

Terraform 기반 다중 웹 서비스 보안 인프라 구축

PROJECT REPORT

IaC 기반 하이브리드
클라우드 인프라 및
보안 솔루션 구축 과정 3기



메가스터디 IT 아카데미

PROJECT

REPORT

목차

01

프로젝트 개요

- 1.1 프로젝트 배경
- 1.2 프로젝트 목표

02

전체 인프라 아키텍처

- 2.1 전체 네트워크 구성도
- 2.2 네트워크 인프라 설계 구조

03

네트워크 및 인프라 설계

- 3.1 Terraform 기반 인프라 자동화 구조

04

보안 설계 (WAF)

- 4.1 AWS WAF 적용 목적
- 4.2 Managed RuleSet 구성

05

검증 결과

- 5.1 Auto Scaling 검증 결과
- 5.2 WAF 검증 결과

06

결론

- 6.1 보완 사항 및 결론

1.1 프로젝트 배경

클라우드 환경에서 다중 웹 서비스를 운영할 경우 SQL Injection, XSS와 같은 웹 공격 위협이 증가합니다.

이에 따라 보안과 가용성을 동시에 고려한 체계적이고 자동화된 보안 인프라 구축이 필요합니다.



다중 웹 서비스 환경의 보안 위협과 운영 한계

- 웹 애플리케이션 대상 SQL Injection, XSS 등 웹 기반 공격 증가
- 단일 서버 환경에서는 보안 대응 및 장애 복구에 한계 존재
- 수동 인프라 구성은 운영 오류 및 일관성 문제 발생
- 보안·가용성을 동시에 고려한 클라우드 기반 인프라 필요성 대두

1.2 프로젝트 목표

본 프로젝트는 Terraform을 활용하여 AWS 환경에서 다중 웹 서비스를 자동으로 구축하고, WAF와 Auto Scaling을 적용해 보안성과 가용성을 동시에 확보하는 것을 목표로 합니다.

Terraform 기반 AWS 인프라 자동 구축

- VPC, Subnet, Security Group 등 코드 기반 구성
- 동일한 환경을 안정적으로 재구성 가능한 구조 확보

다중 웹 서비스 배포 및 관리

- DVWA, GNUBoard, WordPress 자동 배포
- 서비스별 독립 구성
- 공통 인프라 기반 통합 운영

Application Load Balancer 와 WAF 연동 보안 구조 구현

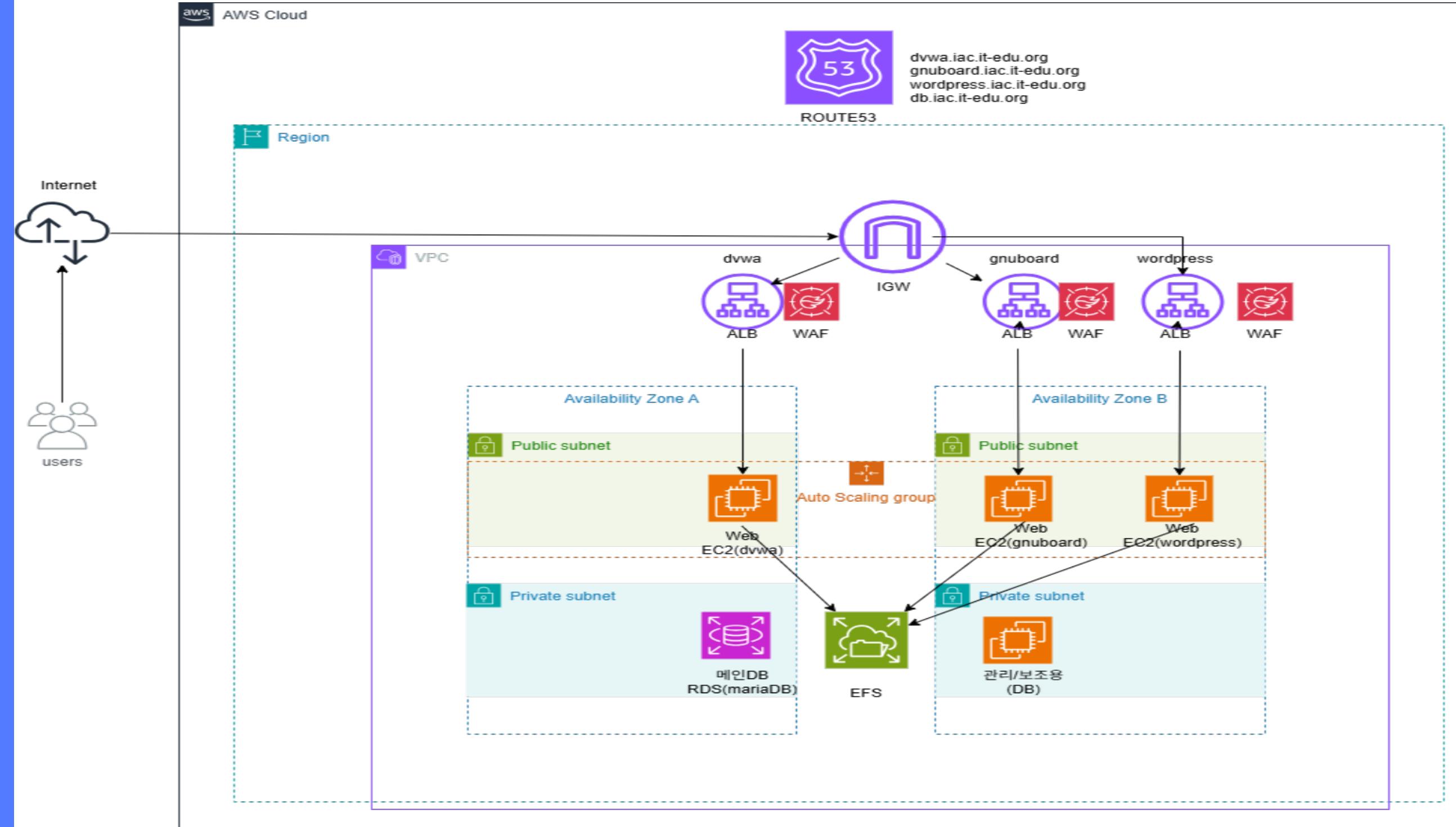
- ALB에 WAF 연동
- SQL Injection, XSS 공격 차단 검증
- 서비스별 Managed RuleSet 적용

Auto Scaling을 통한 장애 대응 및 가용성 확보

- 인스턴스 상태 감지를 통한 자동 교체 및 복구
- 서비스 중단을 최소화

2.1 전체 네트워크 구성도

Terraform으로 구현된 본 아키텍처는 보안과 가용성을 동시에 고려하여 다중 웹 서비스를 안정적으로 운영하도록 설계되었습니다.



AWS 기반 다중 웹 서비스 보안 인프라 아키텍처

- VPC는 퍼블릭·프라이빗 서브넷으로 분리하여 웹 서버를 프라이빗 서브넷에 배치
- ALB는 퍼블릭 서브넷에 위치하며 WAF와 연동되어 외부 트래픽을 사전 차단
- Auto Scaling Group을 통해 DVWA, GNUBoard, WordPress 서비스를 독립적으로 운영
- 각 서비스는 개별 확장 및 장애 대응이 가능하도록 설계

2.2 네트워크 인프라 설계 구조

Public/Private Subnet 분리와 IGW 라우팅으로 외부 접근 흐름을 구성하였다.

Public 영역(ALB)에서 트래픽을 받고, 서비스는 Private Subnet에 분리 배치했습니다



Subnet 분리

- Public / Private 영역 분리

Route Table 구조

- Public 1개 / Private 2개 분리 운영

Internet Gateway

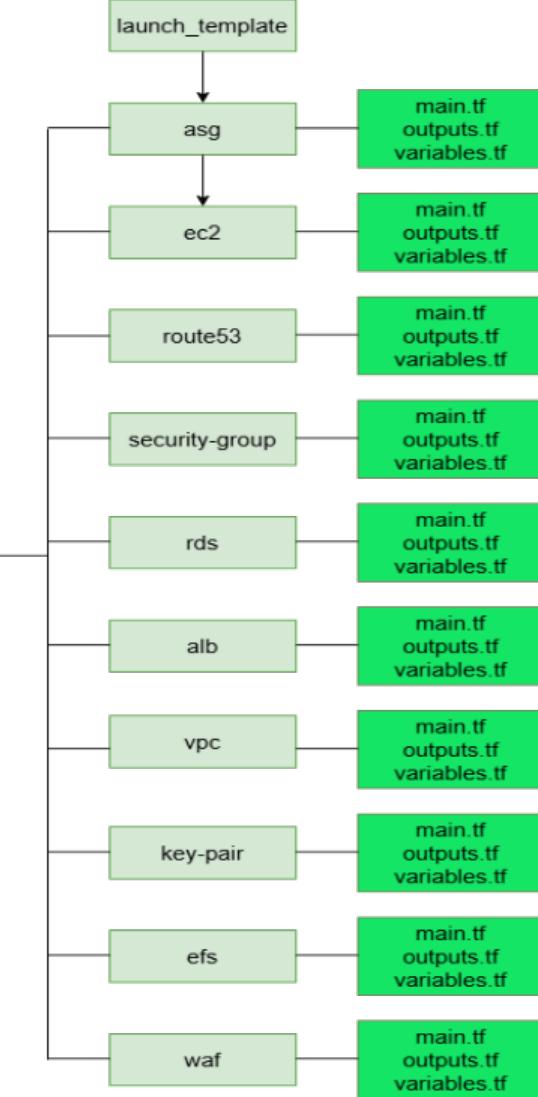
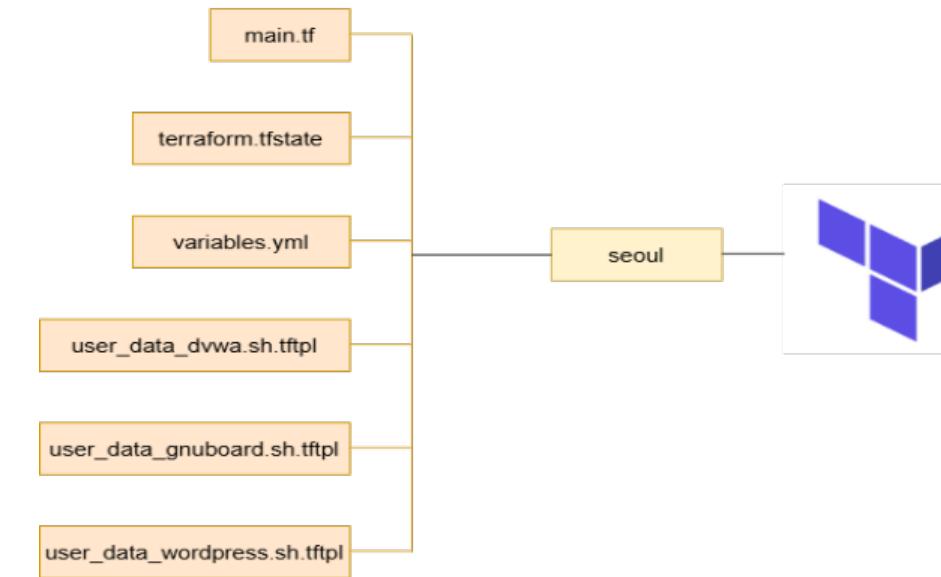
- 외부 트래픽 진입 지점 통제

보안 관점 설계

- 서비스는 Private Subnet 배치

3.1 Terraform 기반 인프라 자동화 구조

Terraform 모듈화 구조로 VPC부터
WAF, Auto Scaling, RDS/EFS까지
인프라를 코드로 통합 관리하였다.



Terraform 구조

- 모듈화: 서비스/기능 단위로 재사용 가능한 구조(VPC, ALB, ASG, WAF, RDS 등)
- 재현성: 동일 환경을 반복 배포할 수 있어 변경 이력 및 구성 일관성 확보
- 자동화: user_data 템플릿으로 웹 서버 초기 설정 자동화

4.1 AWS WAF 적용 목적

본 프로젝트는 ALB 앞단에 AWS WAF를 적용하여 DVWA, GNUMBoard, WordPress 웹 서비스에 대한 SQL Injection, XSS 등 주요 웹 공격을 사전에 차단하고, 서비스 가용성과 보안을 동시에 확보하는 것을 목표로 설계되었습니다.

보호 팩(웹 ACL) (3) 정보

보호 팩(웹 ACL)을 사용하면 위협으로부터 인프라를 보호하는 규칙과 규칙 세트를 결합할 수 있습니다.

The screenshot shows the AWS WAF console interface. At the top right, there are buttons for '작업' (Work), '보호 팩(웹 ACL) 생성' (Create Protected Web ACL), and other navigation options. Below the header, there's a search bar and a dropdown menu set to '리전 범위 CloudFront(글로벌) 및 리전별'. The main area displays a table with three rows, each representing a protected rule set:

ID	ARN	범위	로깅 및 지표	샘플링된 요청	대시보드	리소스	규칙	이름
bd850013-367a-471e-9501-0f89311f1630	arn:aws:wafv2:...:bd850013-367a-471e-9501-0f89311f1630	리전별	활성화하지 않음	보기	보기	관리	3개 규칙	seoul-waf-dvwa
2125c474-aab7-493a-b965-726da23a90a5	arn:aws:wafv2:...:2125c474-aab7-493a-b965-726da23a90a5	리전별	활성화하지 않음	보기	보기	관리	3개 규칙	seoul-waf-gnuboard
6c93d807-657b-42ed-a65d-348ee57c702a	arn:aws:wafv2:...:6c93d807-657b-42ed-a65d-348ee57c702a	리전별	활성화하지 않음	보기	보기	관리	4개 규칙	seoul-waf-wordpress

WAF 적용 목적

- 웹 공격 사전 차단 구조 구성
- 서비스 공통 보안 정책 적용
- SQL Injection 등 웹 취약점 대응

4.2 Managed RuleSet 구성

본 프로젝트는 AWS Managed RuleSet을 활용하여 별도의 룰 개발 없이도 표준화된 웹 보안 정책을 적용하고, 실시간 공격 탐지 및 차단이 가능하도록 구성하였습니다.

seoul-waf-dvwa에 대한 규칙 관리

< seoul-waf-dvwa

규칙은 규칙 우선 순위 오름차순으로 표시됩니다.

rate-limit

2 WCU >

AWSManagedRulesCommonRuleSet

700 WCU >

AWSManagedRulesSQLRuleSet

200 WCU >

규칙 추가

Managed RuleSet 구성

- Common Web Attack RuleSet 구성
- SQL Injection RuleSet 구성

5.1 Auto Scaling 검증 결과

EC2 인스턴스의 웹 서비스를 중단하여
Auto Scaling 기반 장애 감지 및 복구 동
작을 유도하였다.

seoul-dvwa-alb-tg

세부 정보

arn:aws:elasticloadbalancing:ap-northeast-2:279791963612:targetgroup/seoul-dvwa-alb-tg/a9d5c46fcda7849e

대상 유형 인스턴스	프로토콜 : 포트 HTTP: 80	프로토콜 버전 HTTP1	VPC vpc-0e85d06aaa05e7115 ↗		
IP 주소 유형 IPv4	로드 밸런서 seoul-dvwa-alb ↗				
2 대상 합계	❶ 0 정상 0 이상	❷ 0 비정상	❸ 0 사용되지 않음	❹ 1 초기	❺ 1 드레이닝

▶ 가용 영역별 대상 배포
아래의 등록된 대상 테이블에 적용된 해당 필터를 보려면 이 테이블에서 값을 선택합니다.

대상 모니터링 상태 검사 속성 태그

등록된 대상 (2) 정보

대상 그룹은 지정한 프로토콜 및 포트 번호를 사용하여 등록된 개별 대상으로 요청을 라우팅합니다. 상태 확인은 대상 그룹의 상태 확인 설정에 따라 등록된 모든 대상에 대해 수행됩니다. 이상 탐지는 정상 대상이 3개 이상 있는 HTTP/HTTPS 대상 그룹에 자동으로 적용됩니다.

대상 필터링	인스턴스 ID	이름	포트	영역	상태 확인	상태 확인 세부 정보	관리상 재정의	재정의 세부 정보	시작 시간
<input type="checkbox"/>	i-0c0a6e65948e024a2	seoul-dvwa-asg	80	ap-northeast-...	❶ Initial	Target registration is in progress	❷ No override	No override is curre...	2026년 1...
<input checked="" type="checkbox"/>	i-0c494e23978f2df2b	seoul-dvwa-asg	80	ap-northeast-...	❸ Draining	Target deregistration is in progr...	❹ No override	No override is curre...	2026년 1...

Auto Scaling 장애 발생 단계

- DVWA 인스턴스 웹 서비스 중단 발생
- ALB Target Group 상태가 Draining로 전환
- Auto Scaling이 비정상 인스턴스를 감지하는 단계

5.1 Auto Scaling 검증 결과

EC2 인스턴스의 웹 서비스를 중단하여
Auto Scaling 기반 장애 감지 및 복구 동
작을 유도하였다.

seoul-dvwa-alb-tg

세부 정보

arn:aws:elasticloadbalancing:ap-northeast-2:279791963612:targetgroup/seoul-dvwa-alb-tg/a9d5c46fcda7849e

대상 유형 인스턴스	프로토콜 : 포트 HTTP: 80	프로토콜 버전 HTTP1	VPC vpc-0e85d06aaa05e7115
IP 주소 유형 IPv4	로드 밸런서 seoul-dvwa-alb		
2 대상 합계	❶ 1 정상	❷ 0 비정상	❸ 0 사용되지 않음
	0 이상		
			❹ 0 초기
			❺ 1 드레이닝

▶ 가용 영역별 대상 배포
아래의 등록된 대상 테이블에 적용된 해당 필터를 보려면 이 테이블에서 값을 선택합니다.

대상 | 모니터링 | 상태 검사 | 속성 | 태그

등록된 대상 (2) 정보

① 이상 원화: 해당되지 않음 등록 취소 대상 등록

대상 그룹은 지정한 프로토콜 및 포트 번호를 사용하여 등록된 개별 대상으로 요청을 라우팅합니다. 상태 확인은 대상 그룹의 상태 확인 설정에 따라 등록된 모든 대상에 대해 수행됩니다. 이상 탐지는 정상 대상이 3개 이상 있는 HTTP/HTTPS 대상 그룹에 자동으로 적용됩니다.

대상 필터링	인스턴스 ID	이름	포트	영역	상태 확인	상태 확인 세부 정보	관리상 재정의	재정의 세부 정보	시작 시간
<input type="checkbox"/>	i-0c0a6e65948e024a2	seoul-dvwa-asg	80	ap-northeast-...	<input checked="" type="radio"/> Healthy	-	<input type="radio"/> No override	No override is curre...	2026년 1...
<input checked="" type="checkbox"/>	i-0c494e23978f2df2b	seoul-dvwa-asg	80	ap-northeast-...	<input type="radio"/> Draining	Target deregistration is in progr...	<input type="radio"/> No override	No override is curre...	2026년 1...

Auto Scaling 복구 완료 단계

- Auto Scaling이 신규 EC2 인스턴스를 자동 생성
- ALB Target Group 상태가 Healthy로 전환됨

5.1 Auto Scaling

검증 결과

EC2 인스턴스의 웹 서비스를 중단하여
Auto Scaling 기반 장애 감지 및 복구 동
작을 유도하였다.

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are both documented and undocumented vulnerabilities with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! Do not upload it to your hosting provider's public html folder or any Internet facing servers, as they will be compromised. It is recommend using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- [Mutillidae](#)
- [OWASP Vulnerable Web Applications Directory](#)

You have logged in as 'admin'

Auto Scaling 접속 성공 단계

- DVWA 서비스 정상 복구 확인

5.2 WAF 검증 결과

WAF 규칙 적용 후, SQL Injection 공격 등을 시도하여, 비정상 요청이 정상적으로 차단되는지를 검증하였다.

seoul-waf-dvwa에 대한 규칙 관리

< [seoul-waf-dvwa](#)

규칙은 규칙 우선 순위 오름차순으로 표시됩니다.

rate-limit

2 WCU >

AWSManagedRulesCommonRuleSet

700 WCU >

AWSManagedRulesSQLiRuleSet

200 WCU >

[규칙 추가](#)

WAF 규칙 구성 단계

- Common / SQLi Managed RuleSet 적용
- 비정상 트래픽 사전 차단 정책 구성

5.2 WAF 검증 결과

WAF 규칙 적용 후, SQL Injection 공격 등을 시도하여, 비정상 요청이 정상적으로 차단되는지를 검증하였다.

```
PS C:\Users\dg6356> cd "C:\Users\dg6356\Desktop\terraform real"
PS C:\Users\dg6356\Desktop\terraform real> python test_waf_dvwa.py
=====
[DVWA WAF 테스트 스크립트]
=====
대상 URL: http://dvwa.iac.it-edu.org/dvwa
시작 시간: 2026-01-10 15:25:59
=====

[테스트 1] SQL Injection 공격 시작...
=====
[SQL Injection: ' OR '1'='1...] [BLOCKED]
=====
Status Code: 403
Response Time: 0.38s
URL: http://dvwa.iac.it-edu.org/dvwa/vulnerabilities/sqli/?id='%20OR%20'1'='1&id=%27+OR+%271%27%3D%271&username=%27+OR+%271%27%3D%271&password=%27+OR+%271%27%3D%271
Response Length: 118 bytes
=====

=====
[SQL Injection: ' OR 1=1--...] [BLOCKED]
=====
Status Code: 403
Response Time: 0.01s
URL: http://dvwa.iac.it-edu.org/dvwa/vulnerabilities/sqli/?id='%20OR%201=1--&id=%27+OR+1%3D1--&username=%27+OR+1%3D1--&password=%27+OR+1%3D1--
Response Length: 118 bytes
=====

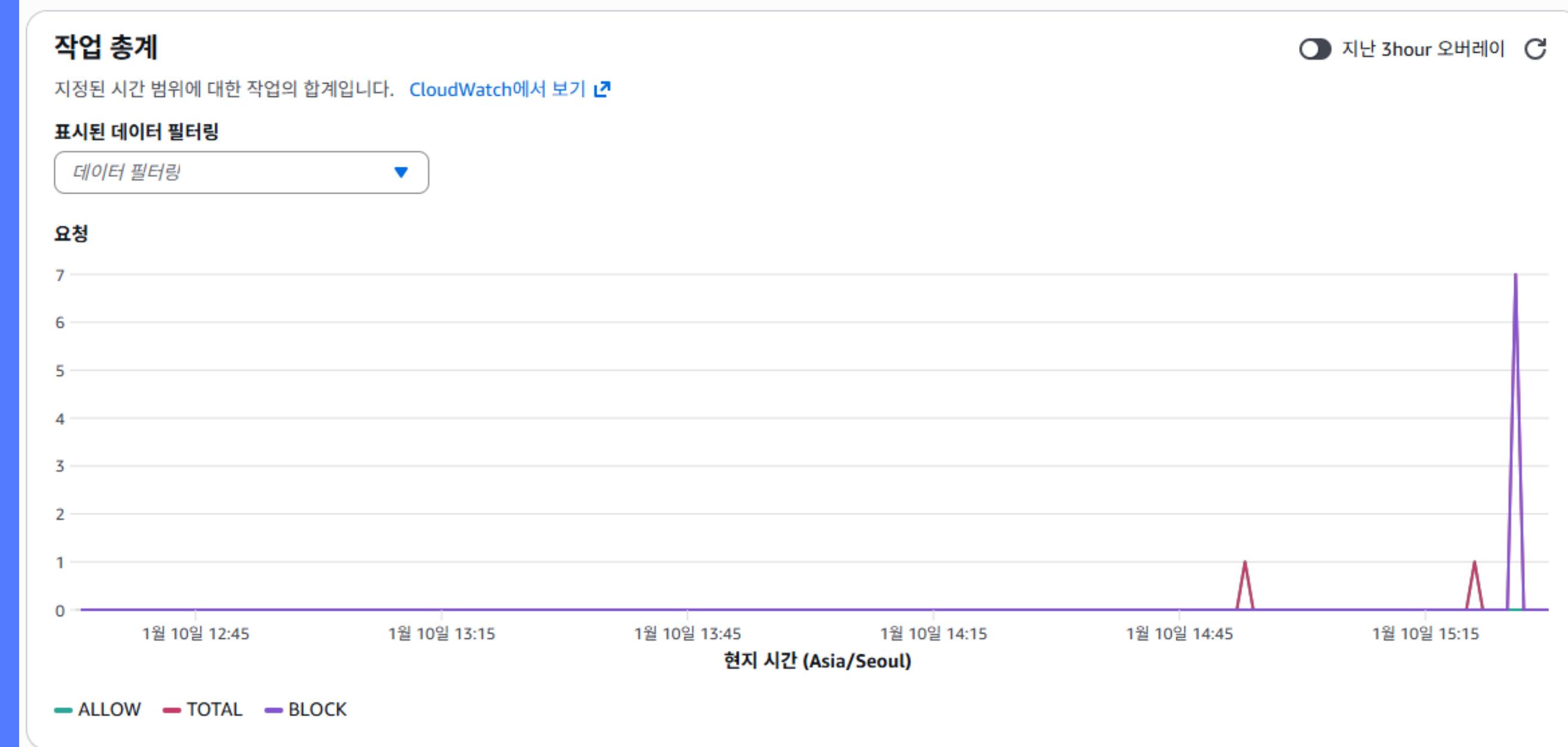
=====
[SQL Injection: ' UNION SELECT NULL--...] [BLOCKED]
=====
Status Code: 403
Response Time: 0.01s
URL: http://dvwa.iac.it-edu.org/dvwa/vulnerabilities/sqli/?id='%20UNION%20SELECT%20NULL--&id=%27+UNION+SELECT+NULL--&username=%27+UNION+SELECT+NULL--&password=%27+UNION+SELECT+NULL--
Response Length: 118 bytes
=====
```

공격 시도 단계

- SQL Injection 공격
- Common 공격
- WAF에 의해 요청이 HTTP 403으로 차단

5.2 WAF 검증 결과

WAF 규칙 적용 후, SQL Injection 공격 등을 시도하여, 비정상 요청이 정상적으로 차단되는지를 검증하였다.



차단 결과 단계

- BLOCK 지표 증가로 보안 정책 동작 확인

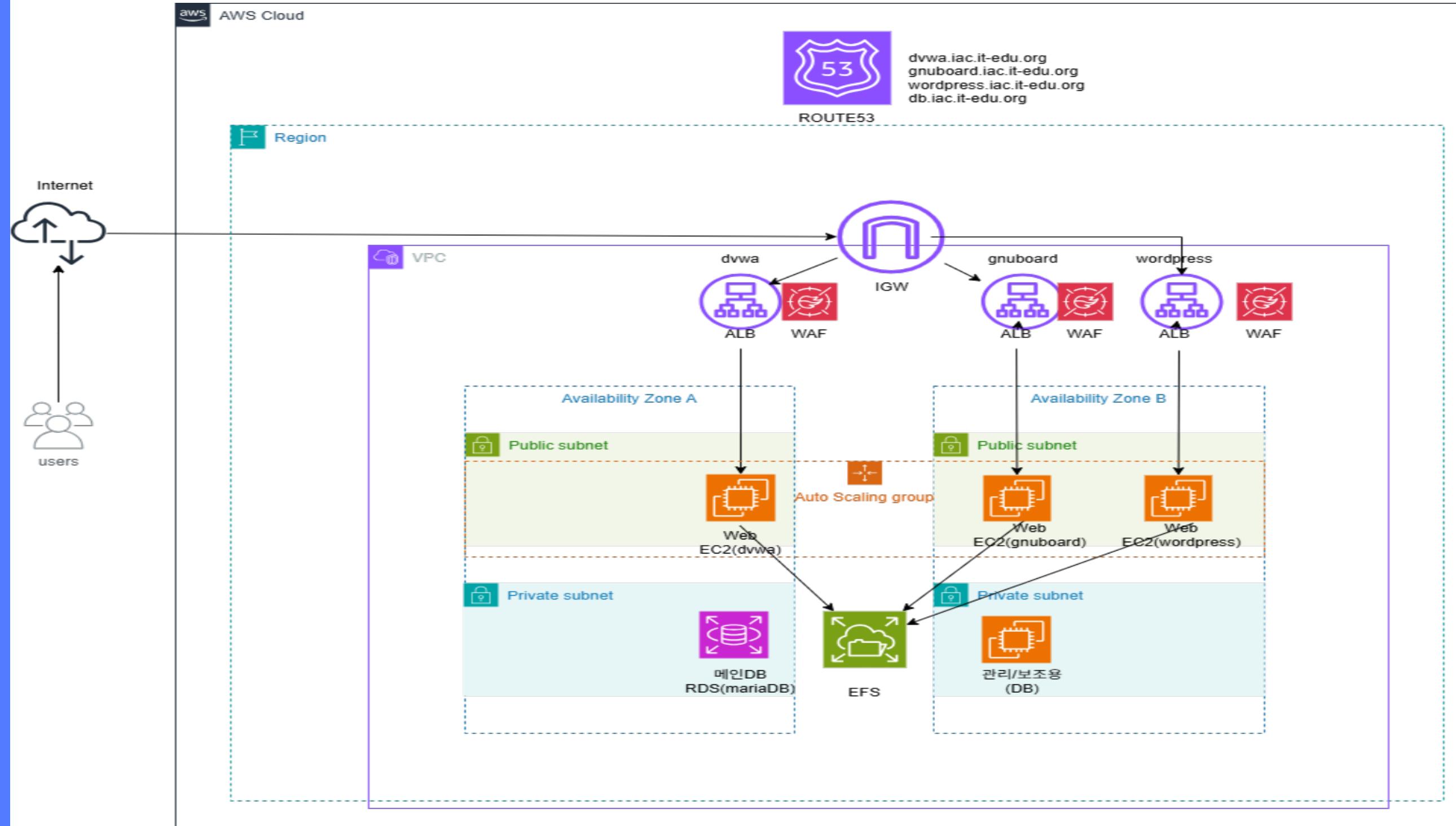
보완 사항 및 결론

본 프로젝트를 통해 Terraform 기반 AWS 다중 웹 서비스 인프라와 WAF, Auto Scaling을 적용한 보안·가용성 구조를 성공적으로 구현하였다.

- WAF 고도화: Managed RuleSet 기반 차단 검증 완료, 서비스 특성에 맞는 Custom Rule 확장 가능
- IAM 및 인프라 코드화: 권한 및 리소스를 IaC로 관리하여 구성 일관성과 재현성 확보
- 가용성 검증: Auto Scaling을 통한 장애 자동 감지 및 서비스 무중단 복구 확인
- 운영 관점 개선: 수동 구축 대비 배포 속도 향상 및 운영 리스크 감소
- 향후 확장: 보안 정책, 자동화 범위를 확장하여 실무 적용 가능한 클라우드 보안 인프라로 발전 가능

Terraform 구조

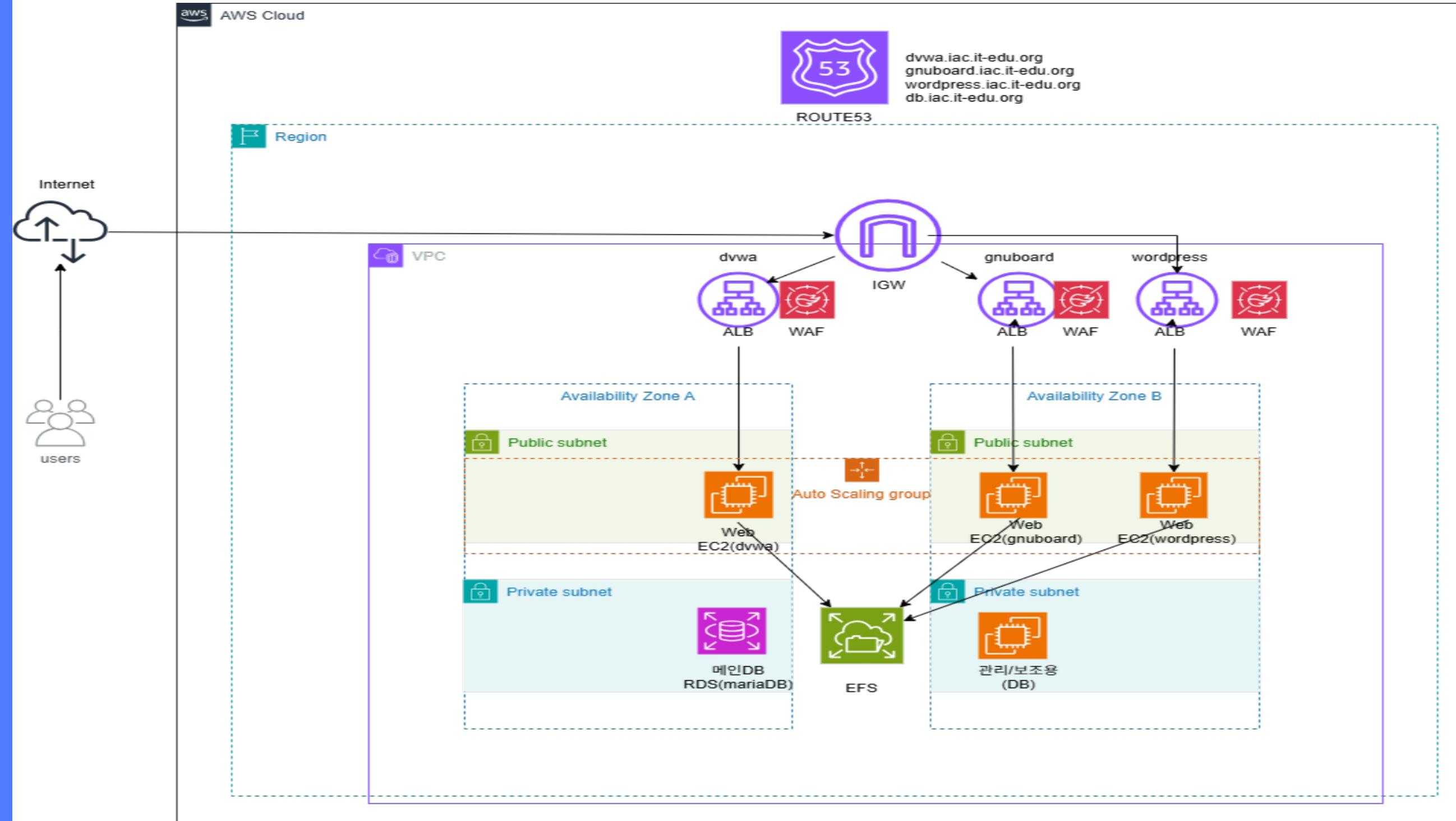
본 프로젝트는 Terraform을 활용하여 AWS 인프라를 코드 기반으로 관리하고, 동일한 환경을 반복적으로 안정적으로 배포할 수 있도록 구성하였습니다.



- VPC, ALB, Auto Scaling, WAF, RDS, EFS 모듈화 구성
- Root(seoul)에서 전체 리소스 중앙 관리
- user_data.tfpl 기반 웹 서버 초기 설정 자동화
- Terraform 코드 기반으로 인프라 구성 일관성 유지

IaC 적용 효과

수동 설정이 아닌 IaC 방식을 적용하여 배포 오류를 최소화하고, 인프라 변경 이력 관리 및 재현성을 확보하였습니다.



- 동일 환경 재구성 가능
- 설정 누락 및 운영 오류 감소
- 인프라 변경 시 코드 기반 추적 가능
- 운영 표준화 및 관리 효율 향상

Terraform 기반 AWS 다중 웹 서비스 인프라 설계

본 프로젝트는 Terraform을 활용하여 DVWA, GNUBoard, WordPress 다중 웹 서비스를 독립적이면서도 공통 인프라 기반으로 운영할 수 있도록 설계하였습니다.

각 서비스는 장애 및 트래픽 증가 상황에서도 안정적으로 운영될 수 있도록 구성되었습니다.

다중 웹 서비스 인프라 설계

- DVWA, GNUBoard, WordPress 서비스를 각각 독립된 구성으로 배포
- 서비스별 Application Load Balancer(ALB) 구성
- 각 웹 서비스는 개별 Auto Scaling Group으로 운영
- 공통 VPC 및 네트워크 자원 기반의 통합 인프라 구조

고가용성 및 확장성 고려 설계

- 다중 Availability Zone(AZ)에 인스턴스 분산 배치
- Auto Scaling을 통해 트래픽 증가 및 인스턴스 장애 시 자동 대응
- 서비스 간 영향 없이 개별 확장 및 관리 가능

데이터 및 공통 리소스 구성

- 데이터베이스는 Private Subnet에 배치하여 외부 접근 차단
- Amazon EFS를 활용하여 웹 서버 간 공통 데이터 공유
- 인프라 변경 시 Terraform 코드 기반으로 일관성 유지