CCCR IaC 미니 프로젝트

Ansible을 사용한 DB/웹 서버 자동화

과정명	CCCR 4기 – DevOps 과정
팀 원	최휘녕
일 자	2025.05.07 - 2025.05.09

목차

1. 프로젝트 서론

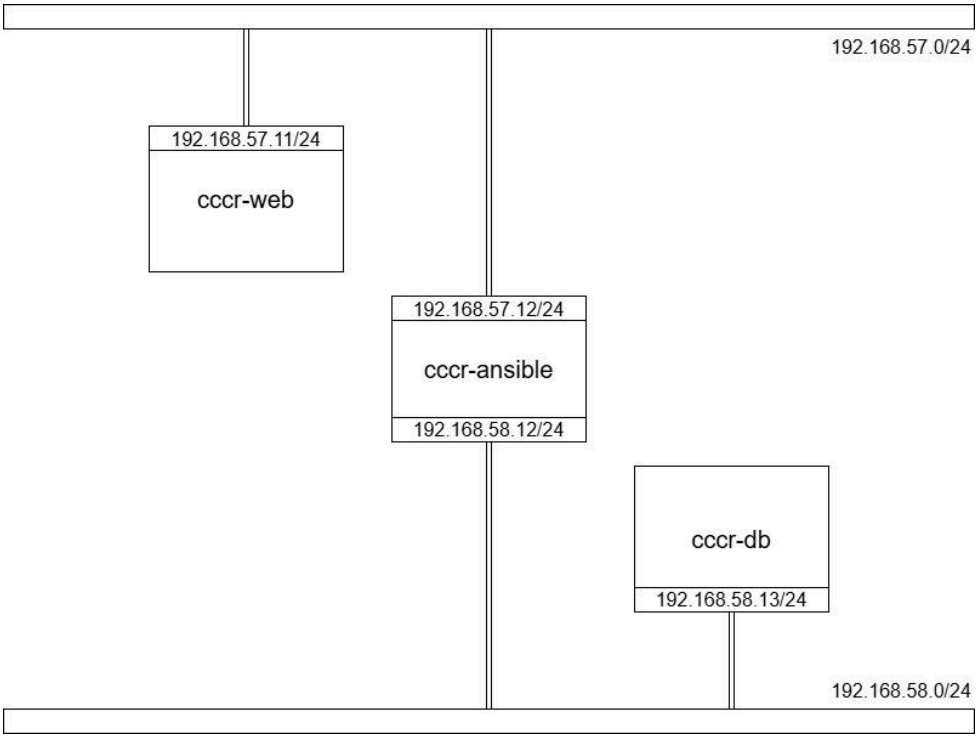
CCCR 4기 DevOps 과정의 일환으로 진행된 IaC(Infrastructure as Code) 강의 말미에는 구성 관리 자동화에 대한 실습 프로젝트가 주어졌습니다. 해당 프로젝트는 강의에서 배운 Ansible을 활용하여 웹 서버와 DB 서버의 초기 환경을 자동으로 구성합니다. 수동 설정 과정에서 발생할 수 있는 오류를 줄이고 재사용 가능한 인프라 자동화 구축을 통해 DevOps의 핵심인 효율성과 안정성을 경험할 수 있는 기회였습니다.

이 장에서는 프로젝트의 개요, 아키텍처 설계 및 요구사항 분석, 그리고 프로젝트 목표 및 기대 효과에 대해 자세히 기술합니다.

1.1 프로젝트 개요

본 프로젝트는 IaC 강의에서 학습한 내용을 실습하기 위한 구성 관리 자동화 프로젝트입니다. Ansible을 활용하여 웹 서버와 DB 서버의 초기 환경을 자동으로 구축하고, 이를 통해 IaC에 대한 이해도를 높이고 패키지 설치, 서비스 설정 등 리눅스 기반 시스템 관리 작업에 대한 경험을 쌓는 것을 목표로 합니다. 프로젝트 범위는 두 서버 간의 연결 설정을 포함하며, 웹에 접속했을 때 정상적으로 화면이 출력될 수 있도록 하는 초기 설정까지 진행됩니다. 본 프로젝트는 개인 단위로 진행되었으며, 전체 기간은 3일입니다.

1.2 아키텍처 설계



본 프로젝트는 총 3개의 VirtualBox 가상 머신을 활용하여 진행됩니다. 각 가상 머신의 호스트명은 각각 cccr-web, cccr-db, cccr-ansible이며, 역할에 따라 다음과 같이 구성됩니다.

cccr-web 가상 머신에는 HTTP 서버와 WordPress를 설치하여 웹 서버를 구축하고,, cccr-db 가상 머신에는 MySQL을 설치하여 DB 서버가 구축됩니다. 이 두 서버는 cccr-ansible 가상 머신에서 실행되는 Ansible을 사용해 자동으로 설치 및 구성됩니다. Ansible 관점에서 cccr-ansible 가상 머신은 제어 노드 역할을 하며 cccr-web, cccr-db 가상 머신은 제어 노드에 의해 관리되는 관리 노드로 동작합니다. 각 가상 머신의 Host-Only 네트워크 구성은 위 그림과 같습니다.

1.3 단계별 구현 과정 설명

Ansible을 활용한 이번 프로젝트는 다음의 단계를 거쳐 진행됩니다.

0단계, Ansible 초기 설정: 인벤토리, ansible.cfg 파일 등의 Ansible 실행을 위한 설정 파일을 작성합니다.

1단계, 플레이북 초기 구현: 변수나 템플릿 등의 기능을 사용하지 않고, 웹 서버와 DB 서버를 각 가상 머신에 구축하는 하나의 플레이북을 작성합니다.

2단계, 변수 활용: 플레이북 내에서 반복적으로 사용되거나 가독성을 향상시키기 위해 분리할 수 있는 문자열을 변수로 정의하고, 이를 활용하여 플레이북을 수정합니다.

3단계, jinja2 템플릿 활용: 반복문, 조건문, 변수 삽입이 필요한 구성 파일은 jinja2 템플릿으로 작성하여 재사용성과 유연성을 높입니다.

4단계, 핸들러와 imports/includes 활용: 수정 사항이 감지되었을 때와 같은 특정 조건에서만 실행되어야 하는 작업은 핸들러로 분리하고, 여러 곳에서 공통적으로 사용되는 작업은 별도의 파일로 작성하여 **imports/includes**를 통해 불러오도록 구성합니다.

5단계, 역할 활용: 작업, 변수, 템플릿 등을 기능 단위로 나누어 역할로 분리함으로써, 구성 요소를 독립적으로 관리하고 실행할 수 있도록 합니다.

1.4 프로젝트 목표 및 기대 효과

본 프로젝트는 인프라 자동화 도구인 Ansible을 이용해 웹 서버와 DB 서버를 자동으로 구축하고 설정하는 과정을 실습함으로써, 반복적인 서버 구축 작업을 효율적으로 처리하는 자동화 경험을 쌓는 것을 목표로 합니다. 특히, 프로젝트를 진행하는 동안 유지보수성과 확장성을 중요한 가치로 삼고 이를 향상시키기 위한 구조와 방식에 대해 깊이 고민하며 설계하고자 합니다.

이 프로젝트를 통해, YAML 문법, 변수 활용법, jinja2 템플릿과 역할의 구조 등 Ansible의 주요 기능을 실습하며 IaC에 대한 이해와 숙련도를 높일 수 있습니다. 또한, 프로젝트 진행 중 발생하는 오류를 직접 해결해나가며 문제 해결 능력을 향상시킬 수 있습니다.

2. 프로젝트 본론

이 장에서 다루는 모든 내용은 Ansible이 실행되는 cccr-ansible 가상 머신에서 진행되는 작업으로 구성되어 있습니다. 이후 절에서는 프로젝트 진행 과정과 트러블슈팅에 대해 자세히 기술합니다.

2.1 프로젝트 진행 과정

1.3 단계별 구현 과정 설명에서 작성한 구현 단계를 따라서 프로젝트를 순차적으로 진행하였습니다. 각 단계는 이전 단계의 결과를 기반으로 확장하거나 개선하는 방식으로 구성되어 있으며, 이에 따라 본 문서에서는 각 단계에서의 이전 단계 구성과 해당 단계에서 어떤 수정이 이루어졌는지를 중점으로 기술합니다. 모든 작업은 cccr-ansible 가상 머신의 project 디렉토리에서 진행됩니다.

2.1.1. 구현 0단계, Ansible 초기 설정

구현 0단계에서는 Ansible 설정을 작성한 ansible.cfg 파일, 웹 서버와 DB 서버를 구축할 cccr-web과 cccr-db의 IP 주소를 작성한 인벤토리 (hosts.ini) 파일을 작성하여 Ansible 초기 설정을 하고 관리할 관리 노드(cccr-web, cccr-db)와 SSH 통신을 위한 키 교환을 합니다.

디렉토리 구조

```
project/
├── ansible.cfg
└── inventory/
    └── hosts.ini
```

이번 단계에서 작성할 파일들은 위와 같은 구조로 구성됩니다.

Ansible 설치

```
[vagrant@cccr-ansible ~]$ sudo dnf install -y ansible
```

인벤토리 (inventory/hosts.ini) 작성

```
# project/inventory/hosts.ini
[webservers]
192.168.57.11

[dbservers]
192.168.58.13
```

cccr-web의 IP 주소(192.168.57.11)를 webservers 그룹의 호스트로 정의하고 cccr-db의 IP 주소(192.168.58.13)는 dbservers 그룹의 호스트로 정의합니다. 추후에 웹 서버를 구축할 호스트가 추가된다면 webservers 그룹명 아래에 작성하면 됩니다.

Ansible 설정 파일 (ansible.cfg) 작성

```
# project/ansible.cfg
[defaults]
inventory=./inventory/hosts.ini
remote_user=vagrant
ask_pass=false

[privilege_escalation]
become=true
become_method=sudo
become_user=root
become_ask_pass=false
```

[defaults] 섹션에서는 앞서 작성한 hosts.ini 파일을 인벤토리로 정의하고 관리 노드에 접속할 기본 원격 사용자로 vagrant를 설정합니다. 또한, SSH 접속 시 비밀번호를 입력하지 않도록 설정하여 작업의 편의성을 높입니다. 비밀번호 없이 접속하기 위해서는 SSH 키 기반 인증 방식을 사용해야 하기 때문에 이후 각 관리 노드와 공개 키를 교환하는 작업을 수행합니다.

[privilege_escalation] 섹션은 원격 사용자의 권한을 상승시켜야 하는 경우에 대한 설정입니다. sudo 명령어를 통해 root 권한을 사용할 수 있도록 하고, 이때도 비밀번호를 입력하지 않도록 설정합니다.

SSH 키 교환

```
[vagrant@cccr-db ~]$ sudo vi /etc/ssh/sshd_config
[vagrant@cccr-web ~]$ sudo vi /etc/ssh/sshd_config
# web과 db에서 모두 PasswordAuthentication yes
[vagrant@cccr-db ~]$ sudo systemctl restart sshd
[vagrant@cccr-web ~]$ sudo systemctl restart sshd
```

SSH 키 기반 인증 방식으로 접속하기 위해 키 교환 작업을 수행합니다. 각 관리 노드인 cccr-web, cccr-db에서 /etc/ssh/sshd_config 파일을 수정해 PasswordAuthentication yes를 작성합니다. 키 교환을 할 때, 관리 노드 원격 사용자의 비밀번호를 입력해야 하기 때문에 이 옵션이 yes로 설정되어 있어야 합니다.

```
[vagrant@cccr-ansible project]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vagrant/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vagrant/.ssh/id_rsa
Your public key has been saved in /home/vagrant/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:EezXjWH9EapYCxW/vDQZyz/h3jWqjI/n5UvfNLFXu5c vagrant@cccr-ansible
The key's randomart image is:
+---[RSA 3072]----+
|       ..  o.. ..|
|        ... + o. |
|       ... + B ..|
|        ..= B * .|
|        So o O o.|
|            . = *|
|             + O=|
|          +.+ +E@|
|         o+=.+.++|
+----[SHA256]-----+

[vagrant@cccr-ansible project]$ ssh-copy-id vagrant@192.168.58.13
[vagrant@cccr-ansible project]$ ssh-copy-id vagrant@192.168.57.11
```

이제 제어 노드(cccr-ansible)로 돌아와 ssh-keygen 명령어를 사용하여 SSH 키 페어를 생성합니다. 별도로 지정하지 않으면 기본적으로 id_rsa, id_rsa_pub 파일이 생성됩니다.

ssh-copy-id 명령어를 사용하여 공개 키를 각 관리 노드와 복사함으로써 비밀번호 없이 키 기반 인증 방식으로 SSH 연결이 가능하게 됩니다. 이때도 키 파일을 별도로 지정하지 않으면 기본 공개 키(id_rsa.pub)가 사용됩니다.

2.1.2. 구현 1단계, 플레이북 초기 구현

구현 1단계에서는 관리 노드에서 웹 서버와 DB 서버를 구축하는 작업들을 플레이북에 작성합니다.

디렉토리 구조

```
project/
├── ansible.cfg
├── inventory/
│   └── hosts.ini
└── playbook.yml    # 추가
```

이번 단계의 디렉토리 구조는 위와 같은 구조로 구성됩니다. 이전 0단계의 구조에서 playbook.yml 파일이 추가됩니다.

플레이북 작성

```
# project/playbook.yml
---
- name: configure DB server
  hosts: dbservers
  tasks:

---
- name: configure Web server
  hosts: webservers
  tasks:
```

project/playbook.yml 파일에 플레이북의 큰 틀을 작성합니다. 이 플레이북은 두 개의 플레이로 구성되어 있습니다.

현재 프로젝트에서는 dbservers와 webservers 그룹에 현재 각 하나의 관리 노드만 포함되어 있지만, 확장성을 고려하여 단일 호스트가 아닌 호스트 그룹으로 지정하였습니다.

1) configure DB server 플레이

첫 번째 플레이는 configure DB server입니다. dbservers 그룹에 속한 호스트에서 실행되고 DB 서버를 구성하는 작업을 수행합니다.

```
- name: install packages for DB servers
  dnf:
    name:
      - mysql-server
      - python3-pexpect
      - python3-PyMySQL
    state: present
```

DB 서버를 구성하는 관리 노드에 필요한 패키지를 설치하는 작업입니다.

DB를 사용하기 위한 mysql-server, secure MySQL installation 작업에서 expect 모듈을 사용하기 위한 python3-pexpect, create databse, create db user and grant privileges 작업에서 mysql 관련 모듈을 사용하기 위한 python3-PyMySQL 패키지를 설치합니다.

state: present를 설정하면 해당 패키지가 이미 설치되어 있는 경우 별도로 재설치하지 않습니다. 이를 통해 멱등성을 보장합니다.

```
- name: configure firewall for MySQL
  firewalld:
    service: mysql
    permanent: yes
    state: enabled
    immediate: yes
```

MySQL에 웹 서버가 접속할 수 있도록 방화벽을 열어주는 작업입니다.

mysql 서비스를 등록하고 permanent=yes 옵션을 통해 재부팅 후에도 설정이 유지되도록 하며, 변경 사항은 즉시 적용됩니다.

```
- name: enable and start MySQL
  service:
    name: mysqld
    state: started
    enabled: yes
```

MySQL 데몬(mysqld)을 실행시키는 작업입니다.

mysqld 서비스를 시작하고 enabled=yes 옵션을 통해 재부팅 후에도 설정이 유지되도록 합니다.

```
- name: setup MySQL if NOT initial setup
  expect:
    command: mysql_secure_installation
    responses:
      'Would you like to setup VALIDATE PASSWORD component' : 'n'
      'New password:' : 'admin123'
      'Re-enter new password:' : 'admin123'
      'Remove anonymous users?' : 'y'
      'Disallow root login remotely?' : 'y'
      'Remove test database and access to it?' : 'y'
      'Reload privilege tables now?' : 'y'
  when: mysql_check_result.rc == 0
```

MySQL을 설치하고 데이터베이스를 사용하기 전에, mysql_secure_installation 명령어를 사용하여 초기 설정을 진행하는 작업입니다.

이 작업에서 사용된 expect는 대화형 명령어 실행 시 자동으로 응답을 제공해주는 모듈입니다.

mysql_secure_installation 명령어는 총 7가지 프롬포트에 수동으로 응답해야 하는데, responses 파라미터를 이용하면 각 프롬프트 메시지의 일부분을 작성해 해당 프롬포트에 대한 응답을 지정할 수 있습니다. 이 작업에서는 'Would you like to setup VALIDATE PASSWORD component' 라는 문자열이 포함된 프롬포트에 'n'이라고 응답하도록 설정한 것입니다.

responses 파라미터에는 실제 출력되는 프롬프트 메시지의 일부분을 정확히 입력해야 합니다. 하지만 MySQL 초기 설정이 이미 완료된 상태라면, 즉 MySQL root 사용자의 비밀번호가 설정되어 있다면 출력되는 프롬프트가 달라집니다.

이 경우에는 expect 작업이 실패하게 되므로, 초기 설정이 완료된 경우 해당 작업을 실행하지 않도록 조건을 설정해야 합니다.

```
- name: check initial setup
  command: mysql -u root -e "SELECT 1;"
  register: mysql_check_result
  ignore_errors: yes
```

조건 설정을 위해 초기 설정 여부를 확인하는 check initial setup 작업을 추가했습니다.

이 작업은 MySQL 서버에 root 사용자로 비밀번호 없이 접속해 'SELECT 1' 쿼리를 실행하고 결과를 mysql_check_result 변수에 저장합니다. 이 변수에 저장되는 rc 값은 명령 실행 결과의 상태 코드를 의미하며, 성공 시 0, 실패 시 1이 저장됩니다. Ansible 플레이북은 기본적으로 작업이 실패하면 이후 작업을 실행하지 않고 중단됩니다. 그러나나 이 작업은 실패해도 영향이 없기 때문에, ignorre_errors: yes를 설정하여 실패하더라도 다음 작업이 계속 실행되도록 합니다.

비밀번호 없이 접속이 성공했다면, MySQL 초기 설정이 되지 않은 상태라고 판단할 수 있으므로 초기 설정을 진행하는 작업의 조건에 mysql_check_result.rc == 0 을 설정합니다.

```
- name: create database
  mysql_db:
    login_user: root
    login_password: admin123
    state: present
        name: cccr
```

MySQL에 웹 서버에서 사용할 데이터베이스를 생성하는 작업입니다.

root 사용자로 접속해 cccr 데이터베이스를 생성합니다. state: present를 설정하여 멱등성을 보장합니다.

```
- name: create db user and grant privileges
  mysql_user:
    login_user: root
    login_password: admin123
    state: present
    name: admin
    password: admin123
    host: 192.168.58.11
    priv: "cccr.*:SELECT,INSERT,UPDATE,DELETE,CREATE,ALTER,INDEX,CREATE
TEMPORARY TABLES,LOCK TABLES"
```

MySQL에 웹 서버에서 사용할 사용자를 생성하는 작업입니다.

root 사용자로 접속해 admin@192.168.58.11 사용자를 생성하고 비밀번호는 admin123으로 설정합니다. 이때, 호스트는 웹 서버의 IP 주소입니다. state: present를 설정하여 멱등성을 보장합니다.

웹 서버는 WordPress로 구현되기 때문에, WordPress에서 사용하려면 필요한 권한만 부여하도록 설정합니다.

2) configure Web server 플레이

두 번째 플레이는 configure Web server입니다. webservers 그룹에 속한 호스트에서 실행되고 웹 서버를 구성하는 작업을 수행합니다.

```
- name: install packages for web servers
  dnf:
    name:
      - php
      - httpd
      - php-mysqlnd
      - policycoreutils-python-utils
    state: present
```

웹 서버를 구성하는 관리 노드에 필요한 패키지를 설치하는 작업입니다.

웹 서버 구축에 필요한 php, httpd, php-mysqlnd, SELinux boolean 설정을 위한 policycoreutils-python-utils 패키지를 설치합니다.

이 작업에서도 state: present를 설정하여 멱등성을 보장합니다.

```
- name: configure firewall for http
  firewalld:
    service: http
    permanent: yes
    state: enabled
    immediate: yes
```

구성한 웹 서버에 외부에서 접속할 수 있도록 방화벽을 열어주는 작업입니다.

http 서비스를 등록하고 permanent=yes 옵션을 통해 재부팅 후에도 설정이 유지되도록 하며, 변경 사항은 즉시 적용됩니다.

```
- name: enable and start httpd
  service:
    name: httpd
    state: started
    enabled: yes
```

HTTP 데몬(httpd)을 실행시키는 작업입니다.

httpd 서비스를 시작하고 enabled=yes 옵션을 통해 재부팅 후에도 설정이 유지되도록 합니다.

```
- name: download wordpress archive
  command: wget -O wordpress.tar.gz https://wordpress.org/latest.tar.gz
```

WordPress를 다운로드하는 작업입니다.

wget 명령어를 사용하여 가장 최근 버전의 WordPress 파일을 다운로드하여 wordpress.tar.gz라는 이름으로 저장합니다.

```
- name: untar wordpress archive
  command: tar -zxvf wordpress.tar.gz -C /var/www/html/
```

이전 작업에서 다운로드 받은 파일의 압축을 해제하는 작업입니다.

tar 명령어를 사용하여 wordpress.tar.gz 파일의 압축을 해제하여 /var/www/html/ 디렉토리에 저장합니다. 이 과정을 통해서, /var/www/html/wordpress 파일이 생성됩니다.

```
- name: configure wordpress
  copy:
    src: wp-config.php
    dest: /var/www/html/wordpress/wp-config.php
```

WordPress 설정 파일을 복사하는 작업입니다.

copy 모듈은 제어 노드에 있는 WordPress 설정 파일을 관리 노드에 복사해서 저장하는 작업입니다.

이 작업은 제어 노드의 wp-config.php 파일을 관리 노드의 /var/www/html/wordpress 디렉토리에 wp-config.php라는 이름으로 저장합니다.

```
define( 'DB_NAME', 'cccr' );
define( 'DB_USER', 'admin' );
define( 'DB_PASSWORD', 'admin123' );
define( 'DB_HOST', '192.168.58.13' );
```

제어 노드의 wp-config.php 파일은 wp-config-sample.php 파일에서 DB 설정인 DB_NAME, DB_USER, DB_PASSWORD, DB_HOST를 정의하는 부분을 위 코드와 같이 수정한 파일입니다.

```
- name: create virtual host
  copy:
    src: vhosts.conf
    dest: /etc/httpd/conf.d/vhosts.conf
```

가상 호스트 파일을 복사하는 작업입니다.

현재 설정으로는, 웹 서버에 접속했을 때 /var/www/html/index.html 파일이 시작점으로 되어 있습니다. 웹 서버의 구성 파일은 /var/www/html/wordpress 디렉토리가 시작점이기 때문에 이 설정을 해주기 위해서는 가상 호스트 파일을 웹 서버에 생성해야 합니다.

이 작업은 제어 노드의 vhosts.conf 파일을 관리 노드의 /etc/httpd/conf.d 디렉토리에 vhosts.conf라는 이름으로 저장합니다.

```
# project/vhosts.conf
<VirtualHost *:80>
        ServerName miniproject.com
        DocumentRoot "/var/www/html/wordpress"

        <Directory "/var/www/html/wordpress">
                AllowOverride None
                Require all granted
        </Directory>
</VirtualHost>
```

제어 노드의 vhosts.conf 파일은 위처럼 작성합니다.

```
- name: restart httpd
  service:
    name: httpd
    state: restarted
```

HTTP 데몬을 재시작하는 작업입니다.

가상 호스트 파일을 생성함으로써 HTTP 설정이 변경되었기 때문에 데몬 서비스를 재시작합니다.

```
- name: enable SELinux boolean httpd_can_network_connect_db
  seboolean:
    name: httpd_can_network_connect_db
    persistent: yes
    state: yes
```

httpd_can_network_connect_db SELinux boolean을 켜는 작업입니다.

웹 서버에서 DB 서버의 데이터베이스를 사용하기 위해 네트워크 연결을 할 수 있게 허용하는 SELinux boolean인 httpd_can_network_connect_db를 켭니다.

이제 플레이북을 실행한 다음, 웹에 접속해 보면 DB 서버의 데이터베이스를 사용하여 성공적으로 연결되는 것을 확인할 수 있습니다.

2.1.3. 구현 2단계, 변수 활용

구현 2단계에서는 플레이북 내 작업들 중 반복되거나 변경 가능성이 있는 문자열을 변수로 정의하여 재사용할 수 있도록 수정합니다.

또한, 호스트 그룹별로 서로 다른 SSH 키를 사용하도록 구성함으로써 하나의 키가 유출되었을 때 발생할 수 있는 피해 범위를 최소화합니다.

디렉토리 구조

```
project/
├── ansible.cfg
├── inventory/
│   ├── hosts.ini
│   └── group_vars/          # 추가
│       └── all.yml
│       └── dbservers.yml
│       └── webservers.yml
└── playbook.yml
```

이번 단계의 디렉토리 구조는 위와 같은 구조로 구성됩니다. 이전 1단계 구조에 group_vars 디렉토리가 추가되어 호스트 그룹별로 변수를 정의합니다.

group_vars 디렉토리에 호스트 그룹 이름과 동일한 .yml 파일을 생성하면, 해당 그룹에 속한 모든 호스트에서 공통으로 사용하는 변수를 정의할 수 있습니다.

변경 내용

1) 호스트 그룹별 SSH 키 분리

```
[vagrant@cccr-db ~]$ sudo vi ~/.ssh/authorized_keys
[vagrant@cccr-web ~]$ sudo vi ~/.ssh/authorized_keys
```

~/.ssh/authorized_keys 파일은 SSH 접속을 허용할 공개 키를 저장하는 파일입니다.

호스트 그룹별로 서로 다른 공개 키를 사용하도록 구성하기 위해, cccr-mysql 가상 머신과 cccr-web 가상 머신에 기존에 저장되어 있는 공통 공개 키를 삭제합니다.

```
[vagrant@cccr-ansible project]$ ssh_keygen -f ~/.ssh/id_rsa_db  # DB 서버용 키
[vagrant@cccr-ansible project]$ ssh_keygen -f ~/.ssh/id_rsa_web # Web 서버용 키
```

ssh_keygen 명령어로 id_rsa_db, id_rsa_web 키 페어를 생성합니다.

```
[vagrant@cccr-ansible project]$ ssh-copy-id -i /home/vagrant/.ssh/id_rsa_db.pub vagrant@192.168.58.13        # DB 서버에 복사
[vagrant@cccr-ansible project]$ ssh-copy-id -i /home/vagrant/.ssh/id_rsa_web.pub vagrant@192.168.57.11       # Web 서버에 복사
```

ssh-copy-id 명령어의 i 옵션을 활용하여 cccr-db(192.168.58.13) 가상 머신에는 id_rsa_db.pub 공개 키를, cccr-web(192.168.57.11) 가상 머신에는 id_rsa_web.pub 공개 키를 복사합니다.

```
# project/inventory/group_vars/dbservers.yml
ansible_ssh_private_key_file: ~/.ssh/id_rsa_db

# project/inventory/group_vars/webservers.yml
ansible_ssh_private_key_file: ~/.ssh/id_rsa_web
```

ansible_ssh_private_key_file 변수는 Ansible이 SSH 접속 시 사용하는 개인 키의 경로입니다.

dbservers.yml 파일에는 ~/.ssh/id_rsa_db 개인 키를, webservers.yml 파일에는 ~/.ssh/id_rsa_web 개인 키를 지정하여 그룹별로 지정된 SSH 키를 사용하도록 설정합니다.

2) 변수 정의

all.yml - 모든 호스트 공통 변수

```
# project/inventory/group_vars/all.yml
db_name: cccr
db_user: admin
db_password: admin123
db_host: 192.168.58.13
```

DB 서버에서 데이터베이스와 사용자를 생성할 때 사용되고, 웹 서버에서는 DB 서버에 접속할 때 사용되는 값을 정의합니다.

dbservers.yml - DB 서버 전용 변수

```
# project/inventory/group_vars/dbservers.yml
ansible_ssh_private_key_file: ~/.ssh/id_rsa_db

db_packages:
  - mysql-server
  - python3-pexpect
  - python3-PyMySQL
db_firewalld_service: mysql
db_daemon_service: mysqld
db_root_password: admin123
```

이후에 추가로 설치할 패키지가 생길 가능성에 대비하여, 설치할 패키지 목록을 변수로 정의합니다.

또한, 방화벽 서비스 이름과 데몬 서비스 이름은 이후 단계에서 재사용 가능한 작업으로 만들기 위해 변수로 선언합니다.

데이터베이스의 root 사용자 비밀번호는 보안성과 유연성을 고려하여 변수로 관리하며, 변경이 필요한 경우 쉽게 수정할 수 있도록 합니다.

webservers.yml - 웹 서버 전용 변수

```yaml
# project/inventory/group_vars/webservers.yml
ansible_ssh_private_key_file: ~/.ssh/id_rsa_web

web_packages:
  - php
  - httpd
  - php-mysqlnd
  - policycoreutils-python-utils
web_firewalld_service: http
web_daemon_service: httpd
web_sebool: httpd_can_network_connect_db
```

webservers.yml 파일의 변수들 역시 dbservers.yml 파일과 같은 목적으로 정의합니다.

3) 플레이북 수정

configure DB server 플레이 수정

```
- name: install packages for DB servers
  dnf:
    name: "{{ db_packages }}"
    state: present

- name: configure firewall for MySQL
  firewalld:
    service: "{{ db_firewalld_service }}"
    permanent: yes
    state: enabled
    immediate: yes

- name: enable and start MySQL
  service:
    name: "{{ db_daemon_service }}"
    state: started
    enabled: yes

- name: check initial setup
  command: mysql -u root -e "SELECT 1;"
  register: mysql_check_result
  ignore_errors: yes

- name: setup MySQL if NOT initial setup
  expect:
    command: mysql_secure_installation
    responses:
      'Would you like to setup VALIDATE PASSWORD component': 'n'
      'New password:': "{{ db_root_password }}"
      'Re-enter new password:': "{{ db_root_password }}"
      'Remove anonymous users?': 'y'
      'Disallow root login remotely?': 'y'
      'Remove test database and access to it?': 'y'
      'Reload privilege tables now?': 'y'
  when: mysql_check_result.rc == 0
```

```
  - name: create database
    mysql_db:
      login_user: root
      login_password: "{{ db_root_password }}"
      name: "{{ db_name }}"
      state: present

  - name: create db user and grant privileges
    mysql_user:
      login_user: root
      login_password: "{{ db_root_password }}"
      name: "{{ db_user }}"
      password: "{{ db_password }}"
      host: 192.168.58.11
      priv: "{{ db_name
  }}.*:SELECT,INSERT,UPDATE,DELETE,CREATE,ALTER,INDEX,CREATE TEMPORARY
  TABLES,LOCK TABLES"
      state: present
```

1단계의 configure DB server 플레이 작업들 중에서는 MySQL 초기 설정을 확인하는 작업을 제외하고 모든 작업을 변수를 활용한 작업으로 수정합니다.

configure Web server 플레이 수정

```
  - name: install packages for web servers
    dnf:
      name: "{{ web_packages }}"
      state: present

  - name: configure firewall for http
    firewalld:
      service: "{{ web_firewalld_service }}"
      permanent: yes
      state: enabled
      immediate: yes

  - name: enable and start httpd
    service:
      name: "{{ web_daemon_service }}"
      state: started
      enabled: yes
```

```
  - name: download wordpress archive
    command: wget -O wordpress.tar.gz https://wordpress.org/latest.tar.gz

  - name: untar wordpress archive
    command: tar -zxvf wordpress.tar.gz -C /var/www/html/

  - name: configure wordpress
    copy:
      src: wp-config.php
      dest: /var/www/html/wordpress/wp-config.php

  - name: create virtual host
    copy:
      src: vhosts.conf
      dest: /etc/httpd/conf.d/vhosts.conf

  - name: restart httpd
    service:
      name: "{{ web_daemon_service }}"
      state: restarted

  - name: enable SELinux boolean httpd_can_network_connect_db
    seboolean:
      name: "{{ web_sebool }}"
      state: yes
      persistent: yes
```

1단계의 configure DB server 플레이 작업들 중에서는 WordPress를 다운로드하고 압축 파일을 해제하는 작업들과 템플릿에서 변수를 사용해야 하는 WordPress 설정 파일과 가상 호스트 파일을 관리 노드에 저장하는 작업들을 제외하고 모든 작업을 변수를 활용한 작업으로 수정합니다.

2.1.4. 구현 3단계, jinja2 템플릿 활용

구현 3단계에서는 플레이북 내 작업 중 템플릿을 적용할 수 있는 파일들을 jinja2 템플릿으로 수정합니다.

configure Web server 플레이에 포함된 WordPress 설정 파일과 가상 호스트 파일을 템플릿으로 작성하여 변수 활용이 가능하도록 개선합니다.

디렉토리 구조

```
project/
├── ansible.cfg
├── inventory/
│   ├── hosts.ini
│   └── group_vars/
│       └── ...
├── templates/       # 추가
│   └── ...
└── playbook.yml
```

이번 단계의 디렉토리 구조는 위와 같은 구조로 구성됩니다. 이전 2단계 구조에 templates 디렉토리가 추가되어 템플릿 파일들을 저장하는 용도로 사용됩니다.

변경 내용

이전 단계에서는 WordPress 설정 파일과 가상 호스트 파일을 copy 모듈로 관리 노드에 복사했지만, 3단계에서는 template 모듈과 템플릿 파일을 사용합니다.

1) WordPress 설정 파일

```
define( 'DB_NAME', "{{ db_name }}" );
define( 'DB_USER', "{{ db_user }}" );
define( 'DB_PASSWORD', "{{ db_password }}" );
define( 'DB_HOST', "{{ db_host }}" );
```

1단계에서 작성한 제어 노드의 wp-config.php 파일의 DB 설정인 DB_NAME, DB_USER, DB_PASSWORD, DB_HOST를 정의하는 부분을 위 코드와 같이 변수를 활용하여 수정하고 templates/wp-config.php.j2라는 이름으로 저장하여 템플릿으로 생성합니다.

작업 수정 전

```
- name: configure wordpress
  copy:
    src: wp-config.php
    dest: /var/www/html/wordpress/wp-config.php
```

작업 수정 후

```
- name: configure wordpress
  copy:
    src: wp-config.php
    dest: /var/www/html/wordpress/wp-config.php
```

2) 가상 호스트 파일

```
# project/inventory/group_vars/webserver.yml
web_domain_name: miniproject.com
```

ServerName에 사용할 도메인 값을 변수화하기 위해 web_domain_name 변수를 group_vars/webservers.yml에 추가합니다.

vhosts.conf.j2

```
<VirtualHost *:80>
        ServerName "{{ web_domain_name }}"
        DocumentRoot "/var/www/html/wordpress"

        <Directory "/var/www/html/wordpress">
                AllowOverride None
                Require all granted
        </Directory>
</VirtualHost>
```

작업 수정 전

```
- name: create virtual host
  copy:
    src: vhosts.conf
    dest: /etc/httpd/conf.d/vhosts.conf
```

작업 수정 후

```
- name: create virtual host
  template:
    src: templates/vhosts.conf.j2
    dest: /etc/httpd/conf.d/vhosts.conf
```

2.1.5. 구현 4단계, 핸들러와 import/include 활용

구현 4단계에서는 플레이북 내 작업 중 재사용 가능한 작업들을 tasks 디렉토리에 분리하여 작성하고 import_tasks/include_tasks를 사용해 불러오도록 구성합니다. 또한, 핸들러로 작성하는 것이 효율적인 작업들은 플레이북 하단에 handlers로 정의합니다.

설정 파일을 수정한 뒤 데몬 서비스를 재시작해야 하는 경우, 해당 작업은 파일 수정이 감지되었을 때만 수행되어야 합니다. 웹 서버의 가상 호스트 파일을 설정한 후 데몬을 재시작하는 작업은 핸들러로 작성합니다.

DB 서버와 웹 서버에서 공통으로 수행되는 작업인 패키지 설치, 방화벽 설정, 데몬 설정, SELinux boolean 작업은 중복을 줄이기 위해 별도 파일로 분리하고 import_tasks/include_tasks를 사용하여 호출합니다.

디렉토리 구조

```
project/
├── ansible.cfg
├── inventory/
│   ├── hosts.ini
│   └── group_vars/
│       └── ...
├── tasks/        # 추가
│   ├── install_package.yml
│   ├── configure_firewall.yml
│   ├── configure_service.yml
│   └── enable_sebool.yml
├── templates/
│   └── ...
└── playbook.yml
```

이번 단계의 디렉토리 구조는 위와 같은 구조로 구성됩니다. 이전 3단계 구조에 tasks 디렉토리가 추가되어 import_tasks/include_tasks를 사용하여 불러올 작업들을 저장합니다.

변경 내용

1) 핸들러 활용

configure Web server 플레이에서 create virtual host와 restart httpd 작업을 수정합니다.

핸들러 작성

```
handlers:
  - name: restart httpd
    ansible.builtin.service:
      name: "{{ web_daemon_service }}"
      state: restarted
```

restart httpd 작업을 플레이북 하단에 handlers로 작성합니다.

작업 수정 전

```
    - name: create virtual host
      template:
        src: templates/vhosts.conf.j2
        dest: /etc/httpd/conf.d/vhosts.conf
```

작업 수정 후

```
- name: create virtual host
  template:
    src: templates/vhosts.conf.j2
    dest: /etc/httpd/conf.d/vhosts.conf
  notify: restart httpd
```

create virtual host 작업에 notify 파라미터를 넣어서 해당 작업의 결과가 changed일 경우,

restart httpd 핸들러를 실행하도록 합니다.

2) import_tasks/include_tasks로 불러올 작업 작성

tasks/install_package.yml

```
# project/tasks/install_package.yml
- name: install package
  dnf:
    name: "{{ package }}"
    state: present
```

패키지를 설치하는 작업을 작성합니다. 변수 이름은 package로 지정하여 직관적으로 확인할 수 있도록 합니다.

해당 작업은 아래 두 개의 작업을 다음과 같이 수정하여 사용합니다.

```
- name: install packages for DB servers
  include_tasks: tasks/install_package.yml
  loop: "{{ db_packages }}"
  loop_control:
    loop_var: package
```

```
- name: install packages for web servers
  include_tasks: tasks/install_package.yml
  loop: "{{ web_packages }}"
  loop_control:
    loop_var: package
```

loop_control 파라미터의 loop_var에 package를 지정하여 변수 리스트의 각 항목이 package 변수에 매핑되도록 합니다. 리스트가 아닌 단일 값일 경우에는는 vars: package: “{{ package }} “처럼 지정하면 됩니다.

tasks/configure_firewall.yml

```
# project/tasks/configure_firewall.yml
- name: configure firewall
  firewalld:
    service: "{{ firewalld_service }}"
    permanent: yes
    state: enabled
    immediate: yes
```

방화벽을 설정하는 작업입니다.

해당 작업은 아래 두 개의 작업을 다음과 같이 수정하여 사용합니다.

```
- name: configure firewall for MySQL
  import_tasks: tasks/configure_firewall.yml
  vars:
    firewalld_service: {{ db_firewalld_service }}
```

```
- name: configure firewall for http
  import_tasks: tasks/configure_firewall.yml
  vars:
    firewalld_service: "{{ web_firewalld_service }}"
```

tasks/configure_service.yml

```
# project/tasks/configure_service.yml
- name: configure service
  service:
    name: "{{ daemon_service }}"
    state: started
    enabled: yes
```

데몬 설정 작업입니다.

해당 작업은 아래 두 개의 작업을 다음과 같이 수정하여 사용합니다.

```
- name: enable and start MySQL
  import_tasks: tasks/configure_service.yml
  vars:
    daemon_service: "{{ db_daemon_service }}"
```

```
- name: enable and start MySQL
  import_tasks: tasks/configure_service.yml
  vars:
    daemon_service: "{{ web_daemon_service }}"
```

tasks/enable_sebool.yml

```
# project/tasks/enable_sebool.yml
- name: enable SELinux boolean
  seboolean:
    name: "{{ sebool }}"
    persistent: yes
    state: yes
```

SELinux boolean 활성화 작업입니다. 이 작업은 현재 중복되는 작업은 아니지만 중복될 가능성을 고려하여 별도의 작업으로 분리했습니다.

해당 작업은 아래의 작업을 다음과 같이 수정하여 사용합니다.

```
- name: enable SELinux boolean httpd_can_network_connect_db
  import_tasks: tasks/enable_sebool.yml
  vars:
    sebool: "{{ web_sebool }}"
```

2.1.6. 구현 5단계, 역할 활용

구현 5단계에서는 역할을 사용하여 기능별로 독립적인 작업 수행이 가능하도록 수정합니다.

본 프로젝트에서는 웹 서버와 DB 서버의 초기 구성을 구현하므로 comon, web, db 세 가지 역할을 사용합니다.

디렉토리 구조

```
project
├── ansible.cfg
├── inventory
│   ├── group_vars
│   │   └── ...
│   └── hosts.ini
└── site
    ├── site.yaml
    └── roles
        ├── common
        │   └── ...
        ├── db
        │   └── ...
        └── web
            └── ...
```

이번 단계의 디렉토리 구조는 위와 같은 구조로 구성됩니다. 이전 4단계 구조를 역할을 사용하여 재배치합니다.

역할 디렉토리는 ansible-galaxy init 명령어를 사용해 생성합니다.

변경 내용

1) 변수 이름 변경

common 역할에서 동일한 작업을 서로 다른 호스트 그룹에서 실행하려면 기존에 db_packages, web_packages처럼 구분되어 있던 변수들의 이름을 통일해야 합니다.

dbservers.yml

수정 전

```
ansible_ssh_private_key_file: ~/.ssh/id_rsa_db

db_packages:
  - mysql-server
  - python3-pexpect
  - python3-PyMySQL
db_firewalld_service: mysql
db_daemon_service: mysqld
db_root_password: admin123
```

수정 후

```
ansible_ssh_private_key_file: ~/.ssh/id_rsa_db

packages:
  - mysql-server
  - python3-pexpect
  - python3-PyMySQL
firewalld_service: mysql
daemon_service: mysqld
db_root_password: admin123
```

webservers.yml

수정 전

```
ansible_ssh_private_key_file: ~/.ssh/id_rsa_web

web_packages:
  - php
  - httpd
  - php-mysqlnd
  - policycoreutils-python-utils
web_firewalld_service: http
web_daemon_service: httpd
web_sebool: httpd_can_network_connect_db
web_domain_name: miniproject.com
```

수정 후

```
ansible_ssh_private_key_file: ~/.ssh/id_rsa_web

packages:
  - php
  - httpd
  - php-mysqlnd
  - policycoreutils-python-utils
firewalld_service: http
daemon_service: httpd
web_sebool: httpd_can_network_connect_db
web_domain_name: miniproject.com
```

2) common 역할 구성

common 역할에는 웹 서버와 DB 서버에서 공통으로 진행하는 작업을 배치합니다.

패키지 설치, 방화벽, 데몬 설정 작업과 구현 4단계에서 작성한 핸들러, SELinux boolean 활성화 작업을 common 역할 디렉토리에 작성합니다.

common/handlers 디렉토리

main.yml

```
      ---
# handlers file for site/roles/common
- name: restart daemon service
  ansible.builtin.service:
    name: "{{ daemon_service }}"
    state: restarted
```

daemon_service 변수의 값을 이름으로 하는 데몬 서비스를 재시작하는 핸들러입니다.

이 핸들러는 web 역할의 작업에서 호출하지만, DB 역할에서도 사용될 가능성을 고려하여 common 역할에 배치합니다.

common/tasks 디렉토리

main.yml

```
- name: install package
  dnf:
    name: "{{ item }}"
    state: present
  loop: "{{ packages | list }}"
```

```
- name: configure firewall
  firewalld:
    service: "{{ firewalld_service }}"
    permanent: yes
    state: enabled
    immediate: yes

- name: configure service
  service:
    name: "{{ daemon_service }}"
    state: started
    enabled: yes
```

common 역할에서는 필요한 패키지를 설치하고 방화벽을 설정하고 데몬 서비스를 시작하는 작업을 진행합니다.

패키지를 설치하는 작업은 packages 변수 리스트를 필요로 하는데, 단일 변수여도 리스트로 변환하여 처리할 수 있도록 합니다. (단일 변수여도 변수 이름은 packages로 지정해야 동작합니다.)

enable_sebool.yml

```
- name: enable SELinux boolean
  seboolean:
    name: "{{ sebool }}"
    persistent: yes
    state: yes
```

어느 호스트 그룹에서든 필요한 경우에 SELinux boolean을 활성화하는 작업을 사용할 수 있도록 common 역할에 배치합니다.

3) web 역할

web 역할에는 웹 서버를 구성하기 위해 진행하는 작업을 배치합니다.

common/tasks/main.yml

```
---
# tasks file for web
- name: download wordpress archive
  command: wget -O wordpress.tar.gz https://wordpress.org/latest.tar.gz

- name: untar wordpress archive
  command: tar -zxvf wordpress.tar.gz -C /var/www/html/

- name: configure wordpress
  template:
    src: wp-config.php.j2
    dest: /var/www/html/wordpress/wp-config.php

- name: create virtual host
  template:
    src: vhosts.conf.j2
    dest: /etc/httpd/conf.d/vhosts.conf
  notify: restart httpd

- name: enable SELinux boolean httpd_can_network_connect_db
  include_role:
    name: common
    tasks_from: enable_sebool
  vars:
    sebool: "{{ web_sebool }}"
```

마지막 작업은 include_role 모듈을 사용하여 common 역할의 enable_sebool 작업을 불러와 실행합니다.

4) db 역할

db 역할에는 DB 서버를 구성하기 위해 진행하는 작업을 배치합니다.

common/tasks/main.yml

```
---
# tasks file for db
- name: check initial setup
  command: mysql -u root -e "SELECT 1;"
  register: mysql_check_result
  ignore_errors: yes

- name: setup MySQL if NOT initial setup
  expect:
    command: mysql_secure_installation
    responses:
      'Would you like to setup VALIDATE PASSWORD component' : 'n'
      'New password:' : "{{ db_root_password }}"
      'Re-enter new password:' : "{{ db_root_password }}"
      'Remove anonymous users?' : 'y'
      'Disallow root login remotely?' : 'y'
      'Remove test database and access to it?' : 'y'
      'Reload privilege tables now?' : 'y'
  when: mysql_check_result.rc == 0

- name: create database
  mysql_db:
    login_user: root
    login_password: "{{ db_root_password }}"
    state: present
    name: "{{ db_name }}"

- name: create db user and grant privileges
  mysql_user:
    login_user: root
    login_password: "{{ db_root_password }}"
    state: present
    name: "{{ db_user }}"
    password: "{{ db_password }}"
    host: "{{ db_host }}"
    priv: "cccr.*:SELECT,INSERT,UPDATE,DELETE,CREATE,ALTER,INDEX,CREATE
TEMPORARY TABLES,LOCK TABLES"
```

5) site.yml 파일 작성

```
# project/site/site.yml
- name: configure Web server
  hosts: webservers
  roles:
    - common
    - web

- name: configure DB server
  hosts: dbservers
  roles:
    - common
    - db
```

마지막으로, 각 호스트 그룹에 맞는 역할을 실행하는 플레이북을 작성합니다.

이 플레이북을 실행하면 webservers 호스트 그룹의 호스트에는 웹 서버가, dbservers 호스트 그룹의 호스트에는 DB 서버가 자동으로 구축됩니다.

2.2 트러블슈팅

구현 0단계의 SSH 키 교환 과정에서 Permission denied가 발생하는 문제가 있었습니다.

문제 설명

```
[vagrant@cccr-ansible project]$ ssh-copy-id vagrant@192.168.58.13
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/vagrant/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
vagrant@192.168.58.13: Permission denied
(publickey,gssapi-keyex,gssapi-with-mic).
```

제어 노드 (cccr-ansible) IP 주소: 192.168.58.12

관리 노드 (cccr-db) IP 주소: 192.168.58.13

제어 노드에서 관리 노드에 공개 키를 복사하는 ssh-copy-id 명령어를 사용하면 관리 노드에 접속할 수 있는 권한이 없다는 에러가 발생합니다.

해결 과정

1. **관리 노드와 제어 노드의 키 확인**

   관리 노드 - ~/.ssh/authorized_keys

   ```
   ssh-ed25519 {생략}b0JA vagrant
   ```

   제어 노드 - ~/.ssh/id_rsa.pub 확인

   ```
   ssh-rsa {생략}Ixs= vagrant@cccr-ansible
   ```

   제어 노드의 id_rsa.pub 공개 키가 관리 노드에 저장되어 있어야 하는데, 저장되어 있지 않다는 사실을 확인했습니다.

2. 기존 키 삭제 후 재생성

```
[vagrant@cccr-ansible project]$ rm ~/.ssh/id_rsa.pub ~/.ssh/id_rsa_db
[vagrant@cccr-ansible project]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vagrant/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vagrant/.ssh/id_rsa
Your public key has been saved in /home/vagrant/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:xqcl7DqN5Pa1X8FtyUbnITvoKDVAdlDEVWjXeX0zTjw vagrant@cccr-ansible
The key's randomart image is:
+---[RSA 3072]----+
|       +=+..o.o o|
|      o .. o . E+|
|       .  . ..ooB|
|       o.   ..=++|
|        Soo. oo++|
|      .o.=+   oo |
|     o +o...  .  |
|      =.o. . .   |
|     ..o. ...    |
+----[SHA256]-----+
```

키 페어에 문제가 있나 싶어 키 페어를 재생성해보았습니다.

하지만, 이후 키 교환을 시도해도 여전히 같은 오류가 발생했습니다.

3. 비밀번호 인증 방식 SSH 접속 시도

```
[vagrant@cccr-mysql ~]$ sudo vim /etc/ssh/sshd_config
# PasswordAuthentication yes로 전부 설정
[vagrant@cccr-mysql ~]$ sudo systemctl restart sshd
```

```
[vagrant@cccr-ansible project]$ ssh vagrant@192.168.58.13
vagrant@192.168.58.13's password:
Last login: Wed May  7 03:13:16 2025 from 10.0.2.2
[vagrant@cccr-mysql ~]$
```

SSH 연결 자체가 불가능한지 확인하기 위해 비밀번호 인증 방식을 사용하여 SSH 접속을 시도합니다.

SSH 연결에 성공했고, 키 기반 인증 방식에서 문제가 발생하고 있다는 것을 알 수 있었습니다.

4. 비밀번호 인증 방식 no로 변경 후 키 기반 인증 방식 SSH 접속 시도

```
[vagrant@cccr-mysql ~]$ sudo vim /etc/ssh/sshd_config
# PasswordAuthentication no로 전부 설정
[vagrant@cccr-mysql ~]$ sudo systemctl restart sshd
```

```
[vagrant@cccr-ansible project]$ ssh-copy-id vagrant@192.168.58.13
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/vagrant/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
are prompted now it is to install the new keys
vagrant@192.168.58.13: Permission denied
(publickey,gssapi-keyex,gssapi-with-mic).
```

SSH 설정 파일을 원래대로 돌려놓고 다시 키 기반 인증 방식으로 SSH 접속을 시도했습니다.

같은 오류가 발생했습니다.

5. ssh-copy-id 명령어 검색

ssh-copy-id 명령어를 사용한 키 교환이 어떻게 이루어지는지 확인하기 위해 구글링을 진행합니다.

한 게시글에서 키 기반 인증 방식으로 SSH 접속을 처음 시도할 때 비밀번호 입력을 해야 한다는 것을 발견했습니다.

인증 방식을 yes로 설정하고 키 교환을 시도해야 한다는 결론에 도달했습니다.

6. 비밀번호 인증 방식 yes로 변경 후 키 기반 인증 방식 SSH 접속 시도

```
[vagrant@cccr-ansible project]$ ssh-copy-id vagrant@192.168.58.13
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/vagrant/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
are prompted now it is to install the new keys
vagrant@192.168.58.13's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'vagrant@192.168.58.13'"
and check to make sure that only the key(s) you wanted were added.

[vagrant@cccr-ansible project]$ ssh vagrant@192.168.58.13
Last login: Wed May  7 05:25:35 2025 from 192.168.58.12
[vagrant@cccr-mysql ~]$
```

관리 노드의 SSH 설정 파일에서 비밀번호 인증 방식 허용 옵션(PasswordAuthentication)을 yes로 설정한 뒤, 키 교환을 진행하니 SSH 접속이 성공했습니다.

관리 노드 - ~/.ssh/authorized_keys

```
ssh-rsa {생략}BSM= vagrant@cccr-ansible
```

제어 노드 - ~/.ssh/id_rsa.pub

```
ssh-rsa {생략}BSM= vagrant@cccr-ansible
```

관리 노드와 제어 노드에 키 교환이 완료된 것을 확인할 수 있었습니다.

이 과정으로 키 기반 인증 방식 SSH 접속이 불가능했던 문제를 해결했습니다.

해결법 정리

A에서 B로 키 기반 인증 방식으로 SSH 접속하기 위해 키 교환을 진행할 때는 비밀번호 인증 방식이 허용된 상태여야 합니다.

3. 프로젝트 결론

본 프로젝트에서는 웹 서버와 DB 서버의 초기 구성을 자동화하는 Ansible 환경을 성공적으로 구축하였습니다. 이를 통해 반복적인 설정 작업을 최소화하고, 일관된 서버 환경을 확보할 수 있었습니다.

3.1 향후 발전 방향

현재는 구현되지 않았지만 향후 추가하거나 개선할 수 있는 부분과 그 방법에 대한 내용은 다음과 같습니다.

1. MySQL 초기 설정 작업 분리

   현재는 expect 모듈을 사용하여 MySQL 초기 설정(총 7개의 작업)을 한 번에 처리하고 있습니다. 각 설정을 별도의 작업으로 분리하여 db 역할의 tasks 디렉토리에 구성하면 유연성과 가독성이 향상됩니다. 향후 특정 설정만 수정하거나 재사용할 경우에 효율적입니다.

2. MySQL 로그인 정보 중복 제거

   현재는 MySQL 관련 작업에서 login_user와 login_password를 반복해서 지정하고 있습니다. db 역할의 defaults/main.yml 파일에 mysql_login_user, mysql_login_password 변수를 정의하면 작업에서 지정해주지 않아도 해당 변수를 사용하여 MySQL에 접속할 수 있습니다.

3. 민감 정보 보안 강화

   현재는 MySQL 사용자 비밀번호 등 민감 정보를 평문 변수로 관리하고 있습니다. Ansible Vault를 활용하여 해당 정보를 암호화하면 보안을 강화할 수 있습니다.

4. 역할 실행 제어를 위한 태그 활용

   프로젝트가 커지고 역할이 복잡해질 경우, tags를 활용하여 필요한 작업만 선택적으로 실행할 수 있도록 구성할 수 있습니다. 이를 통해 전체 플레이북을 실행하지 않고 특정 기능만 빠르게 적용하는 것이 가능합니다.

3.2 프로젝트 회고

이번 프로젝트에서는 서버 구성 자동화를 구현함에 있어 확장성과 유지보수성을 염두에 두고 진행하였습니다. 중복되는 코드를 최대한 줄이고, 재사용 가능한 구조를 설계하는 데 중점을 두었습니다.

작업을 처음부터 역할로 세분화하여 구성하기보다는, 전체적인 계획을 수립한 뒤 단계적으로 개선하며 구현하는 방식을 택하였습니다. 계획 수립에 많은 시간을 들였지만, 그 덕분에 실제 구현 단계에서는 불필요한 시행착오를 줄이고 효율적으로 개발을 진행할 수 있었습니다.

또한, 구현 과정에서의 흐름과 결정 사항을 정리하고 문서화하려고 노력했으며, 이러한 기록이 보고서 작성에 도움이 되기를 기대했습니다. 다만 보고서를 프로젝트 마지막에 몰아서 작성하다 보니, 모든 내용을 충분히 담아내지 못한 점은 아쉬움으로 남습니다.

이번 프로젝는 자동화된 인프라 구성의 기초를 다지는 경험이 되었으며, 향후 더 복잡한 환경에서도 안정적이고 효율적인 구성이 가능하도록 기반을 마련하는 계기가 되었습니다.