

Terraform AWS & GCP

개념	3
VPC	3
Subnet	3
Routing Table	3
Internet Gateway	3
NAT Gateway	3
VPC Endpoint	3
공통 환경 구성	4
실습 1	7
실습 2	12
VPC 구성	12
Subnet 구성	12
Internet Gateway 구성	13
Route Table 구성	13
Security Group 구성	14
EC2 인스턴스 구성	14
Terraform apply 로 인프라 생성	15
Console 창 확인	16

실습 3.....	21
Compute engine api.....	21
Service account.....	22
IAM API	24
Provider	25
VPC.....	25
방화벽.....	26
인스턴스.....	26
Terraform apply	27
Console 창 확인	27
참조	28

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

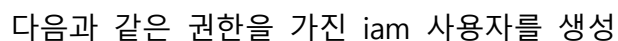
NAT Gateway

- ➔ 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

제공받은 Vagrantfile 을 통해 구성




[IAM](#) > [사용자](#) > [mzc-5303](#)

mzc-5303

정보

요약


ARN

 `arn:aws:iam::471112533833:user/mzc-5303`


생성됨

April 16, 2024, 16:45 (UTC+09:00)

콘솔 액세스

 **MFA 없이 활성화됨**

마지막 콘솔 로그인

 안 함

권한

그룹

태그 (1)

보안 자격 증명

액세스 관리자





권한 정책 (2)

사용자에게 직접 연결된 정책을 통해 또는 그룹을 통해 권한을 정의합니다.

🔍 검색

필터링 기준 유형

모든 유형

<input type="checkbox"/>	정책 이름 	▲	유형	▼	연결 방식: 
<input type="checkbox"/>	 AdministratorAccess		AWS 관리형 - 직무		직접
<input type="checkbox"/>	 IAMUserChangePassword		AWS 관리형		직접

이후 aws cli 를 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 configure 를 설정하여 연결해줌

➔ 지금 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 terraform 을 사용할 준비는 끝남. 본격적으로 실습 시작

실습 1

웹서비스 포트를 입력변수를 50000 번을 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 init 을 통해 현재 디렉토리에 terraform 구성 파일을 읽어들이

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

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

다음과 같이 확인 가능


```

vagrant@ubuntu-jammy:~/terraform_homework$ terraform plan

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.example will be created
+ resource "aws_instance" "example" {
  + ami                    = "ami-09a7535106fbd42d5"
  + arn                   = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone      = (known after apply)
  + cpu_core_count         = (known after apply)
  + cpu_threads_per_core   = (known after apply)
  + disable_api_stop       = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized          = (known after apply)
  + get_password_data      = false
  + host_id                = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile   = (known after apply)
  + id                     = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle     = (known after apply)
  + instance_state         = (known after apply)
  + instance_type          = "t2.micro"
  + ipv6_address_count     = (known after apply)
  + ipv6_addresses         = (known after apply)
  + key_name               = (known after apply)
  + monitoring             = (known after apply)
  + outpost_arn            = (known after apply)
  + password_data          = (known after apply)
  + placement_group        = (known after apply)
  + placement_partition_number = (known after apply)
  + primary_network_interface_id = (known after apply)
  + private_dns            = (known after apply)
  + private_ip             = (known after apply)
  + public_dns             = (known after apply)
  + public_ip              = (known after apply)
  + secondary_private_ips  = (known after apply)
  + security_groups        = (known after apply)
  + source_dest_check      = true
  + spot_instance_request_id = (known after apply)
  + subnet_id              = (known after apply)
  + tags                   = {
    + "Name" = "homework-1"
  }
  + tags_all              = {
    + "Name" = "homework-1"
  }
  + tenancy                = (known after apply)
  + user_data              = "6c019cd43f5ba98924452d94faec7cae0cd5a583"
  + user_data_base64      = (known after apply)
  + user_data_replace_on_change = true
}

```

Terraform apply -auto-approve 으로 구성 후 결과 확인

```
+ tags_all = {
+   "Name" = "homework-1"
+ }
+ tenancy = (known after apply)
+ user_data = "6c019cd43f5ba98924452d94faec7cae0cd5a5"
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = true
+ vpc_security_group_ids = (known after apply)
}

# aws_security_group.instance will be created
resource "aws_security_group" "instance" {
+ arn = (known after apply)
+ description = "Managed by Terraform"
+ egress = (known after apply)
+ id = (known after apply)
+ ingress = [
+   {
+     cidr_blocks = [
+       "0.0.0.0/0",
+     ]
+     from_port = 50000
+     ipv6_cidr_blocks = []
+     prefix_list_ids = []
+     protocol = "tcp"
+     security_groups = []
+     self = false
+     to_port = 50000
+   }
+ ],
+ name = "terraform-example-instance"
+ name_prefix = (known after apply)
+ owner_id = (known after apply)
+ revoke_rules_on_delete = false
+ tags_all = (known after apply)
+ vpc_id = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ public_ip = (known after apply)
aws_security_group.instance: Creating...
aws_security_group.instance: Creation complete after 2s [id=sg-0e821be279b64e57d]
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Still creating... [30s elapsed]
aws_instance.example: Creation complete after 32s [id=i-0e432f48071b72226]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:
public_ip = "43.200.3.71"
vagrant@ubuntu-jammy:~/terraform_homework$
```

Instance	Public IP	Status
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running
homework-1	43.200.3.71	running

```
4. 127.0.0.1 (vagrant)

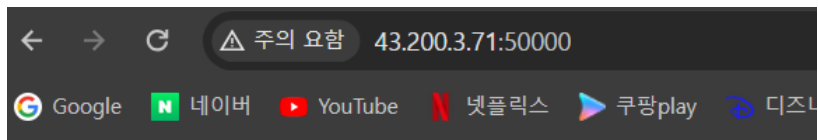
Re-attach Fullscreen Stay on top Duplicate Hide toolbar

Wed Apr 17 08:23:55 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:23:56 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:23:57 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:23:58 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:24:00 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:24:01 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:24:02 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:24:04 UTC 2024
Hello, MZC-CLOUD 50000
Wed Apr 17 08:24:05 UTC 2024
```

console 에서 다음과 같이 인스턴스가 생성된 것을 확인

Instances (2) Info							
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>				All states			
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	homework-1	i-0e432f48071b72226	Running	t2.micro	Initializing	View alarms +	ap-northeast-2c
<input type="checkbox"/>	Single-MyWebSrv	i-0d35156a69e0ed634	Terminated	t2.micro	-	View alarms +	ap-northeast-2c

할당된 ip 주소에서 내용 확인



Hello, MZC-CLOUD 50000

실습 2

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

VPC 구성

```
# VPC
resource "aws_vpc" "main" {
  cidr_block = "10.10.0.0/16"
}
```

Subnet 구성

```
# Public Subnet 1
resource "aws_subnet" "public-1" {
  vpc_id      = aws_vpc.main.id
  cidr_block  = "10.10.1.0/24"
  availability_zone = "ap-northeast-2a"
  map_public_ip_on_launch = true

  tags = {
    Name = "public-1"
  }
}

# Public Subnet 2
resource "aws_subnet" "public-2" {
  vpc_id      = aws_vpc.main.id
  cidr_block  = "10.10.2.0/24"
  availability_zone = "ap-northeast-2c"
  map_public_ip_on_launch = true

  tags = {
    Name = "public-2"
  }
}
```

Internet Gateway 구성

```
# Internet Gateway
resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "gw"
  }
}
```

Route Table 구성

```
# Route Table
resource "aws_route_table" "rt" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "rt"
  }
}

resource "aws_route_table_association" "public-subnet-1" {
  subnet_id      = aws_subnet.public-1.id
  route_table_id = aws_route_table.rt.id
}

resource "aws_route_table_association" "public-subnet-2" {
  subnet_id      = aws_subnet.public-2.id
  route_table_id = aws_route_table.rt.id
}

# Route Rule
resource "aws_route" "r" {
  route_table_id      = aws_route_table.rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id          = aws_internet_gateway.gw.id
}
```

Security Group 구성

```
resource "aws_security_group" "public_sg" {
  name = var.security_group_name
  vpc_id = aws_vpc.main.id

  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

variable "security_group_name" {
  description = "The name of the security group"
  type        = string
  default     = "terraform-my-instance"
}
```

EC2 인스턴스 구성

```
# EC2 - 2a
resource "aws_instance" "main1" {
  ami           = "ami-0e9b9db247cc8de84"
  instance_type = "t2.nano"
  subnet_id     = aws_subnet.public-1.id
  vpc_security_group_ids = [aws_security_group.public_sg.id]

  user_data = <<-EOF
  #!/bin/bash
  sudo apt update -y
  sudo apt install -y apache2
  sudo systemctl start apache2
  sudo systemctl enable apache2

  echo "<img src=\"https://i.namu.wiki/i/LVX-N10dzyEF_B76STh0R9SpkwVgcMU5h0n-hwLI-qQ1ly3I...\"
  sudo systemctl restart apache2
  EOF

  tags = {
    Name = "project1"
  }
}
```

```
# EC2 - 2c
resource "aws_instance" "main2" {
  ami           = "ami-0e9bfdb247cc8de84"
  instance_type = "t2.nano"
  subnet_id     = aws_subnet.public-2.id
  vpc_security_group_ids = [aws_security_group.public_sg.id]

  user_data = <<-EOF
  #!/bin/bash
  sudo apt update -y
  sudo apt install -y apache2
  sudo systemctl start apache2
  sudo systemctl enable apache2

  echo "<img src='\"https://i.namu.wiki/i/A809JxPjAWINHn-SH-UwIzpSy0Pc28ui2br21WQX\"'"
  sudo systemctl restart apache2
  EOF

  tags = {
    Name = "project2"
  }
}
```

Terraform apply로 인프라 생성

```
+ dhcp_options_id           = (known after apply)
+ enable_dns_hostnames      = (known after apply)
+ enable_dns_support        = true
+ enable_network_address_usage_metrics = (known after apply)
+ id                        = (known after apply)
+ instance_tenancy          = "default"
+ ipv6_association_id       = (known after apply)
+ ipv6_cidr_block           = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id       = (known after apply)
+ owner_id                  = (known after apply)
+ tags_all                  = (known after apply)
}
```

Plan: 11 to add, 0 to change, 0 to destroy.

Changes to Outputs:

```
+ public_ip_1 = (known after apply)
+ public_ip_2 = (known after apply)
```

```
aws_vpc.main: Creating...
aws_vpc.main: Creation complete after 1s [id=vpc-0ca0cda67d620095b]
aws_route_table.r: Creating...
aws_subnet.public-2: Creating...
aws_internet_gateway.gw: Creating...
aws_subnet.public-1: Creating...
aws_security_group.public_sg: Creating...
aws_internet_gateway.gw: Creation complete after 0s [id=igw-06beb78766413d989]
aws_route_table.r: Creation complete after 1s [id=rtb-0b9b8746d30336575]
aws_route.r: Creating...
aws_route.r: Creation complete after 0s [id=r-rtb-0b9b8746d303365751080289494]
aws_security_group.public_sg: Creation complete after 2s [id=sg-0f04836d9eb635bc8]
aws_subnet.public-2: Still creating... [10s elapsed]
aws_subnet.public-1: Still creating... [10s elapsed]
aws_subnet.public-1: Creation complete after 11s [id=subnet-02242c4363fbae074]
aws_subnet.public-2: Creation complete after 11s [id=subnet-03d3eda499384972a]
aws_route_table_association.public-subnet-1: Creating...
aws_route_table_association.public-subnet-2: Creating...
aws_instance.main2: Creating...
aws_instance.main1: Creating...
aws_route_table_association.public-subnet-2: Creation complete after 0s [id=rtbassoc-06a68fe509385dd50]
aws_route_table_association.public-subnet-1: Creation complete after 0s [id=rtbassoc-06927cd2c8ebe8abb]
aws_instance.main2: Still creating... [10s elapsed]
aws_instance.main1: Still creating... [10s elapsed]
aws_instance.main2: Still creating... [20s elapsed]
aws_instance.main1: Still creating... [20s elapsed]
aws_instance.main1: Still creating... [30s elapsed]
aws_instance.main2: Still creating... [30s elapsed]
aws_instance.main2: Creation complete after 32s [id=i-021c5cb886e7fb352]
aws_instance.main1: Creation complete after 32s [id=i-05e3f05ad65751130]
```

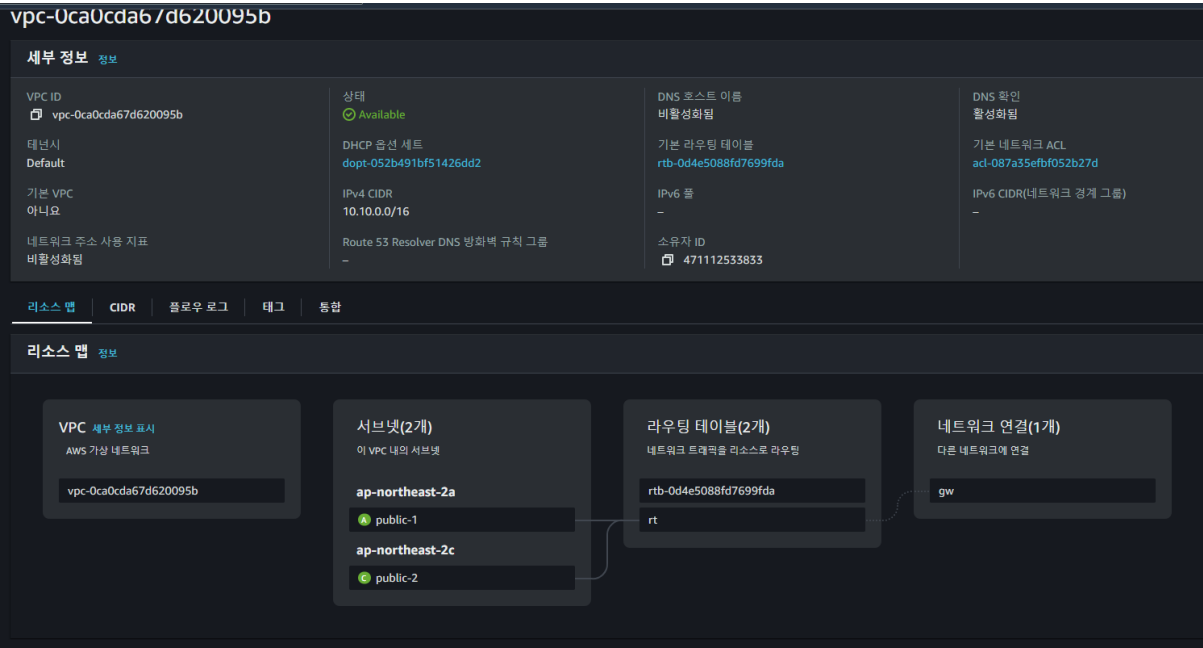
Apply complete! Resources: 11 added, 0 changed, 0 destroyed.

Outputs:

```
public_ip_1 = "52.78.204.34"
public_ip_2 = "13.125.207.135"
vagrant@ubuntu-jammy:~/terraform_homework$
```

Cloud console 에서 확인해보겠습니다

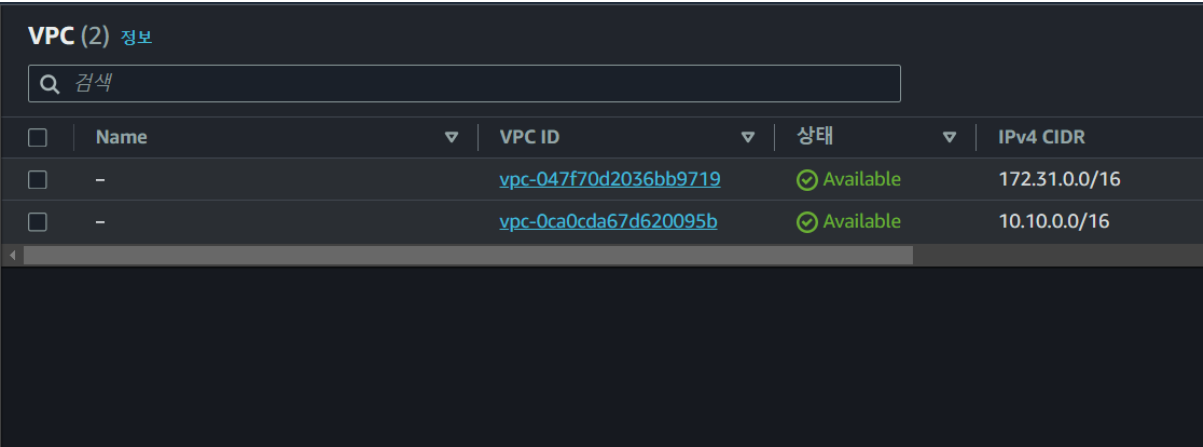
리소스 맵을 통해 다음과 같이 구성된 것을 확인



개별적으로 확인해보겠습니다

Console 창 확인

VPC



Subnet

서브넷 (6) 정보

Find resources by attribute or tag

<input type="checkbox"/>	Name	서브넷 ID	상태	VPC	IPv4 CIDR
<input type="checkbox"/>	-	subnet-0438602db307a9ff5	Available	vpc-047f70d2036bb9719	172.31.48.0/20
<input type="checkbox"/>	-	subnet-01c0cc2a82d76346e	Available	vpc-047f70d2036bb9719	172.31.32.0/20
<input type="checkbox"/>	-	subnet-0754787d8f8328b24	Available	vpc-047f70d2036bb9719	172.31.16.0/20
<input type="checkbox"/>	-	subnet-0ab0135c39dfd4ab6	Available	vpc-047f70d2036bb9719	172.31.0.0/20
<input type="checkbox"/>	public-1	subnet-02242c4363fbae074	Available	vpc-0ca0cda67d620095b	10.10.1.0/24
<input type="checkbox"/>	public-2	subnet-03d3eda499384972a	Available	vpc-0ca0cda67d620095b	10.10.2.0/24

Internet Gateway

인터넷 게이트웨이 (2) 정보

검색

<input type="checkbox"/>	Name	인터넷 게이트웨이 ID	상태	VPC ID
<input type="checkbox"/>	-	igw-01823fa57e6d8a38b	Attached	vpc-047f70d2036bb9719
<input type="checkbox"/>	gw	igw-06bbeb78766413d989	Attached	vpc-0ca0cda67d620095b

Route table

라우팅 테이블 (3) 정보

Find resources by attribute or tag

<input type="checkbox"/>	Name	라우팅 테이블 ID	명시적 서브넷 연결	엣지 연결
<input type="checkbox"/>	-	rtb-0d4e5088fd7699fda	-	-
<input type="checkbox"/>	-	rtb-0bcd09b07c96553e1	-	-
<input type="checkbox"/>	rt	rtb-0b9b8746d30336575	2 서브넷	-

VPC > 라우팅 테이블 > rtb-0b9b8746d30336575

rtb-0b9b8746d30336575 / rt

세부 정보 정보

라우팅 테이블 ID

rtb-0b9b8746d30336575

기본

아니요

명시적 서브넷 연결

2 서브넷

엣지 연결

-

VPC

vpc-0ca0cda67d620095b

소유자 ID

471112533833

라우팅

서브넷 연결

엣지 연결

라우팅 전파

태그

라우팅 (2)

모

Q 라우팅 필터링

대상	대상	상태	전파됨
0.0.0.0/0	igw-06beb78766413d989	✓ 활성화	아니요
10.10.0.0/16	local	✓ 활성화	아니요

VPC > 라우팅 테이블 > rtb-0b9b8746d30336575

rtb-0b9b8746d30336575 / rt

작업 ▼

세부 정보 정보

라우팅 테이블 ID

rtb-0b9b8746d30336575

기본

아니요

명시적 서브넷 연결

2 서브넷

엣지 연결

-

VPC

vpc-0ca0cda67d620095b

소유자 ID

471112533833

라우팅

서브넷 연결

엣지 연결

라우팅 전파

태그

명시적 서브넷 연결 (2)

서브넷 연결 편집

Q 서브넷 연결 검색

< 1 > ⚙

이름	서브넷 ID	IPv4 CIDR	IPv6 CIDR
public-1	subnet-02242c4363fbae074	10.10.1.0/24	-
public-2	subnet-03d3eda499384972a	10.10.2.0/24	-

Security Group

보안 그룹 (3) 정보

🔄

작업 ▼

보안 그룹을 CSV로 내보내기

🔍 Find resources by attribute or tag

<input type="checkbox"/>	Name ▼	보안 그룹 ID ▼	보안 그룹 이름 ▼	VPC ID
<input type="checkbox"/>	-	sg-0f23804259ebe754d	default	vpc-0ca0cda67d620095b
<input type="checkbox"/>	-	sg-03d01c6b39bab63eb	default	vpc-047f70d2036bb9719
<input type="checkbox"/>	-	sg-0f04836d9eb635bc8	terraform-my-instance	vpc-0ca0cda67d620095b

sg-0f04836d9eb635bc8 - terraform-my-instance

작업 ▼

세부 정보

보안 그룹 이름

terraform-my-instance

보안 그룹 ID

sg-0f04836d9eb635bc8

설명

Managed by Terraform

VPC ID

[vpc-0ca0cda67d620095b](#)

소유자

471112533833

인바운드 규칙 수

1 권한 항목

아웃바운드 규칙 수

1 권한 항목

인바운드 규칙

아웃바운드 규칙

태그

인바운드 규칙 (1)

🔄

태그 관리

인바운드 규칙 편집

🔍 검색

< 1 > ⚙️

보안 그룹 규칙 ID ▼	IP 버전 ▼	유형 ▼	프로토콜 ▼	포트 범위 ▼	소스
sg-r07274b79a0d832...	IPv4	HTTP	TCP	80	0.0.0.0/0

sg-0f04836d9eb635bc8 - terraform-my-instance

작업 ▼

세부 정보

보안 그룹 이름

terraform-my-instance

보안 그룹 ID

sg-0f04836d9eb635bc8

설명

Managed by Terraform

VPC ID

[vpc-0ca0cda67d620095b](#)

소유자

471112533833

인바운드 규칙 수

1 권한 항목

아웃바운드 규칙 수

1 권한 항목

인바운드 규칙

아웃바운드 규칙

태그

아웃바운드 규칙 (1)

🔄

태그 관리

아웃바운드 규칙 편집

🔍 검색

< 1 > ⚙️

보안 그룹 규칙 ID ▼	IP 버전 ▼	유형 ▼	프로토콜 ▼	포트 범위 ▼	대상
sg-r0c4b7578262e277fc	IPv4	모든 트래픽	전체	전체	0.0.0.0/0

EC2 인스턴스

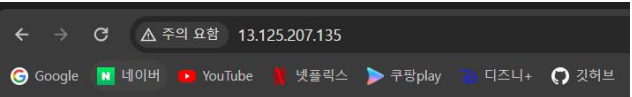
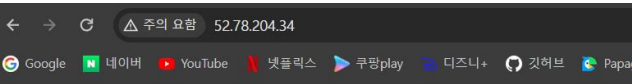
인스턴스 (2) 정보

인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

모든 상태

< 1 >

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역
project1	i-05e3f05ad65751130	실행 중	t2.nano	2/2개 검사 통과...	경보 보기	ap-northeast-2a
project2	i-021c5cb886e7fb352	실행 중	t2.nano	2/2개 검사 통과...	경보 보기	ap-northeast-2c



실습 3

Azure 나 GCP 환경에서 기본 인스턴스를 배포

우선 GCP Project 를 생성

➔ GCP는 Project가 필요. Project는 gcp 리소스들의 집합

Google Cloud 리소스, 문서, 제품 등 검색(/)

새 프로젝트

projects 할당량이 10개 남았습니다. 할당량 증가를 요청하거나 프로젝트를 삭제하세요. [자세히 알아보기](#) [MANAGE QUOTAS](#)

프로젝트 이름 *
terraformtest-5303

프로젝트 ID: terraformtest-5303. 나중에 변경할 수 없습니다. 수정

조직 *
kmu.kr

프로젝트에 연결할 조직을 선택하세요. 선택한 후에는 변경할 수 없습니다.

위치 *
kmu.kr [찾아보기](#)

상위 조직 또는 폴더

[만들기](#) [취소](#)

다음으로 compute engine api 를 활성화시킴

Compute engine api

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

Google Cloud terraformtest-5303

제품 세부정보

Compute Engine API
Google Enterprise API

Compute Engine API

[사용](#) [API 사용해 보기](#)

개요 문서 지원 관련 제품

개요
Creates and runs virtual machines on Google Cloud Platform.

추가 세부정보
유형: SaaS & APIs
최종 제품 업데이트: 23. 3. 24.
카테고리: [Compute](#), [Networking](#), [Google Enterprise APIs](#)
서비스 이름: compute.googleapis.com

가이드 및 문서

Service account

Service account 를 만들어줌. 다음과 같이 설정

➔ Google api에 인증하는데 사용



← 서비스 계정 만들기

1 서비스 계정 세부정보

서비스 계정 이름
terraform-account-5303
이 서비스 계정의 표시 이름입니다.

서비스 계정 ID *
terraform-account-5303
이메일 주소: terraform-account-5303@terraformtest-5303.iam.gserviceaccount.com

서비스 계정 설명
account for terraform-5303
이 서비스 계정에서 수행할 작업을 설명하세요.

[만들고 계속하기](#)

2 이 서비스 계정에 프로젝트에 대한 액세스 권한 부여 (선택사항)

← 서비스 계정 만들기

✓ 서비스 계정 세부정보

2 이 서비스 계정에 프로젝트에 대한 액세스 권한 부여 (선택사항)

프로젝트의 리소스에서 특정 작업을 완료할 수 있도록 이 서비스 계정에 terraformtest-5303에 대한 액세스 권한을 부여합니다. [자세히 알아보기](#)

역할
관리자
IAM 조건(선택사항)
+ IAM 조건 추가

대부분의 Google Cloud 리소스를 확인, 생성, 업데이트, 삭제합니다. 포함된 권한 목록을 참조하세요.

[+ 다른 역할 추가](#)

[계속](#)

3 사용자에게 이 서비스 계정에 대한 액세스 권한 부여 (선택사항)

[완료](#) 취소

← 서비스 계정 만들기

✓ 서비스 계정 세부정보

✓ 이 서비스 계정에 프로젝트에 대한 액세스 권한 부여 (선택사항)

3 사용자에게 이 서비스 계정에 대한 액세스 권한 부여 (선택사항)

Grant access to users or groups that need to perform actions as this service account. [Learn more](#)

서비스 계정 사용자 역할

사용자에게 이 서비스 계정을 사용하여 작업 및 VM을 배포할 권한을 부여합니다.

서비스 계정 관리자 역할

kimgh5303@kmu.kr

사용자에게 이 서비스 계정을 관리할 권한을 부여합니다.

완료

취소

이후 json 형태의 key 를 생성해줌. 다음과 같이 생성

← Compute Engine default service account

세부정보

권한

키

속성항목

로그

키

⚠ 보안 침해 시 서비스 계정 키로 인해 보안 위험이 발생할 수 있습니다. 서비스 계정 하는 데 적합한 방법은 [여기](#)에서 자세히 알아볼 수 있습니다.

새 키 쌍을 추가하거나 기존 키 쌍의 공개키 인증서를 업로드하세요.

[조직 정책](#)을 사용하여 서비스 계정 키 생성을 차단합니다.

[서비스 계정의 조직 정책 설정 자세히 알아보기](#)

키 추가

유형

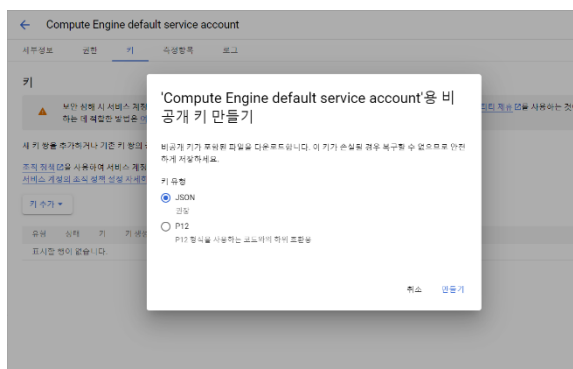
상태

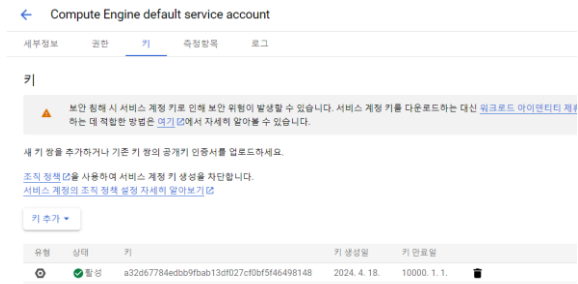
키

키 생성일

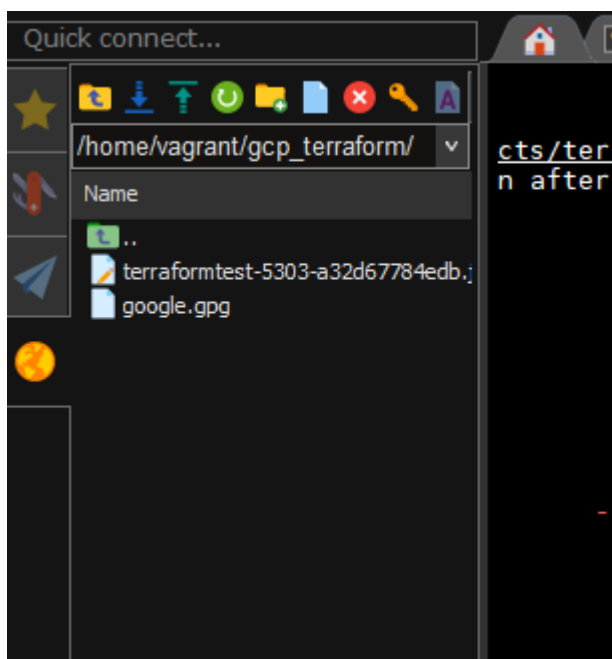
키 만료일

표시할 행이 없습니다.





해당 json 파일은 다운로드 하여 ubuntu terraform 경로에 둘 것



IAM API

➔ 활성화시켜줌



Identity and Access Management (IAM) API

[Google Enterprise API](#)

Manages identity and access control for Google Cloud resources, including the creation of service...

관리

[API 사용해 보기](#)

✓ API 사용 설정됨

개요

문서

관련 제품

작성한 코드를 바탕으로 main.tf 를 구성함

Provider

```
cat <<EOT > main.tf
# Google Cloud Provider
provider "google" {
  credentials = file("/home/vagrant/gcp_terraform/terraformtest-5303-a32d67784edb.json") # 서비스 계정 키 파일 경로
  project     = "terraformtest-5303" # 프로젝트 ID
  region      = "asia-northeast3" # 기본 리전 설정
  zone        = "asia-northeast3-a"
}
```

VPC

```
# 기본 VPC 네트워크 정보 조회
resource "google_compute_network" "main" {
  name = "terraform-network" # 네트워크 이름
}
```

방화벽

```
# 기본 VPC 네트워크 정보 조회
resource "google_compute_network" "main" {
  name = "terraform-network" # 네트워크 이름
}

# HTTP 트래픽을 허용하는 방화벽 규칙 생성
resource "google_compute_firewall" "allow_http" {
  name      = "allow-http"
  network   = google_compute_network.main.name

  allow {
    protocol = "tcp"
    ports    = ["80", "8080"]
  }

  target_tags = ["web"]
  source_ranges = ["0.0.0.0/0"]
}
```

인스턴스

```
# 가상 머신 인스턴스 생성
resource "google_compute_instance" "instance" {
  name         = "my-instance" # 인스턴스 이름
  machine_type = "n2-standard-2" # 머신 타입
  tags         = ["web"]

  # 부팅 디스크 설정
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11" # 사용할 이미지
    }
  }

  # 네트워크 인터페이스 설정
  network_interface {
    network = google_compute_network.main.self_link
    access_config {
      # Ephemeral public IP 할당 설정
    }
  }

  # 스타트업 스크립트 설정
  metadata_startup_script = <<-EOF
  #!/bin/bash
  sudo apt update -y
  sudo apt install -y apache2
  sudo systemctl start apache2
  sudo systemctl enable apache2

  echo "<img src='\"https://i.namu.wiki/i/LVX-N10dzyEF_B76STHoR9SpkwVgcMU5h0n-hwL3\"'"
  sudo systemctl restart apache2
EOF
}
```

Terraform apply

적용 후 결과를 확인해보겠습니다

```
asia-northeast3-a/instances/my-instance]
google_compute_instance.instance: Still destroying... [id=projects/terraformtest-5303/
zones/asia-northeast3-a/instances/my-instance, 10s elapsed]
google_compute_instance.instance: Still destroying... [id=projects/terraformtest-5303/
zones/asia-northeast3-a/instances/my-instance, 20s elapsed]
google_compute_instance.instance: Still destroying... [id=projects/terraformtest-5303/
zones/asia-northeast3-a/instances/my-instance, 30s elapsed]
google_compute_instance.instance: Still destroying... [id=projects/terraformtest-5303/
zones/asia-northeast3-a/instances/my-instance, 40s elapsed]
google_compute_instance.instance: Still destroying... [id=projects/terraformtest-5303/
zones/asia-northeast3-a/instances/my-instance, 50s elapsed]
google_compute_instance.instance: Destruction complete after 51s
google_compute_instance.instance: Creating...
google_compute_instance.instance: Still creating... [10s elapsed]
google_compute_instance.instance: Creation complete after 14s [id=projects/terraformte
st-5303/zones/asia-northeast3-a/instances/my-instance]

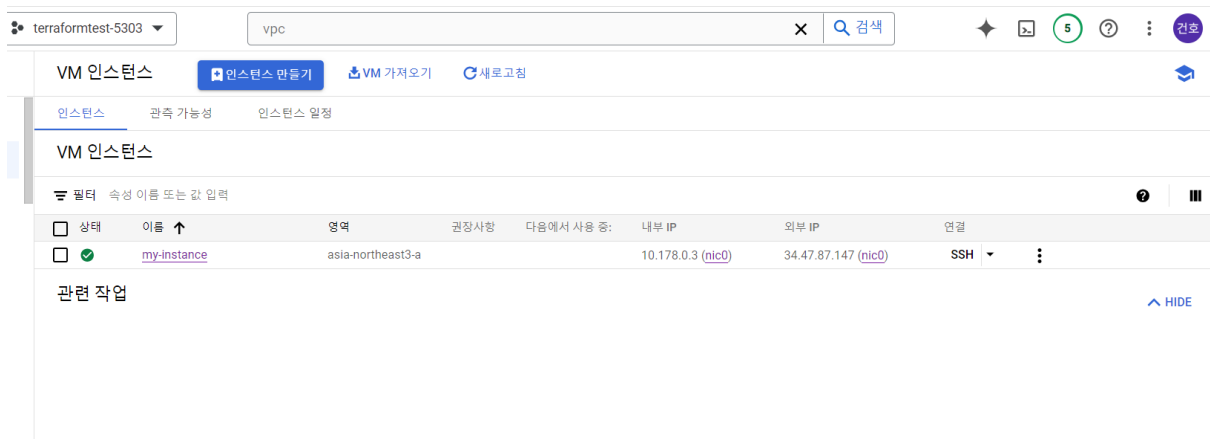
Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

public_ip = "34.47.87.147"
vagrant@ubuntu-jammy:~/gcp_terraform$
```

다음과 같이 적용 되었습니다

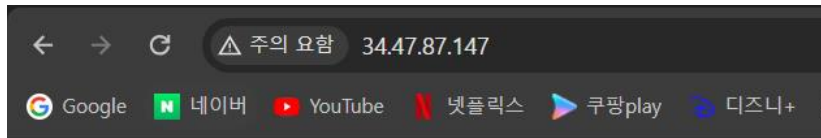
Console 창 확인



상태	이름	영역	권장사항	다음에서 사용 중:	내부 IP	외부 IP	연결
<input checked="" type="checkbox"/>	my-instance	asia-northeast3-a			10.178.0.3 (nic0)	34.47.87.147 (nic0)	SSH

GCP console 에도 생성된 것을 확인

링크로 들어가보면



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>