**Terraform AWS & GCP**

[개념 3](#_Toc164316175)

[VPC 3](#_Toc164316176)

[Subnet 3](#_Toc164316177)

[Routing Table 3](#_Toc164316178)

[Internet Gateway 3](#_Toc164316179)

[NAT Gateway 3](#_Toc164316180)

[VPC Endpoint 3](#_Toc164316181)

[공통 환경 구성 4](#_Toc164316182)

[실습 1 7](#_Toc164316183)

[실습 2 12](#_Toc164316184)

[VPC 구성 12](#_Toc164316185)

[Subnet 구성 12](#_Toc164316186)

[Internet Gateway 구성 13](#_Toc164316187)

[Route Table 구성 13](#_Toc164316188)

[Security Group 구성 14](#_Toc164316189)

[EC2 인스턴스 구성 14](#_Toc164316190)

[Terraform apply로 인프라 생성 15](#_Toc164316191)

[Console 창 확인 16](#_Toc164316192)

[실습 3 21](#_Toc164316193)

[Compute engine api 21](#_Toc164316194)

[Service account 22](#_Toc164316195)

[IAM API 24](#_Toc164316196)

[Provider 25](#_Toc164316197)

[VPC 25](#_Toc164316198)

[방화벽 26](#_Toc164316199)

[인스턴스 26](#_Toc164316200)

[Terraform apply 27](#_Toc164316201)

[Console 창 확인 27](#_Toc164316202)

[참조 28](#_Toc164316203)

# 개념

## VPC

* 사용자가 AWS 클라우드에서 논리적으로 격리된 가상 네트워크 공간을 만들 수 있게함

## Subnet

* VPC 내의 IP 주소를 논리적으로 나누어 사용
* 서브넷은 특정 가용 영역에 위치하며, 리소스 그룹을 격리하는 데 사용

## Routing Table

* 네트워크 트래픽이 VPC 내외부로 어떻게 이동할지 결정하는 규칙의 집합

## Internet Gateway

* VPC와 인터넷 간의 트래픽을 허용하는 VPC 구성 요소
* 공개 서브넷 내의 리소스에 대해 필수적임

## NAT Gateway

* 프라이빗 서브넷의 인스턴스가 인터넷에 액세스할 수 있도록 해줌
* 하지만, 인터넷에서 해당 인스턴스로의 직접 접근은 허용하지 않음. 이를 통해 보안성을 높임

## VPC Endpoint

* VPC와 AWS 서비스(S3, DynamoDB)간의 프라이빗 연결을 가능하게 해줌
* 인터넷을 경유하지 않고 AWS 서비스에 안전하게 접근하게 함

# 공통 환경 구성

원활한 실습을 위해 window가 아닌 ubuntu 환경에서 실습

제공받은 Vagrantfile을 통해 구성

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<https://developer.hashicorp.com/terraform/install?product_intent=terraform#linux>

* 해당 사이트에서 linux terraform package 설치 가이드를 참조

Terraform 설치가 완료되면 aws console에서 iam 사용자를 생성하여 콘솔 접근을 가능하게 해야함

다음과 같은 권한을 가진 iam 사용자를 생성

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이후 aws cli를 설치

sudo apt update

sudo apt install unzip -y

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

설치가 완료되면 aws configure를 설정하여 연결해줌

* 지급 받은 accesskey, secret accesskey와 region을 입력하여 설정

명령 완성 활성화

complete -C '/usr/local/bin/aws\_completer' aws

모니터링 패키지 설치

sudo apt-get install tree jq watch

여기까지 완료하면 terraform을 사용할 준비는 끝남. 본격적으로 실습 시작

# 실습 1

웹서비스 포트를 입력변수를 50000번을 통해 배포 후 접속 결과 확인

다음 코드를 통해 main.tf 파일 구성

cat <<EOT > main.tf

// AWS 공급자 정의

provider "aws" {

region = "ap-northeast-2"

}

// AWS 인스턴스 리소스 생성

resource "aws\_instance" "example" {

ami = "ami-09a7535106fbd42d5"

instance\_type = "t2.micro"

vpc\_security\_group\_ids = [aws\_security\_group.instance.id]

user\_data = <<-EOF

#!/bin/bash

echo "Hello, MZC-CLOUD 50000" > index.html

nohup busybox httpd -f -p 50000 &

EOF

user\_data\_replace\_on\_change = true

tags = {

Name = "homework-1"

}

}

resource "aws\_security\_group" "instance" {

name = var.security\_group\_name

ingress {

from\_port = 50000

to\_port = 50000

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

}

variable "security\_group\_name" {

description = "The name of the security group"

type = string

default = "terraform-example-instance"

}

output "public\_ip" {

value = aws\_instance.example.public\_ip

description = "The public IP of the Instance"

}

EOT

Tf 구성 완료 후 terraform init을 통해 현재 디렉토리에 terraform 구성 파일을 읽어들임

* 해당 구성 파일에서 사용된 플러그인과 백엔드에 대한 종속성을 확인하고 다운
* Terraforn이 사용할 수 있는 모든 플러그인을 가져와서 초기화
* 백엔드를 구성하거나 연결하여 상태를 저장할 위치를 설정
* Git init을 생각하면 됨

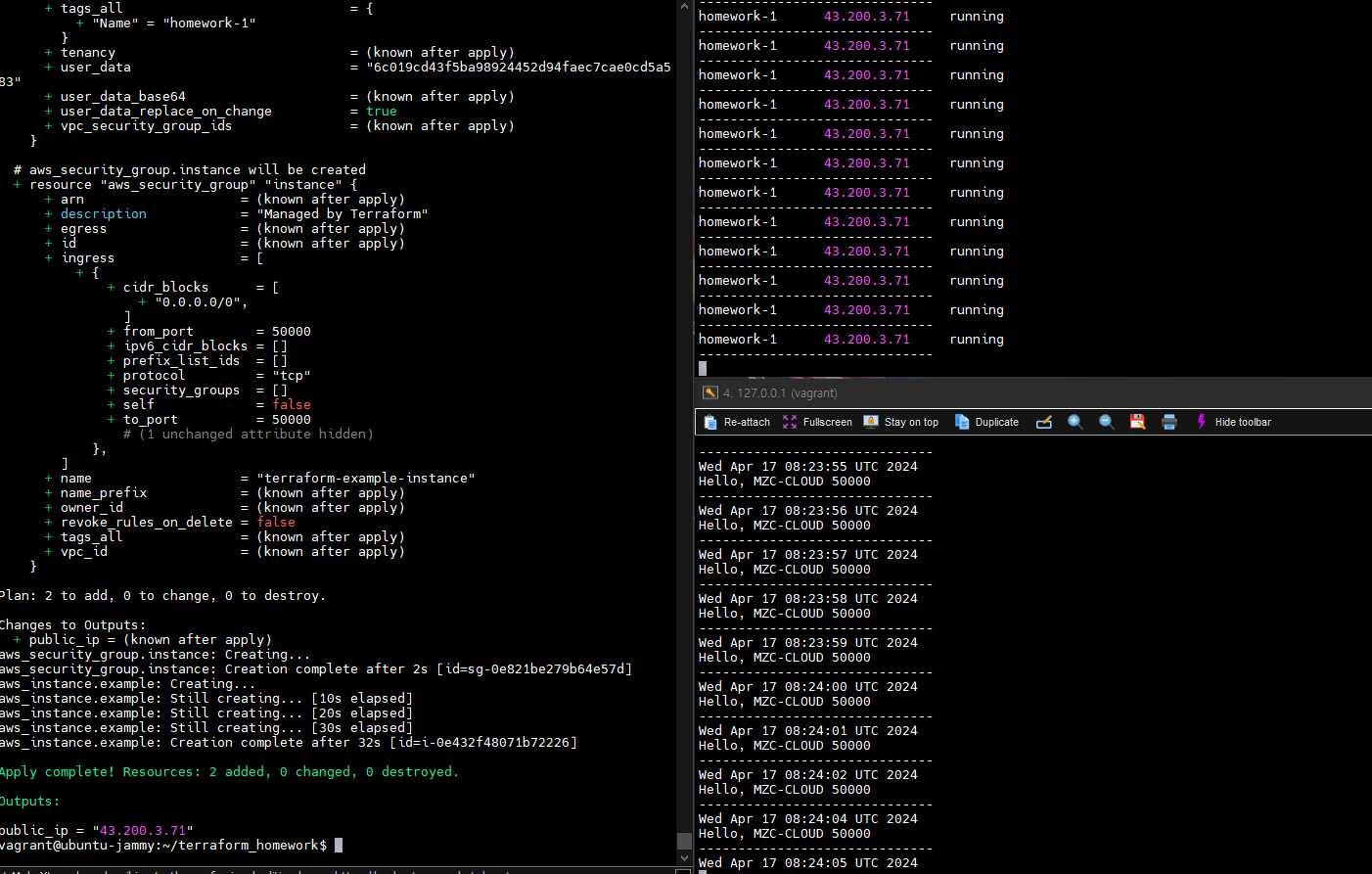
Terraform plan을 통해 terraform 구성 파일에 정의된 리소스들을 계획하고 변경 사항을 미리 볼 수 있음

다음과 같이 확인 가능

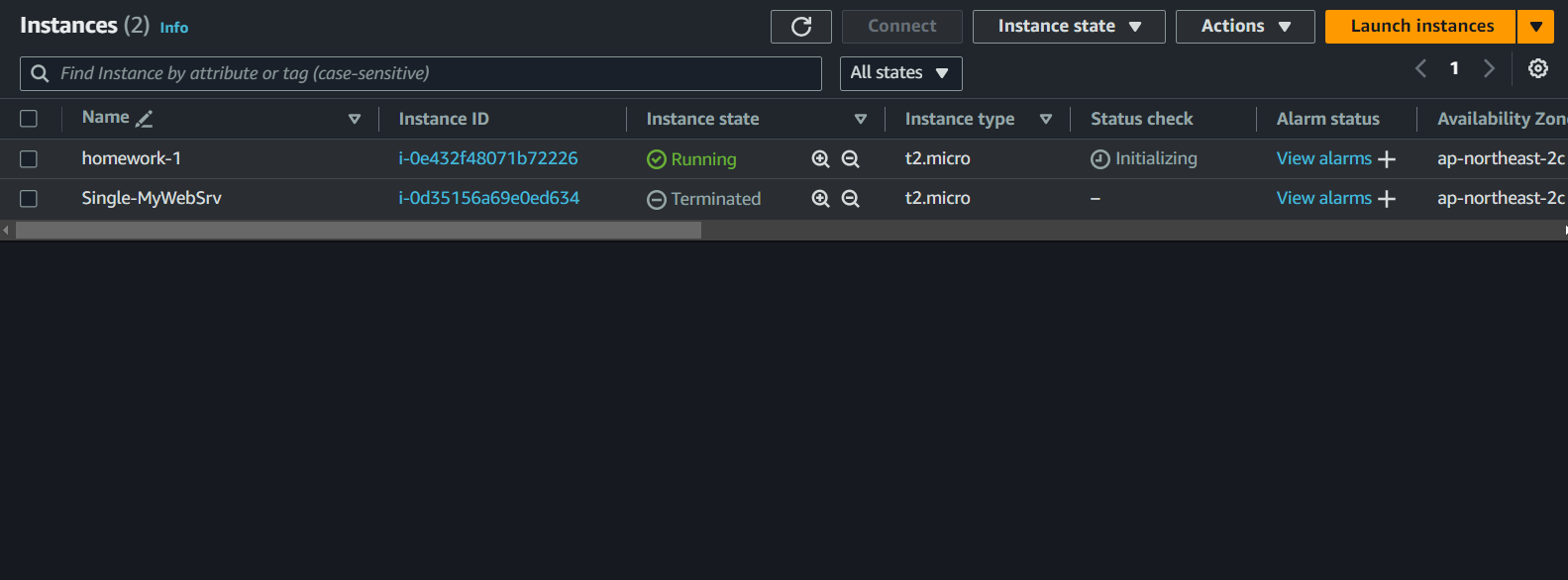
텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

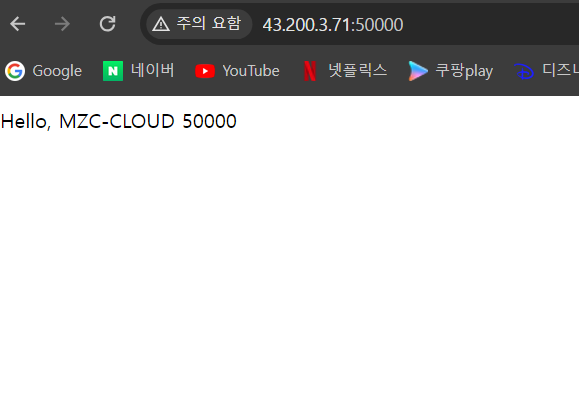
Terraform apply -auto-approve으로 구성 후 결과 확인



console에서 다음과 같이 인스턴스가 생성된 것을 확인



할당된 ip 주소에서 내용 확인



# 실습 2

AWS VPC(Subnet, IGW 등)을 코드로 배포한 환경에서 EC2 웹 서버 배포

## VPC 구성

텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

## Subnet 구성

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

## Internet Gateway 구성

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

## Route Table 구성

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

## Security Group 구성

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

## EC2 인스턴스 구성

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

## Terraform apply로 인프라 생성

텍스트, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

Cloud console에서 확인해보겠음

리소스 맵을 통해 다음과 같이 구성된 것을 확인

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

개별적으로 확인해보겠음

## Console 창 확인

VPC

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Subnet

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Internet Gateway

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Route table

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Security Group

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

EC2 인스턴스

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

포유류, 레서판다, 너구리 판다, 판다이(가) 표시된 사진

자동 생성된 설명

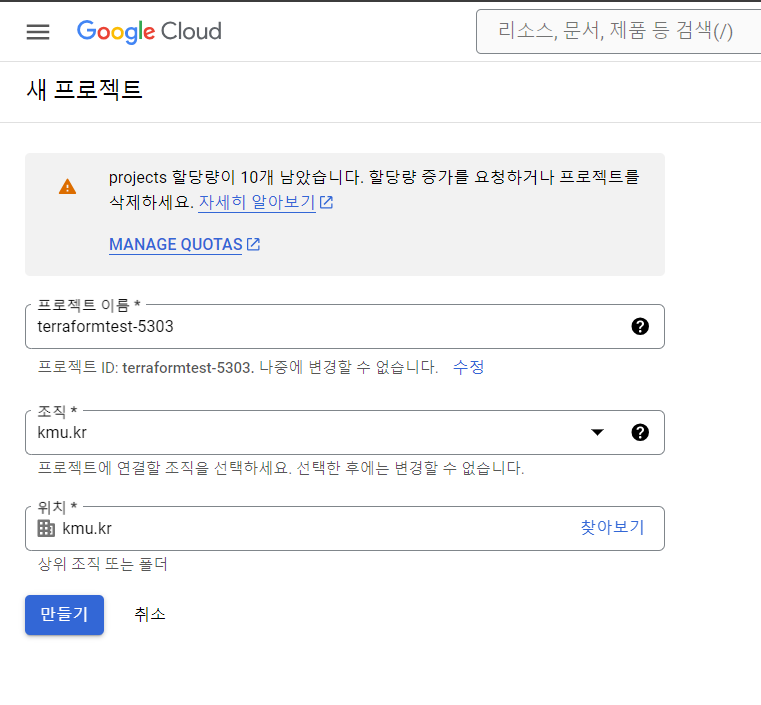
포유류, 레서판다, 너구리 판다, 육상동물이(가) 표시된 사진

자동 생성된 설명

# 실습 3

Azure나 GCP 환경에서 기본 인스턴스를 배포

우선 GCP Project를 생성

* GCP는 Project가 필요. Project는 gcp 리소스들의 집합

다음으로 compute engine api를 활성화시킴

## Compute engine api

* 구글에서 제공하는 가상머신을 만들고 실행할 수 있는 컴퓨터 및 호스팅 서비스

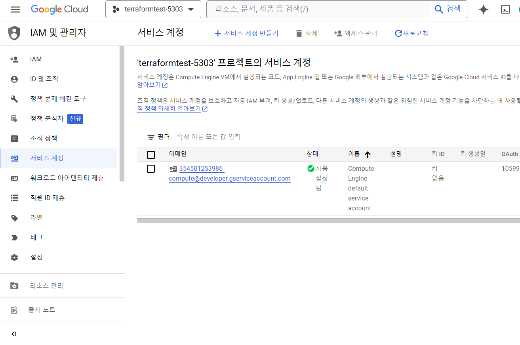
텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

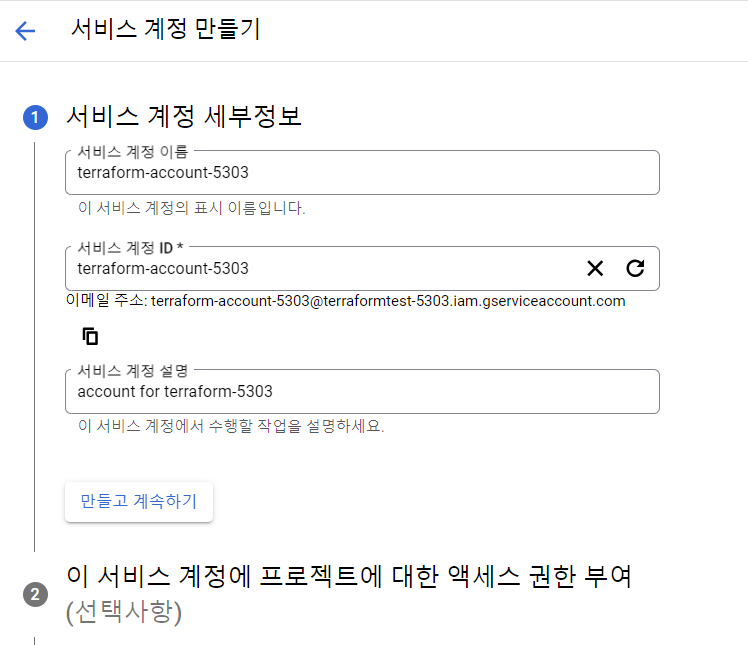
자동 생성된 설명

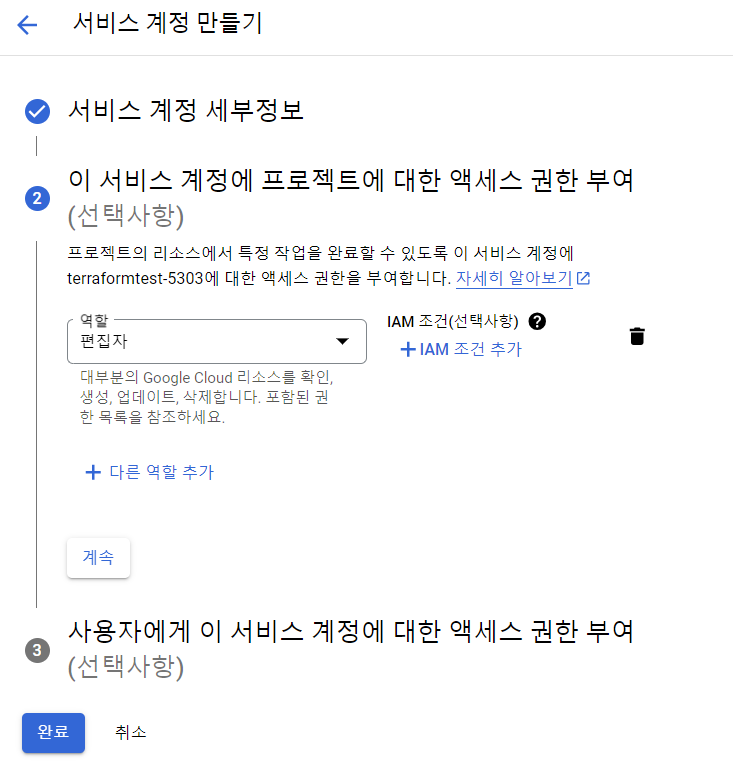
## Service account

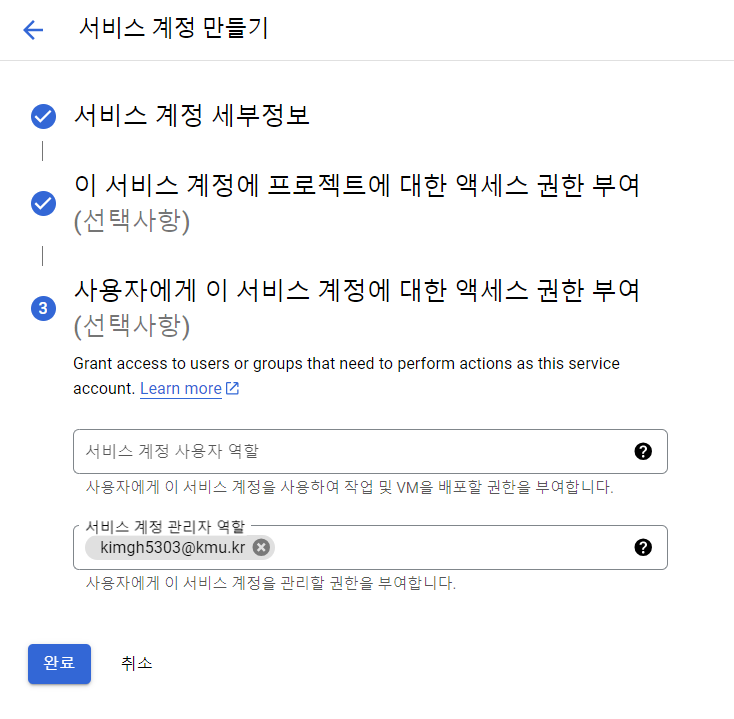
Service account를 만들어줆. 다음과 같이 설정

* Google api에 인증하는데 사용





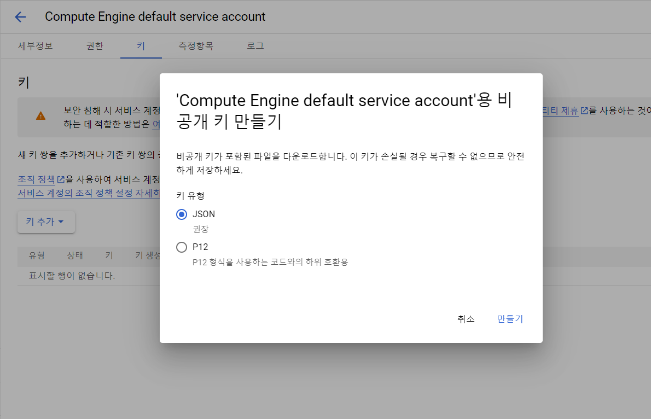




이후 json 형태의 key를 생성해줌. 다음과 같이 생성

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명



텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

해당 json 파일은 다운로드 하여 ubuntu terraform 경로에 둘 것

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

## IAM API

* 활성화시켜줌

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

작성한 코드를 바탕으로 main.tf를 구성함

## Provider

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

## VPC

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

## 방화벽

텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진

자동 생성된 설명

## 인스턴스

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

## Terraform apply

적용 후 결과를 확인해보겠음

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

다음과 같이 적용 되었고

## Console 창 확인

텍스트, 라인, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

GCP console에도 생성된 것을 확인

링크로 들어가보면

포유류, 레서판다, 너구리 판다, 판다이(가) 표시된 사진

자동 생성된 설명

다음과 같이 사진이 나오는 것을 확인!

# 참조

<https://yooloo.tistory.com/181>

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs>

<https://registry.terraform.io/providers/hashicorp/google/latest/docs>