Terraform 및 Ansible을 이용한 AWS EC2에 Wordpress 배포하기

목표: Terraform으로 AWS EC2 등 리소스를 프로비저닝 하고, AWS EC2 인스턴스에 Wordpress CMS 및 MySQL 데이터베이스를 배포하는 Ansible 역할을 이용하여 애플리케이션을 자동화 배포 및 구성 관리한다.

1. Wordpress 및 MySQL 설치 및 구성 명령

시나리오: Wordpress와 MySQL 서버를 같은 EC2 인스턴스에 배포한다.

1) Wordpress 설치 및 구성

(1) Wordpress 설치

패키지 인덱스를 업데이트 한다.

```
sudo apt update
```

wordpress 패키지 및 관련 패키지를 설치한다.

```
sudo apt install wordpress php libapache2-mod-php php-mysql
```

(2) Wordpress를 위한 Apache 구성

wordpress.conf 구성 파일을 작성한다.

> /etc/apache2/sites-available/wordpress.conf

```
Alias /wp /usr/share/wordpress
<Directory /usr/share/wordpress>
    Options FollowSymLinks
    AllowOverride Limit Options FileInfo
    DirectoryIndex index.php
    Order allow,deny
    Allow from all
</Directory>
<Directory /usr/share/wordpress/wp-content>
    Options FollowSymLinks
    Order allow,deny
    Allow from all
</Directory>
```

wordpress 사이트를 활성화 한다.

```
sudo a2ensite wordpress
```

rewrite 모듈을 활성화 한다.

```
sudo a2enmod rewrite
```

apache2 서비스를 리로드 한다.

```
sudo systemctl reload apache2
```

2) MySQL 데이터베이스 구성

(1) MySQL 설치

패키지 인덱스를 업데이트 한다.

```
sudo apt update
```

mysql-server 및 mysql-client 패키지를 설치한다.

```
sudo apt install mysql-server mysql-client
```

(2) 데이터베이스 구성

데이터베이스에 접속한다.

```
sudo mysql -u root
```

wordpress 데이터베이스를 생성한다.

```
mysql> CREATE DATABASE wordpress;
```

wordpress 사용자를 생성하고 wordpress 데이터베이스에 모든 호스트에서 접근할 수 있는 권한을 할당한다.

```
mysql> CREATE USER 'wordpress'@'%' IDENTIFIED BY
'P@ssw0rd';
```

wordpress 사용자가 wordpress 데이터베이스에 권한을 할당한다.

```
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO
'wordpress'@'%';
```

재구성한 권한을 다시 읽는다.

```
mysql> FLUSH PRIVILEGES;
```

데이터베이스를 빠져나간다.

```
mysql> quit
```

(3) Wordpress 데이터베이스 구성

데이터베이스 관련 접속정보를 설정하는 구성파일을 생성한다.

> /etc/wordpress/config-default.php

```
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'P@ssw0rd');
define('DB_HOST', 'localhost');
define('DB_COLLATE', 'utf8_general_ci');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-
content');
?>
```

3. 접속

wordpress에 접속해서 동작을 확인한다.

```
http://192.168.200.101/wp
```

2. Wordpress 및 MySQL 설치 및 구성 플레이북 작성

Wordpress 및 MySQL 데이터베이스를 배포하기 위한 Ansible Playbook을 역할 기반으로 작성한다.

- 요구사항
  - 역할
    - wordpress
    - mysql
  - wordpress 역할
    - 작업: 패키지 설치, 구성파일 복사, 모듈 및 사이트 활성화, 서비스 시작
    - 핸들러: 모듈 및 사이트 활성화, 서비스 재시작
    - 변수: 데이터베이스명, 데이터베이스 사용자, 데이터베이스 패스워드(암호화) 등
    - 파일: 구성파일
  - mysql 역할
    - 작업: 패키지 설치, 데이터베이스 생성, 사용자 생성, 사용자 권한 부여, 구성파일 복사(위임 또는 wordpress에서 작업 가능), 서비스 시작
    - 핸들러: 서비스 재시작
    - 변수: 데이터베이스명, 데이터베이스 사용자, 데이터베이스 패스워드(암호화) 등
    - 파일: 구성파일
  - Main 플레이북 작성

1) wordpress 역할 생성

(1) wordpress 역할 빼대 생성

(2) 변수 관련 파일 작성

(3) 구성 파일 작성

(4) 작업 작성

(5) 핸들러 작성

2) mysql 역할 생성

(1) mysql 역할 빼대 생성

(2) 변수 관련 파일 작성

(3) 구성 파일 작성

(4) 작업 작성

(5) 핸들러 작성

3) main 플레이북 작성

wordpress 및 mysql 역할을 사용하는 플레이북을 작성한다.

4) 플레이북 검증

로컬 VM에서 wordpress 및 mysql이 적절하게 배포되고 작동하는지 검증한다.

3. AWS 리소스 배포 Terraform HCL 작성

1) AWS EC2 리소스 배포용 HCL 작성

(1) provider 작성

AWS 프로바이더, 프로바이더 요구사항, Terraform 연격 백엔드를 정의한다.

> provider.tf

```

```

(2) 변수 정의

필요한 변수를 정의한다.

> vars.tf

```

```

(3) 변수 값 설정

정의한 변수의 값을 설정한다.

> terraform.tfvars

```

```

(4) 로컬 값 설정

리소스의 태그에 설정한 값을 정의하는 로컬 값을 설정한다.

> local.tf

(5) 데이터 소스 설정

AWS AMI 이미지 ID를 가져올 수 있는 데이터 소스를 정의한다.

> data_source.tf

(6) main 리소스 작성

EC2, EIP, VPC, SSH 키 쌍 리소스 등 구성한다.

EC2 인스턴스 정의시 Ansible 플레이북을 실행할 수 있는 프로비저너를 정의한다.

> main.tf

(7) 보안그룹 리소스 작성

외부에서 접근하는 SSH 및 HTTP 포트를 연다.

> security-group.tf

(8) 출력 값 설정

wordpress에 접근할 수 있는 EC2 인스턴스의 외부 IP 및 도메인 주소를 출력을 설정한다.

> output.tf

2) Terraform HCL 계획 및 적용

(1) 배포 계획

wordpress 배포를 계획하여 문제가 없는지 검증한다.

(2) 배포 적용

AWS 리소스가 배포되고, Ansible 플레이북으로 Wordpress가 배포된다.

3) 검증

(1) AWS 리소스 배포 확인 및 검증

> 스크린샷

(2) Wordpress 접속 확인 및 검증

> 스크린샷

(옵션) 4. 추가 도전 과제

- 시나리오
  - 케이스1
    - Wordpress와 MySQL 서버를 각각의 다른 EC2 인스턴스에 배포한다.
  - 케이스2
    - Wordpress는 EC2 인스턴스에 배포하고, MySQL 서버는 AWS RDS를 사용한다.
  - 케이스3
    - Ansible-pull을 이용해 Ansible 제어 노드 없이 EC2 인스턴스에서 직접 Ansible 플레이북을 실행한다.