Hannctl.

hannctl.com

웹 포트폴리오



Public Cloud Infrastructure & Security Engineer

김준현 Kim JunHyun

사람의 기억에 기대던 인프라와 보안을, 코드가 지키는 구조로 바꿉니다

2020~	4	3,500+	20+	15
클라우드 커리어 시작	CSP 운영	최대 운영 인프라 규모	서비스 프로젝트	자격증 취득

kimjhtlzk@naver.com 010-9092-0615 github.com/kimjhtlzk linkedin.com/in/ihanni hann-blume.tistory.com

요약

게임 서비스부터 금융권까지 AWS·GCP·Azure·Tencent 4개 클라우드(CSP)에서 최대 서버 3,500대·DB 650개 규모의 인프라를 운영했습니다. Terragrunt 전환으로 배포 시간을 40% 줄였고, CSPM(클라우드 보안 설정 점검) 조치율을 60%에서 85% 이상으로 올렸던 경험을 가지고 있습니다.

핵심 역량

IaC 자동화	Terraform·Terragrunt로 인프라를 코드로 정의합니다
멀티클라우드 운영	4개 CSP를 함께 설계하고 운영합니다
클라우드 보안	탐지·대응, 최소 권한, 감사 대응을 맡습니다
운영 자동화	Python·PowerShell로 반복 업무를 없앱니다
AI 에이전트	역할과 권한이 있는 멀티 에이전트를 만듭니다

경력

우리FIS

2026.02 – 2026.10

정보보안부
대리 · Cloud Security Engineer

우리은행·우리카드·우리금융지주 등 계열사의 하이브리드 환경(퍼블릭·프라이빗 클라우드, 온프레미스)에서 클라우드 보안 아키텍처를 설계·구축·운영했습니다.

- 우리금융그룹에 없던 클라우드 보안 정책 정의서를 입사 후 직접 설계·완성해 그룹사에 배포
- Terraform이 없던 환경에 AWS Network Firewall·WAF·Shield 보안 인프라를 코드로 처음 도입
- 다중 AWS 계정의 네트워크 통신 구성을 자동 수집·분석하는 Python 파이프라인 구축
- GuardDuty·Security Hub·WAF·Network Firewall·Shield를 Splunk 기반 SIEM/SOAR와 연동
- 전자금융감독규정 망분리 요건과 Zero Trust 원칙을 반영한 IAM 권한 체계를 Terraform으로 정의해 과도한 권한 부여를 구조적으로 차단
- 하이브리드 보안 솔루션 통합 운영: FortiGate, NxPortrait, WIPS
- 금융권 퍼블릭 클라우드 도입 초기 멤버로 보안 정책 수립 참여

컴투스플랫폼

2023.02 – 2026.02

클라우드본부 인프라기술실
대리 · Cloud Engineer

게임 서비스를 위한 AWS·GCP·Azure·Tencent 4개 CSP 멀티클라우드 인프라의 설계·구축·운영·장애 대응을 총괄한 메인 엔지니어입니다.

- Terraform → Terragrunt 전환으로 배포 시간 40% 단축
- GCP → Tencent Cloud 무중단 이관 (서버 80대·DB 20개, 1주일 내 완료)
- CSPM(클라우드 보안 설정 점검) 조치율 60% → 85% 이상
- 드래곤스카이 인프라 비용 최대 62% 절감
- Tencent·GCP 서버 600대·DB 70개 규모의 클라우드 간 방화벽 정책을 전수 분석해 재설계
- 4개 CSP 자산·감사 로그를 공통 스키마로 수집하는 Python 점검 자동화. 권한 부여, Access Key 발급, 주요 서버·DB 변경은 Slack·Telegram으로 즉시 알림
- 4개 CSP 감사 대응: 반복 지적은 Terraform 모듈 기본값과 CI/CD의 validate·plan 필수 게이트로 이전
- Zabbix·Grafana·Slack·Telegram 통합 알림, GCP Cloud Armor로 DDoS 방어 운영

씨앤토트

2020.08 – 2023.02

Cloud 서비스팀
사원~대리 · Cloud Engineer (Azure MSP)

Azure 매니지드 서비스(MSP) 기업에서 게임사 고객을 전담한 메인 엔지니어로 설계·구축·운영·보안 감사·배포 자동화를 맡았습니다.

- Mir4 Global(서버 3,500대·DB 650개) 전담 운영을 비롯해 국내외 여러 게임 서비스의 Azure 인프라 운영
- Azure Log Analytics + PowerShell DDoS 자동 탐지·차단
- PowerShell 배포 자동화로 배포 속도 80% 개선 (Mir4 Korea)
- Dedicated Host 기반 고가용성 구조와 실시간 장애 대응
- Telegraf·InfluxDB·Grafana(TIG 스택) 실시간 대시보드 구축
- CSPM·ISMS 기반 고객사 클라우드 보안 감사 대응

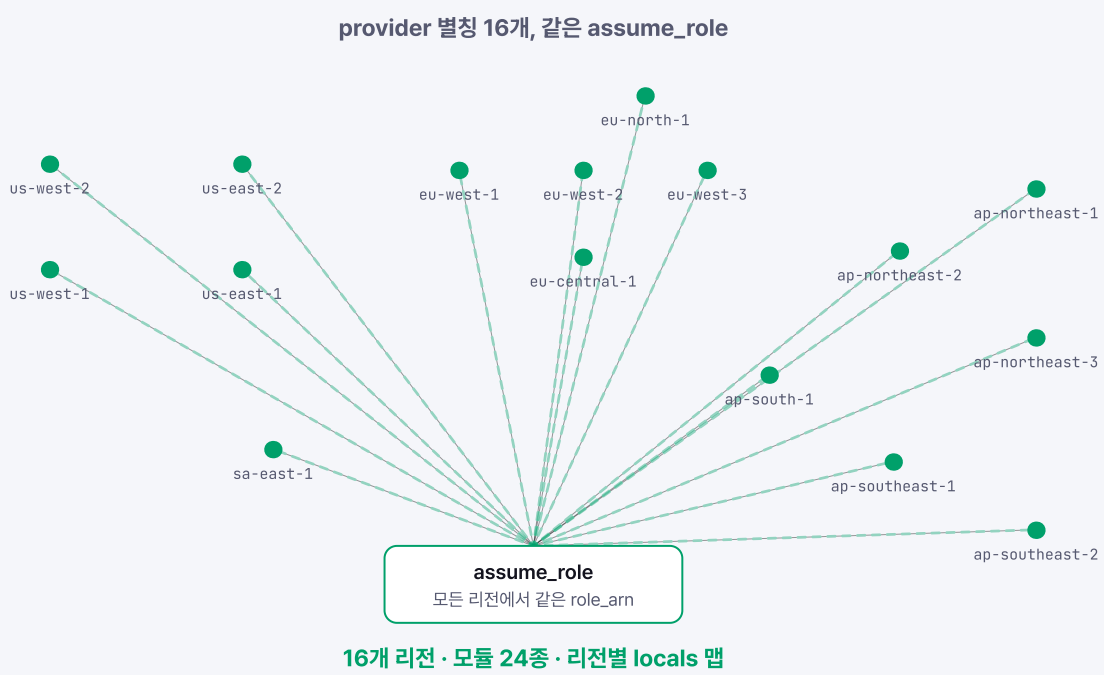
AWS 멀티 리전 Terraform

리전별 locals 맵 하나를 고치면 16개 AWS 리전에 같은 인프라가 만들어지는 Terraform 모듈 모음입니다.

16
리전별 provider 별칭

24
공개한 Terraform 모듈

3
검증한 예제



16개 리전 · 모듈 24종 · 리전별 locals 맵

문제

같은 형태의 리소스를 여러 리전에 만들어야 했습니다. 리전이 늘수록 복사한 코드가 늘고 리전마다 설정이 달라질 위험이 커졌습니다.

접근

리전별 locals 맵에 원하는 상태를 선언하고 module과 for_each로 일괄 생성합니다. provider는 리전마다 별칭을 두고 같은 role을 assume합니다. for_each가 provider를 동적으로 고를 수 없어 리전별 module 블록은 반복합니다.

결과

변경은 맵 항목 하나로 끝나고, 키가 고정이라 순서를 바꿔도 리소스가 다시 생성되지 않습니다. 모듈 24종에 암호화, IMDSv2, 비공개 S3 같은 기본값을 넣었고 EKS와 알림 예제까지 3종을 공개했습니다.

설계 · 구현 · 개인 프로젝트 · 2026.11 공개 · Terraform, AWS, EKS, CloudWatch, AWS Chatbot

github.com/kimjhtlzk/pf-01-aws-terraform-multi-region

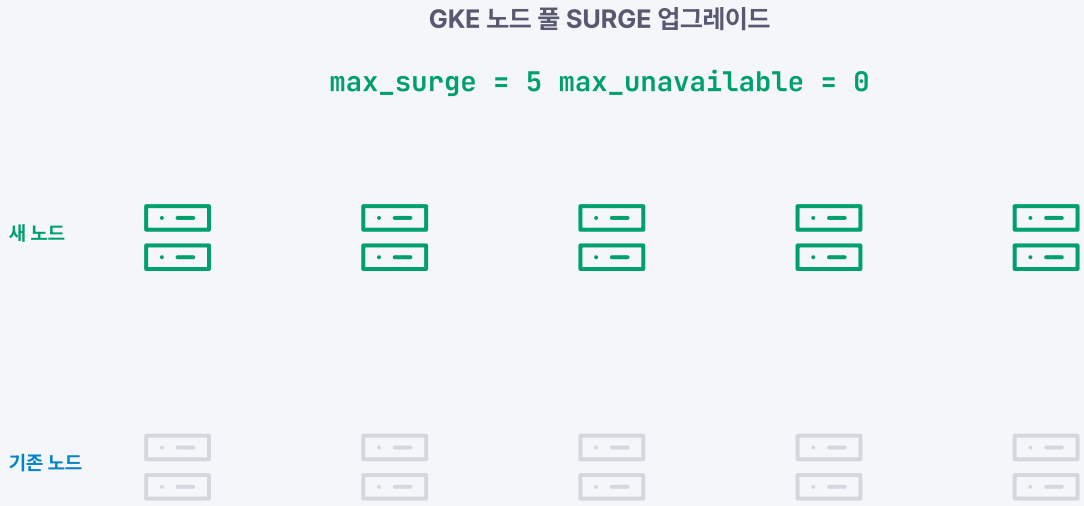
GCP Terraform GKE 플랫폼

노드 풀 업그레이드 중에도 용량을 유지하는 SURGE 설정과, VPN 대역만 허용하는 방화벽 기본선을 코드로 담은 모듈 모음입니다.

22
공개한 Terraform 모듈

5
max_surge로 먼저 뜨는 새 노드

0
max_unavailable, 동시에 빠지는 노드



문제

GKE 노드를 업그레이드하는 동안 서비스 용량이 줄면 안 됐고, 관리 포트는 VPN 대역에서만 열려야 했습니다.

접근

노드 풀 업그레이드 전략을 SURGE로 두어 새 노드 5대를 먼저 올리고 사용 불가 노드를 0으로 유지합니다. c2sfirewall 모듈은 우선순위 1000으로 VPN의 관리 포트만 열고, 우선순위 65534로 그 외 tcp와 udp 아웃바운드를 막습니다.

결과

새 노드가 준비된 뒤에야 기존 노드가 빠집니다. 호스트 장애는 hostError 감사 로그에서 로그 기반 지표, 알림 정책, Slack 채널 순으로 이어지도록 구성했습니다.

설계 · 구현 · 개인 프로젝트 · 2026.11 공개 · Terraform, GCP, GKE, Cloud Logging, Cloud Monitoring

github.com/kimjhtlzk/pf-02-gcp-terraform-gke-platform

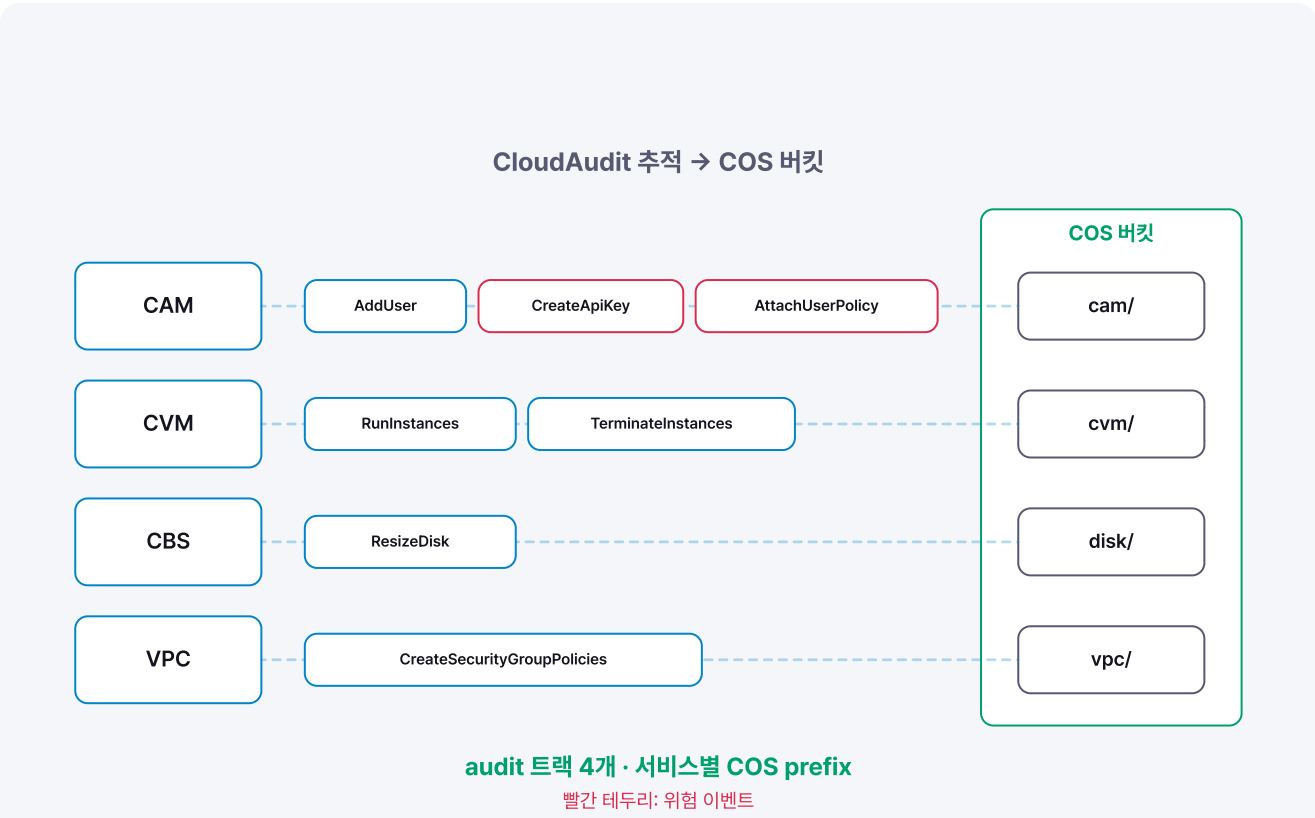
Tencent Cloud 감사 대비 Terraform

CloudAudit 추적을 서비스별로 코드에 정의해 위험한 변경을 COS 버킷에 남기는 Tencent Cloud 모듈 모음입니다.

18
공개한 Terraform 모듈

4
서비스별 감사 트랙

19
코드로 추적하는 감사 이벤트



문제

Tencent Cloud에서 누가 무엇을 바꿨는지 서비스별로 남겨야 했습니다. 또 provider 릴리스 주기가 불규칙해 버전이 바뀌면 코드 변경 없이도 plan이 달라질 수 있었습니다.

접근

CloudAudit 추적을 CAM, CVM, CBS, VPC 서비스별로 코드에 정의하고 COS 버킷의 서비스별 prefix에 저장합니다. provider는 모든 모듈에서 = 1.81.163으로 고정합니다.

결과

권한, 서버, 디스크, 보안그룹 변경 이벤트가 코드로 정의한 추적에 따라 COS에 남습니다. CreateApiKey와 AttachUserPolicy 같은 위험 이벤트도 포함합니다.

설계 · 구현 · 개인 프로젝트 · 2026.11 공개 · Terraform, Tencent Cloud, CloudAudit, COS, CAM, CLB

github.com/kimjhtlzk/pf-03-tencent-terraform-audit-ready

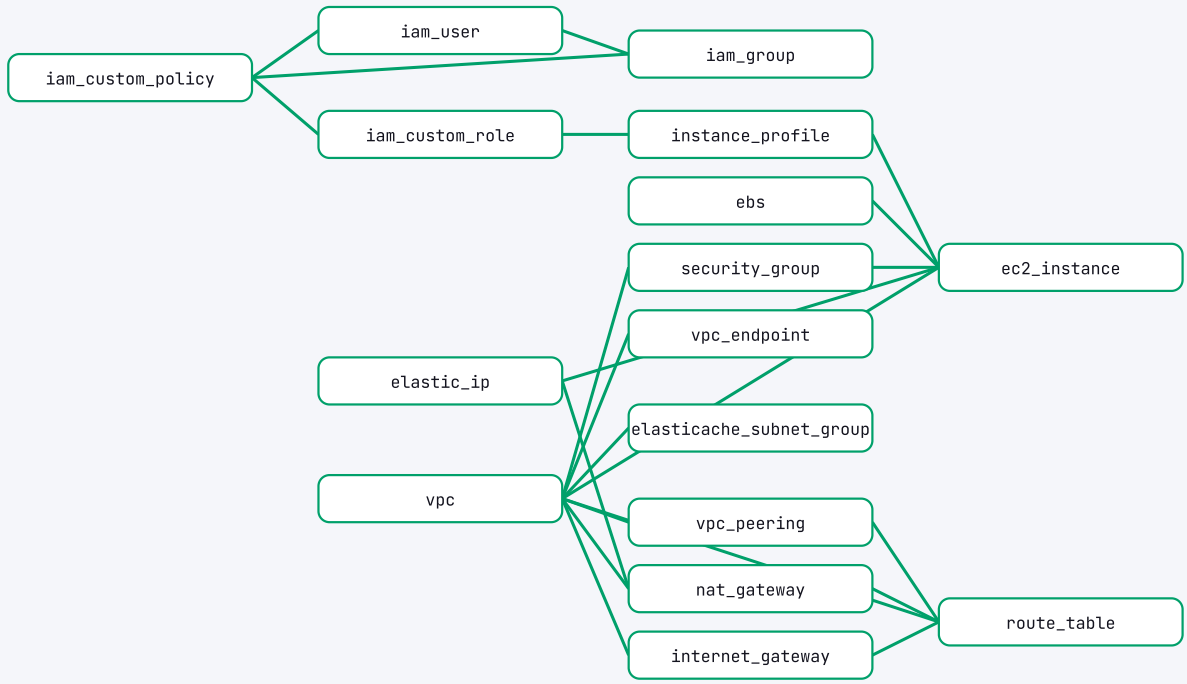
AWS Terragrunt DRY IaC

Terraform 모듈 16개를 루트 설정 하나로 묶고, Terragrunt가 의존성 그래프를 읽어 순서대로 실행하게 한 구조입니다.

16
Terraform 모듈 · 유닛

1
복사본 16개를 대체한 루트 설정

-40%
배포 시간 단축 (실무 적용 결과)



문제

Terraform 루트마다 backend와 provider 블록을 반복해서 썼고, 루트 사이의 실행 순서는 사람이 기억해야 했습니다.

접근

모듈은 일반 Terraform으로 두고 Terragrunt로 감쌌습니다. backend, provider, 공통 inputs는 live/root.hcl 한 곳에서 생성합니다. _envcommon이 모듈 source와 dependency를 맡고, 각 유닛은 자기 inputs만 적습니다. dependency 블록으로 그래프를 만들고 mock_outputs로 클라우드 없이 전체를 검증합니다.

결과

이 레포는 모듈 16개와 유닛 16개를 의존성 4단계로 묶고, 클라우드 자격증명 없이 전체를 검증합니다. 배포 시간 40% 단축은 실제 운영 환경을 이 패턴으로 옮겼을 때 얻은 결과입니다.

설계 · 구현 · 개인 프로젝트 · 2026.11 공개 · Terragrunt, Terraform, AWS

github.com/kimjhtlzk/pf-04-aws-terragrunt-dry-iac

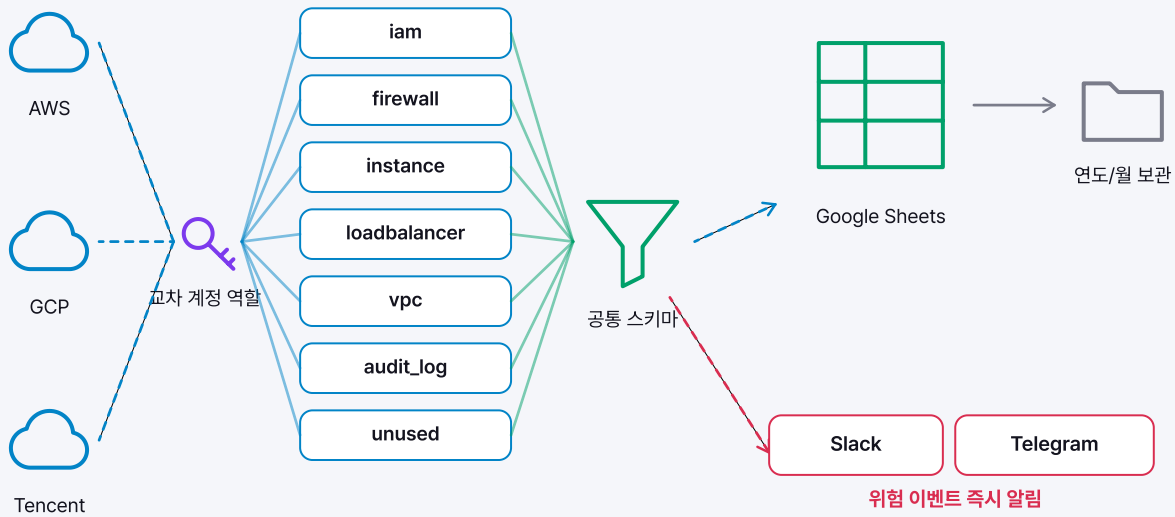
멀티 클라우드 자산·감사 자동화

AWS, GCP, Tencent Cloud의 자산·권한·감사 로그를 공통 스키마로 모으고, 위험 이벤트가 생기면 바로 알리는 Python 도구입니다.

3
공개 코드가 다루는 클라우드

7
자원 종류별 수집기

49
추적하는 감사 액션 (16종은 즉시 알림)



문제

세 클라우드는 같은 개념을 서로 다른 형태로 돌려줍니다. 권한이나 방화벽을 비교하려면 사람이 세 형식을 맞춰 읽어야 했습니다.

접근

계정별로 STS AssumeRole 등 교차 계정 역할로 접근하고, 자원 종류별 수집기 7개로 모은 결과를 자원별 데이터 클래스 하나로 정규화합니다. Google Sheets와 CSV로 내보내고, 위험 이벤트 16종은 critical_events.yaml 기준으로 Slack과 Telegram에 바로 보냅니다.

결과

바탕이 된 실무 도구는 Azure를 포함한 4개 클라우드를 매일 새벽에 점검해 월말 며칠의 수동 점검을 없앴고, Terraform 기본값과 CI 관문과 함께 보안 지적 조치율을 60%에서 85% 이상으로 올리는 데 기여했습니다.

설계 · 구현 · 개인 프로젝트 · 2026.11 공개 · Python, boto3, Google Cloud SDK, Tencent Cloud SDK, Google Sheets, Slack, Telegram

github.com/kimjhtlzk/pf-05-multicloud-asset-audit-automation

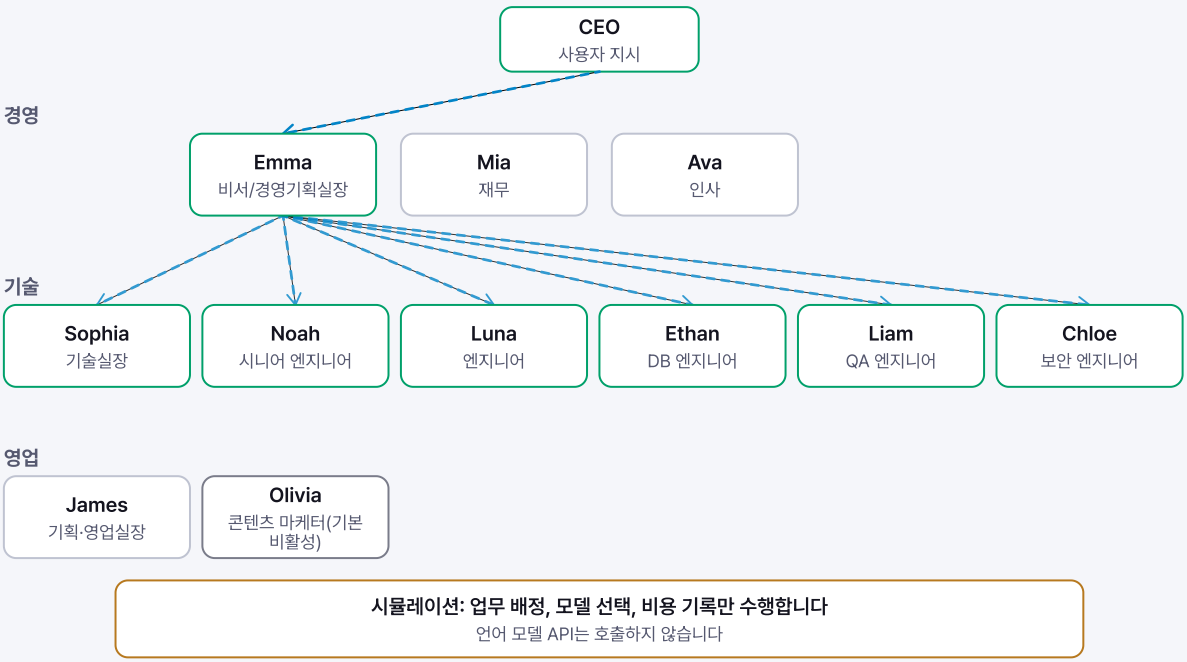
AetherCrew: AI 직원 11명의 가상 회사

직접 만든 오케스트레이터로 AI 직원 11명의 업무 배정, 모델 선택, 비용 추적을 구현한 가상 회사입니다. 에이전트 작업은 시뮬레이션이며 언어 모델 API는 호출하지 않습니다.

11
AI 직원

3
부서

8
웹 대시보드 페이지



문제

여러 AI 직원이 역할별로 일을 나눠 맡으면 누가 어떤 모델을 쓰는지, 비용이 어디까지 갔는지, 외부 시스템에 무엇을 쓸 수 있는지 코드로 통제해야 합니다.

접근

CEO 지시를 비서(경영기획실장)가 받아 계획을 쓰고 역할별 Task로 나눕니다. 직원별로 Anthropic, OpenAI, Google, DeepSeek 모델의 우선순위를 두고 실패하면 다음 모델로 전환합니다. 예상 비용을 쌓아 80%와 95%에서 경고하고 100%에서 작업을 막습니다. 외부 쓰기는 기본 차단, 파일 접근은 프로젝트 루트로 제한합니다.

결과

FastAPI 웹 대시보드 8페이지에서 조직도, 비용, 로그, 픽셀 오피스 CCTV를 봅니다. API 키 없이 실행되고 언어 모델은 호출하지 않으므로 실제 작업 결과물은 나오지 않습니다. Slack, Jira, GitHub 클라이언트는 실제 코드이며 INTEGRATIONS_LIVE_WRITE를 켤 때만 동작합니다.

설계 · 구현 · 개인 프로젝트 · 2026.11 공개 · Python, FastAPI, WebSocket, pytest, Slack, Jira, GitHub

github.com/kimjhtlzk/pf-06-aethercrew-ai-company

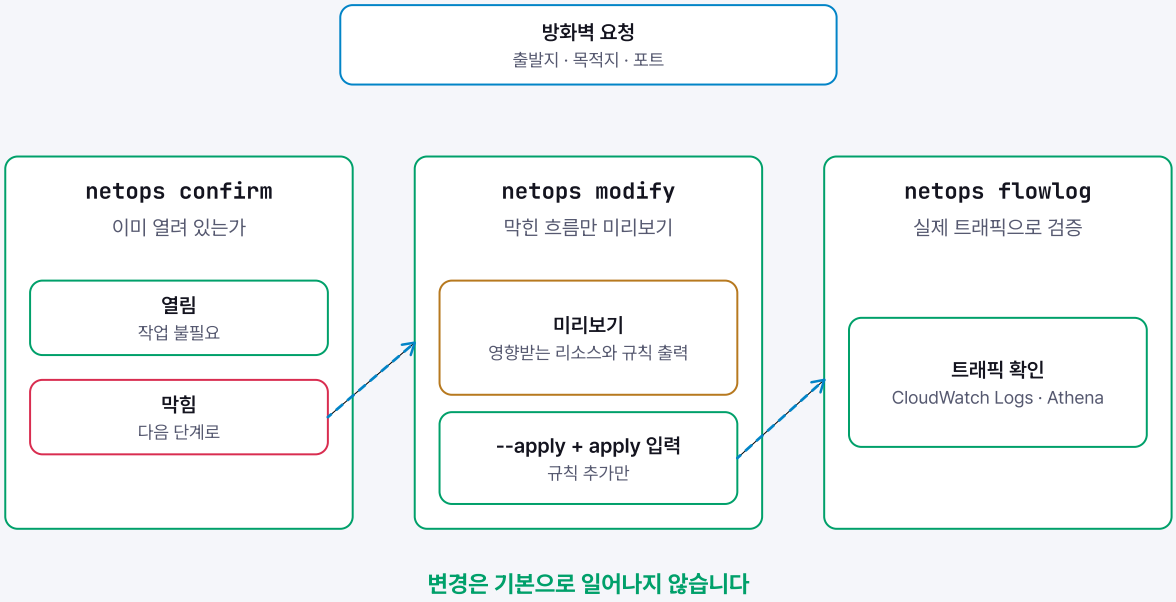
AWS 방화벽 요청 자동화

방화벽 요청이 이미 열려 있는지 먼저 확인하고, 막힌 것만 미리보기 뒤 적용하며, 실제 트래픽으로 검증하는 CLI입니다.

3
단계: confirm, modify, flowlog

4
입력 형식

4
식별하는 리소스 종류



문제

방화벽 요청은 이미 열린 통신을 다시 여는 경우가 많아 중복 규칙이 쌓였고, 어떤 리소스와 규칙이 걸리는지 계정마다 콘솔에서 찾아야 했습니다.

접근

netops confirm이 주소를 AWS 리소스로 식별하고 보안그룹 규칙과 라우트를 평가합니다. 막힌 통신만 modify로 넘겨 계획을 먼저 출력합니다. flowlog는 CloudWatch Logs나 Athena에서 실제 트래픽을 찾습니다.

결과

기본 실행은 아무것도 바꾸지 않습니다. 규칙 추가에는 --apply와 apply 입력이 필요하고, 규칙은 추가만 가능하며, 슬롯이 없는 보안그룹은 고르지 않습니다.

설계 · 구현 · 개인 프로젝트 · 2026.11 공개 · Python, boto3, AWS, CloudWatch Logs, Athena, Excel

github.com/kimjhtlzk/pf-09-aws-netops-firewall-automation

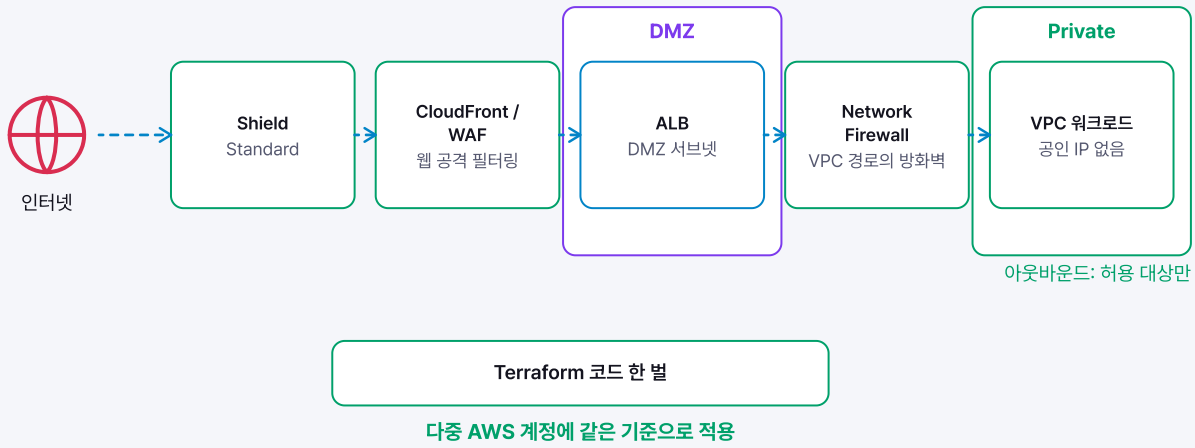
우리금융그룹 AWS 클라우드 보안 아키텍처

다중 AWS 계정의 망분리 요건을 서브넷과 방화벽 구조로 옮기고, 보안 구성을 Terraform 코드 한 벌로 표준화했습니다.

5
Splunk SIEM/SOAR로 연동한 보안 서비스

3
망분리를 옮긴 설계 원칙

1
다중 AWS 계정에 적용하는 Terraform 코드



문제

그룹사들이 퍼블릭 클라우드를 처음 도입하는 단계에 Terraform이 없었고, 계정마다 담당자에 따라 보안 설정이 달랐습니다. 망분리 규정을 문서로만 지키기도 어려웠습니다.

접근

망분리(중요도가 다른 네트워크를 나눠 외부 위협이 내부로 바로 들어오지 못하게 하는 금융권 통제) 요건을 DMZ·Private 서브넷 분리, 백엔드 공인 IP 금지, 허용된 대상으로만 아웃바운드라는 세 원칙으로 옮겼습니다. WAF, Network Firewall, Shield Standard 구성을 Terraform으로 정의해 모든 계정에 같은 코드를 적용하고, GuardDuty와 Security Hub까지 Splunk SIEM/SOAR로 연동했습니다.

결과

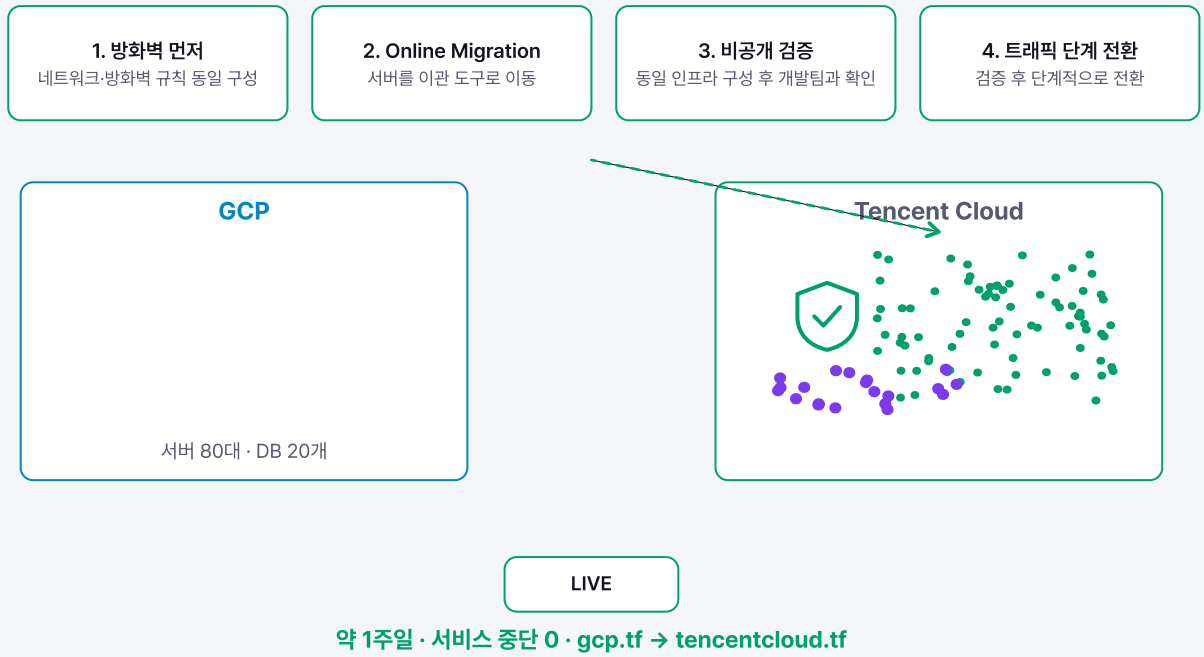
계정이 늘어나도 같은 보안 기준이 코드로 반복되어 담당자별 정책 편차가 사라졌습니다. 다중 AWS 계정의 통신 구성은 Python 파이프라인으로 자동 수집·분석해 반복 가능한 점검이 됐습니다. IAM 권한도 Terraform으로 정의해 과도한 권한을 구조적으로 막았습니다.

Cloud Security Engineer · 정보보안부 · 2026.02 – 2026.10 · Terraform, AWS Network Firewall, AWS WAF, AWS Shield Standard, GuardDuty, Security Hub, Splunk, Python

GCP → Tencent Cloud 무중단 마이그레이션

서머너즈워: 크로니클의 서버 80대·DB 20개를 약 1주일 만에 서비스 중단 없이 GCP에서 Tencent Cloud로 옮긴 프로젝트입니다.

80	20	0
이관한 서버	이관한 DB	서비스 중단



문제

라이브 게임의 인프라를 클라우드 간에 옮기면서 서비스를 멈출 수 없었고, 옮기는 과정에서 방화벽 정책이 다르게 적용되면 안 됐습니다.

접근

네트워크와 방화벽 규칙을 Tencent에 GCP와 동일하게 먼저 구성했습니다. 서버는 Online Migration 도구로 옮기고 같은 인프라를 구성한 뒤, 비공개로 운영하며 개발팀과 이슈를 확인하고 트래픽을 단계적으로 전환했습니다. 전 과정은 GCP Terraform 코드를 Tencent Terraform 코드로 옮기는 방식으로 진행했습니다.

결과

약 1주일 만에 서비스 중단 없이 이관을 끝냈습니다. 일부는 GCP에 남기고 대부분의 게임 서버와 DB를 Tencent로 옮겼으며, 방화벽 정책은 전부 검토해 다시 세운 상태로 운영했습니다.

Cloud Engineer · 메인 엔지니어 · 2023.02 – 2026.02 · Terraform, GCP, Tencent Cloud, Online Migration

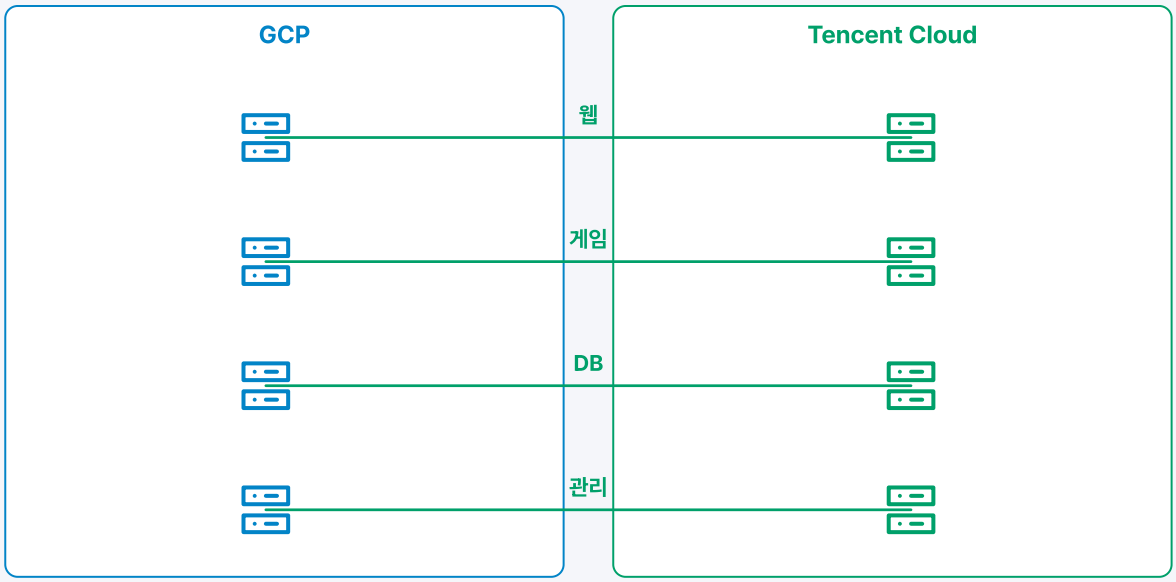
서머너즈워: 천공의 아레나 멀티클라우드와 방화벽 재설계

GCP와 Tencent Cloud에 걸친 서버 600대·DB 70개 환경의 방화벽 정책을 전수 재설계하고, 무점검 원칙으로 운영한 대규모 게임 인프라입니다.

600
서버

70
DB

-40%
Terragrunt 전환 뒤 배포 시간



서버 600대 · DB 70개 (두 클라우드 합계)

문제

회사에서 가장 큰 프로젝트로, 중국 서비스를 위해 Tencent를 함께 운영했습니다. 두 클라우드의 방화벽 정책이 오래 쌓여 중복되고 복잡해졌고, 여러 리전으로 커진 코드는 Terraform만으로 다루기 어려워졌습니다.

접근

클라우드 간 방화벽 정책을 전수 분석해 중복과 불필요한 규칙을 제거하고 웹, 게임, DB, 관리 역할 기준으로 다시 묶었습니다. Terraform을 Terragrunt 구조로 옮겨 코드 중복을 걷어 냈습니다. 대부분의 서버를 로드밸런서 백엔드로 두어 서버를 하나씩 빼고 넣는 방식으로 무점검 패치를 했습니다.

결과

방화벽 정책이 단순해져 검토와 변경이 쉬워지고 공격에 노출되는 범위가 줄었습니다. Terragrunt 전환으로 배포 시간을 40% 줄였고, GCP Cloud Armor로 DDoS 방어를 운영했습니다.

Cloud Engineer · 메인 엔지니어 · 2023.02 – 2026.02 · GCP, Tencent Cloud, Terraform, Terragrunt, Cloud Armor

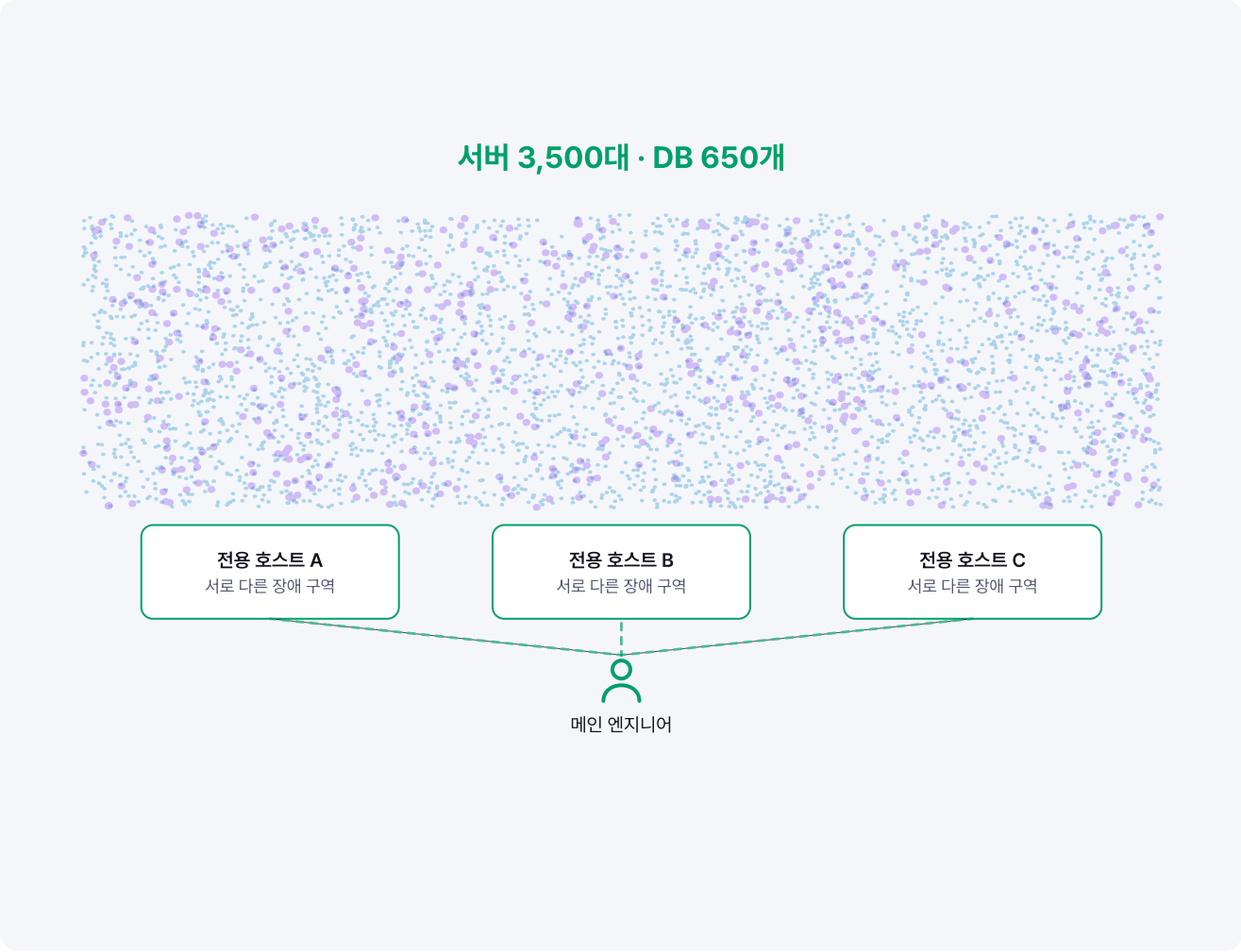
Mir4 Global 초대형 Azure 인프라와 DDoS 자동 차단

서버 3,500대·DB 650개 규모의 글로벌 게임을 Azure에서 전담 운영하고, DDoS 공격 IP를 로그에서 찾아 NSG에 자동으로 등록하는 구조를 만들었습니다.

3,500
서버

650
DB

0
탐지에서 차단까지 사람 개입



문제

규모가 커서 물리 장애와 공격의 영향이 전 세계 서비스로 번질 수 있었고, 야간과 주말의 DDoS 공격은 사람 손으로 막는 동안 대응이 늦어졌습니다.

접근

설계 초기부터 참여해 Azure Dedicated Host 기반 고가용성 구조를 함께 잡았습니다. Azure DDoS 방어가 남기는 완화 흐름 로그를 Log Analytics에서 주기적으로 조회해 비정상 TCP SYN 출발지를 찾고, PowerShell이 해당 IP를 NSG Deny 규칙에 등록합니다.

결과

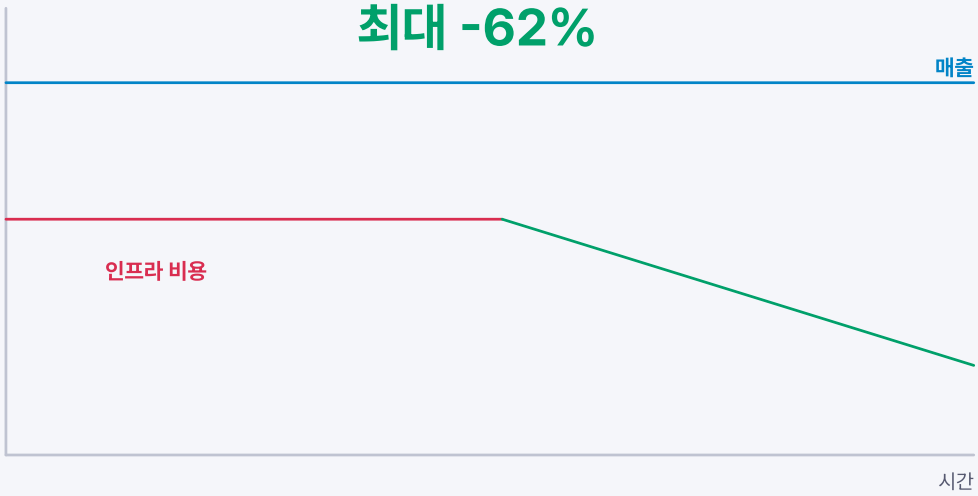
탐지부터 차단까지 사람 개입 없이 이어져 야간과 주말의 대응 지연을 없앴습니다. 리소스 배포, 장애 자동 조치, 자산 수집도 PowerShell로 자동화하며 전담 운영했습니다.

Cloud Engineer · 메인 엔지니어 (고객사 WEMADE) · 2021.07 – 2023.02 · Azure, Dedicated Host, PowerShell, Log Analytics, NSG

드래곤스카이 인프라 비용 62% 절감

매출 지표와 인프라 비용을 연동해 모니터링하고 Python으로 불필요 리소스를 매일 자동 탐지해 인프라 비용을 최대 62% 줄였습니다.

-62%	10	5
인프라 비용 최대 절감	서버	DB



문제

게임 매출 대비 인프라 비용 비율이 높아 수익성에 문제가 있었습니다.

접근

매출 지표와 인프라 비용을 연동해 함께 모니터링했습니다. Python으로 전 CSP의 불필요 리소스를 매일 자동 탐지했습니다.

결과

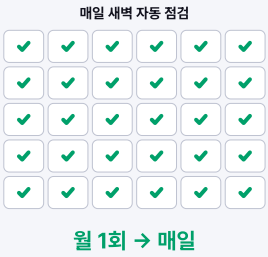
인프라 비용을 최대 62% 절감했습니다.

Cloud Engineer · 메인 엔지니어 · 2023.02 – 2026.02 · Azure, Python, Billing monitoring

현장 노트

한 달에 한 번이던 보안 점검을, 매일 자동으로

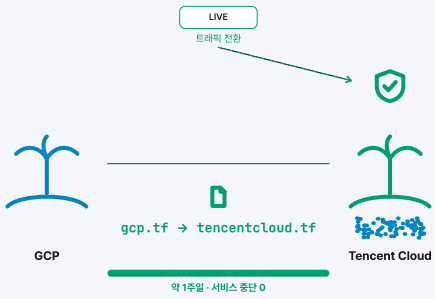
월 1회 → 매일 · 4 Clouds



문제	보안 점검은 월말에 한 번, 며칠씩 수동으로 했습니다. 누군가 자신에게 과도한 권한을 주거나 주요 서버를 지워도 최대 한 달 가까이 모를 수 있었습니다. 누가 시킨 일은 아니었지만 직접 바꾸기로 했습니다.
판단	같은 "권한 변경"이라도 클라우드마다 기록 방식이 모두 달랐습니다. 제각각인 로그를 "누가, 언제, 어떤 리소스에, 무엇을 했나"라는 공통 기준으로 바꾸기로 했습니다.
실행	4개 클라우드의 모든 프로젝트에서 감사 로그를 가져와 정규화하고, 매일 새벽 자동으로 점검합니다. 사고로 이어질 수 있는 행위만 크리티컬로 정해 텔레그램·슬랙으로 알리고, 행위자 확인 뒤 조치 완료로 기록합니다.
결과	점검 주기는 월 1회에서 매일로, 위험 행위 인지 시점은 최대 한 달 뒤에서 발생 당일로 바뀌었습니다. 사람이 따로 시간을 내 점검하지 않아도, 매일 정해진 시간에 점검이 자동으로 끝나 있게 됐습니다.
배운 점	사고로 이어질 수 있는 신호를 먼저 정의하고 놓치지 않는 구조가 더 중요합니다. 권한이 없을 때는 "이렇게 해봤더니 좋아졌다"는 결과가 가장 강한 설득입니다.

라이브 게임을 멈추지 않고 클라우드를 옮기다

1 Week · 0 Downtime · 80 Servers / 20 DBs



문제	라이브 서비스의 GCP 인프라(서버 80대, DB 20개)를 Tencent Cloud로 옮겨야 했습니다. 서비스 중단은 허용되지 않았습니다.
판단	가장 큰 위험은 이관 후 보안 정책이 빠지거나 다르게 적용되는 것이라고 봤습니다. 그래서 서버보다 네트워크와 방화벽을 먼저 옮기기로 했습니다.
실행	Tencent Cloud에 방화벽 규칙을 GCP와 동일하게 먼저 구성하고, Online Migration으로 서버를 옮겼습니다. 바로 전환하지 않고 비공개 운영으로 개발팀과 함께 이슈 여부를 확인했습니다.
결과	검증 후 서비스를 전환했고, 서비스 중단 없이 약 1주일 만에 이관을 마쳤습니다.
배운 점	클라우드를 옮길 때는 보안 기준이 같이 따라가게 만드는 것이 핵심입니다. 이관 내용이 Terraform 코드로 남아 이후 운영과 변경도 쉬워졌습니다.

새벽 3시의 DDoS, 사람 없이 막다

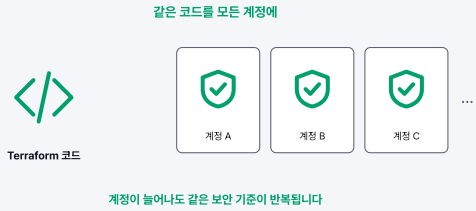
탐지 → 차단 자동화 · 사람 개입 0



문제	공격은 대응 인력이 가장 적은 야간과 주말에 집중됐습니다. 사람이 알림을 보고, 접속해서, 공격 IP를 찾아, 차단 규칙을 넣는 동안 서비스가 흔들렸습니다.
판단	인력을 늘리거나 새 솔루션을 사는 대신, 이미 쌓이고 있던 로그를 쓰기로 했습니다. Azure의 DDoS 방어 기능이 공격 출발지 정보가 담긴 로그를 이미 남기고 있었습니다.
실행	비정상적으로 많은 TCP SYN 패킷을 보내는 출발지를 공격 후보로 정의했습니다. 주기적으로 조회해 뽑은 IP를 PowerShell 스크립트가 NSG(방화벽 규칙 묶음)의 차단 규칙에 자동으로 추가합니다.
결과	탐지부터 차단까지 사람의 개입 없이 이어져 야간·주말 대응 지연을 없앴습니다. 새 솔루션이나 인력은 필요하지 않았습니다.
배운 점	새 도구를 들이기 전에 이미 쌓이고 있는 로그로 무엇을 할 수 있는지 먼저 봅니다. 당장 막는 것과 함께, 같은 상황이 다시 와도 버틸 구조를 만듭니다.

아무것도 없던 금융 클라우드에 보안 기준을 세우다

IaC 0 → Terraform 표준화



문제	그룹사들이 퍼블릭 클라우드를 처음 도입하던 단계였고, IaC(코드로 인프라를 관리하는 방식) 체계가 전혀 없었습니다. 계정마다 요구사항이 달랐지만 공통 보안 기준은 일관되게 지켜야 했습니다.
판단	망분리(중요도가 다른 네트워크를 나눠 외부 위협이 내부로 바로 들어오지 못하게 하는 금융권 핵심 통제)를 설계 원칙 세 가지로 옮겼습니다. DMZ와 Private 서브넷 분리, 공인 IP 할당 원천 차단, 허용된 대상으로만 나가는 아웃바운드입니다.
실행	Terraform을 도입해 Network Firewall과 WAF 구성을 코드로 정의하고, 같은 코드를 모든 계정에 적용했습니다.
결과	계정이 늘어나도 같은 보안 기준이 코드로 반복됩니다. 담당자마다 설정이 달라지던 문제가 사라졌습니다.
배운 점	규정을 문서로 두지 않고 구조와 코드 안에 넣으면, 누가 배포해도 같은 수준이 유지됩니다.

그 밖의 경험

제목	한 줄 요약	결과
서버 3,500대 글로벌 게임의 전담 엔지니어	서버 3,500대, DB 650개 규모 글로벌 MMORPG의 전담 엔지니어로 Azure 운영과 기술 자문을 맡았습니다.	**3,500 Servers, 650 DBs**
서버 500대 배포, 80% 빠르게	사람이 서버 묶음마다 반복하던 배포 단계를 PowerShell 스크립트로 묶어 배포 속도를 80% 개선했습니다.	**80% 개선, 500 Servers / 100 DBs**
인프라 상태를 실시간으로 한눈에	Telegraf, InfluxDB, Grafana로 실시간 대시보드를 만들고 Zabbix와 Grafana로 모니터링을 운영했습니다.	
같은 지적이 다시 나오지 않게	보안 지적을 서비스 영향도로 나누고 보완 통제로 합의하며, 반복 지적은 코드 기본값으로 막아 감사 조치율을 60%에서 85% 이상으로 올렸습니다.	**60% → 85%+**
복사해서 쓰던 인프라 코드를 하나로	운영 중인 인프라를 코드로 옮기고 환경마다 복사하던 Terraform 코드를 Terragrunt 공통 구조로 묶어 배포 시간을 40% 줄였습니다.	**-40% 배포 시간**
엉킨 방화벽 규칙을 다시 설계하다	Tencent Cloud와 GCP에 걸친 서버 600대, DB 70개 규모의 방화벽 규칙을 전수 분석해 역할 기준으로 재설계했습니다.	**600 Servers / 70 DBs**
게임 서버를 향한 공격을 막다	서버 600대 규모 게임의 DDoS 탐지와 방어를 Tencent Cloud와 GCP에서 운영했습니다.	
문제가 생기면 먼저 알게	서버·DB 지표를 Zabbix와 Grafana로 보고, 이상이 생기면 Slack과 Telegram 알림으로 대응했으며, 월별 비용도 분석했습니다.	
말로 요청하면 인프라 코드를 만들어주는 AI	로컬 LLM과 벡터 DB로 인프라 코드 생성 프로토타입을 만들어 팀에 시연했고, 검증에서 한계를 찾아 다음 설계까지 정리했습니다. 실무 정착까지는 이어가지 못했습니다.	
필요한 만큼만, 코드로 남기는 권한	서비스 계정과 역할별 권한을 Terraform으로 정의해, 과도한 권한 부여가 구조적으로 어렵도록 만들었습니다.	
여러 계정의 경보를 한 화면으로	GuardDuty, Security Hub, Splunk SIEM을 연동해 다중 AWS 계정 전반의 보안 이벤트를 한곳에서 모니터링했습니다.	
"켜져 있다"와 "막고 있다"는 다르다	CDN + 로드밸런서 구조에서 해외 접속 차단 정책이 우회될 수 있는 지점을 찾아 고치고, 차단 결과를 로그로 증명했습니다. 이미 수정된 사례입니다.	
인터넷이 없는 PC에서도 돌아가는 운영 도구	인터넷이 막힌 업무 PC에서 돌아가는 오프라인 개발 환경과 Python 도구 4종을 만들어, 반복되던 통신 요청 처리를 표준 절차로 바꿨습니다.	**도구 4종, Python 패키지 31종 오프라인 번들**
상대가 아는 언어로 설명하기	보안그룹 정책을 담당자들이 익숙한 물리 방화벽 정책서 구조로 문서화해, AWS가 처음인 담당자도 바로 업무를 시작할 수 있게 했습니다.	
패킷을 잡는 시간을 줄이다	FortiGate 패킷 캡처 명령 구성과 pcapng 파일 변환을 Python으로 자동화해 분석 준비 시간을 줄였습니다.	
매달 반복되는 보안 점검을 책임지다	방화벽과 WAF, WIPS를 운영하며 매월 정기 점검 보고서를 작성했고, 퇴사할 때 이 업무를 인수인계서로 정리해 넘겼습니다.	
요구사항을 숫자로 검증한 보안 검토	신규 가상화 인프라 도입 심의의 보안 부문을 맡아, 요건을 정리하고 수치로 검증한 뒤 통신 방향별 역할 분담을 제안했습니다.	
AI 직원 11명이 일하는 가상 회사	직접 설계한 오케스트레이터로 AI 직원 11명의 가상 회사를 만들며, 에이전트가 서로를 검증 없이 믿는 신뢰 경계 문제를 체감했습니다.	**11 Agents, 3 Divisions**
클릭 두 번으로 "이 통신 되나요?"에 답하는 서비스	콘솔에서 하나씩 따라가며 확인하던 AWS 통신 경로를, 다이어그램에서 두 리소스만 고르면 판정하는 서비스로 만들고 있습니다.	
포트폴리오용 공개 저장소 시리즈	기존 GitHub 실습·도구 저장소를 번호 붙인 공개 저장소로 재구성하고, 각 경험과 연결하고 있습니다.	

실무 프로젝트

프로젝트	CSP	서버 / DB	기간
우리금융그룹 AWS 클라우드 보안	AWS		2026.02 – 2026.10
서머너즈워: 천공의 아레나	GCP, Tencent	600 / 70	2023.02 – 2026.02
Mir4 Global	Azure	3,500 / 650	2021.07 – 2023.02
Mir M Korea	Azure	700 / 150	2022.02 – 2023.02
Mir4 Korea	Azure	500 / 100	2020.12 – 2023.02
Mir2/3 Korea	Azure	200 / 40	2020.12 – 2023.02
서머너즈워: 크로니클	GCP, Tencent	80 / 20	2023.02 – 2026.02
컴투스프로야구V25	GCP	90 / 8	2023.02 – 2026.02
서머너즈워: 러쉬	Tencent	76 / 11	2024.03 – 2026.02
갓앤데몬	GCP	66 / 12	2024.07 – 2026.02
스타시드: 아스니아 트리거	AWS	50 / 20	2023.09 – 2026.02
MLB 라이벌	GCP	23 / 5	2023.03 – 2026.02
ADT-CAPS (SK Shieldus)	Azure	10 / 2	2021.03 – 2023.02
드래곤스카이: 슈팅 방치형 RPG	Azure	10 / 5	2023.02 – 2026.02
서머너즈워: 백년전쟁	GCP	8 / 2	2023.02 – 2026.02
골프스타	GCP	6 / 2	2023.02 – 2026.02
낚시의 신: 크루	Tencent	4 / 2	2023.02 – 2026.02
전투기 키우기: Strikers 1945	Azure	4 / 1	2023.10 – 2026.02
Strikers1945: RE	Tencent	3 / 2	2023.06 – 2026.02
미니게임천국	GCP	3 / 4	2024.03 – 2026.02

학력 · 교육

경민대학교 정보통신학과 — 2015.03 – 2020.02 · 병역 기간 포함

학점은행제 학사 — 국가평생교육진흥원 — 2019.12 – 2020.08 · 정보통신학과 학위 수여

KG IT뱅크 — AWS 기반 아키텍처 설계 전문가 양성과정 — 2020.02.17 – 2020.07.17 · 6개월 / 800시간. 가상의 고객사를 AWS로 이전하는 프로젝트 수행

육군 병장 만기전역 — 2016.12 – 2018.09 · 통신병(교환병)

자격증

NCP — Naver Cloud Platform Certified Professional
2021.08.18

NCA — Naver Cloud Platform Certified Associate
2021.07.07

정보처리기사
2021.06.02

Microsoft Azure Fundamentals (AZ-900)
2020.10.22

리눅스마스터 2급
2019.10.11

네트워크관리사 2급
2019.07.09

HashiCorp Certified: Terraform Associate
2022.12.18 · 이력

CKA — Certified Kubernetes Administrator
2022.12.16 · 이력

Microsoft Azure Database Administrator Associate
2022.06.09 · 이력

Microsoft Azure Solutions Architect Expert
2022.06.08 · 이력

Microsoft Azure Administrator Associate
2022.06.07 · 이력

AWS Certified Solutions Architect – Professional
2022.05.29 · 이력

Microsoft Azure Architect Technologies
2021.07.01 · 이력

Microsoft Azure Security Engineer Associate
2021.06.18 · 이력

AWS Certified Solutions Architect – Associate
2020.06.23 · 이력

일하며 얻은 원칙

> **새 도구를 들이기 전에 이미 쌓이고 있는 로그로 무엇을 할 수 있는지 먼저 본다.**
> 새벽 3시의 DDoS, 사람 없이 막다

> **권한이 없을 때는 "이렇게 하자"는 제안보다 "이렇게 해봤더니 좋아졌다"는 결과가 가장 강한 설득이다.**
> 한 달에 한 번이던 보안 점검을, 매일 자동으로

> **"안 된다"로 끝내면 대화가 끝나고, 대안을 들고 가야 실제로 조치가 일어난다.**
> 같은 지적이 다시 나오지 않게

> **클라우드를 옮길 때는 보안 기준이 같이 따라가게 만든다.**
> 라이브 게임을 멈추지 않고 클라우드를 옮기다

> **규정을 문서로 두지 않고 구조와 코드 안에 넣으면, 누가 배포해도 같은 수준이 유지된다.**
> 아무것도 없던 금융 클라우드에 보안 기준을 세우다

> **보안 기능을 켜두는 것과, 그 기능이 위협을 막고 있다는 것을 증명하는 것은 다른 일이다.**
> "켜져 있다"와 "막고 있다"는 다르다

> **말하는 해결책("열어주세요")과 실제 문제는 다를 수 있다. 무엇이 안 되는지부터 확인한다.**
> 인터넷이 없는 PC에서도 돌아가는 운영 도구

> **AI는 판단을 보조하고, 실제 조치는 사람이 승인한다.**
> AI 직원 11명이 일하는 가상 회사