HashiCorp

Terraform

코드로 인프라 관리하기

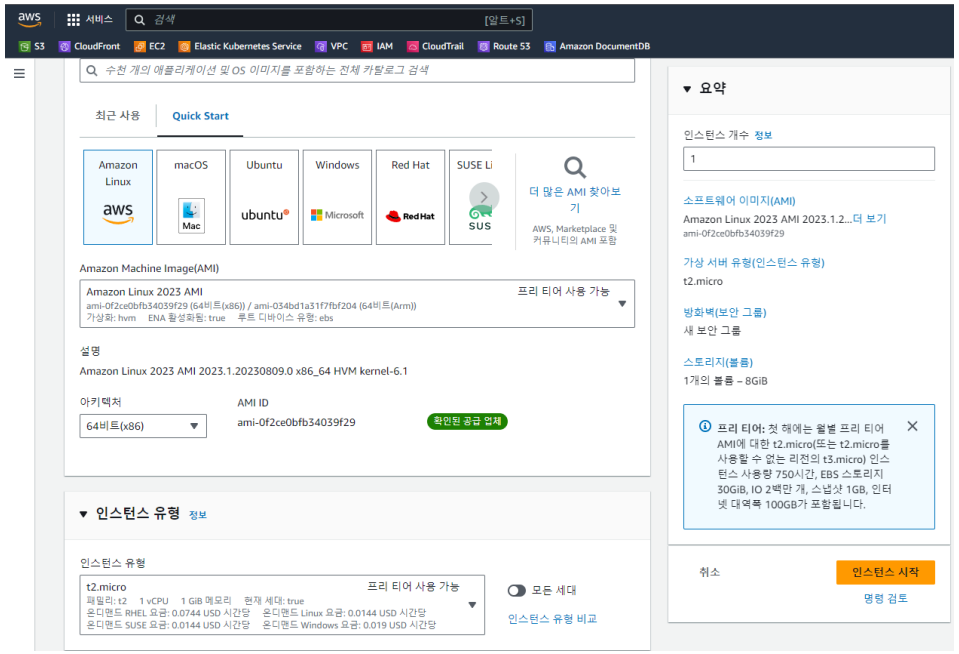
유 퀴즈 온 더 소마 팀

유임성

Infrastructure as Code

Infrastructure as Code, IaC → 코드로 인프라를 정의(선언) 한다.

인프라 일관성 유지 / 배포 용이 / 버전 관리 / 코드 그 자체



→

```hcl
resource "aws_instance" "my_ec2_instance" {
  ami                         = "ami-03221589fd7c8f183"
  instance_type               = "t2.micro"
  associate_public_ip_address = true

  tags = {
    Name = "soma"
  }
}
```

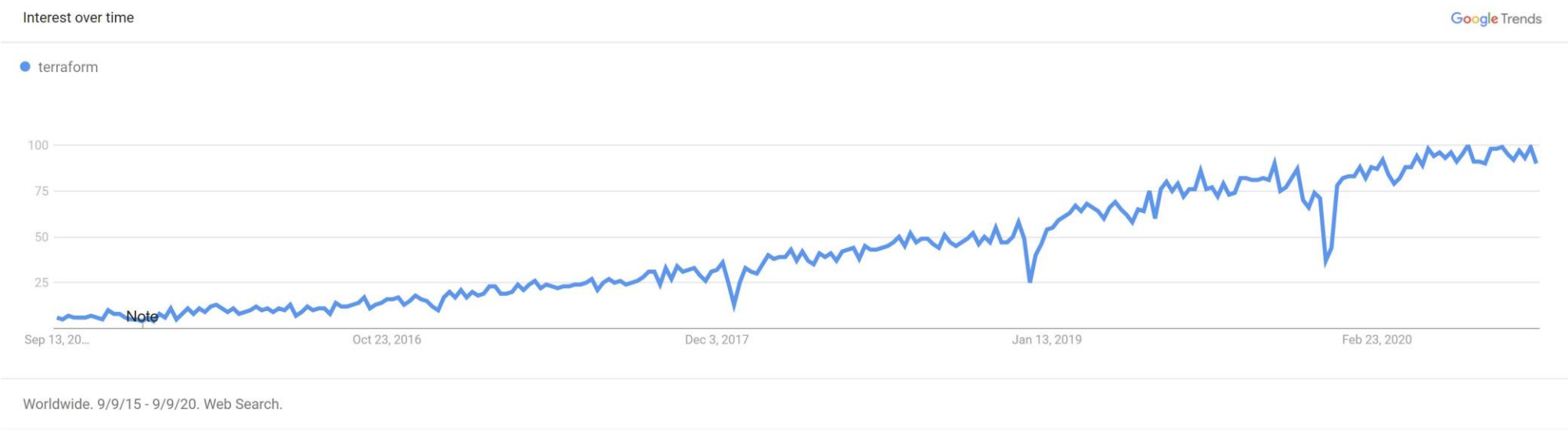
Terraform

Infrastructure as Code → 코드로 인프라를 관리 / 콘솔 대체

HCL → 인간 친화적 언어 / json / yaml

DRY! → Don’t Repeat Yourself / 같은 코드를 복붙하지 않기

HashiCorp



Terraform 장점

버전 관리 / 추적 → 코드로 남기에, 변경 사항을 쉽게 확인할 수 있음

문서화가 따로 필요 없다. 코드 그 자체가 문서

모듈 단위 구성 → 인프라 리소스를 모듈로 구성

재사용성 / 개발 환경 분리 / 외부 모듈 사용

충분히 성숙됨 → 레퍼런스가 풍부함 / 많은 기업들이 사용

좌충우돌 Terraform 입문기 - 우아한형제들 기술블로그

DevOps팀의 Terraform 모험 - 컬리 기술 블로그

Terraform Cloud – YOGIYO Tech Blog

여러 플랫폼 지원 → aws Google Cloud Azure kubernetes

Terraform 단점

러닝 커브가 높다

"코드로 인프라를 관리 / 콘솔 대체"는 좋은데.... How?

인프라 리소스들의 **정확한** 구성 요소와, 그들 간의 **의존성**을 알아야 함

ex) VPC => Subnet, CIDR, Nat GW, IGW, AZ ....

ex) S3 => IAM Role, Policy, User, ACL, ...

어쨌든 **새로운 언어**이다 보니 학습이 필요하다

모든것을 커버하긴 애매

IAM User, Password 같은 민감 정보

Security Group, Rule 같은 민감 정보

Auto Scaling Group 같이 동적으로 빠르게 변해야 하는 리소스

소마 프로젝트에 Terraform을 도입해야 하는 이유

백엔드 2명인 팀 → 보통은 한명이 대부분의 인프라 관리를 맡음

나머지 팀원들은 인프라에 대해 잘 이해하지 못함

지식 공유 → 코드를 통해 언제, 어떤 작업을 했는지 명확하게 볼 수 있음

나중에 다른 팀원들이 이해하기 쉬움

코드 리뷰 → 팀 내 개발문화에 자연스럽게 포함될 수 있음

프로젝트가 끝난 이후 → “11월 말에 계정 일괄 삭제”

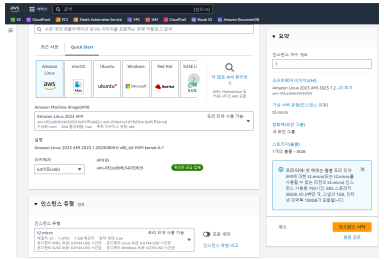
만약 프로젝트를 계속 진행하고 싶다면...?

하지만 이미 콘솔로 인프라 구성을 다 끝냈다면?

기존 인프라 구성

S3 EC2

콘솔로 구성



terraform import

Terraform이 관리

```hcl
resource "aws_instance" "my_ec2_instance" {
  ami                         = "ami-03221589fd7c8f183"
  instance_type               = "t2.micro"
  associate_public_ip_address = true

  tags = {
    Name = "soma"
  }
}
```

이제는 terraform이 관리

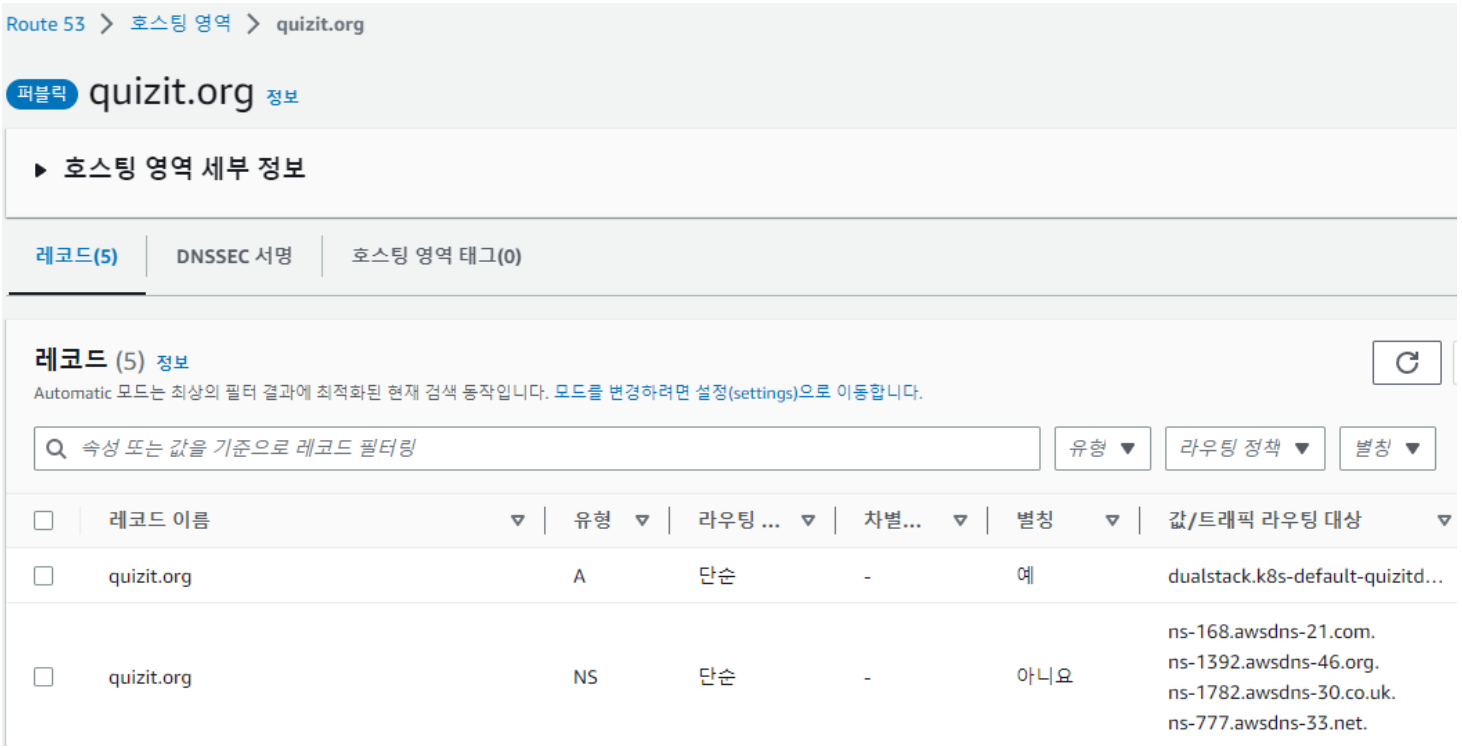
terraform.tfstate

Terraform 구성 요소

tfstate

- Terraform의 가장 핵심
- json 파일로 구성
- **Terraform이 생성한** 리소스들의 정보

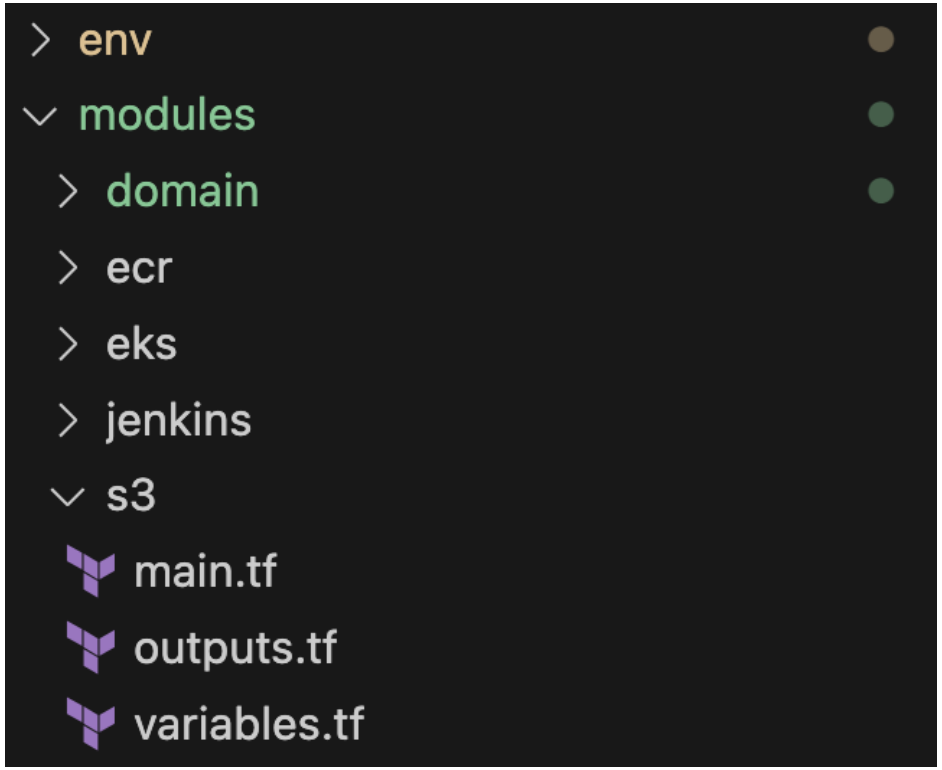

```
{
  "version": 4,
  "terraform_version": "1.5.3",
  "serial": 1,
  "lineage": "8789a55a-56f4-d159-0bd6-4cae0412932f",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_route53_zone",
      "name": "quizit",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "arn": "arn:aws:route53:::hostedzone/Z0112346CTZPZKKS432I",
            "comment": "HostedZone created by Route53 Registrar",
            "delegation_set_id": "",
            "force_destroy": null,
            "id": "Z0112346CTZPZKKS432I",
            "name": "quizit.org",
            "name_servers": [
              "ns-1392.awsdns-46.org",
              "ns-168.awsdns-21.com",
              "ns-1782.awsdns-30.co.uk",
              "ns-777.awsdns-33.net"
            ],
```

Terraform 구성 요소

module

- 우리에게 친숙한 모듈
- 재사용성
- 여러 tf 파일들이 모인 폴더
- 외부 모듈도 사용 가능



main

- 해당 모듈의 핵심
- 실질적인 리소스들의 모임

```
resource "aws_s3_bucket" "swagger" {
  bucket = "quizit-swagger"

  tags = var.tags
}
```

output

- 다른 모듈에서 참조할 변수

```
output "swagger_bucket_arn" {
  value = aws_s3_bucket.swagger.arn
}
```

variable

- 해당 모듈에서 필요한 변수

```
variable "tags" {
  type = map(string)
}
```

Terraform 명령어와 동작 방식

init

- tfstate 저장 위치 설정
- 모듈 초기화
- provider 설정

plan

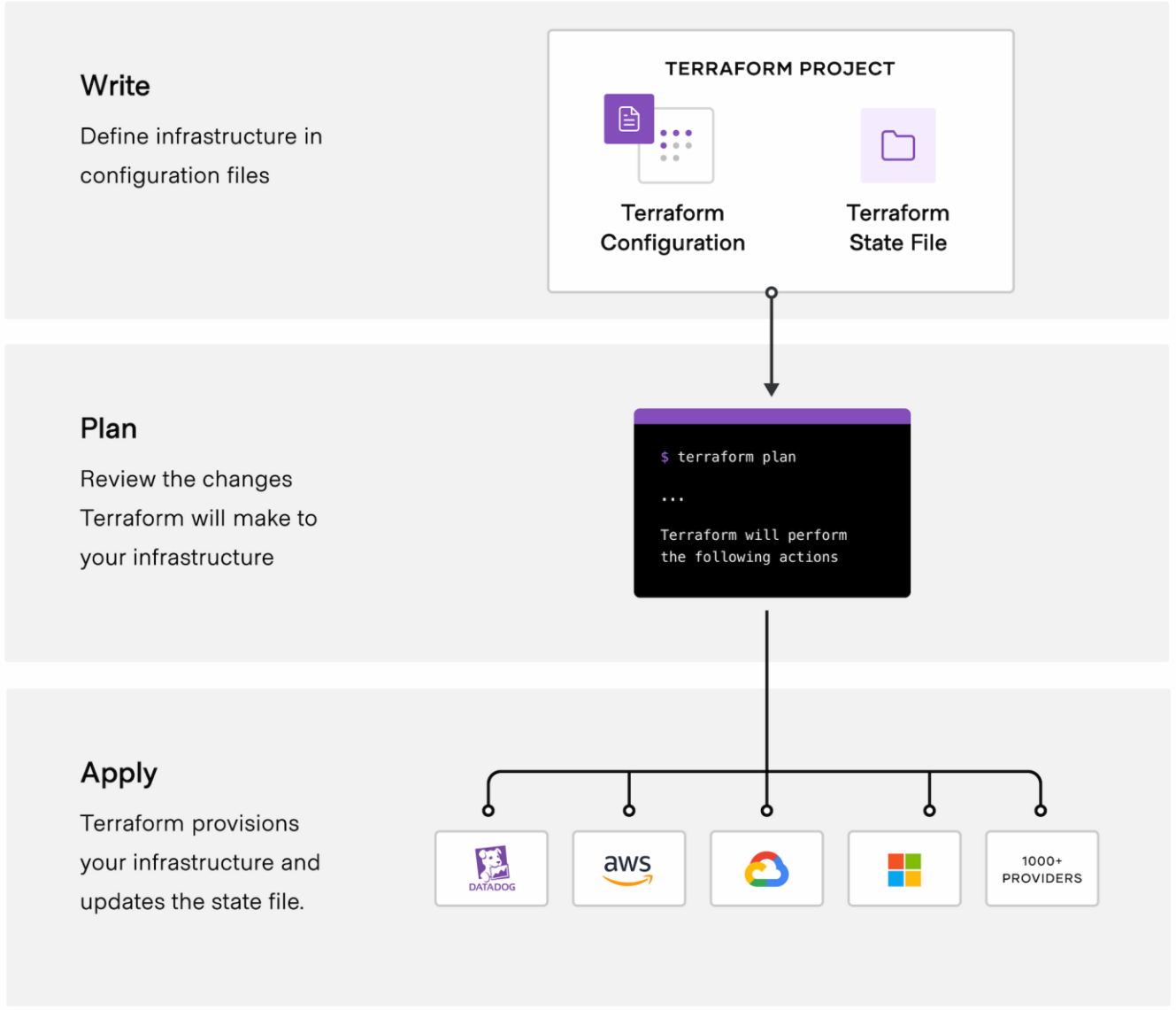
- 현재 코드와 tfstate 파일을 비교
- 실제로 생성될 리소스 출력
- dry-run

apply

- 실제로 리소스 생성
- tfstate 업데이트

show

- 지금까지 만든 리소스 출력



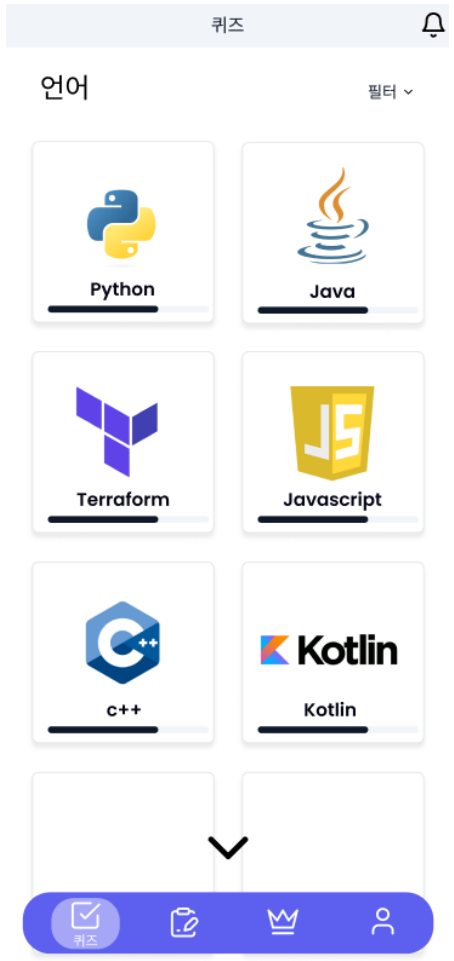
Terraform에 대해 아직 잘 모르겠다면?

Terraform 공부하고 싶은데 쉽지 않네,,,,,

개발 지식을 쉽게 떠먹여 주는 서비스 어디 없나,,,,

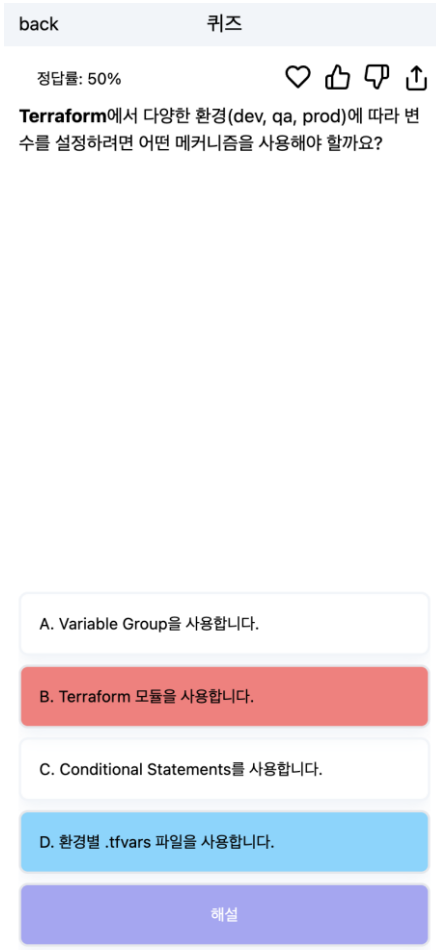
언제 어디서든 간단하게 퀴즈를 풀 수 있는 서비스 어디 없나,,,,,

QuizIt !



선택

채점



해설

감사합니다