



Terraform 프로젝트

Terraform을 활용한 AWS 인프라 구축

Team Rest | 강승환 고동우 유세종 최성민 한시완





CONTENTS

01 Terraform 정의



02 환경 구축



03 Terraform 기본 개념



04 Terraform 작동 원리



05 TF 파일 분석



06 GitHub, Terraform Cloud, AWS 연동



07 Route53 도메인 등록





1. Terraform 정의

1. Terraform 정의



Terraform

- HashiCorp에서 오픈소스로 개발한 인프라 관리 도구, 프로비저닝 도구
- 리소스들을 코드로 작성해 관리
- HCL 또는 JSON 형식으로 작성된 TF 파일 사용



IaC

- IaC(Infrastructure as Code)는 "인프라를 코드로 관리한다"라는 개념
- 서버, 네트워크, 데이터베이스 같은 IT 인프라를 수동 설정하는 대신, 코드로 작성하고 자동화
- Terraform, AWS CloudFormation, Ansible 등



프로비저닝

- IT에서 필요한 자원(서버, 네트워크, 데이터베이스 등)을 준비하고 설정하는 과정
- 반복적인 작업을 줄이고, 복잡한 인프라도 빠르게 배포할 수 있어 효율적

2. 환경 구축

2-1. Terraform 연동



2-2. AWS CLI 연동



2-3. VSCode 연동



2-1. Terraform 설치

The screenshot shows the HashiCorp Terraform installation page. The browser address bar shows `https://developer.hashicorp.com/terraform/install`. The page has a navigation bar with links for Terraform, Install, Tutorials, Documentation, Registry, and Try Cloud. A sidebar on the left lists operating systems (macOS, Windows, Linux, FreeBSD, OpenBSD, Solaris) and other resources. The main content area is titled "Install Terraform" and shows the latest version, 1.13.0. It provides instructions for macOS using Homebrew and for Windows using binary downloads. The Windows section is highlighted with a red box, and the "Download" link for the AMD64 version is also highlighted with a red box. On the right, there are sections for "About Terraform", "Featured docs", and "HCP Terraform".

HashiCorp | The full conference agenda's now LIVE. Buy your pass and plan your schedule. [Register](#)

[Terraform](#) [Install](#) [Tutorials](#) [Documentation](#) [Registry](#) [Try Cloud](#)

[Terraform Home](#)

[Install Terraform](#)

Operating Systems

- macOS
- Windows
- Linux
- FreeBSD
- OpenBSD
- Solaris

Release information

Next steps

Resources

- Tutorial Library
- Certifications
- Community Forum
- Support
- GitHub
- Terraform Registry

Developer / Terraform

Install Terraform

1.13.0 (latest)

macOS

Package manager

```
brew tap hashicorp/tap
brew install hashicorp/tap/terraform
```

Binary download

AMD64 Version: 1.13.0	Download
ARM64 Version: 1.13.0	Download

Windows

Binary download

386 Version: 1.13.0	Download
AMD64 Version: 1.13.0	Download

About Terraform

Define cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.

Featured docs

- [Introduction to Terraform](#)
- [Configuration Language](#)
- [Terraform CLI](#)
- [HCP Terraform](#)
- [Provider Use](#)

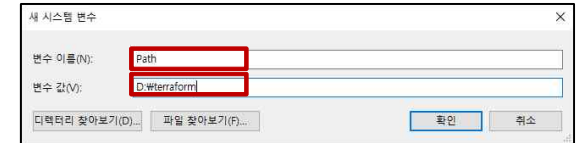
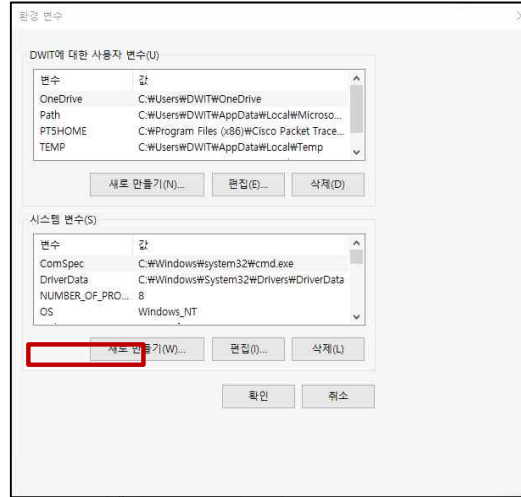
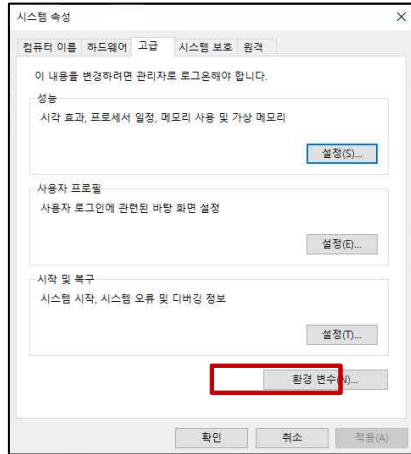
HCP Terraform

Automate your infrastructure provisioning at any scale

[Try HCP Terraform for free](#)

Terraform 설치 후 terraform.exe 파일을 원하는 경로 디렉터리에 저장 → D:\terraform

2-1. Windows 환경변수 등록

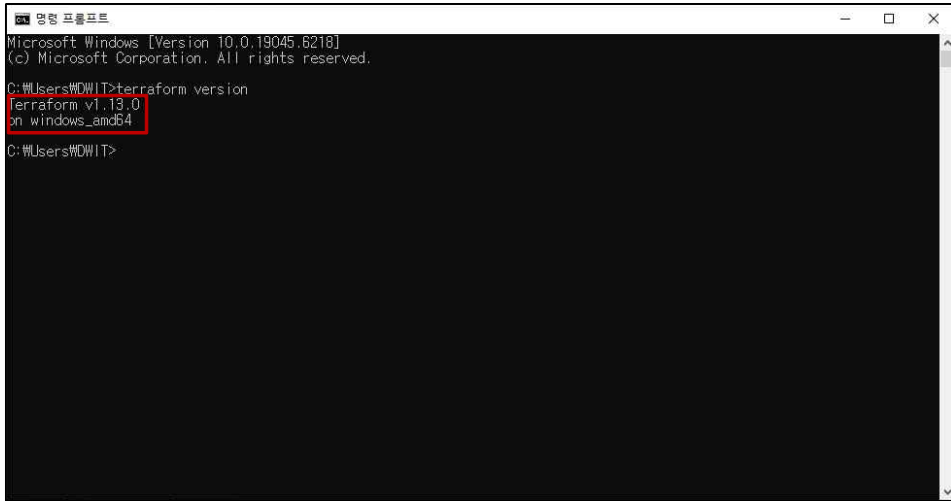


윈도우 버튼 우클릭 → 시스템
→ 고급 시스템 설정 → 환경 변수

시스템 변수 새로 만들기

변수 이름 : Path
변수 값 : D:\terraform

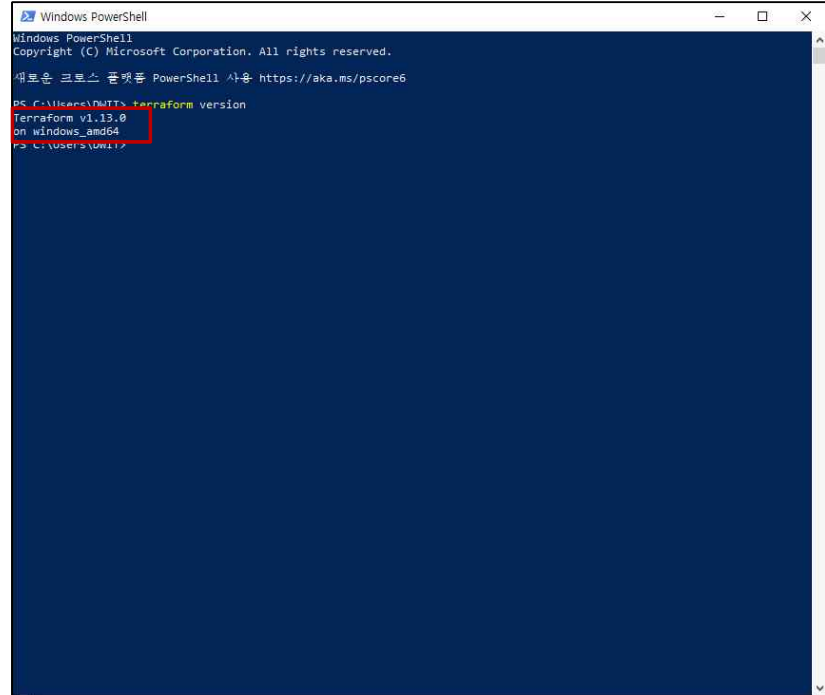
2-1. Terraform 연동 확인



```
Microsoft Windows [Version 10.0.19045.6218]
(c) Microsoft Corporation. All rights reserved.

C:\Users\WDW\IT>terraform version
Terraform v1.13.0
on windows_amd64

C:\Users\WDW\IT>
```



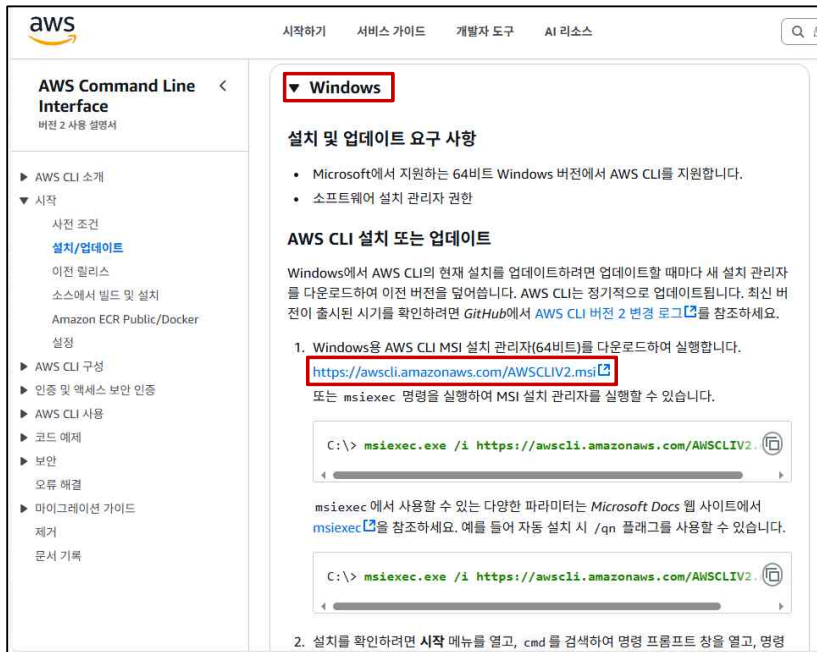
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\WDW\IT> terraform version
Terraform v1.13.0
on windows_amd64
PS C:\Users\WDW\IT>
```

CMD나 Windows PowerShell에서 **terraform version** 입력

2-2. AWS CLI 설치 및 확인



aws

시작하기 서비스 가이드 개발자 도구 AI 리소스

AWS Command Line Interface
버전 2 사용 설명서

▶ AWS CLI 소개
▼ 시작
사전 조건
설치/업데이트
이전 릴리스
소스에서 빌드 및 설치
Amazon ECR Public/Docker
설치
▶ AWS CLI 구성
▶ 인증 및 액세스 보안 인증
▶ AWS CLI 사용
▶ 코드 예제
▶ 보안
오류 해결
▶ 마이그레이션 가이드
제거
문서 기록

▼ Windows

설치 및 업데이트 요구 사항

- Microsoft에서 지원하는 64비트 Windows 버전에서 AWS CLI를 지원합니다.
- 소프트웨어 설치 관리자 권한

AWS CLI 설치 또는 업데이트

Windows에서 AWS CLI의 현재 설치를 업데이트하려면 업데이트할 때마다 새 설치 관리자를 다운로드하여 이전 버전을 덮어씁니다. AWS CLI는 정기적으로 업데이트됩니다. 최신 버전이 출시된 시기를 확인하려면 [GitHub에서 AWS CLI 버전 2 변경 로그](#)를 참조하세요.

1. Windows용 AWS CLI MSI 설치 관리자(64비트)를 다운로드하여 실행합니다.

<https://awscli.amazonaws.com/AWSCLIV2.msi>

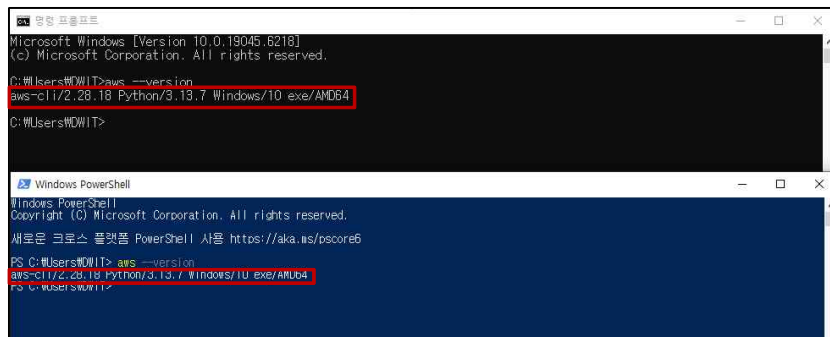
또는 msixec 명령을 실행하여 MSI 설치 관리자를 실행할 수 있습니다.

```
C:\> msixec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

msixec에서 사용할 수 있는 다양한 파라미터는 [Microsoft Docs](#) 웹 사이트에서 [msixec](#)을 참조하세요. 예를 들어 자동 설치 시 /qn 플래그를 사용할 수 있습니다.

```
C:\> msixec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi /qn
```

2. 설치를 확인하려면 **시작** 메뉴를 열고, **cmd**를 검색하여 명령 프롬프트 창을 열고, 명령



```
Microsoft Windows [Version 10.0.19045.6218]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\WDW\Documents> aws --version  
aws-cli/2.28.18 Python/3.13.7 Windows/10 exe/AMD64  
  
C:\Users\WDW\Documents>
```

```
Windows PowerShell  
Copyright (c) Microsoft Corporation. All rights reserved.  
  
새로운 크로스 플랫폼 PowerShell 사용: https://aka.ms/pscore6  
  
PS C:\Users\WDW\Documents> aws --version  
aws-cli/2.28.18 Python/3.13.7 Windows/10 exe/AMD64  
No profile found for 'WDW\WDW\Documents'.
```

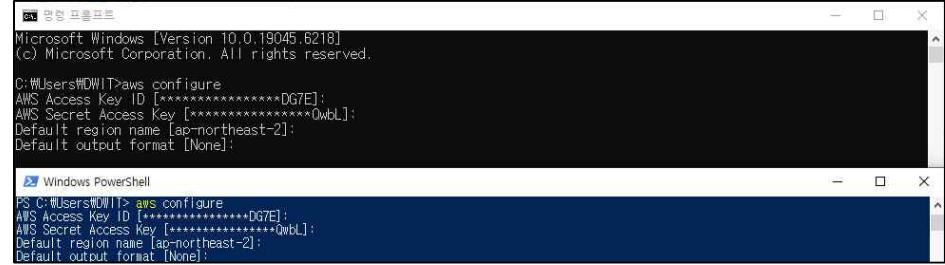
Windows → <https://awscli.amazonaws.com/AWSCLIV2.msi> 다운로드

CMD나 Windows PowerShell에서 **aws --version** 입력

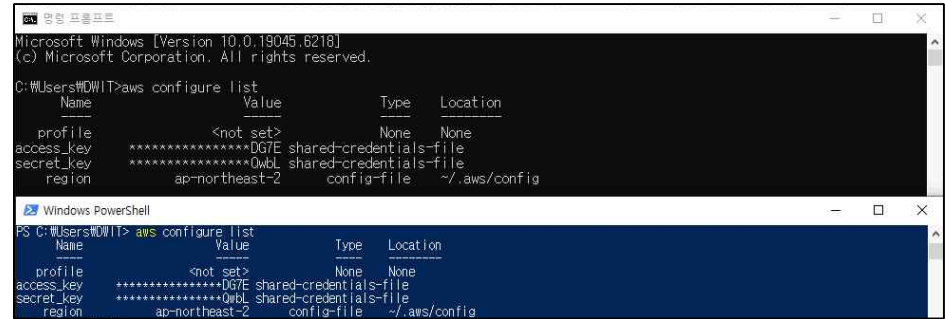
2-2. AWS 액세스 키로 로그인 및 확인



AWS → IAM → 사용자 → 보안 자격 증명 → 액세스 키 만들기

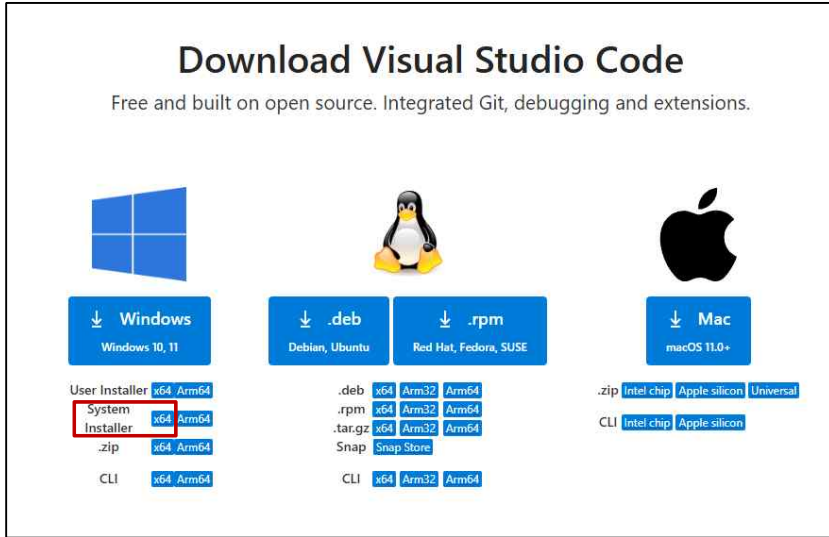


CMD나 Windows PowerShell에서 **aws configure** 입력
Access Key ID, Secret Access Key, Default region 입력

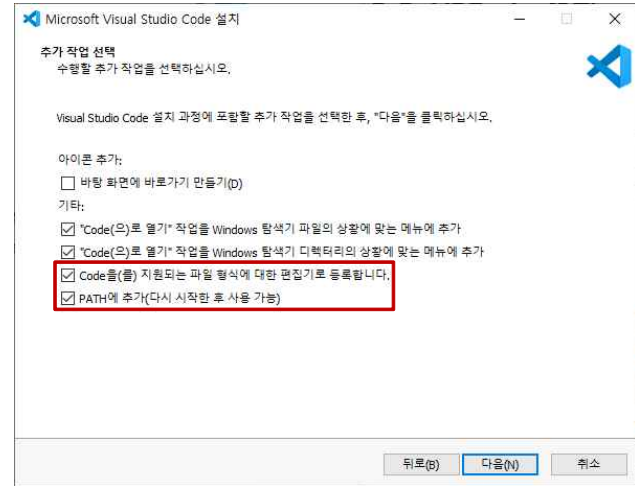


CMD나 Windows PowerShell에서 **aws configure list** 입력

2-3. VSCode 설치



Download → Windows System installer x64 다운로드



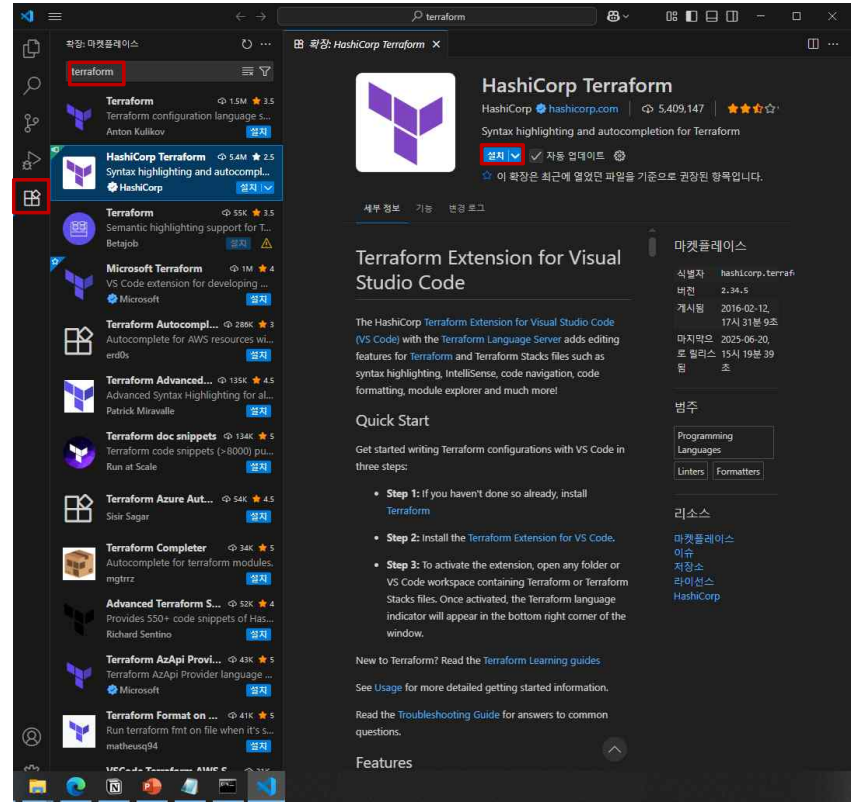
필수 체크

- Code을(를) 지원되는 파일 형식에 대한 편집기로 등록합니다.
- PATH에 추가(다시 시작한 후 사용 가능)

2-3. Terraform 및 한국어 확장 설치

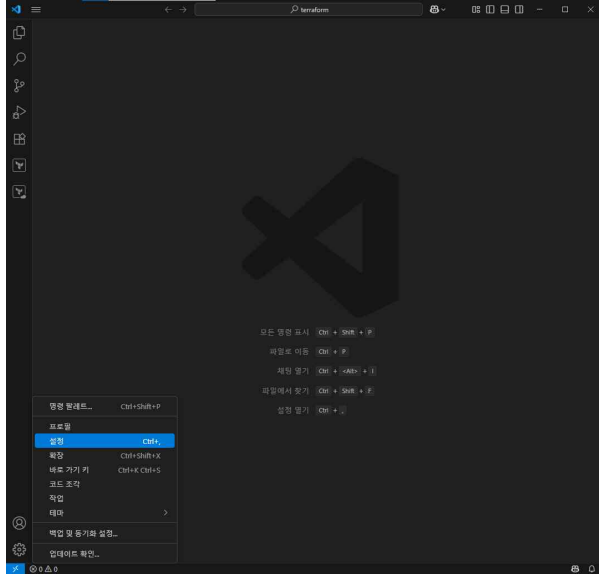


VSCode에서 korean 검색
Korean Language Pack for Visual 설치

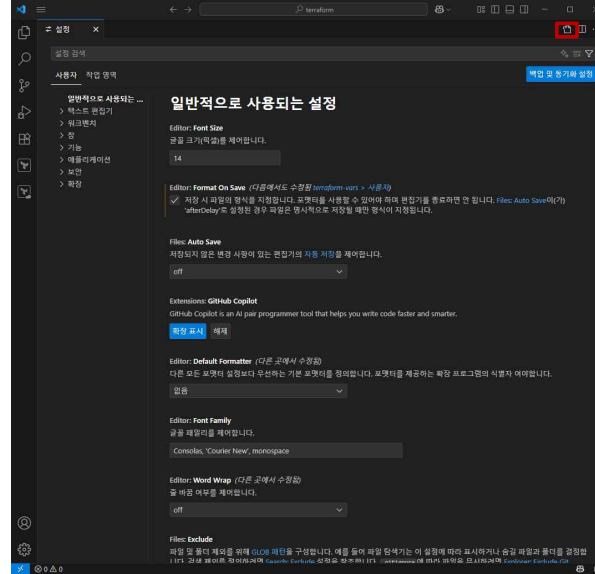


VSCode에서 terraform 검색
HashiCorp Terraform 설치

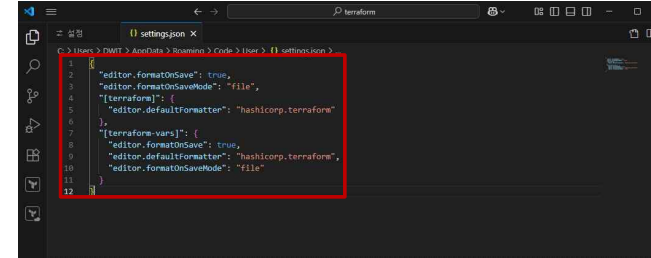
2-3. VSCode 설정



좌측 하단 톱니바퀴 → 설정



우측 상단 파일 아이콘 (설정열기 JSON) 클릭



JSON 코드 추가 후 저장

3. Terraform 기본 개념

3. Terraform 기본 개념 - 1

Step.01

terraform.tfstate 파일

- Terraform이 관리하는 인프라의 현재 상태를 저장하는 JSON 파일
- Terraform이 어떤 리소스가 생성되었고, 어떤 속성을 가지고 있는지 추적
- 협업 시 Terraform Backend 설정



Step.02

provider

- Terraform이 어떤 클라우드나 서비스를 사용할지 정의
- AWS, Azure, Kubernetes 등 다양한 Provider 지원
- 각 클라우드 플랫폼에 맞는 provider를 지정해야함



Step.03

resource

- 실제로 생성할 인프라 자원 정의
- EC2, VPC, S3, RDS 등 다양한 리소스 프로비저닝



Step.04

variable

- 코드에서 유동적인 값을 설정할 수 있는 파라미터
- 코드의 재사용성과 유연성을 높여줌

3. Terraform 기본 개념 - 2

Step.05

output

- 생성된 리소스의 정보를 외부에 출력하는 기능
- IP주소, DNS 이름 등을 사용자에게 출력
- 다른 Terraform 구성이나 스크립트에서 값을 가져와야 사용 가능



Step.06

module

- 반복되는 리소스 구성을 재사용 가능하게 만든 구조
- 팀 프로젝트나 멀티 환경 구성시 유용



Step.07

locals

- 반복되는 값이나 계산식을 변수처럼 정의
- 코드 가독성과 유지보수성 향상



Step.08

data

- 이미 존재하는 외부 리소스 참조
- 최신 AMI, 기존 VPC 등



4. Terraform 작동 원리

4. Terraform 작동 원리 - 1



Step.01

Init

- Terraform 작업 디렉터리를 초기화하는 단계
- 구성 파일에 정의된 Provider 플러그인을 다운로드
- .terraform 디렉터리 생성

Step.02

Validate

- 작성한 .tf 파일의 문법과 구조가 올바른지 검사
- 변수 누락, 타입 오류, 잘못된 참조 등을 사전에 확인 가능

Step.03

Plan

- 현재 인프라 상태와 코드의 차이를 비교
- 변경 사항을 미리 검토하는 단계
- 수행할 작업을 시뮬레이션하여 예측 결과를 보여줌

4. Terraform 작동 원리 - 1



Step.04

Apply

- plan에서 계산된 변경 사항을 실제 클라우드 인프라에 적용
- 적용 후, 그 결과를 상태 파일(terraform.tfstate)에 저장



Step.05

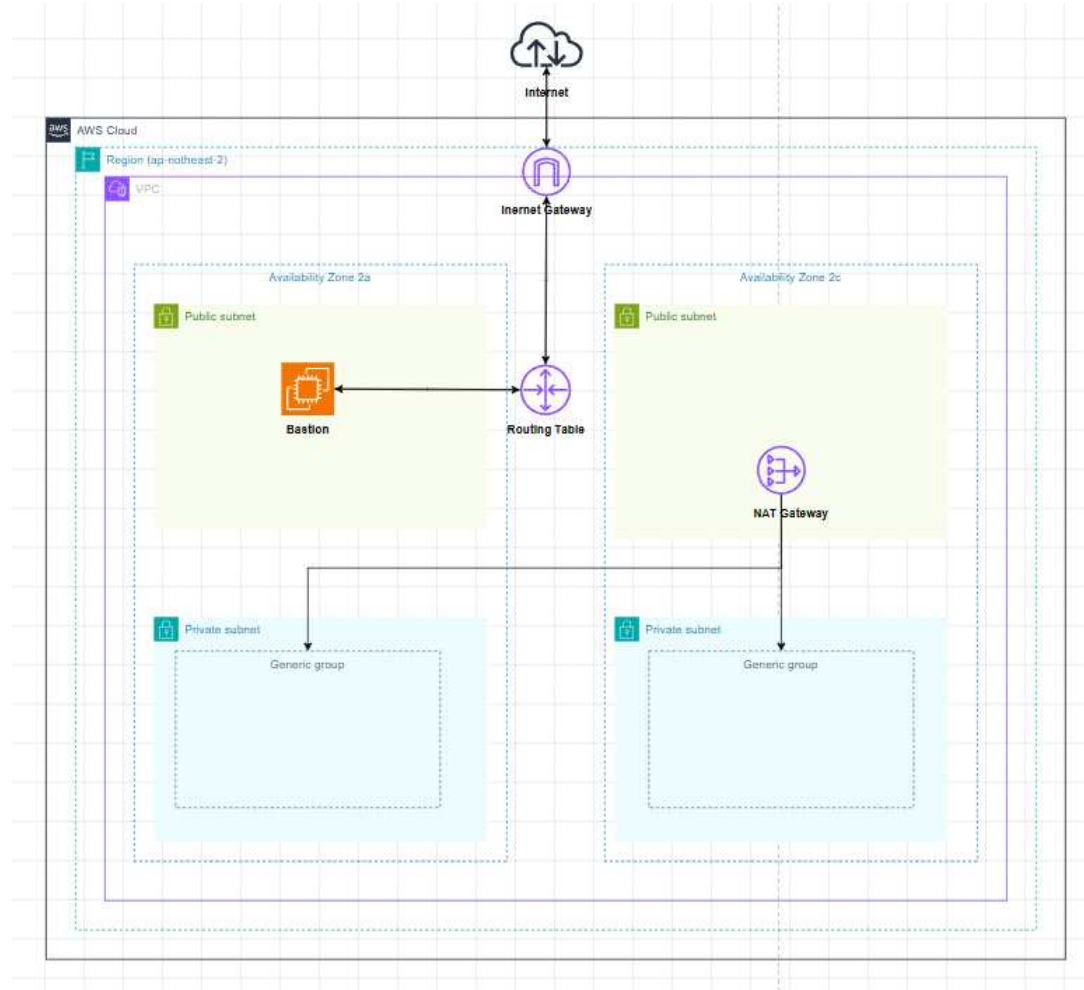
Destroy

- 모든 리소스를 제거하는 단계
- terraform.tfstate도 함께 업데이트



5. TF 파일 분석

5. TF 파일 분석 - 구성도



5. TF 파일 분석 - provider.tf

- Terraform이 어떤 클라우드나 서비스를 사용할지 지정하는 설정 파일
- AWS Provider를 통해 리전, 인증 정보 등을 설정하여 Terraform이 AWS API와 통신 가능
- Terraform이 리소스를 올바르게 프로비저닝 가능

TF 파일 구조

```
terraform
├── 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
```

provider.tf 코드

```
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
  }
}
```

5. TF 파일 분석 - vpc.tf

- VPC(Virtual Private Cloud)는 AWS에서 제공하는 가상 네트워크
- 사용자가 정의한 IP주소 범위 내에서 리소스를 배치
- 트래픽 제어와 외부 접근 조절 가능

TF 파일 구조

```
terraform
├── 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
```

vpc.tf 코드

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

5. TF 파일 분석 – subnet.tf

- VPC 내에서 IP 주소 범위를 더 작은 네트워크로 나눈 논리적 구역
- 퍼블릭 서브넷은 IGW를 통해 외부와 통신, 프라이빗 서브넷은 내부 네트워크에서만 접근 가능
- 이를 통해 가용 영역 분산 및 보안성 강화 구현

TF 파일 구조

```
terraform
├── 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
```

subnet.tf 코드

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"
  }
}
```

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"
  }
}
```


5. TF 파일 분석 - igw.tf

- IGW(Internet Gateway)는 외부 통신이 불가능한 VPC와 인터넷을 연결해주는 리소스
- 퍼블릭 서브넷의 인스턴스가 외부와 통신할수 있도록 라우팅 테이블과 함께 사용

TF 파일 구조

```
terraform
├── 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
```

igw.tf 코드

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

5. TF 파일 분석 - ngw.tf

- NAT Gateway는 프라이빗 서브넷의 리소스가 인터넷으로 아웃바운드 통신을 할 수 있게 해주는 AWS 서비스
- 프라이빗 네트워크의 사설 IP를 공인 IP로 변환하여 외부와의 통신을 가능하게 함

TF 파일 구조

```
terraform
├── 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
```

ngw.tf 코드

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"
  }
}
```

5. TF 파일 분석 – routing-table.tf

- VPC 내 트래픽의 경로를 정의하는 AWS 네트워크 구성 요소
- IGW나 NGW를 통한 외부 통신 경로를 설정

TF 파일 구조

```
terraform
├── 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
```

routing-table.tf 코드

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
}
```

명시적 서브넷 연결

```
resource "aws_route_table_association" "terraform-pub-rt-associate-2a" {
  subnet_id      = aws_subnet.terraform-pub-subnet-2a.id
  route_table_id = aws_route_table.terraform-pub-rt.id
}
```

```
resource "aws_route_table_association" "terraform-pub-rt-associate-2c" {
  subnet_id      = aws_subnet.terraform-pub-subnet-2c.id
  route_table_id = aws_route_table.terraform-pub-rt.id
}
```

```
resource "aws_route_table_association" "terraform-pri-rt-associate-2a" {
  subnet_id      = aws_subnet.terraform-pri-subnet-2a.id
  route_table_id = aws_route_table.terraform-pri-rt.id
}
```

```
resource "aws_route_table_association" "terraform-pri-rt-associate-2c" {
  subnet_id      = aws_subnet.terraform-pri-subnet-2c.id
  route_table_id = aws_route_table.terraform-pri-rt.id
}
```

5. TF 파일 분석 – security-group.tf

- AWS에서 네트워크 트래픽을 제어하는 가상 방화벽
- 리소스에 대한 인바운드 및 아웃바운드 트래픽 제어
- 모든 프로토콜 허용은 -1로 표현

TF 파일 구조

```
terraform
├── 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
```

security-group.tf 코드

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

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"
  }
}
```

EKS SG

```
resource "aws_security_group" "terraform-sg-eks-node-group"
{
  name        = "for EKS-managed server"
  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"
  }
}
```

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"
  }
}
```

5. TF 파일 분석 – ec2.tf

- EC2(Elastic Compute Cloud)는 AWS에서 제공하는 확장 가능한 가상 서버 서비스
- 사용자가 설정한 성능과 크기의 가상 서버를 생성하고, 관리하며, 사용한 만큼 비용 지불

TF 파일 구조

```
terraform
├── 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
```

ec2.tf 코드

```
resource "aws_instance" "terraform-pub-ec2-bastion-2a" {
  ami              = "ami-056a29f2eddc40520"
  instance_type    = "t3.micro"
  vpc_security_group_ids = [aws_security_group.terraform-sg-bastion.id]
  subnet_id        = aws_subnet.terraform-pub-subnet-2a.id
  key_name          = "soonge97"
  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"
}
```

5. TF 파일 분석 – key_pair.tf

- EC2 인스턴스에 안전하게 로그인 하기 위한 공개키와 개인키 쌍
- 이를 통해 비밀번호 없이 안전하게 인증 가능
- 개인 키를 분실하면 해당 인스턴스에 접근 불가

TF 파일 구조

```
terraform
├── 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
```

key_pair.tf 코드

```
resource "tls_private_key" "soonge97_key" {
  algorithm = "RSA"
  rsa_bits  = 2048
}

resource "aws_key_pair" "soonge97_aws_key" {
  key_name   = "soonge97"
  public_key = tls_private_key.soonge97_key.public_key_openssh
}

resource "local_file" "soonge97_private_key" {
  content = tls_private_key.soonge97_key.private_key_pem
  filename = "soonge97.pem"
}
```

5. TF 파일 분석 – rds.tf

- RDS(Relational Database Service)는 관리형 데이터베이스 서비스
- MySQL, PostgreSQL, MariaDB 등 다양한 엔진 지원
- 사용자는 서버 관리, 백업, 패치 등을 직접 관리하지 않고, 데이터베이스를 쉽게 생성 및 운영 가능

TF 파일 구조

```
terraform
├── 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
```

rds.tf 코드

RDS 서브넷 그룹 생성

```
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"
  }
}
```

MariaDB Parameter Group 설정

```
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"
  }
}
```

RDS 생성

```
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"
  }
}
```

5. TF 파일 분석 – s3.tf

- S3(Simple Storage Service)는 AWS에서 제공하는 객체 스토리지 서비스
- 데이터를 안정적으로 저장하고 인터넷을 통해 접근
- 파일, 이미지, 백업 등 다양한 데이터를 버킷 단위로 관리

TF 파일 구조

```
terraform
├── 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
```

s3.tf 코드

```
resource "aws_s3_bucket" "rest-s3-bucket1" {
  bucket = "rest-s3-bucket1"
}
```


5. TF 파일 분석 – acm.tf

- ACM(Amazon Certificate Manager)은 SSL/TLS 인증서를 간편하게 생성, 관리, 배포할 수 있는 서비스
- 웹사이트의 HTTPS를 활성화하는데 필수

TF 파일 구조

```
terraform
├── 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
```

acm.tf 코드

인증서 생성

```
resource "aws_acm_certificate" "cert" {
  domain_name      = "*.mjc-hwan.shop"
  validation_method = "DNS"

  tags = {
    Name = "mjc-hwan.shop"
  }

  lifecycle {
    create_before_destroy = true
  }
}
```

인증서 검증

```
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
}
```

5. TF 파일 분석 – route53.tf

- AWS에서 제공하는 확장 가능하고 고가용성의 DNS(Domain Name System) 웹 서비스
- 도메인 이름을 IP 주소로 변환하여 인터넷 트래픽을 라우팅

TF 파일 구조

```
terraform
├── 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
```

route53.tf 코드

```
resource "aws_route53_zone" "route53" {
  name = "mjc-hwan.shop"
}
```

5. TF 파일 분석 – alb.tf

- ALB(Application Load Balancer)는 AWS에서 애플리케이션 계층(HTTP/HTTPS) 트래픽을 분산시키는 로드밸런서
- URL 경로, 호스트 이름 등 트래픽을 특정 대상(Target Group)으로 라우팅 가능

TF 파일 구조

```
terraform
├── 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
```

alb.tf 코드

lb 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"
  }
}
```

Target Group

```
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"
  }
}
```

Listener

```
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"
  }
}
```

target group attachment

```
/*
resource "aws_lb_target_group_attachment" "terraform-prd-tg-
attachment1" {
  target_group_arn = aws_lb_target_group.terraform-prd-tg.arn
  target_id = aws_instance.terraform-pub-ec2-bastion-2a.id
  port = 80
}
*/
```

5. TF 파일 분석 – auto.tf

- Auto Scaling은 AWS에서 EC2 인스턴스 수를 자동으로 조절해주는 기능
- 트래픽 증가 시 인스턴스를 늘리고, 감소 시 줄여서 비용과 성능을 최적화

TF 파일 구조

```
terraform
├── 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
```

auto.tf 코드

Launch Template (기존 Launch Configuration 대체)

```
resource "aws_launch_template" "as_conf" {
  name_prefix  = "terraform-lt-backend"
  image_id     = "ami-056a29f2eddc40520"
  instance_type = "t3.micro"
  key_name     = "soonge97"

  user_data = base64encode(<<-EOF
    #!/bin/bash
    sudo apt update
    sudo apt install -y nginx
  EOF
)

  vpc_security_group_ids = [aws_security_group.terraform-sg-
bastion.id]

  tag_specifications {
    resource_type = "instance"
    tags = {
      Name = "jeff-userdata"
    }
  }
  lifecycle {
    create_before_destroy = true
  }
}
```

Auto Scaling Group (Launch Template 기반으로 수정)

```
resource "aws_autoscaling_group" "terraform-prd-asg" {
  name                  = "terraform-prd-asg"
  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]
  launch_template {
    id      = aws_launch_template.as_conf.id
    version = "$Latest"
  }
  tag {
    key      = "Name"
    value    = "terraform-prd-asg"
    propagate_at_launch = true
  }
  lifecycle {
    create_before_destroy = true
  }
}
```



6. GitHub, Terraform Cloud, AWS 연동

6. GitHub, Terraform Cloud, AWS 연동

- Terraform Cloud와 Github을 연동하면 코드 변경 시 자동으로 인프라 배포가 트리거되어 CI/CD 환경을 구축 가능
- AWS와 연동함으로써 실제 클라우드 리소스를 자동으로 생성 및 관리
- 이를 통해 코드 기반 인프라 관리의 자동화, 일관성, 협업 효율성 확보

6. GitHub Repository 생성

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 * Han-siwan Repository name * terraformclass
✓ terraformclass is available.

Great repository names are short and memorable. How about [super-doodle?](#)

Description

0 / 350 characters

2 Configuration

Choose visibility * Public
Choose who can see and commit to this repository.

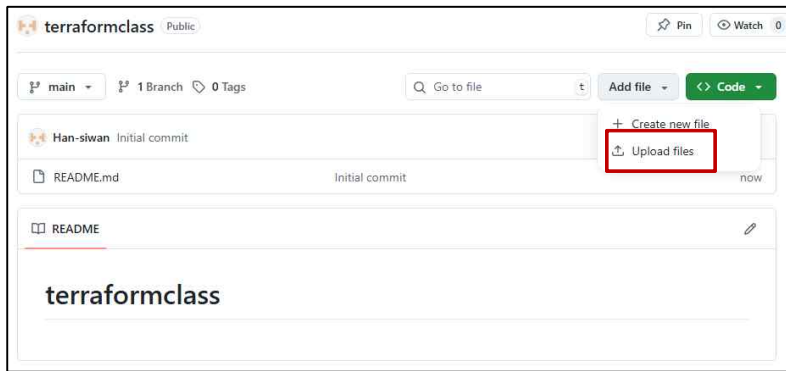
Add README On
READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore No .gitignore
.gitignore tells git which files not to track. [About ignoring files](#)

Add license No license
Licenses explain how others can use your code. [About licenses](#)

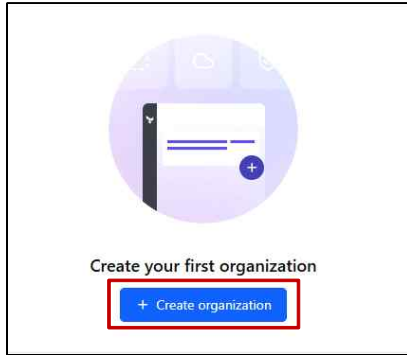
Create repository

새로운 Repository 생성



Repository에 .tf 파일 업로드

6. Terraform Cloud 초기 구성 & GitHub 연동



Terraform.io 가입 후
Create Organization



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 (_).

Rest-Terraform

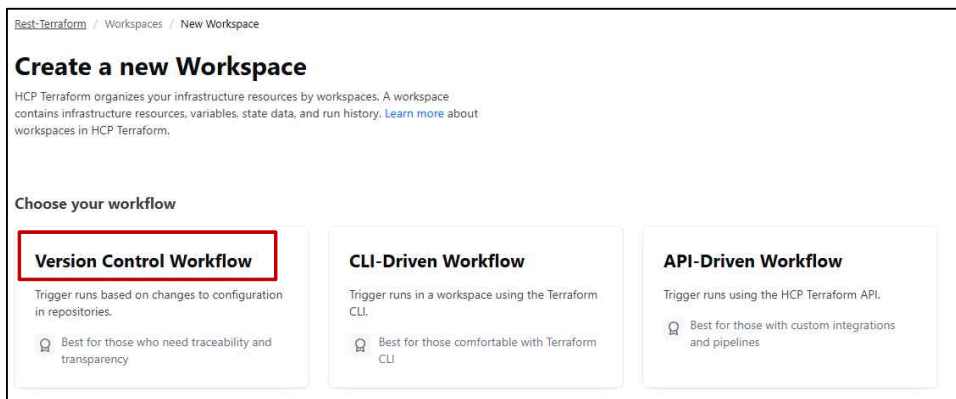
Email address

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

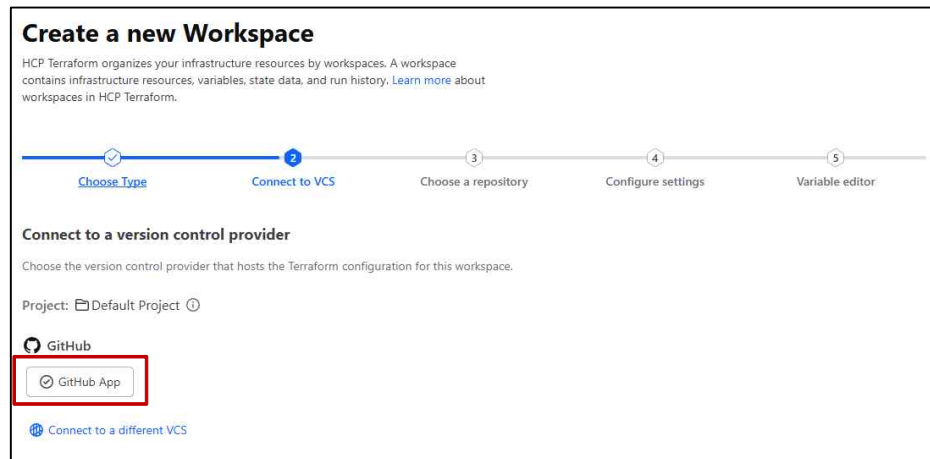
Create organization Cancel

이름과 Email 주소를 추가하여
Organization 생성

6. Terraform Cloud 초기 구성 & GitHub 연동



Version Control Workflow 선택



GitHub 선택 후 연동할
GitHub Repository 선택

6. Terraform Cloud 초기 구성 & GitHub 연동

Create a new Workspace

HCP Terraform organizes your infrastructure resources by workspaces. A workspace contains infrastructure resources, variables, state data, and run history. [Learn more](#) about workspaces in HCP Terraform.

Choose Type Connect to VCS **Choose a repository** Configure settings Variable editor

Choose a repository

Choose the repository that hosts your Terraform source code. We'll watch this for commits and pull requests.

Don't have a repo? Here's an [example repo](#) you can copy to get started.

Han-siwan ▾ 1 repository

Filter acme-corp/infrastructure

terraformclass >

Can't see your repository? Enter its ID below, e.g. `acme-corp/infrastructure` :

acme-corp/infrastructure >



Create a new Workspace

HCP Terraform organizes your infrastructure resources by workspaces. A workspace contains infrastructure resources, variables, state data, and run history. [Learn more](#) about workspaces in HCP Terraform.

Choose Type Connect to VCS Choose a repository **Configure settings** Variable editor

Configure Settings

Workspace Name

terraformclass

The name of your workspace is unique and used in tools, routing, and UI. Dashes, underscores, and alphanumeric characters are permitted. [Learn more about naming workspaces](#).

Description (Optional)

Workspace description

Tags

Use tags to correlate, organize, or filter your resources using single values or key/value pairs.

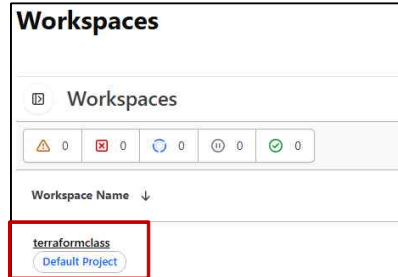
+ Add tag

Advanced options

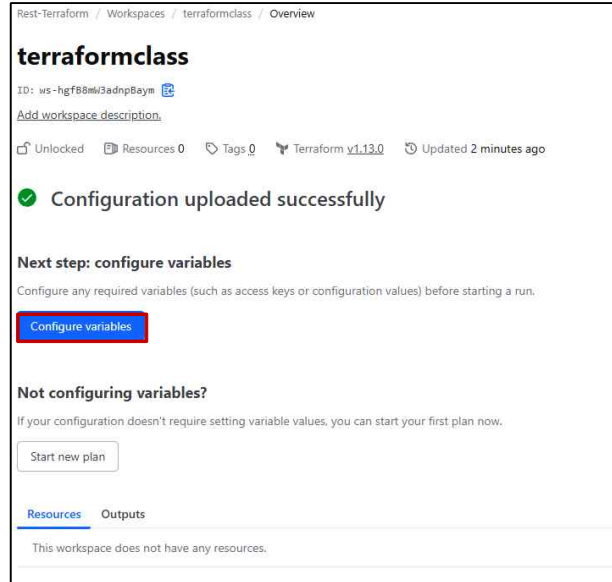
< Previous Cancel Create

Workspace 이름 지정

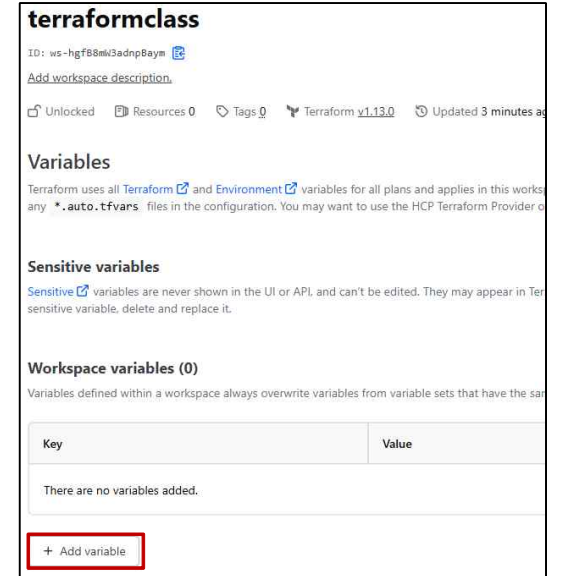
6. Terraform Cloud & AWS 연동



작업할 Workspace 선택



Configure variables 선택



Add variable 선택

6. Terraform Cloud & AWS 연동

Add variable [X]

Select variable category

☒ Terraform variable
These variables should match the declarations in your configuration. Click the HCL box to use interpolation or set a non-string value.

☐ Environment variable
These variables are available in the Terraform runtime environment.

Key
AWS_ACCESS_KEY_ID

Value
AKIA4X2DY6NA6L4FDG7E

☒ HCL ⓘ ☐ Sensitive ⓘ

Description (Optional)
description (optional)

Add variable Cancel



Add variable [X]

Select variable category

☒ Terraform variable
These variables should match the declarations in your configuration. Click the HCL box to use interpolation or set a non-string value.

☐ Environment variable
These variables are available in the Terraform runtime environment.

Key
AWS_SECRET_ACCESS_KEY

Value
.....

☒ HCL ⓘ ☒ Sensitive ⓘ

Description (Optional)
description (optional)

Add variable Cancel



Workspace variables (2)

Variables defined within a workspace always overwrite variables from variable sets that have the same type and the same key. Learn more about variable set [precedence](#).

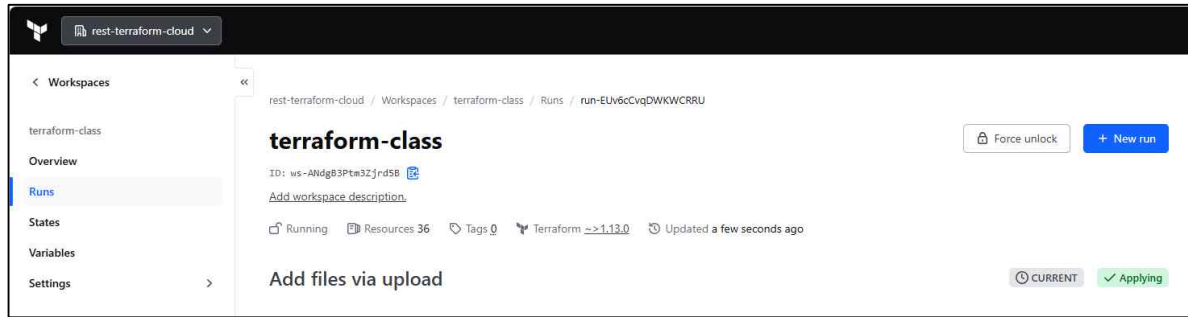
Key	Value	Category	Actions
AWS_ACCESS_KEY_ID HCL	AKIA4X2DY6NA6L4FDG7E	terraform	...
AWS_SECRET_ACCESS_KEY HCL Sensitive	Sensitive - write only	terraform	...

AWS_SECRET_ACCESS_KEY 입력
→ Add variable

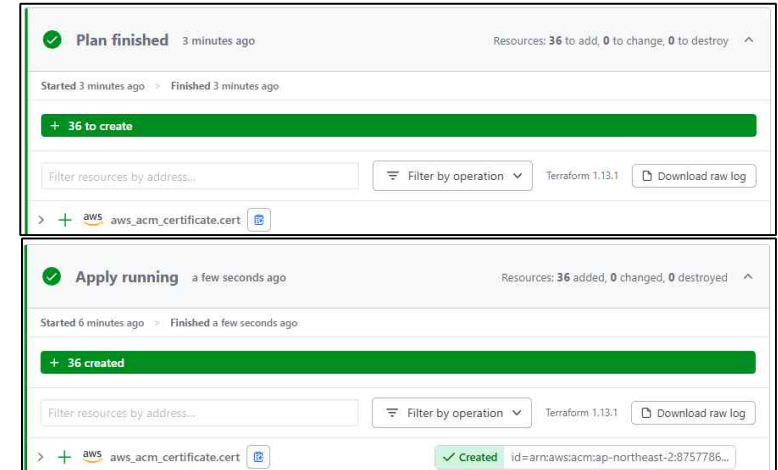
AWS_SECRET_ACCESS_KEY 입력
→ Add variable

생성 확인

6. Terraform Cloud & AWS 연동

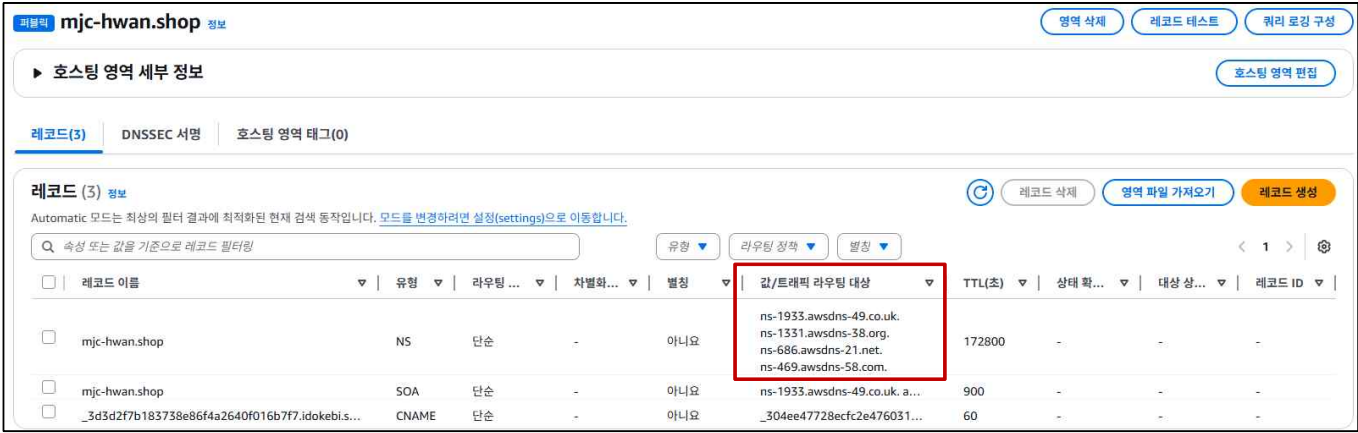


WorkSpace → Runs → New run



연동 확인

7. Route 53 도메인 등록



AWS Route 53에서 호스팅 영역 생성



네임서버 설정	
1차	ns-1933.awsdns-49.co.uk
2차	ns-1331.awsdns-38.org
3차	ns-686.awsdns-21.net
4차	ns-469.awsdns-58.com
5차	데이터 없음
6차	데이터 없음
7차	데이터 없음

생성된 NS 코드 값 가비아에
입력



감사합니다

Team Rest | 강승환 고동우 유세종 최성민 한시완

