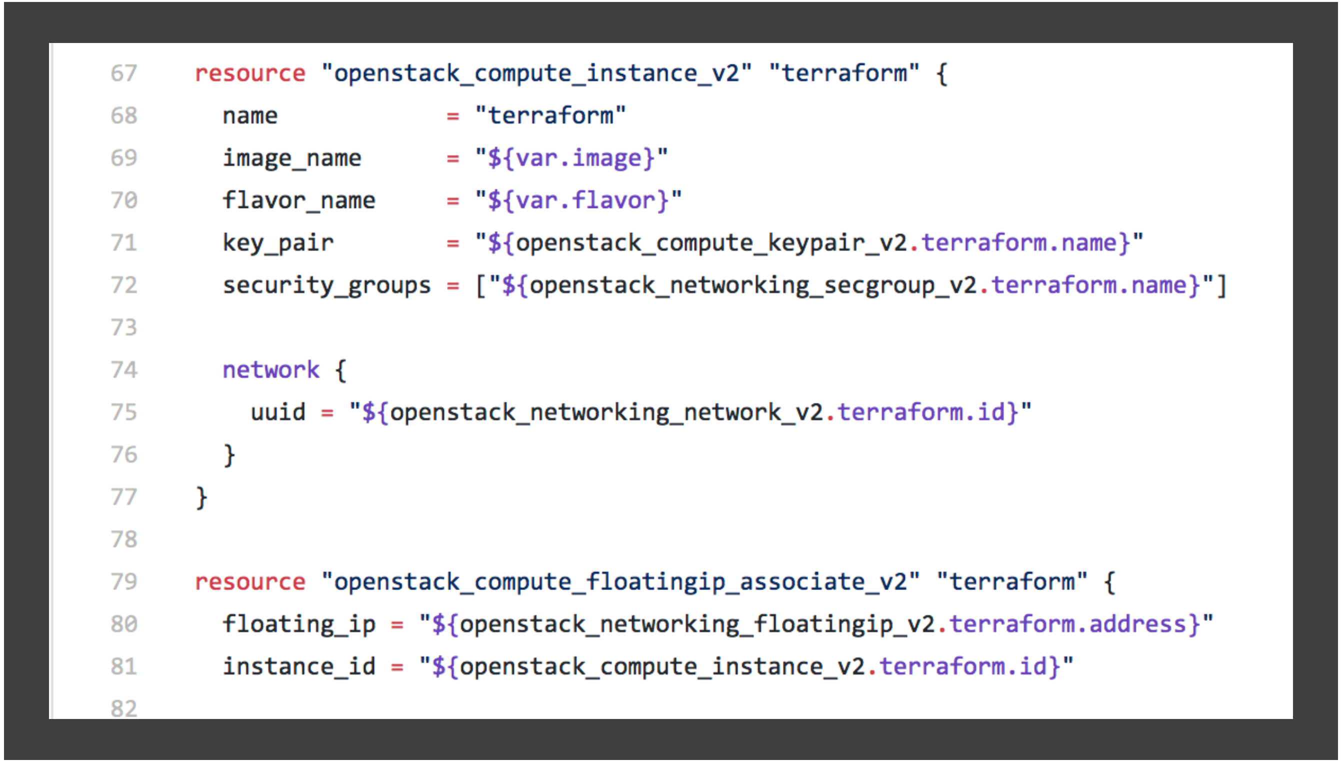
```
resources:
  server:
    type: OS::Nova::Server
    properties:
      block_device_mapping:
        - device_name: vda
          delete_on_termination: true
          volume_id: { get_resource: volume }
      flavor: {get_param: flavor}
      key_name: {get_param: key_name}
      metadata: {get_param: metadata}
      networks:
        - port: { get_resource: port }

  port:
    type: OS::Neutron::Port
    properties:
      network: {get_param: network}
      security_groups:
        - default

  floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: {get_param: external_network}

  floating_ip_assoc:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      floatingip_id: { get_resource: floating_ip }
      port_id: { get_resource: port }

  volume:
    type: OS::Cinder::Volume
    properties:
      image: {get_param: cirros}
      size: 1
```

redhat.

# 인스턴스 배포

## Terraform 코드

- 테라폼 전용 언어인 tf 형식
- 선언형 언어
- 쉬운 코드 및 적용



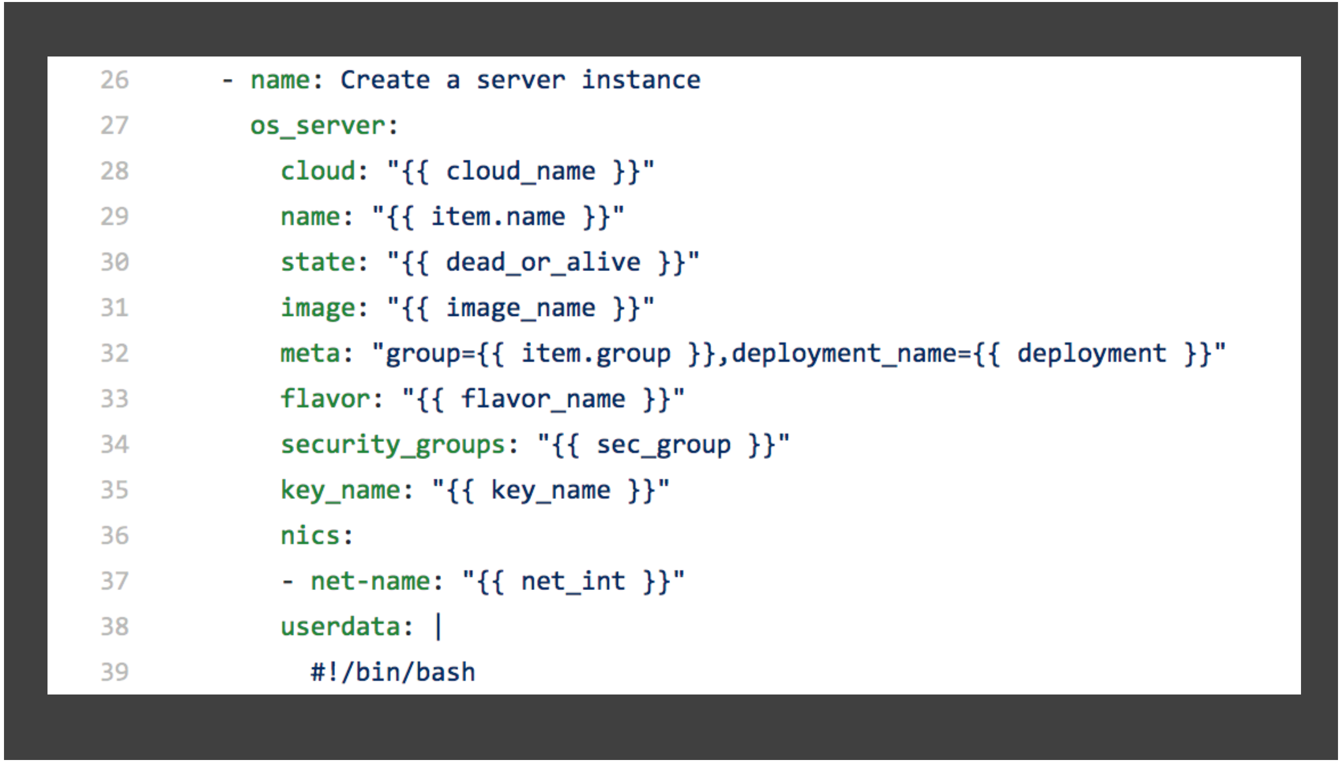
```
resource "openstack_compute_instance_v2" "terraform" {
  name            = "terraform"
  image_name      = "${var.image}"
  flavor_name     = "${var.flavor}"
  key_pair        = "${openstack_compute_keypair_v2.terraform.name}"
  security_groups = ["${openstack_networking_secgroup_v2.terraform.name}"]

  network {
    uuid = "${openstack_networking_network_v2.terraform.id}"
  }
}

resource "openstack_compute_floatingip_associate_v2" "terraform" {
  floating_ip = "${openstack_networking_floatingip_v2.terraform.address}"
  instance_id = "${openstack_compute_instance_v2.terraform.id}"

```

참고: https://github.com/terraform-providers/terraform-provider-openstack/blob/master/examples/app-with-networking/main.tf

# 인스턴스 배포

## Ansible 코드

- YAML 형식
- 선언형 언어
- 상대적으로 간단한 코드
- 인스턴스 배포 이후, OS 및 APP 관련 설정까지 일원화

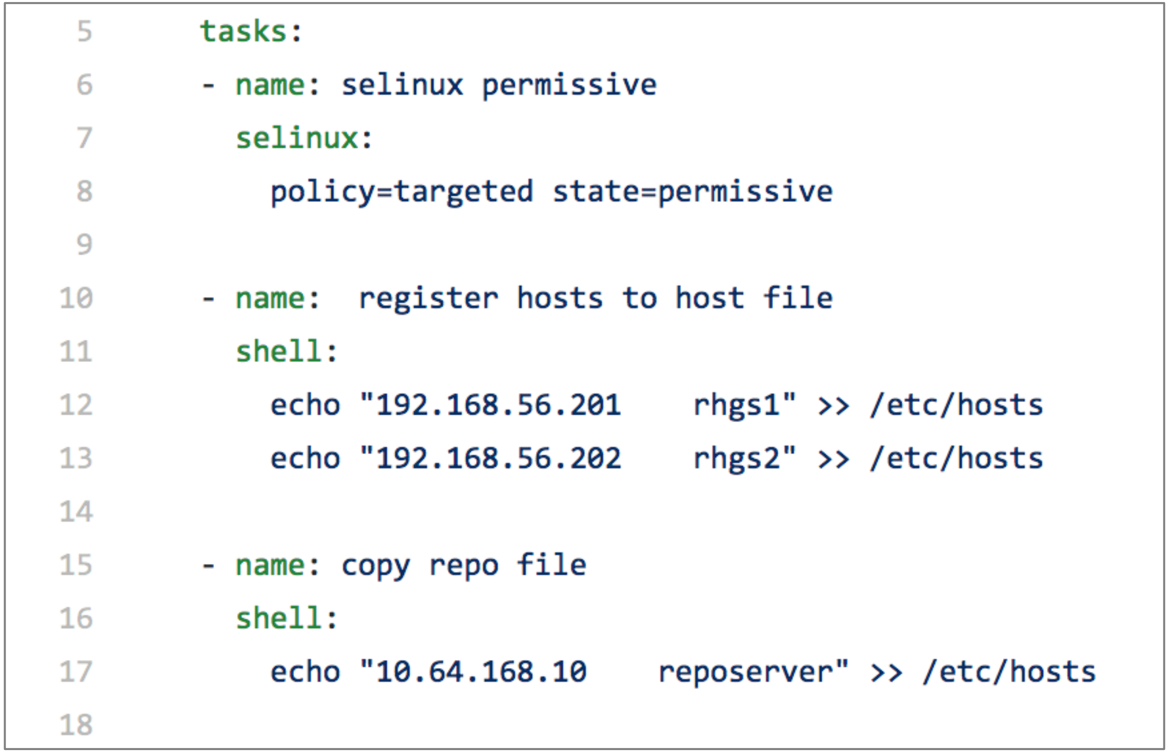


```
    - name: Create a server instance
      os_server:
        cloud: "{{ cloud_name }}"
        name: "{{ item.name }}"
        state: "{{ dead_or_alive }}"
        image: "{{ image_name }}"
        meta: "group={{ item.group }},deployment_name={{ deployment }}"
        flavor: "{{ flavor_name }}"
        security_groups: "{{ sec_group }}"
        key_name: "{{ key_name }}"
        nics:
        - net-name: "{{ net_int }}"
        userdata: |
          #!/bin/bash
```

# 인스턴스 설정

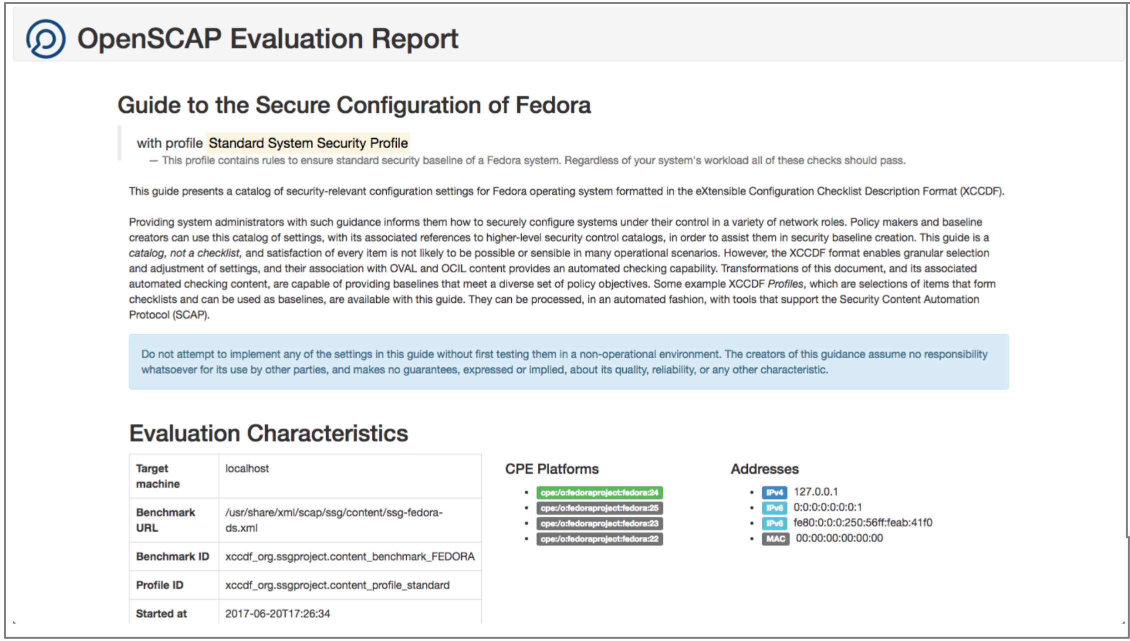
## 표준 OS 환경 설정

- cron 등록
- systemctl 설정
- ulimit 설정
- ntp 설정
- repo 등록
- 추가 패키지 설치
- 등등



```
tasks:
- name: selinux permissive
  selinux:
    policy=targeted state=permissive

- name:  register hosts to host file
  shell:
    echo "192.168.56.201    rhgs1" >> /etc/hosts
    echo "192.168.56.202    rhgs2" >> /etc/hosts

- name: copy repo file
  shell:
    echo "10.64.168.10    reposerver" >> /etc/hosts

```

# OpenScap +Ansible 을 통한 보안 점검



## OpenSCAP Evaluation Report

### Guide to the Secure Configuration of Fedora

> with profile Standard System Security Profile
> — This profile contains rules to ensure standard security baseline of a Fedora system. Regardless of your system's workload all of these checks should pass.

This guide presents a catalog of security-relevant configuration settings for Fedora operating system formatted in the eXtensible Configuration Checklist Description Format (XCCDF).

Providing system administrators with such guidance informs them how to securely configure systems under their control in a variety of network roles. Policy makers and baseline creators can use this catalog of settings, with its associated references to higher-level security control catalogs, in order to assist them in security baseline creation. This guide is a *catalog, not a checklist,* and satisfaction of every item is not likely to be possible or sensible in many operational scenarios. However, the XCCDF format enables granular selection and adjustment of settings, and their association with OVAL and OCIL content provides an automated checking capability. Transformations of this document, and its associated automated checking content, are capable of providing baselines that meet a diverse set of policy objectives. Some example XCCDF *Profiles*, which are selections of items that form checklists and can be used as baselines, are available with this guide. They can be processed, in an automated fashion, with tools that support the Security Content Automation Protocol (SCAP).

Do not attempt to implement any of the settings in this guide without first testing them in a non-operational environment. The creators of this guidance assume no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic.

### Evaluation Characteristics

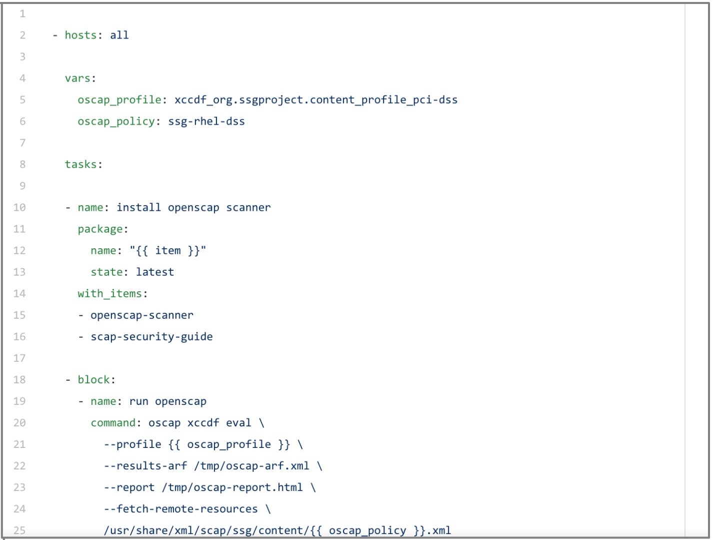
| | |
|---|---|
| **Target machine** | localhost |
| **Benchmark URL** | /usr/share/xml/scap/ssg/content/ssg-fedora-ds.xml |
| **Benchmark ID** | xccdf_org.ssgproject.content_benchmark_FEDORA |
| **Profile ID** | xccdf_org.ssgproject.content_profile_standard |
| **Started at** | 2017-06-20T17:26:34 |

#### CPE Platforms

- cpe:/o:fedoraproject:fedora:24
- cpe:/o:fedoraproject:fedora:25
- cpe:/o:fedoraproject:fedora:23
- cpe:/o:fedoraproject:fedora:22

#### Addresses

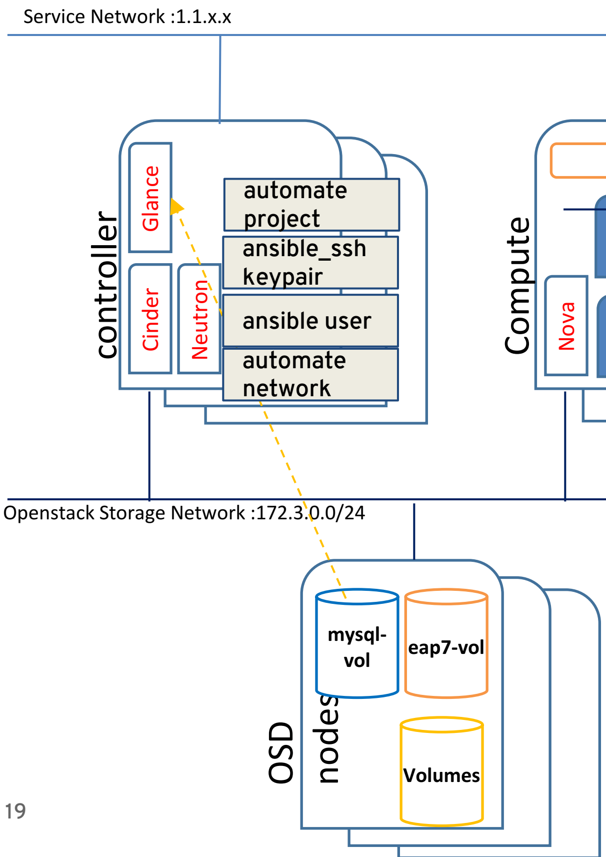
- IPv4 127.0.0.1
- IPv6 0:0:0:0:0:0:0:1
- IPv6 fe80:0:0:0:250:56ff:feab:41f0
- MAC 00:00:00:00:00:00



```

- hosts: all

  vars:
    oscap_profile: xccdf_org.ssgproject.content_profile_pci-dss
    oscap_policy: ssg-rhel-dss

  tasks:

  - name: install openscap scanner
    package:
      name: "{{ item }}"
      state: latest
    with_items:
    - openscap-scanner
    - scap-security-guide

  - block:
    - name: run openscap
      command: oscap xccdf eval \
        --profile {{ oscap_profile }} \
        --results-arf /tmp/oscap-arf.xml \
        --report /tmp/oscap-report.html \
        --fetch-remote-resources \
        /usr/share/xml/scap/ssg/content/{{ oscap_policy }}.xml
```

https://medium.com/@jackprice/ansible-openscap-for-compliance-automation-14200fe70663

# 애플리케이션별 환경 설정



## WEB1-2

httpd.conf 자동 설정 변경
mod_jk.conf
workers.properties

httpd 서비스 실행

## WAS1

standalone.xml 의 DB연결
module.xml 에서 jdbc 등록

jboss eap 서비스 실행

## DB1

my.conf 수정

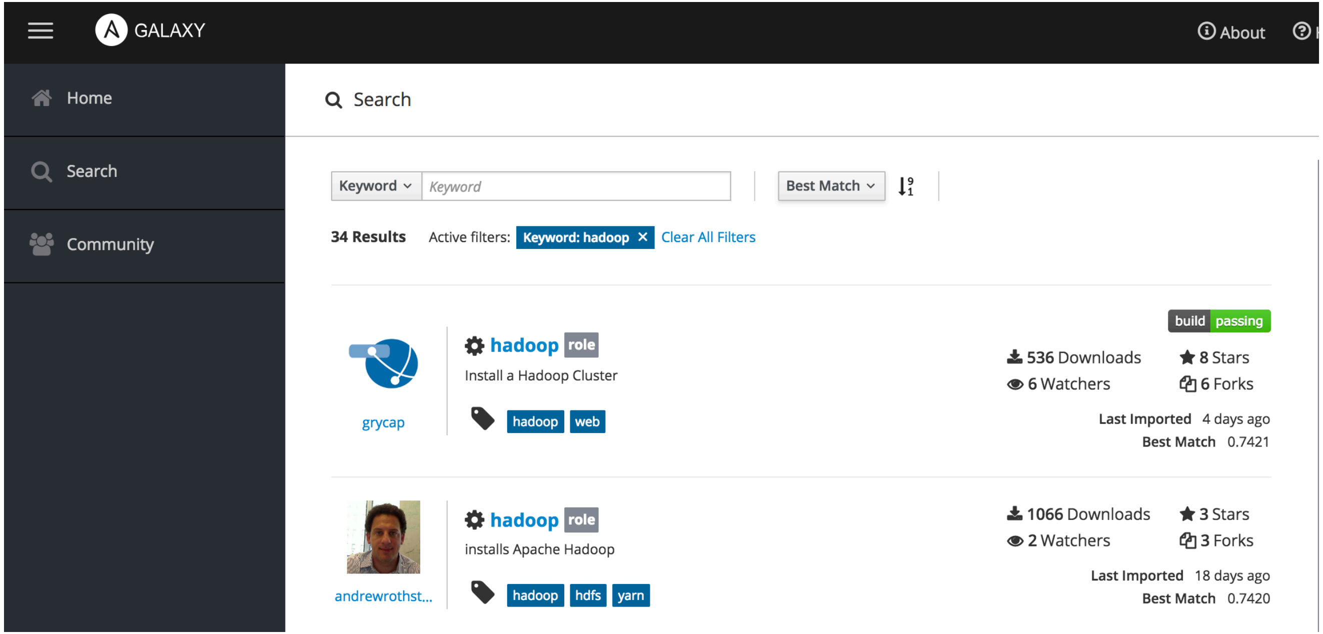
mariadb 서비스 실행

## standalone.xml.template

```
<datasource jta="false" jndi-
name="java:jboss/postgresDS"
pool-name="postgresDS"
enabled="true" use-java-
context="true" use-
ccm="false">

<connection-
url>jdbc:postgresql://{{
hostvars['director']['dblb']
}}:{{ dbport }}/{{ dbsid|upper
}}</connection-url>

<driver-
class>org.postgresql.Driver</d
river-class>

<driver>postgresql</driver>
```

# Ansible Galaxy를 통한 패키지 설치 구성 자동화

https://galaxy.ansible.com/



34 Results Active filters: Keyword: hadoop ✕ Clear All Filters

build passing

hadoop role

Install a Hadoop Cluster

grycap

hadoop web

536 Downloads 8 Stars

6 Watchers 6 Forks

Last Imported 4 days ago

Best Match 0.7421

hadoop role

installs Apache Hadoop

andrewrothst...

hadoop hdfs yarn

1066 Downloads 3 Stars

2 Watchers 3 Forks

Last Imported 18 days ago

Best Match 0.7420

# 애플리케이션 배포

## git과 jenkins를 연동하여 App 자동 배포

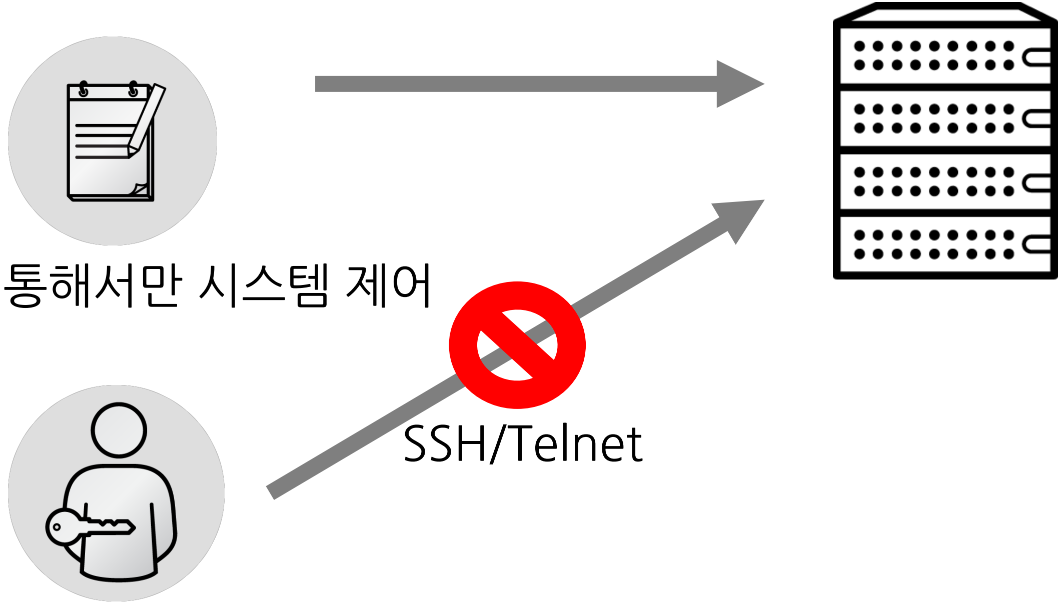
# IAC Best Practices

# IAC Best Practices

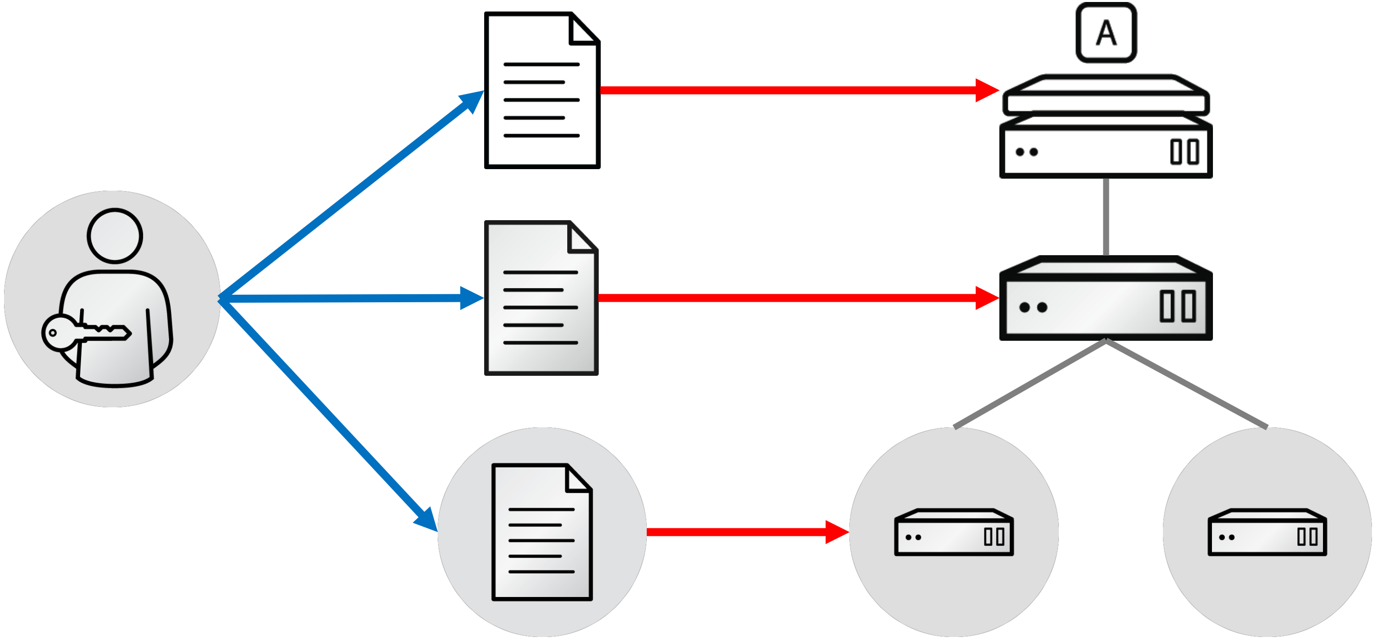
## How to be up to date

- Limited Direct Console Access
- Self-Documentation
- Code Versioning
- Continuous Test & Process
- Keep Services Available

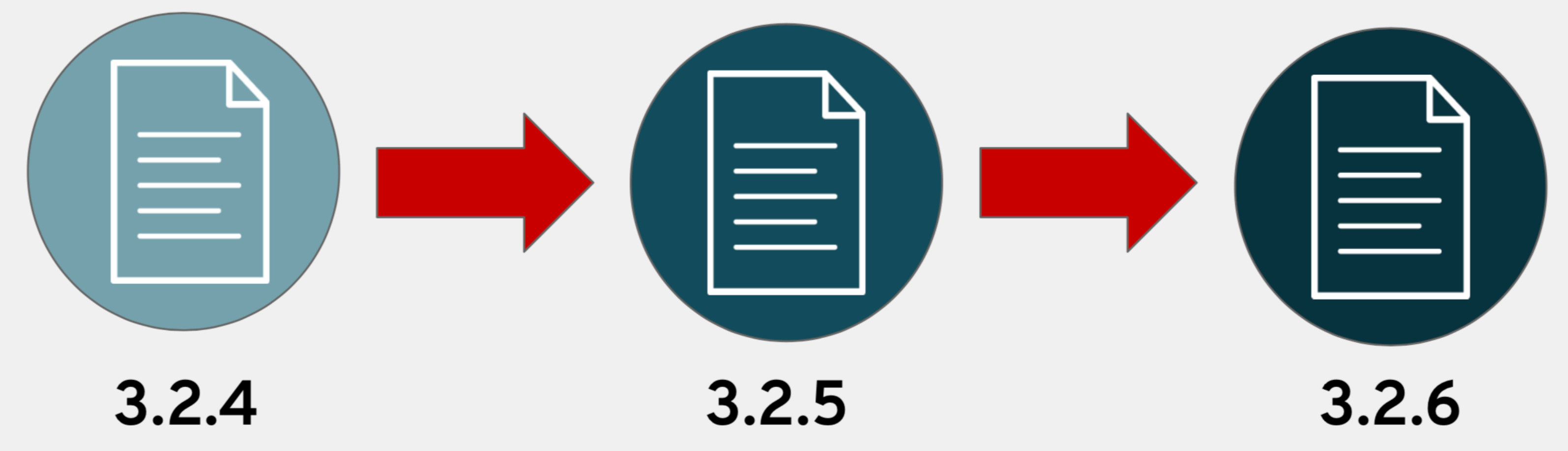
# Limited Direct Console Access



## Self-Documentation



## Code Versioning

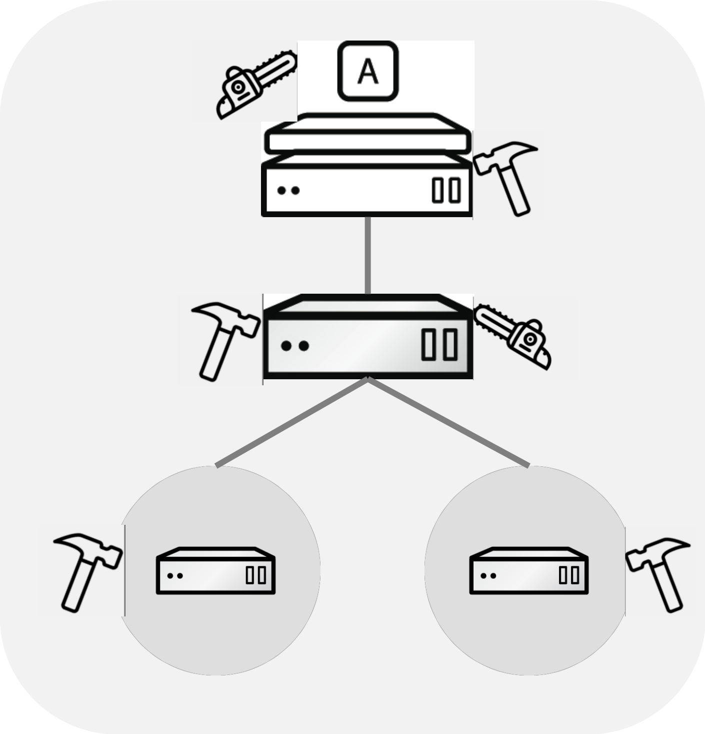


장점:

- 변경 히스토리 관리
- 원복 가능
- 가시성 증대

## Continuous Test & Process

- 테스트 시나리오 현실화
- 테스트 자동화
- 프로세스 표준화
- 사람 간섭 최소화



# PLAYBOOK EXAMPLES

이 문서에서 사용한 예제 사용 코드: HatSAri Github
https://github.com/hatsari/

LAMP + HAPROXY + NAGIOS
github.com/ansible/ansible-examples/tree/master/lamp_haproxy

WINDOWS
github.com/ansible/ansible-examples/tree/master/windows

SECURITY COMPLIANCE
github.com/ansible/ansible-lockdown

NETWORK
github.com/privateip/network-demo

MORE...
galaxy.ansible.com
github.com/ansible/ansible-examples

RED HAT ANSIBLE Tower

# 감사합니다