



★
APEX
Terraform Project

이정일 김민호 서재권 임재근



<목차>

1. Terraform 정의

- Terraform
- IaC
- 프로비저닝
- 장점

2. 환경 구축

- Terraform
- AWS CLI
- VSCode

3. Terraform 기본 개념

- provider
- resource
- variable
- output
- terraform.tfstate 파일

4. Terraform 작동 원리

- init
- Validate
- Plan
- apply
- destroy

5. TF 파일 분석

- 구조 및 구성도
- Code

6. Github ,Terraform Cloud, AWS 연동

- GitHub Repository 만들기
- Terraform Cloud 초기 구성 & GitHub 연동
- Terraform Cloud & AWS 연동
- GitHub를 통한 자동화 배포

7. Route53 도메인 등록

- 도메인 확인
- 도메인 등록업체 NS 레코드 입력
- 가비아에서 확인
- 호스트 영역 생성



1. Terraform의 정의

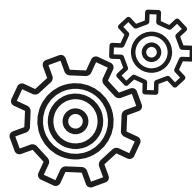
1 - 1 Terraform

1 - 2 IaC

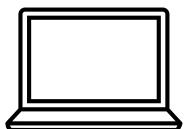
1 - 3 프로비저닝

1 - 4 장점

1-1. Terraform



HashiCorp에서 개발한 오픈소스 인프라 관리 도구로,
인프라를 코드(IaC) 형태로 작성해 자동화된 관리 가능



선언적 구문(HCL/JSON)을 사용하여 사용자가 원하는
인프라를 정의하면 Terraform이 자동으로 구축 처리



AWS, Azure, GCP 등 멀티 클라우드 및 온프레미스 환
경의 자원 프로비저닝과 관리 지원



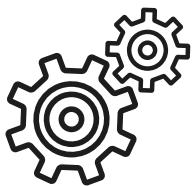
1-2. IaC

IaC는 “인프라를 코드로 관리한다”는 개념으로, 서버·네트워크·DB 설정
을 코드화하여 자동화

선언적(Declarative) 방식과 명령적(Imperative) 방식을 통해 인프라
를 정의하고 운영

Terraform 외에도 CloudFormation, Ansible, Chef, Kubernetes
Manifests 등 다양한 도구 존재

1-3. 프로비저닝



IT 시스템에서 서버, 네트워크, 데이터베이스 등 필요한 자원을 준비하고 설정하는 과정



클라우드에서 서버 생성, 데이터베이스 설정, 네트워크 할당 등이 포함



자동화 도구를 사용하면 구축 시간을 단축하고 오류를 줄이며, 확장성 있는 인프라 관리가 가능



1-4. 장점

자동화

Terraform으로 코드 기반 인프라 관리를 통해 반복 작업을 줄이고, 빠르고 정확하게 리소스를 배포

일관성

동일한 코드를 사용해 개발, 테스트, 운영 환경 간 차이를 없애고 예측 가능한 인프라 구성이 가능

효율성 및 버전 관리

Git으로 코드 변경 이력을 관리하고, CI/CD와 연동해 안정적으로 배포 할 수 있으며, 복잡한 인프라도 빠르게 운영



2. 환경 구축

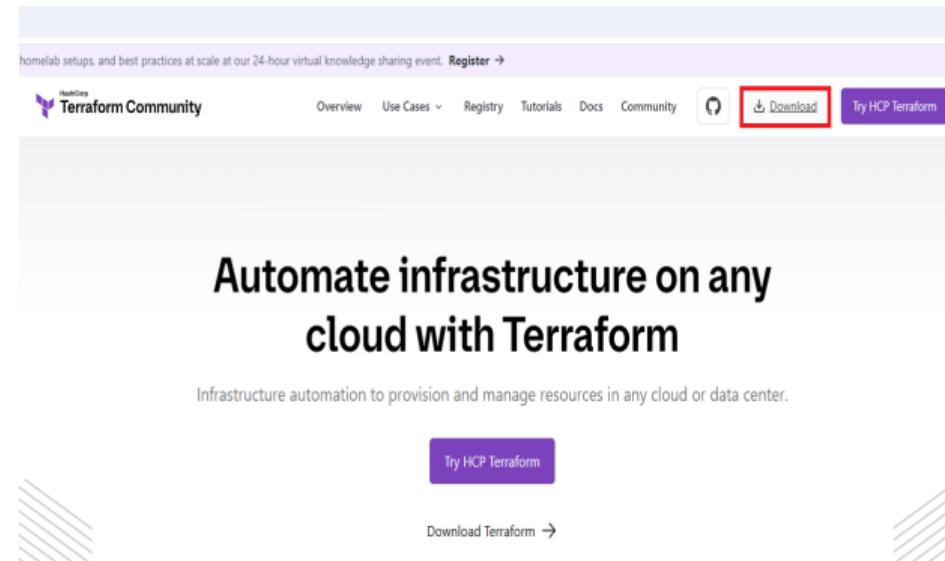
2 - 1 Terraform

2 - 2 AWS CLI

2- 3 VSCode

기간 : XX-XX-XX ~ XX-XX-XX | 기여도 : 100%

2-1. Terraform 설치

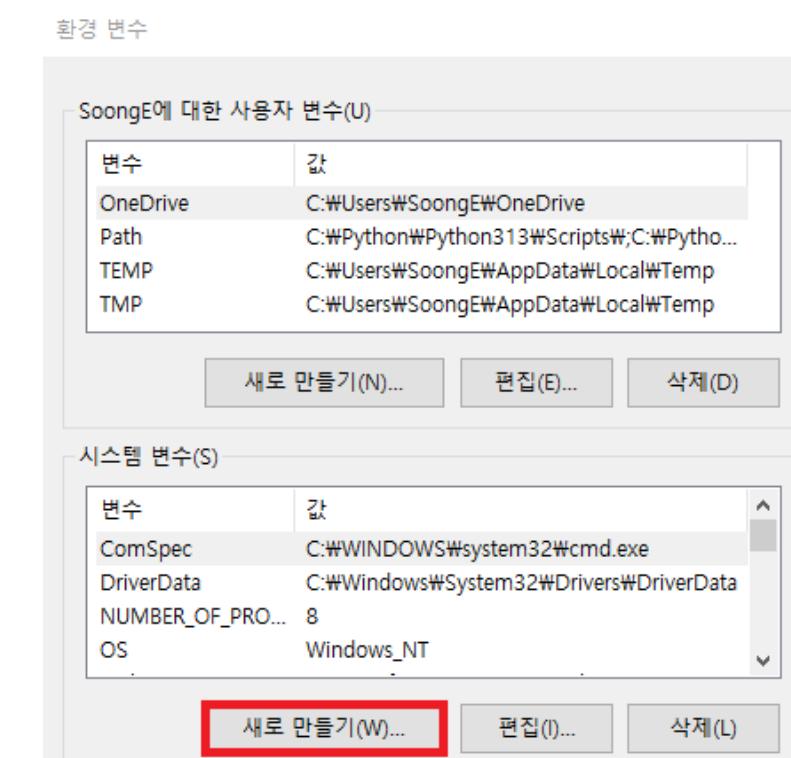


Download -> Windows AMD64 다운로드(32bit는 386) -> 압축 해제

terraform.exe 파일을 원하는 경로 디렉터리에 저장 (ex. C:\Terraform)



윈도우 버튼 클릭 -> 시스템 환경 변수 편집 -> 고급 -> 환경 변수



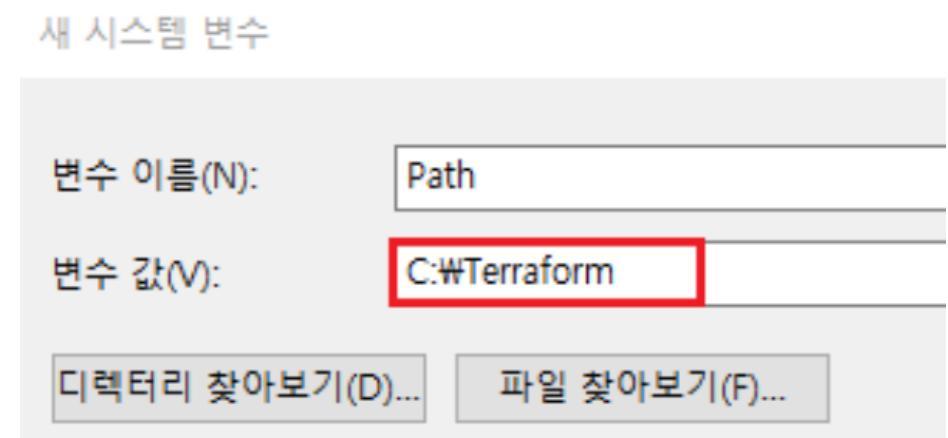
시스템 변수의 Path에 Terraform 경로 추가

```
명령 프롬프트
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SoongE>terraform version
Terraform v1.9.8
on windows_amd64

Your version of Terraform is out of date! The latest version
is 1.10.2. You can update by downloading from https://www.terraform.io/downloads.html
```

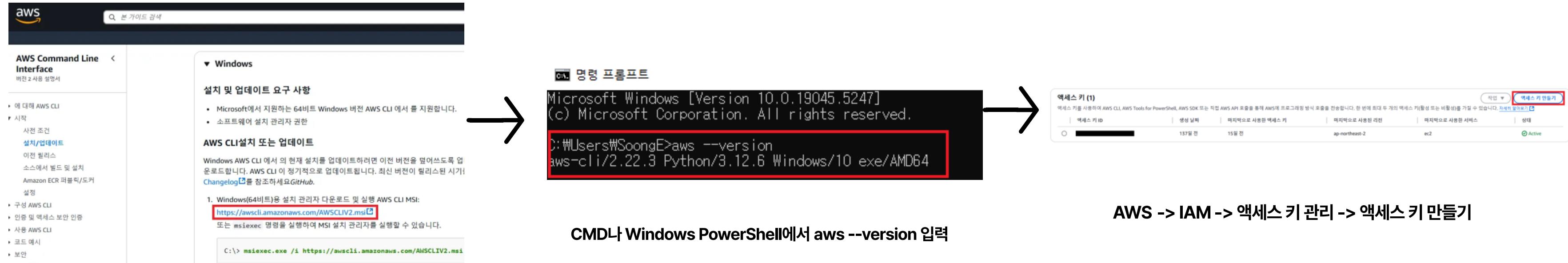
CMD나 Windows PowerShell에서 terraform version 입력 -> terraform 설치 확인



새로 만들기 -> 변수 이름 = Path, 변수 값 = C:\Terraform 입력

변수 이름을 Path로 주지 않으면 인식 못함

2-2. AWS CLI 연동



Windows -> <https://awscli.amazonaws.com/AWSCLIV2.msi> 다운로드 후 실행

```
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SoongE>aws configure list
  Name                Value            Type    Location
  ----              -----          ----    -----
  profile
  access_key        <not set>      None    None
  secret_key        ****V7PL****      shared-credentials-file
  region            ap-northeast-2    config-file  ~/.aws/config
```

CMDL Windows PowerShell에서 aws configure list 입력 -> AWS 로그인 확인

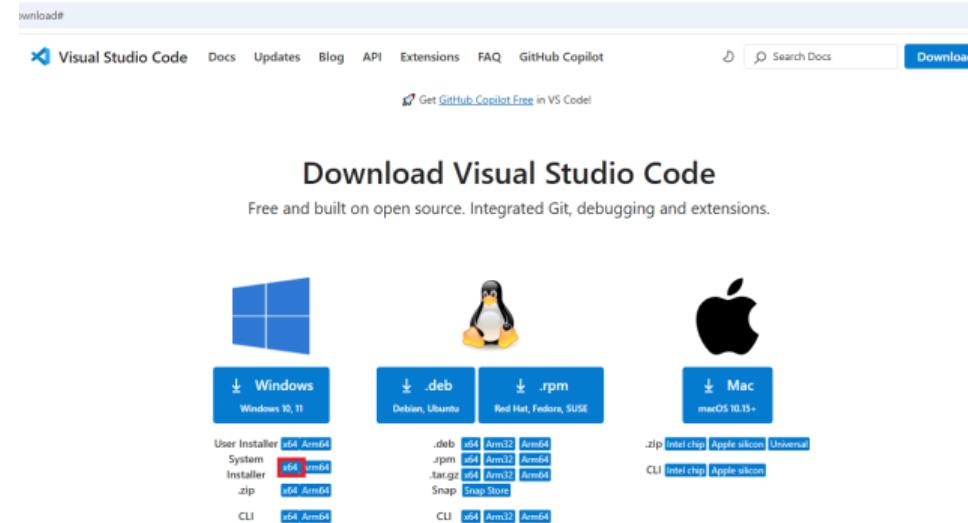
```
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SoongE>aws configure
AWS Access Key ID [*****V7PL*]:
AWS Secret Access Key [*****CGlc*]:
Default region name [ap-northeast-2]:
Default output format [None]:
```

CMDL Windows PowerShell에서 aws configure 입력

Access Key ID, Secret Access Key, Default region 입력

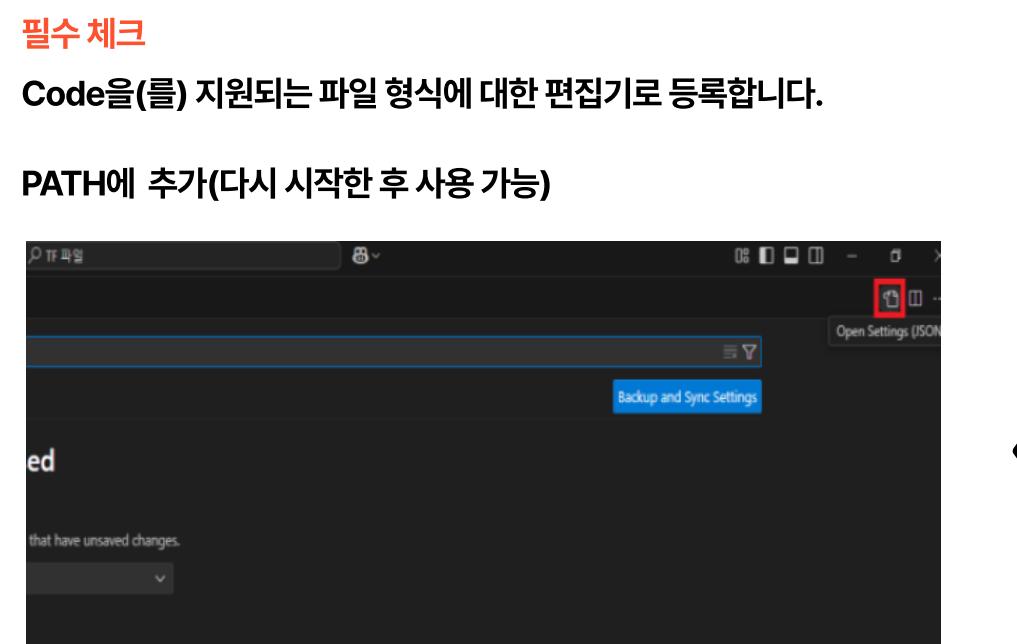
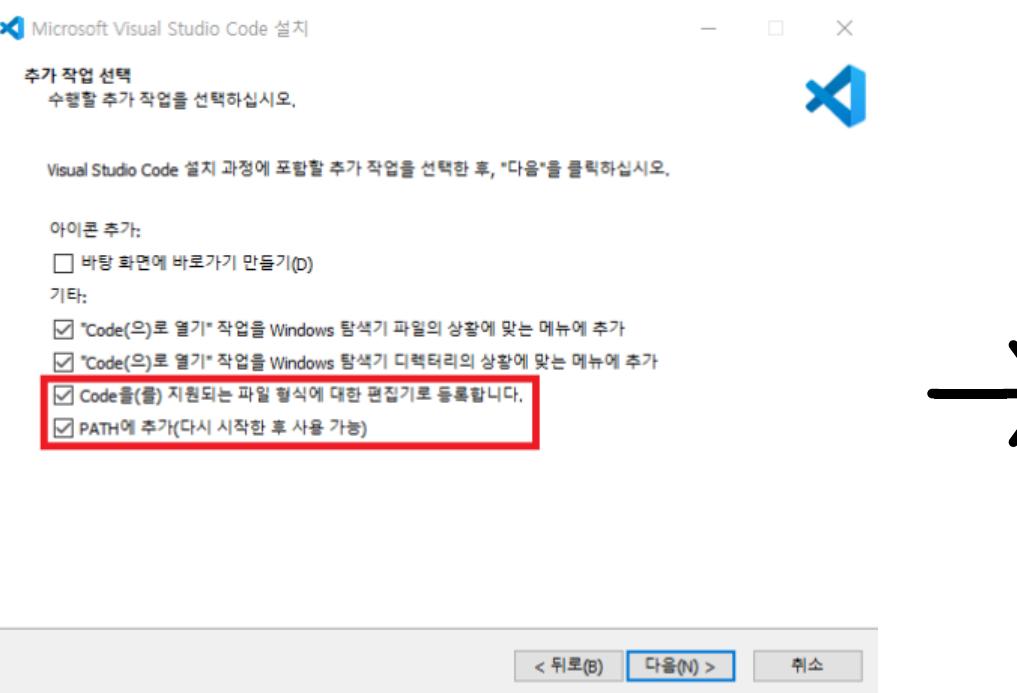
2-3. VSCode 설치



Download → Windows System Installer x64

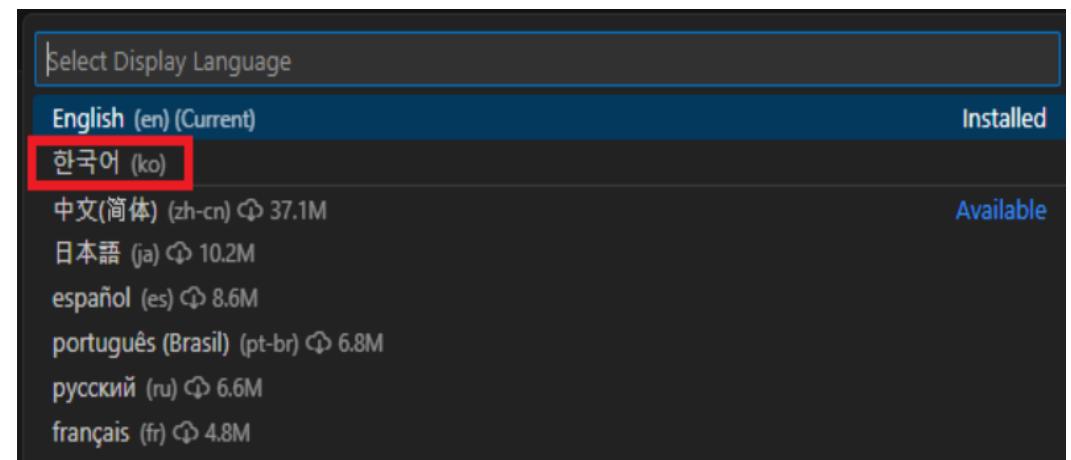
```
Settings settings.json X
C: > Users > SoongE > AppData > Roaming > Code > User > settings.json > ...
1  {
2    "editor.formatOnSave": true,
3    "editor.formatOnSaveMode": "file",
4    "[terraform]": {
5      "editor.defaultFormatter": "hashicorp.terraform"
6    },
7    "[terraform-vars)": {
8      "editor.formatOnSave": true,
9      "editor.defaultFormatter": "hashicorp.terraform",
10     "editor.formatOnSaveMode": "file"
11   },
12   "security.workspace.trust.untrustedFiles": "open"
13 }
14 }
```

아래 JSON 내용 추가 후 저장(Ctrl+S)

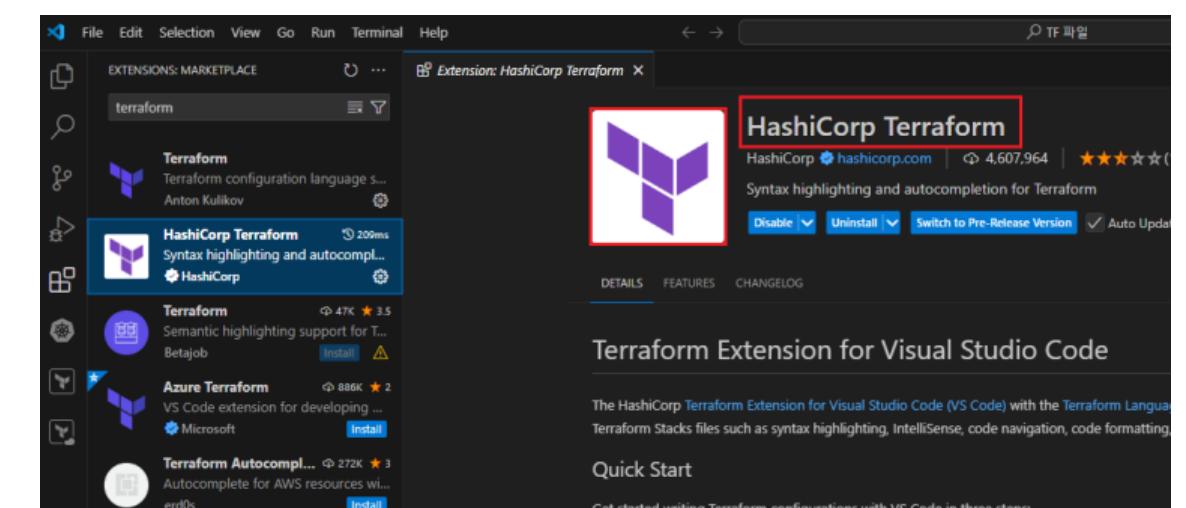


Terraform 파일을 자동으로 포맷팅하도록 구성하는 설정

왼쪽 하단 톱니바퀴 클릭 -> Settings -> 오른쪽 상단 파일 아이콘(Open Settings JSON) 클릭



language 검색 -> Configure Display Language(표시 언어 구성) 선택 → 한국어 선택



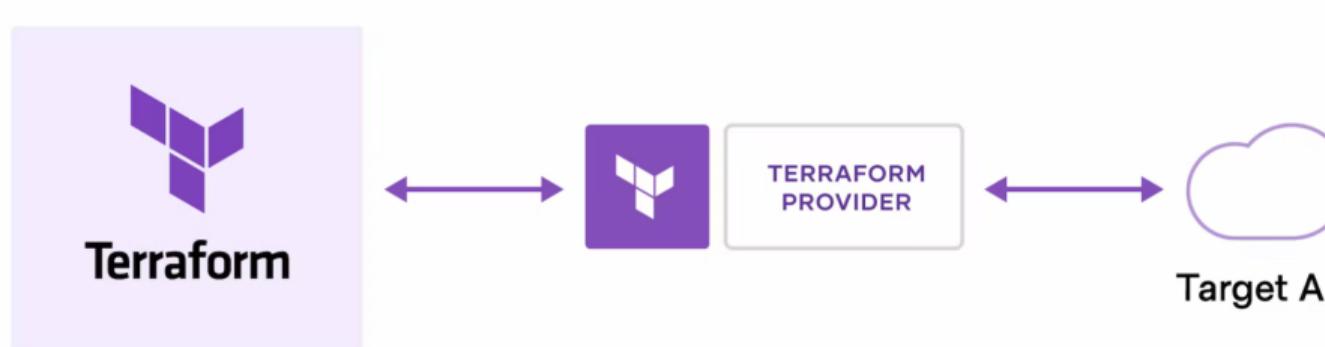
검색 창에 Terraform 입력 → HashiCorp Terraform 설치



3. Terraform 기본 개념

- 3-1. provider
 - 3-2. resource
 - 3-3. variable
 - 3-4. output
 - 3-5. terraform.tfstate 파일
-

3-1. provider



<provider 기본 구조>

```
provider "<PROVIDER_NAME>" {  
    # Configuration arguments  
}
```



Terraform이 인프라를 생성할 때 사용할 클라우드 플랫폼이나 서비스를 정의

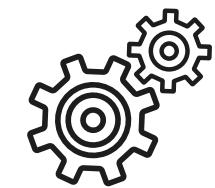


각 클라우드 플랫폼에 맞는 provider를 지정해야 인프라를 생성할 수 있음



프로바이더는 Terraform 핵심 소프트웨어와는 독립적으로 출시되는 플러그인이며, 자체적인 버전 관리를 가집니다.

3-2. resource



실제로 생성할 인프라 자원을 정의
EC2 인스턴스, VPC, RDS 등 리소스를 **프로비저닝**

<resource 기본 구조>

```
resource "<PROVIDER>_<RESOURCE_TYPE>" "<NAME>" {  
  # Configuration arguments  
}
```



리소스 블럭은 테라폼에서 관리한 인프라 구성 요소를 정의하는
기본적인 빌딩 블럭

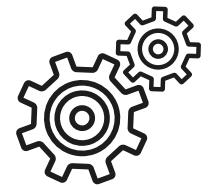


사용법: resource 키워드를 사용해 정의, 각 블럭은 리소스 유
형과 구성 내에서 이를 참조할 이름을 지정

3-3. variable

<variable의 기본구조>

```
variable "<NAME>" {
  type    = <TYPE>
  description = "<DESCRIPTION>"
  default   = <DEFAULT_VALUE>
}
```



인프라 구성 코드를 유연하고 재사용 가능하게 만드는데 필수적인 요소
마치 프로그래밍 언어의 변수처럼, 하드코딩된 값을 직접 사용하는 대신
변수를 통해 값을 전달받아 사용 가능

<variable의 타입>

Type	설명 (Description)	Example
String	텍스트를 나타냄	"example_string"
Number	숫자 값을 나타냄	42
Bool	참 또는 거짓 값을 나타냄	true/false
List	값들의 순서가 있는 시퀀스	["item1", "item2"]
Map	키-값 쌍의 집합	{"key1" = "value1"}
Set	고유한 값들의 집합	set("item1", "item2")
Object	이름이 지정된 속성들의 집합	object({name=string})
Tuple	서로 다른 타입 값들의 시퀀스	tuple([string, number])

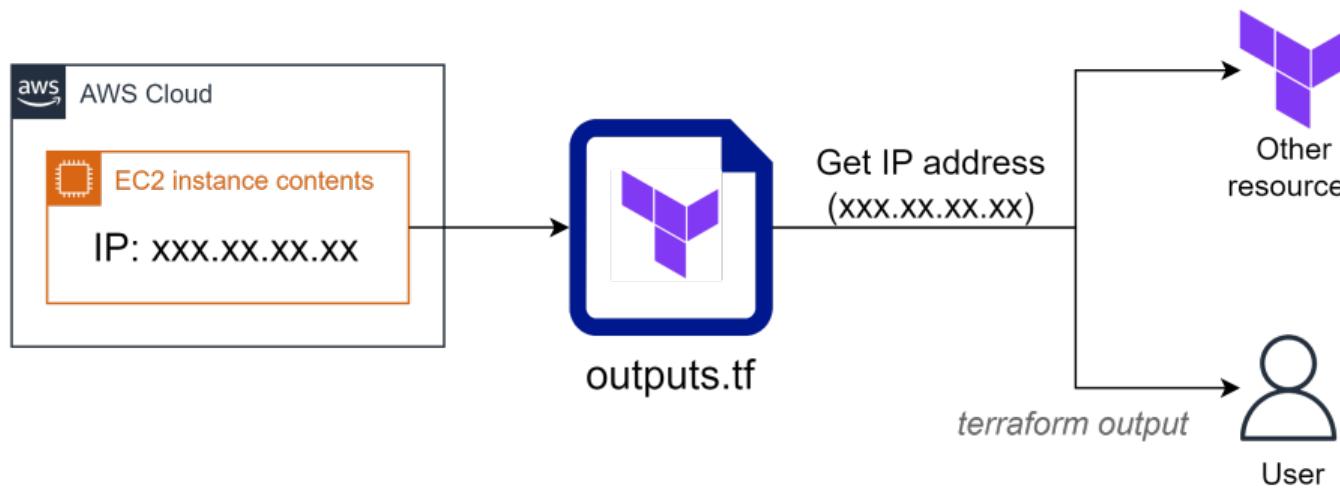


코드에서 값이 변경될 수 있는 파라미터를 정의
변수는 사용자로부터 입력받거나, 다른 모듈에서 값을 전달받을
때 사용



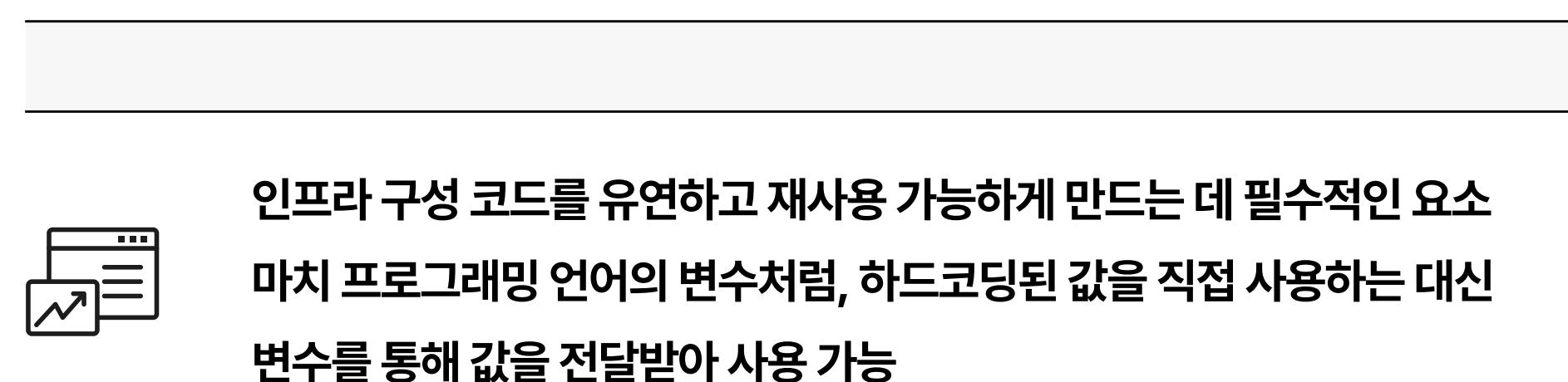
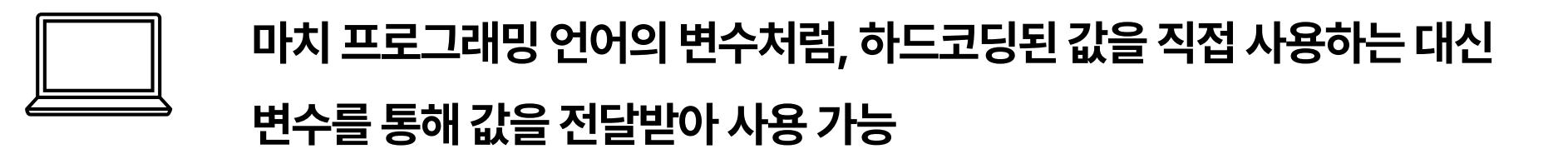
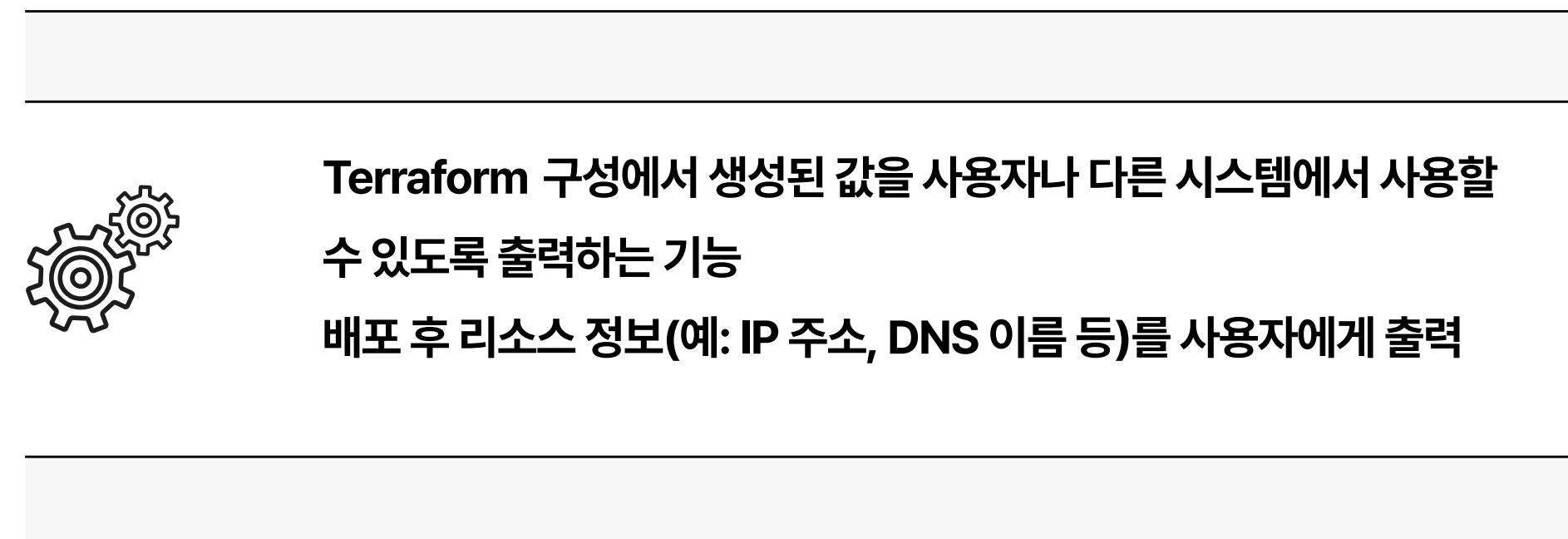
variable을 사용하면 코드의 재사용성과 유연성을 높일 수 있음

3-4. Output

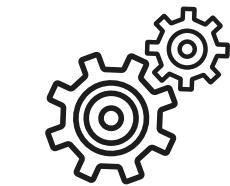
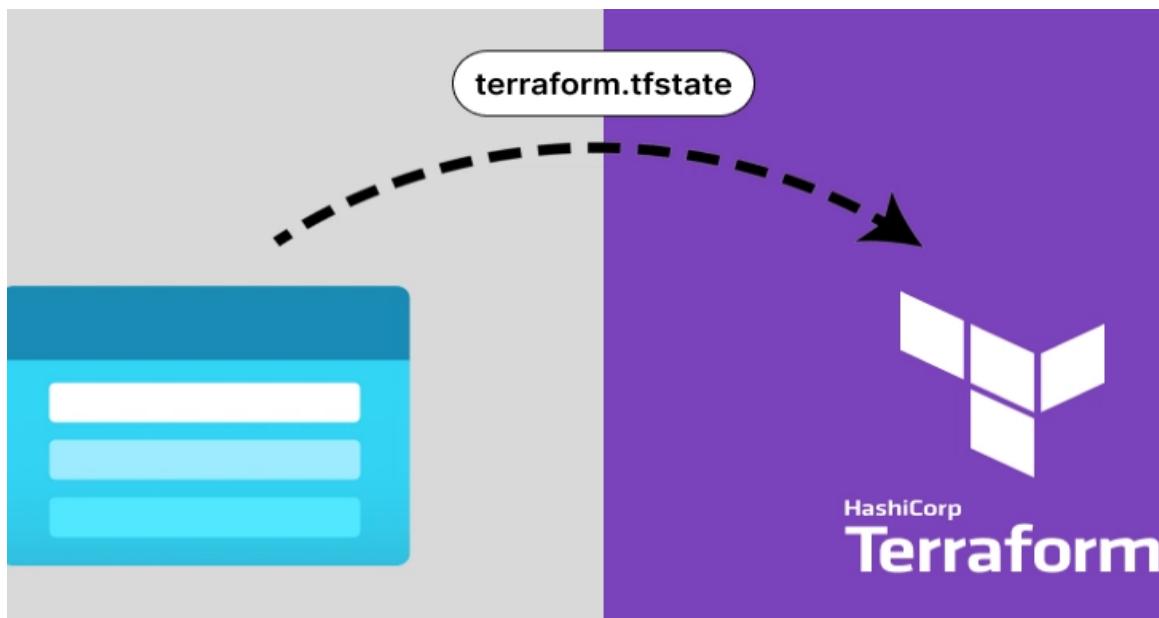


<output의 기본 구조>

terraform output [NAME]



3-5. `terraform.tfstate` 파일



Terraform이 관리하는 인프라의 현재 상태를 저장하는 파일

Terraform이 인프라 리소스의 상태를 추적하고 관리 명령을 실행할 때 참조



Terraform이 배포한 리소스(VPC, EC2 인스턴스 등)의 현재 상태 정보를
JSON 형식으로 저장

version, resources, outputs, modules 등의 정보가 저장되어 있음



팀 협업 시 Terraform Backend(S3, Azure Blob Storage 등)를
설정하여 상태 파일을 원격에서 관리



4. Terraform 작동원리

4-1. init

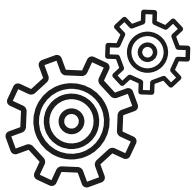
4-2. Validate

4-3. Plan

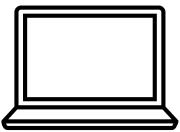
4-4. apply

4-5. destroy

4-1. init



Terraform 작업 디렉토리를 초기화하고,
구성 파일에 정의된 Provider 플러그인을 다운로드
작업 디렉토리를 테라폼이 사용할 수 있는 환경으로 설정



.terraform 디렉토리가 생성됨
사용자의 테라폼 코드가 들어갈 작업 디렉터리를 초기화



모듈과 플러그인 같은 보조적인 구성요소를 다운로드
테라폼 상태 파일을 저장할 백 엔드를 구성



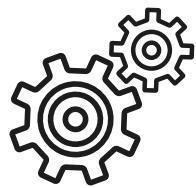
4-2. Validate

Terraform 구성 파일의 문법과 논리적 유효성을 검사하여
오류를 미리 확인하는 명령어

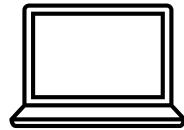
terraform plan에서도 구문 오류를 확인하기에 생략 가능
로컬에서만 검사를 수행하며 원격 서비스나 상태와 상호작용하지 않음

CI/CD 파이프라인에서는 유용함
파이프라인 단계에서 validate를 먼저 실행해 코드 오류를 빠르게 확인하고, 코드 구
문이 올바른 경우에만 plan을 실행하도록 설정해서 불필요한 작업 및 리소스를 줄일
수 있음

4-3. plan



현재 상태와 코드에서 정의된 원하는 상태 간의 차이를 계산하여, 적용할 변경 사항을 미리 보여줌
변경사항이 실제 인프라에 적용되기 전에 검토할 수 있도록 하는 기능



수행할 작업을 시뮬레이션하여 예측 결과를 보여줌
kubernetes 파드 생성 명령어중에서 --dry-run=client와 유사



상태 파일(`terraform.tfstate`) 및 클라우드 환경과 상호작용
사용자는 변경 사항을 적용하기 전에 어떤 작업이 수행될지
미리 검토 가능



4-4. apply

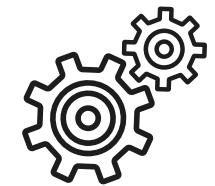
`terraform plan`에서 계산된 변경 사항을 실제로 인프라에 적용
적용 후, 변경된 상태를 상태 파일(`terraform.tfstate`)에 저장

- 옵션

- `auto-approve` : 승인 없이 실행
- `replace=<resource>` : 특정 리소스 강제 재생성

일부 리소스 생성/변경이 실패하면, 그 시점에서 멈추고 상태는 반영된 부분까지만 저장
자동으로 롤백하지 않기 때문에 수동 정리 필요

4-5. destroy



테라폼 구성 파일에 정의된 모든 리소스를 삭제
상태 파일(`terraform.tfstate`)도 업데이트됨



단, `destroy`는 `apply`로 생성한 인프라 리소스의 설정 사항 변동이 없을 때만 가능
(임의 변경 후 `destroy` 실행 시, 지워지지 않고 에러 발생)



`terraform destroy -auto-approve` : 확인 없이 바로 삭제
`terraform destroy -target=aws_instance.my_ec2` : 특정 리소스만 삭제



5. TF 파일 분석

5 - 1 구조 및 구성도

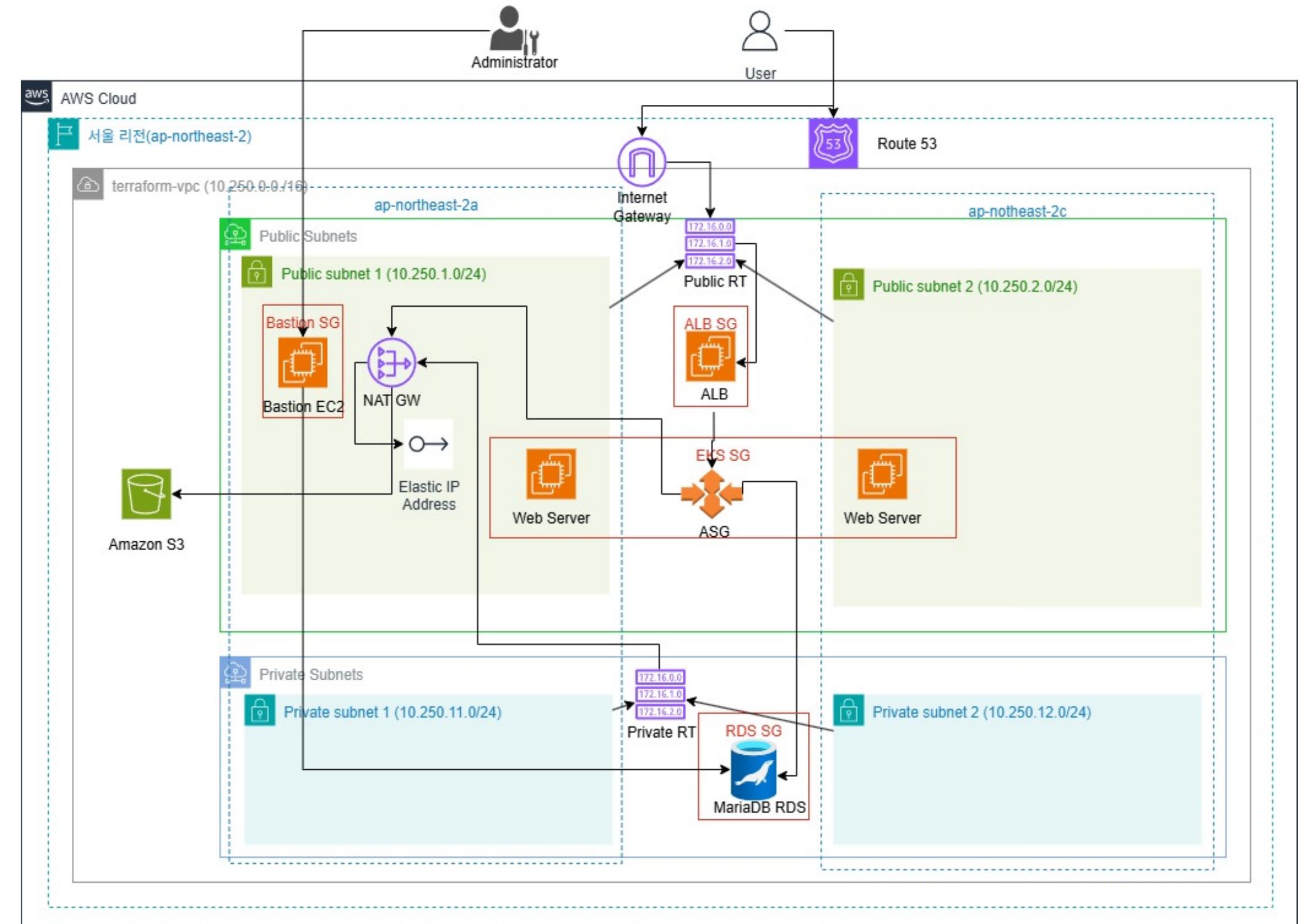
5 - 2 Code

기간 : XX-XX-XX ~ XX-XX-XX | 기여도 : 100%

5-1. 구조 및 구성도

<구조>

```
terraform-class/
├── 01.provider.tf
├── 02.vpc.tf
├── 03.subnet.tf
├── 04.igw.tf
├── 05.ngw.tf
├── 06.routing-table.tf
├── 07.security-group.tf
├── 08.ec2.tf
├── 09.key_pair.tf
├── 10.rds.tf
├── 11.s3.tf
├── 12.acm.tf
├── 13.route53.tf
├── 14.alb.tf
└── 15.Auto.tf
```



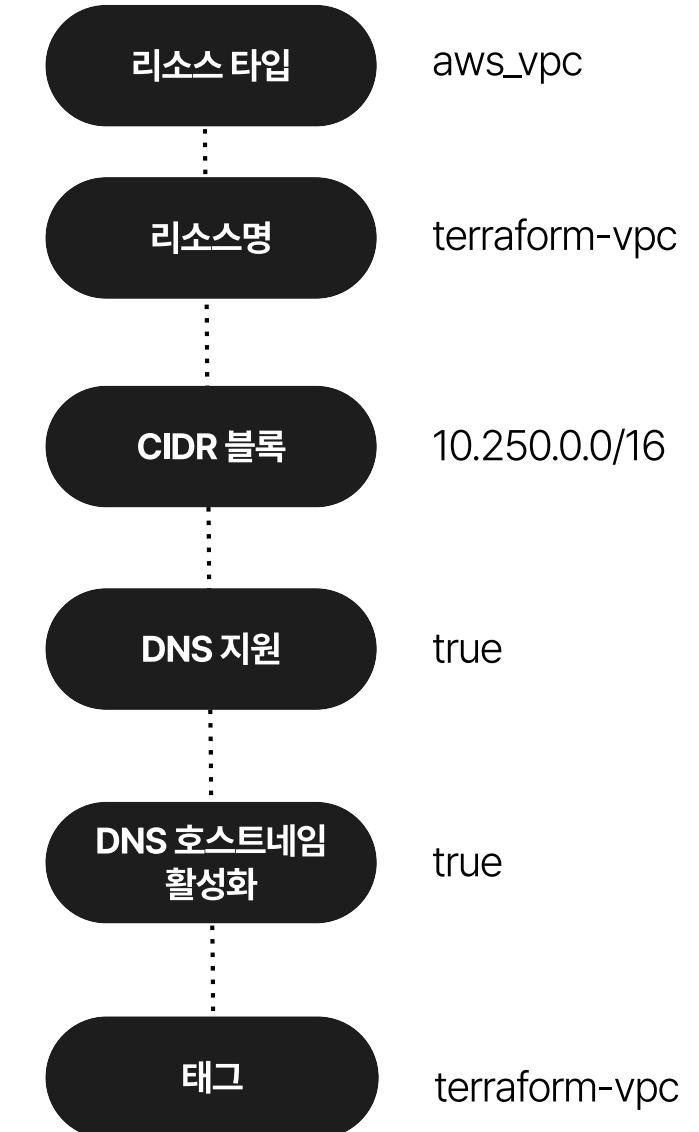
1.Provider.tf

```
terraform {  
    required_version = ">= 1.0.0, <2.0.0"  
  
    required_providers {  
        aws = {  
            source  = "hashicorp/aws"  
            version = "~> 5.0"  
        }  
    }  
}  
  
provider "aws" {  
    region = "ap-northeast-2" #Asia  
    Pacific (seoul) region  
}
```



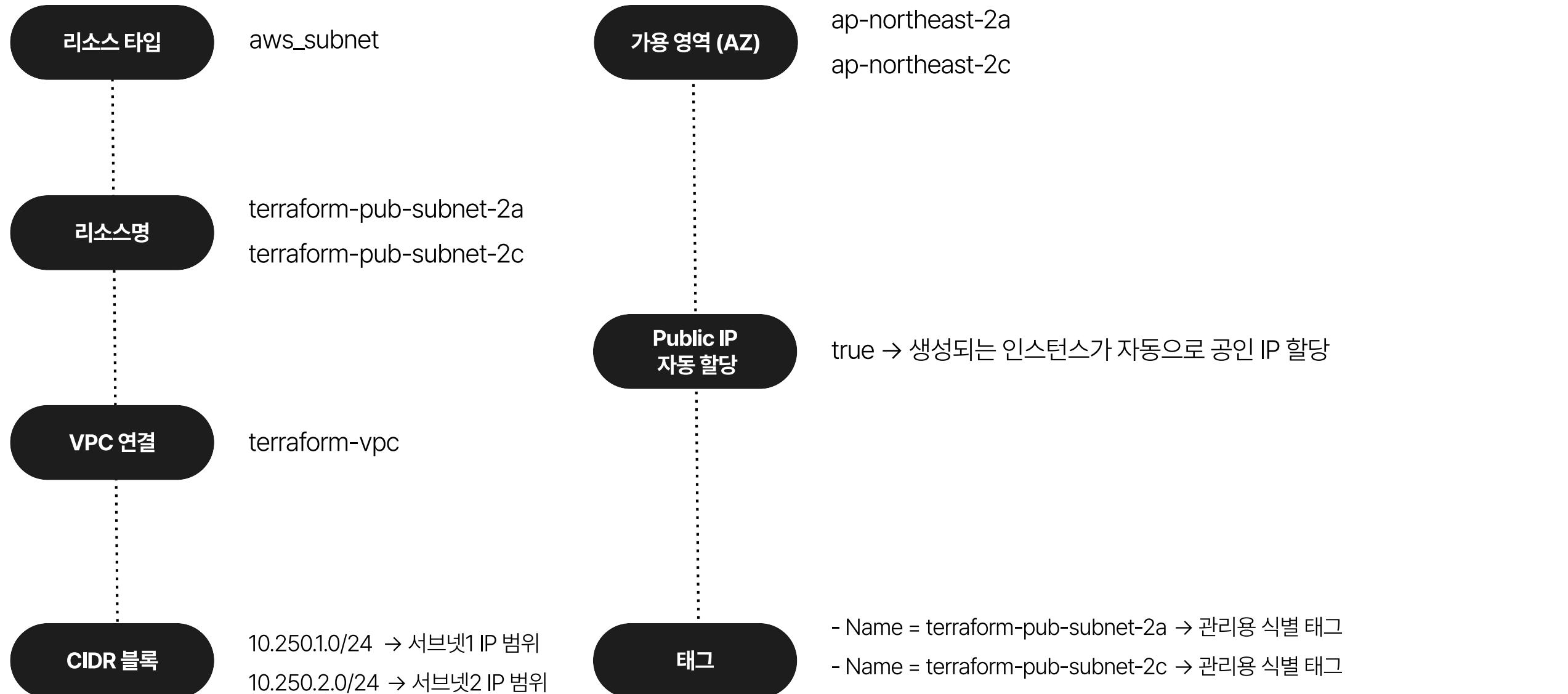
2. VPC.tf

```
resource "aws_vpc" "terraform-vpc" {  
cidr_block      = "10.250.0.0/16"  
enable_dns_support = true  
enable_dns_hostnames = true  
tags = {  
  "Name" = "terraform-vpc"  
}  
}
```



3.Subnet.tf(Public)

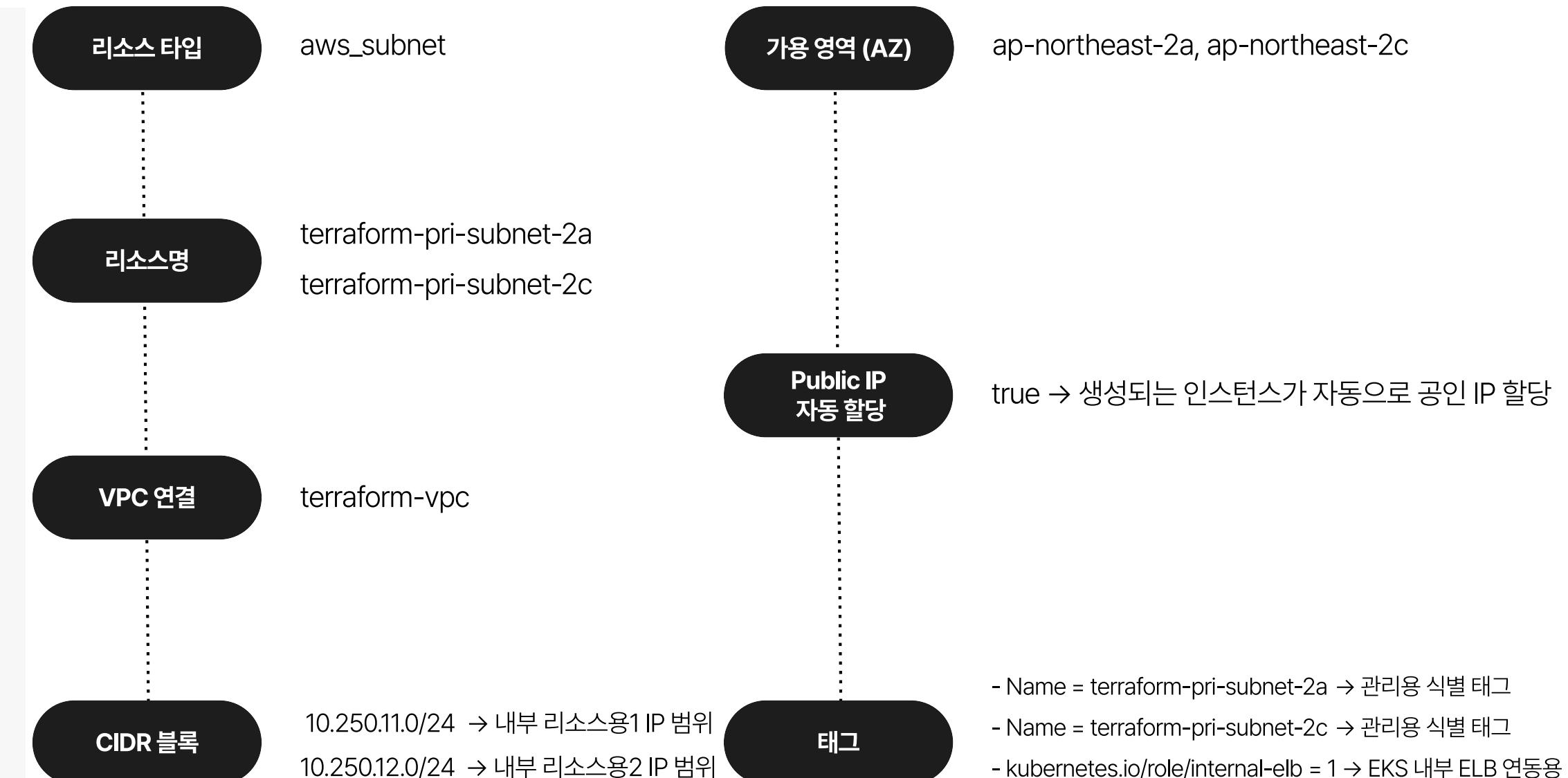
```
# Public Subnet
resource "aws_subnet" "terraform-pub-subnet-2a" {
  vpc_id      = aws_vpc.terraform-vpc.id
  cidr_block   = "10.250.1.0/24"
  availability_zone = "ap-northeast-2a"
  map_public_ip_on_launch = "true"
  tags = {
    "Name"      = "terraform-pub-subnet-2a"
    "kubernetes.io/role/elb" = "1"
    "kubernetes.io/cluster/terraform-eks-cluster" = "shared"
  }
}
resource "aws_subnet" "terraform-pub-subnet-2c" {
  vpc_id      = aws_vpc.terraform-vpc.id
  cidr_block   = "10.250.2.0/24"
  availability_zone = "ap-northeast-2c"
  map_public_ip_on_launch = "true"
  tags = {
    "Name"      = "terraform-pub-subnet-2c"
    "kubernetes.io/role/elb" = "1"
    "kubernetes.io/cluster/terraform-eks-cluster" = "shared"
  }
}
```



3. Subnet.tf(Private)

```
# Private Subnet
resource "aws_subnet" "terraform-pri-subnet-2a" {
  vpc_id      = aws_vpc.terraform-vpc.id
  cidr_block   = "10.250.11.0/24"
  availability_zone = "ap-northeast-2a"
  tags = {
    "Name"          = "terraform-pri-subnet-2a"
    "kubernetes.io/role/internal-elb" = "1"
    "kubernetes.io/cluster/terraform-eks-cluster" =
    "shared"
  }
}

resource "aws_subnet" "terraform-pri-subnet-2c" {
  vpc_id      = aws_vpc.terraform-vpc.id
  cidr_block   = "10.250.12.0/24"
  availability_zone = "ap-northeast-2c"
  tags = {
    "Name"          = "terraform-pri-subnet-2c"
    "kubernetes.io/role/internal-elb" = "1"
    "kubernetes.io/cluster/terraform-eks-cluster" =
    "shared"
  }
}
```



4. IGW.tf

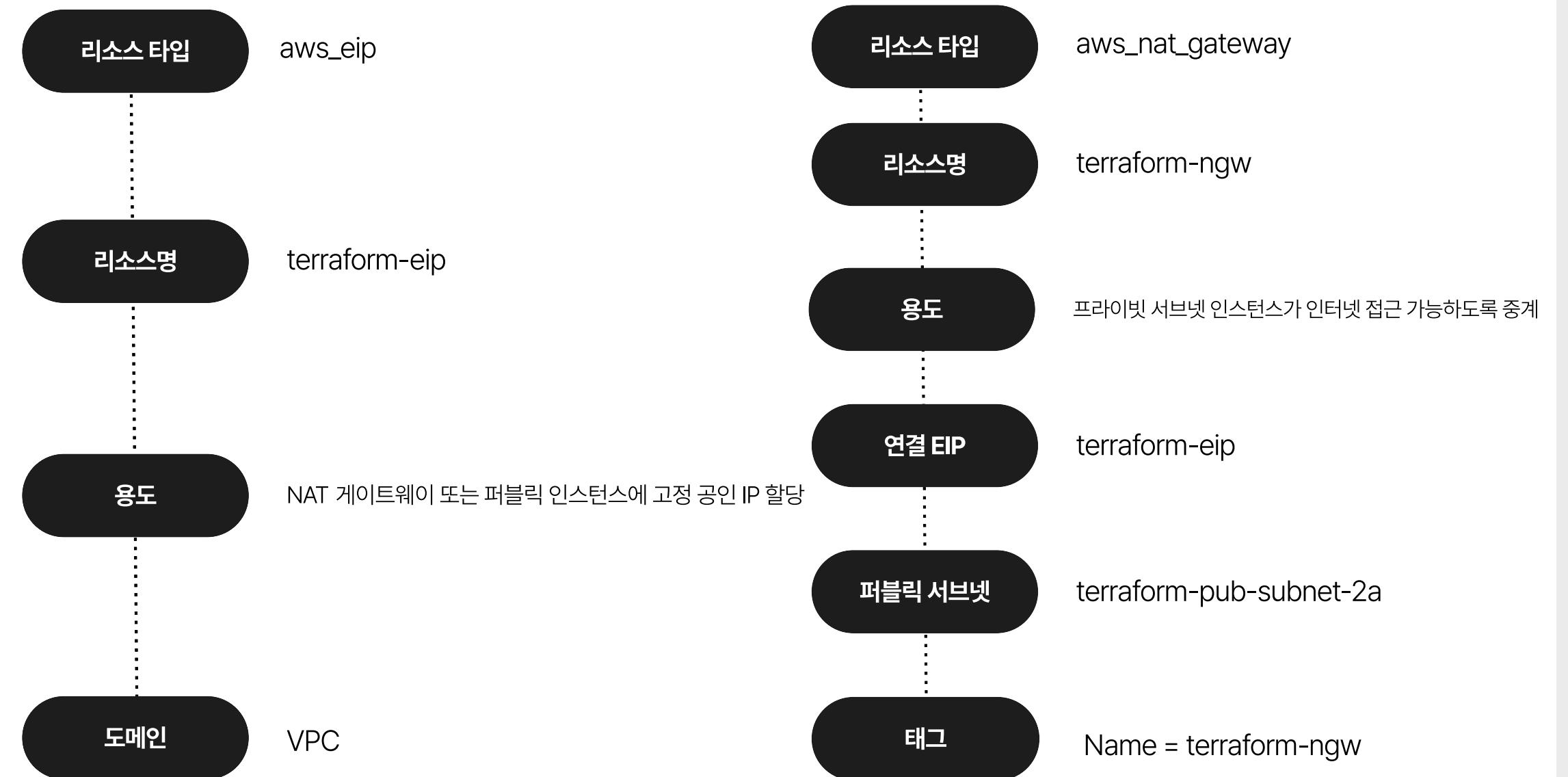
```
resource "aws_internet_gateway"  
"terraform-igw" {  
    vpc_id = aws_vpc.terraform-vpc.id  
    tags = {  
        "Name" = "terraform-igw"  
    }  
}
```



5.NGW.tf

```
# 탄력적 IP
resource "aws_eip" "terraform-eip" {
  domain = "vpc"
}

# NAT 게이트웨이
resource "aws_nat_gateway" "terraform-
ngw" {
  allocation_id = aws_eip.terraform-eip.id
  subnet_id    = aws_subnet.terraform-
pub-subnet-2a.id
  tags = {
    "Name" = "terraform-ngw"
  }
}
```



6. Routing Table.tf(Public)

```
# 라우팅 테이블
resource "aws_route_table" "terraform-pub-rt" {
  vpc_id = aws_vpc.terraform-vpc.id

  tags = {
    "Name" = "terraform-pub-rt"
  }
}

resource "aws_route_table" "terraform-pri-rt" {
  vpc_id = aws_vpc.terraform-vpc.id

  tags = {
    "Name" = "terraform-pri-rt"
  }
}

# 라우팅
resource "aws_route" "terraform-pub-rt" {
  route_table_id      = aws_route_table.terraform-pub-rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id          = aws_internet_gateway.terraform-igw.id
}

resource "aws_route" "terraform-pri-rt" {
  route_table_id      = aws_route_table.terraform-pri-rt.id
  destination_cidr_block = "0.0.0.0/0"
  nat_gateway_id       = aws_nat_gateway.terraform-ngw.id
}
```



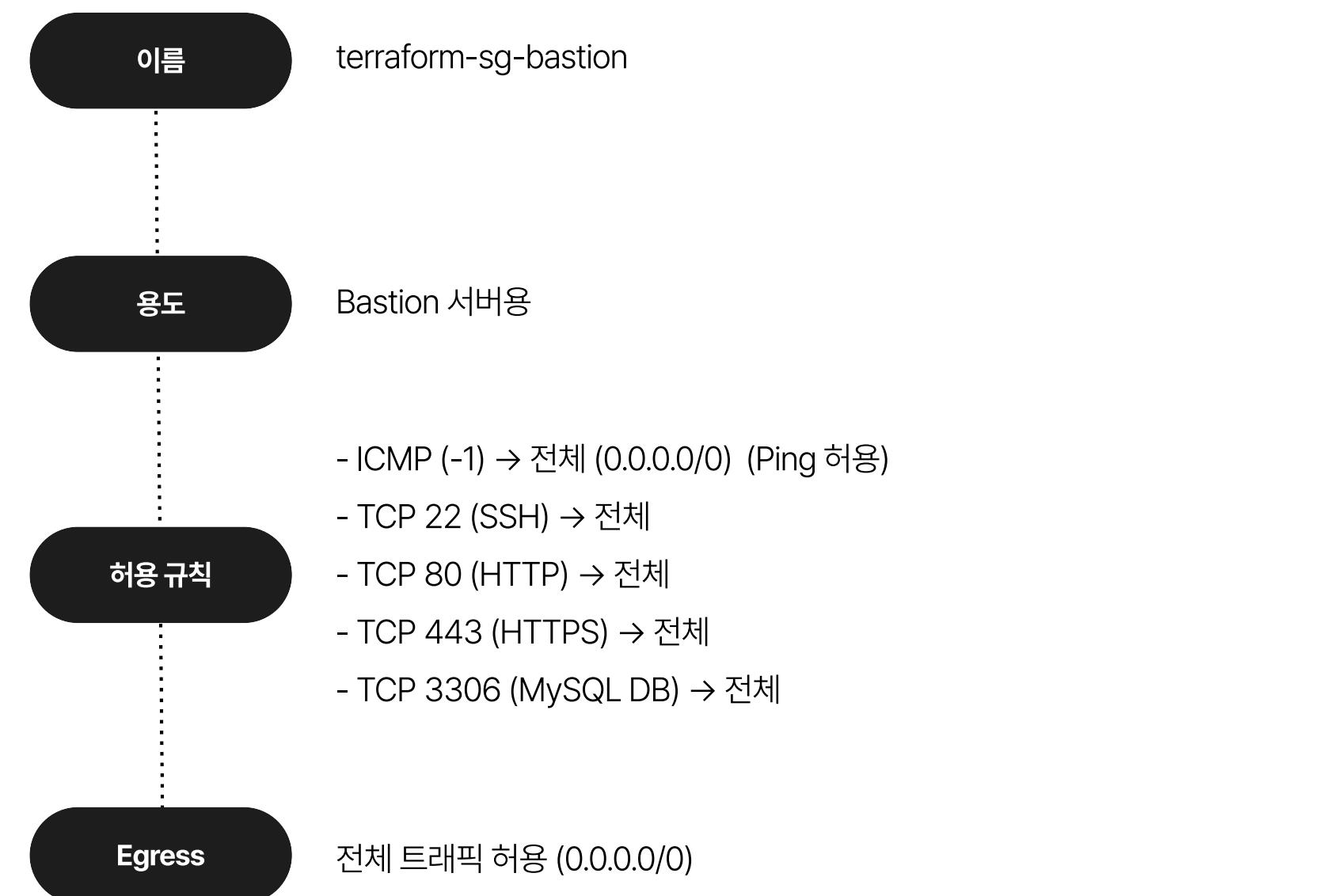
7. Security Group.tf(Bastion SG)

```
# Bastion SG
resource "aws_security_group" "terraform-sg-bastion" {
  name      = "terraform-sg-bastion"
  description = "for Bastion Server"
  vpc_id    = aws_vpc.terraform-vpc.id
  tags = {
    "Name" = "terraform-sg-bastion"
  }

  ingress {
    ingress {
      from_port  = -1
      to_port    = -1
      protocol   = "icmp"
      cidr_blocks = ["0.0.0.0/0"]
      description = "for ping"
    }
    ingress {
      from_port  = 80
      to_port    = 80
      protocol   = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
      description = "for HTTP"
    }
    ingress {
      from_port  = 3306
      to_port    = 3306
      protocol   = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
      description = "for DB"
    }
  }

  ingress {
    from_port  = 22
    to_port    = 22
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
    description = "for SSH"
  }
}

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



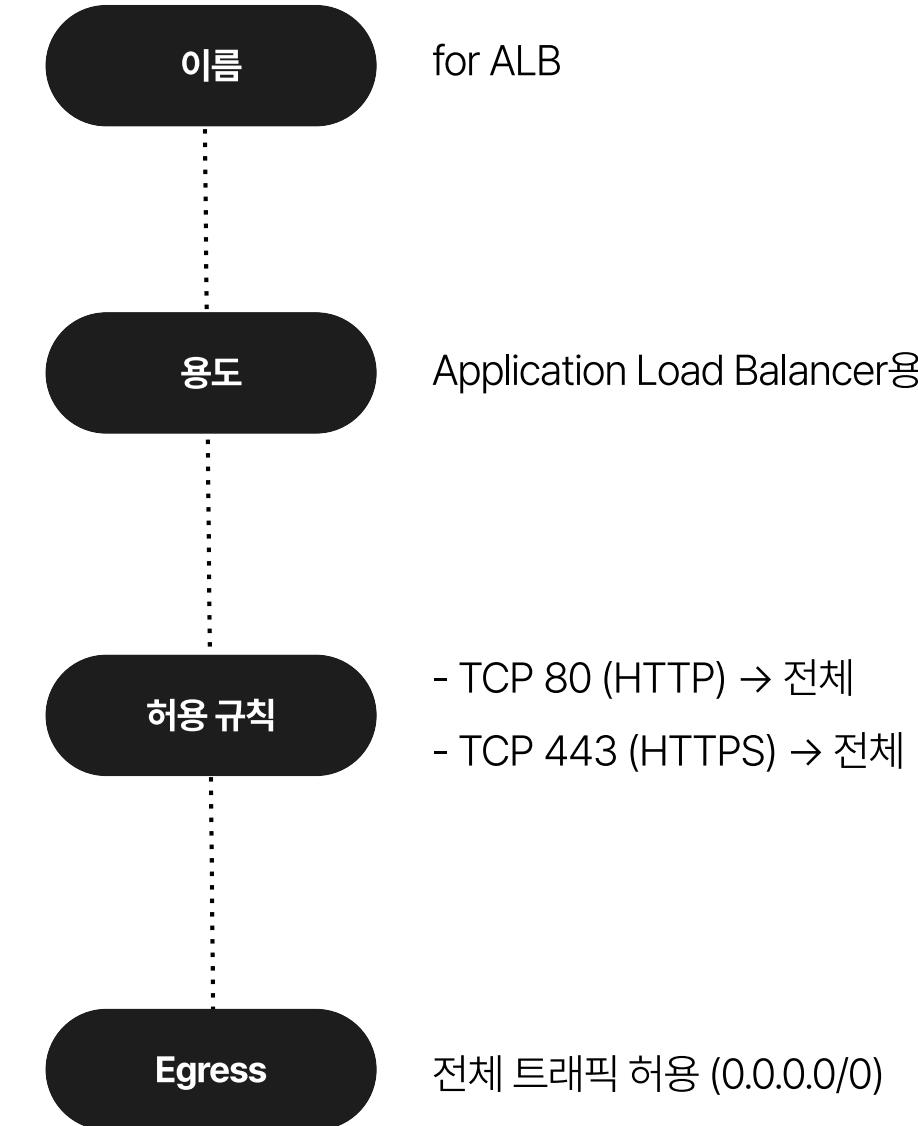
7. Security Group.tf(ALB SG)

```
# ALB SG
resource "aws_security_group" "terraform-sg-alb" {
    name  = "for ALB"
    vpc_id = aws_vpc.terraform-vpc.id

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

    ingress {
        from_port  = 443
        to_port    = 443
        protocol   = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
        description = "for HTTPS"
    }

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



7. Security Group.tf(RDS SG)

```
# RDS SG
resource "aws_security_group" "terraform-sg-rds" {
    name  = "for RDS"
    vpc_id = aws_vpc.terraform-vpc.id

    ingress {
        from_port  = 3306
        to_port    = 3306
        protocol   = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
        description = "for DB"
    }

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



8. ec2.tf

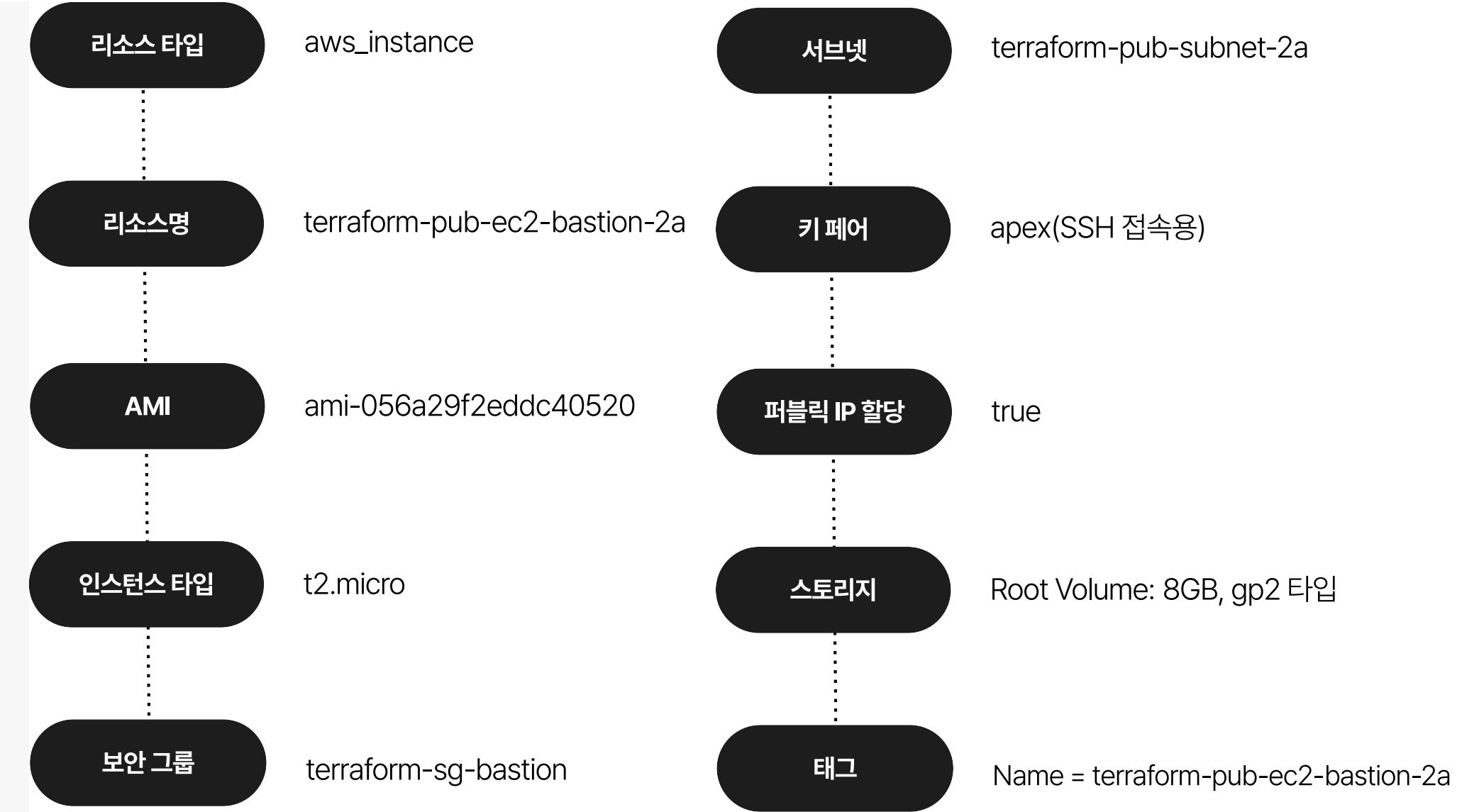
```

resource "aws_instance" "terraform-pub-ec2-bastion-2a" {
  ami           = "ami-056a29f2eddc40520"
  instance_type      = "t2.micro"
  vpc_security_group_ids  = [aws_security_group.terraform-sg-
    bastion.id]
  subnet_id       = aws_subnet.terraform-pub-subnet-2a.id
  key_name        = "apex"
  associate_public_ip_address = true

  root_block_device {
    volume_size = "8"
    volume_type = "gp2"
    tags = {
      "Name" = "terraform-pub-ec2-bastion-2a"
    }
  }

  tags = {
    "Name" = "terraform-pub-ec2-bastion-2a"
  }
}

```



9. key_pair.tf

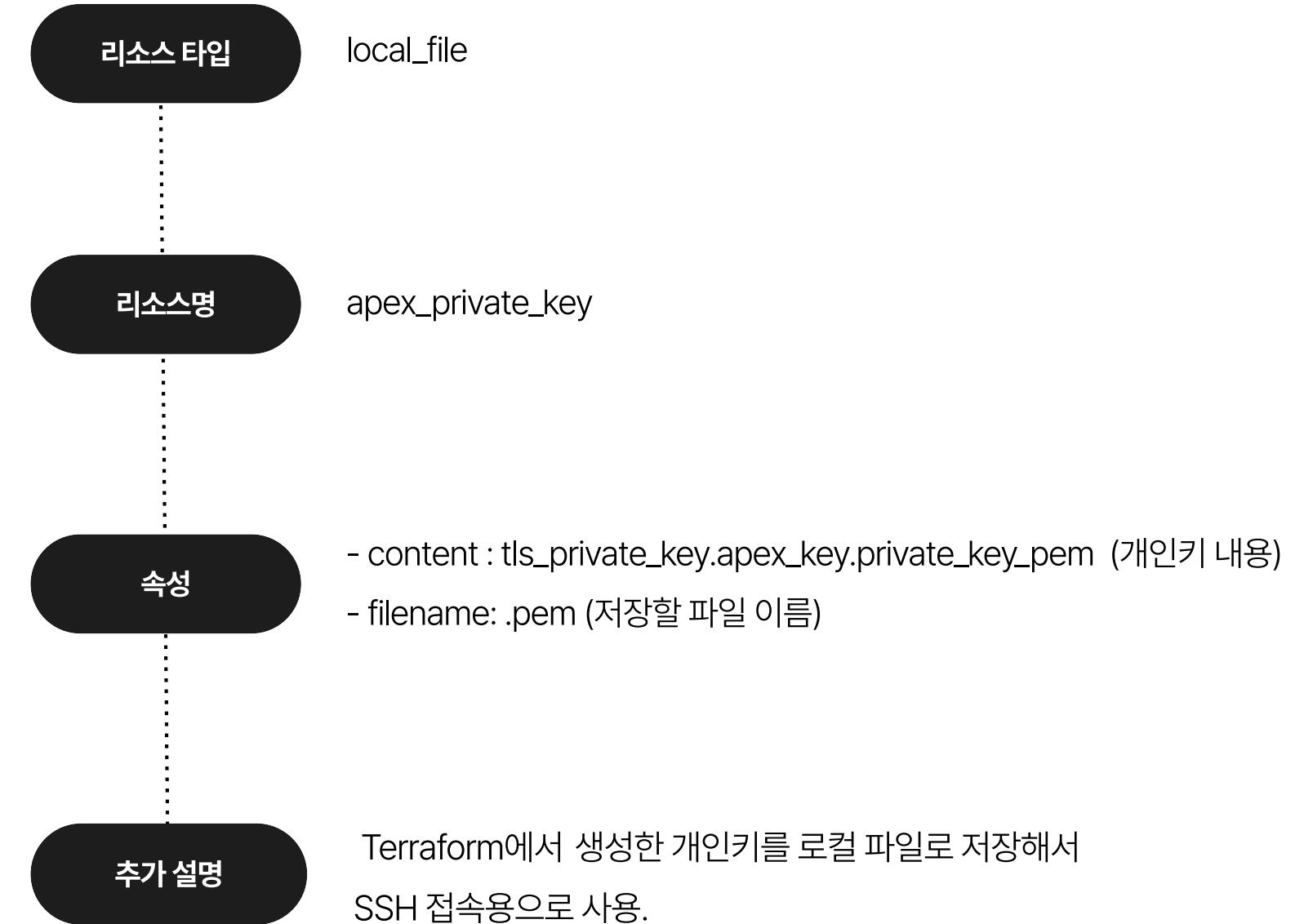
```
resource "tls_private_key" "apex_key" {
  algorithm = "RSA"
  rsa_bits  = 2048
}

resource "aws_key_pair" "apex_aws_key" {
  key_name   = "apex"
  public_key =
  tls_private_key.apex_key.public_key_openssh
}
```



9. key_pair.tf (local_file)

```
resource "local_file" "apex_private_key" {  
    content = tls_private_key.apex_key.private_key_pem  
    filename = ".pem"  
}
```



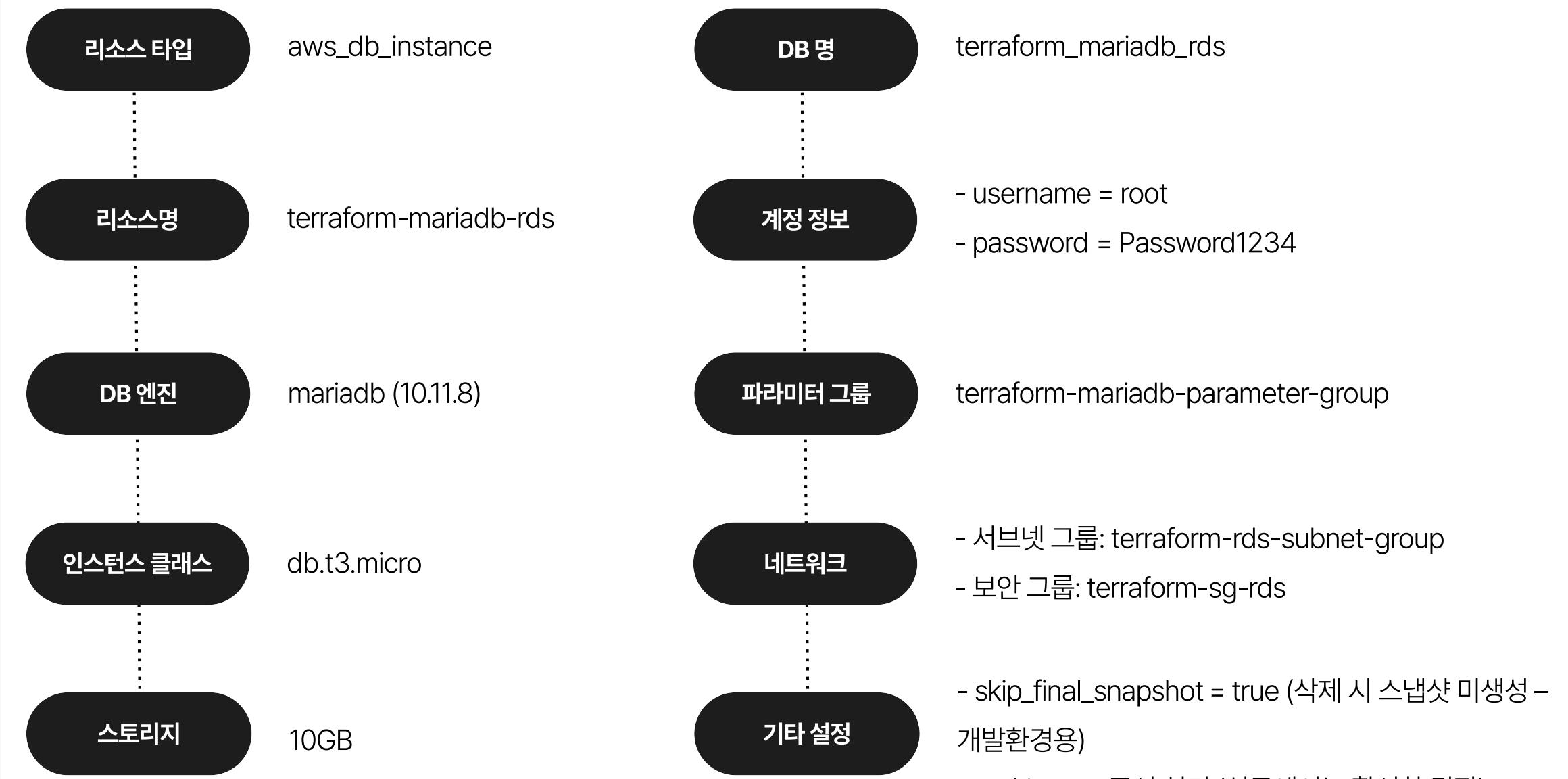
10. rds.tf(RDS Instance {MariaDB})

```

resource "aws_db_instance" "terraform-mariadb-rds" {
  identifier_prefix = "terraform-mariadb-rds"
  allocated_storage = 10
  engine           = "mariadb"
  engine_version   = "10.11.8"
  instance_class   = "db.t3.micro"
  db_name          = "terraform_mariadb_rds" # 영문자로 시작해야 하며 영문자와 숫자, _만 가능
  username          = "root"
  password          = "Password1234" # 8자 이상, 영문 대소문자, 숫자 및 특수 문자를 혼합하여 사용
  parameter_group_name = "terraform-mariadb-parameter-group"
  skip_final_snapshot = true
  #multi_az        = true
  db_subnet_group_name =
  aws_db_subnet_group.terraform-rds-subnet-group.name
  vpc_security_group_ids = [aws_security_group.terraform-sg-rds.id]

  tags = {
    Name = "terraform-mariadb-rds"
  }
}

```

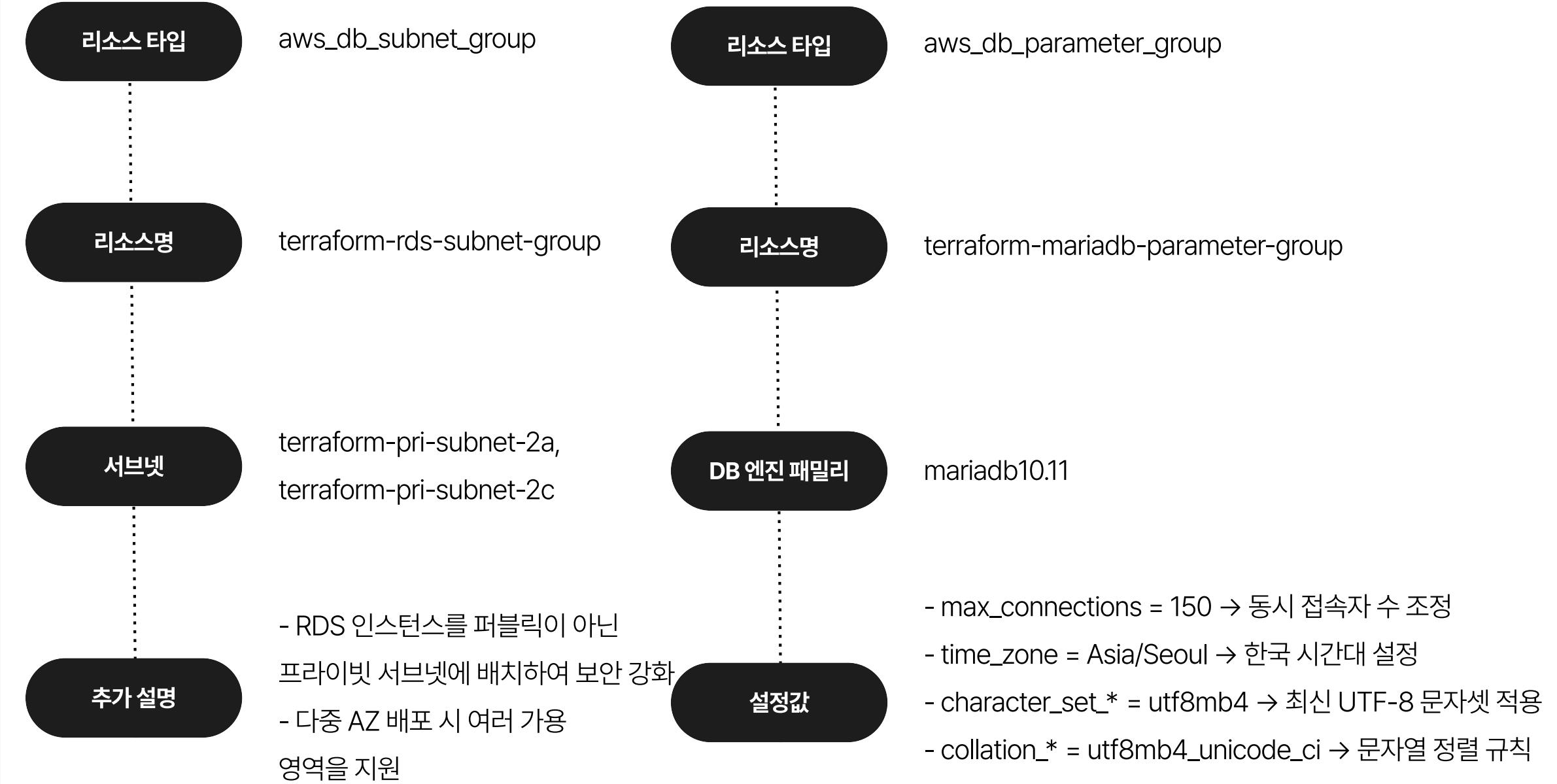


10. rds.tf

```

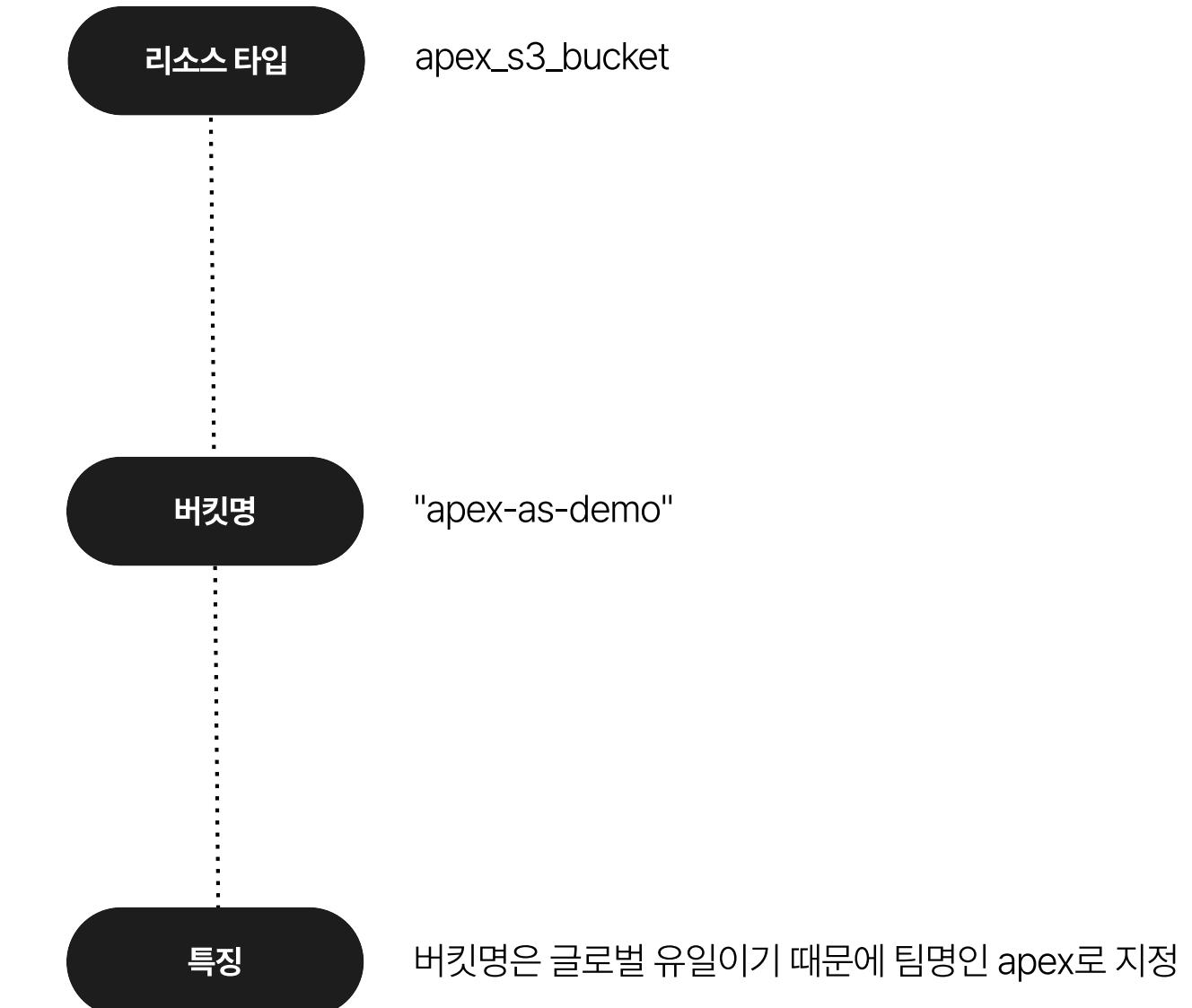
resource "aws_db_subnet_group" "terraform-rds-subnet-group" {
  name      = "terraform-rds-subnet-group"
  subnet_ids = [aws_subnet.terraform-pri-subnet-2a.id,
    aws_subnet.terraform-pri-subnet-2c.id]
  tags      = {Name = "terraform-rds-subnet-group"}
}
resource "aws_db_parameter_group" "terraform-mariadb-parameter-group" {
  name      = "terraform-mariadb-parameter-group"
  family    = "mariadb10.11" # 사용 중인 MariaDB 버전에 맞게 조정
  description = "Custom parameter group for MariaDB"
  parameter {
    name   = "max_connections"
    value  = "150"
  }
  parameter {
    name   = "time_zone"
    value  = "Asia/Seoul"
  }
  parameter {
    name   = "character_set_client"
    value  = "utf8mb4"
  }
  parameter {
    name   = "character_set_connection"
    value  = "utf8mb4"
  }
  parameter {
    name   = "character_set_database"
    value  = "utf8mb4"
  }
  parameter {
    name   = "character_set_filesystem"
    value  = "utf8mb4"
  }
  parameter {
    name   = "character_set_results"
    value  = "utf8mb4"
  }
  parameter {
    name   = "character_set_server"
    value  = "utf8mb4"
  }
  parameter {
    name   = "collation_connection"
    value  = "utf8mb4_unicode_ci"
  }
  parameter {
    name   = "collation_server"
    value  = "utf8mb4_unicode_ci"
  }
  parameter {
    name   = "binlog_format"
    value  = "ROW"
  }
}

```



11. s3.tf

```
resource "apex_s3_bucket" "apex-as-demo" {  
  bucket = "apex-as-demo"  
}
```



12. acm.tf

```

resource "aws_acm_certificate" "cert" {
  domain_name      = "*.idokebi.store"
  validation_method = "DNS"

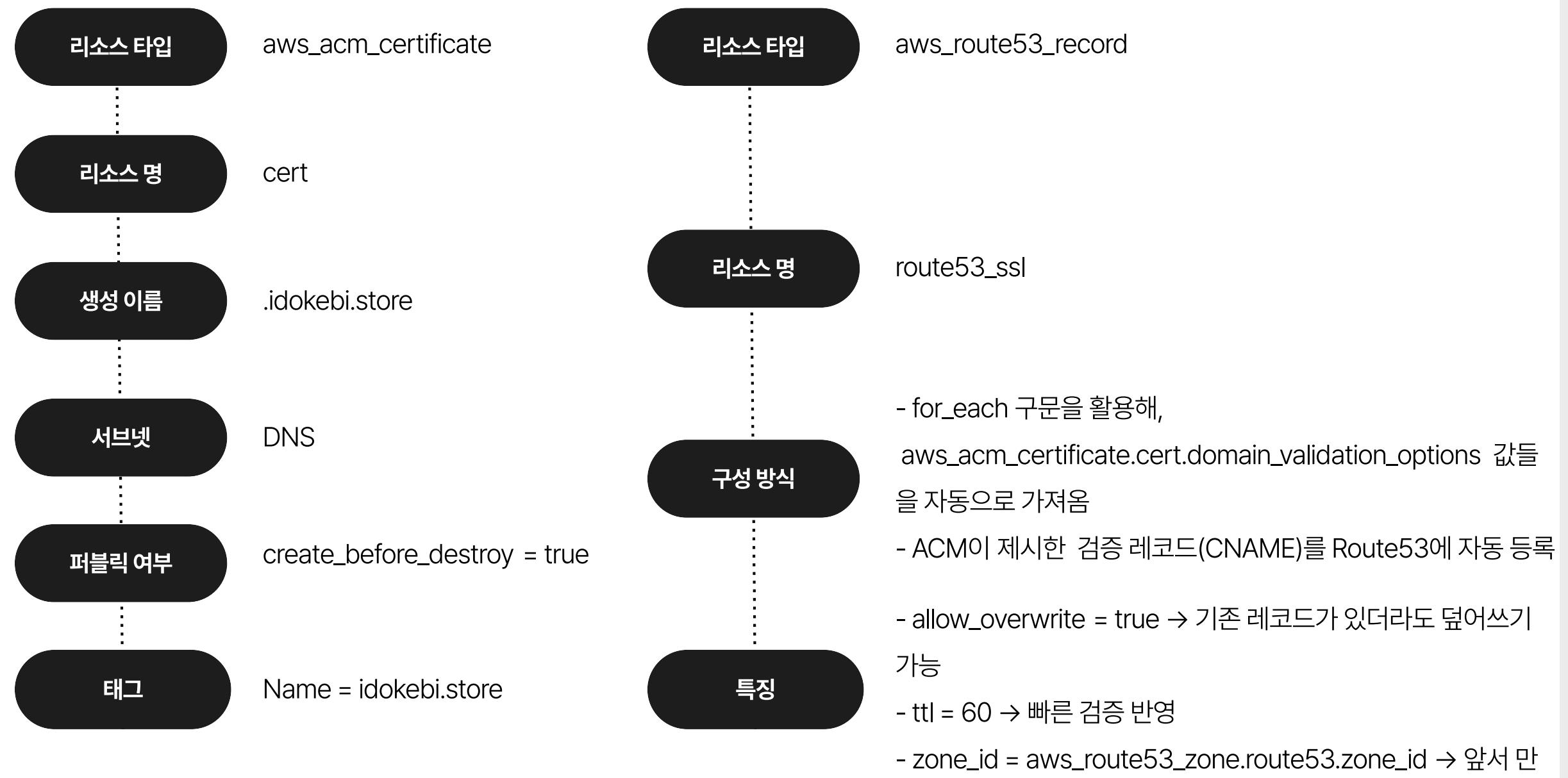
  tags = {
    Name = "idokebi.store"
  }

  lifecycle {
    create_before_destroy = true
  }
}

resource "aws_route53_record" "route53_ssl" {
  for_each = {
    for dvo in aws_acm_certificate.cert.domain_validation_options :
    dvo.domain_name => {
      name  = dvo.resource_record_name
      record = dvo.resource_record_value
      type   = dvo.resource_record_type
    }
  }

  allow_overwrite = true
  name           = each.value.name
  records        = [each.value.record]
  ttl            = 60
  type           = each.value.type
  zone_id        = aws_route53_zone.route53.zone_id
}

```



13. route53.tf

```
resource "aws_route53_zone"  
"route53" {  
  name = "leoandres.store"  
}
```

리소스 타입

aws_route53_zone(AWS Route53에 호스팅 존 생성)

도메인 명

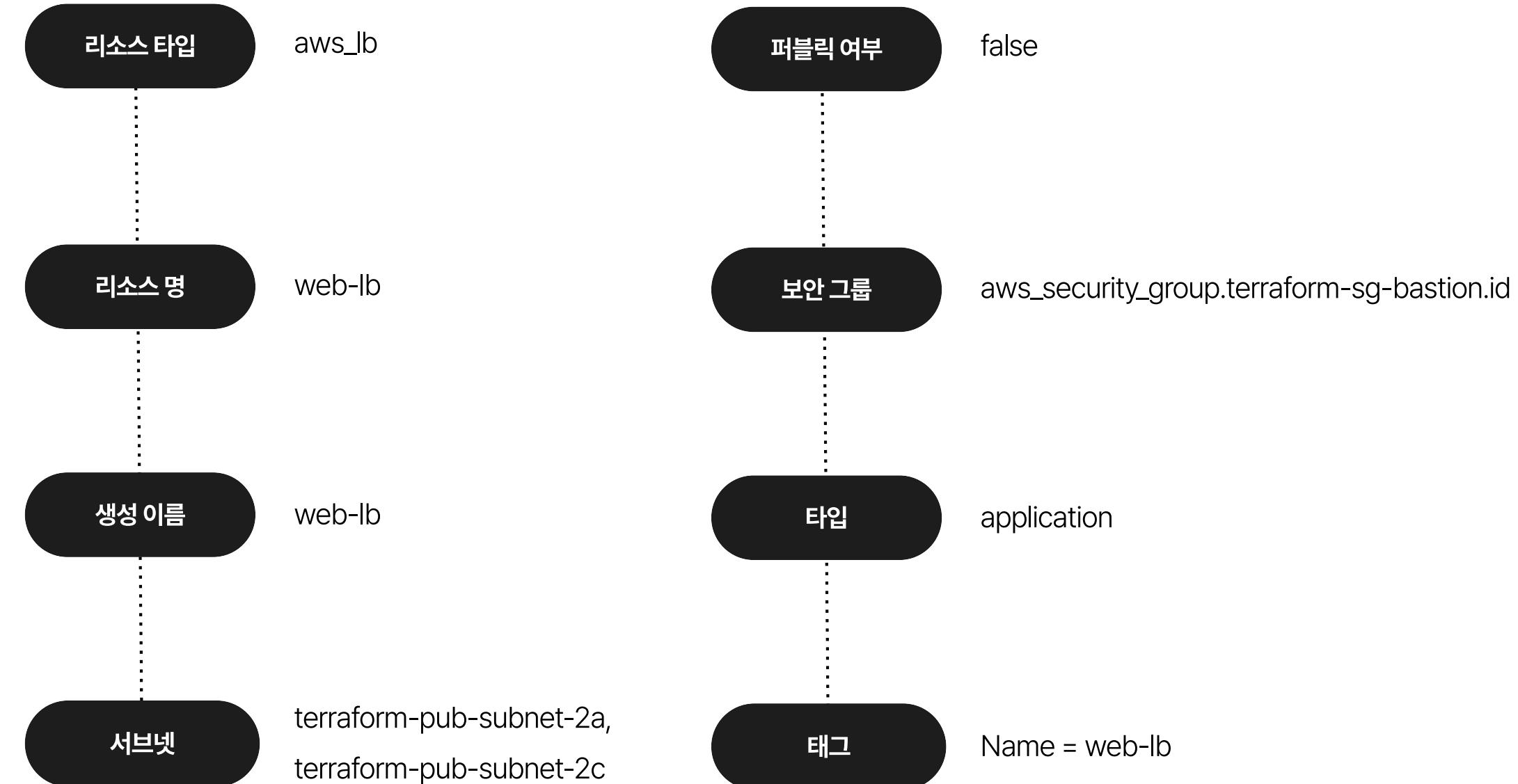
name = "idokebi.store"

리소스 명

route53

14. alb.tf(Application Load Balancer 생성)

```
resource "aws_lb" "web-lb" {
  name = "web-lb"
  subnets = [ aws_subnet.terraform-
    pub-subnet-2a.id,
  aws_subnet.terraform-pub-subnet-2c.id
]
  internal = false
  security_groups = [
    aws_security_group.terraform-sg-
    bastion.id ]
  load_balancer_type = "application"
  tags = {
    "Name" = "web-lb"
  }
}
```



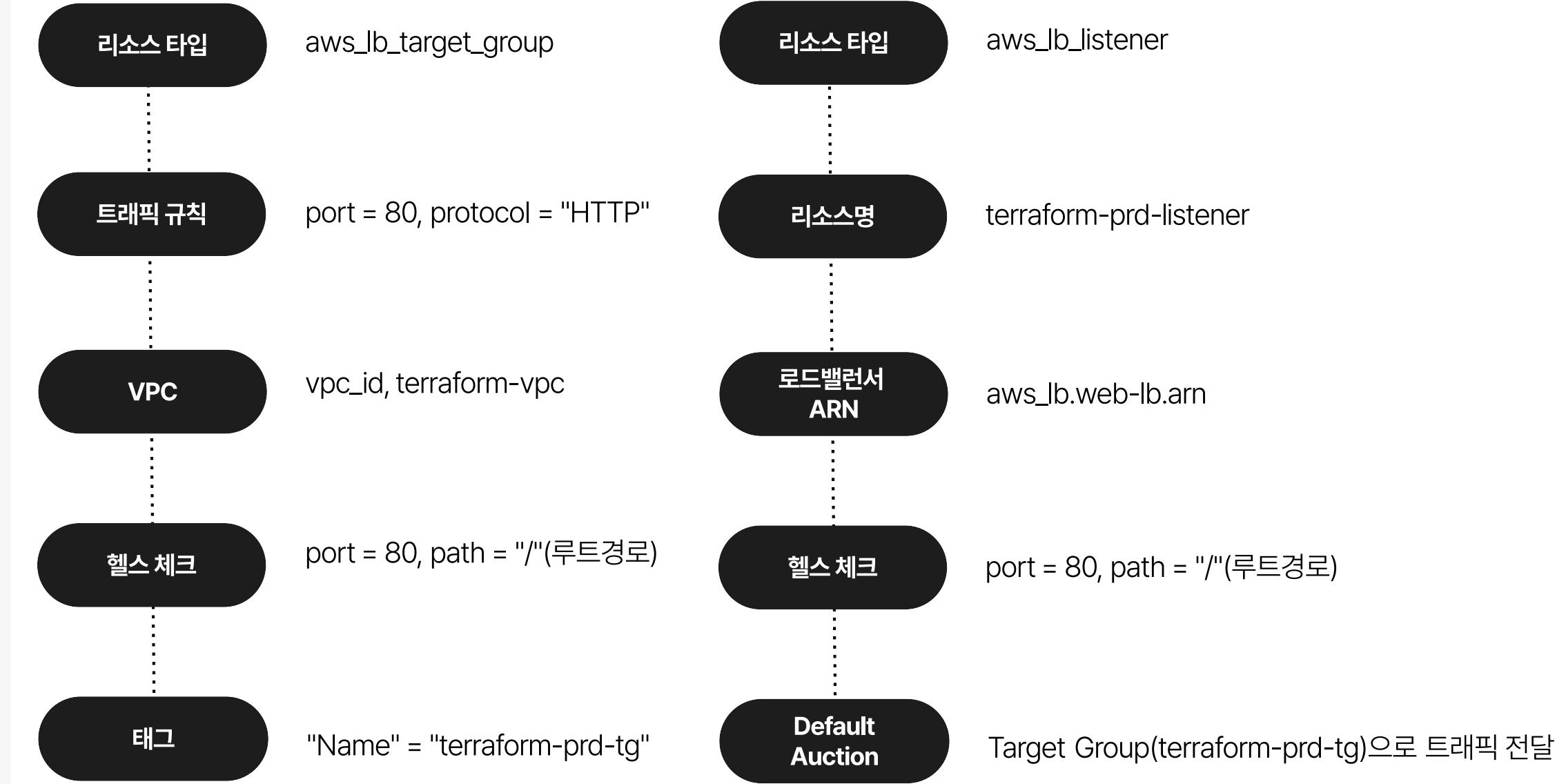
14. alb.tf

```

resource "aws_lb_target_group" "terraform-prd-tg" {
  name = "terraform-prd-tg"
  port = 80
  protocol = "HTTP"
  vpc_id = aws_vpc.terraform-vpc.id
  health_check {
    port = 80
    path = "/"
  }
  tags = {
    "Name" = "terraform-prd-tg"
  }
}

resource "aws_lb_listener" "terraform-prd-listener" {
  load_balancer_arn = aws_lb.web-lb.arn
  port = "80"
  protocol = "HTTP"
  default_action {
    target_group_arn = aws_lb_target_group.terraform-prd-tg.arn
    type = "forward"
  }
}

```

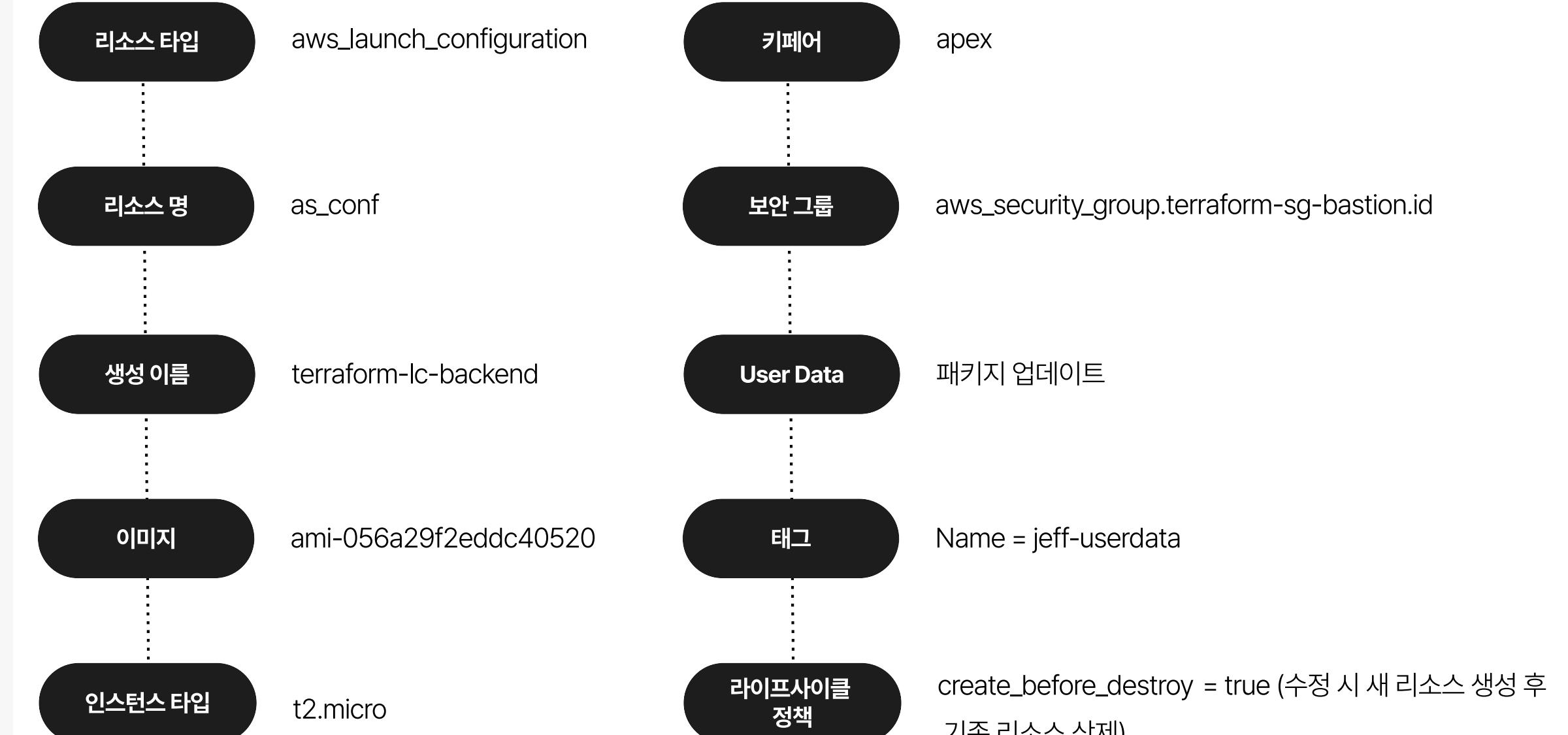


15. Auto.tf(Launch Configuration)

```

resource "aws_launch_configuration" "as_conf" {
  name_prefix = "terraform-lc-backend"
  image_id    = "ami-056a29f2eddc40520"
  instance_type = "t2.micro"
  key_name = "apex"
  user_data = <<EOF
#!/bin/bash
sudo apt update
sudo apt install -y nginx
EOT
tags = {
  Name = "jeff-userdata"
}
}
EOF
  security_groups = [aws_security_group.terraform-sg-bastion.id]
  lifecycle {
    create_before_destroy = true
  }
}

```



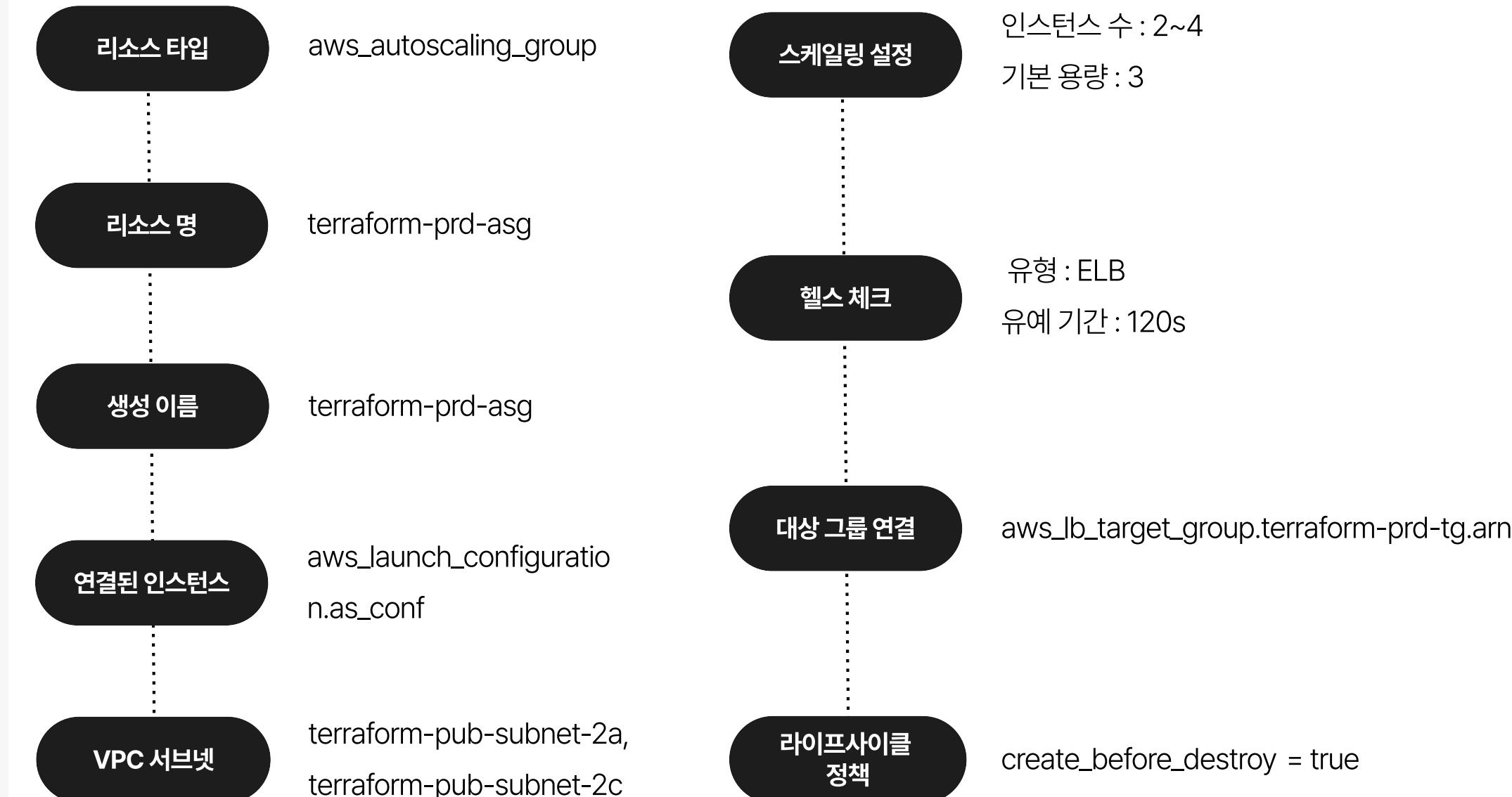
15. Auto.tf(Auto Scaling Group)

```

resource "aws_autoscaling_group" "terraform-prd-asg" {
    name          = "terraform-prd-asg"
    launch_configuration =
    aws_launch_configuration.as_conf.name
    vpc_zone_identifier =
    [aws_subnet.terraform-pub-subnet-2a.id,
    aws_subnet.terraform-pub-subnet-2c.id]

    min_size = 2
    max_size = 4
    desired_capacity = 3
    health_check_grace_period = 120
    health_check_type = "ELB"
    target_group_arns =
    [aws_lb_target_group.terraform-prd-tg.arn]
    lifecycle {
        create_before_destroy = true
    }
}

```



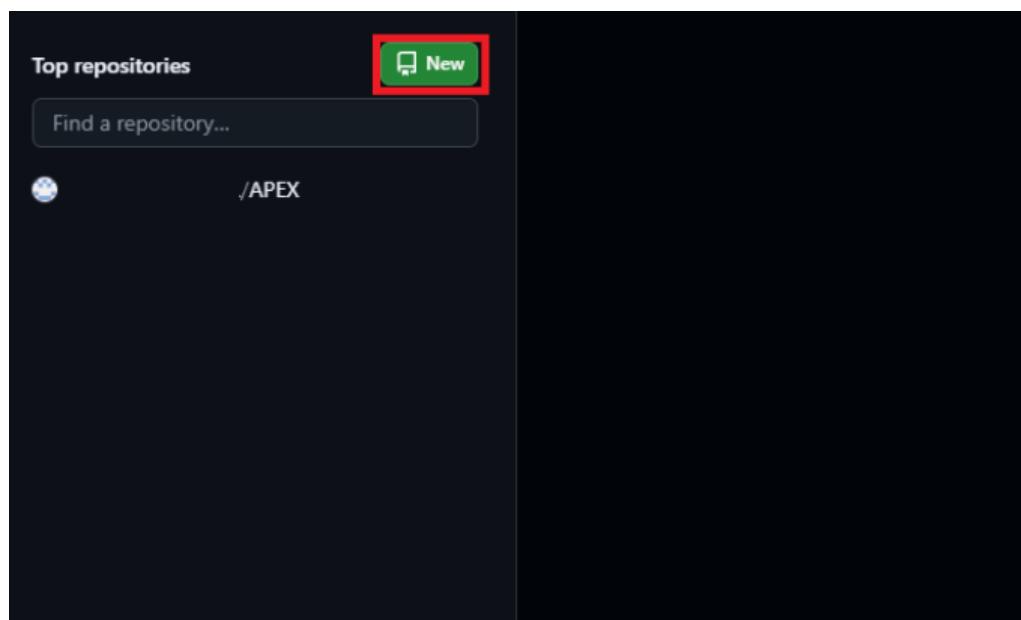


6. Github ,Terraform Cloud, AWS 연동

- GitHub Repository 만들기
- Terraform Cloud 초기 구성 & GitHub 연동
- Terraform Cloud & AWS 연동

기간 : 2023-07-10 ~ 2023-07-14

6-1. GitHub Repository 만들기



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1 General

Owner * / Repository name * **APEX-Project**

Great repository names are short and memorable. How about [special-octo-robot](#)?

Description
0 / 350 characters

2 Configuration

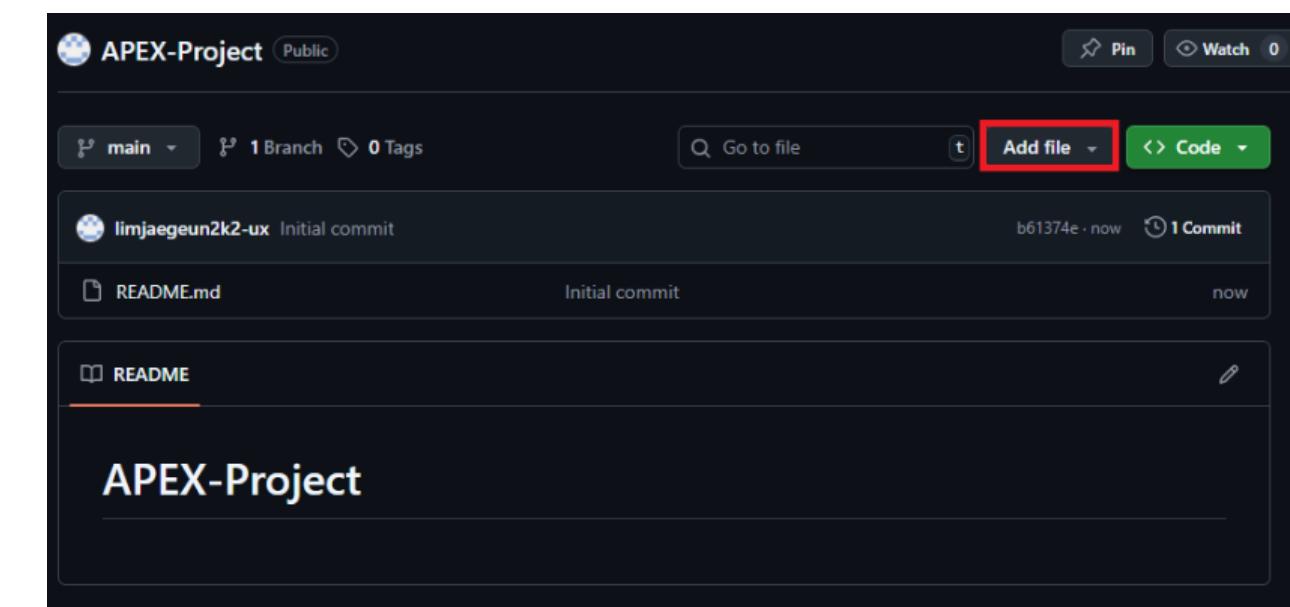
Choose visibility * **Public**

Add README

Add .gitignore

Add license

Create repository

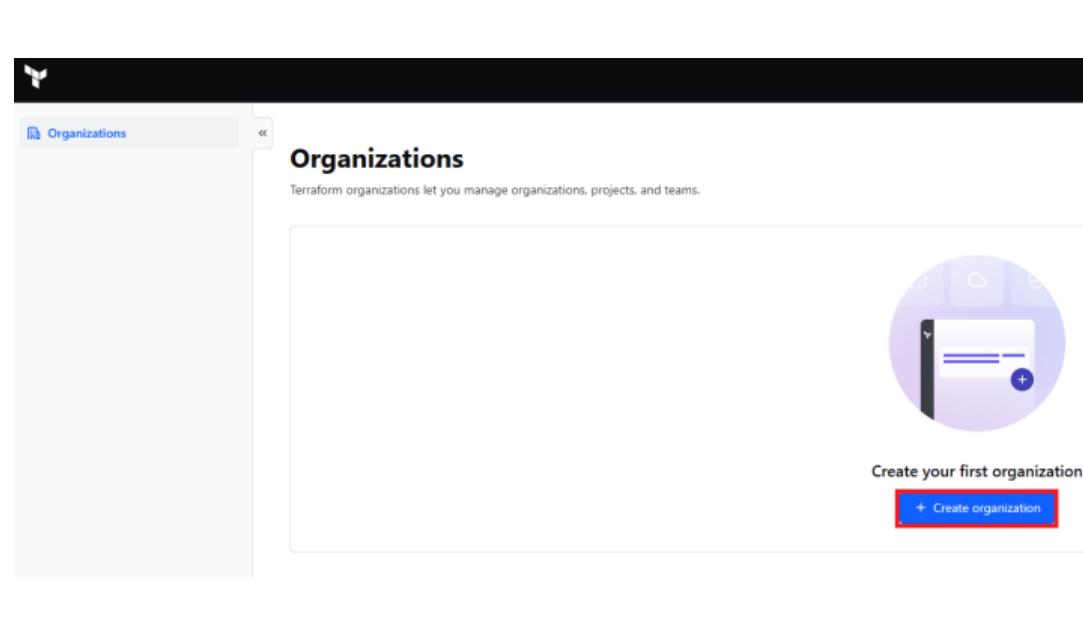


GitHub Repository 만들기

Repository 생성

- Repository 이름 설정
- README file 추가

6-2.Terraform Cloud 초기 구성 & GitHub 연동



Organizations / New

Create a new organization

Organizations are privately shared spaces for teams to collaborate on infrastructure. [Learn more](#) about organizations in HCP Terraform.

How will you use this organization? Not editable after creation

Business

Best for company or institution use (even if this is just for testing for now).

Ensures proper handling of taxes.

Ensures business continuity by tying this organization to the business rather than individuals.

Personal

Best for individual use for your own personal projects that have no affiliation to a business.

Terraform organization name

• Must be unique.
• May contain valid characters including ASCII letters, numbers, spaces, as well as dashes (-), and underscores (_).

APEX-Project

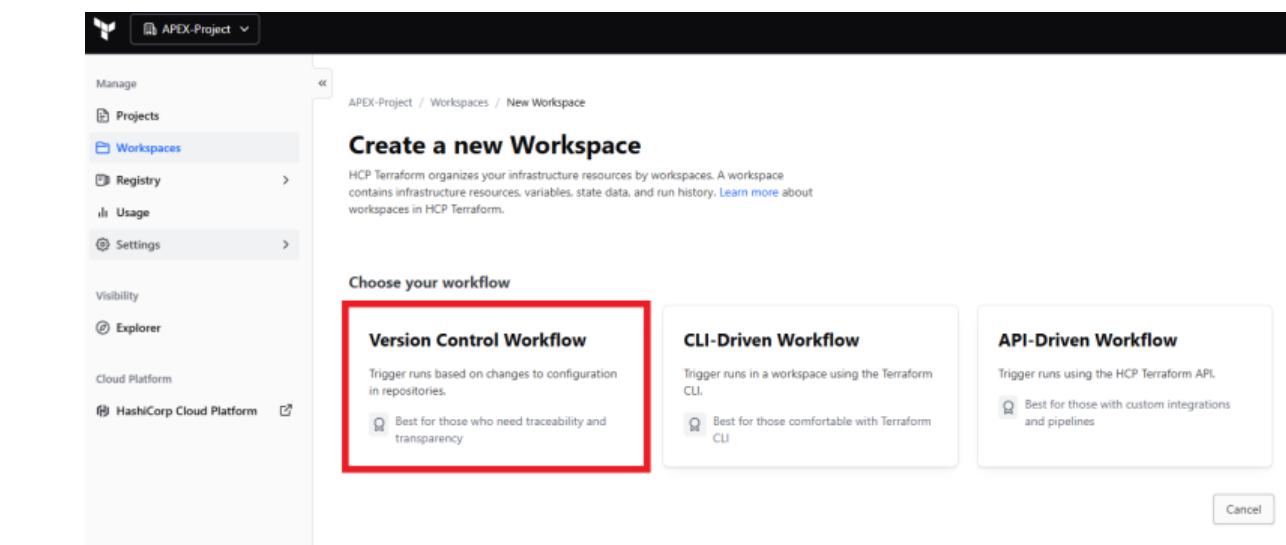
Email address

The organization email is used for any future notifications, such as billing alerts, and the organization avatar, via [gravatar.com](#).

Create organization Cancel

Organizations 생성

이름과 Email 주소를 추가하여
Organization 생성



GitHub와 연동할 WorkSpace 생성
Version Control Workflow 선택

6-2.Terraform Cloud 초기 구성 & GitHub 연동

The screenshot shows the 'Create a new Workspace' wizard in the Terraform Cloud interface. It consists of three main sections:

- Step 1: Choose Type** (Completed): A brief introduction to workspaces.
- Step 2: Connect to VCS** (In Progress): A step to connect the workspace to a version control provider. It shows options for GitHub (selected) and GitHub App (highlighted with a red box). Other options include 'Connect to a different VCS'.
- Step 3: Choose a repository** (In Progress): A step to choose a repository from a list. It shows a list of repositories under 'APEC-Project' with one repository selected. A search bar at the bottom allows entering a repository ID.

Create a new Workspace 생성

GitHub 선택

연동할 GitHub Repository 선택

6-3. Terraform Cloud & AWS 연동

The screenshot shows the 'Explorer' interface of HashiCorp Cloud Platform. A new workspace named 'APEX-Project' is being created. The 'Workspace Name' field contains 'APEX-Project' and 'Default Project'. The 'Key' column of the 'Workspace variables (0)' table has a red box around it, indicating where variables will be defined.

The screenshot shows the 'Overview' page for the 'APEX-Project' workspace. It displays a message: 'Configuration uploaded successfully'. Below this, there are two variable entries: 'AWS_ACCESS_KEY_ID' and 'AWS_SECRET_ACCESS_KEY'. Both entries have their 'Key' and 'Value' fields highlighted with red boxes. The 'HCL' checkbox is checked for both, and the 'Sensitive' checkbox is unchecked.

The screenshot shows the 'General' tab of the workspace settings. Under the 'Auto-apply' section, two checkboxes are checked: 'Auto-apply API, UI, & VCS runs' and 'Auto-apply run triggers'. The 'Terraform Version' dropdown is set to '>> 1.13.0 latest (1.13.1)'. The 'Save settings' button at the bottom right is highlighted with a red box.

연동할 GitHub Work space 선택
variable 생성

AWS와 연동하기 위한 AWS Access Key를 Terraform Cloud에 variable로 등록

자동 apply 활성화

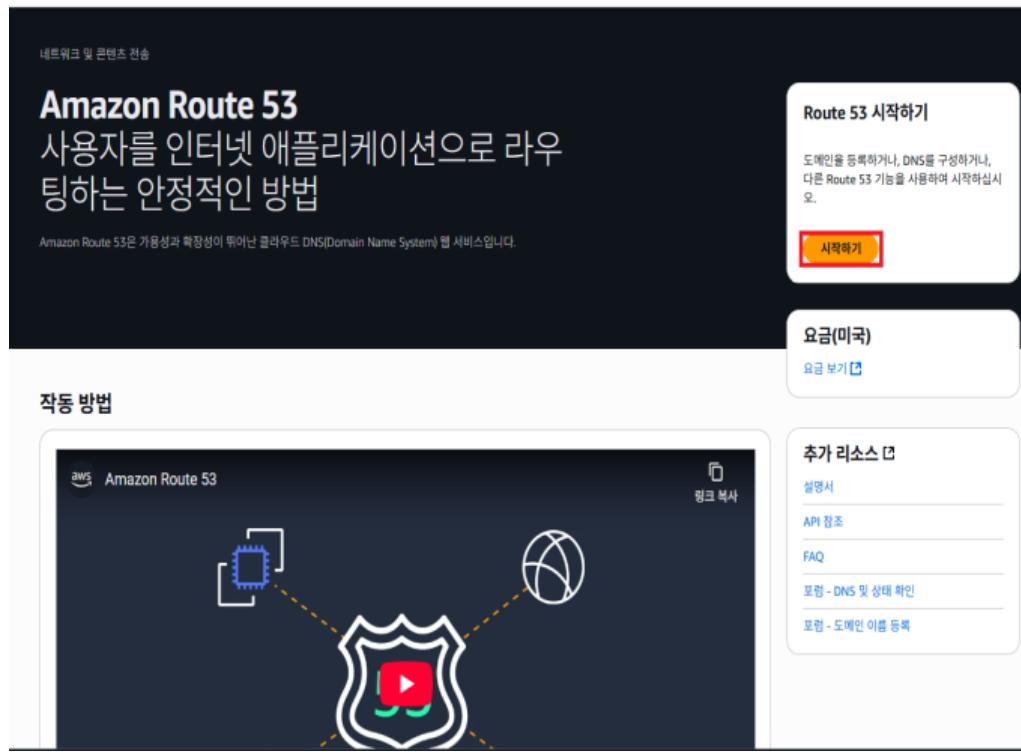


7. Route53 도메인 등록

- 호스트 영역 생성
- 도메인 등록업체 NS 레코드 입
- 가비아에서 확인

기간: XX-XX-XX ~ XX-XX-XX | 가이드: 102

7-1. 호스트 영역 생성



Route 53 > 시작하기

시작하기 정보

시작 지점 선택

- 도메인 등록
- 도메인 이전
- 호스팅 영역 생성

호스팅 영역은 Route 53에 example.com과 같은 도메인의 DNS 쿼리에 응답하는 방법을 알려줍니다.

Route 53 > 호스팅 영역 > 호스팅 영역 생성

호스팅 영역 생성 정보

호스팅 영역 구성

호스팅 영역은 example.com 같은 도메인과 관련 하위 도메인에 대한 트래픽을 리우팅하는 방식에 대한 정보를 포함하는 컨테이너입니다.

도메인 이름: leoandres.store
트래픽을 리우팅할 도메인의 이름입니다.

설명 - 선택 사용: 정보
이 글을 사용하면 이름이 동일한 호스팅 영역을 구별할 수 있습니다.
호스팅 영역이 사용되는 경우...

설명은 최대 256자입니다. 0/256

유형: 정보

유형은 인터넷 또는 Amazon VPC에서 트래픽을 리우팅할지 여부를 가리킵니다.

- 마법의 호스팅 영역
- 프라이빗 호스팅 영역

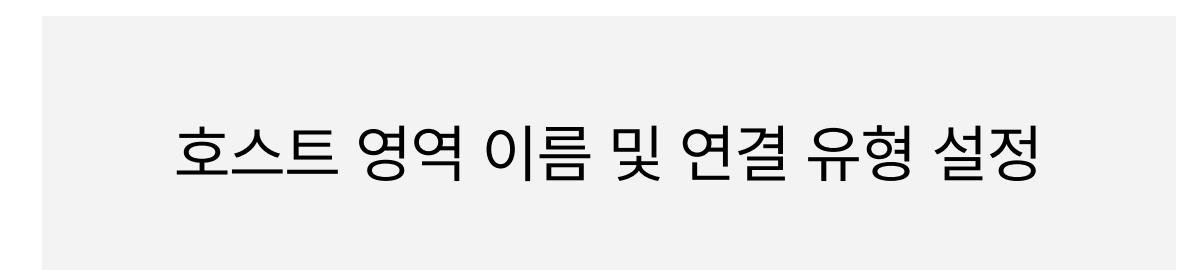
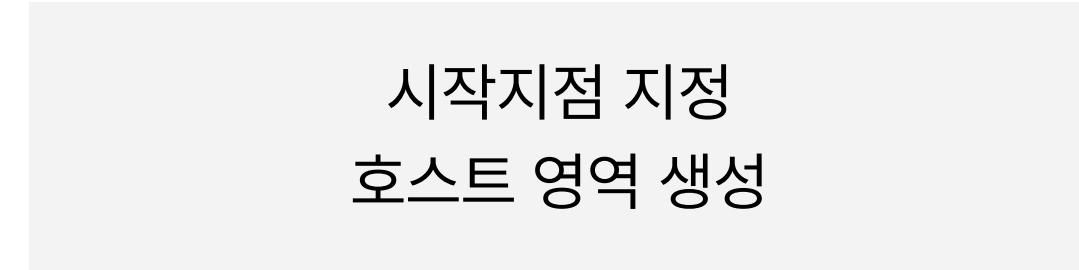
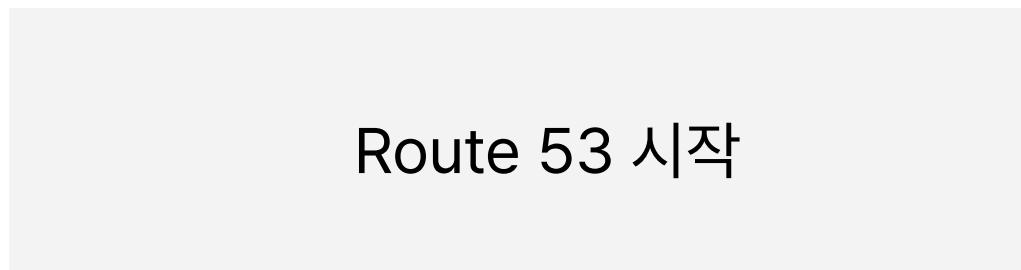
마법의 호스팅 영역은 인터넷에서 트래픽을 리우팅하는 방식을 결정합니다.

프라이빗 호스팅 영역은 Amazon VPC 내에서 트래픽을 리우팅하는 방식을 결정합니다.

태그: 정보

호스팅 영역에 태그를 적용하면 호스팅 영역을 쉽게 구분하고 식별할 수 있습니다.
리소스와 연결된 태그가 없습니다.

태그 추가
최대 50개의 태그를 더 추가할 수 있습니다.



7-2. 도메인 등록업체 NS 레코드 입력

퍼블릭 **leoandres.store** 정보

영역 삭제 레코드 테스트 쿼리 로깅 구성

▶ 호스팅 영역 세부 정보 호스팅 영역 편집

레코드(2) DNSSEC 서명 호스팅 영역 태그(0)

레코드 (2) 정보

Automatic 모드는 최상의 필터 결과에 최적화된 현재 검색 동작입니다. [모드를 변경하려면 설정\(settings\)으로 이동합니다.](#)

속성 또는 값을 기준으로 레코드 필터링 유형 라우팅 정책 별칭 값/트래픽 라우팅 대상 TTL(초)

레코드 이름	유형	라우팅...	차별화...	별칭	값/트래픽 라우팅 대상	TTL(초)
leoandres.store	NS	단순	-	아니요	ns-534.awsdns-02.net. ns-382.awsdns-47.com. ns-1723.awsdns-23.co.uk. ns-1362.awsdns-42.org.	172800
leoandres.store	SOA	단순	-	아니요	ns-534.awsdns-02.net. awsd...	900

네임서버 설정

leoandres.store

ns-534.awsdns-02.net
ns-382.awsdns-47.com
ns-1723.awsdns-23.co.uk
ns-1362.awsdns-42.org

7-3. 가비아에서 확인

Gabia 도메인 관리

김민호님 로그아웃

전체 도메인 01개

My 가비아 | 1:1 문의

홈 > 전체도메인 > 도메인 상세 페이지

메뉴얼

전체 도메인 < leoandres.store 등록 확인증 인증 코드

도메인 정보

도메인 leoandres.store

등록일 2025-08-22

만기일 2026-08-22 (남은 기간: 358일)

만기일 맞춤 도메인 연장 연장 알림

도메인 카테고리

소유자 설정 소유권 이전 관리자 설정

고객센터 네임서버

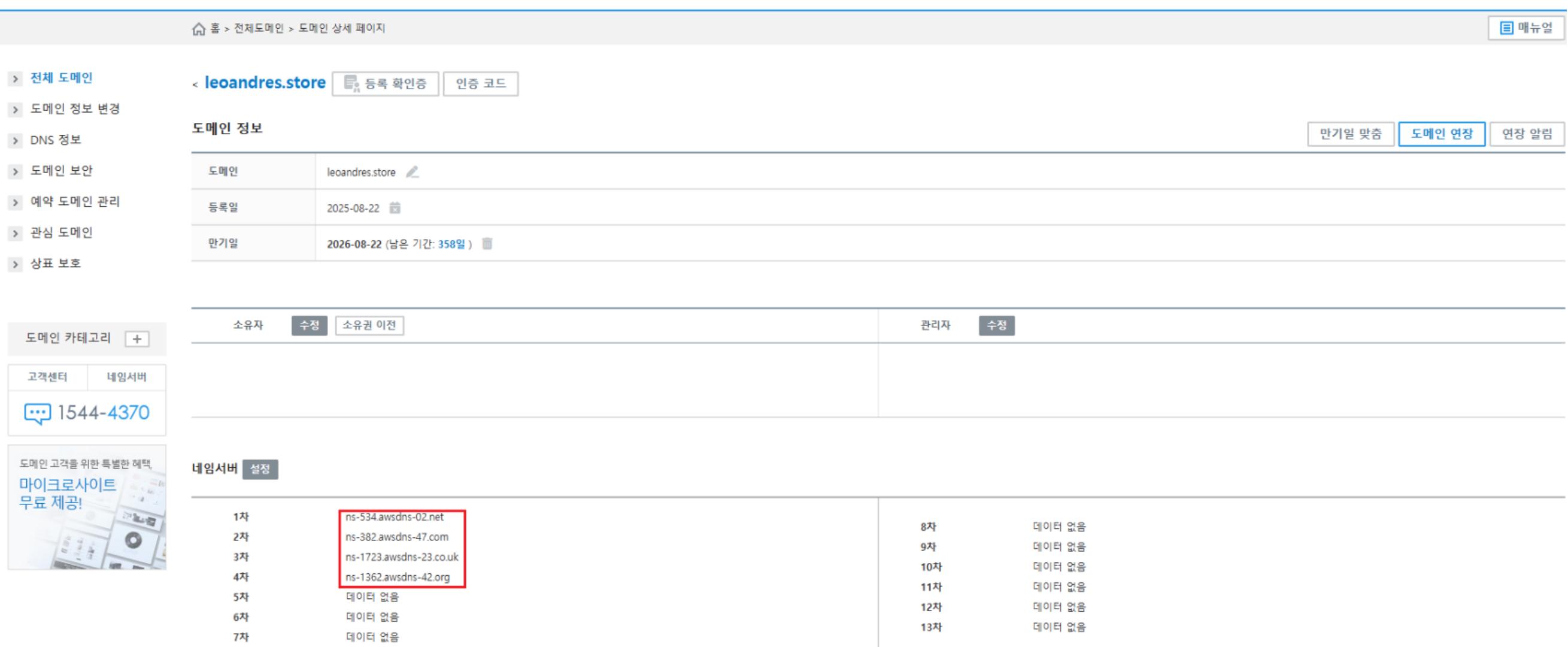
1544-4370

도메인 고객을 위한 특별한 혜택, 마이크로사이트 무료 제공!

네임서버 설정

1차	ns-534.awsdns-02.net
2차	ns-382.awsdns-47.com
3차	ns-1723.awsdns-23.co.uk
4차	ns-1362.awsdns-42.org
5차	데이터 없음
6차	데이터 없음
7차	데이터 없음
8차	데이터 없음
9차	데이터 없음
10차	데이터 없음
11차	데이터 없음
12차	데이터 없음
13차	데이터 없음

보안 서비스 DNS 정보





★
APEX
Thank you

이정일 김민호 서재권 임재근