



이준형

보안 · 모니터링 · 관측

세부 구현 내용

1 보안 & 감사

CloudTrail 감사 로깅 · 실시간 Teams 알림 3종 · WAF 탐지 · 비용 경보 · 멀티클라우드 DR/SSO

2 인프라 모니터링

Prometheus·Grafana 수집 · Alertmanager 알림 4종 · 리소스 대시보드 · HTTPS 접속

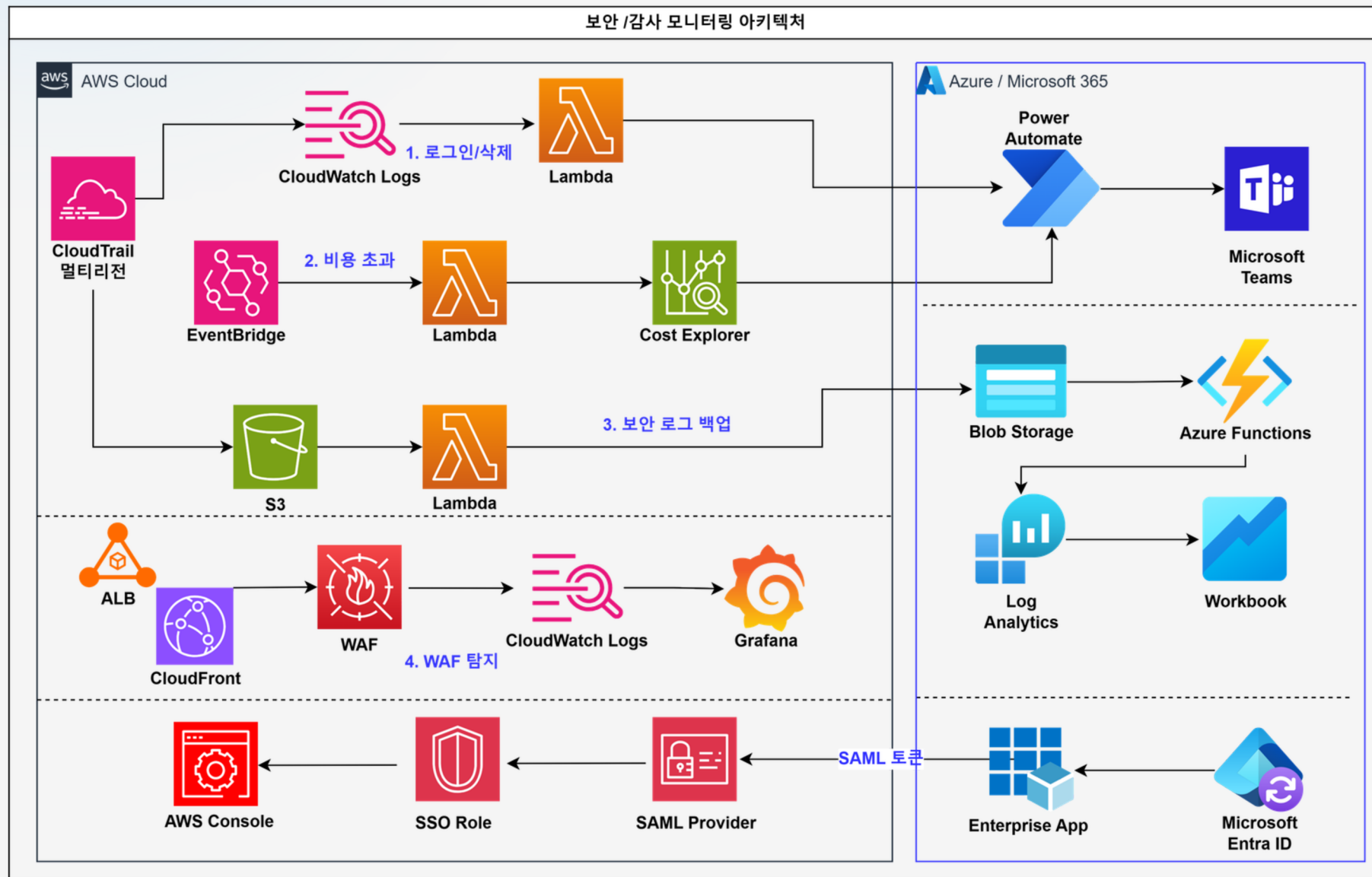
3 애플리케이션 관측

로그·메트릭·추적(X-Ray) 3축 · 멀티리전 · IoT 파이프라인

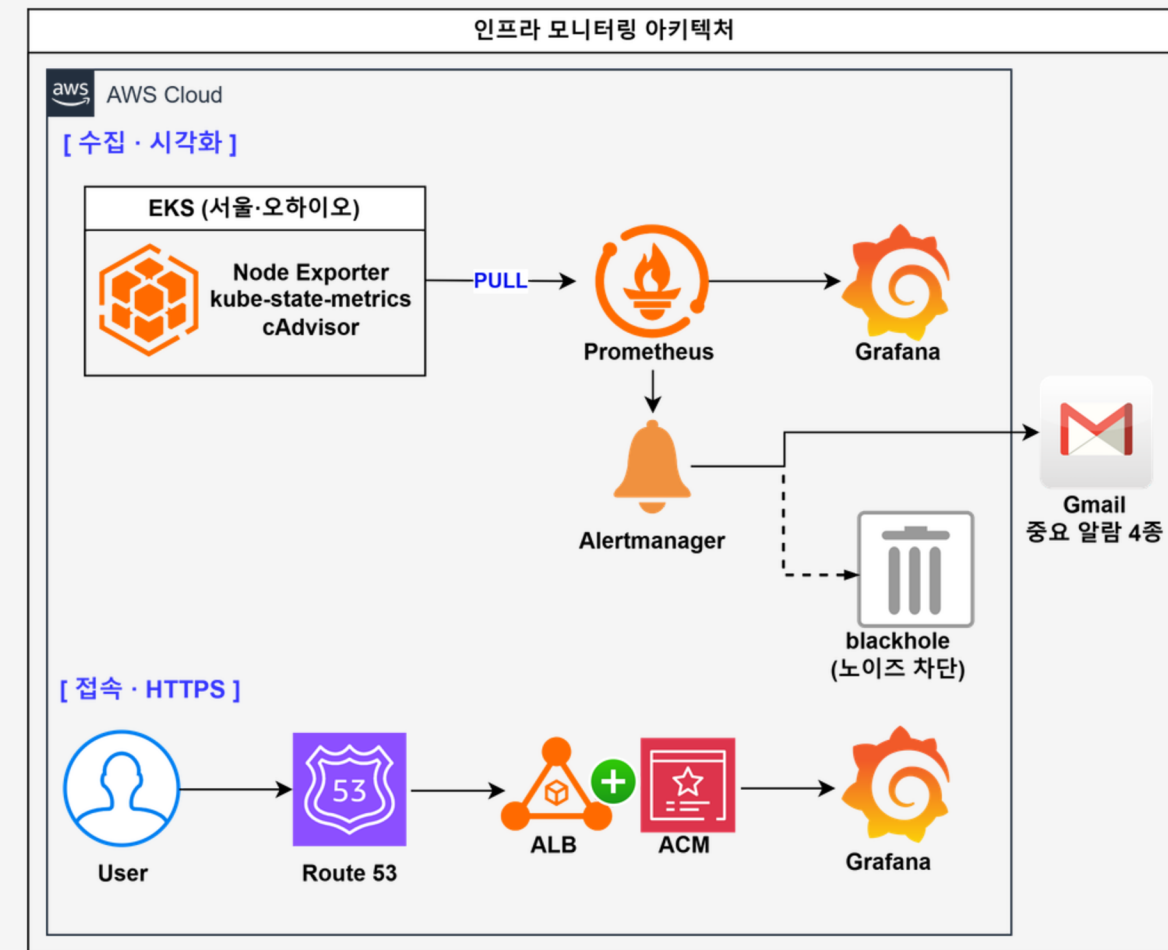
사용 서비스 및 기술

- 보안 CloudTrail · CloudWatch · S3 · WAF · Lambda · EventBridge · Cost Explorer · Power Automate · Teams · IAM SAML
- 모니터링 Prometheus · Grafana · Alertmanager · Helm · IRSA
- 관측 Fluent Bit · OpenTelemetry(ADOT) · X-Ray · IoT Core · Firehose · Glue · Athena
- 멀티클라우드 Azure (Blob · Functions · Log Analytics · Entra ID SSO)
- 공통 EKS · Terraform · Route 53 · ACM · S3 Backend · 멀티리전

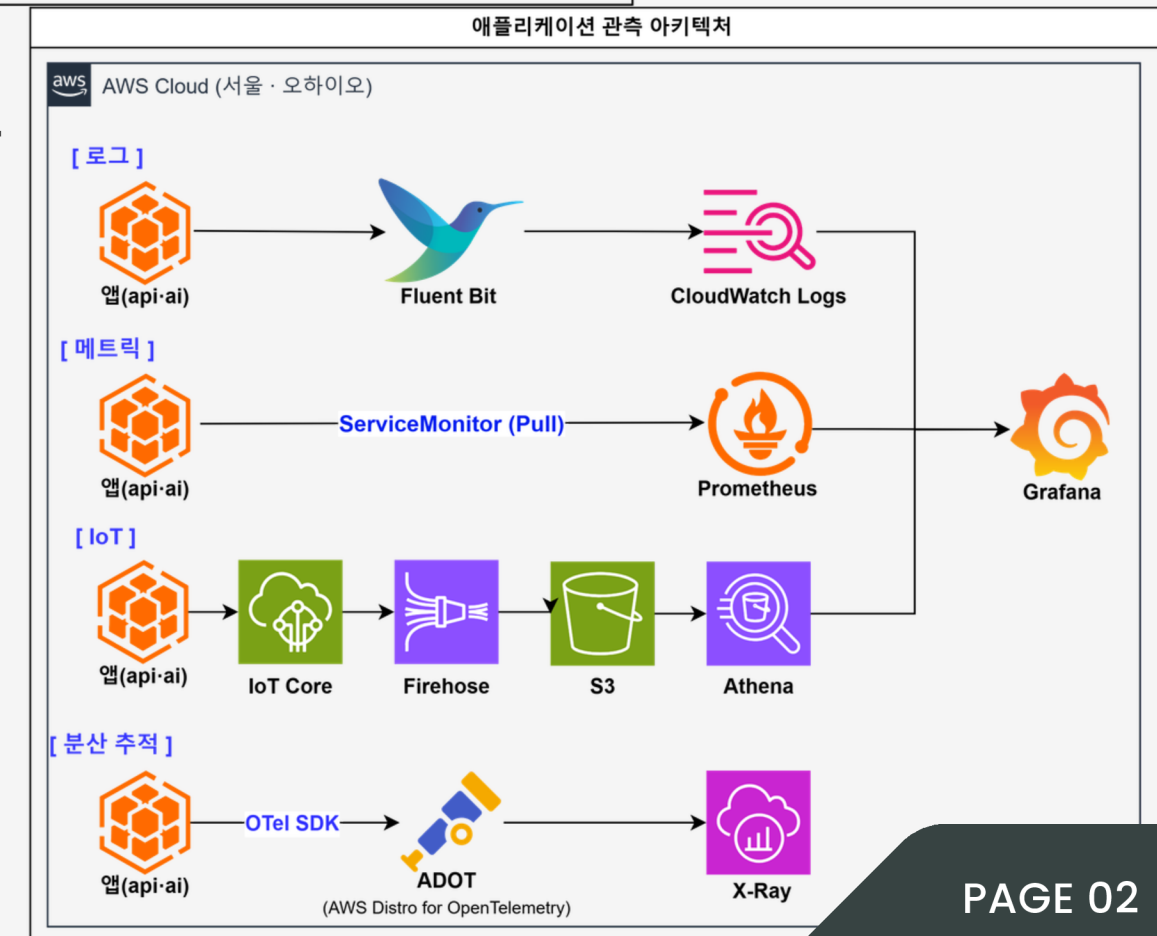
1 보안 & 감사



2 인프라 모니터링



3 애플리케이션 관측



멀티클라우드 계정 체계

aws

AWS 계정: IAM Identity Center

- 장기 액세스 키 → 임시 자격증명 (AWS 권장사항)
- root·IAM User 키 발급 없이 SSO 단일 진입
- 팀원 4명 권한을 코드(IaC)로 중앙 관리

Azure 계정: Entra ID 연동

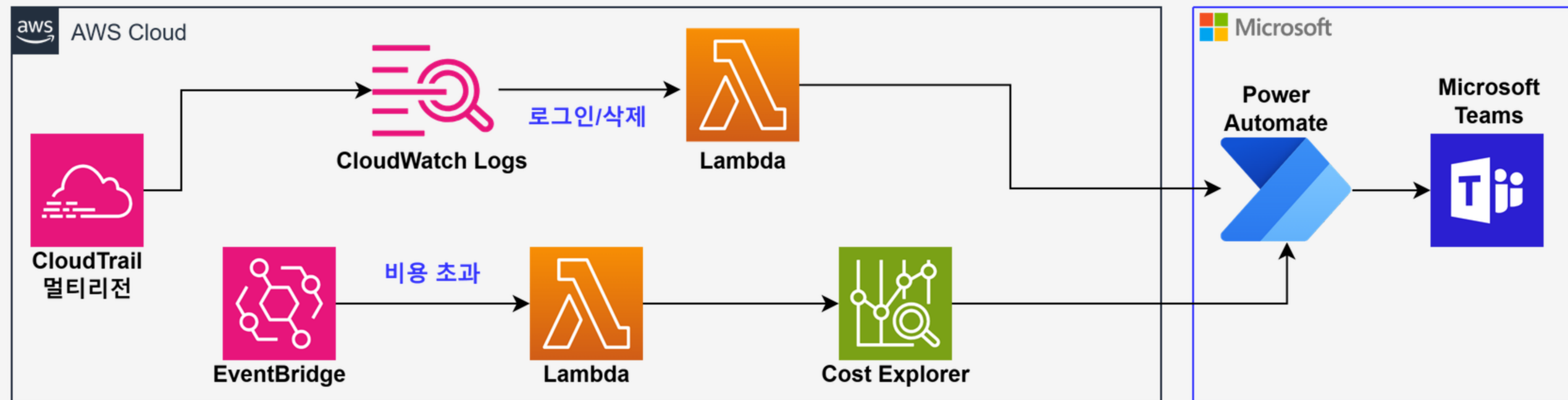
- 알림(Teams·Power Automate)이 이미 M365 기반 → Entra ID로 자연스럽게 통합
- 별도 인증 체계 추가 없이 기존 Azure 자산 활용
- 사용자·디렉토리를 Entra ID에서 일원 관리

멀티클라우드 SSO 통합

- Azure 계정 하나로 AWS 콘솔 단일 로그인
- 퇴사자·권한 변경을 중앙에서 즉시 통제 (운영 확장성 고려)
- 자격증명 분산 없는 중앙 관리 (연동 상세 → PAGE 06)

실시간 감사 알림

로그인·삭제·비용 알림 흐름



01

Lambda & Power Automate

⚡ Lambda

- CloudTrail 로그 파싱 (사용자·IP·시각 추출)
- 이벤트 감지 3종 (로그인/삭제/비용)
- AWS 서비스 노이즈 필터링 (destroy 등 자동작업 제외)

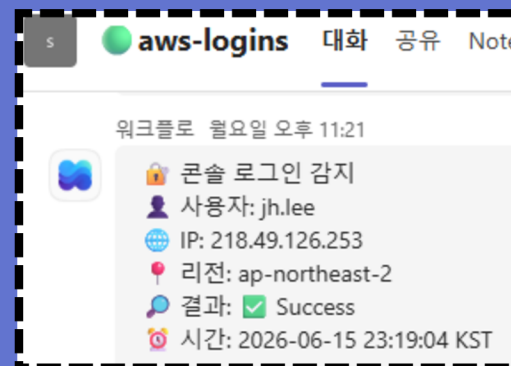
함수 이름	이름
siseon-lambda-billing-alert	aws-billings
siseon-lambda-delete-alert	aws-alerts
siseon-lambda-login-alert	aws-logins

➡ Power Automate

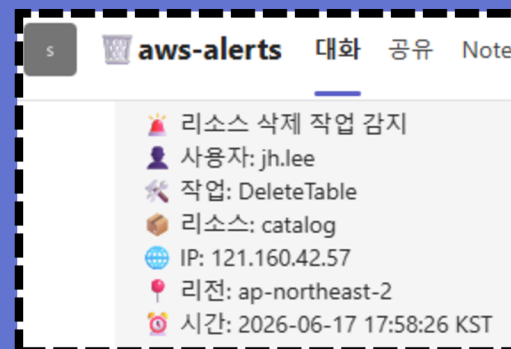
- Webhook 중계 (Lambda → Teams)
- 채널별 분배 (aws-logins/alerts/billing)
- 메시지 동적 렌더링

02

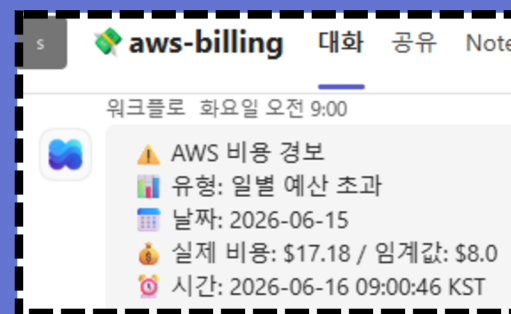
Teams 채팅



1
채널별 분리 발송
→ 로그인/삭제/비용



2
실시간
→ 이벤트 발생 즉시



3
상세 정보 파싱
→ 사용자 · IP · 시각

03

billing 구성

```

1 import json
2 import urllib.request
3 import urllib.error
4 import os
5 import boto3
6 from datetime import datetime, timezone, timedelta
7
8 WEBHOOK_URL = os.environ["TEAMS_WEBHOOK_URL"]
9 DAILY_THRESHOLD = 8.0
10 MONTHLY_THRESHOLD = 135.0
11 KST = timezone(timedelta(hours=9))
12
13 ce = boto3.client("ce", region_name="us-east-1")
14
15 def get_cost(granularity, start, end):
16     response = ce.get_cost_and_usage(
17         TimePeriod={"Start": start, "End": end},
18         Granularity=granularity,
19         Metrics=["UnblendedCost"])
20
21     return float(response["ResultsByTime"][0][0]["Total"][0]["UnblendedCost"]["Amount"])
  
```

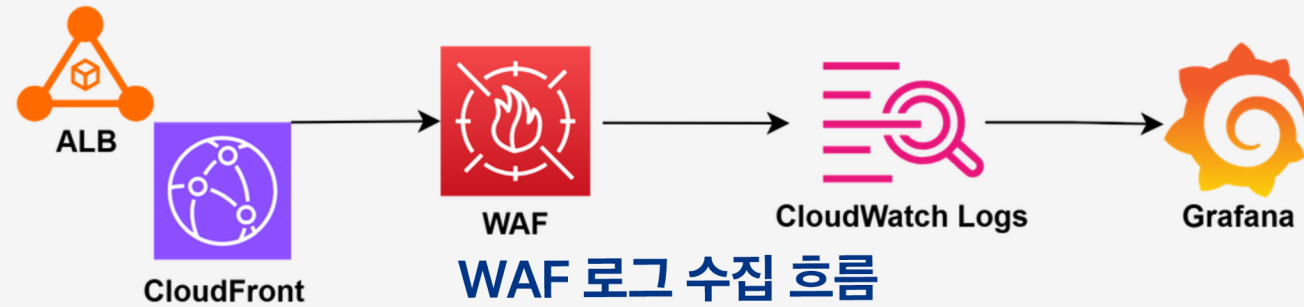
EventBridge — 매일 KST 09:00 스케줄

Cost Explorer — 전일 비용 조회

임계값 — 일 \$8 / 월 \$135 초과 시 알림

Alarm/Budgets 한계
"상태 변경 시에만" 발송 → 능동 폴링으로 해결

멀티리전 WAF 통합 관제



WAF 대상 서울 ALB

총 차단(BLOCK) 5

총 탐지(공격 시도) 32

공격 소스 IP 수 19

terminatingRuleId	blocks
AWSManagedRulesKnownBadInputsR...	5

attackRule	detections
aws:waf-managed:aws:core-rule-set:N...	18
aws:waf-managed:aws:core-rule-set:R...	7
aws:waf-managed:aws:known-bad-inp...	5
aws:waf-managed:aws:core-rule-set:C...	1
aws:waf-managed:aws:core-rule-set:B...	1

httpRequest.clientIp	request
183.101.245.210	293
121.160.42.57	277
180.80.107.14	206
118.235.6.72	89
98.88.19.20	46
18.216.244.128	26
20.250.14.139	18

Time	action
2026-06-18 15:22:34	ALLOW
2026-06-18 15:22:36	ALLOW
2026-06-18 15:22:41	ALLOW
2026-06-18 15:22:44	ALLOW

WAF 대상: 서울 ALB, 오하이오 ALB, CloudFront

Point 01

핵심 설계: 차단(BLOCK)과 탐지(count) 분리

운영 안전을 위해 대부분 룰을 count 모드로 운영. 단순 BLOCK 집계 시 SQLi·XSS 공격이 화면에서 누락되는 함정 → awswaf:managed 라벨을 파싱해 ALLOW로 통과된 공격 시도까지 포착

Point 02

멀티리전: 드롭다운 하나로 3개 WAF

서울 ALB / 오하이오 ALB / CloudFront — 리전·로그그룹 다른 3개를 단일 대시보드에서 전환

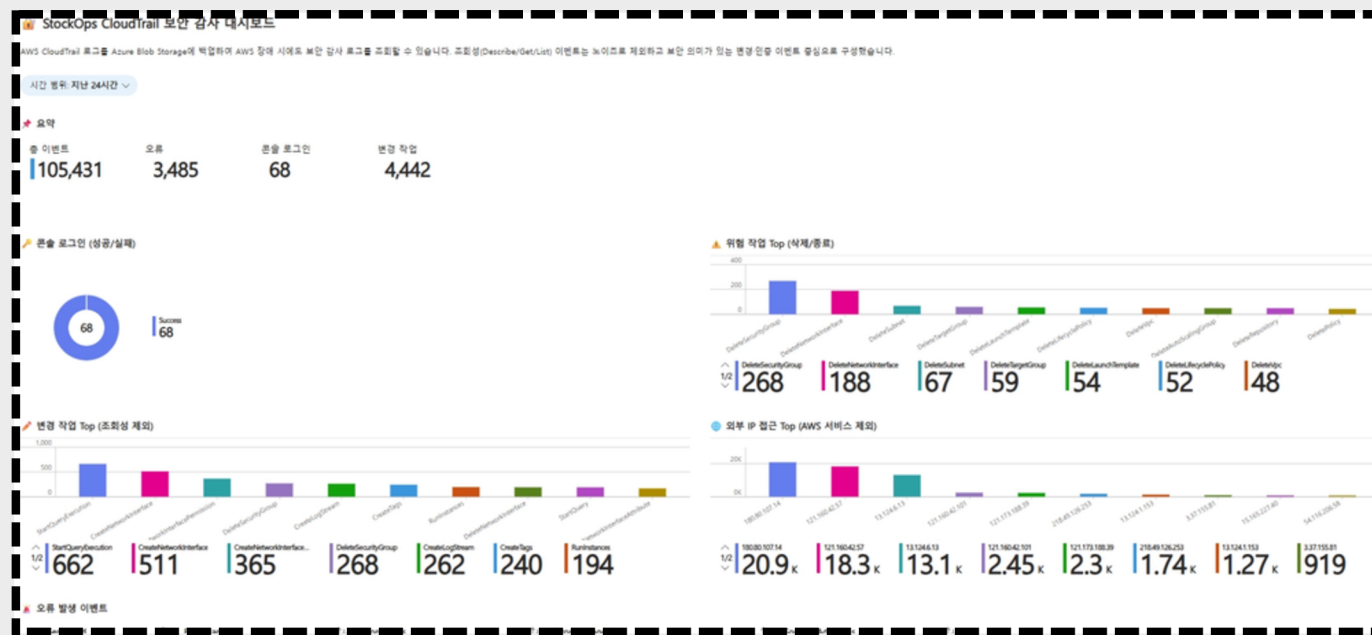
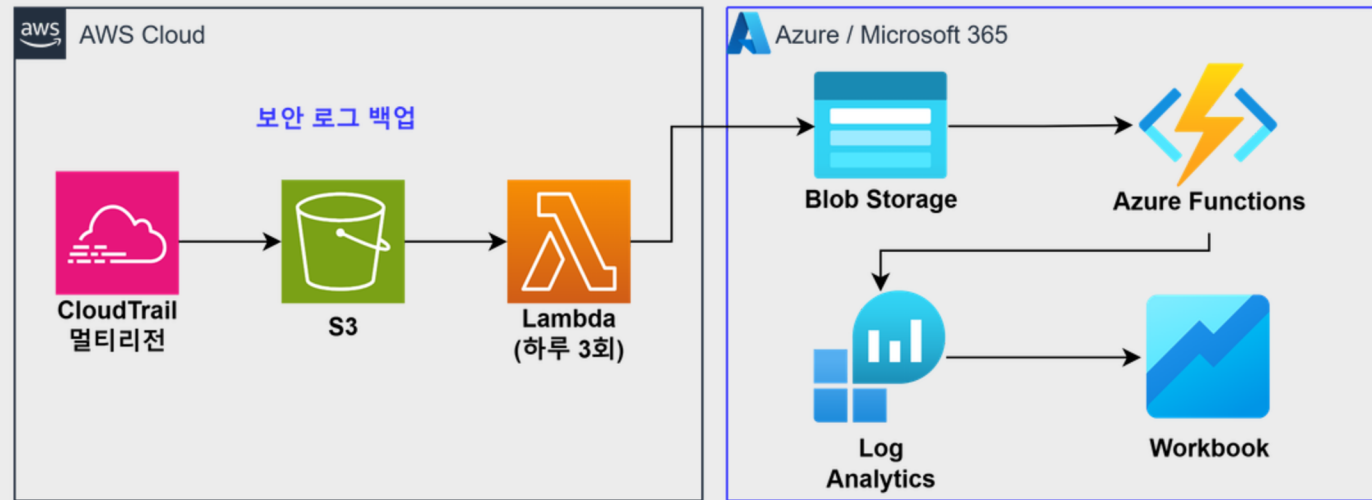
Point 03

실제 공격으로 검증

외부에서 유입된 phpunit 스캐너 봇이 KnownBadInputs 룰에 BLOCK으로 차단되는 실제 공격을 포착. 직접 날린 SQLi·XSS 패턴으로 탐지 로직까지 검증

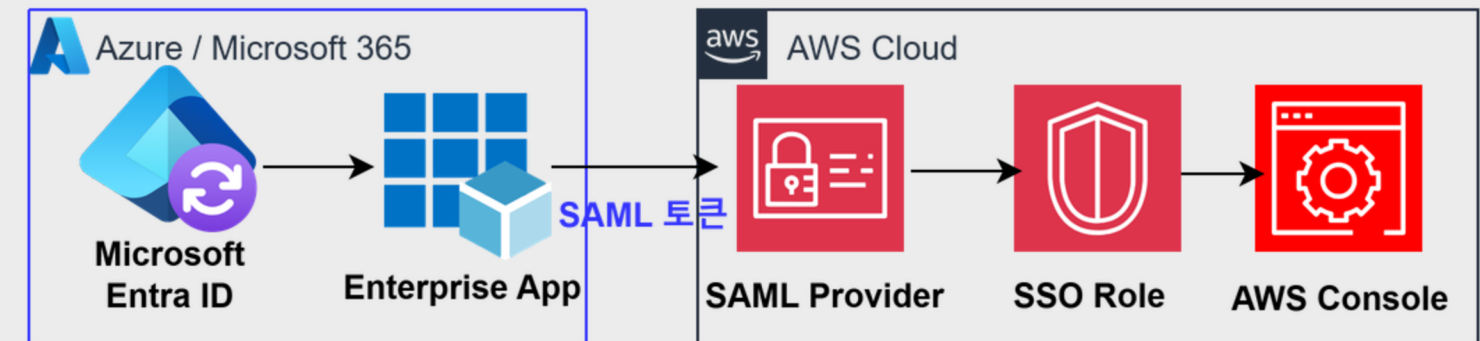
AWS ↔ Azure 백업 · SSO 연동

☁ 재해복구 — CloudTrail 로그 Azure 백업



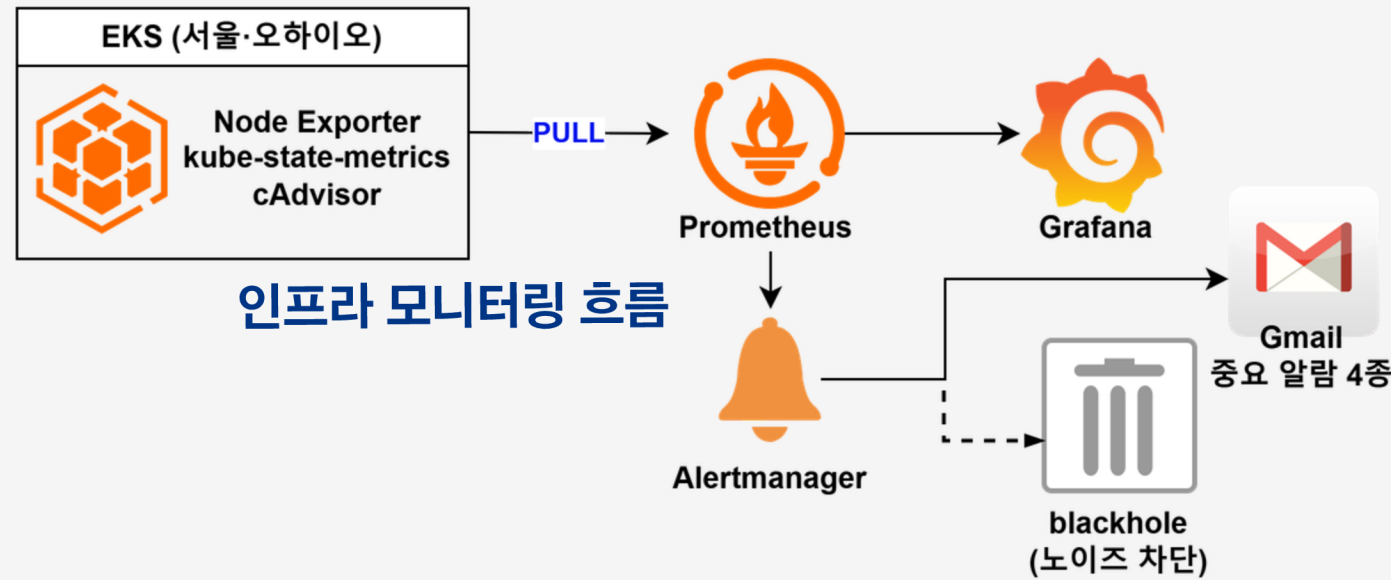
- ✓ CloudTrail 감사 로그를 Azure Blob에 백업 → AWS 장애 시에도 감사 지속
- ✓ 하루 3회 동기화 — 백업 공백 최대 8시간
- ✓ Azure Monitor Workbook으로 페일오버 시각화

🔑 통합 인증 — Entra ID SAML Federation



- ✓ AWS 콘솔 로그인을 Azure Entra ID로 페더레이션
- ✓ 별도 IAM 사용자 없이 Azure 계정으로 AWS 접근
- ✓ 자격증명 일원화 -> 퇴사자 차단·권한 회수를 Entra ID 한 곳에서

Prometheus 기반 인프라 모니터링



템플릿 없이 직접 설계

- Grafana 공식 템플릿 그대로 X
- EKS 환경에 맞춰 패널 직접 구성
- 대시보드 정의를 Terraform 코드로 관리 → GitOps

IRSA 기반 권한

- Grafana → AWS 리소스(Athena 등) 접근
- 액세스 키 없이 IRSA로
- "키 없이 역할 기반" 보안 원칙 적용



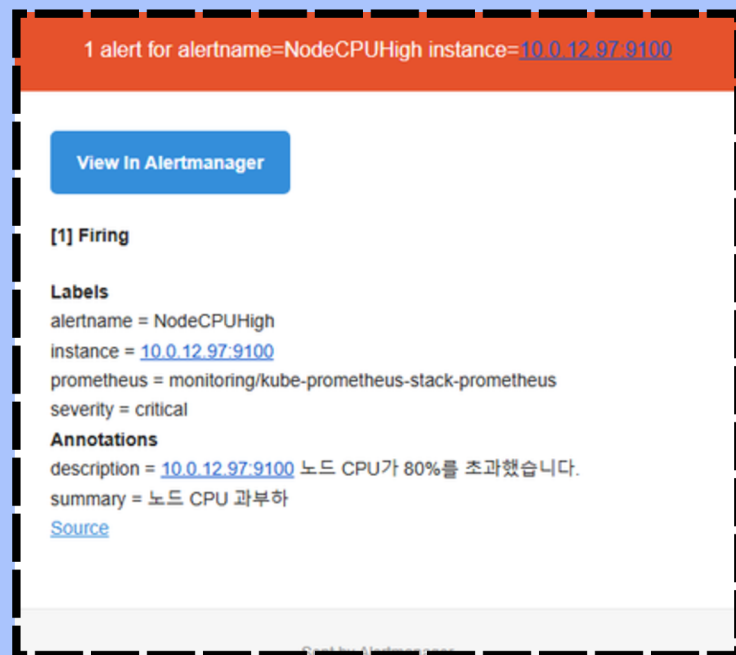
직접 구성한 핵심 패널

- 노드별 CPU 사용률 (Karpenter 오토스케일링 검증)
- 노드별 메모리 사용률 (로그 스케일 — 파드·노드 편차 흡수)
- 노드별 네트워크 I/O (트래픽 분산 확인)
- 파드 상태 / 재시작 현황 (OOM·크래시 조기 감지)
- 노드 범례 테이블 (다수 노드 한눈에)

알림 노이즈 제어 · HTTPS 보안

Alertmanager — 노이즈 억제 알림

Firing



1 alert for alertname=NodeCPUHigh instance=10.0.12.97.9100

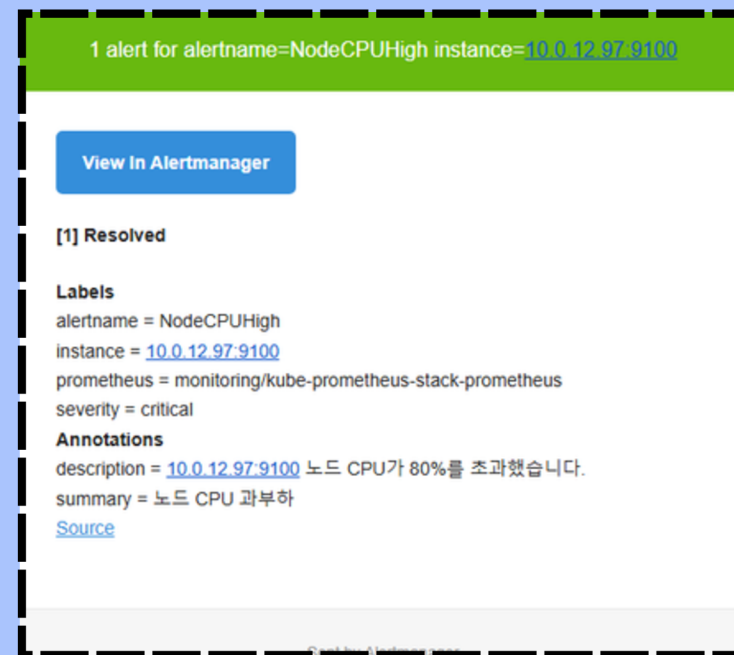
View In Alertmanager

[1] Firing

Labels
 alertname = NodeCPUHigh
 instance = [10.0.12.97.9100](#)
 prometheus = monitoring/kube-prometheus-stack-prometheus
 severity = critical

Annotations
 description = [10.0.12.97.9100](#) 노드 CPU가 80%를 초과했습니다.
 summary = 노드 CPU 과부하
[Source](#)

Resolved



1 alert for alertname=NodeCPUHigh instance=10.0.12.97.9100

View In Alertmanager

[1] Resolved

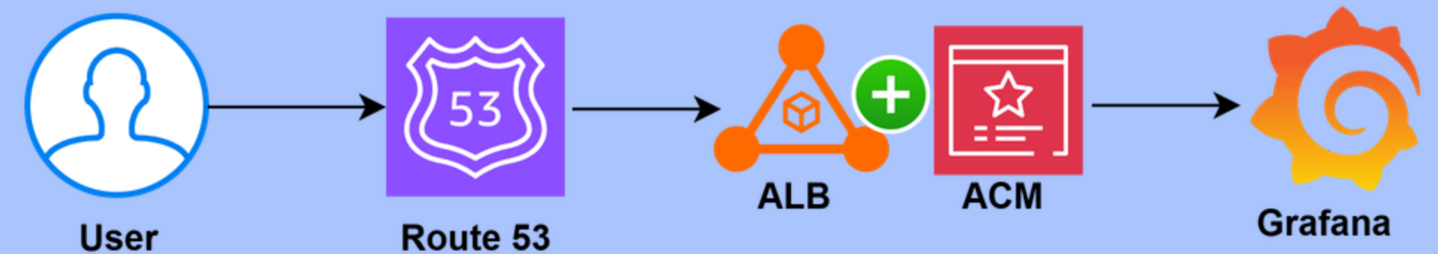
Labels
 alertname = NodeCPUHigh
 instance = [10.0.12.97.9100](#)
 prometheus = monitoring/kube-prometheus-stack-prometheus
 severity = critical


Annotations
 description = [10.0.12.97.9100](#) 노드 CPU가 80%를 초과했습니다.
 summary = 노드 CPU 과부하
[Source](#)

- 중요 4종만 Gmail 발송 (Pod장애/재시작/노드CPU/노드메모리)
- blackhole 라우팅으로 나머지 노이즈 차단 → 화이트리스트 방식
- Firing → Resolved 상태까지 추적 (발생·해소 둘 다 알림)

Grafana HTTPS — 도메인 + 인증서

[접속 · HTTPS]



 <https://grafana.siseon.live>

- grafana.siseon.live 도메인 (Route 53)
- ACM 와일드카드 인증서 (*.siseon.live) → HTTPS 암호화
- NLB → ALB Ingress 전환 + Route 53 alias 자동 연결

센서 데이터 분석 · 시각화



MQTT 센서 수집

- 창고 센서 7종 → MQTT → IoT Core
- IoT Rule이 토픽 와일드카드 구독·라우팅 (+/sensors/+)

서버리스 분석 파이프라인

- Firehose → S3(Hive 파티션) → Glue 카탈로그 → Athena
- 별도 DB·서버 없이 S3 직접 쿼리

⚡ 파티션 프로젝트 최적화

- year/month/day 파티션 자동 인식
- 쿼리 시 해당 날짜만 스캔
 - 스캔량·비용 최소화



애플리케이션 로그 · 메트릭 관측

로그 : Fluent Bit → CloudWatch



```

API 로그 (stockops-api)
> 2026-06-19 16:03:49.327 {"log":"2026-06-19T07:03:49.323868813Z stdout F 2026-06-19 07:03:49.323 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.314 {"log":"2026-06-19T07:03:49.311732797Z stdout F 2026-06-19 07:03:49.311 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.301 {"log":"2026-06-19T07:03:49.301761219Z stdout F 2026-06-19 07:03:49.301 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.287 {"log":"2026-06-19T07:03:49.287811667Z stdout F 2026-06-19 07:03:49.287 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.210 {"log":"2026-06-19T07:03:49.203266172Z stdout F 2026-06-19 07:03:49.203 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.196 {"log":"2026-06-19T07:03:49.195988963Z stdout F 2026-06-19 07:03:49.195 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.164 {"log":"2026-06-19T07:03:49.156231207Z stdout F 2026-06-19 07:03:49.156 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.149 {"log":"2026-06-19T07:03:49.1491868Z stdout F 2026-06-19 07:03:49.149 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for unk
> 2026-06-19 16:03:49.117 {"log":"2026-06-19T07:03:49.11495641Z stdout F 2026-06-19 07:03:49.111 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.117 {"log":"2026-06-19T07:03:49.103811228Z stdout F 2026-06-19 07:03:49.103 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.096 {"log":"2026-06-19T07:03:49.095988684Z stdout F 2026-06-19 07:03:49.095 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.013 {"log":"2026-06-19T07:03:49.995952063Z stdout F 2026-06-19 07:03:49.995 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:49.013 {"log":"2026-06-19T07:03:49.004843398Z stdout F 2026-06-19 07:03:49.003 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:48.931 {"log":"2026-06-19T07:03:48.921388451Z stdout F 2026-06-19 07:03:48.921 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:48.914 {"log":"2026-06-19T07:03:48.905770141Z stdout F 2026-06-19 07:03:48.905 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:48.914 {"log":"2026-06-19T07:03:48.913780036Z stdout F 2026-06-19 07:03:48.913 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:48.898 {"log":"2026-06-19T07:03:48.898889594Z stdout F 2026-06-19 07:03:48.897 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
> 2026-06-19 16:03:48.833 {"log":"2026-06-19T07:03:48.833485122Z stdout F 2026-06-19 07:03:48.833 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u

시 로그 (stockops-ai)
> 2026-06-19 16:03:16.134 {"log":"2026-06-19T07:03:16.134831656Z stdout F INFO: 10.0.11.198:55476 - \"GET /metrics HTTP/1.1\" 200 OK"}
> 2026-06-19 16:03:01.586 {"log":"2026-06-19T07:03:01.566226296Z stdout F INFO: 10.0.2.112:19732 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:03:01.541 {"log":"2026-06-19T07:03:01.540781821Z stdout F INFO: 10.0.1.96:15576 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:02:31.555 {"log":"2026-06-19T07:02:31.555107195Z stdout F INFO: 10.0.2.112:1574 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:02:31.534 {"log":"2026-06-19T07:02:31.534600319Z stdout F INFO: 10.0.1.96:24616 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:02:16.134 {"log":"2026-06-19T07:02:16.134830391Z stdout F INFO: 10.0.11.198:41710 - \"GET /metrics HTTP/1.1\" 200 OK"}
> 2026-06-19 16:02:01.580 {"log":"2026-06-19T07:02:01.580331597Z stdout F INFO: 10.0.2.112:16502 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:02:01.533 {"log":"2026-06-19T07:02:01.533014522Z stdout F INFO: 10.0.1.96:13366 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:01:46.135 {"log":"2026-06-19T07:01:46.135443152Z stdout F INFO: 10.0.11.198:38904 - \"GET /metrics HTTP/1.1\" 200 OK"}
> 2026-06-19 16:01:31.551 {"log":"2026-06-19T07:01:31.551194607Z stdout F INFO: 10.0.2.112:44680 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:01:31.529 {"log":"2026-06-19T07:01:31.529111182Z stdout F INFO: 10.0.1.96:18692 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:01:16.134 {"log":"2026-06-19T07:01:16.134883599Z stdout F INFO: 10.0.11.198:58662 - \"GET /metrics HTTP/1.1\" 200 OK"}
> 2026-06-19 16:01:01.583 {"log":"2026-06-19T07:01:01.583716622Z stdout F INFO: 10.0.2.112:37460 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:01:01.530 {"log":"2026-06-19T07:01:01.530230411Z stdout F INFO: 10.0.1.96:62608 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:00:46.135 {"log":"2026-06-19T07:00:46.135644989Z stdout F INFO: 10.0.11.198:55382 - \"GET /metrics HTTP/1.1\" 200 OK"}
> 2026-06-19 16:00:31.542 {"log":"2026-06-19T07:00:31.53500576Z stdout F INFO: 10.0.2.112:8874 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:00:31.525 {"log":"2026-06-19T07:00:31.525541794Z stdout F INFO: 10.0.1.96:7646 - \"GET /health HTTP/1.1\" 200 OK"}
> 2026-06-19 16:00:16.135 {"log":"2026-06-19T07:00:16.135534852Z stdout F INFO: 10.0.11.198:58256 - \"GET /metrics HTTP/1.1\" 200 OK"}

API 경고/에러 (WARN / ERROR)
> 2026-06-19 16:03:49.327 {"log":"2026-06-19T07:03:49.323868813Z stdout F 2026-06-19 07:03:49.323 [WARN] [stockops-sqs-ingestion] c.s.e.i.TelemetryIngestionService - Skipping telemetry for u
  
```

- ✓ 로그그룹 선택·쿼리 없이 검색어·레벨 필터로 즉시 조회
- ✓ 서울·오하이오 멀티리전 통합 (드롭다운)
- ✓ Fluent Bit DaemonSet + IRSA (노드별 수집, K8s 메타데이터 부착)

메트릭 : ServiceMonitor → Prometheus



- ✓ api 6 + ai 4 패널 (ServiceMonitor 기반 Pull 수집)
- ✓ api=summary(평균) / ai=histogram(p95 분위수)
- ✓ ai 예측 캐시 적중률 ~97%
- ✓ 로그=Push, 메트릭=Pull (수집 방향 분리)
- ✓ Bedrock 회로차단기·DB 커넥션풀 등 앱 내부 지표까지

LLM 호출 분산 추적 (X-Ray)

Trace Map 1. 전체 Trace 목록

최근 15분간 수집된 모든 요청

🔍 X-Ray 그룹으로 필터링

기록 세부 정보

노드를 선택하여 세부 정보 보기

2. 개별 Trace 선택

클라이언트 → API → AI-module → DB

3. Waterfall

노드별 소요 시간 분해 (병목 진단)

이름	세그먼트 상태	응답 코드	지속 시간	Hosted in
▼ bedrock-assistant-converse				
bedrock-assistant-converse	✅ 확인	-	25.43초	
bedrock-converse-turn_0	✅ 확인	-	4.00초	
tool_search-inventory	✅ 확인	-	24밀리초	
connection	✅ 확인	-	24밀리초	
stockops	✅ 확인	-	12밀리초	
stockops	✅ 확인	-	0밀리초	
stockops	✅ 확인	-	1밀리초	
stockops	✅ 확인	-	0밀리초	
bedrock-converse-turn_1	✅ 확인	-	2.69초	
tool_search-inventory	✅ 확인	-	6밀리초	
connection	✅ 확인	-	6밀리초	
stockops	✅ 확인	-	2밀리초	
stockops	✅ 확인	-	0밀리초	

추적 데이터 수집 흐름

앱 코드 레벨 OTEL 계측

- api (Spring): micrometer-tracing-bridge-otel로 OTeI 브리지 + @Observed 커스텀 span
- ai (FastAPI): OpenTelemetry SDK + Instrumentor로 HTTP·DB 자동 계측
- sampling 1.0 전량 수집 → ADOT Collector → X-Ray

LLM 에이전트 호출 체인 추적

- 질문 1건 → Bedrock 멀티턴 → forecast 툴 → prophet 예측 → RDS
- 클라이언트 → api → ai-module → DB 단일 trace 4노드
- 핵심: 계측된 HTTP 클라이언트가 W3C traceparent를 전파 → 코드로 안 이어도 서비스 경계 자동 연결

구간별 지연 분해

- Waterfall로 각 구간(보안필터·LLM 추론·prophet·DB) 소요시간 분리
- 어느 구간이 병목인지 정량 확인 — 실측 병목은 Bedrock LLM 추론
- 동기 호출 체인이라 구간별 시간이 누적되는 구조까지 가시화



트러블슈팅

영역	문제	원인 진단	해결
보안	일부 로그인이 <u>알림 없이 누락됨</u>	CloudWatch Alarm은 상태 변경 시에만 발송 → (이미 ALARM이면 무시)	Subscription Filter → Lambda 이벤트 방식 전환
인프라	백엔드(API) 메트릭만 No data (AI는 정상)	대시보드 아닌 Prometheus 수집 차단 → 인증에 막혀 스크랩 거부 (재배포 시 설정 누락)	메트릭 공개 설정 주입 → Prometheus 타겟 정상화, 패널 복원
관측	API 호출은 <u>200 성공인데</u> X-Ray 추적 선이 안 생김	성공 응답 ≠ 정상 동작 → 연동·DB·데이터 3단 막혀 AI 호출 폴백	연동 env + 시드 데이터 → 실호출 복원, 추적 선 생성



프로젝트 핵심 성과

1 전 구간 IaC

EKS·대시보드·알림·추적까지 Terraform으로 코드화하고 GitOps로 관리. 수작업 없이 재현 가능한 인프라 구성

2 Observability 3축 직접 구축

로그·메트릭·추적을 공식 템플릿 없이 SDK 계층부터 대시보드 시각화까지 직접 구성. 모놀리식 앱의 AI 호출까지 분산 추적

3 멀티리전·멀티클라우드 운영

서울·오하이오 크로스리전 피어링으로 로그·WAF 통합 + AWS 감사로그를 Azure로 백업하는 멀티클라우드 DR 구성