



Terraform Module로

코드 재사용성 높이기



AUSG 6th 정지우

FOR WHOM?

Terraform 매력에 빠질 준비가 된 사람들

laC? Terraform? 궁금했으나 아직 경험해보지 못한 분들,
입문자분들, 모두 환영합니다!

내 이야기 들어줄 사람들

빅챗으로 TMT 욕구 해소하기 ☺

CONTENTS

1. Terraform Basic Concept

Terraform을 이해하기 위한 개념과 Terraform Workflow를 소개합니다

2. Terraform Module

모듈화를 왜 하는 걸까요? 모듈화를 진행하게 된 제 이야기와 그 방법을 들려드릴게요

3. 간단한 Demo

Dev와 Prod 환경의 인프라를 간단하게 구축해볼게요

1. Terraform Basic Concept



1. Terraform Basic Concept

Provisioning

사용자의 요구에 맞게 시스템 자원을 할당, 배치, 배포해 두었다가 필요 시 시스템을 즉시 사용할 수 있는 상태로 미리 준비해 두는 것
⇒ IT 인프라 환경과 클라우드 리소스를 구성하는 프로세스!

IaC (Infrastructure as Code)

코드형 인프라; 코드로 인프라를 관리하고 프로비저닝하는 것 ⇒ 프로비저닝 자동화

1. Terraform Basic Concept

Terraform

developed by HashiCorp

Ö AWS 리소스만 생성할 수 있나요?

아니요! AWS, GCP, Azure, OCI 등 다양한 클라우드 프로바이더를 제공해요.
그리고 HashiCorp의 Vault, Vagrant와도 통합해서 사용할 수 있어요.

Ö 어떤 언어로 개발됐나요?

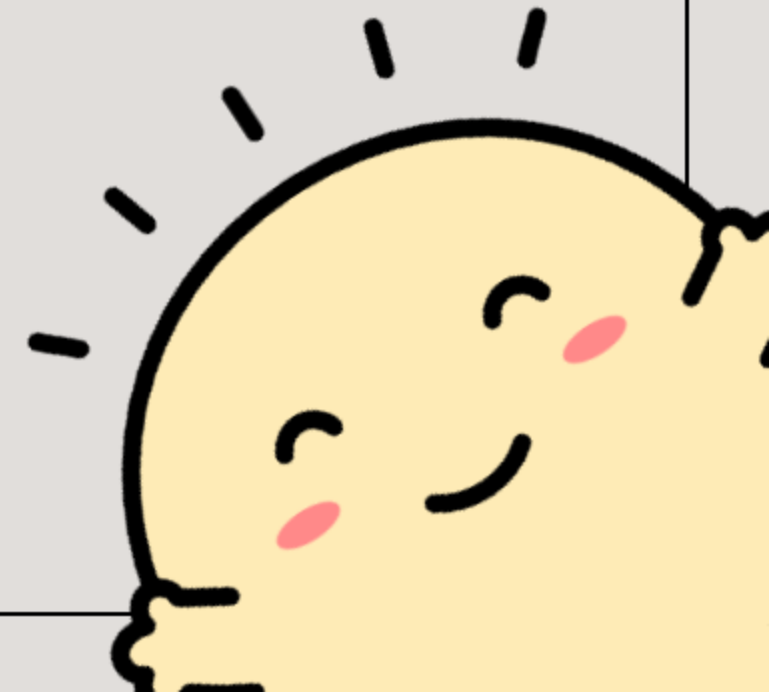
Go 언어로 개발되고, Go 런타임 기반으로 실행됩니다.
그러나 HCL(HashiCorp Configuration Language)로 인프라 구성을 작성해요.

Ö 그럼 실행은 어떻게 해요?!

Terraform이 HCL 파일을 읽어 Go로 코드를 변환하고, Terraform Provider를 사용해서
인프라 리소스를 생성, 업데이트, 삭제합니다.

>> Terraform Registry

<https://registry.terraform.io/>



1. Terraform Basic Concept

Terraform Registry vs Terraform Provider

○ Terraform Registry

Terraform 모듈을 검색하고 다운로드할 수 있는 저장소

○ Terraform Provider

Terraform이 사용 가능한 인프라 리소스를 제공하는 플러그인



1. Terraform Basic Concept

Terraform Workflow

Configure resources

- terraform provider docs에서 제공하는 code와 argument을 참고하여 원하는 대로 구성

terraform init

- 프로젝트의 모듈과 플러그인 버전을 확인하고, 그 버전에 맞게 다운로드
- 로컬/원격 저장소 초기화 (backend initialized)
- 해당 디렉토리에 .terraform 폴더 생성 → 모듈과 프로바이더 정보 포함
- .terraform.lock.hcl 파일 생성 → 현재 사용 중인 Terraform 모듈과 플러그인 버전 정보 포함

terraform plan

- 기존 인프라 상태와 현재 Terraform 코드를 비교해서 추가, 변경, 삭제 리소스 정보 나열
⇒ apply하기 전에 리소스의 변경 사항을 미리 검토하는 단계로, 필수는 아니지만 권장 단계

terraform apply

- terraform plan 명령 실행 시 보여줬던 리소스의 변경 사항을 적용하는 단계
- terraform.tfstate 상태파일 업데이트
- -auto-approve 옵션 사용 시, approval 과정을 생략하고 프로비저닝

2. Terraform Module

2. Terraform Module

Tree Structure

terraform

└─ network.tf

└─ ec2.tf

└─ sg.tf

└─ lb.tf

└─ db.tf

└─ variables.tf

└─ outputs.tf

...

리소스 정의 파일

- `resource` block : 직접 프로비저닝할 리소스 정의
- `data` block : 프로비저닝되어 있는 리소스 불러오기

`variable` block으로 선언할 변수 정의

- (required)type

- (optional)description, default, sensitive, validation

다른 리소스를 정의한 파일에서 참조할 수 있도록
`output` block을 작성하여 값을 내보내는 파일

2. Terraform Module

Tree Structure

terraform

└─ network.tf

└─ ec2.tf

└─ sg.tf

└─ lb.tf

└─ db.tf

└─ variables.tf

└─ outputs.tf

...

○ 근데 프로비저닝할 리소스가 많아지면?
파일이 계속 늘어날텐데 관리가 어렵지 않을까?



○ 리소스별로 변수값을 따로 관리하고 싶다
하나의 폴더 안에 모든 파일을 관리한다?

○ ...개발 환경을 나눠서 작업할 땐 이걸 다 복붙해야돼?




여기까지 생각이 미치니 갑자기 Terraform의 매력이 뚝 떨어지고,
Terraform을 잘못 활용하고 있는 것 같다-라는 생각



최대한 **협업의 관점**에서 바라보자!

2. Terraform Module

어떻게 개선할 수 있을까? 

① Terraform Workspace : 작업 환경을 격리하자!

② **Terraform Module** : 리소스를 모듈로 관리하자!

+ backend configuration



▽ 'Automation', 'Code Reusability', 'Maintenance' ▽

2. Terraform Module

Module Tree Structure

Root Module



자식 모듈을 호출하고,
필요한 변수값을 정의하는
부모 모듈

dev

- └ main.tf
- └ variables.tf
- └ terraform.tfvars

prod

- └ main.tf
- └ variables.tf
- └ terraform.tfvars

...

Child Module



부모 모듈에서 정의한
변수값에 따라
리소스를 생성하고
관리하는 자식 모듈

modules

- └ network
 - └ main.tf
 - └ variables.tf
 - └ outputs.tf
- └ ec2
 - └ main.tf
 - └ variables.tf
 - └ outputs.tf

...

2. Terraform Module

Module Tree Structure

Root Module

dev

└ main.tf

└ variables.tf

└ **dev.tfvars**

prod

└ main.tf

└ variables.tf

└ **prod.tfvars**

...

`variable` block으로 선언할 변수 정의

- (required)type

- (optional)description, default, sensitive, validation

variables.tf에 선언한 변수의 값을 정의하는 파일

⇒ 이름이 꼭 terraform.tfvars가 아니어도 되지만,

`-var-file` flag로 파일을 지정해줘야 정상적으로 apply된다.

2. Terraform Module

Module Tree Structure

Root Module

dev

- └ main.tf **azs = var.az**
- └ variables.tf **az**
- └ dev.tfvars **az = "ap-northeast-2a"**

prod

- └ main.tf
- └ variables.tf
- └ prod.tfvars

...

Child Module

modules

- └ **availability_zone = var.azs**
 - └ network

- └ main.tf

- └ variables.tf **azs**

- └ outputs.tf

- └ ec2

- └ main.tf

- └ variables.tf

- └ outputs.tf

...

2. Terraform Module

Terraform의 상태 관리: Backend Configuration

① S3 버킷에 state file을 관리하도록 S3 backend 구성

Terraform에서 제공하는 Backend: local, gcs, azurerm, consul, Terraform Cloud/Enterprise, etc...

○ 왜 S3 backend?

- AWS provider 사용할 시 인프라 관리하는 데에 S3 백엔드 구성이 편리
- 버전 관리 가능, 비용 효율적, 그리고 처음 시도해보기에 접근성 굳

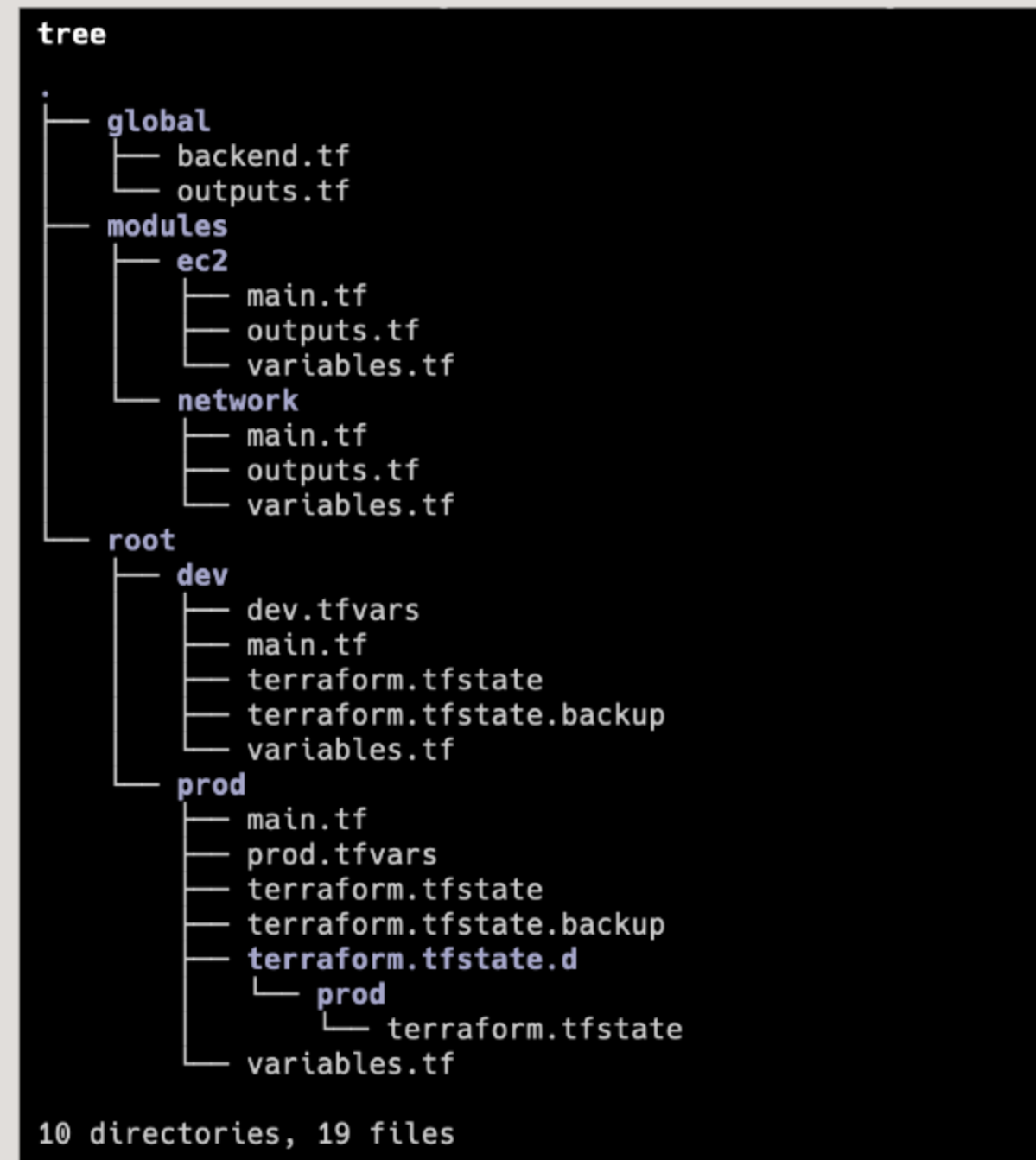
② DynamoDB로 State Locking Mechanism 구현 → 협업 시 상태의 일관성 유지



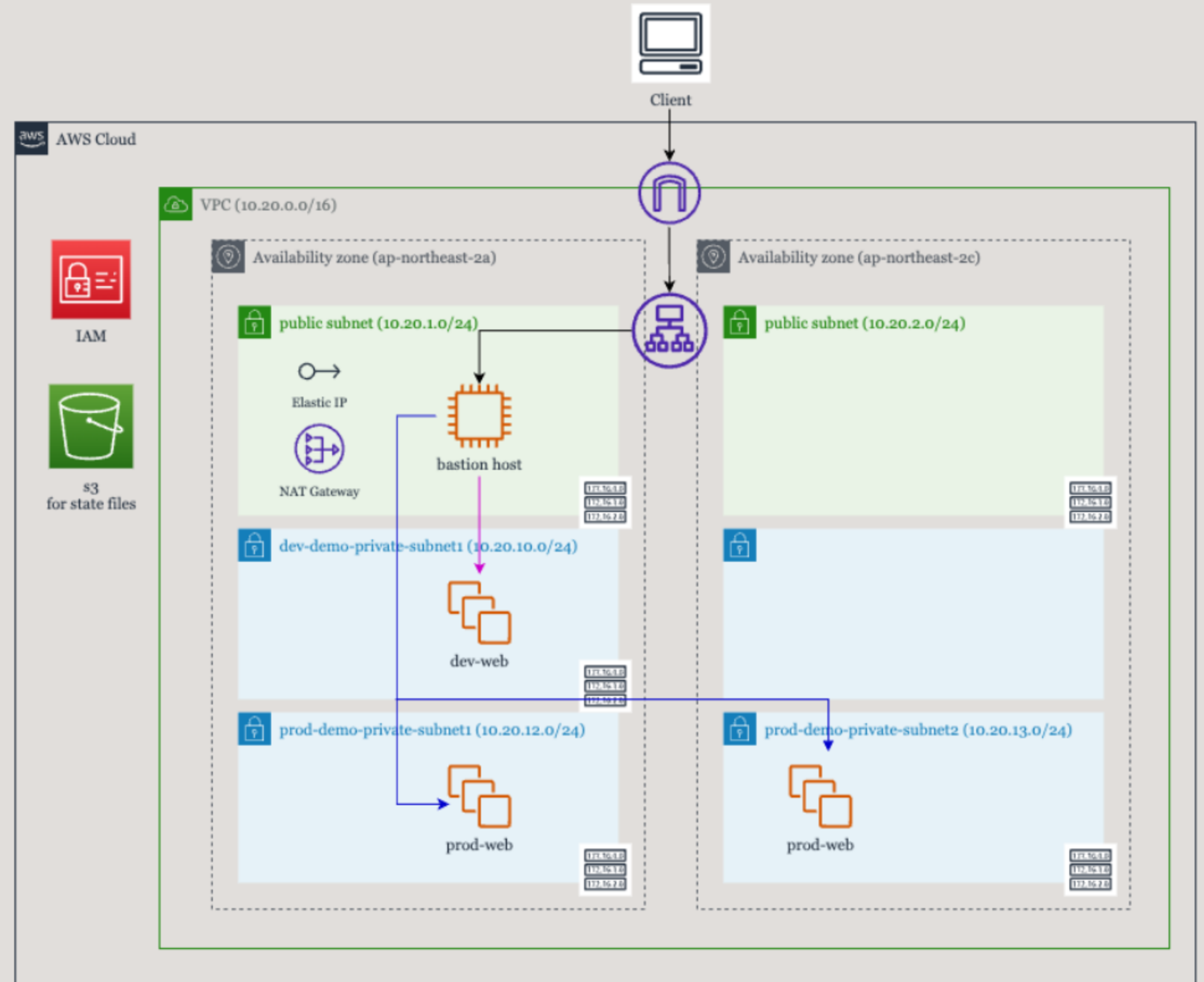
3. Live Demo



Tree Structure



Architecture



3. Live Demo

global/backend.tf

```
terraform {  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 4.0"  
    }  
  }  
}  
  
provider "aws" {  
  region = "ap-northeast-2"  
}
```

리소스를 프로비저닝을 하기 전에 다운받을 프로바이더와 버전 명시

3. Live Demo

dev.tfvars

```
# network
env = "dev"
tags = "demo"
vpc_cidr_block = "10.20.0.0/16"
az = ["ap-northeast-2a"]
az_tag = ["2a"]

# ec2
instance_type = "t2.micro"
type = "gp2"
size = 50
```

prod.tfvars

```
# network
env = "dev"
tags = "demo"
vpc_cidr_block = "10.20.0.0/16"
az = ["ap-northeast-2a", "ap-northeast-2a"]
az_tag = ["2a", "2c"]

# ec2
instance_type = "t2.micro"
type = "gp2"
size = 50
```

.tfvars 파일에 변수값을 다르게 지정하면 리소스 모듈을 재사용하여 프로비저닝할 수 있다!

행복한 테라폼~

맛있는 테라폼(?)~

Q & A

즐거운 테라폼~

재밌는 테라폼(?)~



feat. 안아줘요

AUSG 명예 안아줘요의 조언과 도움으로 탄생한 발표자료입니다 :)