aws

사용 설명서

Amazon CloudWatch Logs



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon CloudWatch Logs: 사용 설명서

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

Amazon CloudWatch Logs란 무엇입니까? 1
기능 1
관련 AWS 서비스 3
가격 책정 4
개념 4
결제 및 비용 5
로그 클래스 6
지원되는 기능 6
Amazon CloudWatch Logs Intelligent Tiering으로 스토리지 비용 최적화 9
CloudWatch Logs Intelligent Tiering 액세스 계층 9
액세스를 구성하는 작업 10
CloudWatch Logs Intelligent Tiering 활성화 10
필수 권한 10
콘솔을 사용하여 활성화 11
AWS CLI를 사용하여 활성화 11
현재 구성 보기 12
계층당 스토리지 모니터링 12
CloudWatch Logs Intelligent Tiering 요금 13
시작하기 14
사전 조건 14
에 가입 AWS 계정 14
Command Line Interface 설정 14
통합 CloudWatch 에이전트 사용 15
이전 CloudWatch 에이전트 사용 15
CloudWatch Logs 에이전트 사전 조건 16
빠른 시작: 실행 중인 EC2 Linux 인스턴스에 에이전트 설치 16
빠른 시작: 시작 시 EC2 Linux 인스턴스에 에이전트 설치 23
빠른 시작: Windows Server 2016 인스턴스에서 CloudWatch Logs 사용 27
빠른 시작: Windows Server 2012 및 Windows Server 2008 인스턴스에서 CloudWatch Logs 사용 38
CloudWatch Logs 에이전트 상태 보고 48
CloudWatch Logs 에이전트 시작 48
CloudWatch Logs 에이전트 중지 49
CloudWatch Logs 에이전트 참조 49

를 사용한 빠른 시작 CloudFormation 61
HTTP 엔드포인트를 통한 로그 수집 62
일반적인 동작 62
베어러 토큰 인증 63
옵션 1: AWS 콘솔을 사용하여 빠른 시작 64
옵션 2: 수동 설정 66
CloudWatch Logs API 키 생성 및 사용에 대한 권한 제어 68
API 키 교체 71
손상된 API 키에 대한 응답 72
API 키에 대한 보안 모범 사례 73
CloudTrail을 사용하여 API 키 사용량 로깅 74
OTLP 엔드포인트 75
요청 형식 75
요청 예제 75
응답 76
OTLP 관련 동작 77
HLC 엔드포인트 77
입력 모드 77
이벤트 필드(필수) 77
시간 필드(선택 사항) 78
Content-Type 78
허용되는 JSON 값 유형 78
엔드포인트 형식 79
요청 형식 80
요청 예제 81
모범 사례 81
제한 사항 81
NDJSON 엔드포인트 82
요청 형식 82
허용되는 JSON 값 유형 82
타임스탬프 필드 84
잘못된 줄 84
요청 예제 84
응답 84
모범 사례 85
제한 사항 86

구조화된 JSON 엔드포인트 ... 86
요청 형식 ... 86
허용되는 JSON 값 유형 ... 87
타임스탬프 필드 ... 87
요청 예제 ... 88
응답 ... 88
모범 사례 ... 89
제한 사항 ... 89
HTTP 수집 엔드포인트 비교 ... 90
엔드포인트 선택 ... 91
Syslog 수집 ... 92
지원되는 프로토콜 및 포트 ... 92
지원되는 syslog 형식 ... 93
RFC 5424 추출 필드 ... 93
RFC 3164 추출 필드 ... 94
Cisco FTD/ASA 추출 필드 ... 94
메시지 전송 ... 95
할당량 및 제한 ... 96
설정 ... 96
사전 조건 ... 97
1단계: VPC 생성 또는 식별 ... 98
2단계: 보안 그룹 만들기 ... 99
3단계: VPC 엔드포인트 생성 ... 100
4단계: 로그 그룹 생성 ... 102
5단계: 리소스 정책 추가 ... 103
6단계: syslog 구성 생성 ... 104
7단계: 전송 및 필드 추출 확인 ... 105
디바이스 구성 ... 106
rsyslog ... 107
syslog-ng ... 108
빠른 검증 ... 110
네트워크 어플라이언스 ... 110
TLS 인증서 신뢰 ... 111
모니터링 ... 111
Syslog 지표 ... 111
삭제 이유 ... 112

연결 거부 이유 112
권장되는 경보 113
자동 대시보드에서 syslog 지표 보기 113
샘플 쿼리 114
보안 및 액세스 모니터링 114
운영 가시성 115
방화벽 및 네트워크 디바이스 쿼리 116
VPC 엔드포인트 정책 116
지원되는 조건 키 117
특정 로그 그룹으로 제한 118
모든 트래픽 거부 118
기본 정책으로 재설정 119
정책 전파 119
문제 해결 119
로그 그룹에 표시되지 않는 메시지 119
TLS 핸드셰이크 실패 121
UDP 메시지 손실 121
연결이 거부됨 121
속도 제한으로 인해 삭제된 메시지 122
AWS SDKs 작업 123
로그 관리 125
데이터 소스 검색 및 관리 125
CloudWatch Logs 데이터 소스란 무엇입니까? 125
시작하는 방법 126
시스템 필드 127
데이터 원본 액세스 128
로그 그룹과의 관계 128
데이터 소스에서 활성화된 기능 129
지원되는 데이터 원본 129
데이터 소스 AWS 서비스 에 지원됨 130
데이터 소스에 지원되는 타사 소스 134
사용자 지정 소스 152
CloudWatch Log Analytics를 사용하여 로그 데이터 분석 153
Log Analytics로 수행할 수 있는 작업 153
Log Analytics 시작하기 153
CloudWatch Logs Insights를 사용한 로그 분석 154

지원되는 쿼리 언어 157
CloudWatch Logs Insights 쿼리 언어(Logs Insights QL) 157
OpenSearch Piped Processing Language(PPL) 243
OpenSearch Structured Query Language(SQL) 253
자연어를 사용하여 CloudWatch Logs Insights 쿼리를 생성하고 업데이트 263
예제 쿼리 264
서비스 개선을 위한 데이터 사용 선택 해제 267
지원되는 로그 및 검색되는 필드 267
JSON 로그의 필드 269
쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성 272
자동으로 인덱싱된 필드 275
필드 인덱스 구문 및 할당량 276
계정 수준 필드 인덱스 정책 생성 281
로그 그룹 수준 필드 인덱스 정책 생성 282
쿼리 생성 시 로그 그룹 선택 옵션 283
필드 인덱스 정책 삭제의 영향 284
패싯을 사용하여 로그 그룹화 및 탐색 284
패싯 기반 쿼리를 실행하려면 285
패싯 기반 쿼리를 저장하려면 285
계정 수준 패싯을 생성하려면 286
APIs 사용한 패싯 관리 286
CloudWatch Logs Insights에서 주변 로그 보기 286
주변 로그의 작동 방식 287
주변 로그 보기 287
주변 로그 내에서 검색 288
패턴 분석 289
패턴 분석 시작하기 290
패턴 명령에 대한 세부 정보 292
쿼리 저장 및 재실행 293
파라미터와 함께 저장된 쿼리 사용 298
대시보드에 쿼리 추가 또는 쿼리 결과 내보내기 300
실행 중인 쿼리 또는 쿼리 기록 보기 301
를 사용하여 쿼리 결과 암호화 AWS Key Management Service 302
제한 사항 302
1단계: 생성 AWS KMS key 302
2단계: KMS 키에 대한 권한 설정 303

3단계: KMS 키를 쿼리 결과와 연결 305
4단계: 계정의 쿼리 결과에서 키 연결 해제 305
CloudWatch Logs Insights 쿼리 결과에서 자연어 요약 생성 305
작동 방식 305
리전 가용성 및 데이터 처리 306
시작하기 307
권한 307
데이터 개인정보보호 307
예약된 쿼리를 사용하여 로그 분석 자동화 308
예약된 쿼리 개념 이해 308
IAM 역할 분리 308
교차 리전 및 교차 계정 사용 311
일정 표현식 및 시간대 처리 312
쿼리 언어 선택 313
대상 선택 및 사용 사례 314
쿼리 결과 형식 및 구조 315
recordsScanned, recordsMatched 및 resultCount 이해 317
일정 표현식 참조 320
모범 사례 323
예약된 쿼리 시작하기 324
예약된 쿼리 생성 324
예약된 쿼리 보기 및 관리 328
예약된 쿼리 실행 기록 보기 330
예약된 쿼리 업데이트 331
예약된 쿼리에 대한 S3 대상 구성 333
동일한 계정의 Amazon S3 버킷에 결과 전송 333
다른 계정의 Amazon S3 버킷에 결과 전송 334
고객 관리형 AWS KMS 키를 사용하여 결과 암호화 336
예약된 쿼리에 대한 조회 테이블 대상 구성 338
예약된 쿼리 문제 해결 339
권한 오류와 함께 쿼리 실행 실패 339
쿼리 제한 시간 339
대상 처리 실패 340
잘못된 쿼리 오류 340
동시성 오류 쿼리 341
로그 이상 탐지 342

이상 및 패턴의 심각도 및 우선순위 ... 343
이상 가시성 시간 ... 343
이상 항목 억제 ... 343
자주 묻는 질문(FAQ) ... 344
CloudWatch Logs Insights에서 이상 탐지 사용 ... 345
로그 그룹에서 이상 탐지 활성화 ... 345
발견된 이상 보기 ... 346
로그 이상 탐지기에 대한 경보 생성 ... 349
로그 이상 탐지기가 게시한 지표 ... 351
를 사용하여 이상 탐지기 및 그 결과 암호화 AWS KMS ... 351
제한 사항 ... 352
CloudWatch Logs Live Tail로 문제 해결 ... 357
를 사용하여 Live Tail 세션 시작 AWS CLI ... 357
print-only ... 358
interactive ... 358
콘솔에서 Live Tail 세션 시작 ... 360
교차 계정 교차 리전 로그 중앙 집중화 ... 363
데이터 중앙 집중화 개념 ... 363
로그 중앙 집중화 설정 ... 365
사전 조건 ... 365
대상 로그 그룹 이름 사용자 지정 ... 367
중앙 집중화 규칙 생성 ... 369
중앙 집중화 규칙 수정 ... 371
중앙 집중화 규칙 보기 ... 372
중앙 집중화 규칙 삭제 ... 372
중앙 집중화 규칙 모니터링 및 문제 해결 ... 373
중앙 집중화 규칙 상태 ... 373
를 사용하여 중앙 집중화 API 호출 모니터링 AWS CloudTrail ... 374
모니터링 권장 사항 ... 374
로그 그룹 및 로그 스트림 작업 ... 377
로그 그룹 생성 ... 377
로그 그룹에 로그 전송 ... 378
로그 데이터 보기 ... 378
필터 패턴을 사용하여 로그 데이터 검색 ... 379
콘솔을 사용하여 로그 항목 검색 ... 379
를 사용하여 로그 항목 검색 AWS CLI ... 380

지표에서 로그로 피벗 적용 380
문제 해결 381
로그 데이터 보존 기간 변경 381
로그 그룹이 삭제되지 않도록 보호 382
로그 그룹이 삭제되지 않도록 보호 382
삭제 방지 활성화 382
로그 그룹 태그 지정 383
태그 기본 사항 384
태그 지정을 사용하여 비용 추적 384
태그 제한 사항 384
를 사용하여 로그 그룹에 태그 지정 AWS CLI 385
CloudWatch Logs API를 사용하여 로그 그룹에 태그 지정 386
를 사용하여 로그 데이터 암호화 AWS KMS 386
제한 사항 387
1단계: AWS KMS 키 생성 302
2단계: KMS 키에 대한 권한 설정 388
3단계: 호출 IAM 보안 주체에 대한 권한 설정 391
4단계: KMS 키를 로그 그룹에 연결 356
5단계: 로그 그룹에서 키 연결 해제 356
KMS 키 및 암호화 컨텍스트 394
마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원 397
데이터 보호 정책 이해 400
데이터 보호 정책을 생성하거나 사용하는 데 필요한 IAM 권한 402
계정 전체 데이터 보호 정책 생성 407
단일 로그 그룹에 대한 데이터 보호 정책 생성 411
마스킹 처리되지 않은 데이터 보기 414
감사 결과 보고서 414
보호할 수 있는 데이터 유형 416
수집 중 로그 변환 458
로그 트랜스포머 생성 및 관리 459
계정 수준 트랜스포머 정책 생성 460
계정 수준 트랜스포머 정책 편집 또는 삭제 462
처음부터 로그 그룹 수준 로그 트랜스포머 생성 462
기존 트랜스포머를 복사하여 로그 그룹 수준 트랜스포머 생성 463
로그 그룹 수준 트랜스포머 편집 464
로그 그룹 수준 트랜스포머 삭제 465

구성 가능한 구문 분석기 유형 프로세서 ... 465
parseJSON ... 466
grok ... 467
csv ... 496
parseKeyValue ... 498
AWS 벤딩 로그용 기본 제공 프로세서 ... 500
parseWAF ... 500
parsePostgres ... 503
parseCloudfront ... 504
parseRoute53 ... 506
parseVPC ... 507
parseToOCSF ... 508
문자열 변형 프로세서 ... 509
lowerCaseString ... 510
upperCaseString ... 511
splitString ... 512
substituteString ... 514
trimString ... 517
JSON 변형 프로세서 ... 518
addKeys ... 519
deleteKeys ... 520
moveKeys ... 521
renameKeys ... 523
copyValue ... 524
listToMap ... 526
데이터타입 컨버터 프로세서 ... 531
typeConverter ... 531
datetimeConverter ... 533
변환 지표 및 오류 ... 534
S3 Tables Integration을 사용한 액세스 로그 ... 536
S3 테이블 통합 이해 ... 536
핵심 구성 요소 ... 536
S3 테이블로의 데이터 흐름 ... 536
S3 테이블 통합을 사용해야 하는 경우 ... 537
사전 조건 ... 537
IAM 권한 ... 538

KMS 키 정책(암호화된 데이터용) ... 539
시작하기 ... 540
Amazon S3 테이블과 AWS 분석 서비스 통합 - Amazon S3 콘솔 사용(링크) ... 542
Lake Formation 권한 구성 ... 542
분석 도구를 사용하여 연결 ... 543
지표 필터 ... 544
개념 ... 545
지표 필터에 대한 필터 패턴 구문 ... 546
지표 필터에 대한 지표 값 구성 ... 547
로그 이벤트에서 지표로 차원 게시 ... 548
로그 이벤트의 값을 사용하여 지표 값 증가 ... 551
지표 필터 생성 ... 552
로그 그룹에 대한 지표 필터 생성 ... 552
예제: 로그 이벤트 수 계산 ... 554
예제: 단어의 출현 횟수 계산 ... 555
예제: HTTP 404 코드 수 계산 ... 557
예제: HTTP 4xx 코드 수 계산 ... 559
예제: Apache 로그에서 필드 추출 후 차원 할당 ... 560
지표 필터 나열 ... 562
지표 필터 삭제 ... 563
구독 필터 ... 565
개념 ... 566
로그 그룹 수준 구독 필터 ... 568
예제 1: Amazon Kinesis Data Streams를 사용한 구독 필터 ... 569
예제 2:를 사용한 구독 필터 AWS Lambda ... 575
예제 3: Amazon Data Firehose에 대한 구독 필터 ... 578
예제 4: Amazon OpenSearch Service를 사용한 구독 필터 ... 585
계정 수준 구독 필터 ... 586
예제 1: Amazon Kinesis Data Streams를 사용한 구독 필터 ... 587
예제 2:를 사용한 구독 필터 AWS Lambda ... 593
예제 3: Amazon Data Firehose에 대한 구독 필터 ... 597
교차 계정 교차 리전 구독 ... 604
Amazon Kinesis Data Streams를 사용한 교차 계정 교차 리전 로그 데이터 공유 ... 605
Firehose를 사용한 교차 계정 교차 리전 로그 데이터 공유 ... 624
Amazon Kinesis Data Streams를 사용한 교차 계정 교차 리전 계정 수준 구독 ... 638
Firehose를 사용한 교차 계정 교차 리전 계정 수준 구독 ... 656

혼동된 대리자 방지 668
로그 재귀 방지 669
필터 패턴 구문 671
지원되는 정규식 671
정규식을 사용하여 일치하는 용어 검색 674
비정형 로그 이벤트에서 일치하는 용어 검색 675
JSON 로그 이벤트에서 일치하는 용어 검색 678
공백으로 구분된 로그 이벤트에서 일치하는 용어 검색 686
AWS 서비스에서 로깅 활성화 691
지원되는 로그 대상 및 권한 694
추가 권한 [V1]이 필요한 로깅 702
CloudWatch Logs로 전송된 로그 703
Amazon S3로 전송된 로그 706
Firehose에 전송된 로그 711
추가 권한 [V2]가 필요한 로깅 713
로그 전송 설정 예제 713
CloudWatch Logs로 전송된 로그 717
Amazon S3로 전송된 로그 721
Firehose에 전송된 로그 727
X-Ray로 전송된 트레이스 730
서비스별 권한 733
콘솔별 권한 734
교차 계정 전송 예제 735
전송 소스 생성 736
Amazon S3 버킷으로의 전송 구성 737
Firehose 스트림으로의 전송 구성 740
교차 서비스 혼동된 대리인 방지 742
원격 측정 활성화 규칙을 사용하여 로그 활성화 자동화 743
규칙 평가 계층 구조 744
활성화 규칙 생성 744
타사 소스에서 로깅 활성화 745
직접 타사 통합 745
AWS Security Hub CSPM을 통한 추가 소스 745
Amazon S3로 로그 데이터 내보내기 751
개념 752
콘솔을 사용하여 Amazon S3로 로그 데이터 내보내기 753

동일 계정 내보내기(콘솔) ... 753
교차 계정 내보내기(콘솔) ... 760
를 사용하여 Amazon S3로 로그 데이터 내보내기 AWS CLI ... 769
동일 계정 내보내기(CLI) ... 769
교차 계정 내보내기(CLI) ... 776
내보내기 작업 설명(CLI) ... 786
내보내기 작업 취소(CLI) ... 787
OpenSearch Service로 데이터 스트리밍 ... 788
사전 조건 ... 788
OpenSearch Service에 대한 로그 그룹 구독 ... 789
코드 예제 ... 791
기본 사항 ... 792
CloudWatch Logs 시작 ... 793
작업 ... 794
시나리오 ... 869
컨테이너 서비스 연결 구성 ... 869
첫 번째 서버리스 함수 생성 ... 886
syslog 수집 기본 사항 알아보기 ... 898
대용량 쿼리 실행 ... 909
예약된 이벤트를 사용하여 Lambda 함수 간접 호출 ... 947
를 사용하여 조회 테이블 암호화 AWS KMS ... 949
CloudWatch Logs가 조회 테이블에 AWS KMS 를 사용하는 방법 ... 949
필요한 권한 ... 950
1단계: AWS KMS 키 생성 ... 950
2단계: KMS 키에 대한 권한 설정 ... 951
3단계: KMS 키를 조회 테이블에 연결 ... 953
고려 사항 ... 953
보안 ... 954
데이터 보호 ... 955
저장된 데이터 암호화 ... 956
전송 중 데이터 암호화 ... 956
ID 및 액세스 관리 ... 956
Authentication ... 956
액세스 관리 ... 957
액세스 관리 개요 ... 957
ID 기반 정책(IAM 정책) 사용 ... 962

규정 준수 확인 990
복원력 990
인프라 보안 990
인터페이스 VPC 엔드포인트 991
가용성 991
CloudWatch Logs에 대한 VPC 엔드포인트 생성 991
VPC와 CloudWatch Logs 간 연결 테스트 992
CloudWatch Logs VPC 엔드포인트에 대한 액세스 제어 992
VPC 컨텍스트 키에 대한 지원 994
를 사용하여 API 및 콘솔 작업 로깅 AWS CloudTrail 995
CloudTrail에서의 쿼리 생성 정보 999
로그 파일 항목 이해 1001
CloudWatch 지표를 사용한 사용량 모니터링 1003
CloudWatch Logs 지표 1003
CloudWatch Logs 지표의 차원 1007
로그 트랜스포머 지표 및 차원 1008
중앙 집중화 지표 및 차원 1009
CloudWatch Logs 서비스 사용량 지표 1012
서비스 할당량 1015
CloudWatch Logs 서비스 할당량 관리 1025
문서 기록 1027
AWS 용어집 1042
..... mxliii

Amazon CloudWatch Logs란 무엇입니까?

Amazon CloudWatch Logs를 사용하여 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스, Route 53 및 기타 소스에서 로그 파일을 모니터링 AWS CloudTrail, 저장 및 액세스할 수 있습니다.

CloudWatch Logs를 사용하면 확장성이 뛰어난 단일 AWS 서비스에서 사용하는 모든 시스템, 애플리케이션 및 서비스의 로그를 중앙 집중화할 수 있습니다. 그런 다음 로그를 쉽게 보고, 특정 오류 코드 또는 패턴이 있는지 검색하고, 특정 필드를 기반으로 필터링하거나, 향후 분석을 위해 안전하게 보관할 수 있습니다. CloudWatch Logs를 사용하면 소스와 관계없이 모든 로그를 시간순으로 정렬된 일관된 단일 이벤트 흐름으로 볼 수 있습니다.

또한 CloudWatch Logs는 강력한 쿼리 언어로 로그를 쿼리하고, 로그의 민감한 데이터를 감사 및 마스킹 처리하고, 필터 또는 임베디드 로그 형식을 사용하여 로그에서 지표를 생성할 수 있도록 지원합니다.

CloudWatch Logs는 두 가지 로그 클래스를 지원합니다. CloudWatch Logs Standard 로그 클래스의 로그 그룹은 모든 CloudWatch Logs 기능을 지원합니다. CloudWatch Logs Infrequent Access 로그 클래스의 로그 그룹은 수집 요금이 보다 저렴하고 Standard 클래스 기능의 일부를 지원합니다. 자세한 내용은 로그 클래스 단원을 참조하십시오.

기능

- 유연성을 위한 두 가지 로그 클래스 - CloudWatch Logs는 두 가지 로그 클래스를 제공하므로 자주 액세스하지 않는 로그에 비용 효율적인 옵션을 사용할 수 있습니다. 실시간 모니터링 또는 기타 기능이 필요한 로그를 위한 전체 기능 옵션도 있습니다. 자세한 내용은 로그 클래스 단원을 참조하십시오.
- 로그 데이터 쿼리 - CloudWatch Logs Insights를 사용하여 로그 데이터를 대화식으로 검색 및 분석할 수 있습니다. 작업상의 문제에 보다 효율적이고 효과적으로 대처할 수 있도록 쿼리를 수행할 수 있습니다. CloudWatch Logs Insights에는 몇 가지 간단하지만 강력한 명령과 함께 특별히 구축된 쿼리 언어가 포함되어 있습니다. 시작하는 데 도움이 되는 샘플 쿼리, 명령 설명, 쿼리 자동 완성 및 로그 필드 검색이 제공됩니다. 샘플 쿼리는 여러 유형의 AWS 서비스 로그에 포함됩니다. 시작하려면 CloudWatch Logs Insights를 사용한 로그 분석을 참조하세요.
- 쿼리를 보다 효율적으로 만들기 위한 필드 인덱스 생성 - 로그 이벤트에서 필드의 필드 인덱스를 생성할 수 있습니다. 그런 다음 CloudWatch Logs Insights 쿼리에서 필드 인덱스를 사용하면 쿼리는 인덱싱된 필드를 포함하지 않는 것으로 알려진 로그 이벤트 처리를 건너뛰려고 시도합니다. 이 쿼리는 쿼리의 스캔 볼륨을 줄여 결과를 더 빠르게 반환할 수 있습니다. 시작하려면 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성을 참조하세요.

- Live Tail을 사용한 탐지 및 디버깅 - Live Tail을 사용하면 모은 새 로그 이벤트의 스트리밍 목록을 확인하여 인시던트 문제를 신속하게 해결할 수 있습니다. 모은 로그를 거의 실시간으로 보고 필터링하고 강조 표시할 수 있으므로 문제를 빠르게 탐지하고 해결할 수 있습니다. 지정한 항을 기반으로 로그를 필터링하고 원하는 항목을 빠르게 찾을 수 있도록 지정된 항이 포함된 로그를 강조 표시할 수도 있습니다. 자세한 내용은 CloudWatch Logs Live Tail로 문제 해결 단원을 참조하십시오.
- Amazon EC2 인스턴스에서 로그 모니터링 - CloudWatch Logs를 사용하여 로그 데이터를 통해 애플리케이션 및 시스템을 모니터링할 수 있습니다. 예를 들어 CloudWatch Logs에서는 애플리케이션 로그에서 발생하는 오류의 수를 추적하고 오류 비율이 지정한 임곗값을 초과할 때마다 알림을 전송할 수 있습니다. 는 모니터링하는 데 로그 데이터를 사용하므로 코드를 변경할 필요가 없습니다. CloudWatch Logs는 모니터링하는 데 로그 데이터를 사용하므로 코드를 변경할 필요가 없습니다. 예를 들어 특정 리터럴 문자(예: 'NullReferenceException')에 대한 애플리케이션 로그를 모니터링하거나 로그 데이터의 특정 위치(예: Apache 액세스 로그의 '404' 상태 코드)에서 리터럴 문자의 출현 횟수를 계산할 수 있습니다. 검색할 단어가 발견되면 CloudWatch Logs는 지정한 CloudWatch 지표로 데이터를 보고합니다. 로그 데이터는 전송 시는 물론 저장 시에도 암호화됩니다. 시작하려면 CloudWatch Logs 시작하기을 참조하세요.
- AWS CloudTrail 로깅된 이벤트 모니터링 - CloudWatch에서 경보를 생성하고 CloudTrail에서 캡처한 특정 API 활동에 대한 알림을 수신하고 알림을 사용하여 문제 해결을 수행할 수 있습니다. 시작하려면 AWS CloudTrail 사용 설명서의 CloudWatch Logs로 CloudTrail 이벤트 보내기를 참조하세요.
- 민감한 데이터 감사 및 마스킹 - 로그에 민감한 데이터가 있는 경우 데이터 보호 정책을 통해 이를 보호할 수 있습니다. 이러한 정책을 통해 민감한 데이터를 감사하고 마스킹 처리할 수 있습니다. 데이터 보호를 활성화하면 선택한 데이터 식별자와 일치하는 중요한 데이터가 기본적으로 마스킹 처리됩니다. 자세한 내용은 마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원 단원을 참조하십시오.
- 로그 보존 - 기본적으로 로그는 무기한으로 보존되며, 만료되지 않습니다. 로그 그룹별로 보존 정책을 조정할 수 있어 무기한 보존 유지하거나 10년 및 하루 중 보존 기간을 선택할 수 있습니다.
- 삭제 방지 - 로그 그룹 및 해당 로그 스트림의 우발적 삭제를 방지하는 보호 장치입니다. 로그 그룹에서 활성화하면 삭제 방지 기능은 명시적으로 비활성화될 때까지 모든 삭제 작업을 차단합니다. 기본적으로 삭제 방지는 활성화되어 있지 않습니다. 이 선택적 기능은 감사 데이터가 포함된 로그 그룹, 문제 해결 및 분석을 위한 프로덕션 애플리케이션 로그 등 중요한 운영 및 규정 준수 데이터가 의도하지 않게 제거되지 않도록 보호하는 데 도움이 됩니다.
- 로그 데이터 아카이브 - CloudWatch Logs를 사용하여 내구성이 뛰어난 스토리지에 로그 데이터를 저장할 수 있습니다. CloudWatch Logs 에이전트를 사용하면 호스트에서 로그 서비스로 로테이션 된 로그 데이터와 로테이션 되지 않은 로그 데이터를 모두 쉽고 빠르게 전송할 수 있습니다. 그런 다음 필요할 때 원시 로그 데이터에 액세스할 수 있습니다.

- Route 53 DNS 쿼리 로그 - CloudWatch Logs를 사용하여 Route 53이 수신하는 DNS 쿼리에 대한 정보를 로그할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 DNS 쿼리 로깅을 참조하세요.
- 계정 및 리전 간 로그 중앙 집중화 - CloudWatch Logs 중앙 집중화를 사용하여 다중 계정 및 리전 환경에서 수집된 로그 데이터를 중앙 리전 및 계정으로 복제하는 교차 계정 및 교차 리전 중앙 집중화 규칙을 정의할 수 있습니다. 복원력을 높이기 위해 중앙 계정 내에서 백업 리전을 구성하고, 중앙 계정에서 새로 생성된 로그 그룹에 대한 암호화 설정을 정의하고, 향상된 필터링 기능으로 특정 소스 계정 및 리전의 로그 이벤트에 대한 지표 및 구독 필터를 생성할 수 있습니다.

관련 AWS 서비스

다음 서비스가 CloudWatch Logs와 함께 사용됩니다.

- AWS CloudTrail는 , AWS Management Console AWS Command Line Interface (AWS CLI) 및 기타 서비스의 호출을 포함하여 계정의 CloudWatch Logs API에 대한 호출을 모니터링할 수 있는 웹 서비스입니다. CloudTrail 로깅이 활성화되면 CloudTrail은 계정에서 API 호출을 포착하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 각 로그 파일에는 하나 이상의 레코드가 포함될 수 있으며, 요청을 충족하기 위해 수행해야 하는 작업의 수에 따라 그 수가 결정됩니다. 에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 란 무엇입니까 AWS CloudTrail?를 AWS CloudTrail참조하세요. CloudWatch가 CloudTrail 로그 파일에 기록하는 데이터 유형의 예제는 에서 CloudWatch Logs API 및 콘솔 작업 로깅 AWS CloudTrail을 참조하세요.
- AWS Identity and Access Management (IAM)는 사용자의 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 되는 웹 서비스입니다. IAM을 사용하여 AWS 리소스를 사용할 수 있는 사람을 제어(인증)하고 이들이 사용할 수 있는 리소스 및 그 사용 방법을 제어(권한 부여)합니다. 자세한 내용은 IAM 사용 설명서의 IAM이란 무엇입니까?를 참조하세요.
- Amazon Kinesis Data Streams는 신속하고 지속적인 데이터 인테이크 및 집계를 위해 사용할 수 있는 웹 서비스입니다. 사용되는 데이터 유형으로는 IT 인프라 로그 데이터, 애플리케이션 로그, 소셜 미디어, 시장 데이터 피드, 웹 클릭스트림 데이터 등이 있습니다. 데이터 인테이크 및 처리에 대한 응답이 실시간으로 이루어지기 때문에 일반적으로 간소화된 방식으로 처리가 됩니다. 자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서의 Amazon Kinesis Data Streams 이란 무엇입니까?를 참조하세요.
- AWS Lambda는 새 정보에 신속하게 응답하는 애플리케이션을 구축하는 데 사용할 수 있는 웹 서비스입니다. Lambda 함수 형태로 애플리케이션 코드를 업로드하면 는 고가용성 컴퓨팅 인프라에서 코드를 실행하고 서버 및 운영 체제 유지 관리, 용량 프로비저닝 및 자동 조정, 코드 및 보안 패치 배포, 코드 모니터링 및 로깅 등 모든 컴퓨팅 리소스 관리를 수행합니다. Lambda가 지원하는 언어 중 하

나로 코드를 공급하기만 하면 됩니다. 자세한 내용은 AWS Lambda 개발자 안내서의 What is AWS Lambda?를 참조하세요.

가격 책정

가입하면 AWS 프리 티어를 사용하여 CloudWatch Logs를 무료로 시작할 AWS수 있습니다.

표준 요금은 CloudWatch Logs를 사용하여 다른 서비스가 저장한 로그(예: Amazon VPC 흐름 로그 및 Lambda 로그)에 적용됩니다.

요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하세요.

CloudWatch Logs 및 CloudWatch의 비용 및 사용량을 분석하는 방법에 대한 자세한 정보와 비용을 절감하는 방법에 대한 모범 사례는 CloudWatch 결제 및 비용을 참조하세요.

Amazon CloudWatch Logs 개념

CloudWatch Logs의 이해 및 사용에 핵심이 되는 용어 및 개념에 대한 설명은 다음과 같습니다.

로그 클래스

CloudWatch Logs는 두 가지 로그 그룹 클래스를 제공합니다. Standard 로그 클래스는 실시간 모니터링이 필요한 로그 또는 자주 액세스하는 로그를 위한 전체 기능 옵션입니다. Infrequent Access 로그 클래스는 자주 액세스하지 않는 로그를 위한 저렴한 옵션입니다. 이 로그 클래스는 Standard 로그 클래스 기능의 일부를 지원합니다.

로그 이벤트

로그 이벤트는 모니터링 중인 애플리케이션 또는 리소스에 기록된 일부 활동에 대한 레코드입니다. CloudWatch Logs가 파악한 로그 이벤트 레코드에는 이벤트가 발생한 시점에 대한 타임스탬프와 원시 이벤트 메시지 등 두 개의 속성이 포함되어 있습니다. 각 메시지는 UTF-8로 인코딩되어야 합니다.

로그 스트림

로그 스트림은 동일한 소스를 공유하는 로그 이벤트 시퀀스입니다. 보다 구체적으로 말하자면, 로그 스트림은 모니터링 중인 애플리케이션 인스턴스나 리소스에서 나온 이벤트의 시퀀스를 표시하는 데 주로 사용됩니다. 예를 들어, 로그 스트림은 특정 호스트의 Apache 액세스 로그에 연결될 수 있습니다. 로그 스트림이 더 이상 필요하지 않으면 aws logs delete-log-stream 명령을 사용하여 이를 삭제할 수 있습니다.

로그 그룹

로그 그룹은 동일한 보존 기간, 모니터링 및 액세스 제어 설정을 공유하는 로그 스트림 그룹을 정의합니다. 각 로그 스트림은 하나의 로그 그룹에 속해야 합니다. 예를 들어, 각 호스트의 Apache 액세스 로그에 대해 별도의 로그 스트림이 있으면 로그 스트림을 `MyWebsite.com/Apache/access_log`라는 하나의 로그 그룹으로 묶을 수 있습니다.

하나의 로그 그룹이 가질 수 있는 로그 스트림의 수는 제한이 없습니다.

지표 필터

지표 필터를 사용하여 수집된 이벤트에서 지표 관찰값을 추출하고 이를 CloudWatch 지표의 데이터 요소로 변환할 수 있습니다. 지표 필터는 로그 그룹에 할당이 되고, 로그 그룹에 할당된 모든 필터는 로그 스트림에 적용됩니다.

보존 기간 설정

보존 기간 설정은 CloudWatch Logs에 로그 이벤트를 보관하는 기간을 설정하는 데 사용할 수 있습니다. 기간이 만료된 로그 이벤트는 자동으로 삭제됩니다. 지표 필터와 마찬가지로 보존 기간 설정 역시 로그 그룹에 할당이 되며, 로그 그룹에 할당된 보존 기간은 로그 스트림에 적용됩니다.

삭제 방지

삭제 방지는 로그 그룹 및 해당 로그 스트림의 우발적 삭제를 방지하는 보호 장치입니다. 로그 그룹에서 활성화하면 삭제 방지 기능은 명시적으로 비활성화될 때까지 모든 삭제 작업을 차단합니다. 기본적으로 삭제 방지는 활성화되어 있지 않습니다. 이 선택적 기능은 감사 데이터가 포함된 로그 그룹, 문제 해결 및 분석을 위한 프로덕션 애플리케이션 로그 등 중요한 운영 및 규정 준수 데이터가 의도하지 않게 제거되지 않도록 보호하는 데 도움이 됩니다.

Amazon CloudWatch Logs 과금 정보 및 비용

CloudWatch Logs 및 CloudWatch의 비용 및 사용량을 분석하는 방법에 대한 자세한 정보와 비용을 절감하는 방법에 대한 모범 사례는 CloudWatch 결제 및 비용을 참조하세요.

요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하세요.

가입하면 AWS 프리 티어를 사용하여 CloudWatch Logs를 무료로 시작할 AWS수 있습니다.

표준 요금은 CloudWatch Logs를 사용하여 다른 서비스가 저장한 로그(예: Amazon VPC 흐름 로그 및 Lambda 로그)에 적용됩니다.

로그 클래스

CloudWatch Logs는 두 가지 로그 그룹 클래스를 제공합니다.

- CloudWatch Logs Standard 로그 클래스는 실시간 모니터링이 필요한 로그 또는 자주 액세스하는 로그를 위한 전체 기능 옵션입니다.
- CloudWatch Logs Infrequent Access 로그 클래스는 로그를 비용 효율적으로 통합하는 데 사용할 수 있는 새 로그 클래스입니다. 이 로그 클래스는 관리형 수집, 스토리지, 교차 계정 로그 분석, 암호화를 비롯한 CloudWatch Logs 기능 일부를 더 저렴한 GB당 수집 요금으로 제공합니다. Infrequent Access 로그 클래스는 자주 액세스하지 않는 로그에 대한 임시 쿼리 및 사후 포렌식 분석에 적합합니다.

> **Note**
>
> 요금의 경우 Standard 액세스 로그 클래스와 Infrequent Access 로그 클래스는 수집 비용만 다릅니다. 스토리지 요금과 CloudWatch Logs Insights 요금은 각 로그 클래스에서 동일합니다.

CloudWatch Logs 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

> **Important**
>
> 로그 그룹이 생성된 후에는 해당 로그 클래스를 변경할 수 없습니다.

지원되는 기능

다음 표에는 각 로그 클래스의 기능이 나열되어 있습니다.

기능	표준	Infrequent Access	
완전 관리형 로그 수집 및 스토리지	예 ✓	예 ✓	
교차 계정 기능	예 ✓	예 ✓	
를 사용한 암호화 AWS KMS	예 ✓	예 ✓	

기능	표준	Infrequent Access	
CloudWatch Logs Insights 쿼리 명령	예 ✓	예 ✓ (대부분의 명령 - 로그 클래스에서 지원되는 Logs Insights QL 명령을 참조하세요.)	
CloudWatch Logs Insights 검색된 필드	예 ✓	예 ✓	
패싯	예 ✓	아니요	
CloudWatch Pipeline을 사용하여 로그 변환	예 ✓	예 ✓	
Amazon S3로 내보내기	예 ✓	예 ✓	
S3 테이블 통합	예 ✓	예 ✓	
예약된 쿼리	예 ✓	예 ✓	
OpenSearch PPL 또는 OpenSearch SQL을 사용하여 CloudWatch Logs Insights에서 쿼리	예 ✓	예 ✓	
자연어 쿼리 지원	예 ✓	아니요	
CloudWatch Logs 이상 탐지	예 ✓	아니요	
Live Tail	예 ✓	아니요	
필드 인덱싱	예 ✓	아니요	
이전 시간 범위와 비교	예 ✓	아니요	
구독 필터	예 ✓	아니요	

기능	표준	Infrequent Access	
GetLogEvents 및 FilterLogEvents API 작업	예 ✓	지원되지 않음. Infrequent Access 로그 클래스의 로그 그룹에 저장된 로그 이벤트를 보려면 CloudWatch Logs Insights를 사용합니다.	
지표 필터	예 ✓	아니요	
Container Insights 로그 수집	예 ✓	아니요	
Lambda Insights 로그 수집	예 ✓	아니요	
마스킹을 통해 민감한 데이터 보호	예 ✓	예 ✓	
임베디드 지표 형식	예 ✓	아니요	

> **Note**
>
> 이 두 로그 클래스 외에도 `Delivery` 로그 클래스가 있습니다. `Delivery` 또는 Amazon S3 에 저장할 로그를 전송할 때만 AWS Lambda 로그 클래스를 사용합니다 Amazon Data Firehose. 전송 클래스의 로그 그룹에 있는 로그 이벤트는 이틀 CloudWatch Logs 동안에 보관됩니다. 이 보존 기간은 고정되어 있으며 변경할 수 없습니다. 이 로그 클래스는 CloudWatch Logs Insights 쿼리와 같은 풍부한 CloudWatch Logs 기능을 제공하지 않습니다.

Amazon CloudWatch Logs Intelligent Tiering으로 스토리지 비용 최적화

Amazon CloudWatch Logs Intelligent Tiering은 액세스 패턴인 Standard, Infrequent Access 및 Archive Instant Access에 따라 세 가지 액세스 계층에서 로그 데이터를 자동으로 분류합니다. Intelligent Tiering을 사용하면 운영 오버헤드 없이 저렴한 계층에서 대용량 장기 보존 로그를 유지할 수 있습니다. 로그를 필터링하거나 별도의 스토리지 솔루션으로 오프로드하는 대신 CloudWatch Logs에서 기본적으로 로그를 유지하고 데이터가 상주하는 계층에 관계없이 동일한 쿼리 환경의 이점을 누릴 수 있습니다.

CloudWatch Logs는 액세스 패턴을 모니터링하고 30일 동안 Infrequent Access 계층에 액세스하지 않은 데이터와 90일 동안 Archive Instant Access 계층에 액세스하지 않은 데이터를 자동으로 재분류합니다. 이전 데이터에 액세스하면 CloudWatch Logs는 30일 동안 자동으로 표준 계층으로 다시 승격합니다. CloudWatch Logs의 모든 로그를 통합하면 단일 도구에서 완전한 가시성을 얻을 수 있습니다. 즉, 여러 스토리지 솔루션을 관리하는 데 드는 운영 오버헤드를 없애고 모든 로그를 한 곳에서 쿼리, 분석 및 알림하여 평균 해결 시간(MTTR)을 줄일 수 있습니다.

스토리지 비용 최적화를 시작하려면 CloudWatch 콘솔을 사용하거나를 호출하여 리전 내 계정 수준에서 Intelligent Tiering을 활성화합니다`PutStorageTierPolicy`. Intelligent Tiering을 활성화하면 해당 리전 내 계정의 모든 로그 그룹에 적용됩니다.

Amazon CloudWatch Logs Intelligent Tiering은 중동(바레인) 및 중동(UAE) 리전을 제외한 모든 AWS 상용 리전에서 사용할 수 있습니다.

CloudWatch Logs Intelligent Tiering 액세스 계층

쿼리 환경은 로그 데이터가 상주하는 액세스 계층에 관계없이 동일하게 유지됩니다.

표준(기본값)

이 값이 기본 계층입니다. 새로 수집된 모든 로그 데이터는 표준 계층에서 시작됩니다. 로그 데이터는 적극적으로 액세스하는 동안(마지막 액세스 후 0~30일) 표준 계층에 남아 있습니다. 표준 티어는 쿼리, 내보내기, 라이브 테일 및 로그 검색에 짧은 지연 시간과 높은 처리량 성능을 제공합니다.

Infrequent Access

연속 30일 동안 로그 데이터에 액세스하지 않으면 Infrequent Access 계층으로 이동합니다. 이 계층에는 31~90일 전에 마지막으로 액세스한 데이터가 들어 있습니다. Infrequent Access 계층은 표준 계층과 동일한 쿼리 지연 시간과 처리량을 더 낮은 스토리지 비용으로 제공합니다.

아카이브 인스턴트 액세스

연속 90일 동안 로그 데이터에 액세스하지 않으면 Archive Instant Access 계층으로 이동합니다. 이 계층에는 90일 이상 전에 마지막으로 액세스한 데이터가 들어 있습니다. Archive Instant Access 계층은 최저 스토리지 비용으로 다른 계층과 동일한 밀리초 쿼리 지연 시간을 제공합니다.

액세스를 구성하는 작업

다음 액세스 패턴은 Infrequent Access 계층 또는 Archive Instant Access 계층에서 표준 계층으로 로그 데이터를 자동으로 승격합니다. 이러한 작업 중 하나라도 저렴한 계층에서 로그 이벤트를 읽으면 해당 로그 이벤트는 30일 동안 표준 계층으로 돌아갑니다. 타이머는 이후의 각 액세스에서 재설정됩니다.

> 로그 중앙 집중화를 위한 계층 분류
>
> 교차 계정 교차 리전 로그 중앙 집중화를 통해 대상 계정의 복제된 로그에 액세스하면 대상 계정의 로그 데이터만 승격됩니다.

- 쿼리 또는 경보 구성을 사용한 송신 - `StartQuery` 또는 `CreateScheduledQuery`를 사용하여 CloudWatch Logs Insights로 로그 데이터를 쿼리하거나를 사용하여 로그에 대한 경보를 구성합니다.
- 로그 이벤트 필터링 및 검색 - `FilterLogEvents` 또는 `GetLogEvents`를 사용하여 로그 이벤트 읽기
- 내보내기 -를 사용하여 Amazon S3로 로그 데이터 내보내기 `CreateExportTask`

CloudWatch Logs Intelligent Tiering 활성화

필수 권한

계정의 스토리지 계층을 관리하려면 다음 IAM 권한이 필요합니다.

- `logs:PutStorageTierPolicy` - 계정의 스토리지 계층을 설정하거나 변경하는 데 필요합니다.
- `logs:GetStorageTierPolicy` - 현재 구성된 스토리지 계층을 검색하는 데 필요합니다. 콘솔에서 Intelligent Tiering을 활성화하는 데도 필요합니다.

`CloudWatchLogsFullAccess` 관리형 정책이 있는 사용자는 이미 두 권한을 모두 가지고 있습니다. `CloudWatchLogsReadOnlyAccess` 관리형 정책이 있는 사용자에게는 `logs:GetStorageTierPolicy` 권한이 있습니다.

다음 예제에서는 두 권한을 모두 부여하는 정책 설명을 보여줍니다.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:PutStorageTierPolicy",
        "logs:GetStorageTierPolicy"
    ],
    "Resource": "*"
}
```

콘솔을 사용하여 활성화

CloudWatch 콘솔을 사용하여 Intelligent Tiering을 활성화하려면:

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. 설정 페이지에서 로그 탭을 선택합니다.
4. Intelligent Tiering 섹션에서 활성화를 선택합니다.

AWS CLI를 사용하여 활성화

Intelligent Tiering을 활성화하려면 명령을 사용합니다`put-storage-tier-policy`.

```
aws logs put-storage-tier-policy --storage-tier INTELLIGENT_TIERING
```

명령은 설정된 스토리지 계층과 마지막으로 업데이트된 시간을 반환합니다.

```
{
    "storageTier": "INTELLIGENT_TIERING",
    "lastUpdatedTime": 1751328000000
}
```

Intelligent Tiering을 비활성화하려면 스토리지 계층을 다시 로 설정합니다`STANDARD`.

```
aws logs put-storage-tier-policy --storage-tier STANDARD
```

스토리지 계층을 다시 로 설정하면 계정의 STANDARD모든 로그 그룹에 표준 스토리지 요금이 청구됩니다.

현재 구성 보기

계정의 현재 스토리지 계층을 보려면 `get-storage-tier-policy` 명령을 사용합니다.

```
aws logs get-storage-tier-policy
```

명령은 현재 스토리지 계층과 마지막으로 업데이트된 시간을 반환합니다.

```
{
    "storageTier": "INTELLIGENT_TIERING",
    "lastUpdatedTime": 1751328000000
}
```

계층당 스토리지 모니터링

Intelligent Tiering이 활성화되면 CloudWatch Logs는 각 로그 그룹의 각 계층에 저장된 데이터의 양을 보여주는 판매 지표를 게시합니다.

- 네임스페이스: `AWS/Logs`
- 지표 이름: `StoredBytes`
- 차원: `LogGroupName`, `StorageType` (값: `Standard`, `Infrequent_Access`, `Archival_Instant_Access`)

이 지표를 사용하여 계층별 스토리지 분석을 표시하거나, 계층 크기에 대한 경보를 설정하거나, 시간 경과에 따른 비용 최적화를 추적하는 CloudWatch 대시보드를 생성할 수 있습니다.

> **Note**
>
> 이 지표는 하루에 한 번 게시되며 티어 전환을 실시간으로 반영하지 않을 수 있습니다.

CloudWatch Logs Intelligent Tiering 요금

Intelligent Tiering을 사용하면 데이터가 있는 계층에 따라 다른 스토리지 요금을 지불합니다. Intelligent Tiering을 활성화하는 데는 추가 요금이 부과되지 않으며 계층 전환에 대한 요금도 부과되지 않습니다. 각 계층에서 사용된 스토리지에 대해서만 비용을 지불합니다.

요금에 대한 자세한 정보는 Amazon CloudWatch 비용을 참조하세요.

계층별 스토리지 비용을 보려면 Athena를 사용하고 `TimedStorage-ByteHrs` (Standard), `TimedStorage-IA-ByteHrs` (Infrequent Access) 및 `TimedStorage-AIA-ByteHrs` (Archive Instant Access) 사용 유형을 기준으로 필터링합니다. 청구서의 CloudWatch Logs 사용 유형에 대한 자세한 내용은 CloudWatch 비용 분석, 최적화 및 절감을 참조하세요.

CloudWatch Logs 시작하기

Amazon EC2 인스턴스 및 온프레미스 서버의 로그를 CloudWatch Logs로 수집하려면 통합 CloudWatch 에이전트를 사용합니다. 에이전트 하나로 로그와 고급 지표를 모두 수집할 수 있습니다. Windows Server를 실행하는 서버를 포함하여 운영 체제 전반에 걸쳐 지원을 제공합니다. 또한 이 에이전트는 우수한 성능을 제공합니다.

통합 CloudWatch 에이전트를 통해 CloudWatch 지표를 수집하는 경우 인게스트 표시 여부에 대해 추가 시스템 지표를 수집할 수도 있습니다. 또한 `StatsD` 또는 `collectd`를 사용하는 사용자 지정 지표 수집을 지원합니다.

자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 에이전트 설치를 참조하세요.

Linux를 실행하는 서버에서의 로그 수집만 지원하는 이전 CloudWatch Logs 에이전트는 사용 중단되어 더 이상 지원되지 않습니다. 이전 CloudWatch Logs 에이전트에서 통합 에이전트로 마이그레이션하는 방법에 대한 자세한 내용은 마법사로 CloudWatch 에이전트 구성 파일 생성을 참조하세요.

내용

- 사전 조건
- 통합 CloudWatch 에이전트를 사용하여 CloudWatch Logs 시작하기
- 이전 CloudWatch 에이전트를 사용하여 CloudWatch Logs 시작하기
- 빠른 시작: CloudWatch Logs를 시작하는 CloudFormation 데 사용

사전 조건

Amazon CloudWatch Logs를 사용하려면 AWS 계정이 필요합니다. AWS 계정을 사용하면 서비스(예: Amazon EC2)를 사용하여 웹 기반 인터페이스인 CloudWatch 콘솔에서 볼 수 있는 로그를 생성할 수 있습니다. 또한 AWS Command Line Interface ()를 설치하고 구성할 수 있습니다AWS CLI.

에 가입 AWS 계정

를 시작하려면이 AWS필요합니다 AWS 계정. 생성에 대한 자세한 AWS 계정내용은 AWS Account Management 참조 안내서의 시작하기 AWS 계정를 참조하세요.

Command Line Interface 설정

AWS CLI 를 사용하여 CloudWatch Logs 작업을 수행할 수 있습니다.

설치 및 구성 방법에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설정 단원을 AWS CLI참조하십시오.

통합 CloudWatch 에이전트를 사용하여 CloudWatch Logs 시작하기

통합 CloudWatch 에이전트를 사용하여 CloudWatch Logs 시작에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버로부터 지표 및 로그 수집을 참조하세요. 이 섹션에 나열된 단계를 완료하여 에이전트를 설치, 구성 및 시작합니다. 에이전트를 사용하여 CloudWatch 지표를 수집하지 않는 경우 지표를 참조하는 모든 섹션을 무시할 수 있습니다.

현재 기존 CloudWatch Logs 에이전트를 사용하고 있으며, 새 통합 에이전트를 사용하기 위해 마이그레이션하려는 경우 새 에이전트 패키지에 포함된 마법사를 사용하는 것이 좋습니다. 이 마법사는 현재 CloudWatch Logs 에이전트 구성 파일을 읽고 동일한 로그를 수집하도록 CloudWatch 에이전트를 설정할 수 있습니다. 마법사에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 마법사로 CloudWatch 에이전트 구성 파일 생성을 참조하세요.

이전 CloudWatch 에이전트를 사용하여 CloudWatch Logs 시작하기

> **Important**
>
> CloudWatch는 EC2 인스턴스 및 온프레미스 서버에서 로그와 지표를 모두 수집할 수 있는 통합 CloudWatch 에이전트가 새롭게 추가되었습니다. 이전 로그 전용 에이전트는 사용 중단되어 더 이상 지원되지 않습니다.
>
> 이전 로그 전용 에이전트에서 통합 에이전트로 마이그레이션하는 방법에 대한 자세한 내용은 마법사로 CloudWatch 에이전트 구성 파일 생성을 참조하세요.
>
> 이번 섹션의 나머지 부분에서는 여전히 이전 CloudWatch Logs 에이전트를 사용 중인 고객을 위해 이전 CloudWatch Logs 에이전트의 사용에 대해서 설명합니다.

CloudWatch Logs 에이전트를 사용하면 Linux 또는 Windows Server를 실행하는 Amazon EC2 인스턴스에서 나온 로그 데이터와 AWS CloudTrail에서 나온 로그 이벤트를 게시할 수 있습니다. 대신 CloudWatch 통합 에이전트를 사용하여 로그 데이터를 게시하는 것이 좋습니다. 새 에이전트에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버로부터 지표 및 로그 수집을 참조하세요.

내용

- CloudWatch Logs 에이전트 사전 조건

- 빠른 시작: 실행 중인 EC2 Linux 인스턴스에 CloudWatch Logs 에이전트를 설치 및 구성
- 빠른 시작: 시작 시 EC2 Linux 인스턴스에 CloudWatch Logs 에이전트 설치 및 구성
- 빠른 시작: CloudWatch Logs 에이전트를 사용하여 Windows Server 2016을 실행하는 Amazon EC2 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정
- 빠른 시작: Windows Server 2012 및 Windows Server 2008을 실행하는 Amazon EC2 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정
- CloudWatch Logs 에이전트 상태 보고
- CloudWatch Logs 에이전트 시작
- CloudWatch Logs 에이전트 중지
- CloudWatch Logs 에이전트 참조

CloudWatch Logs 에이전트 사전 조건

CloudWatch Logs 에이전트에는 Python 버전 2.7, 3.0 또는 3.3과 다음과 같은 버전의 Linux가 필요합니다.

- Amazon Linux 버전 2014.03.02 이상 Amazon Linux 2는 지원되지 않습니다.
- Ubuntu Server 버전 12.04, 14.04 또는 16.04
- CentOS 버전 6, 6.3, 6.4, 6.5 또는 7.0
- Red Hat Enterprise Linux(RHEL) 버전 6.5 또는 7.0
- Debian 8.0

빠른 시작: 실행 중인 EC2 Linux 인스턴스에 CloudWatch Logs 에이전트를 설치 및 구성

Important

이전 로그 에이전트는 사용이 중단됩니다. CloudWatch는 EC2 인스턴스 및 온프레미스 서버에서 로그와 지표를 모두 수집할 수 있는 통합 에이전트를 포함합니다. 자세한 내용은 CloudWatch Logs 시작하기 단원을 참조하십시오.

이전 CloudWatch Logs 에이전트에서 통합 에이전트로 마이그레이션하는 방법에 대한 자세한 내용은 마법사로 CloudWatch 에이전트 구성 파일 생성을 참조하세요.

이전 로그 에이전트는 Python 버전 2.6부터 3.5까지만 지원합니다. 또한 이전 CloudWatch Logs 에이전트는 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 지원하지 않습니다. 서버에

서 IMDSv2를 사용하는 경우 이전 CloudWatch Logs 에이전트 대신 최신 통합 에이전트를 사용해야 합니다.
이번 섹션의 나머지 부분에서는 여전히 이전 CloudWatch Logs 에이전트를 사용 중인 고객을 위해 이전 CloudWatch Logs 에이전트의 사용에 대해서 설명합니다.

Tip

CloudWatch는 EC2 인스턴스 및 온프레미스 서버에서 로그와 지표를 모두 수집할 수 있는 통합 에이전트가 새롭게 추가되었습니다. 이미 이전 CloudWatch Logs 에이전트의 사용을 중단하였다면 새롭게 통합된 CloudWatch 에이전트 사용을 권장합니다. 자세한 내용은 CloudWatch Logs 시작하기 단원을 참조하십시오.
또한 이전 에이전트는 Instance Metadata Service 버전 2(IMDSv2)를 지원하지 않습니다. 서버에서 IMDSv2를 사용하는 경우 이전 CloudWatch Logs 에이전트 대신 최신 통합 에이전트를 사용해야 합니다.
이번 섹션의 나머지 부분에서는 이전 CloudWatch Logs 에이전트의 사용에 대해서 설명하겠습니다.

실행 중인 EC2 Linux 인스턴스에서 이전 CloudWatch Logs 에이전트 구성

기존 EC2 인스턴스에서 CloudWatch Logs 에이전트 설치 프로그램을 사용하여 CloudWatch Logs 에이전트를 설치 및 구성할 수 있습니다. 설치가 완료되면 로그가 자동으로 인스턴스에서 에이전트 설치 도중 생성한 로그 스트림으로 흐릅니다. 에이전트는 인스턴트가 시작되었고 비활성화를 할 때까지 실행이 유지되는지 확인합니다.

에이전트를 사용하는 것 외에도, AWS CLI CloudWatch Logs SDK 또는 CloudWatch Logs API를 사용하여 로그 데이터를 게시할 수도 있습니다. AWS CLI 는 명령줄 또는 스크립트를 통해 데이터를 게시하는 데 가장 적합합니다. CloudWatch Logs SDK는 애플리케이션에서 직접 로그 데이터를 게시하거나 자체적으로 로그 게시 애플리케이션을 구축하기에 가장 적합합니다.

1단계: CloudWatch Logs에 대한 IAM 역할 또는 사용자 구성

CloudWatch Logs 에이전트는 IAM 역할 및 사용자를 지원합니다. 인스턴스가 이미 IAM 역할에 연결되어 있으면 아래의 IAM 정책이 포함되어 있는지 확인합니다. 인스턴스에 IAM 역할이 할당되어 있지 않으면 다음 단계에서 IAM 자격 증명을 사용하거나 해당 인스턴스에 IAM 역할을 할당할 수 있습니다. 자세한 내용은 IAM 역할을 인스턴스에 연결을 참조하세요.

CloudWatch Logs에 대한 IAM 역할 또는 사용자를 구성하려면

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 이름을 선택하여 역할을 선택합니다(이름 옆의 확인란은 선택하지 않음).
4. 정책 연결에서 정책 생성을 선택합니다.

   새 브라우저 탭 또는 창이 열립니다.
5. JSON 탭을 선택하고 다음 JSON 정책 문서를 입력합니다.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
    ],
      "Resource": [
        "*"
    ]
  }
 ]
}
```

6. 작업이 완료되면 정책 검토를 선택합니다. 정책 검증자가 모든 구문 오류를 보고합니다.
7. 정책 검토 페이지에서 생성하는 정책에 대한 이름과 설명(선택 사항)을 입력합니다. 정책 요약을 검토하여 정책이 부여한 권한을 확인합니다. 그런 다음 정책 생성을 선택하여 작업을 저장합니다.
8. 브라우저 탭 또는 창을 닫고, 역할의 권한 추가 페이지로 돌아옵니다. 새로 고침을 선택한 후 새 정책을 선택하여 역할에 연결합니다.
9. 정책 연결을 선택합니다.

2단계: 기존 Amazon EC2 인스턴스에 CloudWatch Logs 설치 및 구성

CloudWatch Logs 에이전트 설치 프로세스는 Amazon EC2 인스턴스가 Amazon Linux, Ubuntu, CentOS 또는 Red Hat 중 무엇을 실행하느냐에 따라 다릅니다. 인스턴스에서 Linux 버전에 적합한 단계를 사용합니다.

기존 Amazon Linux 인스턴스에서 CloudWatch Logs를 설치 및 구성하려면

2014년 9월에 출시된 Amazon Linux AMI 2014.09부터 CloudWatch Logs 에이전트를 awslogs 패키지에서 RPM 설치 프로그램으로 사용할 수 있습니다. 이전 버전의 Amazon Linux는 `sudo yum update -y` 명령을 통해 인스턴스를 업데이트하여 awslogs 패키지에 액세스할 수 있습니다. CloudWatch Logs 설치 관리자를 사용하는 대신 awslogs 패키지를 RPM으로 설치하면 인스턴스는 CloudWatch Logs 에이전트를 수동으로 다시 설치할 필요 AWS 없이에서 정기적인 패키지 업데이트 및 패치를 수신합니다.

> **Warning**
>
> 이전에 Python 스크립트를 사용해 에이전트를 설치한 경우에는 RPM 설치 방법을 이용해 CloudWatch Logs 에이전트를 업데이트하지 마세요. 이로 인해 구성 문제가 발생하여 CloudWatch Logs 에이전트가 CloudWatch에 로그를 전송하지 못할 수 있습니다.

1. Amazon Linux 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하세요.

   연결 문제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결 문제 해결을 참조하세요.
2. Amazon Linux 인스턴스를 업데이트하여 패키지 리포지토리에서 최신 변경 사항을 찾아냅니다.

   ```
   sudo yum update -y
   ```
3. `awslogs` 패키지를 설치합니다. 이는 Amazon Linux 인스턴스에서 awslogs를 설치하는 데 있어 권장되는 방법입니다.

   ```
   sudo yum install -y awslogs
   ```
4. `/etc/awslogs/awslogs.conf` 파일을 편집하여 추적할 로그를 구성합니다. 이 파일 편집에 대한 자세한 내용은 CloudWatch Logs 에이전트 참조을 참조하세요.
5. 기본적으로 `/etc/awslogs/awscli.conf`는 us-east-1 리전을 가리킵니다. 다른 리전으로 로그를 푸시하려면 `awscli.conf` 파일을 편집하고 해당 리전을 지정합니다.

6. awslogs 서비스를 시작합니다.

   ```
   sudo service awslogs start
   ```

   Amazon Linux 2를 실행 중인 경우 다음 명령과 함께 awslogs 서비스를 시작합니다.

   ```
   sudo systemctl start awslogsd
   ```

7. (선택 사항) 서비스 시작 시 기록된 오류에 대해서는 /var/log/awslogs.log 파일을 확인하세요.
8. (선택 사항) 다음 명령을 실행하여 시스템 부팅 때마다 awslogs 서비스를 시작합니다.

   ```
   sudo chkconfig awslogs on
   ```

   Amazon Linux 2를 실행 중인 경우 다음 명령으로 각 시스템 부팅에서 서비스를 시작합니다.

   ```
   sudo systemctl enable awslogsd.service
   ```

9. 에이전트가 몇 분 동안 실행된 이후에 CloudWatch 콘솔에 새로 생성된 로그 그룹 및 로그 스트림이 나타납니다.

   자세한 정보는 CloudWatch Logs로 전송된 로그 데이터 보기 섹션을 참조하세요.

기존의 Ubuntu Server, CentOS 또는 Red Hat 인스턴스에 CloudWatch Logs를 설치 및 구성하려면

Ubuntu Server, CentOS 또는 Red Hat을 실행하는 AMI를 사용하고 있는 경우에는 다음 절차를 이용해 인스턴스에 CloudWatch Logs 에이전트를 수동으로 설치합니다.

1. EC2 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하세요.

   연결 문제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결 문제 해결을 참조하세요.
2. 두 옵션 중 하나를 사용하여 CloudWatch Logs 에이전트 설치 프로그램을 실행합니다. 인터넷에서 직접 실행하거나 파일을 다운로드하여 독립적으로 실행할 수 있습니다.

> **Note**
>
> CentOS 6.x, Red Hat 6.x, 또는 Ubuntu 12.04를 사용 중인 경우 독립 실행형 설치 관리자의 다운로드 및 실행 단계를 사용합니다. 이들 시스템에서는 CloudWatch Logs 에이전트를 인터넷에서 직접 설치하는 것이 지원되지 않습니다.

> **Note**
>
> 아래 명령을 실행하기 전에 Ubuntu에서 `apt-get update`을 실행합니다.

인터넷에서 직접 실행하려면 다음 명령을 사용하고 프롬프트의 메시지를 따릅니다.

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-
setup.py -O
```

```
sudo python ./awslogs-agent-setup.py --region us-east-1
```

위의 명령이 작동하지 않으면 다음 작업을 시도합니다.

```
sudo python3 ./awslogs-agent-setup.py --region us-east-1
```

단독으로 다운로드 및 실행하려면 다음 명령을 사용하고 프롬프트의 메시지를 따릅니다.

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-
setup.py -O
```

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/
AgentDependencies.tar.gz -O
```

```
tar xvf AgentDependencies.tar.gz -C /tmp/
```

```
sudo python ./awslogs-agent-setup.py --region us-east-1 --dependency-path /tmp/
AgentDependencies
```

us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1 또는 sa-east-1 리전을 지정하여 CloudWatch Logs 에이전트를 설치할 수 있습니다.

> **Note**
>
> 최신 버전과 `awslogs-agent-setup`의 버전 이력에 대한 자세한 내용은 CHANGELOG.txt를 참조하세요.

CloudWatch Logs 에이전트 설치 프로그램은 설정 단계에서 특정 정보를 요구합니다. 시작에 앞서 모니터링할 로그 파일과 타임스탬프 형식을 알아야 합니다. 또한 다음 정보를 이미 확보하고 있어야 합니다.

항목	설명
AWS 액세스 키 ID	IAM 역할을 사용하고 있는 경우에는 Enter 키를 누릅니다. 그렇지 않으면 AWS 액세스 키 ID를 입력합니다.
AWS 보안 액세스 키	IAM 역할을 사용하고 있는 경우에는 Enter 키를 누릅니다. 그렇지 않으면 AWS 보안 액세스 키를 입력합니다.
기본 리전 이름	입력을 누릅니다. 기본값은 us-east-2입니다. us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1 또는 sa-east-1 리전으로 설정할 수 있습니다.
기본 출력 형식	공백으로 남겨 두고 입력을 누릅니다.
업로드할 로그 파일의 경로	전송할 로그 데이터가 포함된 파일의 위치입니다. 설치 프로그램이 경로를 제안해 줍니다.
대상 로그 그룹 이름	로그 그룹의 이름. 설치 프로그램이 로그 그룹 이름을 제안합니다.
대상 로그 스트림 이름	기본적으로 이 이름은 호스트 이름입니다. 설치 프로그램이 호스트 이름을 제안합니다.

항목	설명
타임스탬프 형식	지정된 로그 파일 내의 타임스탬프 형식을 지정합니다. 사용자 지정을 선택해서 자체 형식을 지정합니다.
초기 위치	데이터가 업로드된 방법. 이를 start_of_file로 설정하면 데이터 파일에 모든 내용물이 업로드됩니다. end_of_file로 설정하면 새로 추가된 데이터만 업로드됩니다.

이러한 단계가 완료되면 설치 프로그램이 또 다른 로그 파일 구성에 대해 묻습니다. 각 로그 파일에서 원하는 수 만큼 프로세스를 실행할 수 있습니다. 더 이상 모니터링할 로그 파일이 없고 설치 프로그램에 또 다른 로그를 설정하라는 프롬프트 메시지가 나타나면 N을 선택합니다. 에이전트 구성 파일에서의 설정에 대한 자세한 내용은 CloudWatch Logs 에이전트 참조을 참조하세요.

> **Note**
>
> 단일 로그 스트림으로 데이터를 전송하기 위해 여러 개의 로그 소스를 구성하는 것이 불가능합니다.

3. 에이전트가 몇 분 동안 실행된 이후에 CloudWatch 콘솔에 새로 생성된 로그 그룹 및 로그 스트림이 나타납니다.

   자세한 정보는 CloudWatch Logs로 전송된 로그 데이터 보기 섹션을 참조하세요.

빠른 시작: 시작 시 EC2 Linux 인스턴스에 CloudWatch Logs 에이전트 설치 및 구성

> **Tip**
>
> 이 섹션에서 설명하는 이전 CloudWatch Logs 에이전트는 사용 중단되는 과정에 있습니다. 로그와 지표 둘 다를 수집할 수 있는 새로운 통합 CloudWatch 에이전트를 대신 사용하는 것이 좋습니다. 또한 이전 CloudWatch Logs 에이전트에는 Python 3.3 또는 이전 버전이 필요하며 이러한 버전은 기본적으로 새 EC2 인스턴스에 설치되지 않습니다. 통합 CloudWatch 에이전트에 대한 자세한 내용은 CloudWatch 에이전트 설치를 참조하세요.
> 이번 섹션의 나머지 부분에서는 이전 CloudWatch Logs 에이전트의 사용에 대해서 설명하겠습니다.

EC2 Linux 인스턴스를 시작할 때 이전 CloudWatch Logs 에이전트 설치

시작 시 파라미터 정보를 인스턴스에 전달할 수 있게 해주는 Amazon EC2 기능인 Amazon EC2 사용자 데이터를 사용하여 해당 인스턴스에 CloudWatch Logs 에이전트를 설치 및 구성합니다. CloudWatch Logs 에이전트 설치 및 구성 정보를 Amazon EC2로 전달하려면 Amazon S3 버킷 같은 네트워크 위치에 구성 파일을 제공할 수 있습니다.

단일 로그 스트림으로 데이터를 전송하기 위해 여러 개의 로그 소스를 구성하는 것이 불가능합니다.

사전 조건

모든 로그 그룹 및 로그 스트림을 설명하는 에이전트 구성 파일을 생성합니다. 모니터링할 로그 파일을 비롯해 로그 파일을 업로드할 로그 그룹 및 로그 스트림을 설명하는 텍스트 파일입니다. 에이전트는 이 구성 파일을 사용하여 여기에 설명된 모든 로그 파일들에 대한 모니터링 및 업로드를 시작합니다. 에이전트 구성 파일에서의 설정에 대한 자세한 내용은 CloudWatch Logs 에이전트 참조을 참조하세요.

다음은 Amazon Linux 2를 위한 에이전트 구성 파일 예제입니다.

```
[general]
state_file = /var/lib/awslogs/state/agent-state

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

다음은 Ubuntu을 위한 에이전트 구성 파일 예제입니다.

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/syslog]
file = /var/log/syslog
log_group_name = /var/log/syslog
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

IAM 역할을 구성하려면

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.

2. 탐색 창에서 정책을 선택한 후 정책 생성을 선택합니다.
3. 정책 생성 페이지에서 자체 정책 생성에 대해 선택을 선택합니다. 사용자 지정 정책 생성에 대한 자세한 내용은 Amazon EC2 User Guide의 IAM Policies for Amazon EC2을 참조하세요.
4. 정책 검토 페이지의 정책 이름에 정책 이름을 입력합니다.
5. 정책 문서에 다음 정책을 붙여넣습니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

6. 정책 생성을 선택합니다.
7. 탐색 창에서 역할, 새 역할 생성을 선택합니다.
8. 역할 이름 설정 페이지에서 역할 이름을 입력한 다음, 다음 단계를 선택합니다.
9. 역할 유형 선택 페이지에서 Amazon EC2 옆에 있는 선택을 선택합니다.

10. 정책 연결 페이지의 테이블 헤더에서 정책 유형과 고객 관리형을 선택합니다.
11. 생성한 IAM 정책을 선택하고 다음 단계를 선택합니다.
12. 역할 생성을 선택합니다.

    사용자 및 정책에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자 및 그룹과 IAM 정책 관리를 참조하세요.

새 인스턴스를 시작하고 CloudWatch Logs를 활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 시작을 선택합니다.

    자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 시작을 참조하세요.
3. 1단계: Amazon Machine Image(AMI) 선택 페이지에서 시작할 Linux 인스턴스 유형을 선택하고 2단계: 인스턴스 유형 선택 페이지에서 다음: 인스턴스 세부 정보 구성을 선택합니다.

    cloud-init가 Amazon Machine Image(AMI)에 포함되어 있는지 확인합니다. Amazon Linux AMIs와 Ubuntu 및 RHEL용 AMIs에는 이미 cloud-init가 포함되어 있지만의 CentOS 및 기타 AMIs에는 포함되어 있지 않을 AWS Marketplace 수 있습니다.
4. 3단계: 인스턴스 정보 구성 페이지에서 IAM 역할에 대해 생성한 IAM 역할을 선택합니다.
5. 고급 세부 정보의 사용자 데이터에 다음 스크립트를 붙여넣습니다. 그런 다음 -c 옵션의 값을 에이전트 구성 파일의 위치로 변경하여 해당 스크립트를 업데이트합니다.

    ```
    #!/bin/bash
    curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-
    setup.py -O
    chmod +x ./awslogs-agent-setup.py
    ./awslogs-agent-setup.py -n -r us-east-1 -c s3://amzn-s3-demo-bucket/my-config-file
    ```
6. 인스턴스를 더 변경하고 시작 설정을 검토한 다음 시작을 선택합니다.
7. 에이전트가 몇 분 동안 실행된 이후에 CloudWatch 콘솔에 새로 생성된 로그 그룹 및 로그 스트림이 나타납니다.

    자세한 정보는 CloudWatch Logs로 전송된 로그 데이터 보기 섹션을 참조하세요.

빠른 시작: CloudWatch Logs 에이전트를 사용하여 Windows Server 2016을 실행하는 Amazon EC2 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정

> **Tip**
>
> CloudWatch는 EC2 인스턴스 및 온프레미스 서버에서 로그와 지표를 모두 수집할 수 있는 통합 에이전트가 새롭게 추가되었습니다. 새롭게 통합된 CloudWatch 에이전트의 사용을 권장합니다. 자세한 정보는 CloudWatch Logs 시작하기 섹션을 참조하세요.
> 이번 섹션의 나머지 부분에서는 이전 CloudWatch Logs 에이전트의 사용에 대해서 설명하겠습니다.

이전 CloudWatch Logs 에이전트를 사용하여 Windows Server 2016을 실행하는 Amazon EC2 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정

Windows Server 2016을 실행하는 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정하는 방법에는 여러 가지가 있습니다. 이 섹션의 단계에서는 Systems Manager Run Command를 사용합니다. 다른 가능한 방법에 대한 자세한 내용은 Amazon CloudWatch로 로그, 이벤트, 성능 카운터 전송을 참조하세요.

단계(Steps)

- 샘플 구성 파일 다운로드
- CloudWatch에 JSON 파일 구성
- Systems Manager 대한 IAM 역할 생성
- Systems Manager 사전 조건 확인
- 인터넷 액세스 확인
- Systems Manager Run Command를 사용하여 CloudWatch Logs 활성화

샘플 구성 파일 다운로드

컴퓨터에 다음 샘플 파일을 다운로드합니다. `AWS.EC2.Windows.CloudWatch.json`

CloudWatch에 JSON 파일 구성

구성 파일에서 선택 항목을 지정하여 CloudWatch로 전송할 로그를 결정합니다. 이 파일을 만들고 선택 항목을 지정하는 프로세스를 완료하는 데 30분 이상 걸릴 수 있습니다. 이 작업을 한 번 완료한 후 모든 인스턴스에 구성 파일을 재사용할 수 있습니다.

단계(Steps)

- 1단계: CloudWatch Logs 활성화
- 2단계: CloudWatch의 설정 구성
- 3단계: 전송할 데이터 구성
- 4단계: 흐름 제어 구성
- 5단계: JSON 내용 저장

1단계: CloudWatch Logs 활성화

JSON 파일의 상단에서 `IsEnabled`을 'false'에서 'true'로 변경합니다.

```
"IsEnabled": true,
```

2단계: CloudWatch의 설정 구성

자격 증명, 리전, 로그 그룹 이름, 로그 스트림 네임스페이스를 지정합니다. 그러면 인스턴스가 CloudWatch Logs로 로그 데이터를 보낼 수 있습니다. 동일한 로그 데이터를 여러 위치로 보내려는 경우 ID가 고유하고(예: 'CloudWatchLogs2', 'CloudWatchLogs3') ID별로 리전이 다른 섹션을 추가할 수 있습니다.

CloudWatch Logs로 로그 데이터를 전송하기 위해 설정을 구성하려면

1. JSON 파일에서 `CloudWatchLogs` 섹션을 찾습니다.

```
{
    "Id": "CloudWatchLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "AccessKey": "",
        "SecretKey": "",
        "Region": "us-east-1",
        "LogGroup": "Default-Log-Group",
```

```
        "LogStream": "{instance_id}"
    }
},
```

2. AccessKey 및 SecretKey 필드는 비워둡니다. IAM 역할을 사용하여 자격 증명을 구성합니다.
3. Region에 로그 데이터를 보내려는 리전을 입력합니다(예: us-east-2).
4. LogGroup에 로그 그룹의 이름을 입력합니다. 이 이름은 CloudWatch 콘솔에서 로그 그룹 화면에 나타납니다.
5. LogStream에 대상 로그 스트림을 입력합니다. 이 이름은 CloudWatch 콘솔에서 로그 그룹 > 스트림 화면에 나타납니다.

   {instance_id}을 사용하는 경우 기본적으로 로그 스트림 이름은 이 인스턴스의 인스턴스 ID입니다.

   아직 존재하지 않는 로그 스트림 이름을 지정하면 CloudWatch Logs에서 이 이름을 자동으로 생성합니다. 리터럴 문자열, 미리 정의된 변수 {instance_id}, {hostname} 및 {ip_address} 또는 이들의 조합을 사용하여 로그 스트림 이름을 정의할 수 있습니다.

3단계: 전송할 데이터 구성

이벤트 로그 데이터, ETW(Windows용 이벤트 추적) 데이터 및 기타 로그 데이트를 CloudWatch Logs로 전송할 수 있습니다.

CloudWatch Logs로 Windows 애플리케이션 이벤트 로그 데이터를 보내려면

1. JSON 파일에서 ApplicationEventLog 섹션을 찾습니다.

```
{
    "Id": "ApplicationEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Application",
        "Levels": "1"
    }
},
```

2. Levels에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.

   - **1** - 오류 메시지만 업로드됩니다.

- **2** - 경고 메시지만 업로드됩니다.
- **4** - 정보 메시지만 업로드됩니다.

값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 보안 로그 데이터를 보내려면

1. JSON 파일에서 `SecurityEventLog` 섹션을 찾습니다.

```
{
    "Id": "SecurityEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Security",
        "Levels": "7"
    }
},
```

2. 모든 메시지를 업로드하려면 `Levels`에 **7**을 입력합니다.

CloudWatch Logs에 시스템 이벤트 로그 데이터를 보내려면

1. JSON 파일에서 `SystemEventLog` 섹션을 찾습니다.

```
{
    "Id": "SystemEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "System",
        "Levels": "7"
    }
},
```

2. `Levels`에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.

- **1** - 오류 메시지만 업로드됩니다.

- **2** - 경고 메시지만 업로드됩니다.
- **4** - 정보 메시지만 업로드됩니다.

값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 다른 유형의 이벤트 로그 데이터를 보내려면

1. JSON 파일에서 새 섹션을 추가합니다. 각 섹션에는 고유한 `Id`이 있어야 합니다.

```
{
    "Id": "Id-name",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Log-name",
        "Levels": "7"
    }
},
```

2. `Id`에 업로드할 로그의 이름을 입력합니다(예: **`WindowsBackup`**).
3. `LogName`에 업로드할 로그의 이름을 입력합니다. 로그 이름은 다음과 같이 확인할 수 있습니다.
   a. 이벤트 뷰어를 엽니다.
   b. 탐색 창에서 애플리케이션 및 서비스 로그를 선택합니다.
   c. 로그로 이동한 다음 작업, 속성을 선택합니다.
4. `Levels`에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.
   - **1** - 오류 메시지만 업로드됩니다.
   - **2** - 경고 메시지만 업로드됩니다.
   - **4** - 정보 메시지만 업로드됩니다.

   값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 Windows용 이벤트 추적 데이터를 보내려면

ETW(Windows용 이벤트 추적)는 애플리케이션이 로그를 기록할 수 있는 효율적이고 세부적인 로깅 메커니즘을 제공합니다. 로깅 세션을 시작하고 중지할 수 있는 세션 관리자가 각 ETW를 제어합니다. 각 세션에는 한 공급자와 하나 또는 그 이상의 소비자가 있습니다.

1. JSON 파일에서 ETW 섹션을 찾습니다.

```
{
    "Id": "ETW",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Microsoft-Windows-WinINet/Analytic",
        "Levels": "7"
    }
},
```

2. LogName에 업로드할 로그의 이름을 입력합니다.
3. Levels에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.

   - **1** - 오류 메시지만 업로드됩니다.
   - **2** - 경고 메시지만 업로드됩니다.
   - **4** - 정보 메시지만 업로드됩니다.

   값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 사용자 지정 로그(텍스트 기반 로그 파일)를 보내려면

1. JSON 파일에서 CustomLogs 섹션을 찾습니다.

```
{
    "Id": "CustomLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\CustomLogs\\",
        "TimestampFormat": "MM/dd/yyyy HH:mm:ss",
```

```
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "Local",
        "LineCount": "5"
    }
},
```

2. `LogDirectoryPath`에서 인스턴스에 로그가 저장된 경로를 입력합니다.
3. `TimestampFormat`에 사용할 타임스탬프 형식을 입력합니다. 지원되는 값에 대한 자세한 내용은 MSDN의 Custom Date and Time Format Strings 주제를 참조하세요.

> **Important**
>
> 원본 로그 파일에는 각 로그 줄의 시작 부분에 타임스탬프가 있어야 하며, 타임스탬프 뒤에는 공백이 있어야 합니다.

4. `Encoding`에 사용할 파일 인코딩을 입력합니다(예: UTF-8). 지원되는 값 목록은 MSDN에서 Encoding Class 항목을 참조하세요.

> **Note**
>
> 표시 이름이 아니라 인코딩 이름을 사용합니다.

5. (선택 사항) `Filter`에 로그 이름의 접두사를 입력합니다. 모든 파일을 모니터링하려면 이 파라미터를 공백으로 둡니다. 지원되는 값에 대한 자세한 내용은 MSDN의 FileSystemWatcherFilter 속성 주제를 참조하세요.
6. (선택 사항) `CultureName`에 타임스탬프가 로그되는 로케일을 입력합니다. `CultureName`이 공백이면 기본적으로 Windows 인스턴스에서 현재 사용 중인 것과 같은 로케일로 설정됩니다. 자세한 내용은 MSDN에서 제품 동작 주제의 표에 있는 `Language tag` 열을 참조하세요.

> **Note**
>
> `div`, `div-MV`, `hu` 및 `hu-HU` 값은 지원되지 않습니다.

7. (선택 사항) `TimeZoneKind`에 `Local` 또는 `UTC`를 입력합니다. 로그의 타임스탬프에 표준 시간대 정보가 포함되어 있지 않을 때 이렇게 설정하여 표준 시간대 정보를 제공할 수 있습니다. 이 파라미터가 공백으로 남겨져 있고 타임스탬프에 시간대 정보가 포함되어 있지 않으면 CloudWatch

Logs가 기본적으로 현지 표준 시간대로 설정됩니다. 타임스탬프에 표준 시간대 정보가 이미 포함된 경우 이 파라미터는 무시됩니다.

8. (선택 사항) `LineCount`에 로그 파일을 식별할 헤더의 줄의 수를 입력합니다. 예를 들어, IIS 로그 파일에 있는 헤더들은 사실상 동일합니다. **5**을 입력하면 로그 파일 헤더에서 처음 나오는 세 줄을 읽어 식별하는 식입니다. IIS 로그 파일에서 처음 나오는 세 줄은 날짜와 타임스탬프이지만 로그 파일 간에 타임스탬프가 반드시 다르지는 않습니다. 이러한 이유로, 로그 파일에 고유한 지문을 남기기 위해 실제 로그 데이터를 한 줄 이상 포함하는 것이 좋습니다.

CloudWatch Logs로 IIS 로그 데이터를 보내려면

1. JSON 파일에서 `IISLog` 섹션을 찾습니다.

```
{
    "Id": "IISLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
        "TimestampFormat": "yyyy-MM-dd HH:mm:ss",
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "UTC",
        "LineCount": "5"
    }
},
```

2. `LogDirectoryPath`에 개별 사이트에 대해 IIS 로그가 저장되는 폴더를 입력합니다(예: `C:\inetpub\logs\LogFiles\W3SVC`*n*).

> **Note**
>
> W3C 로그 형식만 지원됩니다. IIS, NCSA 및 사용자 지정 형식은 지원되지 않습니다.

3. `TimestampFormat`에 사용할 타임스탬프 형식을 입력합니다. 지원되는 값에 대한 자세한 내용은 MSDN의 사용자 지정 날짜 및 시간 형식 문자열 주제를 참조하세요.
4. `Encoding`에 사용할 파일 인코딩을 입력합니다(예: UTF-8). 지원되는 값에 대한 자세한 내용은 MSDN의 인코딩 클래스 주제를 참조하세요.

> **Note**
>
> 표시 이름이 아니라 인코딩 이름을 사용합니다.

5. (선택 사항) `Filter`에 로그 이름의 접두사를 입력합니다. 모든 파일을 모니터링하려면 이 파라미터를 공백으로 둡니다. 지원되는 값에 대한 자세한 내용은 MSDN의 FileSystemWatcherFilter 속성 주제를 참조하세요.
6. (선택 사항) `CultureName`에 타임스탬프가 로그되는 로케일을 입력합니다. `CultureName`이 공백이면 기본적으로 Windows 인스턴스에서 현재 사용 중인 것과 같은 로케일로 설정됩니다. 지원되는 값에 대한 자세한 내용은 MSDN에서 제품 동작 주제의 표에 있는 `Language tag` 열을 참조하세요.

> **Note**
>
> `div`, `div-MV`, `hu` 및 `hu-HU` 값은 지원되지 않습니다.

7. (선택 사항) `TimeZoneKind`에 `Local` 또는 `UTC`를 입력합니다. 로그의 타임스탬프에 표준 시간대 정보가 포함되어 있지 않을 때 이렇게 설정하여 표준 시간대 정보를 제공할 수 있습니다. 이 파라미터가 공백으로 남겨져 있고 타임스탬프에 시간대 정보가 포함되어 있지 않으면 CloudWatch Logs가 기본적으로 현지 표준 시간대로 설정됩니다. 타임스탬프에 표준 시간대 정보가 이미 포함된 경우 이 파라미터는 무시됩니다.
8. (선택 사항) `LineCount`에 로그 파일을 식별할 헤더의 줄의 수를 입력합니다. 예를 들어, IIS 로그 파일에 있는 헤더들은 사실상 동일합니다. **5**을 입력하면 로그 파일 헤더에서 처음 나오는 다섯 줄을 읽어 식별하는 식입니다. IIS 로그 파일에서 처음 나오는 세 줄은 날짜와 타임스탬프이지만 로그 파일 간에 타임스탬프가 반드시 다르지는 않습니다. 이러한 이유로, 로그 파일에 고유한 지문을 남기기 위해 실제 로그 데이터를 한 줄 이상 포함하는 것이 좋습니다.

4단계: 흐름 제어 구성

각 데이터 형식의 `Flows` 섹션에 해당 대상이 있어야 합니다. 예를 들어 사용자 지정 로그, ETW 로그 및 시스템 로그를 CloudWatch Logs로 전송하려면 `Flows` 섹션에 `(CustomLogs,ETW,SystemEventLog),CloudWatchLogs`를 추가합니다.

> **Warning**
>
> 유효하지 않은 단계를 추가하면 흐름이 차단됩니다. 예를 들어, 디스크 지표 단계를 추가하지만 인스턴스에 디스크가 없는 경우 흐름의 모든 단계가 차단됩니다.

같은 로그 파일을 두 개 이상의 대상으로 보낼 수 있습니다. 예를 들어, 애플리케이션 로그를 CloudWatchLogs 섹션에 정의된 두 개의 대상으로 전송하려면 Flows 섹션에 ApplicationEventLog,(CloudWatchLogs,CloudWatchLogs2)를 추가합니다.

흐름 제어를 구성하는 방법

1. AWS.EC2.Windows.CloudWatch.json 파일에서 Flows 섹션을 찾습니다.

```
"Flows": {
    "Flows": [
      "PerformanceCounter,CloudWatch",
      "(PerformanceCounter,PerformanceCounter2), CloudWatch2",
      "(CustomLogs, ETW, SystemEventLog),CloudWatchLogs",
      "CustomLogs, CloudWatchLogs2",
      "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)"
    ]
}
```

2. Flows에서 업로드할 각 데이터 형식(예: ApplicationEventLog) 및 대상(예: CloudWatchLogs)을 추가합니다.

5단계: JSON 내용 저장

이제 JSON 파일 편집이 끝났습니다. 이것을 저장하여 다른 창의 텍스트 편집기에 파일 내용을 붙여넣습니다. 이 절차의 이후 단계에서 이 파일 내용이 필요할 것입니다.

Systems Manager 대한 IAM 역할 생성

Systems Manager Run Command를 사용할 때 인스턴스 자격 증명에 대한 IAM 역할이 필요합니다. 이 역할은 Systems Manager가 인스턴스에 대한 작업을 수행하도록 허용합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 Systems Manager 대한 보안 역할 구성을 참조하세요. 기존 인스턴스에 IAM 역할을 연결하는 방법에 대한 내용은 Amazon EC2 사용 설명서의 IAM 역할을 인스턴스에 연결을 참조하세요.

Systems Manager 사전 조건 확인

Systems Manager Run Command를 사용하여 CloudWatch Logs와의 통합을 구성하기 전에 인스턴스가 최소 요구 사항을 충족하는지 확인해야 합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 Systems Manager 사전 조건을 참조하세요.

인터넷 액세스 확인

로그 및 이벤트 데이터를 CloudWatch로 보내려면 Amazon EC2 Windows Server 인스턴스와 관리형 인스턴스에 아웃바운드 인터넷 액세스 권한이 있어야 합니다. 인터넷 액세스를 구성하는 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 게이트웨이를 참조하세요.

Systems Manager Run Command를 사용하여 CloudWatch Logs 활성화

'명령 실행'을 통해 요청 시 인스턴스의 구성을 관리할 수 있습니다. Systems Manager 문서, 파라미터를 지정하고 하나 이상의 인스턴스에 명령을 실행합니다. 인스턴스의 SSM 에이전트는 명령을 처리하고 지정된 대로 인스턴스를 구성합니다.

Run Command를 사용하여 CloudWatch Logs와 통합을 구성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. https://console.aws.amazon.com/systems-manager/에서 SSM 콘솔을 엽니다.
3. 탐색 창에서 명령 실행을 선택합니다.
4. 명령 실행을 선택합니다.
5. 명령 문서에서 AWS-ConfigureCloudWatch를 선택합니다.
6. 대상 인스턴스에서 CloudWatch Logs와 통합할 인스턴스를 선택합니다. 이 목록에서 인스턴스가 보이지 않는 경우 Run Command에 대해 구성되지 않은 인스턴스일 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 Systems Manager 사전 조건을 참조하세요.
7. 상태에서 활성을 선택합니다.
8. 속성에 이전 작업에서 생성한 JSON 내용을 복사하여 붙여넣습니다.
9. 나머지 옵션 필드를 완성하고 Run을 선택합니다.

다음 절차를 이용해 Amazon EC2 콘솔에서 명령 실행 결과를 확인합니다.

콘솔에서 명령 출력을 보는 방법

1. 명령을 선택합니다.

2. 출력 탭을 선택합니다.
3. 출력 보기를 선택합니다. 명령 출력 페이지에 명령 실행 결과가 표시됩니다.

빠른 시작: Windows Server 2012 및 Windows Server 2008을 실행하는 Amazon EC2 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정

> **Tip**
>
> CloudWatch는 EC2 인스턴스 및 온프레미스 서버에서 로그와 지표를 모두 수집할 수 있는 통합 에이전트가 새롭게 추가되었습니다. 새롭게 통합된 CloudWatch 에이전트의 사용을 권장합니다. 자세한 정보는 CloudWatch Logs 시작하기 섹션을 참조하세요.
> 이번 섹션의 나머지 부분에서는 이전 CloudWatch Logs 에이전트의 사용에 대해서 설명하겠습니다.

Windows Server 2012 및 Windows Server 2008을 실행하는 Amazon EC2 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정

다음 단계를 사용하면 Windows Server 2012 및 Windows Server 2008을 실행하는 인스턴스가 로그를 CloudWatch Logs로 보내도록 설정할 수 있습니다.

샘플 구성 파일 다운로드

컴퓨터에 다음 샘플 JSON 파일을 다운로드합니다. `AWS.EC2.Windows.CloudWatch.json` 다음 단계에서 이 파일을 편집합니다.

CloudWatch에 JSON 파일 구성

JSON 구성 파일에서 선택 항목을 지정하여 CloudWatch로 전송할 로그를 결정합니다. 이 파일을 만들고 선택 항목을 지정하는 프로세스를 완료하는 데 30분 이상 걸릴 수 있습니다. 이 작업을 한 번 완료한 후 모든 인스턴스에 구성 파일을 재사용할 수 있습니다.

단계(Steps)

- 1단계: CloudWatch Logs 활성화
- 2단계: CloudWatch의 설정 구성
- 3단계: 전송할 데이터 구성
- 4단계: 흐름 제어 구성

1단계: CloudWatch Logs 활성화

JSON 파일의 상단에서 `IsEnabled`을 'false'에서 'true'로 변경합니다.

```
"IsEnabled": true,
```

2단계: CloudWatch의 설정 구성

자격 증명, 리전, 로그 그룹 이름, 로그 스트림 네임스페이스를 지정합니다. 그러면 인스턴스가 CloudWatch Logs로 로그 데이터를 보낼 수 있습니다. 동일한 로그 데이터를 여러 위치로 보내려는 경우 ID가 고유하고(예: 'CloudWatchLogs2', 'CloudWatchLogs3') ID별로 리전이 다른 섹션을 추가할 수 있습니다.

CloudWatch Logs로 로그 데이터를 전송하기 위해 설정을 구성하려면

1. JSON 파일에서 `CloudWatchLogs` 섹션을 찾습니다.

   ```
   {
       "Id": "CloudWatchLogs",
       "FullName":
    "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
       "Parameters": {
           "AccessKey": "",
           "SecretKey": "",
           "Region": "us-east-1",
           "LogGroup": "Default-Log-Group",
           "LogStream": "{instance_id}"
       }
   },
   ```

2. `AccessKey` 및 `SecretKey` 필드는 비워둡니다. IAM 역할을 사용하여 자격 증명을 구성합니다.
3. `Region`에 로그 데이터를 보내려는 리전을 입력합니다(예: `us-east-2`).
4. `LogGroup`에 로그 그룹의 이름을 입력합니다. 이 이름은 CloudWatch 콘솔에서 로그 그룹 화면에 나타납니다.
5. `LogStream`에 대상 로그 스트림을 입력합니다. 이 이름은 CloudWatch 콘솔에서 로그 그룹 > 스트림 화면에 나타납니다.

   `{instance_id}`을 사용하는 경우 기본적으로 로그 스트림 이름은 이 인스턴스의 인스턴스 ID입니다.

아직 존재하지 않는 로그 스트림 이름을 지정하면 CloudWatch Logs에서 이 이름을 자동으로 생성합니다. 리터럴 문자열, 미리 정의된 변수 `{instance_id}`, `{hostname}` 및 `{ip_address}` 또는 이들의 조합을 사용하여 로그 스트림 이름을 정의할 수 있습니다.

3단계: 전송할 데이터 구성

이벤트 로그 데이터, ETW(Windows용 이벤트 추적) 데이터 및 기타 로그 데이트를 CloudWatch Logs로 전송할 수 있습니다.

CloudWatch Logs로 Windows 애플리케이션 이벤트 로그 데이터를 보내려면

1. JSON 파일에서 `ApplicationEventLog` 섹션을 찾습니다.

```
{
    "Id": "ApplicationEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Application",
        "Levels": "1"
    }
},
```

2. `Levels`에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.

   - **1** - 오류 메시지만 업로드됩니다.
   - **2** - 경고 메시지만 업로드됩니다.
   - **4** - 정보 메시지만 업로드됩니다.

   값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 보안 로그 데이터를 보내려면

1. JSON 파일에서 `SecurityEventLog` 섹션을 찾습니다.

```
{
```

```
    "Id": "SecurityEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Security",
        "Levels": "7"
    }
},
```

2. 모든 메시지를 업로드하려면 `Levels`에 **7**을 입력합니다.

CloudWatch Logs에 시스템 이벤트 로그 데이터를 보내려면

1. JSON 파일에서 `SystemEventLog` 섹션을 찾습니다.

```
{
    "Id": "SystemEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "System",
        "Levels": "7"
    }
},
```

2. `Levels`에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.

   - **1** - 오류 메시지만 업로드됩니다.
   - **2** - 경고 메시지만 업로드됩니다.
   - **4** - 정보 메시지만 업로드됩니다.

   값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 다른 유형의 이벤트 로그 데이터를 보내려면

1. JSON 파일에서 새 섹션을 추가합니다. 각 섹션에는 고유한 `Id`이 있어야 합니다.

```
{
```

```
    "Id": "Id-name",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Log-name",
        "Levels": "7"
    }
},
```

2. `Id`에 업로드할 로그의 이름을 입력합니다(예: **`WindowsBackup`**).
3. `LogName`에 업로드할 로그의 이름을 입력합니다. 로그 이름은 다음과 같이 확인할 수 있습니다.

   a. 이벤트 뷰어를 엽니다.

   b. 탐색 창에서 애플리케이션 및 서비스 로그를 선택합니다.

   c. 로그로 이동한 다음 작업, 속성을 선택합니다.
4. `Levels`에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.

   - **1** - 오류 메시지만 업로드됩니다.
   - **2** - 경고 메시지만 업로드됩니다.
   - **4** - 정보 메시지만 업로드됩니다.

   값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 Windows용 이벤트 추적 데이터를 보내려면

ETW(Windows용 이벤트 추적)는 애플리케이션이 로그를 기록할 수 있는 효율적이고 세부적인 로깅 메커니즘을 제공합니다. 로깅 세션을 시작하고 중지할 수 있는 세션 관리자가 각 ETW를 제어합니다. 각 세션에는 한 공급자와 하나 또는 그 이상의 소비자가 있습니다.

1. JSON 파일에서 `ETW` 섹션을 찾습니다.

```
{
    "Id": "ETW",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Microsoft-Windows-WinINet/Analytic",
```

```
        "Levels": "7"
    }
},
```

2. LogName에 업로드할 로그의 이름을 입력합니다.
3. Levels에 업로드할 메시지의 유형을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.
   - **1** - 오류 메시지만 업로드됩니다.
   - **2** - 경고 메시지만 업로드됩니다.
   - **4** - 정보 메시지만 업로드됩니다.

   값을 적절히 조합하여 두 가지 이상의 메시지 유형을 포함할 수 있습니다. 예를 들어, **3**의 값은 오류 메시지(**1**) 및 경고 메시지(**2**)를 업로드합니다. 값 **7**을 지정하면 오류 메시지(**1**), 경고 메시지(**2**) 및 정보 메시지(**4**)가 업로드됩니다.

CloudWatch Logs로 사용자 지정 로그(텍스트 기반 로그 파일)를 보내려면

1. JSON 파일에서 CustomLogs 섹션을 찾습니다.

```
{
    "Id": "CustomLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\CustomLogs\\",
        "TimestampFormat": "MM/dd/yyyy HH:mm:ss",
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "Local",
        "LineCount": "5"
    }
},
```

2. LogDirectoryPath에서 인스턴스에 로그가 저장된 경로를 입력합니다.
3. TimestampFormat에 사용할 타임스탬프 형식을 입력합니다. 지원되는 값에 대한 자세한 내용은 MSDN의 Custom Date and Time Format Strings 주제를 참조하세요.

> **Important**
>
> 원본 로그 파일에는 각 로그 줄의 시작 부분에 타임스탬프가 있어야 하며, 타임스탬프 뒤에는 공백이 있어야 합니다.

4. `Encoding`에 사용할 파일 인코딩을 입력합니다(예: UTF-8). 지원되는 값에 대한 자세한 내용은 MSDN의 인코딩 클래스 주제를 참조하세요.

> **Note**
>
> 표시 이름이 아니라 인코딩 이름을 사용합니다.

5. (선택 사항) `Filter`에 로그 이름의 접두사를 입력합니다. 모든 파일을 모니터링하려면 이 파라미터를 공백으로 둡니다. 지원되는 값에 대한 자세한 내용은 MSDN의 FileSystemWatcherFilter 속성 주제를 참조하세요.
6. (선택 사항) `CultureName`에 타임스탬프가 로그되는 로케일을 입력합니다. `CultureName`이 공백이면 기본적으로 Windows 인스턴스에서 현재 사용 중인 것과 같은 로케일로 설정됩니다. 지원되는 값에 대한 자세한 내용은 MSDN에서 제품 동작 주제의 표에 있는 `Language tag` 열을 참조하세요.

> **Note**
>
> `div`, `div-MV`, `hu` 및 `hu-HU` 값은 지원되지 않습니다.

7. (선택 사항) `TimeZoneKind`에 `Local` 또는 `UTC`를 입력합니다. 로그의 타임스탬프에 표준 시간대 정보가 포함되어 있지 않을 때 이렇게 설정하여 표준 시간대 정보를 제공할 수 있습니다. 이 파라미터가 공백으로 남겨져 있고 타임스탬프에 시간대 정보가 포함되어 있지 않으면 CloudWatch Logs가 기본적으로 현지 표준 시간대로 설정됩니다. 타임스탬프에 표준 시간대 정보가 이미 포함된 경우 이 파라미터는 무시됩니다.
8. (선택 사항) `LineCount`에 로그 파일을 식별할 헤더의 줄의 수를 입력합니다. 예를 들어, IIS 로그 파일에 있는 헤더들은 사실상 동일합니다. **5**을 입력하면 로그 파일 헤더에서 처음 나오는 세 줄을 읽어 식별하는 식입니다. IIS 로그 파일에서 처음 나오는 세 줄은 날짜와 타임스탬프이지만 로그 파일 간에 타임스탬프가 반드시 다르지는 않습니다. 이러한 이유로, 로그 파일에 고유한 지문을 남기기 위해 실제 로그 데이터를 한 줄 이상 포함하는 것이 좋습니다.

CloudWatch Logs로 IIS 로그 데이터를 보내려면

1. JSON 파일에서 `IISLog` 섹션을 찾습니다.

```
{
    "Id": "IISLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
        "TimestampFormat": "yyyy-MM-dd HH:mm:ss",
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "UTC",
        "LineCount": "5"
    }
},
```

2. `LogDirectoryPath`에 개별 사이트에 대해 IIS 로그가 저장되는 폴더를 입력합니다(예: `C:\inetpub\logs\LogFiles\W3SVC`*n*).

> **Note**
>
> W3C 로그 형식만 지원됩니다. IIS, NCSA 및 사용자 지정 형식은 지원되지 않습니다.

3. `TimestampFormat`에 사용할 타임스탬프 형식을 입력합니다. 지원되는 값에 대한 자세한 내용은 MSDN의 사용자 지정 날짜 및 시간 형식 문자열 주제를 참조하세요.
4. `Encoding`에 사용할 파일 인코딩을 입력합니다(예: UTF-8). 지원되는 값에 대한 자세한 내용은 MSDN의 인코딩 클래스 주제를 참조하세요.

> **Note**
>
> 표시 이름이 아니라 인코딩 이름을 사용합니다.

5. (선택 사항) `Filter`에 로그 이름의 접두사를 입력합니다. 모든 파일을 모니터링하려면 이 파라미터를 공백으로 둡니다. 지원되는 값에 대한 자세한 내용은 MSDN의 FileSystemWatcherFilter 속성 주제를 참조하세요.
6. (선택 사항) `CultureName`에 타임스탬프가 로그되는 로케일을 입력합니다. `CultureName`이 공백이면 기본적으로 Windows 인스턴스에서 현재 사용 중인 것과 같은 로케일로 설정됩니다. 지원

되는 값에 대한 자세한 내용은 MSDN에서 제품 동작 주제의 표에 있는 Language tag 열을 참조하세요.

> **Note**
>
> div, div-MV, hu 및 hu-HU 값은 지원되지 않습니다.

7. (선택 사항) TimeZoneKind에 Local 또는 UTC를 입력합니다. 로그의 타임스탬프에 표준 시간대 정보가 포함되어 있지 않을 때 이렇게 설정하여 표준 시간대 정보를 제공할 수 있습니다. 이 파라미터가 공백으로 남겨져 있고 타임스탬프에 시간대 정보가 포함되어 있지 않으면 CloudWatch Logs가 기본적으로 현지 표준 시간대로 설정됩니다. 타임스탬프에 표준 시간대 정보가 이미 포함된 경우 이 파라미터는 무시됩니다.
8. (선택 사항) LineCount에 로그 파일을 식별할 헤더의 줄의 수를 입력합니다. 예를 들어, IIS 로그 파일에 있는 헤더들은 사실상 동일합니다. **5**을 입력하면 로그 파일 헤더에서 처음 나오는 다섯 줄을 읽어 식별하는 식입니다. IIS 로그 파일에서 처음 나오는 세 줄은 날짜와 타임스탬프이지만 로그 파일 간에 타임스탬프가 반드시 다르지는 않습니다. 이러한 이유로, 로그 파일에 고유한 지문을 남기기 위해 실제 로그 데이터를 한 줄 이상 포함하는 것이 좋습니다.

4단계: 흐름 제어 구성

각 데이터 형식의 Flows 섹션에 해당 대상이 있어야 합니다. 예를 들어 사용자 지정 로그, ETW 로그 및 시스템 로그를 CloudWatch Logs로 전송하려면 Flows 섹션에 `(CustomLogs,ETW,SystemEventLog),CloudWatchLogs`를 추가합니다.

> **Warning**
>
> 유효하지 않은 단계를 추가하면 흐름이 차단됩니다. 예를 들어, 디스크 지표 단계를 추가하지만 인스턴스에 디스크가 없는 경우 흐름의 모든 단계가 차단됩니다.

같은 로그 파일을 두 개 이상의 대상으로 보낼 수 있습니다. 예를 들어, 애플리케이션 로그를 CloudWatchLogs 섹션에 정의된 두 개의 대상으로 전송하려면 Flows 섹션에 `ApplicationEventLog,(CloudWatchLogs,CloudWatchLogs2)`를 추가합니다.

흐름 제어를 구성하는 방법

1. AWS.EC2.Windows.CloudWatch.json 파일에서 Flows 섹션을 찾습니다.

```
"Flows": {
    "Flows": [
      "PerformanceCounter,CloudWatch",
      "(PerformanceCounter,PerformanceCounter2), CloudWatch2",
      "(CustomLogs, ETW, SystemEventLog),CloudWatchLogs",
      "CustomLogs, CloudWatchLogs2",
      "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)"
    ]
}
```

2. Flows에서 업로드할 각 데이터 형식(예: ApplicationEventLog) 및 대상(예: CloudWatchLogs)을 추가합니다.

이제 JSON 파일 편집이 끝났습니다. 이후 단계에서 사용하게 됩니다.

에이전트 시작

Windows Server 2012 또는 Windows Server 2008을 실행하는 Amazon EC2 인스턴스가 CloudWatch Logs로 로그를 전송하도록 설정하려면 EC2Config 서비스(EC2Config.exe)를 사용합니다. 인스턴스에 EC2Config 4.0 이상이 있어야 다음 절차를 사용할 수 있습니다.

EC2Config 4.x를 사용하여 CloudWatch를 구성하려면

1. 이 절차에서 앞서 편집한 AWS.EC2.Windows.CloudWatch.json 파일의 인코딩을 확인합니다. BOM 없는 UTF-8 인코딩만 지원됩니다. Windows Server 2008 - 2012 R2 instance: C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch\에서 다음 폴더에 파일을 저장합니다.
2. Windows Services 제어판을 사용하거나 PowerShell에서 다음 명령을 사용하여 SSM 에이전트(AmazonSSMAgent.exe)를 시작하거나 다시 시작합니다.

```
PS C:\> Restart-Service AmazonSSMAgent
```

SSM 에이전트가 다시 시작한 후 구성 파일을 감지하고 CloudWatch 통합에 맞게 인스턴스를 구성합니다. 로컬 구성 파일에서 파라미터와 설정을 변경할 경우 SSM 에이전트를 다시 시작하여 변경 사항을 선택해야 합니다. 인스턴스에서 CloudWatch 통합을 비활성화하려면 IsEnabled을 false로 변경하고 구성 파일에 변경 사항을 저장합니다.

CloudWatch Logs 에이전트 상태 보고

다음 절차를 사용하여 EC2 인스턴스에 CloudWatch Logs 에이전트의 상태를 보고합니다.

에이전트 상태를 보고하려면

1. EC2 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하세요.

   연결 문제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결 문제 해결을 참조하세요.
2. 명령 프롬프트에서 다음 명령을 입력합니다.

   ```
   sudo service awslogs status
   ```

   Amazon Linux 2를 실행 중인 경우 다음 명령을 입력합니다.

   ```
   sudo service awslogsd status
   ```
3. /var/log/awslogs.log 파일에서 CloudWatch Logs 에이전트와 관련된 모든 오류, 경고 또는 문제를 확인할 수 있습니다.

CloudWatch Logs 에이전트 시작

설치 이후에 EC2 인스턴스의 CloudWatch Logs 에이전트가 자동으로 시작되지 않았거나 에이전트가 중단된 경우에는 아래 절차를 사용하여 에이전트를 시작할 수 있습니다.

에이전트를 시작하려면

1. EC2 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하세요.

   연결 문제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결 문제 해결을 참조하세요.
2. 명령 프롬프트에서 다음 명령을 입력합니다.

   ```
   sudo service awslogs start
   ```

   Amazon Linux 2를 실행 중인 경우 다음 명령을 입력합니다.

```
sudo service awslogsd start
```

CloudWatch Logs 에이전트 중지

다음 절차를 사용하여 EC2 인스턴스에서 CloudWatch Logs 에이전트를 중지합니다.

에이전트를 중지하려면

1. EC2 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하세요.

   연결 문제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결 문제 해결을 참조하세요.
2. 명령 프롬프트에서 다음 명령을 입력합니다.

   ```
   sudo service awslogs stop
   ```

   Amazon Linux 2를 실행 중인 경우 다음 명령을 입력합니다.

   ```
   sudo service awslogsd stop
   ```

CloudWatch Logs 에이전트 참조

> **Important**
>
> 이 섹션은 사용이 중단된 CloudWatch Logs 에이전트를 사용 중인 사용자를 위한 참조입니다. 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 사용 중인 경우 새 통합 CloudWatch 에이전트를 사용해야 합니다. 그러나 IMDSv2를 사용하고 있지 않더라도 사용이 중단된 CloudWatch Logs 에이전트 대신 새 통합 CloudWatch 에이전트를 사용하는 것이 좋습니다. 새 통합 CloudWatch 에이전트에 대한 자세한 내용은 CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버로부터 지표 및 로그 수집을 참조하세요. 사용이 중단된 CloudWatch Logs 에이전트에서 통합 에이전트로 마이그레이션하는 방법에 대한 자세한 내용은 마법사로 CloudWatch 에이전트 구성 파일 생성을 참조하세요.

CloudWatch Logs 에이전트는 Amazon EC2 인스턴스에서 CloudWatch Logs로 로그 데이터를 자동 전송할 수 있게 해줍니다. 에이전트에는 다음 구성 요소가 포함되어 있습니다.

- CloudWatch Logs에 로그 데이터를 푸시 AWS CLI 하는에 대한 플러그인입니다.
- CloudWatch Logs로 데이터를 푸시하는 프로세스를 시작하는 스크립트(데몬)입니다.
- 데몬이 항상 실행되도록 하는 cron 작업.

에이전트 구성 파일

CloudWatch Logs 에이전트 구성 파일에는 CloudWatch Logs 에이전트에 필요한 정보가 기술되어 있습니다. 에이전트 구성 파일의 [general] 섹션은 모든 로그 스트림에 적용되는 공통 구성을 정의합니다. [logstream] 섹션은 원격 로그 스트림에 로컬 파일을 전송하는 데 필요한 정보를 정의합니다. [logstream] 섹션은 하나 이상 정의가 가능하지만, 각 섹션은 구성 파일 내에 고유한 이름을 가져야 합니다(예: [logstream1], [logstream2] 등). 로그 파일에 있는 데이터의 첫 줄과 함께 [logstream] 값은 로그 파일의 ID를 정의합니다.

```
[general]
state_file = value
logging_config_file = value
use_gzip_http_content_encoding = [true | false]

[logstream1]
log_group_name = value
log_stream_name = value
datetime_format = value
time_zone = [LOCAL|UTC]
file = value
file_fingerprint_lines = integer | integer-integer
multi_line_start_pattern = regex | {datetime_format}
initial_position = [start_of_file | end_of_file]
encoding = [ascii|utf_8|..]
buffer_duration = integer
batch_count = integer
batch_size = integer

[logstream2]
...
```

state_file

상태 파일이 저장되는 장소를 지정합니다.

logging_config_file

(선택 사항) 에이전트 로깅 구성 파일의 위치를 지정합니다. 여기에서 에이전트 로깅 구성 파일을 지정하지 않으면 기본 파일인 awslogs.conf가 사용됩니다. 스크립트를 통해 에이전트를 설치한 경우 기본 파일 위치는 `/var/awslogs/etc/awslogs.conf`이고, rpm을 통해 에이전트를 설치한 경우 `/etc/awslogs/awslogs.conf`입니다. 파일에는 Python 구성 파일 형식이 있습니다 (https://docs.python.org/2/library/logging.config.html#logging-config-fileformat). 다음 이름을 가진 로거를 사용자 지정할 수 있습니다.

```
cwlogs.push
cwlogs.push.reader
cwlogs.push.publisher
cwlogs.push.event
cwlogs.push.batch
cwlogs.push.stream
cwlogs.push.watcher
```

아래 샘플은 기본 값이 INFO인 경우 독자와 게시자의 수준을 WARNING으로 수준을 변경합니다.

```
[loggers]
keys=root,cwlogs,reader,publisher

[handlers]
keys=consoleHandler

[formatters]
keys=simpleFormatter

[logger_root]
level=INFO
handlers=consoleHandler

[logger_cwlogs]
level=INFO
handlers=consoleHandler
qualname=cwlogs.push
propagate=0
```

```
[logger_reader]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.reader
propagate=0

[logger_publisher]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.publisher
propagate=0

[handler_consoleHandler]
class=logging.StreamHandler
level=INFO
formatter=simpleFormatter
args=(sys.stderr,)

[formatter_simpleFormatter]
format=%(asctime)s - %(name)s - %(levelname)s - %(process)d - %(threadName)s -
 %(message)s
```

use_gzip_http_content_encoding

true(기본)로 설정되어 있으면 gzip http 콘텐츠 인코딩을 활성화하여 CloudWatch Logs로 압축된 페이로드를 전송합니다. 이렇게 하면 CPU 사용량을 줄이고, NetworkOut을 낮추고, PUT 레코드 지연 시간을 줄일 수 있습니다. 이 기능을 비활성화하려면 CloudWatch Logs 에이전트 구성 파일의 [일반] 섹션에 use_gzip_http_content_encoding = false를 추가한 다음 에이전트를 다시 시작합니다.

> **Note**
>
> 이 설정은 awscli-cwlogs 버전 1.3.3 이상에서만 사용할 수 있습니다.

log_group_name

대상 로그 그룹을 지정합니다. 로그 그룹이 없는 경우 이 설정에 따라 로그 그룹이 자동으로 생성됩니다. 로그 그룹 이름에 포함되는 문자 길이는 1~512자입니다. 허용되는 문자: a-z, A-Z, 0-9, '_'(밑줄), '- (하이픈), '/'(슬래시) 및 '.'(마침표)

log_stream_name

대상 로그 스트림을 지정합니다. 리터럴 문자열이나 미리 정의된 변수({instance_id}, {hostname}, {ip_address}) 또는 이 둘의 조합을 사용하여 로그 스트림 이름을 정의할 수 있습니다. 로그 스트림이 없는 경우 이 설정에 따라 로그 스트림이 자동으로 생성됩니다.

datetime_format

타임스탬프가 로그에서 추출되는 방법을 지정합니다. 타임스탬프는 로그 이벤트를 검색하고 지표를 생성하는 데 사용됩니다. datetime_format이 제공되지 않는 경우에는 각 로그 이벤트에서 현재 시간이 사용됩니다. 제공된 datetime_format 값이 해당 로그 메시지에서 유효하지 않으면 타임스탬프가 성공적으로 구문 분석된 마지막 로그 이벤트에서 나온 타임스탬프가 사용됩니다. 이전 로그 이벤트가 존재하지 않는 경우 현재 시간이 사용됩니다.

공통적인 datetime_format 코드가 아래에 나열되어 있습니다. Python, datetime.strptime()에서 지원되는 datetime_format 코드면 무엇이든 사용 가능합니다. 시간대 오프셋(%z)도 지원되기는 하지만, 콜론(:)이 없는 python 3.2, [+-]HHMM이 나올 때까지는 지원되지 않습니다. 자세한 내용은 strftime() 및 strptime() 동작을 참조하세요.

%y: 제로 패딩된 10진수 형태의 세기를 제외한 연도. 00, 01, ..., 99

%Y: 10진수 형태의 세기를 포함한 연도. 1970, 1988, 2001, 2013

%b: 로케일 약어 형태의 월. Jan, Feb, ..., Dec(en_US)

%B: 로케일 전체 이름 형태의 월. January, February, ..., December(en_US)

%m: 제로 패딩된 10진수 형태의 월. 01, 02, ..., 12

%d: 제로 패딩된 10진수 형태의 월 날짜. 01, 02, ..., 31

%H: 제로 패딩된 10진수 형태의 시간(24시간 방식). 00, 01, ..., 23

%I: 제로 패딩된 10진수 형태의 시간(12시간 방식). 01, 02, ..., 12

%p: AM 또는 PM 중 하나에 상응하는 로케일.

%M: 제로 패딩된 10진수 형태의 분. 00, 01, ..., 59

%S: 제로 패딩된 10진수 형태의 초. 00, 01, ..., 59

%f: 왼쪽이 제로 패딩된 10진수 형태의 마이크로초. 000000, ..., 999999

%z: +HHMM 또는 -HHMM 형태의 UTC 오프셋. +0000, -0400, +1030

예제 형식:

```
Syslog: '%b %d %H:%M:%S', e.g. Jan 23 20:59:29

Log4j: '%d %b %Y %H:%M:%S', e.g. 24 Jan 2014 05:00:00

ISO8601: '%Y-%m-%dT%H:%M:%S%z', e.g. 2014-02-20T05:20:20+0000
```

time_zone

로그 이벤트 타임스탬프의 시간대를 지정합니다. UTC 및 LOCAL 등 2개의 값이 지원됩니다. 기본값인 LOCAL은 datetime_format에 따라 시간대를 추정할 수 없을 때 사용됩니다.

파일

CloudWatch Logs에 푸시하고 싶은 로그 파일을 지정합니다. 파일은 특정 파일을 가리키거나 여러 개의 파일을 가리킬 수 있습니다(/var/log/system.log* 같은 와일드카드를 사용). 파일 수정 시간에 따라 최신 파일만 CloudWatch Logs로 푸시됩니다. 와일드카드는 여러 종류의 파일(예: access_log_80 및 access_log_443)이 아니라 종류가 같은 일련의 파일(예: access_log.2014-06-01-01, access_log.2014-06-01-02 등)을 지정할 때 사용하는 것이 좋습니다. 여러 종류의 파일을 지정하려면 로그 파일의 종류에 따라 다른 로그 스트림에 들어가도록 구성 파일에 또 다른 로그 스트림 항목을 추가합니다. 압축 파일은 지원되지 않습니다.

file_fingerprint_lines

파일을 식별하기 위한 줄의 범위를 지정합니다. 유효한 값은 하나의 숫자(예: '1')나 대시로 구분된 두 개의 숫자(예: '2-5')입니다. 기본값은 '1'이기 때문에 지문 산출을 위해 첫 번째 줄이 사용됩니다. 지정된 모든 줄이 사용 가능한 상태가 되지 않는 한, 지문 줄은 CloudWatch Logs로 전송되지 않습니다.

multi_line_start_pattern

로그 메시지의 시작을 식별하기 위해 패턴을 지정합니다. 로그 메시지는 패턴과 일치하는 하나의 줄과 패턴과 일치하지 않는 나머지 줄들로 이루어져 있습니다. 유효한 값은 정규식 또는 {datetime_format}입니다. {datetime_format}을 사용할 때는 반드시 datetime_format 옵션이 지정되어 있어야 합니다. 기본값은 '^[^\s]'이기 때문에 공백이 아닌 문자로 시작되는 줄이 있으면 이전의 로그 메시지가 종료되고 새 로그 메시지가 시작됩니다.

initial_position

데이터 읽기를 시작할 지점(start_of_file 또는 end_of_file)을 지정합니다. 기본값은 파일의 시작 지점입니다. 해당 로그 스트림에서 지속되는 상태가 없을 때만 사용됩니다.

인코딩

파일을 정확하게 읽을 수 있도록 로그 파일의 인코딩을 설정합니다. 기본값은 utf_8입니다. Python codecs.decode()에서 지원되는 인코딩을 여기에서 사용할 수 있습니다.

> **Warning**
>
> 인코딩을 잘못 지정하면 디코딩할 수 없는 문자를 다른 문자들이 대체하면서 데이터 손실이 야기될 수 있습니다.

다음은 몇 가지 일반적인 인코딩입니다.

```
ascii, big5, big5hkscs, cp037, cp424, cp437, cp500, cp720, cp737,
cp775, cp850, cp852, cp855, cp856, cp857, cp858, cp860, cp861, cp862,
cp863, cp864, cp865, cp866, cp869, cp874, cp875, cp932, cp949, cp950,
cp1006, cp1026, cp1140, cp1250, cp1251, cp1252, cp1253, cp1254, cp1255,
cp1256, cp1257, cp1258, euc_jp, euc_jis_2004, euc_jisx0213, euc_kr,
gb2312, gbk, gb18030, hz, iso2022_jp, iso2022_jp_1, iso2022_jp_2,
iso2022_jp_2004, iso2022_jp_3, iso2022_jp_ext, iso2022_kr, latin_1,
iso8859_2, iso8859_3, iso8859_4, iso8859_5, iso8859_6, iso8859_7,
iso8859_8, iso8859_9, iso8859_10, iso8859_13, iso8859_14, iso8859_15,
iso8859_16, johab, koi8_r, koi8_u, mac_cyrillic, mac_greek, mac_iceland,
mac_latin2, mac_roman, mac_turkish, ptcp154, shift_jis, shift_jis_2004,
shift_jisx0213, utf_32, utf_32_be, utf_32_le, utf_16, utf_16_be,
utf_16_le, utf_7, utf_8, utf_8_sig
```

buffer_duration

로그 이벤트를 일괄 처리하는 기간을 지정합니다. 최솟값은 5,000ms이고, 기본값은 5,000ms입니다.

batch_count

일괄 처리할 최대 로그 이벤트의 수를 지정합니다(10,000까지 가능). 기본값은 10,000입니다.

batch_size

일괄 처리할 로그 이벤트의 최대 크기를 바이트로 지정합니다(1,048,576바이트까지 가능). 기본값은 1048576바이트입니다. 이 크기는 UTF-8에서 모든 이벤트 메시지를 합한 값에 각 로그 이벤트마다 26바이트를 추가하여 계산한 값입니다.

HTTP 프록시와 함께 CloudWatch Logs 에이전트 사용

CloudWatch Logs 에이전트를 HTTP 프록시와 함께 사용할 수 있습니다.

> **Note**
>
> HTTP 프록시는 awslogs-agent-setup.py 버전 1.3.8 이상에서 지원됩니다.

HTTP 프록시와 함께 CloudWatch Logs 에이전트를 사용하려면

1. 다음 중 하나를 수행합니다.

   a. CloudWatch Logs 에이전트를 새로 설치하려면 아래 명령을 실행합니다.

   ```
   curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
   ```

   ```
   sudo python awslogs-agent-setup.py --region us-east-1 --http-proxy http://your/proxy --https-proxy http://your/proxy --no-proxy 169.254.169.254
   ```

   EC2 인스턴스에서 Amazon EC2 메타데이터 서비스에 대한 액세스를 유지하려면 --no-proxy 169.254.169.254를 사용합니다(권장 사항). 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 및 사용자 데이터를 참조하세요.

   `http-proxy` 및 `https-proxy` 값에서 전체 URL을 지정합니다.

   b. 기존에 CloudWatch Logs 에이전트가 설치되어 있는 경우에는 /var/awslogs/etc/proxy.conf를 편집해서 프록시를 추가하세요.

   ```
   HTTP_PROXY=
   HTTPS_PROXY=
   NO_PROXY=
   ```

2. 에이전트를 다시 시작하면 변경 사항이 적용됩니다.

   ```
   sudo service awslogs restart
   ```

   Amazon Linux 2를 사용 중인 경우 다음 명령과 함께 에이전트를 다시 시작합니다.

```
sudo service awslogsd restart
```

CloudWatch Logs 에이전트 구성 파일 구획화

awslogs-agent-setup.py 버전 1.3.8 이상을 awscli-cwlogs 1.3.3 이상과 함께 사용하면 /var/awslogs/etc/config/ 디렉터리에서 추가 구성 파일을 생성하여 다양한 구성 요소들에 대해 서로 다른 스트림 구성을 독립적으로 가져올 수 있습니다. CloudWatch Logs 에이전트가 시작될 때 이러한 추가 구성 파일에 스트림 구성이 포함됩니다. [general] 섹션의 구성 속성들은 반드시 기본 구성 파일(/var/awslogs/etc/awslogs.conf)에 정의되어 있어야 하며, /var/awslogs/etc/config/에 있는 추가 구성 파일에서는 무시가 됩니다.

rpm을 통해 에이전트를 설치하지 않았기 때문에 /var/awslogs/etc/config/ 디렉터리가 없는 경우 /etc/awslogs/config/ 디렉터리를 대신 사용할 수 있습니다.

에이전트를 다시 시작하면 변경 사항이 적용됩니다.

```
sudo service awslogs restart
```

Amazon Linux 2를 사용 중인 경우 다음 명령과 함께 에이전트를 다시 시작합니다.

```
sudo service awslogsd restart
```

CloudWatch Logs 에이전트 FAQ

어떤 종류의 파일 로테이션이 지원됩니까?

다음과 같은 파일 로테이션 메커니즘이 지원되고 있습니다.

- 숫자 접미사를 붙여서 기존 로그 파일의 이름을 바꾸고 원래 빈 로그 파일을 다시 생성하는 방법입니다. 예를 들어, /var/log/syslog.log는 /var/log/syslog.log.1로 이름이 바뀝니다. 이전 로테이션에서 /var/log/syslog.log.1이 이미 존재하는 경우에는 /var/log/syslog.log.2로 이름이 바뀝니다.
- 복사본을 생성한 후에 원래 로그 파일을 잘라냅니다. 예를 들어, /var/log/syslog.log는 /var/log/syslog.log.1에 복사가 된 후 잘립니다. 이 경우 데이터 손실이 발생할 수 있기 때문에 이 파일 로테이션 메커니즘을 사용할 때는 조심해야 합니다.
- 기존 파일과 같은 공통 패턴을 따르는 파일을 새로 생성하는 방법입니다. 예를 들어, /var/log/syslog.log.2014-01-01이 그대로 남아 있는 상태에서 /var/log/syslog.log.2014-01-02가 생성됩니다.

파일의 지문(소스 ID)은 로그 스트림 키와 파일 콘텐츠의 첫 줄을 해싱하여 산출됩니다. 이 동작을 재정의하기 위해 file_fingerprint_lines 옵션을 사용할 수 있습니다. 파일 로테이션이 일어나면 새 파일은 새 콘텐츠를 가진 것으로 간주되지만, 기존 파일은 콘텐츠가 추가된 것으로 간주되지 않습니다. 따라서 에이전트는 기존 파일에 대한 읽기를 마치고 나면 새 파일을 푸시합니다.

사용 중인 에이전트의 버전을 어떻게 확인할 수 있습니까?

설정 스크립트를 사용해 CloudWatch Logs 에이전트를 설치했다면 /var/awslogs/bin/awslogs-version.sh를 사용해 사용 중인 에이전트의 버전을 확인할 수 있습니다. 에이전트의 버전과 중요한 플러그인들이 출력됩니다. yum을 사용해 CloudWatch Logs 에이전트를 설치했다면 'yum info awslogs' 및 'yum info aws-cli-plugin-cloudwatch-logs'를 사용해 CloudWatch Logs 에이전트의 버전과 플러그인을 확인할 수 있습니다.

로그 항목들은 어떻게 로그 이벤트로 변환됩니까?

로그 이벤트에는 이벤트가 발생한 시점에 대한 타임스탬프와 원시 로그 메시지 등 두 개의 속성이 포함되어 있습니다. 기본적으로 공백이 아닌 문자로 시작되는 줄이 있으면 이전의 로그 메시지가 종료되고 새 로그 메시지가 시작됩니다. 이 동작을 재정의하기 위해 multi_line_start_pattern을 사용할 수 있으면 패턴과 일치하는 모든 줄에서 새로운 로그 메시지가 시작됩니다. 어떤 regex 또는 {datetime_format}이든 패턴이 될 수 있습니다. 예를 들어, 모든 로그 메시지의 첫 줄에 '2014-01-02T13:13:01Z' 같은 타임스탬프가 포함되어 있으면 multi_line_start_pattern을 '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z'로 설정할 수 있습니다. 간편한 구성을 위해 datetime_format option이 지정된 경우에 '{datetime_format}' 변수를 사용할 수 있습니다. 같은 예제에서 datetime_format이 '%Y-%m-%dT%H:%M:%S%z'로 설정되어 있으면 multi_line_start_pattern이 간단히 '{datetime_format}'이 될 수 있습니다.

datetime_format이 제공되지 않는 경우에는 각 로그 이벤트에서 현재 시간이 사용됩니다. 제공된 datetime_format 값이 해당 로그 메시지에서 유효하지 않으면 타임스탬프가 성공적으로 구문 분석된 마지막 로그 이벤트에서 나온 타임스탬프가 사용됩니다. 이전 로그 이벤트가 존재하지 않는 경우 현재 시간이 사용됩니다. 로그 이벤트가 현재 시간이나 이전 로그 이벤트 시간으로 돌아가면 경고 메시지가 기록됩니다.

타임스탬프는 로그 이벤트를 검색하고 지표를 생성하는 데 사용되기 때문에 형식을 잘못 지정하면 로그 이벤트 검색이 불가능해지고 잘못된 지표가 생성될 수 있습니다.

로그 이벤트는 어떻게 일괄 처리됩니까?

아래 조건 중 어떤 것이든 충족이 되면 배치(batch)가 가득 차면서 게시가 됩니다.

1. 첫 번째 로그 이벤트가 추가되었기 때문에 buffer_duration 시간이 경과되었습니다.
2. batch_size 보다 작은 로그 이벤트들이 누적되었지만 새 로그 이벤트를 추가하면 batch_size를 초과하게 됩니다.

3. 로그 이벤트의 수가 batch_count에 도달했습니다.
4. 배치에서 나온 로그 이벤트들이 24시간 이상 지속되지 않지만 새 로그 이벤트를 추가하면 24시간이라는 제약 조건을 넘어서게 됩니다.

로그 항목, 로그 이벤트 또는 배치는 어떤 이유로 건너 뛰기가 되거나 잘라집니까?

PutLogEvents 작업의 제약 조건을 따르다 보면 다음과 같은 문제들로 인해 로그 이벤트나 배치를 건너 뛰는 상황이 발생할 수 있습니다.

> **Note**
>
> 데이터를 건너뛰면 CloudWatch Logs 에이전트가 로그에 경고를 기록합니다.

1. 로그 이벤트의 크기가 256KB를 초과하면 로그 이벤트가 완전히 건너 뛰기 됩니다.
2. 로그 이벤트의 타임스탬프가 미래에 2시간 이상이면 해당 로그 이벤트가 건너 뛰기 됩니다.
3. 로그 이벤트의 타임스탬프가 과거에 14일 이상이었으면 해당 로그 이벤트가 건너 뛰기 됩니다.
4. 로그 그룹의 보존 기간을 지난 로그 이벤트가 있으면 전체 배치가 건너 뛰기 됩니다.
5. 단일 PutLogEvents 요청에서 로그 이벤트에 대한 배치가 24시간 이상 지속된 경우에는 PutLogEvents 작업이 실패합니다.

에이전트 중지로 인해 데이터 손실/중복이 발생하고 있습니까?

상태 파일이 사용 가능하고 마지막 실행 이후로 파일 로테이션이 발생하지 않은 한 그렇지 않습니다. CloudWatch Logs 에이전트는 중지된 지점부터 시작해서 로그 데이터를 계속해서 푸시할 수 있습니다.

같은 호스트나 다른 호스트에서 나온 서로 다른 로그 파일이 동일한 로그 스트림으로 가리키도록 할 수 있습니까?

단일 로그 스트림으로 데이터를 전송하기 위해 여러 개의 로그 소스를 구성하는 것이 불가능합니다.

에이전트가 어떤 API를 직접 호출합니까?(또는 IAM 정책에 어떤 작업을 추가해야 합니까?)

CloudWatch Logs 에이전트에는 CreateLogGroup, CreateLogStream, DescribeLogStreams, DescribeLogGroups, PutLogEvents 및 PutRetentionPolicy 작업을 수행할 수 있는 권한이 필요합니다. 최신 에이전트를 사용하고 있는 경우에는 DescribeLogStreams가 필요하지 않습니다. IAM 정책 샘플은 아래를 참조하세요.

JSON

```
{
"Version":"2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents",
      "logs:DescribeLogStreams",
      "logs:DescribeLogGroups",
      "logs:PutRetentionPolicy"
    ],
    "Resource": [
      "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

CloudWatch Logs 에이전트에서 로그 그룹이나 로그 스트림이 자동 생성되도록 하고 싶지 않습니다. 어떻게 하면 에이전트가 로그 그룹 또는 로그 스트림을 다시 생성하지 않게 할 수 있습니까?

IAM 정책에서 `DescribeLogStreams`, `PutLogEvents`작업만 수행하도록 에이전트 작업을 제한할 수 있습니다.

에이전트에서 `CreateLogGroup` 및 `CreateLogStream` 권한을 취소하기 전에 에이전트에서 사용할 로그 그룹과 로그 스트림을 모두 생성해야 합니다. 로그 에이전트는 `CreateLogGroup` 및 `CreateLogStream` 권한이 모두 없는 한 사용자가 생성한 로그 그룹에 로그 스트림을 생성할 수 없습니다.

문제 해결 시 어떤 로그를 살펴봐야 합니까?

에이전트 설치 로그는 `/var/log/awslogs-agent-setup.log`에, 에이전트 로그는 `/var/log/awslogs.log`에 있습니다.

빠른 시작: CloudWatch Logs를 시작하는 CloudFormation 데 사용

AWS CloudFormation 를 사용하면 AWS 리소스를 JSON 형식으로 설명하고 프로비저닝할 수 있습니다. 이 방법의 이점으로는 AWS 리소스 모음을 단일 단위로 관리할 수 있고 리전 간에 AWS 리소스를 쉽게 복제할 수 있는 기능이 있습니다.

AWS 를 사용하여 프로비저닝 CloudFormation할 때 사용할 리소스를 설명하는 AWS 템플릿을 생성합니다. 다음은 로그 그룹과 404 출현 횟수를 계산해서 로그 그룹에 전송하는 지표 필터를 생성하는 템플릿 코드 조각의 예제입니다.

```
"WebServerLogGroup": {
    "Type": "AWS::Logs::LogGroup",
    "Properties": {
        "RetentionInDays": 7
    }
},

"404MetricFilter": {
    "Type": "AWS::Logs::MetricFilter",
    "Properties": {
        "LogGroupName": {
            "Ref": "WebServerLogGroup"
        },
        "FilterPattern": "[ip, identity, user_id, timestamp, request, status_code =
 404, size, ...]",
        "MetricTransformations": [
            {
                "MetricValue": "1",
                "MetricNamespace": "test/404s",
                "MetricName": "test404Count"
            }
        ]
    }
}
```

이는 기본적인 예제입니다. 를 사용하여 훨씬 더 풍부한 CloudWatch Logs 배포를 설정할 수 있습니다 CloudFormation. 템플릿 예제에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 Amazon CloudWatch Logs 템플릿 조각을 참조하세요. 시작하기에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation시작하기를 참조하세요.

HTTP 엔드포인트를 통한 로그 수집

Amazon CloudWatch Logs는 간단한 HTTP POST 요청을 사용하여 CloudWatch Logs로 직접 로그를 전송할 수 있는 HTTP 엔드포인트를 제공합니다. 이러한 엔드포인트는 SigV4 및 보유자 토큰 인증을 모두 지원합니다.

> **Important**
>
> AWS SDK 통합이 가능한 모든 프로덕션 워크로드에 SigV4 인증을 사용하는 것이 좋습니다. SigV4는 단기 자격 증명을 사용하고 가장 강력한 보안 태세를 제공합니다. 베어러 토큰(API 키) 인증은 AWS SDK 통합을 지원하지 않는 타사 로그 전달자와 같이 SigV4가 불가능한 시나리오를 위한 것입니다. 자세한 내용은 IAM 사용 설명서의 장기 액세스 키의 대안을 참조하세요.

CloudWatch Logs는 다음 HTTP 수집 엔드포인트를 지원합니다.

엔드포인트	경로	Content-Type	형식
OpenTelemetry Logs	`/v1/logs`	`application/json` 또는 `application/x-protobuf`	OTLP JSON 또는 Protobuf
HLC Logs	`/services/collector/event`	`application/json`	HLC 형식
ND-JSON Logs	`/ingest/bulk`	`application/json` 또는 `application/x-ndjson`	줄 바꿈으로 구분된 JSON
Structured JSON Logs	`/ingest/json`	`application/json`	JSON 객체 또는 배열

일반적인 동작

모든 HTTP 수집 엔드포인트는 다음 동작을 공유합니다.

Authentication

모든 엔드포인트는 SigV4 및 보유자 토큰 인증을 모두 지원합니다.

- SigV4(권장) - 표준 AWS 서명 버전 4 서명. 애플리케이션 또는 인프라가 AWS SDK를 지원하거나 요청에 서명할 수 있을 때마다 SigV4를 사용합니다. SigV4는 단기 자격 증명을 사용하며 가장 안전한 인증 방법입니다.
- 베어러 토큰 - `Authorization: Bearer <ACWL token>` 헤더를 사용합니다.
  - 토큰은 유효한 ACWL 보유자 토큰이어야 합니다. 설정 지침은 단원을 참조하십시오the section called “베어러 토큰 인증”.
  - `logs:PutLogEvents` 및 `logs:CallWithBearerToken` IAM 권한이 필요합니다.

로그 그룹 및 로그 스트림

- 헤더를 통해 제공: `x-aws-log-group` 및 `x-aws-log-stream`
- 쿼리 파라미터`?logGroup=<name>&logStream=<name>`는 OTLP를 제외한 모든 엔드포인트에서도 지원됩니다.
- 동일한 파라미터에 대해 쿼리 파라미터와 헤더를 모두 사용할 수는 없습니다.
- 로그 그룹과 로그 스트림이 모두 필요합니다.

응답

- 성공: 본문 `HTTP 200` 포함 `{}`
- 검증 오류: `HTTP 400`
- 인증 실패: `HTTP 401`

보유자 토큰 인증 설정

HTTP 수집 엔드포인트에서 보유자 토큰 인증을 사용하여 로그를 전송하려면 먼저 다음을 수행해야 합니다.

- CloudWatch Logs 권한이 있는 IAM 사용자 생성
- 서비스별 자격 증명 생성(베어러 토큰)
- 로그 그룹 및 로그 스트림 생성
- 로그 그룹에서 보유자 토큰 인증 활성화

> **Important**
>
> 가능한 경우 모든 워크로드에 대해 단기 자격 증명과 함께 SigV4 인증을 사용하는 것이 좋습니다. SigV4는 가장 강력한 보안 태세를 제공합니다. API 키(베어러 토큰) 사용을 단기 자격 증명 기반 인증이 불가능한 시나리오로 제한합니다. 보안 요구 사항이 더 큰 애플리케이션에 CloudWatch Logs를 통합할 준비가 되면 단기 자격 증명으로 전환해야 합니다. 자세한 내용은 IAM 사용 설명서의 장기 액세스 키의 대안을 참조하세요.

옵션 1: AWS 콘솔을 사용하여 빠른 시작

AWS Management Console은 HTTP 엔드포인트 액세스를 위한 API 키를 생성하는 간소화된 워크플로를 제공합니다.

콘솔을 사용하여 HTTP 엔드포인트 액세스를 설정하려면

1. AWS Management Console에 로그인합니다.
2. CloudWatch > 설정 > 로그로 이동합니다.
3. API 키 섹션에서 API 키 생성을 선택합니다.
4. API 키 만료가 된 경우 다음 중 하나를 수행하세요..
   - 1, 5, 30, 90 또는 365일의 API 키 만료 기간을 선택합니다.
   - 사용자 지정 기간을 선택하여 사용자 지정 API 키 만료 날짜를 지정합니다.
   - 만료되지 않음을 선택합니다(권장되지 않음).
5. API 키 생성을 선택합니다.

   콘솔은 자동으로 다음을 수행합니다.

   - 적절한 권한을 가진 새 IAM 사용자를 생성합니다.
   - CloudWatchLogsAPIKeyAccess 관리형 정책(`logs:PutLogEvents` 및 `logs:CallWithBearerToken` 권한 포함)을 연결합니다.
   - 서비스별 자격 증명(API 키)을 생성합니다.
6. 표시된 자격 증명을 복사하고 안전하게 저장합니다.
   - API 키 ID(서비스별 자격 증명 ID)
   - API 키 보안 암호(베어러 토큰)

> **Important**
>
> API 키 보안 암호를 즉시 저장합니다. 나중에는 조회할 수 없습니다. 분실한 경우 새 API 키를 생성해야 합니다.

7. 로그가 저장될 로그 그룹 및 로그 스트림을 생성합니다.

```
# Create the log group
aws logs create-log-group \
    --log-group-name /aws/hlc-logs/my-application \
    --region us-east-1

# Create the log stream
aws logs create-log-stream \
    --log-group-name /aws/hlc-logs/my-application \
    --log-stream-name application-stream-001 \
    --region us-east-1
```

8. 로그 그룹에서 보유자 토큰 인증을 활성화합니다.

```
aws logs put-bearer-token-authentication \
    --log-group-identifier /aws/hlc-logs/my-application \
    --bearer-token-authentication-enabled \
    --region us-east-1
```

구성을 확인합니다.

```
aws logs describe-log-groups \
    --log-group-name-prefix /aws/hlc-logs/my-application \
    --region us-east-1
```

포함된 권한: 자동으로 생성된 IAM 사용자에게는 다음과 같은 권한이 있습니다.

- `logs:PutLogEvents` - CloudWatch Logs로 로그 이벤트 전송
- `logs:CallWithBearerToken` - 보유자 토큰을 사용하여 인증
- `kms:Describe*`, `kms:GenerateDataKey*`, `kms:Decrypt` - KMS로 암호화된 로그 그룹에 액세스(로그 서비스로 제한되는 조건 사용)

옵션 2: 수동 설정

IAM 구성에 대한 더 많은 제어를 원하거나 권한을 사용자 지정해야 하는 경우 HTTP 엔드포인트 액세스를 수동으로 설정할 수 있습니다.

1단계: IAM 사용자 생성

로그 수집에 사용할 IAM 사용자를 생성합니다.

1. AWS Management Console에 로그인하고 IAM으로 이동합니다.
2. 왼쪽 탐색 창에서 사용자를 선택합니다.
3. 사용자 생성을 선택합니다.
4. 사용자 이름(예: `cloudwatch-logs-hlc-user`)을 입력합니다.
5. 다음을 선택합니다.
6. 다음 IAM 정책 중 하나를 연결합니다.

   옵션 A: 관리형 정책 사용(권장)

   CloudWatchLogsAPIKeyAccess 관리형 정책을 연결합니다.

   옵션 B: 사용자 지정 정책 생성

   다음 IAM 정책을 생성하고 연결합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LogsAPIs",
            "Effect": "Allow",
            "Action": [
                "logs:CallWithBearerToken",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "KMSAPIs",
            "Effect": "Allow",
            "Action": [
```

```
                "kms:Describe*",
                "kms:GenerateDataKey*",
                "kms:Decrypt"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "logs.*.amazonaws.com"
                    ]
                }
            },
            "Resource": "arn:aws:kms:*:*:key/*"
        }
    ]
}
```

7. 다음을 선택한 후 사용자 생성을 선택합니다.

> **Note**
>
> KMS 암호화 로그 그룹에 로그를 보내려는 경우 KMS 권한이 필요합니다. 이 조건은 KMS 액세스를 CloudWatch Logs 서비스를 통해 사용되는 키로만 제한합니다.

2단계: 서비스별 자격 증명 생성(API 키)

CreateServiceSpecificCredential API를 사용하여 CloudWatch Logs API 키를 생성합니다. create-service-specific-credential CLI 명령을 사용할 수도 있습니다. 자격 증명 기간에 대해 1~36600일 사이의 값을 지정할 수 있습니다. 자격 증명 기간을 지정하지 않으면 API 키가 만료되지 않습니다.

만료 기간이 30일인 API 키를 생성하려면:

```
aws iam create-service-specific-credential \
    --user-name cloudwatch-logs-hlc-user \
    --service-name logs.amazonaws.com \
    --credential-age-days 30
```

응답은 ServiceSpecificCredential 객체입니다. `ServiceCredentialSecret` 값은 CloudWatch Logs API 키(베어러 토큰)입니다.

> **Important**
>
> 나중에 검색할 수 없으므로 `ServiceCredentialSecret` 값을 안전하게 저장하세요. 분실한 경우 새 API 키를 생성해야 합니다.

3단계: 로그 그룹 및 로그 스트림 생성

로그가 저장될 로그 그룹 및 로그 스트림을 생성합니다.

```
# Create the log group
aws logs create-log-group \
    --log-group-name /aws/hlc-logs/my-application \
    --region us-east-1

# Create the log stream
aws logs create-log-stream \
    --log-group-name /aws/hlc-logs/my-application \
    --log-stream-name application-stream-001 \
    --region us-east-1
```

4단계: 보유자 토큰 인증 활성화

로그 그룹에서 보유자 토큰 인증을 활성화합니다.

```
aws logs put-bearer-token-authentication \
    --log-group-identifier /aws/hlc-logs/my-application \
    --bearer-token-authentication-enabled \
    --region us-east-1
```

구성을 확인합니다.

```
aws logs describe-log-groups \
    --log-group-name-prefix /aws/hlc-logs/my-application \
    --region us-east-1
```

CloudWatch Logs API 키 생성 및 사용에 대한 권한 제어

CloudWatch Logs API 키의 생성 및 사용은 CloudWatch Logs 및 IAM 서비스 모두에서 작업 및 조건 키로 제어됩니다.

CloudWatch Logs API 키 생성 제어

iam:CreateServiceSpecificCredential 작업은 서비스별 키(예: CloudWatch Logs API 키)의 생성을 제어합니다. 이 작업의 범위를 리소스인 IAM 사용자로 지정하여 키를 생성할 수 있는 사용자를 제한할 수 있습니다.

다음 조건 키를 사용하여 `iam:CreateServiceSpecificCredential` 작업에 대한 권한에 조건을 적용할 수 있습니다.

- iam:ServiceSpecificCredentialAgeDays - 조건에서 키의 만료 시간을 일 단위로 지정할 수 있습니다. 예를 들어, 이 조건 키를 사용하여 90일 이내에 만료되는 API 키의 생성만 허용할 수 있습니다.
- iam:ServiceSpecificCredentialServiceName - 조건에서 서비스 이름을 지정할 수 있습니다. 예를 들어이 조건 키를 사용하여 다른 서비스가 아닌 CloudWatch Logs에 대한 API 키 생성만 허용할 수 있습니다.

CloudWatch Logs API 키 사용 제어

`logs:CallWithBearerToken` 작업은 CloudWatch Logs API 키의 사용을 제어합니다. 자격 증명이 CloudWatch Logs API 키를 사용하지 못하도록 하려면 키와 연결된 IAM 사용자에게 `logs:CallWithBearerToken` 작업을 거부하는 정책을 연결합니다.

정책 예제

자격 증명이 CloudWatch Logs API 키를 생성하고 사용하지 못하도록 방지

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyCWLAPIKeys",
            "Effect": "Deny",
            "Action": [
                "iam:CreateServiceSpecificCredential",
                "logs:CallWithBearerToken"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Warning**
>
> 이 정책은 AWS 서비스별 자격 증명 생성을 지원하는 모든 서비스에 대한 자격 증명 생성을 방지합니다. 자세한 내용은 IAM 사용자를 위한 서비스별 자격 증명을 참조하세요.

자격 증명이 CloudWatch Logs API 키를 사용하지 못하도록 방지

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "logs:CallWithBearerToken",
            "Resource": "*"
        }
    ]
}
```

90일 이내에 만료되는 경우에만 CloudWatch Logs 키 생성 허용

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceSpecificCredential",
            "Resource": "arn:aws:iam::123456789012:user/username",
            "Condition": {
                "StringEquals": {
                    "iam:ServiceSpecificCredentialServiceName": "logs.amazonaws.com"
                },
                "NumericLessThanEquals": {
                    "iam:ServiceSpecificCredentialAgeDays": "90"
                }
            }
        }
    ]
}
```

API 키 교체

API 키를 정기적으로 교체하면 무단 액세스 위험이 줄어듭니다. 조직의 보안 정책에 맞는 교체 일정을 설정하는 것이 좋습니다.

교체 프로세스

로그 전송을 중단하지 않고 API 키를 교체하려면 다음 절차를 따르세요.

1. IAM 사용자를 위한 새 (보조) 자격 증명을 생성합니다.

```
aws iam create-service-specific-credential \
    --user-name cloudwatch-logs-hlc-user \
    --service-name logs.amazonaws.com \
    --credential-age-days 90
```

2. (선택 사항) 보안 검색 및 자동 교체를 AWS Secrets Manager 위해에 새 자격 증명을 저장합니다.
3. 새 자격 증명을 공급업체의 포털로 가져오거나 새 API 키를 사용하도록 애플리케이션 구성을 업데이트합니다.
4. 원래 자격 증명을 비활성으로 설정합니다.

```
aws iam update-service-specific-credential \
    --user-name cloudwatch-logs-hlc-user \
    --service-specific-credential-id ACCA1234EXAMPLE1234 \
    --status Inactive
```

5. CloudWatch에서 로그 그룹의 `IncomingBytes` 지표를 모니터링하여 로그 전송이 영향을 받지 않는지 확인합니다. 자세한 내용은 CloudWatch 지표를 사용한 모니터링을 참조하세요.
6. 새 키로 전송 성공 여부를 확인한 후 이전 자격 증명을 삭제합니다.

```
aws iam delete-service-specific-credential \
    --service-specific-credential-id ACCA1234EXAMPLE1234
```

키 만료 모니터링

기존 API 키의 생성 날짜 및 상태를 확인하려면 list-service-specific-credentials 명령을 사용합니다.

```
aws iam list-service-specific-credentials \
    --user-name cloudwatch-logs-hlc-user \
```

```
    --service-name logs.amazonaws.com
```

응답에는 각 자격 증명Status에 대한 CreateDate 및가 포함됩니다. 이 정보를 사용하여 만료에 가까워지거나 교체 정책에서 허용하는 것보다 오래 활성화된 키를 식별할 수 있습니다.

손상된 API 키에 대한 응답

API 키가 손상되었다고 의심되는 경우 즉시 다음 단계를 수행합니다.

1. 추가 무단 사용을 방지하려면 키를 즉시 비활성화합니다.

```
aws iam update-service-specific-credential \
    --user-name cloudwatch-logs-hlc-user \
    --service-specific-credential-id ACCA1234EXAMPLE1234 \
    --status Inactive
```

2. CloudTrail 로그를 검토하여 무단 액세스 범위를 결정합니다. API 키 사용에 the section called “CloudTrail을 사용하여 API 키 사용량 로깅” 대한 감사를 활성화하는 방법은 섹션을 참조하세요.
3. 에 설명된 교체 프로세스에 따라 대체 키를 생성합니다the section called “교체 프로세스”.
4. 교체가 적용된 후 손상된 키를 삭제합니다.

```
aws iam delete-service-specific-credential \
    --service-specific-credential-id ACCA1234EXAMPLE1234
```

5. 조사하는 동안 IAM 사용자의 모든 보유자 토큰 액세스를 즉시 차단해야 하는 경우 거부 정책을 연결합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "logs:CallWithBearerToken",
        "Resource": "*"
    }
}
```

> **Note**
>
> API를 통해 이러한 작업을 수행하려면 CloudWatch Logs API 키가 아닌 AWS 자격 증명으로 인증해야 합니다.

다음 IAM API 작업을 사용하여 손상된 키를 관리할 수도 있습니다.

- ResetServiceSpecificCredential - 자격 증명을 삭제하지 않고 새 암호를 생성하려면 키를 재설정합니다. 키가 만료되지 않았어야 합니다.

API 키에 대한 보안 모범 사례

CloudWatch Logs API 키를 보호하려면 다음 모범 사례를 따르세요.

- 소스 코드에 API 키를 포함시키지 마십시오. 애플리케이션 코드에서 API 키를 하드 코딩하거나 버전 제어 시스템에 커밋하지 마십시오. 키가 실수로 퍼블릭 리포지토리에 커밋된 경우 AWS 자동 스캔에서 해당 키에 플래그가 지정될 수 있으므로 키를 즉시 교체해야 합니다.
- 보안 암호 관리자를 사용합니다. API 키를 AWS Secrets Manager 또는 이에 상응하는 보안 암호 관리 솔루션에 저장합니다. 이를 통해 중앙 집중식 액세스 제어, 감사 로깅 및 자동 교체가 가능합니다.
- 모든 키에 만료를 설정합니다. API 키를 생성할 때는 항상 `--credential-age-days` 값을 지정합니다. 조직 전체에 최대 키 수명을 적용하려면 `iam:ServiceSpecificCredentialAgeDays` IAM 조건 키를 사용합니다. 예제는 the section called “90일 이내에 만료되는 경우에만 CloudWatch Logs 키 생성 허용” 섹션을 참조하세요.
- 최소 권한 권한을 적용합니다. IAM 사용자의 권한 범위를 필요한 로그 그룹 및 작업으로만 지정합니다. 관리형 CloudWatchLogsAPIKeyAccess 정책을 시작점으로 사용하고 필요에 따라 추가로 제한합니다.
- CloudTrail 로깅을 활성화합니다. 에 대한 CloudTrail 데이터 이벤트를 활성화하여 API 키 사용을 감사합니다`AWS::Logs::LogGroupAuthorization`. the section called “CloudTrail을 사용하여 API 키 사용량 로깅”을(를) 참조하세요.
- IAM Access Analyzer로 모니터링합니다. IAM Access Analyzer를 사용하여 API 키 IAM 사용자와 연결된 미사용 자격 증명 및 지나치게 허용적인 정책을 식별합니다.
- 키를 정기적으로 교체합니다. 교체 일정을 설정하고에 설명된 프로세스를 따릅니다the section called “API 키 교체”.

CloudTrail을 사용하여 API 키 사용량 로깅

AWS CloudTrail 를 사용하여 CloudWatch Logs API 키 사용에 대한 데이터 이벤트를 로깅할 수 있습니다. CloudWatch Logs는 `CallWithBearerToken` 호출에 대한 `AWS::Logs::LogGroupAuthorization` 데이터 이벤트를 내보내므로 API 키를 사용하여 로그를 전송하는 시기와 방법을 감사할 수 있습니다.

CloudWatch Logs API 키 사용에 대해 CloudTrail 로깅을 활성화하려면: CloudWatch

> **Note**
>
> 추적에 지정하는 S3 버킷에는 CloudTrail이 로그 파일을 쓸 수 있도록 허용하는 버킷 정책이 있어야 합니다. 자세한 내용은 CloudTrail에 대한 Amazon S3 버킷 정책을 참조하세요.

1. 추적 생성:

```
aws cloudtrail create-trail \
    --name cloudwatch-logs-api-key-audit \
    --s3-bucket-name my-cloudtrail-bucket \
    --region us-east-1
```

2. CloudWatch Logs 로그 그룹 권한 부여 이벤트를 캡처하도록 고급 이벤트 선택기를 구성합니다.

```
aws cloudtrail put-event-selectors \
    --region us-east-1 \
    --trail-name cloudwatch-logs-api-key-audit \
    --advanced-event-selectors '[{
        "Name": "CloudWatch Logs API key authorization events",
        "FieldSelectors": [
            { "Field": "eventCategory", "Equals": ["Data"] },
            { "Field": "resources.type", "Equals":
 ["AWS::Logs::LogGroupAuthorization"] }
        ]
    }]'
```

3. 추적 로깅 시작:

```
aws cloudtrail start-logging \
    --name cloudwatch-logs-api-key-audit \
    --region us-east-1
```

OTLP 엔드포인트를 사용하여 로그 전송(OpenTelemetry Logs)

OpenTelemetry Logs 엔드포인트(`/v1/logs`)는 JSON 또는 Protobuf 인코딩의 OpenTelemetry Protocol(OTLP) 로그 데이터를 수락합니다. 구성 및 사용량을 포함하여 OTLP 엔드포인트에 대한 자세한 내용은 OpenTelemetry를 사용하여 CloudWatch로 지표 및 추적 전송을 참조하세요.

보유자 토큰 인증을 사용하는 경우 계속하기 the section called “베어러 토큰 인증” 전에의 설정 단계를 완료합니다.

요청 형식

- 메서드: `POST`
- Content-Type: `application/json` 또는 `application/x-protobuf`
- 로그 그룹: `x-aws-log-group` 헤더만(쿼리 파라미터는 지원되지 않음)
- 로그 스트림: `x-aws-log-stream` 헤더

요청 예제

```
curl -X POST "https://logs.<region>.amazonaws.com/v1/logs" \
  -H "Authorization: Bearer ACWL<token>" \
  -H "Content-Type: application/json" \
  -H "x-aws-log-group: MyLogGroup" \
  -H "x-aws-log-stream: MyLogStream" \
  -d '{
  "resourceLogs": [
    {
      "resource": {
        "attributes": [
          {
            "key": "service.name",
            "value": { "stringValue": "my-service" }
          }
        ]
      },
      "scopeLogs": [
        {
          "scope": {
            "name": "my-library",
            "version": "1.0.0"
```

```
          },
          "logRecords": [
            {
              "timeUnixNano": "1741900000000000000",
              "severityNumber": 9,
              "severityText": "INFO",
              "body": {
                "stringValue": "User logged in successfully"
              },
              "attributes": [
                {
                  "key": "user.id",
                  "value": { "stringValue": "12345" }
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}'
```

응답

성공(모든 이벤트 수락됨):

```
HTTP 200 OK
{}
```

부분 성공(일부 이벤트가 거부됨):

```
{
  "partialSuccess": {
    "rejectedLogRecords": 5,
    "errorMessage": "{\"tooOldLogEventCount\": 3, \"tooNewLogEventCount\": 1,
 \"expiredLogEventCount\": 1}"
  }
}
```

요청 Content-Type이 인 경우 `application/x-protobuf`응답은 필드가 동일한 직렬화된 `ExportLogsServiceResponse` protobuf 메시지로 반환됩니다.

OTLP 관련 동작

다음 동작은 OTLP 엔드포인트에 고유하며 다른 HTTP 수집 엔드포인트에는 없습니다.

- Retry-After 헤더 - 클라이언트가 언제 재시도해야 하는지를 나타내는 503 및 429 응답에 포함됩니다.

HLC 엔드포인트를 사용하여 로그 전송(HLC 로그)

HLC Logs 엔드포인트(`/services/collector/event`)는 HTTP Log Collector(HLC) 형식을 기반으로 합니다.

보유자 토큰 인증을 사용하는 경우 계속하기 the section called “베어러 토큰 인증” 전에의 설정 단계를 완료합니다.

입력 모드

각 이벤트는 필수 `"event"` 필드가 있는 JSON 객체입니다. 선택적 메타데이터 필드: `"time"`, `"host"`, `"source"`, `"sourcetype"`, `"index"`.

단일 이벤트:

```
{"event":"Hello world!","time":1486683865.0}
```

이벤트의 JSON 배열:

```
[
  {"event":"msg1","time":1486683865.0},
  {"event":"msg2","time":1486683866.0}
]
```

연결된/배치된 이벤트(어레이 래퍼 없음):

```
{"event":"msg1","time":1486683865.0}{"event":"msg2","time":1486683866.0}
```

이벤트 필드(필수)

`"event"` 필드는 필수입니다. 값은 모든 JSON 유형일 수 있습니다.

```
{"event":"a string message"}
```

```
{"event":{"message":"structured data","severity":"INFO"}}
{"event":42}
{"event":true}
```

"event" 필드가 없는 객체는 자동으로 건너뜁니다.

```
{"message":"this is skipped — no event field"}
```

시간 필드(선택 사항)

"time" 필드는 에포크 초(밀리초 아님) 단위이며, 10진수는 1초 미만의 정밀도에 대한 선택적 10진수입니다.

형식	예제	다음으로 해석됨
Float	"time":1486683865.500	1486683865500ms
Integer	"time":1486683865	1486683865000ms
문자열(부동 소수점)	"time":"1486683865.500"	1486683865500ms
문자열(정수)	"time":"1486683865"	1486683865000ms
누락됨	(시간 필드 없음)	서버 현재 시간
잘못됨	"time":"invalid"	서버 현재 시간

Content-Type

만 application/json 허용됩니다.

허용되는 JSON 값 유형

최상위 유형	동작
가 있는 객체 "event"	수락됨
가 없는 객체 "event"	건너뜀

최상위 유형	동작
객체 배열	개별적으로 처리되는 각 요소
연결된 객체	개별적으로 처리되는 각 객체
프리미티브(문자열, 숫자, 부울, null)	건너뜀

엔드포인트 형식

HLC 엔드포인트 URL은 다음 형식을 따릅니다.

```
https://logs.<region>.amazonaws.com/services/collector/event?
logGroup=<name>&logStream=<name>[&entityName=<name>&entityEnvironment=<environment>]
```

필수 파라미터:

- `<region>` – AWS 리전(예: , `us-east-1`eu-west-1)
- `logGroup` - URL로 인코딩된 로그 그룹 이름
- `logStream` - URL로 인코딩된 로그 스트림 이름

선택적 파라미터:

필요에 따라 다음 쿼리 파라미터를 포함하여 로그 이벤트를 `Service`엔터티와 연결할 수 있습니다. HLC 엔드포인트를 통해 전송된 로그는 사용자 지정 원격 측정이므로 엔터티와 자동으로 연결되지 않습니다. CloudWatch Logs는 이러한 파라미터를 제공하여가 로 `KeyAttributes.Type` 설정된 개체를 생성하고 이를 로그 이벤트와 `Service` 연결합니다. 이를 통해 CloudWatch의 관련 기능 탐색은 이러한 로그를 동일한 서비스의 다른 원격 측정(지표, 추적 및 로그)과 상호 연관시킬 수 있으므로 다양한 신호 유형에서 애플리케이션의 문제를 더 쉽게 해결하고 모니터링할 수 있습니다. 개체 및 관련 원격 측정에 대한 자세한 내용은 사용자 지정 원격 측정에 관련 정보 추가를 참조하세요.

- `entityName` - 로그 이벤트와 연결할 서비스 엔터티의 이름입니다. 이 값은 엔터티`KeyAttributes.Name`(예: `my-application` 또는 `api.myservice.com`)로 저장됩니다.
- `entityEnvironment` - 서비스가 호스팅되는 환경 또는 서비스가 속한 환경입니다. 이 값은 개체`KeyAttributes.Environment`(예: , `production` `ec2:default`또는 `eks:my-cluster/default`)로 저장됩니다.

요청 형식

다음 헤더 및 본문과 함께 HTTP POST를 사용하여 로그를 전송합니다.

헤더:

- `Authorization: Bearer <your-bearer-token>`
- `Content-Type: application/json`

본문 형식:

요청 본문은 이벤트 배열과 함께 JSON 형식이어야 합니다.

```
{
    "event": [
        {
            "time": 1730141374.001,
            "event": "Application started successfully",
            "host": "web-server-1",
            "source": "application.log",
            "severity": "info"
        },
        {
            "time": 1730141374.457,
            "event": "User login successful",
            "host": "web-server-1",
            "source": "auth.log",
            "user": "john.doe"
        }
    ]
}
```

필드 설명:

- `time` - Unix epoch 타임스탬프, 1초 미만의 정밀도에 대한 선택적 10진수(선택 사항)
- `event` - 로그 메시지 또는 이벤트 데이터(필수)
- `host` - 소스 호스트 이름 또는 식별자(선택 사항)
- `source` - 로그 소스 식별자(선택 사항)

필요에 따라 추가 사용자 지정 필드를 포함할 수 있습니다.

요청 예제

```
curl -X POST "https://logs.<region>.amazonaws.com/services/collector/event?
logGroup=MyLogGroup&logStream=MyStream" \
  -H "Authorization: Bearer ACWL<token>" \
  -H "Content-Type: application/json" \
  -d '{"event":{"message":"User logged
 in","user_id":"u-123"},"time":1486683865.0,"host":"web-01","source":"auth-service"}'
```

모범 사례

이벤트 일괄 처리

성능 및 효율성 향상을 위해:

- 가능한 경우 단일 요청으로 여러 이벤트 배치
- 권장 배치 크기: 요청당 이벤트 10~100개
- 최대 요청 크기: 1MB

오류 처리

애플리케이션에서 적절한 오류 처리를 구현합니다. 일반적인 HTTP 상태 코드:

- `200 OK` - 로그가 성공적으로 수집됨
- `400 Bad Request` - 잘못된 요청 형식 또는 파라미터
- `401 Unauthorized` - 유효하지 않거나 만료된 보유자 토큰
- `403 Forbidden` - 권한 부족
- `404 Not Found` - 로그 그룹 또는 스트림이 존재하지 않음
- `429 Too Many Requests` - 속도 제한 초과
- `500 Internal Server Error` - 서비스 오류(지수 백오프로 재시도)

제한 사항

- 최대 이벤트 크기: 이벤트당 256KB
- 최대 요청 크기: 1MB

- 요청당 최대 이벤트 수: 10,000
- 로그 그룹 이름은 CloudWatch Logs 명명 규칙을 따라야 합니다.
- 보유자 토큰 인증을 사용하는 경우 로그 그룹에서 보유자 토큰 인증을 활성화해야 합니다.

NDJSON 엔드포인트를 사용하여 로그 전송(ND-JSON 로그)

ND-JSON Logs 엔드포인트(`/ingest/bulk`)는 NDJSON(Newline Delimited JSON) 형식의 로그를 허용합니다. 각 줄에는 줄 바꿈 문자로 구분된 정확히 하나의 JSON 값이 포함되어 있습니다.

보유자 토큰 인증을 사용하는 경우 계속하기 the section called “베어러 토큰 인증” 전에의 설정 단계를 완료합니다.

요청 형식

(LF) 또는 \n (\r\nCRLF)로 구분하여 줄당 하나의 JSON 값을 전송합니다. 빈 줄은 자동으로 무시됩니다.

```
{"timestamp":1771007942000,"message":"event one","level":"INFO"}
{"timestamp":1771007943000,"message":"event two","level":"ERROR"}
{"timestamp":1771007944000,"message":"event three","level":"DEBUG"}
```

`application/json` 및 `application/x-ndjson` 모두 Content-Type으로 허용됩니다.

허용되는 JSON 값 유형

NDJSON 사양(RFC 8259)에 따라 유효한 JSON 값은 각 행에서 허용됩니다.

JSON 객체(가장 일반적):

```
{"timestamp":1771007942000,"message":"User logged in","service":"auth"}
{"timestamp":1771007943000,"error":"Connection timeout","service":"api"}
```

JSON 배열(개별 이벤트로 확장됨):

```
[{"timestamp":1000,"message":"a"},{"timestamp":2000,"message":"b"}]
```

이 한 줄은 2개의 이벤트를 생성합니다. 각 배열 요소는 별도의 로그 이벤트가 됩니다.

기본 값:

```
"a plain string log message"
42
true
null
```

각 프리미티브는 서버의 현재 타임스탬프가 있는 자체 이벤트가 됩니다.

혼합 유형:

```
{"timestamp":1771007942000,"message":"structured event"}
"unstructured string message"
42
{"timestamp":1771007943000,"error":"something failed"}
```

4줄 모두 유효한 이벤트로 허용됩니다.

라인 콘텐츠	동작
JSON 객체	수락됨, 있는 경우 타임스탬프 추출됨
JSON 배열	평면화됨 - 각 요소는 별도의 이벤트가 됩니다.
빈 배열 []	수락됨, 이벤트 0개 생성
JSON 문자열	이벤트 메시지로 수락됨
JSON 번호	이벤트 메시지로 수락됨
JSON 부울	이벤트 메시지로 수락됨
JSON null	이벤트 메시지로 수락됨
잘못된 JSON	건너뜀(계산됨, 처리 계속)
빈 줄	무시됨(건너뛴 것으로 계산되지 않음)

타임스탬프 필드

"timestamp" 필드는 에포크 밀리초(초 아님) 단위입니다.

형식	예제	다음으로 해석됨
숫자(millis)	"timestamp":1771007942000	1771007942000ms
누락됨	(타임스탬프 필드 없음)	서버 현재 시간
숫자가 아닌	"timestamp":"invalid"	서버 현재 시간
비객체 라인	"hello", 42, true	서버 현재 시간

잘못된 줄

유효한 JSON이 아닌 행은 자동으로 건너뛰고 계산됩니다. 처리는 다음 줄로 계속됩니다.

```
{"message":"valid event"}
this is not valid json
{"message":"another valid event"}
```

결과: 2개의 이벤트가 수집되었고 1개를 건너뛰었습니다. HTTP 200을 반환합니다.

모든 줄이 유효하지 않은 경우는 HTTP 400와 함께를 반환합니다"All events were invalid".

요청 예제

```
curl -X POST "https://logs.<region>.amazonaws.com/ingest/bulk?
logGroup=MyLogGroup&logStream=MyStream" \
  -H "Authorization: Bearer ACWL<token>" \
  -H "Content-Type: application/x-ndjson" \
  -d '{"timestamp":1771007942000,"message":"User logged in","level":"INFO"}
{"timestamp":1771007943000,"message":"Query took 42ms","level":"DEBUG"}
{"timestamp":1771007944000,"error":"Connection refused","level":"ERROR"}'
```

응답

성공(모든 이벤트 수락됨):

```
HTTP 200 OK
{}
```

부분 성공(일부 이벤트가 거부됨):

```
{
  "partialSuccess": {
    "rejectedLogRecords": 5,
    "errorMessage": "{\"tooOldLogEventCount\": 3, \"tooNewLogEventCount\": 1,
 \"expiredLogEventCount\": 1}"
  }
}
```

`rejectedLogRecords` 필드는 거부된 총 이벤트 수입니다. `errorMessage` 필드에는 거부 이유별 JSON 인코딩 분류가 포함되어 있습니다.

- `tooOldLogEventCount` - 보존 기간보다 오래된 타임스탬프가 있는 이벤트
- `tooNewLogEventCount` - 타임스탬프가 너무 먼 미래의 이벤트
- `expiredLogEventCount` - 처리 중에 만료된 이벤트

모범 사례

이벤트 일괄 처리

성능 및 효율성 향상을 위해:

- 가능한 경우 단일 요청으로 여러 이벤트를 일괄 처리
- 권장 배치 크기: 요청당 이벤트 10~100개
- 최대 요청 크기: 1MB

오류 처리

애플리케이션에서 적절한 오류 처리를 구현합니다. 일반적인 HTTP 상태 코드:

- `200 OK` - 로그가 성공적으로 수집됨
- `400 Bad Request` - 잘못된 요청 형식 또는 파라미터
- `401 Unauthorized` - 유효하지 않거나 만료된 보유자 토큰

- `403 Forbidden` - 권한 부족
- `404 Not Found` - 로그 그룹 또는 스트림이 존재하지 않음
- `429 Too Many Requests` - 속도 제한 초과
- `500 Internal Server Error` - 서비스 오류(지수 백오프로 재시도)

제한 사항

- 최대 이벤트 크기: 이벤트당 256KB
- 최대 요청 크기: 1MB
- 요청당 최대 이벤트 수: 10,000
- 로그 그룹 이름은 CloudWatch Logs 명명 규칙을 따라야 합니다.
- 보유자 토큰 인증을 사용하는 경우 로그 그룹에서 보유자 토큰 인증을 활성화해야 합니다.

구조화된 JSON 엔드포인트를 사용하여 로그 전송(구조화된 JSON 로그)

구조화된 JSON 로그 엔드포인트(`/ingest/json`)는 단일 JSON 객체 또는 객체의 JSON 배열인 표준 JSON을 허용합니다. 이 엔드포인트는 각 이벤트가 JSON 객체인 구조화된 로그 데이터를 위해 설계되었습니다.

보유자 토큰 인증을 사용하는 경우 계속하기 the section called “베어러 토큰 인증” 전에의 설정 단계를 완료합니다.

요청 형식

`application/json` 만 Content-Type으로 허용됩니다.

단일 JSON 객체:

```
{"timestamp":1771007942000,"message":"single event","level":"INFO"}
```

객체의 JSON 배열:

```
[
  {"timestamp":1771007942000,"message":"event one","level":"INFO"},
  {"timestamp":1771007943000,"message":"event two","level":"ERROR"}
```

```
]
```

허용되는 JSON 값 유형

이 엔드포인트는 엄격합니다. JSON 객체만 이벤트로 허용됩니다.

Input	동작
단일 JSON 객체	하나의 이벤트로 수락됨
객체의 JSON 배열	각 객체는 별도의 이벤트가 됩니다.
빈 배열 `[]`	수락됨, 이벤트 0개 생성
배열의 비객체(문자열, 숫자 등)	건너뜀
최상위 프리미티브(`"hello"`, `42`)	건너뜀
연결된 객체 `{...}{...}`	첫 번째 객체만 구문 분석됨

예 - 혼합 유형이 있는 배열:

```
[
  {"timestamp":1771007942000,"message":"valid object"},
  "just a string",
  42,
  {"timestamp":1771007943000,"message":"another valid object"}
]
```

결과: 수집된 이벤트 2개(객체), 건너뛴 이벤트 2개(문자열 및 숫자).

타임스탬프 필드

`"timestamp"` 필드는 NDJSON 엔드포인트와 동일한 에포크 밀리초 단위입니다.

형식	예제	다음으로 해석됨
숫자(millis)	`"timestamp":1771007942000`	1771007942000ms

형식	예제	다음으로 해석됨
누락됨	(타임스탬프 필드 없음)	서버 현재 시간
숫자가 아닌	`"timestamp":"invalid"`	서버 현재 시간

요청 예제

```
curl -X POST "https://logs.<region>.amazonaws.com/ingest/json?
logGroup=MyLogGroup&logStream=MyStream" \
  -H "Authorization: Bearer ACWL<token>" \
  -H "Content-Type: application/json" \
  -d '[{"timestamp":1771007942000,"message":"User logged in","user_id":"u-123"},
{"timestamp":1771007943000,"message":"Order placed","order_id":"o-456"}]'
```

응답

성공(모든 이벤트 수락됨):

```
HTTP 200 OK
{}
```

부분 성공(일부 이벤트가 거부됨):

```
{
  "partialSuccess": {
    "rejectedLogRecords": 5,
    "errorMessage": "{\"tooOldLogEventCount\": 3, \"tooNewLogEventCount\": 1,
 \"expiredLogEventCount\": 1}"
  }
}
```

`rejectedLogRecords` 필드는 거부된 총 이벤트 수입니다. `errorMessage` 필드에는 거부 이유별 JSON 인코딩 분류가 포함되어 있습니다.

- `tooOldLogEventCount` - 보존 기간보다 오래된 타임스탬프가 있는 이벤트
- `tooNewLogEventCount` - 타임스탬프가 너무 먼 미래의 이벤트
- `expiredLogEventCount` - 처리 중에 만료된 이벤트

모범 사례

이벤트 일괄 처리

성능 및 효율성 향상을 위해:

- 가능한 경우 단일 요청으로 여러 이벤트 배치
- 권장 배치 크기: 요청당 이벤트 10~100개
- 최대 요청 크기: 1MB

오류 처리

애플리케이션에서 적절한 오류 처리를 구현합니다. 일반적인 HTTP 상태 코드:

- `200 OK` - 로그가 성공적으로 수집됨
- `400 Bad Request` - 잘못된 요청 형식 또는 파라미터
- `401 Unauthorized` - 유효하지 않거나 만료된 보유자 토큰
- `403 Forbidden` - 권한 부족
- `404 Not Found` - 로그 그룹 또는 스트림이 존재하지 않음
- `429 Too Many Requests` - 속도 제한 초과
- `500 Internal Server Error` - 서비스 오류(지수 백오프로 재시도)

제한 사항

- 최대 이벤트 크기: 이벤트당 256KB
- 최대 요청 크기: 1MB
- 요청당 최대 이벤트 수: 10,000
- 로그 그룹 이름은 CloudWatch Logs 명명 규칙을 따라야 합니다.
- 보유자 토큰 인증을 사용하는 경우 로그 그룹에서 보유자 토큰 인증을 활성화해야 합니다.

HTTP 수집 엔드포인트 비교

기능	HLC 로그	ND-JSON 로그	구조화된 JSON 로그	OpenTelemetry 로그
경로	`/services/collector/event`	`/ingest/bulk`	`/ingest/json`	`/v1/logs`
Content-Type	`application/json`	`application/json` 또는 `application/x-ndjson`	`application/json`	`application/json` 또는 `application/x-protobuf`
타임스탬프 필드	`"time"`(초)	`"timestamp"` (밀리초)	`"timestamp"` (밀리초)	`"timeUnixNano"` (나노초)
필수 필드	`"event"`	없음	없음	OTLP 구조 (`"resourceLogs"`)
부분 성공 응답	아니요	예	예	예
쿼리 파라미터 지원	예	예	예	아니요(헤더만 해당)
개체 메타데이터	예	예	예	아니요
프리미티브 허용	아니요	예	아니요	아니요
라인 기반 구문 분석	아니요	예	아니요	아니요
Protobuf 지원	아니요	아니요	아니요	예
재시도 후 헤더	아니요	아니요	아니요	예

엔드포인트 선택

- HLC 형식을 사용하시나요? HLC 로그를 사용합니다. 기존 HLC 페이로드는 최소한의 변경으로 작동합니다.
- line-by-line 로그 스트리밍? ND-JSON 로그를 사용합니다. 행당 하나의 이벤트를 내보내는 로그 파이프라인에 가장 적합합니다. 가장 유연함 - 모든 JSON 값 유형을 허용합니다.
- 구조화된 JSON 페이로드를 보내나요? 구조화된 JSON 로그를 사용합니다. 올바른 형식의 JSON 객체 또는 배열을 생성하는 애플리케이션에 가장 적합합니다.
- 이미 OpenTelemetry를 사용하고 있나요? OpenTelemetry Logs를 사용합니다. OTLP JSON 또는 Protobuf 형식을 수락하고 재시도 의미 체계를 사용하여 부분 성공 응답을 지원합니다.

Syslog 수집

Amazon CloudWatch Logs 관리형 syslog 수집을 사용하면 에이전트를 설치하거나 관리하지 않고도 방화벽, 라우터, 스위치 및 Linux 서버에서 CloudWatch Logs로 직접 syslog 메시지(RFC 5424, RFC 3164 및 Cisco FTD/ASA)를 보낼 수 있습니다. syslog 소스는 TCP, TCP+TLS 또는 UDP를 통해 계정의 VPC 엔드포인트로 메시지를 전송합니다. 트래픽은 AWS PrivateLink 를 통해 CloudWatch Logs syslog 서비스로 터널링되며,이 서비스는 수신 메시지를 자동으로 구문 분석하고 시설, 심각도, 호스트 이름 및 애플리케이션 이름과 같은 구조화된 필드를 추출하므로 사용자 지정 구문 분석 파이프라인이 필요하지 않습니다. 그런 다음 CloudWatch Logs Analytics를 사용하여 이러한 필드를 쿼리하여 보안 이벤트를 조사하거나 연결 문제를 해결할 수 있습니다. 이를 통해 인프라 로그 가시성을 중앙 집중화하고, 운영 워크플로를 간소화하고, 분산 환경에서 로그 수집 에이전트를 배포하고 유지 관리하는 데 드는 오버헤드를 줄일 수 있습니다.

syslog 소스가 이미 Amazon VPC 내에 있는 경우 VPC 엔드포인트로 직접 메시지를 보낼 수 있습니다. 소스가 외부 AWS (온프레미스 데이터 센터, 지사 또는 코로케이션 시설)에 있는 경우 VPN 또는 Direct Connect 연결을 통해 VPC 엔드포인트에 연결할 수 있습니다.

지원되는 프로토콜 및 포트

프로토콜	포트	참고
TCP + TLS	6514	전송 중에 암호화됩니다. 규정 준수 요구 사항에 권장됩니다.
TCP 일반 텍스트	1514	일반 텍스트 over AWS PrivateLink (네트워크 격리).
UDP	514	최선의 전송.

포트 6514의 TLS는 Amazon Trust Services에서 발급한 관리형 인증서를 사용하여 네트워크 로드 AWS밸런서에서 종료됩니다. syslog 클라이언트는 특별한 구성 없이이 인증서를 자동으로 신뢰합니다.

> **Note**
>
> UDP는 최선의 프로토콜입니다. 네트워크 조건으로 인해 메시지가 손실될 수 있습니다. 안정적인 전송을 위해 TCP를 사용합니다.

지원되는 syslog 형식

CloudWatch Logs는 다음 syslog 형식을 자동으로 감지하고 구문 분석합니다.

- RFC 5424(최신 형식) - 구조화된 데이터, ISO 8601 타임스탬프, 명시적 애플리케이션 이름 및 프로세스 ID 필드를 포함합니다.
- RFC 3164(BSD syslog, 레거시 형식) - BSD 스타일 타임스탬프와 TAG 필드를 포함합니다. 방화벽, 라우터 및 스위치와 같은 네트워크 디바이스에서 여전히 널리 사용됩니다.
- Cisco FTD/ASA - Cisco Firepower Threat Defense(FTD) 및 적응형 보안 어플라이언스(ASA) 디바이스에서 사용하는 syslog 형식입니다. 메시지는 메시지 본문의 `%FTD-` 또는 `%ASA-` 태그로 식별됩니다.

메시지는 로그 그룹에 원래 원시 형식으로 저장됩니다. CloudWatch Logs는 각 형식에서 구조화된 필드를 자동으로 추출하며, CloudWatch Logs Insights를 사용하여 쿼리할 수 있습니다.

RFC 5424 추출 필드

필드	설명
`facility`	로그 범주 이름(예: , `kernauth`, `local0`).
`facilityCode`	숫자로 된 시설 코드(0~23).
`severity`	심각도 수준 이름(예: , `emergerr`, `info`).
`severityCode`	숫자 심각도 코드(0~7).
`timestamp`	ISO 8601 형식의 메시지 타임스탬프입니다.
`hostname`	소스 디바이스 호스트 이름입니다.
`appName`	애플리케이션 이름.
`procId`	프로세스 ID
`msgId`	메시지 식별자입니다.
`structuredData`	RFC 5424 구조화된 데이터 요소(키-값 메타데이터).

필드	설명
message	메시지 본문입니다.

RFC 3164 추출 필드

필드	설명
facility	로그 범주 이름입니다.
facilityCode	숫자로 된 시설 코드(0~23).
severity	심각도 수준 이름입니다.
severityCode	숫자 심각도 코드(0~7).
timestamp	메시지 타임스탬프(BSD 형식에서 ISO 8601로 변환됨).
hostname	소스 디바이스 호스트 이름입니다.
appName	애플리케이션 이름(TAG 필드에서 발췌).
procId	프로세스 ID(있는 경우 TAG 필드에서 추출).
message	메시지 본문입니다.

Cisco FTD/ASA 추출 필드

필드	설명
device	디바이스 유형(FTD, ASA또는 FMC-AUDIT-LOG).
timestamp	메시지 타임스탬프(디바이스 구성에 따라 RFC 3164 또는 RFC 5424 형식).
deviceId	디바이스 호스트 이름(어플라이언스에 device-id 로깅이 구성된 경우 존재).

필드	설명
severity	심각도 수준 이름(예: , informational warning, critical).
severityLevel	숫자 심각도 수준(0~7).
messageId	Cisco 숫자 메시지 식별자(예: , 106023302013).
subsystem	하위 시스템 이름(특정 메시지 유형에 해당).
message	본문이 Cisco의 구조화된 형식을 사용하는 경우 메시지 본문(일반 텍스트) 또는 개별 키-값 필드입니다.

메시지 전송

서버가 각 요청에 대한 상태 코드를 반환하는 HTTP 기반 수집과 달리 syslog는 메시지당 성공 확인을 발신자에게 다시 제공하지 않습니다. 서비스에서 메시지를 수신하면 메시지가 버퍼링되고 로그 그룹으로 전송되며 일시적 오류에 대한 재시도가 제공됩니다. 전송 보장은 선택한 전송 프로토콜에 따라 달라집니다.

- TCP(포트 6514 및 1514) - 정상 작동 조건에서 안정적인 전송을 제공합니다.

  전송이 불가능한 경우 서비스는 TCP 연결을 재설정하여 클라이언트에 실패 신호를 보냅니다. 해당 연결에서 진행 중인 메시지는 삭제될 수 있지만 연결 재설정은 즉각적인 역압을 제공하므로 조건이 해결될 때까지 클라이언트가 로컬에서 문제를 감지하고 메시지를 버퍼링할 수 있습니다. 용량 압력에서 서비스는 새 TCP 연결을 조기에 거부하여 동일한 역압 신호를 제공합니다.

  연결 재설정을 유발하는 조건은 다음과 같습니다.

  - VPC 엔드포인트 정책은 액세스를 거부합니다.
  - 계정의 할당PutLogEvents량을 초과했습니다.
  - 대상 로그 그룹이 존재하지 않습니다.
  - 로그 그룹의 리소스 정책이 액세스를 거부합니다.
- UDP(포트 514) - 최선의 전송. 전송할 수 없는 메시지는 발신자에 대한 피드백 없이 삭제됩니다. 네트워크 혼잡 또는 용량 제약으로 인해 메시지가 손실될 수도 있습니다. 사용 사례에 안정적인 전송이 중요한 경우 TCP를 사용합니다.

전송 문제를 감지하고 대응하려면 CloudWatch에서 `SyslogMessagesDropped` 지표를 모니터링합니다. `Reason` 차원은 수정 조치를 취할 수 있도록 메시지가 삭제된 이유를 나타냅니다. 자세한 내용은 the section called “모니터링” 단원을 참조하십시오.

할당량 및 제한

Limit	값	설명
최대 메시지 크기(TCP)	64KB	표준 syslog 메시지는 일반적으로이 제한보다 훨씬 낮습니다. 더 큰 메시지가 필요한 사용 사례가 있는 경우 AWS Support에 문의하세요.
최대 메시지 크기(UDP)	8KB	표준 syslog 메시지는 일반적으로이 제한보다 훨씬 낮습니다. 더 큰 메시지가 필요한 사용 사례가 있는 경우 AWS Support에 문의하세요.
수집 처리량	와 공유됨 `PutLogEvents`	Syslog 수집은 계정의 할당`PutLogEvents` 량(기본적으로 리전별 계정당 초당 5,000개의 요청)에 포함됩니다.

`PutLogEvents` 할당량은 조정이 가능합니다. syslog 트래픽에 더 높은 처리량이 필요한 경우 Service Quotas를 통해 할당량 증가를 요청합니다.

syslog 수집 설정

이 섹션에서는 CloudWatch Logs로 syslog 수집을 설정하는 단계를 안내합니다. syslog 서비스에 대한 VPC 엔드포인트, 메시지를 수신할 로그 그룹, syslog 서비스를 승인하는 리소스 정책, VPC 엔드포인트에서 로그 그룹으로 트래픽을 라우팅하는 syslog 구성을 생성합니다.

AWS Management Console, AWS CLI또는 AWS SDKs. 다음 지침은 콘솔과 AWS CLI 예제를 모두 제공합니다.

사전 조건

syslog 수집을 설정하는 데 사용하는 IAM 자격 증명(사용자 또는 역할)에는 VPC 엔드포인트, 로그 그룹, 리소스 정책 및 syslog 구성을 생성할 수 있는 권한이 있어야 합니다. 다음 예제 정책은 최소 필수 권한을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:PutResourcePolicy",
        "logs:DeleteResourcePolicy",
        "logs:PutSyslogConfiguration",
        "logs:ListSyslogConfigurations",
        "logs:DeleteSyslogConfiguration"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpc",
        "ec2:ModifyVpcAttribute",
        "ec2:CreateSubnet",
        "ec2:CreateSecurityGroup",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateVpcEndpoint",
        "ec2:ModifyVpcEndpoint",
        "ec2:DescribeVpcEndpoints"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 이미 VPC, 서브넷 및 보안 그룹이 있는 경우 EC2 작업에 대한 `ec2:CreateVpcEndpointec2:ModifyVpcEndpoint`, 및 `ec2:DescribeVpcEndpoints` 권한만 있으면 됩니다.

1단계: VPC 생성 또는 식별

syslog 생성 디바이스가 있는 온프레미스 네트워크(VPN 또는 Direct Connect를 통해)에서 연결할 수 있는 VPC가 필요합니다. 이미 데이터 센터에 연결된 VPC가 있는 경우이 단계를 건너뛰고 기존 VPC 및 서브넷 IDs 사용합니다.

Console

VPC(콘솔)를 만들려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 VPCs 선택한 다음 VPC 생성을 선택합니다.
3. 생성할 리소스에서 VPC 전용을 선택합니다.
4. IPv4 CIDR 블록에 `10.0.0.0/16` (또는 온프레미스 네트워크와 충돌하지 않는 CIDR)를 입력합니다.
5. VPC 생성을 선택합니다.
6. 새로 생성된 VPC를 선택하고 작업, VPC 설정 편집을 선택합니다. DNS 확인 및 DNS 호스트 이름을 모두 활성화한 다음 저장을 선택합니다.
7. VPC 엔드포인트에 대한 서브넷을 생성합니다.
   a. 탐색 창에서 서브넷을 선택한 다음 서브넷 생성을 선택합니다.
   b. VPC ID에서 생성한 VPC를 선택합니다.
   c. 가용 영역에서 가용 영역을 선택합니다.
   d. IPv4 서브넷 CIDR 블록에를 입력합니다`10.0.1.0/24`.
   e. 서브넷 생성(Create subnet)을 선택합니다.

AWS CLI

```
REGION=us-east-1
```

```
# Create VPC
VPC_ID=$(aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --region $REGION \
  --query 'Vpc.VpcId' --output text)

aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support --region $REGION
aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames --region
 $REGION

# Create a subnet for the VPC endpoint
SUBNET_ID=$(aws ec2 create-subnet \
  --vpc-id $VPC_ID \
  --cidr-block 10.0.1.0/24 \
  --availability-zone ${REGION}a \
  --region $REGION \
  --query 'Subnet.SubnetId' --output text)

echo "VPC: $VPC_ID, Subnet: $SUBNET_ID"
```

> **Note**
>
> VPC 엔드포인트는 서브넷에 프라이빗 IP가 있는 탄력적 네트워크 인터페이스(ENI)를 생성합니다. 온프레미스 디바이스는 VPN 또는 Direct Connect 연결을 통해이 IP에 도달합니다. 네트워크 라우팅이 디바이스에서 서브넷 CIDR로의 트래픽을 허용하는지 확인합니다.

2단계: 보안 그룹 만들기

VPC의 인바운드 syslog 트래픽을 허용하는 VPC 엔드포인트에 대한 보안 그룹을 생성합니다. 이렇게 하면 엔드포인트에 syslog를 전송할 수 있는 리소스가 제어됩니다.

Console

보안 그룹을 생성하려면(콘솔)

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹을 선택한 다음 보안 그룹 생성을 선택합니다.
3. 보안 그룹 이름(Security group name)에 `syslog-vpce-sg`를 입력합니다.

4. 설명(Description)에 `Allow syslog traffic to VPC endpoint`를 입력합니다.
5. VPC에서 1단계에서 생성하거나 식별한 VPC를 선택합니다.
6. 인바운드 규칙 섹션에서 규칙 추가를 선택하고 다음 규칙을 추가합니다.
   - 규칙 1: 유형 = 사용자 지정 TCP, 포트 범위 = 6514, 소스 = 10.0.0.0/16 (VPC CIDR)
   - 규칙 2: 유형 = 사용자 지정 TCP, 포트 범위 = 1514, 소스 = 10.0.0.0/16
   - 규칙 3: 유형 = 사용자 지정 UDP, 포트 범위 = 514, 소스 = 10.0.0.0/16
7. 보안 그룹 생성을 선택합니다.

AWS CLI

```
VPCE_SG_ID=$(aws ec2 create-security-group \
  --group-name syslog-vpce-sg \
  --description "Allow syslog traffic to VPC endpoint" \
  --vpc-id $VPC_ID \
  --region $REGION \
  --query 'GroupId' --output text)

# Allow TCP+TLS (port 6514), TCP plaintext (port 1514), and UDP (port 514)
aws ec2 authorize-security-group-ingress --group-id $VPCE_SG_ID \
  --protocol tcp --port 6514 --cidr 10.0.0.0/16 --region $REGION
aws ec2 authorize-security-group-ingress --group-id $VPCE_SG_ID \
  --protocol tcp --port 1514 --cidr 10.0.0.0/16 --region $REGION
aws ec2 authorize-security-group-ingress --group-id $VPCE_SG_ID \
  --protocol udp --port 514 --cidr 10.0.0.0/16 --region $REGION
```

> **Tip**
>
> 하나의 프로토콜만 사용하려는 경우(예: 포트 6514의 TCP+TLS) 보안 그룹에서 해당 포트만 열면 됩니다.

3단계: VPC 엔드포인트 생성

syslog AWS PrivateLink 서비스를 가리키는 인터페이스 VPC 엔드포인트를 생성합니다. 이렇게 하면 VPC에 CloudWatch Logs syslog 서비스에 대한 프라이빗 진입점이 제공됩니다.

> **Note**
>
> 엔드포인트를 생성할 때 여러 가용 영역에서 여러 서브넷 IDs를 지정할 수 있습니다. 엔드포인트는 각 서브넷에 ENI를 생성하여 가용 영역당 별도의 VPC 엔드포인트 없이 더 높은 가용성을 제공합니다. 여러 가용 영역에 서브넷이 있는 단일 VPC 엔드포인트면 충분합니다.

Console

VPC 엔드포인트를 생성하려면(콘솔)

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택한 다음 엔드포인트 생성을 선택합니다.
3. 이름 태그에 엔드포인트의 이름(예: `syslog-vpce`)을 입력합니다.
4. 서비스 범주(Service category)에서 AWS 서비스를 선택합니다.
5. 서비스 검색 필드에를 `syslog-logs` 입력하고 서비스를 선택합니다`com.amazonaws.Region.syslog-logs`.
6. VPC에서 1단계에서 생성하거나 식별한 VPC를 선택합니다.
7. 서브넷 섹션에서 하나 이상의 가용 영역을 선택하고 엔드포인트 네트워크 인터페이스를 생성할 서브넷을 선택합니다.
8. 보안 그룹에서 2단계(`syslog-vpce-sg`)에서 생성한 보안 그룹을 선택합니다.
9. (선택 사항) 엔드포인트를 통해 허용되는 트래픽을 제한하려면 VPC 엔드포인트 정책을 구성합니다. 자세한 내용은 the section called “VPC 엔드포인트 정책” 단원을 참조하십시오.
10. 엔드포인트 생성을 선택합니다.
11. 엔드포인트 상태가 사용 가능으로 변경되면 엔드포인트를 선택하고 DNS 이름 값을 기록해 둡니다. 이는 syslog 디바이스가 전송할 주소입니다.

AWS CLI

```
VPCE_ID=$(aws ec2 create-vpc-endpoint \
  --vpc-id $VPC_ID \
  --service-name com.amazonaws.${REGION}.syslog-logs \
  --vpc-endpoint-type Interface \
  --subnet-ids $SUBNET_ID \
  --security-group-ids $VPCE_SG_ID \
  --region $REGION \
```

```
  --query 'VpcEndpoint.VpcEndpointId' --output text)

echo "VPC Endpoint: $VPCE_ID"
```

엔드포인트를 사용할 수 있을 때까지 기다린 다음(약 60초) DNS 이름을 검색합니다.

```
VPCE_DNS=$(aws ec2 describe-vpc-endpoints --vpc-endpoint-ids $VPCE_ID \
  --region $REGION --query 'VpcEndpoints[0].DnsEntries[0].DnsName' --output text)

echo "Endpoint DNS: $VPCE_DNS"
```

VPCE_DNS 값 저장 -이 주소로 전송하도록 syslog 디바이스를 구성합니다.

4단계: 로그 그룹 생성

syslog 메시지가 전달될 CloudWatch Logs 로그 그룹을 생성합니다. 모든 로그 그룹 이름을 사용할 수 있습니다. 조직의 명확성을 위해 /syslog/ 접두사를 사용하는 것이 좋습니다.

Console

로그 그룹을 생성하려면(콘솔)

1. https://console.aws.amazon.com/cloudwatch/ CloudWatch Logs 콘솔을 엽니다.
2. 탐색 창의 로그 관리에서 로그 그룹을 선택합니다.
3. 로그 그룹 생성을 선택합니다.
4. 로그 그룹 이름에를 입력합니다/syslog/my-devices.
5. (선택 사항) 필요에 따라 보존 설정 및 암호화를 구성합니다.
6. 생성(Create)을 선택합니다.

AWS CLI

```
aws logs create-log-group \
  --log-group-name /syslog/my-devices \
  --region $REGION
```

로그 스트림을 생성할 필요가 없습니다. syslog 서비스는 첫 번째 메시지가 전달될 때 라는 로그 스트림*VPCE_ID*_Syslog_*Region*(예: vpce-0abc123def456_Syslog_us-east-1)을 자동으로 생성합니다.

5단계: 리소스 정책 추가

CloudWatch Logs syslog 서비스는 syslog.logs.amazonaws.com 서비스 보안 주체를 사용하여 로그 그룹에 씁니다. 로그 그룹의 리소스 정책을 통해 권한을 부여해야 합니다. 이 aws:SourceArn 조건은 특정 VPC 엔드포인트의 트래픽만이 로그 그룹에 쓸 수 있도록 합니다.

```
ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)

aws logs put-resource-policy \
  --policy-name syslog-ingestion \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "syslog.logs.amazonaws.com"
      },
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:'$REGION':'$ACCOUNT_ID':log-group:/syslog/my-
devices:*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "'$ACCOUNT_ID'"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:ec2:'$REGION':'$ACCOUNT_ID':vpc-
endpoint/'$VPCE_ID'"
        }
      }
    }
  ]
}' \
  --region $REGION
```

리소스 정책의 조건은 다음과 같은 보호를 제공합니다.

- `aws:SourceAccount` - 혼동된 대리자 공격을 방지합니다. 계정의 트래픽만 허용됩니다.
- `aws:SourceArn` - 특정 VPC 엔드포인트에 대한 액세스 범위를 지정합니다. VPC 엔드포인트가 여러 개 있는 경우 조건에 각 엔드포인트를 별도의 ARN으로 추가합니다.

여러 VPC 엔드포인트가 동일한 로그 그룹에 쓰도록 허용하려면 와일드카드와 함께 `ArnLike` 조건 연산자를 사용합니다.

```
"ArnLike": {
    "aws:SourceArn": "arn:aws:ec2:us-east-1:123456789012:vpc-endpoint/*"
}
```

6단계: syslog 구성 생성

이 단계는 CloudWatch Logs syslog 서비스에 VPC 엔드포인트에서 들어오는 트래픽을 로그 그룹으로 라우팅해야 한다고 알려줍니다. 이 구성이 없으면 엔드포인트의 트래픽이 거부됩니다.

Console

syslog 구성을 생성하려면(콘솔)

1. https://console.aws.amazon.com/cloudwatch/ CloudWatch Logs 콘솔을 엽니다.
2. 탐색 창의 로그 관리에서 로그 그룹을 선택합니다.
3. 4단계에서 생성한 로그 그룹(예: `/syslog/my-devices`)을 선택합니다.
4. 로그 그룹 세부 정보에서 Syslog 수집 섹션을 찾습니다.
5. 구성을 선택합니다.
6. VPC 엔드포인트 드롭다운에서 3단계에서 생성한 VPC 엔드포인트를 선택합니다.
7. 생성(Create)을 선택합니다.

AWS CLI

```
aws logs put-syslog-configuration \
  --log-group-identifier /syslog/my-devices \
  --vpc-endpoint-id $VPCE_ID \
  --region $REGION
```

구성을 확인합니다.

```
aws logs list-syslog-configurations \
  --log-group-identifier /syslog/my-devices \
  --region $REGION
```

이제 syslog 수집 파이프라인이 활성화되었습니다. VPC 엔드포인트로 전송된 모든 syslog 메시지는 `/syslog/my-devices` 로그 그룹으로 전달됩니다.

7단계: 전송 및 필드 추출 확인

VPC 엔드포인트에 도달할 수 있는 EC2 호스트 또는 디바이스에서 테스트 메시지를 보낸 다음 CloudWatch Log Analytics를 사용하여 메시지가 전송되었고 구조화된 필드가 올바르게 추출되었는지 확인합니다. 메시지는 일반적으로 10~20초 이내에 나타납니다.

테스트 메시지 전송(TCP 일반 텍스트):

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello from syslog" |
 \
  nc $VPCE_DNS 1514
```

전송 및 추출된 필드를 확인합니다.

Console

Log Analytics를 사용하여 전송을 확인하려면(콘솔)

1. https://console.aws.amazon.com/cloudwatch/ CloudWatch Logs 콘솔을 엽니다.
2. 탐색 창의 로그에서 로그 분석을 선택합니다.
3. 로그 그룹 선택기에서 로그 그룹(예: `/syslog/my-devices`)을 선택합니다.
4. 다음 쿼리를 입력하고 쿼리 실행을 선택합니다.

   ```
   fields @timestamp, facility, severity, hostname, appName, procId, message
   | sort @timestamp desc
   | limit 10
   ```

5. 테스트 메시지가 나타나고 추출된 필드가 올바르게 채워졌는지 확인합니다. 위의 테스트 메시지에는 `facility` = , = `local0`, `severity` = `info` `hostname` `myhost`, `appName` = `myapp`, `procId` =가 표시되어야 합니다`1234`.

AWS CLI

Log Analytics 쿼리를 시작하여 전송 및 필드 추출을 확인합니다.

```
QUERY_ID=$(aws logs start-query \
  --log-group-name /syslog/my-devices \
  --start-time $(date -d '5 minutes ago' +%s 2>/dev/null || echo $(date -v-5M +%s))
 \
  --end-time $(date +%s) \
  --query-string 'fields @timestamp, facility, severity, hostname, appName, procId,
 message | sort @timestamp desc | limit 10' \
  --region $REGION \
  --query 'queryId' --output text)

# Wait a few seconds for the query to complete, then retrieve results
aws logs get-query-results \
  --query-id $QUERY_ID \
  --region $REGION
```

테스트 메시지가 나타나고 추출된 필드가 올바르게 채워졌는지 확인합니다. 위의 테스트 메시지에는 `facility` = , = `local0`, `severity` = `info` `hostname` `myhost`, `appName` = `myapp`, `procId` =가 표시되어야 합니다`1234`.

syslog 디바이스 구성

syslog 수집을 설정한 후 VPC 엔드포인트 DNS 이름으로 메시지를 보내도록 syslog 소스를 구성합니다. 디바이스는 VPC에 대한 VPN 또는 Direct Connect 연결을 통해 엔드포인트에 도달합니다.

> **Note**
>
> 다음 구성이 예제로 제공됩니다. 전체 구성 옵션 및 모범 사례는 syslog 구현에 대한 공식 설명서(rsyslog, syslog-ng 또는 디바이스 공급업체)를 참조하세요.

다음 예제에서 *`VPCE_DNS`*를 설정 절차의 VPC 엔드포인트 DNS 이름으로 바꿉니다(예: `vpce-0abc123def456.syslog-logs.us-east-1.vpce.amazonaws.com`).

rsyslog

rsyslog는 대부분의 Linux 배포판에서 기본 syslog 데몬입니다. 에 다음 구성을 추가합니다`/etc/rsyslog.d/50-cwl-syslog.conf`.

TCP + TLS:

```
# Forward all logs to CloudWatch Logs via TCP+TLS
action(type="omfwd"
  target="VPCE_DNS"
  port="6514"
  protocol="tcp"
  StreamDriver="gtls"
  StreamDriverMode="1"
  StreamDriverAuthMode="x509/certvalid"
  template="RSYSLOG_SyslogProtocol23Format"
  queue.type="LinkedList"
  queue.filename="cwl_fwd"
  queue.saveOnShutdown="on"
  action.resumeRetryCount="-1")
```

TCP 일반 텍스트:

```
# Forward all logs to CloudWatch Logs via plaintext TCP
action(type="omfwd"
  target="VPCE_DNS"
  port="1514"
  protocol="tcp"
  template="RSYSLOG_SyslogProtocol23Format"
  queue.type="LinkedList"
  queue.filename="cwl_fwd"
  queue.saveOnShutdown="on"
  action.resumeRetryCount="-1")
```

UDP:

```
# Forward all logs to CloudWatch Logs via UDP
action(type="omfwd"
  target="VPCE_DNS"
  port="514"
  protocol="udp"
```

```
  template="RSYSLOG_SyslogProtocol23Format")
```

구성을 추가한 후 rsyslog를 다시 시작합니다.

```
sudo systemctl restart rsyslog
```

> **Note**
>
> queue.type="LinkedList" 및 queue.saveOnShutdown="on" 명령은 디스크 지원 대기열을 활성화합니다. VPC 엔드포인트에 대한 연결이 일시적으로 끊어지면 rsyslog는 메시지를 로컬로 버퍼링하고 전송을 재시도합니다.

시설 또는 심각도를 기준으로 필터링:

특정 메시지만 보내려면 작업 앞에 필터를 추가합니다.

```
# Send only auth messages via TCP+TLS
auth,authpriv.* action(type="omfwd"
  target="VPCE_DNS"
  port="6514"
  protocol="tcp"
  StreamDriver="gtls"
  StreamDriverMode="1"
  StreamDriverAuthMode="x509/certvalid"
  template="RSYSLOG_SyslogProtocol23Format")
```

syslog-ng

syslog-ng는 일반적으로 네트워크 어플라이언스 및 엔터프라이즈 Linux 시스템에서 사용됩니다. 에 다음 구성을 추가합니다/etc/syslog-ng/conf.d/cwl-syslog.conf.

TCP + TLS:

```
destination d_cwl_syslog {
  network("VPCE_DNS"
    port(6514)
    transport("tls")
    tls(
      peer-verify(required-trusted)
```

```
      ca-dir("/etc/ssl/certs")
    )
  );
};

log {
  source(s_sys);
  destination(d_cwl_syslog);
};
```

TCP 일반 텍스트:

```
destination d_cwl_syslog {
  network("VPCE_DNS"
    port(1514)
    transport("tcp")
  );
};

log {
  source(s_sys);
  destination(d_cwl_syslog);
};
```

UDP:

```
destination d_cwl_syslog {
  network("VPCE_DNS"
    port(514)
    transport("udp")
  );
};

log {
  source(s_sys);
  destination(d_cwl_syslog);
};
```

구성을 추가한 후 syslog-ng를 다시 시작합니다.

```
sudo systemctl restart syslog-ng
```

빠른 검증

VPC 엔드포인트에 연결할 수 있는 호스트에서 다음 명령을 사용하여 연결을 확인합니다.

TCP 일반 텍스트(포트 1514):

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello from syslog" |
 \
  nc VPCE_DNS 1514
```

TCP + TLS(포트 6514):

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello TLS syslog" | \
  openssl s_client -connect VPCE_DNS:6514 -quiet -no_ign_eof
```

UDP(포트 514):

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello UDP syslog" | \
  nc -u -w1 VPCE_DNS 514
```

테스트 메시지를 전송한 후 CloudWatch Logs에서 전송을 확인합니다(일반적으로 10~20초 이내).

```
aws logs filter-log-events \
  --log-group-name /syslog/my-devices \
  --start-time $(date -d '5 minutes ago' +%s000 2>/dev/null || echo $(($(date -v-5M +
%s) * 1000))) \
  --region $REGION
```

네트워크 어플라이언스

대부분의 방화벽, 라우터 및 스위치는 대상 IP 주소 및 포트를 지정하여 외부 syslog 서버 구성을 지원합니다. VPC 엔드포인트 ENI의 프라이빗 IP 주소(엔드포인트 아래의 Amazon VPC 콘솔에서 볼 수 있음)와 지원되는 포트 중 하나(6514, 1514 또는 514)를 사용합니다.

특정 구성 구문은 디바이스 설명서를 참조하세요. 구성할 키 설정은 다음과 같습니다.

- 서버 주소 - VPC 엔드포인트 DNS 이름 또는 ENI 프라이빗 IP 주소입니다.
- 포트 – 6514(TCP+TLS), 1514(TCP) 또는 514(UDP).
- 프로토콜 - TCP(권장) 또는 UDP.
- 형식 - RFC 5424(기본 설정) 또는 RFC 3164(레거시). 둘 다 지원됩니다.

TLS 인증서 신뢰

포트 6514의 TLS 연결은 Amazon Trust Services에서 발급한 AWS관리형 인증서를 사용합니다. Amazon Trust Services 루트 인증서는 표준 CA 신뢰 저장소에 포함되므로 대부분의 운영 체제 및 syslog 데몬은이 인증서를 자동으로 신뢰합니다.

디바이스가 기본적으로 인증서를 신뢰하지 않는 경우 https://www.amazontrust.com/repository/ Amazon Trust Services 루트 인증서를 다운로드하여 디바이스의 CA 트러스트 스토어에 추가합니다.

syslog 수집 모니터링

syslog 수집 서비스는 AWS/Logs 네임스페이스의 CloudWatch에 지표를 게시합니다. 이러한 지표를 통해 수신된 항목, 삭제된 항목, 이유 등 syslog 수집 파이프라인에 대한 가시성을 확보할 수 있습니다.

CloudWatch 콘솔에서 이러한 지표를 보려면 지표 > 모든 지표 > AWS/Logs로 이동하여 아래 나열된 지표 이름을 기준으로 필터링합니다.

Syslog 지표

지표	측정 기준	설명
SyslogMessagesReceived	LogGroupName	로그 그룹에 성공적으로 수집된 syslog 메시지 수입니다.
SyslogMessagesDropped	LogGroupName , Reason	배달할 수 없는 syslog 메시지 수입니다. 세부 정보는 the section called “삭제 이유” 섹션을 참조하세요.
SyslogConnectionsRejected	Reason	거부된 TCP 연결 수입니다.
SyslogConnectionsEstablished	—	성공적으로 수락된 TCP 연결 수입니다.
SyslogConnectionsClosed	—	닫힌 TCP 연결 수입니다.

삭제 이유

지표에는 메시지가 삭제된 이유를 나타내는 SyslogMessagesDropped Reason차원이 포함됩니다.

이유	설명
MessageRateLimitExceeded	계정의 할당PutLogEvents 량을 초과했습니다. 할당량 증가를 요청하는 것이 좋습니다.
MessageSizeExceeded	UDP 데이터그램이 최대 메시지 크기를 초과했습니다.
ServiceUnavailable	내부 용량 또는 속도 제한을 초과했습니다. 이는 일반적으로 일시적입니다.
ResourceNotFound	대상 로그 그룹이 존재하지 않습니다. 로그 그룹이 삭제되지 않았는지 확인합니다.
AccessDenied	로그 그룹의 리소스 정책은 syslog 서비스에 대한 액세스 권한을 부여하지 않습니다. 리소스 정책이 올바른지 확인합니다.
VpcePolicyDenied	VPC 엔드포인트 정책이 요청을 거부했습니다. VPC 엔드포인트 정책을 검토합니다.
InternalError	예기치 않은 내부 오류가 발생했습니다. 이 문제가 지속되면 AWS Support에 문의하세요.

연결 거부 이유

지표에는 SyslogConnectionsRejected Reason차원이 포함됩니다.

이유	설명
VpcePolicyDenied	VPC 엔드포인트 정책이 연결을 거부했습니다.
ServiceUnavailable	내부 오류로 인해 연결이 끊어졌습니다.

권장되는 경보

문제를 조기에 감지하려면 다음 조건에 대한 CloudWatch 경보를 생성하는 것이 좋습니다.

경보	조건	권장 조치
삭제 중인 메시지	`SyslogMessagesDropped` 5분 동안 > 0	`Reason` 차원을 조사하여 원인을 확인합니다.
액세스 거부됨	`SyslogMessagesDropped` 가 있는 `Reason=AccessDenied`	로그 그룹의 리소스 정책이 올바르게 구성되었는지 확인합니다.
로그 그룹 누락	`SyslogMessagesDropped` 가 있는 `Reason=ResourceNotFound`	로그 그룹이 존재하고 삭제되지 않았는지 확인합니다.
수신된 메시지 없음	`SyslogMessagesReceived` 15분 동안 = 0(일반적으로 > 0인 경우)	디바이스가 아직 전송 중이고 VPC 엔드포인트에 대한 네트워크 연결이 손상되지 않았는지 확인합니다.

CloudWatch 경보 생성에 대한 자세한 내용은 CloudWatch 사용 설명서의 CloudWatch 경보 생성을 참조하세요. CloudWatch

자동 대시보드에서 syslog 지표 보기

CloudWatch는 Syslog Ingestion 섹션을 포함하는 CloudWatch Logs에 대한 자동 대시보드를 제공합니다. 이 대시보드는 계정이 syslog 지표를 게시할 때 자동으로 표시되므로 수동 설정이 필요하지 않습니다. 대시보드에는에 설명된 각 지표에 대해 하나의 차트가 포함되어 있습니다the section called “Syslog 지표”. 각 차트에는 로그 그룹 또는 이유와 같은 관련 차원별로 분류된 지표 비율이 표시됩니다.

대시보드 액세스

syslog 자동 대시보드를 보려면:

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 탐색 창에서 대시보드를 선택한 다음 자동 대시보드를 선택합니다.
3. CloudWatch Logs 대시보드를 선택합니다.
4. Syslog 수집 섹션으로 스크롤합니다.

syslog 데이터에 대한 샘플 쿼리

다음 CloudWatch Logs Insights 쿼리는 syslog 데이터에 대한 일반적인 분석 패턴을 보여줍니다. CloudWatch Logs가 RFC 5424, RFC 3164 및 Cisco FTD/ASA 메시지에서 자동으로 추출하는 구조화된 필드를 사용합니다. 추출된 필드의 전체 목록은 섹션을 참조하세요the section called “지원되는 syslog 형식”.

이러한 쿼리는 CloudWatch 콘솔에서도 사용할 수 있습니다. CloudWatch Logs Insights에서 쿼리를 선택한 다음 Syslog 범주를 확장합니다.

보안 및 액세스 모니터링

Sudo 및 권한 에스컬레이션 이벤트

다음 쿼리는 Linux 호스트에 대한 최근 sudo 권한 부여 활동을 나열합니다. sudo 애플리케이션 이름을 필터링하고 RFC 3164 및 RFC 5424 메시지 모두에서 작동합니다.

```
filter appName="sudo"
| fields @timestamp, hostname, message
| sort @timestamp desc
| limit 50
```

SSH 인증이 실패한 상위 50개 호스트

다음 쿼리는 각 호스트에 대해 실패한 SSH 인증 시도 횟수를 계산합니다. "Failed password" 또는 "Failed publickey"가 포함된 sshd 메시지와 일치합니다. 이를 사용하여 무차별 대입 공격의 대상인 호스트를 식별합니다.

```
filter appName="sshd" and (message like /Failed password/ or message like /Failed
 publickey/)
| stats count(*) as failedAttempts by hostname
| sort failedAttempts desc
| limit 50
```

운영 가시성

시간 경과에 따른 호스트별 로그 볼륨

다음 쿼리는 각 호스트의 메시지를 5분 간격으로 계산합니다. 이를 사용하여 플릿 활동을 측정하거나, 볼륨 스파이크를 식별하거나, 전송을 중지한 호스트를 감지할 수 있습니다. 이 쿼리는 모든 syslog 형식 유형에서 작동합니다.

```
stats count(*) as messages by hostname, bin(5m)
| sort messages desc
```

시설 및 심각도별 메시지 볼륨

다음 쿼리는 시설 및 심각도 수준별로 그룹화된 메시지를 계산합니다. 이를 사용하여 가장 많은 로그 트래픽을 생성하는 하위 시스템과 심각도 수준을 파악할 수 있습니다.

```
stats count(*) as messageCount by facility, severity
| sort messageCount desc
```

애플리케이션별 심각도가 높은 메시지

다음 쿼리는 우선 순위가 높은 메시지(유행, 알림, 위반 및 오류에 해당하는 severityCode 0~3)를 필터링하고 애플리케이션별로 계산합니다. 이를 사용하여 가장 많은 오류를 생성하는 애플리케이션을 식별할 수 있습니다.

```
filter severityCode<=3
| stats count(*) as errorCount by appName
| sort errorCount desc
```

시간 경과에 따른 Warning-and-above 추세

다음 쿼리는 경고 심각도 이상(severityCode 0~4)의 메시지를 5분 간격으로 계산합니다. Logs Insights의 시각화 기능과 함께 사용하여 시간 경과에 따른 오류 급증을 차트로 표시할 수 있습니다.

```
filter severityCode<=4
| stats count(*) as events by bin(5m)
| sort @timestamp
```

방화벽 및 네트워크 디바이스 쿼리

Cisco에서 연결을 거부함(ACL 106023)

다음 쿼리는 Cisco ASA 및 FTD 방화벽용입니다. 디바이스별로 그룹화된 ACL 거부 연결(Cisco 메시지 ID 106023) 수를 계산하므로 트래픽이 가장 거부된 디바이스를 식별하는 데 도움이 됩니다.

```
filter messageId="106023"
| stats count(*) as denies by deviceId
| sort denies desc
| limit 20
```

메시지 ID별 Cisco ASA/FTD 이벤트

다음 쿼리는 Cisco FTD 및 ASA 방화벽용입니다. Cisco 메시지 ID 및 심각도별로 그룹화된 이벤트를 계산하여 방화벽 플릿에서 가장 빈번한 이벤트 유형을 식별하는 데 도움이 됩니다.

```
filter ispresent(messageId)
| stats count(*) as eventCount by messageId, severity
| sort eventCount desc
```

소스 IP별로 삭제된 패킷 방화벽

다음 쿼리는 방화벽 DROP 메시지(예: iptables 또는 nftables)를 구문 분석하여 삭제된 패킷을 소스 IP 주소별로 계산합니다. 이를 사용하여 차단된 트래픽의 상위 소스를 식별합니다.

```
filter message like /DROP/
| parse message /SRC=(?<srcAddr>\S+)/
| stats count(*) as drops by srcAddr
| sort drops desc
| limit 20
```

syslog에 대한 VPC 엔드포인트 정책

syslog VPC 엔드포인트에 VPC 엔드포인트 정책을 연결하여 이를 통해 허용되는 트래픽을 제어할 수 있습니다. 이를 통해 보안 그룹 및 리소스 정책 외에도 추가 액세스 제어 계층이 제공됩니다.

기본적으로 VPC 엔드포인트에는 모든 트래픽을 허용하는 전체 액세스 정책이 있습니다. 사용자 지정 정책을 연결하여 이를 제한할 수 있습니다. VPC 엔드포인트 정책에 대한 자세한 내용은 AWS PrivateLink 가이드의 엔드포인트 정책을 사용하여 액세스 제어를 참조하세요.

> **Important**
>
> Syslog 트래픽은 익명 보안 주체(전선에 IAM 자격 증명 없음)로 도착합니다. 정책은 작업, 리소스 또는 VPC 조건 키를 사용하여 액세스 범위를 `"Principal": "*"` 지정해야 합니다. 보안 주체 범위의 정책(예: 특정 IAM 역할만 허용하는 정책)은 모든 syslog 트래픽을 거부합니다.

지원되는 조건 키

syslog에 대한 VPC 엔드포인트 정책에서 지원되는 조건 키는 다음과 같습니다.

조건 키	설명	예시 값
`aws:SourceVpc`	트래픽이 시작되는 VPC입니다.	`vpc-111bbb22`
`aws:SourceVpce`	트래픽이 도착하는 VPC 엔드포인트입니다.	`vpce-0abc123def456`
`aws:VpcSourceIp`	VPC 내 발신자의 프라이빗 IP 주소입니다.	`10.0.1.50`
`aws:ResourceAccount`	대상 로그 그룹을 소유한 계정입니다.	`123456789012`
`aws:ResourceOrgID`	대상 로그 그룹을 소유한 계정의 조직 ID입니다.	`o-aa111bb222`
`aws:ResourceOrgPaths`	대상 로그 그룹을 소유한 계정의 조직 단위 경로입니다.	`o-aa111bb222/r-ab12/ou-ab12-11111111/*`

syslog 트래픽에는 IAM 보안 주체 자격 증명이 없으므로 다음 조건 키는 지원되지 않습니다.

- `aws:PrincipalArn`
- `aws:PrincipalAccount`
- `aws:PrincipalOrgID`
- `aws:PrincipalOrgPaths`

- aws:PrincipalTag/*

지원되지 않는 조건 키를 사용하는 정책은 암시적 거부를 발생시켜 엔드포인트를 통한 모든 syslog 트래픽을 차단합니다.

특정 로그 그룹으로 제한

다음 정책은 특정 로그 그룹에 대한 syslog 트래픽만 허용합니다.

```
aws ec2 modify-vpc-endpoint \
  --vpc-endpoint-id $VPCE_ID \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:Region:account-id:log-group:/syslog/my-devices"
    }
  ]
}' \
  --region $REGION
```

모든 트래픽 거부

다음 정책은 엔드포인트를 통한 모든 syslog 트래픽을 차단합니다. 이는 엔드포인트를 삭제하지 않고 수집을 일시적으로 중지하는 데 유용합니다.

```
aws ec2 modify-vpc-endpoint \
  --vpc-endpoint-id $VPCE_ID \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "*",
      "Resource": "*"
    }
  ]
```

```
}' \
  --region $REGION
```

기본 정책으로 재설정

기본 allow-all 정책을 복원하려면:

```
aws ec2 modify-vpc-endpoint \
  --vpc-endpoint-id $VPCE_ID \
  --reset-policy \
  --region $REGION
```

정책 전파

VPC 엔드포인트 정책을 생성하거나 수정한 후 정책 변경이 적용되는 데 최대 몇 분 정도 걸릴 수 있습니다. 이 전파 기간 동안 이전 정책(또는 기본 정책)은 계속 적용됩니다.

VPC 엔드포인트 정책이 syslog 트래픽을 거부하면 다음 지표가 표시됩니다.

- TCP 연결은 정책 평가 직후 닫힙니다.
- UDP 데이터그램이 삭제됩니다.
- 의 `SyslogMessagesDropped` 지표`Reason=VpcePolicyDenied`가 내보내집니다.
- 의 `SyslogConnectionsRejected` 지표`Reason=VpcePolicyDenied`는 TCP 연결에 대해 내보내집니다.

syslog 수집 문제 해결

이 섹션에서는 syslog 수집과 관련된 일반적인 문제와 해결 방법을 설명합니다.

로그 그룹에 표시되지 않는 메시지

syslog 메시지를 전송했지만 로그 그룹에 표시되지 않는 경우 다음을 순서대로 확인합니다.

1. syslog 구성이 존재하는지 확인합니다.

   를 실행`list-syslog-configurations`하여 VPC 엔드포인트가 로그 그룹에 등록되어 있는지 확인합니다.

   ```
   aws logs list-syslog-configurations \
   ```

```
  --log-group-identifier /syslog/my-devices \
  --region $REGION
```

구성이 반환되지 않으면를 사용하여 구성을 생성합니다`put-syslog-configuration`.

2. 리소스 정책을 확인합니다.

   로그 그룹의 리소스 정책이 `logs:PutLogEvents` 및 `logs:CreateLogStream`를에 부여`syslog.logs.amazonaws.com`하고 `aws:SourceArn` 조건이 VPC 엔드포인트 ARN과 정확히 일치하는지 확인합니다.

```
aws logs describe-resource-policies --region $REGION
```

   를 `Reason=AccessDenied` 사용하여 `SyslogMessagesDropped` 지표를 확인하여 권한 부여가 문제인지 확인합니다.

3. 보안 그룹을 확인합니다.

   VPC 엔드포인트에 연결된 보안 그룹이 소스 CIDR에서 사용 중인 포트(6514, 1514 또는 514)의 인바운드 트래픽을 허용하는지 확인합니다.

```
aws ec2 describe-security-groups \
  --group-ids $VPCE_SG_ID \
  --region $REGION \
  --query 'SecurityGroups[0].IpPermissions'
```

4. 네트워크 연결을 확인합니다.

   동일한 네트워크의 디바이스 또는 호스트에서 엔드포인트에 대한 TCP 연결을 테스트합니다.

```
# TCP connectivity test
nc -zv $VPCE_DNS 6514
nc -zv $VPCE_DNS 1514
```

   연결이 거부되거나 시간이 초과되면 VPN/Direct Connect 라우팅을 확인하고 디바이스가 VPC 엔드포인트 서브넷에 도달할 수 있는지 확인합니다.

5. VPC 엔드포인트 정책을 확인합니다.

   사용자 지정 VPC 엔드포인트 정책을 연결한 경우 필요한 작업을 허용하는지 확인합니다. 를 사용하여 `SyslogMessagesDropped` 지표를 확인합니다`Reason=VpcePolicyDenied`.

TLS 핸드셰이크 실패

syslog 클라이언트가 포트 6514에서 TLS 연결을 설정하지 못하는 경우:

- 인증서 신뢰 - 디바이스의 CA 신뢰 스토어에 Amazon Trust Services 루트 인증서가 포함되어 있는지 확인합니다. 대부분의 운영 체제에는 기본적으로 이러한 운영 체제가 포함됩니다. 필요한 경우 https://www.amazontrust.com/repository/ 다운로드합니다.
- TLS 버전 - 서비스는 TLS 1.2 및 TLS 1.3을 지원합니다. syslog 클라이언트가 TLS 1.2 이상을 지원하는지 확인합니다.
- 호스트 이름 확인 - 클라이언트가 서버 호스트 이름을 확인하는 경우 VPC 엔드포인트 DNS 이름(IP 주소 아님)을 사용하여 연결하도록 구성되었는지 확인합니다.

OpenSSL을 사용하여 TLS 연결을 수동으로 테스트합니다.

```
openssl s_client -connect $VPCE_DNS:6514 -brief
```

연결에 성공하면 TLS 버전과 암호 제품군이 표시됩니다. 실패하면 오류 메시지에 원인이 표시됩니다.

UDP 메시지 손실

UDP는 최선의 프로토콜입니다. 일부 메시지 손실은 다음 조건에서 예상되는 동작입니다.

- 디바이스와 VPC 엔드포인트 간의 네트워크 정체.
- 메시지가 최대 UDP 데이터그램 크기를 초과합니다.
- 임시 서비스 용량 제약 조건(와 `SyslogMessagesDropped` 함께 로 표시 됨`Reason=ServiceUnavailable`).

안정적인 전송이 필요한 경우 TCP(포트 1514 또는 6514)를 대신 사용합니다. TCP는 전송 확인 및 자동 재전송을 제공합니다.

연결이 거부됨

디바이스가 VPC 엔드포인트에 연결할 때 "연결 거부"를 보고하는 경우:

- VPC 엔드포인트가 `available` 상태인지 확인합니다.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids $VPCE_ID \
```

```
  --region $REGION --query 'VpcEndpoints[0].State'
```

- 지원되는 포트(6514, 1514 또는 514)에서 연결 중인지 확인합니다.
- VPN 또는 Direct Connect 라우팅이 디바이스 네트워크에서 VPC 엔드포인트 서브넷으로의 트래픽을 허용하는지 확인합니다.
- 보안 그룹이 사용 중인 포트에서 인바운드 트래픽을 허용하는지 확인합니다.

속도 제한으로 인해 삭제된 메시지

`SyslogMessagesDropped` 지표에가 표시되면 `Reason=MessageRateLimitExceeded`syslog 트래픽이 계정의 할당`PutLogEvents`량을 초과하는 것입니다.

이 문제를 해결하려면:

1. Service Quotas 콘솔의 Amazon CloudWatch Logs에서 현재 할당량을 확인합니다.
2. syslog 처리량에 필요한 경우 할당량 증가를 요청합니다. Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

AWS SDK에서 CloudWatch Logs 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 많은 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예제
AWS SDK for Python v4	AWS SDK for Python v4 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

CloudWatch Logs에 관련된 예제는 SDK를 사용한 CloudWatch Logs의 코드 예제 AWS SDKs을 참조하세요.

> **예제 가용성**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

로그 관리

CloudWatch Logs는 로그 데이터를 보다 효과적으로 구성, 변환 및 분석하는 데 도움이 되는 고급 로그 관리 기능을 제공합니다. 이러한 기능에는 교차 계정 및 교차 리전 데이터 중앙 집중화, 자동 데이터 검색 및 스키마 관리, 수집 중 로그 변환, 대화형 로그 탐색을 위한 패싯을 사용한 향상된 분석이 포함됩니다.

주제

- 데이터 소스 검색 및 관리
- 데이터 소스에서 활성화된 기능
- 지원되는 데이터 원본

데이터 소스 검색 및 관리

CloudWatch Logs는 데이터 소스 및 유형별로 로그 데이터를 자동으로 검색하고 분류하므로 로그를 대규모로 더 쉽게 이해하고 관리할 수 있습니다. 이 기능은 Amazon VPC 흐름 로그, CloudTrail, Route 53과 같은 AWS 판매 소스와 타사 보안 도구에 대한 스키마 검색을 제공합니다.

로그 관리 콘솔은 로그 그룹만 제공하는 것이 아니라 데이터 소스 및 유형별로 구성된 로그를 개괄적으로 보여줍니다. 이 조직은 다음을 지원합니다.

- AWS 서비스, 타사 소스(예: Okta 또는 CrowdStrike) 및 사용자 지정 소스별로 분류된 로그 보기
- 로그 데이터의 스키마 및 구조를 자동으로 이해
- 검색된 스키마 필드를 기반으로 필드 인덱스 정책 생성
- 다양한 데이터 소스에서 로그를 보다 효율적으로 관리
- 다양한 데이터 소스로 로그 쿼리

지원되는 AWS 서비스에 대해 CloudWatch Logs 로깅을 활성화하면 CloudWatch Logs는 로그에 적절한 스키마를 자동으로 적용합니다. 이 자동 스키마 애플리케이션은 일관성을 유지하고 로그 구조에 대한 즉각적인 인사이트를 제공하는 데 도움이 됩니다.

CloudWatch Logs 데이터 소스란 무엇입니까?

CloudWatch Logs 데이터 소스는 로그를 생성하는 소스를 기반으로 로그 데이터를 구성하고 분류하는 새로운 방법을 제공하는 기능입니다. CloudWatch Logs는 일반적으로 로그 그룹을 사용하여 로그를 구

성하지만 데이터 소스는 원래 서비스 및 로그 유형별로 로그를 그룹화하는 추가 조직 계층을 제공합니다.

데이터 소스 작동 방식

데이터 소스는 AWS 인프라 전반에서 서비스 기반 로그 구성과 간소화된 검색을 제공합니다. 개별 로그 그룹 이름이나 구조를 알 필요 없이 특정 서비스의 로그를 쉽게 찾고 로그 유형별로 필터링할 수 있습니다.

타사 소스 및 선택적으로 애플리케이션 로그 소스의 경우 데이터 소스는 CloudWatch 파이프라인과 함께 작동하여 로그를 분류합니다. 로그를 수집하고 변환하도록 파이프라인을 구성할 때 데이터 소스 이름과 유형을 지정합니다. 그런 다음 CloudWatch Logs는 파이프라인이 처리하는 모든 로그를 자동으로 분류합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 파이프라인을 참조하세요. Amazon CloudWatch

데이터 소스는 두 가지 키 식별자를 사용하여 로그를 분류합니다.

- 데이터 소스 이름: 로그를 생성하는 AWS 서비스, 타사 소스 또는 애플리케이션(예: Route 53, Amazon VPC, CloudTrail, Okta SSO 또는 CrowdStrike Falcon).
- 데이터 소스 유형: 해당 서비스에서 생성된 특정 유형의 로그입니다.

스키마는 존재하는 필드와 정보 구성 방식을 포함하여 로그 데이터의 구조를 정의합니다. 단일 데이터 소스는 스키마와 용도가 다른 여러 유형의 로그를 생성할 수 있습니다. 예를 들어 AWS CloudTrail 데이터 소스에는 관리 이벤트(리소스 생성 또는 삭제와 같은 컨트롤 플레인 작업 추적)와 데이터 이벤트(S3 객체 액세스와 같은 데이터 영역 작업 추적)의 두 가지 유형이 있습니다. 각 유형은 다양한 종류의 정보를 캡처하기 때문에 스키마가 다릅니다.

시작하는 방법

CloudWatch Logs는 로그를 오리진에 따라 데이터 소스로 분류합니다. 메서드는 작업 중인 로그 유형에 따라 달라집니다.

AWS 서비스 로그

지원되는 AWS 서비스의 로그는 구성 없이 데이터 소스별로 자동으로 그룹화됩니다. CloudWatch Logs는 이러한 로그를 인식하고 원본 서비스에 따라 적절한 데이터 소스 이름 및 유형을 적용합니다.

타사 로그

타사 로그에는 데이터 소스 분류를 위한 파이프라인이 필요합니다. Microsoft Office 365, Okta, CrowdStrike 또는 Palo Alto Networks와 같은 지원되는 타사 소스에서 로그를 수집하도록 파이프라인을 구성할 때 파이프라인 구성에 데이터 소스 이름과 유형을 지정합니다. CloudWatch Logs는 파이프라인이 해당 식별자를 사용하여 처리하는 모든 로그를 자동으로 분류합니다.

파이프라인은 표준화된 보안 이벤트 분석을 위해 선택적으로 타사 로그를 OCSF(Open Cybersecurity Schema Framework) 형식으로 변환할 수 있습니다. OCSF 변환이 활성화되면 OCSF 스키마 매핑에 따라 데이터 소스 이름과 유형이 자동으로 결정됩니다. OCSF 변환이 없으면 파이프라인 구성에서 데이터 소스 이름과 유형을 지정합니다.

애플리케이션 로그

사용자 지정 애플리케이션 로그의 경우 다음 방법 중 하나를 사용하여 데이터 소스별로 분류할 수 있습니다.

- 로그 그룹 태그 - `cw:datasource:name` 및 키를 사용하여 로그 그룹에 태그를 추가하여 로그 그룹에 수집된 모든 로그에 대한 데이터 소스 이름과 유형을 각각 `cw:datasource:type` 지정합니다. 태그 값은 최대 64자일 수 있으며 소문자, 숫자 및 밑줄만 포함할 수 있습니다. 문자 또는 숫자로 시작해야 하며 이중 밑줄(__)을 포함할 수 없습니다.
- 파이프라인 구성 - 애플리케이션 로그를 수집할 때 로그 처리 파이프라인을 통해 데이터 소스 정보를 구성합니다.

> **Note**
>
> AWS 서비스 로그와의 충돌을 방지하기 위해 데이터 소스 이름은 "aws" 또는 "amazon"으로 시작할 수 없습니다.

시스템 필드

CloudWatch Logs는 데이터 소스별로 분류된 로그에 3개의 시스템 필드를 자동으로 추가합니다. 이러한 필드는 기본 패싯 역할을 합니다.

- `@data_source_name` - 데이터 소스의 이름을 포함하거나 확인되지 않은 경우 "알 수 없음"
- `@data_source_type` - 데이터 소스의 유형을 포함하거나 확인되지 않은 경우 "알 수 없음"
- `@data_format` - 로그 데이터의 형식을 나타냅니다.

데이터 소스 이름 또는 유형을 확인할 수 없는 경우 이러한 필드는 "알 수 없음"으로 설정됩니다. "알 수 없음" 값이 있는 데이터 소스는 여전히 콘솔의 "로그 관리" 아래의 패싯과 데이터 소스 테이블에 표시되므로 분류되지 않은 로그와 로그 그룹의 출처를 식별할 수 있습니다.

`@data_format` 필드에는 다음 값 중 하나가 포함될 수 있습니다.

- `Default` - 수정 없이 수집된 로그입니다.
- `Custom` - 파이프라인 프로세서를 통해 처리된 로그 또는 데이터 소스 이름/유형 태그를 사용하여 로그 그룹에 수집된 로그입니다.
- `OCSF-<version>` - 파이프라인에서 OCSF(Open Cybersecurity Schema Framework) 프로세서로 처리된 로그입니다.
- `AWS-OTEL-LOG-V<version>` - CloudWatch OTLP 엔드포인트를 통해 수집된 OpenTelemetry 로그입니다.
- `AWS-OTEL-TRACE-V<version>` - CloudWatch OTLP 엔드포인트를 통해 수집된 OpenTelemetry 트레이스입니다.

이러한 시스템 필드를 사용하면 소스 및 형식을 기반으로 로그를 필터링하고 쿼리할 수 있으므로 다양한 오리진 및 처리 파이프라인의 로그로 더 쉽게 작업할 수 있습니다.

데이터 원본 액세스

콘솔

CloudWatch Logs 콘솔에서 로그 관리 탭을 사용하여 데이터 소스에 액세스합니다. CloudWatch Logs는 데이터 소스 및 유형별로 로그 데이터를 자동으로 통합하여 새로 수집된 데이터를 지속적으로 검색합니다. 데이터 소스 목록에서 파이프라인을 생성하고 필드 인덱스와 패싯을 정의할 수 있습니다.

AWS CLI

다음 명령을 사용하여 계정의 고유한 데이터 소스 및 로그 유형을 나열합니다.

```
aws logs list-aggregate-log-group-summaries --group-by DATA_SOURCE_NAME_AND_TYPE
```

로그 그룹과의 관계

데이터 소스는 로그 그룹을 대체하는 대신 보완됩니다. 로그는 이전과 같이 로그 그룹에 계속 저장되지만, 이제는 데이터 소스 정보로도 자동으로 태그가 지정됩니다. 이 이중 조직을 사용하면 다음을 수행할 수 있습니다.

- 세분화된 액세스 제어 및 보존 정책에 로그 그룹 사용
- 서비스 기반 로그 검색 및 분석에 데이터 소스 사용
- 필요에 따라 두 조직 방법을 사용하여 로그 쿼리

데이터 소스를 사용하면 AWS 인프라 전반의 로그 데이터에 대한 서비스 중심 보기를 제공하여 대규모 로그 작업을 더 쉽게 수행할 수 있습니다.

데이터 소스에서 활성화된 기능

데이터 소스를 사용하면 필드 검색 및 일관된 데이터 구조를 통해 고급 로그 처리 및 분석 기능을 사용할 수 있습니다.

- 패싯: 패싯은 쿼리를 작성하지 않고도 대화형 필터링 및 분석을 제공하는 인덱싱된 로그 필드입니다. CloudWatch Logs는 데이터 소스 이름 및 유형에 대한 패싯을 자동으로 생성하며, 검색된 필드에 대한 패싯 정책을 생성하여 문제 해결을 가속화할 수 있습니다. 패싯은 CloudWatch Logs Insights에 값 분포와 개수를 표시하므로 point-and-choose 지점 탐색을 통해 패턴을 쉽게 식별할 수 있습니다.
- 파이프라인: 특정 데이터 소스 이름 및 유형의 모든 로그에 적용되는 변환 파이프라인을 생성합니다. 이렇게 하면 동일한 소스의 로그에 대한 일관된 처리 규칙을 정의할 수 있습니다.
- 필드 검색: CloudWatch Logs는 파이프라인 프로세서를 기반으로 각 데이터 소스 이름 및 유형 조합에 대한 필드와 해당 데이터 유형을 자동으로 검색합니다. AWS 관리형 로그의 경우 필드 구조가 사전 정의되어 있습니다. 애플리케이션 로그의 경우 일관된 로그 형식을 유지하여 잘 정의된 필드 구조가 필요한 Amazon S3 테이블과 같은 분석 도구와의 호환성을 극대화하는 것이 좋습니다.

`GetLogFields` API를 사용하여 모든 데이터 소스의 전체 필드 및 유형 목록을 볼 수 있습니다.

```
aws logs get-log-fields --data-source-name <name>  --data-source-type <type>
```

외부 도구가 로그 데이터를 처리할 때 예측 가능한 필드 구조를 사용할 수 있으므로이 필드 검색 및 일관성을 통해 고급 분석 및 통합이 가능합니다.

지원되는 데이터 원본

CloudWatch Logs는 AWS 서비스, 타사 보안 및 IT 도구, 사용자 지정 애플리케이션 소스를 비롯한 다양한 데이터 소스를 지원합니다. 이 페이지에서는 지원되는 모든 데이터 소스의 포괄적인 카탈로그를 제공합니다. 데이터 소스 검색 및 관리 방법에 대한 자세한 내용은 섹션을 참조하세요the section called “데이터 소스 검색 및 관리”.

주제

- 데이터 소스 AWS 서비스 에 지원됨
- 데이터 소스에 지원되는 타사 소스
- 사용자 지정 소스

데이터 소스 AWS 서비스 에 지원됨

다음 표에는 CloudWatch Logs에 의해 데이터 소스로 자동으로 분류 AWS 서비스 되는가 나열되어 있습니다.

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
amazon_api_gateway	access
amazon_bedrock_agentcore	browser_usage
amazon_bedrock_agentcore	code_interpreter_application
amazon_bedrock_agentcore	code_interpreter_usage
amazon_bedrock_agentcore	gateway_application
amazon_bedrock_agentcore	identity_workload_application
amazon_bedrock_agentcore	memory_application
amazon_bedrock_agentcore	online_evaluation_config
amazon_bedrock_agentcore	runtime_application
amazon_bedrock_agentcore	runtime_usage
amazon_bedrock_agents	application
amazon_bedrock_agents	event
amazon_bedrock_knowledge_bases	application
amazon_cloudfront	access

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
amazon_cloudfront	connection
amazon_cloudwatch	rum_app_monitor
amazon_cognito	user_pool
amazon_ec2	verified_access
amazon_eks	api_server
amazon_eks	audit
amazon_eks	authenticator
amazon_eks	controller_manager
amazon_eks	scheduler
amazon_elasticache	cluster
amazon_eventbridge	eventbus_error
amazon_eventbridge	eventbus_info
amazon_eventbridge	pipes_execution
amazon_interactive_video_service	chat
amazon_managed_prometheus	scraper
amazon_managed_prometheus	workspace
amazon_msk	broker
amazon_msk	connect
amazon_opensearch_service	pipeline
amazon_q_business	events

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
amazon_q_business	sync_job
amazon_q_connect	events
amazon_route53	global_resolver_query
amazon_route53	hosted_zones
amazon_route53	profiles_resolver_query
amazon_route53	resolver_query
amazon_sagemaker	workteam_activity
amazon_ses	ingress_endpoints
amazon_ses	rule_sets
amazon_ses	traffic_policy
amazon_vpc	flow
amazon_vpc	route_server_peer
amazon_vpc_lattice	access
amazon_vpc_lattice	resource_access
amazon_workmail	access_control
amazon_workmail	authentication
amazon_workmail	personal_access
amazon_workmail	workmail_access
amazon_workmail	workmail_availability
aws_b2b_data_interchange	execution

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
aws_backup	data_access
aws_backup	hypervisor
aws_clean_rooms	analysis
aws_client_vpn	connection
aws_client_vpn	event
aws_cloudtrail	data
aws_cloudtrail	management
aws_elemental_mediapackage	egress_access
aws_elemental_mediapackage	ingress_access
aws_elemental_mediatailor	ad_decision
aws_elemental_mediatailor	manifest
aws_elemental_mediatailor	transcode
aws_entity_resolution	id_mapping_workflow
aws_entity_resolution	matching_workflow
aws_iot_fleetwise	error
aws_mainframe_modernization	batch_job
aws_mainframe_modernization	config
aws_mainframe_modernization	console
aws_mainframe_modernization	dataset_import
aws_network_firewall	alert

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
aws_network_firewall	flow
aws_network_firewall	tls
aws_nlb	access
aws_pcs	job_completion
aws_pcs	scheduler
aws_security_hub	compliance_finding
aws_security_hub	data_security_finding
aws_security_hub	detection_finding
aws_security_hub	vulnerability_finding
aws_security_hub_cspm	asff_finding
aws_shield	protection_flow
aws_step_functions	express
aws_step_functions	standard
aws_transfer_family	server
aws_waf	access

데이터 소스에 지원되는 타사 소스

다음 표에는 파이프라인을 통해 수집할 때 CloudWatch Logs에서 자동으로 데이터 소스로 분류하는 타사 소스가 나열되어 있습니다.

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
akamai_datastream_2	base_event

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
akamai_datastream_2	dns_activity
akamai_datastream_2	http_activity
box	account_change
box	authentication
box	detection_finding
box	entity_management
box	file_hosting_activity
box	group_management
box	user_inventory_info
broadcom_carbonblack	authentication
broadcom_carbonblack	detection_finding
broadcom_carbonblack	file_system_activity
broadcom_carbonblack	module_activity
broadcom_carbonblack	network_activity
broadcom_carbonblack	process_activity
broadcom_carbonblack	registry_key_activity
broadcom_carbonblack	registry_value_activity
broadcom_carbonblack	script_activity
checkpoint_nextgenerationfirewall	data_security_finding

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
checkpoint_nextgenerationfirewall	detection_finding
checkpoint_nextgenerationfirewall	entity_management
checkpoint_nextgenerationfirewall	network_activity
cisco_duo	authentication
cisco_duo	base_event
cisco_duo	device_inventory_info
cisco_duo	entity_management
cisco_duo	user_inventory_info
cisco_ftd	authentication
cisco_ftd	authorize_session
cisco_ftd	detection_finding
cisco_ftd	network_activity
cisco_ftd	tunnel_activity
cisco_ise	account_change
cisco_ise	api_activity
cisco_ise	application_error
cisco_ise	application_lifecycle
cisco_ise	authentication

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
cisco_ise	entity_management
cisco_ise	network_activity
cisco_ise	tunnel_activity
cisco_ise	web_resources_activity
cisco_meraki	api_activity
cisco_meraki	detection_finding
cisco_meraki	network_activity
cisco_umbrella	data_security_finding
cisco_umbrella	dns_activity
cisco_umbrella	entity_management
cisco_umbrella	network_activity
crowdstrike_falcon	detection_finding
crowdstrike_falcon	process_activity
drupal_core	application_lifecycle
drupal_core	authentication
drupal_core	entity_management
drupal_core	http_activity
entrust_idaas	authentication
entrust_idaas	entity_management
f5_bigip	http_activity

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
f5_bigip	network_activity
github_auditlogs	account_change
github_auditlogs	api_activity
github_auditlogs	entity_management
gitlab	group_management
gitlab	software_inventory_info
gitlab	user_inventory_info
gitlab	vulnerability_finding
gitlab	web_resources_activity
google_workspace	account_change
google_workspace	api_activity
google_workspace	authentication
google_workspace	base_event
google_workspace	data_security_finding
google_workspace	detection_finding
google_workspace	email_activity
google_workspace	entity_management
google_workspace	file_hosting_activity
google_workspace	user_access_management
google_workspace	web_resources_activity

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
hashicorp_vault	api_activity
jamf_pro	authentication
jamf_pro	device_inventory_info
jamf_pro	entity_management
jamf_protect	detection_finding
jamf_protect	device_inventory_info
jamf_protect	dns_activity
jamf_protect	software_inventory_info
jamf_protect	vulnerability_finding
jumpcloud	account_change
jumpcloud	authentication
jumpcloud	datastore_activity
jumpcloud	detection_finding
jumpcloud	entity_management
microsoft_entraid	account_change
microsoft_entraid	authentication
microsoft_entraid	entity_management
microsoft_entraid	user_access_management
microsoft_office365	account_change
microsoft_office365	application_lifecycle

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
microsoft_office365	authentication
microsoft_office365	compliance_finding
microsoft_office365	detection_finding
microsoft_office365	email_activity
microsoft_office365	file_hosting_activity
microsoft_office365	group_management
microsoft_office365	incident_finding
microsoft_office365	user_access_management
microsoft_office365	vulnerability_finding
microsoft_office365	web_resources_activity
microsoft_windows	account_change
microsoft_windows	authentication
microsoft_windows	entity_management
microsoft_windows	event_log_activity
microsoft_windows	file_system_activity
microsoft_windows	group_management
microsoft_windows	kernel_activity
netskope_cloudexchange	account_change
netskope_cloudexchange	authentication
netskope_cloudexchange	data_security_finding

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
netskope_cloudexchange	detection_finding
netskope_cloudexchange	device_inventory_info
netskope_cloudexchange	entity_management
netskope_cloudexchange	file_hosting_activity
netskope_cloudexchange	network_activity
okta_auth0	api_activity
okta_auth0	authentication
okta_sso	api_activity
okta_sso	authentication
okta_sso	detection_finding
okta_sso	entity_management
onelogin_identity	account_change
onelogin_identity	authentication
onelogin_identity	entity_management
paloaltonetworks_nextgenerationfirewall	authentication
paloaltonetworks_nextgenerationfirewall	detection_finding
paloaltonetworks_nextgenerationfirewall	network_activity
paloaltonetworks_nextgenerationfirewall	process_activity

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
paloaltonetworks_prismacloud	base_event
paloaltonetworks_prismacloud	cloud_resources_inventory_info
paloaltonetworks_prismacloud	compliance_finding
paloaltonetworks_prismacloud	detection_finding
paloaltonetworks_prismacloud	device_inventory_info
paloaltonetworks_prismacloud	entity_management
paloaltonetworks_prismacloud	scan_activity
paloaltonetworks_prismacloud	vulnerability_finding
pingidentity_pingaccess	api_activity
pingidentity_pingaccess	http_activity
pingidentity_pingfederate	authentication
pingidentity_pingfederate	entity_management
pingidentity_pingone	account_change
pingidentity_pingone	authentication
pingidentity_pingone	entity_management
proofpoint_tap	detection_finding
proofpoint_tap	email_activity
proofpoint_tap	osint_inventory_info
qualys_csam	device_inventory_info
qualys_csam	osint_inventory_info

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
qualys_csam	software_inventory_info
qualys_csam	vulnerability_finding
qualys_policyaudit	compliance_finding
qualys_vmdr	account_change
qualys_vmdr	api_activity
qualys_vmdr	authentication
qualys_vmdr	device_inventory_info
qualys_vmdr	entity_management
qualys_vmdr	vulnerability_finding
sailpoint_isc	account_change
sailpoint_isc	authentication
sailpoint_isc	entity_management
sailpoint_isc	user_access_management
salesforce_elf	account_change
salesforce_elf	api_activity
salesforce_elf	authentication
salesforce_elf	datastore_activity
salesforce_elf	detection_finding
salesforce_elf	entity_management
salesforce_elf	file_hosting_activity

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
salesforce_elf	http_activity
salesforce_rtem	api_activity
salesforce_rtem	authentication
salesforce_rtem	datastore_activity
salesforce_rtem	file_hosting_activity
sentinelone_endpointsecurity	dns_activity
sentinelone_endpointsecurity	file_system_activity
sentinelone_endpointsecurity	http_activity
sentinelone_endpointsecurity	process_activity
servicenow_cmdb	api_activity
servicenow_cmdb	datastore_activity
servicenow_cmdb	entity_management
slack_auditlogs	account_change
slack_auditlogs	authentication
slack_auditlogs	detection_finding
slack_auditlogs	entity_management
slack_auditlogs	file_hosting_activity
slack_auditlogs	user_access_management
slack_auditlogs	web_resources_activity
snowflake_platform	authentication

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
snowflake_platform	datastore_activity
snowflake_platform	entity_management
snowflake_platform	group_management
snowflake_platform	user_access_management
snowflake_platform	user_inventory_info
tanium_endpointmanagement	account_change
tanium_endpointmanagement	authentication
tanium_endpointmanagement	compliance_finding
tanium_endpointmanagement	detection_finding
tanium_endpointmanagement	device_inventory_info
tanium_endpointmanagement	entity_management
tanium_endpointmanagement	group_management
tanium_endpointmanagement	vulnerability_finding
tenable_vulnerabilitymanagement	account_change
tenable_vulnerabilitymanagement	api_activity
tenable_vulnerabilitymanagement	authentication
tenable_vulnerabilitymanagement	compliance_finding
tenable_vulnerabilitymanagement	device_inventory_info
tenable_vulnerabilitymanagement	scan_activity
tenable_vulnerabilitymanagement	vulnerability_finding

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
wazuh_platform	api_activity
wazuh_platform	application_lifecycle
wazuh_platform	authentication
wazuh_platform	compliance_finding
wazuh_platform	detection_finding
wazuh_platform	device_inventory_info
wazuh_platform	file_system_activity
wazuh_platform	remediation_activity
wazuh_platform	vulnerability_finding
wiz_cnapp	api_activity
wiz_cnapp	authentication
wiz_cnapp	compliance_finding
wiz_cnapp	detection_finding
wiz_cnapp	vulnerability_finding
zeek	authentication
zeek	base_event
zeek	detection_finding
zeek	dhcp_activity
zeek	dns_activity
zeek	email_activity

데이터 소스 이름(@data_source_name 필드)	데이터 소스 유형(@data_source_type 필드)
zeek	ftp_activity
zeek	http_activity
zeek	network_activity
zeek	rdp_activity
zeek	smb_activity
zeek	software_inventory_info
zeek	ssh_activity
zeek	tunnel_activity
zscaler_internetaccess	authentication
zscaler_internetaccess	dns_activity
zscaler_internetaccess	http_activity
zscaler_internetaccess	network_activity

AWS Security Hub CSPM을 통한 추가 타사 소스

AWS Security Hub CSPM 통합을 통해 추가 타사 보안 조사 결과를 사용할 수 있습니다. 다음 파트너는 조사 결과를 Security Hub CSPM으로 전송합니다. 그러면 CloudWatch Logs에서 데이터 소스로 사용할 수 있습니다. 이러한 통합에 대한 자세한 내용은 Security Hub 사용 설명서의 Security Hub CSPM과의 타사 제품 통합을 참조하세요. AWS

파트너	통합
3CORESec – NTA	Security Hub CSPM을 통해 조사 결과를 전송합니다.

파트너	통합
알림 로직 - SIEMless 위협 관리	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Aqua Security - 클라우드 네이티브 보안 플랫폼	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Aqua Security – Kube-bench	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Armor – Armor Anywhere	Security Hub CSPM을 통해 조사 결과를 전송합니다.
AttackIQ	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Barracuda Networks – 클라우드 보안 가디언	Security Hub CSPM을 통해 조사 결과를 전송합니다.
BigID - BigID Enterprise	Security Hub CSPM을 통해 조사 결과를 전송합니다.
파란색 육각형	Security Hub CSPM을 통해 조사 결과를 전송합니다.
체크포인트 - CloudGuard IaaS	Security Hub CSPM을 통해 조사 결과를 전송합니다.
체크포인트 - CloudGuard Posture Management	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Claroty – xDome	Security Hub CSPM을 통해 조사 결과를 전송합니다.
클라우드 스토리지 보안 - Amazon S3용 바이러스 백신	Security Hub CSPM을 통해 조사 결과를 전송합니다.
대비 보안 - 대비 평가	Security Hub CSPM을 통해 조사 결과를 전송합니다.

파트너	통합
CrowdStrike – CrowdStrike Falcon	Security Hub CSPM을 통해 조사 결과를 전송합니다.
CyberArk - 권한 있는 위협 분석	Security Hub CSPM을 통해 조사 결과를 전송합니다.
데이터 이론	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Drata	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – CASB	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – 클라우드 보안 게이트웨이	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – DLP	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – NGFW	Security Hub CSPM을 통해 조사 결과를 전송합니다.
안개	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Guardicore – Centra	Security Hub CSPM을 통해 조사 결과를 전송합니다.
HackerOne – 취약성 인텔리전스	Security Hub CSPM을 통해 조사 결과를 전송합니다.
JFrog – Xray	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Juniper Networks – vSRX 차세대 방화벽	Security Hub CSPM을 통해 조사 결과를 전송합니다.

파트너	통합
k9 보안 – Access Analyzer	Security Hub CSPM을 통해 조사 결과를 전송합니다.
레이스워크	Security Hub CSPM을 통해 조사 결과를 전송합니다.
McAfee – MVISION CNAPP	Security Hub CSPM을 통해 조사 결과를 전송합니다.
NETSCOUT – 사이버 조사자	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Orca - 클라우드 보안 플랫폼	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Palo Alto Networks – Prisma 클라우드 컴퓨팅	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Palo Alto Networks – Prisma Cloud Enterprise	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Plerion – 클라우드 보안 플랫폼	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Prowler	Security Hub CSPM을 통해 조사 결과를 전송합니다.
품질 - 취약성 관리	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Rapid7 – InsightVM	Security Hub CSPM을 통해 조사 결과를 전송합니다.
SentinelOne	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Snyk	Security Hub CSPM을 통해 조사 결과를 전송합니다.

파트너	통합
Sonrai 보안 - Sonrai Dig	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Sophos - 서버 보호	Security Hub CSPM을 통해 조사 결과를 전송합니다.
StackRox – Kubernetes 보안	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Sumo Logic – 기계 데이터 분석	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Symantec – 클라우드 워크로드 보호	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Tenable.io	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Trend Micro – Cloud One	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Vectra - Cognito Detect	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Wiz	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Caveonix – Caveonix 클라우드	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.
Cloud Custodian	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.
DisruptOps	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.
Kion	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.

파트너	통합
Turbot	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.

> **Note**
>
> 이 목록에는 작성 시 조사 결과를 보내는 Security Hub 파트너 통합이 반영됩니다. AWS Security Hub는 정기적으로 새 파트너 통합을 추가하므로 사용 가능한 파트너의 up-to-date 목록은 Security Hub 사용 설명서의 Security Hub CSPM과의 타사 제품 통합을 참조하세요. AWS

사용자 지정 소스

사용자 지정 애플리케이션 로그의 경우 다음 방법 중 하나를 사용하여 자체 데이터 소스 분류를 정의할 수 있습니다.

- 로그 그룹 태그 - `cw:datasource:name` 및 키를 사용하여 로그 그룹에 태그를 추가하여 로그 그룹에 수집된 모든 로그의 데이터 소스 이름과 유형을 `cw:datasource:type` 지정합니다.
- 파이프라인 구성 - 애플리케이션 로그를 수집할 때 로그 처리 파이프라인을 통해 데이터 소스 정보를 구성합니다.

> **Note**
>
> 사용자 지정 데이터 소스 이름은 AWS 서비스 로그와의 충돌을 방지하기 위해 "aws" 또는 "amazon"으로 시작할 수 없습니다.

CloudWatch Log Analytics를 사용하여 로그 데이터 분석

Log Analytics는 로그 분석 기능을 한 곳에서 통합하는 Amazon CloudWatch Logs의 통합 콘솔 환경입니다. CloudWatch Logs Insights를 사용하여 로그 데이터를 쿼리 및 분석하고, Live Tail을 사용하여 실시간으로 로그를 스트리밍하고, CloudWatch Contributor Insights를 사용하여 상위 기여자를 식별할 수 있습니다. Log Analytics는 CloudWatch Logs 콘솔에서 로그를 분석하기 위한 기본 환경입니다.

Log Analytics로 수행할 수 있는 작업

Log Analytics는 다음 기능을 결합합니다.

- 로그 데이터 쿼리 및 분석 - CloudWatch Logs Insights를 사용하여 로그 그룹에 대해 쿼리를 실행합니다. 여러 탭에서 여러 쿼리를 실행하고, 파라미터를 사용하여 쿼리를 저장 및 재사용하고, 패싯으로 데이터를 탐색하고, 자연어를 사용하여 쿼리를 생성하고, 결과를 시각화할 수 있습니다. 자세한 내용은 CloudWatch Logs Insights를 사용한 로그 분석 단원을 참조하십시오.
- 거의 실시간으로 로그 스트리밍 - Live Tail을 사용하여 수집되는 로그 이벤트의 스트리밍 목록을 보고 중요한 이벤트를 필터링하거나 강조 표시합니다. 자세한 내용은 CloudWatch Logs Live Tail로 문제 해결 단원을 참조하십시오.
- 상위 기여자 식별 - CloudWatch Contributor Insights를 사용하여 높은 카디널리티 로그 데이터를 분석하고 시스템에 가장 큰 영향을 미치는 기여자를 확인합니다. 자세한 내용은 Contributor Insights를 사용하여 높은 카디널리티 데이터 분석을 참조하세요.

Log Analytics 시작하기

Log Analytics를 사용하려면 CloudWatch 콘솔에 로그인하고 탐색 창의 로그에서 로그 분석을 선택합니다.

Log Analytics가 기본 환경입니다. 옵트아웃하면 CloudWatch Logs Insights, Live Tail 및 Contributor Insights를 Log Analytics와 함께 고유한 경험으로 계속 사용할 수 있으며 언제든지 Log Analytics로 다시 전환할 수 있습니다.

Log Analytics는 CloudWatch Logs Insights 쿼리, Live Tail 및 Contributor Insights와 같은 기본 기능과 동일한 요금을 사용합니다. 요금에 대한 자세한 내용은 CloudWatch 요금을 참조하십시오.

CloudWatch Logs Insights를 사용한 로그 분석

CloudWatch Logs Insights를 사용하면 Amazon CloudWatch Logs 내 로그 데이터를 대화식으로 검색하고 분석할 수 있습니다. 작업상의 문제에 보다 효율적이고 효과적으로 대처할 수 있도록 쿼리를 수행할 수 있습니다. 로그 그룹을 사용하여 쿼리하는 것 외에도 패싯, 데이터 소스 및 데이터 유형을 사용하여 쿼리할 수 있습니다. 문제가 발생하면 CloudWatch Logs Insights를 사용해 잠재적인 원인을 식별하고 배포된 수정 사항을 확인할 수 있습니다. 대시보드에 추가된 쿼리를 포함하여 계정당 동시 CloudWatch Logs Insights QL은 100개로 제한됩니다. 또한 OpenSearch Service PPL 또는 OpenSearch Service SQL에 대해 15개의 동시 쿼리를 실행할 수 있습니다.

CloudWatch Logs Insights는 쿼리에 사용할 수 있는 3가지 쿼리 언어를 지원합니다.

- 간단하지만 강력한 명령이 포함된 특수 제작된 Logs Insights 쿼리 언어(Logs Insights QL)입니다.
- OpenSearch Service Piped Processing Language(PPL). OpenSearch PPL을 통해 파이프(|)로 구분된 명령 집합을 사용하여 로그를 분석할 수 있습니다.

  OpenSearch PPL을 사용하면 함께 파이프되는 명령을 사용하여 데이터를 검색, 쿼리 및 분석할 수 있으므로 복잡한 쿼리를 더 쉽게 이해하고 구성할 수 있습니다. 구문을 사용하면 명령을 연결하여 데이터를 변환하고 처리할 수 있습니다. PPL을 사용하면 데이터를 필터링 및 집계하고 분석에 풍부한 수학, 문자열, 날짜, 조건부 및 기타 함수 집합을 사용할 수 있습니다.
- OpenSearch Service Structured Query Language(SQL). OpenSearch SQL 쿼리를 사용하면 선언적 방식으로 로그를 분석할 수 있습니다. SELECT, FROM, WHERE, GROUP BY, HAVING 등의 명령과 SQL에서 사용할 수 있는 기타 다양한 명령 및 함수를 사용할 수 있습니다. 로그 그룹 간에 JOINs 실행하고, 하위 쿼리를 사용하여 로그 간에 데이터를 상호 연관시키고, 풍부한 JSON, 수학, 문자열, 조건부 및 기타 SQL 함수 집합을 사용하여 로그에 대한 강력한 분석을 수행할 수 있습니다.

  SQL 또는 PPL 명령을 사용하는 경우 필드를 성공적으로 쿼리하려면 특수 문자(비영숫자 및 비숫자)가 포함된 필드를 백틱으로 묶어야 합니다. 예를 들어, 백틱에서 `@message`, `Operation.Export` 및 `Test::Field`을 묶습니다. 순전히 알파벳 이름의 필드를 백틱으로 묶을 필요는 없습니다.

CloudWatch Logs Insights는 모든 쿼리 언어에서 사용할 수 있는 다음과 같은 기능을 제공합니다.

- Amazon Route 53, AWS Lambda AWS CloudTrail및 Amazon VPC와 같은 서비스의 로그 AWS 에서 로그 필드를 자동으로 검색하고 로그 이벤트를 JSON으로 내보내는 애플리케이션 또는 사용자 지정 로그를 자동으로 검색합니다.
- 특히 많은 수의 로그 그룹 또는 로그 이벤트의 쿼리의 경우 비용을 절감하고 결과를 가속화하는 필드 인덱스를 생성합니다. 로그 이벤트에서 일반적인 필드의 필드 인덱스를 생성한 후 쿼리에서 사용할

수 있습니다. 쿼리는 인덱싱된 필드를 포함하지 않는 것으로 알려진 로그 이벤트 처리를 건너뛰고 더 적은 데이터를 처리합니다.

> **Note**
>
> `filterIndex` 명령은 Logs Insights QL에서만 사용할 수 있습니다.

- 로그 이벤트의 패턴 감지 및 분석. 패턴은 로그 필드 간에 반복되는 공유 텍스트 구조입니다. 쿼리 결과를 확인할 때 패턴 탭을 선택하면 결과 샘플을 기반으로 CloudWatch Logs가 발견한 패턴을 볼 수 있습니다.
- 쿼리 저장, 쿼리 기록 보기, 저장된 쿼리 다시 실행, 파라미터와 함께 저장된 쿼리 사용.
- 대시보드에 쿼리 추가.
- 를 사용하여 쿼리 결과 암호화 AWS Key Management Service.
- 자연어를 사용하여 쿼리를 생성하면 자연어를 사용하여 CloudWatch Logs Insights 쿼리를 생성할 수 있습니다. 찾고 있는 데이터에 대해 질문하거나 설명할 수 있습니다. 그러면 AI가 프롬프트를 기반으로 쿼리를 생성하고 쿼리가 작동하는 방식에 대한 자세한 설명을 제공합니다.
- 패싯을 사용하여 로그를 그룹화, 필터링 및 대화형으로 탐색할 수 있습니다.
- 로그를 둘러싸면 쿼리 결과의 특정 로그 레코드 전후에 로그 줄을 보고 중요한 로그 이벤트에 대한 즉각적인 컨텍스트를 얻을 수 있습니다. 선택한 레코드 전후에 5, 10, 20, 50 또는 100개의 로그 줄을 보고 주변 로그 내에서 특정 키워드를 검색하도록 범위를 구성할 수 있습니다.

다음 CloudWatch Logs Insights 기능은 Logs Insights QL을 사용하는 경우에만 지원됩니다.

- 로그 그룹의 로그 이벤트를 이전 기간의 로그 이벤트와 비교하는 비교 쿼리.

> **Important**
>
> CloudWatch Logs Insights는 로그 그룹 생성 시간 이전의 타임스탬프가 있는 로그 이벤트에 액세스할 수 없습니다.

CloudWatch 계정 간 가시성에서 모니터링 계정으로 설정된 계정에 로그인하면 이 모니터링 계정에 연결된 소스 계정의 로그 그룹에서 CloudWatch Logs Insights 쿼리를 실행할 수 있습니다. 다른 계정에 있는 여러 로그 그룹을 쿼리하는 쿼리를 실행할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관찰성을 참조하세요.

Logs Insights QL을 사용하여 쿼리를 생성할 때 자연어를 사용하여 CloudWatch Logs Insights 쿼리를 생성할 수도 있습니다. 그러려면 찾고 있는 데이터에 대해 질문하거나 설명합니다. 이 AI 지원 기능은 프롬프트를 기반으로 쿼리를 생성하며, 쿼리 작동 방식에 대한 줄별 설명을 제공합니다. 자세한 내용은 자연어를 사용하여 CloudWatch Log Insights 쿼리 생성 및 업데이트를 참조하세요.

지원되는 쿼리 언어를 사용하는 쿼리는 완료되지 않은 경우 60분 후에 시간 초과됩니다. 쿼리 결과는 7일 동안 사용할 수 있습니다.

CloudWatch Logs Insights 쿼리에는 쿼리 언어에 관계없이 스캔된 압축되지 않은 로그 데이터의 양에 따라 요금이 부과됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

CloudWatch Logs Insights를 사용하면 2018년 11월 5일 이후 CloudWatch Logs로 전송된 로그 데이터를 검색할 수 있습니다.

> **Important**
>
> 네트워크 보안 팀에서 웹 소켓 사용을 허용하지 않는 경우 현재 CloudWatch 콘솔의 CloudWatch Logs Insights 부분에 액세스할 수 없습니다. 이러한 경우 API를 통해 CloudWatch Logs Insights 쿼리 기능을 사용할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs API 참조의 StartQuery를 참조하세요.

내용

- 지원되는 쿼리 언어
- 자연어를 사용하여 CloudWatch Logs Insights 쿼리를 생성하고 업데이트
- 지원되는 로그 및 검색되는 필드
- 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성
- 패싯을 사용하여 로그 그룹화 및 탐색
- CloudWatch Logs Insights에서 주변 로그 보기
- 패턴 분석
- CloudWatch Logs Insights 쿼리 저장 및 재실행
- 대시보드에 쿼리 추가 또는 쿼리 결과 내보내기
- 실행 중인 쿼리 또는 쿼리 기록 보기
- 를 사용하여 쿼리 결과 암호화 AWS Key Management Service
- CloudWatch Logs Insights 쿼리 결과에서 자연어 요약 생성

지원되는 쿼리 언어

다음 섹션에서는 각 쿼리 언어에서 지원되는 명령을 나열합니다. 또한 구문 형식을 설명하고 샘플 쿼리를 제공합니다.

주제

- CloudWatch Logs Insights 쿼리 언어(Logs Insights QL)
- OpenSearch Piped Processing Language(PPL)
- OpenSearch Structured Query Language(SQL)

CloudWatch Logs Insights 쿼리 언어(Logs Insights QL)

이 섹션에는 Logs Insights QL 명령 및 함수에 대한 전체 설명서가 포함되어 있습니다. 또한이 언어에 대한 샘플 쿼리도 포함되어 있습니다.

사용할 수 있는 다른 쿼리 언어에 대한 자세한 내용은 OpenSearch Service PPL, OpenSearch Service SQL 및 CloudWatch Metrics Insights를 참조하세요.

주제

- CloudWatch Logs Insights 언어 쿼리 구문
- Logs Insights QL 시작하기: 쿼리 자습서
- 샘플 쿼리
- (diff)를 이전 시간 범위와 비교
- 그래프로 로그 데이터 시각화

CloudWatch Logs Insights 언어 쿼리 구문

이 섹션에서는 Logs Insights QL에 대한 세부 정보를 제공합니다. 쿼리 구문은 일반 함수, 산술 및 비교 연산, 정규 표현식을 포함하되 이에 국한되지 않는 다양한 함수 및 연산을 지원합니다.

> **Important**
>
> 대규모 쿼리 실행으로 과도한 요금이 발생하는 것을 피하려면 다음 모범 사례를 명심하세요.
>
> - 각 쿼리에 필요한 로그 그룹만 선택합니다.
> - 쿼리의 시간 범위는 항상 가능한 한 가장 좁게 지정합니다.

- 콘솔을 사용하여 쿼리를 실행하는 경우 CloudWatch Logs Insights 콘솔 페이지를 닫기 전에 모든 쿼리를 취소합니다. 그러지 않으면 쿼리가 완료될 때까지 계속 실행됩니다.
- CloudWatch Logs Insights 위젯을 대시보드에 추가할 때는 새로 고침할 때마다 새 쿼리가 시작되므로 대시보드가 자주 새로 고쳐지지 않도록 해야 합니다.

여러 명령이 포함된 쿼리를 생성하려면 파이프 문자(|)로 명령을 구분합니다.

코멘트가 포함된 쿼리를 생성하려면 설명을 해시 문자(#)로 설정합니다.

> **Note**
>
> CloudWatch Logs Insights에서는 다양한 로그 유형의 필드를 자동으로 검색하고 @ 문자로 시작하는 필드를 생성합니다. 이러한 필드에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 지원되는 로그 및 검색된 필드를 참조하세요.

다음 표에는 각 명령에 대한 간략한 설명이 나와 있습니다. 다음 표에는 예제와 함께 각 명령에 대한 보다 포괄적인 설명이 나와 있습니다.

> **Note**
>
> 모든 Log Insights QL 쿼리 명령은 Standard 로그 클래스의 로그 그룹에서 지원됩니다. Infrequent Access 로그 클래스의 로그 그룹은 `pattern`, `diff` 및 `unmask`을 제외한 모든 Log Insights QL 쿼리 명령을 지원합니다.

addtotals	숫자 필드의 행 합계와 열 합계를 계산합니다.
anomaly	기계 학습을 사용하여 로그 데이터의 비정상적인 패턴을 식별합니다.
display	쿼리 결과에 하나 이상의 특정 필드를 표시합니다.
fields	쿼리 결과에 특정 필드를 표시하고, 필드 값을 수정하고 쿼리에 사용할 새 필드를 생성하는 데 사용할 수 있는 함수와 연산을 지원합니다.
filter	하나 이상의 조건과 일치하는 로그 이벤트만 반환하도록 쿼리를 필터링합니다.

filterIndex	쿼리가 필드 인덱스에 언급된 필드에 인덱싱되고 해당 필드 인덱스의 값도 포함하는 로그 그룹만 스캔하도록 강제합니다. 이렇게 하면이 필드 인덱스에 대한 쿼리에 지정된 값이 포함된 이러한 로그 그룹의 로그 이벤트만 스캔하려고 시도하여 스캔 볼륨이 줄어듭니다. 이 명령은 Infrequent Access 로그 클래스의 로그 그룹에는 지원되지 않습니다.
pattern	로그 데이터를 패턴으로 자동 클러스터링합니다. 패턴은 로그 필드 간에 반복되는 공유 텍스트 구조입니다. CloudWatch Logs Insights는 로그 이벤트에서 발견된 패턴을 분석할 수 있는 방법을 제공합니다. 자세한 내용은 패턴 분석 섹션을 참조하세요.
diff	요청된 기간에서 발견된 로그 이벤트를 길이가 동일한 이전 기간의 로그 이벤트와 비교하여 추세를 찾고 특정 로그 이벤트가 새 이벤트인지 확인할 수 있습니다.
parse	로그 필드에서 데이터를 추출하여 쿼리에서 처리할 수 있는 추출된 필드를 생성합니다. **parse**는 와일드카드를 사용하는 글로브 모드와 정규 표현식을 모두 지원합니다.
sort	반환된 로그 이벤트를 오름차순(asc) 또는 내림차순(desc)으로 표시합니다.
SOURCE	쿼리SOURCE에를 포함시키는 것은 로그 그룹 이름 접두사, 계정 식별자, 로그 그룹 클래스, 데이터 소스 또는 로그 그룹 태그를 기반으로 쿼리에 포함할 대량의 로그 그룹을 지정하는 유용한 방법입니다. 이 명령은 CloudWatch 콘솔이 아닌 AWS CLI 또는 프로그래밍 방식으로 쿼리를 생성하는 경우에만 지원됩니다.
stats	로그 필드의 값을 사용하여 집계 통계를 계산합니다.
limit	쿼리에서 반환할 최대 로그 이벤트의 수를 지정합니다. '상위 20개' 또는 '최근 20개' 결과를 반환하는 **sort**에 유용합니다. 충분한 결과가 발견되면 `limit any`를 사용하여 스캔을 조기에 중지합니다.
dedup	지정한 필드의 특정 값을 기준으로 중복 결과를 제거합니다.

unmask	데이터 보호 정책으로 인해 일부 내용이 마스킹 처리된 로그 이벤트의 모든 내용을 표시합니다. 로그 그룹의 데이터 보호에 대한 자세한 내용은 마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원 섹션을 참조하세요.
unnest	입력으로 취해진 목록을 평면화하여 목록의 각 요소에 대해 단일 레코드로 여러 레코드를 생성합니다.
lookup	필드 값을 일치시켜 조회 테이블의 데이터로 로그 이벤트를 강화합니다. 조회 테이블을 사용하여 사용자 세부 정보, 애플리케이션 이름 또는 제품 정보와 같은 참조 데이터를 쿼리 결과에 추가합니다.
join	소스 로그 그룹의 로그 이벤트를 일치하는 필드를 기반으로 다른 로그 그룹 또는 쿼리 결과의 이벤트와 결합합니다. 조인 명령을 사용하여 일치하는 요청 식별자 또는 트랜잭션 IDs.
subqueries	하위 쿼리는 다른 쿼리에 대한 입력으로 사용할 수 있는 중첩된 Logs Insights 쿼리입니다. 하위 쿼리를 사용하여 중간 결과 세트를 도출한 다음 후속 명령에서 사용할 수 있습니다.
autoregress	필드 값의 지연된(이전 행) 복사본을 생성합니다.
logcompare	현재 기간을 기간만큼 뒤로 이동한 기준 기간과 비교합니다.
filldown	마지막 null이 아닌 값을 전달하여 간격을 채웁니다.
fillmissing	통계 뒤에 빈()으로 빈 시간 빈 행을 삽입하고 선택적으로 필드를 상수로 채웁니다.
cidrlookup	조회 테이블의 CIDR 범위에 대해 IP 필드를 일치시켜 이벤트를 강화합니다.
outlier	사분위수 범위(IQR)를 기반으로 통계 이상치를 감지하고 이상치 행을 제거하거나 변환할 수 있습니다.
accum	숫자 필드의 실행 중인 누적 합계를 계산합니다.
appendcols	위치 행 일치를 통해 하위 쿼리의 열을 현재 결과에 추가합니다.

sessionize	자격 증명 필드와 비활성 간격을 기준으로 이벤트를 세션으로 그룹화합니다.
countFrequent	각 고유 필드-값 조합의 대략적인 개수를 내림차순으로 정렬하여 반환합니다.
where	`filter` 명령에 대한 문법 별칭 역할을 하며 동일한 구문을 허용합니다.
추정	쿼리를 실행하지 않고 선택한 로그 그룹 및 시간 범위에서 쿼리가 스캔할 예상 바이트를 반환합니다.
기타 연산 및 함수	CloudWatch Logs Insights는 또한 많은 비교, 산술, 날짜/시간, 숫자, 문자열, IP 주소, 일반 함수 및 연산을 지원합니다.

다음 섹션에서는 CloudWatch Logs Insights 쿼리 명령에 대한 자세한 내용을 제공합니다.

주제

- 로그 클래스에서 지원되는 Logs Insights QL 명령
- addtotals
- 이상
- 표시
- 필드
- 필터
- filterIndex
- 소스
- pattern
- DIFF
- parse
- 관련 필드
- 확장
- sort
- 통계
- limit
- dedup

- unmask
- unnest
- lookup
- join
- 하위 쿼리
- 자동 회귀
- 로그 비교
- 채우기
- 채우기
- cidrlookup
- 이상치
- accum
- appendcols
- 세션화
- countFrequent
- where
- 추정
- 부울, 비교, 숫자, 날짜/시간 및 기타 함수
- 해시 함수
- 시계열 함수
- 특수 문자가 포함된 필드
- 쿼리에 별칭 및 코멘트 사용

로그 클래스에서 지원되는 Logs Insights QL 명령

모든 Log Insights QL 쿼리 명령은 Standard 로그 클래스의 로그 그룹에서 지원됩니다. Infrequent Access 로그 클래스의 로그 그룹은 `pattern`, `diff`. `filterIndex`, `unmask`를 제외한 모든 쿼리 명령을 지원합니다.

addtotals

addtotals를 사용하여 숫자 필드의 행 합계와 열 합계를 계산합니다. 행 합계는 청크당 계산됩니다. 분산 청크 간의 정확성을 위해 병합 후 열 합계(col=true)가 계산됩니다. 사용자 지정 필드 선택, 사용자 지정 필드 이름 및를 사용하여 행 합계를 비활성화row=false할 수 있습니다.

구문

```
| addtotals [fieldname=Name] [row=bool] [col=bool] [field1, field2, ...]
```

- fieldname=*Name* - 행 합계 열의 이름을 설정합니다. 기본값은 Total입니다.
- row=*bool* - 행 합계를 활성화하거나 비활성화합니다. 기본값은 true입니다.
- col=*bool* - 열 합계를 활성화하거나 비활성화하여 하단에 요약 행을 추가합니다. 기본값은 false입니다.
- *field1*, *field2*, ... - 합계할 특정 필드의 선택적 목록입니다. 생략하면 모든 숫자 필드가 합산됩니다.

예제

```
# Row totals (default) - adds a Total column summing all numeric fields
fields a, b
| addtotals
```

```
# Sum only specific fields
fields price, tax, qty
| addtotals price, tax
```

```
# Custom total column name
fields val
| addtotals fieldname=RowSum
```

```
# Column totals - adds a Summary row at the bottom
fields x
| addtotals col=true x
```

이상

anomaly를 사용하면 기계 학습을 사용하여 로그 데이터 내 비정상적인 패턴과 잠재적 문제를 자동으로 식별할 수 있습니다.

이 anomaly 명령은 기존 pattern 기능을 확장하고 고급 분석을 사용하여 로그 데이터의 잠재적 이상을 식별하는 데 도움이 됩니다. anomaly를 사용하면 로그에서 비정상적인 패턴이나 동작을 자동으로 표시하여 연산 문제를 식별하고 해결하는 데 걸리는 시간을 줄일 수 있습니다.

anomaly 명령은 pattern 명령과 작동하여 먼저 로그 패턴을 식별한 다음 해당 패턴 내에서 이상을 감지합니다. anomaly를 filter 또는 sort 명령과 결합하여 데이터의 특정 하위 집합에 이상 탐지를 집중할 수도 있습니다.

이상 명령 입력

anomaly 명령은 일반적으로 pattern 명령 다음에 로그 데이터에서 식별된 패턴을 분석하는 데 사용됩니다. 명령에는 추가 파라미터가 필요하지 않으며 쿼리에서 이전 명령의 출력을 분석합니다.

식별된 이상 유형

anomaly 명령은 다음과 같은 5가지 유형의 이상을 식별합니다.

- 패턴 빈도 이상: 애플리케이션이 평소보다 더 많은 오류 메시지를 생성하기 시작하는 경우와 같이 특정 로그 패턴의 비정상적인 빈도.
- 새 패턴 이상: 로그에 나타나는 새 유형의 오류 또는 메시지를 나타낼 수 있는 이전에 보이지 않았던 로그 패턴입니다.
- 토큰 변형 이상: 예상 로그 형식의 비정상적인 변형을 나타낼 수 있는 로그 메시지 내용의 예기치 않은 변경 사항입니다.
- 숫자 토큰 이상: 잠재적 성능 문제 또는 예상치 못한 지표 변동을 감지하는 데 도움이 될 수 있는 로그 내 숫자 값의 비정상적인 변경입니다.
- HTTP 오류 코드 이상: HTTP 오류 응답과 관련된 패턴으로, 웹 애플리케이션 및 API를 모니터링할 때 특히 유용합니다.

이상 명령 출력

anomaly 명령은 입력 데이터의 모든 필드를 보존하고 이상 탐지 결과를 추가하여 로그 데이터에서 비정상적인 패턴을 식별하는 데 도움이 됩니다.

예제

다음 명령은 로그 데이터의 패턴을 식별한 다음 해당 패턴 내에서 이상을 감지합니다.

```
fields @timestamp, @message
| pattern @message
| anomaly
```

anomaly 명령을 필터링과 함께 사용하여 특정 로그 유형에 집중할 수 있습니다.

```
fields @timestamp, @message
| filter @type = "REPORT"
| pattern @message
| anomaly
```

anomaly 명령을 정렬과 결합하여 결과를 구성할 수 있습니다.

```
fields @timestamp, @message
| filter @type = "ERROR"
| pattern @message
| anomaly
| sort @timestamp desc
```

표시

display 명령을 사용하여 쿼리 결과에 하나 또는 여러 개의 특정 필드를 표시할 수 있습니다.

display 명령은 사용자가 지정한 필드만 표시합니다. 쿼리에 여러 display 명령이 포함된 경우 쿼리 결과는 마지막 display 명령에서 지정한 필드만 표시합니다.

예: 하나의 필드 표시

아래 코드 조각은 parse 명령을 사용하여 @message에서 데이터를 추출하여 추출된 필드 loggingType 및 loggingMessage를 생성하는 쿼리의 예를 보여줍니다. 쿼리는 loggingType에 대한 값이 ERROR인 모든 로그 이벤트를 반환합니다. display은 쿼리 결과에서 loggingMessage에 대한 값만 표시합니다.

```
fields @message
| parse @message "[*] *" as loggingType, loggingMessage
| filter loggingType = "ERROR"
| display loggingMessage
```

> **Tip**
>
> 쿼리에서 display를 한 번만 사용하세요. 쿼리에서 display를 두 번 이상 사용할 경우, 쿼리 결과에는 마지막으로 사용된 display 명령에서 지정한 필드만 표시됩니다.

필드

fields를 사용하여 쿼리 결과에 특정 필드를 표시할 수 있습니다.

쿼리에 fields 명령이 여러 개 포함되어 있는데 display 명령은 포함되지 않은 경우 결과에 fields 명령에 지정된 모든 필드가 표시됩니다.

예: 특정 필드 표시

다음 예제에서는 20개의 로그 이벤트를 반환하고 내림차순으로 표시하는 쿼리를 보여줍니다. 쿼리 결과에 @timestamp 및 @message에 대한 값이 표시됩니다.

```
fields @timestamp, @message
| sort @timestamp desc
| limit 20
```

필드 값을 수정하고 쿼리에 사용할 수 있는 새 필드를 생성하기 위해 fields에서 지원하는 다양한 함수와 연산을 사용하려는 경우 display 대신 fields를 사용하세요.

as 키워드와 함께 fields 명령을 사용하면 로그 이벤트에 있는 필드 및 함수를 사용하는 추출된 필드를 생성할 수 있습니다. 예를 들어, fields ispresent as isRes은 이름이 isRes인 추출된 필드를 생성하며, 쿼리의 나머지 부분에서 추출된 필드를 사용할 수 있습니다.

필터

filter 명령을 사용하여 하나 이상의 조건과 일치하는 로그 이벤트를 가져올 수 있습니다.

예: 하나의 조건을 사용한 로그 이벤트 필터링

아래 코드 조각은 range에 대한 값이 3000보다 큰 모든 로그 이벤트를 반환하는 쿼리의 예를 보여줍니다. 쿼리는 결과를 20개의 로그 이벤트로 제한하며, 로그 이벤트를 내림차순의 @timestamp로 정렬합니다.

```
fields @timestamp, @message
| filter (range>3000)
| sort @timestamp desc
```

```
| limit 20
```

예: 둘 이상의 조건을 사용한 로그 이벤트 필터링

키워드 and 및 or을 사용하여 둘 이상의 조건을 결합할 수 있습니다.

아래 코드 조각은 range에 대한 값이 3000보다 크고 accountId에 대한 값이 123456789012와 동일한 로그 이벤트를 반환하는 쿼리의 예를 보여줍니다. 쿼리는 결과를 20개의 로그 이벤트로 제한하며, 로그 이벤트를 내림차순의 @timestamp로 정렬합니다.

```
fields @timestamp, @message
| filter (range>3000 and accountId=123456789012)
| sort @timestamp desc
| limit 20
```

인덱싱된 필드 및 필터 명령

로그 그룹에 대한 필드 인덱스를 생성한 경우 해당 필드 인덱스를 사용하여 filter 쿼리를 더 효율적으로 만들고 스캔 볼륨을 줄일 수 있습니다. 예를 들어, requestId에 대한 필드 인덱스를 생성했다고 가정해 보겠습니다. 그런 다음 인덱싱된 필드를 포함하지 않는 것으로 알려진 로그 이벤트 처리를 건너뛰filter requestId = *value*거나 건너뛰filter requestId IN [*value*, *value*, ...]려고 시도하는 해당 로그 그룹에 대한 CloudWatch Logs Insights 쿼리입니다. 인덱싱된 필드를 포함하는 것으로 알려진 로그 이벤트만 스캔하려고 하면 스캔 볼륨을 줄일 수 있으며 쿼리 속도가 빨라집니다.

필드 인덱스와 필드 인덱스 생성 방법에 대한 자세한 내용은 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성 섹션을 참조하세요.

> **Important**
>
> filter *fieldName* =... 및 filter *fieldName* IN...을 사용하는 쿼리만 필드 인덱스 개선의 이점을 누릴 수 있습니다. filter *fieldName* like을 사용하는 쿼리는 인덱스를 사용하지 않으며 항상 선택한 로그 그룹의 모든 로그 이벤트를 스캔합니다.

예: 인덱스를 사용하여 특정 요청 ID와 관련된 로그 이벤트 찾기

이 예제에서는 requestId에서 필드 인덱스를 생성했다고 가정합니다. 이 필드 인덱스를 사용하는 로그 그룹의 경우 쿼리는 필드 인덱스를 사용하여 값이 인 이벤트를 찾기 위해 가장 적은 수의 로그 이벤트를 스캔하려고 시도requestId합니다. 123456

```
fields @timestamp, @message
| filter requestId = "1234656"
| limit 20
```

필터 명령의 일치 항목 및 정규식

필터 명령은 정규 표현식 사용을 지원합니다. 비교 연산자(=, !=, <, <=, >, >=) 및 부울 연산자(and, or, not)를 사용할 수 있습니다.

in 키워드를 사용하여 설정된 멤버십을 테스트하고 배열의 요소를 확인할 수 있습니다. 배열에서 요소를 확인하려면 in 뒤에 해당 배열을 넣습니다. in과 함께 부울 연산자 not을 사용할 수 있습니다. in을 사용하여 필드가 문자열과 일치하는 로그 이벤트를 반환하는 쿼리를 생성할 수 있습니다. 필드는 완전한 문자열이어야 합니다. 예를 들어, 다음 코드 조각은 in을 사용하여 logGroup 필드가 완전한 문자열 example_group인 로그 이벤트를 반환하는 쿼리를 보여줍니다.

```
fields @timestamp, @message
| filter logGroup in ["example_group"]
```

키워드 구문 like 및 not like를 사용하여 하위 문자열을 일치시킬 수 있습니다. 정규 표현식 연산자 =~를 사용하여 하위 문자열을 일치시킬 수 있습니다. like 및 not like로 하위 문자열을 일치시키려면, 일치해야 하는 하위 문자열을 큰따옴표 또는 작은따옴표로 둘러쌉니다. like 및 not like와 함께 정규식 패턴을 사용할 수 있습니다. 하위 문자열을 정규식 연산자와 일치시키려면 원하는 하위 문자열을 슬래시로 둘러쌉니다. 다음 예제에는 filter 명령을 사용하여 하위 문자열을 일치시키는 방법을 보여주는 코드 조각이 포함되어 있습니다.

예: 하위 문자열 일치

다음 예제에서는 f1에 단어 Exception이 포함된 모든 이벤트를 반환합니다. 세 예제 모두 대/소문자를 구별합니다.

첫 번째 예제에서는 like를 사용하여 하위 문자열과 일치시킵니다.

```
fields f1, f2, f3
| filter f1 like "Exception"
```

두 번째 예제에서는 like 및 정규 표현식 패턴을 사용하여 하위 문자열을 일치시킵니다.

```
fields f1, f2, f3
| filter f1 like /Exception/
```

세 번째 예제에서는 정규 표현식을 사용하여 하위 문자열을 일치시킵니다.

```
fields f1, f2, f3
| filter f1 =~ /Exception/
```

예: 와일드카드를 사용하여 하위 문자열 일치

정규 표현식에서 마침표 기호(.)를 와일드카드로 사용하여 하위 문자열을 일치시킬 수 있습니다. 다음 예에서 쿼리는 f1에 대한 값이 ServiceLog 문자열로 시작하는 일치 항목을 반환합니다.

```
fields f1, f2, f3
| filter f1 like /ServiceLog./
```

마침표 기호(.*) 뒤에 별표 기호를 배치하여 가능한 한 많은 일치 항목을 반환하는 탐욕적 수량자(Greedy Quantifier)를 생성할 수 있습니다. 예를 들어, 다음 쿼리는 f1 값이 ServiceLog 문자열로 시작할 뿐만 아니라 ServiceLog 문자열도 포함하는 일치 항목을 반환합니다.

```
fields f1, f2, f3
| filter f1 like /ServiceLog.*/
```

가능한 일치 항목은 다음과 같습니다.

- ServiceLogSampleApiLogGroup
- SampleApiLogGroupServiceLog

예제: 일치 항목에서 하위 문자열 제외

다음 예제에서는 f1에 단어 Exception이 포함되지 않은 모든 로그 이벤트를 반환하는 쿼리를 보여줍니다. 이 예제에서는 대/소문자를 구분합니다.

```
fields f1, f2, f3
| filter f1 not like "Exception"
```

예제: 대/소문자를 구분하지 않는 패턴으로 하위 문자열 일치

대/소문자를 구분하지 않는 하위 문자열을 like 및 정규 표현식을 사용하여 일치시킬 수 있습니다. 일치시키려는 하위 문자열 앞에 다음 파라미터(?i)를 입력합니다. 다음 예제에서는 f1에 단어 Exception 또는 exception이 포함된 모든 로그 이벤트를 반환하는 쿼리를 보여줍니다.

```
fields f1, f2, f3
| filter f1 like /(?i)Exception/
```

filterIndex

`filterIndex`를 사용하여 쿼리에서 지정한 필드에 인덱싱된 로그 그룹만 검색하도록 강제하여 인덱스된 데이터만 반환합니다. 해당 필드에 인덱싱된 이러한 로그 그룹의 경우, 인덱싱된 필드의 쿼리에 지정된 필드가 포함된 로그 이벤트가 없는 로그 그룹을 건너뛰어 쿼리를 추가로 최적화합니다. 또한 해당 필드 인덱스의 쿼리에 지정된 값과 일치하는 이러한 로그 그룹의 로그 이벤트만 스캔하려고 시도하여 스캔 볼륨을 줄입니다. 필드 인덱스와 필드 인덱스 생성 방법에 대한 자세한 내용은 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성 섹션을 참조하세요.

인덱싱된 필드와 `filterIndex`를 사용하면 실제 검색 공간을 필드 인덱스가 있는 로그 그룹 및 로그 이벤트로 제한하여 페타바이트의 로그 데이터가 포함된 로그 그룹을 효율적으로 쿼리할 수 있습니다.

예를 들어, 계정의 일부 로그 그룹에서 `IPaddress`에 대한 필드 인덱스를 생성했다고 가정해 보겠습니다. 그런 다음 아래 쿼리를 생성하고 계정의 모든 로그 그룹을 쿼리하여 `IPaddress` 필드에 `198.51.100.0` 값이 포함된 로그 이벤트를 찾도록 선택할 수 있습니다.

```
fields @timestamp, @message
| filterIndex IPaddress = "198.51.100.0"
| limit 20
```

`filterIndex` 명령을 실행하면 이 쿼리가 `IPaddress`에 대해 인덱싱되지 않은 모든 로그 그룹을 건너뛰려고 시도합니다. 또한 인덱싱된 로그 그룹 내에서 쿼리는 `IPaddress` 필드가 있지만 해당 필드의 값으로 관찰된 `198.51.100.0`은 없는 로그 이벤트를 건너뜁니다.

`IN` 연산자를 사용하여 인덱싱된 필드의 여러 값 중 하나로 결과를 확장합니다. 다음 예제는 `IPaddress` 필드에 값 `198.51.100.0` 또는 `198.51.100.1`을 포함하는 로그 이벤트를 찾습니다.

```
fields @timestamp, @message
| filterIndex IPaddress in ["198.51.100.0", "198.51.100.1"]
| limit 20
```

CloudWatch Logs는 Standard 로그 클래스의 모든 로그 그룹에 대한 기본 필드 인덱스를 제공합니다. 기본 필드 인덱스는 다음 필드에 자동으로 사용할 수 있습니다.

- `@logStream`

- @aws.region
- @aws.account
- @source.log
- @data_source_name
- @data_source_type
- @data_format
- traceId
- severityText
- attributes.session.id

CloudWatch Logs는 특정 데이터 소스 이름 및 유형 조합에 대한 기본 필드 인덱스도 제공합니다. 기본 필드 인덱스는 다음 데이터 소스 이름 및 유형 조합에 자동으로 사용할 수 있습니다.

데이터 소스 이름 및 유형	기본 필드 인덱스
amazon_vpc.flow	action logStatus region flowDirection type
amazon_route53.resolver_query	query_type transport rcode
aws_waf.access	action httpRequest.country
aws_cloudtrail.data aws_cloudtrail.management	eventSource eventName

데이터 소스 이름 및 유형	기본 필드 인덱스
	awsRegion userAgent errorCode eventType managementEvent readOnly eventCategory requestId

기본 필드 인덱스는 정책 내에서 정의하는 사용자 지정 필드 인덱스에 추가됩니다. 기본 필드 인덱스는 필드 인덱스 할당량에 포함되지 않습니다.

filterIndex와 filter 비교

filterIndex 및 filter의 차이를 이해하려면 다음 예제 쿼리를 고려합니다. 로그 그룹 4개에 대해 IPaddress에 대한 필드 인덱스를 생성했지만 다섯 번째 로그 그룹에 대해서는 생성하지 않았다고 가정합니다. filterIndex를 사용하는 다음 쿼리는 필드가 인덱싱되지 않은 로그 그룹의 스캔을 건너뜁니다. 각 인덱싱된 로그 그룹에 대해 인덱싱된 필드가 있는 로그 이벤트만 스캔하려고 시도하고 필드 인덱스가 생성된 후의 결과만 반환합니다.

```
fields @timestamp, @message
| filterIndex IPaddress = "198.51.100.0"
| limit 20
```

반대로 동일한 5개 로그 그룹의 쿼리에 filterIndex 대신 filter를 사용하는 경우 쿼리는 인덱싱된 로그 그룹의 값을 포함하는 로그 이벤트뿐만 아니라 인덱싱되지 않은 다섯 번째 로그 그룹도 스캔하려고 시도하고 해당 다섯 번째 로그 그룹의 모든 로그 이벤트를 스캔합니다.

```
fields @timestamp, @message
| filter IPaddress = "198.51.100.0"
| limit 20
```

소스

쿼리SOURCE에를 포함시키는 것은 AWS CLI 또는 API를 사용하여 쿼리를 생성할 때 쿼리에 포함할 로그 그룹 및/또는 데이터 소스를 지정하는 유용한 방법입니다. SOURCE 명령은 CloudWatch 콘솔이 아닌 AWS CLI 및 API에서만 지원됩니다. CloudWatch 콘솔을 사용하여 쿼리를 시작할 때는 콘솔 인터페이스를 사용하여 로그 그룹을 지정합니다.

쿼리 로그 그룹

SOURCE를 사용하여 쿼리할 로그 그룹을 지정하려면 다음 키워드를 사용할 수 있습니다.

- `namePrefix`는 지정한 문자열로 시작하는 이름이 있는 로그 그룹에 대해 쿼리를 실행합니다. 이를 생략하면 모든 로그 그룹이 쿼리됩니다.

  목록에 최대 5개의 접두사를 포함할 수 있습니다.
- `accountIdentifier`는 지정된 AWS 계정의 로그 그룹에 대해 쿼리를 실행합니다. 이는 모니터링 계정에서 쿼리를 실행하는 경우에만 작동합니다. 이를 생략하면 연결된 모든 소스 계정과 현재 모니터링 계정을 쿼리하는 것이 기본값입니다. 교차 계정 관찰성에 대한 자세한 내용은 CloudWatch 크로스 계정 관측성을 참조하세요.

  목록에 최대 20개의 계정 식별자를 포함할 수 있습니다.
- `logGroupClass`는 지정된 로그 클래스인 Standard 또는 Infrequent Access에 있는 로그 그룹에 대해 쿼리를 실행합니다. 이를 생략하면 기본적으로 Standard 로그 클래스가 사용됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스을 참조하세요.

이러한 방식으로 쿼리할 많은 수의 로그 그룹을 지정할 수 있으므로 생성한 필드 인덱스를 사용하는 쿼리SOURCE에만를 사용하는 것이 좋습니다. 로그 그룹에서의 필드 인덱싱에 대한 자세한 내용은 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성 섹션을 참조하세요.

다음 예제에서는 계정의 모든 로그 그룹을 선택합니다. 모니터링 계정인 경우 모니터링 및 모든 소스 계정의 로그 그룹이 선택됩니다. 총 로그 그룹의 수가 10,000개를 초과하면 다른 로그 그룹 선택 방법을 사용하여 로그 그룹의 수를 줄이라는 오류가 표시됩니다.

```
SOURCE logGroups()
```

다음 예제에서는 111122223333 소스 계정의 로그 그룹을 선택합니다. CloudWatch 크로스 계정 관측성에서 모니터링 계정에서 쿼리를 시작하면 모든 소스 계정과 모니터링 계정의 로그 그룹이 기본적으로 선택됩니다.

```
SOURCE logGroups(accountIdentifier:['111122223333'])
```

다음 예제에서는 이름 접두사를 기반으로 로그 그룹을 선택합니다.

```
SOURCE logGroups(namePrefix: ['namePrefix1', 'namePrefix2'])
```

다음 예제에서는 Infrequent Access 로그 클래스의 모든 로그 그룹을 선택합니다. `class` 식별자를 포함하지 않으면 쿼리는 기본값인 Standard 로그 클래스의 로그 그룹에만 적용됩니다.

```
SOURCE logGroups(class: ['INFREQUENT_ACCESS'])
```

다음 예제에서는 특정 이름 접두사로 시작하고 Standard 로그 클래스에 있는 111122223333 계정의 로그 그룹을 선택합니다. Standard가 기본 로그 클래스 값이므로 명령에 클래스가 언급되지 않습니다.

```
SOURCE logGroups(accountIdentifier:['111122223333'], namePrefix: ['namePrefix1',
 'namePrefix2'])
```

마지막 예제에서는 SOURCE 명령과 함께 `start-query` AWS CLI 명령을 사용하는 방법을 보여줍니다.

```
aws logs start-query
--region us-east-1
--start-time 1729728200
--end-time 1729728215
--query-string "SOURCE logGroups(namePrefix: ['Query']) | fields @message | limit 5"
```

데이터 소스 쿼리

SOURCE를 사용하여 쿼리할 데이터 소스를 지정하려면 `dataSource` 키워드를 사용할 수 있습니다. 목록에 최대 10개의 데이터 소스를 포함할 수 있습니다.

다음 예제에서는 `amazon_vpc.flow` 데이터 소스를 선택합니다.

```
SOURCE dataSource(['amazon_vpc.flow'])
```

다음 예제에서는 `amazon_vpc.flow` 데이터 소스를 선택하고 로그 그룹 이름 접두사를 기반으로 로그 그룹을 제한합니다.

```
SOURCE dataSource(['amazon_vpc.flow']) logGroups(namePrefix: ['namePrefix1'])
```

태그별 쿼리 로그 그룹

SOURCE를 사용하여 태그별로 로그 그룹을 필터링하려면 logGroupTags 함수를 사용합니다. 각각 key 및 선택적 values 배열이 있는 태그 필터 목록으로 태그를 지정합니다.

- 키가 다른 여러 태그 필터가 AND 로직과 결합됩니다.
- 동일한 태그 필터 내의 여러 값이 OR 로직과 결합됩니다.
- 와일드카드 일치*에 사용합니다. 예를 들어는 로 시작하는 값과 payment* 일치합니다payment.
- 를 부정의 접두사!로 사용합니다. 예를 들어는가 아닌 값과 !production 일치합니다production.
- 각각 최대 5개의 값을 가진 최대 5개의 태그 필터를 포함할 수 있습니다.

다음 예제에서는 태그가 지정된 모든 로그 그룹을 선택합니다team=team1 OR team=team2.

```
SOURCE logGroupTags([{"key":"team", "values":["team1", "team2"]}])
| fields @message, @timestamp
```

다음 예제에서는 service 태그가 로 시작하고 payment environment 태그가가 아닌 로그 그룹을 선택합니다production.

```
SOURCE logGroupTags([{"key":"service", "values":["payment*"]}, {"key":"environment",
 "values":["!production"]}])
| fields @message, @timestamp
```

다음 예제에서는 태그 필터링을 이름 접두사 필터와 결합합니다.

```
SOURCE logGroups(namePrefix: ['/aws/lambda']) logGroupTags([{"key":"environment",
 "values":["production"]}])
| fields @message, @timestamp
```

pattern

pattern을 사용하여 로그 데이터를 패턴으로 자동 클러스터링합니다.

패턴은 로그 필드 간에 반복되는 공유 텍스트 구조입니다. pattern을 사용하여 새로운 추세를 파악하고, 알려진 오류를 모니터링하고, 자주 발생하거나 비용이 많이 드는 로그 라인을 식별할 수 있습니다. CloudWatch Logs Insights는 로그 이벤트의 패턴을 찾고 추가로 분석하는 데 사용할 수 있는 콘솔 환경도 제공합니다. 자세한 내용은 패턴 분석 섹션을 참조하세요.

pattern 명령은 공통 패턴을 자동으로 식별하므로 로그를 검색하고 분석하기 위한 시작점으로 사용할 수 있습니다. pattern을 filter, parse 또는 sort 명령과 결합하여 더 세밀하게 조정된 쿼리에서 패턴을 식별할 수도 있습니다.

패턴 명령 입력

pattern 명령에는 @message 필드, parse 명령을 사용하여 생성한 추출된 필드 또는 하나 이상의 문자열 함수를 사용하여 조작된 문자열과 같은 입력이 필요합니다.

CloudWatch Logs는 동적 토큰이 나타내는 데이터 유형을 추론할 수 없는 경우 해당 데이터를 <Token-*number*>로 표시하며, *number*는 패턴에서 이 토큰이 나타나는 위치를 다른 동적 토큰과 비교하여 나타냅니다.

동적 토큰의 일반적인 예로는 오류 코드, IP 주소, 타임스탬프, 요청 ID가 있습니다.

패턴 명령 출력

pattern 명령으로 다음 출력이 생성됩니다.

- @pattern: 로그 이벤트 필드 간에 반복되는 공유 텍스트 구조. 요청 ID 또는 타임스탬프와 같이 패턴 내에서 서로 다른 필드는 토큰으로 표시됩니다. CloudWatch Logs는 동적 토큰이 나타내는 데이터 유형을 확인할 수 있는 경우 토큰을 <*string*-*number*>으로 표시합니다. *string*은 토큰이 나타내는 데이터 유형에 대한 설명입니다. *number*는 다른 동적 토큰과 비교하여 패턴에서 이 토큰이 나타나는 위치를 보여줍니다.

  CloudWatch Logs는 문자열을 포함하는 로그 이벤트의 콘텐츠 분석을 기반으로 이름의 문자열 부분을 할당합니다.

  CloudWatch Logs는 동적 토큰이 나타내는 데이터 유형을 추론할 수 없는 경우 해당 데이터를 <Token-*number*>로 표시하며, *number*는 패턴에서 이 토큰이 나타나는 위치를 다른 동적 토큰과 비교하여 나타냅니다.

  예를 들어, `[INFO] Request time: <Time-1> ms`은 로그 메시지 `[INFO] Request time: 327 ms`의 잠재적 출력입니다.
- @ratio: 선택한 기간 및 식별된 패턴과 일치하는 지정된 로그 그룹의 로그 이벤트 비율입니다. 예를 들어 선택한 로그 그룹 및 기간에 있는 로그 이벤트의 절반이 패턴과 일치할 경우 @ratio는 0.50을 반환합니다.
- @sampleCount: 선택한 기간 및 식별된 패턴과 일치하는 지정된 로그 그룹의 로그 이벤트의 수입니다.

- `@severityLabel`: 로그에 포함된 정보의 유형을 나타내는 로그 심각도 또는 수준입니다. 예: `Error`, `Warning`, `Info` 또는 `Debug`.

예제

다음 명령은 선택한 시간 범위 동안 지정된 로그 그룹에서 구조가 비슷한 로그를 식별하여 패턴 및 개수별로 그룹화합니다.

```
pattern @message
```

`pattern` 명령은 `filter` 명령과 함께 사용할 수 있습니다.

```
filter @message like /ERROR/
| pattern @message
```

`pattern` 명령은 `parse` 및 `sort` 명령과 함께 사용할 수 있습니다.

```
filter @message like /ERROR/
| parse @message 'Failed to do: *' as cause
| pattern cause
| sort @sampleCount asc
```

DIFF

요청한 기간에서 발견된 로그 이벤트를 길이가 동일한 이전 기간의 로그 이벤트와 비교합니다. 이렇게 하면 추세를 찾고 특정 로그 이벤트가 새로운지 여부를 확인할 수 있습니다.

`diff` 명령에 한정자를 추가하여 비교할 기간을 지정합니다.

- `diff`는 현재 선택된 시간 범위의 로그 이벤트를 바로 이전 시간 범위의 로그 이벤트와 비교합니다.
- `diff previousDay`는 현재 선택된 시간 범위의 로그 이벤트를 전날 같은 시간의 로그 이벤트와 비교합니다.
- `diff previousWeek`는 현재 선택된 시간 범위의 로그 이벤트를 이전 주의 같은 시간의 로그 이벤트와 비교합니다.
- `diff previousMonth`는 현재 선택된 시간 범위의 로그 이벤트를 전월의 같은 시간의 로그 이벤트와 비교합니다.

자세한 내용은 (diff)를 이전 시간 범위와 비교 섹션을 참조하세요.

parse

`parse`를 사용하여 로그 필드에서 데이터를 추출하고 쿼리에서 처리할 수 있는 추출된 필드를 생성합니다. `parse` 명령은 glob 표현식, 정규 표현식, logfmt 및 CSV의 네 가지 모드를 지원합니다.

`fieldName`이 생략되면 `@message`가 기본적으로 사용됩니다. 필드 이름을 첫 번째 인수로 지정하여 명명된 필드에서 구문 분석할 수 있습니다.

로그 이벤트가 지정된 패턴과 일치하지 않는 경우에도 결과에는 여전히 표시되지만 추출된 필드는 표시되지 않습니다.

글롭 모드

와일드카드(*)를 추출하려는 값의 자리 표시자로 사용하고를 사용하여 명명된 필드에 할당합니다`as`.

구문

```
parse fieldName "pattern" as alias1, alias2
```

* 와일드카드 수는 별칭 수와 같아야 합니다.

예제

```
parse @message "user=*, method:*, latency := *" as @user,
    @method, @latency | stats avg(@latency) by @method, @user
```

```
parse @logStream "*/*/*/*" as env, service, instance, shard
| stats count(*) by env, service
```

연결된 구문 분석

필드를 추출한 다음 추출된 필드를 추가로 구문 분석합니다.

```
parse @message "url=*" as url
| parse url "/api/*/users/*" as apiVersion, userId
| display apiVersion, userId
```

정규식 모드

명명된 캡처 그룹과 함께 정규식을 사용하여 필드를 추출합니다. 정규 표현식 구문에 대한 자세한 내용은 지원되는 정규식 구문 섹션을 참조하세요.

구문

```
parse fieldName /regex/
```

명명된 캡처 그룹을 사용하여 추출된 필드를 (?<*name*>...) 정의합니다.

예제

명명된 캡처 그룹을 사용하여 필드 추출

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
    latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
    @user2
```

명명된 캡처 그룹을 사용하여 VPC 흐름 로그에서 ENI 추출

```
parse @message /(?<NetworkInterface>eni-.*?) /
| display NetworkInterface, @message
```

다중 일치 모드

다중 일치 모드를 사용하여 필드에서 정규 표현식의 모든 일치 항목을 추출하여 로그 이벤트당 여러 행을 생성합니다. 정규식 패턴 `multi` 뒤에 키워드를 추가합니다.

구문

```
parse fieldName /regex/ multi
```

예제

로그 줄에서 모든 IP 주소 추출(다중 일치)

```
parse @message /(\d+\.\d+\.\d+\.\d+)/ as ip_addr multi
| stats count(*) by ip_addr
```

Logfmt 모드

`parse logfmt`를 사용하여 logfmt 형식의 로그 줄을 키-값 페어로 구문 분석합니다. Logfmt는 각 줄에 공백으로 구분된 `key=value` 페어가 포함된 구조화된 로깅 형식입니다.

구문

```
parse fieldName logfmt as alias
```

그 결과 점 표기법으로 액세스하는 맵이 생성됩니다(예: , `lf.level`lf.msg).

예제

```
parse @message logfmt as lf
| filter lf.level = "error"
| display lf.msg, lf.duration
```

```
parse @message logfmt as lf
| stats count(*) by lf.host
```

CSV 모드

`parse csv`를 사용하여 CSV 형식의 로그 선을 구조화된 필드로 구문 분석합니다. 쉼표로 구분된 각 값은 해당 별칭에 할당됩니다.

구문

```
parse fieldName csv as alias1, alias2, alias3
```

예제

```
parse @message csv as timestamp, level, message
| filter level = "ERROR"
| display timestamp, message
```

```
parse @message csv as host, method, path, status, duration
| stats avg(duration) by method
```

JSON 필드 추출

이전에 구문 분석된 객체 필드에서 명시적으로 연결된 JSON 추출에 `json field=`*fieldName* 사용합니다. 이렇게 하면 원시 메시지를 다시 구문 분석하지 않고도 구조화된 필드에서 중첩된 키를 추출할 수 있습니다.

구문

```
json field=fieldName "key.subkey" as alias
```

예제

```
parse @message /(?<payload>\{.*\})/ as payload
| json field=payload "user.name" as username
| display username
```

```
json field=requestContext "identity.sourceIp" as caller_ip
| stats count(*) by caller_ip
```

관련 필드

relevantfields를 사용하여 로그 데이터의 어떤 필드가 특정 조건과 가장 관련이 있는지 자동으로 식별합니다. 명령은 기준 로그를 조건과 일치하는 하위 집합과 비교하고 관련성 점수에 따라 순위가 지정된 필드를 반환합니다.

구문

```
relevantfields [field1, field2, ...] where condition
```

필드 목록은 선택 사항입니다. 생략하면 모든 필드가 분석됩니다.

명령은 다음 출력 필드를 반환합니다.

- @fieldName - 필드 이름
- @relevanceScore - 0~1의 척도로 관련성을 나타내는 점수
- @topRelevanceContributors - 범주형 필드의 경우 빈도가 가장 높은 항목을 표시합니다.
- @conditionalMedian - 숫자 필드의 경우 조건을 충족하는 로그의 값 중앙값
- @baselineMedian - 숫자 필드의 경우 조건을 충족하지 않는 로그의 값 중앙값

예제

모든 필드의 느린 API 응답 분석

```
relevantfields where Time > 400
```

특정 필드에 대한 포커스 분석

```
relevantfields Controller, Path, Service where Time > 400
```

Lambda 제한 시간의 동인 식별

```
relevantfields where Duration > 5000
```

확장

expand를 사용하여 JSON 배열이 포함된 필드를 만들고 배열의 각 요소에 대해 별도의 로그 이벤트를 생성합니다. 원래 로그 이벤트의 다른 모든 필드는 각 새 이벤트에서 중복됩니다.

구문

```
expand fieldName
```

예제

로그 이벤트에 `items = ["apple","banana","cherry"]` 및가 포함된 경우는 `{items: "apple", host: "web-01"}`, 및 `{items: "banana", host: "web-01"}`의 세 가지 로그 이벤트를 `host = "web-01"` `expand items` 생성합니다`{items: "cherry", host: "web-01"}`.

```
expand items
| stats count(*) by items, host
```

sort

sort를 사용하여 지정된 필드에 로그 이벤트를 오름차순(asc) 또는 내림차순(desc)으로 표시합니다. 이를 limit 명령과 함께 사용하여 '상위 N개' 또는 '하위 N개' 쿼리를 생성할 수 있습니다.

이 정렬 알고리즘은 자연 정렬의 업데이트된 버전입니다. 오름차순으로 정렬하면 다음 로직이 사용됩니다.

- 숫자가 아닌 모든 값은 모든 숫자 값 앞에 옵니다. 숫자 값은 숫자와 다른 문자의 혼합이 아니라 숫자만 포함하는 값입니다.
- 숫자가 아닌 값의 경우 알고리즘은 비교를 위해 연속된 숫자와 연속된 영문자를 별도의 청크로 그룹화합니다. 알고리즘은 숫자가 아닌 부분을 유니코드 값으로 정렬하고 숫자 부분은 먼저 길이로 정렬한 다음 숫자 값으로 정렬합니다.

유니코드 순서에 대한 자세한 내용은 유니코드 문자 목록을 참조하세요.

예를 들어, 다음은 오름차순 정렬 결과입니다.

```
!: >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> sorted by unicode order
#
*%04
0# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Alphanumeric starting with numbers
5A
111A   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  Starts with more digits than 5A, so it sorted
 to be later than 5A
2345_
@ >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 2345 is compared with @ in the unicode order,
@_
A >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Values starting with letters
A9876fghj
a12345hfh
0 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Number values
01
1
2
3
```

내림차순으로 정렬하면 정렬 결과가 반대가 됩니다.

예를 들어, Amazon VPC 흐름 로그에 대한 다음 쿼리는 호스트 전반의 상위 15개 패킷 전송을 찾습니다.

```
stats sum(packets) as packetsTransferred by srcAddr, dstAddr
    | sort packetsTransferred  desc
    | limit 15
```

통계

`stats`를 사용하여 막대형 차트, 꺾은선형 차트, 누적 영역 차트와 같은 로그 데이터의 시각화를 생성합니다. 이를 통해 로그 데이터에서 패턴을 더 효율적으로 식별할 수 있습니다. CloudWatch Logs Insights는 `stats` 함수와 집계 함수를 하나 이상 사용하는 쿼리에 대한 시각화를 생성합니다.

예를 들어, Route 53 로그 그룹의 다음 쿼리는 시간당 Route 53 레코드의 분포를 쿼리 유형별로 보여주는 시각화를 반환합니다.

```
stats count(*) by queryType, bin(1h)
```

다음 쿼리는 각 서비스에 대한 고유한 상태 값을 수집합니다.

```
stats values(status) as statuses by service
```

이러한 쿼리는 모두 막대 차트를 생성할 수 있습니다. 쿼리가 bin() 함수를 사용하여 필드별로 데이터를 그룹화하는 경우 선형 차트 및 누적 영역 차트도 볼 수 있습니다.

bin 함수에서는 다음과 같은 시간 단위 및 약어가 지원됩니다. 둘 이상의 문자를 포함하는 모든 단위 및 약어의 경우 s를 추가하여 복수화할 수 있습니다. 따라서 hr 및 hrs 모두 시간을 지정하는 데 사용됩니다.

- millisecond ms msec
- second s sec
- minute m min
- hour h hr
- day d
- week w
- month mo mon
- quarter q qtr
- year y yr

주제

- 시계열 데이터 시각화
- 필드별로 그룹화된 로그 데이터 시각화
- 단일 쿼리에서 여러 통계 명령 사용
- 통계와 함께 사용할 함수

시계열 데이터 시각화

시계열 시각화는 다음과 같은 특성을 가진 쿼리에 사용할 수 있습니다.

- 쿼리에 집계 함수가 하나 이상 포함되어 있습니다. 자세한 정보는 Aggregation Functions in the Stats Command 섹션을 참조하세요.
- 쿼리가 bin() 함수를 사용하여 필드 하나를 기준으로 데이터를 그룹화합니다.

이러한 쿼리는 선형 차트, 누적 영역 차트, 막대 차트 및 파이 차트를 생성할 수 있습니다.

예제

전체 자습서는 the section called “자습서: 시계열 시각화를 생성하는 쿼리 실행” 섹션을 참조하세요.

다음은 시계열 시각화에 사용할 수 있는 예제 쿼리입니다.

다음 쿼리는 5분마다 생성된 데이터 포인트와 함께 myfield1 필드의 평균 값에 대한 시각화를 생성합니다. 각 데이터 포인트는 이전 5분 동안 생성된 로그의 myfield1 값 평균을 집계한 것입니다.

```
stats avg(myfield1) by bin(5m)
```

다음 쿼리는 5분마다 생성된 데이터 포인트와 함께 여러 필드를 기준으로 세 값의 시각화를 생성합니다. 쿼리가 집계 함수를 포함하고 있고 그룹화 필드로 bin()을 사용하기 때문에 시각화가 생성됩니다.

```
stats avg(myfield1), min(myfield2), max(myfield3) by bin(5m)
```

선형 차트 및 누적 영역 차트 제한 사항

로그 항목 정보를 집계하지만 bin() 함수를 사용하지 않는 쿼리는 막대 차트를 생성할 수 있습니다. 그러나 이러한 쿼리는 선형 차트 또는 누적 영역 차트를 생성할 수 없습니다. 이러한 쿼리 유형에 대한 자세한 내용은 the section called “필드별로 그룹화된 로그 데이터 시각화”을 참조하세요.

필드별로 그룹화된 로그 데이터 시각화

stats 함수와 하나 이상의 집계 함수를 사용하는 쿼리에 대해 막대 차트를 생성할 수 있습니다. 자세한 정보는 Aggregation Functions in the Stats Command 섹션을 참조하세요.

시각화를 보려면 쿼리를 실행합니다. 그런 다음 시각화 탭을 선택하고 줄 옆의 화살표를 선택한 다음, 막대를 선택합니다. 막대 차트에서는 시각화가 최대 100개 막대로 제한됩니다.

예제

전체 자습서는 the section called “자습서: 로그 필드로 그룹화된 시각화를 생성하는 쿼리 실행” 섹션을 참조하세요. 다음 단락에는 필드별 시각화에 대한 더 많은 예제 쿼리가 포함되어 있습니다.

다음 VPC 흐름 로그 쿼리는 각 대상 주소에 대해 세션당 전송된 평균 바이트의 수를 찾습니다.

```
stats avg(bytes) by dstAddr
```

각 결과 값에 대해 둘 이상의 막대가 포함된 차트를 생성할 수도 있습니다. 예를 들어, 다음 VPC 흐름 로그 쿼리는 각 대상 주소에 대해 세션당 전송된 평균 및 최대 바이트의 수를 찾습니다.

```
stats avg(bytes), max(bytes) by dstAddr
```

다음 쿼리는 각 쿼리 유형에 대해 Amazon Route 53 쿼리 로그의 수를 찾습니다.

```
stats count(*) by queryType
```

단일 쿼리에서 여러 통계 명령 사용

단일 쿼리에서 여러 `stats` 명령을 사용할 수 있습니다. 이렇게 하면 첫 번째 집계의 출력에 대해 추가 집계를 수행할 수 있습니다. 쿼리에 허용되는 최대 `stats` 명령 수는 로그 그룹의 로그 클래스에 따라 다릅니다.

예: 두 개의 **`stats`** 명령을 사용한 쿼리

예를 들어 다음 쿼리는 먼저 총 트래픽 볼륨을 5분 빈에서 찾은 다음 해당 5분 빈 중에서 가장 높고 가장 낮으며 평균적인 트래픽 볼륨을 계산합니다.

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length)/1024/1024 as logs_mb BY bin(5m)
| STATS max(logs_mb) AS peak_ingest_mb,
        min(logs_mb) AS min_ingest_mb,
        avg(logs_mb) AS avg_ingest_mb
```

예: 여러 통계 명령을 다른 함수(**`filter`**, **`fields`**, **`bin`**)와 결합

여러 `stats` 명령을 및와 같은 다른 명령`filter`과 단일 쿼리`fields`로 결합할 수 있습니다. 예를 들어, 다음 쿼리는 세션에서 고유한 IP 주소의 수 및 클라이언트 플랫폼별 세션의 수를 찾고, 해당 IP 주소를 필터링한 다음, 마지막으로 클라이언트 플랫폼별 평균 세션 요청을 찾습니다.

```
STATS countDistinct(client_ip) AS session_ips,
      count(*) AS requests BY session_id, client_platform
| FILTER session_ips > 1
| STATS count(*) AS multiple_ip_sessions,
        sum(requests) / count(*) AS avg_session_requests BY client_platform
```

여러 `stats` 명령을 사용하여 쿼리에서 `bin` 및 `dateceil` 함수를 사용할 수 있습니다. 예를 들어, 다음 쿼리는 먼저 메시지를 5분 블록으로 결합한 다음 5분 블록을 10분 블록으로 집계하고 각 10분 블록 내의 최고, 최저 및 평균 트래픽 볼륨을 계산합니다.

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length) / 1024 / 1024 AS logs_mb BY BIN(5m) as @t
| STATS max(logs_mb) AS peak_ingest_mb,
        min(logs_mb) AS min_ingest_mb,
        avg(logs_mb) AS avg_ingest_mb BY dateceil(@t, 10m)
```

참고 및 제한 사항

쿼리에는 최대 10개의 stats 명령이 있을 수 있습니다. 이 할당량은 변경할 수 없습니다.

sort 또는 limit 명령을 사용하는 경우 마지막 stats 명령 뒤에 표시되어야 합니다. 마지막 stats 명령 앞에 있는 경우 쿼리가 유효하지 않습니다.

쿼리에 여러 stats 명령이 있는 경우 첫 번째 stats 집계가 완료될 때까지 쿼리의 부분 결과가 표시되지 않습니다.

단일 쿼리의 후속 stats 명령에서는 이전 stats 명령에 정의된 필드만 참조할 수 있습니다. 예를 들어, 첫 번째 stats 집계 후에는 @message 필드를 사용할 수 없으므로 다음 쿼리는 유효하지 않습니다.

```
FIELDS @message
| STATS SUM(Fault) by Operation
# You can only reference `SUM(Fault)` or Operation at this point
| STATS MAX(strlen(@message)) AS MaxMessageSize # Invalid reference to @message
```

stats 명령 다음에 참조하는 모든 필드는 해당 stats 명령에 정의되어야 합니다.

```
STATS sum(x) as sum_x by y, z
| STATS max(sum_x) as max_x by z
# You can only reference `max(sum_x)`, max_x or z at this point
```

Important

bin 함수는 항상 @timestamp 필드를 묵시적으로 사용합니다. 즉, 앞의 stats 명령을 사용하여 timestamp 필드를 전파하지 않으면 후속 stats 명령bin에서를 사용할 수 없습니다. 예를 들어, 다음 쿼리는 유효하지 않습니다.

```
FIELDS strlen(@message) AS message_length
 | STATS sum(message_length) AS ingested_bytes BY @logStream
```

```
 | STATS avg(ingested_bytes) BY bin(5m) # Invalid reference to @timestamp field
```

대신 이전 stats 명령에서 @timestamp 필드를 정의한 다음 다음 예제와 같이 후속 stats 명령dateceil에서와 함께 사용할 수 있습니다.

```
FIELDS strlen(@message) AS message_length
 | STATS sum(message_length) AS ingested_bytes, max(@timestamp) as @t BY
 @logStream
 | STATS avg(ingested_bytes) BY dateceil(@t, 5m)
```

통계와 함께 사용할 함수

CloudWatch Logs Insights는 통계 집계 함수와 통계 비집계 함수를 모두 지원합니다.

stats 명령에서 statsaggregation 함수를 사용하고 다른 함수의 인수로 사용합니다.

함수	결과 유형	설명
avg(fieldName: NumericLogField)	number	지정된 필드의 값 평균.
count() count(fieldName: LogField)	number	로그 이벤트를 계산합니다. count()(또는count(*))는 쿼리에서 반환하는 이벤트 수를 모두 세고 count(fieldName) 은 지정한 필드 이름이 포함된 레코드 수를 모두 계산합니다.
countDistinct(fieldName: LogField)	number	필드에 대해 고유한 값의 수를 반환합니다. 필드의 카디널리티가 매우 높은 경우(고유한 값이 많이 포함되어 있음) countDistinct 가 반환하는 값은 근사치입니다.
max(fieldName: LogField)	LogFieldValue	쿼리된 로그에서 이 로그 필드에 대한 최댓값입니다.
min(fieldName: LogField)	LogFieldValue	쿼리된 로그에서 이 로그 필드에 대한 최솟값입니다.

함수	결과 유형	설명
`pct(fieldName: LogFieldValue, percent: number)`	LogFieldValue	백분위수는 데이터세트에서 값의 상대적 위치를 나타냅니다. 예를 들어, `pct(@duration, 95)`는 `@duration` 의 값 중 95퍼센트가 이 값보다 낮고 5퍼센트는 이 값보다 큰 `@duration` 값을 반환합니다.
`stddev(fieldName: NumericLogField)`	number	지정된 필드의 값에 대한 표준 편차.
`sum(fieldName: NumericLogField)`	number	지정된 필드의 값에 대한 합계.
`values(fieldName: LogField)`	배열	각 그룹에 대해 지정된 필드의 고유한 값을 수집합니다. 라고도 합니다`COLLECT_VALUES` .
`variance(fieldName: NumericLogField)`	number	지정된 필드의 값의 모집단 분산입니다. 예시: `stats variance(responseTime) as rt_var by service`
`topk(k: number, fieldName: LogField)`	배열	지정된 필드의 가장 빈번한 k 값을 반환합니다. k에 유효한 값은 1~10000입니다. 이 함수는 by 또는 다른 집계 함수와 결합할 수 없습니다. 예시: `stats topk(3, status)`

통계 비집계 함수

비집계 함수를 `stats` 명령에서 사용하고 다른 함수의 인수로 사용합니다.

함수	결과 유형	설명
`earliest(fieldName: LogField)`	LogField	쿼리된 로그에서 가장 이른 타임스탬프가 있는 로그 이벤트에서 `fieldName` 의 값을 반환합니다.

함수	결과 유형	설명
`latest(fieldName: LogField)`	LogField	쿼리된 로그에서 최신 타임스탬프가 있는 로그 이벤트에서 `fieldName` 의 값을 반환합니다.
`sortsFirst(fieldName: LogField)`	LogField	쿼리된 로그에서 가장 빨리 정렬된 `fieldName` 의 값을 반환합니다.
`sortsLast(fieldName: LogField)`	LogField	쿼리된 로그에서 가장 늦게 정렬된 `fieldName` 의 값을 반환합니다.

limit

`limit`을 사용하여 쿼리에서 반환할 로그 이벤트의 수를 지정할 수 있습니다. `limit`를 생략하면 쿼리는 기본적으로 최대 10,000개의 로그 이벤트를 반환합니다. 최대 100,000의 `limit` 값을 지정할 수 있습니다.

예를 들어, 다음 예제에서는 최신 로그 이벤트 25개만 반환합니다.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

제한

나머지 데이터를 스캔하지 않고 *N*개의 일치하는 로그 이벤트가 발견되는 즉시 쿼리를 중지`limit any` *N*하려면를 사용합니다. 이렇게 하면 특정 순서 집합이 아닌 일치하는 이벤트의 샘플만 필요한 경우 스캔되는 데이터의 양을 크게 줄이고 쿼리 속도를 높일 수 있습니다.

일반를 사용하면 쿼리`limit`가 시간 범위의 모든 데이터를 스캔한 다음 상위 *N*개의 결과를 반환합니다. `limit any`를 사용하면 충분한 결과가 발견되면 쿼리가 조기에 완료됩니다. 즉, 반환된 이벤트는 시간 범위의 어느 부분에서든 발생할 수 있습니다.

예를 들어 다음 쿼리는 1개의 로그 이벤트를 반환하고 일치 항목을 찾는 즉시 스캔을 중지합니다.

```
fields @message | limit any 1
```

> **Note**
>
> 는 스캔을 조기에 `limit` any 중지하므로 결과가 특정 순서로 보장되지 않습니다. 정렬된 결과가 필요한 경우와 any 함께를 사용하지 `limit` 않고를 사용합니다`sort`.

dedup

dedup을 사용하여 지정한 필드의 특정 값을 기준으로 중복 결과를 제거합니다. 하나 이상의 필드와 함께 dedup을 사용할 수 있습니다. dedup을 사용하여 하나의 필드를 지정하면 해당 필드의 각 고유 값에 대해 하나의 로그 이벤트만 반환됩니다. 여러 필드를 지정하면 해당 필드의 고유한 값 조합마다 하나의 로그 이벤트가 반환됩니다.

중복 항목은 정렬 순서에 따라 삭제되며 정렬 순서의 첫 번째 결과만 유지됩니다. dedup 명령을 실행하기 전에 결과를 정렬하는 것이 좋습니다. dedup을 실행하기 전에 결과가 정렬되지 않으면 `@timestamp`을 사용하는 기본 내림차순 정렬 순서가 사용됩니다.

Null 값은 평가를 위해 중복으로 간주되지 않습니다. 지정된 필드에 대해 null 값이 있는 로그 이벤트는 유지됩니다. Null 값이 있는 필드를 제거하려면 `isPresent(field)` 함수를 사용하여 **`filter`**을 사용합니다.

dedup 명령 이후 쿼리에서 사용할 수 있는 유일한 쿼리 명령은 `limit`입니다.

쿼리dedup에서를 사용하면 콘솔에 Y 레코드 X 표시와 같은 메시지가 표시됩니다. 여기서 X는 중복 제거된 결과 수이고 Y는 중복 제거 전에 일치하는 총 레코드 수입니다. 이는 중복 레코드가 제거되었음을 나타내며 데이터가 누락되었음을 의미하지 않습니다.

예: 이름이 **`server`**인 필드의 각 고유 값에 대한 최신 로그 이벤트만 보기

다음 예제에서는 `server`의 각 고유 값에 대해 최신 이벤트에 대한 `timestamp`, `server`, `severity` 및 `message` 필드를 표시합니다.

```
fields @timestamp, server, severity, message
| sort @timestamp desc
| dedup server
```

CloudWatch Logs Insights 쿼리의 더 많은 샘플은 일반 쿼리 섹션을 참조하세요.

unmask

unmask를 사용하여 데이터 보호 정책으로 인해 일부 내용이 마스킹된 로그 이벤트의 모든 내용을 표시할 수 있습니다. 이 명령을 사용하려면 logs:Unmask 권한이 있어야 합니다.

로그 그룹의 데이터 보호에 대한 자세한 내용은 마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원 섹션을 참조하세요.

unnest

unnest를 사용하면 입력으로 가져온 목록을 평면화하여 목록의 각 요소에 대해 단일 레코드로 여러 레코드를 생성할 수 있습니다. 필드에 포함된 항목의 수에 따라 이 명령은 현재 레코드를 삭제하고 새 레코드를 생성합니다. 각 레코드에는 항목을 나타내는 unnested_field가 포함됩니다. 다른 모든 필드는 원본 레코드에서 가져옵니다.

unnest의 입력값은 LIST이며, 이는 jsonParse 함수에서 나옵니다. 자세한 내용은 Structure types을 참조하세요. MAP, String 및 numbers와 같은 다른 모든 유형은 unnest에서 하나의 항목이 있는 목록으로 처리됩니다.

명령 구조

다음 예제는 이 명령의 형식을 설명합니다.

```
unnest field into unnested_field
```

예제 쿼리

다음 예제에서는 JSON 객체 문자열을 구문 분석하고 필드 이벤트 목록을 확장합니다.

```
fields jsonParse(@message) as json_message
| unnest json_message.events into event
| display event.name
```

이 예제 쿼리의 로그 이벤트는 다음과 같은 JSON 문자열일 수 있습니다.

```
{
   "events": [
        {
            "name": "exception"
        },
```

```
        {
            "name": "user action"
        }
    ]
}
```

이 경우 샘플 쿼리는 쿼리 결과에서 두 개의 레코드를 생성합니다. 하나는 `event.name`을 `exception`로, 다른 하나는 `event.name`을 사용자 작업으로 생성합니다.

예제 쿼리

다음 예제에서는 목록을 평면화한 다음 항목을 필터링합니다.

```
fields jsonParse(@message) as js
| unnest js.accounts into account
| filter account.type = "internal"
```

예제 쿼리

다음 예제에서는 집계 목록을 평면화합니다.

```
fields jsonParse(trimmedData) as accounts
| unnest accounts into account
| stats sum(account.droppedSpans) as n by account.accountId
| sort n desc
| limit 10
```

lookup

`lookup`를 사용하여 조회 테이블의 참조 데이터로 쿼리 결과를 보강합니다. Amazon CloudWatch Logs에 업로드하는 CSV 데이터 또는 완료되거나 취소된 로그 쿼리의 결과로 조회 테이블을 채울 수 있습니다. 쿼리가 실행되면 `lookup` 명령은 로그 이벤트의 필드를 조회 테이블의 필드와 일치시키고 지정된 출력 필드를 결과에 추가합니다.

사용자 IDs를 사용자 세부 정보에 매핑하거나, 제품 코드를 제품 정보에 매핑하거나, 오류 코드를 오류 설명에 매핑하는 등의 데이터 보강 시나리오에는 조회 테이블을 사용합니다.

조회 테이블 생성 및 관리

쿼리에서 `lookup` 명령을 사용하려면 먼저 조회 테이블을 생성해야 합니다. CloudWatch 콘솔에서 또는 Amazon CloudWatch Logs API를 사용하여 조회 테이블을 생성하고 관리할 수 있습니다.

조회 테이블을 생성하려면(콘솔)

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택한 다음 로그 탭을 선택합니다.
3. 조회 테이블로 스크롤하고 관리를 선택합니다.
4. 조회 테이블 생성을 선택합니다.
5. 조회 테이블의 이름을 입력합니다. 이름은 영숫자와 밑줄만 포함할 수 있습니다.
6. (선택 사항) 설명을 입력합니다.
7. CSV 파일을 업로드합니다. 파일에는 열 이름이 있는 헤더 행이 포함되어야 하며 UTF-8 인코딩을 사용하고 10MB를 초과하지 않아야 합니다.
8. (선택 사항) 테이블 데이터를 암호화할 AWS KMS 키를 지정합니다.
9. 생성을 선택합니다.

조회 테이블을 생성한 후 CloudWatch Logs Insights 쿼리 편집기에서 조회 테이블을 볼 수 있습니다. 테이블 조회 탭을 선택하여 사용 가능한 테이블과 해당 필드를 찾습니다.

조회 테이블을 업데이트하려면 테이블을 선택하고 작업, 업데이트를 선택합니다. 기존 콘텐츠를 모두 바꾸려면 새 CSV 파일을 업로드합니다. 조회 테이블을 삭제하려면 작업, 삭제를 선택합니다.

CSV 데이터를 업로드하는 대신 Log Analytics 콘솔 또는 Amazon CloudWatch Logs API를 사용하여 실행한 완료 또는 취소된 로그 쿼리의 결과에서 조회 테이블을 생성하거나 업데이트할 수도 있습니다. `CreateLookupTable` 또는 `UpdateLookupTable` API의 `queryId` 파라미터를 사용하여 쿼리 ID를 지정합니다. 취소된 쿼리는 쿼리가 중지될 때 사용 가능한 부분 결과로 테이블을 채웁니다. CSV 콘텐츠 또는 쿼리 ID를 지정해야 하지만 둘 다 지정해서는 안 됩니다. 테이블 업데이트는 전체 대체 작업이며, 기존 콘텐츠는 소스에 관계없이 모두 대체됩니다. 일정에 따라 쿼리 결과가 자동으로 포함된 조회 테이블을 새로 고치려면 조회 테이블 대상과 함께 예약된 쿼리를 사용합니다. 자세한 내용은 예약된 쿼리에 대한 조회 테이블 대상 구성 섹션을 참조하세요.

> **Note**
>
> 당 계정당 최대 100개의 조회 테이블을 생성할 수 있습니다 AWS 리전. CSV 파일은 최대 10MB일 수 있습니다. Amazon CloudWatch Logs API를 사용하여 조회 테이블을 관리할 수도 있습니다. 자세한 내용은 Amazon CloudWatch Logs API 참조의 CreateLookupTable을 참조하세요. Amazon CloudWatch

> **Note**
>
> 조회 테이블이 KMS 키로 암호화된 경우 호출자는 조회 테이블을 참조하는 쿼리와 함께 `StartQuery` API를 사용하려면 키(조회 테이블을 암호화하는 데 사용되는 KMS 키)에 대한 `kms:Decrypt` 권한이 있어야 합니다. 자세한 내용은 를 사용하여 CloudWatch Logs의 조회 테이블 암호화 AWS Key Management Service 섹션을 참조하세요.

조회를 위한 쿼리 구문

명령 구조

다음은이 명령의 형식을 보여줍니다.

```
lookup table match-fields output-mode output-field[,...]
```

명령은 다음 인수를 사용합니다.

- *table* - 사용할 조회 테이블의 이름입니다.
- *match-fields* - 조회 테이블과 로그 이벤트를 일치시킬 필드를 하나 이상 지정합니다. 다음 양식 중 하나를 사용할 수 있습니다.
  - *lookup-field* `as` *log-field* `[,...]` - 조회 테이블 열 이름이 로그 이벤트 필드 이름과 다를 `as` 때 사용합니다. 예를 들어는 조회 테이블의 `ip_address` 열을 로그 이벤트의 `srcAddr` 필드와 `ip_address as srcAddr` 일치시킵니다.
  - *lookup-field* `[,...]` - 로그 이벤트 필드 이름이 조회 테이블 열 이름과 동일한 경우 필드 이름을 생략as하고 직접 지정할 수 있습니다. 예를 들어는 두 열을 동일한 이름의 로그 이벤트 필드와 `department, role` 일치시킵니다.

  여러 일치 필드를 지정하는 경우 결과를 생성하려면 조회 테이블의 행이 모든 필드와 일치해야 합니다(AND 로직).
- *output-mode* - 출력 필드가 결과에 추가되는 방법을 지정합니다. 다음 중 하나를 사용하세요.
  - `OUTPUT` - 결과에 출력 필드를 추가합니다. 동일한 이름의 필드가 로그 이벤트에 이미 있는 경우 조회 테이블 값으로 덮어씁니다. 일치하는 항목이 없으면 필드가 null로 설정됩니다.
  - `OUTPUTNEW` - 필드가 로그 이벤트에 아직 없는 경우에만 출력 필드를 결과에 추가합니다. 필드에 이미 값이 있는 경우 원래 값이 유지됩니다. 일치하는 항목이 없으면 필드는 변경되지 않습니다.
- *output-field* - 조회 테이블에서 결과에 추가할 하나 이상의 필드입니다.

예: 사용자 세부 정보로 로그 이벤트 강화

id 필드가 포함된 이벤트가 있는 로그 그룹과 id, , name email및 열이 user_data 있는 이름이 인 조회 테이블이 있다고 가정해 보겠습니다department. 다음 쿼리는 조회 테이블의 사용자 이름, 이메일 및 부서로 각 로그 이벤트를 보강합니다.

```
fields action, status, name, email, department
| lookup user_data id OUTPUT name, email, department
```

예: 집계와 함께 조회 사용

집계 함수와 함께 조회 출력 필드를 사용할 수 있습니다. 다음 쿼리는 사용자 세부 정보로 로그 이벤트를 보강한 다음 이메일 주소별로 그룹화된 이벤트를 계산합니다.

```
fields user_id, action, username, email, department
| lookup user_data user_id OUTPUT username, email, department
| stats count(*) by email
```

예: 필터와 함께 조회 사용

조회에서 반환된 필드를 기준으로 결과를 필터링할 수 있습니다. 다음 쿼리는 로그 이벤트를 보강한 다음 특정 부서의 이벤트만 표시하도록 필터링합니다.

```
fields user_id, action
| lookup user_data user_id OUTPUT username, email, department
| filter department = "Engineering"
```

예: OUTPUTNEW를 사용하여 덮어쓰지 않고 보강

로그 이벤트에 이미 hostname 필드가 포함되어 있지만 경우에 따라 비어 있는 경우 OUTPUTNEW를 사용하여 기존 필드를 덮어쓰지 않고 누락된 값을 채웁니다.

```
fields srcAddr, hostname
| lookup known_hosts ip_address as srcAddr OUTPUTNEW hostname, region
```

예: 여러 일치 필드에 조회 사용

둘 이상의 필드에서 일치시킬 수 있습니다. 다음 쿼리는 및를 조회 테이블srcAddrdstPort과 일치시켜 알려진 네트워크 서비스를 식별합니다.

```
fields @timestamp, srcAddr, dstAddr, dstPort
| lookup network_services ip_address as srcAddr, port as dstPort OUTPUT service_name,
 owner
| filter ispresent(service_name)
```

예: 일치하는 필드 이름과 함께 조회 사용

로그 이벤트 필드 이름이 조회 테이블 열 이름과 정확히 일치하면 as 키워드를 생략할 수 있습니다. 다음 쿼리는 조회 테이블department과 직접 및 role 필드를 모두 일치시킵니다.

```
fields @timestamp, department, role
| lookup employees department, role OUTPUT office, manager
```

join

소스 로그 그룹의 로그 이벤트를 일치하는 필드를 기반으로 다른 로그 그룹 또는 쿼리 결과의 이벤트와 결합합니다.

join 명령을 사용하여 일치하는 요청 식별자 또는 트랜잭션 IDs.

구문

```
join type=<join_type> left=<left_alias> right=<right_alias>
    where <left_alias>.<field>=<right_alias>.<field>
    (SOURCE <right_log_group>)
```

파라미터

- <right_log_group> - 조인할 보조 데이터 소스입니다.
- <left_alias> 및 <right_alias> - 필드를 왼쪽(기본) 및 오른쪽(보조) 데이터 소스와 구분하는 별칭입니다.
- where <field> - 조인 키로 사용되는 필드를 지정합니다. 필드는 두 데이터 소스 모두에 있어야 합니다.
- type=<join_type> (선택 사항) - 조인 유형을 지정합니다. 유효값은 다음과 같습니다.
  - inner (기본값) - 일치하는 레코드만 반환합니다.
  - left - 기본 데이터 소스의 모든 레코드와 보조 데이터 소스의 일치하는 레코드를 반환합니다.

예제

Example예제 1: API Gateway 요청을 Lambda 실행 로그와 상관관계 분석

이 예제에서는 API Gateway 액세스 로그를 Lambda 함수 로그와 조인하여 수신 요청을 백엔드 처리와 연관시키는 방법을 보여줍니다. 이는 end-to-end 요청 흐름 문제를 해결하고 특정 API 요청에 해당하는 Lambda 호출을 식별하는 데 유용합니다.

```
filter status >= 500
| join type=inner left=api right=lambda
    where api.requestId=lambda.requestId
    (SOURCE '/aws/lambda/my-function')
| fields api.requestId, api.status, api.latency, lambda.duration, lambda.memoryUsed
| sort api.latency desc
```

이 쿼리는 다음과 같습니다.

1. API Gateway 액세스 로그 및 서버 오류 필터 쿼리(상태 >= 500)
2. 두 로그 소스 모두에 표시되는 `requestId` 필드를 사용하여 Lambda 함수 로그와 조인합니다.
3. 별칭(`api` 및 `lambda`)을 사용하여 각 소스와 필드를 구분합니다.
4. Lambda 실행 기간 및 메모리 사용량과 함께 API 지연 시간을 보여주는 결합된 정보를 반환합니다.
5. API 지연 시간을 기준으로 결과를 정렬하여 가장 느린 요청을 식별합니다.

Example예제 2: 마이크로서비스 전반의 분산 트랜잭션 추적

마이크로서비스 아키텍처에서 문제를 디버깅할 때 여러 서비스에서 트랜잭션을 추적해야 하는 경우가 많습니다. 이 예제에서는 공통 트랜잭션 ID를 사용하여 서로 다른 두 서비스의 로그를 조인하는 방법을 보여줍니다.

```
filter eventType = "ORDER_CREATED"
| join type=left left=order right=payment
    where order.transactionId=payment.transactionId
    (SOURCE '/aws/lambda/payment-service')
| filter payment.eventType = "PAYMENT_PROCESSED" or !ispresent(payment.eventType)
| fields order.transactionId, order.orderId, order.customerId,
    payment.paymentStatus, payment.amount
| filter payment.paymentStatus != "SUCCESS" or !ispresent(payment.paymentStatus)
```

이 쿼리는 다음과 같습니다.

1. 주문 서비스의 주문 생성 이벤트로 시작합니다.
2. `left join`를 사용하여 결제 레코드가 일치하지 않는 주문이라도 모든 주문을 포함합니다.
3. 공유 `transactionId` 필드를 사용하여 결제 처리 이벤트에 참여합니다.
4. 최종 결과를 필터링하여 결제 실패 또는 결제 레코드 누락이 있는 주문만 표시합니다.

왼쪽 조인은 생성되었지만 시스템 장애를 나타낼 수 있는 해당 결제 이벤트가 없었던 주문을 볼 수 있기 때문에 여기서 중요합니다.

동작

- 기본 데이터 소스(왼쪽)가 먼저 처리됩니다.
- 보조 데이터 소스는 지정된 조인 키를 사용하여 평가되고 일치됩니다.
- 매칭은 조인 필드에서 등식 비교를 사용하여 수행됩니다.
- 왼쪽 조인의 경우 기본 데이터 소스의 일치하지 않는 레코드는 보조 필드에 대해 null 값으로 유지됩니다.

참고 및 제한 사항

- 등식(=) 조건만 지원됩니다.
- 쿼리당 하나의 조인 명령만 지원됩니다.
- 조인 키는 두 데이터 소스 모두에 존재해야 하며 호환되는 유형이어야 합니다.
- 조인을 사용하는 쿼리는 더 많은 데이터를 스캔하고 더 많은 비용이 발생할 수 있습니다.
- 쿼리 성능을 보장하기 위해 보조 데이터 소스의 고유 키 값 수는 50,000개로 제한됩니다.
- 조인 오른쪽에 있는 하위 쿼리는 지원되지 않습니다.

관련 명령

- 필드
- filter
- 구문 분석
- stats
- 정렬

- limit

하위 쿼리

하위 쿼리는 다른 쿼리에 대한 입력으로 사용할 수 있는 중첩된 Logs Insights 쿼리입니다. 하위 쿼리를 사용하여 중간 결과 세트를 도출한 다음 후속 명령에서 사용할 수 있습니다.

구문

필터의 하위 쿼리

```
filter <field> in (
    <subquery>
)
```

파라미터

- <subquery> - 결과 집합을 반환하는 유효한 Logs Insights 쿼리입니다. 하위 쿼리는 외부 쿼리에서 참조하는 필드를 생성해야 합니다.

예제

Example예제 1: 다운스트림 서비스에서 오류가 발생한 요청 찾기

이 예제에서는 하위 쿼리를 사용하여 기본 서비스에서 다운스트림 서비스에서 오류가 발생한 요청을 식별하는 방법을 보여줍니다. 이는 분산 시스템의 계단식 장애 문제를 해결하는 데 유용합니다.

```
filter requestId in (
    SOURCE '/aws/lambda/database-service'
    | filter errorType = "DatabaseConnectionTimeout"
    | fields requestId
)
| fields @timestamp, requestId, endpoint, userId, responseTime
| sort @timestamp desc
```

이 쿼리는 다음과 같습니다.

1. 하위 쿼리는 데이터베이스 서비스에서 연결 시간 초과가 발생한 모든 requestId 값을 찾습니다.

2. 외부 쿼리는 기본 서비스의 로그를 필터링하여 오류가 발생하기 쉬운 요청 IDs
3. 결과에는 영향을 받은 엔드포인트 및 사용자를 포함하여 다운스트림에서 실패한 요청의 전체 컨텍스트가 표시됩니다.

이 패턴은 다운스트림 장애의 업스트림 영향을 이해하는 데 도움이 됩니다.

Example예제 2: 대상 조사에 대한 자주 실패하는 요청 식별

이 예제에서는 집계와 함께 하위 쿼리를 사용하여 반복적으로 실패하는 요청을 찾는 방법을 보여줍니다. 이는 일시적인 오류가 아닌 체계적인 문제를 나타내는 경우가 많습니다.

```
filter requestId in (
    SOURCE '/aws/lambda/payment-processor'
    | filter status = "FAILED"
    | stats count(*) as failureCount by requestId
    | filter failureCount > 3
    | fields requestId
)
| fields @timestamp, requestId, customerId, amount, failureReason
| sort @timestamp asc
```

이 쿼리는 다음과 같습니다.

1. 하위 쿼리는 실패한 결제 시도를 집계하고 3회 이상 실패한 요청 IDs 식별합니다.
2. 외부 쿼리는 문제가 있는 요청 IDs
3. 재시도 진행 상황을 표시하기 위해 결과가 시간순으로 정렬됩니다.

이를 통해 일시적인 장애(단일 발생)와 심층 조사가 필요한 지속적인 문제(다중 장애)를 구별할 수 있습니다.

동작

- 하위 쿼리는 외부 쿼리와 독립적으로 실행됩니다.
- 외부 쿼리에서 사용하기 전에 결과가 구체화됩니다.
- 외부 쿼리에는 하위 쿼리에서 명시적으로 선택한 필드만 사용할 수 있습니다.

참고 및 제한 사항

- 하위 쿼리는 외부 쿼리에서 참조하는 필드를 반환해야 합니다.
- 중첩 하위 쿼리는 지원되지 않습니다.
- 하위 쿼리는 쿼리 실행 시간과 비용을 증가시킬 수 있습니다.
- 상관관계가 있는 하위 쿼리는 지원되지 않습니다.
- 내부 쿼리 실행은 30초로 제한됩니다.

관련 명령

- 필드
- filter
- 구문 분석
- stats
- 정렬
- limit

자동 회귀

`autoregress` 명령을 사용하여 필드 값의 지연된(이전 행) 복사본을 생성합니다. 이는 CloudWatch Logs Insights의 지연/납과 동일합니다.

구문

```
| autoregress field [AS alias] [p=start[-end]]
```

명령은 다음 인수를 사용합니다.

- *field* - 지연 값을 생성할 필드입니다.
- AS *alias* (선택 사항) - 출력 필드의 별칭입니다.
- p=*N* (선택 사항) - 단일 지연 깊이(기본값 1). 지연 범위에 p=*N*-*M*를 사용합니다.

예제

다음 쿼리는 4개의 지연 필드(bytes_p1~)를 생성합니다bytes_p4.

```
fields @timestamp, bytes
| autoregress bytes p=1-4
```

로그 비교

logcompare 명령을 사용하여 현재 기간을 지정된 기간만큼 뒤로 이동된 기준 기간과 비교합니다.

구문

```
| logcompare timeshift duration
```

명령은 다음 인수를 사용합니다.

- timeshift *duration* - 기준 기간(예: 7d)으로 돌아가는 기간입니다.

예제

다음 쿼리는 지난 날의 로그를 1주일 전의 동일한 기간과 비교합니다.

```
SOURCE lg start=-1d end=now
| logcompare timeshift 7d
```

채우기

filldown 명령을 사용하여 null이 아닌 마지막 값을 앞으로 전달하여 간격을 채웁니다. 필드 이름을 지정하거나 와일드카드를 사용할 수 있습니다. 필드를 지정하지 않으면 모든 필드가 채워집니다.

구문

```
| filldown [field1 [field2 ...]]
```

명령은 다음 인수를 사용합니다.

- *field* (선택 사항) - 하나 이상의 필드 이름 또는 와일드카드 패턴입니다. 생략하면 모든 필드가 채워집니다.

예제

다음 쿼리는 host 필드 및와 일치하는 모든 필드에 null이 아닌 값을 전달합니다cpu*.

```
fields @timestamp, host, cpu_user, cpu_system
| filldown host cpu*
```

채우기

`fillmissing` 명령을 사용하여 뒤에 빈 시간 빈의 행을 삽입합니다`stats ... by bin()`. 선택적으로 필드를 상수 값으로 채울 수 있습니다.

구문

```
| fillmissing [with value for field [, value for field ...]]
```

명령은 다음 인수를 사용합니다.

- `with` *`value`* `for` *`field`* (선택 사항) - 삽입된 행의 지정된 필드에 할당할 상수 값입니다.

예제

다음 쿼리는 빈 1분 빈을 `avg_latency` 필드에 대해 0으로 채웁니다.

```
fields @timestamp, latency
| stats avg(latency) as avg_latency by bin(1m)
| fillmissing with 0 for avg_latency
```

cidrlookup

`cidrlookup` 명령을 사용하여 조회 테이블의 CIDR 범위에 대해 IP 필드를 일치시켜 이벤트를 보강합니다.

구문

```
| cidrlookup table ipField [AS cidrColumn]
    OUTPUT|OUTPUTNEW lookupField [AS eventField] [, ...]
```

명령은 다음 인수를 사용합니다.

- *`table`* - CIDR 범위를 포함하는 조회 테이블의 이름입니다.
- *`ipField`* - 일치시킬 IP 주소가 포함된 이벤트 필드입니다.
- `AS` *`cidrColumn`* (선택 사항) - CIDR 범위를 포함하는 조회 테이블의 열입니다.

- OUTPUT|OUTPUTNEW - 기존 필드를 OUTPUT 덮어쓰거나 빈 필드만 채우OUTPUTNEW는 데 사용합니다.
- *lookupField* - 조회 테이블에서 이벤트에 추가할 하나 이상의 필드입니다.

예제

다음 쿼리는 네트워크 조회 테이블에서 리전, 소유자 및 데이터 센터를 사용하여 이벤트를 보강합니다.

```
fields @timestamp, srcAddr
| cidrlookup net_table srcAddr OUTPUT region, owner, datacenter
```

이상치

outlier 명령을 사용하여 사분위수 범위(IQR)를 기반으로 숫자 필드의 통계 이상값을 감지합니다. 이상치 행을 제거하거나 고정할 수 있습니다.

구문

```
| outlier [action=remove|transform] [param=n] [uselower=true|false] [mark=true|
false] field [field ...]
```

명령은 다음 인수를 사용합니다.

- action (선택 사항) - remove (특이 행을 삭제) 또는 transform (경계에 고정)입니다. 기본값: transform.
- param (선택 사항) - IQR 승수입니다. 기본값: 2.5.
- uselower (선택 사항) - 하한 특이값을 감지할지 여부입니다. 기본값: false.
- mark (선택 사항) - 부울 마커 필드를 추가할지 여부입니다. 기본값: false.
- *field* - 이상치를 확인할 하나 이상의 숫자 필드입니다.

예제

다음 쿼리는 낮은 경계 감지 및 표시와 함께 3.0 IQR 승수를 사용하여 cpu 또는 메모리가 이상치인 행을 제거합니다.

```
fields @timestamp, cpu, memory
| outlier action=remove param=3.0 uselower=true mark=true cpu memory
```

accum

`accum` 명령을 사용하여 숫자 필드의 실행 중인 누적 합계를 계산합니다.

구문

```
| accum field [AS out]
```

명령은 다음 인수를 사용합니다.

- *field* - 누적할 숫자 필드입니다.
- AS *out* (선택 사항) - 출력 필드의 별칭입니다.

예제

다음 쿼리는 실행 중인 총 바이트 수를 계산합니다.

```
fields @timestamp, bytes
| accum bytes AS totalBytes
```

appendcols

`appendcols` 명령을 사용하여 위치 행 일치를 기준으로 하위 쿼리의 열을 현재 결과에 추가합니다.

구문

```
| appendcols [override=true|false] [max=n] ( subquery )
```

명령은 다음 인수를 사용합니다.

- `override` (선택 사항) - 기존 필드를 덮어쓸지 여부입니다. 기본값: `false`.
- `max` (선택 사항) - 처리할 최대 행 수(1~100000, 기본값 10000).
- *subquery* - 괄호로 묶인 전체 CloudWatch Logs Insights 쿼리입니다.

예제

다음 쿼리는 다른 로그 그룹의 서비스당 평균 지연 시간을 추가합니다.

```
fields svc
```

```
| stats count(*) as cnt by svc
| appendcols override=true ( SOURCE lg | stats avg(latency) as avgLat by svc )
```

세션화

`sessionize` 명령을 사용하여 자격 증명 필드 및 비활성 간격별로 이벤트를 세션으로 그룹화합니다. 명령은 세션 식별자 필드를 내보냅니다.

구문

```
| sessionize field[, field ...] [maxspan duration] [AS out]
```

명령은 다음 인수를 사용합니다.

- *field* - 그룹화할 하나 이상의 자격 증명 필드입니다.
- `maxspan` *duration* (선택 사항) - 새 세션을 시작하기 전 최대 비활성 간격입니다. 기본값: `30m`.
- `AS` *out* (선택 사항) - 출력 필드 이름입니다. 기본값: `session_id`.

예제

다음 쿼리는 1시간의 비활성 간격이 있는 사용자 및 디바이스별로 이벤트를 세션으로 그룹화합니다.

```
fields @timestamp, userId, deviceId
| sessionize userId, deviceId maxspan 1h as my_session
```

countFrequent

`countFrequent` 명령(로도 사용 가능`count_frequent`)을 사용하여 각 고유 필드-값 조합의 대략적인 개수를 내림차순으로 정렬하여 계산합니다. 명령은 `_approxcount` 필드를 내보냅니다.

구문

```
| countFrequent field[, field ...]
```

명령은 다음 인수를 사용합니다.

- *field* - 빈도를 계산할 하나 이상의 필드입니다.

예제

다음 쿼리는 메서드 및 상태 조합의 빈도를 계산합니다.

```
fields method, status
| countFrequent method, status
```

where

where 명령을 명령의 별칭으로 사용합니다filter. 동일한 구문과 동작을 허용합니다.

구문

```
| where condition
```

명령은 다음 인수를 사용합니다.

- *condition* - filter 명령이 허용하는 것과 동일한 부울 표현식입니다.

예제

다음 쿼리는 "오류"가 포함된 로그 이벤트를 필터링합니다.

```
fields @timestamp, @message
| where @message like /error/
```

추정

estimate 명령을 사용하여 쿼리를 실행하지 않고 선택한 로그 그룹 및 시간 범위에서 쿼리가 스캔할 예상 바이트를 반환합니다. 이 명령에서 반환된 예상 바이트를 사용하여 쿼리를 실행하기 전에 로그 그룹 선택, 시간 범위 및 필터를 세분화할 수 있습니다. estimate 명령에는 CloudWatch Logs Insights 쿼리 요금이 발생하지 않습니다.

estimate 명령은 쿼리의 마지막 명령이어야 합니다.

> **Note**
>
> 가 estimate 반환하는 값은 근사치이며 쿼리를 실행할 때 스캔한 실제 데이터 볼륨과 다를 수 있습니다.

구문

```
| estimate
```

예제

다음 쿼리는 쿼리가 실행하지 않고 스캔할 데이터 볼륨의 추정치를 바이트 단위로 반환합니다.

```
fields @timestamp, @message
| filter @message like /error/
| estimate
```

부울, 비교, 숫자, 날짜/시간 및 기타 함수

CloudWatch Logs Insights는 다음 섹션에 설명된 것처럼 쿼리에서 다른 많은 연산과 함수를 지원합니다.

주제

- 산술 연산자
- 부울 연산
- 비교 연산자
- 숫자 연산자
- 구조 유형
- 날짜/시간 함수
- 일반 함수
- JSON 함수
- IP 주소 문자열 함수
- 문자열 함수

산술 연산자

산술 연산자에서는 숫자 데이터 형식을 인수로 수락하고 숫자 결과를 반환합니다. 산술 연산자를 `filter` 및 `fields` 명령에서 사용하고 다른 함수의 인수로 사용합니다.

연산	설명
`a + b`	덧셈

연산	설명
`a - b`	뺄셈
`a * b`	곱셈
`a / b`	나눗셈
`a ^ b`	거듭제곱(`2 ^ 3`에서 8 반환)
`a % b`	나머지 또는 모듈러스(`10 % 3`에서 1 반환)

부울 연산

부울 연산자 **`and`**, **`or`** 및 **`not`**을 사용합니다.

> **Note**
>
> TRUE 또는 FALSE의 값을 반환하는 함수에서만 부울 연산자를 사용합니다.

비교 연산자

비교 연산자에서는 모든 데이터 형식을 인수로 수락하고 부울 결과를 반환합니다. 비교 연산은 `filter` 명령에서 사용하고 다른 함수의 인수로 사용합니다.

연산자	설명
`=`	같음
`!=`	같지 않음
`<`	보다 작음
`>`	보다 큼
`<=`	작거나 같음
`>=`	크거나 같음

숫자 연산자

숫자 연산은 숫자 데이터 형식을 인수로 수락하고 숫자 결과를 반환합니다. 숫자 연산은 `filter` 및 `fields` 명령에서 사용하고 다른 함수의 인수로 사용합니다.

연산	결과 유형	설명
`abs(a: number)`	number	절대값
`ceil(a: number)`	number	천장값으로 반올림(a의 값보다 큰 수 중 가장 작은 정수)
`floor(a: number)`	number	바닥값으로 반올림(a 값보다 작은 수 중 가장 큰 정수)
`greatest(a: number, ...numbers: number[])`	number	가장 큰 값 반환
`least(a: number, ...numbers: number[])`	number	가장 작은 값 반환
`log(a: number)`	number	자연 로그
`round(a: number [, d: number])`	number	값을 반올림합니다a. 인수 하나를 사용하여 가장 가까운 정수로 반올림합니다. 인수가 두 개인 경우 소수d점 자리까지 반올림합니다.
`sqrt(a: number)`	number	제곱근
`haversine(lat1: number, lon1: number, lat2: number, lon2: number)`	number	위도와 경도로 지정된 두 지리적 지점 사이의 큰 원 거리를 킬로미터 단위로 계산합니다.

연산	결과 유형	설명
`toNumber(fieldName: string)`	number	문자열 필드 값을 숫자 값으로 변환합니다.
`toInt(fieldName: string)`	number	필드 값을 정수(32비트)로 변환합니다.
`toLong(fieldName: string)`	number	필드 값을 긴 정수(64비트)로 변환합니다.
`toDouble(fieldName: string)`	number	필드 값을 이중 정밀도 부동 소수점 숫자로 변환합니다.

구조 유형

맵 또는 목록은 쿼리에 속성을 액세스하고 사용할 수 있는 CloudWatch Logs Insights의 구조 유형입니다.

예: 맵 또는 목록을 가져오는 방법

`jsonParse`를 사용하여 json 문자열인 필드를 맵 또는 목록으로 구문 분석합니다.

```
fields jsonParse(@message) as json_message
```

예: 속성에 액세스하려면

점 액세스 연산자(map.attribute)를 사용하여 맵의 항목에 액세스합니다. 맵의 속성에 특수 문자가 포함된 경우 백틱을 사용하여 속성 이름(map.attributes.`special.char`)을 묶습니다.

```
fields jsonParse(@message) as json_message
| stats count() by json_message.status_code
```

브래킷 액세스 연산자(list[index])를 사용하여 목록 내 특정 위치에서 항목을 검색합니다.

```
fields jsonParse(@message) as json_message
| filter json_message.users[1].action = "PutData"
```

키 이름에 특수 문자가 있는 경우 특수 문자를 백틱(``)으로 래핑합니다.

```
fields jsonParse(@message) as json_message
| filter json_message.`user.id` = "123"
```

예: 빈 결과

맵과 목록은 문자열, 숫자 및 날짜/시간 함수에 대해 null로 처리됩니다.

```
fields jsonParse(@message) as json_message
| display toupper(json_message)
```

맵 및 목록을 다른 필드와 비교하면 `false`가 됩니다.

> **Note**
>
> `dedup`, `pattern`, `sort` 및 `stats`에서 맵 및 목록을 사용하는 것은 지원되지 않습니다.

날짜/시간 함수

날짜/시간 함수

날짜/시간 함수를 `fields` 및 `filter` 명령에서 사용하고 다른 함수의 인수로 사용합니다. 이러한 함수를 사용하여 집계 함수가 포함된 쿼리에 대한 시간 버킷을 생성합니다. 숫자와 다음 중 하나로 구성된 기간을 사용합니다.

- `ms` - 밀리초
- `s` - 초
- `m` - 분
- `h` - 시간

예를 들어, `10m`은 10분을, `1h`은 1시간을 나타냅니다.

> **Note**
>
> datetime 함수에 가장 적합한 시간 단위를 사용합니다. CloudWatch Logs는 선택한 시간 단위에 따라 요청의 한도를 지정합니다. 예를 들어, CloudWatch Logs는 `s`을 사용하는 모든 요청의 최대값을 60으로 한정합니다. 따라서 `bin(300s)`을 지정하면 CloudWatch Logs는 실제로 이

를 60초로 구현합니다. 60은 분당 초의 수이므로 CloudWatch Logs는 s에 60보다 큰 숫자를 사용하지 않기 때문입니다. 5분 버킷을 생성하려면 bin(5m)을 대신 사용합니다.
ms의 한도는 1000이고, s와 m의 한도는 60이며, h의 한도는 24입니다.

다음 표에는 쿼리 명령에 사용할 수 있는 다양한 날짜 시간 함수의 목록이 포함되어 있습니다. 이 표에는 각 함수의 결과 유형이 나열되며 각 함수에 대한 설명이 포함되어 있습니다.

Tip

쿼리 명령을 생성할 때 시간 간격 선택기를 사용하여 쿼리할 기간을 선택할 수 있습니다. 예를 들어, 5~30분 간격, 1, 3, 12시간 간격 또는 사용자 지정 시간 범위 중에서 설정할 수 있습니다. 특정 날짜 사이의 기간을 설정할 수도 있습니다.

함수	결과 유형	설명

bin(period: Period)	타임스탬프	@timestamp 값을 지정한 기간으로 반올림한 다음 자릅니다. 예를 들어, bin(5m)은 @timestamp 의 값을 가장 가까운 5분 단위로 반올림합니다.
		
		이를 사용하여 쿼리에서 여러 로그 항목을 함께 그룹화할 수 있습니다. 다음 예제에서는 시간당 발생한 예외 수를 반환합니다.
		
		`filter @message like /Exception/`
		`  | stats count(*) as exceptionCount`
		`by bin(1h)`
		`  | sort exceptionCount desc`
		
		bin 함수에서는 다음과 같은 시간 단위 및 약어가 지원됩니다. 둘 이상의 문자를 포함하는 모든 단위 및 약어의 경우 s를 추가하여 복수화할 수 있습니다. 따라서 hr 및 hrs 모두 시간을 지정하는 데 사용됩니다.
		
		• millisecond ms msec

함수	결과 유형	설명
		• second s sec • minute m min • hour h hr • day d • week w • month mo mon • quarter q qtr • year y yr
datefloor(timestamp: Timestamp, period: Period)	타임스탬프	타임스탬프를 지정한 기간으로 자릅니다. 예를 들어, datefloor(@timestamp, 1h) 은 @timestamp 의 모든 값을 해당 시간 아래로 자릅니다.
dateceil(timestamp : Timestamp, period: Period)	타임스탬프	타임스탬프를 지정한 기간으로 반올림한 다음 자릅니다. 예를 들어, dateceil(@timestamp, 1h) 은 @timestamp 의 모든 값을 해당 시간 위로 자릅니다.
fromMillis(fieldName: number)	타임스탬프	입력 필드를 Unix 에포크 밀리초로 해석하여 타임스탬프로 변환합니다.
toMillis(fieldName: Timestamp)	number	지정된 필드에 있는 타임스탬프를 Unix Epoch 밀리초를 나타내는 숫자로 변환합니다. 예를 들어, toMillis(@timestamp) 은 타임스탬프를 2022-01-14T13:18:031.000-08:00 에서 1642195111000 로 변환합니다.

함수	결과 유형	설명

now()	number	쿼리 처리가 시작된 시간을 epoch 초 단위로 반환합니다. 이 함수는 인수를 필요로 하지 않습니다.
		
		이를 사용하여 현재 시간에 따라 쿼리 결과를 필터링할 수 있습니다.
		
		예를 들어, 다음 쿼리는 지난 2시간 동안의 모든 4xx 오류를 반환합니다.
		
		`parse @message "Status Code: *;" as statusCode\n`
		`| filter statusCode >= 400 and statusCode <= 499 \n`
		`| filter toMillis(@timestamp) >= (now() * 1000 - 7200000)`
		
		다음 예제에서는 `error` 또는 `failure` 단어가 포함된 지난 5시간 동안의 모든 로그 항목을 반환합니다.
		
		`fields @timestamp, @message`
		`| filter @message like /(?i)(error|failure)/`
		`| filter toMillis(@timestamp) >= (now() * 1000 - 18000000)`

`parseDate(fieldName: string, format: string [, timezone: string])`	number	Java DateTimeFormatter 패턴을 사용하여 날짜 문자열을 밀리초 단위로 구문 분석합니다. 선택적 `timezone` 인수는 구문 분석에 사용할 시간대를 지정합니다. 예시: `fields parseDate(field, "yyyy-MM-dd", "UTC") as epoch`

함수	결과 유형	설명
`formatDate(timestamp: LogField, format: string [, timezone: string])`	문자열	strftime 스타일 형식 문자열을 사용하여 타임스탬프의 형식을 지정합니다. 별칭으로도 사용할 수 있습니다`strftime`. 선택적 `timezone` 인수는 서식 지정을 위한 시간대를 지정합니다. 예시: `fields formatDate(@timestamp, "%Y-%m-%d", "UTC") as date`

> **Note**
>
> 현재 CloudWatch Logs Insights는 사람이 읽을 수 있는 타임스탬프가 있는 로그 필터링을 지원하지 않습니다.

일반 함수

일반 함수

일반 함수를 `fields` 및 `filter` 명령에서 사용하고 다른 함수의 인수로 사용합니다.

함수	결과 유형	설명
`ispresent(fieldName: LogField)`	부울	이 필드가 존재하는 경우 `true` 반환
`coalesce(fieldName: LogField, ...fieldNames: LogField[])`	LogField	목록에서 null이 아닌 첫 번째 값 반환
`case(cond1, val1, cond2, val2, ..., [default])`	LogField	조건을 순서대로 평가하고 첫 번째 true 조건에 대한 값을 반환합니다. 조건이 true가 아니고 기본값이 제공된 경우는 기본값을 반환합니다. 최대 10개의 브랜치를 지원합니다.

함수	결과 유형	설명
`if(condition: Boolean, trueValue: LogField, falseValue: LogField)`	LogField	조건을 평가하고 조건이 true 인 `trueValue` 경우 또는 `falseValue` 그렇지 않은 경우를 반환합니다.
`isNumeric(fieldName: LogField)`	부울	필드 값을 숫자로 구문 분석할 수 있는 `true` 경우를 반환합니다. 예시: `filter isNumeric (@duration)`
`messageSize(fieldName: LogField)`	number	문자열 필드의 바이트 길이를 반환합니다. 예시: `fields messageSize(@message) as size`
`queryStartTime()`	number	쿼리 기간 시작 시간을 epoch 밀리초로 반환합니다.
`queryEndTime()`	number	쿼리 기간 종료 시간을 epoch 밀리초로 반환합니다.
`queryTimeRange()`	number	쿼리 기간 지속 시간을 밀리초 단위로 반환합니다.

JSON 함수

JSON 함수

JSON 함수를 `fields` 및 `filter` 명령과 다른 함수의 인수로 사용합니다.

함수	결과 유형	설명
`jsonParse(fieldName: string)`	맵 | 목록 | 비어 있음	입력이 JSON 객체 또는 JSON 배열의 문자열 표현일 때 맵 또

함수	결과 유형	설명
		는 목록을 반환합니다. 입력이 표현 중 하나가 아닌 경우 빈 값을 반환합니다.
`jsonStringify(fieldName: Map | List)`	문자열	맵 또는 목록 데이터에서 JSON 문자열을 반환합니다.
`jsonArraySize(fieldName: LogField)`	number	JSON 배열 문자열 필드의 요소 수를 반환합니다. 예시: `fields jsonArraySize(Operation) as arrayLen`
`jsonArrayContains(fieldName: LogField, value: string)`	부울	필드의 JSON 배열에 지정된 값이 포함된 경우 `true` (1)을 반환합니다. 잘못된 JSON, 비배열 또는 빈 배열에 대해 `false` (0)을 반환합니다. 대/소문자 구분 비교를 사용합니다. 예시: `filter jsonArrayContains(@roles, "admin")`

IP 주소 문자열 함수

IP 주소 문자열 함수

IP 주소 문자열 함수를 `filter` 및 `fields` 명령에서 사용하고 다른 함수의 인수로 사용합니다.

함수	결과 유형	설명
`isValidIp(fieldName: string)`	부울	필드가 유효한 IPv4 또는 IPv6 주소인 경우 `true`을 반환합니다.

함수	결과 유형	설명
`isValidIpV4(fieldName: string)`	부울	필드가 유효한 IPv4 주소인 경우 `true`을 반환합니다.
`isValidIpV6(fieldName: string)`	부울	필드가 유효한 IPv6 주소인 경우 `true`을 반환합니다.
`isIpInSubnet(fieldName: string, subnet: string)`	부울	필드가 지정된 v4 또는 v6 서브넷 내 유효한 IPv4 또는 IPv6 주소인 경우 `true`을 반환합니다. 서브넷을 지정할 때는 `192.0.2.0/24` 또는 `2001:db8::/32` 와 같은 CIDR 표기법을 사용합니다. 여기서는 `192.0.2.0` 또는`2001:db8::` 이 CIDR 블록의 시작입니다.
`isIpv4InSubnet(fieldName: string, subnet: string)`	부울	필드가 지정된 v4 서브넷 내 유효한 IPv4 주소인 경우 `true`을 반환합니다. 서브넷을 지정할 때는 `192.0.2.0/24` 와 같은 CIDR 표기법을 사용합니다. 여기서는 `192.0.2.0` 이 CIDR 블록의 시작입니다.
`isIpv6InSubnet(fieldName: string, subnet: string)`	부울	필드가 지정된 v6 서브넷 내 유효한 IPv6 주소인 경우 `true`을 반환합니다. 서브넷을 지정할 때는 `2001:db8::/32` 와 같은 CIDR 표기법을 사용합니다. 여기서는 `2001:db8::` 이 CIDR 블록의 시작입니다.
`ipv4ToNumber(fieldName: string)`	number	IPv4 주소 문자열을 숫자 표현으로 변환합니다.
`isPrivateIP(fieldName: string)`	부울	IP 주소가 프라이빗 범위(RFC 1918)에 있는 `true` 경우를 반환합니다.
`isPublicIP(fieldName: string)`	부울	IP 주소가 공개적으로 라우팅 가능한 `true` 경우를 반환합니다.
`isReservedIP(fieldName: string)`	부울	IP 주소가 예약 범위에 있는 `true` 경우를 반환합니다.

문자열 함수

문자열 함수

문자열 함수를 `fields` 및 `filter` 명령에서 사용하고 다른 함수의 인수로 사용합니다.

함수	결과 유형	설명
`isempty(fieldName: string)`	숫자	필드가 누락되어 있거나 빈 문자열인 경우 1을 반환합니다.
`isblank(fieldName: string)`	숫자	필드가 누락되어 있거나 빈 문자열이거나 빈 공백만 포함된 경우 1을 반환합니다.
`concat(str: string, ...strings: string[])`	문자열	문자열을 연결합니다.

`ltrim(str: string)`
 	
`ltrim(str: string, trimChars: string)`
	문자열	함수에 두 번째 인수가 없는 경우에는 문자열의 왼쪽에서 공백을 제거합니다. 함수에 두 번째 문자열 인수가 없는 경우에는 공백을 제거하지 않습니다. 대신 `str`의 왼쪽에서 `trimChars` 의 문자를 제거합니다. 예를 들어, `ltrim("xyZxyfooxyZ","xyZ")` 은 `"fooxyZ"`을 반환합니다.

`rtrim(str: string)`
 	
`rtrim(str: string, trimChars: string)`
	문자열	함수에 두 번째 인수가 없는 경우에는 문자열의 오른쪽에서 공백을 제거합니다. 함수에 두 번째 문자열 인수가 없는 경우에는 공백을 제거하지 않습니다. 대신 `str`의 오른쪽에서 `trimChars` 의 문자를 제거합니다. 예를 들어, `rtrim("xy`

함수	결과 유형	설명
		ZfooxyxyZ","xyZ") 은 "xyZfoo"을 반환합니다.
trim(str: string) trim(str: string, trimChars: string)	문자열	함수에 두 번째 인수가 없는 경우에는 문자열의 양쪽 끝에서 공백을 제거합니다. 함수에 두 번째 문자열 인수가 없는 경우에는 공백을 제거하지 않습니다. 대신 str의 양쪽에서 trimChars 의 문자를 제거합니다. 예를 들어, trim("xyZxyfooxyxyZ","xyZ") 은 "foo"을 반환합니다.
strlen(str: string)	number	문자열 길이를 Unicode 코드 포인트로 반환합니다.
toupper(str: string)	문자열	문자열을 대문자로 변환합니다.
tolower(str: string)	문자열	문자열을 소문자로 변환합니다.
substr(str: string, startIndex: number) substr(str: string, startIndex: number, length: number)	문자열	숫자 인수가 지정한 인덱스의 하위 문자열을 문자열 끝에 반환합니다. 함수에 두 번째 숫자 인수가 있는 경우 해당 인수에는 검색되는 하위 문자열의 길이가 포함됩니다. 예를 들어, substr("xyZfooxyZ",3, 3) 은 "foo"을 반환합니다.

함수	결과 유형	설명
`replace(fieldName: string, searchValue: string, replaceValue: string)`	문자열	`fieldName: string` 에서 `searchValue` 의 모든 인스턴스를 `replaceValue` 로 바꿉니다. 예를 들어, `replace(logGroup,"smoke_test","Smoke")` 함수는 `logGroup` 필드에 문자열 값 `smoke_test` 를 포함한 로그 이벤트를 검색하고 해당 값을 `Smoke` 문자열로 바꿉니다.
`regexReplace(fieldName: string, pattern: string, replacement: string)`	문자열	정규식과 일치하는 모든 하위 문자열을 `pattern`로 바꿉니다`replacement` . RE2 정규식 구문을 사용합니다.

`strcontains(str: string, searchValue: string)`
 	
`strcontains(str: string, searchValue: string, caseInsensitive: boolean)`
	number	`str`에 `searchValue` 와 0이 포함되어 있으면 1을 반환합니다. 세 번째 파라미터가 로 설정된 경우 일치 항목은 대/소문자`true`를 구분하지 않습니다.

`startsWith(str: string, searchValue: string)`	number	가 로 `str` 시작하면 1을 반환`searchValue` 하고 그렇지 않으면 0을 반환합니다.
`endsWith(str: string, searchValue: string)`	number	가 로 `str` 끝나는 경우 1을 반환`searchValue` 하고 그렇지 않으면 0을 반환합니다.
`urlencode(str: string)`	문자열	URL - 문자열을 인코딩합니다.

함수	결과 유형	설명
`urldecode(str: string)`	문자열	문자열을 URL로 디코딩합니다.
`base64encode(str: string)`	문자열	Base64-encodes.
`base64decode(str: string)`	문자열	Base64-decodes.
`split(str: string, delimiter: string)`	배열	지정된 구분 기호로 문자열을 분할하고 하위 문자열의 배열을 반환합니다.
`hexToAscii(value: string)`	문자열	16진수 문자열을 ASCII 텍스트로 변환합니다. 예시: `fields hexToAscii("48656c6c6f") as text`
`hexToDec(value: string)`	number	16진수 문자열을 10진수로 변환합니다. 예시: `fields hexToDec("0xff") as dec`
`decToHex(value: number)`	문자열	10진수 정수를 `0x` 접두사가 있는 소문자 16진수 문자열로 변환합니다. 음수는 `-0x` 접두사를 생성합니다. 비통합자는 잘립니다. 예시: `fields decToHex(255) as hex`

해시 함수

`fields` 및 `filter` 명령에서 해싱 함수를 사용할 수 있습니다.

함수	결과 유형	설명
`md5(fieldName: string)`	문자열	문자열 값의 MD5 해시를 계산합니다.
`sha256(fieldName: string)`	문자열	문자열 값의 SHA-256 해시를 계산합니다.

시계열 함수

`stats` 명령과 함께 시계열 함수를 사용하여 시간 기간 동안의 지표와 컴퓨팅 변화율을 분석할 수 있습니다.

함수	결과 유형	설명
`rate(fieldName: NumericLogField, interval: Period)`	number	숫자 필드의 간격당 변경 속도를 계산합니다.
`countOverTime(fieldName: LogField)`	number	시간 빈당 로그 이벤트를 계산합니다. 와 함께 `by bin(interval)` 를 사용하여 창을 설정합니다.
`sumOverTime(fieldName: NumericLogField)`	number	시간 빈당 필드 값을 합산합니다. 와 함께 `by bin(interval)` 를 사용하여 창을 설정합니다.
`histogram(fieldName: NumericLogField, buckets: number)`	map	숫자 필드 값을 지정된 동일 너비 범위 수로 버킷팅하고 배포를 반환합니다.

offset

`stats ... by bin()` 절 끝에 `offset` 있는를 사용하여 시계열 빈을 지정된 기간만큼 이동합니다. 이렇게 하면 현재 지표를 이전 시간 또는 날짜의 동일한 기간과 비교하는 등 시간 이동 비교가 가능합니다.

구문

```
stats <aggregation> by bin(<period>) offset <duration>
```

예제

```
stats count(*) by bin(5m) offset 1h
```

```
stats avg(latency) by bin(1m) offset 1d
```

특수 문자가 포함된 필드

필드에 @ 기호 또는 마침표(.) 이외의 영숫자가 아닌 문자가 포함된 경우 필드를 백틱 문자(`)로 둘러싸야 합니다. 예를 들어, foo-bar 로그 필드는 영숫자가 아닌 문자 하이픈(-)을 포함하고 있기 때문에 백틱(`foo-bar`)으로 묶어야 합니다.

로 시작하고 나머지 이름에 특수 문자가 @ 포함된 시스템 필드의 경우 백틱 외부에 @ 기호를 배치하고 필드 이름 부분만 묶습니다. 예를 들어 리소스 태그 필드는 패턴을 사용합니다@aws.tag.<tagKey>. 태그 키에 콜론과 같은 특수 문자가 포함된 경우 다음 구문을 사용해야 합니다.

```
@`aws.tag.aws:cloudformation:stack-name`
```

다음 예제에서는이 구문을 사용하는 필터 쿼리를 보여줍니다.

```
filter @`aws.tag.aws:cloudformation:stack-name` = "my-stack"
```

쿼리에 별칭 및 코멘트 사용

별칭이 포함된 쿼리를 생성합니다. 별칭은 로그 필드의 이름을 바꾸거나 값을 필드로 추출할 때 사용합니다. 키워드 as를 사용하여 로그 필드 또는 결과에 별칭을 제공합니다. 쿼리에서 별칭을 2개 이상 사용할 수 있습니다. 다음과 같은 명령에서 별칭을 사용할 수 있습니다.

- `fields`

- parse
- sort
- stats

다음 예제에서는 별칭이 있는 쿼리를 생성하는 방법을 보여줍니다.

예제

쿼리의 명령 fields에 별칭이 포함되어 있습니다.

```
fields @timestamp, @message, accountId as ID
| sort @timestamp desc
| limit 20
```

쿼리를 통해 @timestamp, @message 및 accountId 필드에 대한 값이 반환됩니다. 결과는 내림차순으로 정렬되며 20개로 제한됩니다. accountId에 대한 값이 ID라는 별칭 아래에 나열됩니다.

예제

쿼리의 명령 sort 및 stats에 별칭이 포함되어 있습니다.

```
stats count(*) by duration as time
| sort time desc
```

쿼리에서는 로그 그룹에서 발생하는 duration 필드의 수를 계산하고 결과를 내림차순으로 정렬합니다. duration에 대한 값이 time라는 별칭 아래에 나열됩니다.

코멘트 사용

CloudWatch Logs Insights는 쿼리의 코멘트를 지원합니다. 해시 문자(#)를 사용하여 코멘트를 시작합니다. 코멘트를 사용하여 쿼리 또는 문서 쿼리의 줄을 무시할 수 있습니다.

예: 쿼리

다음 쿼리가 실행되면 두 번째 줄이 무시됩니다.

```
fields @timestamp, @message, accountId
# | filter accountId not like "7983124201998"
| sort @timestamp desc
| limit 20
```

Logs Insights QL 시작하기: 쿼리 자습서

다음 섹션에는 Logs Insights QL을 시작하는 데 도움이 되는 샘플 쿼리 자습서가 포함되어 있습니다.

주제

- 자습서: 샘플 쿼리 실행 및 수정
- 자습서: 집계 함수를 사용하여 쿼리 실행
- 자습서: 로그 필드로 그룹화된 시각화를 생성하는 쿼리 실행
- 자습서: 시계열 시각화를 생성하는 쿼리 실행

자습서: 샘플 쿼리 실행 및 수정

다음 자습서에서는 사용자가 CloudWatch Logs Insights를 시작할 수 있도록 도와줍니다. Log Insights QL에서 샘플 쿼리를 실행한 다음, 수정하고 다시 실행하는 방법을 봅니다.

쿼리를 실행하려면 CloudWatch Logs에 이미 저장된 로그가 있어야 합니다. CloudWatch Logs를 이미 사용 중이고 로그 그룹과 로그 스트림이 설정되어 있으면 시작할 준비가 된 것입니다. 또한 AWS CloudTrail, Amazon Route 53 또는 Amazon VPC 등의 서비스를 사용하고 있고 이러한 서비스의 로그가 CloudWatch Logs로 전송되도록 설정한 경우에는 이미 로그가 있을 수도 있습니다. CloudWatch Logs로 로그 전송에 대한 자세한 내용은 CloudWatch Logs 시작하기을 참조하세요.

CloudWatch Logs Insights의 쿼리가 로그 이벤트에서 일련의 필드를 반환하거나 로그 이벤트에 대해 수행된 수학적 집계 또는 다른 연산의 결과를 반환합니다. 이 자습서에서는 로그 이벤트 목록을 반환하는 쿼리를 보여줍니다.

샘플 쿼리 실행

CloudWatch Logs Insights 샘플 쿼리를 실행하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.

   로그 인사이트 페이지의 쿼리 편집기에는 Logs Insights QL에 최신 로그 이벤트 20개를 반환하는 쿼리가 포함되어 있습니다.
3. 로그 그룹 선택 드롭다운에서 쿼리할 로그 그룹을 하나 이상 선택합니다.

   CloudWatch 크로스 계정 관측성에서 모니터링 계정인 경우 소스 계정과 모니터링 계정에서 로그 그룹을 선택할 수 있습니다. 하나의 쿼리로 다른 계정의 로그를 한 번에 쿼리할 수 있습니다.

로그 그룹 이름, 계정 ID 또는 계정 레이블을 기준으로 로그 그룹을 필터링할 수 있습니다.

Standard 로그 클래스의 로그 그룹을 선택하면 CloudWatch Logs Insights가 로그 그룹의 데이터 필드를 자동으로 탐지합니다. 검색된 필드를 보려면 페이지 오른쪽 상단의 필드 메뉴를 선택합니다.

> **Note**
>
> 검색된 필드는 Standard 로그 클래스의 로그 그룹에 대해서만 지원됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스를 참조하세요.

4. (선택 사항) 시간 간격 선택기를 사용하여 쿼리할 기간을 선택합니다.

   5~30분 간격, 1, 3, 12시간 간격 또는 사용자 지정 시간 범위 중에서 선택할 수 있습니다.

5. 실행을 선택하고 결과를 봅니다.

   이 자습서에서는 가장 최근에 추가된 로그 이벤트 20개가 결과에 포함되어 있습니다.

   CloudWatch Logs는 이 로그 그룹의 로그 이벤트를 시간의 흐름에 따라 보여주는 막대 그래프를 표시합니다. 이 막대 그래프는 테이블의 이벤트뿐만 아니라 쿼리 및 시간 범위와 일치하는 로그 그룹 내 이벤트의 분포를 보여줍니다.

6. 반환된 로그 이벤트의 모든 필드를 보려면 번호가 매겨진 이벤트의 왼쪽에 있는 삼각형 드롭다운 아이콘을 선택합니다.

샘플 쿼리 수정

이 자습서에서는 최신 로그 이벤트 50개를 표시하도록 샘플 쿼리를 수정합니다.

이전 자습서를 아직 실행하지 않은 경우 지금 실행하는 것이 좋습니다. 이 자습서는 이전 자습서를 마친 지점에서 시작합니다.

> **Note**
>
> CloudWatch Logs Insights에서 제공하는 일부 샘플 쿼리는 `limit` 대신 `head` 또는 `tail` 명령을 사용합니다. 이러한 명령은 더 이상 사용되지 않으며 `limit`로 대체되었습니다. 작성하는 모든 쿼리에 `head` 또는 `tail` 대신 `limit`를 사용합니다.

CloudWatch Logs Insights 샘플 쿼리를 수정하려면

1. 쿼리 편집기에서 20을 50으로 변경한 다음 실행을 선택합니다.

   새 쿼리의 결과가 표시됩니다. 기본 시간 범위 내에서 로그 파일에 데이터가 충분하다고 가정하고 이제 로그 이벤트 50개가 나열됩니다.
2. (선택 사항) 생성한 쿼리를 저장할 수 있습니다. 이 쿼리를 저장하려면 저장을 선택합니다. 자세한 정보는 CloudWatch Logs Insights 쿼리 저장 및 재실행 섹션을 참조하세요.

샘플 쿼리에 필터 명령 추가

이 자습서에서는 쿼리 편집기에서 쿼리를 보다 과감하게 변경하는 방법에 대해 살펴봅니다. 이 자습서에서는 검색된 로그 이벤트의 필드를 기반으로 이전 쿼리의 결과를 필터링합니다.

이전 자습서를 아직 실행하지 않은 경우 지금 실행하는 것이 좋습니다. 이 자습서는 이전 자습서를 마친 지점에서 시작합니다.

이전 쿼리에 필터 명령을 추가하려면

1. 필터링할 필드를 결정합니다. 지난 15분 동안 선택한 로그 그룹에 포함된 로그 이벤트에서 CloudWatch Logs가 감지한 가장 일반적인 필드와 각 필드가 나타나는 로그 이벤트의 비율을 보려면 페이지 오른쪽에서 필드를 선택합니다.

   특정 로그 이벤트에 포함된 필드를 확인하려면 해당 행 왼쪽에 있는 아이콘을 선택합니다.

   로그에 포함된 이벤트에 따라 로그 이벤트에 awsRegion 필드가 나타날 수 있습니다. 이 자습서의 나머지 부분에서는 필터 필드로 awsRegion을 사용하는데, 이 필드를 사용할 수 없는 경우에는 다른 필드를 사용할 수 있습니다.
2. 쿼리 편집기 상자에서 50 뒤에 커서를 놓고 입력을 누릅니다.
3. 새 줄에 먼저 |(파이프 문자)와 공백을 입력합니다. CloudWatch Logs Insights 쿼리 내 명령은 파이프 문자로 구분해야 합니다.
4. **`filter awsRegion="us-east-1"`**을 입력합니다.
5. 실행을 선택합니다.

   다시 쿼리를 실행하면 이제 새 필터와 일치하는 최신 결과 50개가 표시됩니다.

   다른 필드를 필터링했는데 오류 결과가 표시되면 해당 필드 이름을 이스케이프해야 할 수 있습니다. 필드 이름에 영숫자가 아닌 문자가 포함되어 있으면 필드 이름의 앞/뒤에 백틱 문자(`)를 입력해야 합니다(예: **`` `error-code`="102" ``**).

영숫자가 아닌 문자를 포함하는 필드 이름에 백틱 문자를 사용해야 하지만 값은 그렇지 않습니다. 값은 항상 따옴표(") 안에 포함됩니다.

Logs Insights QL에는 여러 가지 명령과 정규 표현식, 수학적 연산 및 통계적 연산 지원을 비롯하여 강력한 쿼리 기능이 있습니다. 자세한 내용은 CloudWatch Logs Insights 언어 쿼리 구문 섹션을 참조하세요.

자습서: 집계 함수를 사용하여 쿼리 실행

집계 함수를 `stats` 명령과 사용하고 다른 함수의 인수로 사용할 수 있습니다. 이 자습서에서는 지정된 필드를 포함하는 로그 이벤트의 수를 계산하는 쿼리 명령을 실행합니다. 쿼리 명령은 지정된 필드의 하나 이상의 값으로 그룹화된 총 개수를 반환합니다. 집계 함수에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 지원되는 연산 및 함수를 참조하세요.

집계 함수를 사용하여 쿼리 실행

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. Logs Insights QL 탭이 선택되어 있는지 확인합니다.
4. 로그 그룹 선택 드롭다운에서 쿼리할 로그 그룹을 하나 이상 선택합니다.

   CloudWatch 크로스 계정 관측성에서 모니터링 계정인 경우 소스 계정과 모니터링 계정에서 로그 그룹을 선택할 수 있습니다. 하나의 쿼리로 다른 계정의 로그를 한 번에 쿼리할 수 있습니다.

   로그 그룹 이름, 계정 ID 또는 계정 레이블을 기준으로 로그 그룹을 필터링할 수 있습니다.

   로그 그룹을 선택하면 CloudWatch Logs Insights가 로그 그룹의 데이터 필드를 자동으로 탐지합니다(Standard 클래스 로그 그룹인 경우). 검색된 필드를 보려면 페이지 오른쪽 상단의 필드 메뉴를 선택합니다.
5. 쿼리 편집기에서 기본 쿼리를 삭제하고 다음 명령을 입력합니다.

   ```
   stats count(*) by fieldName
   ```
6. *fieldName*을 필드 메뉴의 검색된 필드로 바꿉니다.

   필드 메뉴는 페이지 오른쪽 상단에 있으며 CloudWatch Logs Insights가 로그 그룹에서 탐지한 모든 검색된 필드를 표시합니다.
7. 실행을 선택하고 쿼리 결과를 봅니다.

쿼리 결과에는 쿼리 명령과 일치하는 로그 그룹의 레코드 수와 지정된 필드의 하나 이상의 값으로 그룹화된 총 개수가 표시됩니다.

자습서: 로그 필드로 그룹화된 시각화를 생성하는 쿼리 실행

`stats` 함수를 사용하여 반환된 결과를 로그 항목에 있는 하나 이상의 필드 값으로 그룹화하는 쿼리를 실행하면 결과를 막대 차트, 파이 차트, 선 그래프 또는 누적 영역 그래프로 볼 수 있습니다. 이렇게 하면 로그에서 추세를 보다 효율적으로 시각화할 수 있습니다.

시각화를 위한 쿼리를 실행하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. 로그 그룹 선택 드롭다운에서 쿼리할 로그 그룹을 하나 이상 선택합니다.

   CloudWatch 크로스 계정 관측성에서 모니터링 계정인 경우 소스 계정과 모니터링 계정에서 로그 그룹을 선택할 수 있습니다. 하나의 쿼리로 다른 계정의 로그를 한 번에 쿼리할 수 있습니다.

   로그 그룹 이름, 계정 ID 또는 계정 레이블을 기준으로 로그 그룹을 필터링할 수 있습니다.
4. 쿼리 편집기에서 현재 내용을 삭제하고 다음 `stats` 함수를 입력한 다음, 쿼리 실행을 선택합니다.

   ```
   stats count(*) by @logStream
       | limit 100
   ```

   결과는 로그 그룹에서 각 로그 스트림에 대한 로그 이벤트의 수를 보여줍니다. 결과는 100개 행으로 제한됩니다.
5. 시각화 탭을 선택합니다.
6. 줄 옆에 있는 화살표를 선택한 다음, 막대를 선택합니다.

   막대 차트가 나타나고 로그 그룹의 각 로그 스트림에 대한 막대가 표시됩니다.

자습서: 시계열 시각화를 생성하는 쿼리 실행

`bin()` 함수를 사용해 기간별로 반환되는 결과를 그룹화하는 쿼리를 실행하는 경우 결과를 선 그래프, 누적 영역 그래프, 파이 차트 또는 막대 차트로 볼 수 있습니다. 이렇게 하면 로그 이벤트의 경시적 추세를 보다 효율적으로 시각화할 수 있습니다.

시각화를 위한 쿼리를 실행하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. Logs Insights QL 탭이 선택되어 있는지 확인합니다.
4. 로그 그룹 선택 드롭다운에서 쿼리할 로그 그룹을 하나 이상 선택합니다.

   CloudWatch 크로스 계정 관측성에서 모니터링 계정인 경우 소스 계정과 모니터링 계정에서 로그 그룹을 선택할 수 있습니다. 하나의 쿼리로 다른 계정의 로그를 한 번에 쿼리할 수 있습니다.

   로그 그룹 이름, 계정 ID 또는 계정 레이블을 기준으로 로그 그룹을 필터링할 수 있습니다.
5. 쿼리 편집기에서 현재 내용을 삭제하고 다음 `stats` 함수를 입력한 다음, 쿼리 실행을 선택합니다.

   ```
   stats count(*) by bin(30s)
   ```

   결과에는 CloudWatch Logs가 30초 동안 매번 수신한 로그 그룹의 로그 이벤트의 수가 표시됩니다.
6. 시각화 탭을 선택합니다.

   결과가 선 그래프로 표시됩니다. 막대 차트, 파이 차트 또는 누적 영역 차트로 전환하려면 그래프 오른쪽 위 줄 옆에 있는 화살표를 선택합니다.

샘플 쿼리

이 섹션에는 CloudWatch 콘솔에서 실행할 수 있는 일반적이고 유용한 쿼리 명령 목록이 포함되어 있습니다. 쿼리 명령을 실행하는 방법에 대한 자세한 내용은 Amazon CloudWatch Logs User Guide의 Tutorial: Run and modify a sample query을 참조하세요.

쿼리 구문에 대한 자세한 내용은 CloudWatch Logs Insights 언어 쿼리 구문을 참조하세요.

주제

- 일반 쿼리
- Lambda 로그에 대한 쿼리
- Amazon VPC 흐름 로그에 대한 쿼리
- Route 53 로그에 대한 쿼리
- CloudTrail 로그에 대한 쿼리

- 에 대한 쿼리 Amazon API Gateway
- NAT 게이트웨이에 대한 쿼리
- Apache 서버 로그에 대한 쿼리
- Amazon EventBridge에 대한 쿼리
- 리소스 태그에 대한 쿼리
- 구문 분석 명령의 예제

일반 쿼리

최근에 추가된 로그 이벤트 25개를 찾습니다.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

시간당 발생한 예외 횟수 목록을 가져옵니다.

```
filter @message like /Exception/
    | stats count(*) as exceptionCount by bin(1h)
    | sort exceptionCount desc
```

예외에 해당되지 않는 로그 이벤트 목록을 가져옵니다.

```
fields @message | filter @message not like /Exception/
```

server 필드의 각 고유 값에 대한 최신 로그 이벤트를 가져옵니다.

```
fields @timestamp, server, severity, message
| sort @timestamp asc
| dedup server
```

각 **severity** 유형에 대한 **server** 필드의 각 고유 값에 대한 최신 로그 이벤트를 가져옵니다.

```
fields @timestamp, server, severity, message
| sort @timestamp desc
| dedup server, severity
```

Lambda 로그에 대한 쿼리

과다 프로비저닝된 메모리의 양을 확인합니다.

```
filter @type = "REPORT"
    | stats max(@memorySize / 1000 / 1000) as provisonedMemoryMB,
        min(@maxMemoryUsed / 1000 / 1000) as smallestMemoryRequestMB,
        avg(@maxMemoryUsed / 1000 / 1000) as avgMemoryUsedMB,
        max(@maxMemoryUsed / 1000 / 1000) as maxMemoryUsedMB,
        provisonedMemoryMB - maxMemoryUsedMB as overProvisionedMB
```

지연 보고서를 생성합니다.

```
filter @type = "REPORT" |
    stats avg(@duration), max(@duration), min(@duration) by bin(5m)
```

느린 함수 간접 호출을 검색하고 재시도 또는 클라이언트측 코드에서 발생할 수 있는 중복 요청을 제거합니다. 이 쿼리에서 **@duration**은 밀리초 단위입니다.

```
fields @timestamp, @requestId, @message, @logStream
| filter @type = "REPORT" and @duration > 1000
| sort @timestamp desc
| dedup @requestId
| limit 20
```

Amazon VPC 흐름 로그에 대한 쿼리

호스트 간에 상위 15개의 패킷 전송을 찾습니다.

```
stats sum(packets) as packetsTransferred by srcAddr, dstAddr
    | sort packetsTransferred  desc
    | limit 15
```

지정된 서브넷의 호스트에 대해 상위 15개 바이트 전송을 찾습니다.

```
filter isIpv4InSubnet(srcAddr, "192.0.2.0/24")
    | stats sum(bytes) as bytesTransferred by dstAddr
    | sort bytesTransferred desc
    | limit 15
```

데이터 전송 프로토콜로 UDP를 사용하는 IP 주소를 찾습니다.

```
filter protocol=17 | stats count(*) by srcAddr
```

캡처 기간 중 흐름 레코드를 건너뛴 IP 주소를 찾습니다.

```
filter logStatus="SKIPDATA"
    | stats count(*) by bin(1h) as t
    | sort t
```

네트워크 연결 문제 해결에 도움이 되도록 각 연결에 대한 단일 레코드를 찾습니다.

```
fields @timestamp, srcAddr, dstAddr, srcPort, dstPort, protocol, bytes
| filter logStream = 'vpc-flow-logs' and interfaceId = 'eni-0123456789abcdef0'
| sort @timestamp desc
| dedup srcAddr, dstAddr, srcPort, dstPort, protocol
| limit 20
```

Route 53 로그에 대한 쿼리

시간당 레코드 배포를 쿼리 유형별로 찾습니다.

```
stats count(*) by queryType, bin(1h)
```

요청 횟수가 가장 많은 DNS 해석기 10개를 찾습니다.

```
stats count(*) as numRequests by resolverIp
    | sort numRequests desc
    | limit 10
```

서버가 DNS 요청을 완료하지 못한 도메인 및 하위 도메인별 레코드의 수를 찾습니다.

```
filter responseCode="SERVFAIL" | stats count(*) by queryName
```

CloudTrail 로그에 대한 쿼리

각 서비스, 이벤트 유형 및 AWS 리전에 대한 로그 항목 수를 찾습니다.

```
stats count(*) by eventSource, eventName, awsRegion
```

지정된 AWS 리전에서 시작되거나 중지된 Amazon EC2 호스트를 찾습니다.

```
filter (eventName="StartInstances" or eventName="StopInstances") and awsRegion="us-
east-2"
```

새로 생성된 IAM 사용자의 AWS 리전, 사용자 이름 및 ARNs 찾습니다.

```
filter eventName="CreateUser"
    | fields awsRegion, requestParameters.userName, responseElements.user.arn
```

API **UpdateTrail**을 간접 호출하는 중 예외가 발생한 레코드의 수를 찾습니다.

```
filter eventName="UpdateTrail" and ispresent(errorCode)
    | stats count(*) by errorCode, errorMessage
```

TLS 1.0 또는 1.1이 사용된 로그 항목을 찾습니다.

```
filter tlsDetails.tlsVersion in [ "TLSv1", "TLSv1.1" ]
| stats count(*) as numOutdatedTlsCalls by userIdentity.accountId, recipientAccountId,
 eventSource, eventName, awsRegion, tlsDetails.tlsVersion, tlsDetails.cipherSuite,
 userAgent
| sort eventSource, eventName, awsRegion, tlsDetails.tlsVersion
```

TLS 버전 1.0 또는 1.1을 사용한 서비스당 직접 호출의 수를 찾습니다.

```
filter tlsDetails.tlsVersion in [ "TLSv1", "TLSv1.1" ]
| stats count(*) as numOutdatedTlsCalls by eventSource
| sort numOutdatedTlsCalls desc
```

에 대한 쿼리 Amazon API Gateway

마지막 10개의 4XX 오류 찾기

```
fields @timestamp, status, ip, path, httpMethod
| filter status>=400 and status<=499
```

```
| sort @timestamp desc
| limit 10
```

Amazon API Gateway 액세스 로그 그룹에서 가장 오래 실행되는 Amazon API Gateway 요청 10개 식별

```
fields @timestamp, status, ip, path, httpMethod, responseLatency
| sort responseLatency desc
| limit 10
```

Amazon API Gateway 액세스 로그 그룹에서 가장 많이 사용되는 API 경로 목록 반환

```
stats count(*) as requestCount by path
| sort requestCount desc
| limit 10
```

Amazon API Gateway 액세스 로그 그룹에 대한 통합 지연 시간 보고서 생성

```
filter status=200
| stats avg(integrationLatency), max(integrationLatency),
min(integrationLatency) by bin(1m)
```

NAT 게이트웨이에 대한 쿼리

AWS 청구서에서 일반 비용보다 높은 비용이 발견되면 CloudWatch Logs Insights를 사용하여 상위 기여자를 찾을 수 있습니다. 다음 쿼리 명령에 대한 자세한 내용은 VPC에서 NAT 게이트웨이를 통한 트래픽에 대한 상위 기여자를 찾으려면 어떻게 해야 합니까?를 참조하세요. 는 AWS 프리미엄 지원 페이지에서 확인할 수 있습니다.

> **Note**
>
> 다음 쿼리 명령에서 'x.x.x.x'를 NAT 게이트웨이의 프라이빗 IP로 바꾸고 'y.y'를 VPC CIDR 범위의 처음 두 옥텟으로 바꿉니다.

NAT 게이트웨이를 통해 가장 많은 트래픽을 전송하는 인스턴스를 찾습니다.

```
filter (dstAddr like 'x.x.x.x' and srcAddr like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
```

```
| limit 10
```

NAT 게이트웨이의 인스턴스에서 송수신되는 트래픽을 확인합니다.

```
filter (dstAddr like 'x.x.x.x' and srcAddr like 'y.y.') or (srcAddr like 'xxx.xx.xx.xx'
 and dstAddr like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

VPC 인스턴스가 업로드 및 다운로드를 위해 가장 자주 통신하는 인터넷 대상을 결정합니다.

업로드용

```
filter (srcAddr like 'x.x.x.x' and dstAddr not like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

다운로드용

```
filter (dstAddr like 'x.x.x.x' and srcAddr not like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

Apache 서버 로그에 대한 쿼리

CloudWatch Logs Insights를 사용하여 Apache 서버 로그를 쿼리할 수 있습니다. 다음 쿼리에 대한 자세한 내용은 Cloud Operations & Migrations 블로그의 Simplifying Apache server logs with CloudWatch Logs Insights를 참조하세요. AWS

가장 관련성이 높은 필드를 찾아서 액세스 로그를 검토하고 애플리케이션의 /admin 경로에서 트래픽을 확인할 수 있습니다.

```
fields @timestamp, remoteIP, request, status, filename| sort @timestamp desc
| filter filename="/var/www/html/admin"
| limit 20
```

상태 코드 '200'(성공)을 사용하여 기본 페이지에 액세스한 고유 GET 요청 수를 찾습니다.

```
fields @timestamp, remoteIP, method, status
| filter status="200" and referrer= http://34.250.27.141/ and method= "GET"
| stats countDistinct(remoteIP) as UniqueVisits
| limit 10
```

Apache 서비스가 다시 시작된 횟수를 찾습니다.

```
fields @timestamp, function, process, message
| filter message like "resuming normal operations"
| sort @timestamp desc
| limit 20
```

Amazon EventBridge에 대한 쿼리

이벤트 세부 정보 유형별로 그룹화된 EventBridge 이벤트의 수 가져오기

```
fields @timestamp, @message
| stats count(*) as numberOfEvents by `detail-type`
| sort numberOfEvents desc
```

리소스 태그에 대한 쿼리

리소스 태그 값을 기준으로 로그를 필터링합니다.

```
filter @aws.tag.env = "production"
```

특수 문자가 포함된 태그 키를 기준으로 필터링합니다.

```
filter @`aws.tag.aws:cloudformation:stack-name` = "my-stack"
```

태그 값으로 그룹화된 로그 이벤트 수를 가져옵니다.

```
stats count(*) by @aws.tag.team
```

구문 분석 명령의 예제

glob 표현식을 사용하여 로그 필드 **@message**에서 필드 **@user**, **@method** 및 **@latency**를 추출하고 **@method** 및 **@user**의 고유한 개별 조합에 대한 평균 지연 시간을 반환합니다.

```
parse @message "user=*, method:*, latency := *" as @user,
    @method, @latency | stats avg(@latency) by @method,
    @user
```

정규식을 사용하여 로그 필드 **@message**에서 필드 **@user2**, **@method2** 및 **@latency2**를 추출하고 **@method2** 및 **@user2**의 고유한 개별 조합에 대한 평균 지연 시간을 반환합니다.

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
    latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
    @user2
```

loggingTime, **loggingType** 및 **loggingMessage** 필드를 추출하고 **ERROR** 또는 **INFO** 문자열이 포함된 이벤트를 기록하도록 필터링한 다음 **ERROR** 문자열이 포함된 이벤트에 대해 **loggingMessage** 및 **loggingType** 필드만 표시합니다.

```
FIELDS @message
    | PARSE @message "* [*] *" as loggingTime, loggingType, loggingMessage
    | FILTER loggingType IN ["ERROR", "INFO"]
    | DISPLAY loggingMessage, loggingType = "ERROR" as isError
```

(diff)를 이전 시간 범위와 비교

CloudWatch Logs Insights와 Logs Insights QL을 함께 사용하여 시간 경과에 따른 로그 이벤트의 변경 사항을 비교할 수 있습니다. 최근 시간 범위 동안 수집된 로그 이벤트를 직전 기간의 로그와 비교할 수 있습니다. 또는 유사한 과거 기간과 비교할 수 있습니다. 이렇게 하면 로그의 오류가 최근에 유입되었는지 또는 이미 발생하고 있었는지 확인하는 데 도움이 되고, 다른 추세를 찾는 데 도움이 될 수 있습니다.

비교 쿼리는 원시 로그 이벤트가 아니라 패턴만 결과에 반환합니다. 반환된 패턴은 시간 경과에 따른 로그 이벤트의 추세와 변경 사항을 빠르게 확인하는 데 도움이 됩니다. 비교 쿼리를 실행하고 패턴 결과를 얻은 후 관심 있는 패턴의 샘플 원시 로그 이벤트를 볼 수 있습니다. 로그 패턴에 대한 자세한 내용은 패턴 분석을 참조하세요.

비교 쿼리를 실행하면 두 가지 기간(선택한 원래 쿼리 기간과 비교 기간)에 대해 쿼리가 분석됩니다. 비교 기간은 항상 원래 쿼리 기간과 길이가 동일합니다. 비교의 기본 시간 간격은 다음과 같습니다.

- 이전 기간 - 쿼리 기간 직전의 기간과 비교합니다.
- 전일 - 쿼리 기간 하루 전의 기간과 비교합니다.
- 이전 주 - 쿼리 기간 1주 전의 기간과 비교합니다.

- 이전 달 - 쿼리 기간 1개월 전의 기간과 비교합니다.

> **Note**
>
> 비교를 사용하는 쿼리에는 결합된 시간 범위에서 단일 CloudWatch Logs Insights 쿼리를 실행하는 것과 비슷한 요금이 발생합니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

비교 쿼리를 실행하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, Logs Insights를 선택합니다.

   기본 쿼리가 쿼리 상자에 나타납니다.
3. Logs Insights QL 탭이 선택되어 있는지 확인합니다.
4. 기본 쿼리를 유지하거나 다른 쿼리를 입력합니다.
5. 로그 그룹 선택 드롭다운에서 쿼리할 로그 그룹을 하나 이상 선택합니다.
6. (선택 사항) 시간 간격 선택기를 사용하여 쿼리할 기간을 선택합니다. 기본 쿼리는 로그 데이터의 이전 1시간에 대한 쿼리입니다.
7. 시간 범위 선택기에서 비교를 선택합니다. 그런 다음 원본 로그를 비교할 이전 기간을 선택하고 적용을 선택합니다.
8. 쿼리 실행을 선택합니다.

   쿼리가 비교 기간의 데이터를 가져오도록 하기 위해 `diff` 명령이 쿼리에 추가됩니다.
9. 패턴 탭을 선택하여 결과를 확인합니다.

   테이블에는 다음과 같은 정보가 표시됩니다.

   - 패턴의 가변 부분이 동적 토큰 기호 <*string*-*number*>로 대체된 각 패턴. *string*은 토큰이 나타내는 데이터 유형에 대한 설명입니다. *number*는 다른 동적 토큰과 비교하여 패턴에서 이 토큰이 나타나는 위치를 보여줍니다. 자세한 내용은 패턴 분석 섹션을 참조하세요.
   - 이벤트 개수는 원래의 더 최근 기간에 해당 패턴이 있는 로그 이벤트의 수입니다.
   - 이벤트 수 차이는 현재 기간과 비교 기간의 일치하는 로그 이벤트 수의 차이입니다. 양수의 차이는 현재 기간에 이러한 이벤트가 더 많다는 의미입니다.
   - 차이점 설명은 현재 기간과 비교 기간 간 해당 패턴의 변화를 간략하게 요약합니다.

- 심각도 유형은 FATAL, ERROR 및 WARN 같은 로그 이벤트에서 발견되는 단어를 기반으로 이 패턴이 있는 로그 이벤트에서의 가능한 심각도입니다.

10. 목록의 패턴 중 하나를 추가로 검사하려면 패턴 중 하나의 검사 열에서 아이콘을 선택합니다.

    패턴 검사 창이 나타나고 다음이 표시됩니다.

- 패턴. 패턴 내에서 토큰을 선택하여 해당 토큰의 값을 분석합니다.
- 쿼리된 시간 범위에서 패턴의 발생 횟수를 보여주는 히스토그램. 이는 패턴 발생의 갑작스러운 증가 등 흥미로운 추세를 식별하는 데 도움이 될 수 있습니다.
- 로그 샘플 탭에는 선택한 패턴과 일치하는 몇 가지 로그 이벤트가 표시됩니다.
- 토큰 값 탭에는 선택한 동적 토큰의 값이 표시됩니다(동적 토큰을 선택한 경우).

> **Note**
>
> 각 토큰마다 최대 10개의 토큰 값이 캡처됩니다. 토큰 개수가 정확하지 않을 수 있습니다. CloudWatch Logs는 확률 계수기를 사용하여 절대값이 아닌 토큰 개수를 생성합니다.

- 관련 패턴 탭에는 검사 중인 패턴과 거의 동일한 시간에 자주 발생한 다른 패턴이 표시됩니다. 예를 들어, ERROR 메시지의 패턴에 일반적으로 추가 세부 정보와 함께 INFO로 표시된 다른 로그 이벤트가 수반된 경우 해당 패턴이 여기에 표시됩니다.

그래프로 로그 데이터 시각화

막대 차트, 선형 차트 및 누적 영역 차트와 같은 시각화를 사용하여 로그 데이터의 패턴을 보다 효율적으로 식별할 수 있습니다. CloudWatch Logs Insights는 stats 함수와 집계 함수를 하나 이상 사용하는 쿼리에 대한 시각화를 생성합니다. 자세한 내용은 통계를 참조하세요.

OpenSearch Piped Processing Language(PPL)

이 섹션에는 OpenSearch PPL을 사용한 CloudWatch Logs 쿼리에 대한 기본 소개가 포함되어 있습니다. PPL을 사용하면 파이프 조합 명령을 사용하여 데이터를 검색, 쿼리 및 분석할 수 있으므로 복잡한 쿼리를 더 쉽게 이해하고 구성할 수 있습니다. 구문은 Unix 파이프를 기반으로 하며 데이터를 변환하고 처리하기 위한 명령 체인을 활성화합니다. PPL을 사용하면 데이터를 필터링 및 집계하고 분석에 풍부한 수학, 문자열, 날짜, 조건부 및 기타 함수 집합을 사용할 수 있습니다.

PPL 쿼리SOURCE에를 포함시키는 것은 또는 AWS CLI API를 사용하여 쿼리를 생성할 때 로그 그룹 필드 인덱스와 쿼리에 포함할 데이터 소스를 지정하는 유용한 방법입니다. SOURCE 명령은 CloudWatch

콘솔이 아닌 AWS CLI 및 API에서만 지원됩니다. CloudWatch 콘솔을 사용하여 쿼리를 시작할 때 콘솔 인터페이스를 사용하여 로그 그룹과 데이터 소스 이름 및 유형을 지정합니다.

`aws:fieldIndex`를 사용하여 쿼리에서 지정한 필드에 인덱싱된 로그 그룹만 검색하도록 강제하여 인덱스된 데이터만 반환합니다. `filterIndex` 명령에 지정된 필드를 기반으로 관련 로그 그룹이 자동으로 선택됩니다. 이렇게 하면 쿼리에 지정된 필드가 포함된 로그 이벤트가 없는 로그 그룹을 건너뛰고이 필드 인덱스의 쿼리에 지정된 값과 일치하는 로그 그룹만 스캔하여 스캔 볼륨이 줄어듭니다. 지정된 필드와 값이 포함된 인덱싱된 데이터만 쿼리`aws:fieldIndex`하려면 소스 명령의 필드 이름 및 값과 함께 필드 이름을 지정하는 데 사용합니다. 자세한 내용은 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성 섹션을 참조하세요.

표준 로그 클래스의 로그 그룹 쿼리에 OpenSearch PPL을 사용할 수 있습니다.

> **Note**
>
> CloudWatch Logs에서 지원되는 모든 OpenSearch PPL 쿼리 명령에 대한 자세한 내용과 구문 및 제한에 대한 자세한 내용은 OpenSearch Service 개발자 가이드의 Supported PPL commands을 참조하세요.
> 사용할 수 있는 다른 쿼리 언어에 대한 자세한 내용은 CloudWatch Logs Insights, OpenSearch Service SQL 및 CloudWatch Metrics Insights을 참조하세요.

명령 또는 함수	예제 쿼리	설명
`fields`	`fields field1, field2`	프로젝션이 필요한 필드 집합을 표시합니다.
`join`	`LEFT JOIN left=l, right=r on l.id = r.id` `` `join_right_lg` `` `| fields l.field_1, r.field_2`	두 데이터세트를 함께 조인합니다.
`where`	`where field1="success" | where field2 != "i-023fe0a90929d8822" | fields field3, field4, field5,field6 | head 1000`	지정한 조건에 따라 데이터를 필터링합니다.
`aws:fieldIndex`	`source = [` `` `aws:fieldIndex` `` `="region",` `` `region` `` `= "us-west-2"] | where status = 200 | head 10`	쿼리에서 지정한 필드에 인덱싱된 로그 그룹만 검색

명령 또는 함수	예제 쿼리	설명
		하도록 쿼리를 강제하여 인덱싱된 데이터만 반환합니다.
stats	`stats count(), count(field1), min(field1), max(field1), avg(field1) by field2 | head 1000`	집계 및 계산 수행
parse	`parse field1 ".*/(?<field2>[^/]+$)" | where field2 = "requestId" | fields field1, field2 | head 1000`	문자열에서 정규식 패턴을 추출하고 추출된 패턴을 표시합니다. 추출된 패턴을 사용하여 새 필드를 생성하거나 데이터를 필터링할 수 있습니다.
sort	`stats count(), count(field1), min(field1) as field1Alias, max(`field1`), avg(`field1`) by field2 | sort -field1Alias | head 1000`	필드 이름을 기준으로 표시된 결과를 정렬합니다. sort -FieldName을 사용하여 내림차순으로 정렬합니다.

명령 또는 함수	예제 쿼리	설명
eval	`eval field2 = field1 * 2 | fields field1, field2 | head 20`	필드 값을 수정 또는 처리하고 다른 필드에 저장합니다. 이는 열을 수학적으로 수정하거나, 열에 문자열 함수를 적용하거나, 열에 날짜 함수를 적용하는 데 유용합니다.
rename	`rename field2 as field1 | fields field1;`	검색 결과에서 하나 이상의 필드 이름을 바꿉니다.
head	`fields `@message` | head 20`	표시된 쿼리 결과를 첫 번째 N개 행으로 제한합니다.
top	`top 2 field1 by field2`	필드의 가장 빈번한 값을 찾습니다.
dedup	`dedup field1 | fields field1, field2, field3`	지정한 필드를 기반으로 중복 항목을 제거합니다.
rare	`rare field1 by field2`	필드 목록에 있는 모든 필드의 가장 빈도가 낮은 값을 찾습니다.

명령 또는 함수	예제 쿼리	설명
subquery	`where field_1 IN [ search source= `subquery_lg` | fields field_2 ] | fields id, field_1`	PPL 명령문 내에서 복잡한 중첩 쿼리를 수행합니다.
trendline	`trendline sma(2, field1) as field1Alias`	필드의 이동 평균을 계산합니다.
eventStats	`eventstats sum(field1) by field2`	계산된 요약 통계로 이벤트 데이터를 강화합니다. 이벤트 내에서 지정된 필드를 분석하고 다양한 통계 측정값을 계산한 다음 이러한 결과를 각 원래 이벤트에 새 필드로 추가합니다.
expand	`eval tags_array_string = json_extract(`@message`, '$.tags')\| eval tags_array = json_array(json_extract(tags_string, '$[0]'), json_extract(tags_string, '$[1]'))| expand tags_array as color_tags`	여러 값이 포함된 필드를 별도의 행으로 분할하여 지정된 필드의 각 값에 대해 새 행을 생성합니다.
fillnull	`fields `@timestamp`, error_code, status_code | fillnull using status_code = "UNKNOWN", error_code = "UNKNOWN"`	입력한 값으로 null 필드를 채웁니다. 하나 이상의 필드에서 사용할 수 있습니다.

명령 또는 함수	예제 쿼리	설명
flatten	`eval metadata_struct = json_object('size', json_extract(metadata_string, '$.size'), 'color', json_extract(metadata_string, '$.color')) | flatten metadata_struct as (meta_size, meta_color)`	필드를 평면화합니다. 필드는 `struct<?,?>` 또는 `array<struct<?,?>>` 유형이어야 합니다.
cidrmatch	`where cidrmatch(ip, '2003:db8::/32') | fields ip`	지정된 IP 주소가 지정된 CIDR 범위 내에 있는지 확인합니다.
fieldsummary	`where field1 != 200 | fieldsummary includefields= field1 nulls=true`	각 필드(개수, 고유 개수, 최소, 최대, 평균, stddev 및 평균)에 대한 기본 통계를 계산합니다.
grok	`grok email '.+@%{HOSTNAME:host}' | fields email, host`	grok 패턴으로 텍스트 필드를 구문 분석하고 검색 결과에 결과를 추가합니다.

명령 또는 함수	예제 쿼리	설명
문자열 함수	`eval field1Len = LENGTH(field1) | fields field1Len`	PPL 쿼리 내에서 문자열 및 텍스트 데이터를 조작하고 변환할 수 있는 PPL의 내장 함수입니다. 예를 들어, 변환 사례, 문자열 결합, 부분 추출, 텍스트 정리 등이 있습니다.
날짜/시간 함수	`eval newDate = ADDDATE(DATE('2020-08-26'), 1) | fields newDate`	PPL 쿼리에서 날짜 및 타임스탬프 데이터를 처리하고 변환하기 위한 내장 함수입니다. 예: date_add, date_format, datediff, date-sub, timestamp add, timestamp diff, current_timezone, utc_timestamp, current_date
조건 함수	`eval field2 = isnull(field1) | fields field2, field1, field3`	특정 필드 조건을 확인하고 조건부로 표현식을 평가하는 내장 함수입니다. 예를 들어, field1이 null인 경우 field2를 반환합니다.

명령 또는 함수	예제 쿼리	설명
수학 함수	`eval field2 = ACOS(field1) | fields field1`	PPL 쿼리에서 수학 계산 및 변환을 수행하기 위한 내장 함수입니다. 예: abs(절대값), round(숫자 반올림), sqrt(제곱근), pow(검정력 계산), ceil(가장 가까운 정수로 반올림)
CryptoGraphic 함수	`eval crypto = MD5(field)| head 1000`	지정된 필드의 해시를 계산하려면
JSON 함수	`eval valid_json = json('[1,2,3,{"f1":1,"f2":[5,6]},4]') | fields valid_json`	배열, 추출 및 검증을 포함하여 JSON을 처리하기 위한 내장 함수입니다. 예: json_object, json_array, to_json_string, json_array_length, json_extract, json_keys 및 json_valid

쿼리 범위

쿼리에 SOURCE를 포함시키는 것은 AWS CLI 또는 API를 사용하여 쿼리를 생성할 때 쿼리에 포함할 로그 그룹을 지정하는 유용한 방법입니다. SOURCE 명령은 CloudWatch 콘솔이 아닌 AWS CLI 및 API에서만 지원됩니다. CloudWatch 콘솔을 사용하여 쿼리를 시작할 때 콘솔 인터페이스를 사용하여 로그 그룹과 데이터 소스 이름 및 유형을 지정합니다.

PPL의 소스 명령은 이제 여러 가지 방법으로 지정할 수 있습니다.

1. 로그 그룹:
2. 필드 인덱스 - 신규
3. 데이터 소스 및 유형 - 신규

로그 그룹

로그 그룹 소스 선택은 고객이 검색해야 하는 정확한 로그 그룹(들)을 알 때 사용할 수 있습니다.

```
source = [lg:`/aws/lambda/my-function`] | where status = 200 | head 10
```

필드 인덱스

필드 인덱스 기반 소스 선택은 인덱싱된 대상 필드를 필터링할 때 결과를 인덱싱된 데이터로만 제한하여 쿼리되는 데이터의 양을 줄입니다. `filterIndex` 명령에 지정된 필드를 기반으로 관련 로그 그룹이 자동으로 선택됩니다. 필드 인덱스 및 필드 인덱스 생성 방법에 대한 자세한 내용은 필드 인덱스 생성을 참조하여 쿼리 성능을 개선하고 스캔 볼륨을 줄입니다.

`aws:fieldIndex`를 사용하여 쿼리에서 지정한 필드에 인덱싱된 로그 그룹만 검색하도록 강제하여 인덱스된 데이터만 반환합니다. 해당 필드에 인덱싱된 이러한 로그 그룹의 경우, 인덱싱된 필드의 쿼리에 지정된 필드가 포함된 로그 이벤트가 없는 로그 그룹을 건너뛰어 쿼리를 추가로 최적화합니다. 또한 해당 필드 인덱스의 쿼리에 지정된 값과 일치하는 이러한 로그 그룹의 로그 이벤트만 스캔하려고 시도하여 스캔 볼륨을 줄입니다. 필드 인덱스 및 필드 인덱스 생성 방법에 대한 자세한 내용은 필드 인덱스 생성을 참조하여 쿼리 성능을 개선하고 스캔 볼륨을 줄입니다.

PPL에서 `aws:fieldIndex`는 인덱스로 처리해야 하는 키 값 페어를 지정하는 데 사용됩니다. 구문은 다음과 같습니다.

```
source = [`aws:fieldIndex`="region", `region` = "us-west-2"] | where status = 200 |
 head 10
```

여기서 각 항목은 다음과 같습니다.

1. `` `aws:fieldIndex`="region" ``는 리전을 필드 인덱스로 식별합니다.
   a. 참고: = 대신 고객은 IN을 사용하여 여러 인덱스를 지정할 수 있습니다(아래 예).
2. `` `region`="us-west-2" `` 적용할 필터 조건을 식별합니다.

a. 참고: 대신 = 고객이 IN을 사용하여 여러 값을 지정할 수 있음(아래 예)

고객은 다음과 같이 여러 fieldIndexes를 지정할 수 있습니다.

```
source = [`aws:fieldIndex` IN ("status", "region"), `status` = 200, `region` IN ("us-
west-2", "us-east-1")] | head 10
```

데이터 소스 및 유형

데이터 소스 및 유형 기반 소스 선택은 고객이 쿼리해야 하는 정확한 데이터 소스를 알 때 사용할 수 있습니다. 이 쿼리는 지정된 데이터 소스 및 유형이 포함된 하나 이상의 로그 그룹을 통해 실행됩니다.

```
source = [ds:`data_source.type`] | where status = 200 | head 10
```

데이터 소스 쿼리에 지원되는 PPL

PPL에서 데이터 소스를 쿼리하는 사용 사례를 지원하기 위해 동적 소스 선택기 절을 사용할 수 있습니다. 이 구문을 사용하면 검색 명령에 지정하여 데이터 소스를 쿼리할 수 있습니다. 최대 10개의 데이터 소스를 지정할 수 있습니다.

구문

```
source=[ds:`DataSource1.Type1`, ds:`DataSource2.Type2`, ...ds:`DataSourcen.Typen`]
```

쿼리 예

```
search source=[ds:`DataSource1.Type1`, ds:`DataSource2.Type2`] | fields field1, field2
```

결합된 예제

고객은 어떤 순서로든 모든 소스 선택 연산자를 지정할 수 있으며, 결과는 적용된 모든 조건의 교차점입니다.

예를 들어 /aws/lambda/my-function-1에는 다양한 인덱스를 포함한 여러 데이터 소스 및 유형이 포함될 수 있습니다. 다음 쿼리가 실행되면 반환된 결과에는 소스 및 유형 DataSource1.Type1의 이벤트만 있으며 'status' = 200의 기준과 일치합니다.

```
search source=[
```

```
    ds:`DataSource1.Type1`,
    lg:`/aws/lambda/my-function-1`,
    `aws:fieldIndex` IN ("status"), `status` = 200
]
```

제한 사항

OpenSearch PPL을 사용하여 CloudWatch Logs Insights에서 쿼리할 때 다음 제한이 적용됩니다.

- 데이터 소스 쿼리에는 조인 또는 하위 쿼리 명령을 사용할 수 없습니다.

OpenSearch Structured Query Language(SQL)

이 섹션에는 OpenSearch SQL을 사용한 CloudWatch Logs 쿼리에 대한 기본 소개가 포함되어 있습니다. 관계형 데이터베이스로 작업하는 데 익숙한 옵션을 제공합니다. OpenSearch SQL은 SQL 기능의 하위 집합을 제공하므로 임시 쿼리 및 데이터 분석 작업을 수행하는 데 적합합니다. OpenSearch SQL을 사용하면 SELECT, FROM, WHERE, GROUP BY, HAVING 및 기타 다양한 SQL 명령 및 함수와 같은 명령을 사용할 수 있습니다. 로그 그룹 간에 조인하고, 하위 쿼리를 사용하여 로그 그룹 간에 데이터를 상호 연관시키고, 풍부한 JSON, 수학, 문자열, 조건부 및 기타 SQL 함수 집합을 사용하여 로그 및 보안 데이터에 대한 강력한 분석을 수행할 수 있습니다.

`filterIndex`를 사용하여 쿼리에서 지정한 필드에 인덱싱된 로그 그룹만 검색하도록 강제하여 인덱스된 데이터만 반환합니다. 쿼리에 지정된 필드가 포함된 로그 이벤트가 없는 로그 그룹을 건너뛰고이 필드 인덱스의 쿼리에 지정된 값과 일치하는 로그 그룹만 스캔하여 스캔 볼륨을 줄입니다. `filterIndex`를 사용하여 필드 이름을 지정하고 필드 이름 및 값을 지정하여 지정된 필드 및 값이 포함된 인덱싱된 데이터만 쿼리합니다.

표준 로그 클래스의 로그 그룹 쿼리에 OpenSearch SQL을 사용할 수 있습니다. SQL은 데이터 소스 이름 및 데이터 소스 유형을 사용한 쿼리도 지원합니다.

> **Note**
>
> 다음 표에는 CloudWatch Logs에서 지원되는 SQL 명령 및 함수가 나열되어 있습니다. 구문을 포함한 모든 OpenSearch SQL 명령에 대한 자세한 내용은 OpenSearch Service 개발자 가이드의 지원되는 SQL 명령을 참조하세요.
> 사용할 수 있는 다른 쿼리 언어에 대한 자세한 내용은CloudWatch Logs Insights, OpenSearch Service PPL 및 CloudWatch Metrics Insights를 참조하세요.

지원되는 SQL 명령

> **Note**
>
> 예제 쿼리 열에서 쿼리하려는 데이터 소스에 따라 필요에 따라 *<logGroup>*을 바꿉니다.

명령 또는 함수	예제 쿼리	설명
SELECT	`SELECT \`@message\`, Operation FROM \`LogGroupA\``	프로젝션된 값을 표시합니다.
FROM	`SELECT \`@message\`, Operation FROM \`LogGroupA\``	다양한 유형의 조인 및 하위 쿼리를 지원하여 데이터를 검색할 소스 테이블 또는 보기를 지정하는 기본 제공 절입니다.
WHERE	`SELECT * FROM \`LogGroupA\` WHERE Operation = 'x'`	제공된 필드 기준에 따라 로그 이벤트를 필터링합니다.
filterIndex	`SELECT * FROM \`filterIndex('region' = 'us-east-1')\` WHERE status = 200 LIMIT 10;`	쿼리에서 지정한 필드에 인덱싱된 로그 그룹만 검색하도록 쿼리를 강제하여 인덱싱된 데이터만 반환합니다.
GROUP BY	`SELECT \`@logStream\`, COUNT(*) as log_count FROM \`LogGroupA\` GROUP BY \`@logStream\``	카테고리를 기반으로 로그 이벤트를 그룹화하고 통

명령 또는 함수	예제 쿼리	설명
		계를 기반으로 평균을 찾습니다.
HAVING	`SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream` HAVING log_count > 100`	그룹화 조건을 기준으로 결과를 필터링합니다.
ORDER BY	`SELECT * FROM `LogGroupA` ORDER BY `@timestamp` DESC`	ORDER BY 절의 필드를 기반으로 결과를 정렬합니다. 내림차순 또는 오름차순으로 정렬할 수 있습니다.
JOIN	`SELECT A.`@message`, B.`@timestamp`FROM `LogGroupA` as A INNER JOIN `LogGroupB` as B ON A.`requestId` = B.`requestId``	공통 필드를 기반으로 두 테이블의 결과를 조인합니다. 내부 조인 또는 왼쪽 외부 조인을 지정해야 합니다.
LIMIT	`Select * from `LogGroupA` limit 10`	표시된 쿼리 결과를 첫 번째 N개 행으로 제한합니다.

명령 또는 함수	예제 쿼리	설명
문자열 함수	`SELECT upper(Operation) , lower(Operation), Operation FROM `LogGroupA``	SQL 쿼리 내에서 문자열 및 텍스트 데이터를 조작하고 변환할 수 있는 SQL의 내장 함수입니다. 예를 들어, 변환 사례, 문자열 결합, 부분 추출, 텍스트 정리 등이 있습니다.
날짜 함수	`SELECT current_date() as today, date_add(current_date(), 30) as thirty_days_later, last_day(current_date()) as month_end FROM `LogGroupA``	SQL 쿼리에서 날짜 및 타임스탬프 데이터를 처리하고 변환하기 위한 내장 함수입니다. 예: date_add, date_format, datediff, current_date
조건 함수	`SELECT Operation, IF(Error > 0, 'High', 'Low') as error_category FROM `LogGroupA`;`	지정된 조건을 기반으로 작업을 수행하거나 조건부로 표현식을 평가하는 내장 함수입니다. 예: CASE 및 IF

명령 또는 함수	예제 쿼리	설명
집계 함수	`SELECT AVG(bytes) as bytesWritten FROM `LogGroupA``	여러 행에서 계산을 수행하여 단일 요약 값을 생성하는 내장 함수입니다. 예: SUM, COUNT, AVG, MAX 및 MIN
JSON 함수	`SELECT get_json_object(json_column, '$.name') as name FROM `LogGroupA``	데이터세트에서 JSON 구조를 조작할 수 있도록 SQL 쿼리(예: from_json, to_json, get_json_object, json_tuple) 내에서 JSON 형식의 데이터를 구문 분석, 추출, 수정 및 쿼리하는 기본 제공 함수입니다.
배열 함수	`SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;`	SQL 쿼리에서 배열 유형 열로 작업하기 위한 내장 함수를 사용하면 배열 데이터 액세스, 수정 및 분석과 같은 연산(예: size, explode, array_contains)을 수행할 수 있습니다.

명령 또는 함수	예제 쿼리	설명
윈도우 함수	`SELECT field1, field2, RANK() OVER (ORDER BY field2 DESC) as field2Rank FROM `LogGroupA`;`	현재 행(윈도우)과 관련된 지정된 행 세트에서 계산을 수행하여 순위, 합계 실행 및 이동 평균과 같은 연산을 활성화하는 내장 함수입니다. 예: ROW_NUMBER, RANK, LAG 및 LEAD
변환 함수	`SELECT CAST('123' AS INT) as converted_number, CAST(123 AS STRING) as converted_string FROM `LogGroupA``	SQL 쿼리 내에서 데이터를 한 유형에서 다른 유형으로 변환하여 데이터 유형 변환 및 형식 변환을 지원하는 내장 함수. 예: CAST, TO_DATE, TO_TIMESTAMP 및 BINARY
조건자 함수	`SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;`	조건을 평가하고 지정된 기준 또는 패턴에 따라 부울 값(true/false)을 반환하는 내장 함수입니다. 예: IN, LIKE, BETWEEN, IS NULL 및 EXISTS

명령 또는 함수	예제 쿼리	설명
여러 로그 그룹 선택	`SELECT lg1.field1, lg1.field2 from `logGroups(logGroupIdentifier: ['LogGroup1', 'LogGroup2'])` as lg1 where lg1.field3= "Success"`	SELECT 명령문에서 여러 로그 그룹 지정 가능
여러 데이터 소스 선택	`SELECT ds1.field1, ds1.field2 from `dataSource(['DataSource1', 'DataSource2'])` as ds1 where ds1.field3= "Success"`	SELECT 문에서 여러 데이터 소스를 지정할 수 있습니다.

multi-log-group 쿼리에 지원되는 SQL

SQL에서 여러 로그 그룹을 쿼리하는 사용 사례를 지원하려면 `logGroups` 명령을 사용할 수 있습니다. 이 구문을 사용하여 FROM 명령에 지정하여 여러 로그 그룹을 쿼리할 수 있습니다.

구문:

```
`logGroups(
    logGroupIdentifier: ['LogGroup1','LogGroup2', ...'LogGroupn']
)`
```

이 구문에서는 `logGroupIdentifier` 파라미터에 최대 50개의 로그 그룹을 지정할 수 있습니다. 모니터링 계정의 로그 그룹을 참조하려면 `LogGroup` 이름 대신 ARN을 사용합니다.

예제 쿼리:

```
SELECT LG1.Column1, LG1.Column2 from `logGroups(
    logGroupIdentifier: ['LogGroup1', 'LogGroup2']
)` as LG1 WHERE LG1.Column1 = 'ABC'
```

CloudWatch Logs를 쿼리할 때 `FROM` 명령문 뒤에 여러 로그 그룹이 포함된 다음 구문은 지원되지 않습니다.

```
SELECT Column1, Column2 FROM 'LogGroup1', 'LogGroup2', ...'LogGroupn'
WHERE Column1 = 'ABC'
```

데이터 소스 쿼리에 지원되는 SQL

SQL에서 데이터 소스를 쿼리하는 사용 사례를 지원하려면 dataSource 명령을 사용하면 됩니다. 이 구문을 사용하면 FROM 명령에서 데이터 소스를 지정하여 쿼리할 수 있습니다. 최대 10개의 데이터 소스를 지정할 수 있습니다.

구문

```
`dataSource(
    ['DataSource1', 'DataSource2', ...'DataSourcen']
)`
```

쿼리 예제

```
SELECT DS1.Column1, DS1.Column2 from `dataSource(
    ['DataSource1', 'DataSource2']
)` as DS1 WHERE DS1.Column1 = 'ABC'
```

쿼리 범위

AWS CLI 및 API에서 로그 그룹, 데이터 소스 및 유형, 필드 인덱스를 사용하여 쿼리할 로그를 지정할 수 있습니다.

로그 그룹

로그 그룹 소스 선택은 고객이 검색해야 하는 정확한 로그 그룹(들)을 알 때 사용할 수 있습니다.

```
SELECT * FROM `logGroups(logGroupIdentifier: ['/aws/lambda/my-function'])`;
```

데이터 소스 및 유형

고객은 데이터 소스 이름 및 데이터 소스 유형을 사용하여 로그를 쿼리할 수 있습니다.

데이터 소스 및 유형 기반 소스 선택은 고객이 쿼리해야 하는 정확한 데이터 소스를 알 때 사용할 수 있습니다. 이 쿼리는 지정된 데이터 소스 및 유형이 포함된 하나 이상의 로그 그룹을 통해 실행됩니다.

SQL에서 데이터 소스를 쿼리하는 사용 사례를 지원하려면 dataSource 명령을 사용하면 됩니다. 이 구문을 사용하면 FROM 명령에 지정하여 데이터 소스를 쿼리할 수 있습니다. 최대 10개의 데이터 소스를 지정할 수 있습니다.

구문:

```
`dataSource(
    ['DataSource1.Type1', 'DataSource2.Type2', ...'DataSourcen.Typen']
)`
```

예제 쿼리:

```
SELECT DS1.Column1, DS1.Column2 from `dataSource(
    ['DataSource1.Type1', 'DataSource2.Type2']
)` as DS1 WHERE DS1.Column1 = 'ABC'
```

데이터 소스별 쿼리에 대한 자세한 내용은 섹션을 참조하세요패싯을 사용하여 로그 그룹화 및 탐색.

결합된 예제

고객은 백틱 내의 모든 소스 선택 연산자를 어떤 순서로든 지정할 수 있으며 결과는 적용된 모든 조건의 교차점을 기반으로 합니다.

예를 들어 /aws/lambda/my-function-1에는 다양한 인덱스를 포함한 여러 데이터 소스 및 유형이 포함될 수 있습니다. 다음 쿼리가 실행되면 반환된 결과에는 소스 및 유형 DataSource1.Type1의 이벤트만 있으며 'status' = 200의 기준과 일치합니다.

```
SELECT * FROM `
   logGroups(logGroupIdentifier: ['/aws/lambda/my-function'])
   filterIndex('status' = 200)
   dataSource(['DataSource1.Type1'])
`;
```

필드 인덱스

필드 인덱스 기반 소스 선택은 필터가 인덱싱된 필드를 대상으로 할 때 관련 로그 그룹을 자동으로 식별하여 스캔 볼륨과 쿼리 런타임을 줄입니다.

`filterIndex`를 사용하여 쿼리에서 지정한 필드에 인덱싱된 로그 그룹만 검색하도록 강제하여 인덱스된 데이터만 반환합니다. 해당 필드에 인덱싱된 이러한 로그 그룹의 경우, 인덱싱된 필드의 쿼리에 지정된 필드가 포함된 로그 이벤트가 없는 로그 그룹을 건너뛰어 쿼리를 추가로 최적화합니다. 또한 해당 필드 인덱스의 쿼리에 지정된 값과 일치하는 이러한 로그 그룹의 로그 이벤트만 스캔하려고 시도하여 스캔 볼륨을 줄입니다. 필드 인덱스 및 필드 인덱스 생성 방법에 대한 자세한 내용은 필드 인덱스 생성을 참조하여 쿼리 성능을 개선하고 스캔 볼륨을 줄입니다.

SQL에서 filterIndex는 인덱스로 처리해야 하는 키 값 페어를 지정하는 데 사용됩니다. 구문은 다음과 같습니다.

```
SELECT * FROM `filterIndex('region' = 'us-east-1')`;
```

여기서 각 항목은 다음과 같습니다.

1. filterIndex(...)는 해당 값 내의 키 값을 필드 인덱스로 취급하도록 지정합니다. 각 키 값 페어는 쉼표로 구분됩니다(아래 예).
2. 'region' = 'us-east-1'은 적용할 실제 조건을 지정합니다.
   a. 참고: 대신 = 고객이 IN을 사용하여 여러 값을 지정할 수 있음(아래 예)

여러 filterIndex를 사용하면 "AND"를 사용하여 조건을 결합합니다. 이 예에서는 로그 일치 상태=200이고 us-east-1 또는 us-west-2의 리전이 쿼리됩니다.

```
SELECT * FROM `filterIndex('status' = 200, 'region' IN ['us-east-1', 'us-west-2'])`;
```

제한 사항

OpenSearch SQL을 사용하여 CloudWatch Logs Insights에서 쿼리할 때 다음 제한 사항이 적용됩니다.

- SELECT 명령문에는 JOIN을 하나만 포함할 수 있습니다.
- JOIN 또는 하위 쿼리는 데이터 소스 쿼리와 함께 사용할 수 없습니다.
- 중첩 하위 쿼리는 한 수준만 지원됩니다.
- 세미콜론(;)으로 구분된 여러 명령문 쿼리는 지원되지 않습니다.
- 동일하지만 경우에 따라서만 다른 필드 이름을 포함하는 쿼리(예: field1 및 FIELD1)는 지원되지 않습니다.

  예를 들어, 다음 쿼리는 지원되지 않습니다.

  ```
  Select AWSAccountId, AwsAccountId from LogGroup
  ```

  그러나 필드 이름(`@logStream`)이 두 로그 그룹에서 동일하기 때문에 다음 쿼리가 지원됩니다.

  ```
  Select a.`@logStream`, b.`@logStream` from Table A INNER Join Table B on a.id = b.id
  ```

- 함수와 표현식은 필드 이름에서 작동해야 하며 FROM 절에 지정된 로그 그룹이 있는 SELECT 명령문의 일부여야 합니다.

  예를 들어 다음 쿼리는 지원되지 않습니다.

```
SELECT cos(10) FROM LogGroup
```

이 쿼리는 다음과 같이 지원됩니다.

```
SELECT cos(field1) FROM LogGroup
```

- SQL 또는 PPL 명령을 사용하는 경우 특정 필드를 백틱에 묶어 성공적으로 쿼리합니다. 백틱은 특수 문자(비영문자 및 비숫자)가 있는 필드에 필요합니다. 예를 들어, 백틱에서 `@message`, `Operation.Export` 및 `Test::Field`을 묶습니다. 순전히 알파벳 이름의 필드를 백틱으로 묶을 필요는 없습니다.

  간단한 필드가 있는 예제 쿼리:

```
SELECT SessionToken, Operation, StartTime  FROM `LogGroup-A`
LIMIT 1000;
```

  백틱이 추가된 유사한 쿼리:

```
SELECT `@SessionToken`, `@Operation`, `@StartTime`  FROM `LogGroup-A` LIMIT 1000;
```

자연어를 사용하여 CloudWatch Logs Insights 쿼리를 생성하고 업데이트

CloudWatch Logs는 CloudWatch Logs Insights, OpenSearch Service PPL, OpenSearch Service SQL 및 CloudWatch Metrics Insights에 대한 쿼리를 생성하고 업데이트하는 데 도움이 되는 자연어 쿼리 기능을 지원합니다.

이 기능을 사용하면 찾고 있는 CloudWatch Logs 데이터에 대해 일반적 영어로 질문하거나 설명할 수 있습니다. 자연어 기능은 사용자가 입력한 프롬프트에 따라 쿼리를 생성하고 쿼리 작동 방식을 한 줄씩 설명합니다. 쿼리를 업데이트하여 데이터를 더 자세히 조사할 수도 있습니다.

환경에 따라 '전송된 바이트 기준으로 상위 100개의 소스 IP 주소는?' 및 '가장 슬로우한 Lambda 함수 요청 10개를 찾아 줘.' 같은 프롬프트를 입력할 수 있습니다.

> **Note**
>
> 자연어 쿼리 기능은 리전 서비스입니다. 일부 리전의 경우 이 기능은 쿼리 프롬프트를 처리하기 위해 미국 내 리전에 교차 리전 직접 호출을 수행합니다. 자세한 내용은 Amazon CloudWatch expands region support for natural language query result summarization and query generation을 참조하세요.

이 기능을 사용하여 CloudWatch Logs Insights 쿼리를 생성하려면 CloudWatch Logs Insights 쿼리 편집기를 열고 쿼리하려는 로그 그룹을 선택한 다음, 쿼리 생성을 선택합니다.

> **Important**
>
> 자연어 쿼리 기능을 사용하려면 CloudWatchLogsFullAccess, CloudWatchLogsReadOnlyAccess, AdministratorAccess 또는 ReadOnlyAccess IAM 정책에 로그인하거나 `cloudwatch:GenerateQuery` 권한이 있어야 합니다.

예제 쿼리

이 섹션의 예제에서는 자연어 기능을 사용하여 쿼리를 생성하고 업데이트하는 방법을 설명합니다.

> **Note**
>
> CloudWatch Logs Insights 쿼리 편집기 및 구문에 대한 자세한 내용은 CloudWatch Logs Insights 쿼리 구문을 참조하세요.

예: 자연어 쿼리 생성

자연어를 사용하여 쿼리를 생성하려면 프롬프트를 입력하고 새 쿼리 생성을 선택합니다. 이 예제에서는 기본 검색을 수행하는 쿼리를 보여줍니다.

프롬프트

다음은 가장 느린 Lambda 함수 간접 호출 10개를 검색하도록 기능에 지시하는 프롬프트의 예입니다.

```
Find the 10 slowest requests
```

Query

다음은 자연어 기능이 프롬프트를 기반으로 생성한 CloudWatch Logs Insights 쿼리 언어를 사용하는 쿼리입니다. 쿼리 전에 코멘트에 프롬프트가 어떻게 나타나는지 확인합니다. 쿼리가 끝나면 쿼리 작동 방식을 설명하는 설명을 읽을 수 있습니다.

```
# Find the 10 slowest requests
fields @timestamp, @message, @duration
| sort @duration desc
| limit 10
# This query retrieves the timestamp, message and duration fields from the logs and
 sorts them in descending order by duration to find the 10 slowest requests.
```

> **Note**
>
> 프롬프트 표시 및 쿼리 작동 방식에 대한 설명을 끄려면 편집기의 톱니바퀴 아이콘을 사용합니다.

프롬프트

OpenSearch SQL 쿼리를 생성하려면 OpenSearch SQL 탭을 선택한 다음 쿼리 생성기 프롬프트 상자를 열어 자연어 프롬프트를 입력합니다. 다음은 자연어 기능을 사용하여 OpenSearch SQL 쿼리를 생성하는 프롬프트의 예제입니다.

```
Give me the number of errors and exceptions per hour
```

Query

다음은 시간당 집계된 오류 및 예외 횟수를 찾는 데 사용할 수 있는 프롬프트에서 생성된 SQL 쿼리입니다.

```
SELECT DATE_FORMAT(`@timestamp`, 'yyyy-MM-dd HH') AS hour,
       COUNT(*) AS error_count
FROM `/aws/lambda/CloudWatchOdysseyQueryGen`
WHERE `@message` LIKE '%error%'
  OR `@message` LIKE '%exception%'
GROUP BY DATE_FORMAT(`@timestamp`, 'yyyy-MM-dd HH')
ORDER BY hour
```

프롬프트

OpenSearch PPL 쿼리를 생성하려면 OpenSearch PPL 탭을 선택한 다음 쿼리 생성기 프롬프트 상자를 열어 자연어 프롬프트를 입력합니다. 다음은 자연어 기능을 사용하여 OpenSearch PPL 쿼리를 생성하는 프롬프트의 예제입니다.

```
Give me all unique exception messages
```

Query

다음은 로그에서 고유한 예외 메시지를 찾는 데 사용할 수 있는 프롬프트에서 생성된 PPL 쿼리입니다.

```
dedup @message
| fields @message
```

예: 자연어 쿼리 업데이트

초기 프롬프트를 편집한 다음, 쿼리 업데이트를 선택하여 쿼리를 업데이트할 수 있습니다.

업데이트된 프롬프트

다음 예제는 이전 프롬프트의 업데이트된 버전을 보여줍니다. 이 프롬프트는 가장 느린 Lambda 함수 간접 호출 10개를 검색하는 프롬프트 대신 이제 가장 느린 Lambda 함수 간접 호출 20개를 검색하고 추가 로그 이벤트를 위한 다른 열을 포함하도록 기능에 지시합니다.

```
Show top 20 slowest requests instead and display requestId as a column
```

업데이트된 쿼리

다음은 CloudWatch Logs Insights 쿼리 언어를 사용하여 업데이트된 쿼리의 예제입니다. 업데이트된 쿼리가 나오기 전에 코멘트에 업데이트된 프롬프트가 어떻게 나타나는지 확인합니다. 쿼리가 끝나면 원래 쿼리가 어떻게 업데이트되었는지에 대한 설명을 읽을 수 있습니다.

```
# Show top 20 slowest requests instead and display requestId as a column
fields @timestamp, @message, @requestId, @duration
| sort @duration desc
| limit 20
# This query modifies the original query by replacing the @message field with the
 @requestId field and changing the limit from 10 to 20 to return the top 20 log events
 by duration instead of the top 10.
```

서비스 개선을 위한 데이터 사용 선택 해제

AI 모델을 학습시키고 관련 쿼리를 생성하기 위해 제공하는 자연어 프롬프트 데이터는 서비스를 제공하고 유지하는 용도로만 사용됩니다. 이 데이터는 CloudWatch Logs Insights의 품질을 개선하는 데 사용할 수 있습니다. 사용자의 신뢰와 개인 정보 보호는 물론 콘텐츠 보안도 저희의 최우선 과제입니다. 자세한 내용은 AWS 서비스 약관과 AWS 책임감 있는 AI 정책을 참조하세요.

AI 서비스 옵트아웃 정책을 생성하여 콘텐츠가 자연어 쿼리의 개발 또는 품질 향상에 사용되는 것을 거부할 수 있습니다. 쿼리 생성 기능을 포함하여 모든 CloudWatch Logs AI 기능의 데이터 수집을 거부하려면 CloudWatch Logs의 옵트아웃 정책을 생성해야 합니다. 자세한 내용은 AWS Organizations 사용 설명서의 AI 서비스 옵트아웃 정책을 참조하세요.

지원되는 로그 및 검색되는 필드

CloudWatch Logs Insights는 다양한 유형의 로그를 지원합니다. Amazon CloudWatch Logs에서 Standard 클래스 로그 그룹으로 전송되는 모든 로그에 대해 CloudWatch Log Insights는 자동으로 5개의 시스템 필드를 생성합니다.

- `@message`에는 구문 분석되지 않은 원시 로그 이벤트가 포함되어 있습니다. 이 필드는 InputLogevent의 `message` 필드와 동등합니다.
- `@timestamp`에는 로그 이벤트 `timestamp` 필드의 이벤트 타임스탬프가 포함되어 있습니다. 이 필드는 InputLogevent의 `timestamp` 필드와 동등합니다.
- `@ingestionTime`에는 CloudWatch Logs가 로그 이벤트를 수신한 시간이 포함되어 있습니다.
- `@logStream`에는 로그 이벤트가 추가된 로그 스트림의 이름이 포함되어 있습니다. 로그 스트림은 로그를 생성한 프로세스와 동일한 프로세스를 통해 로그를 그룹화합니다.
- `@log`은 *`account-id`*`:`*`log-group-name`* 형식의 로그 그룹 식별자입니다. 이 필드는 여러 로그 그룹을 쿼리할 때 특정 이벤트가 속한 로그 그룹을 식별하는 데 유용할 수 있습니다.
- `@entity`에는 관련 원격 측정 탐색 기능을 위한 엔터티와 관련된 평면화된 JSON이 포함되어 있습니다.

  예를 들어, 이 JSON은 개체를 나타낼 수 있습니다.

```
{
  "Entity": {
    "KeyAttributes": {
      "Type": "Service",
      "Name": "PetClinic"
    },
```

```
    "Attributes": {
      "PlatformType": "AWS::EC2",
      "EC2.InstanceId": "i-1234567890123"
    }
  }
}
```

이 개체의 경우 추출된 시스템 필드는 다음과 같습니다.

```
@entity.KeyAttributes.Type = Service
@entity.KeyAttributes.Name = PetClinic
@entity.Attributes.PlatformType = AWS::EC2
@entity.Attributes.EC2.InstanceId = i-1234567890123
```

> **Note**
>
> 필드 검색은 Standard 로그 클래스의 로그 그룹에서만 지원됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스를 참조하세요.

CloudWatch Logs Insights는 생성하는 필드의 시작 부분에 @ 기호를 삽입합니다.

다수의 로그 유형에 대해 CloudWatch Logs 역시 로그에 포함된 로그 필드를 자동으로 검색합니다. 다음 표에는 이러한 자동 검색 필드가 나와 있습니다.

CloudWatch Logs Insights가 자동으로 검색하지 않는 필드가 포함된 다른 로그 유형에 대해서는 `parse` 명령을 사용하여 추출하고 쿼리에 사용할 추출된 필드를 생성할 수 있습니다. 자세한 내용은 CloudWatch Logs Insights 언어 쿼리 구문 섹션을 참조하세요.

검색한 로그 필드의 이름이 @ 문자로 시작하면 CloudWatch Logs Insights는 해당 필드의 앞에 @를 추가로 붙여 표시합니다. 예를 들어, 로그 필드 이름이 `@example.com`이면 이 필드 이름은 `@@example.com`로 표시됩니다.

> **Note**
>
> `@message`, `@timestamp`또는 `@log`을 제외하고 검색된 필드에 대한 필드 인덱스를 생성할 수 있습니다. 필드 인덱스에 대한 자세한 내용은 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성을 참조하세요.

로그 유형	검색된 로그 필드
Amazon VPC 흐름 로그	`@timestamp` , `@logStream` , `@message`, `accountId` , `endTime`, `interfaceId` , `logStatus` , `startTime` , `version`, `action`, `bytes`, `dstAddr`, `dstPort`, `packets`, `protocol`, `srcAddr`, `srcPort`
Route 53 로그	`@timestamp` , `@logStream` , `@message`, `edgeLocation` , `ednsClientSubnet` , `hostZoneId` , `protocol`, `queryName` , `queryTimestamp` , `queryType` , `resolverIp` , `responseCode` , `version`
Lambda 로그	`@timestamp` , `@logStream` , `@message`, `@requestId` , `@duration`, `@billedDuration` , `@type`, `@maxMemoryUsed` , `@memorySize` Lambda 로그 행에 X-Ray 트레이스 ID가 포함된 경우 `@xrayTraceId` 및 `@xraySegmentId` 필드도 포함됩니다. CloudWatch Logs Insights는 Lambda 로그에서 로그 필드를 자동으로 검색하지만 각 로그 이벤트에 포함된 첫 번째 JSON 조각에 대해서만 검색합니다. Lambda 로그 이벤트에 JSON 조각이 여러 개 포함된 경우 **`parse`** 명령을 사용하여 로그 필드를 구문 분석하고 추출할 수 있습니다. 자세한 정보는 JSON 로그의 필드 섹션을 참조하세요.
CloudTrail 로그 JSON 형식의 로그	자세한 정보는 JSON 로그의 필드 섹션을 참조하세요.
기타 로그 유형	`@timestamp` , `@ingestionTime` , `@logStream` , `@message`, `@log`.

JSON 로그의 필드

CloudWatch Logs Insights에서는 점 표기법을 사용하여 JSON 필드를 나타냅니다. 이 섹션에는 점 표기법을 사용하여 JSON 필드에 액세스하는 방법을 보여주는 JSON 이벤트 예제와 코드 조각이 포함되어 있습니다.

예: JSON 이벤트

```
{
```

```
    "eventVersion": "1.0",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn: aws: iam: : 123456789012: user/Alice",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "accountId": "123456789012",
        "userName": "Alice"
    },
    "eventTime": "2014-03-06T21: 22: 54Z",
    "eventSource": "ec2.amazonaws.com",
    "eventName": "StartInstances",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "ec2-api-tools1.6.12.2",
    "requestParameters": {
        "instancesSet": {
            "items": [
                {
                    "instanceId": "i-abcde123"
                }
            ]
        }
    },
    "responseElements": {
        "instancesSet": {
            "items": [
                {
                    "instanceId": "i-abcde123",
                    "currentState": {
                        "code": 0,
                        "name": "pending"
                    },
                    "previousState": {
                        "code": 80,
                        "name": "stopped"
                    }
                }
            ]
        }
    }
}
```

예제 JSON 이벤트에는 이름이 userIdentity로 지정된 객체가 포함되어 있고, userIdentity에는 이름이 type으로 지정된 필드가 포함되어 있습니다. 점 표기법을 사용하여 type 값을 표시하려면 userIdentity.type을 사용합니다.

예제 JSON 이벤트에는 중첩 필드 이름 및 값 목록으로 병합되는 배열이 포함되어 있습니다. requestParameters.instancesSet의 첫 번째 항목에 대해 instanceId 값을 표시하려면 requestParameters.instancesSet.items.0.instanceId를 사용합니다. 필드 instanceID 앞에 위치한 숫자 0은 필드 items 값의 위치를 나타냅니다. 다음 예제에는 JSON 로그 이벤트의 중첩된 JSON 필드에 액세스하는 방법을 보여주는 코드 조각이 포함되어 있습니다.

예: 쿼리

```
fields @timestamp, @message
| filter requestParameters.instancesSet.items.0.instanceId="i-abcde123"
| sort @timestamp desc
```

중첩된 JSON 필드 instanceId 값에 액세스하기 위해 filter 명령과 함께 점 표기법을 사용하는 쿼리를 보여주는 코드 조각입니다. instanceId 값이 "i-abcde123"와 같은 메시지를 필터링하고 지정된 값을 포함하는 모든 로그 이벤트를 반환하는 쿼리입니다.

점 표기법 및 JSON 인코딩 문자열 필드

점 표기법은 수집 시 구조적으로 중첩된 JSON 객체로 저장된 필드만 통과합니다. 필드의 값이 JSON 인코딩 문자열(콘텐츠가 유효한 JSON인 문자열)인 경우 점 표기법은 이를 불투명 리프 값으로 취급하고 해당 값 내의 하위 필드에 액세스하지 않습니다.

이는 일반적으로 다음과 같은 경우에 발생합니다.

- OCSF 파이프라인에 의해 변환된 로그(예: OCSF 스키마json_t에 로 api.request.data 입력됨)
- 수집 전에 변수 구조 페이로드가 문자열로 직렬화되는 모든 로그 소스

예를 들어 api.request.data가 문자열 값으로 {"startTime":1234,"endTime":5678} 포함된 경우 CloudWatch Logs Insights는 전체 문자열을 중첩된 객체가 아닌 단일 리프 값으로 취급하므로 는 결과를 api.request.data.startTime 반환하지 않습니다.

JSON 인코딩 문자열 내의 하위 필드에 액세스하려면 jsonParse를 사용하여 문자열을 순회 맵으로 변환합니다.

예: JSON 인코딩 문자열 필드에 jsonParse를 사용하여 쿼리

```
fields jsonParse(api.request.data).startTime as startTime
```

이 쿼리는 jsonParse를 사용하여의 JSON 인코딩 문자열을 맵api.request.data으로 구문 분석한 다음 점 표기법으로 startTime 하위 필드에 액세스합니다. 에 대한 자세한 내용은 구조 유형을 jsonParse참조하세요.

> **Note**
>
> CloudWatch Logs Insights는 하나의 JSON 로그에서 최대 200개의 로그 이벤트 필드를 추출할 수 있습니다. 추출되지 않은 추가 필드의 경우 parse 명령을 사용하여 메시지 필드의 구문 분석되지 않은 원시 로그 이벤트에서 이러한 필드를 추출할 수 있습니다. parse 명령에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Query syntax을 참조하세요.

쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성

효율적인 등식 기반 검색을 위해 로그 이벤트에 필드의 필드 인덱스를 생성할 수 있습니다. 그런 다음 CloudWatch Logs Insights 쿼리에서 필드 인덱스를 사용하면 쿼리는 인덱싱된 필드를 포함하지 않는 것으로 알려진 로그 이벤트 처리를 건너뛰려고 시도합니다. 이렇게 하면 필드 인덱스를 사용하는 쿼리의 스캔 볼륨이 줄어들어 결과를 더 빠르게 반환할 수 있습니다. 이를 통해 수천 개의 로그 그룹에서 페타바이트 단위의 총 로그를 빠르게 검색하고 관련 로그를 더 빠르게 파악할 수 있습니다. 인덱싱할 좋은 필드는 자주 쿼리해야 하는 필드입니다. 값의 카디널리티가 높은 필드도 필드 인덱스의 좋은 후보입니다. 이러한 필드 인덱스를 사용하는 쿼리는 대상 값과 매칭하는 로그 이벤트를 제한하기 때문에 더 빠르게 완료되기 때문입니다.

예를 들어, requestId에 대한 필드 인덱스를 생성했다고 가정해 보겠습니다. 그런 다음 해당 인덱싱된 필드와 쿼리된 값을 포함하는 것으로 알려진 로그 이벤트만 처리requestId = *value*하거나 처리requestId IN [*value*, *value*, ...]하려고 시도하는 해당 로그 그룹에 대한 모든 CloudWatch Logs Insights 쿼리는 CloudWatch Logs가 과거에 해당 필드의 값을 감지했습니다.

필드 인덱스를 사용하여 더 많은 수의 로그 그룹에 대한 효율적인 쿼리를 생성할 수도 있습니다. filterIndex 명령 대신 쿼리에서 filter 명령을 사용하면 필드 인덱스가 있는 로그 이벤트에서 선택한 로그 그룹에 대해 쿼리가 실행됩니다. 이러한 쿼리는 최대 5개의 로그 그룹 이름 접두사를 지정하여 선택한 최대 10,000개의 로그 그룹을 스캔할 수 있습니다. CloudWatch 교차 계정 관찰성의 모니터링 계정인 경우 모든 소스 계정을 선택하거나 개별 소스 계정을 지정하여 로그 그룹을 선택할 수 있습니다.

인덱스 필드는 대/소문자를 구분합니다. 예를 들어, RequestId의 필드 인덱스는 requestId가 포함된 로그 이벤트와 일치하지 않습니다.

필드 인덱스는 JSON 및 서비스 로그의 구조화된 로그 형식에만 지원됩니다.

CloudWatch Logs는 Standard 로그 클래스의 모든 로그 그룹에 대한 기본 필드 인덱스를 제공합니다. 기본 필드 인덱스는 다음 필드에 자동으로 사용할 수 있습니다.

- @logStream
- @aws.region
- @aws.account
- @source.log
- @data_source_name
- @data_source_type
- @data_format
- traceId
- severityText
- attributes.session.id

CloudWatch Logs는 특정 데이터 소스 이름 및 유형 조합에 대한 기본 필드 인덱스도 제공합니다. 기본 필드 인덱스는 다음 데이터 소스 이름 및 유형 조합에 자동으로 사용할 수 있습니다.

데이터 소스 이름 및 유형	기본 필드 인덱스
amazon_vpc.flow	action logStatus region flowDirection type
amazon_route53.resolver_query	query_type

데이터 소스 이름 및 유형	기본 필드 인덱스
	transport rcode
aws_waf.access	action httpRequest.country
aws_cloudtrail.data aws_cloudtrail.management	eventSource eventName awsRegion userAgent errorCode eventType managementEvent readOnly eventCategory requestId

기본 필드 인덱스는 정책 내에서 정의하는 사용자 지정 필드 인덱스에 추가됩니다. 기본 필드 인덱스는 필드 인덱스 할당량에 포함되지 않습니다.

CloudWatch Logs는 인덱스 정책이 생성된 후 수집된 로그 이벤트만 인덱싱합니다. 정책이 생성되기 전에 수집된 로그 이벤트는 인덱싱하지 않습니다. 필드 인덱스를 생성한 후 각 매칭 로그 이벤트는 로그 이벤트의 수집 시간으로부터 30일 동안 인덱싱된 상태로 유지됩니다.

> **Note**
>
> 모니터링 계정에서 필드 인덱스 정책을 생성하는 경우 연결된 소스 계정의 로그 그룹에는 해당 정책이 사용되지 않습니다. 필드 인덱스 정책은 생성된 계정에만 적용됩니다.

이 섹션의 나머지 주제에서는 자동으로 인덱싱된 필드와 필드 인덱스를 생성하는 방법을 설명합니다. 쿼리에서 필드 인덱스 참조에 대한 자세한 내용은 filterIndex 및 필터을 참조하세요.

주제

- 자동으로 인덱싱된 필드
- 필드 인덱스 구문 및 할당량
- 계정 수준 필드 인덱스 정책 생성
- 로그 그룹 수준 필드 인덱스 정책 생성
- 쿼리 생성 시 로그 그룹 선택 옵션
- 필드 인덱스 정책 삭제의 영향

자동으로 인덱싱된 필드

CloudWatch Logs는 = 또는 `IN` 연산자를 사용하는 CloudWatch Logs Insights 등식성 필터에서 최근 사용을 기반으로 필드를 자동으로 인덱싱합니다. 이러한 인덱스는 정책에서 구성하는 기본 필드 인덱스 및 필드 인덱스에 추가됩니다. 자동으로 인덱싱된 필드에 대해 정책을 구성할 필요가 없습니다.

CloudWatch Logs는 최근 쿼리 활동을 기반으로 자동으로 인덱싱된 필드 세트를 업데이트합니다. CloudWatch Logs가 세트에서 필드를 제거하면 해당 필드에 대해 새로 수집된 이벤트 인덱싱이 중지됩니다. CloudWatch Logs는 자동으로 인덱싱된 필드를 관리하고 30일 동안 보관합니다. 필드를 영구적으로 인덱싱하려면 계정 수준 또는 로그 그룹 수준 필드 인덱스 정책에 추가합니다.

> ⚠ 자동 인덱싱된 필드에 filterIndex 대신 필터 사용
>
> 자동 인덱싱된 필드 `filter` 대신 `filterIndex`를 사용하는 것이 좋습니다. CloudWatch Logs는 쿼리 패턴에 따라 이러한 필드를 업데이트하거나 제거할 수 있습니다. CloudWatch Logs는 30일 동안만 로그를 보존합니다. `filterIndex` 명령은 인덱싱된 데이터만 반환합니다. CloudWatch Logs는 쿼리 활동을 기반으로 세트를 업데이트하므로 CloudWatch Logs가 필드를 선택하기 전에 또는 CloudWatch Logs가 인덱싱을 CloudWatch 중지한 후에 수집된 이벤트를 `filterIndex` 쿼리가 검색하지 않습니다. 를 사용하는 경우 필드 인덱스 정책에 필드를 `filterIndex`추가하여 쿼리 활동에 관계없이 인덱싱된 상태를 유지합니다. 자세한 내용은 filterIndex와 filter 비교 섹션을 참조하세요.

로그 그룹의 필드 인덱스 탭에는 자동으로 인덱싱된 필드가 나열됩니다.

DescribeFieldIndexes 작업을 사용하여 자동으로 인덱싱된 필드를 나열할 수도 있습니다. indexCategories 요청 파라미터AUTO에를 포함합니다. 자동 인덱싱된 필드에 AUTO 대한 응답은 indexCategory 로 설정됩니다. 생략된 요청은 자동으로 인덱싱된 필드를 반환하지 indexCategories 않습니다.

로그 그룹 수준 인덱스 정책에 자동으로 인덱싱된 필드를 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택합니다.
4. 필드 인덱스 탭을 선택한 다음 필드 인덱스 관리를 선택합니다.
5. 로그 그룹 수준 필드 인덱스 관리 페이지에서 정책의 필드를 검토합니다. 필요에 따라 필드를 추가하거나 제거한 다음 저장을 선택합니다.

필드 인덱스 구문 및 할당량

필드 인덱스 정책을 생성하여 필드 인덱스를 생성합니다. 전체 계정에 적용되는 계정 수준 인덱스 정책을 생성할 수 있으며 단일 로그 그룹에만 적용되는 정책을 생성할 수도 있습니다. 계정 전체 인덱스 정책의 경우 계정의 모든 로그 그룹에 적용되는 정책을 보유할 수 있습니다. 로그 그룹 이름의 접두사로 선택한 계정의 로그 그룹 하위 집합에 적용되는 계정 수준 인덱스 정책을 생성할 수도 있습니다. 동일한 계정에 여러 계정 수준 정책이 있는 경우 이러한 정책의 로그 그룹 이름 접두사는 겹칠 수 없습니다. 마찬가지로 특정 데이터 소스 이름 및 유형 조합에 적용되는 계정 수준 인덱스 정책을 생성할 수 있습니다. 데이터 소스 이름 및 유형 조합당 하나의 계정 정책만 생성할 수 있습니다.

로그 그룹 수준 필드 인덱스 정책은 로그 그룹 전체에 적용되는 계정 수준 필드 인덱스 정책(예: 선택 기준이 없거나 로그 그룹 이름 접두사 기반 선택 기준이 있는 계정 수준 정책)을 재정의합니다. 로그 이벤트 수준에서 일치하는 계정 수준 정책(예: 지정된 데이터 소스 이름 및 유형 조합)은 로그 그룹 전체와 일치하는 정책 외에도 적용됩니다. 로그 그룹 수준 인덱스 정책을 생성하는 경우 해당 로그 그룹은 로그 그룹 수준에서 일치하는 계정 수준 정책을 사용하지 않습니다.

로그 이벤트와 필드 인덱스 이름의 매칭은 대/소문자를 구분합니다. 예를 들어, RequestId의 필드 인덱스는 requestId가 포함된 로그 이벤트와 일치하지 않습니다.

최대 40개의 계정 수준 인덱스 정책을 보유할 수 있으며, 이러한 정책 중 20개는 로그 그룹 이름 접두사 선택 기준을 사용할 수 있고 20개는 데이터 소스 기반 선택 기준을 사용할 수 있습니다. 로그 그룹 이름 접두사로 필터링된 계정 수준 인덱스 정책이 여러 개 있는 경우 두 개는 동일하거나 중복되는 로그 그룹 이름 접두사를 사용할 수 없습니다. 예를 들어, 하나의 정책을 my-log로 시작하는 로그 그룹에 필

터링한 경우 다른 필드 인덱스 정책을 `my-logpprod` 또는 `my-logging`로 필터링할 수 없습니다. 마찬가지로 데이터 소스 이름 및 유형 조합으로 필터링된 계정 수준 인덱스 정책이 여러 개 있는 경우 둘 다 동일한 데이터 소스 이름 및 유형을 사용할 수 없습니다. 예를 들어 데이터 소스 이름 `amazon_vpc` 및 데이터 소스 유형으로 필터링된 정책이 하나 있는 경우이 조합으로 다른 정책을 생성할 `flow` 수 없습니다.

이름 접두사가 없고 모든 로그 그룹에 적용되는 계정 수준 인덱스 정책이 있는 경우 로그 그룹 이름 접두사 필터가 있는 다른 계정 수준 인덱스 정책을 생성할 수 없습니다. 데이터 소스 이름 및 유형 필터를 사용하는 계정 수준 인덱스 정책을 생성할 수 있습니다.

각 인덱스 정책에는 다음과 같은 할당량 및 제한이 있습니다.

- 정책에 최대 20개의 필드를 포함할 수 있습니다.
- 각 필드 이름은 최대 100자를 포함할 수 있습니다.
- `@`로 시작하는 로그 그룹에 사용자 지정 필드의 인덱스를 생성하려면 필드 이름 시작 부분에 추가 `@`이 있는 필드를 지정해야 합니다. 예를 들어, 로그 이벤트에 이름이 `@userId`인 필드가 포함된 경우 `@@userId`을 지정하여 이 필드에 대한 인덱스를 생성해야 합니다.

데이터 소스 이름 및 유형 기반 선택 기준이 있는 계정 수준 인덱스 정책의 경우 추가 제한이 적용됩니다. 모든 필드는 기본 데이터 형식이어야 하며 중첩된 기본 항목은 구조체에서만 지원됩니다.

생성된 필드 및 예약된 필드

CloudWatch Logs Insights는 각 로그 이벤트에서 시스템 필드를 자동으로 생성합니다. 생성된 필드에는 `@`로 접두사가 붙습니다. 생성된 필드에 대한 자세한 내용은 지원되는 로그 및 검색되는 필드을 참조하세요.

생성된 필드 중에서 필드 인덱스로 사용할 수 있는 필드는 다음과 같습니다.

- `@logStream`
- `@ingestionTime`
- `@requestId`
- `@type`
- `@initDuration`
- `@duration`
- `@billedDuration`

- @memorySize
- @maxMemoryUsed
- @xrayTraceId
- @xraySegmentId

생성된 필드를 인덱싱하려면 @로 시작하는 사용자 지정 필드에 대해 해야 하므로 지정할 때 추가 @를 추가할 필요가 없습니다. 예를 들어, @logStream에 대한 필드 인덱스를 생성하려면 @logStream을 필드 인덱스로 지정하면 됩니다.

CloudWatch Logs는 Standard 로그 클래스의 모든 로그 그룹에 대한 기본 필드 인덱스를 제공합니다. 기본 필드 인덱스는 다음 필드에 자동으로 사용할 수 있습니다.

- @logStream
- @aws.region
- @aws.account
- @source.log
- @data_source_name
- @data_source_type
- @data_format
- traceId
- severityText
- attributes.session.id

CloudWatch Logs는 특정 데이터 소스 이름 및 유형 조합에 대한 기본 필드 인덱스도 제공합니다. 기본 필드 인덱스는 다음 데이터 소스 이름 및 유형 조합에 자동으로 사용할 수 있습니다.

데이터 소스 이름 및 유형	기본 필드 인덱스
amazon_vpc.flow	action logStatus region flowDirection

데이터 소스 이름 및 유형	기본 필드 인덱스
	type
amazon_route53.resolver_query	query_type transport rcode
aws_waf.access	action httpRequest.country
aws_cloudtrail.data aws_cloudtrail.management	eventSource eventName awsRegion userAgent errorCode eventType managementEvent readOnly eventCategory requestId

기본 필드 인덱스는 정책 내에서 정의하는 사용자 지정 필드 인덱스에 추가됩니다. 기본 필드 인덱스는 필드 인덱스 할당량에 포함되지 않습니다.

JSON 로그의 하위 필드 및 배열 필드

JSON 로그에서 중첩된 하위 필드 또는 배열 필드인 필드를 인덱싱할 수 있습니다.

예를 들어, 이 로그의 필드 내 accessKeyId 하위 userIdentity 필드의 인덱스를 생성할 수 있습니다.

```
{
    "eventVersion": "1.0",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EXAMPLE_PRINCIPAL_ID",
        "arn": "arn: aws: iam: : 123456789012: user/Alice",
        "accessKeyId": "11112222",
        "accountId": "123456789012",
        "userName": "Alice"
    },
    "eventTime": "2014-03-06T21: 22: 54Z",
    "eventSource": "ec2.amazonaws.com",
    "eventName": "StartInstances",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "ec2-api-tools1.6.12.2",
    "requestParameters": {
        "instancesSet": {
            "items": [{
                "instanceId": "i-abcde123",
                "currentState": {
                    "code": 0,
                    "name": "pending"
                },
                "previousState": {
                    "code": 80,
                    "name": "stopped"
                }
            }]
        }
    }
}
```

이 필드를 생성하려면 필드 인덱스를 생성할 때와 쿼리에서 지정할 때 점 표기법(userIdentity.accessKeyId)을 사용하여 참조합니다. 쿼리는 다음과 같을 수 있습니다.

```
fields @timestamp, @message
| filterIndex userIdentity.accessKeyId = "11112222"
```

이전 예제 이벤트에서 `instanceId` 필드는 `requestParameters.instancesSet.items` 내의 배열에 있습니다. 필드 인덱스를 생성할 때와 쿼리할 때 이 필드를 나타내려면 `requestParameters.instancesSet.items.0.instanceId`로 참조합니다. `0`은 배열에서 해당 필드의 위치를 참조냅니다.

그러면이 필드에 대한 쿼리는 다음과 같을 수 있습니다.

```
fields @timestamp, @message
| filterIndex requestParameters.instancesSet.items.0.instanceId="i-abcde123"
```

계정 수준 필드 인덱스 정책 생성

이 섹션의 단계를 사용하여 계정의 모든 로그 그룹 또는 동일한 문자열로 시작하는 로그 그룹 이름이 있는 여러 로그 그룹에 적용되는 필드 인덱스 정책을 생성합니다.

계정 수준 필드 인덱스 정책을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 설정을 선택한 다음, 로그 탭을 선택합니다.
3. 계정 수준 인덱스 정책 섹션에서 관리를 선택합니다.
4. 인덱스 정책 생성을 선택합니다.
5. 정책 이름에 새 정책의 이름을 입력합니다.
6. 정책 범위 선택에서 다음 중 하나를 수행합니다.
   - 모든 표준 로그 그룹을 선택하여 계정의 모든 Standard 클래스 로그 그룹에 인덱스 정책을 적용합니다.
   - 접두사 일치별 로그 그룹을 선택하여 이름이 모두 동일한 문자열로 시작하는 로그 그룹의 하위 집합에 정책을 적용합니다. 그런 다음 접두사 이름 입력에 이러한 로그 그룹의 접두사를 입력합니다.

     접두사를 입력한 후 접두사 매칭된 로그 그룹 미리 보기를 선택하여 접두사가 예상된 로그 그룹을 매칭하는지 확인할 수 있습니다.

     데이터 소스별 데이터 로깅을 선택하여 정책을 특정 데이터 소스 이름 및 유형 조합에 적용합니다. 그런 다음 드롭다운 메뉴에서 데이터 소스 및 데이터 유형을 선택할 수 있습니다.

     데이터 소스 이름과 유형을 선택한 후 필드 가져오기를 선택하여 필드 인덱스 및 패싯 구성 섹션을 사용 가능한 필드, 포함된 로그 그룹, 기본 및 사용자 지정 필드 인덱스와 같은 관련 정보로 채울 수 있습니다.

7. 사용자 지정 인덱스 필드 구성에서 필드 경로 추가를 선택하여 인덱싱할 첫 번째 필드를 입력합니다.

   그런 다음 필드 이름의 값으로 사용할 문자열을 입력하거나 드롭다운 메뉴에서 필드를 선택합니다. 이는 로그 이벤트에 표시되는 것과 정확한 사례 매칭이어야 합니다. 예를 들어, 로그 이벤트에 `requestId`이 포함된 경우 여기에 `requestId`을 입력해야 합니다. `RequestId`, `requestID` 및 `request Id`은 매칭하지 않습니다.

   @ 문자로 시작하는 사용자 지정 로그 필드를 인덱싱하려면 인덱스 문자열을 입력할 때 추가 @ 문자를 포함해야 합니다. 예를 들어, 사용자 지정 로그 필드 `@emailname`이 있는 경우 필드 경로 추가 상자에 `@@emailname`을 입력합니다.

   CloudWatch Logs가 자동으로 생성하는 `@ingestionTime` 및 `@logStream` 필드에 대한 인덱스를 생성할 수도 있습니다. 이렇게 하면 지정할 때 추가 @를 추가할 필요가 없습니다.
8. (선택 사항)필드 경로를 지정하는 것 외에도 패싯으로 설정을 선택하여 필드를 패싯으로 생성할 수 있습니다.
9. 이전 단계를 반복하여 최대 20개의 필드 인덱스를 추가합니다.
10. 모두 마쳤으면 생성을 선택합니다.

로그 그룹 수준 필드 인덱스 정책 생성

이 섹션의 단계를 사용하여 단일 로그 그룹에 적용되는 필드 인덱스 정책을 생성합니다.

로그 그룹 수준 필드 인덱스 정책을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택합니다.
4. 필드 인덱스 탭을 선택합니다.
5. 이 로그 그룹에 대한 필드 인덱스 관리를 선택합니다.
6. 로그 그룹 수준 필드 인덱스 관리에서 필드 경로 추가를 선택하여 인덱싱할 첫 번째 필드를 입력합니다.

   그런 다음 필드 이름의 값으로 사용할 문자열을 입력합니다. 이는 로그 이벤트에 표시되는 것과 정확한 사례 매칭이어야 합니다. 예를 들어, 로그 이벤트에 `requestId`이 포함된 경우 여기에 `requestId`을 입력해야 합니다. `RequestId`, `requestID` 및 `request Id`은 매칭하지 않습니다.

@ 문자로 시작하는 사용자 지정 로그 필드를 인덱싱하려면 인덱스 문자열을 입력할 때 추가 @ 문자를 포함해야 합니다. 예를 들어, 사용자 지정 로그 필드 @emailname이 있는 경우 필드 경로 추가 상자에 @@emailname을 입력합니다.

CloudWatch Logs가 자동으로 생성하는 @ingestionTime 및 @logStream 필드에 대한 인덱스를 생성할 수도 있습니다. 이렇게 하면 지정할 때 추가 @을 추가할 필요가 없습니다.

7. (선택 사항) 필드 경로를 지정하는 것 외에도 패싯으로 설정을 선택하여 필드를 패싯으로 생성할 수 있습니다.
8. 이전 단계를 반복하여 최대 20개의 필드 인덱스를 추가합니다.
9. 완료되면 저장을 선택합니다.

쿼리 생성 시 로그 그룹 선택 옵션

이 섹션에서는 쿼리에 포함할 로그 그룹을 선택할 수 있는 다양한 방법을 설명합니다.

콘솔에서 쿼리에 대한 로그 그룹을 선택하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 인사이트를 선택합니다.
3. 이 쿼리에 사용하려는 쿼리 언어를 선택합니다. Logs Insights QL, OpenSearch PPL 또는 OpenSearch SQL 중 하나를 선택할 수 있습니다.
4. 쿼리에 대한 로그 그룹을 선택하는 세 가지 방법이 있습니다.
   - 로그 그룹 이름 상자를 사용합니다. 이는 기본 선택 방법입니다. 이 방법을 사용하여 최대 50개의 로그 그룹 이름을 입력할 수 있습니다. CloudWatch 크로스 계정 관측성에서 모니터링 계정인 경우 소스 계정과 모니터링 계정에서 로그 그룹을 선택할 수 있습니다. 하나의 쿼리로 다른 계정의 로그를 한 번에 쿼리할 수 있습니다.
   - 로그 그룹 기준 섹션을 사용합니다. 이 섹션에서는 로그 그룹 이름의 접두사를 기반으로 로그 그룹을 선택할 수 있습니다. 하나의 쿼리에 최대 5개의 접두사를 포함할 수 있습니다. 이름에 이러한 접두사가 있는 로그 그룹이 선택됩니다. 또는 모든 로그 그룹 옵션은 계정에서 모든 로그 그룹을 선택합니다.
   - CloudWatch 크로스 계정 관측성의 모니터링 계정인 경우 계정 드롭다운 메뉴에서 모든 계정을 선택하여 연결된 모든 계정의 로그 그룹을 선택할 수 있습니다. 또는 이 쿼리에 포함할 계정을 개별적으로 선택할 수 있습니다.

   선택 항목이 10,000개 이상의 로그 그룹을 매칭하면 선택 범위를 좁히라는 오류가 표시됩니다.

5. 쿼리의 기본 로그 클래스는 Standard입니다. 로그 클래스를 사용하여 Infrequent Access로 변경할 수 있습니다.

사용 AWS CLI

명령 줄에서 쿼리를 시작할 때 이러한 유형의 선택을 하려면 쿼리에서 `source` 명령을 사용할 수 있습니다. 자세한 내용과 예제는 소스을 참조하세요.

필드 인덱스 정책 삭제의 영향

한동안 유효한 필드 인덱스 정책을 삭제하면 다음과 같은 상황이 발생합니다.

- 정책이 삭제된 후 최대 30일 동안 쿼리는 여전히 인덱싱된 로그 이벤트의 이점을 누릴 수 있습니다.
- 로그 그룹 수준 인덱스 정책을 삭제하고 해당 로그 그룹에 적용되는 계정 수준 정책이 이미 있는 경우 계정 수준 정책은 결국 해당 로그 그룹에 적용됩니다.

패싯을 사용하여 로그 그룹화 및 탐색

패싯은 쿼리를 실행하지 않고도 데이터를 대화식으로 필터링하고 그룹화할 수 있으므로 로그를 분석하는 데 유용합니다. 패싯은 로그 그룹의 필터링, 집계 및 분석을 활성화하는 로그의 필드(예: `ServiceName` 또는 `StatusCode`)입니다. CloudWatch Logs Insights 콘솔에서 패싯 필드 목록과 선택한 시간 범위를 기반으로 각 패싯 값에 대한 로그 이벤트 수를 볼 수 있습니다. 다양한 패싯과 값을 선택하면 패싯 값과 개수가 실시간으로 업데이트되므로 로그를 대화식으로 탐색할 수 있습니다.

각 패싯은 선택한 시간 범위 및 쿼리 범위를 기반으로 로그의 필드에서 자동으로 추출되어 30일 동안 보관된 사용 가능한 값과 개수를 표시합니다. 표시된 패싯 수는 근사치입니다. 데이터 소스 이름 또는 데이터 소스 유형과 같은 기본 패싯을 사용하여 로그를 탐색하거나 로그의 모든 필드에 사용자 지정 패싯을 생성할 수 있습니다. 데이터 소스 이름은 로그(예: Route 53, Amazon VPC 또는 CloudTrail)를 생성하는 AWS 서비스 또는 애플리케이션이며 데이터 소스 유형은 해당 서비스에서 생성되는 특정 유형의 로그입니다. 기본 패싯은 CloudWatch에서 생성하며 `@aws.region`, `@data_source_name`, 및 `@data_source_type`를 포함합니다`@data_format`. 자세한 내용은 로그 관리 섹션을 참조하세요. 패싯은 계정에서 수집된 로그에만 사용할 수 있습니다. 교차 계정 관찰성을 설정한 경우 모니터링 계정은 소스 계정의 로그를 기반으로 패싯을 볼 수 없습니다.

원격 측정을 위한 리소스 태그가 활성화된 경우 리소스 태그 필드(`@aws.tag.*`)가 기본 패싯으로 자동으로 추가됩니다. 이러한 패싯을 사용하면 로그 이벤트를 생성한 리소스의 태그를 기준으로 로그 이벤트를 필터링하고 그룹화할 수 있습니다.

추가 패싯을 생성하려면 문제 해결과 관련된 로그의 필드를 선택하고 인덱스 정책을 사용하여 구성합니다. 사용자 지정 패싯의 경우 카디널리티가 낮은 필드(상태 및 ApplicationName과 같이 하루에 고유 값이 100개 미만인 필드)에서 생성하는 것이 좋습니다. 하루에 고유 값이 100개를 초과하는 패싯은 높은 카디널리티로 분류되며 이러한 패싯의 값은 표시되지 않습니다. 하나 이상의 패싯을 선택하고 로그에서 쿼리를 실행하도록 선택합니다.

CloudWatch Logs Insights에서 패싯을 시작하려면:

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 인사이트를 선택합니다.
3. (선택 사항) 시간 범위 선택기를 사용하여 분석하려는 기간을 선택합니다. 선택한 시간 범위에서 사용 가능한 패싯과 값이 패널에 표시됩니다.
4. 패싯을 선택하여 데이터를 탐색하고 패싯 전반의 값 분포에 대한 실시간 업데이트를 확인합니다.

   고유 값이 100개를 초과하는 패싯은 표시되지 않습니다. 특정 값을 쿼리하려면 대신 쿼리에 필터를 사용합니다.

패싯 기반 쿼리를 실행하려면

1. 패싯에서 하나 이상의 값을 선택합니다.
2. 이벤트 수는 선택한 패싯과 값에 따라 업데이트됩니다.
3. 패싯 값을 선택하면 선택 사항을 반영하도록 쿼리 범위가 업데이트됩니다.
4. 패싯 값을 선택한 후 실행을 선택하여 쿼리를 실행합니다.
5. 패싯당 지원되는 고유 값의 최대 수는 100개입니다. 예를 들어 패싯에 대한 값이 100개를 초과하는 경우 모든 개수가 "-"로 표시되어 값을 알 수 없음을 나타냅니다.

패싯 기반 쿼리를 저장하려면

1. 하나 이상의 패싯 값을 사용하여 쿼리를 생성합니다.
2. 나머지 단계는 Logs Insights 쿼리를 저장하는 것과 동일합니다. CloudWatch Logs Insights 쿼리 저장을 참조하세요.
3. 저장된 쿼리는 저장된 쿼리 섹션에서 사용할 수 있습니다. 저장된 쿼리를 검색하면 쿼리에 사용되는 패싯과 값이 자동으로 포함되므로 로그를 쉽게 분석할 수 있습니다.

계정 수준 패싯을 생성하려면

1. 패싯을 생성하려면 먼저 필드를 인덱스로 생성하고 패싯으로 구성해야 합니다. 탐색 창에서 설정, 로그, 계정 수준 인덱스 정책을 선택합니다. 또는 패싯 패널에서 패싯 관리를 선택할 수 있습니다.
2. 새 인덱스 정책 생성을 선택합니다. 인덱스 정책 생성에 대한 자세한 내용은 섹션을 참조하세요계정 수준 필드 인덱스 정책 생성.
3. 패싯을 생성하려면 인덱스 정책 생성 페이지에서 선택한 필드에 대해 패싯으로 설정을 선택합니다.

APIs 사용한 패싯 관리

패싯 관리는 필드 인덱스 정책을 사용하여 수행할 수 있습니다. 자세한 내용은 `field index` APIs.

필드 인덱스 APIs

아니요	이름	설명
1	PutIndexPolicy	특정 로그 그룹에 대한 필드 인덱스 정책을 생성하거나 업데이트합니다.
2	PutAccountPolicy	계정의 모든 로그 그룹 또는 로그 그룹의 하위 집합에 적용되는 계정 수준 데이터 보호 정책, 구독 필터 정책, 필드 인덱스 정책, 변환기 정책 또는 지표 추출 정책을 생성합니다.
3	DeleteIndexPolicy	단일 로그 그룹에 적용된 로그 그룹 수준 필드 인덱스 정책을 삭제합니다.
4	DeleteAccountPolicy	CloudWatch Logs 계정 정책을 삭제합니다.

CloudWatch Logs Insights에서 주변 로그 보기

CloudWatch Logs Insights 쿼리에서 반환된 특정 로그 레코드 전후에 발생한 로그 이벤트를 볼 수 있습니다. 로그를 둘러싸면 오류, 경고 또는 기타 중요한 이벤트와 관련된 컨텍스트를 이해하는 데 도움이 됩니다.

주변 로그의 작동 방식

CloudWatch Logs Insights 쿼리를 실행하면 결과의 각 로그 레코드에 주변 로그 옵션이 포함됩니다. 이 옵션을 선택하면 CloudWatch Logs Insights는 선택한 레코드 직전과 직후에 발생한 동일한 로그 스트림의 추가 로그 이벤트를 표시합니다. 이러한 이벤트는 원래 쿼리 필터와 일치하지 않더라도 나타납니다.

표시할 주변 로그 줄 수를 구성할 수 있습니다. 선택한 레코드의 위와 아래에 있는 5, 10, 20, 50 또는 100개의 로그 줄 중에서 선택합니다.

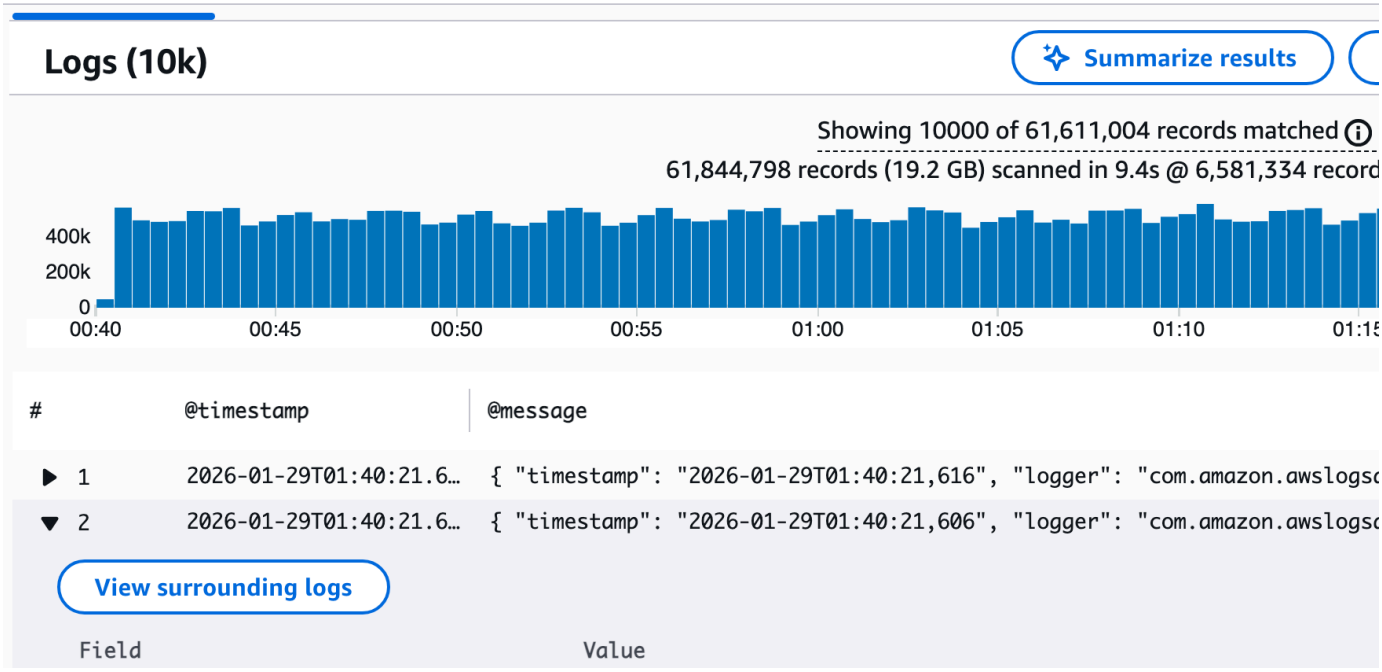
주변 로그는 다음 시나리오에서 유용합니다.

- 실패로 이어진 이벤트를 확인하여 오류 디버깅
- 분산 시스템에서 경고 또는 제한 시간 조사
- 애플리케이션 전반의 이벤트 시퀀스 이해
- 대용량 로그 환경의 문제 분석

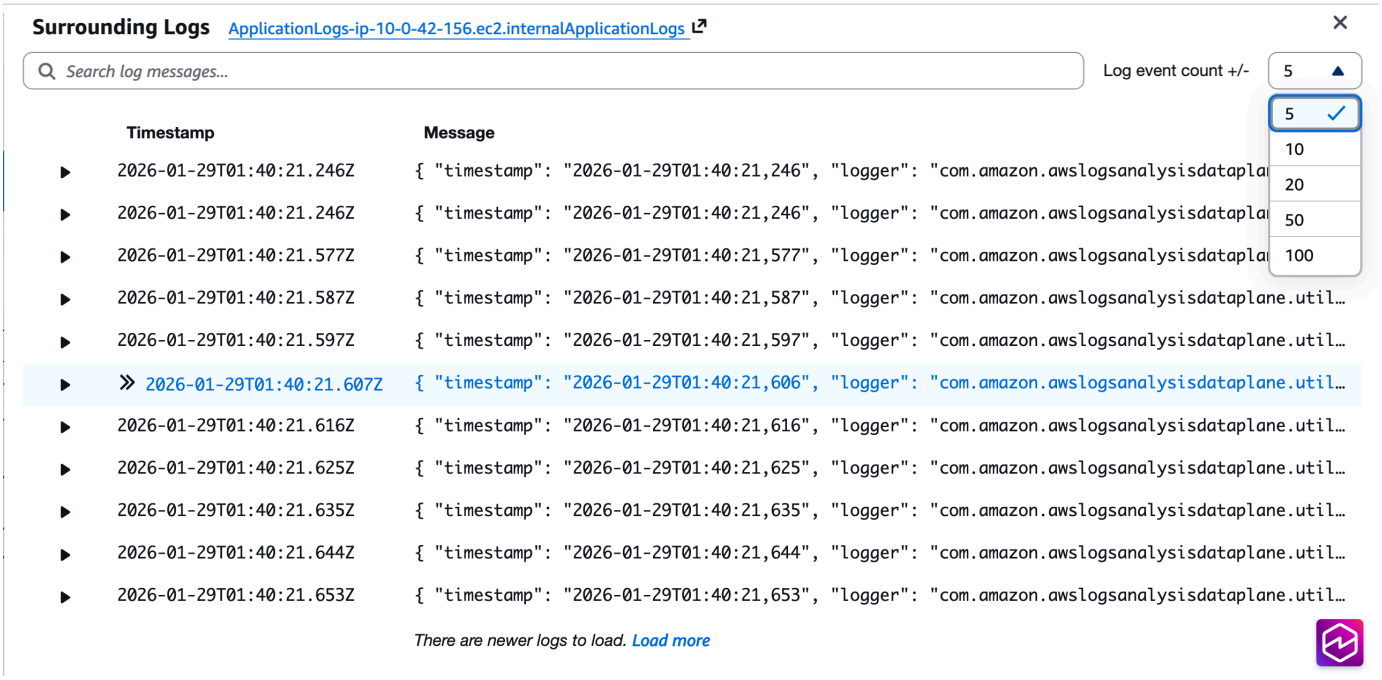
주변 로그 보기

주변 로그를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 인사이트를 선택합니다.
3. 쿼리를 실행합니다.
4. 쿼리 결과에서 조사하려는 로그 레코드를 찾습니다.
5. 로그 주변을 선택합니다.



6. 이벤트 수 드롭다운에서 선택한 레코드의 위와 아래에 표시할 로그 이벤트 수를 선택합니다. 5, 10, 20, 50 또는 100을 선택할 수 있습니다.



주변 로그 내에서 검색

주변 로그 패널을 연 후 특정 키워드를 검색하여 관련 컨텍스트를 찾을 수 있습니다.

주변 로그 내에서 검색하려면

1. 주변 로그 패널의 검색 상자에 검색어를 입력합니다.
2. 강조 표시된 일치하는 로그 줄을 검토합니다.
3. 탐색 컨트롤을 사용하여 매치 간에 이동합니다.



패턴 분석

로그를 쿼리할 때 CloudWatch Logs Insights는 기계 학습 알고리즘을 사용하여 패턴을 찾습니다. 패턴은 로그 필드 간에 반복되는 공유 텍스트 구조입니다. 쿼리 결과를 확인할 때 패턴 탭을 선택하면 결과 샘플을 기반으로 CloudWatch Logs가 발견한 패턴을 볼 수 있습니다. 또는 쿼리에 `pattern` 명령을 추가하여 일치하는 로그 이벤트의 전체 세트에서 패턴을 분석할 수 있습니다.

많은 수의 로그 이벤트를 몇 가지 패턴으로 압축할 수 있기 때문에 패턴은 큰 로그 집합을 분석하는 데 유용합니다.

다음 세 가지 로그 이벤트 샘플을 고려해 보세요.

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for resource id 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for resource id 324892398123-12345
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for resource id 3ff231242342-12345
```

이전 샘플에서 세 로그 이벤트는 모두 한 가지 패턴을 따릅니다.

```
<Time-1> [INFO] Calling DynamoDB to store for resource id <ID-2>
```

패턴 내의 필드를 토큰이라고 합니다. 요청 ID 또는 타임스탬프와 같이 패턴 내에서 다양한 필드는 동적 토큰입니다. 각 동적 토큰은 <*string*-*number*>로 표시됩니다. *string*은 토큰이 나타내는 데이터 유형에 대한 설명입니다. *number*는 다른 동적 토큰과 비교하여 패턴에서 이 토큰이 나타나는 위치를 보여줍니다.

동적 토큰의 일반적인 예로는 오류 코드, 타임스탬프, 요청 ID 등이 있습니다. 토큰 값은 동적 토큰의 특정 값을 나타냅니다. 예를 들어, 동적 토큰이 HTTP 오류 코드를 나타내는 경우 토큰 값은 `501`일 수 있습니다.

패턴 감지는 CloudWatch Logs 이상 탐지기 및 비교 기능에도 사용됩니다. 자세한 내용은 로그 이상 탐지 및 (diff)를 이전 시간 범위와 비교 섹션을 참조하세요.

패턴 분석 시작하기

패턴 감지는 모든 CloudWatch Logs Insights 쿼리에서 자동으로 수행됩니다. `pattern` 명령이 포함되지 않은 쿼리는 결과에 로그 이벤트와 패턴을 모두 가져옵니다.

쿼리에 `pattern` 명령을 포함하면 일치하는 전체 로그 이벤트 세트에서 패턴 분석이 수행됩니다. 이렇게 하면 더 정확한 패턴 결과를 얻을 수 있지만 `pattern` 명령을 사용하면 원시 로그 이벤트가 반환되지 않습니다. 쿼리에 `pattern`이 포함되지 않은 경우 패턴 결과는 반환된 첫 1,000개의 로그 이벤트 또는 쿼리에 사용한 한도 값을 기반으로 합니다. 쿼리에 `pattern`을 포함하면 패턴 탭에 표시되는 결과는 쿼리와 일치하는 모든 로그 이벤트에서 파생됩니다.

CloudWatch Logs Insights에서 패턴 분석을 시작하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, Logs Insights를 선택합니다.

   로그 인사이트 페이지의 쿼리 편집기에는 최신 로그 이벤트 20개를 반환하는 기본 쿼리가 포함되어 있습니다.
3. 쿼리가 다음과 같이 보이도록 쿼리 상자의 `| limit 20` 줄을 제거합니다.

   ```
   fields @timestamp, @message, @logStream, @log
   | sort @timestamp desc
   ```
4. 로그 그룹 선택 드롭다운에서 쿼리할 로그 그룹을 하나 이상 선택합니다.
5. (선택 사항) 시간 간격 선택기를 사용하여 쿼리할 기간을 선택합니다.

   5~30분 간격, 1시간, 3시간, 12시간 간격 또는 사용자 지정 시간 범위 중에서 선택할 수 있습니다.
6. 쿼리 실행을 선택하여 쿼리를 시작합니다.

쿼리 실행이 완료되면 로그 탭에 쿼리에서 반환된 로그 이벤트 테이블이 표시됩니다. 테이블 위에는 일치하는 레코드 71,101개 중 10,000개 표시와 비슷하게 쿼리에 일치한 레코드 수에 대한 메시지가 있습니다.

7. 패턴 탭을 선택합니다.
8. 이제 쿼리에서 발견된 패턴이 테이블에 표시됩니다. 쿼리에 `pattern` 명령이 포함되지 않았기 때문에 이 탭에는 로그탭의 테이블에 표시된 10,000개의 로그 이벤트에서 발견된 패턴만 표시됩니다.

   각 패턴마다 다음 정보가 표시됩니다.

   - 패턴 및 <*string*-*number*>로, 표시된 각 동적 토큰. *string*은 토큰이 나타내는 데이터 유형에 대한 설명입니다. *number*는 다른 동적 토큰과 비교하여 패턴에서 이 토큰이 나타나는 위치를 보여줍니다.
   - 쿼리된 로그 이벤트에 패턴이 나타난 횟수인 이벤트 개수. 패턴을 빈도별로 정렬하려면 이벤트 개수 열 제목을 선택합니다.
   - 이 패턴을 포함하는 쿼리된 로그 이벤트의 백분율인 이벤트 비율.
   - 다음 중 하나인 심각도 유형.
     - 패턴에 Error라는 단어가 포함된 경우 ERROR.
     - 패턴에 Warn이라는 단어는 포함되어 있지만 Error는 포함되어 있지 않은 경우 WARN.
     - 패턴에 Warn 또는 Error가 포함되어 있지 않은 경우INFO.

   심각도 기준으로 패턴을 정렬하려면 Severity info(심각도 정보) 열 제목을 선택합니다.
9. 이제 쿼리를 변경합니다. 전체 쿼리가 다음과 같아지도록 쿼리의 `| sort @timestamp desc` 줄을 `| pattern @message`로 바꿉니다.

```
fields @timestamp, @message, @logStream, @log
| pattern @message
```

10. 쿼리 실행을 선택합니다.

    쿼리가 완료되면 로그 탭에 결과가 없습니다. 그러나 패턴 탭에는 쿼리된 총 로그 이벤트의 수에 따라 더 많은 패턴의 수가 나열될 수 있습니다.
11. 쿼리에 `pattern`을 포함했는지 여부에 관계없이 쿼리가 반환하는 패턴을 추가로 검사할 수 있습니다. 그러려면 패턴 중 하나의 검사 열에서 아이콘을 선택합니다.

    패턴 검사 창이 나타나고 다음이 표시됩니다.

- 패턴. 패턴 내에서 토큰을 선택하여 해당 토큰의 값을 분석합니다.
- 쿼리된 시간 범위에서 패턴의 발생 횟수를 보여주는 히스토그램. 이는 패턴 발생의 갑작스러운 증가 등 흥미로운 추세를 식별하는 데 도움이 될 수 있습니다.
- 로그 샘플 탭에는 선택한 패턴과 일치하는 몇 가지 로그 이벤트가 표시됩니다.
- 토큰 값 탭에는 선택한 동적 토큰의 값이 표시됩니다(동적 토큰을 선택한 경우).

> **Note**
>
> 각 토큰마다 최대 10개의 토큰 값이 캡처됩니다. 토큰 개수가 정확하지 않을 수 있습니다. CloudWatch Logs는 확률 계수기를 사용하여 절대값이 아닌 토큰 개수를 생성합니다.

- 관련 패턴 탭에는 검사 중인 패턴과 거의 동일한 시간에 자주 발생한 다른 패턴이 표시됩니다. 예를 들어, ERROR 메시지의 패턴에 일반적으로 추가 세부 정보와 함께 INFO로 표시된 다른 로그 이벤트가 수반된 경우 해당 패턴이 여기에 표시됩니다.

패턴 명령에 대한 세부 정보

이 섹션에는 pattern 명령 및 명령 사용에 대한 추가 정보가 포함되어 있습니다.

- 이전 자습서에서는 pattern을 추가할 때 sort 명령을 제거했습니다. sort 명령 뒤에 pattern 명령을 포함하는 쿼리는 유효하지 않기 때문입니다. pattern이 sort 앞에 있어야 유효합니다.

  pattern 구문에 대한 자세한 내용은 pattern을 참조하세요.
- 쿼리에서 pattern을 사용하는 경우 @message는 pattern 명령에서 선택한 필드 중 하나여야 합니다.
- pattern 명령 앞에 filter 명령을 포함하여 필터링된 로그 이벤트 집합만 패턴 분석의 입력으로 사용되도록 할 수 있습니다.
- parse 명령에서 파생된 필드 등 특정 필드의 패턴 결과를 보려면 pattern @fieldname을 사용합니다.
- stats 명령이 있는 쿼리와 같이 로그가 아닌 출력이 있는 쿼리는 패턴 결과를 반환하지 않습니다.

CloudWatch Logs Insights 쿼리 저장 및 재실행

쿼리를 생성한 후 나중에 다시 실행할 수 있도록 저장할 수 있습니다. 쿼리는 폴더 구조로 저장되므로 체계적으로 관리할 수 있습니다. 계정별로 리전당 최대 1,000개의 쿼리를 저장할 수 있습니다.

쿼리는 사용자별 수준이 아닌 리전별 수준에 저장됩니다. 쿼리를 생성하고 저장하는 경우 동일한 리전의 CloudWatch Logs에 액세스할 수 있는 다른 사용자는 해당 리전의 저장된 모든 쿼리와 폴더 구조를 볼 수 있습니다.

쿼리를 저장하려면 `logs:PutQueryDefinition` 권한이 있는 역할에 로그인해야 합니다. 저장된 쿼리의 목록을 보려면 `logs:DescribeQueryDefinitions` 권한이 있는 역할에 로그인해야 합니다.

> **Note**
>
> 파라미터, 즉 명명된 자리 표시자가 있는 재사용 가능한 템플릿을 사용하여 쿼리를 생성하고 저장할 수 있습니다. 동일한 쿼리의 여러 변형을 서로 다른 값으로 저장하는 대신 템플릿을 하나 생성하고 실행할 때 서로 다른 파라미터 값을 제공합니다. 이 기능은 현재 Logs Insights 쿼리 언어를 사용하는 쿼리에만 지원됩니다. 자세한 내용은 파라미터와 함께 저장된 쿼리 사용을 참조하세요.

Console

쿼리를 저장하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. 쿼리 편집기에서 쿼리를 생성합니다.
4. 저장을 선택합니다.
5. 쿼리의 이름을 입력합니다.
6. (선택 사항) 쿼리를 저장할 폴더를 선택합니다. 새로 생성을 선택하여 폴더를 만듭니다. 새 폴더를 만드는 경우 폴더 이름에 슬래시(/) 문자를 사용하여 폴더 구조를 정의할 수 있습니다. 예를 들어, 새 폴더의 이름을 **`folder-level-1/folder-level-2`**로 지정하면 **`folder-level-1`**이라는 최상위 폴더가 만들어지고 그 폴더 안에 **`folder-level-2`**라는 다른 폴더가 만들어집니다. 쿼리가 **`folder-level-2`**에 저장됩니다.
7. (선택 사항) 쿼리의 로그 그룹 또는 쿼리 텍스트를 변경합니다.
8. (선택 사항) 쿼리에서 파라미터를 사용하려면 다음 추가 단계를 따릅니다.

a. 쿼리에 파라미터를 추가합니다. {{parameter}} 구문(파라미터 이름 앞뒤에 두 개의 중괄호)을 사용하여 정적 값을 자리 표시자로 바꿉니다.

   예: 정적 값이 있는 원래 쿼리:

   ```
   fields @timestamp, @message
   | filter level = "Error"
   | filter applicationName = "OrderService"
   ```

   파라미터로 쿼리 업데이트:

   ```
   fields @timestamp, @message
   | filter level = {{logLevel}}
   | filter applicationName = {{applicationName}}
   ```

b. 쿼리에 사용되는 파라미터를 정의합니다. 각 자리 표시자 파라미터에 대해 다음을 지정합니다.
   - 이름: 자리 표시자 이름과 정확히 일치해야 합니다(예: , logLevelapplicationName).
   - 기본값(선택 사항): 파라미터 값이 제공되지 않은 경우 사용할 값입니다.
   - 설명(선택 사항): 파라미터의 용도를 설명합니다.

c. 파라미터가 있는 쿼리는 $ 접두사가 있는 쿼리 이름을 사용하고 파라미터 이름을 키-값 페어로 전달하여 실행할 수 있습니다. 자세한 내용은 저장된 쿼리 실행을 참조하세요.

9. 저장을 선택합니다.

AWS CLI

쿼리를 저장하려면를 사용합니다put-query-definition.

```
aws logs put-query-definition \
  --name "ErrorsByLevel" \
  --query-string "fields @timestamp, @message | filter level = \"ERROR\"" \
  --log-group-names "/aws/lambda/my-function" \
  --region us-east-1
```

(선택 사항) 파라미터와 함께 쿼리를 저장하려면 --parameters 옵션을 추가하고 쿼리 문자열에 {{parameterName}} 자리 표시자를 사용합니다.

```
aws logs put-query-definition \
```

```
  --name "ErrorsByLevel" \
  --query-string "fields @timestamp, @message | filter level = {{logLevel}} | filter
 applicationName = {{applicationName}}" \
  --parameters '[{"name":"logLevel","defaultValue":"ERROR","description":"Log level
 to filter"},
{"name":"applicationName","defaultValue":"OrderService","description":"Application
 name to filter"}]' \
  --log-group-names "/aws/lambda/my-function" \
  --region us-east-1
```

폴더에 쿼리를 저장하려면 쿼리 이름 앞에 폴더 경로를 붙입니다.

```
aws logs put-query-definition \
  --name "my-folder/ErrorsByLevel" \
  --query-string "fields @timestamp, @message | filter level = {{logLevel}}" \
  --parameters '[{"name":"logLevel","defaultValue":"ERROR","description":"Log level
 to filter"}]' \
  --log-group-names "/aws/lambda/my-function" \
  --region us-east-1
```

API

쿼리를 저장하려면 PutQueryDefinition을 호출합니다.

```
{
  "name": "ErrorsByLevel",
  "queryString": "fields @timestamp, @message | filter level = \"ERROR\"",
  "logGroupNames": ["/aws/lambda/my-function"]
}
```

(선택 사항) 파라미터와 함께 쿼리를 저장하려면 `parameters` 필드를 포함하고 쿼리 문자열에 `{{parameterName}}` 자리 표시자를 사용합니다.

```
{
  "name": "ErrorsByLevel",
  "queryString": "fields @timestamp, @message | filter level = {{logLevel}} | filter
 applicationName = {{applicationName}}",
  "logGroupNames": ["/aws/lambda/my-function"],
  "parameters": [
    {
      "name": "logLevel",
```

```
      "defaultValue": "ERROR",
      "description": "Log level to filter"
    },
    {
      "name": "applicationName",
      "defaultValue": "OrderService",
      "description": "Application name to filter"
    }
  ]
}
```

> **Tip**
>
> PutQueryDefinition을 사용하여 저장된 쿼리에 대한 폴더를 만들 수 있습니다. 저장된 쿼리에 대한 폴더를 만들려면 슬래시(/)를 사용하여 원하는 쿼리 이름에 원하는 폴더 이름을 접두사로 붙입니다. <*folder-name*>/<*query-name*>. 이 작업에 대한 자세한 내용은 PutQueryDefinition을 참조하세요.

Console

저장된 쿼리를 실행하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. 오른쪽에서 쿼리를 선택합니다.
4. 저장된 쿼리 목록에서 쿼리를 선택합니다. 쿼리 텍스트가 쿼리 편집기에 나타납니다.
5. (선택 사항) 파라미터와 함께 쿼리를 사용하려면:
   a. 저장된 쿼리 사이드 패널에서 쿼리 이름 옆에 있는 + 아이콘을 선택합니다.
   b. 파라미터가 있는 쿼리는 쿼리 편집기에 나타납니다. 예를 들어 옆에 있는 + 아이콘을 선택하면 ErrorsByLevel쿼리 편집기가 다음과 같이 채워집니다. `$ErrorsByLevel(level=, applicationName=)`
   c. 파라미터(level, applicationName) 값을 입력하고 쿼리를 실행합니다. 예: `$ErrorsByLevel(level= "ERROR", applicationName= "OrderService")`.
6. 실행을 선택합니다.

AWS CLI

파라미터로 저장된 쿼리를 실행하려면

구문과 `start-query` 함께 `$QueryName()`를 사용합니다.

```
aws logs start-query \
  --log-group-names "/aws/lambda/my-function" \
  --start-time 1707566400 --end-time 1707570000 \
  --query-string '$ErrorsByLevel(level= "ERROR", applicationName= "OrderService")' \
  --region us-east-1
```

API

파라미터로 저장된 쿼리를 실행하려면

`queryString` 필드에 `$QueryName()` 구문을 사용하여 StartQuery를 호출합니다.

```
{
  "logGroupNames": ["/aws/lambda/my-function"],
  "startTime": 1707566400,
  "endTime": 1707570000,
  "queryString": "$ErrorsByLevel(level=\"ERROR\", applicationName= \"OrderService
\")"
}
```

저장된 쿼리의 새 버전을 저장하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. 오른쪽에서 쿼리를 선택합니다.
4. 저장된 쿼리 목록에서 쿼리를 선택합니다. 이 쿼리가 쿼리 편집기에 나타납니다.
5. 쿼리를 수정합니다. 작업을 확인하기 위해 쿼리를 실행해야 하는 경우 쿼리 실행을 선택합니다.
6. 새 버전을 저장할 준비가 되면 작업, 다른 이름으로 저장을 선택합니다.
7. 쿼리의 이름을 입력합니다.
8. (선택 사항) 쿼리를 저장할 폴더를 선택합니다. 새로 생성을 선택하여 폴더를 만듭니다. 새 폴더를 만드는 경우 폴더 이름에 슬래시(/) 문자를 사용하여 폴더 구조를 정의할 수 있습니다. 예를 들어, 새 폴더의 이름을 **`folder-level-1/folder-level-2`**로 지정하면 **`folder-level-1`**이라는

최상위 폴더가 만들어지고 그 폴더 안에 **folder-level-2**라는 다른 폴더가 만들어집니다. 쿼리가 **folder-level-2**에 저장됩니다.

9. (선택 사항) 쿼리의 로그 그룹 또는 쿼리 텍스트를 변경합니다.
10. 저장을 선택합니다.

쿼리를 삭제하려면 `logs:DeleteQueryDefinition` 권한이 있는 역할에 로그인해야 합니다.

저장된 쿼리를 편집 또는 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. 오른쪽에서 쿼리를 선택합니다.
4. 저장된 쿼리 목록에서 쿼리를 선택합니다. 이 쿼리가 쿼리 편집기에 나타납니다.
5. 작업, 편집 또는 작업, 삭제를 선택합니다.

파라미터와 함께 저장된 쿼리 사용

파라미터가 있는 저장된 쿼리는 명명된 자리 표시자가 있는 재사용 가능한 쿼리 템플릿입니다. 거의 동일한 쿼리의 여러 복사본을 유지하는 대신 쿼리를 실행할 때 템플릿을 저장하고 다른 파라미터 값을 제공할 수 있습니다. 파라미터는 CloudWatch Logs Insights 쿼리 언어에서만 지원됩니다.

작동 방식

쿼리를 저장할 때 자리 표시자는 쿼리 실행 시 제공할 수 있는 값을 식별합니다. 자리 표시자는 `{{parameterName}}` 구문을 사용합니다. 다음은 `logLevel` 파라미터 2개와를 `ErrorsByLevel` 사용하여 라는 저장된 쿼리의 예입니다`applicationName`.

```
fields @timestamp, @message
| filter level = {{logLevel}}
| filter applicationName = {{applicationName}}
```

저장된 쿼리를 실행하려면 접두사가 붙은 쿼리 이름을 사용하고 파라미터 값을 $ 전달하여 호출할 수 있습니다. CloudWatch Logs Insights 쿼리 엔진은 각 자리 표시자를 대체합니다. 파라미터에 기본값이 포함된 경우 다른 값이 제공되지 않으면 해당 값이 사용됩니다.

```
# Run query by using query name and passing parameter values explicitly
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService")
```

```
# Run query without specifying parameter values - default values are used in this case.
$ErrorsByLevel()
```

공백 또는 특수 문자가 포함된 저장된 쿼리 이름은 백틱으로 묶어야 합니다.

```
$`Errors By Level`(logLevel = "WARN")
```

파라미터를 사용하여 저장된 쿼리 샘플

파라미터로 결과 제한 추가

쿼리 이름: 파라미터 ErrorsByLevel 포함logLevel(기본값: "ERROR"), applicationName (기본값: "OrderService") 및 maxResults (기본값: 50)

```
fields @timestamp, @message, @logStream
| filter level = {{logLevel}}
| filter applicationName = {{applicationName}}
| sort @timestamp desc
| limit {{maxResults}}
```

```
# Run the query using the query name and passing parameter values
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService", maxResults = 100)
```

파라미터와 함께 여러 저장된 쿼리 사용

아래 예제에서는 ErrorsByLevel 및 sort @timestamp desc | limit {{count}} (파라미터 count, 기본값)로 RecentN 정의된 두 번째 저장된 쿼리를 사용합니다20. CloudWatch Logs Insights 쿼리 엔진은 실행하기 전에 각 쿼리를 확장합니다.

```
# Using multiple queries with parameters in sequence
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService")
| $RecentN(count = 10)

# Each of the queries is expanded, resulting in the following query when it is run.
fields @timestamp, @message
| filter level = "WARN"
| filter applicationName = "OrderService"
| sort @timestamp desc
| limit 10
```

할당량 및 오류 처리

> **Note**
>
> 저장된 각 쿼리는 최대 20개의 파라미터를 가질 수 있습니다.

확장된 쿼리 문자열은 10,000자를 초과할 수 없습니다. 파라미터 이름은 문자 또는 밑줄로 시작해야 합니다. 저장된 쿼리는 다른 저장된 쿼리를 참조할 수 없습니다(중첩된 호출은 지원되지 않음).

일반적인 오류

오류	원인
파라미터는 CWLI 쿼리 언어에만 지원됩니다.	파라미터는 CloudWatch Logs Insights 쿼리 언어에서만 지원됩니다.
queryString에서 필수 파라미터를 찾을 수 없음	의 파라미터 이름에 쿼리 문자열`{{placeholder}}` 에 일치하는이 `--parameters` 없습니다.
파라미터 수가 최대 20개를 초과합니다.	저장된 쿼리는 현재 20개의 파라미터만 지원합니다.
중복 파라미터 이름	쿼리 정의에에 중복 파라미터가 있습니다`parameters` .

> **Note**
>
> 파라미터로 저장된 쿼리를 생성하거나 업데이트하려면 `logs:PutQueryDefinition` 권한이 필요합니다. 하나를 실행하려면 `logs:StartQuery` 및가 필요합니다`logs:DescribeQueryDefinitions`.

대시보드에 쿼리 추가 또는 쿼리 결과 내보내기

쿼리를 실행한 후 CloudWatch 대시보드에 쿼리를 추가하거나 클립보드에 쿼리 결과를 복사할 수 있습니다.

대시보드에 추가한 쿼리는 대시보드를 로드할 때마다 그리고 대시보드를 새로 고칠 때마다 실행됩니다. 이러한 쿼리는 동시 CloudWatch Logs Insights 쿼리 100개 한도에 포함됩니다.

대시보드에 쿼리 결과를 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. 로그 그룹을 하나 이상 선택하고 쿼리를 실행합니다.
4. 대시보드에 추가를 선택합니다.
5. 대시보드를 선택하거나 새로 생성을 선택하여 쿼리 결과에 대한 새 대시보드를 생성합니다.
6. 쿼리 결과에 사용할 위젯 유형을 선택합니다.
7. 위젯의 이름을 입력합니다.
8. 대시보드에 추가를 선택합니다.

클립보드로 쿼리 결과를 복사하거나 쿼리 결과를 다운로드하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. 로그 그룹을 하나 이상 선택하고 쿼리를 실행합니다.
4. 결과 내보내기를 선택한 다음 원하는 옵션을 선택합니다.

실행 중인 쿼리 또는 쿼리 기록 보기

현재 진행 중인 쿼리와 최신 쿼리 기록을 볼 수 있습니다.

현재 실행 중인 쿼리에는 대시보드에 추가한 쿼리가 포함됩니다. 대시보드에 추가된 쿼리를 포함하여 계정당 동시 CloudWatch Logs Insights 쿼리는 100개로 제한됩니다. 또한 OpenSearch Service PPL 또는 OpenSearch Service SQL에 대해 15개의 동시 쿼리를 실행할 수 있습니다.

최근 쿼리 기록을 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음, 로그 인사이트를 선택합니다.
3. CloudWatch Logs 콘솔에 대해 새 디자인을 사용하는 경우 기록을 선택합니다. 이전 디자인을 사용하는 경우 작업, 이 계정에 대한 쿼리 기록 보기를 선택합니다.

   최근 쿼리 목록이 나타납니다. 쿼리를 선택하고 실행을 선택하여 이 중 하나를 다시 실행할 수 있습니다.

상태 아래의 CloudWatch Logs에서 현재 실행 중인 쿼리가 진행 중으로 표시됩니다.

를 사용하여 쿼리 결과 암호화 AWS Key Management Service

기본적으로 CloudWatch Logs는 기본 CloudWatch Logs 서버 측 암호화 방법을 사용하여 CloudWatch Logs 인사이트 쿼리의 저장된 결과를 암호화합니다. 대신 AWS KMS 키를 사용하여 이러한 결과를 암호화하도록 선택할 수 있습니다. AWS KMS 키를 암호화 결과와 연결하면 CloudWatch Logs는 해당 키를 사용하여 계정에 있는 모든 쿼리의 저장된 결과를 암호화합니다.

나중에 쿼리 결과에서 키 연결을 해제하면 CloudWatch Logs는 이후 쿼리에서 기본 암호화 방법을 다시 사용합니다. 그러나 키가 연결된 동안 실행된 쿼리는 여전히 해당 키로 암호화됩니다. CloudWatch Logs는 계속해서 키를 참조할 수 있으므로 CloudWatch Logs는 KMS 키의 연결이 해제된 후에도 여전히 해당 결과를 반환할 수 있습니다. 그러나 나중에 키를 비활성화하면 CloudWatch Logs는 해당 키로 암호화된 쿼리 결과를 읽을 수 없습니다.

> **Important**
>
> CloudWatch Logs는 대칭 KMS 키만 지원합니다. 비대칭 키를 사용하여 쿼리 결과를 암호화하지 마세요. 자세한 내용은 대칭 및 비대칭 키 사용을 참조하세요.

제한 사항

- 다음 단계를 수행하려면 `kms:CreateKey`, `kms:GetKeyPolicy` 및 `kms:PutKeyPolicy` 권한이 있어야 합니다.
- 쿼리 결과에서 키를 연결하거나 연결 해제하고 난 후 이러한 변경이 적용되기까지 최대 5분의 시간이 소요될 수 있습니다.
- 연결된 키에 대한 CloudWatch Logs 액세스 권한을 취소하거나 연결된 KMS 키를 삭제한 경우에는 CloudWatch Logs에서 암호화된 데이터를 더 이상 검색할 수 없습니다.
- CloudWatch 콘솔을 사용하여 키를 연결할 수는 없으며, AWS CLI 또는 CloudWatch Logs API를 사용해야 합니다.

1단계: 생성 AWS KMS key

다음 create-key 명령을 사용하여 KMS 키 생성:

```
aws kms create-key
```

이 명령의 출력 화면에는 키의 키 ID와 Amazon Resource Name(ARN)이 포함됩니다. 다음은 예제 출력입니다.

```
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
        "AWSAccountId": "123456789012",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

2단계: KMS 키에 대한 권한 설정

기본적으로 모든 KMS 키는 비공개입니다. 리소스 소유자만 이를 사용하여 데이터를 암호화 및 해독할 수 있습니다. 그러나 리소스 소유자가 원한다면 다른 사용자 및 리소스에 키에 대한 액세스 권한을 부여할 수 있습니다. 이 단계에서는 키를 사용할 보안 주체 권한을 CloudWatch Logs 서비스에 제공합니다. 이 서비스 보안 주체는 키가 저장된 리전과 동일한 AWS 리전에 있어야 합니다.

가장 좋은 방법은 키 사용을 지정한 AWS 계정으로만 제한하는 것입니다.

먼저 다음 get-key-policy 명령을 사용하는 `policy.json`로 KMS 키에 대한 기본 정책을 저장합니다.

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./
policy.json
```

텍스트 편집기에서 `policy.json` 파일을 열고 다음 명령문 중 하나에서 굵은 글꼴로 표시된 섹션을 추가합니다. 기존 명령문과 새 명령문을 쉼표로 구분합니다. 이러한 문은 `Condition` 섹션을 사용하

여 AWS KMS 키의 보안을 강화합니다. 자세한 내용은 AWS KMS 키 및 암호화 컨텍스트 섹션을 참조하세요.

이 예제의 Condition 섹션에서는 AWS KMS 키 사용을 지정된 계정의 CloudWatch Logs Insights 쿼리 결과로 제한합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.region.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt*",
                "kms:Decrypt*",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                "aws:SourceArn": "arn:aws:logs:us-east-1:111122223333:query-
result:*"
                },
                "StringEquals": {
                "aws:SourceAccount": "111122223333"
                }
            }
```

```
        }
    ]
}
```

마지막으로 아래 put-key-policy 명령을 사용하여 업데이트된 정책을 추가합니다.

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

3단계: KMS 키를 쿼리 결과와 연결

KMS 키를 계정의 쿼리 결과와 연결하려면

다음과 같이 disassociate-kms-key 명령을 사용합니다.

```
aws logs associate-kms-key --resource-identifier "arn:aws:logs:region:account-id:query-
result:*" --kms-key-id "key-arn"
```

4단계: 계정의 쿼리 결과에서 키 연결 해제

쿼리 결과에 연결된 KMS 키의 연결을 해제하려면 아래 disassociate-kms-key 명령을 사용합니다.

```
aws logs disassociate-kms-key --resource-identifier "arn:aws:logs:region:account-
id:query-result:*"
```

CloudWatch Logs Insights 쿼리 결과에서 자연어 요약 생성

로그 데이터를 분석하는 것은 애플리케이션의 동작을 이해하는 데 중요하지만 대량의 로그 항목을 해석하는 데는 시간이 많이 걸릴 수 있습니다. CloudWatch Logs Insights는 이제 복잡한 쿼리 결과를 명확하고 간결한 요약으로 변환하는 자연어 요약 기능을 제공합니다. 이 기능을 통해 문제를 빠르게 식별하고 로그 데이터에서 실행 가능한 인사이트를 얻을 수 있습니다.

작동 방식

CloudWatch Logs Insights는 Amazon Bedrock을 사용하여 쿼리 결과에서 사람이 읽을 수 있는 요약을 생성할 수 있습니다. 이 기능은 모든 CloudWatch Logs Insights 쿼리 언어를 지원하며, 로그 데이터에서 명확하고 실행 가능한 인사이트를 제공합니다.

리전 가용성 및 데이터 처리

> **Important**
>
> 이 기능을 사용하면 쿼리 결과가 다른 AWS 리전에서 처리될 수 있습니다. 예를 들어, 미국 동부(버지니아 북부)에서 쿼리를 실행하는 경우 미국 서부(오리건)에서 요약이 발생할 수 있습니다.

다음 표에는 쿼리 결과 기능을 사용할 수 있는 다양한 지역에 AWS 리전 대해 가능한 처리가 나열되어 있습니다.

지원되는 CloudWatch Logs 지리	가능한 처리 리전
미국(US)	미국 동부(버지니아 북부) 리전 미국 동부(오하이오) 리전 미국 서부(오리건) 리전
유럽	유럽(프랑크푸르트) 리전 유럽(아일랜드) 리전 유럽(파리) 리전 유럽(스톡홀름) 리전 유럽(런던) 리전
아시아 태평양	미국 동부(버지니아 북부) 리전 미국 동부(오하이오) 리전 미국 서부(오리건) 리전
남미	미국 동부(버지니아 북부) 리전 미국 동부(오하이오) 리전 미국 서부(오리건) 리전

시작하기

자연어 요약을 생성하려면

1. CloudWatch Logs Insights 쿼리를 실행합니다.
2. 쿼리가 완료되면 결과 요약을 선택합니다.

권한

다음 중 하나가 있어야 합니다.

- `CloudWatchLogsFullAccess` 권한
- `CloudWatchLogsReadOnlyAccess` 권한
- `cloudwatch:GenerateQueryResultsSummary`, `logs:GetQueryResults` `logs:DescribeQueries` 및 `logs:FilterLogEvents` 작업을 포함한 사용자 지정 IAM 정책

데이터 개인정보보호

쿼리 결과는 안전하게 처리되며 CloudWatch Logs Insights 또는 Amazon Bedrock을 훈련하거나 개선하는 데 사용되지 않습니다. 피드백 버튼을 사용하여 쿼리 결과 요약에 대한 피드백을 제공하기로 선택한 경우 피드백은 CloudWatch Logs Insights에 제공된 기능에 대한 만족도를 나타냅니다.

예약된 쿼리를 사용하여 로그 분석 자동화

예약된 쿼리를 사용하면 정기적으로 로그 쿼리 실행을 자동화할 수 있습니다. 로그 데이터를 분석하기 위해 쿼리를 수동으로 실행하는 대신 자동으로 실행되고 Amazon S3 버킷, Amazon EventBridge 이벤트 버스 또는 조회 테이블과 같은 대상에 결과를 전달하도록 예약된 쿼리를 구성할 수 있습니다. 이 자동화는 정기 보고서를 생성하거나, 추세를 모니터링하거나, 조회 테이블을 최신 로그 데이터로 최신 상태로 유지하거나, 로그 분석 결과를 기반으로 다운스트림 프로세스를 트리거하는 데 적합합니다.

예약된 쿼리는 로그 쿼리에 사용할 수 있는 세 가지 쿼리 언어를 모두 지원합니다.

- Logs Insights 쿼리 언어(Logs Insights QL)
- OpenSearch Service Piped Processing Language(PPL)
- OpenSearch Service Structured Query Language(SQL)

내용

- 예약된 쿼리 개념 이해
- 일정 표현식 참조
- 모범 사례
- 예약된 쿼리 시작하기
- 예약된 쿼리에 대한 S3 대상 구성
- 예약된 쿼리에 대한 조회 테이블 대상 구성
- 예약된 쿼리 문제 해결

예약된 쿼리 개념 이해

예약된 쿼리를 생성하기 전에 쿼리 실행 방식과 결과가 전달되는 위치에 영향을 미치는 이러한 주요 개념을 이해합니다.

IAM 역할 분리

예약된 쿼리에는 두 개의 개별 IAM 역할이 필요합니다. 하나는 쿼리를 실행하기 위한 역할이고 다른 하나는 Amazon S3 버킷, Amazon EventBridge 이벤트 버스 또는 조회 테이블과 같은 대상에 결과를 전달하기 위한 역할입니다. 이러한 분리가 존재하는 이유를 이해하면 권한을 올바르게 구성하고 해당 권한이 제공하는 보안 및 운영 이점을 사용하는 데 도움이 됩니다.

2역할 아키텍처는 데이터 액세스와 데이터 전송 간의 책임을 나눕니다. 쿼리 실행 역할은 로그 데이터에 액세스하고 쿼리를 실행하는 반면 대상 전송 역할은 선택한 대상에 결과를 기록합니다. 이러한 분리는 최소 권한 원칙을 따릅니다. 각 역할에는 특정 함수에 필요한 권한만 있습니다.

쿼리 실행 역할

CloudWatch Logs가 사용자를 대신하여 CloudWatch Logs Insights 쿼리를 실행하도록 허용합니다. 이 역할은 로그 그룹에 액세스하고 쿼리를 실행할 수 있는 권한이 필요하지만 대상 리소스에 액세스할 필요는 없습니다. 필요한 권한:

- `logs:StartQuery`
- `logs:StopQuery`
- `logs:GetQueryResults`
- `logs:DescribeLogGroups`
- `logs:Unmask` 마스킹 해제 데이터가 필요한 경우

KMS로 암호화된 로그 그룹의 경우: 로그 그룹을 암호화하는 데 사용되는 KMS 키에 대한 `kms:Decrypt` 및 `kms:DescribeKey` 권한. 이러한 권한도 추가해야 합니다.

신뢰 관계 요구 사항: 쿼리 실행 역할에는 CloudWatch Logs 서비스(`logs.amazonaws.com`)가 역할을 수임하도록 허용하는 신뢰 정책이 포함되어야 합니다. 이 신뢰 관계가 없으면 예약된 쿼리가 권한 오류와 함께 실패합니다.

쿼리 실행 역할에 대한 신뢰 정책 예제:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

쿼리 실행 역할에 대한 권한 정책 예제:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:GetQueryResults",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

대상 전송 역할

CloudWatch Logs가 선택한 대상으로 쿼리 결과를 전송할 수 있도록 허용합니다. 이 역할에는 최소 권한 원칙에 따라 특정 대상 서비스에 대한 권한만 필요합니다. 필요한 권한은 대상 유형에 따라 다릅니다.

신뢰 관계 요구 사항: 대상 전송 역할에는 CloudWatch Logs 서비스(`logs.amazonaws.com`)가 역할을 수임하도록 허용하는 신뢰 정책도 포함되어야 합니다.

S3 대상 전송 역할에 대한 권한 정책 예제:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::your-scheduled-query-results-bucket/*"
        }
    ]
}
```

조회 테이블 대상 전송 역할에 대한 권한 정책의 예:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLookupTable",
                "logs:UpdateLookupTable",
                "logs:GetQueryResults"
            ],
            "Resource": "*"
        }
    ]
}
```

이 분리는 운영에 실질적인 이점을 제공합니다. 보안 관점에서 결과가 전달되는 위치를 변경해야 하는 경우 쿼리 실행 권한을 변경하지 않고 대상 전달 역할만 수정합니다. 규정 준수 및 감사를 위해 민감한 로그 데이터에 액세스하는 역할과 외부 시스템에 쓰는 역할을 명확하게 추적할 수 있습니다. 이렇게 하면 로그 분석 인프라가 보안 모범 사례를 따르고 있음을 더 쉽게 입증할 수 있습니다.

교차 리전 및 교차 계정 사용

예약된 쿼리는 특정 리전에서 생성되며 해당 리전에서 실행됩니다. 그러나 로그 그룹을 쿼리하고 리전 및 계정 간에 결과를 전달할 수 있습니다. 하나 이상의 AWS 계정을 모니터링 계정으로 설정하고 여러 소스 계정과 연결해야 합니다. 모니터링 계정은 소스 AWS 계정에서 생성된 관찰성 데이터를 보고 상호 작용할 수 있는 중앙 계정입니다. 소스 계정은 해당 계정에 있는 리소스에 대한 관찰성 데이터를 생성하는 개별 AWS 계정입니다. 소스 계정은 관측성 데이터를 모니터링 계정과 공유합니다. 따라서 연결된 모든 계정의 로그 그룹을 사용하여 모니터링 계정에서 예약된 쿼리를 설정할 수 있습니다.

교차 리전 로그 그룹 쿼리

예약된 쿼리는 모든 리전의 로그 그룹에 액세스할 수 있습니다. 전체 ARN 형식을 사용하여 로그 그룹을 지정합니다`arn:aws:logs:region:account-id:log-group:log-group-name`. 쿼리 실행 역할에는 모든 대상 리전의 로그 그룹에 대한 `logs:StartQuery` 및 `logs:GetQueryResults` 권한이 필요합니다.

> **Important**
>
> 로그 그룹을 쿼리하거나 리전 간에 결과를 전송할 때 로그 데이터는 리전 경계를 넘어갑니다. 다음을 고려하세요.

- 데이터 레지던시 요구 사항 - 리전 간 데이터 전송이 조직의 데이터 거버넌스 정책 및 규제 요구 사항을 준수하는지 확인합니다.
- 데이터 전송 비용 - 리전 간 데이터 전송에는 추가 요금이 발생합니다.
- 네트워크 지연 시간 - 먼 리전의 로그 그룹에 액세스하는 쿼리는 지연 시간이 더 길어질 수 있습니다.

최적의 성능과 비용 효율성을 위해 기본 로그 그룹과 동일한 리전에서 예약된 쿼리를 생성합니다.

대체 접근 방식: CloudWatch Logs 중앙 집중화를 사용하여 여러 계정 및 리전의 로그 데이터를 중앙 모니터링 계정으로 복제합니다. 이렇게 하면 단일 리전에서 모든 중앙 집중식 로그에 액세스하는 예약된 쿼리를 생성하여 리전 간 쿼리를 방지하고 IAM 권한 관리를 간소화할 수 있습니다.

일정 표현식 및 시간대 처리

정의한 일정에 따라 쿼리 실행 시기와 실행 빈도가 결정됩니다. 올바른 일정 표현식을 선택하면 결과를 받는 시기와 쿼리하는 데이터의 양에 영향을 줍니다. 표현식 유형을 이해하면 단순성과 정밀도 중에서 선택할 수 있습니다.

Cron 표현식을 사용하면 타이밍을 정확하게 제어할 수 있으므로 정확한 시간, 요일 또는 요일을 지정할 수 있습니다. 특정 업무 시간에 실행하거나 운영 일정에 맞게 쿼리를 실행해야 하는 경우 cron 표현식을 사용합니다. 콘솔에서 간편한 일정 옵션을 사용하여 쿼리를 예약할 수도 있습니다.

cron 표현식

특정 시간에 쿼리를 실행합니다. 형식: `cron(minute hour day-of-month month day-of-week year)` 예시:

- `cron(0 9 * * ? *)` - 매일 오전 9시 UTC
- `cron(0 18 ? * MON-FRI *)` - 평일 오후 6시 UTC
- `cron(0 0 1 * ? *)` - 매월 1일 자정 UTC
- `cron(0 12 ? * SUN *)` - 매주 일요일 정오 UTC
- `cron(30 8 1 1 ? *)` - 1월 1일 오전 8시 30분 UTC

모든 예약된 쿼리는 현지 시간대 또는 AWS 리소스의 위치에 관계없이 UTC에서 실행됩니다. 이는 업무 시간 또는 시간에 민감한 분석을 위한 쿼리를 예약할 때 특히 중요합니다. 예를 들어 미국 동부 시

간으로 비즈니스를 운영하고 오전 9시에 일일 보고서를 원하는 경우 UTC 오프셋(일광 절약 시간 중 14:00 UTC, 그렇지 않으면 13:00 UTC)을 고려해야 합니다. 쿼리가 의도한 시간에 실행되도록 UTC를 염두에 두고 일정 표현식을 계획합니다.

쿼리 언어 선택

예약된 쿼리는 세 가지 쿼리 언어를 지원하며, 선택 사항은 쿼리를 작성하는 방법과 팀이 쿼리를 쉽게 유지 관리할 수 있는 방법에 영향을 줍니다. 올바른 언어는 분석 요구 사항과 팀의 기존 기술에 따라 달라집니다.

주로 로그 데이터를 필터링하고 집계하는 경우 CloudWatch Logs Insights 쿼리 언어는 가장 간단한 구문을 제공합니다. 여러 단계를 통해 데이터를 재구성하거나 보강해야 하는 복잡한 데이터 변환의 경우 PPL의 파이프라인 접근 방식을 사용하면 로직을 더 쉽게 따를 수 있습니다. 데이터베이스 작업과 유사한 조인 또는 복잡한 집계를 수행해야 하는 경우 SQL은 데이터베이스 경험 팀이 빠르게 채택할 수 있는 익숙한 구문을 제공합니다.

CloudWatch Logs Insights 쿼리 언어(CWLI)

직관적인 구문을 사용하여 로그 분석을 위해 특별히 설계되었습니다. 다음 경우에 가장 적합합니다.

- 텍스트 기반 로그 분석 및 필터링
- 시계열 집계 및 통계
- 로그 분석을 처음 사용하는 팀

OpenSearch Service Piped Processing Language(PPL)

강력한 데이터 변환 기능을 갖춘 파이프라인 기반 쿼리 언어입니다. 다음 경우에 가장 적합합니다.

- 복잡한 데이터 변환 및 보강
- 다단계 데이터 처리 워크플로
- 파이프라인 기반 처리에 익숙한 팀

OpenSearch Service Structured Query Language(SQL)

익숙한 데이터베이스 스타일 쿼리를 위한 표준 SQL 구문입니다. 다음 경우에 가장 적합합니다.

- 복잡한 조인 및 집계
- 비즈니스 인텔리전스 및 보고
- 강력한 SQL 경험을 갖춘 팀

대상 선택 및 사용 사례

쿼리 결과를 전송하는 위치에 따라 쿼리 결과를 사용하여 수행할 수 있는 작업이 결정됩니다. 이 선택은 장기 분석 구축, 자동 응답 트리거 또는 둘 다에 관계없이 전체 다운스트림 워크플로를 구성합니다. 각 대상 유형의 강점을 이해하면 사용 사례에 적합한 아키텍처를 설계하는 데 도움이 됩니다.

Amazon S3 대상은 스토리지 및 배치 처리에 최적화되어 있습니다. 쿼리 결과를 몇 개월 또는 몇 년 동안 유지하거나, 시간 경과에 따른 추세를 분석하거나, 데이터를 분석 플랫폼에 제공해야 하는 경우 Amazon S3는 무제한 보존으로 비용 효율적인 스토리지를 제공합니다. EventBridge 대상은 실시간 자동화에 최적화되어 있습니다. 쿼리 결과가 알림 전송, 워크플로 시작 또는 시스템 업데이트와 같은 즉각적인 작업을 트리거해야 하는 경우 EventBridge는 애플리케이션이 즉시 응답할 수 있는 이벤트로 결과를 제공합니다. 기본적으로 모든 쿼리 완료 이벤트는 이벤트로 기본 이벤트 버스로 자동 전송되므로 다운스트림 처리 시스템, Lambda 함수 또는 기타 이벤트 기반 아키텍처와 통합할 수 있습니다. 쿼리가 성공적으로 실행된 경우에만 결과가 대상에 게시됩니다. 조회 테이블 대상은 참조 데이터를 최신 상태로 유지하도록 최적화되어 있습니다. 조회 테이블 대상은 예약된 각 실행에 대한 쿼리 결과로 지정된 조회 테이블을 자동으로 채우거나 새로 고치므로 다른 쿼리는 `lookup` 명령을 사용하여 최신 데이터를 참조할 수 있습니다.

Amazon S3 대상

장기 보존 및 배치 처리를 위해 쿼리 결과를 JSON 파일로 저장합니다. Amazon S3 대상은 다음 시나리오에 가장 적합합니다.

- 기록 분석 및 데이터 아카이빙
- 데이터 레이크 및 분석 플랫폼과의 통합
- 규정 준수 및 감사 요구 사항
- 대규모 결과 집합의 비용 효율적인 스토리지

EventBridge 대상

쿼리 결과를 실시간 처리 및 자동화를 위한 이벤트로 전송합니다. 이벤트`queryId`에서를 사용하여 쿼리 실행 후 30일 동안 사용 가능한 쿼리 결과를 검색합니다. EventBridge 대상은 다음 시나리오에 가장 적합합니다.

- 쿼리 결과에 대한 자동 응답 트리거
- 서버리스 워크플로 및 Lambda 함수와 통합
- 실시간 알림 및 알림 시스템
- 이벤트 기반 아키텍처 및 마이크로서비스

테이블 대상 조회

예약된 각 실행에 대한 쿼리 결과가 포함된 조회 테이블을 자동으로 생성하거나 새로 고칩니다. 각 새로 고침은 테이블 콘텐츠를 완전히 대체합니다. 조회 테이블 대상은 다음 시나리오에 가장 적합합니다.

- 로그 쿼리에서 `lookup` 명령에 대한 참조 데이터를 최신 상태로 유지
- 로그 데이터에서 파생된 허용 목록, 거부자 또는 엔터티 인벤토리 유지 관리
- 활성 사용자 또는 리소스 목록과 같은 최근 활동 요약으로 쿼리 강화

쿼리 결과 형식 및 구조

예약된 쿼리는 JSON 형식으로 결과를 전달하지만 각 대상 유형은 서로 다른 페이로드를 수신합니다. 조회 테이블 대상의 경우 쿼리 결과는 조회 테이블의 콘텐츠가 되며 각 실행이 해당 콘텐츠를 대체합니다. 자세한 내용은 예약된 쿼리에 대한 조회 테이블 대상 구성 단원을 참조하십시오.

Amazon S3 대상은 쿼리의 결과 행을 수신합니다. 각 객체에는 결과 집합의 각 행에 대해 하나의 항목이 있는 JSON 배열이 포함되며, 각 항목은 쿼리의 출력 필드 이름을 해당 값에 매핑합니다. 객체는 쿼리 메타데이터 또는 쿼리 통계를 포함하지 않으며 쿼리가 요청하더라도 `@ptr` 필드를 생략합니다. 해당 필드는 콘솔에서만 사용할 수 있기 때문입니다.

EventBridge 대상은 쿼리 통계를 포함한 쿼리 메타데이터를 수신하지만 결과 행은 수신하지 않습니다. 완료된 쿼리의 행을 검색하려면 이벤트`queryId`의 값으로 GetQueryResults를 호출합니다.

다음 예제는 예약된 쿼리가 완료될 때 CloudWatch Logs가 EventBridge에 게시하는 이벤트를 보여줍니다.

```
{
    "version": "0",
    "id": "be72061b-eca2-e068-a7e1-83e01d6fe807",
    "detail-type": "Scheduled Query Completed",
    "source": "aws.logs",
    "account": "123456789012",
    "time": "2025-11-18T11:31:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:logs:us-east-1:123456789012:scheduled-query:477b4380-b098-474e-9c5e-
e10a8cc2e6e7"
    ],
    "detail": {
```

```
        "queryId": "2038fd57-ab4f-4018-bb2f-61d363f4a004",
        "queryString": "fields @timestamp, @message, @logStream\n| filter @message
 like /ERROR/\n| sort @timestamp desc\n| limit 10000",
        "logGroupIdentifiers": [
            "/aws/lambda/my-function"
        ],
        "status": "Complete",
        "startTime": 1763465460,
        "statistics": {
            "recordsMatched": 1842,
            "recordsScanned": 48325,
            "estimatedRecordsSkipped": 0,
            "bytesScanned": 12081250,
            "estimatedBytesSkipped": 0,
            "logGroupsScanned": 1,
            "resultCount": 1842
        }
    }
}
```

이 쿼리는 집계되지 않으며 10,000개보다 적은 수limit의 행을 반환했으므로 일치하는 로그 이벤트 1,842개 각각은 하나의 출력 행이 되었고 recordsMatchedresultCount와 동일합니다. 집계되는 쿼리 또는에 의해 결과 집합이 잘린 쿼리는 보다 resultCount 낮은를 limit생성합니다recordsMatched. 자세한 내용은 recordsScanned, recordsMatched 및 resultCount 이해 단원을 참조하십시오.

주요 요소는 다음과 같습니다.

- statistics - 쿼리가 읽은 로그 데이터의 양과 결과 집합의 크기를 설명하는 카운터입니다. 각 필드에 대한 설명은 다음 표를 참조하십시오.
- startTime - 쿼리 실행이 시작된 시간(Unix 타임스탬프)
- queryString - 실행된 실제 쿼리
- queryId - 결과를 검색할 수 있는 쿼리의 쿼리 ID
- logGroupIdentifiers - 쿼리된 로그 그룹 목록
- status - 쿼리 실행 상태(완료, 실패 등)

다음 표에서는 statistics 객체의 각 필드를 설명합니다. 이러한 필드의 API 정의는 QueryStatistics를 참조하세요.

필드	설명
recordsScanned	쿼리 중에 스캔된 총 로그 이벤트 수입니다.
recordsMatched	쿼리 문자열과 일치하는 로그 이벤트 수입니다. 이 값은 출력 행이 아닌 로그 이벤트를 계산합니다. 결과 집합의 행 수에는를 사용합니다resultCount .
resultCount	쿼리 결과 집합의 행 수입니다. 이 값은 쿼리의 모든 작업에서 유지된 행만 계산하므로 보다 작을 수 있습니다recordsMatched . 가 GetQueryResults 반환하는 결과의 모든 페이지를 다룹니다. 자세한 내용은 recordsScanned, recordsMatched 및 resultCount 이해 단원을 참조하십시오.
estimatedRecordsSkipped	쿼리에 인덱싱된 필드가 포함되어 있기 때문에이 쿼리를 처리할 때 건너뛴 로그 이벤트 수의 추정치입니다. 이러한 항목을 건너뛰면 쿼리 비용이 절감되고 쿼리 성능 시간이 향상됩니다. 자세한 내용은 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성 단원을 참조하십시오.
bytesScanned	쿼리 중에 스캔된 로그 이벤트의 총 바이트 수입니다.
estimatedBytesSkipped	쿼리에 인덱싱된 필드가 포함되어 있기 때문에이 쿼리를 처리할 때 건너뛴 로그 이벤트의 바이트 수 추정치입니다.
logGroupsScanned	이 쿼리에서 스캔한 로그 그룹의 수입니다.

recordsScanned, recordsMatched 및 resultCount 이해

쿼리 통계 중 3개는 서로 다른 사물을 계산하며 이를 직접 비교하는 것은 오해의 소지가 있습니다. 각 단계는 서로 다른 쿼리 처리 단계를 측정합니다.

- recordsScanned - 쿼리가 로그 그룹에서 읽은 로그 이벤트 수입니다. 쿼리에 대한 입력입니다.
- recordsMatched - 쿼리 문자열과 일치하는 로그 이벤트의 수입니다. 이 값은 로그 이벤트를 계산합니다.
- resultCount - 쿼리가 생성한 결과 집합의 행 수입니다. 이 값은 출력 행을 계산합니다.

각 단계는 데이터의 범위를 좁힙니다. 와 같은 명령은 많은 로그 이벤트를 단일 출력 행으로 stats 결합하므로 보다 훨씬 작을 resultCount 수 있습니다recordsMatched. recordsMatched가 작다고 resultCount해서 결과 집합에서 행이 누락된 것은 아닙니다.

예 - 집계

다음 쿼리는 1시간 동안 한 로그 그룹의 각 로그 스트림에 있는 오류 메시지를 계산합니다.

```
filter @message like /ERROR/
| stats count(*) as errorCount by @logStream
```

쿼리가 1,500,000개의 로그 이벤트를 읽고, 그 중 24,318개에가 포함되어 있으며ERROR, 이러한 일치하는 로그 이벤트는 12개의 로그 스트림에서 가져온 경우 쿼리는 다음 통계로 완료됩니다.

```
"statistics": {
    "recordsMatched": 24318,
    "recordsScanned": 1500000,
    "estimatedRecordsSkipped": 0,
    "bytesScanned": 450000000,
    "estimatedBytesSkipped": 0,
    "logGroupsScanned": 1,
    "resultCount": 12
}
```

stats는 각 로그 스트림에 대해 하나의 행을 생성하므로 resultCount는 12이고는 24,318recordsMatched입니다. 의 12개 값은 errorCount 최대 24,318입니다.

예 - 사후 집계 필터

다음 쿼리는 1,000개 이상의 오류를 생성한 로그 스트림만 유지합니다.

```
filter @message like /ERROR/
| stats count(*) as errorCount by @logStream
| filter errorCount > 1000
```

최종 filter 명령은 그룹화 후 실행되므로 스캔에서 로그 이벤트 대신 결과 집합에서 행을 제거합니다. 12개의 로그 스트림 중 3개의가 1,000errorCount보다 큰 경우 resultCount는 12 대신 3입니다. 쿼리는 동일한 로그 데이터를 양방향으로 읽기 때문에 recordsScanned와는 변경되지 bytesScanned 않습니다.

예 - 제한

`limit` 명령은 집계 `resultCount` 없이를 줄입니다. 다음 쿼리는 가장 최근의 오류 메시지 100개를 반환합니다.

```
filter @message like /ERROR/
| sort @timestamp desc
| limit 100
```

쿼리가 동일한 1,500,000개의 로그 이벤트를 스캔하고 동일한 24,318개와 일치하는 경우 `resultCount`는 결과 집합을 100개 행으로 `limit` 제한하므로는 100입니다. 콘솔에서이 관계는 일치하는 레코드 24,318개 중 100개를 표시하는 것으로 표시됩니다.

각 통계는 서로 다른 질문에 답합니다.

이 쿼리는 몇 개의 행을 반환했습니까?

`resultCount`를 사용합니다. 값이 0이면 쿼리가 행을 생성하지 않았음을 의미합니다. 행이 아닌 로그 이벤트를 계산하므로이 용도로 `recordsMatched`를 사용하지 마십시오.

이 쿼리는 얼마나 많은 로그 데이터를 읽었나요?

`recordsScanned` 및 `bytesScanned` 사용 스캔 볼륨은 쿼리의 비용과 실행 시간을 결정합니다. 이를 줄이려면 시간 범위를 단축하거나, 더 적은 수의 로그 그룹을 쿼리하거나, 필드 인덱스를 생성합니다. 자세한 내용은 쿼리 성능을 개선하고 스캔 볼륨을 줄이기 위한 필드 인덱스 생성 단원을 참조하십시오.

이 쿼리와 일치하는 로그 이벤트는 몇 개입니까?

`recordsMatched`를 사용합니다.

> **Note**
>
> CloudWatch Logs는 통계를 0으로 보고하는 대신 객체에 사용할 수 있는 값이 없을 때 `statistics` 객체에서 통계를 생략합니다.
> 이벤트 소비자가 이벤트`resultCount`에서를 찾지 못하면 값을 0이 아닌 알 수 없음으로 취급합니다. 이벤트 소비자가 없는 통계를 허용하도록 작성하고 인식하지 못하는 통계는 무시합니다.

일정 표현식 참조

이러한 참조 테이블을 사용하여 예약된 쿼리에 대한 일정 표현식을 구성합니다. 모든 시간은 협정 세계시(UTC)입니다.

Cron 표현식 구문

형식: `cron(minute hour day-of-month month day-of-week year)`

사용 사례	cron 표현식	설명	사용 시기
일별 일정	`cron(0 9 * * ? *)`	매일 오전 9시 UTC	일일 보고서
	`cron(0 */6 * * ? *)`	6시간마다(00:00, 06:00, 12:00, 18:00 UTC)	빈번한 모니터링
	`cron(30 2 * * ? *)`	매일 오전 2시 30분 UTC	사용량이 적은 분석
업무 시간	`cron(0 9-17 ? * MON-FRI *)`	월요일부터 금요일까지 오전 9시부터 오후 5시까지 1시간마다 UTC	비즈니스 모니터링
	`cron(0 18 ? * MON-FRI *)`	평일 오후 6시 UTC	영업일 종료
	`cron(0 8,12,17 ? * MON-FRI *)`	평일 UTC 기준 오전 8시, 정오 및 오후 5시	주요 업무 시간
주간 일정	`cron(0 12 ? * SUN *)`	매주 일요일 정오 UTC	주간 요약
	`cron(0 9 ? * MON *)`	매주 월요일 오전 9시 UTC	주 시작 보고서
	`cron(0 23 ? * FRI *)`	매주 금요일 오후 11시 UTC	주말 정리
월별 일정	`cron(0 0 1 * ? *)`	매월 1일 자정 UTC	월별 보고서

사용 사례	cron 표현식	설명	사용 시기
	`cron(0 9 L * ? *)`	매월 마지막 날 오전 9시 UTC	월말 처리
	`cron(0 10 1 1,4,7,10 ? *)`	각 분기의 첫날 오전 10시 UTC	분기별 분석
높은 빈도	`cron(*/15 * * * ? *)`	15분마다	실시간 모니터링
	`cron(0,30 * * * ? *)`	30분마다(:00 및 :30)	빈번한 검사
	`cron(0 */2 * * ? *)`	2시간마다	일반 간격
특수 사례	`cron(30 8 1 1 ? *)`	1월 1일 오전 8시 30분 UTC	연간 보고서
	`cron(0 6 * * SAT,SUN *)`	UTC 기준 주말 오전 6시	주말 처리
	`cron(0 0 ? * MON#1 *)`	매월 첫 번째 월요일 자정 UTC	월별 계획

Cron 표현식 필드 참조

Field	값	와일드카드	예제
분(1분)	0~59	`* , - /`	`0` (시간 기준), `*/15` (15분마다), `0,30` (시간당 2회)
시간(2번째)	0~23	`* , - /`	9 (오전 9시), `*/2` (2시간마다), `9-17` (영업 시간)
Day-of-month(3일)	1~31, L, W	`* , - / ?`	1 (1일), L (마지막 날), ? (day-of-week 사용 시)
월(4일)	1-12 또는 JAN-DEC	`* , - /`	1 (1월), `JAN`, `1,4,7,10` (분기별)

Field	값	와일드카드	예제
Day-of-week(5일)	1-7 또는 SUN-SAT	* , - / ? # L	MON-FRI (평일), SUN, MON#1 (첫 번째 월요일)
연도(6년차)	1970~2199	* , - /	* (매년), 2024 (특정 연도), 2024-2026 (범위)

와일드카드 문자 및 특수 표현식

*****(별표)

필드의 모든 값과 일치합니다. 예: 시간 필드에서 *는 매시간을 의미합니다.

? (물음표)

특정 값이 없습니다. 다른가 지정된 경우 day-of-monthday-of-week를 사용합니다. 예: ? day-of-month 지정할 때 day-of-weekMON-FRI에 사용합니다.

-(대시)

값의 범위입니다. 예: MON-FRI (월요일~금요일), 9-17 (오전 9시~오후 5시).

, (쉼표)

여러 특정 값. 예: MON,WED,FRI (월요일, 수요일, 금요일), 8,12,17 (오전 8시, 정오, 오후 5시).

/ (슬래시)

단계 값 또는 증분입니다. 예: 분 0/15 단위는 0분(0, 15, 30, 45)부터 15분마다를 의미하고, 시간 */2 단위는 2시간마다를 의미합니다.

L (마지막)

월 마지막 날 또는 평일의 마지막 발생. 예: day-of-monthL은 마지막 날을 의미합니다.는 마지막 금요일을 FRIL 의미합니다.

W (평일)

가장 가까운 평일입니다. 예:은 해당 월의 15일에 가장 가까운 평일을 15W 의미합니다.

(n번째 발생)

월중 평일의 N번째 발생. 예: MON#1는 매월 첫 번째 월요일을 의미하고,는 매월 두 번째 금요일을 FRI#2 의미합니다.

일반적인 패턴 및 모범 사례

- 비즈니스 애플리케이션의 경우: 주말 또는 업무 외 시간에 쿼리를 실행하지 않으려면 MON-FRI 및 업무 시간(예: 9-17)을 사용합니다.
- 고주파 모니터링의 경우: */15 (15분마다)와 같은 증분을 사용하지만 쿼리 동시성 제한에 유의하세요.
- 리소스 효율성: 2-6 UTC와 같은 이른 아침 시간을 사용하여 사용량이 적은 시간에 리소스 집약적인 쿼리를 예약합니다.
- 월별 보고서의 경우: 일관된 타이밍L을 보장하기 위해 월 마지막 날 또는 첫 번째 날과 같은 특정 날짜에 1를 사용합니다.

모범 사례

다음 모범 사례를 따라 안정적이고 효율적인 예약 쿼리 작업을 보장합니다.

쿼리 최적화

- 성능 및 결과를 확인하기 위해 예약하기 전에 쿼리를 수동으로 테스트합니다.
- 쿼리 초기에 필터 인덱스를 사용하여 데이터 처리 감소
- 대용량 로그 그룹의 제한 시간을 피하기 위해 시간 범위 제한
- 쿼리 복잡성 및 실행 시간 제한 고려

일정 계획

- 예약된 다음 실행 전에 쿼리가 완료되도록 하여 중복 실행 방지
- 시간 범위를 설정할 때 로그 수집 지연 고려
- 특정 시간에 cron 표현식 사용
- 쿼리 동시성 한도에 도달하지 않도록 일정을 분산합니다.

모니터링 및 유지 관리

- 실행 기록을 정기적으로 모니터링하여 장애 또는 성능 문제 식별
- 보안을 유지하기 위해 주기적으로 IAM 역할 검토 및 업데이트
- 프로덕션에 배포하기 전에 대상 접근성 테스트

권한 부여

- 예약된 쿼리의 모든 APIs 로그 그룹과 같은 입력에서 사용하는 리소스가 아닌 예약된 쿼리 리소스에 대해 권한을 부여합니다. 그에 따라 IAM 정책 설정
- APIs에 전달된 실행 역할을 사용하여 로그 그룹의 권한 부여 관리

예약된 쿼리 시작하기

예약된 쿼리를 생성할 때 쿼리 실행 방식과 결과가 전달되는 위치를 정의하는 몇 가지 주요 구성 요소를 구성합니다. 이러한 구성 요소를 이해하면 효과적인 자동 로그 분석을 설정하는 데 도움이 됩니다.

예약된 각 쿼리는 다음과 같은 주요 구성 요소로 구성됩니다.

쿼리 구성

분석에 사용할 CloudWatch Logs Insights 쿼리 문자열, 대상 로그 그룹 및 쿼리 언어입니다.

예약 표현식

쿼리 실행 시기를 정의하는 cron 표현식 또는 빈도 달력입니다. 시간대 설정을 지정하여 쿼리가 올바른 현지 시간에 실행되도록 할 수 있습니다. 콘솔에는 “5분 동안 매주 화요일 15:10에 쿼리 실행, UTC에 즉시 적용, 무기한으로 적용”과 같이 사람이 읽을 수 있는 일정 설명이 표시됩니다.

시간 범위

각 쿼리 실행의 룩백 기간으로, 실행 시간의 시작 시간 오프셋으로 정의됩니다. 각 쿼리 실행에서 분석할 기록 데이터의 양을 결정합니다.

실행 일정 미리 보기

콘솔에는 정확한 날짜와 시간(예: 2025/10/28 15:10, UTC, 15:10, UTC, 2025/11/11 2025/11/04 15:10, UTC)이 포함된 다음 3개의 예약된 쿼리 실행이 표시되므로 일정이 올바르게 구성되었는지 확인할 수 있습니다.

대상

실행 성공 후 쿼리 결과가 전달되는 위치입니다. 지원되는 대상에는 Amazon S3 버킷이 포함되며 기본적으로 메타데이터는 기본 이벤트 버스로 전송됩니다.

실행 역할

CloudWatch Logs가 쿼리를 실행하고 결과를 지정된 대상으로 전송하기 위해 수임하는 IAM 역할입니다.

예약된 쿼리를 생성하기 전에 필요한 권한과 리소스가 구성되어 있는지 확인합니다.

예약된 쿼리 생성

CloudWatch Logs Insights 쿼리를 자동으로 실행하고 선택한 대상으로 결과를 전송하는 예약된 쿼리를 생성합니다.

사전 조건

예약된 쿼리를 생성하기 전에 다음이 있는지 확인합니다.

- 로그 그룹 - 분석하려는 데이터가 포함된 하나 이상의 로그 그룹
- 실행 IAM 역할 - 다음 권한이 있는 IAM 역할:
  - `logs:StartQuery` - CloudWatch Logs Insights 쿼리를 시작할 수 있는 권한
  - `logs:GetQueryResults` - 쿼리 결과를 검색할 수 있는 권한
  - `logs:DescribeLogGroups` - 로그 그룹 정보에 액세스할 수 있는 권한. 이는 로그 그룹 검색을 위한 접두사 기반 로그 그룹에만 필요합니다.
- 대상 권한 - 선택한 대상에 대한 추가 IAM 권한:
  - Amazon S3 대상의 경우: `s3:PutObject`
- AWS CLI 및 API 사용 - CloudWatch Logs APIs를 호출할 권한이 있는 구성된 AWS 자격 증명

자세한 IAM 정책 예제는 섹션을 참조하세요Amazon CloudWatch Logs용 Identity and Access Management. 또한 계정당 1,000개의 예약된 쿼리만 가질 수 있습니다.

Console

예약된 쿼리를 생성하려면(콘솔)

1. CloudWatch Logs 콘솔을 여시겠습니까? https://us-east-1.console.aws.amazon.com/cloudwatch/homeregion=us-east-1#logsV2:logs-insights.
2. 탐색 창에서 Logs Insights를 선택합니다.
3. 예약된 쿼리 생성을 선택합니다.
4. 쿼리 정의 섹션에서 다음을 수행합니다.
   a. 쿼리 언어의 경우 목록에서 사용할 쿼리 언어를 선택합니다.
   b. 쿼리 문자열의 경우 상자에 CloudWatch Logs Insights 쿼리를 입력합니다.
   c. 로그 그룹의 경우 목록에서 쿼리할 로그 그룹을 선택합니다.
5. 일정 설정 섹션에서 다음을 수행합니다.
   a. 일정 표현식에서 쿼리가 실행되는 시기를 구성합니다. 사전 정의된 옵션 중에서 선택하거나 사용자 지정 cron 표현식을 입력합니다.
   b. 생성 시 유효에서 일정이 활성화되는 시기를 지정합니다. YYYY/MM/DD 형식을 사용하여 즉시 또는 특정 날짜 및 시간에 시작하도록 선택합니다.

c. 시간 범위에서 각 쿼리 실행의 룩백 기간을 지정합니다. 쿼리할 실행 시간과 얼마나 멀어지는지 정의하는 지속 시간을 분 단위로 입력합니다.

d. 무기한 계속에서 일정이 종료되는 시간을 지정합니다. 무기한으로 실행하거나 YYYY/MM/DD 형식을 사용하여 특정 날짜 및 시간까지 실행하도록 선택합니다.

6. 콘솔에는 구성에 따라 다음 3개의 예약된 쿼리 실행이 표시되며 쿼리가 실행되는 정확한 날짜와 시간을 UTC 단위로 표시합니다.

7. 쿼리 결과를 S3에 게시 - 선택 사항 섹션에서(S3 대상을 사용하는 경우):

a. S3 버킷의 경우 대상 버킷이 동일한 계정에 있는 경우 이 AWS 계정을 선택하고 버킷이 다른 계정에 있는 경우 다른 AWS 계정을 선택하고 버킷 소유 계정의 계정 ID를 입력으로 제공합니다.

b. Amazon S3 URI에 결과를 저장할 Amazon S3 버킷과 접두사(예: `s3://my-bucket/query-results/`)를 입력합니다. 이 계정을 선택한 경우 Amazon S3 찾아보기를 선택하여 기존 Amazon S3 위치를 탐색하고 선택할 수 있습니다.

c. (선택 사항) KMS 키 ARN에 고객 관리형 AWS KMS 키의 ARN을 입력하여 SSE-KMS를 사용하여 쿼리 결과를 암호화합니다. 키는 대상 Amazon S3 버킷과 동일한 AWS 리전에 있어야 합니다.

8. Amazon S3에 쿼리 결과를 게시하기 위한 IAM 역할 섹션에서 다음 옵션 중 하나를 선택합니다.

a. 기본 권한이 있는 새 역할 자동 생성을 선택하여 CloudWatch Logs가 Amazon S3에 쿼리 결과를 전달하는 데 필요한 권한이 있는 IAM 역할을 자동으로 설정합니다.

b. 기존 역할 사용을 선택하여 CloudWatch Logs가 Amazon S3에 쿼리 결과를 전송하는 데 필요한 정책이 있는 기존 IAM 역할을 선택합니다. 검색 필드를 사용하여 목록에서 적절한 IAM 역할을 찾아 선택합니다.

> **Note**
>
> 계정에서 역할 관리자가 활성화된 경우 CloudWatch는 역할을 자동으로 연결하며, 여기에 설명된 역할 옵션(예: 새 역할 생성 버튼)은 사용자 지정 옵션으로 대체됩니다. 다른 역할을 사용하려면 사용자 지정을 선택합니다. IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

9. 예약된 쿼리 실행을 위한 IAM 역할 섹션에서 다음 옵션 중 하나를 선택합니다.

a. 기본 권한이 있는 새 역할 자동 생성을 선택하여 CloudWatch Logs가 예약된 쿼리를 실행하는 데 필요한 권한이 있는 IAM 역할을 자동으로 설정합니다.

b. 기존 역할 사용을 선택하여 CloudWatch Logs가 예약된 쿼리를 실행하는 데 필요한 정책이 있는 기존 IAM 역할을 선택합니다. 검색 필드를 사용하여 목록에서 적절한 IAM 역할을 찾아 선택합니다.

> **Note**
>
> 계정에서 역할 관리자가 활성화된 경우 CloudWatch는 역할을 자동으로 연결하며, 여기에 설명된 역할 옵션(예: 새 역할 생성 버튼)은 사용자 지정 옵션으로 대체됩니다. 다른 역할을 사용하려면 사용자 지정을 선택합니다. IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

10. 일정 생성을 선택하여 예약된 쿼리를 생성합니다.

AWS CLI

예약된 쿼리를 생성하려면(AWS CLI)

- `create-scheduled-query` 명령을 사용하여 새 예약된 쿼리를 생성합니다.

```
aws logs create-scheduled-query \
    --name "ErrorAnalysisQuery" \
    --query-language "CWLI" \
    --query-string "fields @timestamp, @message | filter @message like /ERROR/ |
 stats count() by bin(5m)" \
    --schedule-expression "cron(8 * * * ? *)" \
    --execution-role-arn "arn:aws:iam::123456789012:role/
CloudWatchLogsScheduledQueryRole" \
    --log-group-identifiers "/aws/lambda/my-function" "/aws/apigateway/my-api" \
    --state "ENABLED"
```

API

예약된 쿼리를 생성하려면(API)

- `CreateScheduledQuery` 작업을 사용하여 새 예약된 쿼리를 생성합니다. 다음 예제에서는 1시간마다 실행되는 예약된 쿼리를 생성합니다.

```
{
    "name": "ErrorAnalysisQuery",
    "queryLanguage": "CWLI",
    "queryString": "fields @timestamp, @message | filter @message like /ERROR/ |
 stats count() by bin(5m)",
    "scheduleExpression": "cron(8 * * * ? *)",
    "executionRoleArn": "arn:aws:iam::123456789012:role/
CloudWatchLogsScheduledQueryRole",
    "logGroupIdentifiers": ["/aws/lambda/my-function", "/aws/apigateway/my-
api"],
    "state": "ENABLED"
}
```

예약된 쿼리를 생성한 후 예약된 쿼리 페이지에서 ListScheduledQueries API를 사용하여 해당 쿼리를 보고 관리할 수 있습니다.이 API에는 이름, 생성 날짜, 마지막 실행 상태, 마지막으로 트리거된 시간 및 반복 빈도가 포함된 모든 예약된 쿼리가 표시됩니다.

예약된 쿼리 보기 및 관리

각 쿼리에 대해 다음 정보를 사용할 수 있습니다.

이름

예약된 쿼리에 할당한 고유한 이름입니다. 이름을 선택하여 자세한 구성 및 실행 기록을 봅니다.

Creation(생성) 날짜

예약된 쿼리가 생성된 날짜로, YYYY-MM-DD 형식으로 표시됩니다.

마지막 실행 상태

가장 최근 쿼리 실행의 실행 상태입니다. 가능한 값은 다음과 같습니다.

- 완료 - 쿼리가 성공적으로 실행되었으며 결과가 구성된 모든 대상으로 전송되었습니다.
- 실패 - 쿼리 실행 또는 결과 전송에 실패했습니다. 실행 기록에서 오류 세부 정보를 확인합니다.
- 잘못된 쿼리 - 쿼리가 유효하지 않고 구문 문제가 있습니다.

- 제한 시간 - 쿼리가 제한 시간을 초과했습니다. 60분 후 쿼리 시간이 자동으로 초과됨

마지막 트리거 시간

쿼리가 마지막으로 실행된 날짜와 시간으로, YYYY-MM-DD HH:MM:SS 형식으로 표시됩니다. 쿼리가 아직 실행되지 않은 경우 없음을 표시합니다.

마다 반복

쿼리의 일정 빈도입니다. cron 표현식을 사용하는 쿼리에 대한 사용자 지정 또는 더 간단한 일정에 대한 특정 빈도 설명을 표시합니다.

예약된 쿼리 페이지는 예약된 모든 쿼리에 대한 개요를 제공하며, 중앙 위치에서 예약된 모든 쿼리를 보고, 모니터링하고, 관리할 수 있도록 현재 상태 및 실행 기록을 보여줍니다. 이 정보를 사용하여 쿼리 성능을 모니터링하고, 문제를 식별하고, 자동화된 로그 분석 워크플로를 관리할 수 있습니다.

Console

예약된 쿼리를 보려면(콘솔)

1. CloudWatch Logs 콘솔을 여시겠습니까? https://us-east-1.console.aws.amazon.com/cloudwatch/homeregion=us-east-1#logsV2:logs-insights.
2. CloudWatch Logs 콘솔에서 예약된 쿼리, 예약된 쿼리 보기를 선택합니다.

AWS CLI

예약된 쿼리를 나열하려면(AWS CLI)

- `list-scheduled-queries` 명령을 사용하여 예약된 모든 쿼리를 나열합니다.

```
aws logs list-scheduled-queries --max-results 10
```

API

예약된 쿼리를 나열하려면(API)

- `ListScheduledQueries` 작업을 사용하여 예약된 모든 쿼리를 검색합니다.

```
{
    "maxResults": 10
```

```
}
```

예약된 쿼리 페이지 헤더에는 계정의 총 예약된 쿼리 수가 표시되므로 사용량을 추적하고 자동화된 로그 분석 워크플로를 효과적으로 관리할 수 있습니다.

예약된 쿼리 실행 기록 보기

실행 기록을 사용하여 예약된 쿼리의 성능을 모니터링하고 쿼리 실행 또는 결과 전송 문제를 해결합니다.

실행 내역에는 성공한 실행, 실패 및 대상 처리 결과를 포함하여 각 쿼리 실행의 상태가 표시됩니다. 이 정보를 사용하여 패턴을 식별하고, 문제를 진단하고, 쿼리가 예상대로 실행되고 있는지 확인할 수 있습니다.

Console

실행 기록을 보려면(콘솔)

1. CloudWatch Logs 콘솔에서 예약된 쿼리, 예약된 쿼리 보기를 선택합니다.
2. 검사할 예약된 쿼리를 선택합니다.
3. Execution history(실행 내역) 탭을 선택합니다.

AWS CLI

실행 기록을 보려면(AWS CLI)

1. `get-scheduled-query-history` 명령을 사용하여 예약된 쿼리의 실행 기록을 검색합니다.

```
aws logs get-scheduled-query-history \
    --identifier "DailyErrorMonitoring" \
    --start-time 1743379200 \
    --end-time 1743465600 \
    --max-results 10
```

2. 실행 상태를 기준으로 필터링하려면 `--execution-statuses` 파라미터를 추가합니다.

```
aws logs get-scheduled-query-history \
    --identifier "DailyErrorMonitoring" \
    --start-time 1743379200 \
    --end-time 1743465600 \
```

```
    --max-results 1 \
    --execution-statuses "SUCCEEDED"
```

API

실행 기록을 보려면(API)

- `GetScheduledQueryHistory` 작업을 사용하여 실행 기록을 검색합니다.

```
{
    "identifier": "DailyErrorMonitoring",
    "startTime": 1743379200,
    "endTime": 1743465600,
    "maxResults": 10,
    "executionStatuses": ["SUCCEEDED", "FAILED"]
}
```

실행 내역에 다음이 표시됩니다.

- 실행 상태 - 실행 중, 완료, 실패, 제한 시간 또는 InvalidQuery
- 트리거된 시간 - 쿼리가 실행된 시간
- 대상 - S3, EventBridge 및 조회 테이블을 포함하여 구성된 각 대상의 처리 상태
- 오류 메시지 - 쿼리 실행 또는 대상 처리 실패에 대한 세부 정보

예약된 쿼리 업데이트

요구 사항이 발전함에 따라 예약된 쿼리 구성을 수정하여 쿼리 문자열, 일정, 대상 또는 실행 역할을 변경합니다.

쿼리 문자열, 일정 표현식, 대상 및 실행 역할을 포함하여 예약된 쿼리의 모든 측면을 업데이트할 수 있습니다. 변경 사항은 향후 실행에 즉시 적용됩니다.

Console

예약된 쿼리를 업데이트하려면(콘솔)

1. CloudWatch Logs 콘솔에서 예약된 쿼리, 예약된 쿼리 보기를 선택합니다.
2. 업데이트하려는 예약된 쿼리를 선택합니다.

3. 편집을 선택합니다.
4. 필요에 따라 구성을 수정합니다.
5. 변경 사항 저장을 선택합니다.

AWS CLI

예약된 쿼리를 업데이트하려면(AWS CLI)

- `update-scheduled-query` 명령을 사용하여 기존 예약된 쿼리를 수정합니다.

```
aws logs update-scheduled-query \
    --identifier "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f" \
    --description "Monitor for ERROR level logs daily" \
    --query-language "LogsQL" \
    --query-string "fields @timestamp, @message | filter @message like /ERROR/"
 \
    --log-group-identifiers "/aws/lambda/my-function-1" "/aws/lambda/my-
function-2"
```

API

예약된 쿼리를 업데이트하려면(API)

1. `UpdateScheduledQuery` 작업을 사용하여 예약된 쿼리 구성을 수정합니다.

```
{
    "identifier": "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f",
    "queryString": "fields @timestamp, @message | filter @message like /WARNING|
ERROR/ | stats count() by bin(5m)",
    "scheduleExpression": "cron(0 */2 * * ? *)",
    "state": "ENABLED"
}
```

2. 여러 구성 파라미터를 한 번에 업데이트하려면:

```
{
    "identifier": "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f",
```

```
    "queryString": "fields @timestamp, @message, @level | filter @level =
 'ERROR'",
    "scheduleExpression": "cron(0 8,12,16 * * ? *)",
    "executionRoleArn": "arn:aws:iam::111122223333:role/
UpdatedScheduledQueryRole",
    "logGroupIdentifiers": ["/aws/lambda/my-function", "/aws/lambda/another-
function"],
    "destinationConfiguration": {
        "s3Configuration": {
            "destinationIdentifier": "s3://111122223333-sqn-results-bucket/
processed-results",
            "roleArn": "arn:aws:iam::111122223333:role/Admin"
        }
    }
}
```

예약된 쿼리에 대한 S3 대상 구성

장기 보존 및 분석을 위해 예약된 쿼리 결과를 JSON 파일로 저장할 대상으로 Amazon S3를 구성합니다.

Amazon S3를 대상으로 사용하는 경우 쿼리 결과는 지정된 버킷 및 접두사에 JSON 파일로 저장됩니다. 이 옵션은 결과를 아카이빙하거나, 배치 분석을 수행하거나, S3 데이터를 처리하는 다른 AWS 서비스와 통합하는 데 적합합니다.

쿼리 결과를 예약된 쿼리와 동일한 AWS 계정의 Amazon S3 버킷 또는 다른 계정의 버킷에 전달할 수 AWS 있습니다. 고객 관리형 AWS KMS 키(SSE-KMS)를 사용하여 쿼리 결과를 암호화할 수도 있습니다.

동일한 계정의 Amazon S3 버킷에 결과 전송

대상 Amazon S3 버킷이 예약된 쿼리와 동일한 AWS 계정에 있는 경우 콘솔에서 직접 버킷을 찾아 선택할 수 있습니다.

동일 계정 Amazon S3 대상을 구성하려면(콘솔)

1. 쿼리 결과를 S3에 게시 섹션의 S3 버킷에서 이 계정을 선택합니다.
2. Amazon S3 URI에 결과를 저장할 Amazon S3 버킷과 접두사(예: `s3://my-bucket/query-results/`)를 입력하거나 Amazon S3 찾아보기를 선택하여 기존 Amazon S3 위치를 탐색하고 선택합니다.

3. (선택 사항) 고객 관리형 AWS KMS 키로 결과를 암호화하려면 KMS AWS KMS 키 ARN 필드에 키의 ARN을 입력합니다. 키는 대상 Amazon S3 버킷과 동일한 AWS 리전에 있어야 합니다. AWS KMS 키를 지정하지 않으면 버킷의 기본 암호화 설정이 적용됩니다.
4. Amazon S3에 쿼리 결과를 게시하기 위한 IAM 역할 섹션에서 기본 권한이 있는 새 역할 자동 생성을 선택하여 필요한 권한을 자동으로 설정하거나 기존 역할 사용을 선택하여 필요한 정책이 있는 기존 IAM 역할을 선택합니다.

대상 전송 IAM 역할에는 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

다른 계정의 Amazon S3 버킷에 결과 전송

다른 AWS 계정의 Amazon S3 버킷에 예약된 쿼리 결과를 전달할 수 있습니다. 교차 계정 버킷을 사용하는 경우 Amazon S3 URI와 버킷 소유 계정의 계정 ID를 제공해야 합니다.

교차 계정 Amazon S3 대상을 구성하려면(콘솔)

1. 쿼리 결과를 S3에 게시 섹션의 S3 버킷에서 다른 계정을 선택하고 버킷 소유 계정의 계정 ID를 입력으로 제공합니다.
2. Amazon S3 URI의 경우 대상 버킷의 전체 Amazon S3 URI와 다른 계정의 접두사(예: `s3://cross-account-bucket/query-results/`)를 입력합니다.
3. (선택 사항) 고객 관리형 AWS KMS 키로 결과를 암호화하려면 KMS AWS KMS 키 ARN 필드에 키의 ARN을 입력합니다. 키는 대상 Amazon S3 버킷과 동일한 AWS 리전에 있어야 합니다.

4. Amazon S3에 쿼리 결과를 게시하기 위한 IAM 역할 섹션에서 기본 권한이 있는 새 역할 자동 생성을 선택하여 필요한 권한을 자동으로 설정하거나 기존 역할 사용을 선택하여 필요한 정책이 있는 기존 IAM 역할을 선택합니다.

교차 계정 전송에는 양쪽에 대한 권한이 필요합니다. 소스 계정의 대상 전송 IAM 역할에는 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::cross-account-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

대상 계정의 Amazon S3 버킷 정책은 소스 계정의 IAM 역할에 객체를 쓸 수 있는 권한을 부여해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowScheduledQueryRolePutObject",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/my-s3-delivery-role"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::cross-account-bucket/prefix/*"
        }
    ]
}
```

고객 관리형 AWS KMS 키를 사용하여 결과 암호화

선택적으로 고객 관리형 AWS KMS 키를 지정하여 SSE-KMS를 사용하여 Amazon S3에 전달된 쿼리 결과를 암호화할 수 있습니다. AWS KMS 키는 예약된 쿼리와 동일한 계정 또는 다른 계정에 있을 수 있습니다.

AWS KMS 키를 지정하면 예약된 쿼리는 해당 키를 사용하여 SSE-KMS를 통해 결과를 암호화합니다. AWS KMS 키를 지정하지 않으면 버킷의 기본 암호화 설정이 적용됩니다. 버킷이 고객 관리형 키를 사용하는 기본 SSE-KMS 암호화로 구성된 경우 대상 전송 IAM 역할에는 여전히 해당 키에 대한 `kms:GenerateDataKey` 권한이 있어야 합니다.

대상 전송 IAM 역할에는 AWS KMS 키에 대한 `kms:GenerateDataKey` 권한이 필요합니다. 다음 예제에서는 고객 관리형 AWS KMS 키가 있는 Amazon S3 대상에 필요한 권한을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey",
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::my-bucket*"
                }
            }
        }
    ]
}
```

AWS KMS 키가 대상 전송 IAM 역할과 다른 계정에 있는 경우 AWS KMS 키 소유 계정의 키 정책은 역할 액세스 권한을 명시적으로 부여해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowScheduledQueryRoleToEncrypt",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/my-s3-delivery-role"
            },
            "Action": "kms:GenerateDataKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::my-bucket*"
                }
            }
        }
    ]
}
```

> **Note**
>
> AWS KMS 키와 대상 전송 IAM 역할이 동일한 계정에 있는 경우 AWS KMS 키 정책에 기본 "IAM 정책 활성화" 루트 문이 포함된 경우 IAM 자격 증명 정책만으로도 충분합니다. 명시적 AWS KMS 키 정책 권한 부여는 키 정책이 IAM에 위임하지 않는 경우에만 필요합니다.

Amazon S3에 쿼리 결과를 게시하기 위한 IAM 역할은 예약된 쿼리 실행을 위한 IAM 역할과 별도로 구성해야 합니다. 이렇게 분리하면 세분화된 액세스 제어가 가능하므로 Amazon S3 역할이 특히 결과 전송을 처리하는 동안 실행 역할이 쿼리를 실행할 수 있습니다. 두 역할 모두 CloudWatch Logs 서비스(`logs.amazonaws.com`)가 역할을 수임하도록 허용하는 신뢰 정책을 포함해야 합니다.

예약된 쿼리에 대한 조회 테이블 대상 구성

조회 테이블을 대상으로 구성하여 예약된 쿼리 결과로 조회 테이블을 자동으로 새로 고칩니다. 예약된 각 실행에서 CloudWatch Logs는 지정된 조회 테이블을 쿼리 결과로 채우고 테이블을 최신 로그 데이터로 최신 상태로 유지합니다. 그런 다음 다른 로그 쿼리에서 `lookup` 명령을 사용하여 테이블을 참조할 수 있습니다. 조회 테이블에 대한 자세한 내용은 섹션을 참조하세요lookup.

조회 테이블이 없는 경우 CloudWatch Logs는 첫 번째 성공 실행 시 테이블을 생성합니다. 각 후속 실행은 전체 대체 작업이며, 기존 테이블 콘텐츠는 모두 새 쿼리 결과로 대체됩니다.

조회 테이블 구성에는 다음 설정이 포함됩니다.

- 테이블 이름 - 생성하거나 새로 고칠 조회 테이블의 이름입니다. 이름에는 영숫자와 밑줄만 사용할 수 있습니다.
- 전송 역할 - 쿼리 결과로 조회 테이블을 생성하거나 업데이트할 수 있는 CloudWatch Logs 권한을 부여하는 IAM 역할의 ARN입니다.
- 설명 - (선택 사항) 조회 테이블에 대한 설명입니다.
- AWS KMS key – (선택 사항) 조회 테이블 데이터를 암호화하는 데 사용할 AWS KMS 키의 ARN입니다. 키를 지정하지 않으면 데이터가 AWS소유 키로 암호화됩니다.
- 태그 - (선택 사항) 조회 테이블과 연결할 키-값 페어입니다. 태그는 테이블이 처음 생성될 때만 적용됩니다.

대상 전송 IAM 역할에는 다음 권한이 필요하며 CloudWatch Logs 서비스(`logs.amazonaws.com`)가 역할을 수임하도록 허용하는 신뢰 정책이 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLookupTable",
                "logs:UpdateLookupTable",
                "logs:GetQueryResults"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 조회 테이블 대상 결과에는 계정당 최대 조회 테이블 수와 같이 직접 생성하는 테이블과 동일한 조회 테이블 할당량이 적용됩니다 AWS 리전. 예약된 실행이 테이블을 새로 고치지 못하면 기존 테이블 콘텐츠는 변경되지 않고 예약된 쿼리 실행 기록에 실패가 나타납니다.

예약된 쿼리 문제 해결

이러한 문제 해결 주제를 사용하여 예약된 쿼리와 관련된 일반적인 문제를 해결합니다.

권한 오류와 함께 쿼리 실행 실패

예약된 쿼리가 실행되거나 대상에 결과를 전달하지 못하게 하는 권한 오류를 해결합니다.

권한 오류는 실행 역할에 로그 그룹에서 읽거나 대상 리소스에 쓰는 데 필요한 권한이 없을 때 발생합니다.

권한 오류를 해결하려면

1. 실행 역할에 대상 로그 그룹에 대한 `logs:StartQuerylogs:GetQueryResults`, 및 `logs:DescribeLogGroups` 권한이 있는지 확인합니다.
2. 실행 역할에 대상 리소스(예: S3 버킷)`s3:PutObject`에 대한 쓰기 권한이 있는지 확인합니다.
3. 신뢰 정책에서 CloudWatch Logs가 실행 역할을 수임하도록 허용하는지 확인합니다. 역할은 신뢰 정책에서 로그 서비스 보안 주체(`logs.amazonaws.com`)를 신뢰해야 합니다.

일반적인 원인으로는 IAM 권한 누락, 정책의 잘못된 리소스 ARNs 또는 신뢰 정책 구성 문제가 있습니다.

권한 오류를 방지하려면 프로덕션에 예약된 쿼리를 배포하기 전에 실행 역할을 생성하고 권한을 테스트할 때 최소 권한 원칙을 사용합니다.

쿼리 제한 시간

예약된 쿼리가 최대 실행 시간 제한을 초과할 때 발생하는 제한 시간 오류를 해결합니다.

쿼리 제한 시간은 쿼리가 지정된 데이터 범위를 처리하는 데 60분 이상 걸릴 때 발생합니다. 이는 종종 대규모 데이터 세트 또는 복잡한 쿼리 로직으로 인해 발생합니다.

제한 시간 오류를 해결하려면

1. 시작 시간 오프셋을 줄여 실행당 더 적은 데이터를 처리하여 시간 범위를 줄입니다.
2. 쿼리 초기에 필터를 추가하여 처리되는 데이터의 양을 줄여 쿼리를 최적화합니다. 필터 인덱스를 사용하여 데이터 스캔 크기를 줄입니다.
3. 복잡한 쿼리를 더 간단하고 집중적인 쿼리로 나누는 것이 좋습니다.

일반적인 원인으로는 대규모 시간 범위 쿼리, 대용량 로그 그룹 처리 또는 적절한 필터링 없이 복잡한 집계 사용 등이 있습니다.

제한 시간을 방지하려면 CloudWatch Logs Insights에서 예상 데이터 볼륨으로 쿼리를 수동으로 테스트하고 예약 전에 성능을 최적화합니다.

대상 처리 실패

예약된 쿼리 결과를 구성된 대상으로 전송할 수 없을 때 발생하는 실패를 해결합니다.

대상 Amazon S3 버킷 또는 EventBridge 이벤트 버스에 액세스할 수 없거나 잘못 구성된 경우 대상 처리 실패가 발생합니다.

쿼리 결과가 대상에 게시되지 않는 실패를 해결하려면

1. 지정된 Amazon S3 버킷이 존재하고 액세스할 수 있는지 확인합니다.
2. 대상 구성에서 올바른 URIs 확인합니다.
3. 실행 역할에 대상에 쓰는 데 필요한 권한이 있는지 확인합니다.

일반적인 원인으로는 삭제되거나 이름이 변경된 대상 리소스, 잘못된 대상 URIs 또는 네트워크 연결 문제가 있습니다.

대상 장애를 방지하려면 대상 구성을 정기적으로 검증하고 대상 리소스 가용성을 모니터링합니다.

잘못된 쿼리 오류

성공적인 실행을 방해하는 예약된 쿼리 문자열의 구문 및 로직 오류를 해결합니다.

쿼리 문자열에 구문 오류가 포함되거나 존재하지 않는 필드를 참조하거나 지원되지 않는 쿼리 언어 기능을 사용할 때 잘못된 쿼리 오류가 발생합니다.

잘못된 쿼리 오류를 해결하려면

1. CloudWatch Logs Insights에서 쿼리를 수동으로 테스트하여 구문과 로직을 확인합니다.
2. 참조된 모든 로그 필드가 대상 로그 그룹에 존재하는지 확인합니다.
3. 사용 중인 쿼리 언어 기능이 예약된 쿼리에 지원되는지 확인합니다.

일반적인 원인에는 필드 이름의 오타, 잘못된 쿼리 구문 또는 예약된 실행 환경에서 지원되지 않는 쿼리 기능 사용이 포함됩니다.

잘못된 쿼리 오류를 방지하려면 예약하기 전에 항상 대화형으로 쿼리를 테스트하고 필드 검색 기능을 사용하여 필드 이름을 확인합니다.

동시성 오류 쿼리

예약된 쿼리가 Cloudwatch Logs 인사이트 쿼리와 동일한 할당량을 사용하므로 동시성 오류가 표시될 때 유의해야 할 몇 가지 중요한 사항이 아래에 나와 있습니다. 동시성 한도에 도달하지 않도록 일정을 분산하는 것이 좋습니다.

- 할당량: AWS 계정당 최대 100개의 동시 CloudWatch Logs Insights 쿼리를 실행할 수 있습니다.
- 대시보드: CloudWatch 대시보드에 추가된 쿼리도이 동시성 제한에 포함됩니다. 대시보드가 로드되거나 새로 고쳐질 때 실행되기 때문입니다.
- OpenSearch Service PPL/SQL: AWS 계정당 최대 15개의 동시 OpenSearch PPL 또는 OpenSearch SQL 쿼리를 실행할 수 있습니다.
- 교차 계정 쿼리: 동시성 할당량은 단일 및 교차 계정 쿼리 모두에 적용됩니다. CloudWatch 교차 계정 관찰성을 사용하는 경우 연결된 소스 계정에 대해 모니터링 계정에서 시작된 쿼리도 모니터링 계정의 동시성 한도에 포함됩니다.
- 빈번하지 않은 액세스 로그 그룹: 빈번하지 않은 액세스 로그 클래스의 로그 그룹의 경우 동시 Logs Insights 쿼리의 최대 수는 5개로 제한됩니다.

로그 이상 탐지

로그 데이터에서 이상 징후를 감지할 수 있는 방법은 지속적인 모니터링을 위해 로그 이상 탐지기를 생성하는 것, 그리고 온디맨드 분석을 위해 CloudWatch Log Insights 쿼리에서 `anomaly detection` 명령을 사용하는 것입니다.

로그 이상 탐지기는 로그 그룹에 수집된 로그 이벤트를 스캔하여 로그 데이터에서 이상을 자동으로 찾습니다. 이상 탐지는 기계 학습 및 패턴 인식을 사용하여 일반적인 로그 콘텐츠의 기준을 설정합니다. 온디맨드 분석의 경우 CloudWatch Logs Insights 쿼리에서 `anomaly detection` 명령을 사용하여 시계열 데이터의 비정상적인 패턴을 식별할 수 있습니다. 쿼리 기반 이상 탐지에 대한 자세한 내용은 CloudWatch Logs Insights에서 이상 탐지 사용을 참조하세요.

로그 그룹 이상 탐지기를 생성하면 이상 탐지기는 로그 그룹의 지난 2주 동안의 로그 이벤트를 사용하여 훈련됩니다. 훈련 기간은 최대 15분이 소요될 수 있습니다. 훈련이 완료되면 들어오는 로그를 분석하여 이상을 식별하기 시작하고, 검사할 수 있도록 CloudWatch Logs 콘솔에 이상이 표시됩니다.

CloudWatch Logs 패턴 인식은 로그에서 정적 및 동적 콘텐츠를 식별하여 로그 패턴을 추출합니다. 많은 수의 로그 이벤트를 몇 가지 패턴으로 압축할 수 있기 때문에 패턴은 큰 로그 집합을 분석하는 데 유용합니다.

예를 들어, 다음 세 가지 로그 이벤트 샘플을 참조하세요.

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for ResourceID: 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for ResourceID: 324892398123-1234R
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for ResourceID: 3ff231242342-12345
```

이전 샘플에서 세 로그 이벤트는 모두 한 가지 패턴을 따릅니다.

```
<Date-1> <Time-2> [INFO] Calling DynamoDB to store for resource id <ResourceID-3>
```

패턴 내의 필드를 토큰이라고 합니다. 요청 ID나 타임스탬프 같이 패턴 내에서 달라지는 필드를 동적 토큰이라고 합니다. 동적 토큰에 대해 발견된 각 값을 토큰 값이라고 합니다.

CloudWatch Logs는 동적 토큰이 나타내는 데이터 유형을 추론할 수 있는 경우 토큰을 <*string*-*number*>으로 표시합니다. *string*은 토큰이 나타내는 데이터 유형에 대한 설명입니다. *number*는 다른 동적 토큰과 비교하여 패턴에서 이 토큰이 나타나는 위치를 보여줍니다.

CloudWatch Logs는 문자열을 포함하는 로그 이벤트의 콘텐츠 분석을 기반으로 이름의 문자열 부분을 할당합니다.

CloudWatch Logs는 동적 토큰이 나타내는 데이터 유형을 추론할 수 없는 경우 토큰을 <Token-*number*>로 표시하며, *number*는 패턴에서 이 토큰이 나타나는 위치를 다른 동적 토큰과 비교하여 나타냅니다.

동적 토큰의 일반적인 예로는 오류 코드, IP 주소, 타임스탬프, 요청 ID가 있습니다.

로그 이상 탐지는 이러한 패턴을 사용하여 이상을 찾습니다. 이상 탐지기 모델 훈련 기간이 지나면 알려진 추세와 비교하여 로그가 평가됩니다. 이상 탐지기는 상당한 변동을 이상으로 플래그 지정합니다.

이 장에서는 이상 탐지를 활성화하고, 이상을 보고, 로그 이상 탐지기를 위한 경보를 생성하고, 로그 이상 탐지기가 게시하는 지표를 생성하는 방법을 설명합니다. 또한 이상 탐지기와 그 결과를 암호화하는 방법도 설명합니다 AWS Key Management Service.

로그 이상 탐지기 생성에는 요금이 발생하지 않습니다.

이상 및 패턴의 심각도 및 우선순위

로그 이상 탐지기가 발견한 각 이상에는 우선순위가 할당됩니다. 발견된 각 패턴에는 심각도가 할당됩니다.

- 우선순위는 자동으로 계산되며, 패턴의 심각도 수준 및 예상 값과의 편차 양을 기반으로 합니다. 예를 들어, 특정 토큰 값이 갑자기 500% 증가하는 경우 심각도가 `NONE`이더라도 해당 이상을 `HIGH` 우선순위로 지정할 수 있습니다.
- 심각도는 `FATAL`, `ERROR`, `WARN` 등 패턴에서 발견된 키워드만을 기반으로 합니다. 이러한 키워드를 찾을 수 없는 경우 패턴의 심각도는 `NONE`으로 표시됩니다.

이상 가시성 시간

이상 탐지기를 생성할 때 이상 탐지기의 최대 가시성 기간을 지정합니다. 이 기간은 콘솔에 이상이 표시되고 ListAnomalies API 작업에 의해 이상이 반환되는 일수입니다. 이 기간이 경과한 후에도 계속 발생하는 이상은 자동으로 일반 동작으로 수락되고 이상 탐지기 모델은 이를 이상 현상으로 더 이상 플래그 지정하지 않습니다.

이상 탐지기를 생성할 때 가시성 시간을 조정하지 않으면 21일이 기본값으로 사용됩니다.

이상 항목 억제

이상이 발견된 후 일시적으로 또는 영구적으로 이를 억제하도록 선택할 수 있습니다. 이상 항목 억제를 선택하면 이상 탐지기는 지정한 시간 동안 이를 이상으로 플래그 지정하지 않습니다. 이상을 억제할 때

해당 특정 이상만 억제하거나 이상이 발견된 패턴과 관련된 모든 이상을 억제하도록 선택할 수 있습니다.

콘솔에서는 여전히 억제된 이상을 볼 수 있습니다. 이상 억제를 중지하도록 선택할 수도 있습니다.

자주 묻는 질문(FAQ)

내 데이터를 AWS 사용하여 AWS 사용 또는 다른 고객을 위한 기계 학습 알고리즘을 훈련하나요?

아니요. 훈련에서 생성된 이상 탐지 모델은 로그 그룹의 로그 이벤트를 기반으로 하며 해당 로그 그룹 및 해당 AWS 계정 내에서만 사용됩니다.

이상 탐지에 적합한 로그 이벤트 유형은 무엇입니까?

로그 이상 탐지가 적합한 로그 이벤트 유형은 애플리케이션 로그 및 대부분의 로그 항목이 일반적인 패턴에 맞는 기타 유형의 로그입니다. 로그 수준 또는 INFO, ERROR, DEBUG 같은 심각도 키워드가 포함된 이벤트가 있는 로그 그룹은 로그 이상 탐지에 특히 적합합니다.

로그 이상 탐지가 적합하지 않은 로그 이벤트는 CloudTrail Logs 같이 JSON 구조가 매우 긴 로그 이벤트입니다. 패턴 분석은 로그 줄의 처음 1,500자까지만 분석하므로 해당 한도를 넘는 모든 문자는 건너뜁니다.

VPC 흐름 로그 같은 감사 또는 액세스 로그도 이상 탐지 성공률이 떨어집니다. 이상 탐지는 애플리케이션 문제를 찾기 위한 것이므로 네트워크 또는 액세스 이상에는 적합하지 않을 수 있습니다.

CloudWatch Logs 패턴 분석을 사용하여 그룹의 로그 이벤트에서 패턴의 수를 찾으면 이상 탐지기가 특정 로그 그룹에 적합한지 여부를 결정하는 데 도움이 됩니다. 패턴의 수가 약 300개 이하인 경우 이상 탐지가 잘 작동할 수 있습니다. 패턴 분석에 대한 자세한 내용은 패턴 분석을 참조하세요.

무엇이 이상으로 플래그 지정되나요?

다음과 같은 경우 로그 이벤트가 이상으로 플래그 지정될 수 있습니다.

- 로그 그룹에서 이전에 보이지 않던 패턴이 있는 로그 이벤트.
- 알려진 패턴의 상당한 변형.
- 일반적 값의 이산 집합이 있는 동적 토큰의 새 값.
- 동적 토큰의 특정 값 발생 횟수의 큰 변화.

위의 모든 항목이 이상으로 플래그 지정될 수 있지만, 모두 다 애플리케이션이 제대로 작동하지 않는다는 의미는 아닙니다. 예를 들어, 평소보다 많은 수의 200 성공 값은 이상으로 플래그가 지정될 수 있습니다. 이와 같은 경우 문제를 나타내지 않는 이러한 이상을 억제하는 것이 좋습니다.

마스킹 처리되는 민감한 데이터는 어떻게 되나요?

로그 이벤트에서 민감한 데이터로 마스킹 처리된 부분은 이상을 스캔하지 않습니다. 자세한 내용은 Help protect sensitive log data with masking을 참조하세요.

CloudWatch Logs Insights에서 이상 탐지 사용

지속적인 모니터링을 위한 로그 이상 탐지기를 생성하는 것 외에도 CloudWatch Logs Insights 쿼리에서 anomaly 명령을 사용하여 온디맨드 방식으로 로그 데이터의 비정상적인 패턴을 식별할 수도 있습니다. 이 명령은 기존 pattern 기능을 확장하고 기계 학습을 사용하여 패턴 빈도 변경, 새 패턴, 토큰 변형 등 5가지 유형의 이상을 감지합니다.

anomaly 명령은 다음과 같은 경우에 특히 유용합니다.

- 비정상적인 패턴을 식별하기 위한 기록 로그 데이터의 임시 분석
- 비정상적인 동작에 대한 특정 기간 조사
- Lambda 함수와 같은 애플리케이션의 실행 문제 모니터링

쿼리에서 anomaly 명령 사용에 대한 자세한 내용은 이상을 참조하세요.

이 쿼리 기반 이상 탐지는 다음 섹션에 설명된 연속 이상 탐지기를 보완하여 실시간 모니터링과 온디맨드 분석 기능을 모두 제공합니다.

로그 그룹에서 이상 탐지 활성화

다음 단계에 따라 CloudWatch 콘솔을 사용하여 로그 그룹에 이상이 있는지 스캔하는 로그 이상 탐지기를 생성합니다.

프로그래밍 방식으로 이상 탐지기를 생성할 수도 있습니다. 자세한 내용은 CreateLogAnomalyDetector를 참조하세요.

로그 이상 탐지기를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 로그, 로그 이상 항목을 선택합니다.

3. 이상 탐지기 생성을 선택합니다.
4. 이 이상 탐지기를 생성할 로그 그룹을 선택합니다.
5. 이상 탐지기 이름에 탐지기 이름을 입력합니다.
6. (선택 사항) 평가 빈도를 기본값인 5분에서 변경합니다. 로그 그룹이 새 로그를 수신하는 빈도에 따라 이 값을 설정합니다. 예를 들어, 로그 그룹이 10분마다 배치로 새 로그 이벤트를 수신하는 경우 평가 빈도를 15분으로 설정하는 것이 적절할 수 있습니다.
7. (선택 사항) 특정 단어나 문자열이 포함된 로그 이벤트에서만 이상을 찾도록 이상 탐지기를 구성하려면 패턴 필터링을 선택합니다.

   그런 다음 이상 탐지 필터 패턴에 패턴을 입력합니다. 패턴 구문에 대한 자세한 내용은 지표 필터, 구독 필터, 필터 로그 이벤트 및 Live Tail에 대한 필터 패턴 구문을 참조하세요.

   (선택 사항) 필터 패턴을 테스트하려면 로그 이벤트 메시지에 몇몇 로그 메시지를 입력한 다음 패턴 테스트를 선택합니다.
8. (선택 사항) 이상 표시 기간을 기본값에서 변경하거나 AWS KMS 키를이 이상 탐지기와 연결하려면 고급 구성을 선택합니다.

   a. 이상 가시성 기간을 기본값에서 변경하려면 최대 이상 항목 가시성 기간(일)에 새 값을 입력합니다.

   b. AWS KMS 키를이 이상 탐지기와 연결하려면 KMS 키 ARN에 ARN을 입력합니다. 키를 할당하면 이 탐지기가 발견한 이상 정보가 키를 사용하여 저장 시 암호화됩니다. 사용자는 이 키에 대한 권한과 이상 탐지기에 대한 권한이 있어야 이상 탐지기가 발견한 이상에 대한 정보를 검색할 수 있습니다.

      또한 CloudWatch Logs 서비스 보안 주체에게 키를 사용할 수 있는 권한이 있는지 확인해야 합니다. 자세한 내용은 를 사용하여 이상 탐지기 및 그 결과 암호화 AWS KMS 단원을 참조하십시오.
9. 이상 탐지 활성화를 선택합니다.

   이상 탐지기가 생성되고, 로그 그룹이 수집 중인 로그 이벤트에 따라 모델 훈련을 시작합니다. 약 15분 후에 이상 탐지가 활성화되고 이상을 찾아서 표시하기 시작합니다.

발견된 이상 보기

하나 이상의 로그 이상 탐지기를 생성한 후 CloudWatch 콘솔을 사용하여 이상 탐지기가 발견한 이상을 볼 수 있습니다.

프로그래밍 방식으로 이상을 볼 수 있습니다. 자세한 내용은 ListAnomalies를 참조하세요.

모든 로그 이상 탐지기에서 발견된 이상을 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 로그, 로그 이상 항목을 선택합니다.

   로그 이상 항목 테이블이 나타납니다. 로그 이상 항목 옆의 상단에 있는 숫자는 테이블에 나열된 로그 이상 수를 표시합니다. 테이블의 각 행에는 다음과 같은 정보가 표시됩니다.

   - 이상 열에는 이상의 간략한 요약이 표시됩니다. 이러한 요약은 CloudWatch Logs에서 생성됩니다.
   - 이상 현상의 우선순위. 우선순위는 로그 이벤트의 변화량, 로그 이벤트에서 발생한 `Exception` 같은 키워드 등을 기반으로 자동으로 계산됩니다.
   - 이상의 기반이 되는 로그 패턴. 패턴에 대한 자세한 내용은 로그 이상 탐지을 참조하세요.
   - 이상 로그 추세는 패턴과 일치하는 로그 볼륨을 나타내는 히스토그램을 표시합니다.
   - 마지막 탐지 시간에는 이 이상이 발견된 최신 시간이 표시됩니다.
   - 첫 번째 탐지 시간에는 이 이상이 처음 발견된 시간이 표시됩니다.
   - 이상 탐지기에는 이 이상과 관련된 로그 이벤트를 포함하는 로그 그룹의 이름이 표시됩니다. 이 이름을 선택하면 로그 그룹 세부 정보 페이지를 볼 수 있습니다.
3. 이상 하나를 추가로 검사하려면 해당 행에서 라디오 버튼을 선택합니다.

   패턴 검사 창이 나타나고 다음이 표시됩니다.

   - 이 이상의 기반이 되는 로그 패턴. 패턴 내에서 토큰을 선택하여 해당 토큰의 값을 분석합니다.
   - 쿼리된 시간 범위에서 이상의 발생 횟수를 보여주는 히스토그램.
   - 로그 샘플 탭에는 이상의 일부인 몇 가지 로그 이벤트가 표시됩니다.
   - 토큰 값 탭에는 선택한 동적 토큰의 값이 표시됩니다(동적 토큰을 선택한 경우).

   > **Note**
   >
   > 각 토큰마다 최대 10개의 토큰 값이 캡처됩니다. 토큰 개수가 정확하지 않을 수 있습니다. CloudWatch Logs는 확률 계수기를 사용하여 절대값이 아닌 토큰 개수를 생성합니다.

4. 이상을 억제하려면 해당 행에서 라디오 버튼을 선택한 후 다음을 수행합니다.

a. 작업, 이상 항목 억제를 선택합니다.

b. 그런 다음 이상을 억제할 기간을 지정합니다.

c. 이 패턴과 관련된 모든 이상을 억제하려면 패턴 억제를 선택합니다.

d. 이상 항목 억제를 선택합니다.

단일 로그 그룹에서 발견된 이상을 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 로그, 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택한 다음 이상 탐지 탭을 선택합니다.

   이상 탐지 테이블이 나타납니다. 로그 이상 항목 옆의 상단에 있는 숫자는 테이블에 나열된 로그 이상 수를 표시합니다. 테이블의 각 행에는 다음과 같은 정보가 표시됩니다.

   - 이상 열에는 이상의 간략한 요약이 표시됩니다. 이러한 요약은 CloudWatch Logs에서 생성됩니다.
   - 이상 현상의 우선순위. 우선순위는 로그 이벤트의 변화량, 로그 이벤트에서 발생한 `Exception` 같은 키워드 등을 기반으로 자동으로 계산됩니다.
   - 이상의 기반이 되는 로그 패턴. 패턴에 대한 자세한 내용은 로그 이상 탐지을 참조하세요.
   - 이상 로그 추세는 패턴과 일치하는 로그 볼륨을 나타내는 히스토그램을 표시합니다.
   - 마지막 탐지 시간에는 이 이상이 발견된 최신 시간이 표시됩니다.
   - 첫 번째 탐지 시간에는 이 이상이 처음 발견된 시간이 표시됩니다.
4. 이상 하나를 추가로 검사하려면 해당 행에서 라디오 버튼을 선택합니다.

   패턴 검사 창이 나타나고 다음이 표시됩니다.

   - 이 이상의 기반이 되는 로그 패턴. 패턴 내에서 토큰을 선택하여 해당 토큰의 값을 분석합니다.
   - 쿼리된 시간 범위에서 이상의 발생 횟수를 보여주는 히스토그램.
   - 로그 샘플 탭에는 이상의 일부인 몇 가지 로그 이벤트가 표시됩니다.
   - 토큰 값 탭에는 선택한 동적 토큰의 값이 표시됩니다(동적 토큰을 선택한 경우).

> **Note**
>
> 각 토큰마다 최대 10개의 토큰 값이 캡처됩니다. 토큰 개수가 정확하지 않을 수 있습니다. CloudWatch Logs는 확률 계수기를 사용하여 절대값이 아닌 토큰 개수를 생성합니다.

5. 이상을 억제하려면 해당 행에서 라디오 버튼을 선택한 후 다음을 수행합니다.

   a. 작업, 이상 항목 억제를 선택합니다.

   b. 그런 다음 이상을 억제할 기간을 지정합니다.

   c. 이 패턴과 관련된 모든 이상을 억제하려면 패턴 억제를 선택합니다.

   d. 이상 항목 억제를 선택합니다.

로그 이상 탐지기에 대한 경보 생성

로그 그룹에서 로그 이상 탐지기를 위한 경보를 생성할 수 있습니다. 지정된 기간 동안 로그 그룹에서 지정된 횟수의 이상이 발견되면 ALARM 상태가 되도록 경보를 지정할 수 있습니다. 필터를 사용하여 경보에서 지정된 우선순위의 이상만 계산되도록 할 수도 있습니다.

로그 이상 탐지기에 대한 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.

   로그 이상 탐지기 테이블이 나타납니다.
3. 경보를 설정하려는 이상 탐지기의 라디오 버튼을 선택하고 경보 생성을 선택합니다.

   CloudWatch 경보 생성 마법사가 나타납니다. LogAnomalyDetector 필드에는 선택한 이상 탐지기의 이름이 표시됩니다. 지표 이름 필드에 AnomalyCount가 표시됩니다.
4. (선택 사항) 이상 우선순위로 이 경보를 필터링하려면 다음 중 하나를 수행합니다.
   - 경보에서 우선순위가 높은 이상만 계산되도록 하려면 LogAnomalyPriority에 **`HIGH`**를 입력합니다.
   - 경보에서 우선순위가 높음 및 중간인 이상만 계산되도록 하려면 LogAnomalyPriority에 **`MEDIUM`**를 입력합니다.

   우선순위 수준에 대한 자세한 내용은 이상 및 패턴의 심각도 및 우선순위을 참조하세요.

5. 경보에 정적 또는 지표 이상 탐지 임계값을 사용하도록 선택합니다. 이 선택에 따라 경보 임곗값 설정 방법이 결정됩니다. 정적 임곗값은 경보 임곗값이 선택한 정적 상수 숫자임을 의미합니다. 이상 탐지 임곗값은 CloudWatch가 일반적인 값의 범위를 결정하고 실제 개수가 이 대역의 임곗값을 초과하면 경보가 트리거됨을 의미합니다. 로그 이상 탐지 경보에 대해 이상 탐지를 선택할 필요는 없습니다. 지표 이상 탐지에 대한 자세한 내용은 CloudWatch 이상 탐지 사용을 참조하세요.
6. ***your-metric-name*** is... 일 때마다 Greater, Greater/Equal, Lower/Equal 또는 Lower을 선택합니다. than . . .에 임곗값에 대한 숫자를 지정합니다. 이상 탐지기가 기간에 지정된 시간 동안 이 경보 횟수보다 많은 경보를 찾으면 경보가 ALARM 상태로 전환됩니다.
7. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

   N개 중 M번째 경보를 생성하려면 두 번째 값의 숫자보다 작은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가를 참조하세요.
8. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락된 데이터를 처리하는 방법 구성을 참조하세요.
9. 다음을 선택합니다.
10. 알림에서 알림 추가를 선택한 다음, 경보가 ALARM, OK 또는 INSUFFICIENT_DATA 상태로 전환될 때 알림을 보낼 Amazon SNS 주제를 지정합니다.

    a. (선택 사항) 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내려면 알림 추가를 선택합니다.

    > **Note**
    >
    > 경보 상태가 되는 경우와 함께 데이터 부족 상태가 되는 경우에도 조치를 취하도록 경보를 설정하는 것이 좋습니다. 데이터 소스에 연결되는 Lambda 함수와 관련된 많은 문제로 인해 경보가 데이터 부족 상태로 전환될 수 있기 때문입니다.

    b. (선택 사항) Amazon SNS 알림을 보내지 않으려면 제거를 선택합니다.
11. (선택 사항) 경보가 Amazon EC2 Auto Scaling, Amazon EC2, 티켓에 대한 작업을 수행하도록 하려면 적절한 버튼을 AWS Systems Manager선택하고 경보 상태 및 작업을 지정합니다.

> **Note**
>
> 경보는 ALARM 상태일 때만 Systems Manager 작업을 수행할 수 있습니다. 시스템 관리자 작업에 대한 자세한 내용은 OpsItems 및 Incident 생성을 위한 CloudWatch 구성을 참조하세요.

12. 다음을 선택합니다.
13. 이름 및 설명에서 경보의 이름과 설명을 입력하고 다음을 선택합니다. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.

> **Tip**
>
> 경보 이름에는 UTF-8 문자만 포함되어야 합니다. ASCII 제어 문자를 포함할 수 없습니다.

14. 미리 보기 및 생성에서 경보의 정보와 조건이 올바른지 확인하고 경보 생성을 선택합니다.

로그 이상 탐지기가 게시한 지표

CloudWatch Logs는 AnomalyCount 지표를 CloudWatch 지표에 게시합니다. 이 지표는 AWS/Logs 네임스페이스에 게시됩니다.

AnomalyCount 지표는 다음 차원으로 게시됩니다.

- LogAnomalyDetector - 이상 탐지기의 이름
- LogAnomalyPriority - 이상의 우선순위 수준

를 사용하여 이상 탐지기 및 그 결과 암호화 AWS KMS

이상 탐지기 데이터는 항상 CloudWatch Logs에서 암호화됩니다. 기본적으로 CloudWatch Logs는 저장 데이터에 서버 측 암호화를 사용합니다. 또는이 암호화 AWS Key Management Service 에를 사용할 수 있습니다. 이렇게 하면 AWS KMS 키를 사용하여 암호화가 완료됩니다. AWS KMS 를 사용한 암호화는 KMS 키를 이상 탐지기와 연결하여 이상 탐지기 수준에서 활성화됩니다.

> **Important**
>
> CloudWatch Logs는 대칭 KMS 키만 지원합니다. 비대칭 키를 사용하여 로그 그룹의 데이터를 암호화하지 마세요. 자세한 내용은 대칭 및 비대칭 키 사용을 참조하세요.

제한 사항

- 다음 단계를 수행하려면 `kms:CreateKey`, `kms:GetKeyPolicy` 및 `kms:PutKeyPolicy` 권한이 있어야 합니다.
- 로그 그룹에 키 연결 작업 또는 로그 그룹에서 키 연결 해제 작업이 적용되려면 최대 5분이 소요될 수 있습니다.
- 연결된 키에 대한 CloudWatch Logs 액세스 권한을 취소하거나 연결된 KMS 키를 삭제한 경우에는 CloudWatch Logs에서 암호화된 데이터를 더 이상 검색할 수 없습니다.

1단계: AWS KMS 키 생성

다음 create-key 명령을 사용하여 KMS 키 생성:

```
aws kms create-key
```

이 명령의 출력 화면에는 키의 키 ID와 Amazon Resource Name(ARN)이 포함됩니다. 다음은 예제 출력입니다.

```
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "key-default-1",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/key-default-1",
        "AWSAccountId": "123456789012",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
```

```
        ]
    }
}
```

2단계: KMS 키에 대한 권한 설정

기본적으로 모든 AWS KMS 키는 프라이빗입니다. 리소스 소유자만 이를 사용하여 데이터를 암호화 및 해독할 수 있습니다. 그러나 리소스 소유자가 원한다면 다른 사용자 및 리소스에 KMS 키에 대한 액세스 권한을 부여할 수 있습니다. 이 단계에서는 키를 사용할 보안 주체 권한을 CloudWatch Logs 서비스에 제공합니다. 이 서비스 보안 주체는 KMS 키가 저장된 리전과 동일한 AWS 리전에 있어야 합니다.

가장 좋은 방법은 KMS 키 사용을 지정한 AWS 계정 또는 이상 탐지기로만 제한하는 것입니다.

먼저 다음 get-key-policy 명령을 사용하는 `policy.json`로 KMS 키에 대한 기본 정책을 저장합니다.

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./
policy.json
```

텍스트 편집기에서 `policy.json` 파일을 열고 다음 명령문 중 하나에서 굵은 글꼴로 표시된 섹션을 추가합니다. 기존 명령문과 새 명령문을 쉼표로 구분합니다. 이러한 문은 `Condition` 섹션을 사용하여 AWS KMS 키의 보안을 강화합니다. 자세한 내용은 AWS KMS 키 및 암호화 컨텍스트 단원을 참조하십시오.

이 예제의 `Condition` 섹션에서는 AWS KMS 키 사용을 지정된 계정으로 제한하지만 모든 이상 탐지기에서 사용할 수 있습니다.

JSON

```
{
"Version":"2012-10-17",
"Id": "key-default-1",
"Statement": [
    {
        "Sid": "EnableIAMUserPermissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
        },
        "Action": "kms:*",
        "Resource": "*"
```

```
        },
        {
            "Sid": "AllowCloudWatchLogsEncryption",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
                }
            }
        },
        {
            "Sid": "AllowCloudWatchLogsDescribeKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        },
        {
            "Sid": "AllowCloudWatchLogsReEncryption",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws-crypto-ec:aws:logs:arn":
 "arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
                }
            }
        },
        {
            "Sid": "AllowCloudWatchLogsDescribeKeyForReEncryption",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

마지막으로 아래 put-key-policy 명령을 사용하여 업데이트된 정책을 추가합니다.

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

3단계: KMS 키를 이상 탐지기와 연결

콘솔에서 또는 AWS CLI 또는 APIs.

4단계: 이상 탐지기에서 키 연결 해제

키가 이상 탐지기와 연결된 후에는 키를 업데이트할 수 없습니다. 키를 제거하는 유일한 방법은 이상 탐지기를 삭제한 다음 다시 생성하는 것입니다.

CloudWatch Logs Live Tail로 문제 해결

CloudWatch Logs Live Tail을 사용하면 모이는 새 로그 이벤트의 스트리밍 목록을 보고 인시던트를 신속하게 해결할 수 있습니다. 모든 로그를 거의 실시간으로 보고 필터링하고 강조 표시할 수 있으므로 문제를 빠르게 탐지하고 해결할 수 있습니다. 지정한 항을 기반으로 로그를 필터링하고 원하는 항목을 빠르게 찾을 수 있도록 지정된 항이 포함된 로그를 강조 표시할 수도 있습니다.

Live Tail 세션은 세션 사용 시간(분)별로 비용이 발생합니다. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금의 로그 탭을 참조하세요.

Live Tail은 Standard 로그 클래스의 로그 그룹에서만 지원됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스을 참조하세요.

다음 섹션에서는 콘솔과 AWS CLI에서 Live Tail을 사용하는 방법을 설명합니다. 프로그래밍 방식으로 Live Tail 세션을 시작할 수도 있습니다. 자세한 내용은 StartLiveTail을 참조하세요. SDK 예제는 AWS SDK를 사용하여 Live Tail 세션 시작을 참조하세요.

AWS Toolkit for Visual Studio Code에서 Live Tail을 사용할 수도 있습니다. VS Code Command Palette에서 Live Tail 세션을 시작하려면 AWS Toolkit for Visual Studio Code 사용 설명서의 Amazon CloudWatch Logs Live Tail 섹션을 참조하세요.

Live Tail 기능은 모든 commercial AWS Regions에서 사용할 수 있습니다. 중국 리전 또는 AWS GovCloud(미국) 리전에서는 사용할 수 없습니다.

> **Note**
>
> `StartLiveTail` API는 SDK 호스트 접두사 삽입을 사용하여 요청을 라우팅합니다. 2026년 4월 1일 이전에 릴리스된 SDK 버전은 VPC 엔드포인트를 지원하지 `streaming-logs.`*`Region`*`.amazonaws.com`않는 로 라우팅됩니다. 2026년 4월 1일 이후에 릴리스된 SDK 버전은 VPC 엔드포인트`stream-logs.`*`Region`*`.amazonaws.com`를 지원하는 로 라우팅됩니다. 이 API에 대한 VPC 엔드포인트를 설정하려면 CloudWatch Logs에 대한 VPC 엔드포인트 생성을 참조하세요.

를 사용하여 Live Tail 세션 시작 AWS CLI

`start-live-tail` AWS CLI 명령은 터미널에서 하나 이상의 로그 그룹에 대해 Live Tail 스트리밍 세션을 시작합니다. Live Tail 세션은 최대 3시간 동안 지속될 수 있습니다. 초당 500개 이상의 로그 이벤

트가 필터와 일치하는 경우 표시되는 로그 이벤트는 전체 로그 이벤트의 샘플로서 실시간 테일링 경험을 제공합니다. `start-live-tail` 명령에 대한 자세한 내용은 start–live-tail을 참조하세요.

`start-live-tail`은 다음 두 가지 모드로 사용할 수 있습니다.

- print-only - 기본 모드
- interactive

print-only

`print-only` 모드에서 로그 이벤트는 터미널에서 스트리밍됩니다. 새 이벤트는 1초마다 하단에 추가되어 Linux의 `tail -f`와 유사한 거의 실시간의 테일링 경험을 생성합니다.

print-only 모드에서 Live Tail 세션을 시작하려면 다음 명령을 입력하세요.

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs
```

print-only 모드를 사용하는 경우 다른 Linux 명령과 파이핑하여 분석 기능을 높일 수도 있습니다. 다음 예제에서는 특정 정보를 추출할 수 있도록 `error` 키워드로 로그 이벤트를 필터링하고 이러한 이벤트의 두 번째 및 네 번째 열을 인쇄합니다.

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode print-only | grep "error" | awk '{print
 $2, $4}'
```

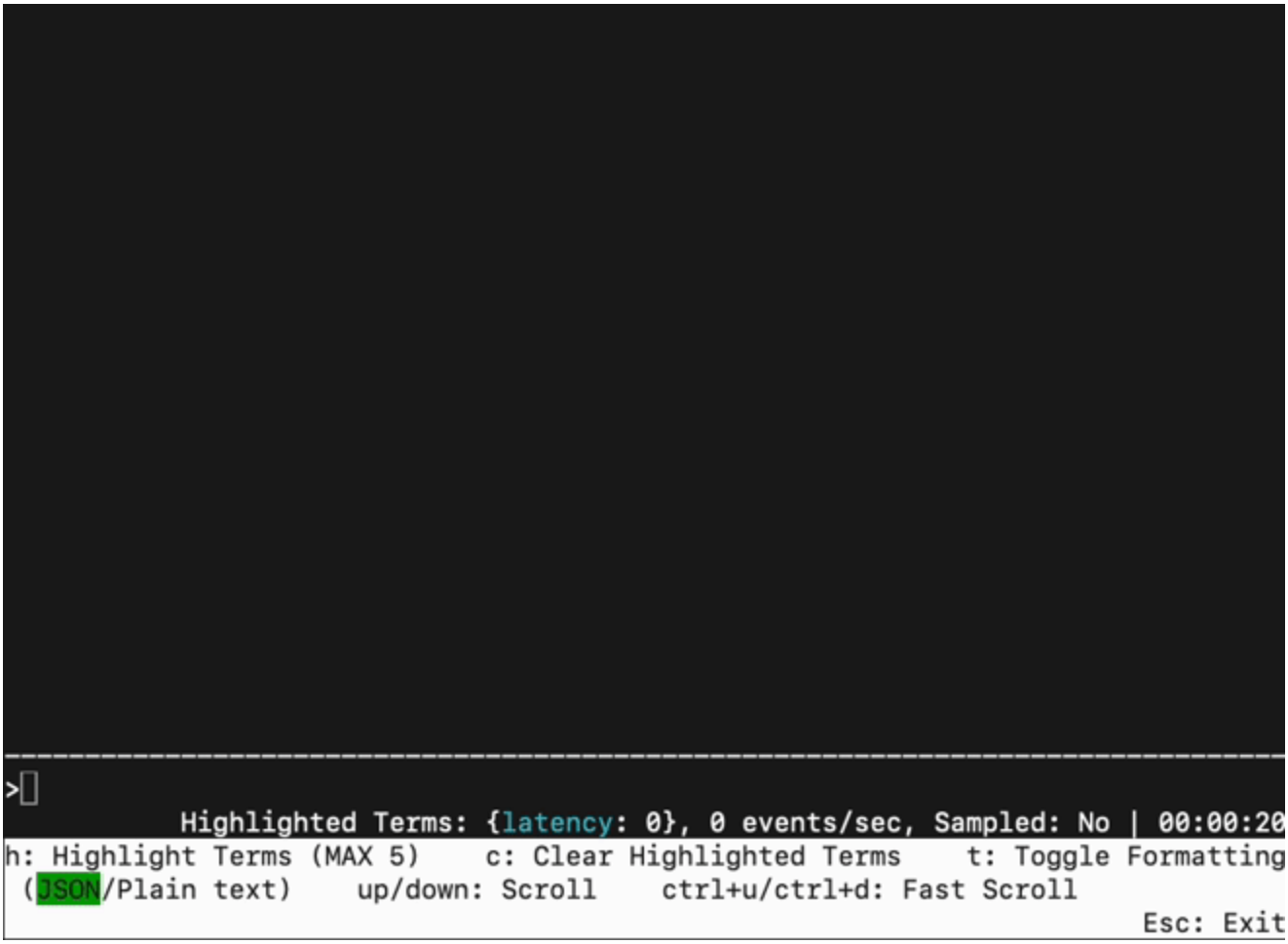
interactive

`interactive` 모드에서는 용어를 강조 표시하고 출력 로그 이벤트 형식을 JSON과 일반 텍스트 간에 전환할 수 있습니다. interactive 모드에는 세션 지속 시간, 세션이 샘플링되는지 여부, 현재 강조 표시된 용어, 해당 용어가 발견된 횟수 등 Live Tail 세션에 대한 정보도 표시됩니다.

interactive 모드에서 Live Tail 세션을 시작하려면 다음 명령을 입력합니다.

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode interactive
```

Live Tail 세션이 시작됩니다. 다음 동영상은 예제 세션의 일부를 보여줍니다.



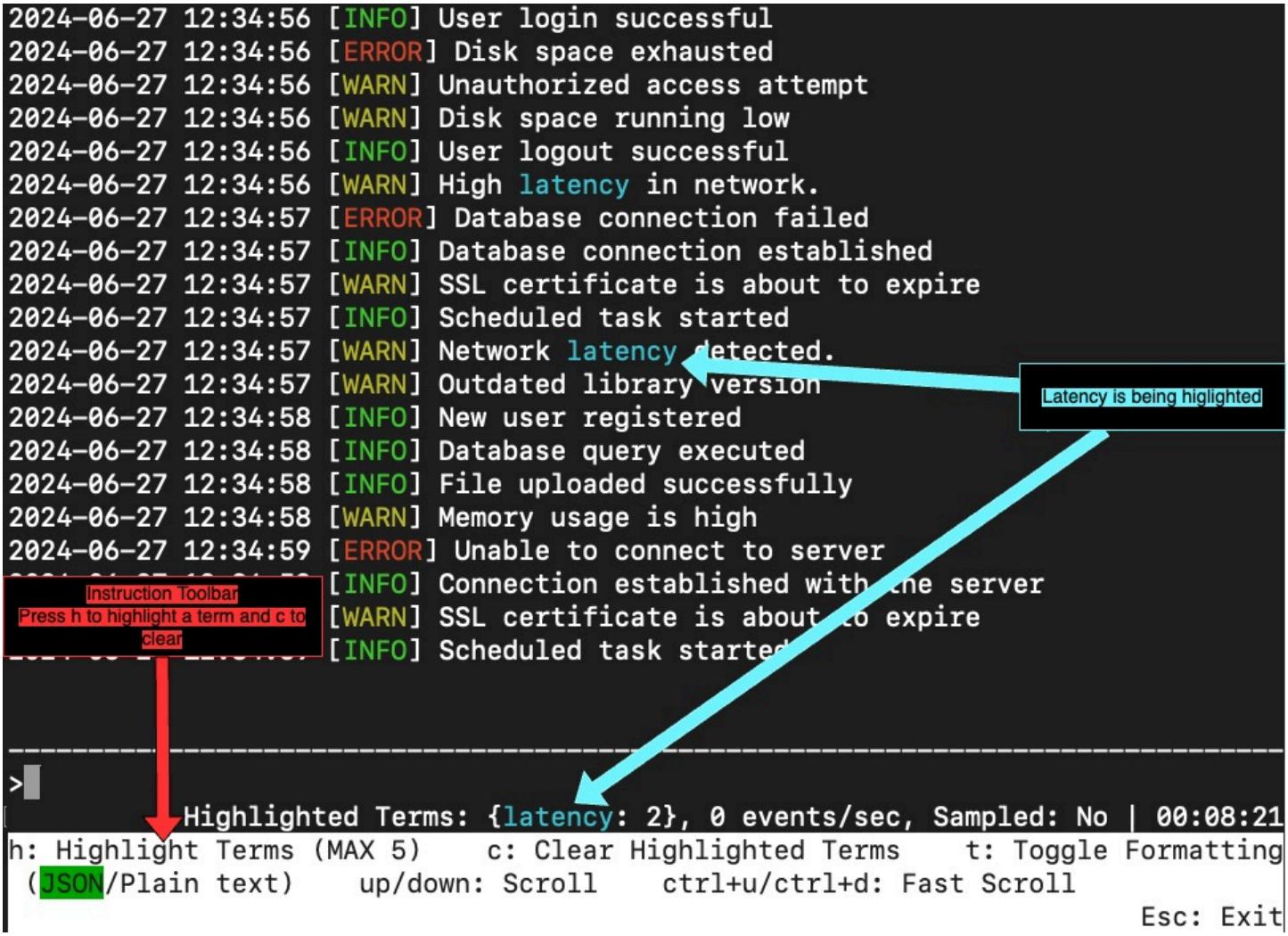
스트리밍 로그에서 용어를 강조 표시하려면 h를 누른 다음 용어를 입력합니다. 다음은 용어 `latency`가 강조 표시된 후의 화면을 보여줍니다.

강조 표시된 용어를 지우려면 c를 누른 다음 강조 표시를 중지하려는 용어를 나타내는 숫자를 입력합니다.

t를 눌러 수신 이벤트의 표시 형식을 JSON과 일반 텍스트 간에 전환할 수 있습니다. 이 토글 기능은 최선(best-effort)이며 로그 이벤트 형식이 호환되는 경우에만 전환됩니다.

위쪽 화살표와 아래쪽 화살표 키를 사용하여 스크롤하고 `CTRL+u`와 `CTRL+d`를 사용하면 더 빠르게 스크롤할 수 있습니다.

다음 이미지는 Live Tail 세션 중 `latency` 용어의 강조 표시를 보여줍니다.



콘솔에서 Live Tail 세션 시작

CloudWatch 콘솔을 사용하여 Live Tail 세션을 시작합니다. 다음 절차에서는 왼쪽 탐색 창의 Live tail 옵션을 사용하여 Live Tail 세션을 시작하는 방법을 설명합니다. 로그 그룹 페이지나 CloudWatch 로그 인사이트 페이지에서 Live Tail 세션을 시작할 수도 있습니다.

데이터 보호 정책을 사용하여 Live Tail로 보고 있는 로그 그룹의 민감한 데이터를 마스킹하는 경우 민감한 데이터는 항상 Live Tail 세션에서 마스킹된 상태로 나타납니다. 로그 그룹의 민감한 데이터 마스킹 처리에 대한 자세한 내용은 마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원을 참조하세요.

> **Important**
>
> 네트워크 보안 팀에서 웹 소켓 사용을 허용하지 않는 경우 현재 CloudWatch 콘솔의 Live Tail 부분에 액세스할 수 없습니다. AWS CLI 또는 APIs. 자세한 내용은 를 사용하여 Live Tail 세션 시작 AWS CLI 및 StartLiveTail을 참조하세요.

Live Tail 세션을 시작하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, Live tail을 선택합니다.
3. 로그 그룹 선택에서 Live Tail 세션에서 이벤트를 보려는 로그 그룹을 선택합니다. 로그 그룹을 10개까지 선택할 수 있습니다.
4. (선택 사항) 로그 그룹을 하나만 선택한 경우 로그 이벤트를 보려는 로그 스트림을 하나 이상 선택하여 Live Tail 세션을 추가로 필터링할 수 있습니다. 이렇게 하려면 로그 스트림 선택 아래의 드롭다운 목록에서 로그 스트림 이름을 선택합니다. 또는 로그 스트림 선택 아래의 두 번째 상자를 사용하여 로그 스트림 이름 접두사를 입력할 수 있습니다. 그러면 접두사와 일치하는 이름의 모든 로그 스트림이 선택됩니다.
5. (선택 사항) 특정 단어나 다른 문자열을 포함하는 로그 이벤트만 표시하려면 `Add filter patterns`에 단어나 문자열을 입력합니다.

   예를 들어 `Warning`이라는 단어가 포함된 로그 이벤트만 표시하려면 **`Warning`**을 입력합니다. 필터 필드는 대소문자를 구분합니다. 이 필드에 여러 항과 패턴 연산자를 포함할 수 있습니다.

   - **`error 404`**는 `error`와 `404`를 모두 포함하는 로그 이벤트만 표시합니다.
   - **`?Error ?error`**는 `Error` 또는 `error`를 포함하는 로그 이벤트를 표시합니다.
   - **`-INFO`**는 `INFO`를 포함하지 않는 모든 로그 이벤트를 표시합니다.
   - **`{ $.eventType = "UpdateTrail" }`**은 이벤트 유형 필드의 값이 `UpdateTrail`인 모든 JSON 로그 이벤트를 표시합니다.

   정규식(regex)을 사용하여 필터링할 수도 있습니다.

   - **`%ERROR%`**은(는) regex를 사용하여 ERROR 키워드로 구성된 모든 로그 이벤트를 표시
   - **`{ $.names = %Steve% }`**은(는) Steve가 속성 `"name"`에 속해 있는 경우 regex를 사용하여 JSON 로그 이벤트를 표시

- **`[ w1 = %abc%, w2 ]`**은(는) regex를 사용하여 첫 단어가 abc인 공백으로 구분된 로그 이벤트를 표시

패턴 구문에 대한 자세한 내용은 필터 패턴 구문을 참조하세요.

6. (선택 사항) 표시된 로그 이벤트 중 일부를 강조 표시하려면 Live Tail에 검색 및 강조 표시할 용어를 입력합니다. 강조 표시 항을 한 번에 하나씩 입력합니다. 강조 표시할 항을 여러 개 추가하면 각 항을 나타내는 다른 색상이 할당됩니다. 강조 표시기는 지정된 항이 포함된 로그 이벤트의 왼쪽에 표시되며, 전체 로그 이벤트를 보기 위해 기본 창에서 로그 이벤트를 확장할 때도 항 자체 아래에 표시됩니다.

   강조 표시와 함께 필터링을 사용하여 문제를 신속하게 해결할 수 있습니다. 예를 들어 이벤트를 필터링하여 `Error`가 포함된 이벤트만 표시한 다음 `404`가 포함된 이벤트를 강조 표시할 수도 있습니다.
7. 세션을 시작하려면 필터 적용을 선택합니다.

   일치하는 로그 이벤트가 창에 나타나기 시작합니다. 다음 정보도 표시됩니다.

   - 타이머는 Live Tail 세션이 활성화된 시간을 표시합니다.
   - 이벤트/초는 설정한 필터와 일치하는 초당 모은 로그 이벤트 수를 표시합니다.
   - 많은 이벤트가 필터와 일치하므로 세션이 너무 빨리 스크롤되지 않도록 CloudWatch Logs에서 일치하는 이벤트 일부만 표시할 수 있습니다. 이 경우 화면에 표시되는 일치하는 이벤트의 비율이 % 표시됨에 표시됩니다.
8. 이벤트 흐름을 일시 중지하여 현재 표시된 내용을 조사하려면 이벤트 창에서 아무 곳이나 선택합니다.
9. 세션 중 다음을 사용하여 각 로그 이벤트에 대한 추가 세부 정보를 볼 수 있습니다.

   - 기본 창에 로그 이벤트의 전체 텍스트를 표시하려면 해당 로그 이벤트 옆에 있는 화살표를 선택합니다.
   - 보조 창에 로그 이벤트의 전체 텍스트를 표시하려면 해당 로그 이벤트 옆에 있는 + 돋보기를 선택합니다. 이벤트 흐름이 일시 중지되고 보조 창이 나타납니다.

     보조 창에 로그 이벤트 텍스트를 표시하면 해당 텍스트를 기본 창의 다른 이벤트와 비교하는 데 유용할 수 있습니다.
10. Live Tail 세션을 중지하려면 중지를 선택합니다.
11. 세션을 다시 시작하려면 필요에 따라 필터 패널을 사용하여 필터링 기준을 수정하고 필터 적용을 선택합니다. 그런 다음 시작을 선택합니다.

교차 계정 교차 리전 로그 중앙 집중화

Amazon CloudWatch Logs 데이터 중앙 집중화는와 함께 작동 AWS Organizations 하여 교차 계정 및 교차 리전 중앙 집중화 규칙을 사용하여 여러 멤버 계정의 로그 데이터를 하나의 데이터 리포지토리로 수집합니다. 여러 계정 및 AWS 리전 의 로그 데이터를 조직 내 중앙 집중화된 계정으로 자동 복제하는 규칙을 정의합니다. 이 기능은 로그 통합을 간소화하여 전체 AWS 인프라에서 중앙 집중식 모니터링, 분석 및 규정 준수를 개선합니다.

CloudWatch Logs 데이터 중앙 집중화는 대상 계정 내에서 규칙 설정 중에 백업 리전을 구성하여 복원력을 높이는 등 운영 및 보안 요구 사항을 충족할 수 있는 구성 유연성을 제공합니다. 또한 소스 계정에서 복사한 로그 그룹의 암호화 동작을 완벽하게 제어하여 고객 관리형 KMS 키로 원래 암호화된 데이터를 처리할 수 있습니다.

> **Note**
>
> CloudWatch Logs 중앙 집중화 기능은 중앙 집중화 규칙을 생성한 후 소스 계정에 도착하는 새 로그 데이터만 처리합니다. 기록 로그 데이터(규칙 생성 전에 존재한 로그)는 중앙 집중화되지 않습니다.

데이터 중앙 집중화 개념

CloudWatch Logs 데이터 중앙 집중화 사용을 시작하기 전에 다음 개념을 숙지합니다.

중앙 집중화 규칙

: 소스 계정 및 리전의 로그 데이터를 대상 계정 및 리전에 복제하는 방법을 정의하는 구성. 규칙은 소스 기준 및 대상 설정을 지정합니다.

소스 계정

: 로그 데이터가 시작되는 AWS 계정입니다. 소스 계정의 로그 이벤트는 정의한 중앙 집중화 규칙에 따라 대상 계정으로 복제됩니다.

대상 계정

: 복제된 로그 데이터가 저장되는 대상 AWS 계정입니다. 이 계정은 로그 분석 및 모니터링을 위한 중앙 집중화 위치 역할을 합니다.

백업 리전

복원력 향상 및 재해 복구를 위해 로그 데이터를 복제할 수 있는 대상 계정 내의 선택적 보조 리전.

태그 전파

소스 로그 그룹에서 해당 대상 로그 그룹으로 리소스 태그를 전파하는 옵트인 기능입니다. 태그 전파는 대상 계정의 고객 관리형 IAM 역할을 사용하여 대상 로그 그룹에 태그를 추가, 업데이트 및 제거합니다. 중앙 집중화 규칙의 대상 구성에 `TagPropagationConfiguration` 블록을 추가하여 태그 전파를 구성합니다.

CloudWatch Logs에서의 암호화

로그 그룹 데이터는 항상 CloudWatch Logs에서 암호화됩니다. 기본적으로 CloudWatch Logs는 256비트 고급 암호화 표준 갈루아/카운터 모드(AES-GCM)를 사용하는 서버 측 암호화로 서버 측 암호화를 사용하여 저장 로그 데이터를 암호화합니다. 또는이 암호화에 AWS Key Management Service를 사용할 수 있습니다. 자세한 내용은 CloudWatch Logs Encryption 설명서를 참조하세요.

- 중앙 집중화 중 암호화 작동 방식: CloudWatch Logs 중앙 집중화는 소스 계정에서 대상 계정으로 수집 시 로그 데이터를 능동적으로 복사합니다. 이 프로세스 중에 데이터는 AWS 소유 서비스 키를 사용하여 전송 중에 암호화된 상태로 유지됩니다. 소스 및 대상 로그 그룹 모두에 저장된 데이터는 선택한 암호화 방법(고객 관리형 또는 AWS 소유 KMS 키)을 사용하여 암호화됩니다. 대상 로그 그룹에서 고객 관리형 KMS 키를 사용하는 경우 중앙 집중화 서비스가 액세스할 수 있도록 `LogsManaged = true` kms 키에 태그를 추가합니다.
- KMS 권한이 필요한 경우:
  - 소스 계정에서 고객 관리형 KMS 키를 사용하는 경우 CloudWatch Logs에는 다음 예제 시나리오에서 KMS 권한이 필요합니다.
    - 처리량 관리: 중앙 집중화 처리량 한도에 도달하면 대역폭을 사용할 수 있을 때까지 고객 관리형 KMS 키로 암호화된 로그 데이터가 일시적으로 저장됩니다.
    - 데이터 보호 및 수정: 소스 로그 그룹에 데이터 보호 정책이 활성화된 경우 CloudWatch Logs는 원시 로그 데이터에 액세스하여 중앙 집중화하려면 복호화 권한이 필요합니다.

> **Important**
>
> 중앙 집중화 규칙은 AWS Organizations 관리 계정 또는 위임된 관리자가 관리합니다. 고객 관리형 KMS 암호화 로그 그룹을 중앙 집중화에서 제외하려면 규칙 설정을 " AWS KMS 키로 암호화된 로그 그룹을 중앙 집중화하지 마십시오."로 구성합니다.

로그 중앙 집중화 설정

CloudWatch Logs Centralization을 설정하려면 소스 계정의 로그 그룹에서 대상 계정의 로그 그룹으로 로그 데이터가 흐르는 방식을 정의하는 중앙 집중화 규칙을 구성해야 합니다.

중앙 집중화 규칙이 활성화되고 로그 이벤트가 대상 계정에 복제되면 향상된 필터링 기능을 사용하여 중앙 집중식 로그 그룹에 지표, 구독 및 계정 필터를 생성할 수 있습니다. 이러한 필터는 특정 소스 계정 및 리전의 로그 이벤트를 대상으로 할 수 있으며 소스 계정 및 리전 정보를 지표 차원으로 내보낼 수 있습니다. 자세한 내용은 필터를 사용하여 로그 이벤트에서 지표 생성 단원을 참조하십시오.

사전 조건

- AWS Organizations 를 설정해야 하며 소스 계정과 대상 계정 모두 조직에 속해야 합니다.
- 로그 데이터에 대한 액세스를 제공하려면 CloudWatch, 관리 계정 및 대상 계정에 대해 신뢰할 수 있는 액세스를 활성화해야 합니다.

> **Note**
>
> 콘솔을 통해 필요한 서비스 연결 역할(SLR)을 자동으로 생성하는 신뢰할 수 있는 액세스를 활성화하는 것이 좋습니다. 다른 방법을 통해 신뢰할 수 있는 액세스가 활성화된 경우 서비스 연결 역할을 별도로 생성해야 합니다.

태그 전파 사전 조건

태그 전파를 활성화하려면 다음 추가 설정을 완료해야 합니다.

고객 관리형 IAM 역할 생성

다음 구성을 사용하여 대상 계정에서 고객 관리형 IAM 역할을 생성해야 합니다.

신뢰 정책

역할의 신뢰 정책은 중앙 집중화 서비스 연결 역할이 이를 수임하도록 허용해야 합니다. 다음 예제 신뢰 정책은 `AWSServiceRoleForObservabilityAdmin_LogsCentralization` 서비스 연결 역할에 대상 역할을 수임할 수 있는 권한을 부여합니다. `<destination-account-id>`를 대상 계정 ID로 바꾸고를 AWS Organizations ID`<organization-id>`로 바꿉니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::<destination-account-id>:role/aws-
service-role/logs-centralization.observabilityadmin.amazonaws.com/
AWSServiceRoleForObservabilityAdmin_LogsCentralization"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "<organization-id>"
      }
    }
  }]
}
```

권한 정책

역할의 권한 정책은 대상 로그 그룹에 태그 작업을 부여해야 합니다. 다음 예제에서는 최소 필수 권한을 부여합니다. <destination-account-id>를 값으로 바꿉니다. 의 범위를 특정 로그 그룹 ARN 패턴으로 Resource 더 지정할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "logs:ListTagsForResource",
      "logs:TagResource",
      "logs:UntagResource"
    ],
    "Resource": "arn:aws:logs:*:<destination-account-id>:log-group:*"
  }]
}
```

iam:PassRole 권한 부여

iam:PassedToService 조건 키를 통해 중앙 집중화 서비스로 범위가 지정된 대상 역할에 대한 iam:PassRole 권한이 있어야 합니다. 다음 예제에서는 역할을 전달할 수 있는 권한을 부여합니다. <destination-account-id> 및 <your-tag-role-name>을 사용자의 값으로 바꿉니다.

> **Note**
>
> 이 문은 대상 역할 자체가 아닌 자격 증명 정책(CreateCentralizationRuleForOrganization 또는를 호출하는 보안 주체UpdateCentralizationRuleForOrganization)에 적용됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::<destination-account-id>:role/<your-tag-role-name>",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "logs-centralization.observabilityadmin.amazonaws.com"
      }
    }
  }]
}
```

대상 로그 그룹 이름 사용자 지정

중앙 집중화 규칙을 생성할 때 속성을 사용하여 대상 로그 그룹 이름을 구성하는 방법을 사용자 지정할 수 있습니다. 로그 그룹이 생성되면 이러한 속성이 실제 값으로 자동으로 대체되므로 대상 계정에서 계층적으로 로그를 구성할 수 있습니다. 기본적으로 속성만 사용되며,이 ${source.logGroup} 속성은 대상 계정에서 이름이 동일한 모든 로그 그룹을 병합합니다. 변수를 확인할 수 없는 경우 계층 구조의 상위 변수에서 값을 상속합니다.

사용 가능한 속성

대상 로그 그룹 이름 패턴에 다음 속성을 사용할 수 있습니다.

대상 로그 그룹 이름 속성

속성	설명
${source.accountId}	로그가 시작된 AWS 계정 ID입니다.
${source.region}	로그가 시작된 AWS 리전 입니다.

속성	설명
${source.logGroup}	소스 계정의 원래 로그 그룹 이름입니다.
${source.org.id}	소스 계정의 AWS Organizations ID입니다.
${source.org.ouId}	소스 계정의 조직 단위 ID
${source.org.rootId}	조직 루트 ID
${source.org.path}	계정에서 루트까지의 전체 조직 경로

예제

원래 로그 그룹 구조 보존

패턴: `/centralized/${source.accountId}${source.logGroup}`

Result: `/centralized/123456789012/aws/lambda/my-function`

계정 및 리전별로 구성

패턴: `/centralized/${source.accountId}/${source.region}`

Result: `/centralized/123456789012/us-east-1`

조직 구조별로 구성

패턴: `/logs/${source.org.id}/${source.org.ouId}/${source.accountId}`

Result: `/logs/o-abc123/ou-xyz-12345678/123456789012`

단순 플랫 구조

패턴: `/centralized-logs`

Result: `/centralized-logs`

모범 사례

- 소스 계정 ID를 포함하여 어떤 계정 로그에서 왔는지 쉽게 식별합니다.

- 여러 리전에서 중앙 집중화하는 경우 소스 리전을 포함합니다.
- 대상 로그 그룹 이름을 512자 미만으로 구성합니다. CloudWatch Logs는 최대 512자의 로그 그룹 이름 길이를 적용합니다.
- 태그 전파를 활성화하는 경우 패턴에는 , 및 `${source.logGroup}`의 세 가지 속성이 모두 포함되어야 합니다`${source.accountId}`. `${source.region}`. 이렇게 하면 각 소스 로그 그룹이 고유한 대상 로그 그룹에 매핑됩니다.

중앙 집중화 규칙 생성

다음 절차에 따라 소스 계정에서 대상 계정으로 로그 데이터를 복제하는 중앙 집중화 규칙을 생성합니다.

중앙 집중화 규칙을 생성하려면

1. 조직의 '관리 또는 위임된 관리자' 계정에서 CloudWatch 콘솔로 이동합니다.
2. 설정을 선택합니다.
3. 조직 탭으로 이동합니다.
4. 규칙 구성을 선택합니다.
5. 다음 필드를 설정하여 소스 세부 정보를 지정한 다음, 다음을 선택합니다.

   a. 중앙 집중화 규칙 이름: 중앙 집중화 규칙의 고유한 이름을 입력합니다.

   b. 소스 계정: 원격 측정 데이터를 중앙 집중화할 계정을 선택하기 위한 소스 선택 기준을 정의합니다. 선택 기준에는 다음이 포함될 수 있습니다.

      - 조직에서의 멤버 계정의 목록
      - 조직에서의 조직 단위의 목록
      - 전체 조직

      두 가지 모드로 선택 기준을 제공할 수 있습니다.

      - 빌더: 소스 선택 기준을 생성하기 위한 클릭 기반 환경
      - 편집기: 소스 선택 기준을 제공하기 위한 자유 형식 텍스트 상자

      소스 선택 기준에 지원되는 구문:

      - 지원되는 키: OrganizationId | OrganizationUnitId | AccountId | *

- 지원되는 연산자: = | IN | OR

c. 소스 리전: 중앙 집중화할 원격 측정 데이터를 찾을 리전 목록을 선택합니다.

6. 다음 필드를 설정하여 대상 세부 정보를 지정한 다음, 다음을 선택합니다.

a. 대상 계정: 원격 측정 데이터의 중앙 대상 역할을 하는 조직의 계정을 선택합니다.

b. 대상 리전: 중앙 집중식 원격 측정 데이터의 사본을 저장하는 기본 리전을 선택합니다.

c. 백업 리전: 중앙 집중식 원격 측정 데이터의 두 번째 복사본을 저장하는 리전을 선택적으로 선택합니다.

7. 다음 필드를 설정하여 원격 측정 데이터를 지정한 다음, 다음을 선택합니다.

a. 로그 그룹: 다음 옵션 중 하나를 선택합니다:

- 모든 로그 그룹: 소스 계정의 모든 로그 그룹에서 로그를 중앙 집중화합니다.
- 로그 그룹 필터링: 선택 기준과 일치하는 소스 계정의 로그 그룹 하위 집합에서 로그를 중앙 집중화합니다. 두 가지 모드로 선택 기준을 제공할 수 있습니다.
  - Builder: 선택 기준을 생성하기 위한 선택 기반 환경
  - 편집기: 선택 기준을 제공하는 자유 형식 텍스트 상자

  로그를 필터링하는 데 사용할 수 있는 두 가지 선택 기준이 있습니다.
  - 로그 그룹 선택 기준: 중앙 집중화할 소스 로그 그룹을 지정하는 선택 기준입니다.
    - 지원되는 키: LogGroupName | *
    - 지원되는 연산자: = | != | IN | NOT IN | 및 | 또는 | LIKE | NOT LIKE
  - 데이터 소스 선택 기준: 중앙 집중화할 데이터 소스를 지정하는 선택 기준입니다.
    - 지원되는 키: DataSourceName | DataSourceType
    - 지원되는 연산자: = | != | IN | NOT IN | 및 | 또는 | LIKE | NOT LIKE

  로그 그룹 선택 기준과 데이터 소스 선택 기준을 모두 지정하면 로그 이벤트가 두 기준과 일치해야 중앙 집중화됩니다.

b. KMS 암호화된 로그 그룹

> **Important**
>
> 중앙 집중화 규칙에 제공된 KMS 키가 CloudWatch Logs의 사용을 허용하지 않는 경우 CloudWatch 중앙 집중화 규칙은 소스 계정에서 대상 로그 그룹으로 로그를 전달하지 못합니다. 대상 로그 그룹에서 고객 관리형 KMS 키를 사용하는 경우

LogsManaged = true 태그를 kms 키에 추가합니다. 자세한 내용은 2단계: KMS 키에 대한 권한 설정 단원을 참조하십시오.

다음 옵션 중 하나를 선택하세요.

- 대상별 고객 관리형 KMS 키를 사용하여 고객 관리형 KMS 키로 암호화된 소스 로그 그룹 중앙 집중화: 고객 관리형 KMS 키로 암호화된 소스 로그 그룹의 로그 이벤트를 대상 계정의 고객 관리형 KMS 키로 암호화된 대상 로그 그룹으로 중앙 집중화합니다.

  이 설정을 선택하면 다음 사항도 설정해야 합니다.

  - 대상 암호화 키 ARN: 새로 생성된 대상 로그 그룹과 연결할 대상 계정 및 기본 대상 리전의 고객 관리형 KMS 키 ARN입니다.
  - 백업 대상 암호화 키 ARN(백업 리전을 선택한 경우): 새로 생성된 대상 로그 그룹과 연결할 대상 계정 및 백업 대상 리전의 고객 관리형 KMS 키 ARN입니다.
  - 암호화되지 않은 대상 로그 그룹으로 중앙 집중화 건너뛰기(선택 사항): 고객 관리형 KMS 키 없이 로그 그룹이 이미 있는 경우 CloudWatch는 암호화를 업데이트할 수 없습니다. 고객 관리형 KMS 키로 암호화된 소스 로그 그룹에서 고객 관리형 KMS 키와 연결되지 않은 대상 로그 그룹으로 로그 이벤트를 중앙 집중화하지 않으려면이 옵션을 선택합니다.
- AWS 소유 KMS 키가 있는 대상 계정에서 고객 관리형 KMS 키로 암호화된 로그 그룹 중앙 집중화: 고객 관리형 KMS 키로 암호화된 소스 로그 그룹의 로그 이벤트를 AWS 소유 KMS 키를 사용하여 암호화된 새로 생성된 대상 로그 그룹으로 중앙 집중화합니다.
- 고객 관리형 KMS 키로 암호화된 로그 그룹을 중앙 집중화하지 마십시오. 고객 관리형 KMS 키로 암호화된 소스 로그 그룹에서 로그 이벤트의 중앙 집중화를 건너뜁니다.

8. 중앙 집중화 규칙을 검토하고 필요에 따라 마지막 순간에 편집한 다음, 중앙 집중화 정책 생성을 선택합니다.

중앙 집중화 규칙 수정

기존 중앙 집중화 규칙을 수정하려면 다음 절차를 따릅니다.

중앙 집중화 규칙을 수정하려면

1. 조직의 '관리 또는 위임된 관리자' 계정에서 CloudWatch 콘솔로 이동합니다.

2. 설정을 선택합니다.
3. 조직 탭으로 이동합니다.
4. 규칙 관리를 선택합니다.
5. 업데이트할 규칙을 선택하고 편집을 선택합니다.
6. 필요에 따라 규칙 구성을 업데이트하고 다음을 선택하여 각 단계를 진행합니다.
7. 4단계, 검토 및 구성에서 중앙 집중화 정책 업데이트를 선택합니다.

중앙 집중화 규칙 보기

기존 중앙 집중화 규칙의 세부 정보를 보려면 다음 절차를 따릅니다.

중앙 집중화 규칙을 보려면

1. 조직의 '관리 또는 위임된 관리자' 계정에서 CloudWatch 콘솔로 이동합니다.
2. 설정을 선택합니다.
3. 조직 탭으로 이동합니다.
4. 규칙 관리를 선택합니다.
5. 모든 기존 중앙 집중화 규칙의 목록을 확인하고 특정 규칙 이름을 선택하여 세부 정보를 확인합니다.

중앙 집중화 규칙 삭제

다음 절차에 따라 기존 중앙 집중화 규칙을 삭제합니다.

중앙 집중화 규칙을 삭제하려면

1. 조직의 '관리 또는 위임된 관리자' 계정에서 CloudWatch 콘솔로 이동합니다.
2. 설정을 선택합니다.
3. 조직 탭으로 이동합니다.
4. 규칙 관리를 선택합니다.
5. 삭제할 규칙을 선택하고 삭제를 선택합니다.
6. 삭제를 확인하고 삭제를 선택합니다.

중앙 집중화 규칙 모니터링 및 문제 해결

CloudWatch 지표, CloudWatch Logs 콘솔 및 AWS CloudTrail 로그를 사용하여 중앙 집중화 규칙의 상태 및 성능을 모니터링할 수 있습니다. 이렇게 하면 로그 데이터가 성공적으로 복제되고 중앙 집중화 구성과 관련된 문제를 식별할 수 있습니다.

CloudWatch Logs는 다음을 제공합니다.

1. 중앙 집중화 규칙당 규칙 상태
   a. 설정을 선택합니다.
   b. 조직 탭으로 이동합니다.
   c. 규칙 관리를 선택합니다.
2. 를 사용하여 API 호출을 로깅합니다. AWS CloudTrail
3. 또한 CloudWatch는 복제된 로그 이벤트, 오류 및 제한을 포함하여 중앙 집중화에 대한 지표를 게시합니다. 이러한 지표 및 차원에 대한 자세한 내용은 섹션을 참조하세요중앙 집중화 지표 및 차원.

중앙 집중화 규칙 상태

각 중앙 집중화 규칙에는 올바르게 작동하는지 여부를 나타내는 상태가 있습니다. 콘솔 또는 프로그래밍 방식으로 API를 사용하여 규칙 상태를 확인할 수 있습니다.

규칙 상태는 다음과 같습니다.

- `HEALTHY`: 규칙이 정상적으로 작동하고 구성된 대로 로그 데이터를 복제합니다.
- `UNHEALTHY`: 규칙에 문제가 발생하여 데이터를 올바르게 복제하지 못할 수 있습니다.
- `PROVISIONING`: 조직의 중앙 집중화가 설정 중입니다.

규칙이 UNHEALTHY로 표시되면 `FailureReason` 필드는 해결해야 하는 특정 문제에 대한 세부 정보를 제공합니다.

태그 전파 상태

태그 전파에는 로그 전송을 `RuleHealth` 위한 전체와 독립적인 자체 상태 (`TagPropagationStatus`)가 있습니다. 대상 역할을 잘못 구성하면 전체 규칙 상태는 영향을 받지 않습니다. 태그 전파만 비정상으로 표시됩니다.

- `Healthy`: 가장 최근의 태그 전파 시도가 성공했습니다.

- `Unhealthy`: 가장 최근의 태그 전파 시도가 실패했습니다. 자세한 내용은 `TagPropagationFailureReason` 필드를 확인하세요. 문제 해결 단계는 아래 문제 해결 섹션을 참조하세요.

를 사용하여 중앙 집중화 API 호출 모니터링 AWS CloudTrail

AWS CloudTrail 는 중앙 집중화 서비스에 대한 API 호출을 로깅하므로 구성 변경을 추적하고의 멤버인 계정의 문제를 해결할 수 있습니다 AWS Organizations.

중앙 집중화를 위한 주요 CloudTrail 이벤트는 다음과 같습니다.

- `CreateCentralizationRuleForOrganization`: 새 중앙 집중화 규칙이 생성되는 경우
- `UpdateCentralizationRuleForOrganization`: 기존 규칙이 수정되는 경우
- `DeleteCentralizationRuleForOrganization`: 규칙이 삭제되는 경우
- `GetCentralizationRuleForOrganization`: 규칙 세부 정보가 검색되는 경우
- `ListCentralizationRulesForOrganization`: 규칙이 나열되는 경우

CloudTrail 로그를 사용하여 중앙 집중화 구성 변경 사항을 감사하고 성능 문제 또는 복제 실패와 상호 연관시킬 수 있습니다.

모니터링 권장 사항

중앙 집중화가 올바르게 작동하도록 하려면 CloudWatch 지표에 제공하는 주요 중앙 집중화 지표에 대해 CloudWatch 경보를 설정하는 것이 좋습니다. 이 사전 모니터링을 통해 문제를 조기에 감지하고 조직 전체에서 안정적인 로그 중앙 집중화를 유지할 수 있습니다.

모니터링할 주요 지표는 다음과 같습니다.

- `IncomingCopiedBytes`: 대상 계정에 성공적으로 복제되는 로그 데이터의 볼륨을 모니터링합니다. 이 지표의 갑작스러운 감소 또는 부재는 중앙 집중화 문제를 나타낼 수 있습니다.
- `CentralizationError`: 중앙 집중화 프로세스의 오류에 대한 경보를 설정하여 문제를 신속하게 식별하고 해결합니다.
- `CentralizationThrottled`: 로그 복제 성능에 영향을 미칠 수 있는 제한 이벤트를 모니터링합니다.

사용 가능한 중앙 집중화 지표 및 해당 차원의 전체 목록은 섹션을 참조하세요중앙 집중화 지표 및 차원.

로그가 예상대로 중앙 집중화되지 않는 경우 로그 중앙 집중화를 방지할 수 있는 다음과 같은 일반적인 시나리오를 검토합니다.

기록 로그 데이터

CloudWatch Logs 중앙 집중화 기능은 중앙 집중화 규칙을 생성한 후 소스 계정에 도착하는 새 로그 데이터만 처리합니다. 기록 로그 데이터(규칙 생성 전에 존재한 로그)는 중앙 집중화되지 않습니다.

KMS 키 권한

중앙 집중화 규칙에 제공된 KMS 키가 CloudWatch Logs의 사용을 허용하지 않는 경우 중앙 집중화 규칙은 소스 계정에서 대상 로그 그룹으로 로그를 전달하지 못합니다. KMS 키 정책이 CloudWatch Logs에 필요한 권한을 부여하는지 확인합니다. 자세한 내용은 2단계: KMS 키에 대한 권한 설정 단원을 참조하십시오.

고객 관리형 KMS 키 구성

규칙 생성 중에 고객 관리형 KMS 키로 암호화된 로그 그룹을 중앙 집중화하지 않음을 선택한 경우 고객 관리형 KMS 키로 암호화된 소스 로그 그룹의 로그 이벤트는 건너뛰고 중앙 집중화되지 않습니다.

대상 암호화 불일치

대상 로그 그룹이 중앙 집중화 규칙이 지정하는 것과 다른 KMS 암호화 구성으로 이미 존재하고 충돌 해결이 SKIP로 설정된 경우 레코드가 삭제되고 `DestinationEncryptionMismatch` 오류가 발생합니다. 예를 들어 대상에 기본 암호화가 있지만 규칙이 고객 관리형 KMS 키를 지정하는 경우에 발생합니다.

신뢰할 수 있는 액세스가 활성화되지 않음

로그 데이터에 대한 액세스를 제공하려면 관리 계정 및 대상 계정에 대해의 CloudWatch에 AWS Organizations 대해 신뢰할 수 있는 액세스를 활성화해야 합니다.

소스 선택 기준

중앙 집중화 규칙의 소스 선택 기준이 올바르게 구성되었는지 확인합니다.

- 계정 및 리전: 로그가 시작된 소스 계정 및 리전이 규칙에 포함되어 있는지 확인합니다. 규칙에 지정되지 않은 계정 또는 리전의 로그 그룹은 중앙 집중화되지 않습니다.
- 로그 그룹 필터: 로그 그룹 필터를 구성한 경우 지정된 기준과 일치하는 로그 그룹만 중앙 집중화됩니다. 로그 그룹 선택 기준에 중앙 집중화할 것으로 예상되는 로그 그룹이 포함되어 있는지 확인합니다.
- 조직 멤버십: 소스 계정과 대상 계정 모두 동일한 AWS Organizations 조직에 속해야 합니다. 조직 외부의 계정은 중앙 집중화에 참여할 수 없습니다.

로그 그룹 할당량 한도에 도달함

대상 계정이 로그 그룹 할당량 한도에 도달한 경우 중앙 집중화를 위해 새 로그 그룹을 생성할 수 없습니다. 대상 계정에 모든 소스 계정의 중앙 집중식 로그 그룹을 수용할 수 있는 충분한 할당량이 있는지 확인합니다. 필요한 경우 할당량 증가를 요청할 수 있습니다.

로그 스트림 이름 길이 제한을 초과했습니다.

로그 스트림 이름에는 최대 길이 제한이 있습니다. 중앙 집중화가 로그 스트림을 대상 계정에 복제하면 로그 스트림 이름에 접미사가 추가됩니다. 결과 로그 스트림 이름이 최대 허용 길이를 초과하면 레코드가 삭제되고 고객 계정에 `InvalidLogStream` 오류가 발생합니다.

규칙 상태

콘솔에서 또는 `GetCentralizationRuleForOrganization` API를 사용하여 중앙 집중화 규칙 상태를 확인합니다. 규칙이 비정상으로 표시된 경우 `FailureReason` 필드를 검토하여 문제에 대한 구체적인 세부 정보를 확인합니다.

태그 전파 상태

에가 `TagPropagationStatus` 표시되면 `TagPropagationFailureReason` 필드를 `Unhealthy`확인합니다.

- `RoleNotAssumable`: 서비스가 대상 역할을 수임할 수 없습니다. 역할의 신뢰 정책이 중앙 집중화 서비스 연결 역할이 이를 수임하도록 허용하고가 조직 ID와 `sts:ExternalId` 일치하는지 확인합니다.
- `RoleLacksPermissions`: 역할이 수임되었지만 태그 API 호출이 거부되었습니다. 역할의 권한 정책이 대상 로그 그룹에 `logs:TagResource`, 및 `logs:ListTagsForResourcelogs:UntagResource`를 부여하는지 확인합니다.

중앙 집중화 문제를 진단하려면 콘솔에서 중앙 집중화 규칙 상태를 검토하고 CloudWatch 지표에서 오류 및 제한을 확인하고 AWS CloudTrail 로그에서 API 호출 실패를 검사합니다. 중앙 집중화 지표에 대한 자세한 내용은 섹션을 참조하세요중앙 집중화 지표 및 차원.

로그 그룹 및 로그 스트림 작업

로그 스트림은 동일한 소스를 공유하는 로그 이벤트 시퀀스입니다. CloudWatch Logs에서 각 별도의 로그 소스가 별도의 로그 스트림을 구성합니다.

로그 그룹은 동일한 보존 기간, 모니터링 및 액세스 제어 설정을 공유하는 로그 스트림 그룹입니다. 로그 그룹을 정의하고 각 그룹에 배치할 스트림을 지정할 수 있습니다. 하나의 로그 그룹이 가질 수 있는 로그 스트림의 수는 제한이 없습니다.

여러 계정 및 리전의 로그 데이터를 통합해야 하는 조직의 경우 CloudWatch Logs Centralization을 사용하여 로그 그룹을 중앙 계정에 자동으로 복제할 수 있습니다. 자세한 내용은 교차 계정 교차 리전 로그 중앙 집중화 단원을 참조하십시오.

이 섹션에 나오는 절차를 사용하여 로그 그룹 및 로그 스트림 작업을 수행할 수 있습니다.

CloudWatch Logs에서의 로그 그룹 생성

Amazon CloudWatch Logs 사용 설명서의 이전 섹션에 나온 단계를 사용하여 Amazon EC2 인스턴스에 CloudWatch Logs 에이전트를 설치하면 해당 프로세스의 일부로 로그 그룹이 생성됩니다. CloudWatch 콘솔에서 직접 로그 그룹을 생성할 수도 있습니다.

로그 그룹을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 관리를 선택합니다.
3. 작업을 선택한 다음, 로그 그룹 생성을 선택합니다.
4. 로그 그룹의 이름을 입력한 다음, 로그 그룹 생성을 선택합니다.

> **Tip**
>
> 탐색 창의 메뉴에서 대시보드와 경보뿐만 아니라 로그 그룹을 즐겨찾기에 추가할 수 있습니다. 최근 방문 열에서 즐겨찾기에 추가하려는 로그 그룹 위에 마우스 포인터를 놓고 그 옆에 있는 별 기호를 선택합니다.

로그 그룹에 로그 전송

CloudWatch Logs는 여러 AWS 서비스에서 로그 이벤트를 자동으로 수신합니다. 다음 방법 중 하나를 사용하여 CloudWatch Logs로 다른 로그 이벤트를 전송할 수도 있습니다.

- CloudWatch 에이전트— 통합 CloudWatch 에이전트는 지표와 로그를 모두 CloudWatch Logs에 전송할 수 있습니다. CloudWatch 에이전트 설치 및 사용에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버로부터 지표 및 로그 수집을 참조하세요.
- AWS CLI—put-log-events는 CloudWatch Logs로 로그 이벤트 배치를 업로드합니다.
- 프로그래밍 방식으로—PutLogEvents API를 사용하면 프로그래밍 방식으로 로그 이벤트 배치를 CloudWatch Logs로 업로드할 수 있습니다.

CloudWatch Logs로 전송된 로그 데이터 보기

CloudWatch Logs 에이전트가 CloudWatch Logs로 전송한 로그 데이터를 스트림별로 확인하고 스크롤할 수 있습니다. 확인할 로그 데이터에 대해 시간 범위를 지정할 수 있습니다.

로그 데이터를 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 관리를 선택합니다.
3. 로그 그룹에서 스트림을 확인할 로그 그룹을 선택합니다.
4. 로그 그룹 목록에서 보려는 로그 그룹의 이름을 선택합니다.
5. 로그 스트림의 목록에서 보려는 로그 스트림의 이름을 선택합니다.
6. 로그 데이터의 표시 방법을 변경하려면 다음 중 하나를 수행합니다.
   - 단일 로그 이벤트를 확장하려면 해당 로그 이벤트 옆에 있는 화살표를 선택합니다.
   - 로그 이벤트 목록에서 모든 로그 이벤트를 확장하고 이를 일반 텍스트 버전으로 확인하려면 텍스트를 선택합니다.
   - 로그 이벤트를 필터링하려면 검색 필드에 원하는 검색 필터를 입력합니다. 자세한 정보는 필터를 사용하여 로그 이벤트에서 지표 생성 섹션을 참조하세요.
   - 지정된 날짜/시간 범위의 로그 데이터를 확인하려면 검색 필터 옆의 날짜/시간 옆에 있는 화살표를 선택합니다. 날짜/시간 범위를 지정하려면 Absolute(절대)를 선택합니다. 미리 정의된 분, 시

간, 일 또는 주의 횟수를 선택하려면 상대를 선택합니다. UTC와 현지 시간대 간을 전환할 수도 있습니다.

필터 패턴을 사용하여 로그 데이터 검색

지표 필터, 구독 필터, 필터 로그 이벤트 및 Live Tail에 대한 필터 패턴 구문을 사용하여 로그 데이터를 검색할 수 있습니다. 로그 그룹 내의 모든 로그 스트림을 검색하거나를 사용하여 특정 로그 스트림을 검색할 AWS CLI 수도 있습니다. 각각의 검색이 실행되면서 데이터의 다음 페이지를 검색하거나 검색을 계속할 수 있도록 발견된 데이터의 첫 페이지와 토큰이 반환됩니다. 결과가 반환되지 않은 경우에는 계속 검색을 할 수 있습니다.

검색 범위를 제한하기 위해 쿼리하고자 하는 시간 범위를 설정할 수 있습니다. 처음에는 시간 범위를 넓게 잡아서 관심 있는 로그 줄이 어디에 있는지 확인한 다음, 시간 범위를 좁혀서 관심 있는 시간 범위로 로그에 대한 뷰의 범위를 지정할 수 있습니다.

로그에서 추출된 지표에서 해당 로그로 직접 피벗을 적용할 수도 있습니다.

CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 로그인하면, 이 모니터링 계정에 연결된 소스 계정에서 로그 이벤트를 검색하고 필터링할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관찰성을 참조하세요.

콘솔을 사용하여 로그 항목 검색

콘솔을 이용하여 지정된 기준을 충족하는 로그 항목을 검색할 수 있습니다.

콘솔을 이용하여 로그 항목을 검색하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 관리를 선택합니다.
3. 로그 그룹에서 검색할 로그 스트림이 포함된 로그 그룹의 이름을 선택합니다.
4. 로그 스트림에서 검색할 로그 스트림의 이름을 선택합니다.
5. 로그 이벤트에서 사용할 필터 구문을 입력합니다.

콘솔을 이용하여 시간 범위에 대한 모든 로그 항목을 검색하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 관리를 선택합니다.

3. 로그 그룹에서 검색할 로그 스트림이 포함된 로그 그룹의 이름을 선택합니다.
4. 로그 그룹 검색을 선택합니다.
5. 로그 이벤트에서 날짜/시간 범위를 선택하고 필터 구문을 입력합니다.

를 사용하여 로그 항목 검색 AWS CLI

를 사용하여 지정된 기준을 충족하는 로그 항목을 검색할 수 있습니다 AWS CLI.

를 사용하여 로그 항목을 검색하려면 AWS CLI

명령 프롬프트에서 아래 filter-log-events 명령을 실행합니다. `--filter-pattern`을 사용하여 지정된 필터 패턴으로 결과를 제한하고 `--log-stream-names`를 사용하여 지정된 로그 스트림으로 결과를 제한합니다.

```
aws logs filter-log-events --log-group-name my-group [--log-stream-
names LIST_OF_STREAMS_TO_SEARCH] [--filter-pattern VALID_METRIC_FILTER_PATTERN]
```

를 사용하여 지정된 시간 범위에서 로그 항목을 검색하려면 AWS CLI

명령 프롬프트에서 아래 filter-log-events 명령을 실행합니다.

```
aws logs filter-log-events --log-group-name my-group [--log-stream-
names LIST_OF_STREAMS_TO_SEARCH] [--start-time 1482197400000] [--end-
time 1482217558365] [--filter-pattern VALID_METRIC_FILTER_PATTERN]
```

지표에서 로그로 피벗 적용

콘솔의 다른 부분에서 특정 로그 항목으로 이동할 수 있습니다.

대시보드 위젯에서 로그로 이동하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택합니다.
3. 대시보드를 선택합니다.
4. 위젯에서 로그 보기 아이콘을 선택한 다음, 이 시간 범위에서 로그 보기를 선택합니다. 지표 필터가 하나 이상 있는 경우 목록에서 하나를 선택합니다. 지표 필터의 수가 목록에 표시할 수 있는 것보다 많은 경우에는 더 많은 지표 필터를 선택하고 지표 필터를 선택 또는 검색합니다.

지표에서 로그로 이동하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 모든 지표 탭의 검색 필드에 지표 이름을 입력하고 Enter 키를 누릅니다.
4. 검색 결과에서 지표를 하나 이상 선택합니다.
5. 작업, 로그 보기를 선택합니다. 지표 필터가 하나 이상 있는 경우 목록에서 하나를 선택합니다. 지표 필터의 수가 목록에 표시할 수 있는 것보다 많은 경우에는 더 많은 지표 필터를 선택하고 지표 필터를 선택 또는 검색합니다.

문제 해결

검색 완료에 너무 많은 시간이 소요

로그 데이터가 많은 경우에는 검색을 완료하는 데 긴 시간이 소요될 수 있습니다. 다음과 같은 방법으로 검색 속도를 높일 수 있습니다.

- 를 사용하는 경우 관심 있는 로그 스트림으로만 검색을 제한할 AWS CLI수 있습니다. 예를 들어 로그 그룹에 1,000개의 로그 스트림이 있지만 관련성이 있는 것으로 알려진 3개의 로그 스트림만 보려는 경우 AWS CLI 를 사용하여 로그 그룹 내의 3개의 로그 스트림으로만 검색을 제한할 수 있습니다.
- 더 짧고 세분화된 시간 범위를 사용하면 검색할 데이터양을 줄이고 쿼리 속도를 높일 수 있습니다.

CloudWatch Logs에서 로그 데이터 보존 기간 변경

기본적으로 CloudWatch Logs에서 로그 데이터는 무기한으로 저장됩니다. 그러나 로그 그룹에서 로그 데이터를 저장할 기간을 구성할 수 있습니다. 현재 보존 기간 설정보다 오래된 데이터는 모두 삭제됩니다. 언제든지 로그 그룹별로 로그 보존 기간을 변경할 수 있습니다.

> **Note**
>
> CloudWatch Logs 는 보존 설정에 도달하면 로그 이벤트를 즉시 삭제하지 않습니다. 일반적으로 로그 이벤트가 삭제되기까지 최대 72시간이 걸리지만 드물게 그보다 더 오래 걸릴 수 있습니다.
>
> 즉, 만료 날짜가 지났지만 실제로 삭제되지 않은 로그 이벤트를 포함하는 경우 더 긴 보존 설정을 갖도록 로그 그룹을 변경하면 새 보존 날짜에 도달한 후 해당 로그 이벤트가 삭제되는 데 최대 72시간이 걸립니다. 로그 데이터를 영구적으로 삭제하려면 이전 보존 기간이 종료된 후 72

시간이 경과하거나 이전 로그 이벤트가 삭제되었음을 확인할 때까지 로그 그룹을 낮은 보존 설정으로 유지합니다.
로그 이벤트가 보존 설정에 도달하면 삭제 대상으로 표시됩니다. 삭제 대상으로 표시된 후에는 이후에 실제로 삭제되지 않더라도 더 이상 아카이브 스토리지 비용에 추가되지 않습니다. 삭제 대상으로 표시된 로그 이벤트는 API를 통해 `storedBytes` 값을 검색하여 로그 그룹에 저장 중인 바이트 수를 확인할 때도 포함되지 않습니다.

로그 보존 기간 설정을 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 업데이트할 로그 그룹을 찾습니다.
4. 해당 로그 그룹의 보존 열에서 만료되지 않음 같은 현재 보존 설정을 선택합니다.
5. 보존 설정의 이벤트 만료 시점에서 로그 보존 값을 선택한 다음 저장을 선택합니다.

로그 그룹이 삭제되지 않도록 보호

선택적으로 삭제 방지를 활성화하여 중요한 로그 그룹이 실수로 삭제되는 것을 방지할 수 있습니다. 삭제 방지에 대한 자세한 내용은 섹션을 참조하세요로그 그룹이 삭제되지 않도록 보호.

로그 그룹이 삭제되지 않도록 보호

삭제 방지 활성화

새 로그 그룹을 생성할 때 또는 기존 로그 그룹에서 삭제 방지를 활성화할 수 있습니다. 로그 그룹을 생성하는 동안 "삭제 방지 활성화"를 선택하거나 파라미터를 전달하여를 선택합니다`--deletion-protection-enabled`. 기본적으로 삭제 방지는 활성화되어 있지 않습니다.

기존 로그 그룹에서 삭제 방지를 활성화하거나 비활성화하려면(콘솔)

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 관리를 선택합니다.
3. 보호하려는 로그 그룹을 선택합니다.
4. 작업, 삭제 방지 편집을 선택합니다.
5. 대화 상자에서 변경 사항을 검토한 다음 제출합니다.

를 사용하는 경우 기존 로그 그룹에서 삭제 방지를 활성화 AWS CLI하려면:

```
aws logs put-log-group-deletion-protection \
--log-group-identifier "/my-application/logs" \
--deletion-protection-enabled
```

기존 로그 그룹에서 삭제 방지를 제거하려면:

```
aws logs put-log-group-deletion-protection \
--log-group-identifier "/my-application/logs" \
--no-deletion-protection-enabled
```

오류 처리

삭제 방지 기능이 활성화된 로그 그룹을 삭제하려고 하면 `ValidationException` "삭제 방지 기능이 활성화된 로그 그룹은 삭제할 수 없습니다. 먼저 삭제 방지 기능을 비활성화합니다."

Amazon CloudWatch Logs의 로그 그룹에 태그 지정

Amazon CloudWatch Logs에서 생성하는 로그 그룹에 고유한 메타데이터를 태그 형식으로 할당할 수 있습니다. 태그는 로그 그룹에 대해 정의된 키-값 쌍입니다. 태그를 사용하는 것은 AWS 리소스를 관리하고 결제 데이터를 포함한 데이터를 구성하는 간단하지만 강력한 방법입니다.

> **Note**
>
> 태그를 사용하여 로그 그룹 및 대상을 비롯한 CloudWatch Logs 리소스에 대한 액세스를 제어할 수 있습니다. 로그 그룹과 로그 스트림 간의 계층적 관계로 인해 로그 스트림에 대한 액세스는 로그 그룹 수준에서 제어됩니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 Amazon Web Services 리소스에 대한 액세스 제어를 참조하세요.

내용

- 태그 기본 사항
- 태그 지정을 사용하여 비용 추적
- 태그 제한 사항
- 를 사용하여 로그 그룹에 태그 지정 AWS CLI
- CloudWatch Logs API를 사용하여 로그 그룹에 태그 지정

태그 기본 사항

AWS CloudFormation AWS CLI또는 CloudWatch Logs API를 사용하여 다음 작업을 완료합니다.

- 생성 시 로그 그룹에 태그를 추가합니다.
- 기존 로그 그룹에 태그를 추가합니다.
- 로그 그룹에 대한 태그를 나열합니다.
- 로그 그룹에서 태그를 제거합니다.

태그를 사용하여 로그 그룹을 분류할 수 있습니다. 예를 들어, 용도, 소유자 또는 환경을 기준별로 로그 그룹을 분류할 수 있습니다. 각 태그에 대해 키 및 값이 정의되기 때문에 특정 요구 사항을 충족하는 사용자 지정 카테고리 집합을 생성할 수 있습니다. 예를 들어, 태그 집합을 정의하여 소유자와 연관 애플리케이션별 로그 그룹을 추적할 수 있습니다. 다음은 태그의 몇 가지 예제입니다.

- 프로젝트: 프로젝트 이름
- 소유자: 이름
- 용도: 로드 테스트
- 애플리케이션: 애플리케이션 이름
- 환경: 프로덕션

태그 지정을 사용하여 비용 추적

태그를 사용하여 AWS 비용을 분류하고 추적할 수 있습니다. 로그 그룹을 포함하여 AWS 리소스에 태그를 적용하면 AWS 비용 할당 보고서에는 태그별로 집계된 사용량 및 비용이 포함됩니다. 비즈니스 카테고리를 나타내는 태그(예: 비용 센터, 애플리케이션 이름 또는 소유자)를 적용하여 여러 서비스에 대한 비용을 정리할 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 사용자 지정 결제 보고서에 비용 할당 태그 사용을 참조하세요.

태그 제한 사항

태그에 적용되는 제한은 다음과 같습니다.

기본 제한 사항

- 로그 그룹당 최대 태그의 수는 50개입니다.

- 태그 키와 값은 대소문자를 구분합니다.
- 삭제된 로그 그룹에 대해 태그를 변경하거나 편집할 수 없습니다.

태그 키 제한 사항

- 각 태그 키는 고유해야 합니다. 이미 사용 중인 키를 가진 태그를 추가하면 기존 키-값 쌍에 새 태그가 덮어쓰기 됩니다.
- 이 접두사는에서 사용하도록 예약되어 `aws:` 있으므로 로 태그 키를 시작할 수 없습니다 AWS.는 사용자를 대신하여이 접두사로 시작하는 태그를 AWS 생성하지만 편집하거나 삭제할 수는 없습니다.
- 태그 키의 길이는 유니코드 1~128자여야 합니다.
- 태그 키의 문자로는 유니코드 문자, 숫자, 공백 그리고 `_ . / = + - @` 같은 특수 문자가 허용됩니다.

태그 값 제한 사항

- 태그 값의 길이는 유니코드 0~255자여야 합니다.
- 태그 값은 공백 상태로 둘 수 있습니다. 아니면 유니코드 문자, 숫자, 공백 그리고 `_ . / = + - @` 같은 특수 문자를 사용할 수 있습니다.

를 사용하여 로그 그룹에 태그 지정 AWS CLI

AWS CLI를 사용하여 태그를 추가, 나열 및 제거할 수 있습니다. 예제는 다음 설명서를 참조하세요.

[create-log-group]

로그 그룹을 생성합니다. 로그 그룹 생성 시 태그를 선택에 따라 추가할 수 있습니다.

[tag-resource]

지정된 CloudWatch Logs 리소스에 하나 이상의 태그(키-값 쌍)를 할당합니다.

[list-tags-for-resource]

CloudWatch Logs 리소스와 연관된 태그를 표시합니다.

[untag-resource]

지정된 CloudWatch Logs 리소스에서 하나 이상의 태그를 제거합니다.

CloudWatch Logs API를 사용하여 로그 그룹에 태그 지정

CloudWatch Logs API를 사용하여 태그를 추가, 나열 및 제거할 수 있습니다. 예제는 다음 설명서를 참조하세요.

CreateLogGroup

로그 그룹을 생성합니다. 로그 그룹 생성 시 태그를 선택에 따라 추가할 수 있습니다.

TagResource

지정된 CloudWatch Logs 리소스에 하나 이상의 태그(키-값 쌍)를 할당합니다.

ListTagsForResource

CloudWatch Logs 리소스와 연관된 태그를 표시합니다.

UntagResource

지정된 CloudWatch Logs 리소스에서 하나 이상의 태그를 제거합니다.

를 사용하여 CloudWatch Logs의 로그 데이터 암호화 AWS Key Management Service

로그 그룹 데이터는 항상 CloudWatch Logs에서 암호화됩니다. 기본적으로 CloudWatch Logs는 256비트 고급 암호화 표준 갈루아/카운터 모드(AES-GCM)를 사용하는 서버 측 암호화로 서버 측 암호화를 사용하여 저장 로그 데이터를 암호화합니다. 대신 이 암호화를 위해 AWS Key Management Service을 사용할 수 있습니다. 이렇게 하면 AWS KMS 키를 사용하여 암호화가 완료됩니다. 를 사용한 암호화 AWS KMS 는 로그 그룹을 생성할 때 또는 로그 그룹이 존재한 후 KMS 키를 로그 그룹과 연결하여 로그 그룹 수준에서 활성화됩니다.

> **Important**
>
> CloudWatch Logs에서는 이제 `kms:EncryptionContext:aws:logs:arn`을 키로 사용하고 로그 그룹의 ARN을 해당 키의 값으로 사용하여 암호화 컨텍스트를 지원합니다. KMS 키로 이미 암호화된 로그 그룹이 있고 단일 계정 및 로그 그룹에서 키를 사용하도록 제한하려면 IAM 정책에 조건을 포함하는 새 KMS 키를 할당해야 합니다. 자세한 내용은 AWS KMS 키 및 암호화 컨텍스트 단원을 참조하십시오.

> **Important**
>
> CloudWatch Logs는 이제 로그`kms:ViaService`가 사용자를 대신하여 AWS KMS 호출할 수 있도록 허용하는를 지원합니다. 키 정책 또는 IAM에서 CloudWatch Logs를 직접 호출하는 역할에 이를 추가해야 합니다. 자세한 내용은 kms:ViaService를 참조하세요.

KMS 키를 로그 그룹에 연결한 후, 로그 그룹에 대해 새로 수집된 모든 데이터는 이 키를 사용하여 암호화됩니다. 이 데이터는 보존되는 동안 암호화된 형식으로 저장됩니다. CloudWatch Logs는 요청이 있을 때마다 이 데이터를 암호화 해제합니다. CloudWatch Logs에는 암호화된 데이터 요청이 발생할 때마다 KMS 키에 대한 권한이 있어야 합니다.

로그 그룹에서 KMS 키가 연결 해제되는 경우 CloudWatch Logs에서는 CloudWatch Logs의 기본 암호화 방법을 사용하여 새로 수집된 데이터를 암호화합니다. KMS 키로 암호화된 이전에 수집된 모든 데이터는 KMS 키로 암호화된 상태가 유지됩니다. CloudWatch Logs는 계속해서 키를 참조할 수 있으므로 CloudWatch Logs는 KMS 키의 연결이 끊어진 후에도 여전히 해당 데이터를 반환할 수 있습니다. 그러나 나중에 키를 비활성화하면 CloudWatch Logs는 해당 키로 암호화된 로그를 읽을 수 없습니다.

> **Important**
>
> CloudWatch Logs는 대칭 KMS 키만 지원합니다. 비대칭 키를 사용하여 로그 그룹의 데이터를 암호화하지 마세요. 자세한 내용은 대칭 및 비대칭 키 사용을 참조하세요.

제한 사항

- 다음 단계를 수행하려면 `kms:CreateKey`, `kms:GetKeyPolicy` 및 `kms:PutKeyPolicy` 권한이 있어야 합니다.
- 로그 그룹에서 키를 연결하거나 연결 해제하고 난 후 이러한 변경이 적용되기까지 최대 5분의 시간이 소요될 수 있습니다.
- 연결된 키에 대한 CloudWatch Logs 액세스 권한을 취소하거나 연결된 KMS 키를 삭제한 경우에는 CloudWatch Logs에서 암호화된 데이터를 더 이상 검색할 수 없습니다.
- CloudWatch 콘솔을 사용하여 KMS 키를 기존 로그 그룹과 연결할 수 없습니다.

1단계: AWS KMS 키 생성

다음 create-key 명령을 사용하여 KMS 키 생성:

```
aws kms create-key
```

이 명령의 출력 화면에는 키의 키 ID와 Amazon Resource Name(ARN)이 포함됩니다. 다음은 예제 출력입니다.

```
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
        "AWSAccountId": "123456789012",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

2단계: KMS 키에 대한 권한 설정

기본적으로 모든 AWS KMS 키는 프라이빗입니다. 리소스 소유자만 이를 사용하여 데이터를 암호화 및 해독할 수 있습니다. 그러나 리소스 소유자가 원한다면 다른 사용자 및 리소스에 KMS 키에 대한 액세스 권한을 부여할 수 있습니다. 이 단계에서는 CloudWatch Logs 서비스 보안 주체와 직접 호출자 역할에게 키를 사용할 수 있는 권한을 부여합니다. 이 서비스 보안 주체는 KMS 키가 저장된 리전과 동일한 AWS 리전에 있어야 합니다.

가장 좋은 방법은 KMS 키 사용을 지정한 AWS 계정 또는 로그 그룹으로만 제한하는 것입니다.

먼저 다음 get-key-policy 명령을 사용하는 `policy.json`로 KMS 키에 대한 기본 정책을 저장합니다.

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./
policy.json
```

텍스트 편집기에서 policy.json 파일을 열고 다음 명령문 중 하나에서 굵은 글꼴로 표시된 섹션을 추가합니다. 기존 명령문과 새 명령문을 쉼표로 구분합니다. 이러한 문은 Condition 섹션을 사용하여 AWS KMS 키의 보안을 강화합니다. 자세한 내용은 AWS KMS 키 및 암호화 컨텍스트 단원을 참조하십시오.

이 예제의 Condition 섹션에서는 키를 단일 로그 그룹 ARN으로 제한합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:111122223333:log-group:log-group-name"
                }
            }
        }
```

```
    ]
}
```

이 예제의 Condition 섹션에서는 AWS KMS 키 사용을 지정된 계정으로 제한하지만 모든 로그 그룹에 사용할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:123456789012:*"
                }
            }
        }
```

```
    ]
}
```

3단계: 호출 IAM 보안 주체에 대한 권한 설정

CloudWatch Logs APIs를 호출할 IAM 보안 주체(사용자 또는 역할)에 권한을 추가합니다. 필요한 권한은 보안 주체가 수행해야 하는 작업에 따라 달라집니다. AWS KMS 키 정책 또는 역할 자체의 IAM을 통해 이러한 권한을 추가할 수 있습니다. CloudWatch Logs는 `kms:ViaService`를 사용하여 고객을 대신하여 AWS KMS 에 전화를 겁니다. 자세한 내용은 kms:ViaService를 참조하세요.

KMS 키를 로그 그룹과 연결하기 위한 권한

`kmsKeyId` 파라미터를 `CreateLogGroup` 사용하여를 호출하거나를 호출하는 IAM 보안 주체는 지정된 KMS 키에 대한 `kms:DescribeKey` 권한이 `AssociateKmsKey`있어야 합니다. 호출자에게이 권한이 없는 경우 API 호출은와 함께 실패합니다`AccessDeniedException`.

다음 예제 키 정책 문은 KMS 키를 로그 그룹과 연결하는 데 필요한 최소 권한을 부여합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}
```

암호화된 로그 데이터를 읽고 쓸 수 있는 권한

IAM 보안 주체가 암호화된 로그 데이터도 읽거나 써야 하는 경우(예: 고객 관리형 KMS 키로 암호화된 `StartQuery` 로그 그룹에서 `PutLogEvents`, `GetLogEvents`, `FilterLogEvents`, 또는 호출) 보안

주체에게 추가 AWS KMS 권한이 필요합니다. 다음 예제 키 정책 문은 키를 연결하고 암호화된 로그 데이터를 읽거나 쓰는 데 필요한 전체 권한 집합을 부여합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Encrypt",
    "kms:ReEncrypt*",
    "kms:Decrypt",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}
```

가장 좋은 방법은 AWS KMS 암호화된 로그 그룹과 상호 작용할 역할로만 정책의 범위를 지정하는 것입니다.

또는 키를 연결하고 IAM을 통해 암호화된 로그 데이터를 읽거나 쓸 수 있는 전체 권한 집합을 부여하려면 호출자 역할에 다음 정책을 추가할 수 있습니다. 이는 기존 역할 정책에 추가하거나 역할에 별도의 추가 정책으로 연결할 수 있습니다. 이 방법을 사용하는 경우 모범 사례로 정책의 범위를 로그 암호화에 사용할 AWS KMS 키로만 지정합니다. 자세한 내용은 IAM 정책 편집을 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "kms:Encrypt",
                "kms:ReEncrypt*",
                "kms:Decrypt",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "logs.us-east-1.amazonaws.com"
                    ]
                }
            },
            "Resource": "arn:aws:kms:us-east-1:444455556666:key/key_id"
        }
    ]
}
```

마지막으로 아래 put-key-policy 명령을 사용하여 업데이트된 정책을 추가합니다.

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

4단계: KMS 키를 로그 그룹에 연결

로그 그룹을 생성할 때와 그 이후에 KMS 키를 로그 그룹에 연결할 수 있습니다.

로그 그룹에 이미 KMS 키가 연결되어 있는지 확인하려면 다음 describe-log-groups 명령을 사용합니다.

```
aws logs describe-log-groups --log-group-name-prefix "log-group-name-prefix"
```

출력에 kmsKeyId 필드가 포함된 경우 로그 그룹은 해당 필드의 값에 대해 표시된 키와 연결됩니다.

생성 시 로그 그룹에 KMS 키를 연결

다음과 같이 create-log-group 명령을 사용합니다.

```
aws logs create-log-group --log-group-name my-log-group --kms-key-id "key-arn"
```

기존 로그 그룹에 KMS 키를 연결

다음과 같이 associate-kms-key 명령을 사용합니다.

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id "key-arn"
```

5단계: 로그 그룹에서 키 연결 해제

로그 그룹에 연결된 KMS 키의 연결을 해제하려면 아래 disassociate-kms-key 명령을 사용합니다.

```
aws logs disassociate-kms-key --log-group-name my-log-group
```

AWS KMS 키 및 암호화 컨텍스트

AWS Key Management Service 키와 암호화된 로그 그룹의 보안을 강화하기 위해 CloudWatch Logs는 이제 로그 데이터를 암호화하는 데 사용되는 암호화 컨텍스트의 일부로 로그 그룹 ARNs을 배치합니다. 암호화 컨텍스트는 추가 인증 데이터로 사용되는 키-값 쌍의 집합입니다. 암호화 컨텍스트를 사용하면 IAM 정책 조건을 사용하여 계정 및 로그 그룹별로 AWS KMS AWS 키에 대한 액세스를 제한할 수 있습니다. 자세한 내용은 암호화 컨텍스트 및 IAM JSON 정책 요소: 조건을 참조하세요.

암호화된 각 로그 그룹에 대해 서로 다른 KMS 키를 사용하는 것이 좋습니다.

이전에 암호화한 로그 그룹이 있고 해당 로그 그룹에만 작동하는 새 KMS 키를 사용하도록 로그 그룹을 변경하려는 경우 다음 단계를 수행합니다.

KMS 키를 해당 로그 그룹으로 제한하는 정책에 따라 사용하도록 암호화된 로그 그룹 변환

1. 다음 명령을 입력하여 로그 그룹의 현재 키의 ARN을 찾습니다.

   ```
   aws logs describe-log-groups
   ```

   출력에는 다음 줄이 포함됩니다. ARN을 기록해 둡니다. 7단계에서 사용해야 합니다.

   ```
   ...
   "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/01234567-89ab-
   cdef-0123-456789abcdef"
   ...
   ```

2. 다음 명령을 입력하여 새 KMS 키 생성:

   ```
   aws kms create-key
   ```

3. 다음 명령을 입력하여 새 키의 정책을 policy.json 파일에 저장합니다.

```
aws kms get-key-policy --key-id new-key-id --policy-name default --output text > ./
policy.json
```

4. 텍스트 편집기를 사용하여 policy.json을 열고 Condition 표현식을 정책에 추가합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:111122223333:log-group:LOG-GROUP-NAME"
                }
            }
        }
```

```
    ]
}
```

5. 다음 명령을 입력하여 업데이트된 정책을 새 KMS 키에 추가:

```
aws kms put-key-policy --key-id new-key-ARN --policy-name default --policy file://
policy.json
```

6. 다음 명령을 입력하여 정책을 로그 그룹과 연결합니다.

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id new-key-ARN
```

CloudWatch Logs는 이제 새 키를 사용하여 모든 새 데이터를 암호화합니다.

7. 그런 다음 이전 키에서 Decrypt를 제외한 모든 권한을 취소합니다. 먼저 다음 명령을 입력하여 이전 정책을 검색합니다.

```
aws kms get-key-policy --key-id old-key-ARN --policy-name default --output text
 > ./policy.json
```

8. 텍스트 편집기를 사용하여 policy.json을 열고 Action 목록에서 kms:Decrypt를 제외한 모든 값을 제거합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.region.amazonaws.com"
```

```
            },
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

9. 다음 명령을 입력하여 업데이트된 정책을 이전 키에 추가합니다.

```
aws kms put-key-policy --key-id old-key-ARN --policy-name default --policy file://
policy.json
```

마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원

로그 그룹 데이터 보호 정책을 사용하여 CloudWatch Logs에서 모든 민감한 데이터를 보호할 수 있습니다. 이러한 정책을 사용하면 계정의 로그 그룹에서 모은 로그 이벤트에 나타나는 민감한 데이터를 감사하고 마스킹할 수 있습니다.

데이터 보호 정책을 생성할 때 선택한 데이터 식별자와 일치하는 민감한 데이터가 기본적으로 CloudWatch Logs Insights, 지표 필터 및 구독 필터를 포함한 모든 송신 지점에서 마스킹됩니다. `logs:Unmask` IAM 권한이 있는 사용자만 마스킹 처리되지 않은 데이터를 볼 수 있습니다.

계정의 모든 로그 그룹에 대한 데이터 보호 정책을 생성할 수 있으며 개별 로그 그룹에 대한 데이터 보호 정책을 생성할 수도 있습니다. 전체 계정에 대한 정책을 생성하면 기존 로그 그룹과 향후 생성되는 로그 그룹 모두에 적용됩니다.

전체 계정에 대한 데이터 보호 정책을 생성하고 단일 로그 그룹에 대한 정책도 생성하는 경우 두 정책 모두 해당 로그 그룹에 적용됩니다. 두 정책 중 하나에 지정된 모든 관리형 데이터 식별자는 해당 로그 그룹에서 감사 및 마스킹 처리됩니다.

> **Note**
>
> 민감한 데이터 마스킹은 Standard 및 Infrequent Access 로그 클래스의 로그 그룹에 대해 지원됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스을 참조하세요.

각 로그 그룹에는 로그 그룹 수준의 데이터 보호 정책이 하나만 있을 수 있지만 해당 정책은 감사하고 마스킹할 여러 관리형 데이터 식별자를 지정할 수 있습니다. 데이터 보호 정책의 한도는 30,720자입니다.

> **Important**
>
> 민감한 데이터는 로그 그룹에 수집될 때 감지 및 마스킹 처리됩니다. 데이터 보호 정책을 설정하면 해당 시간 이전에 로그 그룹에 수집된 로그 이벤트는 마스킹 처리되지 않습니다.

CloudWatch Logs가 지원하는 여러 관리형 데이터 식별자는 금융 데이터, 개인 건강 정보(PHI), 개인 식별 정보(PII) 보호를 위해 선택할 수 있는 사전 구성된 데이터 유형을 제공합니다. CloudWatch Logs 데이터 보호를 통해 패턴 매칭 및 기계 학습 모델을 활용하여 중요한 데이터를 탐지할 수 있습니다. 일부 관리형 데이터 식별자 유형의 경우 민감한 데이터에 근접한 특정 키워드의 검색 결과에 따라 탐지가 달라집니다. 또한 사용자 지정 데이터 식별자를 사용하여 특정 사용 사례에 맞는 데이터 식별자를 생성할 수 있습니다.

선택한 데이터 식별자와 일치하는 중요한 데이터가 감지되면 지표를 CloudWatch로 내보냅니다. 이는 LogEventsWithFindings 지표이며 AWS/Logs 네임스페이스로 내보내집니다. 해당 지표를 사용하여 CloudWatch 경보를 생성하고 그래프와 대시보드로 시각화할 수 있습니다. 데이터 보호에서 내보낸 지표는 판매된 지표이며 무료입니다. CloudWatch Logs가 CloudWatch로 보내는 지표에 대한 자세한 내용은 CloudWatch 지표를 사용한 모니터링을 참조하세요.

각 관리형 데이터 식별자는 신용카드 번호, AWS 비밀 액세스 키 또는 특정 국가 또는 지역의 여권 번호와 같은 특정 유형의 민감한 데이터를 감지하도록 설계되었습니다. 데이터 보호 정책을 생성할 때 이러한 식별자를 사용하여 로그 그룹에서 수집한 로그를 분석하고, 감지되면 조치를 취하도록 구성할 수 있습니다.

CloudWatch Logs 데이터 보호는 관리형 데이터 식별자를 사용하여 다음과 같은 민감한 데이터 카테고리를 감지할 수 있습니다.

- 프라이빗 키 또는 AWS 보안 액세스 키와 같은 자격 증명
- 금융 정보(예: 신용 카드 번호)
- 개인 식별 정보(PII)(예: 운전면허증 또는 사회 보장 번호)
- 보호 대상 건강 정보(PHI)(예: 건강 보험 또는 의료 식별 번호)
- 디바이스 식별자(예: IP 주소 또는 MAC 주소)

보호할 수 있는 데이터 유형에 대한 자세한 내용은 보호할 수 있는 데이터 유형을 참조하세요.

목차

- 데이터 보호 정책 이해
  - 데이터 보호 정책이란 무엇입니까?
  - 데이터 보호 정책은 어떻게 구성되어 있습니까?
    - 데이터 보호 정책의 JSON 속성
    - 정책 명령문을 위한 JSON 속성
    - 정책 명령문 작업을 위한 JSON 속성
- 데이터 보호 정책을 생성하거나 사용하는 데 필요한 IAM 권한
  - 계정 수준 데이터 보호 정책에 필요한 권한
  - 단일 로그 그룹에 대한 데이터 보호 정책에 필요한 권한
  - 샘플 데이터 보호 정책
- 계정 전체 데이터 보호 정책 생성
  - 콘솔
  - AWS CLI
    - AWS CLI 또는 API 작업에 대한 데이터 보호 정책 구문
- 단일 로그 그룹에 대한 데이터 보호 정책 생성
  - 콘솔
  - AWS CLI
    - AWS CLI 또는 API 작업에 대한 데이터 보호 정책 구문
- 마스킹 처리되지 않은 데이터 보기
- 감사 결과 보고서
  - 로 보호되는 버킷으로 감사 결과를 보내는 데 필요한 키 정책 AWS KMS
- 보호할 수 있는 데이터 유형
  - 민감한 데이터 유형에 대한 CloudWatch Logs 관리형 데이터 식별자
    - 자격 증명
      - 자격 증명 데이터 유형용 데이터 식별자 ARN
    - 디바이스 식별자
      - 디바이스 데이터 유형을 위한 데이터 식별자 ARN
    - 금융 정보
      - 금융 데이터 유형용 데이터 식별자 ARN

    - 보호 대상 건강 정보(PHI)
        - 보호 대상 건강 정보(PHI) 데이터 유형에 대한 데이터 식별자 ARN
    - 개인 식별 정보(PII)
        - 운전면허증 식별 번호 키워드
        - 국가 식별 번호 키워드
        - 여권 번호 키워드
        - 납세자 식별 및 참조 번호 키워드
        - 개인 식별 정보(PII)에 대한 데이터 식별자 ARN
- 사용자 지정 데이터 식별자
    - 사용자 지정 데이터 식별자란?
    - 사용자 지정 데이터 식별자 제약
    - 콘솔에서 사용자 지정 데이터 식별자 사용
    - 데이터 보호 정책에서 사용자 지정 데이터 식별자 사용

데이터 보호 정책 이해

주제

- 데이터 보호 정책이란 무엇입니까?
- 데이터 보호 정책은 어떻게 구성되어 있습니까?

데이터 보호 정책이란 무엇입니까?

CloudWatch Logs는 데이터 보호 정책을 사용하여 스캔하려는 민감한 데이터와 해당 데이터를 보호하기 위해 수행하려는 작업을 선택합니다. 관심 있는 민감한 데이터를 선택하려면 데이터 식별자를 사용합니다. 그런 다음 CloudWatch Logs 데이터 보호는 기계 학습 및 패턴 일치를 사용하여 중요한 데이터를 감지합니다. 발견된 데이터 식별자에 따라 조치를 취하기 위해 감사와 비식별화 작업을 정의할 수 있습니다. 이러한 연산을 사용하여 발견된(또는 발견되지 않은) 민감한 데이터를 로깅하고 로그 이벤트를 확인할 때 민감한 데이터를 마스킹 처리할 수 있습니다.

데이터 보호 정책은 어떻게 구성되어 있습니까?

다음 그림처럼 데이터 보호 정책 문서는 다음 요소를 포함합니다.

- 문서 상단에 위치하는 정책 전반의 선택적 정보

- 감사 및 비식별화 작업을 정의하는 명령문 하나

CloudWatch Logs 로그 그룹당 하나의 데이터 보호 정책만 정의할 수 있습니다. 데이터 보호 정책은 하나 이상의 거부 또는 비식별 명령문과 하나의 감사 명령문을 가질 수 있습니다.

데이터 보호 정책의 JSON 속성

데이터 보호 정책에는 식별을 위해 다음과 같은 기본 정책 정보가 필요합니다.

- Name(이름) – 정책 이름입니다.
- Description(설명)(선택 사항) – 정책 설명입니다.
- Version(버전) - 정책 언어 버전입니다. 현재 버전은 2021-06-01입니다.
- Statement(명령문) - 데이터 보호 정책 조치를 지정하는 명령문 목록입니다.

```
{
  "Name": "CloudWatchLogs-PersonalInformation-Protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
        ...
  ]
}
```

정책 명령문을 위한 JSON 속성

정책 명령문은 데이터 보호 작업에 대한 탐지 컨텍스트를 설정합니다.

- Sid(선택 사항) - 명령문 식별자입니다.
- DataIdentifier – CloudWatch Logs에서 스캔해야 하는 민감한 데이터입니다. 이러한 데이터로는 이름, 주소 또는 전화번호가 있습니다.
- Operation - 후속 작업입니다. Audit 또는 De-identify 중 하나입니다. CloudWatch Logs는 민감한 데이터를 발견하면 해당 작업을 수행합니다.

```
{
  ...
  "Statement": [
    {
```

```
      "Sid": "audit-policy",
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Address"
      ],
      "Operation": {
        "Audit": {
          "FindingsDestination": {}
        }
      }
    },
```

정책 명령문 작업을 위한 JSON 속성

정책 명령문은 다음 데이터 보호 작업 중 하나를 설정합니다.

- Audit - 로깅을 중단하지 않고 지표 및 결과 보고서를 내보냅니다. 일치하는 문자열은 CloudWatch Logs가 CloudWatch의 AWS/Logs 네임스페이스에 게시하는 LogEventsWithFindings 지표를 증가시킵니다. 해당 지표를 사용하여 경보를 생성할 수 있습니다.

  결과 보고서의 예는 감사 결과 보고서를 참조하세요.

  CloudWatch Logs가 CloudWatch로 보내는 지표에 대한 자세한 내용은 CloudWatch 지표를 사용한 모니터링을 참조하세요.
- De-identify -로깅을 중단하지 않고 민감한 데이터를 마스킹 처리합니다.

데이터 보호 정책을 생성하거나 사용하는 데 필요한 IAM 권한

로그 그룹에 대한 데이터 보호 정책을 사용하려면 다음 표에 나와 있는 특정 권한이 있어야 합니다. 권한은 계정 전체 데이터 보호 정책과 단일 로그 그룹에 적용되는 데이터 보호 정책에 따라 다릅니다.

계정 수준 데이터 보호 정책에 필요한 권한

> **Note**
>
> Lambda 함수 내에서 이러한 작업을 수행하는 경우 Lambda 실행 역할 및 권한 경계에 다음 권한도 포함되어야 합니다.

연산	IAM 권한 필요	Resource
감사 대상이 없는 데이터 보호 정책 생성	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
CloudWatch Logs를 감사 대상으로 사용하여 데이터 보호 정책 생성	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
	`logs:CreateLogDelivery`	*
	`logs:PutResourcePolicy`	*
	`logs:DescribeResourcePolicies`	*
	`logs:DescribeLogGroups`	*
Firehose를 감사 대상으로 사용하여 데이터 보호 정책 생성	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
	`logs:CreateLogDelivery`	*
	`firehose:TagDeliveryStream`	`arn:aws:logs:::deliverystream/` *YOUR_DELIVERY_STREAM*

연산	IAM 권한 필요	Resource
Amazon S3를 감사 대상으로 사용하여 데이터 보호 정책 생성	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
	`logs:CreateLogDelivery`	*
	`s3:GetBucketPolicy`	`arn:aws:s3::: YOUR_BUCKET`
	`s3:PutBucketPolicy`	`arn:aws:s3::: YOUR_BUCKET`
지정된 로그 그룹에서 마스킹 처리된 로그 이벤트 마스킹 처리 해제	`logs:Unmask`	`arn:aws:logs:::log-group:*`
기존 데이터 보호 정책 보기	`logs:GetDataProtectionPolicy`	*
데이터 보호 정책 삭제	`logs:DeleteAccountPolicy`	*
	`logs:DeleteDataProtectionPolicy`	*

데이터 보호 감사 로그가 이미 대상으로 전송되고 있는 경우 동일한 대상으로 로그를 전송하는 다른 정책에는 `logs:PutDataProtectionPolicy` 및 `logs:CreateLogDelivery` 권한만 있으면 됩니다.

단일 로그 그룹에 대한 데이터 보호 정책에 필요한 권한

> **Note**
>
> Lambda 함수 내에서 이러한 작업을 수행하는 경우 Lambda 실행 역할 및 권한 경계에 다음 권한도 포함되어야 합니다.

연산	IAM 권한 필요	Resource
감사 대상이 없는 데이터 보호 정책 생성	`logs:PutDataProtectionPolicy`	`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`

CloudWatch Logs를 감사 대상으로 사용하여 데이터 보호 정책 생성	`logs:PutDataProtectionPolicy`
	
	`logs:CreateLogDelivery`
	
	`logs:PutResourcePolicy`
	
	`logs:DescribeResourcePolicies`
	
	`logs:DescribeLogGroups`
		`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
		
		`*`
		
		`*`
		
		`*`
		
		`*`

Firehose를 감사 대상으로 사용하여 데이터 보호 정책 생성	`logs:PutDataProtectionPolicy`
	
	`logs:CreateLogDelivery`
	
	`firehose:TagDeliveryStream`
		`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
		
		`*`
		
		`arn:aws:logs:::deliverystream/` *YOUR_DELIVERY_STREAM*

연산	IAM 권한 필요	Resource
Amazon S3를 감사 대상으로 사용하여 데이터 보호 정책 생성	logs:PutDataProtectionPolicy logs:CreateLogDelivery s3:GetBucketPolicy s3:PutBucketPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :* * arn:aws:s3::: *YOUR_BUCKET* arn:aws:s3::: *YOUR_BUCKET*
마스킹된 로그 이벤트 마스크 처리 해제	logs:Unmask	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*
기존 데이터 보호 정책 보기	logs:GetDataProtectionPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*
데이터 보호 정책 삭제	logs:DeleteDataProtectionPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*

데이터 보호 감사 로그가 이미 대상으로 전송되고 있는 경우 동일한 대상으로 로그를 전송하는 다른 정책에는 `logs:PutDataProtectionPolicy` 및 `logs:CreateLogDelivery` 권한만 있으면 됩니다.

샘플 데이터 보호 정책

다음 샘플 정책을 사용하면 사용자는 세 가지 유형의 감사 대상 모두에 감사 결과를 전송할 수 있는 데이터 보호 정책을 생성, 확인 및 삭제할 수 있습니다. 사용자는 마스킹 처리되지 않은 데이터를 볼 수 없습니다.

JSON

```
    {
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogDeliveryConfiguration",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:DescribeLogGroups",
                "logs:DescribeResourcePolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowDataProtectionAndBucketConfiguration",
            "Effect": "Allow",
            "Action": [
                "logs:GetDataProtectionPolicy",
                "logs:DeleteDataProtectionPolicy",
                "logs:PutDataProtectionPolicy",
                "s3:PutBucketPolicy",
                "firehose:TagDeliveryStream",
                "s3:GetBucketPolicy"
            ],
            "Resource": [
            "arn:aws:firehose:us-east-1:111122223333:deliverystream/delivery-
stream-name",
            "arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "arn:aws:logs:us-east-1:111122223333:log-group:log-group-name:*"
            ]
        }
    ]
}
```

계정 전체 데이터 보호 정책 생성

CloudWatch Logs 콘솔 또는 AWS CLI 명령을 사용하여 계정의 모든 로그 그룹에 대해 민감한 데이터를 마스킹하는 데이터 보호 정책을 생성할 수 있습니다. 이는 현재 로그 그룹과 향후 생성하는 로그 그룹 모두에 영향을 미칩니다.

> **Important**
>
> 민감한 데이터는 로그 그룹에 수집될 때 감지 및 마스킹 처리됩니다. 데이터 보호 정책을 설정하면 해당 시간 이전에 로그 그룹에 수집된 로그 이벤트는 마스킹 처리되지 않습니다.

주제

- 콘솔
- AWS CLI

콘솔

콘솔을 사용하여 계정 전체 데이터 보호 정책 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다. 목록 하단 근처에 있습니다.
3. 로그 탭을 선택합니다.
4. 구성을 선택합니다.
5. 관리형 데이터 식별자에서 모든 로그 그룹에 대해 감사하고 마스킹할 데이터 유형을 선택합니다. 선택 상자에 입력하여 원하는 식별자를 찾을 수 있습니다.

   로그 데이터 및 비즈니스와 관련된 데이터 식별자만 선택하는 것이 좋습니다. 여러 유형의 데이터를 선택하면 오탐이 발생할 수 있습니다.

   보호할 수 있는 데이터 유형에 대한 자세한 내용은 보호할 수 있는 데이터 유형을 참조하세요.
6. (선택 사항) 사용자 지정 데이터 식별자를 사용하여 다른 유형의 데이터를 감사하고 마스킹하려면 사용자 지정 데이터 식별자 추가를 선택합니다. 그런 다음 데이터 유형의 이름과 로그 이벤트에서 해당 데이터 유형을 검색하는 데 사용할 정규 표현식을 입력합니다. 자세한 내용은 사용자 지정 데이터 식별자 단원을 참조하십시오.

   하나의 데이터 보호 정책에 최대 10개의 사용자 지정 데이터 식별자가 포함될 수 있습니다. 사용자 지정 데이터 식별자를 정의하는 각 정규 표현식은 200자 이하여야 합니다.
7. (선택 사항) 감사 결과를 전송할 서비스를 하나 이상 선택합니다. 이러한 서비스에 감사 결과를 보내지 않기로 선택하더라도 선택한 민감한 데이터 유형은 여전히 마스킹 처리됩니다.
8. 데이터 보호 활성화를 선택합니다.

AWS CLI

AWS CLI 를 사용하여 데이터 보호 정책을 생성하려면

1. 텍스트 편집기를 사용하여 DataProtectionPolicy.json 정책 파일을 생성합니다. 정책 구문에 대한 자세한 내용은 다음 섹션을 참조하세요.
2. 다음 명령을 입력합니다.

```
aws logs put-account-policy \
--policy-name TEST_POLICY --policy-type "DATA_PROTECTION_POLICY" \
--policy-document file://policy.json \
--scope "ALL" \
--region us-west-2
```

AWS CLI 또는 API 작업에 대한 데이터 보호 정책 구문

AWS CLI 명령 또는 API 작업에 사용할 JSON 데이터 보호 정책을 생성할 때 정책에는 두 개의 JSON 블록이 포함되어야 합니다.

- 첫 번째 블록은 DataIdentifer 배열 및 Audit 작업이 있는 Operation 속성을 모두 포함해야 합니다. DataIdentifer 배열은 마스킹 처리하려는 민감한 데이터 유형을 나열합니다. 이용 가능한 옵션에 대한 자세한 내용은 보호할 수 있는 데이터 유형을 참조하세요.

  민감한 데이터 용어를 찾으려면 Audit 작업이 있는 Operation 속성이 필요합니다. 이 Audit 작업에는 FindingsDestination 객체가 포함되어야 합니다. 선택 사항으로 해당 FindingsDestination 객체를 사용하여 감사 결과 보고서를 보낼 하나 이상의 대상을 나열할 수 있습니다. 로그 그룹, Amazon Data Firehose 스트림, S3 버킷 같은 대상을 지정하는 경우 해당 대상이 이미 존재해야 합니다. 감사 결과 보고서의 예제는 감사 결과 보고서을 참조하세요.
- 두 번째 블록은 DataIdentifer 배열 및 Deidentify 작업이 있는 Operation 속성을 모두 포함해야 합니다. DataIdentifer 배열은 정책의 첫 번째 블록에 있는 DataIdentifer 배열과 정확히 일치해야 합니다.

  Deidentify 작업이 있는 Operation 속성은 실제로 데이터를 마스킹 처리하며, 해당 속성은 "MaskConfig": {} 객체를 포함해야 합니다. "MaskConfig": {} 객체는 비어 있어야 합니다.

다음은 관리형 데이터 식별자만 사용하는 데이터 보호 정책의 예제입니다. 이 정책은 이메일 주소와 미국 운전면허증을 마스킹합니다.

사용자 지정 데이터 식별자를 지정하는 정책에 대한 자세한 내용은 데이터 보호 정책에서 사용자 지정 데이터 식별자 사용을 참조하세요.

```
{
    "Name": "data-protection-policy",
    "Description": "test description",
    "Version": "2021-06-01",
    "Statement": [{
            "Sid": "audit-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "CloudWatchLogs": {
                            "LogGroup": "EXISTING_LOG_GROUP_IN_YOUR_ACCOUNT,"
                        },
                        "Firehose": {
                            "DeliveryStream": "EXISTING_STREAM_IN_YOUR_ACCOUNT"
                        },
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
                        }
                    }
                }
            }
        },
        {
            "Sid": "redact-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Deidentify": {
                    "MaskConfig": {}
                }
            }
        }
    ]
}
```

단일 로그 그룹에 대한 데이터 보호 정책 생성

CloudWatch Logs 콘솔 또는 AWS CLI 명령을 사용하여 민감한 데이터를 마스킹하는 데이터 보호 정책을 생성할 수 있습니다.

각 로그 그룹에 하나의 데이터 보호 정책을 할당할 수 있습니다. 각 데이터 보호 정책은 여러 유형의 정보를 감사할 수 있습니다. 각 데이터 보호 정책은 하나의 감사 명령문을 포함할 수 있습니다.

주제

- 콘솔
- AWS CLI

콘솔

콘솔을 사용하여 데이터 보호 정책을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택합니다.
4. 작업, 데이터 보호 정책 생성을 선택합니다.
5. 관리형 데이터 식별자의 경우 이 로그 그룹에서 감사하고 마스킹 처리할 데이터 유형을 선택합니다. 선택 상자에 입력하여 원하는 식별자를 찾을 수 있습니다.

   로그 데이터 및 비즈니스와 관련된 데이터 식별자만 선택하는 것이 좋습니다. 여러 유형의 데이터를 선택하면 오탐이 발생할 수 있습니다.

   관리형 데이터 식별자를 사용하여 보호할 수 있는 데이터 유형에 대한 자세한 내용은 보호할 수 있는 데이터 유형을 참조하세요.
6. (선택 사항) 사용자 지정 데이터 식별자를 사용하여 다른 유형의 데이터를 감사하고 마스킹하려면 사용자 지정 데이터 식별자 추가를 선택합니다. 그런 다음 데이터 유형의 이름과 로그 이벤트에서 해당 데이터 유형을 검색하는 데 사용할 정규 표현식을 입력합니다. 자세한 내용은 사용자 지정 데이터 식별자 단원을 참조하십시오.

   하나의 데이터 보호 정책에 최대 10개의 사용자 지정 데이터 식별자가 포함될 수 있습니다. 사용자 지정 데이터 식별자를 정의하는 각 정규 표현식은 200자 이하여야 합니다.
7. (선택 사항) 감사 결과를 전송할 서비스를 하나 이상 선택합니다. 이러한 서비스에 감사 결과를 보내지 않기로 선택하더라도 선택한 민감한 데이터 유형은 여전히 마스킹 처리됩니다.

8. 데이터 보호 활성화를 선택합니다.

AWS CLI

AWS CLI 를 사용하여 데이터 보호 정책을 생성하려면

1. 텍스트 편집기를 사용하여 DataProtectionPolicy.json 정책 파일을 생성합니다. 정책 구문에 대한 자세한 내용은 다음 섹션을 참조하세요.
2. 다음 명령을 입력합니다.

```
aws logs put-data-protection-policy --log-group-identifier "my-log-group" --policy-
document file:///Path/DataProtectionPolicy.json --region us-west-2
```

AWS CLI 또는 API 작업에 대한 데이터 보호 정책 구문

AWS CLI 명령 또는 API 작업에 사용할 JSON 데이터 보호 정책을 생성할 때 정책에는 두 개의 JSON 블록이 포함되어야 합니다.

- 첫 번째 블록은 DataIdentifer 배열 및 Audit 작업이 있는 Operation 속성을 모두 포함해야 합니다. DataIdentifer 배열은 마스킹 처리하려는 민감한 데이터 유형을 나열합니다. 이용 가능한 옵션에 대한 자세한 내용은 보호할 수 있는 데이터 유형을 참조하세요.

  민감한 데이터 용어를 찾으려면 Audit 작업이 있는 Operation 속성이 필요합니다. 이 Audit 작업에는 FindingsDestination 객체가 포함되어야 합니다. 선택 사항으로 해당 FindingsDestination 객체를 사용하여 감사 결과 보고서를 보낼 하나 이상의 대상을 나열할 수 있습니다. 로그 그룹, Amazon Data Firehose 스트림, S3 버킷 같은 대상을 지정하는 경우 해당 대상이 이미 존재해야 합니다. 감사 결과 보고서의 예제는 감사 결과 보고서을 참조하세요.
- 두 번째 블록은 DataIdentifer 배열 및 Deidentify 작업이 있는 Operation 속성을 모두 포함해야 합니다. DataIdentifer 배열은 정책의 첫 번째 블록에 있는 DataIdentifer 배열과 정확히 일치해야 합니다.

  Deidentify 작업이 있는 Operation 속성은 실제로 데이터를 마스킹 처리하며, 해당 속성은 "MaskConfig": {} 객체를 포함해야 합니다. "MaskConfig": {} 객체는 비어 있어야 합니다.

다음은 이메일 주소와 미국 운전면허증을 마스킹 처리하는 데이터 보호 정책의 예입니다.

```
{
```

```
    "Name": "data-protection-policy",
    "Description": "test description",
    "Version": "2021-06-01",
    "Statement": [{
            "Sid": "audit-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "CloudWatchLogs": {
                            "LogGroup": "EXISTING_LOG_GROUP_IN_YOUR_ACCOUNT,"
                        },
                        "Firehose": {
                            "DeliveryStream": "EXISTING_STREAM_IN_YOUR_ACCOUNT"
                        },
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
                        }
                    }
                }
            },
            {
                "Sid": "redact-policy",
                "DataIdentifier": [
                    "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                    "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
                ],
                "Operation": {
                    "Deidentify": {
                        "MaskConfig": {}
                    }
                }
            }
    ]
}
```

마스킹 처리되지 않은 데이터 보기

마스킹 처리되지 않은 데이터를 확인하려면 사용자에게 `logs:Unmask` 권한이 있어야 합니다. 이 권한이 있는 사용자는 다음과 같은 방법으로 마스킹 처리되지 않은 데이터를 볼 수 있습니다.

- 로그 스트림에서 이벤트를 볼 때는 표시, 마스크 해제를 선택합니다.
- unmask(@message)[마스크 해제(@message)] 명령이 포함된 CloudWatch Logs Insights 쿼리를 사용합니다. 다음 예제 쿼리는 스트림의 최신 로그 이벤트 20개를 마스킹 처리되지 않은 상태로 표시합니다.

  ```
  fields @timestamp, @message, unmask(@message)
  | sort @timestamp desc
  | limit 20
  ```

  CloudWatch Logs Insights 명령에 대한 자세한 내용은 CloudWatch Logs Insights 언어 쿼리 구문을 참조하세요.
- `unmask` 파라미터와 함께 GetLogEvents 또는 FilterLogEvents 작업을 사용하세요.

CloudWatchLogsFullAccess 정책은 `logs:Unmask` 권한을 포함합니다. CloudWatchLogsFullAccess가 없는 사용자에게 `logs:Unmask` 권한을 부여하려면 해당 사용자에게 사용자 지정 IAM 정책을 연결하면 됩니다. 자세한 내용은 사용자(콘솔)에게 권한 추가를 참조하세요.

감사 결과 보고서

CloudWatch Logs, Amazon S3 또는 Firehose에 감사 보고서를 쓰도록 CloudWatch Logs 데이터 보호 감사 정책을 설정한 경우 해당 결과 보고서는 다음 예와 비슷합니다. CloudWatch Logs는 민감한 데이터가 포함된 각 로그 이벤트에 대해 하나의 결과 보고서를 작성합니다.

```
{
    "auditTimestamp": "2023-01-23T21:11:20Z",
    "resourceArn": "arn:aws:logs:us-west-2:111122223333:log-group:/aws/lambda/
MyLogGroup:*",
    "dataIdentifiers": [
        {
            "name": "EmailAddress",
            "count": 2,
            "detections": [
                {
                    "start": 13,
```

```
                    "end": 26
                },
{
                    "start": 30,
                    "end": 43
                }
            ]
        }
    ]
}
```

보고서의 필드는 다음과 같습니다.

- `resourceArn` 필드는 민감한 데이터가 발견된 로그 그룹을 표시합니다.
- `dataIdentifiers` 객체는 감사 중인 민감한 데이터 유형 중 하나에 대한 결과 정보를 표시합니다.
- `name` 필드는 이 섹션에서 보고하는 민감한 데이터의 유형을 식별합니다.
- `count` 필드는 이 유형의 민감한 데이터가 로그 이벤트에서 나타나는 횟수를 표시합니다.
- `start` 및 `end` 필드는 로그 이벤트에서 민감한 데이터의 각 항목이 나타나는 위치를 문자 수를 기준으로 보여줍니다.

이전 예에서는 하나의 로그 이벤트에서 두 개의 이메일 주소를 발견한 보고서를 보여 줍니다. 첫 번째 이메일 주소는 로그 이벤트의 13번째 문자에서 시작하여 26번째 문자에서 끝납니다. 두 번째 이메일 주소는 30번째 문자부터 43번째 문자까지입니다. 이 로그 이벤트에는 두 개의 이메일 주소가 있지만 `LogEventsWithFindings` 지표는 중요한 데이터가 나타난 횟수가 아닌 중요한 데이터를 포함하는 로그 이벤트 수를 계산하므로 지표의 값은 하나씩만 증가합니다.

로 보호되는 버킷으로 감사 결과를 보내는 데 필요한 키 정책 AWS KMS

Amazon S3-관리형 키(SSE-S3)를 사용한 서버 측 암호화 또는 KMS 키(SSE-KMS)를 사용한 서버 측 암호화를 활성화하여 Amazon S3 버킷의 데이터를 보호할 수 있습니다. 자세한 내용은 Amazon S3 사용 설명서의 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

SSE-S3로 보호되는 버킷에 감사 결과를 보내는 경우 추가 구성이 필요하지 않습니다. Amazon S3은 암호화 키를 처리합니다.

SSE-KMS로 보호되는 버킷에 감사 결과를 보내는 경우 로그 전달 계정이 S3 버킷에 쓸 수 있도록 KMS에 대한 키 정책을 업데이트해야 합니다. SSE-KMS와 함께 사용하는 데 필요한 키 정책에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 Amazon S3를 참조하세요.

보호할 수 있는 데이터 유형

이 섹션에는 CloudWatch Logs 데이터 보호 정책에서 보호할 수 있는 데이터 유형에 대한 정보가 포함되어 있습니다. CloudWatch Logs 관리형 데이터 식별자는 금융 데이터, 개인 건강 정보(PHI) 및 개인 식별 정보(PII)를 보호하기 위해 사전 구성된 데이터 유형을 제공합니다. 또한 사용자 지정 데이터 식별자를 사용하여 특정 사용 사례에 맞는 데이터 식별자를 생성할 수 있습니다.

목차

- 민감한 데이터 유형에 대한 CloudWatch Logs 관리형 데이터 식별자
  - 자격 증명
    - 자격 증명 데이터 유형용 데이터 식별자 ARN
  - 디바이스 식별자
    - 디바이스 데이터 유형을 위한 데이터 식별자 ARN
  - 금융 정보
    - 금융 데이터 유형용 데이터 식별자 ARN
  - 보호 대상 건강 정보(PHI)
    - 보호 대상 건강 정보(PHI) 데이터 유형에 대한 데이터 식별자 ARN
  - 개인 식별 정보(PII)
    - 운전면허증 식별 번호 키워드
    - 국가 식별 번호 키워드
    - 여권 번호 키워드
    - 납세자 식별 및 참조 번호 키워드
    - 개인 식별 정보(PII)에 대한 데이터 식별자 ARN
- 사용자 지정 데이터 식별자
  - 사용자 지정 데이터 식별자란?
  - 사용자 지정 데이터 식별자 제약
  - 콘솔에서 사용자 지정 데이터 식별자 사용
  - 데이터 보호 정책에서 사용자 지정 데이터 식별자 사용

민감한 데이터 유형에 대한 CloudWatch Logs 관리형 데이터 식별자

이 섹션은 관리형 데이터 식별자를 사용하여 보호할 수 있는 데이터 유형과 이러한 각 데이터 유형과 관련된 국가 및 리전에 대한 정보를 포함하고 있습니다.

일부 중요한 데이터 유형의 경우, CloudWatch Logs 데이터 보호는 데이터 근처에 있는 키워드를 스캔하고 해당 키워드를 찾은 경우에만 일치하는 키워드를 찾습니다. 키워드가 특정 유형의 데이터와 인접해야 하는 경우 키워드는 일반적으로 데이터로부터 30자 이내(포함) 이내여야 합니다.

키워드에 공백이 포함된 경우 CloudWatch Logs 데이터 보호는 공백이 누락되었거나 공백 대신 밑줄(_) 또는 하이픈(-)이 포함된 유사 키워드를 자동으로 찾습니다. 일부 경우 CloudWatch Logs는 키워드의 일반적인 변형을 해결하기 위해 키워드를 확장하거나 축약하기도 합니다.

다음 표는 CloudWatch Logs가 관리형 데이터 식별자를 사용하여 탐지할 수 있는 자격 증명, 디바이스, 금융, 의료 및 보호 대상 건강 정보(PHI)의 유형을 나열합니다. 이는 개인 식별 정보(PII)로도 인정될 수 있는 특정 유형의 데이터에 추가됩니다.

언어 및 리전에 구애받지 않는 지원되는 식별자

식별자	카테고리
Address	개인
AwsSecretKey	자격 증명
CreditCardExpiration	금융
CreditCardNumber	금융
CreditCardSecurityCode	금융
EmailAddress	개인
IpAddress	개인
LatLong	개인
Name	개인
OpenSshPrivateKey	자격 증명
PgpPrivateKey	자격 증명
PkcsPrivateKey	자격 증명
PuttyPrivateKey	자격 증명

식별자	카테고리
VehicleIdentificationNumber	개인

리전별 데이터 식별자에는 식별자 이름, 하이픈, 두 글자(ISO 3166-1 alpha-2) 코드를 포함해야 합니다. 예: DriversLicense-US

두 글자로 된 국가 또는 리전 코드를 포함해야 하는 지원되는 식별자

식별자	카테고리	국가 및 언어
BankAccountNumber	금융	DE, ES, FR, GB, IT, US
CepCode	개인	BR
Cnpj	개인	BR
CpfCode	개인	BR
DriversLicense	개인	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Health	US
ElectoralRollNumber	개인	GB
HealthInsuranceCardNumber	Health	EU
HealthInsuranceClaimNumber	Health	US
HealthInsuranceNumber	Health	FR
HealthcareProcedureCode	Health	US

식별자	카테고리	국가 및 언어
IndividualTaxIdentification Number	개인	US
InseeCode	개인	FR
MedicareBeneficiaryNumber	Health	US
NationalDrugCode	Health	US
NationalIdentificationNumber	개인	DE, ES, IT
NationalInsuranceNumber	개인	GB
NationalProviderId	Health	US
NhsNumber	Health	GB
NieNumber	개인	ES
NifNumber	개인	ES
PassportNumber	개인	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	개인	CA
PersonalHealthNumber	Health	CA
PhoneNumber	개인	BR, DE, ES, FR, GB, IT, US
PostalCode	개인	CA
RgNumber	개인	BR
SocialInsuranceNumber	개인	CA
Ssn	개인	ES, US
TaxId	개인	DE, ES, FR, GB
ZipCode	개인	US

자격 증명

CloudWatch Logs 데이터 보호는 다음 유형의 자격 증명을 찾을 수 있습니다.

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전
AWS 보안 액세스 키	`AwsSecretKey`	`aws_secret_access_key` , `credentials` , `secret access key`, `secret key`, `set-awscredential`	모두
OpenSSH 프라이빗 키	`OpenSSHPrivateKey`	없음	모두
PGP 프라이빗 키	`PgpPrivateKey`	없음	모두
Pkcs 프라이빗 키	`PkcsPrivateKey`	없음	모두
PuTTY 프라이빗 키	`PuttyPrivateKey`	없음	모두

자격 증명 데이터 유형용 데이터 식별자 ARN

다음은 데이터 보호 정책에 추가할 수 있는 데이터 식별자의 Amazon Resource Name(ARN) 목록입니다.

자격 증명 데이터 식별자 ARN
`arn:aws:dataprotection::aws:data-identifier/AwsSecretKey`
`arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey`
`arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey`
`arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey`

자격 증명 데이터 식별자 ARN
`arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey`

디바이스 식별자

CloudWatch Logs 데이터 보호는 다음 유형의 디바이스 식별자를 찾을 수 있습니다.

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전
IP 주소	`IpAddress`	없음	모두

디바이스 데이터 유형을 위한 데이터 식별자 ARN

다음은 데이터 보호 정책에 추가할 수 있는 데이터 식별자의 Amazon Resource Name(ARN) 목록입니다.

디바이스 데이터 식별자 ARN
`arn:aws:dataprotection::aws:data-identifier/IpAddress`

금융 정보

CloudWatch Logs 데이터 보호는 다음과 같은 유형의 금융 정보를 찾을 수 있습니다.

데이터 보호 정책을 설정하면 CloudWatch Logs는 로그 그룹이 위치한 지리적 위치에 관계없이 사용자가 지정한 데이터 식별자를 스캔합니다. 이 표에서 Countries and regions(국가 및 리전) 열에 있는 정보는 해당 국가 및 리전에 적합한 키워드를 검색하기 위해 데이터 식별자에 두 글자로 된 국가 코드를 추가해야 하는지 그 여부를 지정합니다.

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
은행 계좌 번호	`BankAccountNumber`	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은	프랑스, 독	국가 코드

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
		이 섹션 뒷부분의 은행 계좌 번호 키워드 표를 참조하세요.	일, 이탈리아, 스페인, 영국, 미국	같은 요소를 포함하여 최대 34개의 영숫자로 구성된 국제 은행 계좌 번호(IBAN)를 포함합니다.
신용 카드 유효 기간	`CreditCardExpiration`	`exp d`, `exp m`, `exp y`, `expiration` , `expiry`	모두	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
신용 카드 번호	CreditCardNumber	account number, american express, amex, bank card, card, card number, card num, cc #, ccn, check card, credit, credit card#, dankort, debit, debit card, diners club, discover, electron, japanese card bureau, jcb, mastercard , mc, pan, payment account number, payment card number, pcn, union pay, visa	모두	탐지하려면 데이터가 Luhn 수표 공식을 준수하고 American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card, Bureau(JCB), Mastercard, UnionPay 및 Visa 유형의 신용 카드에 대해 표준 카드 번

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
				호 접두사를 사용하는 13~19자리 숫자 시퀀스여야 합니다.
신용 카드 인증 코드	CreditCardSecurityCode	card id, card identification code, card identification number , card security code, card validation code , card validation number , card verification data , card verification value, cvc, cvc2, cvv, cvv2, elo verification code	모두	

은행 계좌 번호 키워드

다음 은행 계좌 번호 키워드를 사용합니다. 여기에는 국가 코드 같은 요소를 포함하여 최대 34개의 영숫자로 구성된 국제 은행 계좌 번호(IBAN)가 포함됩니다.

국가	키워드
프랑스	account code, account number, accountno# , accountnumber# , bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte
독일	account code, account number, accountno# , accountnumber# , bankleitzahl , bban, customer account id, customer account number, customer bank account id, geheimzahl , iban, kartennummer , kontonummer , kreditkartennummer , sepa
이탈리아	account code, account number, accountno# , accountnumber# , bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto
스페인	account code, account number, accountno# , accountnumber# , bban, código cuenta, código cuenta bancaria, cuenta cliente id, customer account ID, customer account number, customer bank account id, iban, número cuenta bancaria cliente, número cuenta cliente
영국	account code, account number, accountno# , accountnumber# , bban, customer account ID, customer account number, customer bank account id, iban, sepa
미국	bank account, bank acct, checking account, checking acct, deposit account, deposit acct, savings account, savings acct, chequing account, chequing acct

CloudWatch Logs는 신용 카드 발급 기관이 공개 테스트를 위해 예약한 다음 시퀀스 발생을 보고하지 않습니다.

```
122000000000003, 2222405343248877, 2222990905257051, 2223007648726984,
 2223577120017656,
30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505,
 36148900647913,
```

```
36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237,
 4012888888881881,
4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000,
 4911830000000,
4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742,
 5105105105105100,
5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,
 5204740009900014, 5420923878724339,
5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444,
 5506900510000234, 5506920809243667,
5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194,
 5555555555554444, 5610591081018250,
6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441,
 630495060000000000,
6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.
```

금융 데이터 유형용 데이터 식별자 ARN

다음은 데이터 보호 정책에 추가할 수 있는 데이터 식별자의 Amazon Resource Name(ARN) 목록입니다.

금융 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-DE
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-ES
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-FR
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-GB
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-IT
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-US
arn:aws:dataprotection::aws:data-identifier/CreditCardExpiration
arn:aws:dataprotection::aws:data-identifier/CreditCardNumber
arn:aws:dataprotection::aws:data-identifier/CreditCardSecurityCode

보호 대상 건강 정보(PHI)

CloudWatch Logs 데이터 보호는 다음과 같은 유형의 보호 대상 건강 정보(PHI) 를 찾을 수 있습니다.

데이터 보호 정책을 설정하면 CloudWatch Logs는 로그 그룹이 위치한 지리적 위치에 관계없이 사용자가 지정한 데이터 식별자를 스캔합니다. 이 표에서 Countries and regions(국가 및 리전) 열에 있는 정보는 해당 국가 및 리전에 적합한 키워드를 검색하기 위해 데이터 식별자에 두 글자로 된 국가 코드를 추가해야 하는지 그 여부를 지정합니다.

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전
마약단속국(DEA) 등록 번호	`DrugEnforcementAgencyNumber`	`dea number`, `dea registration`	미국
건강 보험 카드 번호(EHIC)	`HealthInsuranceCardNumber`	`assicurazione sanitaria numero`, `carta assicurazione numero`, `carte d’assurance maladie `, `carte européenne d'assurance maladie `, `ceam`, `ehic`, `ehic#`, `finlandehicnumber#  `, `gesundheitskarte `, `hälsokort `, `health card`, `health card number`, `health insurance card`, `health insurance number`, `insurance card number`, `krankenversicherungskarte `, `krankenversicherungsnummer  `, `medical account number`, `numero conto medico`, `numéro d’assurance maladie `, `numéro`	유럽 연합

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전
		de carte d’assurance , numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin , sairausvakuutuskortti , sairausvakuutusnumero , sjukförsäkring nummer, sjukförsäkringskort , suomi ehic-numero , tarjeta de salud, terveyskortti , tessera sanitaria assicurazione numero , versicherungsnummer	
건강 보험 청구 번호(HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#, hicno#	미국
건강 보험 또는 의료 식별 번호	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	프랑스
Healthcare Common Procedure Coding System(HCPCS) 코드	HealthcareProcedureCode	current procedural terminology , hcpcs, healthcare common procedure coding system	미국

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전
메디케어 수혜자 번호(MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	미국
국가 의약품 코드(NDC)	NationalDrugCode	national drug code, ndc	미국
국가 공급자 식별자(NPI)	NationalProviderId	hipaa, n.p.i., national provider, npi	미국
국가 의료 서비스(NHS) 번호	NhsNumber	national health service, NHS	영국
개인 건강 번호	PersonalHealthNumber	canada healthcare number, msp number, care number, phn, soins de santé	캐나다

보호 대상 건강 정보(PHI) 데이터 유형에 대한 데이터 식별자 ARN

다음은 보호 대상 건강 정보(PHI) 데이터 보호 정책에서 사용할 수 있는 데이터 식별자 Amazon Resource Name(ARN)을 나열합니다.

PHI 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU

PHI 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US
arn:aws:dataprotection::aws:data-identifier/NationalInsuranceNumber-GB
arn:aws:dataprotection::aws:data-identifier/NationalProviderId-US
arn:aws:dataprotection::aws:data-identifier/NhsNumber-GB
arn:aws:dataprotection::aws:data-identifier/PersonalHealthNumber-CA

개인 식별 정보(PII)

CloudWatch Logs 데이터 보호는 다음과 같은 유형의 개인 식별 정보(PII)를 찾을 수 있습니다.

데이터 보호 정책을 설정하면 CloudWatch Logs는 로그 그룹이 위치한 지리적 위치에 관계없이 사용자가 지정한 데이터 식별자를 스캔합니다. 이 표에서 Countries and regions(국가 및 리전) 열에 있는 정보는 해당 국가 및 리전에 적합한 키워드를 검색하기 위해 데이터 식별자에 두 글자로 된 국가 코드를 추가해야 하는지 그 여부를 지정합니다.

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
생년월일	DateOfBirth	dob, date of birth, birthdate , birth	임의	지원에는 모든 숫

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
		`date`, `birthday`, `b-day`, `bday`		자, 숫자 및 월 이름의 조합과 같은 대부분의 날짜 형식이 포함됩니다. 날짜 구성 요소는 공백, 슬래시(/) 또는 하이픈(-)으로 구분할 수 있습니다.
Código de Endereçamento Postal(CEP)	`CepCode`	`cep`, `código de endereçamento postal`, `codigo de endereçamento postal`	브라질	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
Cadastro Nacional da Pessoa Jurídica(CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	브라질	
Cadastro de Pessoas Físicas(CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	브라질	
운전면허증 식별 번호	DriversLicense	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 운전면허증 식별 번호 표를 참조하세요.	여러 국가. 자세한 내용은 운전면허증 식별 번호 표를 참조하세요.	
선거인단 번호	ElectoralRollNumber	electoral #, electoral number, electoral roll #, electoral roll no., electoral roll number, electoral rollno	영국	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
개인 납세자 식별	`IndividualTaxIdentificationNumber`	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 개인 납세자 식별 번호 표를 참조하세요.	브라질, 프랑스, 독일, 스페인, 영국	
국립 통계 경제 연구소(INSEE)	`InseeCode`	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 국가 식별 번호용 키워드 표를 참조하세요.	프랑스	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
국적 식별 번호	`NationalIdentificationNumber`	예. 자세한 내용은 이 섹션 뒷부분의 국가 식별 번호용 키워드 표를 참조하세요.	독일, 이탈리아, 스페인	여기에는 DNI(Documento Nacional de Identidad) 식별자(스페인), Codice fiscale 코드(이탈리아) 및 국가 신분증 번호(독일어)가 포함됩니다.
National Insurance Number(NINO)	`NationalInsuranceNumber`	`insurance no.`, `insurance number`, `insurance#` , `national insurance number`, `nationalinsurance#` , `nationalinsurancenumber` , `nin`, `nino`	영국	–

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
Número de identidad de extranjero(NIE)	`NieNumber`	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 개인 납세자 식별 번호 표를 참조하세요.	스페인	
Número de Identificación Fiscal(NIF)	`NifNumber`	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 개인 납세자 식별 번호 표를 참조하세요.	스페인	
여권 번호	`PassportNumber`	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 여권 번호 키워드 표를 참조하세요.	캐나다, 프랑스, 독일, 이탈리아, 스페인, 영국, 미국	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
영주권 번호	Permanent Residence Number	carte résident permanent , numéro carte résident permanent , numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	캐나다	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
전화번호	PhoneNumber	브라질: 키워드에 cel, celular, fone, móvel, número residencial , numero residencial , telefone을 포함합니다. 기타: cell, contact, fax, fax number, mobile, phone, phone number, tel, telephone , telephone number	브라질, 캐나다, 프랑스, 독일, 이탈리아, 스페인, 영국, 미국	여기에는 미국 내 수신자 부담 전화번호와 팩스 번호가 포함됩니다. 키워드가 데이터에 근접한 경우 번호에 국가 코드를 포함할 필요가 없습니다. 키워드가 데이터에 근접하지 않은 경우 숫자에 국가

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
				코드가 포함되지 않아도 됩니다.
우편 번호	PostalCode	없음	캐나다	
Registro Geral(RG)	RgNumber	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 개인 납세자 식별 번호 표를 참조하세요.	브라질	
Social Insurance Number(SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	캐나다	
사회 보장 번호(SSN)	Ssn	스페인 – número de la seguridad social, social security no., social security no. número de la seguridad social, social security number, socialsecurityno# , ssn, ssn# 미국 – social security, ss#, ssn	스페인, 미국	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
납세자 식별 번호 또는 참조 번호	`TaxId`	예. 국가마다 다른 키워드가 적용됩니다. 자세한 내용은 이 섹션 뒷부분의 개인 납세자 식별 번호 표를 참조하세요. .	프랑스, 독일, 스페인, 영국	여기에는 TIN(프랑스), Steueridentifikationsnummer(독일), CIF(스페인), UTR 및 TRN(영국)이 포함됩니다.
우편 번호	`ZipCode`	`zip code`, `zip+4`	미국	미국 우편번호입니다.

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
우편 주소	`Address`	없음	호주, 캐나다, 프랑스, 독일, 이탈리아, 스페인, 영국, 미국	키워드가 필수 항목은 아니지만 검색하려면 주소에 도시 또는 장소의 이름과 우편 번호를 포함해야 합니다.
전자 메일 주소	`EmailAddress`	없음	임의	

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
위성 항법 시스템(GPS) 좌표	LatLong	coordinate , coordinates , lat long, latitude longitude , location, position	임의	CloudWatch Logs는 위도 및 경도 좌표가 쌍으로 저장되고 DD(10진수 도) 형식인 경우 GPS 좌표를 탐지할 수 있습니다(예: 41.948614,-87.655311). 지원에는 10진수 도(DDM) 형식의 좌표(예: 41°56.9168'N

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
				87°39.3187'W) 또는 도, 분, 초(DMS) 형식의 좌표(예: 41°56'55.0104"N 87°39'19.1196"W)가 포함되지 않습니다.
전체 이름	`Name`	없음	임의	CloudWatch Logs는 전체 이름만 감지할 수 있습니다. 라틴 문자 집합만 지원합니다.

데이터 유형	데이터 식별자 ID	필수 키워드	국가 및 리전	참고 사항
차량 식별 번호(VIN)	VehicleIdentificationNumber	Fahrgestellnummer , niv, numarul de identificare , numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles , numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	임의	CloudWatch Logs는 17자 시퀀스로 구성되고 ISO 3779 및 3780 표준을 준수하는 VIN을 탐지할 수 있습니다. 이 표준은 전 세계에서 사용할 수 있도록 설계되었습니다.

운전면허증 식별 번호 키워드

다양한 유형의 운전면허증 식별 번호를 탐지하기 위해 CloudWatch Logs에 해당 번호에 가까운 키워드가 있어야 합니다. 다음 표는 특정 국가 및 리전에서 CloudWatch Logs가 인식하는 키워드를 나열합니다.

국가 또는 리전	키워드
호주	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
오스트리아	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
벨기에	fuehrerschein, fuehrerschein- nr, fuehrerschheinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrerscheinnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
불가리아	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
캐나다	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit,

국가 또는 리전	키워드
	drivers permit number, driving licence, driving license, driving permit, permis de conduire
크로아티아	vozačka dozvola
사이프러스	άδεια οδήγησης
체코 공화국	číslo licence, číslo licence řidiče, číslo řidičského průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
덴마크	kørekort, kørekortnummer
에스토니아	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
핀란드	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
프랑스	permis de conduire
독일	fuehrerschein, fuehrerschein- nr, fuehrerschein, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrerschein, führerscheinnummer
그리스	δεια οδήγησης, adeia odigisis
헝가리	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
아일랜드	ceadúnas tiomána
이탈리아	patente di guida, patente di guida numero, patente guida, patente guida numero

국가 또는 리전	키워드
라트비아	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
리투아니아	vairuotojo pažymėjimas
룩셈부르크	fahrerlaubnis, führerschäin
몰타	liċenzja tas-sewqan
네덜란드	permis de conduire, rijbewijs, rijbewijsnummer
폴란드	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
포르투갈	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
루마니아	numărul permisului de conducere, permis de conducere
슬로바키아	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
슬로베니아	vozniško dovoljenje

국가 또는 리전	키워드
스페인	carnet conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
스웨덴	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.
영국	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
미국	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

국가 식별 번호 키워드

다양한 유형의 주민등록번호를 탐지하기 위해 CloudWatch Logs에 해당 번호에 가까운 키워드가 있어야 합니다. 여기에는 Documento Nacional de Identidad(DNI) 식별자(스페인), 프랑스 국립 통계 및 경제 연구소(INSEE) 코드, 독일 신분증 번호 및 Registro Geral(RG) 번호(브라질)이 포함됩니다.

다음 표는 특정 국가 및 리전에서 CloudWatch Logs가 인식하는 키워드를 나열합니다.

국가 또는 리전	키워드
브라질	registro geral, rg
프랑스	assurance sociale, carte nationale d’identité, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
독일	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
이탈리아	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
스페인	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationalidno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

여권 번호 키워드

다양한 유형의 여권 번호를 탐지하기 위해 CloudWatch Logs에 해당 번호에 가까운 키워드가 있어야 합니다. 다음 표는 특정 국가 및 리전에서 CloudWatch Logs가 인식하는 키워드를 나열합니다.

국가 또는 리전	키워드
캐나다	passeport, passeport#, passport, passport#, passportno, passportno#
프랑스	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
독일	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
이탈리아	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
스페인	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
영국	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
미국	passport, travel document

납세자 식별 및 참조 번호 키워드

다양한 유형의 납세자 식별 및 참조 번호를 탐지하기 위해 CloudWatch Logs는 숫자 근처에 키워드가 있어야 합니다. 다음 표는 특정 국가 및 리전에서 CloudWatch Logs가 인식하는 키워드를 나열합니다.

국가 또는 리전	키워드
브라질	cadastro de pessoa física, cadastro de pessoa fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj, cpf
프랑스	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
독일	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
스페인	cif, cif número, cifnúmero#, nie, nif, número de contribuyente, número de identidad de extranjero, número de identificación fiscal, número de impuesto corporativo, personal tax number, tax id, tax identification number, tax number, tin, tin#
영국	paye, tax id, tax id no., tax id number, tax identification, tax identification#, tax no., tax number, tax reference, tax#, taxid#, temporary reference number, tin, trn, unique tax reference, unique taxpayer reference, utr
미국	개인 납세자 식별 번호(itin, i.t.i.n.)

개인 식별 정보(PII)에 대한 데이터 식별자 ARN

다음 표는 데이터 보호 정책에 추가할 수 있는 개인 식별 정보(PII) 데이터 식별자의 Amazon Resource Name(ARN) 목록입니다.

PII 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:dataprotection::aws:data-identifier/CepCode-BR
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:dataprotection::aws:data-identifier/CpfCode-BR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BG
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CA
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CY
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CZ
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-EE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-ES
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GB
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HR

PII 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LV
arn:aws:dataprotection::aws:data-identifier/DriversLicense-MT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-NL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-RO
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-US
arn:aws:dataprotection::aws:data-identifier/ElectoralRollNumber-GB
arn:aws:dataprotection::aws:data-identifier/EmailAddress
arn:aws:dataprotection::aws:data-identifier/IndividualTaxIdentificationNumber-US
arn:aws:dataprotection::aws:data-identifier/InseeCode-FR

PII 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-DE
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-ES
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-IT
arn:aws:dataprotection::aws:data-identifier/NieNumber-ES
arn:aws:dataprotection::aws:data-identifier/NifNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-CA
arn:aws:dataprotection::aws:data-identifier/PassportNumber-DE
arn:aws:dataprotection::aws:data-identifier/PassportNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-FR
arn:aws:dataprotection::aws:data-identifier/PassportNumber-GB
arn:aws:dataprotection::aws:data-identifier/PassportNumber-IT
arn:aws:dataprotection::aws:data-identifier/PassportNumber-US
arn:aws:dataprotection::aws:data-identifier/PermanentResidenceNumber-CA
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-BR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-DE
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-ES

PII 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-FR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-GB
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-IT
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US
arn:aws:dataprotection::aws:data-identifier/PostalCode-CA
arn:aws:dataprotection::aws:data-identifier/RgNumber-BR
arn:aws:dataprotection::aws:data-identifier/SocialInsuranceNumber-CA
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:dataprotection::aws:data-identifier/TaxId-DE
arn:aws:dataprotection::aws:data-identifier/TaxId-ES
arn:aws:dataprotection::aws:data-identifier/TaxId-FR
arn:aws:dataprotection::aws:data-identifier/TaxId-GB
arn:aws:dataprotection::aws:data-identifier/VehicleIdentificationNumber
arn:aws:dataprotection::aws:data-identifier/ZipCode-US

사용자 지정 데이터 식별자

주제

- 사용자 지정 데이터 식별자란?
- 사용자 지정 데이터 식별자 제약

- 콘솔에서 사용자 지정 데이터 식별자 사용
- 데이터 보호 정책에서 사용자 지정 데이터 식별자 사용

사용자 지정 데이터 식별자란?

사용자 지정 데이터 식별자(CDI)를 통해 데이터 보호 정책에 사용할 수 있는 사용자 지정 정규 표현식을 정의할 수 있습니다. 사용자 지정 데이터 식별자를 사용하면 관리형 데이터 식별자로는 제공할 수 없는 비즈니스별 개인 식별 정보(PII) 사용 사례를 대상으로 지정할 수 있습니다. 예를 들어, 사용자 지정 데이터 식별자를 사용하여 회사별 직원 ID를 찾을 수 있습니다. 사용자 지정 데이터 식별자는 관리형 데이터 식별자와 함께 사용할 수 있습니다.

사용자 지정 데이터 식별자 제약

CloudWatch 사용자 지정 데이터 식별자에는 다음과 같은 제한 사항이 있습니다.

- 각 데이터 보호 정책에 최대 10개의 사용자 지정 데이터 식별자가 지원됩니다.
- 사용자 지정 데이터 식별자 이름의 최대 길이는 128자입니다. 다음 문자가 지원됩니다.
  - 영숫자: (a-zA-Z0-9)
  - 기호: ('_' | '-')
- RegEx의 최대 길이는 200자입니다. 다음 문자가 지원됩니다.
  - 영숫자: (a-zA-Z0-9)
  - 기호: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - RegEx 예약 문자: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- 사용자 지정 데이터 식별자는 관리형 데이터 식별자와 동일한 이름을 공유할 수 없습니다.
- 사용자 지정 데이터 식별자는 계정 수준 데이터 보호 정책 또는 로그 그룹 수준 데이터 보호 정책에서 지정할 수 있습니다. 관리형 데이터 식별자와 마찬가지로 계정 수준 정책 내에 정의된 사용자 지정 데이터 식별자는 로그 그룹 수준 정책에 정의된 사용자 지정 데이터 식별자와 함께 작동합니다.

콘솔에서 사용자 지정 데이터 식별자 사용

CloudWatch 콘솔을 사용하여 데이터 보호 정책을 생성하거나 편집할 때 사용자 지정 데이터 식별자를 지정하려면 데이터 식별자의 이름과 정규 표현식을 입력하기만 하면 됩니다. 예를 들어 **`Employee_ID`**를 이름으로, **`EmployeeID-\d{9}`**을 정규 표현식으로 입력할 수 있습니다. 이 정규 표현식은 `EmployeeID-` 뒤에 9개의 숫자가 있는 로그 이벤트를 탐지하고 마스킹합니다. 예: `EmployeeID-123456789`

데이터 보호 정책에서 사용자 지정 데이터 식별자 사용

AWS CLI 또는 AWS API를 사용하여 사용자 지정 데이터 식별자를 지정하는 경우 데이터 식별자 이름과 정규식을 데이터 보호 정책을 정의하는 데 사용되는 JSON 정책에 포함해야 합니다. 다음 데이터 보호 정책은 회사별 직원 ID가 있는 로그 이벤트를 탐지하고 마스킹합니다.

1. 데이터 보호 정책 내에 `Configuration` 블록을 생성합니다.
2. 사용자 지정 데이터 식별자에 `Name`을 입력합니다. 예: **`EmployeeId`**
3. 사용자 지정 데이터 식별자에 `Regex`을 입력합니다. 예: **`EmployeeID-\d{9}`** 이 정규 표현식은 `EmployeeID-` 뒤에 9자리 숫자가 있는 `EmployeeID-`가 포함된 로그 이벤트와 일치합니다. 예: `EmployeeID-123456789`
4. 정책 명령문에서 다음 사용자 지정 데이터 식별자를 참조하세요.

```
{
    "Name": "example_data_protection_policy",
    "Description": "Example data protection policy with custom data identifiers",
    "Version": "2021-06-01",
    "Configuration": {
      "CustomDataIdentifier": [
        {"Name": "EmployeeId", "Regex": "EmployeeId-\\d{9}"}
      ]
    },
    "Statement": [
        {
            "Sid": "audit-policy",
            "DataIdentifier": [
                "EmployeeId"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
                        }
                    }
                }
            }
        },
        {
            "Sid": "redact-policy",
            "DataIdentifier": [
```

```
            "EmployeeId"
            ],
            "Operation": {
                "Deidentify": {
                    "MaskConfig": {
                    }
                }
            }
        }
    ]
}
```

5. (선택 사항) 필요에 따라 사용자 지정 데이터 식별자를 `Configuration` 블록에 계속 추가합니다. 데이터 보호 정책은 현재 최대 10개의 사용자 지정 데이터 식별자를 지원합니다.

수집 중 로그 변환

로그 변환 및 보강을 사용하면 CloudWatch Logs로 수집할 때 일관되고 컨텍스트가 풍부한 형식으로 모든 로그를 정규화할 수 있습니다. AWS WAF 및 Amazon Route 53과 같은 일반적인 AWS 서비스에 대해 사전 구성된 템플릿을 사용하여 로그에 구조를 추가하거나 Grok과 같은 네이티브 파서를 사용하여 사용자 지정 변환기를 빌드할 수 있습니다. 기존 속성의 이름을 바꾸고 계정 ID 및 리전과 같은 추가 메타데이터를 로그에 추가할 수도 있습니다.

로그 변환은 애플리케이션 전반의 로그 쿼리를 간소화 및 단축하고 로그에 대한 알림 생성을 간소화하는 데 도움이 됩니다. 이 기능은 VPC 흐름 로그, Route 53 및 같은 주요 AWS 로그 소스에 대한 out-of-the-box 제공 변환 템플릿을 사용하여 일반적인 로그 유형에 대한 변환을 제공합니다 Amazon RDS for PostgreSQL. 사전 구성된 변환 템플릿을 사용하거나 필요에 맞게 사용자 지정 트랜스포머를 생성할 수 있습니다.

로그 변환을 사용하면 형식과 속성 이름이 매우 다양한 소스에서 내보낸 로그를 관리할 수 있습니다.

트랜스포머를 생성하면 수집된 로그 이벤트가 변환되고 표준 형식으로 저장됩니다. 이러한 변환된 로그를 활용하여 다음 기능을 통해 분석 경험을 가속화할 수 있습니다.

- 필드 인덱스
- CloudWatch Logs Insights 검색된 필드
- 지표 필터를 사용한 경보의 유연성
- 구독 필터를 통한 전달
- Contributor Insights를 사용하여 로그 이벤트에서 지표 데이터를 생성하면 Contributor Insights 규칙이 변환 전 또는 후에 로그 이벤트를 평가하도록 선택할 수 있습니다.

변환은 로그 수집 중에만 발생합니다. 이미 수집된 로그 이벤트는 변환할 수 없습니다. 변환은 되돌릴 수 없습니다. 원본 로그와 변환된 로그는 모두 동일한 보존 정책을 사용하여 CloudWatch Logs에 저장됩니다. 로그 변환 및 보강 기능은 기존 표준 로그 클래스 수집 가격에 포함됩니다. 로그 스토리지 비용은 원래 로그 볼륨을 초과할 수 있는 변환 후 로그 크기를 기준으로 합니다.

> **Important**
>
> 로그 이벤트가 변환된 후에는 CloudWatch Logs Insights 쿼리를 사용하여 변환된 로그 버전을 확인해야 합니다. GetLogEvents 및 FilterLogEvents 작업은 변환되기 전에 로그 이벤트의 원래 버전만 반환합니다.

> **Important**
>
> PutLogEvents에서 최대 1MB를 허용하는 단일 로그 이벤트에도 불구하고 로그 변환은 크기가 512kb 미만인 로그 이벤트만 처리할 수 있습니다. 512kb보다 큰 모든 로그 이벤트는 변환에 실패하고 오류를 내보냅니다. PutLogEvents의 총 크기는 여전히 512kb를 초과할 수 있습니다.

다른 형식으로 변환하는 것 외에도 계정 ID, 리전 및 키워드와 같은 추가 컨텍스트로 로그를 보강할 수도 있습니다. 로그 그룹 이름과 정적 키워드에서 추출됩니다.

로그 변환은 형식과 속성 이름이 매우 다양한 소스에서 생성된 로그를 제공하는 데 도움이 됩니다.

로그 변환 및 보강은 표준 로그 클래스의 로그 그룹에 대해서만 지원됩니다.

개별 로그 그룹에 대한 트랜스포머를 생성할 수 있으며 계정의 모든 또는 많은 로그 그룹에 적용되는 계정 수준 트랜스포머를 생성할 수도 있습니다. 로그 그룹에 로그 그룹 수준 트랜스포머가 있는 경우 해당 트랜스포머는 해당 로그 그룹에 적용되는 모든 계정 수준 트랜스포머를 재정의합니다.

주제

- 로그 트랜스포머 생성 및 관리
- 구성 가능한 구문 분석기 유형 프로세서
- AWS 벤딩 로그용 기본 제공 프로세서
- 문자열 변형 프로세서
- JSON 변형 프로세서
- 데이터타입 컨버터 프로세서
- 변환 지표 및 오류

로그 트랜스포머 생성 및 관리

로그 트랜스포머에는 논리적 파이프라인에 함께 있는 하나 이상의 프로세서가 포함됩니다. 각 프로세서는 트랜스포머 구성에 나열된 순서대로 차례로 로그 이벤트에 적용됩니다.

일부 프로세서는 구문 분석기 유형입니다. 각 트랜스포머에는 구문 분석기가 하나 이상 있어야 하며, 트랜스포머의 첫 번째 프로세서는 구문 분석기여야 합니다.

일부 구문 분석기는 특정 유형의 AWS 판매 로그에 대해 구성된 기본 제공 구문 분석기입니다.

다른 프로세서 유형은 문자열 변형기, JSON 변형기 및 데이터 프로세서입니다.

개별 로그 그룹에 대한 트랜스포머를 생성할 수 있으며 계정의 모든 또는 많은 로그 그룹에 적용되는 계정 수준 트랜스포머를 생성할 수도 있습니다. 로그 그룹에 로그 그룹 수준 트랜스포머가 있는 경우 해당 트랜스포머는 해당 로그 그룹에 적용되는 모든 계정 수준 트랜스포머를 재정의합니다. 계정의 한 리전에 최대 20개의 계정 수준 트랜스포머를 보유할 수 있습니다.

트랜스포머를 생성할 때 이러한 지침을 따라야 합니다.

- AWS 판매된 로그 유형에 대해 미리 구성된 구문 분석기를 포함하는 경우 변환기에 나열된 첫 번째 프로세서여야 합니다. 트랜스포머에는 이러한 프로세서를 하나만 포함할 수 있습니다.
- 트랜스포머에는 `grok` 프로세서를 하나만 포함할 수 있습니다.
- 트랜스포머에 구문 분석기 유형 프로세서가 하나 이상 있어야 합니다. 최대 5개의 구문 분석기 유형 프로세서를 포함할 수 있습니다. 이 5개 제한에는 기본 제공 구문 분석기와 구성 가능한 구문 분석기가 모두 포함됩니다.
- 트랜스포머에는 최대 20개의 프로세서가 있을 수 있습니다.
- 트랜스포머에는 addKeys 프로세서를 하나만 포함할 수 있습니다.
- 트랜스포머에는 copyValue 프로세서를 하나만 포함할 수 있습니다.
- 각 트랜스포머는 로그 이벤트에서 최대 200개의 필드를 추출할 수 있습니다.
- 각 로그 이벤트는 512KB 미만이어야 합니다. 로그 이벤트의 총 크기는 여전히 512KB를 초과할 수 있습니다.

주제

- 계정 수준 트랜스포머 정책 생성
- 계정 수준 트랜스포머 정책 편집 또는 삭제
- 처음부터 로그 그룹 수준 로그 트랜스포머 생성
- 기존 트랜스포머를 복사하여 로그 그룹 수준 트랜스포머 생성
- 로그 그룹 수준 트랜스포머 편집
- 로그 그룹 수준 트랜스포머 삭제

계정 수준 트랜스포머 정책 생성

이 섹션의 단계를 사용하여 계정의 모든 로그 그룹 또는 동일한 문자열(접두사)로 시작하는 로그 그룹 이름이 있는 여러 로그 그룹에 적용되는 트랜스포머 정책을 생성합니다. 한 리전에 최대 20개의 계정 수준 트랜스포머 정책을 보유할 수 있습니다.

동일한 접두사를 사용하거나 하나의 접두사가 다른 접두사에 포함된 두 개의 트랜스포머 정책을 동일한 리전에 생성할 수 없습니다. 예를 들어, 문자열 접두사 `/aws/lambda`에 대해 하나의 트랜스포머 정책을 생성하는 경우 접두사 `/aws`로 다른 트랜스포머 정책을 생성할 수 없습니다. 하지만 `/aws/lambda`용 트랜스포머 하나와 `/aws/waf`용 트랜스포머 하나가 있을 수 있습니다.

계정 수준 트랜스포머 정책을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 설정을 선택한 다음, 로그 탭을 선택합니다.
3. 계정의 트랜스포머 정책 섹션에서 트랜스포머 정책 생성을 선택합니다.
4. 트랜스포머 정책 이름에 새 정책의 이름을 입력합니다.
5. 로그 그룹 선택에서 다음 중 하나를 수행합니다.
   - 트랜스포머 정책을 계정의 모든 표준 클래스 로그 그룹에 적용하려면 모든 표준 로그 그룹을 선택합니다.
   - 접두사 매칭별 로그 그룹을 선택하여 이름이 모두 동일한 문자열로 시작하는 로그 그룹의 하위 집합에 정책을 적용합니다. 그런 다음 선택 기준에 이러한 로그 그룹의 접두사를 입력합니다.
6. 구문 분석기 선택 영역에서 구문 분석기를 사용하여 트랜스포머에 포함할 구문 분석기를 선택합니다.

   AWS 벤딩된 로그 유형에 대해 사전 구성된 구문 분석기인 경우 해당 로그에 대한 구성을 지정할 필요가 없습니다.

   다른 구문 분석기인 경우 구성을 지정해야 합니다. 자세한 내용은 구성 가능한 구문 분석기 유형 프로세서에서 해당 프로세서에 대한 정보를 참조하세요.
7. 다른 프로세서를 추가하려면 프로세서 선택을 선택합니다. 그런 다음 프로세서 상자에서 원하는 프로세서를 선택하고 구성 파라미터를 입력합니다.

   프로세서는 트랜스포머에 추가하는 순서대로 로그 이벤트에서 작동합니다.
8. (선택 사항) 프로세서를 추가하려면 + 프로세서를 선택하고 이전 단계를 반복합니다.
9. (선택 사항) 샘플 로그 이벤트에서 지금까지 빌드한 트랜스포머를 언제든지 테스트할 수 있습니다. 이렇게 하려면 트랜스포머 미리 보기 섹션에서 다음 중 하나를 수행합니다.
   - 로그 그룹 선택에서 최대 5개의 로그 그룹을 선택한 다음, 최신 로그 이벤트 로드를 선택합니다. 그런 다음 트랜스포머 테스트를 선택합니다.
   - 로그 이벤트를 샘플 로그 이벤트에 직접 복사한 다음, 트랜스포머 테스트를 선택합니다.

그러면 로그의 변환된 버전이 나타납니다.

10. 프로세서 추가를 마치고 샘플 로그의 테스트에 만족하면 저장을 선택합니다.
11. 모두 마쳤으면 생성을 선택합니다.

계정 수준 트랜스포머 정책 편집 또는 삭제

이 섹션의 단계를 사용하여 계정 수준 트랜스포머 정책을 편집하거나 삭제합니다.

계정 수준 트랜스포머 정책을 편집하거나 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 설정을 선택한 다음, 로그 탭을 선택합니다.
3. 트랜스포머 계정 정책 섹션에서 관리를 선택합니다.
4. 관리하려는 트랜스포머 정책 옆에 있는 버튼을 선택한 다음 편집 또는 삭제를 선택합니다.

   정책을 편집하는 경우 옵션을 보려면 구성 가능한 구문 분석기 유형 프로세서의 5~11단계를 참조하세요.

처음부터 로그 그룹 수준 로그 트랜스포머 생성

다음 단계에 따라 로그 그룹 수준 트랜스포머를 처음부터 생성합니다.

콘솔을 사용하여 로그 그룹에 대한 로그 트랜스포머를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 트랜스포머를 생성하려는 로그 그룹을 선택합니다.
4. 트랜스포머 탭을 선택합니다. 탭 목록을 보려면 오른쪽으로 스크롤해야 할 수 있습니다.
5. 트랜스포머 생성을 선택합니다.
6. 구문 분석기 선택 상자에서 트랜스포머에 포함할 구문 분석기를 선택합니다.

   AWS 벤딩된 로그 유형에 대해 사전 구성된 구문 분석기인 경우 해당 로그에 대한 구성을 지정할 필요가 없습니다.

   다른 구문 분석기인 경우 구성을 지정해야 합니다. 자세한 내용은 구성 가능한 구문 분석기 유형 프로세서에서 해당 프로세서에 대한 정보를 참조하세요.

7. 다른 프로세서를 추가하려면 + 프로세서 추가를 선택합니다. 그런 다음 프로세서 선택 상자에서 원하는 프로세서를 선택하고 구성 파라미터를 입력합니다.

   프로세서는 트랜스포머에 추가하는 순서대로 로그 이벤트에서 작동합니다.

8. (선택 사항) 샘플 로그 이벤트에서 지금까지 빌드한 트랜스포머를 언제든지 테스트할 수 있습니다. 그렇게 하려면 다음을 수행합니다.

   - 변환 미리 보기 섹션에서 샘플 로그 로드를 선택하여 이 트랜스포머가 사용되는 로그 그룹에서 샘플 로그 이벤트를 로드하거나 로그 이벤트를 텍스트 상자에 붙여넣습니다.

     트랜스포머 테스트를 선택합니다. 로그의 변환된 버전이 나타납니다.

9. 프로세서 추가를 마치고 샘플 로그의 테스트에 만족하면 저장을 선택합니다.

AWS CLI 를 사용하여 처음부터 로그 변환기를 생성하려면

- `aws logs put-transformer` 명령을 사용합니다. `parseJSON`을 첫 번째 프로세서로 사용할 때는 `@message`를 소스 필드로 사용하여 전체 로그 이벤트를 구문 분석해야 합니다. 초기 JSON 구문 분석 후 후속 프로세서에서 특정 필드를 조작할 수 있습니다. 다음은 `parseJSON` 및 `addKeys` 프로세서를 포함하는 트랜스포머를 생성하는 예제입니다.

```
 aws logs put-transformer \
  --transformer-config '[{"parseJSON":{"source":"@message"}},{"addKeys":
{"entries":[{"key":"metadata.transformed_in","value":"CloudWatchLogs"},
{"key":"feature","value":"Transformation"}]}},{"trimString":{"withKeys":
["status"]}}]' \
  --log-group-identifier my-log-group-name
```

기존 트랜스포머를 복사하여 로그 그룹 수준 트랜스포머 생성

콘솔을 사용하여 기존 트랜스포머의 JSON 구성을 복사할 수 있습니다. 그런 다음 해당 코드를 사용하여를 사용하여 동일한 변환기를 생성 AWS CLI하거나 먼저 구성을 수정할 수 있습니다.

기존 로그 트랜스포머를 복사하여 로그 트랜스포머를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 복사하려는 트랜스포머가 있는 로그 그룹을 선택합니다.

4. 변환 탭을 선택합니다. 탭 목록을 보려면 오른쪽으로 스크롤해야 할 수 있습니다.
5. 트랜스포머 관리를 선택합니다.
6. 트랜스포머 복사를 선택합니다. 그러면 트랜스포머 JSON이 클립보드에 복사됩니다.
7. 파일을 생성하고 트랜스포머 구성에 붙여 넣습니다. 이 예제에서는 `CopiedTransformer.json` 파일을 직접 호출합니다
8. AWS CLI 를 사용하여 해당 구성으로 새 변환기를 생성합니다.

```
aws logs put-transformer --log-group-identifier my-log-group-name \
--transformer-config file://CopiedTransformer.json
```

로그 그룹 수준 트랜스포머 편집

다음 단계에 따라 기존 로그 트랜스포머를 편집합니다.

로그 트랜스포머를 편집하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 편집하려는 트랜스포머가 있는 로그 그룹을 선택합니다.
4. 변환 탭을 선택합니다. 탭 목록을 보려면 오른쪽으로 스크롤해야 할 수 있습니다.
5. 트랜스포머 관리를 선택합니다.
6. 구문 분석기 및 프로세서 섹션에서 변경합니다.
7. 다른 프로세서를 추가하려면 + 프로세서 추가를 선택합니다. 그런 다음 프로세서 상자에서 원하는 프로세서를 선택하고 구성 파라미터를 입력합니다.

   프로세서는 트랜스포머에 추가하는 순서대로 로그 이벤트에서 작동합니다.
8. (선택 사항) 샘플 로그 이벤트에서 지금까지 빌드한 트랜스포머를 언제든지 테스트할 수 있습니다. 그렇게 하려면 다음을 수행합니다.

   - 변환 미리 보기 섹션에서 샘플 로그 로드를 선택하여 이 트랜스포머가 사용되는 로그 그룹에서 샘플 로그 이벤트를 로드하거나 로그 이벤트를 텍스트 상자에 붙여넣습니다.

     변환 테스트를 선택합니다. 로그의 변환된 버전이 나타납니다.
9. 프로세서 추가를 마치고 샘플 로그의 테스트에 만족하면 저장을 선택합니다.

로그 그룹 수준 트랜스포머 삭제

다음 단계에 따라 로그 트랜스포머를 삭제합니다.

로그 트랜스포머를 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그, 로그 그룹을 선택합니다.
3. 편집하려는 트랜스포머가 있는 로그 그룹을 선택합니다.
4. 변환 탭을 선택합니다. 탭 목록을 보려면 오른쪽으로 스크롤해야 할 수 있습니다.
5. 삭제를 선택합니다.
6. 확인 상자에서 정책 삭제를 선택합니다.

구성 가능한 구문 분석기 유형 프로세서

이 섹션에는 로그 이벤트 변환기에서 사용할 수 있는 구성 가능한 데이터 구문 분석기 프로세서에 대한 정보가 포함되어 있습니다.

목차

- parseJSON
- grok
  - Grok 예제
    - 예제 1: grok를 사용하여 비정형 로그에서 필드 추출
    - 예제 2: parseJSON과 함께 grok를 사용하여 JSON 로그 이벤트에서 필드 추출
    - 예제 3: FIELD_NAME에 점 주석이 있는 Grok 패턴
  - 지원되는 grok 패턴
    - 일반적인 로그 형식 예제
      - Apache 로그 예제
      - NGINX 로그 예제
      - Syslog 프로토콜(RFC 5424) 로그 예제
- csv
- parseKeyValue

parseJSON

parseJSON 프로세서는 JSON 로그 이벤트를 구문 분석하며, 추출된 JSON 키-값 쌍을 대상 아래에 삽입합니다. 대상을 지정하지 않으면 프로세서가 루트 노드 아래에 키-값 쌍을 배치합니다. parseJSON을 첫 번째 프로세서로 사용할 때는 @message를 소스 필드로 사용하여 전체 로그 이벤트를 구문 분석해야 합니다. 초기 JSON 구문 분석 후 후속 프로세서에서 특정 필드를 조작할 수 있습니다.

원래 @message 내용은 변경되지 않으며 새 키가 메시지에 추가됩니다.

필드	설명	필수	기본값	제한 사항
source	구문 분석할 로그 이벤트의 필드 경로. 점 표기법을 사용하여 하위 필드에 액세스합니다. 예: store.book	아니요	@message	최대 길이: 128 최대 중첩 키 깊이: 3
대상	구문 분석된 JSON의 대상 필드	아니요	Parent JSON node	최대 길이: 128 최대 중첩 키 깊이: 3

예제

수집된 로그 이벤트가 다음과 같다고 가정해 보겠습니다.

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

이 parseJSON 프로세서가 있는 경우:

```
[
   {
        "parseJSON": {
            "destination": "new_key"
        }
   }
```

```
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
    "new_key": {
        "outer_key": {
            "inner_key": "inner_value"
        }
    }
}
```

grok

grok 프로세서를 사용하여 패턴 매칭을 사용하여 비정형 데이터를 구문 분석하고 구조화합니다. 이 프로세서는 로그 메시지에서 필드를 추출할 수도 있습니다.

필드	설명	필수	기본값	제한 사항	참고 사항
source	Grok 매칭을 적용할 필드의 경로	아니요	`@messag`	최대 길이: 128 최대 중첩 키 깊이: 3	
match	로그 이벤트와 매칭할 grok 패턴	예		최대 길이: 512 최대 grok 패턴: 20 일부 grok 패턴 유형에는 개별 사용량 제한이 있습니다. {URI, URIPARAM, URIPATHPARAM, SPACE, DATA, GREEDYDATA, GREEDYDATA_MULTILINE} 패턴의 조합은 최대 5	지원되는 모든 Grok 패턴 보기

필드	설명	필수	기본값	제한 사항	참고 사항
				회까지 사용할 수 있습니다. Grok 패턴은 유형 변환을 지원하지 않습니다. 일반적인 로그 형식 패턴(APACHE_ACCESS_LOG, NGINX_ACCESS_LOG, SYSLOG5424)의 경우 일반적인 로그 패턴 뒤에는 DATA, GREEDYDATA 또는 GREEDYDATA_MULTILINE 패턴만 포함될 수 있습니다.	

Grok 패턴의 구조

지원되는 grok 패턴 구조는 다음과 같습니다.

```
%{PATTERN_NAME:FIELD_NAME}
```

- PATTERN_NAME: 특정 유형의 데이터와 일치시키기 위해 미리 정의된 정규식을 나타냅니다. 사전 정의된 grok 패턴만 지원됩니다. 사용자 지정 패턴 생성은 허용되지 않습니다.
- FIELD_NAME: 추출된 값에 이름을 할당합니다. `FIELD_NAME`은 선택 사항이지만 이 값을 지정하지 않으면 추출된 데이터가 변환된 로그 이벤트에서 삭제됩니다. `FIELD_NAME`은 점 표기법(예: 'parent.child')을 사용하는 경우 JSON 경로로 간주됩니다.
- 유형 변환: 명시적 유형 변환은 지원되지 않습니다. TypeConverter 프로세서를 사용하여 grok에서 추출한 값의 데이터타입을 변환합니다.

보다 복잡한 매칭 표현식을 생성하려면 여러 grok 패턴을 결합하면 됩니다. 로그 이벤트를 매칭하도록 최대 20개의 grok 패턴을 결합할 수 있습니다. 예를 들어, 이 `%{NUMBER:timestamp} [%{NUMBER:db} %{IP:client_ip}:%{NUMBER:client_port}] %{GREEDYDATA:data}` 패턴 조합을 사용하여 다음과 같은 Redis 슬로우 로그 항목에서 필드를 추출할 수 있습니다.

```
1629860738.123456 [0 127.0.0.1:6379] "SET" "key1" "value1"
```

Grok 예제

예제 1: grok를 사용하여 비정형 로그에서 필드 추출

샘플 로그:

```
293750 server-01.internal-network.local OK "[Thread-000] token generated"
```

사용된 트랜스포머:

```
[
     {
         "grok": {
             "match": "%{NUMBER:version} %{HOSTNAME:hostname} %{NOTSPACE:status}
 %{QUOTEDSTRING:logMsg}"
         }
    }
]
```

출력:

```
{
  "version": "293750",
  "hostname": "server-01.internal-network.local",
  "status": "OK",
  "logMsg": "[Thread-000] token generated"
}
```

샘플 로그:

```
23/Nov/2024:10:25:15 -0900 172.16.0.1 200
```

사용된 트랜스포머:

```
[
    {
        "grok": {
            "match": "%{HTTPDATE:timestamp} %{IPORHOST:clientip}
 %{NUMBER:response_status}"
        }
    }
]
```

출력:

```
{
  "timestamp": "23/Nov/2024:10:25:15 -0900",
  "clientip": "172.16.0.1",
  "response_status": "200"
}
```

예제 2: parseJSON과 함께 grok를 사용하여 JSON 로그 이벤트에서 필드 추출

샘플 로그:

```
{
    "timestamp": "2024-11-23T16:03:12Z",
    "level": "ERROR",
    "logMsg": "GET /page.html HTTP/1.1"
}
```

사용된 트랜스포머:

```
[
     {
        "parseJSON": {}
    },
    {
         "grok": {
            "source": "logMsg",
             "match": "%{WORD:http_method} %{NOTSPACE:request} HTTP/
%{NUMBER:http_version}"
         }
    }
]
```

출력:

```
{
  "timestamp": "2024-11-23T16:03:12Z",
  "level": "ERROR",
  "logMsg": "GET /page.html HTTP/1.1",
  "http_method": "GET",
  "request": "/page.html",
  "http_version": "1.1"
}
```

예제 3: FIELD_NAME에 점 주석이 있는 Grok 패턴

샘플 로그:

```
192.168.1.1 GET /index.html?param=value 200 1234
```

사용된 트랜스포머:

```
[
    {
        "grok": {
            "match": "%{IP:client.ip} %{WORD:method} %{URIPATHPARAM:request.uri}
 %{NUMBER:response.status} %{NUMBER:response.bytes}"
        }
    }
]
```

출력:

```
{
  "client": {
    "ip": "192.168.1.1"
  },
  "method": "GET",
  "request": {
    "uri": "/index.html?param=value"
  },
  "response": {
    "status": "200",
    "bytes": "1234"
  }
```

```
}
```

지원되는 grok 패턴

다음 표에는 grok 프로세서에서 지원하는 패턴이 나열되어 있습니다.

일반 grok 패턴

Grok 패턴	설명	최대 패턴 제한	예제
사용자 이름 또는 사용자	소문자(a~z), 대문자(A~Z), 숫자(0~9), 점(.), 밑줄(_) 또는 하이픈(-)을 포함할 수 있는 하나 이상의 문자와 일치합니다.	20	입력: `user123.name-TEST` 패턴: `%{USERNAME:name}` 출력: `{"name": "user123.name-TEST"}`
INT	선택 사항인 더하기 또는 빼기 기호 뒤에 하나 이상의 숫자가 오는 것과 일치합니다.	20	입력: `-456` 패턴: `%{INT:num}` 출력: `{"num": "-456"}`
BASE10NUM	부호 및 소수점 옵션이 있는 정수 또는 부동 소수점 숫자와 일치합니다.	20	입력: `-0.67` 패턴: `%{BASE10NUM:num}` 출력: `{"num": "-0.67"}`
BASE16NUM	10진수 및 16진수와 선택적 기호(+ 또는 -) 및 선택적 0x 접두사와 일치합니다.	20	입력: `+0xA1B2` 패턴: `%{BASE16NUM:num}` 출력: `{"num": "+0xA1B2"}`
POSINT	하나 이상의 숫자(1~9 다음에 0~9)로 구성된 앞에 0이 없는 정수와 일치합니다.	20	입력: `123` 패턴: `%{POSINT:num}` 출력: `{"num": "123"}`

Grok 패턴	설명	최대 패턴 제한	예제
NONNEGINT	0과 앞에 0이 있는 숫자를 포함하여 모든 정수(0~9의 숫자로 구성)와 일치합니다.	20	입력: `007` 패턴: `%{NONNEGINT:num}` 출력: `{"num": "007"}`
WORD	문자, 숫자 및 밑줄을 포함하여 하나 이상의 단어 문자(\w)로 구성된 전체 단어와 일치합니다..	20	입력: `user_123` 패턴: `%{WORD:user}` 출력: `{"user": "user_123"}`
NOTSPACE	공백이 없는 하나 이상의 문자와 일치합니다.	5	입력: `hello_world123` 패턴: `%{NOTSPACE:msg}` 출력: `{"msg": "hello_world123"}`
SPACE	0개 이상의 공백 문자와 일치합니다.	5	입력: `" "` 패턴: `%{SPACE:extra}` 출력: `{"extra": " "}`
DATA	모든 문자(줄 바꿈 제외)와 0회 이상 최소 일치합니다.	5	입력: `abc def ghi` 패턴: `%{DATA:x} %{DATA:y}` 출력: `{"x": "abc", "y": "def ghi"}`

Grok 패턴	설명	최대 패턴 제한	예제
GREEDYDATA	모든 문자(줄 바꿈 제외)와 0회 이상 최대 일치합니다.	5	입력: `abc def ghi` 패턴: `%{GREEDYDATA:x} %{GREEDYDATA:y}` 출력: `{"x": "abc def", "y": "ghi"}`

GREEDYDATA_MULTILINE	모든 문자(줄 바꿈 포함)와 0회 이상 최대 일치합니다.	1	입력:
			
			`abc`
			
			`def`
			
			`ghi`
			
			패턴: `%{GREEDYDATA_MULTILINE:data}`
			
			출력: `{"data": "abc\ndef\nghi"}`

QUOTEDSTRING	이스케이프 처리된 문자가 포함된 따옴표(작은따옴표 또는 큰따옴표)로 묶인 문자열과 일치합니다.	20	입력: `"Hello, world!"` 패턴: `%{QUOTEDSTRING:msg}` 출력: `{"msg": "Hello, world!"}`

Grok 패턴	설명	최대 패턴 제한	예제
UUID	표준 UUID 형식과 일치합니다. 즉, 8개의 16진수 문자 뒤에 4개의 16진수 문자로 구성된 3개의 그룹이 오고 12개의 16진수 문자로 끝나며, 모두 하이픈으로 구분됩니다.	20	입력: `550e8400-e29b-41d4-a716-446655440000` 패턴: `%{UUID:id}` 출력: `{"id": "550e8400-e29b-41d4-a716-446655440000"}`
urn	URN(Uniform Resource Name) 구문과 일치합니다.	20	입력: `urn:isbn:0451450523` 패턴: `%{URN:urn}` 출력: `{"urn": "urn:isbn:0451450523"}`

AWS grok 패턴

패턴	설명	최대 패턴 제한	예제
ARN	파티션(`aws`, 또는 `aws-us-gov`), 서비스`aws-cn`, 리전, 계정 ID 및 슬래시로 구분된 최대 5개의 계층적 리소스 식별자를 캡처하여 AWS Amazon 리소스 이름(ARNs)과 일치합니다. 콜론 간에 누락된 정보가 있는 ARN을 매칭하지 않습니다.	5	입력: `arn:aws:iam:us-east-1:123456789012:user/johndoe` 패턴: `%{ARN:arn}` 출력: `{"arn": "arn:aws:iam:us-east-1:123456789012:user/johndoe"}`

네트워킹 grok 패턴

Grok 패턴	설명	최대 패턴 제한	예제
CISCOMAC	4-4-4 16진수 형식의 MAC 주소와 일치합니다.	20	입력: `0123.4567.89AB` 패턴: `%{CISCOMAC:MacAddress}` 출력: `{"MacAddress": "0123.4567.89AB"}`
WINDOWSMAC	하이픈이 포함된 16진수 형식의 MAC 주소와 일치합니다.	20	입력: `01-23-45-67-89-AB` 패턴: `%{WINDOWSMAC:MacAddress}` 출력: `{"MacAddress": "01-23-45-67-89-AB"}`
COMMONMAC	콜론이 포함된 16진수 형식의 MAC 주소와 일치합니다.	20	입력: `01:23:45:67:89:AB` 패턴: `%{COMMONMAC:MacAddress}` 출력: `{"MacAddress": "01:23:45:67:89:AB"}`
MAC	CISCOMAC, WINDOWSMAC 또는 COMMONMAC grok 패턴 중 하나와 일치합니다.	20	입력: `01:23:45:67:89:AB` 패턴: `%{MAC:m1}` 출력: `{"m1":"01:23:45:67:89:AB"}`
IPV6	압축된 양식 및 IPv4 매핑된 IPv6 주소를 포함한 IPv6 주소와 일치합니다.	5	입력: `2001:db8:3333:4444:5555:6666:7777:8888` 패턴: `%{IPV6:ip}`

Grok 패턴	설명	최대 패턴 제한	예제
			출력: `{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}`
IPV4	IPv4 주소와 일치합니다.	20	입력: `192.168.0.1` 패턴: `%{IPV4:ip}` 출력: `{"ip": "192.168.0.1"}`
IP	%{IPv6}에서 지원하는 IPv6 주소 또는 %{IPv4}에서 지원하는 IPv4 주소 중 하나와 일치합니다.	5	입력: `192.168.0.1` 패턴: `%{IP:ip}` 출력: `{"ip": "192.168.0.1"}`
HOSTNAME 또는 HOST	하위 도메인을 포함한 도메인 이름과 일치합니다.	5	입력: `server-01.internal-network.local` 패턴: `%{HOST:host}` 출력: `{"host": "server-01.internal-network.local"}`
IPORHOST	호스트 이름 또는 IP 주소와 일치합니다.	5	입력: `2001:db8:3333:4444:5555:6666:7777:8888` 패턴: `%{IPORHOST:ip}` 출력: `{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}`

Grok 패턴	설명	최대 패턴 제한	예제
HOSTPORT	%{IPORHOST} 패턴에서 지원하는 IP 주소 또는 호스트 이름과 일치하고, 그 뒤에 콜론과 포트 번호가 오며, 출력에서 포트를 'PORT'로 캡처합니다.	5	입력: `192.168.1.1:8080` 패턴: `%{HOSTPORT:ip}` 출력: `{"ip":"192.168.1.1:8080","PORT":"8080"}`
URIHOST	%{IPORHOST} 패턴에서 지원하는 IP 주소 또는 호스트 이름과 일치하고, 선택적으로 그 뒤에 콜론과 포트 번호가 오고, 포트가 있는 경우 'port'로 캡처합니다.	5	입력: `example.com:443 10.0.0.1` 패턴: `%{URIHOST:host} %{URIHOST:ip}` 출력: `{"host":"example.com:443","port":"443","ip":"10.0.0.1"}`

경로 grok 패턴

Grok 패턴	설명	최대 패턴 제한	예제
UNIXPATH	쿼리 파라미터를 포함할 수 있는 URL 경로와 일치합니다.	20	입력: `/search?q=regex` 패턴: `%{UNIXPATH:path}` 출력: `{"path":"/search?q=regex"}`
WINPATH	Windows 파일 경로와 일치합니다.	5	입력: `C:\Users\John\Documents\file.txt` 패턴: `%{WINPATH:path}`

Grok 패턴	설명	최대 패턴 제한	예제
			출력: `{"path": "C:\\Users\\John\\Documents\\file.txt"}`
PATH	URL 또는 Windows 파일 경로와 일치합니다.	5	입력: `/search?q=regex` 패턴: `%{PATH:path}` 출력: `{"path":"/search?q=regex"}`
TTY	터미널 및 가상 터미널의 Unix 디바이스 경로와 일치합니다.	20	입력: `/dev/tty1` 패턴: `%{TTY:path}` 출력: `{"path":"/dev/tty1"}`
URIPROTO	문자와 일치하며, 선택적으로 그 뒤에 더하기(+) 문자와 추가 문자 또는 더하기(+) 문자가 옵니다.	20	입력: `web+transformer` 패턴: `%{URIPROTO:protocol}` 출력: `{"protocol":"web+transformer"}`
URIPATH	URI의 경로 구성 요소와 일치합니다.	20	입력: `/category/sub-category/product_name` 패턴: `%{URIPATH:path}` 출력: `{"path":"/category/sub-category/product_name"}`

Grok 패턴	설명	최대 패턴 제한	예제
URIPARAM	URL 쿼리 파라미터와 일치합니다.	5	입력: `?param1=value1&param2=value2` 패턴: `%{URIPARAM:url}` 출력: `{"url":"?param1=value1&param2=value2"}`
URIPATHPARAM	URI 경로와 일치하며, 선택적으로 그 뒤에 쿼리 파라미터가 옵니다.	5	입력: `/category/sub-category/product?id=12345&color=red` 패턴: `%{URIPATHPARAM:path}` 출력: `{"path":"/category/sub-category/product?id=12345&color=red"}`
URI	전체 URI와 일치합니다.	5	입력: `https://user:password@example.com/path/to/resource?param1=value1&param2=value2` 패턴: `%{URI:uri}` 출력: `{"path":"https://user:password@example.com/path/to/resource?param1=value1&param2=value2"}`

날짜/시간 Grok 패턴

Grok 패턴	설명	최대 패턴 제한	예제
MONTH	전체 또는 축약된 영어 월 이름과 전체 단어로 일치합니다.	20	입력: `Jan` 패턴: `%{MONTH:month}` 출력: `{"month": "Jan"}` 입력: `January` 패턴: `%{MONTH:month}` 출력: `{"month": "January"}`
MONTHNUM	1에서 12까지의 월 수와 일치하며, 한 자릿수 월의 경우 선택적으로 앞에 0이 표시됩니다.	20	입력: `5` 패턴: `%{MONTHNUM:month}` 출력: `{"month":"5"}` 입력: `05` 패턴: `%{MONTHNUM:month}` 출력: `{"month":"05"}`
MONTHNUM	01에서 12까지의 두 자리 숫자와 일치합니다.	20	입력: `05` 패턴: `%{MONTHNUM2:month}` 출력: `{"month":"05"}`

Grok 패턴	설명	최대 패턴 제한	예제
MONTHDAY	1에서 31까지의 일과 일치하며, 선택적으로 앞에 0이 표시됩니다.	20	입력: `31` 패턴: `%{MONTHDAY:monthDay}` 출력: `{"monthDay":"31"}`
YEAR	2자리 또는 4자리 연도와 일치합니다.	20	입력: `2024` 패턴: `%{YEAR:year}` 출력: `{"year":"2024"}` 입력: `24` 패턴: `%{YEAR:year}` 출력: `{"year":"24"}`
DAY	전체 또는 축약된 요일 이름과 일치합니다.	20	입력: `Tuesday` 패턴: `%{DAY:day}` 출력: `{"day":"Tuesday"}`
시간	24시간 형식의 시간(0~23)과 일치하며, 선택적으로 앞에 영(0)이 표시됩니다.	20	입력: `22` 패턴: `%{HOUR:hour}` 출력: `{"hour":"22"}`
분	분(00~59)과 일치합니다.	20	입력: `59` 패턴: `%{MINUTE:min}` 출력: `{"min":"59"}`

Grok 패턴	설명	최대 패턴 제한	예제
SECOND	초((0)0~60)를 나타내는 숫자와 일치하며, 선택적으로 그 뒤에 소수점 또는 콜론과 소수 분을 나타내는 하나 이상의 숫자가 옵니다.	20	입력: `3` 패턴: `%{SECOND:second}` 출력: `{"second":"3"}` 입력: `30.5` 패턴: `%{SECOND:minSec}` 출력: `{"minSec":"30.5"}` 입력: `30:5` 패턴: `%{SECOND:minSec}` 출력: `{"minSec":"30:5"}`
TIME	시간, 분 및 초가 (H)H:mm:(s)s 형식인 시간 형식과 일치합니다. 초에는 윤초(0)0~60이 포함됩니다.	20	입력: `09:45:32` 패턴: `%{TIME:time}` 출력: `{"time":"09:45:32"}`

Grok 패턴	설명	최대 패턴 제한	예제
DATE_US	(M)M/(d)d/(yy)yy 또는 (M)M-(d)d-(yy)yy 형식의 날짜와 일치합니다.	20	입력: `11/23/2024` 패턴: `%{DATE_US:date}` 출력: `{"date":"11/23/2024"}` 입력: `1-01-24` 패턴: `%{DATE_US:date}` 출력: `{"date":"1-01-24"}`
DATE_EU	(d)d/(M)M/(yy)yy, (d)d-(M)M-(yy)yy 또는 (d)d.(M)M.(yy)yy 형식의 날짜와 일치합니다.	20	입력: `23/11/2024` 패턴: `%{DATE_EU:date}` 출력: `{"date":"23/11/2024"}` 입력: `1.01.24` 패턴: `%{DATE_EU:date}` 출력: `{"date":"1.01.24"}`

Grok 패턴	설명	최대 패턴 제한	예제
ISO8601_TIMEZONE	UTC 오프셋 'Z' 또는 선택적 콜론이 포함된 시간대 오프셋과 [+-](H)H(:)mm 형식으로 일치합니다.	20	입력: `+05:30` 패턴: `%{ISO8601_TIMEZONE:tz}` 출력: `{"tz":"+05:30"}` 입력: `-530` 패턴: `%{ISO8601_TIMEZONE:tz}` 출력: `{"tz":"-530"}` 입력: `Z` 패턴: `%{ISO8601_TIMEZONE:tz}` 출력: `{"tz":"Z"}`
ISO8601_SECOND	초((0)0~60)를 나타내는 숫자와 일치하며, 선택적으로 그 뒤에 소수점 또는 콜론과 소수 초를 나타내는 하나 이상의 숫자가 옵니다.	20	입력: `60` 패턴: `%{ISO8601_SECOND:second}` 출력: `{"second":"60"}`

Grok 패턴	설명	최대 패턴 제한	예제
TIMESTAMP_ISO8601	선택적 초 및 시간대가 포함된 ISO8601 날짜/시간 형식 (yy)yy-(M)M-(d)dT(H)H:mm:((s)s)(Z|[+-](H)H:mm)과 일치합니다.	20	입력: 2023-05-15T14:30:00+05:30 패턴: %{TIMESTAMP_ISO8601:timestamp} 출력: {"timestamp":"2023-05-15T14:30:00+05:30"} 입력: 23-5-1T1:25+5:30 패턴: %{TIMESTAMP_ISO8601:timestamp} 출력: {"timestamp":"23-5-1T1:25+5:30"} 입력: 23-5-1T1:25Z 패턴: %{TIMESTAMP_ISO8601:timestamp} 출력: {"timestamp":"23-5-1T1:25Z"}

Grok 패턴	설명	최대 패턴 제한	예제
DATE	%{DATE_US}를 사용하는 미국 형식 또는 %{DATE_EU}를 사용하는 EU 형식의 날짜와 일치합니다.	20	입력: `11/29/2024` 패턴: `%{DATE:date}` 출력: `{"date":"11/29/2024"}` 입력: `29.11.2024` 패턴: `%{DATE:date}` 출력: `{"date":"29.11.2024"}`
DATESTAMP	공백 또는 하이픈으로 구분된 %{DATE} 뒤에 %{TIME}이 오는 패턴과 일치합니다.	20	입력: `29-11-2024 14:30:00` 패턴: `%{DATESTAMP:dateTime}` 출력: `{"dateTime":"29-11-2024 14:30:00"}`
TZ	공통 시간대 약어(PST, PDT, MST, MDT, CST CDT, EST, EDT, UTC)와 일치합니다.	20	입력: `PDT` 패턴: `%{TZ:tz}` 출력: `{"tz":"PDT"}`

Grok 패턴	설명	최대 패턴 제한	예제
DATESTAMP_RFC822	Day MonthName (D)D (YY)YY (H)H:mm:(s)s Timezone 형식의 날짜/시간과 일치합니다.	20	입력: `Monday Jan 5 23 1:30:00 CDT` 패턴: `%{DATESTAMP_RFC822:dateTime}` 출력: `{"dateTime":"Monday Jan 5 23 1:30:00 CDT"}` 입력: `Mon January 15 2023 14:30:00 PST` 패턴: `%{DATESTAMP_RFC822:dateTime}` 출력: `{"dateTime":"Mon January 15 2023 14:30:00 PST"}`

Grok 패턴	설명	최대 패턴 제한	예제
DATESTAMP_RFC2822	Day, (d)d MonthName (yy)yy (H)H:mm:(s)s Z|[+-](H)H:mm 형식의 RFC2822 날짜/시간과 일치합니다.	20	입력: `Mon, 15 May 2023 14:30:00 +0530` 패턴: `%{DATESTAMP_RFC2822:dateTime}` 출력: `{"dateTime":"Mon, 15 May 2023 14:30:00 +0530"}` 입력: `Monday, 15 Jan 23 14:30:00 Z` 패턴: `%{DATESTAMP_RFC2822:dateTime}` 출력: `{"dateTime":"Monday, 15 Jan 23 14:30:00 Z"}`
DATESTAMP_OTHER	Day MonthName (d)d (H)H:mm:(s)s Timezone (yy)yy 형식의 날짜/시간과 일치합니다.	20	입력: `Mon May 15 14:30:00 PST 2023` 패턴: `%{DATESTAMP_OTHER:dateTime}` 출력: `{"dateTime":"Mon May 15 14:30:00 PST 2023"}`

Grok 패턴	설명	최대 패턴 제한	예제
DATESTAMP_EVENTLOG	구분자가 없는 압축 날짜/시간((yy)yyMM(d)d(H)Hmm(s)s)과 일치합니다.	20	입력: 20230515143000 패턴: %{DATESTAMP_EVENTLOG:dateTime} 출력: {"dateTime":"20230515143000"}

로그 grok 패턴

Grok 패턴	설명	최대 패턴 제한	예제
LOGLEVEL	Alert/ALERT , Trace/TRACE , Debug/DEBUG , Notice/NOTICE , Info/INFO , Warn/Warning/WARN/WARNING , Err/Error/ERR/ERROR , Crit/Critical/CRIT/CRITICAL , Fatal/FATAL , Severe/SEVERE , Emerg/Emergency/EMERG/EMERGENCY 를 포함하여 다양한 대문자 및 약어의 표준 로그 수준과 일치합니다.	20	입력: INFO 패턴: %{LOGLEVEL:logLevel} 출력: {"logLevel":"INFO"}
HTTPDATE	로그 파일에 자주 사용되는 날짜/시간 형식과 일치합니다. 형식: (d)d/MonthName/(yy)yy:(H)H:mm:(s)s Timezone	20	입력: 23/Nov/2024:14:30:00 +0640 패턴: %{HTTPDATE:date}

Grok 패턴	설명	최대 패턴 제한	예제
	MonthName: 전체 또는 축약된 영어 월 이름과 일치합니다(예: 'Jan' 또는 'January'). Timezone: %{INT} grok 패턴과 일치합니다.		출력: `{"date":"23/Nov/2024:14:30:00 +0640"}`
SYSLOGTIMESTAMP	MonthName (d)d (H)H:mm:(s)s 날짜 형식과 일치합니다. MonthName : 전체 또는 축약된 영어 월 이름과 일치합니다(예: 'Jan' 또는 'January').	20	입력: `Nov 29 14:30:00` 패턴: `%{SYSLOGTIMESTAMP: dateTime}` 출력: `{"dateTime":"Nov 29 14:30:00"}`
PROG	문자, 숫자, 점, 밑줄, 슬래시, 백분율 기호 및 하이픈 문자로 구성된 문자열로 구성된 프로그램 이름과 일치합니다.	20	입력: `user.profile/settings-page` 패턴: `%{PROG:program}` 출력: `{"program":"user.profile/settings-page"}`

Grok 패턴	설명	최대 패턴 제한	예제
SYSLOGPRO G	PROG grok 패턴과 일치하며, 선택적으로 그 뒤에 대괄호로 묶인 프로세스 ID가 옵니다.	20	입력: `user.profile/settings-page[1234]` 패턴: `%{SYSLOGPROG:programWithId}` 출력: `{"programWithId":"user.profile/settings-page[1234]","program":"user.profile/settings-page","pid":"1234"}`
SYSLOGHOS T	%{HOST} 또는 %{IP} 패턴과 일치합니다.	5	입력: `2001:db8:3333:4444:5555:6666:7777:8888` 패턴: `%{SYSLOGHOST:ip}` 출력: `{"ip":"2001:db8:3333:4444:5555:6666:7777:8888"}`

Grok 패턴	설명	최대 패턴 제한	예제
SYSLOGFAC ILITY	십진수 형식의 syslog 우선 순위와 일치합니다. 값은 각진 대괄호(<>)로 묶어야 합니다.	20	입력: <13.6> 패턴: %{SYSLOGF ACILITY:syslog} 출력: {"syslog" :"<13.6>" ,"facilit y":"13"," priority":"6"}

일반적인 로그 grok 패턴

사전 정의된 사용자 지정 grok 패턴을 사용하여 Apache, NGINX 및 Syslog Protocol(RFC 5424) 로그 형식을 매칭할 수 있습니다. 이러한 특정 패턴을 사용하는 경우 매칭 구성의 첫 번째 패턴이어야 하며, 다른 패턴보다 우선할 수 없습니다. 또한 정확히 하나의 데이터로만 추적할 수 있습니다. GREEDYDATA 또는 GREEDYDATA_MULTILINE 패턴.

Grok 패턴	설명	최대 패턴 제한
APACHE_ACCESS_LOG	Apache 액세스 로그와 일치합니다.	1
NGINX_ACCESS_LOG	NGINX 액세스 로그와 일치합니다.	1
SYSLOG5424	Syslog 프로토콜(RFC 5424) 로그와 일치합니다.	1

다음은 이러한 일반적인 로그 형식 패턴을 사용하기 위한 유효한 예제와 잘못된 예제입니다.

```
"%{NGINX_ACCESS_LOG} %{DATA}" // Valid
"%{SYSLOG5424}%{DATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{GREEDYDATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{SYSLOG5424}" // Invalid (multiple common log patterns used)
"%{NGINX_ACCESS_LOG} %{NUMBER:num}" // Invalid (Only GREEDYDATA and DATA patterns are
 supported with common log patterns)
"%{GREEDYDATA:logMsg} %{SYSLOG5424}" // Invalid (GREEDYDATA and DATA patterns are
 supported only after common log patterns)
```

일반적인 로그 형식 예제

Apache 로그 예제

샘플 로그:

```
127.0.0.1 - - [03/Aug/2023:12:34:56 +0000] "GET /page.html HTTP/1.1" 200 1234
```

트랜스포머:

```
[
     {
        "grok": {
            "match": "%{APACHE_ACCESS_LOG}"
        }
    }
]
```

출력:

```
{
    "request": "/page.html",
    "http_method": "GET",
    "status_code": 200,
    "http_version": "1.1",
    "response_size": 1234,
    "remote_host": "127.0.0.1",
    "timestamp": "2023-08-03T12:34:56Z"
}
```

NGINX 로그 예제

샘플 로그:

```
192.168.1.100 - Foo [03/Aug/2023:12:34:56 +0000] "GET /account/login.html HTTP/1.1"
 200 42 "https://www.amazon.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
```

트랜스포머:

```
[
     {
        "grok": {
            "match": "%{NGINX_ACCESS_LOG}"
        }
    }
]
```

출력:

```
{
    "request": "/account/login.html",
    "referrer": "https://www.amazon.com/",
    "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/92.0.4515.131 Safari/537.36",
    "http_method": "GET",
    "status_code": 200,
    "auth_user": "Foo",
    "http_version": "1.1",
    "response_size": 42,
    "remote_host": "192.168.1.100",
    "timestamp": "2023-08-03T12:34:56Z"
}
```

Syslog 프로토콜(RFC 5424) 로그 예제

샘플 로그:

```
<165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47
 [exampleSDID@32473 iut="3" eventSource= "Application" eventID="1011"]
[examplePriority@32473 class="high"]
```

트랜스포머:

```
[
```

```
    {
       "grok": {
           "match": "%{SYSLOG5424}"
       }
    }
]
```

출력:

```
{
  "pri": 165,
  "version": 1,
  "timestamp": "2003-10-11T22:14:15.003Z",
  "hostname": "mymachine.example.com",
  "app": "evntslog",
  "msg_id": "ID47",
  "structured_data": "exampleSDID@32473 iut=\"3\" eventSource= \"Application\" eventID=
\"1011\"",
  "message": "[examplePriority@32473 class=\"high\"]"
}
```

CSV

csv 프로세서는 로그 이벤트의 쉼표로 구분된 값(CSV)을 열로 구문 분석합니다.

필드	설명	필수	기본값	제한 사항
source	구문 분석할 로그 이벤트의 필드 경로	아니요	`@message`	최대 길이: 128 최대 중첩 키 깊이: 3
delimiter	원래 쉼표로 구분된 값 로그 이벤트에서 각 열을 구분하는 데 사용되는 문자	아니요	`,`	최대 길이: 값이 `\t` 또는 `\s`이 아닌 경우 1
quoteCharacter	단일 데이터 열에 대한 텍스트 한정자로 사용되는 문자	아니요	`"`	최대 길이: 1
columns	변환된 로그 이벤트의 열에 사용할 이름의 목록.	아니요	`[column_1, column_2`	최대 CSV 열: 100 최대 길이: 128

필드	설명	필수	기본값	제한 사항
				최대 중첩 키 깊이: 3
대상	변환된 키 값 페어를 넣을 상위 필드입니다.	아니요	Root node	최대 길이: 128 최대 중첩 키 깊이: 3

delimiter을 \t로 설정하면 탭 문자에서 각 열이 구분되며, \t은 단일 공백 문자에서 각 열을 구분합니다.

예제

수집된 로그 이벤트의 일부가 다음과 같다고 가정해 보겠습니다.

```
'Akua Mansa':28:'New York: USA'
```

csv 프로세서만 사용한다고 가정해 보겠습니다.

```
[
     {
        "csv": {
            "delimiter": ":",
            "quoteCharacter": "'"
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "column_1": "Akua Mansa",
  "column_2": "28",
  "column_3": "New York: USA"
}
```

예제 2

수집된 로그 이벤트가 다음과 같다고 가정해 보겠습니다.

```
{
    "timestamp": "2024-11-23T16:03:12Z",
    "type": "user_data",
    "logMsg": "'Akua Mansa':28:'New York: USA'"
}
```

이벤트를 JSON으로 구문 분석하고 열 이름과 대상을 지정하여 csv 프로세서로 JSON 필드를 구문 분석한다고 가정해 보겠습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "csv": {
            "source": "logMsg",
            "delimiter": ":",
            "quoteCharacter": "'",
            "columns":["name","age","location"],
            "destination": "msg"
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
    "timestamp": "2024-11-23T16:03:12Z",
    "logMsg": "'Akua Mansa':28:'New York: USA'",
    "type": "user_data",
    "msg": {
        "name": "Akua Mansa",
        "age": "28",
        "location": "New York: USA"
    }
}
```

parseKeyValue

parseKeyValue 프로세서를 사용하여 지정된 필드를 키-값 쌍으로 구문 분석합니다. 다음 옵션을 사용하여 필드 정보를 구문 분석하도록 프로세서를 사용자 지정할 수 있습니다.

필드	설명	필수	기본값	제한 사항
source	구문 분석할 로그 이벤트의 필드 경로	아니요	`@message`	최대 길이: 128 최대 중첩 키 깊이: 3
대상	추출된 키-값 쌍을 넣을 대상 필드	아니요		최대 길이: 128
fieldDelimiter	원래 로그 이벤트의 키-값 쌍 간에 사용되는 필드 구분 문자 문자열.	아니요	&	최대 길이: 128
keyValueDelimiter	변환된 로그 이벤트의 각 쌍에서 키와 값 사이에 사용할 구분 문자 문자열	아니요	=	최대 길이: 128
nonMatchValue	키-값 쌍이 성공적으로 분할되지 않은 경우 결과의 값 필드에 삽입할 값.	아니요		최대 길이: 128
keyPrefix	변환된 모든 키에 접두사를 추가하려면 여기에 지정합니다.	아니요		최대 길이: 128
overwriteIfExists	대상 키가 이미 있는 경우 값을 덮어쓸지 여부	아니요	`false`	

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
key1:value1!key2:value2!key3:value3!key4
```

다음 프로세서 구성을 사용한다고 가정해 보겠습니다.

```
[
    {
        "parseKeyValue": {
            "destination": "new_key",
            "fieldDelimiter": "!",
            "keyValueDelimiter": ":",
```

```
            "nonMatchValue": "defaultValue",
            "keyPrefix": "parsed_"
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "new_key": {
    "parsed_key1": "value1",
    "parsed_key2": "value2",
    "parsed_key3": "value3",
    "parsed_key4": "defaultValue"
  }
}
```

AWS 벤딩 로그용 기본 제공 프로세서

이 섹션에는 로그를 제공하는 AWS 서비스와 함께 사용할 수 있는 내장 프로세서에 대한 정보가 포함되어 있습니다.

목차

- parseWAF
- parsePostgres
- parseCloudfront
- parseRoute53
- parseVPC
- parseToOCSF

parseWAF

이 프로세서를 사용하여 AWS WAF 벤딩된 로그를 구문 분석합니다. 각 헤더 이름에서의 콘텐츠를 가져`httpRequest.headers`와서 해당 값과 함께 JSON 키를 생성합니다. 또한 `labels`에 대해서도 동일한 작업을 수행합니다. 이러한 변환을 통해 AWS WAF 로그를 훨씬 쉽게 쿼리할 수 있습니다. AWS WAF 로그 형식에 대한 자세한 내용은 웹 ACL 트래픽에 대한 로그 예제를 참조하세요.

이 프로세서는 @message만 입력으로 허용합니다.

> **Important**
>
> 이 프로세서를 사용하는 경우 트랜스포머의 첫 번째 프로세서여야 합니다.

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": [],
  "httpRequest": {
    "clientIp": "1.1.1.1",
    "country": "AU",
    "headers": [
      { "name": "Host", "value": "localhost:1989" },
      { "name": "User-Agent", "value": "curl/7.61.1" },
      { "name": "Accept", "value": "*/*" },
      { "name": "x-stm-test", "value": "10 AND 1=1" }
    ],
    "uri": "/myUri",
    "args": "",
```

```
    "httpVersion": "HTTP/1.1",
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": [{ "name": "value" }]
}
```

프로세서 구성은 다음과 같습니다.

```
[
    {
        "parseWAF": {}
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "httpRequest": {
    "headers": {
      "Host": "localhost:1989",
      "User-Agent": "curl/7.61.1",
      "Accept": "*/*",
      "x-stm-test": "10 AND 1=1"
    },
    "clientIp": "1.1.1.1",
    "country": "AU",
    "uri": "/myUri",
    "args": "",
    "httpVersion": "HTTP/1.1",
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": { "name": "value" },
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
```

```
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": []
}
```

parsePostgres

이 프로세서를 사용하여 Amazon RDS for PostgreSQL 벤딩된 로그를 구문 분석하고, 필드를 추출하고, JSON 형식으로 변환합니다. RDS for PostgreSQL 로그 형식에 대한 자세한 내용은 RDS for PostgreSQL 데이터베이스 로그 파일을 참조하세요.

이 프로세서는 `@message`만 입력으로 허용합니다.

> **Important**
>
> 이 프로세서를 사용하는 경우 트랜스포머의 첫 번째 프로세서여야 합니다.

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
2019-03-10 03:54:59 UTC:10.0.0.123(52834):postgres@logtestdb:[20175]:ERROR: column
 "wrong_column_name" does not exist at character 8
```

프로세서 구성은 다음과 같습니다.

```
[
    {
        "parsePostgres": {}
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "logTime": "2019-03-10 03:54:59 UTC",
  "srcIp": "10.0.0.123(52834)",
  "userName": "postgres",
  "dbName": "logtestdb",
  "processId": "20175",
  "logLevel": "ERROR"
}
```

parseCloudfront

이 프로세서를 사용하여 Amazon CloudFront 벤딩된 로그를 구문 분석하고, 필드를 추출하고, JSON 형식으로 변환합니다. 인코딩된 필드 값은 디코딩됩니다. 정수 및 2배인 값은 이와 같이 처리됩니다. Amazon CloudFront 로그 형식에 대한 자세한 내용은 표준 로그 구성 및 사용(액세스 로그)을 참조하세요.

이 프로세서는 `@message`만 입력으로 허용합니다.

> **Important**
>
> 이 프로세서를 사용하는 경우 트랜스포머의 첫 번째 프로세서여야 합니다.

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
2019-12-04  21:02:31   LAX1   392    192.0.2.24    GET
 d111111abcdef8.cloudfront.net  /index.html    200    -  Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36  -  -  Hit
    SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ==
 d111111abcdef8.cloudfront.net  https  23 0.001  -  TLSv1.2    ECDHE-RSA-AES128-GCM-
SHA256    Hit    HTTP/2.0   -  -  11040  0.001  Hit    text/html  78 -  -
```

프로세서 구성은 다음과 같습니다.

```
[
```

```
    {
        "parseCloudfront": {}
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "date": "2019-12-04",
  "time": "21:02:31",
  "x-edge-location": "LAX1",
  "sc-bytes": 392,
  "c-ip": "192.0.2.24",
  "cs-method": "GET",
  "cs(Host)": "d111111abcdef8.cloudfront.net",
  "cs-uri-stem": "/index.html",
  "sc-status": 200,
  "cs(Referer)": "-",
  "cs(User-Agent)": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36",
  "cs-uri-query": "-",
  "cs(Cookie)": "-",
  "x-edge-result-type": "Hit",
  "x-edge-request-id": "SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ==",
  "x-host-header": "d111111abcdef8.cloudfront.net",
  "cs-protocol": "https",
  "cs-bytes": 23,
  "time-taken": 0.001,
  "x-forwarded-for": "-",
  "ssl-protocol": "TLSv1.2",
  "ssl-cipher": "ECDHE-RSA-AES128-GCM-SHA256",
  "x-edge-response-result-type": "Hit",
  "cs-protocol-version": "HTTP/2.0",
  "fle-status": "-",
  "fle-encrypted-fields": "-",
  "c-port": 11040,
  "time-to-first-byte": 0.001,
  "x-edge-detailed-result-type": "Hit",
  "sc-content-type": "text/html",
  "sc-content-len": 78,
  "sc-range-start": "-",
  "sc-range-end": "-"
}
```

parseRoute53

이 프로세서를 사용하여 Amazon Route 53 Public Data Plane 벤딩된 로그를 구문 분석하고, 필드를 추출하고, JSON 형식으로 변환합니다. 인코딩된 필드 값은 디코딩됩니다. 이 프로세서는 Amazon Route 53 Resolver 로그를 지원하지 않습니다.

이 프로세서는 `@message`만 입력으로 허용합니다.

> **Important**
>
> 이 프로세서를 사용하는 경우 트랜스포머의 첫 번째 프로세서여야 합니다.

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
1.0 2017-12-13T08:15:50.235Z Z123412341234 example.com AAAA NOERROR TCP IAD12 192.0.2.0
 198.51.100.0/24
```

프로세서 구성은 다음과 같습니다.

```
[
    {
        "parseRoute53": {}
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "version": 1.0,
  "queryTimestamp": "2017-12-13T08:15:50.235Z",
  "hostZoneId": "Z123412341234",
  "queryName": "example.com",
  "queryType": "AAAA",
  "responseCode": "NOERROR",
  "protocol": "TCP",
  "edgeLocation": "IAD12",
  "resolverIp": "192.0.2.0",
```

```
  "ednsClientSubnet": "198.51.100.0/24"
}
```

parseVPC

이 프로세서를 사용하여 Amazon VPC 판매 로그를 구문 분석하고, 필드를 추출하고, JSON 형식으로 변환합니다. 인코딩된 필드 값은 디코딩됩니다.

이 프로세서는 `@message`만 입력으로 허용합니다.

> **Important**
>
> 이 프로세서를 사용하는 경우 트랜스포머의 첫 번째 프로세서여야 합니다.

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
2 123456789010 eni-abc123de 192.0.2.0 192.0.2.24 20641 22 6 20 4249 1418530010
 1418530070 ACCEPT OK
```

프로세서 구성은 다음과 같습니다.

```
[
    {
        "parseVPC": {}
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "version": 2,
  "accountId": "123456789010",
  "interfaceId": "eni-abc123de",
  "srcAddr": "192.0.2.0",
  "dstAddr": "192.0.2.24",
  "srcPort": 20641,
```

```
  "dstPort": 22,
  "protocol": 6,
  "packets": 20,
  "bytes": 4249,
  "start": 1418530010,
  "end": 1418530070,
  "action": "ACCEPT",
  "logStatus": "OK"
}
```

parseToOCSF

parseToOCSF 프로세서는 로그를 OCSF(Open Cybersecurity Schema Framework) 이벤트로 변환합니다. OCSF는 보안 데이터에 대한 공통 스키마를 제공하는 개방형 표준으로, 다양한 보안 도구 및 플랫폼에서 보다 나은 상호 운용성과 분석을 지원합니다.

이 프로세서는 다양한 AWS 서비스의 로그 형식을 다운스트림 분석을 위한 일관된 스키마로 표준화해야 하는 보안 분석 워크플로에 특히 유용합니다.

파라미터

eventSource(필수)

변환할 로그 이벤트를 생성하는 AWS 서비스 또는 프로세스를 지정합니다. 유효값은 다음과 같습니다.

- CloudTrail - CloudTrail 로그
- Route53Resolver - Route 53 Resolver 로그
- VPCFlow - Amazon VPC Flow Logs
- EKSAudit - Amazon EKS 감사 로그
- AWSWAF - AWS WAF 로그

ocsfVersion(필수)

변환된 로그 이벤트에 사용할 OCSF 스키마 버전을 지정합니다. 현재 지원되는 버전: V1.1, V1.5

mappingVersion (선택 사항)

OCSF 변환 매핑 버전을 지정합니다. 로그를 OCSF 형식으로 변환할 때 적용되는 변환 로직을 제어합니다. 지정하지 않으면는 정책 생성 시 사용 가능한 최신 버전을 사용합니다. 새 매핑 버전이 릴리스되면 기존 정책이 자동으로 업그레이드되지 않습니다. 현재 최신 버전: v1.5.0.

참고:이 ocsfVersion인 경우 지원되지 않습니다V1.1.

source (선택 사항)

구문 분석하려는 로그 이벤트의 필드 경로. 생략하면 전체 로그 메시지가 구문 분석됩니다.

예제

다음 예제에서는 parseToOCSF를 사용하여 VPC 흐름 로그를 OCSF 형식으로 변환하는 방법을 보여줍니다.

```
{
  "parseToOCSF": {
    "eventSource": "VPCFlow",
    "ocsfVersion": "V1.1"
  }
}
```

다음 예제에서는 일관된 변환 동작을 위해 특정 매핑 버전을 지정하는 방법을 보여줍니다.

```
{
  "parseToOCSF": {
    "eventSource": "CloudTrail",
    "ocsfVersion": "V1.5",
    "mappingVersion": "v1.5.0"
  }
}
```

문자열 변형 프로세서

이 섹션에는 로그 이벤트 변환기와 함께 사용할 수 있는 문자열 변형 프로세서에 대한 정보가 포함되어 있습니다.

목차

- lowerCaseString
- upperCaseString
- splitString
- substituteString

- trimString

lowerCaseString

lowerCaseString 프로세서는 문자열을 소문자 버전으로 변환합니다.

필드	설명	필수	기본값	제한 사항
withKeys	소문자로 변환할 키의 목록	예		최대 항목: 10

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "INNER_VALUE"
    }
}
```

트랜스포머 구성은 lowerCaseString를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "lowerCaseString": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key": "inner_value"
```

```
  }
}
```

upperCaseString

upperCaseString 프로세서는 문자열을 대문자 버전으로 변환합니다.

필드	설명	필수	기본값	제한 사항
withKeys	대문자로 변환할 키의 목록	예		최대 항목: 10

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

트랜스포머 구성은 upperCaseString를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "upperCaseString": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key": "INNER_VALUE"
```

```
  }
}
```

splitString

splitString 프로세서는 구분 문자를 사용하여 필드를 배열로 분할하는 문자열 변형 프로세서의 한 유형입니다.

필드	설명	필수	기본값	제한 사항
항목	항목의 배열. 배열의 각 항목에는 source 및 delimiter 필드가 포함되어야 합니다.	예		최대 항목: 10
source	분할할 필드 값의 키	예		최대 길이: 128
delimiter	필드 값을 분할할 구분 문자 문자열	예		최대 길이: 128

예제 1

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

트랜스포머 구성은 splitString를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
     {
        "parseJSON": {}
    },
    {
         "splitString": {
            "entries": [
                {
                    "source": "outer_key.inner_key",
```

```
                    "delimiter": "_"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key": [
      "inner",
      "value"
    ]
  }
}
```

예제 2

문자열을 분할할 구분 문자는 여러 문자일 수 있습니다.

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "item1, item2, item3"
    }
}
```

트랜스포머 구성은 다음과 같습니다.

```
[
     {
        "parseJSON": {}
    },
    {
         "splitString": {
            "entries": [
                {
                    "source": "outer_key.inner_key",
```

```
                    "delimiter": ", "
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key": [
      "item1",
      "item2",
      "item3"
    ]
  }
}
```

substituteString

`substituteString` 프로세서는 키 값을 정규식과 비교하여 일치하는 모든 항목을 대체 문자열로 대체하는 문자열 변형 프로세서의 한 유형입니다.

필드	설명	필수	기본값	제한 사항
항목	항목의 배열. 배열의 각 항목에는 `source`, `from` 및 `to` 필드가 포함되어야 합니다.	예		최대 항목: 10
source	수정할 필드의 키	예		최대 길이: 128 최대 중첩 키 깊이: 3
from	대체될 정규식 문자열입니다. 큰따옴표를 사용할 때는 \\를 사용하고 작은따옴표를 사용하거나 AWS Management Console에서 구성하는 경우에는 \를 사용하여 [및]와 같은 특수 정규식 문자를 이스케이	예		최대 길이: 128

필드	설명	필수	기본값	제한 사항
	프 처리해야 합니다. 자세한 내용은 Oracle 웹사이트의 Class Pattern을 참조하세요. (...)에서 패턴을 래핑하여 번호가 지정된 캡처 그룹을 생성하고 to 필드에서 참조할 수 있는 이름이 (?P<group_name>...) 인 캡처 그룹을 생성할 수 있습니다.			
to	캡처 그룹에 대한 from 역참조의 각 매칭으로 대체될 문자열을 사용할 수 있습니다. $1와 같은 번호가 매겨진 그룹에는 $n 형식을 사용하고, ${my_group} 와 같은 명명된 그룹에는 ${group_name} 을 사용합니다.>	예		최대 길이: 128 최대 역참조의 수: 10 최대 중복된 역참조의 수: 2

예제 1

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key1": "[]",
        "inner_key2": "123-345-567",
        "inner_key3": "A cat takes a catnap."
    }
}
```

트랜스포머 구성은 substituteString를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
```

```
        "substituteString": {
            "entries": [
                {
                    "source": "outer_key.inner_key1",
                    "from": "\\[\\]",
                    "to": "value1"
                },
                {
                    "source": "outer_key.inner_key2",
                    "from": "[0-9]{3}-[0-9]{3}-[0-9]{3}",
                    "to": "xxx-xxx-xxx"
                },
                {
                    "source": "outer_key.inner_key3",
                    "from": "cat",
                    "to": "dog"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key1": "value1",
    "inner_key2": "xxx-xxx-xxx",
    "inner_key3": "A dog takes a dognap."
  }
}
```

예제 2

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key1": "Tom, Dick, and Harry",
        "inner_key2": "arn:aws:sts::123456789012:assumed-role/MyImportantRole/
MySession"
    }
}
```

트랜스포머 구성은 substituteString를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "substituteString": {
            "entries": [
                {
                    "source": "outer_key.inner_key1",
                    "from": "(\w+), (\w+), and (\w+)",
                    "to": "$1 and $3"
                },
                {
                    "source": "outer_key.inner_key2",
                    "from": "^arn:aws:sts::(?P<account_id>\\d{12}):assumed-role/(?
P<role_name>[\\w+=,.@-]+)/(?P<role_session_name>[\\w+=,.@-]+)$",
                    "to": "${account_id}:${role_name}:${role_session_name}"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key1": "Tom and Harry",
    "inner_key2": "123456789012:MyImportantRole:MySession"
  }
}
```

trimString

trimString 프로세서는 키의 시작과 끝에서 공백을 제거합니다.

필드	설명	필수	기본값	제한 사항
withKeys	정리할 키의 목록	예		최대 항목: 10

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "   inner_value  "
    }
}
```

트랜스포머 구성은 `trimString`를 `parseJSON`과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "trimString": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

JSON 변형 프로세서

이 섹션에는 로그 이벤트 변환기와 함께 사용할 수 있는 JSON 변형 프로세서에 대한 정보가 포함되어 있습니다.

목차

- addKeys
- deleteKeys
- moveKeys

- renameKeys
- copyValue
- listToMap

addKeys

addKeys 프로세서를 사용하여 로그 이벤트에 새 키-값 쌍을 추가합니다.

필드	설명	필수	기본값	제한 사항
항목	항목의 배열. 배열의 각 항목에는 key, value 및 overwriteIfExists 필드가 포함될 수 있습니다.	예		최대 항목: 5
키	추가할 새 항목의 키	예		최대 길이: 128 최대 중첩 키 깊이: 3
값	추가할 새 항목의 값	예		최대 길이: 256
overwriteIfExists	이 값을 true로 설정하면 이벤트에 key가 이미 있는 경우 기존 값을 덮어씁니다. 기본값은 false입니다.	아니요	false	제한 없음

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

트랜스포머 구성은 addKeys를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
```

```
    {
        "parseJSON": {}
    },
    {
        "addKeys": {
            "entries": [
                {
                    "key": "outer_key.new_key",
                    "value": "new_value"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key": "inner_value",
    "new_key": "new_value"
  }
}
```

deleteKeys

deleteKeys 프로세서를 사용하여 로그 이벤트에서 필드를 삭제합니다. 이러한 필드에는 키-값 쌍이 포함될 수 있습니다.

필드	설명	필수	기본값	제한 사항
withKeys	삭제할 키의 목록.	예	제한 없음	최대 항목: 5

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "inner_value"
```

```
    }
}
```

트랜스포머 구성은 deleteKeys를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "deleteKeys": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {}
}
```

moveKeys

moveKeys 프로세서를 사용하여 한 필드에서 다른 필드로 키를 이동합니다.

필드	설명	필수	기본값	제한 사항
항목	항목의 배열. 배열의 각 항목에는 source, target 및 overwriteIfExists 필드가 포함될 수 있습니다.	예		최대 항목: 5
source	이동할 키	예		최대 길이: 128 최대 중첩 키 깊이: 3
대상	이동할 키	예		최대 길이: 128 최대 중첩 키 깊이: 3

필드	설명	필수	기본값	제한 사항
overwriteIfExists	이 값을 `true`로 설정하면 이벤트에 key가 이미 있는 경우 기존 값을 덮어씁니다. 기본값은 `false`입니다.	아니요	false	제한 없음

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key1": {
        "inner_key1": "inner_value1"
    },
    "outer_key2": {
        "inner_key2": "inner_value2"
    }
}
```

트랜스포머 구성은 `moveKeys`를 `parseJSON`과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "moveKeys": {
            "entries": [
                {
                    "source": "outer_key1.inner_key1",
                    "target": "outer_key2"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
```

```
  "outer_key1": {},
  "outer_key2": {
    "inner_key2": "inner_value2",
    "inner_key1": "inner_value1"
  }
}
```

renameKeys

renameKeys 프로세서를 사용하여 로그 이벤트에서 키의 이름을 바꿉니다.

필드	설명	필수	기본값	제한 사항
항목	항목의 배열. 배열의 각 항목에는 key, target 및 overwriteIfExists 필드가 포함될 수 있습니다.	예	제한 없음	최대 항목: 5
키	이름을 바꿀 키	예	제한 없음	최대 길이: 128
대상	새 키 이름	예	제한 없음	최대 길이: 128 최대 중첩 키 깊이: 3
overwriteIfExists	이 값을 true로 설정하면 이벤트에 key가 이미 있는 경우 기존 값을 덮어씁니다. 기본값은 false입니다.	아니요	false	제한 없음

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

트랜스포머 구성은 renameKeys를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "renameKeys": {
            "entries": [
                {
                    "key": "outer_key",
                    "target": "new_key"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "new_key": {
    "inner_key": "inner_value"
  }
}
```

copyValue

copyValue 프로세서를 사용하여 로그 이벤트 내에서 값을 복사합니다. 또한 이 프로세서를 사용하여 @logGroupName, @logGroupStream, @accountId, @regionName 등 메타데이터 키 값을 로그 이벤트에 복사하여 로그 이벤트에 메타데이터를 추가할 수 있습니다. 다음 예제에 설명되어 있습니다.

필드	설명	필수	기본값	제한 사항
항목	항목의 배열. 배열의 각 항목에는 source, target 및 overwriteIfExists 필드가 포함될 수 있습니다.	예		최대 항목: 5
source	복사할 키	예		최대 길이: 128 최대 중첩 키 깊이: 3

필드	설명	필수	기본값	제한 사항
대상	값을 복사할 키	예	제한 없음	최대 길이: 128 최대 중첩 키 깊이: 3
overwriteIfExists	이 값을 `true`로 설정하면 이벤트에 `key`가 이미 있는 경우 기존 값을 덮어씁니다. 기본값은 `false`입니다.	아니요	false	제한 없음

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

트랜스포머 구성은 `copyValue`를 `parseJSON`과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "copyValue": {
            "entries": [
                {
                    "key": "outer_key.new_key",
                    "target": "new_key"
                },
                {
                    "source": "@logGroupName",
                    "target": "log_group_name"
                },
                {
                    "source": "@logGroupStream",
                    "target": "log_group_stream"
                },
```

```
                {
                    "source": "@accountId",
                    "target": "account_id"
                },
                {
                    "source": "@regionName",
                    "target": "region_name"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
  "outer_key": {
    "inner_key": "inner_value"
  },
  "new_key": "inner_value",
  "log_group_name": "myLogGroupName",
  "log_group_stream": "myLogStreamName",
  "account_id": "012345678912",
  "region_name": "us-east-1"
}
```

listToMap

`listToMap` 프로세서는 키 필드가 포함된 객체 목록을 가져와 대상 키 맵으로 변환합니다.

필드	설명	필수	기본값	제한 사항
source	맵으로 변환될 객체 목록이 있는 ProcessingEvent의 키	예		최대 길이: 128 최대 중첩 키 깊이: 3
키	생성된 맵에서 키로 추출할 필드의 키	예		최대 길이: 128
valueKey	이렇게 지정하면 이 파라미터에서 지정하는 값이 `source` 객체에서	아니요		최대 길이: 128

필드	설명	필수	기본값	제한 사항
	추출되어 생성된 맵의 값에 추가됩니다. 그렇지 않으면 소스 목록의 원본 객체가 생성된 맵의 값에 추가됩니다.			
대상	생성된 맵을 보관할 필드의 키.	아니요	루트 노드	최대 길이: 128 최대 중첩 키 깊이: 3
평면화	목록을 단일 항목으로 평탄화할지 아니면 생성된 맵의 값을 목록으로 만들지 나타내는 부울 값입니다. 기본적으로 매칭 키의 값은 배열로 표시됩니다. `flatten`을 `true`로 설정하여 배열을 `flattened Element` 의 값을 기반으로 하는 단일 값으로 변환합니다.	아니요	false	
flattened Element	`flatten`을 `true`로 설정한 경우 `flattenedElement` 를 사용하여 유지할 요소 `first` 또는 `last`를 지정합니다.	`flatten`이 `true`로 설정된 경우 필요합니다.		값은 `first` 또는 `last`만 될 수 있습니다.

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
```

```
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ]
}
```

사용 사례 1의 트랜스포머: `flatten`은 `false`입니다

```
[
    {
        "parseJSON": {}
    },
    {
        "listToMap": {
            "source": "outer_key"
            "key": "inner_key",
            "valueKey": "inner_value",
            "flatten": false
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
```

```
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": [
        "val-a"
    ],
    "b": [
        "val-b1",
        "val-b2"
    ],
    "c": [
        "val-c"
    ]
}
```

사용 사례 2의 트랜스포머: `flatten`은 `true`이고, `flattenedElement`는 `first`입니다

```
[
    {
        "parseJSON": {}
    },
    {
        "listToMap": {
            "source": "outer_key"
            "key": "inner_key",
            "valueKey": "inner_value",
            "flatten": true,
            "flattenedElement": "first"
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
```

```
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": "val-a",
    "b": "val-b1",
    "c": "val-c"
}
```

사용 사례 3의 트랜스포머: `flatten`은 `true`이고, `flattenedElement`는 `last`입니다

```
[
    {
        "parseJSON": {}
    },
    {
        "listToMap": {
            "source": "outer_key"
            "key": "inner_key",
            "valueKey": "inner_value",
            "flatten": true,
            "flattenedElement": "last"
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
```

```
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": "val-a",
    "b": "val-b2",
    "c": "val-c"
}
```

데이터타입 컨버터 프로세서

이 섹션에는 로그 이벤트 변환기와 함께 사용할 수 있는 데이터 유형 변환기 프로세서에 대한 정보가 포함되어 있습니다.

목차

- typeConverter
- datetimeConverter

typeConverter

`typeConverter` 프로세서를 사용하여 지정된 키와 연결된 값 유형을 지정된 유형으로 변환합니다. 지정된 필드의 유형을 변경하는 캐스팅 프로세서입니다. 값은 `integer`, `double` `string` 및 `boolean` 데이터타입 중 하나로 변환할 수 있습니다.

필드	설명	필수	기본값	제한 사항
항목	항목의 배열. 배열의 각 항목에는 `key` 및 `type` 필드가 포함되어야 합니다.	예		최대 항목: 10

필드	설명	필수	기본값	제한 사항
키	다른 유형으로 변환할 값이 있는 키	예		최대 길이: 128 최대 중첩 키 깊이: 3
type	변환할 유형. 유효값은 `integer`, `double`, `string` 및 `boolean`입니다.	예		

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{
    "name": "value",
    "status": "200"
}
```

트랜스포머 구성은 `typeConverter`를 `parseJSON`과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "typeConverter": {
            "entries": [
                {
                    "key": "status",
                    "type": "integer"
                }
            ]
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
```

```
    "name": "value",
    "status": 200
}
```

datetimeConverter

`datetimeConverter` 프로세서를 사용하여 날짜/시간 문자열을 지정한 형식으로 변환합니다.

필드	설명	필수	기본값	제한 사항
source	날짜 변환을 적용할 키.	예		최대 항목: 10
matchPatterns	`source` 필드를 매칭할 패턴의 목록	예		최대 항목: 5
대상	결과를 저장할 JSON 필드.	예		최대 길이: 128 최대 중첩 키 깊이: 3
targetFormat	대상 필드의 변환된 데이터에 사용할 날짜/시간 형식.	아니요	`yyyy-MM-dd'T'HH:mm:ss.SSS'Z`	최대 길이: 64
sourceTimezone	소스 필드의 시간대. 가능한 값의 목록은 Java 지원되는 영역 ID 및 오프셋을 참조하세요.	아니요	UTC	최소 길이: 1
targetTimezone	대상 필드의 시간대. 가능한 값의 목록은 Java 지원되는 영역 ID 및 오프셋을 참조하세요.	아니요	UTC	최소 길이: 1
locale	소스 필드의 로케일. 가능한 값의 목록은 예제가 포함된 Java의 Locale getAvailableLocales() Method를 참조하세요.	예		최소 길이: 1

예제

다음 예제 로그 이벤트를 살펴봅니다.

```
{"german_datetime": "Samstag 05. Dezember 1998 11:00:00"}
```

트랜스포머 구성은 dateTimeConverter를 parseJSON과 사용하는 경우 다음과 같습니다.

```
[
    {
        "parseJSON": {}
    },
    {
        "dateTimeConverter": {
            "source": "german_datetime",
            "target": "target_1",
            "locale": "de",
            "matchPatterns": ["EEEE dd. MMMM yyyy HH:mm:ss"],
            "sourceTimezone": "Europe/Berlin",
            "targetTimezone": "America/New_York",
            "targetFormat": "yyyy-MM-dd'T'HH:mm:ss z"
        }
    }
]
```

변환된 로그 이벤트는 다음과 같습니다.

```
{
    "german_datetime": "Samstag 05. Dezember 1998 11:00:00",
    "target_1": "1998-12-05T17:00:00 MEZ"
}
```

변환 지표 및 오류

CloudWatch Logs는 CloudWatch에 변환 지표를 게시합니다. 이러한 지표에는 TransformedLogEvents, TransformedBytes 및 TransformationErrors이 있습니다. 자세한 내용은 로그 트랜스포머 지표 및 차원 단원을 참조하십시오.

CloudWatch Logs가 로그 이벤트를 변환하려고 하거나 변환하지 못할 때마다 해당 로그 이벤트에 @transformationError 시스템 필드가 추가됩니다. CloudWatch Logs

Insights 쿼리를 실행하면 변환에 실패한 모든 로그 이벤트에이 필드가 표시됩니다. `filter ispresent(@transformationError)`와 같은 쿼리로 이 필드를 쿼리하여 실패한 모든 변환 이벤트를 찾을 수 있습니다.

S3 Tables Integration을 사용한 액세스 로그

CloudWatch와 S3 Tables 통합을 사용하면 Amazon Athena, Amazon Redshift 및 Apache Iceberg 호환 스토어에 대한 연결을 지원하는 타사 도구와 같은 분석 엔진을 사용하여 CloudWatch에 수집된 로그 데이터에 액세스할 수 있습니다. 이 통합을 통해 원하는 도구를 사용하여 포괄적인 로그 분석을 수행하고 CloudWatch Logs의 데이터를 비 CloudWatch 데이터와 상호 연관시킬 수 있습니다. 이 기능은 추가 비용 없이 사용할 수 있습니다.

S3 테이블 통합 이해

Amazon S3 Tables Integration은 CloudWatch Logs의 로그를 관리형 Amazon S3 테이블로 사용할 수 있게 해주는 완전 관리형 솔루션입니다. 이 통합을 통해 CloudWatch Logs 기능 외에도 로그를 분석하는 방법에 대한 유연성을 높일 수 있습니다.

통합은 관리형 Amazon S3 테이블 버킷(`aws-cloudwatch`)을 생성하고 데이터 소스 이름 및 유형(CloudWatch Logs 콘솔의 로그 관리 > 데이터 소스 탭에서 관리 가능)을 기반으로 특정 로그 소스를 Amazon S3 테이블과 연결하는 방식으로 작동합니다. 연결되면 Apache Iceberg 형식을 사용하여 Amazon S3 Tables를 통해 CloudWatch Logs 데이터에 액세스할 수 있습니다. 이 형식은 다양한 분석 엔진이 데이터를 효율적으로 쿼리할 수 있는 표준화된 방법을 제공합니다.

핵심 구성 요소

Apache Iceberg 테이블

구조화된 데이터 스토리지를 제공하고 여러 분석 엔진과의 호환성을 지원하는 S3 Tables에서 사용하는 기본 테이블 형식입니다.

데이터 소스 연결

데이터 소스 및 유형 기준에 따라 특정 CloudWatch Logs 소스를 S3 Tables 통합에 연결하는 프로세스입니다.

S3 테이블로의 데이터 흐름

CloudWatch Logs와 S3 Tables 간의 데이터 흐름을 이해하면 통합을 계획하고 로그 데이터를 효과적으로 관리하는 데 도움이 됩니다.

연결을 생성하면 CloudWatch Logs는 연결된 데이터 소스 이름 및 유형과 일치하는 새 로그 이벤트를 CloudWatch 관리형 S3 테이블 버킷에 자동으로 전송합니다. 이러한 이벤트는 해당 데이터 소스에 해

당하는 테이블 아래의 로그 네임스페이스에 표시됩니다. 통합은 연결이 생성된 후 수신된 로그 이벤트만 처리합니다. 기존 로그 데이터는 다시 채워지지 않습니다. S3 Tables 통합을 생성할 때 S3 테이블에서 모든 로그 소스 및 유형을 사용할 수 있도록 활성화 확인란을 선택한 상태로 두면 나중에 추가된 데이터 소스를 포함하여 계정의 모든 데이터 소스가 자동으로 연결되고 S3 Tables로 전송됩니다. 특정 데이터 소스만 S3 Tables에 전송하려면 통합 생성 중에이 확인란의 선택을 취소한 다음 포함하려는 데이터 소스를 개별적으로 연결합니다.

S3 테이블 버킷의 데이터 보존은 로그 그룹에 설정된 보존 정책과 일치합니다. 예를 들어 로그 그룹을 1일 보존으로 설정하면 CloudWatch Logs는 하루 후에 CloudWatch Logs와 S3 테이블 모두에서 데이터를 제거합니다. 로그 그룹 또는 로그 스트림을 삭제하면 CloudWatch Logs도 S3 테이블 버킷에서 데이터를 제거합니다.

S3 테이블 통합을 사용해야 하는 경우

S3 Tables 통합을 사용하여 로그 데이터를 다른 외부 또는 비 CloudWatch 데이터와 연관시키거나 Amazon Athena와 같은 다른 분석 도구를 사용하여 CloudWatch Logs 데이터에 대한 분석을 수행하는 것을 선호하는 경우를 고려해 보세요. CloudWatch Logs에서 사용할 수 있는 것 이상의 기능이 필요한 경우이 통합을 사용합니다. 이 통합은 다음과 같은 경우에 특히 유용합니다.

- 대량의 로그 데이터에서 복잡한 SQL과 유사한 쿼리를 실행해야 합니다.
- 로그 분석을 기존 분석 워크플로 및 도구와 통합하려는 경우
- 여러 데이터 소스에 걸친 포괄적인 로그 분석 기능이 필요합니다.

기존 CloudWatch 수집 및 스토리지 요금 외에이 통합을 통해 생성된 S3 테이블에 대한 추가 스토리지 또는 테이블 유지 관리 요금은 없습니다.

사전 조건

통합을 구현하기 전에 다음이 있는지 확인합니다.

- 기존 CloudWatch Logs 데이터
- 다음 섹션에 설명된 대로 CloudWatch Logs와 S3 Tables 간의 교차 서비스 액세스를 위한 적절한 IAM 권한

IAM 권한

CloudWatch Logs를 S3 Tables와 통합하려면 통합을 설정하는 사용자 또는 역할과 CloudWatch Logs가 S3 Tables에 데이터를 쓰기 위해 수임하는 서비스 역할이라는 두 개의 개별 엔터티에 대한 IAM 권한을 구성해야 합니다.

통합을 생성하는 역할 또는 사용자의 경우

통합을 설정하는 사용자 또는 역할에는 다음 권한이 필요합니다.

- `observabilityadmin:CreateS3TableIntegration` 통합을 생성하고 소스를 `logs:AssociateSourceToS3TableIntegration` 추가하려면
- `s3tables:CreateTableBuckets3tables:PutTableBucketEncryption`, 및 `s3tables:PutTableBucketPolicy`- S3 테이블 버킷 구성

서비스 역할의 경우

CloudWatch Logs가 테이블 버킷에 데이터를 쓰는 데 사용하는 IAM 서비스 역할에 다음 IAM 정책을 연결합니다. 이 정책은 테이블에 쓸 수 있는 권한을 부여합니다. *aws-region*, *123456789012* 및 *log-group-name*을 AWS 리전, 계정 ID 및 로그 그룹 이름으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:integrateWithS3Table"
            ],
            "Resource": ["arn:aws:logs:aws-region:123456789012:log-group:log-group-
name"],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

CloudWatch Logs가 로그 데이터를 S3 Tables에 쓰기 위해 수임할 IAM 서비스 역할에 다음 신뢰 정책을 연결합니다. 통합 설정 중에이 역할을 생성하거나 선택합니다. 조건은 CloudWatch Logs가 지정된 계정 및 로그 그룹에 대해서만 역할을 수임할 수 있도록 역할을 제한합니다. *aws-region*, *123456789012* 및 *log-group-name*을 AWS 리전, 계정 ID 및 로그 그룹 이름으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": ["arn:aws:logs:aws-region:123456789012:log-
group:log-group-name"]
                }
            }
        }
    ]
}
```

KMS 키 정책(암호화된 데이터용)

고객 관리형 키를 사용하여 로그 데이터를 암호화하는 경우 CloudWatch 서비스 보안 주체와 S3 Tables 유지 관리 서비스 보안 주체에 키에 대한 액세스 권한을 부여해야 합니다. KMS 키 정책에 다음 문을 추가합니다. 자리 표시자 값을 AWS 계정 ID, 리전, KMS 키 ID 및 S3 테이블 또는 테이블 버킷 ARN으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableSystemTablesKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "systemtables.cloudwatch.amazonaws.com"
```

```
            },
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:aws-region:123456789012:key/key-id",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        },
        {
            "Sid": "EnableKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "maintenance.s3tables.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:aws-region:123456789012:key/key-id",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "<table-or-table-bucket-arn>/*"
                }
            }
        }
    ]
}
```

시작하기

S3 Tables Integration을 시작하려면 CloudWatch Logs와 S3 Tables 간의 통합을 설정해야 합니다. 이 프로세스에는 데이터 소스 연결 구성 및 적절한 IAM 권한 설정이 포함됩니다.

S3 테이블 통합을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/ CloudWatch Logs 콘솔을 엽니다.
2. 설정, 전역, S3 테이블 통합 생성을 선택합니다.

3. S3 Tables에서 로그를 암호화하는 방법과 CloudWatch Logs가 S3 Tables에 로그를 쓰는 데 사용할 역할을 사용자 지정합니다.
4. 모든 데이터 소스를 통합과 자동으로 연결하려면 S3 테이블에서 모든 로그 소스 및 유형을 사용할 수 있도록 활성화 확인란을 선택한 상태로 둡니다(기본적으로 선택됨). 특정 데이터 소스만 연결하려면이 확인란의 선택을 취소합니다.
5. S3 테이블 통합 생성을 선택합니다.

> **Note**
>
> 생성 중에 S3 테이블에서 사용할 수 있도록 모든 로그 소스 및 유형 활성화를 선택한 경우 향후 추가되는 모든 데이터 소스를 포함하여 모든 데이터 소스가 자동으로 연결됩니다. 여기에서 중지할 수 있습니다. 다음 단계는 확인란을 선택 취소하고 특정 데이터 소스를 연결하려는 경우에만 필요합니다.

소스를 S3 테이블 통합에 연결하려면

1. https://console.aws.amazon.com/cloudwatch/ CloudWatch Logs 콘솔을 엽니다.
2. 설정, 전역, S3 테이블 통합 관리를 선택합니다.
3. 데이터 소스 연결을 선택합니다.
4. 통합을 활성화하려는 데이터 소스 이름과 데이터 소스 유형을 선택합니다.
5. 데이터 소스 연결을 선택합니다.

로그 관리 페이지에서 소스를 S3 테이블 통합에 연결하려면

1. https://console.aws.amazon.com/cloudwatch/ CloudWatch Logs 콘솔을 엽니다.
2. 탐색 창에서 로그 관리를 선택합니다.
3. 데이터 소스 탭을 선택합니다.
4. 통합하려는 데이터 소스 이름과 데이터 소스 유형을 선택합니다.
5. 데이터 소스 작업을 선택합니다.
6. S3 테이블 통합과 연결을 선택합니다.
7. 데이터 소스를 검토한 다음 데이터 소스 연결을 선택합니다.

데이터를 사용하려면 먼저 다음 3단계를 수행해야 합니다.

1. Amazon S3 테이블과 AWS 분석 서비스 통합 - Amazon S3 콘솔 사용
2. Lake Formation 권한 구성
3. 분석 도구를 사용하여 연결

Amazon S3 테이블과 AWS 분석 서비스 통합 - Amazon S3 콘솔 사용(링크)

S3 콘솔을 사용하여 S3 Tables 통합을 활성화하려면

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블 버킷을 선택합니다.
3. 상단에서 통합 활성화를 선택합니다.
4. 어떤 리전에서든 테이블 버킷을 처음 통합하면 Amazon S3가 사용자를 대신하여 새 IAM 서비스 역할을 생성합니다. 이 역할을 통해 Lake Formation은 계정의 모든 테이블 버킷에 액세스하고 AWS Glue 데이터 카탈로그의 테이블에 대한 액세스를 페더레이션할 수 있습니다.

Lake Formation 권한 구성

CloudWatch Logs에는 테이블에 쓸 수 있는 권한(이전 단계에서 구성됨)이 있지만 사용자 및 분석 역할에는 데이터를 읽을 수 있는 권한이 자동으로 없습니다. AWS Lake Formation을 사용하여 명시적으로 액세스 권한을 부여해야 합니다. 테이블에 대한 액세스를 제공하려는 모든 IAM 보안 주체에 대해이 작업을 수행해야 합니다.

사용자 또는 역할에 쿼리 액세스 권한을 부여하려면

Athena 또는 Redshift에서 쿼리를 실행할 IAM 보안 주체(사용자 또는 역할)에 SELECT 및 DESCRIBE 권한을 부여해야 합니다.

1. AWS Lake Formation 콘솔을 엽니다.
2. 탐색 창의 권한에서 데이터 레이크 권한을 선택합니다.
3. 권한 부여를 선택합니다.
4. 보안 주체: 액세스가 필요한 IAM 사용자 또는 역할(예: 데이터 분석가 또는 현재 사용 중인 관리자 역할)을 선택합니다.
5. LF 태그 또는 카탈로그 리소스: 명명된 데이터 카탈로그 리소스를 선택합니다.
6. 데이터베이스 및 테이블:

    - CloudWatch 통합()에서 생성한 S3 테이블 버킷을 선택합니다`aws-cloudwatch`.

- 데이터 소스와 연결된 특정 테이블을 선택합니다(선택 사항).

7. 테이블 권한: 선택 및 설명을 선택합니다.
8. 권한 부여를 선택합니다.

> **Note**
>
> Athena에서 로그를 쿼리할 때 "액세스 거부됨" 오류가 발생하는 경우 쿼리를 실행하는 사용자에게 Athena에 대한 적절한 IAM 권한과 위에 정의된 Lake Formation 권한이 모두 있는지 확인합니다.

Lake Formation 권한에 대한 자세한 내용은 https://docs.aws.amazon.com/lake-formation/latest/dg/granting-catalog-permissions.html 참조하십시오.

분석 도구를 사용하여 연결

권한이 부여되면 S3 테이블을 쿼리하도록 선호하는 분석 서비스를 구성할 수 있습니다. S3 테이블은 Amazon Athena, Amazon Redshift 및 Amazon EMR에서 기본적으로 지원하는 Apache Iceberg 형식을 사용합니다.

Amazon Athena에서 로그 데이터를 쿼리하려면

Amazon Athena는 Amazon S3 Tables 카탈로그를 Amazon S3 Tables와 상호 작용합니다.

로그 데이터를 쿼리하도록 Athena를 설정하려면

1. https://console.aws.amazon.com/athena/에서 Amazon Athena 콘솔을 엽니다.
2. 쿼리 편집기의 데이터 소스 드롭다운에서 Amazon S3 Tables 카탈로그를 선택합니다.
3. 카탈로그가 표시되지 않는 경우 특정 사용자 역할에 대해 위의 Lake Formation 권한 단계를 완료했는지 확인합니다.
4. 카탈로그를 선택하면 데이터베이스 목록에 로그 테이블이 나타납니다. 이제 로그 데이터에 대해 표준 SQL 쿼리를 실행할 수 있습니다.

쿼리 예제: `SELECT * FROM "amazon_vpc__flow" LIMIT 100;`

https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-tables-integrating-aws.html Analytics 서비스를 S3 Tables와 연결하는 방법에 대해 자세히 알아봅니다.

필터를 사용하여 로그 이벤트에서 지표 생성

지표 필터를 한 개 이상 생성하여 CloudWatch Logs로 들어오는 로그 데이터를 검색하고 필터링할 수 있습니다. 지표 필터는 CloudWatch Logs 로그로 전송될 때 로그 데이터에서 검색할 단어와 패턴을 정의합니다. CloudWatch Logs는 이러한 지표 필터를 사용하여 로그 데이터를 그래프로 나타내거나 경보를 설정할 수 있는 숫자 CloudWatch 지표로 전환합니다.

로그 필터에서 지표를 생성할 때 차원과 단위를 지표에 할당하도록 선택할 수도 있습니다. 단위를 지정하는 경우 필터를 생성할 때 올바른 단위를 지정해야 합니다. 필터 단위를 나중에 변경하는 것은 효과가 없습니다.

를 구성 AWS Organizations 하고 멤버 계정으로 작업하는 경우 로그 중앙 집중화를 사용하여 소스 계정에서 중앙 모니터링 계정으로 로그 데이터를 수집할 수 있습니다.

중앙 집중식 로그 그룹으로 작업하는 경우에는 지표 필터를 생성할 때 이러한 시스템 필드 차원을 사용할 수 있습니다.

- `@aws.account` -이 차원은 로그 이벤트가 시작된 AWS 계정 ID를 나타냅니다.
- `@aws.region` -이 차원은 로그 이벤트가 생성된 AWS 리전을 나타냅니다.

이러한 차원은 로그 데이터의 소스를 식별하는 데 도움이 되므로 중앙 집중식 로그에서 파생된 지표를 보다 세부적으로 필터링하고 분석할 수 있습니다. 자세한 내용은 교차 계정 교차 리전 로그 중앙 집중화 단원을 참조하십시오.

구독이 있는 로그 그룹이 로그 변환을 사용하는 경우 필터 패턴이 로그 이벤트의 변환된 버전에 적용됩니다. 자세한 내용은 수집 중 로그 변환 단원을 참조하십시오.

> **Note**
>
> 지표 필터는 Standard 로그 클래스의 로그 그룹에서만 지원됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스를 참조하세요.

이러한 지표를 보거나 경보를 설정할 때 백분위수 통계를 포함하여 어떤 유형의 CloudWatch 통계도 사용할 수 있습니다.

> **Note**
>
> 지표 값이 음수가 아닌 경우에만 지표의 백분위수 통계가 지원됩니다. 지표 필터를 음수를 보고할 수 있도록 설정하면 값이 음수인 경우에는 해당 지표의 백분위수 통계를 사용할 수 없습니다. 자세한 내용은 백분위수를 참조하세요.

필터는 데이터를 소급해서 필터링하지 않습니다. 필터는 필터가 생성된 이후에 발생한 이벤트에 대한 지표 데이터 요소만 게시합니다. 필터 패턴을 테스트할 때 필터 결과 미리 보기에는 검증을 위해 일치하는 로그 줄이 처음 50개까지 표시됩니다. 필터링된 결과의 타임스탬프가 지표 생성 시간보다 이전인 경우 로그가 표시되지 않습니다.

> **Note**
>
> 지표 필터는 로그 이벤트를 지정된 지표로 한 번 이상 전송하도록 보장하지만 중복 전송이 가끔 발생할 수 있습니다.

내용

- 개념
- 지표 필터에 대한 필터 패턴 구문
- 지표 필터 생성
- 지표 필터 나열
- 지표 필터 삭제

개념

각 지표 필터는 다음과 같은 키 요소들로 이루어져 있습니다.

기본값

로그를 수집했지만 일치하는 로그를 찾을 수 없는 기간 동안 지표 필터에 보고되는 값입니다. 이 값을 0으로 설정하면 이러한 모든 기간에서 데이터가 보고되어 일치하는 데이터가 없는 기간 때문에 지표가 '불규칙'해지는 것을 방지할 수 있습니다. 1분 동안 로그가 수집되지 않은 경우에는 아무 값도 보고되지 않습니다.

지표 필터로 생성한 지표에 차원을 할당하는 경우 해당 지표에 기본값을 할당할 수 없습니다.

차원

차원은 지표를 추가로 정의하는 키-값 쌍입니다. 지표 필터에서 생성된 지표에 차원을 할당할 수 있습니다. 차원은 지표에 대한 고유한 식별자의 일부이므로 로그에서 고유한 이름/값 쌍을 추출할 때마다 해당 지표의 새로운 변형이 생성되는 것입니다.

필터 패턴

CloudWatch Logs가 각 로그 이벤트에서 데이터를 해석하는 방법을 심볼로 설명한 것입니다. 예를 들어 로그 항목에는 타임스탬프, IP 주소, 문자열 등이 포함될 수 있습니다. 이러한 패턴을 사용하여 로그 파일에서 검색할 내용을 지정합니다.

척도 이름

모니터링된 로그 정보가 게시되는 CloudWatch 지표의 이름입니다. 예를 들어 ErrorCount라는 지표에 게시할 수 있습니다.

지표 네임스페이스

새 CloudWatch 지표에 대한 대상 네임스페이스입니다.

지표 값

일치하는 로그가 발견될 때마다 지표에 게시하는 숫자 값입니다. 예를 들어, 특정 단어(예: 'Error')의 출현 횟수를 계산할 경우 각 출현마다 이 값이 '1'이 됩니다. 전송된 바이트를 계산하는 경우 로그 이벤트에서 발견된 실제 바이트 수만큼 증가시킬 수 있습니다.

지표 필터에 대한 필터 패턴 구문

> **Note**
>
> 지표 필터와 CloudWatch Logs Insights 쿼리의 차이
> 지표 필터는 일치하는 로그를 찾을 때마다 지표 필터에 지정된 숫자 값이 추가된다는 점에서 CloudWatch 로그 인사이트 쿼리와 다릅니다. 자세한 내용은 지표 필터에 대한 지표 값 구성 단원을 참조하십시오.
> Amazon CloudWatch Logs Insights에서 쿼리 언어를 사용하여 로그 그룹을 쿼리하는 방법에 대한 자세한 내용은 CloudWatch Logs Insights 언어 쿼리 구문을 참조하세요.
> 일반 필터 패턴 예제
> 지표 필터와 구독 필터 및 필터 로그 이벤트에 적용할 수 있는 일반 필터 패턴 구문에 대한 자세한 내용은 지표 필터, 구독 필터 및 필터 로그 이벤트의 필터 패턴 구문을 참조하세요. 여기에 다음 예제가 포함되어 있습니다.

- 지원되는 정규 표현식 구문
- 비정형 로그 이벤트에서 일치하는 용어 검색
- JSON 로그 이벤트에서 일치하는 단어 검색
- 공백으로 구분된 로그 이벤트에서 일치하는 용어 검색

지표 필터를 사용하면 CloudWatch Logs로 들어오는 로그 데이터를 검색 및 필터링하고, 필터링된 로그 데이터에서 지표 관찰 내용을 추출하며, 데이터 포인트를 CloudWatch Logs 지표로 변환할 수 있습니다. CloudWatch Logs로 로그 데이터를 전송할 때 사용자가 로그 데이터에서 검색할 용어와 패턴을 정의합니다. 지표 필터는 로그 그룹에 할당이 되고, 로그 그룹에 할당된 모든 필터는 로그 스트림에 적용됩니다.

지표 필터와 일치하는 용어가 검색되면 지표 개수가 지정된 숫자 값만큼 증가합니다. 예를 들어 지표 필터를 생성하여 로그 이벤트에서 ERROR라는 단어의 개수를 계산할 수 있습니다.

지표에 측정 단위 및 차원을 할당할 수 있습니다. 예를 들어 로그 이벤트에서 ERROR라는 단어의 개수를 계산하는 지표 필터를 생성하는 경우, `ErrorCode`라는 차원을 지정하여 단어 ERROR를 포함하는 로그 이벤트의 총 수를 표시하고 보고된 오류 코드를 기준으로 데이터를 필터링할 수 있습니다.

> **Tip**
>
> 지표에 측정 단위를 할당하는 경우 올바른 단위를 지정해야 합니다. 나중에 단위를 변경하면 변경 사항이 적용되지 않을 수 있습니다. CloudWatch에서 지원하는 단위의 전체 목록은 Amazon CloudWatch API 참조의 MetricDatum을 참조하세요.

주제

- 지표 필터에 대한 지표 값 구성
- JSON 값 또는 공백으로 구분된 로그 이벤트의 지표로 차원 게시
- 로그 이벤트의 값을 사용하여 지표 값 증가

지표 필터에 대한 지표 값 구성

지표 필터를 생성할 때 필터 패턴을 정의하고 지표의 값과 기본값을 지정합니다. 지표 값을 숫자, 명명된 식별자 또는 숫자 식별자로 설정할 수 있습니다. 기본값을 지정하지 않으면 지표 필터가 일치하는

항목을 찾지 못할 경우 CloudWatch가 데이터를 보고하지 않습니다. 값이 0인 경우에도 기본값을 지정하는 것이 좋습니다. 기본값을 설정하면 CloudWatch가 데이터를 보다 정확하게 보고하고 CloudWatch가 불규칙한 지표를 집계하는 것을 방지합니다. CloudWatch는 매분마다 지표 값을 집계하고 보고합니다.

지표 필터가 로그 이벤트에서 일치하는 항목을 찾으면 지표 값을 기준으로 지표 수가 증가합니다. 지표 필터가 일치하는 항목을 찾지 못하면 CloudWatch가 지표의 기본값을 보고합니다. 예를 들어 로그 그룹이 1분마다 두 개의 레코드를 게시하고, 지표 값은 1, 기본값은 0입니다. 처음 1분 동안 지표 필터가 두 로그 레코드 모두에서 일치하는 항목을 찾을 경우 이 기간(분)의 지표 값은 2입니다. 두 번째 1분 동안 지표 필터가 두 레코드 모두에서 일치하는 항목을 찾지 못할 경우 해당 기간(분)의 기본값은 0입니다. 지표 필터가 생성하는 지표에 차원을 할당하는 경우 해당 지표에 대한 기본값을 지정할 수 없습니다.

정적 값 대신 로그 이벤트에서 추출한 값으로 지표를 증가시키도록 지표 필터를 설정할 수도 있습니다. 자세한 내용은 로그 이벤트의 값을 사용하여 지표 값 증가 섹션을 참조하세요.

JSON 값 또는 공백으로 구분된 로그 이벤트의 지표로 차원 게시

CloudWatch 콘솔 또는 AWS CLI를 사용하여 JSON 및 공백으로 구분된 로그 이벤트가 생성하는 지표로 차원을 게시하는 지표 필터를 생성할 수 있습니다. 차원은 이름/값 쌍이며 JSON 및 공백으로 구분된 필터 패턴에만 사용할 수 있습니다. JSON 및 공백으로 구분된 지표 필터를 최대 3개의 차원으로 생성할 수 있습니다. 차원에 대한 자세한 내용 및 지표에 차원을 할당하는 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.

- Amazon CloudWatch 사용 설명서의 차원
- Amazon CloudWatch Logs User Guide의 예제: Apache 로그에서 필드 추출 후 차원 할당

> **Important**
>
> 차원에는 사용자 지정 지표와 동일한 요금을 수집하는 값이 포함되어 있습니다. 예기치 않은 요금이 청구되는 것을 방지하려면 `IPAddress` 또는 `requestID`처럼 높은 카디널리티 필드를 차원으로 지정하지 마세요.
>
> 로그 이벤트에서 지표를 추출하면 사용자 지정 지표에 대한 요금이 청구됩니다. 실수로 높은 요금이 부과되는 것을 방지하기 위해 일정 시간 내에 지정한 차원에 대해 1000개의 다른 이름/값 쌍을 생성하는 경우 Amazon이 지표 필터를 사용 중지할 수 있습니다.
>
> 예상 요금을 알려주는 결제 경보를 생성할 수 있습니다. 자세한 내용은 예상 AWS 요금을 모니터링하기 위한 결제 경보 생성을 참조하세요.

JSON 로그 이벤트에서 지표로 차원 게시

다음 예제에는 JSON 지표 필터에서 차원을 지정하는 방법을 설명하는 코드 조각이 포함되어 있습니다.

Example: JSON log event

```
{
  "eventType": "UpdateTrail",
  "sourceIPAddress": "111.111.111.111",
  "arrayKey": [
        "value",
        "another value"
  ],
  "objectList": [
       {"name": "a",
         "id": 1
       },
       {"name": "b",
         "id": 2
       }
  ]

}
```

> **Note**
>
> 예제 JSON 로그 이벤트로 예제 지표 필터를 테스트하는 경우, 예제 JSON 로그를 한 줄에 입력해야 합니다.

Example: Metric filter

지표 필터는 JSON 로그 이벤트에 속성 `eventType` 및 `"sourceIPAddress"`가 포함될 때마다 지표를 증가시킵니다.

```
{ $.eventType = "*" && $.sourceIPAddress != 123.123.* }
```

JSON 지표 필터를 생성할 때 지표 필터에 있는 어떤 속성도 차원으로 지정할 수 있습니다. 예를 들어, eventType을 차원으로 설정하려면 다음을 사용합니다.

```
"eventType" : $.eventType
```

예제 지표에는 이름이 "eventType"인 차원이 포함되어 있으며, 예제 로그 이벤트의 차원 값은 "UpdateTrail"입니다.

공백으로 구분된 로그 이벤트에서 지표로 차원 게시

다음 예제에는 공백으로 구분된 지표 필터에서 차원을 지정하는 방법을 설명하는 코드 조각이 포함되어 있습니다.

Example: Space-delimited log event

```
127.0.0.1 Prod frank [10/Oct/2000:13:25:15 -0700] "GET /index.html HTTP/1.0" 404
 1534
```

Example: Metric filter

```
[ip, server, username, timestamp, request, status_code, bytes > 1000]
```

지표 필터는 공백으로 구분된 로그 이벤트에 필터에 지정된 필드가 포함된 경우 지표를 증가시킵니다. 예를 들어 지표 필터는 공백으로 구분된 로그 이벤트 예제에서 다음 필드 및 값을 찾습니다.

```
{
   "$bytes": "1534",
   "$status_code": "404",

   "$request": "GET /index.html HTTP/1.0",
   "$timestamp": "10/Oct/2000:13:25:15 -0700",
```

```
    "$username": "frank",
    "$server": "Prod",
    "$ip": "127.0.0.1"
}
```

공백으로 구분된 지표 필터를 생성할 때 지표 필터에 있는 어떤 속성도 차원으로 지정할 수 있습니다. 예를 들어, `server`을 차원으로 설정하려면 다음을 사용합니다.

```
"server" : $server
```

예제 지표 필터에는 이름이 `server`인 차원이 있으며, 예제 로그 이벤트의 차원 값은 `"Prod"`입니다.

Example: Match terms with AND (&&) and OR (||)

논리 연산자 AND('&&') 및 OR('||')를 사용하여 조건을 포함하는 공백으로 구분된 지표 필터를 생성할 수 있습니다. 다음 지표 필터는 이벤트의 첫 번째 단어가 WARN을 포함하는 모든 상위 문자열 또는 ERROR인 로그 이벤트를 반환합니다.

```
[w1=ERROR || w1=%WARN%, w2]
```

로그 이벤트의 값을 사용하여 지표 값 증가

로그 이벤트에서 찾은 숫자 값을 게시하는 지표 필터를 생성할 수 있습니다. 이 섹션의 절차에서는 다음 예제 지표 필터를 사용하여 JSON 로그 이벤트의 숫자 값을 지표에 게시하는 방법을 보여 줍니다.

```
{ $.latency = * } metricValue: $.latency
```

로그 이벤트에 값을 게시하는 지표 필터를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 로그(Logs)를 선택한 다음, 로그 그룹(Log groups)을 선택합니다.
3. 로그 그룹을 선택하거나 생성합니다.

   로그 그룹을 생성하는 방법에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 CloudWatch Logs에서 로그 그룹 생성을 참조하세요.

4. 작업을 선택한 후 지표 필터 생성을 선택합니다.
5. 필터 패턴(Filter Pattern)에 **`{ $.latency = * }`**를 입력하고 다음(Next)을 선택합니다.
6. 지표 이름에 myMetric을 입력합니다.
7. 지표 값에 **`$.latency`**를 입력합니다.
8. (선택 사항) 기본값(Default Value)에 0을 입력하고 다음(Next)을 선택합니다.

   값이 0인 경우에도 기본값을 지정하는 것이 좋습니다. 기본값을 설정하면 CloudWatch가 데이터를 보다 정확하게 보고하고 CloudWatch가 불규칙한 지표를 집계하는 것을 방지합니다. CloudWatch는 매분마다 지표 값을 집계하고 보고합니다.
9. 지표 필터 생성을 선택합니다.

예제 지표 필터가 예제 JSON 로그 이벤트의 용어 `"latency"`와 일치하고 숫자 값 50을 지표 myMetric에 게시합니다.

```
{
"latency": 50,
"requestType": "GET"
}
```

지표 필터 생성

다음 절차 및 예제에서는 지표 필터를 생성하는 방법을 보여줍니다.

예제

- 로그 그룹에 대한 지표 필터 생성
- 예제: 로그 이벤트 수 계산
- 예제: 단어의 출현 횟수 계산
- 예제: HTTP 404 코드 수 계산
- 예제: HTTP 4xx 코드 수 계산
- 예제: Apache 로그에서 필드 추출 후 차원 할당

로그 그룹에 대한 지표 필터 생성

로그 그룹에 대한 지표 필터를 만들려면 다음 단계를 따릅니다. 지표는 해당 몇 개의 데이터 포인트가 있기 전에는 표시되지 않습니다.

CloudWatch 콘솔을 사용하여 지표 필터를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택합니다.
4. Actions을 선택한 다음 지표 필터 생성을 선택합니다.
5. Filter pattern(필터 패턴)에 필터 패턴을 입력합니다. 자세한 내용은 지표 필터, 구독 필터, 필터 로그 이벤트 및 Live Tail에 대한 필터 패턴 구문 단원을 참조하십시오.
6. (선택 사항) 중앙 집중식 로그 그룹을 사용하는 경우 필터 선택 기준에서 소스 계정(`@aws.account`), 소스 리전(`@aws.region`) 또는 두 조건 모두를 기준으로 필터를 지정할 수 있습니다.
7. (선택 사항) 필터 패턴을 테스트하려면 패턴 테스트에 패턴을 테스트할 로그 이벤트를 하나 이상 입력합니다. 각 로그 이벤트는 한 줄에 형식이 지정되어야 합니다. 줄 바꿈은 Log event messages(로그 이벤트 메시지) 상자에서 로그 이벤트를 구분하는 데 사용됩니다.
8. Next(다음)를 선택하고 지표 필터의 이름을 입력합니다.
9. 지표 세부 정보에서 지표 네임스페이스에 지표가 게시될 CloudWatch 네임스페이스의 이름을 입력합니다. 네임스페이스가 아직 없는 경우 Create new(새로 생성)가 선택되는지 확인하세요.
10. 지표 이름에 새 지표의 이름을 입력합니다.
11. 지표 필터가 필터에 포함된 키워드의 출현 횟수를 계산하는 경우 지표 값에 1을 입력합니다. 이렇게 하면 키워드 중 하나를 포함하는 각 로그 이벤트에 대해 지표가 1씩 증가합니다.

    또는 토큰(예: **`$size`**)을 입력합니다. 이렇게 하면 `size` 필드가 포함된 모든 로그 이벤트에 대해 `size` 필드의 수치 값만큼씩 지표가 증가합니다.
12. (선택 사항) 단위에서 지표에 할당할 단위를 선택합니다. 단위를 지정하지 않을 경우 단위는 None으로 설정됩니다.
13. (선택 사항) 지표에 대한 세 개의 차원에 대해 각각 이름과 토큰을 입력합니다. 지표 필터가 생성하는 지표에 차원을 할당하는 경우 해당 지표에 대한 기본값을 할당할 수 없습니다.

    > **Note**
    >
    > 차원은 JSON 또는 공백으로 구분된 지표 필터에서만 지원됩니다.
14. 지표 필터 생성을 선택합니다. 생성한 지표 필터를 탐색 창에서 찾을 수 있습니다. Logs(로그)를 선택한 후 Log groups(로그 그룹)를 선택합니다. 지표 필터를 생성한 로그 그룹의 이름을 선택한 다음 지표 필터 탭을 선택합니다.

예제: 로그 이벤트 수 계산

가장 간단한 유형의 로그 이벤트 모니터링은 발생하는 로그 이벤트의 수를 계산하는 것입니다. 모든 이벤트의 수를 유지하거나, '하트비트' 스타일 모니터를 생성하거나, 단순히 지표 필터 생성을 연습하기 위해 계산할 수 있습니다.

다음 CLI 예제에서는 MyNamespace라는 CloudWatch 네임스페이스에서 EventCount라는 지표를 생성하기 위해 MyAppAccessCount라는 지표 필터가 MyApp/access.log라는 로그 그룹에 적용됩니다. 이 필터는 모든 로그 이벤트 콘텐츠와 일치하며 지표를 '1'씩 늘리도록 구성되어 있습니다.

CloudWatch 콘솔을 사용하여 지표 필터를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택합니다.
4. 작업, 지표 필터 생성을 선택합니다.
5. 필터 패턴 및 테스트할 로그 데이터 선택을 비워 둡니다.
6. 다음을 선택한 후 필터 이름에 **EventCount**를 입력합니다.
7. 지표 세부 정보의 지표 네임스페이스에 **MyNameSpace**를 입력합니다.
8. 지표 이름에 **MyAppEventCount**를 입력합니다.
9. 지표 값이 1인지 확인합니다. 이는 모든 로그 이벤트에 대해 개수가 1씩 증가하도록 지정합니다.
10. 기본값에 0을 입력하고 다음을 선택합니다. 기본값을 지정할 경우 로그 이벤트가 발생하지 않는 기간에도 데이터가 보고되므로 때때로 데이터가 존재하지 않아 지표가 불규칙해지는 것을 방지할 수 있습니다.
11. 지표 필터 생성을 선택합니다.

를 사용하여 지표 필터를 생성하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name EventCount \
  --filter-pattern " " \
  --metric-transformations \
```

```
  metricName=MyAppEventCount,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

모든 이벤트 데이터를 게재하여 이 새로운 정책을 테스트할 수 있습니다. 데이터 요소가 MyAppAccessEventCount 지표에 게시되어 있어야 합니다.

를 사용하여 이벤트 데이터를 게시하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs put-log-events \
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \
  --log-events \
    timestamp=1394793518000,message="Test event 1" \
    timestamp=1394793518000,message="Test event 2" \
    timestamp=1394793528000,message="This message also contains an Error"
```

예제: 단어의 출현 횟수 계산

로그 이벤트에는 연산의 성공 또는 실패 횟수같이 계산이 필요한 중요한 메시지가 종종 포함됩니다. 예를 들어 지정된 연산이 실패하면 오류가 발생하고 로그 파일에 오류가 기록될 수 있습니다. 이들 항목을 모니터링하여 오류의 트렌드를 이해하고 싶을 수 있습니다.

아래 예제에서는 Error라는 단어를 모니터링하기 위한 지표 필터가 생성됩니다. 정책이 생성되어 로그 그룹 MyApp/message.log에 추가되었습니다. CloudWatch Logs는 MyApp/message.log 네임스페이스에서 Error가 포함된 모든 이벤트에 대해 값 '1'을 사용하여 데이터 포인트를 CloudWatch 사용자 지정 지표 ErrorCount에 게시합니다. Error라는 단어가 포함된 이벤트가 없으면 값 0이 게시됩니다. CloudWatch 콘솔에서 이 데이터를 그래픽 처리할 때는 반드시 Sum 통계를 사용해야 합니다.

지표 필터를 생성한 후 CloudWatch 콘솔에서 지표를 볼 수 있습니다. 보려는 지표를 선택할 때 로그 그룹 이름과 일치하는 지표 네임스페이스를 선택합니다. 자세한 내용은 사용 가능한 지표 보기를 참조하세요.

CloudWatch 콘솔을 사용하여 지표 필터를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택합니다.
4. 작업, 지표 필터 생성을 선택합니다.
5. 필터 패턴에 **Error**를 입력합니다.

> **Note**
>
> 필터 패턴의 모든 항목들은 대소문자를 구분합니다.

6. (선택 사항) 필터 패턴을 테스트하려면 테스트 패턴에 패턴을 테스트하는 데 사용할 로그 이벤트를 하나 이상 입력합니다. 줄 바꿈은 로그 이벤트 메시지 상자에서 로그 이벤트를 구분할 때 사용하므로 각 로그 이벤트는 한 줄을 넘지 않아야 합니다.
7. 다음을 선택한 후 지표 할당 페이지에서 필터 이름에 **MyAppErrorCount**를 입력합니다.
8. 지표 세부 정보의 지표 네임스페이스에 MyNameSpace를 입력합니다.
9. 지표 이름에 ErrorCount를 입력합니다.
10. 지표 값이 1인지 확인합니다. 이는 'Error'를 포함하는 모든 로그 이벤트에 대해 개수가 1씩 증가하도록 지정합니다.
11. 기본값에 0을 입력하고 다음을 선택합니다.
12. 지표 필터 생성을 선택합니다.

를 사용하여 지표 필터를 생성하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs put-metric-filter \
  --log-group-name MyApp/message.log \
  --filter-name MyAppErrorCount \
  --filter-pattern 'Error' \
  --metric-transformations \
      metricName=ErrorCount,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

메시지에 'Error'라는 단어가 포함된 이벤트를 게재하여 이 새로운 정책을 테스트할 수 있습니다.

를 사용하여 이벤트를 게시하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다. 패턴은 대소문자를 구분한다는 점을 유의하세요.

```
aws logs put-log-events \
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \
  --log-events \
    timestamp=1394793518000,message="This message contains an Error" \
    timestamp=1394793528000,message="This message also contains an Error"
```

예제: HTTP 404 코드 수 계산

CloudWatch Logs를 사용하여 Apache 서버가 발견되지 않은 페이지에 대한 응답 코드인 HTTP 404 응답을 반환하는 횟수를 모니터링할 수 있습니다. 사이트 방문자들이 원하는 리소스를 찾지 못하는 빈도를 파악하기 위해 모니터링을 원할 수 있습니다. 로그 레코드는 각 로그 이벤트(사이트 방문)에 대해 다음 정보를 포함하도록 구성되었다고 가정합니다.

- 요청자 IP 주소
- RFC 1413 ID
- 사용자 이름
- 타임스탬프
- 요청된 리소스와 프로토콜이 포함된 요청 메서드
- 요청할 HTTP 응답 코드
- 요청 시 전송되는 바이트

예를 들어 다음과 같은 형태일 수 있습니다.

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 404 2326
```

다음 예제에서 알 수 있듯이, HTTP 404 오류에서 해당 구조의 이벤트와 매칭을 시도하는 규칙을 지정할 수 있습니다.

CloudWatch 콘솔을 사용하여 지표 필터를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 작업, 지표 필터 생성을 선택합니다.
4. 필터 패턴에 **[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]**를 입력합니다.
5. (선택 사항) 필터 패턴을 테스트하려면 테스트 패턴에 패턴을 테스트하는 데 사용할 로그 이벤트를 하나 이상 입력합니다. 줄 바꿈은 로그 이벤트 메시지 상자에서 로그 이벤트를 구분할 때 사용하므로 각 로그 이벤트는 한 줄을 넘지 않아야 합니다.
6. 다음을 선택하고 필터 이름에 HTTP404Errors를 입력합니다.
7. 지표 세부 정보의 지표 네임스페이스에 **MyNameSpace**를 입력합니다.

8. 지표 이름에 **ApacheNotFoundErrorCount**를 입력합니다.
9. 지표 값이 1인지 확인합니다. 이는 모든 404 오류 이벤트에 대해 개수가 1씩 증가하도록 지정합니다.
10. 기본값에 0을 입력하고 다음을 선택합니다.
11. 지표 필터 생성을 선택합니다.

를 사용하여 지표 필터를 생성하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name HTTP404Errors \
  --filter-pattern '[ip, id, user, timestamp, request, status_code=404, size]' \
  --metric-transformations \
      metricName=ApacheNotFoundErrorCount,metricNamespace=MyNamespace,metricValue=1
```

이 예제에서는 왼쪽/오른쪽 대괄호, 큰따옴표, 문자열 404 같은 리터럴 문자가 사용되었습니다. 패턴은 모니터링하려는 로그 이벤트의 전체 로그 이벤트 메시지와 일치해야 합니다.

describe-metric-filters 명령을 사용하여 지표 필터가 생성되었는지 확인할 수 있습니다. 다음과 유사한 출력 화면이 표시되어야 합니다.

```
aws logs describe-metric-filters --log-group-name MyApp/access.log

{
    "metricFilters": [
        {
            "filterName": "HTTP404Errors",
            "metricTransformations": [
                {
                    "metricValue": "1",
                    "metricNamespace": "MyNamespace",
                    "metricName": "ApacheNotFoundErrorCount"
                }
            ],
            "creationTime": 1399277571078,
            "filterPattern": "[ip, id, user, timestamp, request, status_code=404,
 size]"
        }
```

```
    ]
}
```

수동으로 몇 가지 이벤트를 게재할 수 있습니다.

```
aws logs put-log-events \
--log-group-name MyApp/access.log --log-stream-name hostname \
--log-events \
timestamp=1394793518000,message="127.0.0.1 - bob [10/Oct/2000:13:55:36 -0700] \"GET /
apache_pb.gif HTTP/1.0\" 404 2326" \
timestamp=1394793528000,message="127.0.0.1 - bob [10/Oct/2000:13:55:36 -0700] \"GET /
apache_pb2.gif HTTP/1.0\" 200 2326"
```

이러한 샘플 로그 이벤트를 게재하고 나면 그 즉시 CloudWatch 콘솔에서 ApacheNotFoundErrorCount로 명명된 지표를 검색할 수 있습니다.

예제: HTTP 4xx 코드 수 계산

이전 예제에서와 마찬가지로 사용자는 웹 서비스 액세스 로그를 모니터링하고 HTTP 응답 코드 수준을 모니터링하려고 할 수 있습니다. 예를 들어 HTTP 400 수준 오류를 모두 모니터링하고 싶을 수 있습니다. 그러나 모든 반환 코드에 새로운 지표 필터를 지정하고 싶지 않을 수 있습니다.

다음 예제는 예제: HTTP 404 코드 수 계산 예제에서 Apache 액세스 로그 형식을 사용하여 액세스 로그로부터 400 수준의 모든 HTTP 코드 응답을 포함하는 지표를 생성하는 방법을 보여줍니다.

CloudWatch 콘솔을 사용하여 지표 필터를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. Apache 서버의 로그 그룹 이름을 선택합니다.
4. 작업, 지표 필터 생성을 선택합니다.
5. 필터 패턴에 **[ip, id, user, timestamp, request, status_code=4*, size]**를 입력합니다.
6. (선택 사항) 필터 패턴을 테스트하려면 테스트 패턴에 패턴을 테스트하는 데 사용할 로그 이벤트를 하나 이상 입력합니다. 줄 바꿈은 로그 이벤트 메시지 상자에서 로그 이벤트를 구분할 때 사용하므로 각 로그 이벤트는 한 줄을 넘지 않아야 합니다.
7. 다음을 선택한 후 필터 이름에 **HTTP4xxErrors**를 입력합니다.
8. 지표 세부 정보의 지표 네임스페이스에 **MyNameSpace**를 입력합니다.

9. 지표 이름에 HTTP4xxErrors를 입력합니다.
10. 지표 값에 1을 입력합니다. 이는 4xx 오류를 포함하는 모든 로그 이벤트에 대해 개수가 1씩 증가하도록 지정합니다.
11. 기본값에 0을 입력하고 다음을 선택합니다.
12. 지표 필터 생성을 선택합니다.

를 사용하여 지표 필터를 생성하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name HTTP4xxErrors \
  --filter-pattern '[ip, id, user, timestamp, request, status_code=4*, size]' \
  --metric-transformations \
  metricName=HTTP4xxErrors,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

PUT 이벤트 호출에서 다음 데이터를 사용하여 이 규칙을 테스트할 수 있습니다. 이전 예제에서 모니터링 규칙을 제거하지 않았다면 서로 다른 두 개의 지표가 생성됩니다.

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200 3
```

예제: Apache 로그에서 필드 추출 후 차원 할당

때로는 수를 계산하는 대신 지표 값으로 개별 로그 이벤트 내의 값을 사용하는 것이 도움이 됩니다. 이 예제는 Apache 웹 서버에서 전송된 바이트를 측정하는 지표를 생성하기 위해 추출 규칙을 만드는 방법을 보여줍니다.

이 예제에서는 생성 중인 지표에 차원을 할당하는 방법도 보여줍니다.

CloudWatch 콘솔을 사용하여 지표 필터를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.

3. Apache 서버의 로그 그룹 이름을 선택합니다.
4. 작업, 지표 필터 생성을 선택합니다.
5. 필터 패턴에 **[ip, id, user, timestamp, request, status_code, size]**를 입력합니다.
6. (선택 사항) 필터 패턴을 테스트하려면 테스트 패턴에 패턴을 테스트하는 데 사용할 로그 이벤트를 하나 이상 입력합니다. 줄 바꿈은 로그 이벤트 메시지 상자에서 로그 이벤트를 구분할 때 사용하므로 각 로그 이벤트는 한 줄을 넘지 않아야 합니다.
7. 다음을 선택한 후 필터 이름에 **size**를 입력합니다.
8. 지표 세부 정보의 지표 네임스페이스에 **MyNameSpace**를 입력합니다. 새 네임스페이스이기 때문에 새로 생성이 선택되어 있습니다.
9. 지표 이름에 **BytesTransferred**를 입력합니다.
10. 지표 값에 **$size**를 입력합니다.
11. 단위로 바이트를 선택합니다.
12. 차원 이름에 **IP**를 입력합니다.
13. 차원 값에 **$ip**를 입력하고 다음을 선택합니다.
14. 지표 필터 생성을 선택합니다.

를 사용하여이 지표 필터를 생성하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs put-metric-filter \
--log-group-name MyApp/access.log \
 --filter-name BytesTransferred \
 --filter-pattern '[ip, id, user, timestamp, request, status_code, size]' \
 --metric-transformations  \
 metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue='$size'
```

```
aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name BytesTransferred \
--filter-pattern '[ip, id, user, timestamp, request, status_code, size]' \
--metric-transformations  \
metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue='$size',unit=Bytes,dimensio
$ip}}'
```

> **Note**
>
> 이 명령에서는 이 형식을 사용하여 여러 차원을 지정합니다.
>
> ```
> aws logs put-metric-filter \
> --log-group-name my-log-group-name \
> --filter-name my-filter-name \
> --filter-pattern 'my-filter-pattern' \
> --metric-transformations  \
> metricName=my-metric-name,metricNamespace=my-metric-namespace,metricValue=my-token,unit=unit,dimensions='{dimension1=$dim,dimension2=$dim2,dim3=$dim3}'
> ```

PUT 로그 이벤트 호출에서 다음 데이터를 사용하여 이 규칙을 테스트할 수 있습니다. 이전 예제에서 모니터링 규칙을 제거하지 않았다면 서로 다른 두 개의 지표가 생성됩니다.

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200 3
```

지표 필터 나열

로그 그룹에 속한 모든 지표 필터를 나열할 수 있습니다.

CloudWatch 콘솔을 사용하여 지표 필터를 나열하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 콘텐츠 창에서 로그 그룹 목록의 지표 필터 열에서 필터 수를 선택합니다.

   로그 그룹 > 필터 대상 화면에 해당 로그 그룹과 연관된 모든 지표 필터들이 나열됩니다.

를 사용하여 지표 필터를 나열하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs describe-metric-filters --log-group-name MyApp/access.log
```

다음은 예제 출력입니다.

```
{
    "metricFilters": [
        {
            "filterName": "HTTP404Errors",
            "metricTransformations": [
                {
                    "metricValue": "1",
                    "metricNamespace": "MyNamespace",
                    "metricName": "ApacheNotFoundErrorCount"
                }
            ],
            "creationTime": 1399277571078,
            "filterPattern": "[ip, id, user, timestamp, request, status_code=404,
 size]"
        }
    ]
}
```

지표 필터 삭제

이름과 소속 로그 그룹으로 정책을 식별할 수 있습니다.

CloudWatch 콘솔을 사용하여 지표 필터를 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 콘텐츠 창의 지표 필터 열에서 로그 그룹에 대한 지표 필터 개수를 선택합니다.
4. 지표 필터 화면에서 삭제할 필터 이름 오른쪽에 있는 확인란을 선택합니다. 그런 다음 삭제를 선택합니다.
5. 확인 메시지가 나타나면 삭제를 선택합니다.

를 사용하여 지표 필터를 삭제하려면 AWS CLI

명령 프롬프트에서 다음 명령을 실행합니다.

```
aws logs delete-metric-filter --log-group-name MyApp/access.log \
 --filter-name MyFilterName
```

구독을 통한 로그 데이터 실시간 처리

구독을 사용하여 CloudWatch Logs에서 나온 로그 이벤트의 실시간 피드에 액세스하고, 사용자 지정 처리, 분석 또는 다른 시스템으로의 로드를 위해 다른 서비스(예: Amazon Kinesis 스트림, Amazon Data Firehose 스트림 또는 AWS Lambda)에 이를 전송할 수 있습니다. 로그 이벤트가 수신 서비스로 전송되면 base64로 인코딩되고 gzip 형식으로 압축됩니다.

CloudWatch Logs 중앙 집중화를 사용하여 여러 계정 및 리전의 로그 데이터를 중앙 위치로 복제할 수도 있습니다. 자세한 내용은 교차 계정 교차 리전 로그 중앙 집중화 단원을 참조하십시오.

로그 이벤트 구독을 시작하려면 이벤트가 전달될 Amazon Kinesis Data Streams 스트림과 같은 수신 리소스를 생성합니다. 구독 필터는 AWS 리소스에 전달되는 로그 이벤트와 일치하는 로그 이벤트를 전송할 대상에 대한 정보를 필터링하는 데 사용할 필터 패턴을 정의합니다. 로그 이벤트는 수집 직후, 일반적으로 3분 이내에 수신 리소스로 전송됩니다.

> **Note**
>
> 구독이 있는 로그 그룹이 로그 변환을 사용하는 경우 필터 패턴은 로그 이벤트의 변환된 버전과 비교됩니다. 자세한 내용은 수집 중 로그 변환 단원을 참조하십시오.

계정 수준 및 로그 그룹 수준에서 구독을 생성할 수 있습니다. 각 계정은 리전별 하나의 계정 수준 구독 필터를 가질 수 있습니다. 각 로그 그룹에는 최대 5개의 구독 필터가 연결될 수 있습니다.

> **Note**
>
> 대상 서비스가 스로틀링 예외 또는 재시도 가능한 서비스 예외(예: HTTP 5xx)와 같은 재시도 가능한 오류를 반환하는 경우 CloudWatch Logs는 최대 24시간 동안 계속해서 전송을 다시 시도합니다. AccessDeniedException 또는 ResourceNotFoundException과 같이 다시 시도할 수 없는 오류인 경우 CloudWatch Logs는 다시 전송을 시도하지 않습니다. 이러한 경우 구독 필터가 최대 10분 동안 비활성화된 다음 CloudWatch Logs가 대상에 로그 전송을 재시도합니다. 이 비활성화된 기간 동안에는 로그를 건너뜁니다.

또한 CloudWatch Logs는 구독 서비스로의 로그 이벤트 전달에 대한 CloudWatch 지표를 생성합니다. 자세한 내용은 CloudWatch 지표를 사용한 모니터링 단원을 참조하십시오.

CloudWatch Logs 구독을 사용하여 로그 데이터를 거의 실시간으로 Amazon OpenSearch Service 클러스터로 스트리밍할 수도 있습니다. 자세한 내용은 Streaming CloudWatch Logs data to Amazon OpenSearch Service을 참조하세요.

구독은 Standard 로그 클래스의 로그 그룹에서만 지원됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스를 참조하세요.

> **Note**
>
> 구독 필터는 로그 이벤트를 배칭하여 전송을 최적화하고 대상에 대한 직접 호출의 양을 줄일 수 있습니다. 배칭은 보장되지는 않지만 가능하면 사용됩니다.

일정에 따라 로그 데이터를 일괄 처리하고 분석하려면를 사용하는 것이 좋습니다예약된 쿼리를 사용하여 로그 분석 자동화. 예약된 쿼리는 CloudWatch Logs Insights 쿼리를 자동으로 실행하고 Amazon S3 버킷 또는 Amazon EventBridge 이벤트 버스와 같은 대상에 결과를 전송합니다.

> **Note**
>
> 구독 필터는 이벤트 전송을 한 번 이상 보장하지만 중복 이벤트가 가끔 발생할 수 있습니다.

내용

- 개념
- 로그 그룹 수준 구독 필터
- 계정 수준 구독 필터
- 교차 계정 교차 리전 구독
- 혼동된 대리자 방지
- 로그 재귀 방지

개념

각 구독 필터는 다음과 같은 키 요소로 이루어져 있습니다.

필터 패턴

CloudWatch Logs가 대상 AWS 리소스로 전송되는 항목을 제한하는 필터링 표현식과 함께 각 로그 이벤트의 데이터를 해석하는 방법에 대한 심볼 설명입니다. 필터 패턴 구문에 대한 자세한 내용은 지표 필터, 구독 필터, 필터 로그 이벤트 및 Live Tail에 대한 필터 패턴 구문을 참조하세요.

대상 ARN

구독 피드의 대상으로 사용할 Amazon Kinesis Data Streams 스트림, Firehose 스트림 또는 Lambda 함수의 Amazon 리소스 이름(ARN)입니다.

역할 ARN

선택한 대상으로 데이터를 입력하기 위해 CloudWatch Logs에 필요한 권한을 부여하는 IAM 역할입니다. 이 역할은 Lambda 대상에는 필요하지 않습니다. CloudWatch Logs가 Lambda 함수 자체에 대한 액세스 제어 설정에 필요한 권한을 얻을 수 있기 때문입니다.

> **Note**
>
> 계정에서 역할 관리자가 활성화된 경우 CloudWatch는 역할을 자동으로 연결하며, 여기에 설명된 역할 옵션(예: 새 역할 생성 버튼)은 사용자 지정 옵션으로 대체됩니다. 다른 역할을 사용하려면 사용자 지정을 선택합니다. IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

배포

대상이 Amazon Kinesis Data Streams의 스트림일 때 해당 대상으로 로그 데이터를 배포하는 데 사용되는 방법입니다. 기본적으로 로그 스트림에 따라 로그 데이터가 그룹화됩니다. 보다 균등한 배포를 위해 로그 데이터를 무작위로 그룹화할 수 있습니다.

로그 그룹 수준 구독의 경우 다음 키 요소도 포함됩니다.

log_group_name

구독 필터에 연결되는 로그 그룹. 이 로그 그룹에 업로드되는 모든 로그 이벤트는 구독 필터를 따르게 되며, 필터와 일치하는 로그 이벤트는 일치하는 로그 이벤트를 수신하는 대상 서비스로 전송됩니다.

계정 수준 구독의 경우 다음 키 요소도 포함됩니다.

선택 기준

계정 수준 구독 필터가 적용되는 로그 그룹을 선택하는 데 사용되는 기준입니다. 이를 지정하지 않으면 계정 수준 구독 필터가 계정의 모든 로그 그룹에 적용됩니다. 이 필드는 무한 로그 루프를 방지하는 데 사용됩니다. 무한 로그 루프 문제에 대한 자세한 내용은 로그 재귀 방지을 참조하세요.

선택 기준의 크기 한도는 25KB입니다.

중앙 집중식 로그 그룹의 경우 다음 키 요소도 포함됩니다. 이러한 요소를 필드 선택 기준으로 사용하여 로그 데이터의 소스를 식별할 수 있으므로 중앙 집중식 로그에서 파생된 지표를 보다 세밀하게 필터링하고 분석할 수 있습니다.

@aws.account

이 필드는 로그 이벤트가 시작된 AWS 계정 ID를 식별합니다.

@aws.region

이 필드는 로그 이벤트가 생성된 AWS 리전을 식별합니다.

@source.log

이 필드는 로그 이벤트가 시작된 소스 로그 그룹을 식별합니다.

로그 그룹 수준 구독 필터

Amazon Kinesis Data Streams, Amazon AWS Lambda Data Firehose 또는 Amazon OpenSearch Service에서 구독 필터를 사용할 수 있습니다. 구독 필터를 통해 서비스로 전송되는 로그는 base64로 인코딩되고 gzip 형식으로 압축됩니다. 에서 중앙 집중식 로그를 사용하는 경우 `@aws.account`, `@aws.region`및 `@source.log` 시스템 필드를 내보내 조직의 어떤 계정, 리전 및 소스 로그 그룹에서 어떤 데이터가 오는지 식별할 AWS Organizations수 있습니다. 이 섹션에서는 Firehose, Lambda, Amazon Kinesis Data Streams 및 OpenSearch Service로 로그 데이터를 전송하는 CloudWatch Logs 구독 필터를 생성하는 데 따를 수 있는 예제를 제공합니다.

> **Note**
>
> 로그 데이터를 검색하려면 필터 및 패턴 구문을 참조하세요.

예제

- 예제 1: Amazon Kinesis Data Streams를 사용한 구독 필터
- 예제 2:를 사용한 구독 필터 AWS Lambda
- 예제 3: Amazon Data Firehose에 대한 구독 필터
- 예제 4: Amazon OpenSearch Service를 사용한 구독 필터

예제 1: Amazon Kinesis Data Streams를 사용한 구독 필터

다음 예제에서는 AWS CloudTrail 이벤트를 포함하는 로그 그룹에 구독 필터를 연결합니다. 구독 필터는 "RootAccess"라는 Amazon Kinesis Data Streams의 스트림에 "Root" AWS 자격 증명에 의해 수행된 모든 로깅된 활동을 전달합니다.RootAccess." CloudWatch Logs로 AWS CloudTrail 이벤트를 전송하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudWatch Logs로 CloudTrail CloudTrail 이벤트 전송을 참조하세요.

> **Note**
>
> 스트림을 생성하기 전에 생성할 로그 데이터의 볼륨을 계산합니다. 이 볼륨을 처리하기에 충분한 샤드로 스트림을 생성해야 합니다. 스트림에 샤드가 충분하지 않은 경우 로그 스트림에 스로틀링이 발생합니다. 스트림 볼륨 제한 사항에 대한 자세한 내용은 Quotas and Limits을 참조하세요.
>
> 제한된 결과물은 최대 24시간 동안 재시도됩니다. 24시간이 지나면 실패한 결과물이 삭제됩니다.
>
> 스로틀링 현상의 위험을 완화하려면 다음 단계를 수행합니다.
>
> - PutSubscriptionFilter 또는 put-subscription-filter를 사용하여 구독 필터를 생성할 때 `distribution`에 `random`을 지정합니다. 기본적으로 스트림 필터 배포는 로그 스트림에 의해 이루어지며, 이로 인해 스로틀링 현상이 발생할 수 있습니다.
> - CloudWatch 지표를 사용하여 스트림 모니터링합니다. 이렇게 하면 제한을 식별하고 그에 따라 구성을 조정할 수 있습니다. 예를 들어, `DeliveryThrottling` 지표는 데이터를 구독 대상으로 전달할 때 CloudWatch Logs에 스로틀링이 발생한 로그 이벤트의 수를 추적하는 데 사용될 수 있습니다. 모니터링에 대한 자세한 내용은 CloudWatch 지표를 사용한 모니터링을 참조하세요.
> - Amazon Kinesis Data Streams에서 스트림에 대한 온디맨드 용량 모드를 사용합니다. 온디맨드 모드는 워크로드가 확장 또는 축소될 때 즉시 워크로드를 수용합니다. 온디맨드 용량 모드에 대한 자세한 내용은 온디맨드 모드를 참조하세요.

- Amazon Kinesis Data Streams의 스트림 용량과 일치하도록 CloudWatch 구독 필터 패턴을 제한합니다. 스트림에 너무 많은 데이터를 전송하는 경우 필터 크기를 줄이거나 필터 기준을 조정해야 할 수 있습니다.

Amazon Kinesis Data Streams에 대한 구독 필터를 생성하려면

1. 다음 명령을 사용하여 대상 스트림을 생성합니다.

```
$ C:\>  aws kinesis create-stream --stream-name "RootAccess" --shard-count 1
```

2. 스트림이 활성 상태가 될 때까지 기다립니다(1~2분 정도 소요). 다음 Amazon Kinesis Data Streams describe-stream 명령을 사용하여 StreamDescription.StreamStatus 속성을 확인할 수 있습니다. 또한 이후 단계에서 필요할 수 있기 때문에 StreamDescription.StreamARN 값을 적어둡니다.

```
aws kinesis describe-stream --stream-name "RootAccess"
```

다음은 예제 출력입니다.

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "RootAccess",
        "StreamARN": "arn:aws:kinesis:us-east-1:123456789012:stream/RootAccess",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "49551135218688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 스트림에 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 신뢰 정책을 파일로 생성해야 합니다(예: ~/TrustPolicyForCWL-Kinesis.json). 텍스트 편집기를 사용하여 이 정책을 생성합니다. IAM 콘솔을 사용하여 정책을 생성하지 마세요.

   이 정책에는 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 aws:SourceArn 글로벌 조건 컨텍스트 키가 포함되어 있습니다. 자세한 내용은 혼동된 대리자 방지 섹션을 참조하세요.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
     }
   }
}
```

4. create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이후 단계에서 필요할 수 있기 때문에 반환된 Role.Arn 값을 적어둡니다.

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document
 file://~/TrustPolicyForCWL-Kinesis.json
```

   다음은 출력의 예제입니다.

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": { "arn:aws:logs:region:123456789012:*" }
                    }
                }
            }
        },
```

```
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
    }
}
```

5. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 파일에 권한 정책을 생성합니다(예: ~/PermissionsForCWL-Kinesis.json). 텍스트 편집기를 사용하여 이 정책을 생성합니다. IAM 콘솔을 사용하여 정책을 생성하지 마세요.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:123456789012:stream/RootAccess"
    }
  ]
}
```

6. 다음 put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy  --role-name CWLtoKinesisRole  --policy-name Permissions-Policy-For-CWL  --policy-document file://~/PermissionsForCWL-Kinesis.json
```

7. 스트림이 Active 상태에 있고 IAM 역할을 생성하고 나면 CloudWatch Logs 구독 필터를 생성할 수 있습니다. 그 즉시 구독 필터는 실시간으로 선택한 로그 그룹에서 스트림으로 로그 데이터를 이동시키기 시작합니다.

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail/logs" \
    --filter-name "RootAccess" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:kinesis:region:123456789012:stream/RootAccess" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
```

8. 구독 필터를 설정하고 나면 CloudWatch Logs는 들어오는 모든 로그 이벤트 중에서 필터 패턴과 일치하는 이벤트를 스트림으로 전달합니다. Amazon Kinesis Data Streams 샤드 반복자를 잡고 Amazon Kinesis Data Streams get-records 명령을 사용하여 일부 Amazon Kinesis Data Streams 레코드를 가져와서이 문제가 발생하고 있는지 확인할 수 있습니다.

```
aws kinesis get-shard-iterator --stream-name RootAccess --shard-id
 shardId-000000000000 --shard-iterator-type TRIM_HORIZON
```

```
{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
```

Amazon Kinesis Data Streams가 데이터를 반환하기 전에이 호출을 몇 번 수행해야 할 수 있습니다.

레코드 어레이에서 응답을 확인할 수 있습니다. Amazon Kinesis Data Streams 레코드의 데이터 속성은 base64로 인코딩되고 gzip 형식으로 압축됩니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

```
echo -n "<Content of Data>" | base64 -d | zcat
```

디코딩 및 압축 해제된 base64 데이터는 다음 구조를 가진 JSON으로 포맷됩니다.

```
{
    "owner": "111111111111",
    "logGroup": "CloudTrail/logs",
    "logStream": "111111111111_CloudTrail/logs_us-east-1",
    "subscriptionFilters": [
        "Destination"
    ],
    "messageType": "DATA_MESSAGE",
    "logEvents": [
        {
```

```
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        }
    ]
}
```

위의 데이터 구조에서 키 요소는 다음과 같습니다.

owner

발신 로그 데이터의 AWS 계정 ID입니다.

logGroup

원본 로그 데이터의 로그 그룹 이름.

logStream

원본 로그 데이터의 로그 스트림 이름.

subscriptionFilters

원본 로그 데이터과 일치한 구독 필터 이름 목록입니다.

messageType

데이터 메시지는 'DATA_MESSAGE' 유형을 사용합니다. CloudWatch Logs는 주로 대상에 연결할 수 있는지 확인하기 위해 "CONTROL_MESSAGE" 유형의 Amazon Kinesis Data Streams 레코드를 내보낼 수 있습니다.

logEvents

로그 이벤트 레코드의 배열로 표현되는 실제 로그 데이터. 'ID' 속성은 모든 로그 이벤트의 고유 식별자입니다.

예제 2:를 사용한 구독 필터 AWS Lambda

이 예제에서는 AWS Lambda 함수에 로그 데이터를 전송하는 CloudWatch Logs 구독 필터를 생성합니다.

> **Note**
>
> Lambda 함수를 생성하기 전에 생성할 로그 데이터의 볼륨을 계산합니다. 이 볼륨을 처리할 수 있는 함수를 생성해야 합니다. 함수에 볼륨이 충분하지 않은 경우 로그 스트림에 스로틀링이 발생합니다. AWS Lambda 제한 사항에 대한 자세한 내용은 Lambda 제한 사항을 참조하세요.

Lambda에 대한 구독 필터를 생성하려면

1. AWS Lambda 함수를 생성합니다.

   Lambda 실행 역할이 설정되었는지 확인합니다. 자세한 내용은 AWS Lambda Developer Guide의 Step 2.2: Create an IAM Role (execution role)을 참조하세요.
2. 다음 콘텐츠로 텍스트 편집기를 열고 `helloWorld.js`라는 파일을 생성합니다.

```
var zlib = require('zlib');
exports.handler = function(input, context) {
    var payload = Buffer.from(input.awslogs.data, 'base64');
    zlib.gunzip(payload, function(e, result) {
        if (e) {
            context.fail(e);
        } else {
            result = JSON.parse(result.toString());
            console.log("Event Data:", JSON.stringify(result, null, 2));
            context.succeed();
        }
    });
};
```

3. helloWorld.js 파일을 압축하고 `helloWorld.zip`라는 이름으로 저장합니다.

4. 역할이 첫 단계에서 설정한 Lambda 실행 역할인 경우에는 다음 명령을 사용합니다.

```
aws lambda create-function \
    --function-name helloworld \
    --zip-file fileb://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs12.x
```

5. CloudWatch Logs에 함수를 실행할 권한을 부여합니다. 다음 명령을 사용하여 자리 표시자 계정을 자체 계정으로, 자리 표시자 그룹을 처리할 로그 그룹으로 바꿉니다.

```
aws lambda add-permission \
    --function-name "helloworld" \
    --statement-id "helloworld" \
    --principal "logs.amazonaws.com" \
    --action "lambda:InvokeFunction" \
    --source-arn "arn:aws:logs:region:123456789123:log-group:TestLambda:*" \
    --source-account "123456789012"
```

6. 다음 명령을 사용하여 구독 필터를 생성하여 자리 표시자 계정을 자체 계정으로, 자리 표시자 그룹을 처리할 로그 그룹으로 바꿉니다.

```
aws logs put-subscription-filter \
    --log-group-name myLogGroup \
    --filter-name demo \
    --filter-pattern "" \
    --destination-arn arn:aws:lambda:region:123456789123:function:helloworld
```

7. (선택 사항) 샘플 로그 이벤트를 사용하여 테스트합니다. 명령 프롬프트에서 다음 명령을 실행하여 구독된 스트림으로 간단한 로그 메시지를 보냅니다.

   Lambda 함수의 출력을 보려면 /aws/lambda/helloworld에 출력이 표시되는 Lambda 함수를 탐색합니다.

```
aws logs put-log-events --log-group-name myLogGroup --log-stream-name stream1 --
log-events "[{\"timestamp\":<CURRENT TIMESTAMP MILLIS> , \"message\": \"Simple
 Lambda Test\"}]"
```

   Lambda의 배열로 응답을 확인할 수 있습니다. Lambda 레코드의 데이터 속성은 base64로 인코딩되고 gzip 형식으로 압축됩니다. Lambda가 수신하는 실제 페이로드는 { "awslogs":

{"data": "BASE64ENCODED_GZIP_COMPRESSED_DATA"} } 형식을 따릅니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

```
echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
```

디코딩 및 압축 해제된 base64 데이터는 다음 구조를 가진 JSON으로 포맷됩니다.

```
{
    "owner": "123456789012",
    "logGroup": "CloudTrail",
    "logStream": "123456789012_CloudTrail_us-east-1",
    "subscriptionFilters": [
        "Destination"
    ],
    "messageType": "DATA_MESSAGE",
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        }
    ]
}
```

위의 데이터 구조에서 키 요소는 다음과 같습니다.

owner

발신 로그 데이터의 AWS 계정 ID입니다.

logGroup

원본 로그 데이터의 로그 그룹 이름.

logStream

원본 로그 데이터의 로그 스트림 이름.

subscriptionFilters

원본 로그 데이터과 일치한 구독 필터 이름 목록입니다.

messageType

데이터 메시지는 'DATA_MESSAGE' 유형을 사용합니다. 때로 CloudWatch Logs는 주로 대상이 도달 가능한지 확인하기 위한 목적으로 'CONTROL_MESSAGE' 유형을 가진 Lambda 레코드를 출력할 수 있습니다.

logEvents

로그 이벤트 레코드의 배열로 표현되는 실제 로그 데이터. 'ID' 속성은 모든 로그 이벤트의 고유 식별자입니다.

예제 3: Amazon Data Firehose에 대한 구독 필터

이 예제에서는 수신 로그 이벤트 중에서 정의된 필터와 일치하는 이벤트를 Amazon Data Firehose 전송 스트림에 전송하는 CloudWatch Logs 구독을 생성합니다. CloudWatch Logs에서 Amazon Data Firehose로 전송된 데이터는 이미 gzip 6 수준의 압축이 되었기 때문에 Firehose 전송 스트림 내에서 압축을 사용할 필요가 없습니다. 그런 다음 Firehose의 압축 해제 기능을 사용하여 로그를 자동으로 압축 해제할 수 있습니다. 자세한 내용은 Firehose에 CloudWatch Logs 전송을 참조하세요.

> **Note**
>
> Firehose 스트림을 생성하기 전에 생성될 로그 데이터의 볼륨을 계산합니다. 이 볼륨을 처리할 수 있는 Firehose 스트림을 생성해야 합니다. 스트림이 볼륨을 처리할 수 없는 경우 로그 스트림에 스로틀링이 발생합니다. Firehose 스트림 볼륨 제한 사항에 대한 자세한 내용은 Amazon Data Firehose 데이터 제한 사항을 참조하세요.

Firehose 구독 필터를 생성하려면

1. Amazon Simple Storage Service(Amazon S3) 버킷을 생성합니다. CloudWatch Logs를 위해 특별히 생성한 버킷을 사용하는 것이 좋습니다. 그러나 기존 버킷을 사용하고 싶으면 2단계로 건너뛸 수 있습니다.

   다음 명령을 실행하여 자리 표시자 리전을 사용하고자 하는 리전으로 바꿉니다.

   ```
   aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration
    LocationConstraint=region
   ```

   다음은 예제 출력입니다.

   ```
   {
       "Location": "/amzn-s3-demo-bucket2"
   }
   ```

2. Amazon S3 버킷에 데이터를 입력하는 데 필요한 권한을 Amazon Data Firehose에 부여하는 IAM 역할을 생성합니다.

   자세한 내용은 Amazon Data Firehose 개발자 가이드의 Amazon Data Firehose를 사용하여 액세스 제어를 참조하세요.

   먼저 텍스트 편집기를 사용하여 다음과 같이 ~/TrustPolicyForFirehose.json 파일에 신뢰 정책을 생성합니다.

   ```
   {
     "Statement": {
       "Effect": "Allow",
       "Principal": { "Service": "firehose.amazonaws.com" },
       "Action": "sts:AssumeRole"
       }
   }
   ```

3. create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이후 단계에서 필요할 수 있기 때문에 반환된 Role.Arn 값을 적어둡니다.

   ```
   aws iam create-role \
    --role-name FirehosetoS3Role \
    --assume-role-policy-document file://~/TrustPolicyForFirehose.json
   ```

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "FirehosetoS3Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/FirehosetoS3Role"
    }
}
```

4. Firehose가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 ~/PermissionsForFirehose.json 파일엣 권한 정책을 생성합니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "s3:AbortMultipartUpload",
          "s3:GetBucketLocation",
          "s3:GetObject",
          "s3:ListBucket",
          "s3:ListBucketMultipartUploads",
          "s3:PutObject" ],
      "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket2",
          "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}
```

5. 다음 put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name Permissions-
Policy-For-Firehose --policy-document file://~/PermissionsForFirehose.json
```

6. 다음과 같이 대상 Firehose 전송 시스템을 생성하여 RoleARN 및 BucketARN의 자리 표시자 값을 생성한 역할 및 버킷 ARN으로 바꿉니다.

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

Firehose는 제공된 Amazon S3 객체에서 YYYY/MM/DD/HH UTC 시간 형식의 접두사를 자동으로 사용합니다. 시간 형식 접두사 앞에 추가할 또 다른 접두사를 지정할 수 있습니다. 슬래시(/)로 끝난 접두사는 Amazon S3 버킷에서 자리 표시자로 표시됩니다.

7. 스트림이 활성 상태가 될 때까지 기다립니다(몇 분 소요). Firehose describe-delivery-stream 명령을 사용하여 DeliveryStreamDescription.DeliveryStreamStatus 속성을 확인할 수 있습니다. 또한 이후 단계에서 필요할 수 있기 때문에 DeliveryStreamDescription.DeliveryStreamARN 값을 적어 둡니다.

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
    "DeliveryStreamDescription": {
        "HasMoreDestinations": false,
        "VersionId": "1",
        "CreateTimestamp": 1446075815.822,
        "DeliveryStreamARN": "arn:aws:firehose:us-
east-1:123456789012:deliverystream/my-delivery-stream",
        "DeliveryStreamStatus": "ACTIVE",
        "DeliveryStreamName": "my-delivery-stream",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
```

```
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. Firehose 전송 스트림으로 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 텍스트 편집기를 사용하여 ~/TrustPolicyForCWL.json 파일에 신뢰 정책을 생성합니다.

   이 정책에는 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 aws:SourceArn 글로벌 조건 컨텍스트 키가 포함되어 있습니다. 자세한 내용은 혼동된 대리자 방지 섹션을 참조하세요.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
         "ArnLike": {
             "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
         }
     }
  }
}
```

9. create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이후 단계에서 필요할 수 있기 때문에 반환된 Role.Arn 값을 적어둡니다.

```
aws iam create-role \
--role-name CWLtoKinesisFirehoseRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
```

```
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Condition": {
                     "StringLike": {
                         "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
                     }
                 }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
    }
}
```

10. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 권한 정책을 파일로 생성합니다(예: ~/PermissionsForCWL.json).

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
      }
    ]
}
```

11. put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

12. Amazon Data Firehose 전송 스트림이 활성 상태가 되고 IAM 역할을 생성하고 나면 CloudWatch Logs 구독 필터를 생성할 수 있습니다. 그 즉시 구독 필터는 실시간으로 선택한 로그 그룹에서 Amazon Data Firehose 전송 스트림으로 로그 데이터를 이동시키기 시작합니다.

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "Destination" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:firehose:region:123456789012:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
```

13. 구독 필터를 설정하고 나면 CloudWatch Logs는 모든 수신 로그 이벤트 중에서 필터 패턴과 일치하는 이벤트를 Amazon Data Firehose 전송 스트림으로 전달합니다. Amazon Data Firehose 전송 스트림에 설정된 시간 버퍼 간격에 따라 Amazon S3에 데이터가 나타나기 시작합니다. 충분한 시간이 지나고 나면 Amazon S3 버킷을 확인하여 데이터를 확인할 수 있습니다.

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
    "Contents": [
        {
            "LastModified": "2015-10-29T00:01:25.000Z",
            "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
            },
            "Size": 593
        },
        {
            "LastModified": "2015-10-29T00:35:41.000Z",
            "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-
stream-2015-10-29-00-35-40-7cc92023-7e66-49bc-9fd4-fc9819cc8ed3",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
```

```
}
```

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2015/10/29/00/
my-delivery-stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
 testfile.gz

{
    "AcceptRanges": "bytes",
    "ContentType": "application/octet-stream",
    "LastModified": "Thu, 29 Oct 2015 00:07:06 GMT",
    "ContentLength": 593,
    "Metadata": {}
}
```

Amazon S3 객체의 데이터는 gzip 형식으로 압축됩니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

```
zcat testfile.gz
```

예제 4: Amazon OpenSearch Service를 사용한 구독 필터

이 예제에서는 정의된 필터와 일치하는 수신 로그 이벤트를 OpenSearch Service 도메인으로 전송하는 CloudWatch Logs 구독을 생성합니다.

OpenSearch Service에 구독 필터를 생성하려면

1. OpenSearch Service 도메인을 생성합니다. 자세한 내용은 Creating OpenSearch Service domains을 참조하세요.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 탐색 창에서 로그 그룹을 선택합니다.
4. 로그 그룹의 이름을 선택합니다.
5. 작업, 구독 필터, Amazon OpenSearch Service 구독 필터 생성을 선택합니다.
6. 이 계정의 클러스터로 스트리밍하려고 하는지 아니면 다른 계정의 클러스터로 스트리밍하려고 하는지 선택합니다.
   - 이 계정을 선택한 경우 1단계에서 생성된 도메인을 선택합니다.
   - 다른 계정을 선택한 경우 해당 도메인의 ARN 및 엔드포인트를 입력합니다.

7. 다른 계정을 선택한 경우 도메인 ARN 및 엔드포인트를 제공합니다.
8. Amazon OpenSearch Service 클러스터에서 로그 그룹 데이터가 전달될 클러스터의 이름을 선택합니다.
9. 로그 형식을 선택합니다.
10. 구독 필터 패턴에 로그 이벤트에서 찾을 용어나 패턴을 입력합니다. 이렇게 하면 관심 있는 데이터만 OpenSearch Service 클러스터로 전송할 수 있습니다. 자세한 내용은 지표 필터에 대한 필터 패턴 구문 단원을 참조하십시오.
11. (선택 사항) 테스트할 로그 데이터 선택에서 로그 스트림을 선택한 다음 패턴 테스트를 선택해서 검색 필터가 예상한 결과를 반환하고 있는지 확인합니다.
12. 스트리밍 시작을 선택합니다.

계정 수준 구독 필터

Important

구독 필터는 무한 재귀 루프를 발생시킬 위험이 있으며, 이를 해결하지 않으면 수집 요금 청구가 크게 증가할 수 있습니다. 이러한 위험을 완화하려면 계정 수준 구독 필터에서 선택 기준을 사용하여 구독 전송 워크플로의 일부인 리소스에서 로그 데이터를 수집하는 로그 그룹을 제외하는 것이 좋습니다. 이 문제 및 제외할 로그 그룹 결정에 대한 자세한 내용은 로그 재귀 방지을 참조하세요.

계정의 로그 그룹 하위 집합을 포함하는 계정 수준 구독 정책을 설정할 수 있습니다. 계정 구독 정책은 Amazon Kinesis Data Streams AWS Lambda또는 Amazon Data Firehose에서 작동할 수 있습니다. 계정 수준 구독 정책을 통해 서비스로 전송되는 로그는 base64로 인코딩되고 gzip 형식으로 압축됩니다. 이 섹션에서는 Amazon Kinesis Data Streams, Lambda 및 Firehose에 대한 계정 수준 구독을 생성하기 위해 따를 수 있는 예제를 제공합니다.

Note

계정의 모든 구독 필터 정책 목록을 확인하려면 `--policy-type` 파라미터 값 `SUBSCRIPTION_FILTER_POLICY`를 사용하여 `describe-account-policies` 명령을 사용합니다. 자세한 내용은 describe-account-policies¶를 참조하세요.

예제

- 예제 1: Amazon Kinesis Data Streams를 사용한 구독 필터
- 예제 2:를 사용한 구독 필터 AWS Lambda
- 예제 3: Amazon Data Firehose에 대한 구독 필터

예제 1: Amazon Kinesis Data Streams를 사용한 구독 필터

계정 수준 구독 정책에 사용할 Amazon Kinesis Data Streams 데이터 스트림을 생성하기 전에 생성될 로그 데이터의 볼륨을 계산합니다. 이 볼륨을 처리하기에 충분한 샤드로 스트림을 생성해야 합니다. 스트림에 샤드가 충분하지 않은 경우 스트림에 스로틀링이 발생합니다. 스트림 볼륨 제한에 대한 자세한 내용은 Amazon Kinesis Data Streams 설명서의 할당량 및 제한을 참조하세요.

> **Warning**
>
> 여러 로그 그룹의 로그 이벤트가 대상에 전달되므로 스로틀링의 위험이 있습니다. 스로틀링이 발생한 전송물은 최대 24시간 동안 재시도됩니다. 24시간이 지나면 실패한 결과물이 삭제됩니다.
> 스로틀링 현상의 위험을 완화하려면 다음 단계를 수행합니다.
>
> - CloudWatch 지표를 사용하여 Amazon Kinesis Data Streams 스트림을 모니터링합니다. 이렇게 하면 스로틀링 현상을 식별하며, 이에 따라 구성을 조정할 수 있습니다. 예를 들어 `DeliveryThrottling` 지표는 데이터를 구독 대상에 전달할 때 CloudWatch Logs에 스로틀링이 발생한 로그 이벤트 수를 추적합니다. 자세한 내용은 CloudWatch 지표를 사용한 모니터링 단원을 참조하십시오.
> - Amazon Kinesis Data Streams에서 스트림에 대한 온디맨드 용량 모드를 사용합니다. 온디맨드 모드는 워크로드가 확장 또는 축소될 때 즉시 워크로드를 수용합니다. 자세한 내용은 온디맨드 모드를 참조하세요.
> - CloudWatch Logs 구독 필터 패턴을 Amazon Kinesis Data Streams의 스트림 용량과 일치하도록 제한합니다. 스트림에 너무 많은 데이터를 전송하는 경우 필터 크기를 줄이거나 필터 기준을 조정해야 할 수 있습니다.

다음 예제에서는 계정 수준 구독 정책을 사용하여 모든 로그 이벤트를 Amazon Kinesis Data Streams의 스트림으로 전달합니다. 필터 패턴은 모든 로그 이벤트를 텍스트와 일치`Test`시키고 Amazon Kinesis Data Streams의 스트림으로 전달합니다.

Amazon Kinesis Data Streams에 대한 계정 수준 구독 정책을 생성하려면

1. 다음 명령을 사용하여 대상 스트림을 생성합니다.

```
$ C:\>  aws kinesis create-stream –stream-name "TestStream" –shard-count 1
```

2. 스트림이 활성화될 때까지 몇 분 정도 기다립니다. describe-stream 명령을 사용하여 StreamDescription.StreamStatus 속성을 확인하면 스트림이 활성인지 여부를 확인할 수 있습니다.

```
aws kinesis describe-stream --stream-name "TestStream"
```

다음은 예제 출력입니다.

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "TestStream",
        "StreamARN": "arn:aws:kinesis:region:123456789012:stream/TestStream",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "EXAMPLE846346337460743176821145 5",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "EXAMPLE688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 스트림에 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 신뢰 정책을 파일로 생성해야 합니다(예: ~/TrustPolicyForCWL-Kinesis.json). 텍스트 편집기를 사용하여 이 정책을 생성합니다.

이 정책에는 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 aws:SourceArn 글로벌 조건 컨텍스트 키가 포함되어 있습니다. 자세한 내용은 혼동된 대리자 방지 섹션을 참조하세요.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
     }
   }
}
```

4. create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이후 단계에서 필요할 수 있기 때문에 반환된 Role.Arn 값을 적어둡니다.

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document
 file://~/TrustPolicyForCWL-Kinesis.json
```

다음은 출력의 예제입니다.

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": { "arn:aws:logs:region:123456789012:*" }
                    }
                }
            }
        },
        "RoleId": "EXAMPLE450GAB4HC5F431",
        "CreateDate": "2023-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
    }
```

```
}
```

5. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 파일에 권한 정책을 생성합니다(예: ~/PermissionsForCWL-Kinesis.json). 텍스트 편집기를 사용하여 이 정책을 생성합니다. IAM 콘솔을 사용하여 정책을 생성하지 마세요.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:123456789012:stream/TestStream"
    }
  ]
}
```

6. 다음 put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy  --role-name CWLtoKinesisRole  --policy-name Permissions-
Policy-For-CWL  --policy-document file://~/PermissionsForCWL-Kinesis.json
```

7. 스트림이 Active 상태에 있고 IAM 역할을 생성하고 나면 CloudWatch Logs 구독 필터 정책을 생성할 수 있습니다. 이 정책은 스트림으로 가는 실시간 로그 데이터 흐름을 즉시 시작합니다. 이 예제에서는 LogGroupToExclude1과 LogGroupToExclude2라는 로그 그룹의 이벤트를 제외하고 ERROR 문자열을 포함하는 모든 로그 이벤트가 스트리밍됩니다.

```
aws logs put-account-policy \
    --policy-name "ExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/
CWLtoKinesisRole", "DestinationArn":"arn:aws:kinesis:region:123456789012:stream/
TestStream", "FilterPattern": "Test", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

8. 구독 필터를 설정하고 나면 CloudWatch Logs는 모든 수신 로그 이벤트 중에서 필터 패턴 및 선택 기준과 일치하는 이벤트를 스트림으로 전달합니다.

   selection-criteria 필드는 선택 사항이지만 구독 필터에서 무한 로그 재귀를 일으킬 수 있는 로그 그룹을 제외하는 데 중요합니다. 이 문제 및 제외할 로그 그룹 결정에 대한 자세한 내용은 로

그 재귀 방지을 참조하세요. 현재 NOT IN은 `selection-criteria`에 지원되는 유일한 연산자입니다.

Amazon Kinesis Data Streams 샤드 반복기를 사용하고 Amazon Kinesis Data Streams `get-records` 명령을 사용하여 일부 Amazon Kinesis Data Streams 레코드를 가져와서 로그 이벤트의 흐름을 확인할 수 있습니다.

```
aws kinesis get-shard-iterator --stream-name TestStream --shard-id
 shardId-000000000000 --shard-iterator-type TRIM_HORIZON
```

```
{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
```

Amazon Kinesis Data Streams가 데이터를 반환하기 전에이 명령을 몇 번 사용해야 할 수 있습니다.

레코드 어레이에서 응답을 확인할 수 있습니다. Amazon Kinesis Data Streams 레코드의 데이터 속성은 base64로 인코딩되고 gzip 형식으로 압축됩니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

```
echo -n "<Content of Data>" | base64 -d | zcat
```

디코딩 및 압축 해제된 base64 데이터는 다음 구조를 가진 JSON으로 포맷됩니다.

```
{
    "messageType": "DATA_MESSAGE",
    "owner": "123456789012",
```

```
    "logGroup": "Example1",
    "logStream": "logStream1",
    "subscriptionFilters": [
        "ExamplePolicy"
    ],
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        }
    ],
    "policyLevel": "ACCOUNT_LEVEL_POLICY"
}
```

데이터 구조에서 키 요소는 다음과 같습니다.

messageType

데이터 메시지는 'DATA_MESSAGE' 유형을 사용합니다. CloudWatch Logs는 주로 대상에 연결할 수 있는지 확인하기 위해 "CONTROL_MESSAGE" 유형의 Amazon Kinesis Data Streams 레코드를 내보낼 수 있습니다.

owner

발신 로그 데이터의 AWS 계정 ID입니다.

logGroup

원본 로그 데이터의 로그 그룹 이름.

logStream

원본 로그 데이터의 로그 스트림 이름.

subscriptionFilters

원본 로그 데이터과 일치한 구독 필터 이름 목록입니다.

logEvents

로그 이벤트 레코드의 배열로 표현되는 실제 로그 데이터. 'ID' 속성은 모든 로그 이벤트의 고유 식별자입니다.

policyLevel

정책이 적용된 수준. 'ACCOUNT_LEVEL_POLICY'는 계정 수준 구독 필터 정책의 `policyLevel`입니다.

예제 2:를 사용한 구독 필터 AWS Lambda

이 예제에서는 로그 데이터를 AWS Lambda 함수로 전송하는 CloudWatch Logs 계정 수준 구독 필터 정책을 생성합니다.

> **Warning**
>
> Lambda 함수를 생성하기 전에 생성할 로그 데이터의 볼륨을 계산합니다. 이 볼륨을 처리할 수 있는 함수를 생성해야 합니다. 함수가 볼륨을 처리할 수 없는 경우 로그 스트림에 스로틀링이 발생합니다. 모든 로그 그룹 또는 계정 로그 그룹의 하위 집합의 로그 이벤트가 대상으로 전달되므로 스로틀링의 위험이 있습니다. AWS Lambda 제한 사항에 대한 자세한 내용은 Lambda 제한 사항을 참조하세요.

Lambda의 계정 수준 구독 필터 정책을 생성하려면

1. AWS Lambda 함수를 생성합니다.

   Lambda 실행 역할이 설정되었는지 확인합니다. 자세한 내용은 AWS Lambda Developer Guide의 Step 2.2: Create an IAM Role (execution role)을 참조하세요.
2. 다음 콘텐츠로 텍스트 편집기를 열고 `helloWorld.js`라는 파일을 생성합니다.

```
var zlib = require('zlib');
```

```
exports.handler = function(input, context) {
    var payload = Buffer.from(input.awslogs.data, 'base64');
    zlib.gunzip(payload, function(e, result) {
        if (e) {
            context.fail(e);
        } else {
            result = JSON.parse(result.toString());
            console.log("Event Data:", JSON.stringify(result, null, 2));
            context.succeed();
        }
    });
};
```

3. helloWorld.js 파일을 압축하고 `helloWorld.zip`라는 이름으로 저장합니다.
4. 역할이 첫 단계에서 설정한 Lambda 실행 역할인 경우에는 다음 명령을 사용합니다.

```
aws lambda create-function \
    --function-name helloworld \
    --zip-file fileb://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs18.x
```

5. CloudWatch Logs에 함수를 실행할 권한을 부여합니다. 다음 명령을 사용하고 자리 표시자 계정을 자신의 계정으로 바꿉니다.

```
aws lambda add-permission \
    --function-name "helloworld" \
    --statement-id "helloworld" \
    --principal "logs.amazonaws.com" \
    --action "lambda:InvokeFunction" \
    --source-arn "arn:aws:logs:region:123456789012:log-group:*" \
    --source-account "123456789012"
```

6. 다음 명령을 사용하고 자리 표시자 계정을 자신의 계정으로 바꾸어 계정 수준 구독 필터 정책을 생성합니다. 이 예제에서는 `LogGroupToExclude1`과 `LogGroupToExclude2`라는 로그 그룹의 이벤트를 제외하고 `ERROR` 문자열을 포함하는 모든 로그 이벤트가 스트리밍됩니다.

```
aws logs put-account-policy \
    --policy-name "ExamplePolicyLambda" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
```

```
    --policy-document
'{"DestinationArn":"arn:aws:lambda:region:123456789012:function:helloWorld",
"FilterPattern": "Test", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
"LogGroupToExclude2"]' \
    --scope "ALL"
```

구독 필터를 설정하고 나면 CloudWatch Logs는 모든 수신 로그 이벤트 중에서 필터 패턴 및 선택 기준과 일치하는 이벤트를 스트림으로 전달합니다.

`selection-criteria` 필드는 선택 사항이지만 구독 필터에서 무한 로그 재귀를 일으킬 수 있는 로그 그룹을 제외하는 데 중요합니다. 이 문제 및 제외할 로그 그룹 결정에 대한 자세한 내용은 로그 재귀 방지을 참조하세요. 현재 NOT IN은 `selection-criteria`에 지원되는 유일한 연산자입니다.

7. (선택 사항) 샘플 로그 이벤트를 사용하여 테스트합니다. 명령 프롬프트에서 다음 명령을 실행하여 구독된 스트림으로 간단한 로그 메시지를 보냅니다.

   Lambda 함수의 출력을 보려면 /aws/lambda/helloworld에 출력이 표시되는 Lambda 함수를 탐색합니다.

   ```
   aws logs put-log-events --log-group-name Example1 --log-stream-name logStream1 --
   log-events "[{\"timestamp\":CURRENT TIMESTAMP MILLIS , \"message\": \"Simple Lambda
    Test\"}]"
   ```

   Lambda의 배열로 응답을 확인할 수 있습니다. Lambda 레코드의 데이터 속성은 base64로 인코딩되고 gzip 형식으로 압축됩니다. Lambda가 수신하는 실제 페이로드는 `{ "awslogs": {"data": "BASE64ENCODED_GZIP_COMPRESSED_DATA"} }` 형식을 따릅니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

   ```
   echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
   ```

   디코딩 및 압축 해제된 base64 데이터는 다음 구조를 가진 JSON으로 포맷됩니다.

   ```
   {
       "messageType": "DATA_MESSAGE",
       "owner": "123456789012",
       "logGroup": "Example1",
       "logStream": "logStream1",
       "subscriptionFilters": [
   ```

```
        "ExamplePolicyLambda"
    ],
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        }
    ],
    "policyLevel": "ACCOUNT_LEVEL_POLICY"
}
```


> **Note**
>
> 계정 수준 구독 필터는 대상 Lambda 함수의 로그 그룹에 적용되지 않습니다. 이는 수집 요금 청구를 증가시킬 수 있는 무한 로그 재귀를 방지하기 위한 것입니다. 이 문제에 대한 자세한 내용은 로그 재귀 방지을 참조하세요.

데이터 구조에서 키 요소는 다음과 같습니다.

messageType

데이터 메시지는 'DATA_MESSAGE' 유형을 사용합니다. CloudWatch Logs는 주로 대상에 연결할 수 있는지 확인하기 위해 "CONTROL_MESSAGE" 유형의 Amazon Kinesis Data Streams 레코드를 내보낼 수 있습니다.

owner

발신 로그 데이터의 AWS 계정 ID입니다.

logGroup

원본 로그 데이터의 로그 그룹 이름.

logStream

원본 로그 데이터의 로그 스트림 이름.

subscriptionFilters

원본 로그 데이터과 일치한 구독 필터 이름 목록입니다.

logEvents

로그 이벤트 레코드의 배열로 표현되는 실제 로그 데이터. 'ID' 속성은 모든 로그 이벤트의 고유 식별자입니다.

policyLevel

정책이 적용된 수준. “ACCOUNT_LEVEL_POLICY”는 계정 수준 구독 필터 정책의 `policyLevel`입니다.

예제 3: Amazon Data Firehose에 대한 구독 필터

이 예제에서는 수신 로그 이벤트 중에서 정의된 필터와 일치하는 이벤트를 Amazon Data Firehose 전송 스트림에 전송하는 CloudWatch Logs 계정 수준 구독 필터 정책을 생성합니다. CloudWatch Logs에서 Amazon Data Firehose로 전송된 데이터는 이미 gzip 6 수준의 압축이 되었기 때문에 Firehose 전송 스트림 내에서 압축을 사용할 필요가 없습니다. 그런 다음 Firehose의 압축 해제 기능을 사용하여 로그를 자동으로 압축 해제할 수 있습니다. 자세한 내용은 CloudWatch Logs를 사용하여 Kinesis Data Firehose에 쓰기를 참조하세요.

> **Warning**
>
> Firehose 스트림을 생성하기 전에 생성될 로그 데이터의 볼륨을 계산합니다. 이 볼륨을 처리할 수 있는 Firehose 스트림을 생성해야 합니다. 스트림이 볼륨을 처리할 수 없는 경우 로그 스트림에 스로틀링이 발생합니다. Firehose 스트림 볼륨 제한 사항에 대한 자세한 내용은 Amazon Data Firehose 데이터 제한 사항을 참조하세요.

Firehose 구독 필터를 생성하려면

1. Amazon Simple Storage Service(Amazon S3) 버킷을 생성합니다. CloudWatch Logs를 위해 특별히 생성한 버킷을 사용하는 것이 좋습니다. 그러나 기존 버킷을 사용하고 싶으면 2단계로 건너뛸 수 있습니다.

   다음 명령을 실행하여 자리 표시자 리전을 사용하고자 하는 리전으로 바꿉니다.

   ```
   aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration
    LocationConstraint=region
   ```

   다음은 예제 출력입니다.

   ```
   {
       "Location": "/amzn-s3-demo-bucket2"
   }
   ```

2. Amazon S3 버킷에 데이터를 입력하는 데 필요한 권한을 Amazon Data Firehose에 부여하는 IAM 역할을 생성합니다.

   자세한 내용은 Amazon Data Firehose 개발자 가이드의 Amazon Data Firehose를 사용하여 액세스 제어를 참조하세요.

   먼저 텍스트 편집기를 사용하여 다음과 같이 ~/TrustPolicyForFirehose.json 파일에 신뢰 정책을 생성합니다.

   ```
   {
     "Statement": {
       "Effect": "Allow",
       "Principal": { "Service": "firehose.amazonaws.com" },
       "Action": "sts:AssumeRole"
       }
   }
   ```

3. create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이후 단계에서 필요하므로 반환된 Role.Arn 값을 적어둡니다.

   ```
   aws iam create-role \
    --role-name FirehosetoS3Role \
    --assume-role-policy-document file://~/TrustPolicyForFirehose.json
   ```

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                }
            }
        },
        "RoleId": "EXAMPLE50GAB4HC5F431",
        "CreateDate": "2023-05-29T13:46:29.431Z",
        "RoleName": "FirehosetoS3Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/FirehosetoS3Role"
    }
}
```

4. Firehose가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 ~/PermissionsForFirehose.json 파일엣 권한 정책을 생성합니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "s3:AbortMultipartUpload",
          "s3:GetBucketLocation",
          "s3:GetObject",
          "s3:ListBucket",
          "s3:ListBucketMultipartUploads",
          "s3:PutObject" ],
      "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket2",
          "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}
```

5. 다음 put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name Permissions-
Policy-For-Firehose --policy-document file://~/PermissionsForFirehose.json
```

6. 다음과 같이 대상 Firehose 전송 시스템을 생성하여 RoleARN 및 BucketARN의 자리 표시자 값을 생성한 역할 및 버킷 ARN으로 바꿉니다.

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

Firehose는 제공된 Amazon S3 객체에서 YYYY/MM/DD/HH UTC 시간 형식의 접두사를 자동으로 사용합니다. 시간 형식 접두사 앞에 추가할 또 다른 접두사를 지정할 수 있습니다. 슬래시(/)로 끝난 접두사는 Amazon S3 버킷에서 폴더로 표시됩니다.

7. 스트림이 활성화될 때까지 몇 분 정도 기다립니다. Firehose describe-delivery-stream 명령을 사용하여 DeliveryStreamDescription.DeliveryStreamStatus 속성을 확인할 수 있습니다. 또한 이후 단계에서 필요할 수 있기 때문에 DeliveryStreamDescription.DeliveryStreamARN 값을 적어둡니다.

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
    "DeliveryStreamDescription": {
        "HasMoreDestinations": false,
        "VersionId": "1",
        "CreateTimestamp": 1446075815.822,
        "DeliveryStreamARN": "arn:aws:firehose:us-
east-1:123456789012:deliverystream/my-delivery-stream",
        "DeliveryStreamStatus": "ACTIVE",
        "DeliveryStreamName": "my-delivery-stream",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
```

```
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. Firehose 전송 스트림으로 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 텍스트 편집기를 사용하여 ~/TrustPolicyForCWL.json 파일에 신뢰 정책을 생성합니다.

   이 정책에는 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 aws:SourceArn 글로벌 조건 컨텍스트 키가 포함되어 있습니다. 자세한 내용은 혼동된 대리자 방지 섹션을 참조하세요.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
         "ArnLike": {
             "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
         }
     }
  }
}
```

9. create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이후 단계에서 필요하므로 반환된 Role.Arn 값을 적어둡니다.

```
aws iam create-role \
--role-name CWLtoKinesisFirehoseRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
```

```
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Condition": {
                     "StringLike": {
                         "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
                     }
                 }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
    }
}
```

10. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 권한 정책을 파일로 생성합니다(예: ~/PermissionsForCWL.json).

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
      }
    ]
}
```

11. put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

12. Amazon Data Firehose 전송 스트림이 활성 상태가 되고 IAM 역할을 생성하고 나면 CloudWatch Logs 계정 수준 구독 필터 정책을 생성할 수 있습니다. 그 즉시 정책은 선택한 로그 그룹에서 Amazon Data Firehose 전송 스트림으로 실시간 로그 데이터 흐름을 시작합니다.

```
aws logs put-account-policy \
```

```
    --policy-name "ExamplePolicyFirehose" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/
CWLtoKinesisFirehoseRole", "DestinationArn":"arn:aws:firehose:us-
east-1:123456789012:deliverystream/delivery-stream-name", "FilterPattern": "Test",
 "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

13. 구독 필터를 설정하고 나면 CloudWatch Logs는 수신 로그 이벤트 중에서 필터 패턴과 일치하는 이벤트를 Amazon Data Firehose 전송 스트림으로 전달합니다.

    `selection-criteria` 필드는 선택 사항이지만 구독 필터에서 무한 로그 재귀를 일으킬 수 있는 로그 그룹을 제외하는 데 중요합니다. 이 문제 및 제외할 로그 그룹 결정에 대한 자세한 내용은 로그 재귀 방지을 참조하세요. 현재 NOT IN은 `selection-criteria`에 지원되는 유일한 연산자입니다.

    Amazon Data Firehose 전송 스트림에 설정된 시간 버퍼 간격에 따라 Amazon S3에 데이터가 나타나기 시작합니다. 충분한 시간이 지나고 나면 Amazon S3 버킷을 확인하여 데이터를 확인할 수 있습니다.

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
    "Contents": [
        {
            "LastModified": "2023-10-29T00:01:25.000Z",
            "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
            },
            "Size": 593
        },
        {
            "LastModified": "2015-10-29T00:35:41.000Z",
            "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
            "StorageClass": "STANDARD",
```

```
            "Key": "firehose/2023/10/29/00/my-delivery-stream-2023-10-29-00-35-40-
EXAMPLE-7e66-49bc-9fd4-fc9819cc8ed3",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "EXAMPLE6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
}
```

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2023/10/29/00/
my-delivery-stream-2023-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
 testfile.gz

{
    "AcceptRanges": "bytes",
    "ContentType": "application/octet-stream",
    "LastModified": "Thu, 29 Oct 2023 00:07:06 GMT",
    "ContentLength": 593,
    "Metadata": {}
}
```

Amazon S3 객체의 데이터는 gzip 형식으로 압축됩니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

```
zcat testfile.gz
```

교차 계정 교차 리전 구독

다른 AWS 계정의 소유자와 협업하고 Amazon Kinesis 또는 Amazon Data Firehose 스트림(이를 교차 계정 데이터 공유라고 함)과 같은 AWS 리소스에 대한 로그 이벤트를 수신할 수 있습니다. 예를 들어이 로그 이벤트 데이터는 중앙 집중식 Amazon Kinesis Data Streams 또는 Firehose 스트림에서 읽어 사용자 지정 처리 및 분석을 수행할 수 있습니다. 사용자 지정 처리는 특히 다수의 계정에서 데이터를 공동 사용 및 분석할 때 유용합니다.

예를 들어, 기업의 정보 보안 그룹은 실시간 침입 탐지나 이상 행동에 대한 데이터를 분석하고자 중앙 처리를 위해 연동된 프로덕션 로그를 수집하여 모든 부서의 계정에 대해 감사를 실시할 수 있습니다.

Amazon Kinesis Data Streams를 사용하여 기존 보안 분석 시스템에 데이터를 연결할 수 있는 정보 보안 그룹에 이러한 계정 전반의 이벤트 데이터 스트림을 수집하여 전송할 수 있습니다.

> **Note**
>
> 로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상이 가리키는 AWS 리소스는 다른 리전에 위치할 수 있습니다. 다음 섹션의 예제에서는 모든 리전 관련 리소스가 미국 동부(버지니아 북부)에서 생성됩니다.

를 구성 AWS Organizations 하고 멤버 계정으로 작업하는 경우 로그 중앙 집중화를 사용하여 소스 계정에서 중앙 모니터링 계정으로 로그 데이터를 수집할 수 있습니다.

중앙 집중식 로그 그룹으로 작업할 때 구독 필터를 생성할 때 다음 시스템 필드 차원을 사용할 수 있습니다.

- `@aws.account` -이 차원은 로그 이벤트가 시작된 AWS 계정 ID를 나타냅니다.
- `@aws.region` -이 차원은 로그 이벤트가 생성된 AWS 리전을 나타냅니다.
- `@source.log` -이 차원은 로그 이벤트가 시작된 소스 로그 그룹을 나타냅니다.

이러한 차원은 로그 데이터의 소스를 식별하는 데 도움이 되므로 중앙 집중식 로그에서 파생된 지표를 보다 세부적으로 필터링하고 분석할 수 있습니다.

주제

- Amazon Kinesis Data Streams를 사용한 교차 계정 교차 리전 로그 데이터 공유
- Firehose를 사용한 교차 계정 교차 리전 로그 데이터 공유
- Amazon Kinesis Data Streams를 사용한 교차 계정 교차 리전 계정 수준 구독
- Firehose를 사용한 교차 계정 교차 리전 계정 수준 구독

Amazon Kinesis Data Streams를 사용한 교차 계정 교차 리전 로그 데이터 공유

교차 계정 구독을 생성할 때 단일 계정 또는 조직을 발신자로 지정할 수 있습니다. 조직을 지정하는 경우 이 절차를 통해 조직의 모든 계정에서 수신자 계정으로 로그를 보낼 수 있습니다.

계정에서 로그 데이터를 공유하려면 로그 데이터 발신자 및 수신자를 설정해야 합니다.

- 로그 데이터 발신자 - 수신자로부터 대상 정보를 얻고 지정된 대상으로 로그 이벤트를 전송할 준비가 되었음을 CloudWatch Logs에 알립니다. 이 섹션의 나머지 부분에서는 로그 데이터 발신자가 가상 AWS 계정 번호 111111111111로 표시됩니다.

  한 조직 내의 여러 계정에서 한 수신자 계정으로 로그를 보내려면 해당 조직의 모든 계정에서 수신자 계정으로 로그를 보낼 수 있는 권한을 부여하는 정책을 만들 수 있습니다. 각 발신자 계정에 대해 별도의 구독 필터를 설정해야 합니다.
- 로그 데이터 수신자 - Amazon Kinesis Data Streams 스트림을 캡슐화하는 대상을 설정하고 CloudWatch Logs에 수신자가 로그 데이터를 수신하고 싶다는 사실을 알립니다. 그런 다음 수신자는 이 대상에 대한 정보를 발신자와 공유합니다. 이 섹션의 나머지 부분에서는 로그 데이터 수신자가 가상 AWS 계정 번호 999999999999로 표시됩니다.

교차 계정 사용자로부터 로그 이벤트 수신을 시작하기 위해 로그 데이터 수신자는 먼저 CloudWatch Logs 대상을 생성합니다. 각 대상은 다음과 같은 키 요소로 이루어져 있습니다.

대상 이름

생성하고자 하는 대상의 이름.

대상 ARN

구독 피드의 대상으로 사용할 리소스의 Amazon AWS 리소스 이름(ARN)입니다.

역할 ARN

선택한 스트림에 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 AWS Identity and Access Management (IAM) 역할입니다.

액세스 정책

대상에 쓰기 권한이 허용된 사용자들에게 적용되는 IAM 정책 문서(JSON 형식, IAM 정책 문법을 사용해 작성)입니다.

> **Note**
>
> 로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상이 가리키는 AWS 리소스는 다른 리전에 위치할 수 있습니다. 다음 섹션의 예제에서는 모든 리전 관련 리소스가 미국 동부(버지니아 북부)에서 생성됩니다.

주제

- 새 교차 계정 구독 설정
- 기존 교차 계정 구독 업데이트

새 교차 계정 구독 설정

이러한 섹션의 단계에 따라 새 교차 계정 로그 구독을 설정합니다.

주제

- 1단계: 대상 생성
- 2단계: (조직을 사용하는 경우에만) IAM 역할 생성
- 3단계: 교차 계정 대상에 대한 IAM 권한 추가/검증
- 4단계: 구독 필터 생성
- 로그 이벤트 흐름 검증
- 런타임 시 대상 멤버십 수정

1단계: 대상 생성

> **Important**
> 이 절차의 모든 단계는 로그 데이터 수신자 계정에서 수행되어야 합니다.

이 예제에서 로그 데이터 수신자 계정의 AWS 계정 ID는 999999999999이고 로그 데이터 발신자 AWS 계정 ID는 111111111111입니다.

이 예제에서는 RecipientStream이라는 Amazon Kinesis Data Streams 스트림과 CloudWatch Logs가 데이터를 쓸 수 있는 역할을 사용하여 대상을 생성합니다.

대상이 생성되면 CloudWatch Logs는 수신자 계정을 대신하여 대상에게 테스트 메시지를 전송합니다. 구독 필터가 나중에 활성화되면 CloudWatch Logs는 소스 계정을 대신하여 대상에게 로그 이벤트를 보냅니다.

대상을 생성하려면

1. 수신자 계정에서 Amazon Kinesis Data Streams에 대상 스트림을 생성합니다. 명령 프롬프트에서 다음과 같이 입력합니다.

```
aws kinesis create-stream --stream-name "RecipientStream" --shard-count 1
```

2. 스트림이 활성 상태가 될 때까지 기다립니다. aws kinesis describe-stream 명령을 사용하여 StreamDescription.StreamStatus 속성을 확인할 수 있습니다. 또한 CloudWatch Logs의 이후 단계에서 필요할 수 있기 때문에 StreamDescription.StreamARN 값을 기록해 둡니다.

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
        }
      }
    ]
  }
}
```

스트림이 활성 상태가 될 때까지 1~2분 정도 기다려야 할 수 있습니다.

3. 스트림에 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 ~/TrustPolicyForCWL.json 파일에서 신뢰 정책을 생성해야 합니다. 텍스트 편집기를 사용하여 이 정책 파일을 생성하고 IAM 콘솔은 사용하지 마세요.

   이 정책은 `sourceAccountId`를 지정하여 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 포함합니다. 첫 번째 직접 호출에서 소스 계정 ID를 아직 모르는 경우 소스 ARN 필드에 대상 ARN을 넣는 것이 좋습니다. 후속 직접 호출에서는 소스 ARN을 첫 번째 직접 호출에서 수집한 실제 소스 ARN으로 설정해야 합니다. 자세한 내용은 혼동된 대리자 방지 단원을 참조하십시오.

```
{
```

```
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.amazonaws.com"
        },
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
        },
        "Action": "sts:AssumeRole"
    }
}
```

4. aws iam create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이 또한 CloudWatch Logs의 이후 단계에서 필요할 수 있기 때문에 반환된 Role.Arn 값을 적어 둡니다.

```
aws iam create-role \
--role-name CWLtoKinesisRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                },
                "Principal": {
                    "Service": "logs.amazonaws.com"
                }
            }
        },
```

```
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
    }
}
```

5. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 권한 정책을 ~/PermissionsForCWL.json 파일로 생성합니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}
```

6. aws iam put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy \
    --role-name CWLtoKinesisRole \
    --policy-name Permissions-Policy-For-CWL \
    --policy-document file://~/PermissionsForCWL.json
```

7. 스트림이 활성 상태이고 IAM 역할을 생성하고 나면 CloudWatch Logs 대상을 생성할 수 있습니다.

   a. 이 단계를 수행해도 액세스 정책이 대상에 연결되는 것은 아니며, 대상 생성을 완료하기 위한 두 단계 중 첫 번째 단계를 완료한 것일 뿐입니다. 페이로드에 반환되는 DestinationArn을 적어둡니다.

```
aws logs put-destination \
    --destination-name "testDestination" \
    --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
    --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
```

```
  "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
  "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
}
```

b. 7a 단계를 완료한 후 로그 데이터 수신자 계정에서 액세스 정책을 대상과 연결합니다. 이 정책은 logs:PutSubscriptionFilter 작업을 지정해야 하며, 발신자 계정에 대상에 액세스할 권한을 부여합니다.

이 정책은 로그를 전송하는 AWS 계정에 권한을 부여합니다. 정책에서 이 계정 하나만 지정할 수 있으며, 또는 발신자 계정이 조직의 구성원인 경우 정책은 해당 조직의 조직 ID를 지정할 수 있습니다. 이렇게 하면 정책 하나만 생성하여 한 조직의 여러 계정이 이 대상 계정으로 로그를 보내도록 할 수 있습니다.

텍스트 편집기를 사용하여 이름이 ~/AccessPolicy.json이고 다음 정책 명령문 중 하나를 포함한 파일을 생성합니다.

이 첫 번째 예제 정책은 ID가 o-1234567890인 조직의 모든 계정이 수신자 계정으로 로그를 보내도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
```

```
}
```

다음 예에서는 로그 데이터 발신자 계정(111111111111)에서만 로그 데이터 수신자 계정으로 로그를 보내도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111111111111"
            },
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
    ]
}
```

c. 이전 단계에서 생성한 정책을 대상에 연결합니다.

```
aws logs put-destination-policy \
    --destination-name "testDestination" \
    --access-policy file://~/AccessPolicy.json
```

이 액세스 정책을 사용하면 ID가 111111111111인 AWS 계정의 사용자가 ARN arn:aws:logs:region::destination:testDestination인 대상에 대해 PutSubscriptionFilter를 호출할 수 있습니다.999999999999 testDestination 다른 사용자가 이 대상에 대해 PutSubscriptionFilter를 직접 호출하려는 시도는 모두 거부됩니다.

액세스 정책에 대한 사용자 권한의 유효성을 검사하려면 IAM 사용 설명서의 정책 검사기 사용을 참조하세요.

작업을 마쳤으면 교차 계정 권한 AWS Organizations 에를 사용하는 경우의 단계를 따릅니다2단계: (조직을 사용하는 경우에만) IAM 역할 생성. Organizations를 사용하는 대신 다른 계정에 직접 권한을 부여하는 경우 해당 단계를 건너뛰고 4단계: 구독 필터 생성 단계로 넘어갑니다.

2단계: (조직을 사용하는 경우에만) IAM 역할 생성

이전 섹션에서 111111111111 계정이 속한 조직에 권한을 부여하는 액세스 정책을 사용하여 대상을 생성한 경우 111111111111 계정에 직접 권한을 부여하는 대신 이 섹션의 단계를 따릅니다. 그렇지 않다면 4단계: 구독 필터 생성 단계로 건너뛰어도 됩니다.

이 섹션의 단계에서는 CloudWatch가 수신자 대상에 대한 구독 필터를 생성할 수 있는 권한이 발신자 계정에 있는지를 가정하고 검증할 수 있는 IAM 역할을 생성합니다.

발신자 계정에서 이 섹션의 단계를 수행합니다. 역할은 발신자 계정에 있어야 하며 구독 필터에서 이 역할의 ARN을 지정합니다. 이 예제에서 발신자 계정은 111111111111입니다.

를 사용하여 교차 계정 로그 구독에 필요한 IAM 역할을 생성하려면 AWS Organizations

1. /TrustPolicyForCWLSubscriptionFilter.json 파일에 다음 트러스트 정책을 생성합니다. 텍스트 편집기를 사용하여 이 정책 파일을 생성하고, IAM 콘솔은 사용하지 마세요.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. 이 정책을 사용하는 IAM 역할을 생성합니다. 명령에 의해 반환되는 Arn 값은 이 절차의 뒷부분에 필요하므로 메모해 둡니다. 이 예제에서 생성하는 역할의 이름으로는 CWLtoSubscriptionFilterRole을 사용합니다.

```
aws iam create-role \
     --role-name CWLtoSubscriptionFilterRole \
     --assume-role-policy-document file://~/
TrustPolicyForCWLSubscriptionFilter.json
```

3. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다.

   a. 먼저 텍스트 편집기를 사용하여 다음 권한 정책을 이름이 ~/PermissionsForCWLSubscriptionFilter.json인 파일로 생성합니다.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
        }
    ]
}
```

b. 다음 명령을 입력하여 방금 생성한 권한 정책을 2단계에서 생성한 역할에 연결합니다.

```
aws iam put-role-policy
    --role-name CWLtoSubscriptionFilterRole
    --policy-name Permissions-Policy-For-CWL-Subscription-filter
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

작업을 마쳤으면 4단계: 구독 필터 생성 단계로 넘어갑니다.

3단계: 교차 계정 대상에 대한 IAM 권한 추가/검증

AWS 교차 계정 정책 평가 로직에 따라 교차 계정 리소스(예: 구독 필터의 대상으로 사용되는 Kinesis 또는 Firehose 스트림)에 액세스하려면 전송 계정에 교차 계정 대상 리소스에 대한 명시적 액세스를 제공하는 자격 증명 기반 정책이 있어야 합니다. 정책 평가 로직에 대한 자세한 내용은 교차 계정 정책 평가 로직을 참조하세요.

구독 필터를 생성하는 데 사용하는 IAM 역할 또는 IAM 사용자에게 자격 증명 기반 정책을 연결할 수 있습니다. 전송 계정에 이 정책이 있어야 합니다. 관리자 역할을 사용하여 구독 필터를 만드는 경우 이 단계를 건너뛰고 4단계: 구독 필터 생성로 넘어갈 수 있습니다.

교차 계정에 필요한 IAM 권한을 추가 또는 검증하려면

1. 다음 명령을 입력하여 AWS 로그 명령을 실행하는 데 사용되는 IAM 역할 또는 IAM 사용자를 확인합니다.

   ```
   aws sts get-caller-identity
   ```

   이 명령은 다음과 비슷한 출력을 반환합니다.

```
{
"UserId": "User ID",
"Account": "sending account id",
"Arn": "arn:aws:sending account id:role/user:RoleName/UserName"
}
```

RoleName 또는 *UserName*으로 나타낸 값을 기록해 둡니다.

2. 전송 계정의 AWS Management Console 에 로그인하고 1단계에서 입력한 명령의 출력에서 반환된 IAM 역할 또는 IAM 사용자가 있는 연결된 정책을 검색합니다.
3. 이 역할 또는 사용자에게 연결된 정책이 교차 계정 대상 리소스에서 `logs:PutSubscriptionFilter`을 직접 호출할 수 있는 명시적 권한을 제공하는지 확인합니다.

   다음 정책은 단일 AWS 계정인 계정 에서만 모든 대상 리소스에 구독 필터를 생성할 수 있는 권한을 제공합니다999999999999.

   JSON

   ```
   {
        "Version":"2012-10-17",
        "Statement": [
           {
               "Sid": "AllowSubscriptionFiltersOnAccountResources",
               "Effect": "Allow",
               "Action": "logs:PutSubscriptionFilter",
               "Resource": [
                   "arn:aws:logs:*:*:log-group:*",
                   "arn:aws:logs:*:123456789012:destination:*"
               ]
           }
       ]
   }
   ```

   다음 정책은 단일 AWS 계정인 계정에서 라는 특정 대상 리소스에 대해서만 구독 필터를 생성할 `sampleDestination` 수 있는 권한을 제공합니다123456789012.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSubscriptionFiltersonAccountResource",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:sampleDestination"
            ]
        }
    ]
}
```

4단계: 구독 필터 생성

대상을 생성하고 나면 로그 데이터 수신자 계정에서 다른 AWS 계정이 로그 이벤트를 동일한 대상으로 전송할 수 있도록 대상 ARN(arn:aws:logs:us-east-1:999999999999:destination:testDestination)을 이들과 공유할 수 있습니다. 그러면 이러한 다른 전송 계정 사용자는 이 대상에 해당되는 로그 그룹에 대한 구독 필터를 생성합니다. 그 즉시 구독 필터는 실시간으로 선택한 로그 그룹에서 지정된 스트림으로 로그 데이터를 이동시키기 시작합니다.

> **Note**
>
> 전체 조직에 구독 필터에 대한 권한을 부여하는 경우 2단계: (조직을 사용하는 경우에만) IAM 역할 생성에서 생성한 IAM 역할의 ARN을 사용해야 합니다.

다음 예제에서는 전송 계정에 구독 필터가 생성됩니다.이 필터는 이벤트가 포함된 AWS CloudTrail 로그 그룹과 연결되어 "루트" AWS 자격 증명에 의해 기록된 모든 활동이 이전에 생성한 대상으로 전달됩니다. 이 대상에서는 'RecipientStream'이라는 스트림을 캡슐화합니다.

다음 섹션의 나머지 단계에서는 AWS CloudTrail 사용 설명서의 CloudWatch Logs로 CloudTrail 이벤트 전송 지침을 따른 후 CloudTrail 이벤트를 포함하는 로그 그룹을 만들었다고 가정합니다. 이 단계에서는 이 로그 그룹의 이름이 `CloudTrail/logs`라고 가정합니다.

다음 명령을 입력할 때는3단계: 교차 계정 대상에 대한 IAM 권한 추가/검증에서 IAM 사용자로 로그인하거나 정책을 추가한 IAM 역할을 사용하여 로그인해야 합니다.

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail/logs" \
    --filter-name "RecipientStream" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:logs:region:999999999999:destination:testDestination"
```

로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상은 다른 리전에 있는 Amazon Kinesis Data Streams 스트림과 같은 AWS 리소스를 가리킬 수 있습니다.

로그 이벤트 흐름 검증

구독 필터를 생성하고 나면 CloudWatch Logs가 들어오는 모든 로그 이벤트 중에서 필터 패턴과 일치하는 이벤트를 "RecipientStream"이라는 대상 스트림 내에서 캡슐화된 스트림으로 전달합니다. 대상 소유자는 aws kinesis get-shard-iterator 명령을 사용하여 Amazon Kinesis Data Streams 샤드를 캡처하고 aws kinesis get-records 명령을 사용하여 일부 Amazon Kinesis Data Streams 레코드를 가져와서 이 문제가 발생하고 있는지 확인할 수 있습니다.

```
aws kinesis get-shard-iterator \
      --stream-name RecipientStream \
      --shard-id shardId-000000000000 \
      --shard-iterator-type TRIM_HORIZON

{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
      --limit 10 \
      --shard-iterator
      "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

> **Note**
>
> Amazon Kinesis Data Streams가 데이터를 반환하기 전에 get-records 명령을 몇 번 다시 실행해야 할 수 있습니다.

Amazon Kinesis Data Streams 레코드 배열이 포함된 응답이 표시됩니다. Amazon Kinesis Data Streams 레코드의 데이터 속성은 gzip 형식으로 압축된 다음 base64로 인코딩됩니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

```
echo -n "<Content of Data>" | base64 -d | zcat
```

디코딩 및 압축 해제된 base64 데이터는 다음 구조를 가진 JSON으로 포맷됩니다.

```
{
    "owner": "111111111111",
    "logGroup": "CloudTrail/logs",
    "logStream": "111111111111_CloudTrail/logs_us-east-1",
    "subscriptionFilters": [
        "RecipientStream"
    ],
    "messageType": "DATA_MESSAGE",
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        }
    ]
```

```
}
```

이 데이터 구조에서의 키 요소는 다음과 같습니다.

owner

발신 로그 데이터의 AWS 계정 ID입니다.

logGroup

원본 로그 데이터의 로그 그룹 이름.

logStream

원본 로그 데이터의 로그 스트림 이름.

subscriptionFilters

원본 로그 데이터과 일치한 구독 필터 이름 목록입니다.

messageType

데이터 메시지는 'DATA_MESSAGE' 유형을 사용합니다. CloudWatch Logs는 주로 대상에 연결할 수 있는지 확인하기 위해 "CONTROL_MESSAGE" 유형의 Amazon Kinesis Data Streams 레코드를 내보낼 수 있습니다.

logEvents

로그 이벤트 레코드의 배열로 표현되는 실제 로그 데이터. ID 속성은 모든 로그 이벤트의 고유 식별자입니다.

런타임 시 대상 멤버십 수정

소유된 대상에서 몇몇 사용자의 멤버십을 추가 또는 제거해야 하는 상황에 직면할 수 있습니다. 새 액세스 정책과 함께 대상에 `put-destination-policy` 명령을 사용할 수 있습니다. 다음 예제에서는 이전에 추가한 계정인 111111111111이 추가적인 로그 데이터 전송을 중단하고 계정 222222222222가 활성화됩니다.

1. 대상 testDestination에 현재 연결된 정책을 가져와서 AccessPolicy를 적어둡니다.

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"

{
```

```
 "Destinations": [
   {
     "DestinationName": "testDestination",
     "RoleArn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
     "DestinationArn":
 "arn:aws:logs:region:999999999999:destination:testDestination",
     "TargetArn": "arn:aws:kinesis:region:999999999999:stream/RecipientStream",
     "AccessPolicy": "{\"Version\": \"2012-10-17\", \"Statement\":
 [{\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"AWS\":
 \"111111111111\"}, \"Action\": \"logs:PutSubscriptionFilter\", \"Resource\":
 \"arn:aws:logs:region:999999999999:destination:testDestination\"}] }"
   }
 ]
}
```

2. 계정 1111111111111이 중단되고 계정 222222222222가 활성화되었음을 반영하도록 이 정책을 업데이트합니다. ~/NewAccessPolicy.json 파일에 이 정책을 저장합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "222222222222"
            },
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
    ]
}
```

3. PutDestinationPolicy를 직접 호출하여 NewAccessPolicy.json 파일에 정의된 정책을 대상에 연결합니다.

```
aws logs put-destination-policy \
--destination-name "testDestination" \
--access-policy file://~/NewAccessPolicy.json
```

이렇게 하면 계정 ID 111111111111에서 로그 이벤트가 비활성화됩니다. 계정 ID 222222222222에서 나온 로그 이벤트는 계정 222222222222 의 소유자가 구독 필터를 생성하고 나면 그 즉시 대상으로 이동하기 시작합니다.

기존 교차 계정 구독 업데이트

현재 대상 계정이 특정 발신자 계정에만 권한을 부여하는 교차 계정 로그 구독이 있고 대상 계정이 조직의 모든 계정에 대한 액세스 권한을 부여하도록 이 구독을 업데이트하려는 경우 이 섹션의 단계를 따릅니다.

주제

- 1단계: 구독 필터 업데이트
- 2단계: 기존 대상 액세스 정책 업데이트

1단계: 구독 필터 업데이트

> **Note**
>
> 이 단계는 AWS 서비스에서 로깅 활성화에 나열된 서비스에서 생성된 로그에 대한 교차 계정 구독에만 필요합니다. 해당 로그 그룹 중 하나에서 생성된 로그로 작업하지 않는 경우 2단계: 기존 대상 액세스 정책 업데이트 섹션으로 건너뛸 수 있습니다.

경우에 따라 대상 계정으로 로그를 보내는 모든 발신자 계정의 구독 필터를 업데이트해야 합니다. 업데이트에는 IAM 역할이 추가됩니다. IAM 역할은 CloudWatch에서 발신자 계정이 수신자 계정에 로그를 보낼 수 있는 권한이 있는지를 가정하고 검증할 수 있습니다.

교차 계정 구독 권한에 대해 조직 ID를 사용하도록 업데이트하려는 모든 발신자 계정에 대해서는 이 섹션의 단계를 따릅니다.

이 섹션의 예제에서 두 개의 111111111111 계정 및 222222222222 계정은 999999999999 계정에 로그를 전송하기 위해 생성된 구독 필터를 이미 가지고 있습니다. 기존 구독 필터 값은 다음과 같습니다.

```
## Existing Subscription Filter parameter values
    \ --log-group-name "my-log-group-name"
    \ --filter-name "RecipientStream"
    \ --filter-pattern "{$.userIdentity.type = Root}"
```

```
    \ --destination-arn "arn:aws:logs:region:999999999999:destination:testDestination"
```

현재 구독 필터 파라미터 값을 찾아야 하는 경우 다음 명령을 입력합니다.

```
aws logs describe-subscription-filters
    \ --log-group-name "my-log-group-name"
```

교차 계정 로그 권한에 대한 조직 ID 사용을 시작하도록 구독 필터 업데이트

1. ~/TrustPolicyForCWL.json 파일에 다음 트러스트 정책을 생성합니다. 텍스트 편집기를 사용하여 이 정책 파일을 생성하고, IAM 콘솔은 사용하지 마세요.

   ```
   {
     "Statement": {
       "Effect": "Allow",
       "Principal": { "Service": "logs.amazonaws.com" },
       "Action": "sts:AssumeRole"
     }
   }
   ```

2. 이 정책을 사용하는 IAM 역할을 생성합니다. 명령에 의해 반환되는 Arn 값의 Arn 값은 이 절차의 뒷부분에 필요하므로 메모해 둡니다. 이 예제에서 생성하는 역할의 이름으로는 CWLtoSubscriptionFilterRole을 사용합니다.

   ```
   aws iam create-role
       \ --role-name CWLtoSubscriptionFilterRole
       \ --assume-role-policy-document file://~/TrustPolicyForCWL.json
   ```

3. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다.

   a. 먼저 텍스트 편집기를 사용하여 다음 권한 정책을 이름이 / PermissionsForCWLSubscriptionFilter.json인 파일로 생성합니다.

      ```
      {
          "Statement": [
              {
                  "Effect": "Allow",
                  "Action": "logs:PutLogEvents",
                  "Resource": "arn:aws:logs:region:111111111111:log-
      group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
              }
      ```

```
    ]
}
```

b. 다음 명령을 입력하여 방금 생성한 권한 정책을 2단계에서 생성한 역할에 연결합니다.

```
aws iam put-role-policy
    --role-name CWLtoSubscriptionFilterRole
    --policy-name Permissions-Policy-For-CWL-Subscription-filter
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

4. 다음 명령을 입력하여 구독 필터를 업데이트합니다.

```
aws logs put-subscription-filter
    \ --log-group-name "my-log-group-name"
    \ --filter-name "RecipientStream"
    \ --filter-pattern "{$.userIdentity.type = Root}"
    \ --destination-arn
 "arn:aws:logs:region:999999999999:destination:testDestination"
    \ --role-arn "arn:aws:iam::111111111111:role/CWLtoSubscriptionFilterRole"
```

2단계: 기존 대상 액세스 정책 업데이트

모든 발신자 계정에서 구독 필터를 업데이트한 후, 수신자 계정에서 대상 액세스 정책을 업데이트할 수 있습니다.

다음 예제에서 수신자 계정은 `999999999999`이며 대상의 이름은 `testDestination`입니다.

업데이트를 통해 ID가 `o-1234567890`인 조직의 모든 계정이 수신자 계정으로 로그를 보낼 수 있습니다. 구독 필터가 생성된 계정만이 실제로 수신자 계정에 로그를 보냅니다.

수신자 계정의 대상 액세스 정책을 업데이트하여 권한에 대한 조직 ID 사용

1. 수신자 계정에서 텍스트 편집기를 사용하여 다음 내용을 포함한 `~/AccessPolicy.json` 파일을 생성합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
}
```

2. 다음 명령을 입력하여 방금 생성한 정책을 기존 대상에 연결합니다. 특정 AWS 계정 ID를 나열하는 액세스 정책 대신 조직 IDs로 액세스 정책을 사용하도록 대상을 업데이트하려면 `force` 파라미터를 포함합니다.

> **Warning**
>
> 에 나열된 AWS 서비스에서 전송한 로그로 작업AWS 서비스에서 로깅 활성화하는 경우이 단계를 수행하기 전에 먼저에 설명된 대로 모든 발신자 계정의 구독 필터를 업데이트해야 합니다1단계: 구독 필터 업데이트.

```
aws logs put-destination-policy
    \ --destination-name "testDestination"
    \ --access-policy file://~/AccessPolicy.json
    \ --force
```

Firehose를 사용한 교차 계정 교차 리전 로그 데이터 공유

계정에서 로그 데이터를 공유하려면 로그 데이터 발신자 및 수신자를 설정해야 합니다.

- 로그 데이터 발신자 - 수신자로부터 대상 정보를 얻고 지정된 대상으로 로그 이벤트를 전송할 준비가 되었음을 CloudWatch Logs에 알립니다. 이 섹션의 나머지 부분에서는 로그 데이터 발신자가 가상 AWS 계정 번호 111111111111로 표시됩니다.

- 로그 데이터 수신자 - Amazon Kinesis Data Streams 스트림을 캡슐화하는 대상을 설정하고 CloudWatch Logs에 수신자가 로그 데이터를 수신하고 싶다는 사실을 알립니다. 그런 다음 수신자는 이 대상에 대한 정보를 발신자와 공유합니다. 이 섹션의 나머지 부분에서는 로그 데이터 수신자가 가상 AWS 계정 번호 222222222222로 표시됩니다.

이 섹션의 예에서는 Amazon S3 스토리지와 함께 Firehose 전송 스트림을 사용합니다. 다른 설정을 사용하여 Firehose 전송 스트림을 설정할 수도 있습니다. 자세한 내용은 Creating a Firehose Delivery Stream을 참조하세요.

> **Note**
>
> 로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상이 가리키는 AWS 리소스는 다른 리전에 위치할 수 있습니다.

> **Note**
>
> 동일한 계정 및 교차 리전 전송 스트림에 대한 Firehose 구독 필터가 지원됩니다.

주제

- 1단계: Firehose 전송 스트림 생성
- 2단계: 대상 생성
- 3단계: 교차 계정 대상에 대한 IAM 권한 추가/검증
- 4단계: 구독 필터 생성
- 로그 이벤트 흐름 검증
- 런타임 시 대상 멤버십 수정

1단계: Firehose 전송 스트림 생성

> **Important**
>
> 다음 단계를 완료하기 전에 Firehose가 Amazon S3 버킷에 액세스할 수 있도록 액세스 정책을 사용해야 합니다. 자세한 내용은 Amazon Data Firehose 개발자 가이드의 액세스 제어를 참조하세요.
>
> 이 섹션(1단계)의 모든 단계는 로그 데이터 수신자 계정에서 수행해야 합니다.

미국 동부(버지니아 북부)가 다음 샘플 명령에 사용됩니다. 이 리전을 배포에 적합한 리전으로 바꿉니다.

대상으로 사용할 Firehose 전송 스트림을 생성하려면

1. Amazon S3 버킷 생성:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-1
```

2. 버킷에 데이터를 입력하는 데 필요한 권한을 Firehose에 부여하는 IAM 역할을 생성합니다.

   a. 먼저 텍스트 편집기를 사용하여 ~/TrustPolicyForFirehose.json 파일에 신뢰 정책을 생성합니다.

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
 "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
 { "StringEquals": { "sts:ExternalId":"222222222222" } } } }
```

   b. IAM 역할을 생성하여 방금 생성한 신뢰 정책 파일을 지정합니다.

```
aws iam create-role \
    --role-name FirehosetoS3Role \
    --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

   c. 이 명령의 출력은 다음과 비슷합니다. 역할 이름과 역할 ARN을 기록해둡니다.

```
{
    "Role": {
        "Path": "/",
        "RoleName": "FirehosetoS3Role",
        "RoleId": "AROAR3BXASEKW7K635M53",
        "Arn": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
        "CreateDate": "2021-02-02T07:53:10+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
```

```
                "Condition": {
                    "StringEquals": {
                        "sts:ExternalId": "222222222222"
                    }
                }
            }
        }
    }
```

3. Firehose가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다.

   a. 먼저 텍스트 편집기를 사용하여 다음 권한 정책을 이름이 ~/PermissionsForFirehose.json인 파일로 생성합니다. 사용 사례에 따라 이 파일에 권한을 더 추가해야 할 수도 있습니다.

```
{
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
    }]
}
```

   b. 다음 명령을 입력하여 방금 생성한 권한 정책을 IAM 역할에 연결합니다.

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name
 Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. 다음 명령을 입력하여 Firehose 전송 스트림을 생성합니다. *my-role-arn* 및 *amzn-s3-demo-bucket2-arn*을 배포에 올바른 값으로 바꿉니다.

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
```

```
  '{"RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket"}'
```

출력은 다음과 비슷하게 보여야 합니다.

```
{
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
}
```

2단계: 대상 생성

> **Important**
>
> 이 절차의 모든 단계는 로그 데이터 수신자 계정에서 수행되어야 합니다.

대상이 생성되면 CloudWatch Logs는 수신자 계정을 대신하여 대상에게 테스트 메시지를 보냅니다. 구독 필터가 나중에 활성화되면 CloudWatch Logs는 소스 계정을 대신하여 대상에게 로그 이벤트를 보냅니다.

대상을 생성하려면

1. 1단계: Firehose 전송 스트림 생성에서 생성한 Firehose 스트림이 활성화될 때까지 기다립니다. 다음 명령을 사용하여 StreamDescription.StreamStatus 속성을 확인할 수 있습니다.

   ```
   aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
   ```

   또한 이후 단계에서 사용해야 하기 때문에 DeliveryStreamDescription.DeliveryStreamARN 값을 기록해 둡니다. 이 명령의 샘플 출력:

   ```
   {
       "DeliveryStreamDescription": {
           "DeliveryStreamName": "my-delivery-stream",
           "DeliveryStreamARN": "arn:aws:firehose:us-
   east-1:222222222222:deliverystream/my-delivery-stream",
           "DeliveryStreamStatus": "ACTIVE",
           "DeliveryStreamEncryptionConfiguration": {
               "Status": "DISABLED"
   ```

```
        },
        "DeliveryStreamType": "DirectPut",
        "VersionId": "1",
        "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
        ],
        "HasMoreDestinations": false
    }
}
```

전송 스트림이 활성 상태가 될 때까지 1~2분 정도 기다려야 할 수 있습니다.

2. 전송 스트림이 활성 상태가 되면 Firehose 스트림으로 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 ~/TrustPolicyForCWL.json 파일에서 신뢰 정책을 생성해야 합니다. 텍스트 편집기를 사용하여 이 정책을 생성합니다. CloudWatch Logs 엔드포인트에 대한 자세한 내용은 Amazon CloudWatch Logs 엔드포인트 및 할당량을 참조하세요.

   이 정책은 `sourceAccountId`를 지정하여 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 포함합니다. 첫 번째 직접 호출에서 소스 계정 ID를 아직 모르는 경우 소스 ARN 필드에 대상 ARN을 넣는 것이 좋습니다. 후속 직접 호출에서는 소스 ARN을 첫 번째 직접 호출에서 수집한 실제 소스 ARN으로 설정해야 합니다. 자세한 내용은 혼동된 대리자 방지 단원을 참조하십시오.

```
{
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.region.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
        }
     }
}
```

3. aws iam create-role 명령을 사용하여 IAM 역할을 생성해 방금 만든 신뢰 정책 파일을 지정합니다.

```
aws iam create-role \
      --role-name CWLtoKinesisFirehoseRole \
      --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

   다음은 샘플 출력입니다. 이후 단계에서 사용해야 하므로 반환된 `Role.Arn` 값을 메모해 둡니다.

```
{
```

```
    "Role": {
        "Path": "/",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "RoleId": "AROAR3BXASEKYJYWF243H",
        "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "CreateDate": "2021-02-02T08:10:43+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.region.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "ArnLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                }
            }
        }
    }
}
```

4. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 권한 정책을 ~/PermissionsForCWL.json 파일로 생성합니다.

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:*"],
        "Resource":["arn:aws:firehose:region:222222222222:*"]
      }
    ]
}
```

5. 다음 명령을 입력하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-name
 Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

6. Firehose 전송 스트림이 활성 상태이고 IAM 역할을 생성하고 나면 CloudWatch Logs 대상을 생성할 수 있습니다.

   a. 이 단계를 수행했다고 액세스 정책이 대상에 연결되는 것은 아니며, 대상 생성을 완료하기 위한 두 단계 중 첫 번째 단계일 뿐입니다. 페이로드에서 반환된 새 대상의 ARN은 이후 단계에서 `destination.arn`으로 사용할 것이므로 메모해 둡니다.

```
aws logs put-destination \

    --destination-name "testFirehoseDestination" \
    --target-arn "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole"

{
    "destination": {
        "destinationName": "testFirehoseDestination",
        "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
        "roleArn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "arn": "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"}
}
```

   b. 이전 단계를 완료한 후 로그 데이터 수신자 계정(222222222222)에서 액세스 정책을 대상과 연결합니다.

      이 정책을 사용하면 로그 데이터 발신자 계정(111111111111)이 로그 데이터 수신자 계정(222222222222)만의 대상에 액세스할 수 있습니다. 텍스트 편집기를 사용하여 이 정책을 ~/AccessPolicy.json 파일에 저장할 수 있습니다.

      JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
```

```
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

c. 이렇게 하면 대상에 대해 쓰기 액세스 권한을 가진 사람을 정의하는 정책이 생성됩니다. 이 정책은 대상 액세스를 위한 logs:PutSubscriptionFilter 작업을 지정하게 됩니다. 교차 계정 사용자는 PutSubscriptionFilter 작업을 사용해 대상에 로그 이벤트를 전송합니다.

```
aws logs put-destination-policy \
    --destination-name "testFirehoseDestination" \
    --access-policy file://~/AccessPolicy.json
```

3단계: 교차 계정 대상에 대한 IAM 권한 추가/검증

AWS 교차 계정 정책 평가 로직에 따라 교차 계정 리소스(예: 구독 필터의 대상으로 사용되는 Kinesis 또는 Firehose 스트림)에 액세스하려면 전송 계정에 교차 계정 대상 리소스에 대한 명시적 액세스를 제공하는 자격 증명 기반 정책이 있어야 합니다. 정책 평가 로직에 대한 자세한 내용은 교차 계정 정책 평가 로직을 참조하세요.

구독 필터를 생성하는 데 사용하는 IAM 역할 또는 IAM 사용자에게 자격 증명 기반 정책을 연결할 수 있습니다. 전송 계정에 이 정책이 있어야 합니다. 관리자 역할을 사용하여 구독 필터를 만드는 경우 이 단계를 건너뛰고 4단계: 구독 필터 생성로 넘어갈 수 있습니다.

교차 계정에 필요한 IAM 권한을 추가 또는 검증하려면

1. 다음 명령을 입력하여 AWS 로그 명령을 실행하는 데 사용되는 IAM 역할 또는 IAM 사용자를 확인합니다.

```
aws sts get-caller-identity
```

이 명령은 다음과 비슷한 출력을 반환합니다.

```
{
"UserId": "User ID",
"Account": "sending account id",
"Arn": "arn:aws:sending account id:role/user:RoleName/UserName"
```

```
}
```

RoleName 또는 *UserName*으로 나타낸 값을 기록해 둡니다.

2. 전송 계정의 AWS Management Console 에 로그인하고 1단계에서 입력한 명령의 출력에서 반환된 IAM 역할 또는 IAM 사용자가 있는 연결된 정책을 검색합니다.
3. 이 역할 또는 사용자에게 연결된 정책이 교차 계정 대상 리소스에서 `logs:PutSubscriptionFilter`을 직접 호출할 수 있는 명시적 권한을 제공하는지 확인합니다.

   다음 정책은 단일 AWS 계정인 계정 에서만 모든 대상 리소스에 구독 필터를 생성할 수 있는 권한을 제공합니다999999999999.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Sid":
    "AllowSubscriptionFiltersOnAnyResourceInOneSpecificAccount",
               "Effect": "Allow",
               "Action": "logs:PutSubscriptionFilter",
               "Resource": [
                   "arn:aws:logs:*:*:log-group:*",
                   "arn:aws:logs:*:123456789012:destination:*"
               ]
           }
       ]
   }
   ```

   다음 정책은 단일 AWS 계정인 계정에서 라는 특정 대상 리소스에 대해서만 구독 필터를 생성할 `sampleDestination` 수 있는 권한을 제공합니다123456789012.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
   ```

```
        {
        "Sid": "AllowSubscriptionFiltersOnSpecificResource",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

4단계: 구독 필터 생성

이 예제에서는 111111111111인 전송 계정으로 전환합니다. 이제 전송 계정에서 구독 필터를 생성합니다. 이 예제에서 필터는 AWS CloudTrail 이벤트가 포함된 로그 그룹과 연결되어 "루트" AWS 자격 증명에 의해 기록된 모든 활동이 이전에 생성한 대상으로 전달됩니다. CloudWatch Logs로 AWS CloudTrail 이벤트를 전송하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudWatch Logs로 CloudTrail CloudTrail 이벤트 전송을 참조하세요.

다음 명령을 입력할 때는3단계: 교차 계정 대상에 대한 IAM 권한 추가/검증에서 IAM 사용자로 로그인하거나 정책을 추가한 IAM 역할을 사용하여 로그인해야 합니다.

```
aws logs put-subscription-filter \
    --log-group-name "aws-cloudtrail-logs-111111111111-300a971e" \
    --filter-name "firehose_test" \
    --filter-pattern "{$.userIdentity.type = AssumedRole}" \
    --destination-arn "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
```

로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상은 다른 리전에 있는 Firehose 스트림과 같은 AWS 리소스를 가리킬 수 있습니다.

로그 이벤트 흐름 검증

구독 필터를 생성하고 나면 CloudWatch Logs는 모든 수신 로그 이벤트 중에서 필터 패턴과 일치하는 이벤트를 Firehose 전송 스트림으로 전달합니다. Firehose 전송 스트림에 설정된 시간 버퍼 간격에 따라 Amazon S3 버킷에 데이터가 나타나기 시작합니다. 충분한 시간이 지나고 나면 Amazon S3 버킷을 확인하여 데이터를 확인할 수 있습니다. 버킷을 확인하려면 다음 명령을 입력합니다.

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

해당 명령은 다음과 비슷하게 출력됩니다.

```
{
    "Contents": [
        {
            "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
            "LastModified": "2021-02-02T09:00:26+00:00",
            "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
            "Size": 198,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "firehose+2test",
                "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
            }
        }
    ]
}
```

다음 명령을 입력하여 버킷에서 특정 객체를 검색할 수 있습니다. 이전 명령에서 찾은 값으로 key의 값을 바꿉니다.

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Amazon S3 객체의 데이터는 gzip 형식으로 압축됩니다. 다음 명령 중 하나를 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

Linux:

```
zcat testfile.gz
```

macOS:

```
zcat <testfile.gz
```

런타임 시 대상 멤버십 수정

소유한 대상에서 로그 발신자를 추가 또는 제거해야 하는 상황에 직면할 수 있습니다. 새 액세스 정책에서는 대상에 대해 PutDestinationPolicy 작업을 사용할 수 있습니다. 다음 예제에서는 이전에 추가한 계정인 111111111111이 추가적인 로그 데이터 전송을 중단하고 계정 333333333333이 활성화됩니다.

1. 대상 testDestination에 현재 연결된 정책을 가져와서 AccessPolicy를 적어둡니다.

```
aws logs describe-destinations \
    --destination-name-prefix "testFirehoseDestination"

{
    "destinations": [
        {
            "destinationName": "testFirehoseDestination",
            "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
            "roleArn": "arn:aws:iam:: 222222222222:role/CWLtoKinesisFirehoseRole",
            "accessPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement
\" : [\n    {\n      \"Sid\" : \"\",\n      \"Effect\" : \"Allow\",\n
 \"Principal\" : {\n        \"AWS\" : \"111111111111 \"\n      },\n      \"Action
\" : \"logs:PutSubscriptionFilter\",\n      \"Resource\" : \"arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination\"\n    }\n  ]\n}\n\n",
            "arn": "arn:aws:logs:us-east-1:
 222222222222:destination:testFirehoseDestination",
            "creationTime": 1612256124430
        }
    ]
}
```

2. 계정 111111111111이 중단되고 계정 333333333333이 활성화되었음을 반영하도록 이 정책을 업데이트합니다. ~/NewAccessPolicy.json 파일에 이 정책을 저장합니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
```

```
      "Principal" : {
        "AWS" : "333333333333 "
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

3. 다음 명령을 사용하여 NewAccessPolicy.json 파일에 정의된 정책을 대상과 연결합니다.

```
aws logs put-destination-policy \
    --destination-name "testFirehoseDestination" \

    --access-policy file://~/NewAccessPolicy.json
```

이렇게 하면 계정 ID 111111111111에서 로그 이벤트가 비활성화됩니다. 계정 ID 333333333333에서 나온 로그 이벤트는 계정 333333333333의 소유자가 구독 필터를 생성하고 나면 그 즉시 대상으로 이동하기 시작합니다.

Amazon Kinesis Data Streams를 사용한 교차 계정 교차 리전 계정 수준 구독

교차 계정 구독을 생성할 때 단일 계정 또는 조직을 발신자로 지정할 수 있습니다. 조직을 지정하는 경우 이 절차를 통해 조직의 모든 계정에서 수신자 계정으로 로그를 보낼 수 있습니다.

계정에서 로그 데이터를 공유하려면 로그 데이터 발신자 및 수신자를 설정해야 합니다.

- 로그 데이터 발신자 - 수신자로부터 대상 정보를 얻고 지정된 대상으로 로그 이벤트를 전송할 준비가 되었음을 CloudWatch Logs에 알립니다. 이 섹션의 나머지 부분에서는 가상 AWS 계정 번호가 111111111111인 로그 데이터 발신자가 표시됩니다.

  한 조직 내의 여러 계정에서 한 수신자 계정으로 로그를 보내려면 해당 조직의 모든 계정에서 수신자 계정으로 로그를 보낼 수 있는 권한을 부여하는 정책을 만들 수 있습니다. 각 발신자 계정에 대해 별도의 구독 필터를 설정해야 합니다.
- 로그 데이터 수신자 - Amazon Kinesis Data Streams 스트림을 캡슐화하는 대상을 설정하고 CloudWatch Logs에 수신자가 로그 데이터를 수신하고 싶다는 사실을 알립니다. 그런 다음 수신자는

이 대상에 대한 정보를 발신자와 공유합니다. 이 섹션의 나머지 부분에서는 로그 데이터 수신자가 가상 AWS 계정 번호 999999999999로 표시됩니다.

교차 계정 사용자로부터 로그 이벤트 수신을 시작하기 위해 로그 데이터 수신자는 먼저 CloudWatch Logs 대상을 생성합니다. 각 대상은 다음과 같은 키 요소로 이루어져 있습니다.

대상 이름

생성하고자 하는 대상의 이름.

대상 ARN

구독 피드의 대상으로 사용할 리소스의 Amazon AWS 리소스 이름(ARN)입니다.

역할 ARN

선택한 스트림에 데이터를 넣는 데 필요한 권한을 CloudWatch Logs에 부여하는 AWS Identity and Access Management (IAM) 역할입니다.

액세스 정책

대상에 쓰기 권한이 허용된 사용자들에게 적용되는 IAM 정책 문서(JSON 형식, IAM 정책 문법을 사용해 작성)입니다.

> **Note**
>
> 로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상이 가리키는 AWS 리소스는 다른 리전에 위치할 수 있습니다. 다음 섹션의 예제에서는 모든 리전 관련 리소스가 미국 동부(버지니아 북부)에서 생성됩니다.

주제

- 새 교차 계정 구독 설정
- 기존 교차 계정 구독 업데이트

새 교차 계정 구독 설정

이러한 섹션의 단계에 따라 새 교차 계정 로그 구독을 설정합니다.

주제

- 1단계: 대상 생성
- 2단계: (조직을 사용하는 경우에만) IAM 역할 생성
- 3단계: 계정 수준 구독 필터 정책 생성
- 로그 이벤트의 흐름 검증
- 런타임 시 대상 멤버십 수정

1단계: 대상 생성

> **Important**
>
> 이 절차의 모든 단계는 로그 데이터 수신자 계정에서 수행되어야 합니다.

이 예제에서 로그 데이터 수신자 계정의 AWS 계정 ID는 999999999999이고 로그 데이터 발신자 AWS 계정 ID는 111111111111입니다.

이 예제에서는 RecipientStream이라는 Amazon Kinesis Data Streams 스트림과 CloudWatch Logs가 데이터를 쓸 수 있는 역할을 사용하여 대상을 생성합니다.

대상이 생성되면 CloudWatch Logs는 수신자 계정을 대신하여 대상에게 테스트 메시지를 전송합니다. 구독 필터가 나중에 활성화되면 CloudWatch Logs는 소스 계정을 대신하여 대상에게 로그 이벤트를 보냅니다.

대상을 생성하려면

1. 수신자 계정에서 Amazon Kinesis Data Streams에 대상 스트림을 생성합니다. 명령 프롬프트에서 다음과 같이 입력합니다.

   ```
   aws kinesis create-stream --stream-name "RecipientStream" --shard-count 1
   ```

2. 스트림이 활성 상태가 될 때까지 기다립니다. aws kinesis describe-stream 명령을 사용하여 StreamDescription.StreamStatus 속성을 확인할 수 있습니다. 또한 CloudWatch Logs의 이후 단계에서 필요할 수 있기 때문에 StreamDescription.StreamARN 값을 기록해 둡니다.

   ```
   aws kinesis describe-stream --stream-name "RecipientStream"
   {
     "StreamDescription": {
       "StreamStatus": "ACTIVE",
       "StreamName": "RecipientStream",
   ```

```
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
        }
      }
    ]
  }
}
```

스트림이 활성 상태가 될 때까지 1~2분 정도 기다려야 할 수 있습니다.

3. 스트림에 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 ~/TrustPolicyForCWL.json 파일에서 신뢰 정책을 생성해야 합니다. 텍스트 편집기를 사용하여 이 정책 파일을 생성하고 IAM 콘솔은 사용하지 마세요.

   이 정책은 sourceAccountId를 지정하여 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 aws:SourceArn 글로벌 조건 컨텍스트 키를 포함합니다. 첫 번째 직접 호출에서 소스 계정 ID를 아직 모르는 경우 소스 ARN 필드에 대상 ARN을 넣는 것이 좋습니다. 후속 직접 호출에서는 소스 ARN을 첫 번째 직접 호출에서 수집한 실제 소스 ARN으로 설정해야 합니다. 자세한 내용은 혼동된 대리자 방지 단원을 참조하십시오.

```
{
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.amazonaws.com"
        },
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
```

```
        },
        "Action": "sts:AssumeRole"
    }
}
```

4. aws iam create-role 명령을 사용하여 신뢰 정책 파일을 지정하는 IAM 역할을 생성합니다. 이 또한 CloudWatch Logs의 이후 단계에서 필요할 수 있기 때문에 반환된 Role.Arn 값을 적어 둡니다.

```
aws iam create-role \
--role-name CWLtoKinesisRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                },
                "Principal": {
                    "Service": "logs.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2023-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
    }
}
```

5. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 권한 정책을 ~/PermissionsForCWL.json 파일로 생성합니다.

```
{
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}
```

6. aws iam put-role-policy 명령을 사용하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy \
    --role-name CWLtoKinesisRole \
    --policy-name Permissions-Policy-For-CWL \
    --policy-document file://~/PermissionsForCWL.json
```

7. 스트림이 활성 상태이고 IAM 역할을 생성하고 나면 CloudWatch Logs 대상을 생성할 수 있습니다.

   a. 이 단계를 수행해도 액세스 정책이 대상에 연결되는 것은 아니며, 대상 생성을 완료하기 위한 두 단계 중 첫 번째 단계를 완료한 것일 뿐입니다. 페이로드에 반환되는 DestinationArn을 적어둡니다.

```
aws logs put-destination \
    --destination-name "testDestination" \
    --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
    --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
  "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
  "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
}
```

   b. 7a 단계를 완료한 후 로그 데이터 수신자 계정에서 액세스 정책을 대상과 연결합니다. 이 정책은 logs:PutSubscriptionFilter 작업을 지정해야 하며, 발신자 계정에 대상에 액세스할 권한을 부여합니다.

      이 정책은 로그를 전송하는 AWS 계정에 권한을 부여합니다. 정책에서 이 계정 하나만 지정할 수 있으며, 또는 발신자 계정이 조직의 구성원인 경우 정책은 해당 조직의 조직 ID를 지정할

수 있습니다. 이렇게 하면 정책 하나만 생성하여 한 조직의 여러 계정이 이 대상 계정으로 로그를 보내도록 할 수 있습니다.

텍스트 편집기를 사용하여 이름이 ~/AccessPolicy.json이고 다음 정책 명령문 중 하나를 포함한 파일을 생성합니다.

이 첫 번째 예제 정책은 ID가 o-1234567890인 조직의 모든 계정이 수신자 계정으로 로그를 보내도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
}
```

다음 예에서는 로그 데이터 발신자 계정(111111111111)에서만 로그 데이터 수신자 계정으로 로그를 보내도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111111111111"
            },
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
    ]
}
```

c. 이전 단계에서 생성한 정책을 대상에 연결합니다.

```
aws logs put-destination-policy \
    --destination-name "testDestination" \
    --access-policy file://~/AccessPolicy.json
```

이 액세스 정책을 사용하면 ID가 111111111111인 AWS 계정의 사용자가 ARN arn:aws:logs:region::destination:testDestination인 대상에 대해 PutSubscriptionFilter를 호출할 수 있습니다.999999999999 testDestination 다른 사용자가 이 대상에 대해 PutSubscriptionFilter를 직접 호출하려는 시도는 모두 거부됩니다.

액세스 정책에 대한 사용자 권한의 유효성을 검사하려면 IAM 사용 설명서의 정책 검사기 사용을 참조하세요.

작업을 마쳤으면 교차 계정 권한 AWS Organizations 에를 사용하는 경우의 단계를 따릅니다2단계: (조직을 사용하는 경우에만) IAM 역할 생성. Organizations를 사용하는 대신 다른 계정에 직접 권한을 부여하는 경우 해당 단계를 건너뛰고 3단계: 계정 수준 구독 필터 정책 생성 단계로 넘어갑니다.

2단계: (조직을 사용하는 경우에만) IAM 역할 생성

이전 섹션에서 111111111111 계정이 속한 조직에 권한을 부여하는 액세스 정책을 사용하여 대상을 생성한 경우 111111111111 계정에 직접 권한을 부여하는 대신 이 섹션의 단계를 따릅니다. 그렇지 않다면 3단계: 계정 수준 구독 필터 정책 생성 단계로 건너뛰어도 됩니다.

이 섹션의 단계에서는 CloudWatch가 수신자 대상에 대한 구독 필터를 생성할 수 있는 권한이 발신자 계정에 있는지를 가정하고 검증할 수 있는 IAM 역할을 생성합니다.

발신자 계정에서 이 섹션의 단계를 수행합니다. 역할은 발신자 계정에 있어야 하며 구독 필터에서 이 역할의 ARN을 지정합니다. 이 예제에서 발신자 계정은 111111111111입니다.

를 사용하여 교차 계정 로그 구독에 필요한 IAM 역할을 생성하려면 AWS Organizations

1. /TrustPolicyForCWLSubscriptionFilter.json 파일에 다음 트러스트 정책을 생성합니다. 텍스트 편집기를 사용하여 이 정책 파일을 생성하고, IAM 콘솔은 사용하지 마세요.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. 이 정책을 사용하는 IAM 역할을 생성합니다. 명령에 의해 반환되는 Arn 값은 이 절차의 뒷부분에 필요하므로 메모해 둡니다. 이 예제에서 생성하는 역할의 이름으로는 CWLtoSubscriptionFilterRole을 사용합니다.

```
aws iam create-role \
     --role-name CWLtoSubscriptionFilterRole \
     --assume-role-policy-document file://~/
TrustPolicyForCWLSubscriptionFilter.json
```

3. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다.

   a. 먼저 텍스트 편집기를 사용하여 다음 권한 정책을 이름이 ~/PermissionsForCWLSubscriptionFilter.json인 파일로 생성합니다.

```
{
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
        }
    ]
}
```

b. 다음 명령을 입력하여 방금 생성한 권한 정책을 2단계에서 생성한 역할에 연결합니다.

```
aws iam put-role-policy
    --role-name CWLtoSubscriptionFilterRole
    --policy-name Permissions-Policy-For-CWL-Subscription-filter
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

작업을 마쳤으면 3단계: 계정 수준 구독 필터 정책 생성 단계로 넘어갑니다.

3단계: 계정 수준 구독 필터 정책 생성

대상을 생성하고 나면 로그 데이터 수신자 계정에서 다른 AWS 계정이 로그 이벤트를 동일한 대상으로 전송할 수 있도록 대상 ARN(arn:aws:logs:us-east-1:999999999999:destination:testDestination)을 이들과 공유할 수 있습니다. 그러면 이러한 다른 전송 계정 사용자는 이 대상에 해당되는 로그 그룹에 대한 구독 필터를 생성합니다. 그 즉시 구독 필터는 실시간으로 선택한 로그 그룹에서 지정된 스트림으로 로그 데이터를 이동시키기 시작합니다.

> **Note**
>
> 전체 조직에 구독 필터에 대한 권한을 부여하는 경우 2단계: (조직을 사용하는 경우에만) IAM 역할 생성에서 생성한 IAM 역할의 ARN을 사용해야 합니다.

다음 예제에서는 전송 계정에 계정 수준 구독 필터 정책이 생성됩니다. 필터는 발신자 계정 111111111111과 연결되어 필터 및 선택 기준과 일치하는 모든 로그 이벤트가 이전에 생성한 대상으로 전달됩니다. 이 대상에서는 'RecipientStream'이라는 스트림을 캡슐화합니다.

`selection-criteria` 필드는 선택 사항이지만 구독 필터에서 무한 로그 재귀를 일으킬 수 있는 로그 그룹을 제외하는 데 중요합니다. 이 문제 및 제외할 로그 그룹 결정에 대한 자세한 내용은 로그 재귀 방지을 참조하세요. 현재 NOT IN은 `selection-criteria`에 지원되는 유일한 연산자입니다.

```
aws logs put-account-policy \
```

```
    --policy-name "CrossAccountStreamsExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
 "FilterPattern": "", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

발신자 계정의 로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상은 다른 리전에 있는 Amazon Kinesis Data Streams 스트림과 같은 AWS 리소스를 가리킬 수 있습니다.

로그 이벤트의 흐름 검증

계정 수준 구독 필터 정책을 생성하고 나면 CloudWatch Logs가 모든 수신 로그 이벤트 중에서 필터 패턴 및 선택 기준과 일치하는 이벤트를 "RecipientStream"이라는 대상 스트림 내에서 캡슐화된 스트림으로 전달합니다. 대상 소유자는 aws kinesis get-shard-iterator 명령을 사용하여 Amazon Kinesis Data Streams 샤드를 캡처하고 aws kinesis get-records 명령을 사용하여 일부 Amazon Kinesis Data Streams 레코드를 가져와서이 문제가 발생하고 있는지 확인할 수 있습니다.

```
aws kinesis get-shard-iterator \
      --stream-name RecipientStream \
      --shard-id shardId-000000000000 \
      --shard-iterator-type TRIM_HORIZON

{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
      --limit 10 \
      --shard-iterator
      "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

> **Note**
>
> Amazon Kinesis Data Streams가 데이터를 반환하기 전에 `get-records` 명령을 몇 번 다시 실행해야 할 수 있습니다.

Amazon Kinesis Data Streams 레코드 배열이 포함된 응답이 표시됩니다. Amazon Kinesis Data Streams 레코드의 데이터 속성은 gzip 형식으로 압축된 다음 base64로 인코딩됩니다. 다음 Unix 명령을 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

```
echo -n "<Content of Data>" | base64 -d | zcat
```

디코딩 및 압축 해제된 base64 데이터는 다음 구조를 가진 JSON으로 포맷됩니다.

```
{
    "owner": "111111111111",
    "logGroup": "CloudTrail/logs",
    "logStream": "111111111111_CloudTrail/logs_us-east-1",
    "subscriptionFilters": [
        "RecipientStream"
    ],
    "messageType": "DATA_MESSAGE",
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        }
```

```
    ]
}
```

데이터 구조에서 키 요소는 다음과 같습니다.

messageType

데이터 메시지는 'DATA_MESSAGE' 유형을 사용합니다. CloudWatch Logs는 주로 대상에 연결할 수 있는지 확인하기 위해 "CONTROL_MESSAGE" 유형의 Amazon Kinesis Data Streams 레코드를 내보낼 수 있습니다.

owner

원본 로그 데이터의 AWS 계정 ID입니다.

logGroup

원본 로그 데이터의 로그 그룹 이름.

logStream

원본 로그 데이터의 로그 스트림 이름.

subscriptionFilters

원본 로그 데이터과 일치한 구독 필터 이름 목록입니다.

logEvents

로그 이벤트 레코드의 배열로 표현되는 실제 로그 데이터. 'ID' 속성은 모든 로그 이벤트의 고유 식별자입니다.

policyLevel

정책이 적용된 수준. 'ACCOUNT_LEVEL_POLICY'는 계정 수준 구독 필터 정책의 `policyLevel`입니다.

런타임 시 대상 멤버십 수정

소유된 대상에서 몇몇 사용자의 멤버십을 추가 또는 제거해야 하는 상황에 직면할 수 있습니다. 새 액세스 정책과 함께 대상에 `put-destination-policy` 명령을 사용할 수 있습니다. 다음 예제에서는 이전에 추가한 계정인 111111111111이 추가적인 로그 데이터 전송을 중단하고 계정 222222222222가 활성화됩니다.

1. 대상 testDestination에 현재 연결된 정책을 가져와서 AccessPolicy를 적어둡니다.

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"

{
 "Destinations": [
   {
     "DestinationName": "testDestination",
     "RoleArn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
     "DestinationArn":
 "arn:aws:logs:region:999999999999:destination:testDestination",
     "TargetArn": "arn:aws:kinesis:region:999999999999:stream/RecipientStream",
     "AccessPolicy": "{\"Version\": \"2012-10-17\", \"Statement\":
 [{\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"AWS\":
 \"111111111111\"}, \"Action\": \"logs:PutSubscriptionFilter\", \"Resource\":
 \"arn:aws:logs:region:999999999999:destination:testDestination\"}] }"
   }
 ]
}
```

2. 계정 111111111111이 중단되고 계정 222222222222가 활성화되었음을 반영하도록 이 정책을 업데이트합니다. ~/NewAccessPolicy.json 파일에 이 정책을 저장합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "222222222222"
            },
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
```

```
    ]
}
```

3. PutDestinationPolicy를 직접 호출하여 NewAccessPolicy.json 파일에 정의된 정책을 대상에 연결합니다.

```
aws logs put-destination-policy \
--destination-name "testDestination" \
--access-policy file://~/NewAccessPolicy.json
```

이렇게 하면 계정 ID 111111111111에서 로그 이벤트가 비활성화됩니다. 계정 ID 222222222222에서 나온 로그 이벤트는 계정 222222222222 의 소유자가 구독 필터를 생성하고 나면 그 즉시 대상으로 이동하기 시작합니다.

기존 교차 계정 구독 업데이트

현재 대상 계정이 특정 발신자 계정에만 권한을 부여하는 교차 계정 로그 구독이 있고 대상 계정이 조직의 모든 계정에 대한 액세스 권한을 부여하도록 이 구독을 업데이트하려는 경우 이 섹션의 단계를 따릅니다.

주제

- 1단계: 구독 필터 업데이트
- 2단계: 기존 대상 액세스 정책 업데이트

1단계: 구독 필터 업데이트

> **Note**
>
> 이 단계는 AWS 서비스에서 로깅 활성화에 나열된 서비스에서 생성된 로그에 대한 교차 계정 구독에만 필요합니다. 해당 로그 그룹 중 하나에서 생성된 로그로 작업하지 않는 경우 2단계: 기존 대상 액세스 정책 업데이트 섹션으로 건너뛸 수 있습니다.

경우에 따라 대상 계정으로 로그를 보내는 모든 발신자 계정의 구독 필터를 업데이트해야 합니다. 업데이트에는 IAM 역할이 추가됩니다. IAM 역할은 CloudWatch에서 발신자 계정이 수신자 계정에 로그를 보낼 수 있는 권한이 있는지를 가정하고 검증할 수 있습니다.

교차 계정 구독 권한에 대해 조직 ID를 사용하도록 업데이트하려는 모든 발신자 계정에 대해서는 이 섹션의 단계를 따릅니다.

이 섹션의 예제에서 두 개의 111111111111 계정 및 222222222222 계정은 999999999999 계정에 로그를 전송하기 위해 생성된 구독 필터를 이미 가지고 있습니다. 기존 구독 필터 값은 다음과 같습니다.

```
## Existing Subscription Filter parameter values
{
    "DestinationArn": "arn:aws:logs:region:999999999999:destination:testDestination",
    "FilterPattern": "{$.userIdentity.type = Root}",
    "Distribution": "Random"
}
```

현재 구독 필터 파라미터 값을 찾아야 하는 경우 다음 명령을 입력합니다.

```
aws logs describe-account-policies \
--policy-type "SUBSCRIPTION_FILTER_POLICY" \
--policy-name "CrossAccountStreamsExamplePolicy"
```

교차 계정 로그 권한에 대한 조직 ID 사용을 시작하도록 구독 필터 업데이트

1. ~/TrustPolicyForCWL.json 파일에 다음 트러스트 정책을 생성합니다. 텍스트 편집기를 사용하여 이 정책 파일을 생성하고, IAM 콘솔은 사용하지 마세요.

   ```
   {
     "Statement": {
       "Effect": "Allow",
       "Principal": { "Service": "logs.amazonaws.com" },
       "Action": "sts:AssumeRole"
     }
   }
   ```

2. 이 정책을 사용하는 IAM 역할을 생성합니다. 명령에 의해 반환되는 Arn 값의 Arn 값은 이 절차의 뒷부분에 필요하므로 메모해 둡니다. 이 예제에서 생성하는 역할의 이름으로는 CWLtoSubscriptionFilterRole을 사용합니다.

   ```
   aws iam create-role
       \ --role-name CWLtoSubscriptionFilterRole
       \ --assume-role-policy-document file://~/TrustPolicyForCWL.json
   ```

3. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다.

   a. 먼저 텍스트 편집기를 사용하여 다음 권한 정책을 이름이 `/PermissionsForCWLSubscriptionFilter.json`인 파일로 생성합니다.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
        }
    ]
}
```

   b. 다음 명령을 입력하여 방금 생성한 권한 정책을 2단계에서 생성한 역할에 연결합니다.

```
aws iam put-role-policy
    --role-name CWLtoSubscriptionFilterRole
    --policy-name Permissions-Policy-For-CWL-Subscription-filter
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

4. 다음 명령을 입력하여 구독 필터 정책을 업데이트합니다.

```
aws logs put-account-policy \
    --policy-name "CrossAccountStreamsExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
 "FilterPattern": "{$.userIdentity.type = Root}", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

2단계: 기존 대상 액세스 정책 업데이트

모든 발신자 계정에서 구독 필터를 업데이트한 후, 수신자 계정에서 대상 액세스 정책을 업데이트할 수 있습니다.

다음 예제에서 수신자 계정은 `999999999999`이며 대상의 이름은 `testDestination`입니다.

업데이트를 통해 ID가 o-1234567890인 조직의 모든 계정이 수신자 계정으로 로그를 보낼 수 있습니다. 구독 필터가 생성된 계정만이 실제로 수신자 계정에 로그를 보냅니다.

수신자 계정의 대상 액세스 정책을 업데이트하여 권한에 대한 조직 ID 사용

1. 수신자 계정에서 텍스트 편집기를 사용하여 다음 내용을 포함한 ~/AccessPolicy.json 파일을 생성합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
}
```

2. 다음 명령을 입력하여 방금 생성한 정책을 기존 대상에 연결합니다. 특정 AWS 계정 ID를 나열하는 액세스 정책 대신 조직 IDs로 액세스 정책을 사용하도록 대상을 업데이트하려면 force 파라미터를 포함합니다.

> **Warning**
>
> 에 나열된 AWS 서비스에서 전송한 로그로 작업AWS 서비스에서 로깅 활성화하는 경우이 단계를 수행하기 전에 먼저에 설명된 대로 모든 발신자 계정의 구독 필터를 업데이트해야 합니다1단계: 구독 필터 업데이트.

```
aws logs put-destination-policy
    \ --destination-name "testDestination"
    \ --access-policy file://~/AccessPolicy.json
    \ --force
```

Firehose를 사용한 교차 계정 교차 리전 계정 수준 구독

계정에서 로그 데이터를 공유하려면 로그 데이터 발신자 및 수신자를 설정해야 합니다.

- 로그 데이터 발신자 - 수신자로부터 대상 정보를 얻고 지정된 대상으로 로그 이벤트를 전송할 준비가 되었음을 CloudWatch Logs에 알립니다. 이 섹션의 나머지 부분에서는 가상 AWS 계정 번호가 111111111111인 로그 데이터 발신자가 표시됩니다.
- 로그 데이터 수신자 - Amazon Kinesis Data Streams 스트림을 캡슐화하는 대상을 설정하고 CloudWatch Logs에 수신자가 로그 데이터를 수신하고 싶다는 사실을 알립니다. 그런 다음 수신자는 이 대상에 대한 정보를 발신자와 공유합니다. 이 섹션의 나머지 부분에서는 로그 데이터 수신자가 가상 AWS 계정 번호 222222222222로 표시됩니다.

이 섹션의 예에서는 Amazon S3 스토리지와 함께 Firehose 전송 스트림을 사용합니다. 다른 설정을 사용하여 Firehose 전송 스트림을 설정할 수도 있습니다. 자세한 내용은 Creating a Firehose Delivery Stream을 참조하세요.

> **Note**
>
> 로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상이 가리키는 AWS 리소스는 다른 리전에 위치할 수 있습니다.

> **Note**
>
> 동일한 계정 및 교차 리전 전송 스트림에 대한 Firehose 구독 필터가 지원됩니다.

주제

- 1단계: Firehose 전송 스트림 생성
- 2단계: 대상 생성
- 3단계: 계정 수준 구독 필터 정책 생성
- 로그 이벤트 흐름 검증
- 런타임 시 대상 멤버십 수정

1단계: Firehose 전송 스트림 생성

> **Important**
>
> 다음 단계를 완료하기 전에 Firehose가 Amazon S3 버킷에 액세스할 수 있도록 액세스 정책을 사용해야 합니다. 자세한 내용은 Amazon Data Firehose 개발자 가이드의 액세스 제어를 참조하세요.
>
> 이 섹션(1단계)의 모든 단계는 로그 데이터 수신자 계정에서 수행해야 합니다.
>
> 미국 동부(버지니아 북부)가 다음 샘플 명령에 사용됩니다. 이 리전을 배포에 적합한 리전으로 바꿉니다.

대상으로 사용할 Firehose 전송 스트림을 생성하려면

1. Amazon S3 버킷 생성:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-1
```

2. 버킷에 데이터를 입력하는 데 필요한 권한을 Firehose에 부여하는 IAM 역할을 생성합니다.

   a. 먼저 텍스트 편집기를 사용하여 ~/TrustPolicyForFirehose.json 파일에 신뢰 정책을 생성합니다.

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
 "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
 { "StringEquals": { "sts:ExternalId":"222222222222" } } } }
```

b. IAM 역할을 생성하여 방금 생성한 신뢰 정책 파일을 지정합니다.

```
aws iam create-role \
    --role-name FirehosetoS3Role \
    --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

c. 이 명령의 출력은 다음과 비슷합니다. 역할 이름과 역할 ARN을 기록해둡니다.

```
{
    "Role": {
        "Path": "/",
        "RoleName": "FirehosetoS3Role",
        "RoleId": "AROAR3BXASEKW7K635M53",
        "Arn": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
        "CreateDate": "2021-02-02T07:53:10+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringEquals": {
                        "sts:ExternalId": "222222222222"
                    }
                }
            }
        }
    }
}
```

3. Firehose가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다.

a. 먼저 텍스트 편집기를 사용하여 다음 권한 정책을 이름이 ~/PermissionsForFirehose.json인 파일로 생성합니다. 사용 사례에 따라 이 파일에 권한을 더 추가해야 할 수도 있습니다.

```
{
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
    }]
}
```

b. 다음 명령을 입력하여 방금 생성한 권한 정책을 IAM 역할에 연결합니다.

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name
 Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. 다음 명령을 입력하여 Firehose 전송 스트림을 생성합니다. *my-role-arn* 및 *amzn-s3-demo-bucket2-arn*을 배포에 올바른 값으로 바꿉니다.

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket"}'
```

출력은 다음과 비슷하게 보여야 합니다.

```
{
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
}
```

2단계: 대상 생성

> **Important**
>
> 이 절차의 모든 단계는 로그 데이터 수신자 계정에서 수행되어야 합니다.

대상이 생성되면 CloudWatch Logs는 수신자 계정을 대신하여 대상에게 테스트 메시지를 보냅니다. 구독 필터가 나중에 활성화되면 CloudWatch Logs는 소스 계정을 대신하여 대상에게 로그 이벤트를 보냅니다.

대상을 생성하려면

1. 1단계: Firehose 전송 스트림 생성에서 생성한 Firehose 스트림이 활성화될 때까지 기다립니다. 다음 명령을 사용하여 StreamDescription.StreamStatus 속성을 확인할 수 있습니다.

   ```
   aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
   ```

   또한 이후 단계에서 사용해야 하기 때문에 DeliveryStreamDescription.DeliveryStreamARN 값을 기록해 둡니다. 이 명령의 샘플 출력:

   ```
   {
       "DeliveryStreamDescription": {
           "DeliveryStreamName": "my-delivery-stream",
           "DeliveryStreamARN": "arn:aws:firehose:us-
   east-1:222222222222:deliverystream/my-delivery-stream",
           "DeliveryStreamStatus": "ACTIVE",
           "DeliveryStreamEncryptionConfiguration": {
               "Status": "DISABLED"
           },
           "DeliveryStreamType": "DirectPut",
           "VersionId": "1",
           "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
           "Destinations": [
               {
                   "DestinationId": "destinationId-000000000001",
                   "S3DestinationDescription": {
                       "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                       "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
                       "BufferingHints": {
                           "SizeInMBs": 5,
   ```

```
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
        ],
        "HasMoreDestinations": false
    }
}
```

전송 스트림이 활성 상태가 될 때까지 1~2분 정도 기다려야 할 수 있습니다.

2. 전송 스트림이 활성 상태가 되면 Firehose 스트림으로 데이터를 입력하는 데 필요한 권한을 CloudWatch Logs에 부여하는 IAM 역할을 생성합니다. 먼저 ~/TrustPolicyForCWL.json 파일에서 신뢰 정책을 생성해야 합니다. 텍스트 편집기를 사용하여 이 정책을 생성합니다. CloudWatch Logs 엔드포인트에 대한 자세한 내용은 Amazon CloudWatch Logs 엔드포인트 및 할당량을 참조하세요.

   이 정책은 `sourceAccountId`를 지정하여 혼동된 대리자 보안 문제를 방지하는 데 도움이 되는 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 포함합니다. 첫 번째 직접 호출에서 소스 계정 ID를 아직 모르는 경우 소스 ARN 필드에 대상 ARN을 넣는 것이 좋습니다. 후속 직접 호출에서는 소스

ARN을 첫 번째 직접 호출에서 수집한 실제 소스 ARN으로 설정해야 합니다. 자세한 내용은 혼동된 대리자 방지 단원을 참조하십시오.

```
{
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
        }
     }
}
```

3. aws iam create-role 명령을 사용하여 IAM 역할을 생성해 방금 만든 신뢰 정책 파일을 지정합니다.

```
aws iam create-role \
      --role-name CWLtoKinesisFirehoseRole \
      --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

다음은 샘플 출력입니다. 이후 단계에서 사용해야 하므로 반환된 Role.Arn 값을 메모해 둡니다.

```
{
    "Role": {
        "Path": "/",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "RoleId": "AROAR3BXASEKYJYWF243H",
        "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "CreateDate": "2023-02-02T08:10:43+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
```

```
                "Action": "sts:AssumeRole",
                "Condition": {
                    "ArnLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                }
            }
        }
    }
}
```

4. CloudWatch Logs가 계정에서 수행할 수 있는 작업을 정의하는 권한 정책을 생성합니다. 먼저 텍스트 편집기를 사용하여 권한 정책을 ~/PermissionsForCWL.json 파일로 생성합니다.

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:*"],
        "Resource":["arn:aws:firehose:region:222222222222:*"]
      }
    ]
}
```

5. 다음 명령을 입력하여 권한 정책을 역할에 연결합니다.

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-name
 Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

6. Firehose 전송 스트림이 활성 상태이고 IAM 역할을 생성하고 나면 CloudWatch Logs 대상을 생성할 수 있습니다.

   a. 이 단계를 수행했다고 액세스 정책이 대상에 연결되는 것은 아니며, 대상 생성을 완료하기 위한 두 단계 중 첫 번째 단계일 뿐입니다. 페이로드에서 반환된 새 대상의 ARN은 이후 단계에서 `destination.arn`으로 사용할 것이므로 메모해 둡니다.

```
aws logs put-destination \

    --destination-name "testFirehoseDestination" \
```

```
    --target-arn "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole"

{
    "destination": {
        "destinationName": "testFirehoseDestination",
        "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
        "roleArn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "arn": "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"}
}
```

b. 이전 단계를 완료한 후 로그 데이터 수신자 계정(222222222222)에서 액세스 정책을 대상과 연결합니다. 이 정책을 사용하면 로그 데이터 발신자 계정(111111111111)이 로그 데이터 수신자 계정(222222222222)만의 대상에 액세스할 수 있습니다. 텍스트 편집기를 사용하여 이 정책을~/AccessPolicy.json 파일에 저장할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

c. 이렇게 하면 대상에 대해 쓰기 액세스 권한을 가진 사람을 정의하는 정책이 생성됩니다. 이 정책은 대상에 액세스하기 위해 logs:PutSubscriptionFilter 및 logs:PutAccountPolicy 작업을 지정해야 합니다. 교차 계정 사용자는

PutSubscriptionFilter 및 PutAccountPolicy 작업을 사용하여 대상에 로그 이벤트를 전송합니다.

```
aws logs put-destination-policy \
    --destination-name "testFirehoseDestination" \
    --access-policy file://~/AccessPolicy.json
```

3단계: 계정 수준 구독 필터 정책 생성

이 예제에서는 111111111111인 전송 계정으로 전환합니다. 이제 전송 계정에서 계정 수준 구독 필터 정책을 생성합니다. 이 예제에서 필터는 두 로그 그룹을 제외한 모든 로그 그룹에서 ERROR 문자열이 포함된 모든 로그 이벤트를 이전에 생성한 대상으로 전송하도록 합니다.

```
aws logs put-account-policy \
    --policy-name "CrossAccountFirehoseExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"DestinationArn":"arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination", "FilterPattern":
 "{$.userIdentity.type = AssumedRole}", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

전송 계정의 로그 그룹과 대상은 동일한 AWS 리전에 있어야 합니다. 그러나 대상은 다른 리전에 있는 Firehose 스트림과 같은 AWS 리소스를 가리킬 수 있습니다.

로그 이벤트 흐름 검증

구독 필터를 생성하고 나면 CloudWatch Logs는 모든 수신 로그 이벤트 중에서 필터 패턴 및 선택 기준과 일치하는 이벤트를 Firehose 전송 스트림으로 전달합니다. Firehose 전송 스트림에 설정된 시간 버퍼 간격에 따라 Amazon S3 버킷에 데이터가 나타나기 시작합니다. 충분한 시간이 지나고 나면 Amazon S3 버킷을 확인하여 데이터를 확인할 수 있습니다. 버킷을 확인하려면 다음 명령을 입력합니다.

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

해당 명령은 다음과 비슷하게 출력됩니다.

```
{
```

```
    "Contents": [
        {
            "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
            "LastModified": "2023-02-02T09:00:26+00:00",
            "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
            "Size": 198,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "firehose+2test",
                "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
            }
        }
    ]
}
```

다음 명령을 입력하여 버킷에서 특정 객체를 검색할 수 있습니다. 이전 명령에서 찾은 값으로 key의 값을 바꿉니다.

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Amazon S3 객체의 데이터는 gzip 형식으로 압축됩니다. 다음 명령 중 하나를 사용하여 명령 줄에서 원시 데이터를 검토할 수 있습니다.

Linux:

```
zcat testfile.gz
```

macOS:

```
zcat <testfile.gz
```

런타임 시 대상 멤버십 수정

소유한 대상에서 로그 발신자를 추가 또는 제거해야 하는 상황에 직면할 수 있습니다. 새 액세스 정책에서는 대상에 대해 PutDestinationPolicy 및 PutAccountPolicy 작업을 사용할 수 있습니다. 다음 예제에서는 이전에 추가한 계정인 111111111111이 추가적인 로그 데이터 전송을 중단하고 계정 333333333333이 활성화됩니다.

1. 대상 testDestination에 현재 연결된 정책을 가져와서 AccessPolicy를 적어둡니다.

```
aws logs describe-destinations \
    --destination-name-prefix "testFirehoseDestination"
```

반환된 데이터는 다음과 같을 수 있습니다.

```
{
    "destinations": [
        {
            "destinationName": "testFirehoseDestination",
            "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
            "roleArn": "arn:aws:iam:: 222222222222:role/CWLtoKinesisFirehoseRole",
            "accessPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement
\" : [\n    {\n      \"Sid\" : \"\",\n      \"Effect\" : \"Allow\",\n
 \"Principal\" : {\n        \"AWS\" : \"111111111111 \"\n      },\n      \"Action
\" : \"logs:PutSubscriptionFilter\",\n      \"Resource\" : \"arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination\"\n    }\n  ]\n}\n\n",
            "arn": "arn:aws:logs:us-east-1:
 222222222222:destination:testFirehoseDestination",
            "creationTime": 1612256124430
        }
    ]
}
```

2. 계정 111111111111이 중단되고 계정 333333333333이 활성화되었음을 반영하도록 이 정책을 업데이트합니다. ~/NewAccessPolicy.json 파일에 이 정책을 저장합니다.

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement" : [
       {
         "Sid" : "",
         "Effect" : "Allow",
         "Principal" : {
           "AWS" : "333333333333 "
         },
         "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
         "Resource" : "arn:aws:logs:us-
   east-1:222222222222:destination:testFirehoseDestination"
   ```

```
    }
  ]
}
```

3. 다음 명령을 사용하여 NewAccessPolicy.json 파일에 정의된 정책을 대상과 연결합니다.

```
aws logs put-destination-policy \
    --destination-name "testFirehoseDestination" \

    --access-policy file://~/NewAccessPolicy.json
```

이렇게 하면 계정 ID 111111111111에서 로그 이벤트가 비활성화됩니다. 계정 ID 333333333333에서 나온 로그 이벤트는 계정 333333333333의 소유자가 구독 필터를 생성하고 나면 그 즉시 대상으로 이동하기 시작합니다.

혼동된 대리자 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 AWS 제공합니다.

리소스 정책에서 aws:SourceArn, aws:SourceAccount, aws:SourceOrgID 및 aws:SourceOrgPaths 글로벌 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스를 제공하는 권한을 제한하는 것이 좋습니다. aws:SourceArn을 사용하면 하나의 리소스만 교차 서비스 액세스 권한과 연결됩니다. aws:SourceAccount를 사용하면 해당 계정의 모든 리소스가 교차 서비스 사용 권한과 연결됩니다. aws:SourceOrgID를 사용하면 조직 내 모든 계정의 모든 리소스가 교차 서비스 사용 권한과 연결될 수 있습니다. aws:SourceOrgPaths를 사용하면 AWS Organizations 경로 내 계정의 모든 리소스가 교차 서비스 사용 권한과 연결됩니다. 경로 사용 및 이해에 대한 자세한 내용은 AWS Organizations 개체 경로 이해를 참조하세요.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드

카드 문자(*)를 포함한 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용합니다. 예: arn:aws:*servicename*:*:*123456789012*:*

만약 aws:SourceArn 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 aws:SourceAccount 및 aws:SourceArn를 모두 사용해야 합니다.

혼동된 대리자 문제로부터 보호하려면 리소스 기반 정책에서 리소스의 조직 ID 또는 조직 경로와 함께 aws:SourceOrgID 또는 aws:SourceOrgPaths 글로벌 조건 컨텍스트 키를 사용합니다. aws:SourceOrgID 또는 aws:SourceOrgPaths 키가 포함된 정책에는 올바른 계정이 자동으로 포함되며 조직에서 계정을 추가, 제거 또는 이동할 때 정책을 수동으로 업데이트할 필요가 없습니다.

CloudWatch Logs에 대한 액세스 권한을 부여하여에서 Amazon Kinesis Data Streams 및 Firehose에 데이터를 쓰1단계: 대상 생성고 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 2단계: 대상 생성 보여주는 문서화된 정책입니다.

로그 재귀 방지

구독 필터에는 방지하지 않을 경우 CloudWatch Logs와 대상 모두에서 수집 요금 청구를 크게 증가시킬 수 있는 로그 재귀를 초래할 위험이 있습니다. 이는 구독 전송 워크플로의 결과로 로그 이벤트를 수신하는 로그 그룹과 구독 필터가 연결될 때 발생할 수 있습니다. 로그 그룹에 수집된 로그는 대상으로 전달되어 로그 그룹이 더 많은 로그를 수집하게 되고, 이 로그는 대상으로 다시 전달되어 재귀 루프를 생성합니다.

예를 들어, 로그 이벤트를 Amazon S3로 전달하는 Firehose가 대상인 구독 필터를 생각해 보세요. 또한 Amazon S3에 전달되는 새 이벤트를 처리하고 일부 로그를 자체 생성하는 Lambda 함수도 있습니다. 구독 필터가 Lambda 함수의 로그 그룹에 적용되는 경우 함수가 생성한 로그 이벤트가 대상의 Firehose와 Amazon S3로 전달되고, 그러면 함수가 다시 간접 호출되어 더 많은 로그가 생성되고 Firehose와 Amazon S3로 전달되어 또 다른 함수 간접 호출을 초래하는 현상이 계속됩니다. 이 현상이 무한 루프로 발생하여 로그 수집, Firehose, Amazon S3에서 예상치 못한 요금 청구 증가로 이어집니다.

CloudWatch Logs에 대해 흐름 로그가 활성화된 VPC에 Lambda 함수가 연결된 경우 VPC의 로그 그룹도 로그 재귀를 일으킬 수 있습니다.

구독 전송 워크플로의 일부인 로그 그룹에는 구독 필터를 적용하지 않는 것이 좋습니다. 계정 수준 구독 필터의 경우 PutAccountPolicy API의 selectionCriteria 파라미터를 사용하여 정책에서 이러한 로그 그룹을 제외하세요.

로그 그룹을 제외할 때는 로그를 생성하고 구독 전송 워크플로의 일부일 수 있는 다음 AWS 서비스를 고려하십시오.

- Amazon EC2와 Fargate
- Lambda
- AWS 단계 함수
- CloudWatch Logs에 대해 활성화된 Amazon VPC 흐름 로그

> **Note**
>
> Lambda 대상의 로그 그룹에서 생성된 로그 이벤트는 계정 수준 구독 필터 정책의 경우 Lambda 함수로 다시 전달되지 않습니다. 이 경우 계정 구독 정책에는 `selectionCriteria`를 사용하여 대상 Lambda 함수의 로그 그룹을 제외할 필요가 없습니다.

지표 필터, 구독 필터, 필터 로그 이벤트 및 Live Tail에 대한 필터 패턴 구문

> **Note**
>
> Amazon CloudWatch 로그 인사이트에서 쿼리 언어를 사용하여 로그 그룹을 쿼리하는 방법에 대한 자세한 내용은 CloudWatch Logs Insights 언어 쿼리 구문을 참조하세요.

CloudWatch Logs를 사용하면 지표 필터를 사용하여 로그 데이터를 실행 가능한 지표로 변환하고, 구독 필터를 사용하여 로그 이벤트를 다른 AWS 서비스로 라우팅하고, 로그 이벤트를 필터링하여 로그 이벤트를 검색하고, Live Tail을 사용하여 로그가 수집될 때 실시간으로 대화식으로 로그를 볼 수 있습니다.

필터 패턴은 지표 필터, 구독 필터, 로그 이벤트 및 Live Tail을 사용하여 로그 이벤트에서 일치하는 용어를 검색하는 구문을 구성합니다. 용어는 단어, 정확한 문구 또는 숫자 값일 수 있습니다. 정규식을 사용하여 독립 실행형 필터 패턴을 생성하거나 JSON 및 공백으로 구분된 필터 패턴에 정규식을 통합할 수 있습니다.

일치시키려는 용어를 사용하여 필터 패턴을 생성합니다. 필터 패턴은 정의한 용어가 포함된 로그 이벤트만 반환합니다. CloudWatch 콘솔에서 필터 패턴을 테스트할 수 있습니다.

주제

- 지원되는 정규식 구문
- 필터 패턴을 사용하여 정규식과 일치하는 용어 검색
- 필터 패턴을 사용하여 비정형 로그 이벤트에서 일치하는 용어 검색
- 필터 패턴을 사용하여 JSON 로그 이벤트에서 일치하는 용어 검색
- 필터 패턴을 사용하여 공백으로 구분된 로그 이벤트에서 일치하는 용어 검색

지원되는 정규식 구문

지원되는 정규식 구문

정규식을 사용하여 로그 데이터를 검색하고 필터링하는 경우 표현식을 %로 묶어야 합니다.

정규식을 포함하는 필터 패턴에는 다음만 포함할 수 있습니다.

- 영숫자 - 영숫자는 문자(A~Z 또는 a~z) 또는 숫자(0~9)에 해당하는 문자입니다.
- 지원되는 기호 문자 – 여기에는 ':', '_', '#', '=', '@','/', ';', ',' 및 '-'이 포함됩니다. 예를 들어, `%something!%`은 '!'이 지원되지 않으므로 거부됩니다.
- 지원되는 연산자 - 여기에는 '^', '$', '?', '[', ']', '{', '}', '|', '\', '*', '+' 및 '.'가 포함됩니다.

(및) 연산자는 지원되지 않습니다. 하위 패턴을 정의할 때 괄호를 사용할 수 없습니다.

멀티바이트 문자는 지원되지 않습니다.

> **Note**
>
> 할당량
> 지표 필터 또는 구독 필터를 생성하는 경우 각 로그 그룹에 대한 정규식이 포함된 최대 5개의 필터 패턴이 있습니다.
> 지표 필터 및 구독 필터에 대해 구분된 필터 패턴 또는 JSON 필터 패턴을 생성하거나 로그 이벤트 또는 Live Tail을 필터링하는 경우 각 필터 패턴에 대해 2개의 정규식으로 제한됩니다.

지원되는 연산자 사용

- ^: 문자열의 시작 위치로 일치 항목을 고정합니다. 예를 들어, `%^[hc]at%`는 문자열의 시작 위치만 일치시켜 'hat' 및 'cat'과 일치합니다.
- $: 문자열의 끝 위치로 일치 항목을 고정합니다. 예를 들어, `%[hc]at$%`는 문자열의 끝 위치만 일치시켜 'hat' 및 'cat'과 일치합니다.
- ?: 앞에 오는 용어가 0회 또는 1회 나타나는 경우와 일치합니다. 예를 들어, `%colou?r%`는 'color' 및 'colour' 모두와 일치할 수 있습니다.
- []: 문자 클래스를 정의합니다. 대괄호 안에 포함된 문자 목록 또는 문자 범위와 일치합니다. 예를 들어, `%[abc]%`는 'a', 'b' 또는 'c'와 일치하고, `%[a-z]%`는 'a'에서 'z'까지의 모든 소문자와 일치하며, `%[abcx-z]%`는 'a', 'b', 'c', 'x', 'y' 또는 'z'와 일치합니다.
- {m, n}: 앞에 오는 용어와 m~n번만큼 일치합니다. 예를 들어, `%a{3,5}%` 는 'aaa', 'aaaa' 및 'aaaaa'와만 일치합니다.

> **Note**
>
> 최솟값 또는 최댓값을 정의하지 않으려는 경우 m 또는 n을 생략할 수 있습니다.

- |: 부울 'Or' 연산자로, 세로 막대의 양쪽에 있는 용어 중 하나와 일치합니다. 예제:
  - %gray|grey%는 'gray' 또는 'grey'와 일치할 수 있습니다.
  - %^starting|^initializing|^shutting down%은 'starting ...' 또는 'initializing ...' 또는 'shutting down'과 일치할 수 있지만 'skipping initializing ...'과는 일치하지 않습니다.
  - %abcc|ab[^c]$%는 "abcc ..." 및 "aba"와 일치할 수 있지만 "aac ..."와는 일치하지 않습니다.
- \: 이스케이프 문자로, 이 문자를 사용하면 연산자의 특수한 의미 대신, 문자 그대로의 의미를 사용할 수 있습니다. 예를 들어, %\[.\]%는 대괄호가 이스케이프 처리되었으므로 '[a]', '[b]', '[7]', '[@]', '[]]' 및 '[ ]'와 같이 '[' 및 ']'로 묶인 모든 단일 문자와 일치합니다.

  > **Note**
  >
  > %10\.10\.0\.1%는 IP 주소 10.10.0.1과 일치하는 정규식을 생성하는 올바른 방법입니다.

- *: 앞에 오는 용어의 0개 이상의 인스턴스와 일치합니다. 예를 들어, %ab*c% 는 'ac', 'abc' 및 'abbbc'와 일치할 수 있으며, %ab[0-9]*%는 'ab', 'ab0' 및 'ab129'와 일치할 수 있습니다.
- +: 앞에 오는 용어의 1개 이상의 인스턴스와 일치합니다. 예를 들어, %ab+c%는 'abc', 'abbc' 및 'abbbc'와 일치할 수 있지만, 'ac'와는 일치하지 않습니다.
- .: 모든 문자와 일치합니다. 예를 들어, %.at%는 'hat', 'cat', 'bat', '4at', '#at' 및 ' at'(공백으로 시작)을 포함하여 'at'으로 끝나는 3자리 문자열과 일치합니다.

  > **Note**
  >
  > IP 주소와 일치하는 정규식을 생성하는 경우 . 연산자를 이스케이프 처리하지 않아야 합니다. 예를 들어, %10.10.0.1%는 '10010,051'과 일치할 수 있으며, 이는 식의 의도된 실제 용도가 아닐 수 있습니다.

- \d, \D: 숫자 및 숫자 외 문자와 일치합니다. 예를 들어, %\d%는 %[0-9]%와 같고, %\D%는 %[^0-9]%와 같습니다.

  > **Note**
  >
  > 대문자인 연산자는 소문자인 연산자와 의미가 반대입니다.

- \s, \S: 공백 문자 및 공백 외 문자와 일치합니다.

> **Note**
>
> 대문자인 연산자는 소문자인 연산자와 의미가 반대입니다. 공백 문자에는 탭(\t), 공백( ), 줄 바꿈(\n) 문자가 포함됩니다.

- \w, \W: 영숫자 및 영숫자 외 문자와 일치합니다. 예를 들어, %\w%는 %[a-zA-Z_0-9]%와 같고, %\W%는 %[^a-zA-Z_0-9]%와 같습니다.

> **Note**
>
> 대문자인 연산자는 소문자인 연산자와 의미가 반대입니다.

- \xhh: 2자리 16진수 문자의 ASCII 매핑을 일치합니다. \x는 다음 문자가 ASCII의 16진수 값임을 나타내는 이스케이프 시퀀스입니다. hh는 ASCII 테이블의 문자를 가리키는 2자리 16진수(0~9 및 A~F)를 지정합니다.

> **Note**
>
> \xhh를 사용하여 필터 패턴에서 지원하지 않는 기호 문자를 일치시킬 수 있습니다. 예를 들어, %\x3A%는 :과 일치하며, %\x28%는 (와 일치합니다.

필터 패턴을 사용하여 정규식과 일치하는 용어 검색

정규식을 사용하여 일치하는 용어 검색

%(정규식 패턴 앞뒤의 백분율 기호)로 묶인 정규식 패턴을 사용하여 로그 이벤트에서 일치하는 용어를 검색할 수 있습니다. 다음 코드 조각은 AUTHORIZED 키워드로 구성된 모든 로그 이벤트를 반환하는 필터 패턴의 예시를 보여줍니다.

지원되는 정규식 목록은 지원되는 정규식을 참조하세요.

```
%AUTHORIZED%
```

이 필터 패턴은 다음과 같은 로그 이벤트 메시지를 반환합니다.

- [ERROR 401] UNAUTHORIZED REQUEST
- [SUCCESS 200] AUTHORIZED REQUEST

필터 패턴을 사용하여 비정형 로그 이벤트에서 일치하는 용어 검색

비정형 로그 이벤트에서 일치하는 용어 검색

다음 예제에는 필터 패턴을 사용하여 비정형 로그 이벤트에서 일치하는 용어를 검색하는 방법을 보여 주는 코드 조각이 포함되어 있습니다.

> **Note**
>
> 필터 패턴은 대/소문자를 구분합니다. 영숫자가 아닌 문자가 포함된 정확한 문구 및 용어를 큰 따옴표("")로 묶습니다.

Example: Match a single term

다음 코드 조각은 메시지에 ERROR라는 단어가 포함된 모든 로그 이벤트를 반환하는 단일 용어 필터 패턴의 예를 보여 줍니다.

```
ERROR
```

이 필터 패턴은 다음과 같이 로그 이벤트 메시지와 일치합니다.

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match multiple terms

다음 코드 조각은 메시지에 ERROR 및 ARGUMENTS라는 단어가 포함된 모든 로그 이벤트를 반환하는 다중 용어 필터 패턴의 예를 보여 줍니다.

```
ERROR ARGUMENTS
```

필터는 다음과 같은 로그 이벤트 메시지를 반환합니다.

- `[ERROR 419] MISSING ARGUMENTS`
- `[ERROR 420] INVALID ARGUMENTS`

이 필터 패턴은 필터 패턴에 지정된 두 용어를 모두 포함하지 않으므로 다음의 로그 이벤트 메시지를 반환하지 않습니다.

- `[ERROR 400] BAD REQUEST`
- `[ERROR 401] UNAUTHORIZED REQUEST`

Example: Match optional terms

패턴 일치를 사용하여 선택적 용어가 포함된 로그 이벤트를 반환하는 필터 패턴을 생성할 수 있습니다. 일치시키려는 용어 앞에 물음표('?')를 입력합니다. 다음 코드 조각은 메시지에 ERROR라는 단어 또는 ARGUMENTS라는 단어가 포함된 모든 로그 이벤트를 반환하는 필터 패턴의 예를 보여줍니다.

```
?ERROR ?ARGUMENTS
```

이 필터 패턴은 다음과 같이 로그 이벤트 메시지와 일치합니다.

- `[ERROR 400] BAD REQUEST`
- `[ERROR 401] UNAUTHORIZED REQUEST`
- `[ERROR 419] MISSING ARGUMENTS`
- `[ERROR 420] INVALID ARGUMENTS`

> **Note**
>
> 물음표('?')를 포함 및 제외 용어와 같은 다른 필터 패턴과 결합할 수 없습니다. '?'를 다른 필터 패턴과 결합하면 모든 물음표 용어는 무시됩니다.

예를 들어, 다음 필터 패턴은 REQUEST라는 단어가 포함된 모든 이벤트를 검색하지만 물음표('?') 필터 용어는 무시되고 효과가 없습니다.

```
?ERROR ?ARGUMENTS REQUEST
```

로그 이벤트 일치

- [INFO] REQUEST FAILED
- [WARN] UNAUTHORIZED REQUEST
- [ERROR] 400 BAD REQUEST

Example: Match exact phrases

다음 코드 조각은 메시지에 INTERNAL SERVER ERROR라는 정확한 구가 포함된 모든 로그 이벤트를 반환하는 필터 패턴의 예를 보여 줍니다.

```
"INTERNAL SERVER ERROR"
```

이 필터는 다음과 같은 로그 이벤트 메시지를 반환합니다.

- [ERROR 500] INTERNAL SERVER ERROR

Example: Include and exclude terms

메시지에 일부 용어가 포함되고 다른 용어가 제외된 로그 이벤트를 반환하는 필터 패턴을 만들 수 있습니다. 제외하려는 용어 앞에 빼기 기호('-')를 입력합니다. 다음 코드 조각은 메시지에 ERROR라는 용어를 포함하고 ARGUMENTS라는 용어를 제외한 로그 이벤트를 반환하는 필터 패턴의 예를 보여 줍니다.

```
ERROR -ARGUMENTS
```

이 필터 패턴은 다음과 같은 로그 이벤트 메시지를 반환합니다.

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST

이 필터 패턴은 단어 ARGUMENTS를 포함하므로 다음 로그 이벤트 메시지를 반환하지 않습니다.

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match everything

로그 이벤트의 모든 항목을 큰따옴표로 일치시킬 수 있습니다. 다음 코드 조각은 모든 로그 이벤트를 반환하는 필터 패턴의 예를 보여 줍니다.

```
" "
```

필터 패턴을 사용하여 JSON 로그 이벤트에서 일치하는 용어 검색

JSON 로그 이벤트에 대한 필터 패턴 작성

다음 예제에서는 문자열 및 숫자 값을 포함하는 JSON 용어와 일치하는 필터 패턴에 대한 구문을 작성하는 방법을 설명합니다.

Writing filter patterns that match strings

JSON 로그 이벤트에서 일치하는 문자열을 검색하도록 지표 필터를 생성할 수 있습니다. 다음 코드 조각은 문자열 기반 필터 패턴 구문 예제를 보여줍니다.

```
{ PropertySelector EqualityOperator String }
```

필터 패턴은 중괄호('{}')로 묶습니다. 문자열 기반 필터 패턴은 다음 부분을 포함해야 합니다.

- 속성 선택기

뒤에 마침표가 있는 달러 기호('$.')로 속성 선택기를 설정합니다. 속성 선택기는 하이픈('-') 및 밑줄('_') 문자도 지원하는 영숫자 문자열입니다. 문자열은 과학적 표기법을 지원하지 않습니다. 속성 선택기는 JSON 로그 이벤트의 값 노드를 가리킵니다. 값 노드는 문자열이나 숫자일 수 있습니다. 속성 선택기 뒤에 배열을 배치합니다. 배열의 요소는 0부터 시작하는 번호 매기기 체계를 따릅니다. 즉, 배열의 첫 번째 요소는 요소 0이고, 두 번째 요소는 요소 1이 되는 식입니다. 요소를 대괄호('[]')로 묶습니다. 속성 선택기가 배열이나 객체를 가리키는 경우 필터 패턴은 로그 형식과 일치하지 않습니다. JSON 속성에 마침표("."")가 포함된 경우 대괄호 표기법을 사용하여 해당 속성을 선택할 수 있습니다.

> **Note**
>
> 와일드카드 선택기
> JSON 와일드카드를 사용하여 배열 요소 또는 JSON 객체 필드를 선택할 수 있습니다.
> 할당량
> 속성 선택기에서는 와일드카드 선택기를 최대 1개만 사용할 수 있습니다.

- 같음 연산자

  같음('=') 또는 같지 않음('!=') 기호 중 하나를 사용하여 같음 연산자를 설정합니다. 같음 연산자는 부울 값(true 또는 false)을 반환합니다.

- 문자열

  문자열을 큰따옴표("")로 묶을 수 있습니다. 영숫자 이외의 형식과 밑줄 기호가 포함된 문자열은 큰따옴표로 묶어야 합니다. 별표('*')를 와일드카드로 사용하여 텍스트와 일치시킵니다.

> **Note**
>
> JSON 로그 이벤트에서 일치하는 용어를 검색하기 위해 필터 패턴을 생성하는 경우 모든 조건부 정규식을 사용할 수 있습니다. 지원되는 정규식 목록은 지원되는 정규식을 참조하세요.

다음 코드 조각에는 문자열과 JSON 용어를 일치시키도록 필터 패턴의 서식을 지정하는 방법을 보여주는 필터 패턴 예제가 포함되어 있습니다.

```
{ $.eventType = "UpdateTrail" }
```

Writing filter patterns that match numeric values

JSON 로그 이벤트에서 일치하는 숫자 값을 검색하도록 필터 패턴을 생성할 수 있습니다. 다음 코드 조각은 숫자 값과 일치하는 필터 패턴 구문 예제를 보여줍니다.

```
{ PropertySelector NumericOperator Number }
```

필터 패턴은 중괄호('{}')로 묶습니다. 숫자 값과 일치하는 필터 패턴은 다음 부분을 포함해야 합니다.

- 속성 선택기

  뒤에 마침표가 있는 달러 기호('$.')로 속성 선택기를 설정합니다. 속성 선택기는 하이픈('-') 및 밑줄('_') 문자도 지원하는 영숫자 문자열입니다. 문자열은 과학적 표기법을 지원하지 않습니다. 속성 선택기는 JSON 로그 이벤트의 값 노드를 가리킵니다. 값 노드는 문자열이나 숫자일 수 있습니다. 속성 선택기 뒤에 배열을 배치합니다. 배열의 요소는 0부터 시작하는 번호 매기기 체계를 따릅니다. 즉, 배열의 첫 번째 요소는 요소 0이고, 두 번째 요소는 요소 1이 되는 식입니다. 요소를 대괄호('[]')로 묶습니다. 속성 선택기가 배열이나 객체를 가리키는 경우 필터 패턴은 로그 형식과 일치하지 않습니다. JSON 속성에 마침표('.')가 포함된 경우 대괄호 표기법을 사용하여 해당 속성을 선택할 수 있습니다.

  > **Note**
  >
  > 와일드카드 선택기
  > JSON 와일드카드를 사용하여 배열 요소 또는 JSON 객체 필드를 선택할 수 있습니다.
  > 할당량
  > 속성 선택기에서는 와일드카드 선택기를 최대 1개만 사용할 수 있습니다.

- 숫자 연산자

  다음 기호 중 하나를 사용하여 숫자 연산자를 설정합니다. 보다 큼('>'), 보다 작음('<'), 같음('='), 같지 않음('!='), 이상('>=') 또는 이하('<=') 등.

- 숫자

더하기('+') 또는 빼기('-') 기호를 포함하는 정수를 사용하고 과학적 표기법을 따를 수 있습니다. 별표('*')를 와일드카드로 사용하여 숫자와 일치시킵니다.

다음 코드 조각에는 숫자 값과 JSON 용어를 일치시키도록 필터 패턴의 서식을 지정하는 방법을 보여주는 예제가 포함되어 있습니다.

```
// Filter pattern with greater than symbol
{ $.bandwidth > 75 }
// Filter pattern with less than symbol
{ $.latency < 50 }
// Filter pattern with greater than or equal to symbol
{ $.refreshRate >= 60 }
// Filter pattern with less than or equal to symbol
{ $.responseTime <= 5 }
// Filter pattern with equal sign
{ $.errorCode = 400}
// Filter pattern with not equal sign
{ $.errorCode != 500 }
// Filter pattern with scientific notation and plus symbol
{ $.number[0] = 1e+3 }
// Filter pattern with scientific notation and minus symbol
{ $.number[0] != 1e-3 }
```

간단한 표현식을 사용하여 JSON 로그 이벤트에서 일치하는 용어 검색

다음 예제에는 필터 패턴을 통해 JSON 로그 이벤트에서 일치하는 용어를 검색하는 방법을 보여주는 코드 조각이 포함되어 있습니다.

> **Note**
>
> JSON 로그 이벤트 예제에서 필터 패턴 예제를 테스트하는 경우 JSON 로그 예제를 한 줄에 입력해야 합니다.

JSON 로그 이벤트

```
{
      "eventType": "UpdateTrail",
      "sourceIPAddress": "111.111.111.111",
```

```
      "arrayKey": [
            "value",
            "another value"
      ],
      "objectList": [
           {
             "name": "a",
             "id": 1
           },
           {
             "name": "b",
             "id": 2
           }
      ],
      "SomeObject": null,
      "cluster.name": "c"
}
```

Example: Filter pattern that matches string values

이 필터 패턴은 "eventType" 속성에서 "UpdateTrail" 문자열과 일치합니다.

```
{ $.eventType = "UpdateTrail" }
```

Example: Filter pattern that matches string values (IP address)

이 필터 패턴은 와일드카드를 포함하며 "123.123." 접두사가 있는 숫자는 포함하지 않으므로 "sourceIPAddress" 속성과 일치합니다.

```
{ $.sourceIPAddress != 123.123.* }
```

Example: Filter pattern that matches a specific array element with a string value

이 필터 패턴은 "arrayKey" 배열에서 "value" 요소와 일치합니다.

```
{ $.arrayKey[0] = "value" }
```

Example: Filter pattern that matches a string using regex

이 필터 패턴은 "eventType" 속성에서 "Trail" 문자열과 일치합니다.

```
{ $.eventType = %Trail% }
```

Example: Filter pattern that uses a wildcard to match values of any element in the array using regex

필터 패턴에는 "arrayKey" 배열에서 "value" 요소와 일치하는 정규식이 포함되어 있습니다.

```
{ $.arrayKey[*] = %val.{2}% }
```

Example: Filter pattern that uses a wildcard to match values of any element with a specific prefix and subnet using regex (IP address)

이 필터 패턴에는 "sourceIPAddress" 속성에서 "111.111.111.111" 요소와 일치하는 정규식이 포함되어 있습니다.

```
{ $.* = %111\.111\.111\.1[0-9]{1,2}% }
```

> **Note**
>
> 할당량
> 속성 선택기에서는 와일드카드 선택기를 최대 1개만 사용할 수 있습니다.

Example: Filter pattern that matches a JSON property with a period (.) in the key

대괄호 표기법을 사용하여 마침표(.)가 포함된 JSON 속성 이름과 일치하도록 필터 패턴을 생성할 수 있습니다. 속성 이름을 작은따옴표(')로 묶습니다. 큰따옴표(")는 대괄호 표기법 내에서 지원되지 않으며 구문 오류와 함께 필터 패턴이 실패합니다. 다음 필터 패턴은 속성과 일치합니다cluster.name.

```
{ $.['cluster.name'] = "c" }
```

Example: Filter pattern that matches JSON logs using IS

IS 변수와 JSON 로그의 필드를 일치시키는 필터 패턴을 생성할 수 있습니다. IS 변수는 NULL, TRUE 또는 FALSE 값을 포함하는 필드와 일치할 수 있습니다. 다음 필터 패턴은 SomeObject 값이 NULL인 JSON 로그를 반환합니다.

```
{ $.SomeObject IS NULL }
```

Example: Filter pattern that matches JSON logs using not_exists

로그 데이터에서 특정 필드를 포함하지 않는 JSON 로그를 반환하는 NOT EXISTS 변수를 사용하여 필터 패턴을 생성할 수 있습니다. 다음 필터 패턴은 SomeOtherObject 필드를 포함하지 않는 JSON 로그를 반환하는 NOT EXISTS를 사용합니다.

```
{ $.SomeOtherObject NOT EXISTS }
```

> **Note**
>
> 변수 IS NOT 및 EXISTS은 현재 지원되지 않습니다.

복합 표현식을 사용하여 JSON 객체에서 일치하는 용어 검색

필터 패턴에서 논리 연산자 AND('&&') 및 OR('||')을 사용하여 둘 이상의 조건이 참인 로그 이벤트와 일치하는 복합 표현식을 생성할 수 있습니다. 복합 표현식은 괄호('()') 사용과 () > && > || 표준 연산 순서를 지원합니다. 다음 예제에는 JSON 객체에서 일치하는 용어를 검색하도록 복합 표현식에서 필터 패턴을 사용하는 방법을 보여주는 코드 조각이 포함되어 있습니다.

JSON 객체

```
{
    "user": {
        "id": 1,
        "email": "John.Stiles@example.com"
    },
```

```
    "users": [
        {
         "id": 2,
         "email": "John.Doe@example.com"
        },
        {
         "id": 3,
         "email": "Jane.Doe@example.com"
        }
    ],
    "actions": [
        "GET",
        "PUT",
        "DELETE"
    ],
    "coordinates": [
        [0, 1, 2],
        [4, 5, 6],
        [7, 8, 9]
    ]
}
```

Example: Expression that matches using AND (&&)

이 필터 패턴에는 숫자 값이 1인 "user"의 "id"와 일치하고 문자열이 "John.Doe@example.com"인 "users" 배열 첫 번째 요소의 "email"과 일치하는 복합 표현식이 포함되어 있습니다.

```
{ ($.user.id = 1) && ($.users[0].email = "John.Doe@example.com") }
```

Example: Expression that matches using OR (||)

이 필터 패턴에는 문자열이 "John.Stiles@example.com"인 "user"의 "email"과 일치하는 복합 표현식이 포함되어 있습니다.

```
{ $.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = "nonmatch" &&
 $.actions[2] = "nonmatch" }
```

Example: Expression that doesn't match using AND (&&)

이 필터 패턴에는 표현식이 "actions"의 세 번째 작업과 일치하지 않아서 일치 항목을 찾지 못하는 복합 표현식이 포함되어 있습니다.

```
{ ($.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = "nonmatch") &&
 $.actions[2] = "nonmatch" }
```

> **Note**
>
> 할당량
> 속성 선택기에서 와일드카드 선택기를 최대 1개만 사용할 수 있으며, 복합 표현식이 포함된 필터 패턴에서는 와일드카드 선택기를 3개까지 사용할 수 있습니다.

Example: Expression that doesn't match using OR (||)

이 필터 패턴에는 표현식이 "users"의 첫 번째 속성 또는 "actions"의 세 번째 작업과 일치하지 않아서 일치 항목을 찾지 못하는 복합 표현식이 포함되어 있습니다.

```
{ ($.user.id = 2 && $.users[0].email = "nonmatch") || $.actions[2] = "GET" }
```

필터 패턴을 사용하여 공백으로 구분된 로그 이벤트에서 일치하는 용어 검색

공백으로 구분된 로그 이벤트에 대한 필터 패턴 작성

공백으로 구분된 로그 이벤트에서 일치하는 용어를 검색하도록 필터 패턴을 생성할 수 있습니다. 다음에서는 공백으로 구분된 로그 이벤트의 예제를 제공하고, 공백으로 구분된 로그 이벤트에서 일치하는 용어를 검색하는 필터 패턴 구문을 작성하는 방법을 설명합니다.

> **Note**
>
> 공백으로 구분된 로그 이벤트에서 일치하는 용어를 검색하기 위해 필터 패턴을 생성하는 경우 모든 조건부 정규식을 사용할 수 있습니다. 지원되는 정규식 목록은 지원되는 정규식을 참조하세요.

Example: Space-delimited log event

다음 코드 조각은 7개의 필드(`ip`, `user`, `username`, `timestamp`, `request`, `status_code`, `bytes`)를 포함하는 공백으로 구분된 로그 이벤트를 보여 줍니다.

```
127.0.0.1 Prod frank [10/Oct/2000:13:25:15 -0700] "GET /index.html HTTP/1.0" 404
 1534
```

> **Note**
>
> 대괄호('[]') 와 큰따옴표("") 사이의 문자는 단일 필드로 간주됩니다.

Writing filter patterns that match terms in a space-delimited log event

공백으로 구분된 로그 이벤트에서 일치하는 용어를 검색하는 필터 패턴을 생성하려면 필터 패턴을 대괄호('[]')로 묶고 쉼표(',')로 이름이 구분된 필드를 지정합니다. 다음 필터 패턴은 7개의 필드를 구문 분석합니다.

```
[ip=%127\.0\.0\.[1-9]%, user, username, timestamp, request =*.html*, status_code =
 4*, bytes]
```

숫자 연산자(>, <, =, !=, >= 또는 <=) 및 별표(*)를 와일드카드 또는 정규식으로 사용하여 필터 패턴에 조건을 제공할 수 있습니다. 필터 패턴 예제에서 `ip`는 IP 주소 범위 127.0.0.1~127.0.0.9와 일치

하는 정규식을 사용하고, request에는 .html을 포함하는 값을 추출해야 함을 나타내는 와일드카드가 포함되어 있으며, status_code에는 4로 시작하는 값을 추출해야 함을 나타내는 와일드카드가 포함되어 있습니다.

공백으로 구분된 로그 이벤트에서 구문 분석하는 필드 수를 모르는 경우 말줄임표(...)를 사용하여 이름이 지정되지 않은 필드를 참조할 수 있습니다. 필요 시 말줄임표는 많은 필드를 참조할 수 있습니다. 다음 예제에서는 이전 필터 패턴 예제에 나온 이름이 없는 처음 4개의 필드를 나타내는 말줄임표를 포함하는 필터 패턴을 보여줍니다.

```
[..., request =*.html*, status_code = 4*, bytes]
```

논리 연산자 AND(&&) 및 OR(||)를 사용하여 복합 표현식을 만들 수도 있습니다. 다음 필터 패턴에는 status_code 값은 404 또는 410이어야 함을 나타내는 복합 표현식이 포함되어 있습니다.

```
[ip, user, username, timestamp, request =*.html*, status_code = 404 || status_code =
 410, bytes]
```

패턴 일치를 사용하여 공백으로 구분된 로그 이벤트에서 일치하는 용어 검색

패턴 일치를 사용하여 특정 순서로 용어를 일치시키는 공백으로 구분된 필터 패턴을 생성할 수 있습니다. 표시기로 용어의 순서를 지정합니다. 첫 번째 용어를 나타낼 때는 w1을 사용하고 후속 용어의 순서를 나타낼 때는 w2 등을 사용합니다. 용어 사이에 쉼표(',')를 입력합니다. 다음 예제에는 공백으로 구분된 필터 패턴으로 패턴 일치를 사용하는 방법을 보여주는 코드 조각이 포함되어 있습니다.

> **Note**
>
> 공백으로 구분된 로그 이벤트에서 일치하는 용어를 검색하기 위해 필터 패턴을 생성하는 경우 모든 조건부 정규식을 사용할 수 있습니다. 지원되는 정규식 목록은 지원되는 정규식을 참조하세요.

공백으로 구분된 로그 이벤트

```
INFO 09/25/2014 12:00:00 GET /service/resource/67 1200
```

```
INFO 09/25/2014 12:00:01 POST /service/resource/67/part/111 1310
WARNING 09/25/2014 12:00:02 Invalid user request
ERROR 09/25/2014 12:00:02 Failed to process request
```

Example: Match terms in order

다음과 같은 공백으로 구분된 필터 패턴은 로그 이벤트의 첫 번째 단어가 ERROR인 로그 이벤트를 반환합니다.

```
[w1=ERROR, w2]
```

> **Note**
>
> 패턴 일치를 사용하는 공백으로 구분된 지표 필터를 생성하는 경우 용어 순서를 지정한 후 빈 표시기를 포함해야 합니다. 예를 들어, 첫 번째 단어가 ERROR인 로그 이벤트를 반환하는 필터 패턴을 생성하는 경우 w1 용어 뒤에 빈 w2 표시기를 포함합니다.

Example: Match terms with AND (&&) and OR (||)

논리 연산자 AND('&&') 및 OR('||')을 사용하여 조건을 포함하는 공백으로 구분된 필터 패턴을 생성할 수 있습니다. 다음 필터 패턴은 이벤트의 첫 번째 단어가 ERROR 또는 WARNING인 로그 이벤트를 반환합니다.

```
[w1=ERROR || w1=WARNING, w2]
```

Example: Exclude terms from matches

하나 이상의 용어를 제외한 로그 이벤트를 반환하는 공백으로 구분된 필터 패턴을 생성할 수 있습니다. 제외하려는 용어 앞에 같지 않음 기호('!=')를 입력합니다. 다음 코드 조각은 첫 번째 단어가 ERROR 및 WARNING이 아닌 로그 이벤트를 반환하는 필터 패턴 예제를 보여줍니다.

```
[w1!=ERROR && w1!=WARNING, w2]
```

Example: Match the top level item in a resource URI

다음 코드 조각은 정규식을 사용하여 리소스 URI의 상위 항목과 일치하는 필터 패턴 예제를 보여줍니다.

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+$%, response_time]
```

Example: Match the child level item in a resource URI

다음 코드 조각은 정규식을 사용하여 리소스 URI의 하위 항목과 일치하는 필터 패턴 예제를 보여줍니다.

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+/part/[0-9]+$%,
 response_time]
```

AWS 서비스에서 로깅 활성화

검색 가능한 카탈로그를 사용하여 AWS 서비스를 찾고 로깅 설정 가이드를 엽니다. 각 서비스의 대상 및 권한 모델에 대한 자세한 내용은 지원되는 로그 대상 및 권한을 참조하세요.

많은 서비스가 CloudWatch Logs에만 로그를 게시하지만, 벤딩 로그 전송을 사용하여 Amazon Simple Storage Service 또는 Amazon Data Firehose로 직접 로그를 전송하는 서비스도 있습니다. 직접 전송은 주요 요구 사항이 이러한 대상 중 하나에서 장기 저장 또는 처리인 경우에 유용합니다.

Amazon S3 또는 Firehose에 로그를 직접 게시하더라도 CloudWatch 전송 요금이 적용됩니다. Amazon S3로 로그를 보내면 Cost Explorer 또는 청구서에 *AWS_REGION*-S3-Egress-Bytes 요금이 표시됩니다. Firehose로 로그를 보내면 *AWS_REGION*-FH-Egress-Bytes 요금이 표시됩니다. 판매 로그 요금에 대한 자세한 내용은 Amazon CloudWatch 요금의 로그 탭을 참조하세요.

일부 서비스는 로그를 전송하기 전에 추가 권한이 필요합니다. 이러한 권한이 없으면 로그 전송이 실패합니다. 비교 표에서 원래 권한 모델을 사용하는 서비스에는 지원되는(V1 권한) 레이블이 지정됩니다. 현재 모델을 사용하는 서비스에는 Supported(V2 권한) 레이블이 지정됩니다. 각 레이블은 필수 정책에 연결됩니다.
<integration-catalog>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>

<integration></integration>
<integration></integration>
<integration></integration>

<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
</integration-catalog>

지원되는 로그 대상 및 권한

다음 표에는 각 AWS 서비스가 로그 전송을 지원하는 대상과 필요한 권한 버전이 나와 있습니다.

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
Amazon API Gateway 액세스 로그	벤딩 로그	지원됨 (V1 권한)			
Application Load Balancer 액세스 로그	벤딩 로그	지원되는 [V2 권한]	지원되는 [V2 권한]	지원되는 [V2 권한]	
AWS AppSync 로그	사용자 지정 로그	지원됨			
Amazon Aurora MySQL 로그	사용자 지정 로그	지원됨			
Amazon Bedrock 지식 기반 로깅	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)
Amazon Bedrock 에이전트 로깅	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Bedrock AgentCore 런타임	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)
Amazon Bedrock AgentCore 게이트웨이	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)
Amazon Bedrock AgentCore 자격 증명	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
Amazon Bedrock AgentCore 메모리	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)
Amazon Bedrock AgentCore 결제	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)
Amazon Bedrock AgentCore 도구	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)
Amazon Chime 미디어 품질 지표 로그 및 SIP 메시지 로그	벤딩 로그	지원됨 (V1 권한)			
CloudFront: 액세스 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
AWS CloudHSM 감사 로그	사용자 지정 로그	지원됨			
CloudWatch Evidently 평가 이벤트 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)		
CloudWatch Internet Monitor 로그	벤딩 로그		지원됨 (V1 권한)		
the section called “CloudWatch Logs Insights 쿼리 실행 로그에 대한 전송 소스 생성”	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
CloudTrail 로그	사용자 지정 로그	지원됨			

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
AWS CodeBuild 로그	사용자 지정 로그	지원됨			
Amazon CodeWhisperer 이벤트 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Cognito 로그	벤딩 로그	지원됨 (V1 권한)			
Amazon Connect 고객 로그	사용자 지정 로그	지원됨			
AWS DataSync 로그	사용자 지정 로그	지원됨			
AWS DevOps Agent 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon ElastiCache(Redis OSS) 로그	벤딩 로그	지원됨 (V1 권한)		지원됨 (V1 권한)	
AWS Elastic Beanstalk 로그	사용자 지정 로그	지원됨			
Amazon Elastic Container Service 로그	사용자 지정 로그	지원됨			
Amazon Elastic Kubernetes 서비스 자동 모드 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
Amazon Elastic Kubernetes 서비스 기능 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Elastic Kubernetes Service 컨트롤 플레인 로그	벤딩 로그	지원됨			
AWS Elemental MediaPackage 액세스 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
AWS Elemental MediaTailor 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
AWS Entity Resolution 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon EventBridge 파이프 로깅	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon EventBridge 이벤트 버스	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon EventBridge 이벤트 버스 (사용자 지정) 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
AWS Fargate 로그	사용자 지정 로그	지원됨			
AWS Fault Injection Service 실험 로그	벤딩 로그		지원됨 (V1 권한)		

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
Amazon FinSpace	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
AWS Global Accelerator 흐름 로그	벤딩 로그		지원됨 (V1 권한)		
AWS Glue 작업 로그	사용자 지정 로그	지원됨			
IAM Identity Center 오류 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Interactive Video Service 채팅 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
AWS IoT 로그	사용자 지정 로그	지원됨			
AWS IoT FleetWise 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
AWS Lambda 로그	벤딩 로그	지원됨	지원됨	지원됨	
Amazon Macie 로그	사용자 지정 로그	지원됨			
AWS Mainframe Modernization	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon Managed Service for Prometheus 로그	벤딩 로그	지원됨 (V1 권한)			

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
Amazon MSK 브로커 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon MSK Connect 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon MQ 로그	사용자 지정 로그	지원됨			
AWS 네트워크 방화벽 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
AWS Network Firewall 프록시 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Network Load Balancer 액세스 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
OpenSearch 로그	사용자 지정 로그	지원됨			
Amazon OpenSearch Service 수집 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
AWS PCS 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Q Business 커넥터 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
Amazon Q Business 대화 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Q in Connect AI 에이전트 이벤트 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Quick 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Relational Database Service PostgreSQL 로그	사용자 지정 로그	지원됨			
Amazon Route 53 Global Resolver 쿼리 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Route 53 퍼블릭 DNS 쿼리 로그	벤딩 로그	지원됨			
Amazon Route 53 해석기 쿼리 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)		
AWS RTB Fabric 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
AWS Security Hub CSPM	벤딩 로그	지원됨 (V2 권한)			
AWS Security Hub	벤딩 로그	지원됨 (V2 권한)			

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
AWS Shield Advanced 공격 흐름 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Simple Storage Service 서버 액세스 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon SageMaker AI 이벤트	벤딩 로그	지원됨 (V1 권한)			
Amazon SageMaker AI 작업자 이벤트	벤딩 로그	지원됨 (V1 권한)			
AWS Site-to-Site VPN 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon Simple Email Service 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
Amazon Simple Notification Service 로그	사용자 지정 로그	지원됨			
Amazon Simple Notification Service 데이터 보호 정책 로그	사용자 지정 로그	지원됨			
EC2 스팟 인스턴스 데이터 피드 파일	벤딩 로그		지원됨 (V1 권한)		
AWS Step Functions Express 워크플로 및 표준 워크플로 로그	벤딩 로그	지원됨 (V1 권한)			

로그 소스	로그 유형	the section called “CloudWatch Logs로 전송된 로그”	the section called “Amazon S3로 전송된 로그”	the section called “Firehose에 전송된 로그”	the section called “X-Ray로 전송된 트레이스”
Storage Gateway 감사 로그 및 상태 로그	벤딩 로그	지원됨 (V1 권한)			
AWS Transfer Family 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
AWS Verified Access 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon Virtual Private Cloud 흐름 로그	벤딩 로그	지원됨	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon VPC Lattice 액세스 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨 (V1 권한)	
Amazon VPC Route Server	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	
AWS WAF 로그	벤딩 로그	지원됨 (V1 권한)	지원됨 (V1 권한)	지원됨	
Amazon WorkMail 감사 로그	벤딩 로그	지원됨 (V2 권한)	지원됨 (V2 권한)	지원됨 (V2 권한)	

추가 권한 [V1]이 필요한 로깅

일부 AWS 서비스는 공통 인프라를 사용하여 CloudWatch Logs, Amazon S3 또는 Firehose로 로그를 전송합니다. 위 표에 나열된 AWS 서비스가 이러한 대상으로 로그를 전송하도록 하려면 특정 권한이 있는 사용자로 로그인해야 합니다.

또한 로그 전송을 활성화 AWS 하려면에 권한을 부여해야 합니다.는 로그가 설정될 때 이러한 권한을 AWS 자동으로 생성하거나 로깅을 설정하기 전에 직접 생성할 수 있습니다. 교차 계정 전송의 경우 권한 정책을 직접 수동으로 생성해야 합니다.

사용자 또는 조직의 누군가가 로그 전송을 처음 설정할 때 필요한 권한 및 리소스 정책을 AWS 자동으로 설정하도록 선택한 경우 로그 전송을 설정하는 사용자에게는이 섹션의 뒷부분에 설명된 대로 특정 권한이 있어야 합니다. 또는 리소스 정책을 직접 만들면 로그 전송을 설정한 사용자에게 많은 권한이 필요하지 않습니다.

다음 주제에서는 각 대상에 대한 더 자세한 내용을 제공합니다.

주제

- CloudWatch Logs로 전송된 로그
- Amazon S3로 전송된 로그
- Firehose에 전송된 로그

CloudWatch Logs로 전송된 로그

> **Important**
>
> CloudWatch Logs로 전송되도록 다음 목록의 로그 유형을 설정하면 AWS 에서는 로그 그룹과 연결된 리소스 정책을 생성하거나 변경하여 필요한 경우 로그를 수신합니다. 자세한 내용을 확인하려면 이 섹션을 계속 읽습니다.

이 섹션은 이전 섹션의 표에 나열된 로그 유형이 CloudWatch Logs로 전송될 때 적용됩니다.

사용자 권한

처음으로, 이러한 유형의 로그를 CloudWatch Logs로 전송하도록 설정하려면 다음 권한이 있는 계정에 로그인해야 합니다.

- `logs:CreateLogDelivery`
- `logs:PutResourcePolicy`
- `logs:DescribeResourcePolicies`
- `logs:DescribeLogGroups`

> **Note**
>
> `logs:DescribeLogGroups`, `logs:DescribeResourcePolicies` 또는 `logs:PutResourcePolicy` 권한을 지정할 때는 단일 로그 그룹 이름만 지정하는 대신 * 와일드카드를 사용할 수 있도록 `Resource` 줄의 ARN을 설정해야 합니다. 예: `"Resource": "arn:aws:logs:us-east-1:111122223333:log-group:*"`

이러한 유형의 로그가 이미 CloudWatch Logs의 로그 그룹으로 전송되고 있는 경우 해당 유형 로그 중 다른 하나를 동일한 로그 그룹으로 보내도록 설정하려면 `logs:CreateLogDelivery` 권한만 있으면 됩니다.

로그 그룹 리소스 정책

로그를 보내는 로그 그룹에는 특정 권한이 포함된 리소스 정책이 있어야 합니다. 로그 그룹에 현재 리소스 정책이 없고 로깅을 설정하는 사용자에게 로그 그룹에 대한 `logs:PutResourcePolicy`,`logs:DescribeResourcePolicies` 및 `logs:DescribeLogGroups` 권한이 있는 경우 AWS 에서는 로그를 CloudWatch Logs 로그로 전송하기 시작할 때 자동으로 다음 정책을 생성합니다. 새로 생성된 구독의 경우 리소스 정책은 로그 그룹 수준에서 구성되며 최대 크기는 51,200바이트입니다. 기존 계정 수준 리소스 정책이 이미 와일드카드를 통해 권한을 부여하는 경우 별도의 로그 그룹 수준 정책은 생성되지 않습니다. 특정 로그 그룹에 대한 logGroup 수준 리소스 정책을 확인하려면 `--resource-arn` 파라미터를 로그 그룹 ARN으로 설정하고 `--policy-scope` 파라미터를 로 설정한 `describe-resource-policies` 명령을 사용합니다.`RESOURCE`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
```

```
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-
stream:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
}
```

로그 그룹의 리소스 정책 한도는 51,200바이트입니다. 이 한도에 도달하면 AWS는 새 권한을 추가할 수 없습니다. 이렇게 하려면 고객이 정책을 수동으로 수정하여 `logs:CreateLogStream` 및 `logs:PutLogEvents` 작업에 대한 `delivery.logs.amazonaws.com` 서비스 보안 주체 권한을 부여해야 합니다. 고객은와 같은 와일드카드와 함께 로그 그룹 이름 접두사를 사용해야 `/aws/vendedlogs/*` 하며 나중에 전송을 생성할 때이 로그 그룹 이름을 사용해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
```

```
                ]
            },
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group/aws/
vendedlogs/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
}
```

Amazon S3로 전송된 로그

Amazon S3로 로그를 전송하도록 설정하면 필요한 경우 로그를 수신하는 S3 버킷과 연결된 리소스 정책을 AWS 생성하거나 변경합니다.

Amazon S3에 게시되는 로그는 사용자가 지정한 기존 버킷에 게시됩니다. 지정된 버킷에 5분마다 하나 이상의 로그 파일이 생성됩니다.

Amazon S3 버킷에 처음으로 로그를 전송할 때 로그를 전송하는 서비스는 버킷의 소유자를 기록하여 로그가 이 계정에 속한 버킷에만 전달되도록 합니다. 따라서 Amazon S3 버킷 소유자를 변경하려면 원본 서비스에서 로그 구독을 다시 생성하거나 업데이트해야 합니다.

> **Note**
>
> CloudFront는 판매 로그를 S3에 전송하는 다른 서비스와는 다른 권한 모델을 사용합니다. 자세한 내용은 표준 로깅 구성 및 로그 파일에 액세스에 필요한 권한을 참조하세요.
> 또한 CloudFront 액세스 로그와 다른 로그 소스에 동일한 S3 버킷을 사용하는 경우 CloudFront용 버킷에서 ACL을 활성화하면 이 버킷을 사용하는 다른 모든 로그 소스에도 권한이 부여됩니다.

> **Important**
>
> Amazon S3 버킷에 로그를 전송하고 버킷 정책에 `NotAction` 또는 `NotPrincipal` 요소가 포함된 경우 버킷에 로그 전송 권한을 자동으로 추가하고 로그 구독을 생성하려고 하면 실패합니다. 로그 구독을 성공적으로 생성하려면 버킷 정책에 로그 전송 권한을 수동으로 추가한 다음 로그 구독을 생성해야 합니다. 자세한 내용은 이 섹션의 지침을 참조하세요.
> 버킷에 고객 관리형 AWS KMS 키를 사용한 서버 측 암호화가 있는 경우 고객 관리형 키에 대한 키 정책도 추가해야 합니다. 자세한 내용은 Amazon S3 섹션을 참조하세요.
> 대상 버킷에 SSE-KMS가 있고 버킷 키가 활성화된 경우 연결된 고객 관리형 KMS 키 정책이 더 이상 모든 요청에 대해 예상대로 작동하지 않습니다. 자세한 내용은 Amazon S3 버킷 키를 사용하여 SSE-KMS 비용 절감을 참조하세요.
> 고객 관리 AWS KMS 형 키와 함께 판매 로그 및 S3 암호화를 사용하는 경우 버킷을 구성할 때 AWS KMS 키 ID 대신 정규화된 키 ARN을 사용해야 합니다. 자세한 내용은 put-bucket-encryption을 참조하세요.

사용자 권한

처음으로, 이러한 유형의 로그를 Amazon S3로 전송하도록 설정하려면 다음 권한이 있는 계정에 로그인해야 합니다.

- `logs:CreateLogDelivery`
- `S3:GetBucketPolicy`
- `S3:PutBucketPolicy`

이러한 유형의 로그가 이미 Amazon S3 버킷으로 전송되고 있는 경우 해당 유형 로그 중 다른 하나를 동일한 버킷으로 보내도록 설정하려면 `logs:CreateLogDelivery` 권한만 있으면 됩니다.

S3 버킷 리소스 정책

로그가 전송되는 S3 버킷에는 특정 권한을 포함하는 리소스 정책이 있어야 합니다. 버킷에 현재 리소스 정책이 없고 로깅을 설정하는 사용자에게 해당 버킷에 대한 `S3:GetBucketPolicy` 및 `S3:PutBucketPolicy` 권한이 있는 경우 AWS 에서는 로그를 Amazon S3로 전송하기 시작할 때 자동으로 다음 정책을 생성합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryAclCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
```

```
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
}
```

이전 정책에서 `aws:SourceAccount`에 대해 이 버킷으로 로그를 전달할 계정 IDS의 목록을 지정합니다. `aws:SourceArn`에 대해 로그를 생성하는 리소스의 ARN 목록을 `arn:aws:logs:`*source-region*`:`*source-account-id*`:*` 형식으로 지정합니다.

버킷에 리소스 정책이 있지만 해당 정책에 이전 정책에 표시된 명령문이 포함되어 있지 않고 로깅을 설정하는 사용자에게 버킷에 대한 `S3:GetBucketPolicy` 및 `S3:PutBucketPolicy` 권한이 있는 경우 해당 명령문이 로그 그룹의 리소스 정책에 추가됩니다.

> **Note**
>
> 경우에 따라에 `s3:ListBucket` 권한이 부여되지 않은 AWS CloudTrail 경우에 `AccessDenied` 오류가 표시될 수 있습니다`delivery.logs.amazonaws.com`. CloudTrail 로그에서 이러한 오류를 방지하려면 `delivery.logs.amazonaws.com`에 `s3:ListBucket` 권한을 부여하고 앞의 버킷 정책에서 설정한 `s3:GetBucketAcl` 권한과 함께 표시된 `Condition` 파라미터를 포함해야 합니다. 이 작업을 더 간단하게 하려면 새 `Statement`을 생성하는 대신 `AWSLogDeliveryAclCheck`을 “`Action”: [“s3:GetBucketAcl”, “s3:ListBucket”]`로 직접 업데이트할 수 있습니다.

Amazon S3 버킷 서버 측 암호화

Amazon S3 S3-managed 키를 사용한 서버 측 암호화(SSE-S3) 또는에 저장된 AWS KMS 키를 사용한 서버 측 암호화 AWS Key Management Service (SSE-KMS)를 활성화하여 Amazon S3 버킷의 데이터를 보호할 수 있습니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

SSE-S3를 선택하면 추가 구성이 필요하지 않습니다. Amazon S3는 암호화 키를 처리합니다.

> **Warning**
>
> 이 시나리오에서는 관리형 키 사용이 지원되지 않으므로 SSE-KMS를 선택하는 경우 고객 AWS 관리형 키를 사용해야 합니다. AWS 관리형 키를 사용하여 암호화를 설정하면 로그가 읽을 수 없는 형식으로 전송됩니다.

고객 관리형 AWS KMS 키를 사용하는 경우 버킷 암호화를 활성화할 때 고객 관리형 키의 Amazon 리소스 이름(ARN)을 지정할 수 있습니다. 로그 전달 계정이 S3 버킷에 쓸 수 있으려면 다음 사항을 S3 버킷의 버킷 정책이 아니라 고객 관리형 키의 키 정책에 추가해야 합니다.

SSE-KMS를 선택하면 해당 시나리오에서는 AWS 관리형 키가 지원되지 않기 때문에 고객 관리형 키를 사용해야 합니다. 고객 관리형 AWS KMS 키를 사용하는 경우 버킷 암호화를 활성화할 때 고객 관리형 키의 Amazon 리소스 이름(ARN)을 지정할 수 있습니다. 로그 전달 계정이 S3 버킷에 쓸 수 있으려면 다음 사항을 S3 버킷의 버킷 정책이 아니라 고객 관리형 키의 키 정책에 추가해야 합니다.

```
{
    "Sid": "Allow Logs Delivery to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": [ "delivery.logs.amazonaws.com" ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": ["0123456789"]
```

```
        },
        "ArnLike": {
            "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
        }
        }
}
```

aws:SourceAccount에 대해 이 버킷으로 로그를 전달할 계정 IDS의 목록을 지정합니다. aws:SourceArn에 로그를 생성하는 리소스의 ARN 목록을 arn:aws:logs:*source-region*:*source-account-id*:* 형식으로 지정합니다.

Firehose에 전송된 로그

이 섹션은 이전 섹션의 테이블에 나열된 로그 유형이 Firehose로 전송될 때 적용됩니다.

사용자 권한

이러한 유형의 로그를 처음으로 Firehose에 전송하도록 설정하려면 다음 권한이 있는 계정에 로그인해야 합니다.

- logs:CreateLogDelivery
- firehose:TagDeliveryStream
- iam:CreateServiceLinkedRole

이러한 유형의 로그가 이미 Firehose로 전송되고 있는 경우 해당 유형 로그 중 다른 하나를 Firehose로 전송하도록 설정하려면 logs:CreateLogDelivery 및 firehose:TagDeliveryStream 권한만 있으면 됩니다.

권한에 사용되는 IAM 역할

Firehose는 리소스 정책을 사용하지 않으므로 이러한 로그를 Firehose로 전송하도록 설정할 때 IAM 역할을 AWS 사용합니다.는 라는 서비스 연결 역할을 AWS 생성합니다AWSServiceRoleForLogDelivery. 이 서비스 연결 역할에는 다음 권한이 포함됩니다.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:ListTagsForDeliveryStream"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/LogDeliveryEnabled": "true"
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

이 서비스 연결 역할은 LogDeliveryEnabled 태그가 로 설정된 모든 Firehose 전송 스트림에 대한 권한을 부여합니다true. 로깅을 설정할 때는이 태그를 대상 전송 스트림에 AWS 제공합니다.

또한 이 서비스 연결 역할에는 delivery.logs.amazonaws.com 서비스 보안 주체가 필요한 서비스 연결 역할을 맡도록 허용하는 신뢰 정책이 있습니다. 해당 신뢰 정책은 다음과 같습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

추가 권한 [V2]가 필요한 로깅

일부 AWS 서비스는 새 메서드를 사용하여 로그를 전송합니다. 이는 이러한 서비스에서 CloudWatch Logs, Amazon S3 또는 트레이스 전송용 Firehose 및 X-Ray라는 대상 중 하나 이상으로 로그 전송을 설정할 수 있는 유연한 방법입니다.

작동하는 로그 전송은 다음 세 가지 요소로 구성됩니다.

- 실제로 로그를 전송하는 리소스를 나타내는 논리적 객체인 `DeliverySource`.
- 실제 전달 대상을 나타내는 논리적 객체인 `DeliveryDestination`.
- 전송 소스를 전송 대상에 연결하는 `Delivery`.

지원되는 AWS 서비스와 대상 간에 로그 전송을 구성하려면 다음을 수행해야 합니다.

- PutDeliverySource를 사용하여 전송 소스를 생성합니다.
- PutDeliveryDestination을 사용하여 전송 대상을 생성합니다.
- 교차 계정 로그를 전송하는 경우 Amazon S3 및 Firehose 대상만 사용할 수 있습니다. 대상 계정에 IAM 정책을 할당하려면 대상 계정에서 PutDeliveryDestinationPolicy를 사용해야 합니다. 이 정책은 계정 A의 전송 소스에서 계정 B의 전송 대상으로의 전송을 생성할 수 있는 권한을 부여합니다. 교차 계정 전송의 경우 권한 정책을 직접 수동으로 생성해야 합니다. 설정 예제는 단원을 참조하십시오교차 계정 전송 예제.
- CreateDelivery를 사용하여 정확히 하나의 전송 소스와 하나의 전송 대상을 페어링하여 전송을 생성합니다.

로그 전송 설정 예제

다음 예제에서는 동일한 AWS 계정에서 전송 소스와 전송 대상을 생성합니다. 소스 리소스 ARN 및 로그 유형을 로그를 생성하는 서비스에서 지원하는 값으로 바꿉니다. 이 예제에서는 전송 대상 정책이 필요하지 않습니다.

CloudWatch Logs Insights 쿼리 실행 로그에 대한 전송 소스 생성

다음 예시에서는 CloudWatch Logs Insights 쿼리 실행 로그에 대한 전송 소스를 생성합니다. 전체 `logType` 및 리소스 ARN 요구 사항은 PutDeliverySource를 참조하세요.

> **Note**
>
> 쿼리 실행 로그log-group:*가 하나의 로그 그룹에 연결되어 있지 않기 때문에의 후행 와일드카드가 필요합니다. AWS 는이 로그 유형에 대한 특정 로그 그룹 ARN을 지원하지 않습니다.

```
aws logs put-delivery-source \
    --name insights-query-logs \
    --resource-arn arn:aws:logs:region:account-id:log-group:* \
    --log-type INSIGHTS_QUERY_LOGS
```

전송 소스 생성

전송 소스를 생성한 다음 다음 대상 예제 중 하나를 사용합니다.

```
aws logs put-delivery-source \
    --name my-delivery-source \
    --resource-arn source-resource-arn \
    --log-type log-type
```

CloudWatch Logs에 대한 전송 생성

기존 로그 그룹의 전송 대상을 생성합니다.

```
aws logs put-delivery-destination \
    --name my-cwl-delivery-destination \
    --delivery-destination-configuration \
    "destinationResourceArn=arn:aws:logs:region:account-id:log-group:log-group-name"
```

전송을 생성합니다.

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
cwl-delivery-destination
```

Amazon S3로 전송 생성

기존 버킷의 전송 대상을 생성합니다.

```
aws logs put-delivery-destination \
    --name my-s3-delivery-destination \
    --delivery-destination-configuration \
    "destinationResourceArn=arn:aws:s3:::bucket-name"
```

전송을 생성합니다.

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
s3-delivery-destination
```

대상 접두사, 접미사 경로 또는 Hive 호환 경로를 구성하려면 섹션을 참조하세요Amazon S3 객체 키 형식.

Firehose로 전송 생성

기존 DirectPut 전송 스트림의 전송 대상을 생성합니다.

```
aws logs put-delivery-destination \
    --name my-firehose-delivery-destination \
    --delivery-destination-configuration \
    "destinationResourceArn=arn:aws:firehose:region:account-id:deliverystream/delivery-
stream-name"
```

전송을 생성합니다.

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
firehose-delivery-destination
```

X-Ray로 전송 생성

추적 전송의 경우 X-Ray로의 전송을 지원하는 소스 서비스 및 로그 유형을 사용합니다. 논리적 X-Ray 전송 대상을 생성합니다.

```
aws logs put-delivery-destination \
    --name my-xray-delivery-destination \
    --delivery-destination-type XRAY
```

전송을 생성합니다.

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
xray-delivery-destination
```

전송을 확인하려면 다음 명령을 사용합니다.

```
aws logs describe-deliveries \
    --delivery-source-name-prefix my-delivery-source
```

다음 섹션에서는 V2 프로세스를 사용하여 각 유형의 대상에 대한 로그 전달을 설정하기 위해 로그인할 때 필요한 권한의 세부 정보를 제공합니다. 이러한 권한은 로그인한 IAM 역할에 부여할 수 있습니다.

Important

로그 생성 리소스를 삭제한 후 로그 전송 리소스를 제거하는 것은 사용자의 책임입니다. 이렇게 하려면 다음 단계를 따릅니다.

1. DeleteDelivery 작업을 사용하여 `Delivery`를 삭제합니다.
2. DeleteDeliverySource 작업을 사용하여 `DeliverySource`를 삭제합니다.
3. 방금 삭제한 `DeliverySource`와 연결된 `DeliveryDestination`이 해당 특정 `DeliverySource`에만 사용되는 경우 DeleteDeliveryDestinations 작업을 사용하여 이를 제거할 수 있습니다.

목차

- CloudWatch Logs로 전송된 로그
  - 사용자 권한
  - 로그 그룹 리소스 정책
- Amazon S3로 전송된 로그
  - 사용자 권한
  - Amazon S3 버킷 리소스 정책
  - Amazon S3 버킷 서버 측 암호화
  - Amazon S3 객체 키 형식

- Firehose에 전송된 로그
  - 사용자 권한
  - 리소스 권한에 사용되는 IAM 역할
- X-Ray로 전송된 트레이스
  - 사용자 권한
  - X-Ray 리소스 정책
  - 트랜잭션 검색 활성화

CloudWatch Logs로 전송된 로그

AWS CLI 예제는 섹션을 참조하세요CloudWatch Logs에 대한 전송 생성.

사용자 권한

CloudWatch Logs로 로그를 전송하려면 다음 권한으로 로그인해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
```

```
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
            "arn:aws:logs:us-east-1:444455556666:delivery-source:*",
            "arn:aws:logs:us-east-1:777788889999:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyCWL",
            "Effect": "Allow",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:123456789012:*"
            ]
        }
    ]
}
```

로그 그룹 리소스 정책

로그를 보내는 로그 그룹에는 특정 권한이 포함된 리소스 정책이 있어야 합니다. 로그 그룹에 현재 리소스 정책이 없고 로깅을 설정하는 사용자에게 로그 그룹에 대한 `logs:PutResourcePolicy`,`logs:DescribeResourcePolicies` 및 `logs:DescribeLogGroups` 권한이 있는 경우 AWS 에서는 로그를 CloudWatch Logs 로그로 전송하기 시작할 때 자동으로 다음 정책을 생성합니다. 새로 생성된 구독의 경우 리소스 정책은 로그 그룹 수준에서 구성되며 최대 크기는 51,200바이트입니다. 기존 계정 수준 리소스 정책이 이미 와일드카드를 통해 권한을 부여하는 경우 별도의 로그 그룹 수준 정책은 생성되지 않습니다. 특정 로그 그룹에 대

한 logGroup 수준 리소스 정책을 확인하려면 `--resource-arn` 파라미터를 로그 그룹 ARN으로 설정하고 `--policy-scope` 파라미터를 로 설정한 `describe-resource-policies` 명령을 사용합니다RESOURCE.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-
stream:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
}
```

로그 그룹의 리소스 정책 한도는 51,200바이트입니다. 이 한도에 도달하면 AWS는 새 권한을 추가할 수 없습니다. 이렇게 하려면 고객이 정책을 수동으로 수정하여 `logs:CreateLogStream` 및 `logs:PutLogEvents` 작업에 대한 `delivery.logs.amazonaws.com` 서비스 보안 주체 권한을 부여해야 합니다. 고객은와 같은 와일드카드와 함께 로그 그룹 이름 접두사를 사용해야 `/aws/vendedlogs/*` 하며 나중에 전송을 생성할 때이 로그 그룹 이름을 사용해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group/aws/
vendedlogs/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
```

```
}
```

Amazon S3로 전송된 로그

AWS CLI 예제는 섹션을 참조하세요Amazon S3로 전송 생성.

사용자 권한

Amazon S3로 로그를 전송하려면 다음 권한으로 로그인해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyS3",
            "Effect": "Allow",
            "Action": [
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::bucket-name"
        }
    ]
}
```

Amazon S3 버킷 리소스 정책

로그가 전송되는 S3 버킷에는 특정 권한을 포함하는 리소스 정책이 있어야 합니다. 버킷에 현재 리소스 정책이 없고 로깅을 설정하는 사용자에게 해당 버킷에 대한 `S3:GetBucketPolicy` 및 `S3:PutBucketPolicy` 권한이 있는 경우 AWS 에서는 로그를 Amazon S3로 전송하기 시작할 때 자동으로 다음 정책을 생성합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
```

```
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:delivery-source:*"
                    ]
                }
            }
        }
    ]
}
```

이전 정책에서 aws:SourceAccount에 대해 이 버킷으로 로그를 전달할 계정 IDS의 목록을 지정합니다. aws:SourceArn에 대해 로그를 생성하는 리소스의 ARN 목록을 arn:aws:logs:*source-region*:*source-account-id*:* 형식으로 지정합니다.

버킷에 리소스 정책이 있지만 해당 정책에 이전 정책에 표시된 명령문이 포함되어 있지 않고 로깅을 설정하는 사용자에게 버킷에 대한 S3:GetBucketPolicy 및 S3:PutBucketPolicy 권한이 있는 경우 해당 명령문이 로그 그룹의 리소스 정책에 추가됩니다.

> **Note**
>
> 경우에 따라에 s3:ListBucket 권한이 부여되지 않은 AWS CloudTrail 경우에 AccessDenied 오류가 표시될 수 있습니다delivery.logs.amazonaws.com. CloudTrail 로그에서 이러한 오류를 방지하려면 delivery.logs.amazonaws.com에 s3:ListBucket 권한을 부여하고 앞의 버킷 정책에서 설정한 s3:GetBucketAcl 권한과 함께 표시된 Condition 파라미터를 포함해야 합니다. 이 작업을 더 간단하게 하려면 새 Statement을 생성하는 대신 AWSLogDeliveryAclCheck을 “Action”: [“s3:GetBucketAcl”, “s3:ListBucket”]로 직접 업데이트할 수 있습니다.

Amazon S3 버킷 서버 측 암호화

서버 측 암호화를 활성화하여 Amazon S3 버킷의 데이터를 보호할 수 있습니다. Amazon S3-managed 키(SSE-S3) 또는에 저장된 AWS KMS 키 AWS Key Management Service (SSE-KMS)를 사용할 수 있습니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

SSE-S3를 선택하면 추가 구성이 필요하지 않습니다. Amazon S3는 암호화 키를 처리합니다.

> ⚠ 고객 관리형 키 필요
>
> SSE-KMS를 선택하면 고객 관리형 키를 사용해야 합니다. AWS 관리형 키는 사용할 수 없습니다. AWS 관리형 키를 사용하여 암호화를 구성하면 CloudWatch Logs는 로그를 읽을 수 없는 형식으로 전송합니다.

SSE-KMS에서 버킷 암호화를 활성화할 때 키의 Amazon 리소스 이름(ARN)을 지정합니다. 로그 전송 계정이 S3 버킷에 쓸 수 있도록 키 정책(S3 버킷의 버킷 정책 아님)에 다음을 추가합니다.

```
{
    "Sid": "Allow Logs Delivery to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": [ "delivery.logs.amazonaws.com" ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": ["012345678901"]
        },
        "ArnLike": {
            "aws:SourceArn": ["arn:aws:logs:us-east-1:012345678901:delivery-source:*"]
        }
        }
}
```

의 경우 로그가이 버킷에 전달되는 계정 IDs를 aws:SourceAccount지정합니다. 의 경우 전송 소스 ARNs을 형식으로 aws:SourceArn지정합니다arn:aws:logs:*source-region*:*source-account-id*:delivery-source:*.

Amazon S3 객체 키 형식

V2 권한을 사용하는 전송의 경우 Amazon S3 객체 키는 대상 접두사, 로그 유형, 전송의 접미사 경로, Hive 호환 경로의 활성화 여부에 따라 결정됩니다. 정확한 서비스 정의 경로와 지원되는 접미사 변수는 로그 유형에 따라 다릅니다.

대상 접두사

PutDeliveryDestination을 호출할 때 버킷 ARN에 추가하는 선택적 경로입니다. 예: arn:aws:s3:::*bucket-name*/*MyLogPrefix* 전송된 객체는이 접두사로 시작하고 그 뒤에와 같은 로그 유형의 서비스 정의 경로가 옵니다AWSLogs/*source-account-id*/*service-name*/. CloudFront 표준 로깅(v2)의 경우 대상 접두사는 앞이 아닌 기본 경로를 대체합니다.

접미사 경로

S3DeliveryConfiguration에서 개별 전송에 대해 구성하는 선택적 경로입니다. 접미사는 정적 텍스트와 변수를 포함할 수 있습니다. 로그 유형에서 지원하는 변수를 찾으려면 DescribeConfigurationTemplates를 호출하고를 확인합니다allowedSuffixPathFields. 접미사 경로를 지정하지 않으면 로그 유형의 기본 접미사 경로가 사용 가능한 경우 사용됩니다.

Hive 호환 경로

이 enableHiveCompatiblePath인 경우 명시적에서 true지원되는 변수suffixPath는 로 렌더링됩니다*key*=*value*. Application Load Balancer 액세스 로그와 같은 로그 유형의 경우이 형식 지정은 서비스 정의 계정 세그먼트도 변경합니다. 예를 들어, AWSLogs/*source-account-id*/는 AWSLogs/aws-account-id=*source-account-id*/가 됩니다.

를 생략하면 로그 유형의 기본 접미사 경로가 지원하는 경우에만 suffixPathHive 호환 형식이 적용됩니다. 예를 들어 Application Load Balancer 액세스 로그는이 형식을 지원합니다. Amazon S3 서버 액세스 로그를 포함한 다른 로그 유형은 리소스 및 날짜 디렉터리를 포함하여 원래 기본 접미사를 비 Hive 형식으로 유지합니다.

기본값이 Hive 호환 형식 지정을 지원하지 않는 로그 유형의 경우 지원되는 변수suffixPath가 포함된 유효한를 지정합니다. 명시적 접미사는 기본 접미사를 대체합니다. allowedSuffixPathFields DescribeConfigurationTemplates의 로그 유형을 사용하여 필요한 리소스 및 날짜 변수를 포함합니다.

다음 예제에서는의 계정에 대한 Application Load Balancer access-log 객체 키의 시작111122223333을 보여줍니다us-east-1. 별도의 언급이 없는 한이 예제에서는 대상 접두사를 가정하지 않습니다.

구성	객체 키의 시작
Hive 호환 경로 비활성화, 접미사 생략	AWSLogs/111122223333/elasticloadbalancing/us-east-1/2026/09/10/
Hive 호환 경로 활성화됨, 접미사 생략됨	AWSLogs/aws-account-id=111122223333/elasticloadbalancing/region=us-east-1/year=2026/month=09/day=10/
Hive 호환 경로 활성화됨, 접미사 myFolder/{yyyy}/{MM}/{dd}	AWSLogs/aws-account-id=111122223333/elasticloadbalancing/myFolder/year=2026/month=09/day=10/
Hive 호환 경로 활성화됨, 접미사 myFolder/	AWSLogs/111122223333/elasticloadbalancing/myFolder/
대상 접두사 MyLogPrefix , Hive 호환 경로 비활성화됨, 접미사 생략됨	MyLogPrefix/AWSLogs/111122223333/elasticloadbalancing/us-east-1/2026/09/10/

⚠ 정적 접미사 경로 및 버킷 권한

Application Load Balancer 액세스 로그와 같은 로그 유형의 경우 정적 텍스트만 suffixPath 포함하는는 서비스 정의 계정 접두사의 Hive 호환 형식을 활성화하지 않습니다. 예를 들어는 테이블에 표시된 일반 계정 접두사를 myFolder/ 사용합니다. CloudWatch Logs가 일반 계정 접두사로 쓰기를 시도하고 버킷 정책에서 Hive 호환 접두사만 허용하는 경우 전송이 실패합니다.

를 로 enableHiveCompatiblePath 설정하여 Application Load Balancer 전송을 생성하면 true가 자동으로 생성된 버킷 정책을 Hive 호환 접두사s3:PutObject에 부여합니다. 에 있는 로그 유형의 allowedSuffixPathFields에서 지원되는 변수를 하나 이상 포함suffixPath하거나 기본 접미사 경로가 Hive 호환 형식 지정을 지원하는 로그 유형의 suffixPath 경우를 생략합니다.

CloudFront 표준 로깅(v2)은의 변수와 독립적으로 기본 접두사를 포맷합니다`suffixPath`.

> **Note**
>
> CloudFront는 Amazon S3로 로그 전송에 표준 로깅(v2) 경로 동작 및 예제를 문서화합니다.

> **Note**
>
> 접미사 경로 또는 Hive 호환 설정을 업데이트하면 구성이 전파된 후 후속 전송에 영향을 미칩니다. 기존 객체는 이동되지 않습니다. 활성 전송이 해당 대상을 참조하는 동안에는 접두사를 포함하여 대상의 ARN을 변경할 수 없습니다.
> 버킷 정책은 `s3:PutObject` 결과 접두사를 허용해야 합니다. 버킷 정책을 관리할 때는의 Amazon S3 버킷 정책에 표시된 `aws:SourceAccount` 및 `aws:SourceArn` 조건을 유지하고 필요한 접두사에만 액세스 권한을 Amazon S3 버킷 리소스 정책부여합니다.

Firehose에 전송된 로그

AWS CLI 예제는 섹션을 참조하세요Firehose로 전송 생성.

사용자 권한

Firehose로 로그 전송을 활성화하려면 다음 권한으로 로그인해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
```

```
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyFH",
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": [
            "arn:aws:firehose:us-east-1:111122223333:deliverystream/*"
            ]
        },
        {
            "Sid": "CreateServiceLinkedRole",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
```

```
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
        }
    ]
}
```

리소스 권한에 사용되는 IAM 역할

Firehose는 리소스 정책을 사용하지 않으므로 이러한 로그를 Firehose로 전송하도록 설정할 때 IAM 역할을 AWS 사용합니다.는 라는 서비스 연결 역할을 AWS 생성합니다AWSServiceRoleForLogDelivery. 이 서비스 연결 역할에는 다음 권한이 포함됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:ListTagsForDeliveryStream"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/LogDeliveryEnabled": "true"
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

이 서비스 연결 역할은 `LogDeliveryEnabled` 태그가 로 설정된 모든 Firehose 전송 스트림에 대한 권한을 부여합니다`true`. 로깅을 설정할 때는이 태그를 대상 전송 스트림에 AWS 제공합니다.

또한 이 서비스 연결 역할에는 `delivery.logs.amazonaws.com` 서비스 보안 주체가 필요한 서비스 연결 역할을 맡도록 허용하는 신뢰 정책이 있습니다. 해당 신뢰 정책은 다음과 같습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

X-Ray로 전송된 트레이스

AWS CLI 예제는 섹션을 참조하세요X-Ray로 전송 생성.

사용자 권한

추적 전송을 활성화하려면 다음 권한으로 로그인해야 AWS X-Ray합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
              "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
```

```
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
            "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
            "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyXRay",
            "Effect": "Allow",
            "Action": [
                "xray:PutResourcePolicy",
                "xray:ListResourcePolicies",
                "xray:GetTraceSegmentDestination"
            ],
            "Resource": "*"
        }
    ]
}
```

X-Ray 리소스 정책

트레이스가 전송되는 대상 계정에는 특정 권한이 포함된 리소스 정책이 있어야 합니다. 추적을 설정하는 사용자가 계정에 `xray:PutResourcePolicy` 및 `xray:ListResourcePolicies` 권한이 있으면 X-Ray로 추적 전송을 시작할 때가 리소스 정책을 AWS 자동으로 생성합니다. 생성되는 정책은 소스 서비스에 따라 다릅니다.

Amazon Bedrock AgentCore 리소스

AWS 는 리소스 유형당 하나의 리소스 정책을 생성합니다. 이 정책은 계정 경계로 범위가 지정된 와일드카드 패턴을 사용하며, 계정에서 동일한 Amazon Bedrock AgentCore 리소스 유형의 모든 리소스를 다룹니다. 예를 들어 Amazon Bedrock AgentCore 메모리 리소스가 추적 전송에 대해 활성화된 경우 정책은 향후 생성된 메모리 리소스를 포함하여 해당 계정의 모든 메모리 리소스를 포함합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "xray:PutTraceSegments",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ForAllValues:ArnLike": {
          "logs:LogGeneratingResourceArns": "arn:aws:bedrock-agentcore:us-
east-1:123456789012:memory/*"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:delivery-
source:*"
        }
      }
    }
  ]
}
```

기타 AWS 서비스

추적 전송을 지원하는 다른 서비스의 경우 AWS 는 특정 소스 리소스로 범위가 지정된 리소스 정책을 생성합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "xray:PutTraceSegments",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ForAllValues:ArnLike": {
          "logs:LogGeneratingResourceArns": "arn:aws:bedrock:us-
east-1:123456789012:knowledge-base/KnowledgeBaseId"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:delivery-
source:xray-test"
        }
      }
    }
  ]
}
```

트랜잭션 검색 활성화

X-Ray로 트레이스 전송을 활성화하려면 트랜잭션 검색을 활성화해야 합니다.

서비스별 권한

이전 섹션에 나열된 대상별 권한 외에도 일부 서비스는 보안 강화를 위해 고객이 리소스에서 로그를 전송할 수 있도록 허용하는 명시적 권한 부여가 필요합니다. 이 정책은 해당 서비스 내에서 로그를 제공하는 리소스에 대한 `AllowVendedLogDeliveryForResource` 작업을 승인합니다. 이러한 서비스의 경우 다음 정책을 사용하고 서비스 네임스페이스(*###*), 리전(*##*), AWS 계정 ID(*account-*

id) 및 리소스 유형(*### ##*)을 리소스에 적합한 값으로 바꿉니다. 로그를 제공하는 모든 서비스가 이 작업을 사용하는 것은 아닙니다. 액세스 권한을 부여하는 서비스에 대한 판매 로그 설명서에서 `AllowVendedLogDeliveryForResource` 작업 적용 여부를 포함하여 필요한 권한과 값을 확인합니다. 다음 예제에서는 Amazon SES에서 로그를 벤딩하는 정책을 보여줍니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceLevelAccessForLogDelivery",
            "Effect": "Allow",
            "Action": [
                "ses:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:ses:us-east-1:123456789012:resource-type/*"
        }
    ]
}
```

콘솔별 권한

API 대신 콘솔을 사용하여 로그 전송을 설정하는 경우 이전 섹션에 나열된 권한 외에도 다음과 같은 추가 권한이 필요합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogDeliveryActionsConsoleCWL",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup"
            ],
            "Resource": [
```

```
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleFH",
            "Effect": "Allow",
            "Action": [
                "firehose:ListDeliveryStreams",
                "firehose:DescribeDeliveryStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

교차 계정 전송 예제

이 예제에서는 두 개의 계정이 관련되어 있습니다. 로그 생성 리소스가 있는 계정은 계정 A, ID: *123456789012*이며, 로그 소모 리소스가 있는 계정은 계정 B, ID: *111122223333*입니다.

계정 A는 ARN arn:aws:bedrock:*us-east-1*:*123456789012*:knowledge-base/*kb-12345678*을 사용하여 계정의 Amazon Bedrock 지식 기반에서 로그를 전달하려고 합니다.

이 예제에서 계정 A에는 다음 권한이 필요합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowVendedLogDeliveryForKnowledgeBase",
            "Effect": "Allow",
            "Action": [
                "bedrock:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:bedrock:us-east-1:123456789012:knowledge-
base/XXXXXXXXXX"
        },
        {
            "Sid": "CreateLogDeliveryPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliverySource",
                "logs:CreateDelivery"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:123456789012:delivery-source:*",
    "arn:aws:logs:us-east-1:123456789012:delivery:*",
    "arn:aws:logs:us-east-1:444455556666:delivery-destination:*"
            ]
        }
    ]
}
```

전송 소스 생성

먼저 계정 A는 bedrock 지식 기반을 사용하여 전송 소스를 생성합니다.

```
aws logs put-delivery-source --name my-delivery-source --log-type APPLICATION_LOGS --
resource-arn arn:aws:bedrock:region:AAAAAAAAAAAA:knowledge-base/XXXXXXXXXX
```

다음으로 계정 B는 아래 흐름 중 하나를 사용하여 전송 대상을 생성해야 합니다.

- Amazon S3 버킷으로의 전송 구성

- Firehose 스트림으로의 전송 구성

Amazon S3 버킷으로의 전송 구성

계정 B는 ARN arn:aws:s3:::amzn-s3-demo-bucket을 사용하여 S3 버킷에 로그를 수신하려고 합니다. 이 예제에서 계정 B에는 다음 권한이 필요합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PutLogDestinationPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliveryDestination",
                "logs:PutDeliveryDestinationPolicy"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:*"
        }
    ]
}
```

버킷은 버킷 정책에서 다음 권한이 필요합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogsDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": [
```

```
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/123456789012/
*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                    "123456789012"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
                    ]
                }
            }
        }
    ]
}
```

버킷이 SSE-KMS로 암호화된 경우 AWS KMS 키 정책에 적절한 권한이 있는지 확인합니다. 예를 들어, KMS 키가 `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`인 경우 다음을 사용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogsGenerateDataKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey"
            ],
```

```
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                    "123456789012"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
                    ]
                }
            }
        }
    ]
}
```

그런 다음 계정 B는 S3 버킷을 대상 리소스로 사용하여 전송 대상을 생성할 수 있습니다.

```
aws logs put-delivery-destination --name my-s3-delivery-destination --delivery-
destination-configuration "destinationResourceArn=arn:aws:s3:::amzn-s3-demo-bucket"
```

다음으로 계정 B는 새로 생성된 전송 대상에 전송 대상 정책을 생성하여 계정 A가 로그 전송을 생성할 수 있는 권한을 부여합니다. 새로 생성된 전송 대상에 추가할 정책은 다음과 같습니다.

JSON

```
    {
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDelivery",
            "Effect": "Allow",
            "Principal": {
                "AWS": "123456789012"
            },
            "Action": [
                "logs:CreateDelivery"
```

```
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:amzn-s3-demo-bucket"
        }
    ]
}
```

이 정책은 계정 B의 컴퓨터에 destination-policy-s3.json으로 저장됩니다. 이 리소스를 연결하기 위해 계정 B는 다음 명령을 실행합니다.

```
aws logs put-delivery-destination-policy --delivery-destination-name my-s3-delivery-
destination --delivery-destination-policy file://destination-policy-s3.json
```

마지막으로 계정 A는 계정 A의 전송 소스를 계정 B의 전송 대상에 연결하는 전송을 생성합니다.

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-s3-delivery-
destination
```

Firehose 스트림으로의 전송 구성

이 예제에서 계정 B는 Firehose 스트림에 로그를 수신하려고 합니다. Firehose 스트림에는 다음 ARN이 있으며, DirectPut 전송 스트림 유형을 사용하도록 구성되어 있습니다.

arn:aws:firehose:*us-east-1*:*111122223333*:deliverystream/*log-delivery-stream*

이 예제에서 계정 B에는 다음 권한이 필요합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFirehoseCreateSLR",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
```

```
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
        },
        {
            "Sid": "AllowFirehoseTagging",
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": "arn:aws:firehose:us-
east-1:111122223333:deliverystream/X"
        },
        {
            "Sid": "AllowFirehoseDeliveryDestination",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliveryDestination",
                "logs:PutDeliveryDestinationPolicy"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:*"
        }
    ]
}
```

Firehose 스트림의 `true` 태그가 `LogDeliveryEnabled`로 설정되어 있어야 합니다.

그런 다음 계정 B는 Firehose 스트림을 대상 리소스로 사용하여 전송 대상을 생성합니다.

```
aws logs put-delivery-destination --name my-fh-delivery-destination --delivery-
destination-configuration
 "destinationResourceArn=arn:aws:firehose:region:BBBBBBBBBBBB:deliverystream/X"
```

다음으로 계정 B는 새로 생성된 전송 대상에 전송 대상 정책을 생성하여 계정 A가 로그 전송을 생성할 수 있는 권한을 부여합니다. 새로 생성된 전송 대상에 추가할 정책은 다음과 같습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowCreateDelivery",
            "Effect": "Allow",
            "Principal": {
            "AWS": "123456789012"
            },
            "Action": [
                "logs:CreateDelivery"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:amzn-s3-demo-bucket"
        }
    ]
}
```

이 정책은 계정 B의 컴퓨터에 destination-policy-fh.json으로 저장됩니다. 이 리소스를 연결하기 위해 계정 B는 다음 명령을 실행합니다.

```
aws logs put-delivery-destination-policy --delivery-destination-name my-fh-delivery-
destination --delivery-destination-policy file://destination-policy-fh.json
```

마지막으로 계정 A는 계정 A의 전송 소스를 계정 B의 전송 대상에 연결하는 전송을 생성합니다.

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-fh-delivery-
destination
```

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 AWS 제공합니다.

리소스 정책에서 aws:SourceArn, aws:SourceAccount, aws:SourceOrgID 및 aws:SourceOrgPaths 글로벌 조건 컨텍스트 키를 사용하여 CloudWatch Logs가 리소스에 제공하

는 다른 서비스에 대한 권한을 제한하는 것이 좋습니다. aws:SourceArn을 사용하면 하나의 리소스만 교차 서비스 액세스 권한과 연결됩니다. aws:SourceAccount를 사용하면 해당 계정의 모든 리소스가 교차 서비스 사용 권한과 연결됩니다. aws:SourceOrgID를 사용하면 조직 내 모든 계정의 모든 리소스가 교차 서비스 사용 권한과 연결될 수 있습니다. aws:SourceOrgPaths를 사용하여 AWS Organizations 경로 내 계정의 모든 리소스를 교차 서비스 사용과 연결합니다. 경로 사용 및 이해에 대한 자세한 내용은 AWS Organizations 개체 경로 이해를 참조하세요.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자(*)를 포함한 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용합니다. 예: arn:aws:*servicename*:*:*123456789012*:*

만약 aws:SourceArn 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 aws:SourceAccount 및 aws:SourceArn를 모두 사용해야 합니다.

혼동된 대리자 문제로부터 보호하려면 리소스 기반 정책에서 리소스의 조직 ID 또는 조직 경로와 함께 aws:SourceOrgID 또는 aws:SourceOrgPaths 글로벌 조건 컨텍스트 키를 사용합니다. aws:SourceOrgID 또는 aws:SourceOrgPaths 키가 포함된 정책에는 올바른 계정이 자동으로 포함되며 조직에서 계정을 추가, 제거 또는 이동할 때 정책을 수동으로 업데이트할 필요가 없습니다.

1단계: 대상 생성 및 2단계: 대상 생성에서 Kinesis Data Streams와 Firehose에 데이터를 쓰기 위해 CloudWatch Logs에 대한 접근 권한을 부여하는 정책은 혼동된 대리인 문제를 방지하기 위해 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용하는 방법을 보여줍니다.

원격 측정 활성화 규칙을 사용하여 로그 활성화 자동화

원격 측정 활성화 규칙을 사용하여 AWS 리소스에 대한 로그 수집을 자동으로 구성할 수 있습니다. 규칙은 조직 또는 계정 전체에서 로그 수집을 표준화하고 일관된 모니터링 범위를 보장하는 데 도움이 됩니다.

활성화 규칙을 사용하면 다음을 수행할 수 있습니다.

- 규칙 범위와 일치하는 신규 및 기존 리소스에 대한 로깅 자동 활성화
- 조직, 조직 단위(OU) 또는 개별 계정 수준에서 규칙 적용
- 태그를 사용하여 규칙이 영향을 미치는 리소스 필터링

활성화 규칙은 현재 Amazon VPC 흐름 로그, AWS WAF 로그, Route 53 Resolver 쿼리 로그, NLB 액세스 로그, Amazon EKS 제어 플레인 로그, CloudTrail 데이터 및 관리 이벤트, Amazon Bedrock AgentCore 로그, Amazon EC2 세부 지표, AWS Security Hub, Amazon Bedrock AgentCore Gateway, Amazon Bedrock AgentCore 메모리 및 CloudFront 배포와 같은 AWS 원격 측정 소스를 지원합니다.

규칙 동작, 규칙 관리, 문제 해결 및 서비스별 고려 사항을 포함한 활성화 규칙에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 원격 측정 활성화 규칙을 참조하세요.

규칙 평가 계층 구조

활성화 규칙은 조직 규칙 먼저, OU 수준 규칙, 계정 수준 규칙 순으로 계층적으로 평가됩니다. 상위 수준의 규칙은 기준 원격 측정을 제공합니다. 하위 수준 규칙은 원격 측정을 추가할 수 있지만 줄일 수는 없습니다. 충돌하는 규칙이 동일한 범위에 있는 경우 충돌이 해결될 때까지는 적용되지 않습니다.

활성화 규칙 생성

원격 분석 활성화 규칙을 생성하는 방법

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 원격 측정 구성을 선택합니다.
3. 활성화 규칙 탭을 선택하세요.
4. 규칙 추가를 선택합니다.
5. 규칙 이름, 범위(조직, OU 또는 계정), 데이터 소스 및 원격 측정 유형을 지정합니다.
6. 선택적으로 태그를 추가하여 규칙이 영향을 미치는 리소스를 필터링할 수 있습니다.
7. 규칙 생성을 선택합니다.

타사 소스에서 로깅 활성화

CloudWatch Logs는 직접 API 통합 및 Amazon S3 버킷 통합을 통해 타사 소스에서 로그 수집을 지원합니다. AWS Security Hub CSPM을 통해 추가 타사 보안 조사 결과를 받을 수도 있습니다.

직접 타사 통합

CloudWatch Logs는 다음과 같은 타사 소스와의 직접 통합을 제공합니다. 이러한 통합은 직접 API 연결 또는 Amazon S3 버킷 통합을 사용하여 CloudWatch Logs에 로그를 수집합니다.

- CrowdStrike Falcon
- Microsoft Office 365
- Okta Auth0
- Microsoft Entra ID
- Palo Alto Networks NGFW
- Microsoft Windows Events
- Wiz
- Zscaler 인터넷 액세스
- Okta SSO
- SentinelOne 엔드포인트 보안
- GitHub 감사 로그
- ServiceNow CMDB
- Cisco Umbrella

설정 지침은 Amazon CloudWatch 사용 설명서의 타사 통합 설정 가이드를 참조하세요. Amazon CloudWatch

AWS Security Hub CSPM을 통한 추가 소스

타사 보안 파트너는 직접 통합 외에도 조사 결과를 AWS Security Hub CSPM에 전송한 다음 CloudWatch Logs에서 데이터 소스로 사용할 수 있습니다. 다음 표에는 Security Hub CSPM 파트너 통합과 해당 통합 유형이 나와 있습니다.

Security Hub CSPM 조사 결과를 CloudWatch Logs에서 데이터 소스로 활성화하려면 CloudWatch 콘솔에서 AWS Security Hub에 대한 원격 측정 활성화 규칙을 생성합니다. 활성화 규칙은 Security Hub CSPM의 결과를 관리형 로그 그룹으로 자동으로 수집하도록 CloudWatch를 구성합니다. step-by-step 지침은 Amazon CloudWatch 사용 설명서의 원격 측정 활성화 규칙을 참조하세요.

파트너	통합
3CORESec – NTA	Security Hub CSPM을 통해 조사 결과를 전송합니다.
알림 로직 - SIEMless 위협 관리	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Aqua Security - 클라우드 네이티브 보안 플랫폼	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Aqua Security – Kube-bench	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Armor – Armor Anywhere	Security Hub CSPM을 통해 조사 결과를 전송합니다.
AttackIQ	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Barracuda Networks – 클라우드 보안 가디언	Security Hub CSPM을 통해 조사 결과를 전송합니다.
BigID - BigID Enterprise	Security Hub CSPM을 통해 조사 결과를 전송합니다.
파란색 육각형	Security Hub CSPM을 통해 조사 결과를 전송합니다.
체크포인트 - CloudGuard IaaS	Security Hub CSPM을 통해 조사 결과를 전송합니다.
체크포인트 - CloudGuard 자세 관리	Security Hub CSPM을 통해 조사 결과를 전송합니다.

파트너	통합
Claroty – xDome	Security Hub CSPM을 통해 조사 결과를 전송합니다.
클라우드 스토리지 보안 - Amazon S3용 바이러스 백신	Security Hub CSPM을 통해 조사 결과를 전송합니다.
대비 보안 - 대비 평가	Security Hub CSPM을 통해 조사 결과를 전송합니다.
CrowdStrike – CrowdStrike Falcon	Security Hub CSPM을 통해 조사 결과를 전송합니다.
CyberArk - 권한 있는 위협 분석	Security Hub CSPM을 통해 조사 결과를 전송합니다.
데이터 이론	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Drata	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – CASB	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – 클라우드 보안 게이트웨이	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – DLP	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Forcepoint – NGFW	Security Hub CSPM을 통해 조사 결과를 전송합니다.
안개	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Guardicore – Centra	Security Hub CSPM을 통해 조사 결과를 전송합니다.

파트너	통합
HackerOne – 취약성 인텔리전스	Security Hub CSPM을 통해 조사 결과를 전송합니다.
JFrog – Xray	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Juniper Networks – vSRX 차세대 방화벽	Security Hub CSPM을 통해 조사 결과를 전송합니다.
k9 보안 - Access Analyzer	Security Hub CSPM을 통해 조사 결과를 전송합니다.
레이스워크	Security Hub CSPM을 통해 조사 결과를 전송합니다.
McAfee – MVISION CNAPP	Security Hub CSPM을 통해 조사 결과를 전송합니다.
NETSCOUT – 사이버 조사자	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Orca - 클라우드 보안 플랫폼	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Palo Alto Networks – Prisma 클라우드 컴퓨팅	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Palo Alto Networks – Prisma Cloud Enterprise	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Plerion – 클라우드 보안 플랫폼	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Prowler	Security Hub CSPM을 통해 조사 결과를 전송합니다.
품질 - 취약성 관리	Security Hub CSPM을 통해 조사 결과를 전송합니다.

파트너	통합
Rapid7 – InsightVM	Security Hub CSPM을 통해 조사 결과를 전송합니다.
SentinelOne	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Snyk	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Sonrai 보안 - Sonrai Dig	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Sophos - 서버 보호	Security Hub CSPM을 통해 조사 결과를 전송합니다.
StackRox – Kubernetes 보안	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Sumo Logic – 기계 데이터 분석	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Symantec – 클라우드 워크로드 보호	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Tenable.io	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Trend Micro – Cloud One	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Vectra - Cognito Detect	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Wiz	Security Hub CSPM을 통해 조사 결과를 전송합니다.
Caveonix – Caveonix 클라우드	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.

파트너	통합
Cloud Custodian	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.
DisruptOps	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.
Kion	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.
Turbot	Security Hub CSPM을 통해 조사 결과를 보내고 받습니다.

> **Note**
>
> 이 목록에는 작성 시 조사 결과를 보내는 Security Hub 파트너 통합이 반영됩니다. AWS Security Hub는 정기적으로 새 파트너 통합을 추가하므로 사용 가능한 파트너의 up-to-date 목록은 Security Hub 사용 설명서의 Security Hub CSPM과의 타사 제품 통합을 참조하세요. AWS

Amazon S3로 로그 데이터 내보내기

이 장에서는 로그 그룹에서 Amazon S3 버킷으로 로그 데이터를 내보내 사용자 지정 처리 및 분석에 사용하거나 다른 시스템에 로드하는 방법을 설명합니다. 동일한 계정 또는 다른 계정의 S3 버킷으로 내보낼 수 있습니다.

다음을 수행할 수 있습니다.

- AWS Key Management Service (AWS KMS)에서 SSE-KMS로 암호화된 S3 버킷으로 로그 데이터 내보내기
- 보존 기간이 있고 S3 Object Lock이 사용 설정된 S3 버킷으로 로그 데이터 내보내기

로그를 지속적으로 아카이브하는 방법으로 Amazon S3로 정기적으로 내보내지 않는 것이 좋습니다. 이 사용 사례에서는 대신 구독을 사용하는 것이 좋습니다. 구독에 대한 자세한 내용은 구독을 통한 로그 데이터 실시간 처리을 참조하세요.

내보내기 프로세스를 시작하려면 S3 버킷을 생성해서 내보낸 로그 데이터를 저장해야 합니다. S3 버킷에 내보낸 파일을 저장하고 Amazon S3 수명 주기 규칙을 정의하여 내보낸 파일을 자동으로 보관하거나 삭제할 수 있습니다.

AES-256 또는 SSE-KMS로 암호화된 S3 버킷으로 내보낼 수 있습니다. DSSE-KMS로 암호화된 버킷으로의 내보내기는 지원되지 않습니다.

여러 로그 그룹이나 시간 범위에서 내보낸 로그를 동일한 S3 버킷으로 내보낼 수 있습니다. 내보낸 데이터를 구성하려면 내보낸 모든 객체의 Amazon S3 키 접두사로 사용할 각 내보내기 작업의 접두사를 지정합니다. 예: `prod/app-logs/2026-01-03/` 또는 `log-group-name/backup/`

> **Note**
>
> 내보낸 파일 내의 로그 데이터 청크에 대한 시간 기반 정렬은 보장되지 않습니다. Linux 유틸리티를 사용하여 내보낸 로그 필드 데이터를 정렬할 수 있습니다. 예를 들어, 다음 유틸리티 명령은 단일 폴더에 있는 모든 `.gz` 파일의 이벤트를 정렬합니다.
>
> ```
> find . -exec zcat {} + | sed -r 's/^[0-9]+/\x0&/' | sort -z
> ```
>
> 다음 유틸리티 명령은 여러 하위 폴더에서.gz 파일을 정렬합니다.
>
> ```
> find ./*/ -type f -exec zcat {} + | sed -r 's/^[0-9]+/\x0&/' | sort -z
> ```

또한 다른 `stdout` 명령을 사용하여 정렬된 출력을 다른 파일로 파이프하여 저장할 수 있습니다.

로그 데이터가 내보내기가 가능한 상태가 되는 데는 최대 12시간이 걸릴 수 있습니다. 내보내기 작업은 24시간 이후 시간 초과됩니다. 내보내기 작업의 시간이 초과될 경우 내보내기 작업을 생성할 때 시간 범위를 줄이세요.

로그 데이터를 실시간에 가깝게 분석하려면 대신에 CloudWatch Logs Insights를 사용한 로그 분석 또는 구독을 통한 로그 데이터 실시간 처리을 참조하세요.

내용

- 개념
- 콘솔을 사용하여 Amazon S3로 로그 데이터 내보내기
- 를 사용하여 Amazon S3로 로그 데이터 내보내기 AWS CLI
- 내보내기 작업 설명(CLI)
- 내보내기 작업 취소(CLI)

개념

내보내기를 시작하기 전에 다음 개념을 익힙니다.

log_group_name

내보내기 작업과 연관된 로그 그룹의 이름. 이 로그 그룹의 로그 데이터는 지정된 S3 버킷으로 내보내집니다.

(타임스탬프)부터

1970년 1월 1일 00:00:00 UTC 이후 경과된 시간(밀리초)으로 표현되는 필수 타임스탬프. 이 시간에 또는 그 이후에 수집된 로그 그룹의 모든 로그 이벤트가 내보내기됩니다.

(타임스탬프)까지

1970년 1월 1일 00:00:00 UTC 이후 경과된 시간(밀리초)으로 표현되는 필수 타임스탬프. 이 시간이 내보내기되기 전에 수집된 로그 그룹의 모든 로그 이벤트.

대상 버킷

내보내기 작업과 연관된 S3 버킷의 이름. 이 버킷은 지정된 로그 그룹에서 로그 데이터를 내보내는 데 사용됩니다.

대상 접두사

내보낸 모든 객체에 대한 Amazon S3 키 접두사로 사용되는 선택적인 속성. 이는 버킷에 폴더 같은 조직을 생성하는 데 도움이 됩니다.

콘솔을 사용하여 Amazon S3로 로그 데이터 내보내기

다음 예제에서는 Amazon CloudWatch 콘솔을 사용하여 이름이 `amzn-s3-demo-bucket`인 Amazon S3 버킷으로 이름이 `my-log-group`인 Amazon CloudWatch Logs 로그 그룹의 모든 데이터를 내보냅니다.

SSE-KMS로 암호화된 S3 버킷으로 로그 데이터 내보내기가 지원됩니다. DSSE-KMS로 암호화된 버킷으로의 내보내기는 지원되지 않습니다.

내보내기 설정 방법에 대한 세부 정보는 내보내려는 Amazon S3 버킷이 내보내는 로그와 동일한 계정에 있는지, 또는 다른 계정에 있는지에 따라 다릅니다.

주제

- 동일 계정 내보내기(콘솔)
- 교차 계정 내보내기(콘솔)

동일 계정 내보내기(콘솔)

Amazon S3 버킷이 내보내는 로그와 동일한 계정에 있는 경우 이 섹션의 지침을 사용합니다.

주제

- Amazon S3 버킷 생성(콘솔)
- 액세스 권한 설정(콘솔)
- Amazon S3 버킷에 대한 권한 설정(콘솔)
- (선택 사항) SSE-KMS로 암호화된 대상 Amazon S3 버킷으로 내보내기(콘솔)
- 내보내기 작업 생성(콘솔)

Amazon S3 버킷 생성(콘솔)

CloudWatch Logs를 위해 특별히 생성한 버킷을 사용하는 것이 좋습니다. 그러나 기존 버킷을 사용하고 싶으면 2단계로 건너뛸 수 있습니다.

> **Note**
>
> Amazon S3 버킷은 내보내려는 로그 데이터와 동일한 리전에 있어야 합니다. CloudWatch Logs는 다른 리전에 있는 Amazon S3 버킷으로 데이터 내보내기를 지원하지 않습니다.

Amazon S3 버킷을 생성하려면

1. Amazon S3 콘솔(https://console.aws.amazon.com/s3/)을 엽니다.
2. 필요한 경우 리전을 변경합니다. 탐색 모음에서 CloudWatch Logs가 상주하는 리전을 선택합니다.
3. 버킷 생성을 선택합니다.
4. 버킷 이름에 버킷의 이름을 입력합니다.
5. 리전에서 CloudWatch Logs 데이터가 상주하는 리전을 선택합니다.
6. 생성을 선택합니다.

액세스 권한 설정(콘솔)

내보내기 작업을 생성하려면 `AmazonS3ReadOnlyAccess` IAM 역할 및 다음 권한으로 로그인해야 합니다.

- `logs:CreateExportTask`
- `logs:CancelExportTask`
- `logs:DescribeExportTasks`
- `logs:DescribeLogStreams`
- `logs:DescribeLogGroups`

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.

- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

Amazon S3 버킷에 대한 권한 설정(콘솔)

기본적으로 모든 Amazon S3 버킷 및 객체는 프라이빗입니다. 버킷을 AWS 계정 생성한 리소스 소유자만 버킷과 버킷에 포함된 모든 객체에 액세스할 수 있습니다. 그러나 리소스 소유자는 액세스 정책을 작성하여 다른 리소스 및 사용자에게 액세스 권한을 부여할 수 있습니다.

정책을 설정할 때 의도한 로그 스트림만 버킷으로 내보내도록 임의로 생성되는 문자열을 버킷의 접두사로 포함할 것을 권장합니다.

> **Important**
>
> Amazon S3 버킷으로 더 안전하게 내보내려면 로그 데이터를 S3 버킷으로 내보낼 수 있는 소스 계정 목록을 지정해야 합니다.
> 다음 예제에서 `aws:SourceAccount` 키의 계정 ID 목록은 사용자가 로그 데이터를 Amazon S3 버킷으로 내보낼 수 있는 계정입니다. `aws:SourceArn` 키는 작업이 수행되는 리소스가 됩니다. 이를 특정 로그 그룹으로 제한하거나 이 예와 같이 와일드카드를 사용할 수 있습니다.
> 동일한 계정 내에서 내보내기를 허용하려면 S3 버킷이 생성된 계정의 계정 ID도 포함하는 것이 좋습니다.

Amazon S3 버킷에 대한 권한을 설정하려면

1. Amazon S3 콘솔에서 생성한 버킷을 선택합니다.
2. 권한, 버킷 정책을 선택합니다.

3. 버킷 정책 편집기에 다음 정책을 추가합니다. `amzn-s3-demo-bucket`을 S3 버킷의 이름으로 변경합니다. 보안 주체에 올바른 리전 엔드포인트(예: `us-west-1`)를 지정해야 합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
          "Sid": "AllowCloudWatchLogsGetBucketAcl",
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
          "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
          }
      },
      {
          "Sid": "AllowCloudWatchLogsPutObject",
          "Action": "s3:PutObject",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
          "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
```

```
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
          }
      }
    ]
}
```

4. 저장을 선택하여 버킷에서 액세스 정책으로 방금 추가한 정책을 설정합니다. 이 정책은 CloudWatch Logs가 Amazon S3 버킷으로 로그 데이터를 내보낼 수 있도록 합니다. 버킷 소유자는 내보낸 모든 개체에 대해 모든 권한을 가집니다.

> **Warning**
>
> 기존 버킷에 이미 하나 이상의 정책이 연결된 경우 CloudWatch Logs 액세스용 명령문을 해당 정책이나 정책들에 추가합니다. 발생한 권한 집합이 버킷에 액세스하는 사용자에게 적절한지를 여부를 평가하는 것이 좋습니다.

(선택 사항) SSE-KMS로 암호화된 대상 Amazon S3 버킷으로 내보내기(콘솔)

이 단계는 서버 측 암호화를 사용하는 Amazon S3 버킷으로 내보내는 경우에만 필요합니다 AWS KMS keys. 이 암호화를 SSE-KMS라고 합니다.

SSE-KMS로 암호화된 버킷으로 내보내려면

1. https://console.aws.amazon.com/kms AWS KMS 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.

   키 생성을 선택합니다.
4. 키 유형에 대칭을 선택합니다.
5. 키 사용에서 암호화 및 해독을 선택한 다음, 다음을 선택합니다.

6. Add labels(레이블 추가)에서 키의 별칭을 입력하고 필요에 따라 설명이나 태그를 추가합니다. 그런 다음 다음을 선택합니다.
7. 키 관리자에서 이 키를 관리할 수 있는 사용자를 선택한 후 다음을 선택합니다.
8. 키 사용 권한 정의에서 변경하지 않고 다음을 선택합니다.
9. 설정을 검토한 다음, 완료를 선택합니다.
10. 고객 관리형 키 페이지로 돌아가서 방금 생성한 키의 이름을 선택합니다.
11. 키 정책 탭에서 정책 보기로 전환을 선택합니다.
12. 키 정책 섹션에서 편집을 선택합니다.
13. 키 정책 문 목록에 다음 문을 추가합니다. 그런 다음 *##*을 로그의 리전으로 바꾸고 *account-ARN*을 KMS 키를 소유한 계정의 ARN으로 바꿉니다.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
```

```
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

14. 변경 사항 저장을 선택합니다.
15. Amazon S3 콘솔(https://console.aws.amazon.com/s3/)을 엽니다.
16. S3 버킷 생성(CLI)에서 생성한 버킷을 찾아 버킷 이름을 선택합니다.
17. 속성 탭을 선택합니다. 기본 암호화에서 편집을 선택합니다.
18. 서버 측 암호화에서 활성화를 선택합니다.
19. 암호화 유형에서 AWS Key Management Service 키(SSE-KMS)를 선택합니다.
20. AWS KMS 키에서 선택을 선택하고 생성한 키를 찾습니다.
21. 버킷 키에서 활성화를 선택합니다.
22. 변경 사항 저장을 선택합니다.

내보내기 작업 생성(콘솔)

이 절차에서는 로그 그룹에서 로그를 내보내기 위한 내보내기 작업을 생성합니다.

CloudWatch 콘솔을 사용하여 Amazon S3 로 데이터를 내보내려면

1. 액세스 권한 설정(콘솔)에 설명된 대로 충분한 권한으로 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 탐색 창에서 로그 그룹을 선택합니다.
4. 로그 그룹 화면에서 로그 그룹의 이름을 선택합니다.
5. 작업에서 Amazon S3로 데이터 내보내기를 선택합니다.
6. Amazon S3로 데이터 내보내기 화면의 데이터 내보내기 정의에서 시작 및 끝을 사용하여 내보낼 데이터에 대한 시간 범위를 설정합니다.
7. 로그 그룹에 여러 개의 로그 스트림이 있는 경우에는 로그 스트림 접두사를 제공하여 로그 그룹 데이터를 특정 스트림으로 제한할 수 있습니다. 고급을 선택하고 스트림 접두사에 로그 스트림 접두사를 입력합니다.
8. S3 버킷 선택에서 S3 버킷과 연관된 계정을 선택합니다.

9. S3 bucket name(S3 버킷 이름)에서 Amazon S3 버킷을 선택합니다.
10. S3 Bucket prefix(S3 버킷 접두사)에 버킷 정책에서 지정한 임의로 생성된 문자열을 입력합니다.
11. 내보내기를 선택하여 Amazon S3로 로그 데이터를 내보냅니다.
12. Amazon S3로 내보낸 로그 데이터의 상태를 확인하려면 작업을 선택한 다음, Amazon S3에 대한 모든 내보내기 보기를 선택합니다.

교차 계정 내보내기(콘솔)

Amazon S3 버킷이 내보내는 로그와 다른 계정에 있는 경우 이 섹션의 지침을 사용합니다.

주제

- 교차 계정 내보내기를 위한 Amazon S3 버킷 생성(콘솔)
- 교차 계정 내보내기에 대한 액세스 권한 설정(콘솔)
- 교차 계정 내보내기를 위한 S3 버킷에 대한 권한 설정(콘솔)
- (선택 사항) 교차 계정 내보내기를 위해 SSE-KMS로 암호화된 대상 Amazon S3 버킷으로 내보내기(콘솔)
- 교차 계정 내보내기를 위한 내보내기 작업 생성(콘솔)

교차 계정 내보내기를 위한 Amazon S3 버킷 생성(콘솔)

CloudWatch Logs를 위해 특별히 생성한 버킷을 사용하는 것이 좋습니다. 그러나 기존 버킷을 사용하려면이 절차를 건너뛸 수 있습니다.

> **Note**
>
> Amazon S3 버킷은 내보내려는 로그 데이터와 동일한 리전에 있어야 합니다. CloudWatch Logs는 다른 리전에 있는 Amazon S3 버킷으로 데이터 내보내기를 지원하지 않습니다.

Amazon S3 버킷을 생성하려면

1. Amazon S3 콘솔(https://console.aws.amazon.com/s3/)을 엽니다.
2. 필요한 경우 리전을 변경합니다. 탐색 모음에서 CloudWatch Logs가 상주하는 리전을 선택합니다.
3. 버킷 생성을 선택합니다.
4. 버킷 이름에 버킷의 이름을 입력합니다.

5. 리전에서 CloudWatch Logs 데이터가 상주하는 리전을 선택합니다.
6. 생성을 선택합니다.

교차 계정 내보내기에 대한 액세스 권한 설정(콘솔)

먼저 CloudWatch Logs가 대상 계정에서 대상 Amazon S3 버킷에 대한 `s3:PutObject` 작업을 수행할 수 있도록 하려면 새 IAM 정책을 생성해야 합니다.

`s3:PutObject` 작업과 함께 정책에 포함된 추가 작업은 대상 버킷이 AWS KMS 암호화를 사용하는지 또는 S3 객체 소유권 설정을 사용하여 활성화된 ACLs이 있는지에 따라 달라집니다.

- KMS 암호화를 사용하는 경우 키 리소스에 대한 `kms:GenerateDataKey` 및 `kms:Decrypt` 작업을 추가합니다.
- 버킷에서 ACL 활성화된 경우 버킷 리소스에 대한 `s3:PutObjectAcl` 작업을 추가합니다.

다음 정책에서 대상 S3 버킷의 이름으로 `amzn-s3-demo-bucket`를 변경합니다.

Amazon S3 버킷으로 로그를 내보내는 IAM 정책 생성:

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택합니다.
4. 정책 편집기 섹션에서 JSON을 선택합니다.
5. 대상 버킷이 AWS KMS 암호화를 사용하지 않는 경우 다음 정책을 편집기에 붙여넣습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

대상 버킷이 AWS KMS 암호화를 사용하는 경우 다음 정책을 편집기에 붙여 넣습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ]
}
```

대상 버킷에서 ACLs 활성화된 경우 위의 정책에서 s3:PutObjectAcl을 s3:PutObject 작업 블록에 추가합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:PutObject",
               "s3:PutObjectAcl"
            ],
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

6. 다음(Next)을 선택합니다.
7. 정책 이름을 입력합니다. 이 이름을 사용하여 정책을 IAM 역할에 연결합니다.
8. 정책 생성을 선택하여 새 정책을 저장합니다.

내보내기 작업을 생성하려면 AmazonS3ReadOnlyAccess 관리형 정책이 연결된 IAM 역할, 위에서 생성한 IAM 정책 및 다음 권한으로 로그인해야 합니다.

- `logs:CreateExportTask`
- `logs:CancelExportTask`
- `logs:DescribeExportTasks`
- `logs:DescribeLogStreams`
- `logs:DescribeLogGroups`

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

교차 계정 내보내기를 위한 S3 버킷에 대한 권한 설정(콘솔)

기본적으로 모든 S3 버킷 및 객체는 비공개입니다. 버킷을 AWS 계정 생성한 리소스 소유자만 버킷과 버킷에 포함된 모든 객체에 액세스할 수 있습니다. 그러나 리소스 소유자는 액세스 정책을 작성하여 다른 리소스 및 사용자에게 액세스 권한을 부여할 수 있습니다.

정책을 설정할 때 의도한 로그 스트림만 버킷으로 내보내도록 임의로 생성되는 문자열을 버킷의 접두사로 포함할 것을 권장합니다.

> **Important**
>
> S3 버킷을 더 안전하게 내보내려면 이제 로그 데이터를 S3 버킷으로 내보낼 수 있는 소스 계정 목록을 지정해야 합니다.
> 다음 예제에서 `aws:SourceAccount` 키의 계정 ID 목록은 사용자가 로그 데이터를 S3 버킷으로 내보낼 수 있는 계정입니다. `aws:SourceArn` 키는 작업이 수행되는 리소스가 됩니다. 이를 특정 로그 그룹으로 제한하거나 이 예와 같이 와일드카드를 사용할 수 있습니다.
> 동일한 계정 내에서 내보내기를 허용하려면 S3 버킷이 생성된 계정의 계정 ID도 포함하는 것이 좋습니다.

Amazon S3 버킷에 대한 권한을 설정하려면

1. Amazon S3 콘솔에서 생성한 버킷을 선택합니다.
2. 권한, 버킷 정책을 선택합니다.
3. 버킷 정책 편집기에 다음 정책을 추가합니다. `amzn-s3-demo-bucket`을 S3 버킷의 이름으로 변경합니다. 보안 주체에 올바른 리전 엔드포인트(예: `us-east-1`)를 지정해야 합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
          "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
          "Condition": {
```

```
            "StringEquals": {
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
        }
    },
    {
        "Action": "s3:PutObject",
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
        "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/role_name"
        },
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Condition": {
            "StringEquals": {
```

```
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
        }
    }
  ]
}
```

4. 저장을 선택하여 버킷에서 액세스 정책으로 방금 추가한 정책을 설정합니다. 이 정책은 CloudWatch Logs가 S3 버킷으로 로그 데이터를 내보낼 수 있도록 합니다. 버킷 소유자는 내보낸 모든 개체에 대해 모든 권한을 가집니다.

> **Warning**
>
> 기존 버킷에 이미 하나 이상의 정책이 연결된 경우 CloudWatch Logs 액세스용 명령문을 해당 정책이나 정책들에 추가합니다. 발생한 권한 집합이 버킷에 액세스하는 사용자에게 적절한지를 여부를 평가하는 것이 좋습니다.

(선택 사항) 교차 계정 내보내기를 위해 SSE-KMS로 암호화된 대상 Amazon S3 버킷으로 내보내기(콘솔)

이 절차는 서버 측 암호화를 사용하는 S3 버킷으로 내보내는 경우에만 필요합니다 AWS KMS keys. 이 암호화를 SSE-KMS라고 합니다.

SSE-KMS로 암호화된 버킷으로 내보내려면

1. https://console.aws.amazon.com/kms AWS KMS 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.

   키 생성을 선택합니다.
4. 키 유형에 대칭을 선택합니다.
5. 키 사용에서 암호화 및 해독을 선택한 다음, 다음을 선택합니다.
6. Add labels(레이블 추가)에서 키의 별칭을 입력하고 필요에 따라 설명이나 태그를 추가합니다. 그런 다음 다음을 선택합니다.
7. 키 관리자에서 이 키를 관리할 수 있는 사용자를 선택한 후 다음을 선택합니다.
8. 키 사용 권한 정의에서 변경하지 않고 다음을 선택합니다.

9. 설정을 검토한 다음, 완료를 선택합니다.
10. 고객 관리형 키 페이지로 돌아가서 방금 생성한 키의 이름을 선택합니다.
11. 키 정책 탭에서 정책 보기로 전환을 선택합니다.
12. 키 정책 섹션에서 편집을 선택합니다.
13. 키 정책 문 목록에 다음 문을 추가합니다. 이 경우 *us-east-1*을 로그의 리전으로, *account-ARN*을 KMS 키를 소유한 계정의 ARN으로, *123456789012*을 KMS 키를 소유한 계정 번호로,*key_id*를 kms-key Id로, *role_name*을 내보내기 작업을 생성하는 데 사용되는 역할로 바꿉니다.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id"
        }
    ]
}
```

14. 변경 사항 저장을 선택합니다.
15. Amazon S3 콘솔(https://console.aws.amazon.com/s3/)을 엽니다.
16. S3 버킷 생성(CLI)에서 생성한 버킷을 찾아 버킷 이름을 선택합니다.
17. 속성 탭을 선택합니다. 기본 암호화에서 편집을 선택합니다.
18. 서버 측 암호화에서 활성화를 선택합니다.
19. 암호화 유형에서 AWS Key Management Service 키(SSE-KMS)를 선택합니다.
20. AWS KMS 키에서 선택을 선택하고 생성한 키를 찾습니다.
21. 버킷 키에서 활성화를 선택합니다.
22. 변경 사항 저장을 선택합니다.

교차 계정 내보내기를 위한 내보내기 작업 생성(콘솔)

이 절차에서는 로그 그룹에서 로그를 내보내기 위한 내보내기 작업을 생성합니다.

CloudWatch 콘솔을 사용하여 Amazon S3 로 데이터를 내보내려면

1. 액세스 권한 설정(콘솔)에 설명된 대로 충분한 권한으로 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 탐색 창에서 로그 그룹을 선택합니다.
4. 로그 그룹 화면에서 로그 그룹의 이름을 선택합니다.

5. 작업에서 Amazon S3로 데이터 내보내기를 선택합니다.
6. Amazon S3로 데이터 내보내기 화면의 데이터 내보내기 정의에서 시작 및 끝을 사용하여 내보낼 데이터에 대한 시간 범위를 설정합니다.
7. 로그 그룹에 여러 개의 로그 스트림이 있는 경우에는 로그 스트림 접두사를 제공하여 로그 그룹 데이터를 특정 스트림으로 제한할 수 있습니다. 고급을 선택하고 스트림 접두사에 로그 스트림 접두사를 입력합니다.
8. Choose S3 bucket(S3 버킷 선택)에서 S3 버킷과 연관된 계정을 선택합니다.
9. S3 버킷 이름에서 S3 버킷을 선택합니다.
10. S3 버킷 접두사에 버킷 정책에서 지정한 임의로 생성된 문자열을 입력합니다.
11. 내보내기를 선택하여 Amazon S3로 로그 데이터를 내보냅니다.
12. Amazon S3로 내보낸 로그 데이터의 상태를 확인하려면 작업을 선택한 다음, Amazon S3에 대한 모든 내보내기 보기를 선택합니다.

를 사용하여 Amazon S3로 로그 데이터 내보내기 AWS CLI

다음 예제에서는 내보내기 작업을 사용하여 이름이 `my-log-group`인 CloudWatch Logs 로그 그룹의 모든 데이터를 이름이 `amzn-s3-demo-bucket`인 Amazon S3 버킷으로 내보낼 수 있습니다. 이 예제에서는 이름이 `my-log-group`인 로그 그룹을 이미 생성했다고 가정합니다.

에서 암호화된 S3 버킷으로 로그 데이터 내보내기가 지원 AWS KMS 됩니다. DSSE-KMS로 암호화된 버킷으로의 내보내기는 지원되지 않습니다.

내보내기 설정 방법에 대한 세부 정보는 내보내려는 Amazon S3 버킷이 내보내는 로그와 동일한 계정에 있는지, 또는 다른 계정에 있는지에 따라 다릅니다.

주제

- 동일 계정 내보내기(CLI)
- 교차 계정 내보내기(CLI)

동일 계정 내보내기(CLI)

Amazon S3 버킷이 내보내는 로그와 동일한 계정에 있는 경우 이 섹션의 지침을 사용합니다.

주제

- S3 버킷 생성(CLI)

- 액세스 권한 설정(CLI)
- S3 버킷에 대한 권한 설정(CLI)
- (선택 사항) SSE-KMS(CLI)로 암호화된 대상 Amazon S3 버킷으로 내보내기
- 내보내기 작업 생성(CLI)

S3 버킷 생성(CLI)

CloudWatch Logs를 위해 특별히 생성한 버킷을 사용하는 것이 좋습니다. 그러나 기존 버킷을 사용하려면이 절차를 건너뛸 수 있습니다.

> **Note**
>
> S3 버킷은 내보내는 로그 데이터와 동일한 리전에 있어야 합니다. CloudWatch Logs는 다른 리전에 있는 S3 버킷으로 데이터 내보내기를 지원하지 않습니다.

를 사용하여 S3 버킷을 생성하려면 AWS CLI

명령 프롬프트에서 다음 create-bucket 명령을 입력합니다. 여기서 `LocationConstraint`는 로그 데이터를 내보내는 리전입니다.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-2
```

다음은 예제 출력입니다.

```
{
    "Location": "/amzn-s3-demo-bucket"
}
```

액세스 권한 설정(CLI)

나중에 내보내기 작업을 생성하려면 `AmazonS3ReadOnlyAccess` IAM 역할 및 다음 권한으로 로그인해야 합니다.

- `logs:CreateExportTask`
- `logs:CancelExportTask`

- `logs:DescribeExportTasks`
- `logs:DescribeLogStreams`
- `logs:DescribeLogGroups`

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.

- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

S3 버킷에 대한 권한 설정(CLI)

기본적으로 모든 S3 버킷 및 객체는 비공개입니다. 리소스 소유자(버킷을 생성한 계정)만 해당 버킷과 버킷에 포함된 객체에 액세스할 수 있습니다. 그러나 리소스 소유자는 액세스 정책을 작성하여 다른 리소스 및 사용자에게 액세스 권한을 부여할 수 있습니다.

> **Important**
>
> S3 버킷을 더 안전하게 내보내려면 이제 로그 데이터를 S3 버킷으로 내보낼 수 있는 소스 계정 목록을 지정해야 합니다.
> 다음 예제에서 `aws:SourceAccount` 키의 계정 ID 목록은 사용자가 로그 데이터를 S3 버킷으로 내보낼 수 있는 계정입니다. `aws:SourceArn` 키는 작업이 수행되는 리소스가 됩니다. 이를 특정 로그 그룹으로 제한하거나 이 예와 같이 와일드카드를 사용할 수 있습니다.
> 동일한 계정 내에서 내보내기를 허용하려면 S3 버킷이 생성된 계정의 계정 ID도 포함하는 것이 좋습니다.

S3 버킷에 대한 권한을 설정하려면

1. 이름이 policy.json인 파일을 생성하고 다음 액세스 정책을 추가합니다. amzn-s3-demo-bucket을 S3 버킷의 이름으로 변경하고 Principal을 로그 데이터(예: us-east-1)를 내보내는 리전의 엔드포인트로 변경합니다. 텍스트 편집기를 사용하여 이 정책 파일을 생성합니다. IAM 콘솔을 사용하지 마세요.

   JSON

```
{
"Version":"2012-10-17",
"Statement": [
  {
      "Sid": "AllowGetBucketAcl",
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
            "aws:SourceAccount": [
                "123456789012",
                "111122223333"
            ]
        },
        "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:123456789012:log-group:*",
                    "arn:aws:logs:us-east-1:111122223333:log-group:*"
                ]
        }
      }
  },
  {
      "Sid": "AllowPutObject",
      "Action": "s3:PutObject",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
```

```
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
        }
    }
  ]
}
```

2. put-bucket-policy 명령을 사용하여 버킷에 액세스 정책으로 방금 추가한 정책을 설정합니다. 이 정책은 CloudWatch Logs가 S3 버킷으로 로그 데이터를 내보낼 수 있도록 합니다. 버킷 소유자는 내보낸 모든 개체에 대해 모든 권한을 갖게 됩니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json
```

> **Warning**
>
> 기존 버킷에 이미 하나 이상의 정책이 연결된 경우 CloudWatch Logs 액세스용 명령문을 해당 정책이나 정책들에 추가합니다. 발생한 권한 집합이 버킷에 액세스하는 사용자에게 적절한지를 여부를 평가하는 것이 좋습니다.

(선택 사항) SSE-KMS(CLI)로 암호화된 대상 Amazon S3 버킷으로 내보내기

이 절차는 서버 측 암호화를 사용하는 S3 버킷으로 내보내는 경우에만 필요합니다 AWS KMS keys. 이 암호화를 SSE-KMS라고 합니다.

SSE-KMS로 암호화된 버킷으로 내보내려면

1. 텍스트 편집기를 사용하여 이름이 `key_policy.json`인 파일을 생성하고 다음 액세스 정책을 추가합니다. 정책을 추가할 때 다음과 같이 변경합니다.

- *Region*을 로그의 리전으로 바꿉니다.
- *account-ARN*을 KMS 키를 소유하는 계정의 ARN으로 바꿉니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

2. 다음 명령을 입력합니다.

```
aws kms create-key --policy file://key_policy.json
```

다음은 이 명령의 출력 예시입니다.

```
{
    "KeyMetadata": {
        "AWSAccountId": "account_id",
        "KeyId": "key_id",
        "Arn": "arn:aws:kms:us-east-2:account-ARN:key/key_id",
        "CreationDate": "time",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
```

3. 텍스트 편집기를 사용하여 다음 내용을 포함하는 bucketencryption.json이라는 파일을 생성합니다.

```
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

4. 다음 명령을 입력하여 *amzn-s3-demo-bucket*을 로그를 내보내는 버킷의 이름으로 바꿉니다.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration file://bucketencryption.json
```

명령이 오류를 반환하지 않으면 프로세스가 성공한 것입니다.

내보내기 작업 생성(CLI)

다음 명령을 사용하여 내보내기 작업을 생성합니다. 생성하고 나면 내보낼 데이터의 크기에 따라 내보내기 작업에 몇 초부터 몇 시간까지 소요될 수 있습니다.

를 사용하여 Amazon S3로 데이터를 내보내려면 AWS CLI

1. 액세스 권한 설정(CLI)에 설명된 대로 충분한 권한으로 로그인합니다.
2. 명령 프롬프트에서 다음 create-export-task 명령을 사용하여 내보내기 작업을 생성합니다.

```
aws logs create-export-task --profile CWLExportUser --task-name "my-log-
group-09-10-2015" --log-group-name "my-log-group" --from 1441490400000 --
to 1441494000000 --destination "amzn-s3-demo-bucket" --destination-prefix "export-
task-output"
```

다음은 예제 출력입니다.

```
{
    "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

교차 계정 내보내기(CLI)

Amazon S3 버킷이 내보내는 로그와 다른 계정에 있는 경우 이 섹션의 지침을 사용합니다.

주제

- 교차 계정 내보내기(CLI)를 위한 S3 버킷 생성
- 교차 계정 내보내기(CLI)에 대한 액세스 권한 설정
- 교차 계정 내보내기(CLI)를 위한 S3 버킷에 대한 권한 설정
- (선택 사항) 교차 계정 내보내기(CLI)를 위해 SSE-KMS로 암호화된 대상 Amazon S3 버킷으로 내보내기

- 교차 계정 내보내기를 위한 내보내기 작업 생성(CLI)

교차 계정 내보내기(CLI)를 위한 S3 버킷 생성

CloudWatch Logs를 위해 특별히 생성한 버킷을 사용하는 것이 좋습니다. 그러나 기존 버킷을 사용하고 싶으면 2단계로 건너뛸 수 있습니다.

> **Note**
>
> S3 버킷은 내보내는 로그 데이터와 동일한 리전에 있어야 합니다. CloudWatch Logs는 다른 리전에 있는 S3 버킷으로 데이터 내보내기를 지원하지 않습니다.

를 사용하여 S3 버킷을 생성하려면 AWS CLI

명령 프롬프트에서 다음 create-bucket 명령을 입력합니다. 여기서 `LocationConstraint`는 로그 데이터를 내보내는 리전입니다.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-2
```

다음은 예제 출력입니다.

```
{
    "Location": "/amzn-s3-demo-bucket"
}
```

교차 계정 내보내기(CLI)에 대한 액세스 권한 설정

먼저 CloudWatch Logs가 대상 계정에서 대상 Amazon S3 버킷에 대한 `s3:PutObject` 작업을 수행할 수 있도록 하려면 새 IAM 정책을 생성해야 합니다.

`s3:PutObject` 작업과 함께 정책에 포함된 추가 작업은 대상 버킷이 AWS KMS 암호화를 사용하는지 아니면 S3 객체 소유권 설정을 사용하여 ACLs 활성화되어 있는지에 따라 달라집니다.

- KMS 암호화를 사용하는 경우 키 리소스에 대한 `kms:GenerateDataKey` 및 `kms:Decrypt` 작업을 추가합니다.
- 버킷에서 ACL 활성화된 경우 버킷 리소스에 대한 `s3:PutObjectAcl` 작업을 추가합니다.

다음 정책에서 대상 S3 버킷의 이름으로 amzn-s3-demo-bucket를 변경합니다.

생성하는 정책은 대상 버킷의 AWS KMS 암호화 사용 여부에 따라 달라집니다. AWS KMS 암호화를 사용하지 않는 경우 다음 내용이 포함된 정책을 생성합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

대상 버킷이 AWS KMS 암호화를 사용하는 경우 다음 내용이 포함된 정책을 생성합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

대상 버킷에서 ACLs 활성화된 경우 위의 정책에서 s3:PutObjectAcl을 s3:PutObject 작업 블록에 추가합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:PutObject",
               "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

내보내기 작업을 생성하려면 AmazonS3ReadOnlyAccess 관리형 정책이 연결된 IAM 역할, 위에서 생성한 IAM 정책 및 다음 권한으로 로그인해야 합니다.

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

교차 계정 내보내기(CLI)를 위한 S3 버킷에 대한 권한 설정

기본적으로 모든 S3 버킷 및 객체는 비공개입니다. 리소스 소유자(버킷을 생성한 계정)만 해당 버킷과 버킷에 포함된 객체에 액세스할 수 있습니다. 그러나 리소스 소유자는 액세스 정책을 작성하여 다른 리소스 및 사용자에게 액세스 권한을 부여할 수 있습니다.

> **Important**
>
> S3 버킷을 더 안전하게 내보내려면 이제 로그 데이터를 S3 버킷으로 내보낼 수 있는 소스 계정 목록을 지정해야 합니다.
> 다음 예제에서 `aws:SourceAccount` 키의 계정 ID 목록은 사용자가 로그 데이터를 S3 버킷으로 내보낼 수 있는 계정입니다. `aws:SourceArn` 키는 작업이 수행되는 리소스가 됩니다. 이를 특정 로그 그룹으로 제한하거나 이 예와 같이 와일드카드를 사용할 수 있습니다.
> 동일한 계정 내에서 내보내기를 허용하려면 S3 버킷이 생성된 계정의 계정 ID도 포함하는 것이 좋습니다.

S3 버킷에 대한 권한을 설정하려면

1. 라는 파일을 생성하고 대상 S3 버킷의 이름으로 변경`amzn-s3-demo-bucket`하여와 같은 로그 데이터를 내보내는 리전의 `Principal` 엔드포인트에 다음 액세스 정책을 `policy.json` 추가합니다`us-west-1`. 텍스트 편집기를 사용하여 이 정책 파일을 생성합니다. IAM 콘솔을 사용하지 마세요.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
            "Condition": {
              "StringEquals": {
                  "aws:SourceAccount": [
                  "123456789012",
                  "111122223333"
                  ]
              },
              "ArnLike": {
                      "aws:SourceArn": [
                      "arn:aws:logs:us-east-1:123456789012:log-group:*",
                      "arn:aws:logs:us-east-1:111122223333:log-group:*"
                       ]
              }
            }
        },
        {
            "Action": "s3:PutObject" ,
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control",
                  "aws:SourceAccount": [
                  "123456789012",
                  "111122223333"
                  ]
              },
              "ArnLike": {
                      "aws:SourceArn": [
                      "arn:aws:logs:us-east-1:123456789012:log-group:*",
                      "arn:aws:logs:us-east-1:111122223333:log-group:*"
                      ]
              }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/role_name"
            },
```

```
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::>amzn-s3-demo-bucket/*",
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control"
              }
            }
        }
    ]
}
```

2. put-bucket-policy 명령을 사용하여 버킷에 액세스 정책으로 방금 추가한 정책을 설정합니다. 이 정책은 CloudWatch Logs가 S3 버킷으로 로그 데이터를 내보낼 수 있도록 합니다. 버킷 소유자는 내보낸 모든 개체에 대해 모든 권한을 갖게 됩니다.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json
```

> **Warning**
>
> 기존 버킷에 이미 하나 이상의 정책이 연결된 경우 CloudWatch Logs 액세스용 명령문을 해당 정책이나 정책들에 추가합니다. 발생한 권한 집합이 버킷에 액세스하는 사용자에게 적절한지를 여부를 평가하는 것이 좋습니다.

(선택 사항) 교차 계정 내보내기(CLI)를 위해 SSE-KMS로 암호화된 대상 Amazon S3 버킷으로 내보내기

이 절차는 서버 측 암호화를 사용하는 S3 버킷으로 내보내는 경우에만 필요합니다 AWS KMS keys. 이 암호화를 SSE-KMS라고 합니다.

SSE-KMS로 암호화된 버킷으로 내보내려면

1. 텍스트 편집기를 사용하여 이름이 `key_policy.json`인 파일을 생성하고 다음 액세스 정책을 추가합니다. 정책을 추가할 때 다음과 같이 변경합니다.

   - *`us-east-1`*을 로그의 리전으로 바꿉니다.
   - *`account-ARN`*을 KMS 키를 소유하는 계정의 ARN으로 바꿉니다.
   - *`123456789012`*을 KMS 키를 소유한 계정 번호로 바꿉니다.

- kms-key ID가 있는 *key_id*입니다.
- *role_name*은 내보내기 작업을 생성하는 데 사용되는 역할을 포함합니다.

JSON

```
{
"Version":"2012-10-17",
"Statement": [
    {
        "Sid": "AllowCWLServicePrincipalUsage",
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
        },
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "*"
    },
    {
        "Sid": "EnableIAMUserPermissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "account-ARN"
        },
        "Action": [
            "kms:GetKeyPolicy*",
            "kms:PutKeyPolicy*",
            "kms:DescribeKey*",
            "kms:CreateAlias*",
            "kms:ScheduleKeyDeletion*",
            "kms:Decrypt"
        ],
        "Resource": "*"
    },
    {
        "Sid": "EnableIAMRolePermissions",
        "Effect": "Allow",
        "Principal": {
```

```
                "AWS": "arn:aws:iam::111122223333:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

2. 다음 명령을 입력합니다.

```
aws kms create-key --policy file://key_policy.json
```

다음은 이 명령의 출력 예시입니다.

```
{
    "KeyMetadata": {
        "AWSAccountId": "account_id",
        "KeyId": "key_id",
        "Arn": "arn:aws:kms:us-east-1:123456789012:key/key_id",
        "CreationDate": "time",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
```

3. 텍스트 편집기를 사용하여 다음 내용을 포함하는 bucketencryption.json이라는 파일을 생성합니다.

```
{
```

```
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

4. 다음 명령을 입력하여 *amzn-s3-demo-bucket*을 로그를 내보내는 버킷의 이름으로 바꿉니다.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration file://bucketencryption.json
```

명령이 오류를 반환하지 않으면 프로세스가 성공한 것입니다.

교차 계정 내보내기를 위한 내보내기 작업 생성(CLI)

다음 명령을 사용하여 내보내기 작업을 생성합니다. 생성하고 나면 내보낼 데이터의 크기에 따라 내보내기 작업에 몇 초부터 몇 시간까지 소요될 수 있습니다.

를 사용하여 Amazon S3로 데이터를 내보내려면 AWS CLI

1. 액세스 권한 설정(CLI)에 설명된 대로 충분한 권한으로 로그인합니다.
2. 명령 프롬프트에서 다음 create-export-task 명령을 사용하여 내보내기 작업을 생성합니다.

```
aws logs create-export-task --profile CWLExportUser --task-name "my-log-
group-09-10-2015" --log-group-name "my-log-group" --from 1441490400000 --
to 1441494000000 --destination "amzn-s3-demo-bucket" --destination-prefix "export-
task-output"
```

다음은 예제 출력입니다.

```
{
    "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

내보내기 작업 설명(CLI)

내보내기 작업을 생성하고 나면 작업의 현재 상태를 파악할 수 있습니다.

를 사용하여 내보내기 작업을 설명하려면 AWS CLI

명령 프롬프트에서 다음 describe-export-tasks 명령을 사용합니다.

```
aws logs --profile CWLExportUser describe-export-tasks --task-id
 "cda45419-90ea-4db5-9833-aade86253e66"
```

다음은 예제 출력입니다.

```
{
   "exportTasks": [
   {
      "destination": "my-exported-logs",
      "destinationPrefix": "export-task-output",
      "executionInfo": {
         "creationTime": 1441495400000
      },
      "from": 1441490400000,
      "logGroupName": "my-log-group",
      "status": {
         "code": "RUNNING",
         "message": "Started Successfully"
      },
      "taskId": "cda45419-90ea-4db5-9833-aade86253e66",
      "taskName": "my-log-group-09-10-2015",
      "tTo": 1441494000000
   }]
}
```

세 가지 방법으로 `describe-export-tasks` 명령을 사용할 수 있습니다.

- 필터가 없는 경우 - 생성 순서의 역순으로 모든 내보내기 작업을 나열합니다.
- 작업 ID 필터링 - 지정된 ID를 가진 내보내기 작업(존재하는 경우)을 나열합니다.
- 작업 상태 필터링 - 지정된 상태를 가진 내보내기 작업을 나열합니다.

예를 들어, 다음 명령을 사용하여 `FAILED` 상태를 필터링합니다.

```
aws logs --profile CWLExportUser describe-export-tasks --status-code "FAILED"
```

다음은 예제 출력입니다.

```
{
   "exportTasks": [
   {
      "destination": "amzn-s3-demo-bucket",
      "destinationPrefix": "export-task-output",
      "executionInfo": {
         "completionTime": 1441498600000
         "creationTime": 1441495400000
      },
      "from": 1441490400000,
      "logGroupName": "my-log-group",
      "status": {
         "code": "FAILED",
         "message": "FAILED"
      },
      "taskId": "cda45419-90ea-4db5-9833-aade86253e66",
      "taskName": "my-log-group-09-10-2015",
      "to": 1441494000000
   }]
}
```

내보내기 작업 취소(CLI)

PENDING 또는 RUNNING 상태인 경우 내보내기 작업을 취소할 수 있습니다.

를 사용하여 내보내기 작업을 취소하려면 AWS CLI

명령 프롬프트에서 다음 cancel-export-task 명령을 사용합니다.

```
aws logs --profile CWLExportUser cancel-export-task --task-id "cda45419-90ea-4db5-9833-aade86253e66"
```

describe-export-tasks 명령을 사용하여 작업이 성공적으로 취소되었는지 확인할 수 있습니다.

Amazon OpenSearch Service로 CloudWatch Logs 데이터 스트리밍

Amazon CloudWatch Logs에서 로그 그룹을 구성하여 데이터를 Amazon OpenSearch Service 클러스터로 거의 실시간으로 스트리밍할 수 있습니다. 자세한 내용은 구독을 통한 로그 데이터 실시간 처리 단원을 참조하십시오.

> **Note**
>
> OpenSearch Service로의 스트리밍은 Standard 로그 클래스의 로그 그룹에서만 지원됩니다. 로그 클래스에 대한 자세한 내용은 로그 클래스를 참조하세요.

스트리밍되는 로그 데이터의 양에 따라 함수 수준 동시성 제한을 설정하는 것이 좋습니다. 자세한 내용은 Lambda 함수 크기 조정을 참조하세요.

> **Note**
>
> 많은 양의 CloudWatch Logs 데이터를 OpenSearch Service로 스트리밍하면 사용 요금이 높아질 수 있으므로 AWS 결제 및 비용 관리 콘솔에서 예산을 생성하는 것이 좋습니다. 자세한 내용은 AWS 예산으로 비용 관리를 참조하세요.

이 섹션에서는 로그 그룹의 OpenSearch Service 구독 전에 완료해야 하는 사전 조건을 설명합니다. 로그 그룹의 OpenSearch Service 구독 방법도 설명합니다.

사전 조건

시작에 앞서 OpenSearch Service 도메인을 생성합니다. 도메인은 퍼블릭 액세스 또는 VPC 액세스 중 하나를 가질 수 있지만 도메인을 생성한 후에는 액세스 유형을 수정할 수 없습니다. 나중에 OpenSearch Service 도메인 설정을 검토하고 클러스터가 처리할 데이터 양에 따라 클러스터 구성을 수정하고 싶을 수 있습니다. 도메인을 생성하기 위한 지침은 OpenSearch Service 도메인 생성을 참조하세요.

OpenSearch Service에 대한 자세한 내용은 Amazon OpenSearch Service 개발자 가이드를 참조하세요.

OpenSearch Service에 대한 로그 그룹 구독

CloudWatch 콘솔을 사용하여 OpenSearch Service에 대한 로그 그룹을 구독할 수 있습니다.

OpenSearch Service에 대한 로그 그룹을 구독하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 로그 그룹의 이름을 선택합니다.
4. 작업, 구독 필터, Amazon OpenSearch Service 구독 필터 생성을 선택합니다.
5. 이 계정의 클러스터로 스트리밍하려고 하는지 아니면 다른 계정의 클러스터로 스트리밍하려고 하는지 선택합니다.
   - 이 계정을 선택한 경우 이전 단계에서 생성한 도메인을 선택합니다.
   - 다른 계정을 선택한 경우 도메인 ARN 및 엔드포인트를 제공합니다.
6. Lambda IAM 실행 역할에서 OpenSearch에 대한 직접 호출을 실행할 때 Lambda에서 사용해야 하는 IAM 역할을 선택합니다.

   선택된 IAM 역할은 다음 요구 사항을 충족해야 합니다.

   - 신뢰 관계를 맺고 있는 `lambda.amazonaws.com`이 있어야 합니다.
   - 다음 정책이 포함되어야 합니다.

     JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowOpenSearchStreamingAccess",
            "Action": [
                "es:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:es:us-
east-1:123456789012:domain/cloudwatch-logs/*"
        }
    ]
}
```

- 대상 OpenSearch Service 도메인에서 VPC 액세스를 사용하는 경우 역할에 AWSLambdaVPCAccessExecutionRole 정책이 연결되어 있어야 합니다. Amazon에서 관리하는 이 정책은 고객의 VPC에 Lambda 액세스 권한을 부여하여 Lambda가 VPC의 OpenSearch 엔드포인트에 기록할 수 있게 합니다.

7. 로그 형식에서 로그 형식을 선택합니다.
8. 구독 필터 패턴에 로그 이벤트에서 찾을 용어나 패턴을 입력합니다. 이렇게 하면 OpenSearch 클러스터로 원하는 데이터만 전송할 수 있습니다. 자세한 내용은 필터를 사용하여 로그 이벤트에서 지표 생성 단원을 참조하십시오.
9. (선택 사항) 테스트할 로그 데이터 선택에서 로그 스트림을 선택한 다음 패턴 테스트를 선택해서 검색 필터가 예상한 결과를 반환하고 있는지 확인합니다.
10. 스트리밍 시작을 선택합니다.

SDK를 사용한 CloudWatch Logs의 코드 예제 AWS SDKs

다음 코드 예제에서는 AWS 소프트웨어 개발 키트(SDK)와 함께 CloudWatch Logs를 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDKs를 사용한 CloudWatch Logs의 기본 예제
  - CloudWatch Logs 시작
  - AWS SDKs를 사용한 CloudWatch Logs 작업
    - AWS SDK와 AssociateKmsKey 함께 사용
    - AWS SDK와 CancelExportTask 함께 사용
    - AWS SDK와 CreateExportTask 함께 사용
    - AWS SDK 또는 CLI와 CreateLogGroup 함께 사용
    - AWS SDK 또는 CLI와 CreateLogStream 함께 사용
    - AWS SDK 또는 CLI와 DeleteLogGroup 함께 사용
    - AWS SDK와 DeleteSubscriptionFilter 함께 사용
    - AWS SDK와 DeleteSyslogConfiguration 함께 사용
    - AWS SDK와 DescribeExportTasks 함께 사용
    - AWS SDK 또는 CLI와 DescribeLogGroups 함께 사용
    - AWS SDK 또는 CLI와 DescribeLogStreams 함께 사용
    - AWS SDK와 DescribeSubscriptionFilters 함께 사용
    - AWS SDK 또는 CLI와 GetLogEvents 함께 사용
    - AWS SDK와 GetQueryResults 함께 사용
    - AWS SDK와 ListSyslogConfigurations 함께 사용

    - AWS SDK와 PutResourcePolicy 함께 사용
    - AWS SDK와 PutSubscriptionFilter 함께 사용
    - AWS SDK와 PutSyslogConfiguration 함께 사용
    - AWS SDK 또는 CLI와 StartLiveTail 함께 사용
    - AWS SDK와 StartQuery 함께 사용
- AWS SDKs를 사용한 CloudWatch Logs 시나리오
    - 컨테이너 서비스 연결 구성
    - 첫 번째 서버리스 함수 생성
    - AWS SDK를 사용하여 CloudWatch Logs syslog 수집 기본 사항 알아보기
    - CloudWatch Logs를 사용하여 대규모 쿼리 실행
    - 예약된 이벤트를 사용하여 Lambda 함수 간접 호출

AWS SDKs를 사용한 CloudWatch Logs의 기본 예제

다음 코드 예제는 AWS SDK로 Amazon CloudWatch Logs의 기본 기능을 사용하는 방법을 보여줍니다.

예제

- CloudWatch Logs 시작
- AWS SDKs를 사용한 CloudWatch Logs 작업
    - AWS SDK와 AssociateKmsKey 함께 사용
    - AWS SDK와 CancelExportTask 함께 사용
    - AWS SDK와 CreateExportTask 함께 사용
    - AWS SDK 또는 CLI와 CreateLogGroup 함께 사용
    - AWS SDK 또는 CLI와 CreateLogStream 함께 사용
    - AWS SDK 또는 CLI와 DeleteLogGroup 함께 사용
    - AWS SDK와 DeleteSubscriptionFilter 함께 사용
    - AWS SDK와 DeleteSyslogConfiguration 함께 사용
    - AWS SDK와 DescribeExportTasks 함께 사용
    - AWS SDK 또는 CLI와 DescribeLogGroups 함께 사용
    - AWS SDK 또는 CLI와 DescribeLogStreams 함께 사용

- AWS SDK와 DescribeSubscriptionFilters 함께 사용
- AWS SDK 또는 CLI와 GetLogEvents 함께 사용
- AWS SDK와 GetQueryResults 함께 사용
- AWS SDK와 ListSyslogConfigurations 함께 사용
- AWS SDK와 PutResourcePolicy 함께 사용
- AWS SDK와 PutSubscriptionFilter 함께 사용
- AWS SDK와 PutSyslogConfiguration 함께 사용
- AWS SDK 또는 CLI와 StartLiveTail 함께 사용
- AWS SDK와 StartQuery 함께 사용

CloudWatch Logs 시작

다음 코드 예제에서는 CloudWatch Logs 사용을 시작하는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def hello_cloudwatch_logs() -> None:
    """
    Verifies connectivity to CloudWatch Logs by calling DescribeLogGroups
    and displaying the first page of log groups with name, ARN, and
    creation time.
    """
    logs_client = boto3.client("logs")

    try:
        response = logs_client.describe_log_groups()
        log_groups = response.get("logGroups", list())

        if not log_groups:
            print("No log groups found in this Region for your account.")
```

```
        else:
            print(f"Found {len(log_groups)} log group(s):\n")
            for lg in log_groups:
                name = lg.get("logGroupName", "N/A")
                arn = lg.get("arn", "N/A")
                creation_ms = lg.get("creationTime", 0)
                creation_dt = datetime.fromtimestamp(
                    creation_ms / 1000, tz=timezone.utc
                )
                print(f"  Name: {name}")
                print(f"  ARN:  {arn}")
                print(f"  Created: {creation_dt.strftime('%Y-%m-%d %H:%M:%S
UTC')}")
                print()

    except ClientError as error:
        logger.error(
            "Failed to describe log groups: %s",
            error.response["Error"]["Message"],
        )
        raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 DescribeLogGroups를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용한 CloudWatch Logs 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 CloudWatch Logs 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이러한 발췌문은 CloudWatch Logs API를 직접적 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDKs를 사용한 CloudWatch Logs 시나리오에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon CloudWatch Logs API 참조를 참조하세요.

예제

- AWS SDK와 AssociateKmsKey 함께 사용
- AWS SDK와 CancelExportTask 함께 사용
- AWS SDK와 CreateExportTask 함께 사용
- AWS SDK 또는 CLI와 CreateLogGroup 함께 사용
- AWS SDK 또는 CLI와 CreateLogStream 함께 사용
- AWS SDK 또는 CLI와 DeleteLogGroup 함께 사용
- AWS SDK와 DeleteSubscriptionFilter 함께 사용
- AWS SDK와 DeleteSyslogConfiguration 함께 사용
- AWS SDK와 DescribeExportTasks 함께 사용
- AWS SDK 또는 CLI와 DescribeLogGroups 함께 사용
- AWS SDK 또는 CLI와 DescribeLogStreams 함께 사용
- AWS SDK와 DescribeSubscriptionFilters 함께 사용
- AWS SDK 또는 CLI와 GetLogEvents 함께 사용
- AWS SDK와 GetQueryResults 함께 사용
- AWS SDK와 ListSyslogConfigurations 함께 사용
- AWS SDK와 PutResourcePolicy 함께 사용
- AWS SDK와 PutSubscriptionFilter 함께 사용
- AWS SDK와 PutSyslogConfiguration 함께 사용
- AWS SDK 또는 CLI와 StartLiveTail 함께 사용
- AWS SDK와 StartQuery 함께 사용

AWS SDK와 **AssociateKmsKey** 함께 사용

다음 코드 예제는 `AssociateKmsKey`의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to associate an AWS Key Management Service (AWS KMS) key with
    /// an Amazon CloudWatch Logs log group.
    /// </summary>
    public class AssociateKmsKey
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
            string groupName = "cloudwatchlogs-example-loggroup";

            var request = new AssociateKmsKeyRequest
            {
                KmsKeyId = kmsKeyId,
                LogGroupName = groupName,
            };

            var response = await client.AssociateKmsKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine($"Successfully associated KMS key ID:
{kmsKeyId} with log group: {groupName}.");
            }
            else
            {
                Console.WriteLine("Could not make the association between:
{kmsKeyId} and {groupName}.");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 AssociateKmsKey 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **CancelExportTask** 함께 사용

다음 코드 예제는 CancelExportTask의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to cancel an Amazon CloudWatch Logs export task.
```

```
    /// </summary>
    public class CancelExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskId = "exampleTaskId";

            var request = new CancelExportTaskRequest
            {
                TaskId = taskId,
            };

            var response = await client.CancelExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{taskId} successfully canceled.");
            }
            else
            {
                Console.WriteLine($"{taskId} could not be canceled.");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CancelExportTask 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **CreateExportTask** 함께 사용

다음 코드 예제는 CreateExportTask의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Export Task to export the contents of the Amazon
    /// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
    /// bucket.
    /// </summary>
    public class CreateExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskName = "export-task-example";
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string destination = "amzn-s3-demo-bucket";
            var fromTime = 1437584472382;
            var toTime = 1437584472833;

            var request = new CreateExportTaskRequest
            {
                From = fromTime,
                To = toTime,
                TaskName = taskName,
                LogGroupName = logGroupName,
```

```
                Destination = destination,
            };

            var response = await client.CreateExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The task, {taskName} with ID: " +
                                  $"{response.TaskId} has been created
successfully.");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateExportTask를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreateLogGroup** 함께 사용

다음 코드 예제는 CreateLogGroup의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 컨테이너 서비스 연결 구성
- syslog 수집 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Amazon CloudWatch Logs log group.
    /// </summary>
    public class CreateLogGroup
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string logGroupName = "cloudwatchlogs-example-loggroup";

            var request = new CreateLogGroupRequest
            {
                LogGroupName = logGroupName,
            };

            var response = await client.CreateLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create log group.");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateLogGroup을 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 my-logs인 로그 그룹을 생성합니다.

```
aws logs create-log-group --log-group-name my-logs
```

- API 세부 정보는 AWS CLI 명령 참조의 CreateLogGroup를 참조하세요.

JavaScript

SDK for JavaScript(v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { CreateLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new CreateLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateLogGroup 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreateLogStream** 함께 사용

다음 코드 예제는 CreateLogStream의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
/// log group.
/// </summary>
public class CreateLogStream
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string logStreamName = "cloudwatchlogs-example-logstream";

        var request = new CreateLogStreamRequest
        {
            LogGroupName = logGroupName,
```

```
                LogStreamName = logStreamName,
            };

            var response = await client.CreateLogStreamAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{logStreamName} successfully created for
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create stream.");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateLogStream를 참조하세요.

CLI

AWS CLI

다음 명령은 my-logs 로그 그룹에서 이름이 20150601인 로그 스트림을 생성합니다.

```
aws logs create-log-stream --log-group-name my-logs --log-stream-name 20150601
```

- API 세부 정보는 AWS CLI 명령 참조의 CreateLogStream을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteLogGroup** 함께 사용

다음 코드 예제는 DeleteLogGroup의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 컨테이너 서비스 연결 구성
- 첫 번째 서버리스 함수 생성
- syslog 수집 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Uses the Amazon CloudWatch Logs Service to delete an existing
/// CloudWatch Logs log group.
/// </summary>
public class DeleteLogGroup
{
    public static async Task Main()
    {
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new DeleteLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.DeleteLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted CloudWatch log group,
{logGroupName}.");
```

```
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteLogGroup을 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 my-logs인 로그 그룹을 삭제합니다.

```
aws logs delete-log-group --log-group-name my-logs
```

- API 세부 정보는 AWS CLI 명령 참조의 DeleteLogGroup을 참조하세요.

JavaScript

SDK for JavaScript(v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { DeleteLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
```

```
  }
};

export default run();
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteLogGroup 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **DeleteSubscriptionFilter** 함께 사용

다음 코드 예제는 DeleteSubscriptionFilter의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

구독 필터를 삭제합니다.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
```

```
        request.SetFilterName(filter_name);
        request.SetLogGroupName(log_group);

        auto outcome = cwl.DeleteSubscriptionFilter(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        } else {
            std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteSubscriptionFilter를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                  <filter> <logGroup>

                Where:
                  filter - The name of the subscription filter (for example,
 MyFilter).
                  logGroup - The name of the log group. (for example, testgroup).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
 String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
 DeleteSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .logGroupName(logGroup)
                    .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
 filter %s", filter);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteSubscriptionFilter를 참조하세요.

JavaScript

SDK for JavaScript(v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { DeleteSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteSubscriptionFilterCommand({
    // The name of the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteSubscriptionFilter를 참조하세요.

SDK for JavaScript(v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cwl.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteSubscriptionFilter를 참조하세요.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteSubscriptionFilter를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **DeleteSyslogConfiguration** 함께 사용

다음 코드 예제는 DeleteSyslogConfiguration의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- syslog 수집 기본 사항 알아보기

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    def delete_syslog_configuration(
        self, log_group_identifier: str, vpc_endpoint_id: str
    ) -> None:
        """
        Deletes a syslog configuration for a log group. After deletion, syslog
        data is no longer ingested through the specified VPC endpoint.

        :param log_group_identifier: The name or ARN of the log group.
        :param vpc_endpoint_id: The ID of the VPC endpoint associated with the
            syslog configuration.
        :raises ClientError: If the syslog configuration does not exist (in which
            case the error is logged but not re-raised during cleanup).
        """
        try:
            self.logs_client.delete_syslog_configuration(
                logGroupIdentifier=log_group_identifier,
                vpcEndpointId=vpc_endpoint_id,
            )
            logger.info(
                "Deleted syslog configuration for log group '%s' and VPC endpoint
'%s'.",
                log_group_identifier,
                vpc_endpoint_id,
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info(
                    "Syslog configuration does not exist or was already deleted:
%s",
                    error.response["Error"]["Message"],
```

```
            )
        else:
            logger.error(
                "Failed to delete syslog configuration: %s",
                error.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 DeleteSyslogConfiguration을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **DescribeExportTasks** 함께 사용

다음 코드 예제는 DescribeExportTasks의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks.
/// </summary>
```

```
    public class DescribeExportTasks
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            var request = new DescribeExportTasksRequest
            {
                Limit = 5,
            };

            var response = new DescribeExportTasksResponse();

            do
            {
                response = await client.DescribeExportTasksAsync(request);
                response.ExportTasks.ForEach(t =>
                {
                    Console.WriteLine($"{t.TaskName} with ID: {t.TaskId} has
status: {t.Status}");
                });
            }
            while (response.NextToken is not null);
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeExportTasks 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeLogGroups** 함께 사용

다음 코드 예제는 DescribeLogGroups의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- syslog 수집 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Retrieves information about existing Amazon CloudWatch Logs log groups
/// and displays the information on the console.
/// </summary>
public class DescribeLogGroups
{
    public static async Task Main()
    {
        // Creates a CloudWatch Logs client using the default
        // user. If you need to work with resources in another
        // AWS Region than the one defined for the default user,
        // pass the AWS Region as a parameter to the client constructor.
        var client = new AmazonCloudWatchLogsClient();

        bool done = false;
        string newToken = null;

        var request = new DescribeLogGroupsRequest
        {
            Limit = 5,
        };
```

```
            DescribeLogGroupsResponse response;

            do
            {
                if (newToken is not null)
                {
                    request.NextToken = newToken;
                }

                response = await client.DescribeLogGroupsAsync(request);

                response.LogGroups.ForEach(lg =>
                {
                    Console.WriteLine($"{lg.LogGroupName} is associated with the
key: {lg.KmsKeyId}.");
                    Console.WriteLine($"Created on:
{lg.CreationTime.Date.Date}");
                    Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
                });

                if (response.NextToken is null)
                {
                    done = true;
                }
                else
                {
                    newToken = response.NextToken;
                }
            }
            while (!done);
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeLogGroups를 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 my-logs인 로그 그룹을 설명합니다.

```
aws logs describe-log-groups --log-group-name-prefix my-logs
```

출력:

```
{
    "logGroups": [
        {
            "storedBytes": 0,
            "metricFilterCount": 0,
            "creationTime": 1433189500783,
            "logGroupName": "my-logs",
            "retentionInDays": 5,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:*"
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DescribeLogGroups를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});
```

```
export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
    if (page.logGroups?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DescribeLogGroups 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeLogStreams** 함께 사용

다음 코드 예제는 DescribeLogStreams의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 첫 번째 서버리스 함수 생성

CLI

AWS CLI

다음 명령은 로그 그룹 my-logs의 2015 접두사로 시작하는 모든 로그 스트림을 보여 줍니다.

```
aws logs describe-log-streams --log-group-name my-logs --log-stream-name-
prefix 2015
```

출력:

```
{
    "logStreams": [
        {
            "creationTime": 1433189871774,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150531",
            "logStreamName": "20150531",
            "storedBytes": 0
        },
        {
            "creationTime": 1433189873898,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150601",
            "logStreamName": "20150601",
            "storedBytes": 0
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DescribeLogStreams를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

지정된 접두사와 일치하는 지정된 로그 그룹 내의 로그 스트림을 검색합니다.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 WeathertopJavaContainerLogs).
                  logStreamName - The name of the log stream (for example,
 weathertop-java-stream).
                  pattern - the pattern to use (for example, INFO)

                """;

        if (args.length != 3) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = args[0] ;
        String logStreamName = args[1] ;
        String pattern = args[2] ;

        CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
 pattern);
    }

    /**
     * Searches for log streams with a specific prefix within a log group and
 filters log events based on a specified pattern.
     *
     * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName    the name of the log group to search within
```

```
     * @param logStreamPrefix the prefix of the log streams to search for
     * @param pattern         the pattern to filter log events by
     */
    public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
 cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
        DescribeLogStreamsRequest describeLogStreamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .logStreamNamePrefix(logStreamPrefix)
                .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
 cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
 FilterLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNames(logStreamName)
                    .filterPattern(pattern)
                    .build();

            FilterLogEventsResponse filterLogEventsResponse =
 cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }


 System.out.println("--------------------------------------------------"); //
 Separator for better readability
        }
    }
}
```

지정된 로그 그룹의 최신 로그 스트림에 대한 메타데이터를 출력합니다.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """;

        if (args.length != 1) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
 specified log group.
     *
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName the name of the log group
     */
```

```
    public static void describeMostRecentLogStream(CloudWatchLogsClient
 logsClient, String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
                .descending(true)
                .limit(1)
                .build();

        try {
            DescribeLogStreamsResponse streamsResponse =
 logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
 logGroupName);
                return;
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
 stream.firstEventTimestamp());
            System.out.println("  Last Event Time: " +
 stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeLogStreams을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **DescribeSubscriptionFilters** 함께 사용

다음 코드 예제는 DescribeSubscriptionFilters의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

구독 필터를 나열합니다.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
        request.SetLogGroupName(log_group);
        request.SetLimit(1);

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = cwl.DescribeSubscriptionFilters(
                    request);
            if (!outcome.IsSuccess()) {
```

```
                std::cout << "Failed to describe CloudWatch subscription filters
"
                    << "for log group " << log_group << ": " <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(32) << "Name" <<
                    std::setw(64) << "FilterPattern" << std::setw(64) <<
                    "DestinationArn" << std::endl;
                header = true;
            }

            const auto &filters = outcome.GetResult().GetSubscriptionFilters();
            for (const auto &filter : filters) {
                std::cout << std::left << std::setw(32) <<
                    filter.GetFilterName() << std::setw(64) <<
                    filter.GetFilterPattern() << std::setw(64) <<
                    filter.GetDestinationArn() << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeSubscriptionFilters를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

```
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroup>

                Where:
                  logGroup - A log group name (for example, myloggroup).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        describeFilters(logs, logGroup);
        logs.close();
    }

    public static void describeFilters(CloudWatchLogsClient logs, String
 logGroup) {
```

```
        try {
            boolean done = false;
            String newToken = null;

            while (!done) {
                DescribeSubscriptionFiltersResponse response;
                if (newToken == null) {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                            .logGroupName(logGroup)
                            .limit(1).build();

                    response = logs.describeSubscriptionFilters(request);
                } else {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                            .nextToken(newToken)
                            .logGroupName(logGroup)
                            .limit(1).build();
                    response = logs.describeSubscriptionFilters(request);
                }

                for (SubscriptionFilter filter : response.subscriptionFilters())
{
                    System.out.printf("Retrieved filter with name %s, " +
"pattern %s " + "and destination arn %s",
                            filter.filterName(),
                            filter.filterPattern(),
                            filter.destinationArn());
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeSubscriptionFilters를 참조하세요.

JavaScript

SDK for JavaScript(v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  // This will return a list of all subscription filters in your account
  // matching the log group name.
  const command = new DescribeSubscriptionFiltersCommand({
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
    limit: 1,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DescribeSubscriptionFilters를 참조하세요.

SDK for JavaScript(v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cwl.describeSubscriptionFilters(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DescribeSubscriptionFilters를 참조하세요.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { cwlClient
 ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeSubscriptionFilters를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `GetLogEvents` 함께 사용

다음 코드 예제는 `GetLogEvents`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 첫 번째 서버리스 함수 생성

CLI

AWS CLI

다음 명령은 my-logs 로그 그룹에서 이름이 20150601인 로그 스트림의 로그 이벤트를 검색합니다.

```
aws logs get-log-events --log-group-name my-logs --log-stream-name 20150601
```

출력:

```
{
    "nextForwardToken":
 "f/31961209122447488583055879464742346735121166569214640130",
    "events": [
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190516679,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184358,
            "message": "Example Event 2"
        }
    ],
    "nextBackwardToken":
 "b/31961209122358285602261756944988674324553373268216709120"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 GetLogEvents를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
```

```
                  logGroupName - The name of the log group (for example,
 myloggroup).
                  logStreamName - The name of the log stream (for example,
 mystream).

                """;

       // if (args.length != 2) {
       //     System.out.print(usage);
       //     System.exit(1);
//        }

        String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
        String logStreamName = "weathertop-java-stream" ; //args[1];

        Region region = Region.US_EAST_1 ;
        CloudWatchLogsClient cloudWatchLogsClient =
 CloudWatchLogsClient.builder()
                .region(region)
                .build();

        getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
        cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                      String logGroupName,
                                      String logStreamPrefix) {
        try {
            // First, find the exact log stream name
            DescribeLogStreamsRequest describeRequest =
 DescribeLogStreamsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNamePrefix(logStreamPrefix)
                    .limit(1) // get the first matching stream
                    .build();

            DescribeLogStreamsResponse describeResponse =
 cloudWatchLogsClient.describeLogStreams(describeRequest);

            if (describeResponse.logStreams().isEmpty()) {
                System.out.println("No matching log streams found for prefix: " +
 logStreamPrefix);
                return;
```

```
            }

            String exactLogStreamName =
 describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7,
 ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest =
 GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not
 prefix
                    .startTime(startTime)
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
 cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetLogEvents를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 `GetQueryResults` 함께 사용

다음 코드 예제는 `GetQueryResults`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 대용량 쿼리 실행

.NET

SDK for .NET (v4)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Gets the results of a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="queryId">The ID of the query.</param>
    /// <returns>The query results response.</returns>
    public async Task<GetQueryResultsResponse?> GetQueryResultsAsync(string
queryId)
    {
        try
        {
            var request = new GetQueryResultsRequest
            {
                QueryId = queryId
            };

            var response = await
_amazonCloudWatchLogs.GetQueryResultsAsync(request);
            return response;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Query not found: {ex.Message}");
```

```
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while getting query results:
{ex.Message}");
            return null;
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetQueryResults를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetQueryResults를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 GetQueryResults를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_cwl->getqueryresults(
          iv_queryid = iv_query_id ).

        " Display query status and result count
        DATA(lv_status) = oo_result->get_status( ).
        DATA(lt_results) = oo_result->get_results( ).
        DATA(lv_result_count) = lines( lt_results ).

        MESSAGE |Query status: { lv_status }. Retrieved { lv_result_count } log
event(s).| TYPE 'I'.
      CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 GetQueryResults를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **ListSyslogConfigurations** 함께 사용

다음 코드 예제는 `ListSyslogConfigurations`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- syslog 수집 기본 사항 알아보기

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    def list_syslog_configurations(
        self,
        log_group_identifier: Optional[str] = None,
        vpc_endpoint_id: Optional[str] = None,
    ) -> list[dict[str, Any]]:
        """
        Lists syslog configurations, optionally filtered by log group or VPC
endpoint.

        Handles pagination by following nextToken until all results are returned.

        :param log_group_identifier: Optional log group name or ARN to filter by.
        :param vpc_endpoint_id: Optional VPC endpoint ID to filter by.
        :return: A list of syslog configuration dictionaries.
        :raises ClientError: If a filter parameter is invalid.
        """
        try:
            configurations = list()
            params = dict()
            if log_group_identifier is not None:
                params["logGroupIdentifier"] = log_group_identifier
            if vpc_endpoint_id is not None:
                params["vpcEndpointId"] = vpc_endpoint_id

            while True:
                response = self.logs_client.list_syslog_configurations(**params)
```

```
                configurations.extend(response.get("syslogConfigurations",
list()))
                next_token = response.get("nextToken", None)
                if next_token is None:
                    break
                params["nextToken"] = next_token

            logger.info("Listed %d syslog configuration(s).",
len(configurations))
            return configurations
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidParameterException":
                logger.error(
                    "Invalid filter parameter: %s. Ensure the log group
identifier "
                    "or VPC endpoint ID is correctly formatted.",
                    error.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Failed to list syslog configurations: %s",
                    error.response["Error"]["Message"],
                )
            raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 ListSyslogConfigurations를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **PutResourcePolicy** 함께 사용

다음 코드 예제는 PutResourcePolicy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- syslog 수집 기본 사항 알아보기

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    def put_resource_policy(
        self,
        log_group_arn: str,
        vpc_endpoint_id: str,
    ) -> dict[str, Any]:
        """
        Creates or updates a resource-scoped policy on a log group that grants
        the syslog.logs.amazonaws.com service principal permission to call
        logs:PutLogEvents and logs:CreateLogStream.

        A resource-scoped policy (one attached to a specific log group) is
        specified with resourceArn and policyDocument. The PutResourcePolicy
        API does not allow policyName to be combined with resourceArn.

        :param log_group_arn: The ARN of the log group (with trailing :*).
        :param vpc_endpoint_id: The VPC endpoint ID to scope the condition.
        :return: The resource policy response.
        """
        policy_document = json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "SyslogIngestPermissions",
                        "Effect": "Allow",
                        "Principal": {"Service": "syslog.logs.amazonaws.com"},
                        "Action": [
                            "logs:PutLogEvents",
                            "logs:CreateLogStream",
                        ],
                        "Resource": log_group_arn,
                        "Condition": {
```

```
                        "StringEquals": {"aws:sourceVpce": vpc_endpoint_id}
                    },
                }
            ],
        }
    )
    try:
        response = self.logs_client.put_resource_policy(
            policyDocument=policy_document,
            resourceArn=log_group_arn,
        )
        logger.info("Put resource policy on log group '%s'.", log_group_arn)
        return response
    except ClientError as error:
        logger.error(
            "Failed to put resource policy: %s",
            error.response["Error"]["Message"],
        )
        raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 PutResourcePolicy를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **PutSubscriptionFilter** 함께 사용

다음 코드 예제는 PutSubscriptionFilter의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

구독 필터를 생성합니다.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetFilterPattern(filter_pattern);
        request.SetLogGroupName(log_group);
        request.SetDestinationArn(dest_arn);
        auto outcome = cwl.PutSubscriptionFilter(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch logs subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 PutSubscriptionFilter를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 *
 * Also, before running this Java V2 code example,set up your development
 * environment,including your credentials.
 *
 * For more information,see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <pattern> <logGroup> <functionArn>\s

                Where:
                  filter - A filter name (for example, myfilter).
                  pattern - A filter pattern (for example, ERROR).
```

```
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
               """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String filter = args[0];
       String pattern = args[1];
       String logGroup = args[2];
       String functionArn = args[3];
       Region region = Region.US_WEST_2;
       CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
               .region(region)
               .build();

       putSubFilters(cwl, filter, pattern, logGroup, functionArn);
       cwl.close();
   }

   public static void putSubFilters(CloudWatchLogsClient cwl,
           String filter,
           String pattern,
           String logGroup,
           String functionArn) {

       try {
           PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
                   .filterName(filter)
                   .filterPattern(pattern)
                   .logGroupName(logGroup)
                   .destinationArn(functionArn)
                   .build();

           cwl.putSubscriptionFilter(request);
           System.out.printf(
                   "Successfully created CloudWatch logs subscription filter
%s",
                   filter);
```

```
        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutSubscriptionFilter를 참조하세요.

JavaScript

SDK for JavaScript(v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { PutSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutSubscriptionFilterCommand({
    // An ARN of a same-account Kinesis stream, Kinesis Firehose
    // delivery stream, or Lambda function.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
SubscriptionFilters.html
    destinationArn: process.env.CLOUDWATCH_LOGS_DESTINATION_ARN,

    // A name for the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,

    // A filter pattern for subscribing to a filtered stream of log events.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
FilterAndPatternSyntax.html
    filterPattern: process.env.CLOUDWATCH_LOGS_FILTER_PATTERN,

    // The name of the log group. Messages in this group matching the filter
 pattern
    // will be sent to the destination ARN.
```

```
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 PutSubscriptionFilter를 참조하세요.

SDK for JavaScript(v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cwl.putSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
```

```
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 PutSubscriptionFilter 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **PutSyslogConfiguration** 함께 사용

다음 코드 예제는 PutSyslogConfiguration의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- syslog 수집 기본 사항 알아보기

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    def put_syslog_configuration(
        self, log_group_identifier: str, vpc_endpoint_id: str
    ) -> None:
        """
        Creates or updates a syslog configuration for a log group. This enables
        ingestion of syslog data through the specified VPC endpoint.

        :param log_group_identifier: The name or ARN of the log group to
associate
```

```
            with the syslog configuration.
        :param vpc_endpoint_id: The ID of the VPC endpoint to use for syslog
            ingestion.
        :raises ClientError: If the log group or VPC endpoint does not exist.
        """
        try:
            self.logs_client.put_syslog_configuration(
                logGroupIdentifier=log_group_identifier,
                vpcEndpointId=vpc_endpoint_id,
            )
            logger.info(
                "Created syslog configuration for log group '%s' with VPC
endpoint '%s'.",
                log_group_identifier,
                vpc_endpoint_id,
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "The specified log group or VPC endpoint does not exist: %s",
                    error.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Failed to put syslog configuration: %s",
                    error.response["Error"]["Message"],
                )
            raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 PutSyslogConfiguration을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`StartLiveTail`** 함께 사용

다음 코드 예제는 `StartLiveTail`의 사용 방법을 보여 줍니다.

.NET

SDK for .NET

필수 파일을 포함합니다.

```
using Amazon;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

Live Tail 세션을 시작합니다.

```
            var client = new AmazonCloudWatchLogsClient();
            var request = new StartLiveTailRequest
            {
                LogGroupIdentifiers = logGroupIdentifiers,
                LogStreamNames = logStreamNames,
                LogEventFilterPattern = filterPattern,
            };

            var response = await client.StartLiveTailAsync(request);

            // Catch if request fails
            if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Failed to start live tail session");
                return;
            }
```

Live Tail 세션의 이벤트는 두 가지 방법으로 처리할 수 있습니다.

```
            /* Method 1
            * 1). Asynchronously loop through the event stream
            * 2). Set a timer to dispose the stream and stop the Live Tail
 session at the end.
            */
            var eventStream = response.ResponseStream;
            var task = Task.Run(() =>
            {
                foreach (var item in eventStream)
```

```
                {
                    if (item is LiveTailSessionUpdate liveTailSessionUpdate)
                    {
                        foreach (var sessionResult in
liveTailSessionUpdate.SessionResults)
                        {
                            Console.WriteLine("Message : {0}",
sessionResult.Message);
                        }
                    }
                    if (item is LiveTailSessionStart)
                    {
                        Console.WriteLine("Live Tail session started");
                    }
                    // On-stream exceptions are processed here
                    if (item is CloudWatchLogsEventStreamException)
                    {
                        Console.WriteLine($"ERROR: {item}");
                    }
                }
            });
            // Close the stream to stop the session after a timeout
            if (!task.Wait(TimeSpan.FromSeconds(10))){
                eventStream.Dispose();
                Console.WriteLine("End of line");
            }
```

```
            /* Method 2
            * 1). Add event handlers to each event variable
            * 2). Start processing the stream and wait for a timeout using
AutoResetEvent
            */
            AutoResetEvent endEvent = new AutoResetEvent(false);
            var eventStream = response.ResponseStream;
            using (eventStream) // automatically disposes the stream to stop the
session after execution finishes
            {
                eventStream.SessionStartReceived += (sender, e) =>
                {
                    Console.WriteLine("LiveTail session started");
                };
                eventStream.SessionUpdateReceived += (sender, e) =>
```

```
                {
                    foreach (LiveTailSessionLogEvent logEvent in
e.EventStreamEvent.SessionResults){
                        Console.WriteLine("Message: {0}", logEvent.Message);
                    }
                };
                // On-stream exceptions are captured here
                eventStream.ExceptionReceived += (sender, e) =>
                {
                    Console.WriteLine($"ERROR:
{e.EventStreamException.Message}");
                };

                eventStream.StartProcessing();
                // Stream events for this amount of time.
                endEvent.WaitOne(TimeSpan.FromSeconds(10));
                Console.WriteLine("End of line");
            }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 StartLiveTail을 참조하세요.

CLI

AWS CLI

다음 명령은 이름이 인 로그 그룹에서 Live Tail 세션을 시작합니다. `my-logs`

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs
```

다음 명령은 대화형 모드에서 라는 로그 그룹에서 Live Tail 세션을 시작합니다`my-logs`.

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode interactive
```

대화형 모드에서는 테일 로그에서 최대 5개의 용어를 강조 표시할 수 있습니다. 심각도 코드는 기본적으로 강조 표시됩니다. `h`를 눌러 강조 표시 모드로 전환한 다음 강조 표시할 용어를 한 번에 하나씩 입력하고 Enter 키를 누릅니다. 강조 표시된 용어(들)를 지우`c`려면를 누릅니다. JSON/Plain 텍스트 간에 서식을 전환`t`하려면를 누릅니다. `Esc`를 눌러 Live Tail 세션을 종료합

니다. up / down 키를 눌러 로그 이벤트 사이를 위 또는 아래로 스크롤하고 Ctrl + u /Ctrl + d를 사용하여 더 빠르게 스크롤합니다. q를 눌러 최신 로그 이벤트로 스크롤합니다.

- API 세부 정보는 명령 참조의 StartLiveTail을 참조하세요. AWS CLI

Go

SDK for Go V2

필수 파일을 포함합니다.

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

Live Tail 세션의 이벤트를 처리합니다.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
 eventsChan := stream.Events()
 for {
  event := <-eventsChan
  switch e := event.(type) {
  case *types.StartLiveTailResponseStreamMemberSessionStart:
   log.Println("Received SessionStart event")
  case *types.StartLiveTailResponseStreamMemberSessionUpdate:
   for _, logEvent := range e.Value.SessionResults {
    log.Println(*logEvent.Message)
   }
  default:
   // Handle on-stream exceptions
   if err := stream.Err(); err != nil {
    log.Fatalf("Error occured during streaming: %v", err)
   } else if event == nil {
    log.Println("Stream is Closed")
    return
```

```
   } else {
    log.Fatalf("Unknown event type: %T", e)
   }
  }
 }
}
```

Live Tail 세션을 시작합니다.

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
 panic("configuration error, " + err.Error())
}
client := cloudwatchlogs.NewFromConfig(cfg)

request := &cloudwatchlogs.StartLiveTailInput{
 LogGroupIdentifiers:   logGroupIdentifiers,
 LogStreamNames:        logStreamNames,
 LogEventFilterPattern: logEventFilterPattern,
}

response, err := client.StartLiveTail(context.TODO(), request)
// Handle pre-stream Exceptions
if err != nil {
 log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

일정 시간이 경과하면 Live Tail 세션을 중단합니다.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- API 세부 정보는 AWS SDK for Go API 참조의 StartLiveTail을 참조하세요.

Java

SDK for Java 2.x

필수 파일을 포함합니다.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Live Tail 세션의 이벤트를 처리합니다.

```
    private static StartLiveTailResponseHandler
 getStartLiveTailResponseStreamHandler(
            AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
 throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
```

```
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
                        LiveTailSessionStart sessionStart =
(LiveTailSessionStart) event;
                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
                        LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                        List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                        logEvents.forEach(e -> {
                            long timestamp = e.timestamp();
                            Date date = new Date(timestamp);
                            System.out.println("[" + date + "] " + e.message());
                        });
                    } else {
                        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                    }
                }

                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
                }
            })
            .build();
    }
```

Live Tail 세션을 시작합니다.

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();

        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

        cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

일정 시간이 경과하면 Live Tail 세션을 중단합니다.

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartLiveTail을 참조하세요.

JavaScript

SDK for JavaScript(v3)

필수 파일을 포함합니다.

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-
cloudwatch-logs";
```

Live Tail 세션의 이벤트를 처리합니다.

```
async function handleResponseAsync(response) {
    try {
      for await (const event of response.responseStream) {
        if (event.sessionStart !== undefined) {
          console.log(event.sessionStart);
        } else if (event.sessionUpdate !== undefined) {
          for (const logEvent of event.sessionUpdate.sessionResults) {
            const timestamp = logEvent.timestamp;
            const date = new Date(timestamp);
            console.log("[" + date + "] " + logEvent.message);
          }
        } else {
            console.error("Unknown event type");
        }
      }
    } catch (err) {
        // On-stream exceptions are captured here
        console.error(err)
    }
}
```

Live Tail 세션을 시작합니다.

```
    const client = new CloudWatchLogsClient();

    const command = new StartLiveTailCommand({
        logGroupIdentifiers: logGroupIdentifiers,
        logStreamNames: logStreamNames,
        logEventFilterPattern: filterPattern
    });
```

```
    try{
        const response = await client.send(command);
        handleResponseAsync(response);
    } catch (err){
        // Pre-stream exceptions are captured here
        console.log(err);
    }
```

일정 시간이 경과하면 Live Tail 세션을 중단합니다.

```
    /* Set a timeout to close the client. This will stop the Live Tail session.
 */
    setTimeout(function() {
        console.log("Client timeout");
        client.destroy();
      }, 10000);
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 StartLiveTail을 참조하세요.

Kotlin

SDK for Kotlin

필수 파일을 포함합니다.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Live Tail 세션을 시작합니다.

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
        logEventFilterPattern = logEventFilterPatternVal
    }
```

```
    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is
StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StartLiveTail을 참조하세요.

Python

SDK for Python(Boto3)

필수 파일을 포함합니다.

```
import boto3
```

```
import time
from datetime import datetime
```

Live Tail 세션을 시작합니다.

```
    # Initialize the client
    client = boto3.client('logs')

    start_time = time.time()

    try:
        response = client.start_live_tail(
            logGroupIdentifiers=log_group_identifiers,
            logStreamNames=log_streams,
            logEventFilterPattern=filter_pattern
        )
        event_stream = response['responseStream']
        # Handle the events streamed back in the response
        for event in event_stream:
            # Set a timeout to close the stream.
            # This will end the Live Tail session.
            if (time.time() - start_time >= 10):
                event_stream.close()
                break
            # Handle when session is started
            if 'sessionStart' in event:
                session_start_event = event['sessionStart']
                print(session_start_event)
            # Handle when log event is given in a session update
            elif 'sessionUpdate' in event:
                log_events = event['sessionUpdate']['sessionResults']
                for log_event in log_events:
                    print('[{date}]
{log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['me
            else:
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 StartLiveTail을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **StartQuery** 함께 사용

다음 코드 예제는 StartQuery의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 대용량 쿼리 실행

.NET

SDK for .NET (v4)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Starts a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="logGroupName">The name of the log group to query.</param>
    /// <param name="queryString">The CloudWatch Logs Insights query string.</
param>
    /// <param name="startTime">The start time for the query (seconds since
 epoch).</param>
    /// <param name="endTime">The end time for the query (seconds since epoch).</
param>
    /// <param name="limit">The maximum number of results to return.</param>
    /// <returns>The query ID if successful, null otherwise.</returns>
    public async Task<string?> StartQueryAsync(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit = 10000)
```

```
    {
        try
        {
            var request = new StartQueryRequest
            {
                LogGroupName = logGroupName,
                QueryString = queryString,
                StartTime = startTime,
                EndTime = endTime,
                Limit = limit
            };

            var response = await _amazonCloudWatchLogs.StartQueryAsync(request);
            return response.QueryId;
        }
        catch (InvalidParameterException ex)
        {
            _logger.LogError($"Invalid parameter for query: {ex.Message}");
            return null;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Log group not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while starting query:
{ex.Message}");
            return null;
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 StartQuery를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
        endTime: endDate.valueOf(),
        limit: maxLogs,
      }),
    );
  } catch (err) {
    /** @type {string} */
    const message = err.message;
    if (message.startsWith("Query's end date and time")) {
      // This error indicates that the query's start or end date occur
      // before the log group was created.
      throw new DateOutOfBoundsError(message);
    }

    throw err;
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 StartQuery를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
```

```
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
```

```
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 StartQuery를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   TRY.
       " iv_log_group_name = '/aws/lambda/my-function'
       " iv_query_string = 'fields @timestamp, @message | sort @timestamp desc |
limit 20'
       " iv_start_time and iv_end_time must be in Unix epoch milliseconds (ms
since Jan 1, 1970 00:00:00 UTC)
       oo_result = lo_cwl->startquery(
         iv_loggroupname = iv_log_group_name
         iv_starttime    = iv_start_time
         iv_endtime      = iv_end_time
         iv_querystring  = iv_query_string
         iv_limit        = iv_limit ).

       " Display the query ID for tracking
       DATA(lv_query_id) = oo_result->get_queryid( ).
       MESSAGE |Query started successfully with ID: { lv_query_id }| TYPE 'I'.
     CATCH /aws1/cx_cwlinvalidparameterex.
       MESSAGE 'Invalid parameter.' TYPE 'E'.
     CATCH /aws1/cx_cwllimitexceededex.
       MESSAGE 'Limit exceeded.' TYPE 'E'.
     CATCH /aws1/cx_cwlmalformedqueryex.
       MESSAGE 'Malformed query.' TYPE 'E'.
     CATCH /aws1/cx_cwlresourcenotfoundex.
       MESSAGE 'Resource not found.' TYPE 'E'.
     CATCH /aws1/cx_cwlserviceunavailex.
```

```
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 StartQuery를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용한 CloudWatch Logs 시나리오

다음 코드 예제에서는 CloudWatch Logs AWS SDKs에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오는 CloudWatch Logs 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접 호출하여 특정 작업을 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- 컨테이너 서비스 연결 구성
- 첫 번째 서버리스 함수 생성
- AWS SDK를 사용하여 CloudWatch Logs syslog 수집 기본 사항 알아보기
- CloudWatch Logs를 사용하여 대규모 쿼리 실행
- 예약된 이벤트를 사용하여 Lambda 함수 간접 호출

컨테이너 서비스 연결 구성

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- VPC 인프라 생성
- 로깅 설정
- ECS 클러스터 생성
- IAM 역할 구성

- Service Connect를 사용하여 서비스 생성
- 배포 확인
- 리소스 정리

Bash

AWS CLI Bash 스크립트 사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 샘플 개발자 튜토리얼 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
```

```
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}
```

```
# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
```

```
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
```

```
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
```

```
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
```

```
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
```

```
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"
```

```
    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
```

```
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
```

```
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
```

```
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================="
    echo "CREATED RESOURCES"
    echo "============================================="
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================="
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
```

```
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
```

```
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id
```

```
    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================="
    echo "CLEANUP CONFIRMATION"
    echo "============================================="
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
the AWS Management Console."
```

```
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService(AWS Cloud Map)
  - DeleteService(Amazon ECS)
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity

- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- 대기(Amazon EC2)
- 대기(Amazon ECS)

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

첫 번째 서버리스 함수 생성

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Lambda용 IAM 역할 생성
- 함수 코드 생성
- Lambda 함수 생성
- Lambda 함수 테스트
- 리소스 정리

Bash

AWS CLI Bash 스크립트 사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 샘플 개발자 튜토리얼 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash
```

```
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

################################################################################
# Helper functions
################################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
```

```
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
IAM role ${NAME}."
                ;;
```

```
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR: Script failed at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
```

```
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "========================================="
echo "AWS Lambda - Create Your First Function"
echo "========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
```

```
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
```

```
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

# ============================================================================
# Step 1: Create IAM execution role
# ============================================================================

echo ""
echo "=========================================="
echo "Step 1: Create IAM execution role"
echo "=========================================="
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
```

```
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

################################################################################
# Step 2: Create Lambda function
################################################################################

echo ""
echo "========================================="
echo "Step 2: Create Lambda function"
echo "========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
```

```
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

###############################################################################
# Step 3: Invoke the function
###############################################################################
```

```
echo ""
echo "=========================================="
echo "Step 3: Invoke the function"
echo "=========================================="
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

################################################################################
# Step 4: View CloudWatch logs
################################################################################

echo ""
echo "=========================================="
```

```
echo "Step 4: View CloudWatch Logs"
echo "========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
```

```
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

################################################################################
# Summary and cleanup
################################################################################

echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents

- Invoke

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 CloudWatch Logs syslog 수집 기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- syslog VPC 엔드포인트를 프로비저닝하는 CloudFormation 스택을 배포합니다.
- 로그 그룹을 생성하고 syslog 서비스에 대한 리소스 정책을 추가합니다.
- PutSyslogConfiguration 구성을 생성합니다.
- 필터가 있거나 없는 syslog 구성을 나열합니다.
- syslog 구성, 로그 그룹 및 스택을 삭제하여 정리합니다.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
class SyslogIngestionScenario:
    """Interactive scenario demonstrating CloudWatch Logs syslog ingestion."""

    STACK_NAME = "syslog-demo-stack"

    def __init__(
        self,
        cloudwatch_logs_wrapper: CloudWatchLogsWrapper,
        cf_client: boto3.client,
    ) -> None:
        """
```

```
        :param cloudwatch_logs_wrapper: A CloudWatchLogsWrapper instance.
        :param cf_client: A Boto3 CloudFormation client.
        """
        self.wrapper = cloudwatch_logs_wrapper
        self.cf_client = cf_client
        self.vpc_endpoint_id: Optional[str] = None
        self.log_group_name: Optional[str] = None
        self.log_group_arn: Optional[str] = None
        self.stack_deployed = False

    # --------------------------------------------------------------------
    # Setup phase
    # --------------------------------------------------------------------
    def deploy_cfn_stack(self) -> str:
        """
        Deploys the CloudFormation stack that provisions the VPC endpoint
        prerequisite. Waits for CREATE_COMPLETE and returns the VpcEndpointId
        output.

        :return: The VPC endpoint ID.
        """
        print("\n" + "=" * 68)
        print("Step 1: Deploy the CloudFormation prerequisite stack")
        print("=" * 68)
        print(
            f"\nDeploying stack '{self.STACK_NAME}' to create a VPC, subnet, "
            "security group, and syslog VPC endpoint.\n"
            "This can take several minutes..."
        )

        try:
            self.cf_client.create_stack(
                StackName=self.STACK_NAME,
                TemplateBody=_load_cfn_template(),
                Capabilities=["CAPABILITY_IAM"],
            )
            # Only this run's own stack should be torn down in cleanup. Mark it
            # as soon as creation is initiated so a stack that fails partway
            # (e.g. ROLLBACK) is still cleaned up.
            self.stack_deployed = True
        except ClientError as error:
            if error.response["Error"]["Code"] == "AlreadyExistsException":
                # The stack pre-existed this run. Reuse it but do NOT mark it
                # for cleanup — deleting a stack we did not create could destroy
```

```
                # infrastructure the user set up themselves.
                print(
                    f"Stack '{self.STACK_NAME}' already exists. Reusing it; it "
                    "will not be deleted during cleanup."
                )
            else:
                raise

        waiter = self.cf_client.get_waiter("stack_create_complete")
        try:
            waiter.wait(
                StackName=self.STACK_NAME,
                WaiterConfig={"Delay": 30, "MaxAttempts": 40},
            )
        except WaiterError:
            # The waiter fails both when the stack genuinely failed
            # (CREATE_FAILED, ROLLBACK_COMPLETE, ...) and, harmlessly, when the
            # stack already existed in a completed state before this run.
            # Inspect the actual status so a real failure surfaces clearly
            # instead of as a confusing "missing outputs" error later.
            status = self.cf_client.describe_stacks(StackName=self.STACK_NAME)[
                "Stacks"
            ][0]["StackStatus"]
            if status not in ("CREATE_COMPLETE", "UPDATE_COMPLETE"):
                raise RuntimeError(
                    f"CloudFormation stack '{self.STACK_NAME}' did not reach a "
                    f"successful state (status: {status}). Check the stack events
"
                    "in the CloudFormation console for the failure reason."
                )

        response = self.cf_client.describe_stacks(StackName=self.STACK_NAME)
        outputs = response["Stacks"][0].get("Outputs", list())
        vpc_endpoint_id = None
        for output in outputs:
            if output["OutputKey"] == "VpcEndpointId":
                vpc_endpoint_id = output["OutputValue"]
                break

        if vpc_endpoint_id is None:
            raise RuntimeError(
                "VpcEndpointId not found in stack outputs. "
                "Check the CloudFormation stack for errors."
            )
```

```
        self.vpc_endpoint_id = vpc_endpoint_id
        print(f"\nStack deployed. VPC Endpoint ID: {vpc_endpoint_id}")
        return vpc_endpoint_id

    def create_and_verify_log_group(self) -> None:
        """
        Prompts the user for a log group name, creates it, and verifies it
        with DescribeLogGroups.
        """
        print("\n" + "=" * 68)
        print("Step 2: Create and verify a log group")
        print("=" * 68)

        name_pattern = re.compile(r"^[a-zA-Z0-9_\-/.#]{1,512}$")

        while True:
            log_group_name = q.ask(
                "\nEnter a name for the syslog log group (e.g. /syslog/demo): "
            )
            if name_pattern.match(log_group_name):
                break
            print("Invalid name. Use 1-512 characters: a-z, A-Z, 0-9, " "_,
-, /, ., #")

        self.log_group_name = log_group_name
        self.wrapper.create_log_group(log_group_name)

        log_groups = self.wrapper.describe_log_groups(
            log_group_name_prefix=log_group_name
        )
        for lg in log_groups:
            if lg.get("logGroupName") == log_group_name:
                creation_ms = lg.get("creationTime", 0)
                creation_dt = datetime.fromtimestamp(
                    creation_ms / 1000, tz=timezone.utc
                )
                self.log_group_arn = lg.get("logGroupArn", "")
                print(f"\nLog group verified:")
                print(f"  Name:    {lg['logGroupName']}")
                print(f"  ARN:     {lg.get('arn', 'N/A')}")
                print(f"  Created: {creation_dt.strftime('%Y-%m-%d %H:%M:%S
UTC')}")
                return
```

```
        print(f"\nWARNING: Log group '{log_group_name}' not found in describe
results.")

    def add_resource_policy(self) -> None:
        """
        Adds a resource policy to the log group so the syslog service can
        write to it.
        """
        print("\n" + "=" * 68)
        print("Step 3: Add a resource policy for syslog ingestion")
        print("=" * 68)

        if self.log_group_arn is None or self.vpc_endpoint_id is None:
            print(
                "WARNING: Cannot add resource policy - missing log group ARN or
VPC endpoint."
            )
            return

        self.wrapper.put_resource_policy(
            log_group_arn=self.log_group_arn,
            vpc_endpoint_id=self.vpc_endpoint_id,
        )
        print(
            "\nResource policy created. The "
            "syslog.logs.amazonaws.com service principal can now call "
            "logs:PutLogEvents and logs:CreateLogStream on the log group."
        )

    # ------------------------------------------------------------------
    # Create syslog configuration phase
    # ------------------------------------------------------------------
    def create_syslog_configuration(self) -> None:
        """
        Creates a syslog configuration that associates the VPC endpoint with
        the log group.
        """
        print("\n" + "=" * 68)
        print("Step 4: Create the syslog configuration")
        print("=" * 68)

        print(
            "\nAssociating the VPC endpoint with the log group so that syslog "
```

```
            "messages arriving at the endpoint are stored in the log group."
        )
        print(
            "CloudWatch Logs parses common syslog formats (RFC 5424, "
            "RFC 3164, Cisco FTD/ASA) and extracts fields such as "
            "facility, severity, hostname, and appName.\n"
        )

        self.wrapper.put_syslog_configuration(
            log_group_identifier=self.log_group_name,
            vpc_endpoint_id=self.vpc_endpoint_id,
        )

        print(
            f"\nSyslog ingestion is now enabled for log group "
            f"'{self.log_group_name}' through VPC endpoint "
            f"'{self.vpc_endpoint_id}'."
        )
        print("\nSupported transport options:")
        print("  - TCP with TLS on port 6514 (recommended)")
        print("  - Plaintext TCP on port 1514")
        print("  - UDP on port 514")

    # ------------------------------------------------------------------
    # List syslog configurations phase
    # ------------------------------------------------------------------
    def _display_configurations(self, configs: list) -> None:
        """Helper to display a list of syslog configurations."""
        if not configs:
            print("  (no configurations found)")
            return
        for cfg in configs:
            created_ms = cfg.get("createdAt", 0)
            created_dt = datetime.fromtimestamp(created_ms / 1000,
tz=timezone.utc)
            print(f"  Log Group ARN:    {cfg.get('logGroupArn', 'N/A')}")
            print(f"  VPC Endpoint ID:  {cfg.get('vpcEndpointId', 'N/A')}")
            print(f"  Source Type:      {cfg.get('sourceType', 'N/A')}")
            print(f"  Created At:       {created_dt.strftime('%Y-%m-%d %H:%M:%S
UTC')}")
            print()

    def list_all_configurations(self) -> None:
        """Lists all syslog configurations in the account (no filter)."""
```

```
        print("\n" + "=" * 68)
        print("Step 5a: List all syslog configurations (no filter)")
        print("=" * 68)

        configs = self.wrapper.list_syslog_configurations()
        print(f"\nAll syslog configurations ({len(configs)} found):\n")
        self._display_configurations(configs)

    def list_configurations_by_log_group(self) -> None:
        """Lists syslog configurations filtered by the scenario's log group."""
        print("\n" + "=" * 68)
        print("Step 5b: List configurations filtered by log group")
        print("=" * 68)

        configs = self.wrapper.list_syslog_configurations(
            log_group_identifier=self.log_group_name,
        )
        print(
            f"\nConfigurations for log group '{self.log_group_name}' "
            f"({len(configs)} found):\n"
        )
        self._display_configurations(configs)

    def list_configurations_by_vpc_endpoint(self) -> None:
        """Lists syslog configurations filtered by VPC endpoint."""
        print("\n" + "=" * 68)
        print("Step 5c: List configurations filtered by VPC endpoint")
        print("=" * 68)

        configs = self.wrapper.list_syslog_configurations(
            vpc_endpoint_id=self.vpc_endpoint_id,
        )
        print(
            f"\nConfigurations for VPC endpoint '{self.vpc_endpoint_id}' "
            f"({len(configs)} found):\n"
        )
        self._display_configurations(configs)

    # ------------------------------------------------------------------
    # Cleanup phase
    # ------------------------------------------------------------------
    def cleanup(self, prompt: bool = True) -> None:
        """
        Cleans up the resources created during the scenario. Tolerates
```

```
        resources that were never created or are already gone, so it is safe to
        call after a partial failure.

        :param prompt: When True (the normal end-of-scenario path), ask the user
            to confirm before deleting. When False (best-effort cleanup after an
            error), skip the prompt and delete whatever was created.
        """
        print("\n" + "=" * 68)
        print("Cleanup")
        print("=" * 68)

        if prompt:
            do_cleanup = q.ask(
                "\nDo you want to delete all resources created during this "
                "scenario? (y/n) ",
                q.is_yesno,
            )
            # q.is_yesno converts the answer to a bool (True when the user
            # answered 'y'); check it directly rather than calling string
methods.
            if not do_cleanup:
                print("Skipping cleanup. Resources remain in your account.")
                return
        else:
            print(
                "\nAn error occurred; attempting best-effort cleanup of any "
                "resources that were created..."
            )

        # Delete syslog configuration
        if self.log_group_name and self.vpc_endpoint_id:
            print(
                f"\nDeleting syslog configuration for log group "
                f"'{self.log_group_name}'..."
            )
            self.wrapper.delete_syslog_configuration(
                log_group_identifier=self.log_group_name,
                vpc_endpoint_id=self.vpc_endpoint_id,
            )
            print(
                "  After deletion, syslog data is no longer ingested through "
                "the VPC endpoint into this log group."
            )
```

```
            # Verify deletion
            configs = self.wrapper.list_syslog_configurations(
                log_group_identifier=self.log_group_name,
            )
            if not configs:
                print("  Verified: configuration has been removed.")
            else:
                print(
                    f"  WARNING: Still found {len(configs)} configuration(s). "
                    "They may take a moment to be fully removed."
                )

        # Delete log group
        if self.log_group_name:
            print(f"\nDeleting log group '{self.log_group_name}'...")
            self.wrapper.delete_log_group(self.log_group_name)
            print(
                "  The log group and all archived log events have been "
                "permanently deleted."
            )

        # Delete CloudFormation stack
        if self.stack_deployed:
            print(f"\nDeleting CloudFormation stack '{self.STACK_NAME}'...")
            try:
                self.cf_client.delete_stack(StackName=self.STACK_NAME)
                waiter = self.cf_client.get_waiter("stack_delete_complete")
                waiter.wait(
                    StackName=self.STACK_NAME,
                    WaiterConfig={"Delay": 30, "MaxAttempts": 40},
                )
                print("  Stack deleted successfully.")
            except (ClientError, WaiterError) as error:
                logger.warning("Stack deletion issue: %s", error)
                print(
                    f"  WARNING: Stack deletion may still be in progress. "
                    f"Check the CloudFormation console."
                )

    # ------------------------------------------------------------------
    # Run the full scenario
    # ------------------------------------------------------------------
    def run_scenario(self) -> None:
        """Runs all phases of the syslog ingestion scenario."""
```

```
        print("\n" + "=" * 68)
        print("CloudWatch Logs Syslog Ingestion Scenario")
        print("=" * 68)
        print(
            "\nThis scenario demonstrates CloudWatch Logs managed syslog "
            "ingestion, which lets you route syslog data from firewalls, "
            "routers, switches, and Linux servers directly into CloudWatch "
            "Logs through a VPC endpoint - without running a separate "
            "collection tier.\n"
        )

        succeeded = False
        try:
            # Setup
            self.deploy_cfn_stack()
            self.create_and_verify_log_group()
            self.add_resource_policy()

            # Create syslog configuration
            self.create_syslog_configuration()

            q.ask("\nPress Enter to continue to listing configurations...")

            # List syslog configurations
            self.list_all_configurations()
            self.list_configurations_by_log_group()
            self.list_configurations_by_vpc_endpoint()
            succeeded = True

            # On success, prompt before cleaning up.
            self.cleanup(prompt=True)
        finally:
            # On failure, run best-effort cleanup without prompting so that
            # partially-created resources are not left behind.
            if not succeeded:
                self.cleanup(prompt=False)

        print("\n" + "=" * 68)
        print("Scenario complete!")
        print("=" * 68)
        print(
            "\nYou have successfully demonstrated CloudWatch Logs managed "
            "syslog ingestion:\n"
            "  - Deployed a CloudFormation stack with a syslog VPC endpoint.\n"
```

```
            "  - Created a log group and authorized the syslog service.\n"
            "  - Associated a syslog configuration with PutSyslogConfiguration.
\n"
            "  - Listed configurations with and without filters.\n"
            "  - Cleaned up all resources.\n"
        )
```

CloudWatch Logs 작업을 래핑하는 클래스를 생성합니다.

```
class CloudWatchLogsWrapper:
    """Encapsulates Amazon CloudWatch Logs actions."""

    def __init__(self, logs_client: boto3.client) -> None:
        """
        Initializes the CloudWatchLogsWrapper with a CloudWatch Logs client.

        :param logs_client: A Boto3 CloudWatch Logs client.
        """
        self.logs_client = logs_client

    @classmethod
    def from_client(cls) -> "CloudWatchLogsWrapper":
        """
        Creates a CloudWatchLogsWrapper instance with a default CloudWatch Logs
 client.

        :return: An instance of CloudWatchLogsWrapper.
        """
        logs_client = boto3.client("logs")
        return cls(logs_client)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
  - CreateLogGroup
  - DeleteLogGroup
  - DeleteSyslogConfiguration
  - DescribeLogGroups

- ListSyslogConfigurations
- PutResourcePolicy
- PutSyslogConfiguration

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CloudWatch Logs를 사용하여 대규모 쿼리 실행

다음 코드 예제는 CloudWatch Logs를 사용하여 10,000개 이상의 레코드를 쿼리하는 방법을 보여줍니다.

.NET

SDK for .NET (v4)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이는 대규모 쿼리 시나리오를 보여주는 기본 워크플로입니다.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
using CloudWatchLogsActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace CloudWatchLogsScenario;

public class LargeQueryWorkflow
{
```

```
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the CloudWatch Logs
 Large Query workflow:

    1. Prepare the Application:
       - Prompt the user to deploy CloudFormation stack and generate sample logs.
       - Deploy the CloudFormation template for resource creation.
       - Generate 50,000 sample log entries using CloudWatch Logs API.
       - Wait 5 minutes for logs to be fully ingested.

    2. Execute Large Query:
       - Perform recursive queries to retrieve all logs using binary search.
       - Display progress for each query executed.
       - Show total execution time and logs found.

    3. Clean up:
       - Prompt the user to delete the CloudFormation stack and all resources.
       - Destroy the CloudFormation stack and wait until removed.
    */

    public static ILogger<LargeQueryWorkflow> _logger = null!;
    public static CloudWatchLogsWrapper _wrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _logGroupName = "/workflows/cloudwatch-logs/large-
query";
    private static string _logStreamName = "stream1";
    private static long _queryStartDate;
    private static long _queryEndDate;

    public static bool _interactive = true;
    public static string _stackName = "CloudWatchLargeQueryStack";
    private static string _stackResourcePath = "../../../../../../../scenarios/
features/cloudwatch_logs_large_query/resources/stack.yaml";

    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter("Microsoft", LogLevel.Information))
            .ConfigureServices((_, services) =>
```

```
                services.AddAWSService<IAmazonCloudWatchLogs>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<CloudWatchLogsWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<LargeQueryWorkflow>();

            _wrapper = host.Services.GetRequiredService<CloudWatchLogsWrapper>();
            _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Runs the scenario workflow. Used for testing.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        Console.WriteLine();

        try
        {
            var deployStack = !_interactive || GetYesNoResponse(
                "Would you like to deploy the CloudFormation stack and generate
sample logs? (y/n) ");

            if (deployStack)
            {
                if (_interactive)
                {
                    Console.Write(
                        $"Enter a path for the CloudFormation stack
resource .yaml file (or press Enter for default '{_stackResourcePath}'): ");
                    string? inputPath = Console.ReadLine();
                    if (!string.IsNullOrWhiteSpace(inputPath))
                    {
                        _stackResourcePath = inputPath;
                    }
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName);
```

```
                if (deploySuccess)
                {
                    Console.WriteLine();
                    Console.WriteLine("Generating 50,000 sample log entries...");
                    var generateSuccess = await GenerateSampleLogs();

                    if (generateSuccess)
                    {
                        Console.WriteLine();
                        Console.WriteLine("Sample logs created. Waiting 5 minutes
for logs to be fully ingested...");
                        await WaitWithCountdown(300);

                        Console.WriteLine("Application preparation complete.");
                        return true;
                    }
                }
            }
            else
            {
                _logGroupName = PromptUserForInput("Enter the log group name ",
_logGroupName);
                _logStreamName = PromptUserForInput("Enter the log stream name ",
_logStreamName);

                var startDateMs = PromptUserForLong("Enter the query start date
(milliseconds since epoch): ");
                var endDateMs = PromptUserForLong("Enter the query end date
(milliseconds since epoch): ");

                _queryStartDate = startDateMs / 1000;
                _queryEndDate = endDateMs / 1000;

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }

        Console.WriteLine("Application preparation failed.");
```

```
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath)
            };

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
```

```
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
```

```
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Generates sample logs directly using CloudWatch Logs API.
    /// Creates 50,000 log entries spanning 5 minutes.
    /// </summary>
    /// <returns>True if logs were generated successfully.</returns>
    private static async Task<bool> GenerateSampleLogs()
    {
        const int totalEntries = 50000;
        const int entriesPerBatch = 10000;
        const int fiveMinutesMs = 5 * 60 * 1000;

        try
        {
            // Calculate timestamps
            var startTimeMs = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
            var timestampIncrement = fiveMinutesMs / totalEntries;

            Console.WriteLine($"Generating {totalEntries} log entries...");
```

```
            var entryCount = 0;
            var currentTimestamp = startTimeMs;
            var numBatches = totalEntries / entriesPerBatch;

            // Generate and upload logs in batches
            for (int batchNum = 0; batchNum < numBatches; batchNum++)
            {
                var logEvents = new List<InputLogEvent>();

                for (int i = 0; i < entriesPerBatch; i++)
                {
                    logEvents.Add(new InputLogEvent
                    {
                        Timestamp =
 DateTimeOffset.FromUnixTimeMilliseconds(currentTimestamp).UtcDateTime,
                        Message = $"Entry {entryCount}"
                    });

                    entryCount++;
                    currentTimestamp += timestampIncrement;
                }

                // Upload batch
                var success = await _wrapper.PutLogEventsAsync(_logGroupName,
 _logStreamName, logEvents);
                if (!success)
                {
                    _logger.LogError($"Failed to upload batch {batchNum + 1}/
{numBatches}");
                    return false;
                }

                Console.WriteLine($"Uploaded batch {batchNum + 1}/{numBatches}");
            }

            // Set query date range (convert milliseconds to seconds for query
 API)
            _queryStartDate = startTimeMs / 1000;
            _queryEndDate = (currentTimestamp - timestampIncrement) / 1000;

            Console.WriteLine($"Query start date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryStartDate):yyyy-MM-
ddTHH:mm:ss.fffZ}");
```

```
            Console.WriteLine($"Query end date:
{DateTimeOffset.FromUnixTimeSeconds(_queryEndDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Successfully uploaded {totalEntries} log
entries");

            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while generating sample
logs.");
            return false;
        }
    }

    /// <summary>
    /// Executes the large query workflow.
    /// </summary>
    public static async Task ExecuteLargeQuery()
    {
        Console.WriteLine("Starting recursive query to retrieve all logs...");
        Console.WriteLine();

        var queryLimit = PromptUserForInteger("Enter the query limit (max 10000)
", 10000);
        if (queryLimit > 10000) queryLimit = 10000;

        var queryString = "fields @timestamp, @message | sort @timestamp asc";

        var stopwatch = Stopwatch.StartNew();
        var allResults = await PerformLargeQuery(_logGroupName, queryString,
_queryStartDate, _queryEndDate, queryLimit);
        stopwatch.Stop();

        Console.WriteLine();
        Console.WriteLine($"Queries finished in
{stopwatch.Elapsed.TotalSeconds:F3} seconds.");
        Console.WriteLine($"Total logs found: {allResults.Count}");

        // Check for duplicates
        Console.WriteLine();
        Console.WriteLine("Checking for duplicate logs...");
        var duplicates = FindDuplicateLogs(allResults);
        if (duplicates.Count > 0)
```

```
        {
            Console.WriteLine($"WARNING: Found {duplicates.Count} duplicate log
entries!");
            Console.WriteLine("Duplicate entries (showing first 10):");
            foreach (var dup in duplicates.Take(10))
            {
                Console.WriteLine($"  [{dup.Timestamp}] {dup.Message} (appears
{dup.Count} times)");
            }

            var uniqueCount = allResults.Count - duplicates.Sum(d => d.Count -
1);
            Console.WriteLine($"Unique logs: {uniqueCount}");
        }
        else
        {
            Console.WriteLine("No duplicates found. All logs are unique.");
        }
        Console.WriteLine();

        var viewSample = !_interactive || GetYesNoResponse("Would you like to see
a sample of the logs? (y/n) ");
        if (viewSample)
        {
            Console.WriteLine();
            Console.WriteLine($"Sample logs (first 10 of {allResults.Count}):");
            for (int i = 0; i < Math.Min(10, allResults.Count); i++)
            {
                var timestamp = allResults[i].Find(f => f.Field ==
"@timestamp")?.Value ?? "N/A";
                var message = allResults[i].Find(f => f.Field ==
"@message")?.Value ?? "N/A";
                Console.WriteLine($"[{timestamp}] {message}");
            }
        }
    }

    /// <summary>
    /// Performs a large query using recursive binary search.
    /// </summary>
    private static async Task<List<List<ResultField>>> PerformLargeQuery(
        string logGroupName,
        string queryString,
        long startTime,
```

```
        long endTime,
        int limit)
    {
        var queryId = await _wrapper.StartQueryAsync(logGroupName, queryString,
 startTime, endTime, limit);
        if (queryId == null)
        {
            return new List<List<ResultField>>();
        }

        var results = await PollQueryResults(queryId);
        if (results == null || results.Count == 0)
        {
            return new List<List<ResultField>>();
        }

        var startDate =
 DateTimeOffset.FromUnixTimeSeconds(startTime).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var endDate = DateTimeOffset.FromUnixTimeSeconds(endTime).ToString("yyyy-
MM-ddTHH:mm:ss.fffZ");
        Console.WriteLine($"Query date range: {startDate} ({startTime}s) to
 {endDate} ({endTime}s). Found {results.Count} logs.");

        if (results.Count < limit)
        {
            Console.WriteLine($"  -> Returning {results.Count} logs (less than
 limit of {limit})");
            return results;
        }

        Console.WriteLine($"  -> Hit limit of {limit}. Need to split and
 recurse.");

        // Get the timestamp of the last log (sorted to find the actual last one)
        var lastLogTimestamp = GetLastLogTimestamp(results);
        if (lastLogTimestamp == null)
        {
            Console.WriteLine($"  -> No timestamp found in results. Returning
 {results.Count} logs.");
            return results;
        }

        Console.WriteLine($"  -> Last log timestamp: {lastLogTimestamp}");
```

```
        // Parse the timestamp and add 1 millisecond to avoid querying the same
 log again
        var lastLogDate = DateTimeOffset.Parse(lastLogTimestamp + " +0000");
        Console.WriteLine($"  -> Last log as DateTimeOffset: {lastLogDate:yyyy-
MM-ddTHH:mm:ss.fffZ} ({lastLogDate.ToUnixTimeSeconds()}s)");

        var offsetLastLogDate = lastLogDate.AddMilliseconds(1);
        Console.WriteLine($"  -> Offset timestamp (last
 + 1ms): {offsetLastLogDate:yyyy-MM-ddTHH:mm:ss.fffZ}
 ({offsetLastLogDate.ToUnixTimeSeconds()}s)");

        // Convert to seconds, but round UP to the next second to avoid
 overlapping with logs in the same second
        // This ensures we don't re-query logs that share the same second as the
 last log
        var offsetLastLogTime = offsetLastLogDate.ToUnixTimeSeconds();
        if (offsetLastLogDate.Millisecond > 0)
        {
            offsetLastLogTime++; // Move to the next full second
            Console.WriteLine($"  -> Adjusted to next full second:
 {offsetLastLogTime}s
 ({DateTimeOffset.FromUnixTimeSeconds(offsetLastLogTime):yyyy-MM-
ddTHH:mm:ss.fffZ})");
        }

        Console.WriteLine($"  -> Comparing:
 offsetLastLogTime={offsetLastLogTime}s vs endTime={endTime}s");
        Console.WriteLine($"  -> End time as date:
 {DateTimeOffset.FromUnixTimeSeconds(endTime):yyyy-MM-ddTHH:mm:ss.fffZ}");

        // Check if there's any time range left to query
        if (offsetLastLogTime >= endTime)
        {
            Console.WriteLine($"  -> No time range left to query. Offset time
 ({offsetLastLogTime}s) >= end time ({endTime}s)");
            return results;
        }

        // Split the remaining date range in half
        var (range1Start, range1End, range2Start, range2End) =
 SplitDateRange(offsetLastLogTime, endTime);
```

```
        var range1StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range1Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range1EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range1End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range2Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range2End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");

        Console.WriteLine($"  -> Splitting remaining range:");
        Console.WriteLine($"     Range 1: {range1StartDate} ({range1Start}s) to
 {range1EndDate} ({range1End}s)");
        Console.WriteLine($"     Range 2: {range2StartDate} ({range2Start}s) to
 {range2EndDate} ({range2End}s)");

        // Query both halves recursively
        Console.WriteLine($"  -> Querying range 1...");
        var results1 = await PerformLargeQuery(logGroupName, queryString,
 range1Start, range1End, limit);
        Console.WriteLine($"  -> Range 1 returned {results1.Count} logs");

        Console.WriteLine($"  -> Querying range 2...");
        var results2 = await PerformLargeQuery(logGroupName, queryString,
 range2Start, range2End, limit);
        Console.WriteLine($"  -> Range 2 returned {results2.Count} logs");

        // Combine all results
        var allResults = new List<List<ResultField>>(results);
        allResults.AddRange(results1);
        allResults.AddRange(results2);

        Console.WriteLine($"  -> Combined total: {allResults.Count} logs
 ({results.Count} + {results1.Count} + {results2.Count})");

        return allResults;
    }

    /// <summary>
    /// Gets the timestamp string of the most recent log from a list of logs.
    /// Sorts timestamps to find the actual last one.
```

```
    /// </summary>
    private static string? GetLastLogTimestamp(List<List<ResultField>> logs)
    {
        var timestamps = logs
            .Select(log => log.Find(f => f.Field == "@timestamp")?.Value)
            .Where(t => !string.IsNullOrEmpty(t))
            .OrderBy(t => t)
            .ToList();

        if (timestamps.Count == 0)
        {
            return null;
        }

        return timestamps[timestamps.Count - 1];
    }

    /// <summary>
    /// Splits a date range in half.
    /// Range 2 starts at midpoint + 1 second to avoid overlap.
    /// </summary>
    private static (long range1Start, long range1End, long range2Start, long
range2End) SplitDateRange(long startTime, long endTime)
    {
        var midpoint = startTime + (endTime - startTime) / 2;
        // Range 2 starts at midpoint + 1 to avoid querying the same second twice
        return (startTime, midpoint, midpoint + 1, endTime);
    }

    /// <summary>
    /// Polls for query results until complete.
    /// </summary>
    private static async Task<List<List<ResultField>>?> PollQueryResults(string
queryId)
    {
        int retryCount = 0;
        const int maxRetries = 60;
        const int retryDelay = 1000;

        while (retryCount < maxRetries)
        {
            var response = await _wrapper.GetQueryResultsAsync(queryId);
            if (response == null)
            {
```

```
                return null;
            }

            if (response.Status == QueryStatus.Complete)
            {
                return response.Results;
            }

            if (response.Status == QueryStatus.Failed ||
                response.Status == QueryStatus.Cancelled ||
                response.Status == QueryStatus.Timeout ||
                response.Status == QueryStatus.Unknown)
            {
                _logger.LogError($"Query failed with status: {response.Status}");
                return null;
            }

            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for query results.");
        return null;
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    public static async Task<bool> Cleanup()
    {
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete the CloudFormation stack and all resources?
(y/n) ");

        if (cleanup)
        {
            try
            {
                var stackDeleteSuccess = await
DeleteCloudFormationStack(_stackName, false);
                return stackDeleteSuccess;
            }
            catch (Exception ex)
            {
```

```
                _logger.LogError(ex, "An error occurred while cleaning up the
resources.");
                return false;
            }
        }

        Console.WriteLine($"Resources will remain. Stack name: {_stackName}, Log
group: {_logGroupName}");
        _logger.LogInformation("CloudWatch Logs Large Query scenario is
complete.");
        return true;
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
bool forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
            return true;
        }
        else
        {
```

```
            _logger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }

                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus
== StackStatus.DELETE_FAILED)
                {
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
```

```
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Waits with a countdown display.
    /// </summary>
    private static async Task WaitWithCountdown(int seconds)
    {
        for (int i = seconds; i > 0; i--)
        {
            Console.Write($"\rWaiting: {i} seconds remaining...  ");
            await Task.Delay(1000);
        }
        Console.WriteLine("\rWait complete.                      ");
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
```

```
        if (_interactive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = "[a-zA-Z][-a-zA-Z0-9]*";
                if (!Regex.IsMatch(input, regex))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
{_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }

    /// <summary>
    /// Prompts the user for input with a default value.
    /// </summary>
    private static string PromptUserForInput(string prompt, string defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
            string? input = Console.ReadLine();
            return string.IsNullOrWhiteSpace(input) ? defaultValue : input;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for an integer value.
    /// </summary>
    private static int PromptUserForInteger(string prompt, int defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
```

```
            string? input = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(input) || !int.TryParse(input, out var
result))
            {
                return defaultValue;
            }
            return result;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for a long value.
    /// </summary>
    private static long PromptUserForLong(string prompt)
    {
        if (_interactive)
        {
            Console.Write(prompt);
            string? input = Console.ReadLine();
            if (long.TryParse(input, out var result))
            {
                return result;
            }
        }
        return 0;
    }

    /// <summary>
    /// Finds duplicate log entries based on timestamp and message.
    /// </summary>
    private static List<(string Timestamp, string Message, int Count)>
FindDuplicateLogs(List<List<ResultField>> logs)
    {
        var logSignatures = new Dictionary<string, int>();

        foreach (var log in logs)
        {
            var timestamp = log.Find(f => f.Field == "@timestamp")?.Value ?? "";
            var message = log.Find(f => f.Field == "@message")?.Value ?? "";
            var signature = $"{timestamp}|{message}";

            if (logSignatures.ContainsKey(signature))
            {
```

```
                logSignatures[signature]++;
            }
            else
            {
                logSignatures[signature] = 1;
            }
        }

        return logSignatures
            .Where(kvp => kvp.Value > 1)
            .Select(kvp =>
            {
                var parts = kvp.Key.Split('|');
                return (Timestamp: parts[0], Message: parts[1], Count:
 kvp.Value);
            })
            .OrderByDescending(x => x.Count)
            .ToList();
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하세요.
  - GetQueryResults
  - StartQuery

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

진입점입니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
```

```
import { CloudWatchQuery } from "./cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
  throw new Error(
    "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
  );
}

const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
  logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
  dateRange: [
    new Date(Number.parseInt(process.env.QUERY_START_DATE)),
    new Date(Number.parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs
 found: ${cloudWatchQuery.results.length}`,
);
```

필요한 경우 쿼리를 여러 단계로 분할하는 클래스입니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
```

```
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient}
client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
{ limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;

    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }

  /**
```

```
   * Recursively query for logs.
   * @param {[Date, Date]} dateRange
   * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[]
[]>}
   */
  async _largeQuery(dateRange) {
    const logs = await this._query(dateRange, this.limit);

    console.log(
      `Query date range: ${dateRange
        .map((d) => d.toISOString())
        .join(" to ")}. Found ${logs.length} logs.`,
    );

    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
  }

  /**
   * Find the most recent log in a list of logs.
   * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
   */
  _getLastLogDate(logs) {
    const timestamps = logs
      .map(
        (log) =>
          log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
      )
      .filter((t) => !!t)
      .map((t) => `${t}Z`)
      .sort();
```

```
    if (!timestamps.length) {
      throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
  }

  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }

  /**
   * Starts a query and waits for it to complete.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   */
  async _query(dateRange, maxLogs) {
    try {
      const { queryId } = await this._startQuery(dateRange, maxLogs);
      const { results } = await this._waitUntilQueryDone(queryId);
      return results ?? [];
    } catch (err) {
      /**
       * This error is thrown when StartQuery returns an error indicating
       * that the query's start or end date occur before the log group was
       * created.
       */
      if (err instanceof DateOutOfBoundsError) {
        return [];
      }
      throw err;
    }
  }

  /**
   * Wrapper for the StartQueryCommand. Uses a static query string
   * for consistency.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   * @returns {Promise<{ queryId: string }>}
```

```
   */
  async _startQuery([startDate, endDate], maxLogs = 10000) {
    try {
      return await this.client.send(
        new StartQueryCommand({
          logGroupNames: this.logGroupNames,
          queryString: "fields @timestamp, @message | sort @timestamp asc",
          startTime: startDate.valueOf(),
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
    } catch (err) {
      /** @type {string} */
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }

  /**
   * Call GetQueryResultsCommand until the query is done.
   * @param {string} queryId
   */
  _waitUntilQueryDone(queryId) {
    const getResults = async () => {
      const results = await this._getQueryResults(queryId);
      const queryDone = [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
      ].includes(results.status);

      return { queryDone, results };
    };

    return retry(
```

```
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하세요.
  - GetQueryResults
  - StartQuery

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 파일은 10,000개의 결과를 초과하는 CloudWatch 쿼리를 관리하기 위한 예제 모듈을 간접 호출합니다.

```
import logging
import os
import sys

import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities
```

```
# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
        Initializes the CloudWatchLogsQueryRunner class by setting up date
 utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
 configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
 10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
 end dates.
        Fetches the environment variable for log group, returning the default
 value if it
        does not exist.

        :return: Tuple of query start date and end date as integers and the log
 group.
```

```
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
are required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
        except KeyError:
            logging.warning("No QUERY_LOG_GROUP environment variable, using
default value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 =
self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
        )
```

```
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
        query="fields @timestamp, @message | sort @timestamp asc"
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided
 date range.

        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
        :type log_group: str
        :param query: Query string to pass to the CloudWatchQuery instance
        :type query: str
        """
        cloudwatch_query = CloudWatchQuery(
            log_group=log_group,
            query_string=query
        )
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds.
 Total logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
 query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
```

```
    query_start_date, query_end_date, log_group =
 runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601,
 log_group=log_group)


if __name__ == "__main__":
    main()
```

이 모듈은 10,000개의 결과를 초과하는 CloudWatch 쿼리를 처리합니다.

```
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
```

```
    """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and
calculates the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:     {start_date}"
            f"\n       END:       {end_date}"
            f"\n       LOG GROUP: {self.log_group}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.

        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
```

```
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the
final batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
        :type :return List containing log item details
        """
```

```
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
```

```
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
```

```
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")


    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 다음 주제를 참조하세요.
  - GetQueryResults
  - StartQuery

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

예약된 이벤트를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon EventBridge 예약 이벤트에서 호출된 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

Java

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여줍니다. Lambda 함수가 간접 호출될 때 cron 표현식을 사용하여 일정을 예약하도록 EventBridge를 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK for JavaScript(v3)

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여줍니다. Lambda 함수가 간접 호출될 때 cron 표현식을 사용하여 일정을 예약하도록 EventBridge를 구성합니다. 이 예제에서는 Lambda JavaScript 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제는 AWS SDK for JavaScript v3 개발자 가이드에서도 확인할 수 있습니다.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK for Python(Boto3)

이 예제에서는 예약된 Amazon EventBridge 이벤트의 대상으로 AWS Lambda 함수를 등록하는 방법을 보여줍니다. Lambda 핸들러는 나중에 검색할 수 있도록 알기 쉬운 메시지와 전체 이벤트 데이터를 Amazon CloudWatch Logs에 기록합니다.

- Lambda 함수를 배포합니다.
- EventBridge 예약된 이벤트를 생성하고 Lambda 함수를 대상으로 만듭니다.
- EventBridge에 Lambda 함수를 간접 호출할 수 있는 권한을 부여합니다.
- CloudWatch Logs에서 최신 데이터를 인쇄하여 예약된 간접 호출의 결과를 표시합니다.
- 데모 중에 생성된 모든 리소스를 정리합니다.

이 예제는 GitHub에서 가장 잘 볼 수 있습니다. 전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 CloudWatch Logs 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

를 사용하여 CloudWatch Logs의 조회 테이블 암호화 AWS Key Management Service

조회 테이블 데이터는 항상 CloudWatch Logs에서 암호화됩니다. 기본적으로 CloudWatch Logs는 256비트 Advanced Encryption Standard Galois/Counter Mode(AES-GCM)를 사용한 서버 측 암호화를 사용하여 유휴 시 조회 테이블 데이터를 암호화합니다. 대신 이 암호화를 위해 AWS Key Management Service 을 사용할 수 있습니다. 이렇게 하면 AWS KMS 키를 사용하여 암호화가 완료됩니다. 를 사용한 암호화 AWS KMS 는 조회 테이블을 생성하거나 업데이트할 때 KMS 키를 조회 테이블과 연결하여 조회 테이블 수준에서 활성화됩니다.

> **Important**
>
> CloudWatch Logs는 대칭 KMS 키만 지원합니다. 비대칭 키를 사용하여 조회 테이블의 데이터를 암호화하지 마십시오. 자세한 내용은 대칭 및 비대칭 키 사용을 참조하세요.

KMS 키를 조회 테이블에 연결하면 조회 테이블에 저장된 모든 데이터가이 키를 사용하여 암호화됩니다. CloudWatch Logs는 요청이 있을 때마다 이 데이터를 암호화 해제합니다. CloudWatch Logs에는 암호화된 데이터 요청이 발생할 때마다 KMS 키에 대한 권한이 있어야 합니다.

나중에 조회 테이블에서 KMS 키의 연결을 해제하면 CloudWatch Logs는 CloudWatch Logs 기본 암호화 방법을 사용하여 데이터를 암호화합니다. 그러나 연결 해제하기 전에 키가 비활성화되거나 삭제된 경우 CloudWatch Logs는 해당 키로 암호화된 데이터를 읽을 수 없습니다.

CloudWatch Logs가 AWS KMS 를 사용하여 로그 데이터를 암호화하는 방법에 대한 일반적인 내용은 섹션을 참조하세요를 사용하여 CloudWatch Logs의 로그 데이터 암호화 AWS Key Management Service.

CloudWatch Logs가 조회 테이블에 AWS KMS 를 사용하는 방법

CloudWatch Logs는 AWS KMS 봉투 암호화를 사용하여 조회 테이블 데이터를 보호합니다. KMS 키를 조회 테이블에 연결하면 CloudWatch Logs는 `GenerateDataKey` 요청을에 전송합니다 AWS KMS.는 고유한 데이터 암호화 키(DEK)를 AWS KMS 생성하고 일반 텍스트 사본과 DEK의 암호화된 사본을 모두 반환합니다. CloudWatch Logs는 일반 텍스트 DEK를 사용하여 조회 테이블 데이터를 암호화한 다음 암호화된 DEK를 암호화된 데이터와 함께 저장합니다. 일반 텍스트 DEK는 저장되지 않으며 사용 후 메모리에서 삭제됩니다.

CloudWatch Logs가 조회 테이블 데이터를 읽어야 하는 경우 암호화된 DEK AWS KMS 를 사용하여에 Decrypt 요청을 보냅니다.는 DEK를 AWS KMS 복호화하고 일반 텍스트 DEK를 CloudWatch Logs에 반환하며,이 로그는 이를 사용하여 조회 테이블 데이터를 복호화합니다.

CloudWatch Logs는 AWS KMS에 요청할 때 다음 암호화 컨텍스트를 사용합니다.

```
{
    "aws:logs:arn": "arn:aws:logs:region:account-id:lookup-table:lookup-table-name"
}
```

IAM 정책 및 AWS KMS 키 정책에서이 암호화 컨텍스트를 사용하여 KMS 키에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 AWS KMS 키 및 암호화 컨텍스트 단원을 참조하십시오.

필요한 권한

조회 테이블에서 AWS KMS 암호화를 사용하려면 IAM 보안 주체에 KMS 키에 대한 다음 AWS KMS 권한이 있어야 합니다.

- kms:Decrypt
- kms:GenerateDataKey

CloudWatch Logs가 사용자를 대신하여 데이터를 복호화할 수 있도록 KMS 키로 암호화된 조회 테이블GetLookupTable에서를 호출할 때 kms:Decrypt 권한이 필요합니다. 암호화된 조회 테이블에서 lookup 명령을 사용하는 쿼리StartQuery를 사용하여를 호출할 때 키(조회 테이블을 암호화하는 데 사용되는 KMS 키)에 대한 kms:Decrypt 권한도 필요합니다. CloudWatch Logs가 조회 테이블 데이터를 암호화하는 데이터 암호화 키를 생성할 수 있도록 KMS 키를 UpdateLookupTable 사용하여 CreateLookupTable 또는를 호출할 때 kms:GenerateDataKey 권한이 필요합니다.

또한 CloudWatch Logs 서비스에는 KMS 키를 사용할 수 있는 권한이 있어야 합니다. 다음 섹션에 설명된 대로 KMS 키 정책에 정책 설명을 추가하여 이러한 권한을 부여합니다.

1단계: AWS KMS 키 생성

대칭 KMS 키를 생성하려면 다음 create-key 명령을 사용합니다.

```
aws kms create-key
```

이 명령의 출력 화면에는 키의 키 ID와 Amazon Resource Name(ARN)이 포함됩니다. 다음은 예제 출력입니다.

```
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
        "AWSAccountId": "123456789012",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

키 ARN을 기록합니다. 다음 단계에서 필요합니다.

2단계: KMS 키에 대한 권한 설정

기본적으로 모든 AWS KMS 키는 프라이빗입니다. 리소스 소유자만 이를 사용하여 데이터를 암호화 및 해독할 수 있습니다. CloudWatch Logs 서비스 보안 주체에게 키를 사용할 수 있는 권한을 부여하고 호출 역할에 키를 사용할 수 있는 권한도 부여해야 합니다.

먼저 다음 get-key-policy 명령을 사용하는 `policy.json`로 KMS 키에 대한 기본 정책을 저장합니다.

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./
policy.json
```

텍스트 편집기에서 `policy.json` 파일을 열고 다음 문을 추가하여 CloudWatch Logs 서비스 보안 주체에게 키를 사용할 수 있는 권한을 부여합니다. 이 예제에서는 암호화 컨텍스트와 일치하는 `Condition` 섹션을 사용하여 키를 특정 조회 테이블로 제한합니다.

```
{
```

```
    "Effect": "Allow",
    "Principal": {
        "Service": "logs.region.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:lookup-table:lookup-table-name",
            "aws:SourceAccount": "account-id",
            "aws:SourceArn": "arn:aws:logs:region:account-id:lookup-table:lookup-table-
name"
        }
    }
}
```

그런 다음 CloudWatch Logs `CreateLookupTable` 또는 `UpdateLookupTable` APIs를 호출할 역할에 권한을 추가합니다. CloudWatch Logs는 `kms:ViaService`를 사용하여 고객을 대신하여에 전화를 AWS KMS 겁니다. 자세한 내용은 kms:ViaService를 참조하세요.

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::account-id:role/role-name"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": [
                "logs.region.amazonaws.com"
            ]
        }
    }
}
```

마지막으로 아래 put-key-policy 명령을 사용하여 업데이트된 정책을 추가합니다.

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

3단계: KMS 키를 조회 테이블에 연결

`CreateLookupTable` API를 사용하여 KMS 키를 생성할 때 조회 테이블과 연결하거나 `UpdateLookupTable` API를 사용하여 기존 조회 테이블을 업데이트할 수 있습니다. 두 APIs 모두 AWSLogsConfigService의 일부입니다.

조회 테이블을 생성할 때 KMS 키를 조회 테이블과 연결하려면

`CreateLookupTable` API를 사용하고 KMS 키의 ARN으로 `kmsKeyArn` 파라미터를 지정합니다.

```
aws logs create-lookup-table \
    --lookup-table-name my-lookup-table \
    --kms-key-arn "arn:aws:kms:region:account-id:key/key-id"
```

KMS 키를 기존 조회 테이블과 연결하려면

`UpdateLookupTable` API를 사용하고 KMS 키의 ARN으로 `kmsKeyArn` 파라미터를 지정합니다.

```
aws logs update-lookup-table \
    --lookup-table-name my-lookup-table \
    --kms-key-arn "arn:aws:kms:region:account-id:key/key-id"
```

고려 사항

- 조회 테이블에서 KMS 키를 연결하거나 연결 해제한 후 작업이 적용되려면 최대 5분이 걸릴 수 있습니다.
- 연결된 키에 대한 CloudWatch Logs 액세스를 취소하거나 연결된 KMS 키를 삭제하면 CloudWatch Logs에서 암호화된 조회 테이블 데이터를 더 이상 검색할 수 없습니다.
- 이 주제의 단계를 수행하려면 , `kms:CreateKey`, `kms:PutKeyPolicy`및 `CreateLookupTable` 또는 `kms:GetKeyPolicy`를 호출할 적절한 CloudWatch Logs 권한이 있어야 합니다`UpdateLookupTable`.

Amazon CloudWatch Logs의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. WorkSpaces에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 사용자는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다

이 설명서는 Amazon CloudWatch Logs를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 보안 및 규정 준수 목표에 맞게 Amazon CloudWatch Logs를 구성하는 방법을 보여줍니다. 또한 CloudWatch Logs 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

내용

- Amazon CloudWatch Logs의 데이터 보호
- Amazon CloudWatch Logs용 Identity and Access Management
- Amazon CloudWatch Logs 규정 준수 검증
- Amazon CloudWatch Logs의 복원성
- Amazon CloudWatch Logs의 인프라 보안
- 인터페이스 VPC 엔드포인트에서 CloudWatch Logs 사용

Amazon CloudWatch Logs의 데이터 보호

> **Note**
>
> 의 일반 데이터 보호에 대한 다음 정보 외에도 AWS CloudWatch Logs를 사용하면 로그 이벤트에서 민감한 데이터를 마스킹하여 보호할 수 있습니다. 자세한 내용은 마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원 단원을 참조하십시오.

AWS 공동 책임 모델 Amazon CloudWatch Logs의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 일반 데이터 보호 규정 (GDPR) 센터를 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 CloudWatch Logs 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는

자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

저장된 데이터 암호화

CloudWatch Logs는 암호화를 사용하여 저장 시 데이터를 보호합니다. 모든 로그 그룹이 암호화됩니다. 기본적으로 CloudWatch Logs 서비스는 서버 측 암호화를 관리하고 256비트 AES-GCM(Advanced Encryption Standard Galois/Counter Mode)을 사용한 서버 측 암호화를 사용하여 저장 로그 데이터를 암호화합니다.

로그 암호화 및 복호화에 사용되는 키를 관리하려면 AWS KMS 키를 사용합니다. 자세한 내용은 를 사용하여 CloudWatch Logs의 로그 데이터 암호화 AWS Key Management Service 단원을 참조하십시오.

전송 중 데이터 암호화

CloudWatch Logs는 전송 중인 데이터의 종단 간 암호화를 사용합니다. CloudWatch Logs 서비스는 서버 측 암호화 키를 관리합니다.

Amazon CloudWatch Logs용 Identity and Access Management

Amazon CloudWatch Logs에 액세스하려면가 요청을 인증하는 데 사용할 AWS 수 있는 자격 증명이 필요합니다. 이러한 자격 증명에는 클라우드 AWS 리소스에 대한 CloudWatch Logs 데이터를 검색하는 등 리소스에 액세스할 수 있는 권한이 있어야 합니다. 다음 섹션에서는 리소스에 액세스할 수 있는지 대상을 제어하여 리소스를 보호할 수 있도록 AWS Identity and Access Management (IAM) 및 CloudWatch Logs를 사용하는 방법에 대한 세부 정보를 제공합니다.

- Authentication
- 액세스 관리

Authentication

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.

- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서의 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

액세스 관리

요청을 인증하는 데 유효한 자격 증명이 있더라도 권한이 없다면 CloudWatch Logs 리소스를 생성하거나 액세스할 수 없습니다. 예를 들어 로그 스트림과 로그 그룹 생성 등의 권한이 있어야 합니다.

다음 섹션에서는 CloudWatch Logs에 대한 권한을 관리하는 방법에 대해 설명합니다. 먼저 개요를 읽어보면 도움이 됩니다.

- CloudWatch Logs 리소스에 대한 액세스 권한 관리 개요
- CloudWatch Logs에 대한 ID 기반 정책(IAM 정책) 사용
- CloudWatch Logs 권한 참조

CloudWatch Logs 리소스에 대한 액세스 권한 관리 개요

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.

- (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

주제

- CloudWatch Logs 리소스 및 작업
- 리소스 소유권 이해
- 리소스 액세스 관리
- 정책 요소 지정: 작업, 효과, 보안 주체
- 정책에서 조건 지정

CloudWatch Logs 리소스 및 작업

CloudWatch Logs에서 기본 리소스는 로그 그룹, 로그 스트림 및 대상입니다. CloudWatch Logs는 하위 리소스(기본 리소스와 함께 사용되는 다른 리소스)를 지원하지 않습니다.

다음 표에 나와 있는 것처럼 이러한 리소스와 하위 리소스에는 고유한 Amazon 리소스 이름(ARN)이 연결되어 있습니다.

리소스 유형	ARN 형식
로그 그룹:	arn:aws:logs:*region*:*account-id* :log-group:*log_group_name*
로그 스트림	arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* :log-stream:*log-stream-name*
Destination	arn:aws:logs:*region*:*account-id* :destination:*destination_name*

ARN에 대한 자세한 내용은 IAM 사용 설명서의 ARN을 참조하세요. CloudWatch Logs ARN에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon 리소스 이름(ARN)을 참조하세요.

CloudWatch Logs에 적용되는 정책의 예제는 CloudWatch Logs에 대한 ID 기반 정책(IAM 정책) 사용을 참조하세요.

CloudWatch Logs 는 CloudWatch Logs 리소스를 사용하기 위한 일련의 작업을 제공합니다. 사용 가능한 작업 목록은 CloudWatch Logs 권한 참조 섹션을 참조하세요.

리소스 소유권 이해

AWS 계정은 누가 리소스를 생성했는지에 관계없이 계정에 생성된 리소스를 소유합니다. 특히 리소스 소유자는 리소스 생성 요청을 인증하는 AWS 보안 주체 엔터티(즉, 루트 계정, 사용자 또는 IAM 역할)의 계정입니다. 다음 예제에서는 이러한 작동 방법을 설명합니다.

- AWS 계정의 루트 계정 자격 증명을 사용하여 로그 그룹을 생성하는 경우 AWS 계정은 CloudWatch Logs 리소스의 소유자입니다.
- AWS 계정에서 사용자를 생성하고 해당 사용자에게 CloudWatch Logs 리소스를 생성할 수 있는 권한을 부여하는 경우 사용자는 CloudWatch Logs 리소스를 생성할 수 있습니다. 하지만 사용자가 속한 AWS 계정이 CloudWatch Logs 리소스를 소유합니다.
- AWS 계정에서 CloudWatch Logs 리소스를 생성할 권한이 있는 IAM 역할을 생성하는 경우 역할을 수임할 수 있는 사람은 누구나 CloudWatch Logs 리소스를 생성할 수 있습니다. 역할이 속한 AWS 계정이 CloudWatch Logs 리소스를 소유합니다.

리소스 액세스 관리

권한 정책은 누가 무엇에 액세스할 수 있는지를 나타냅니다. 다음 섹션에서는 권한 정책을 만드는 데 사용 가능한 옵션에 대해 설명합니다.

> **Note**
>
> 이 섹션에서는 CloudWatch Logs의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 내용은 다루지 않습니다. IAM 설명서 전체 내용은 IAM 사용 설명서의 IAM이란 무엇입니까?를 참조하세요. IAM 정책 구문 및 설명에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 참조 단원을 참조하세요.

IAM 자격 증명에 연결된 정책을 자격 증명 기반 정책(IAM 정책)이라 하고, 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다. CloudWatch Logs는 계정 구독 간에 활성화하는 데 사용되는 대상에 대해 ID 기반 정책과 리소스 기반 정책을 지원합니다. 자세한 정보는 교차 계정 교차 리전 구독 섹션을 참조하세요.

주제

- 로그 그룹 권한 및 Contributor Insights
- 리소스 기반 정책

로그 그룹 권한 및 Contributor Insights

Contributor Insights는 로그 그룹의 데이터를 분석하고 기여자 데이터를 표시하는 시계열을 생성할 수 있도록 하는 CloudWatch의 기능입니다. 상위 N개의 기고자, 총 고유 기고자 수 및 사용량에 대한 지표를 볼 수 있습니다. 자세한 내용은 Contributor Insights를 사용하여 높은 카디널리티 데이터 분석을 참조하세요.

Contributor Insights 작업은 `cloudwatch:` IAM 네임스페이스를 사용합니다. Contributor Insights가 로그 그룹 데이터를 처리할 때는 CloudWatch Logs(`logs:`) 권한이 평가되지 않습니다. Contributor Insights 권한 모델, 조건 키 및 모범 사례에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 조건 키를 사용하여 로그 그룹에 대한 Contributor Insights 사용자의 액세스 제한 섹션을 참조하세요.

로그 그룹에 민감한 데이터가 포함되어 있는 경우 Contributor Insights 또는 기타 기능이 처리하기 전에 CloudWatch Logs 데이터 보호 정책을 사용하여 해당 데이터를 마스킹합니다. 자세한 내용은 마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원 단원을 참조하십시오.

리소스 기반 정책

CloudWatch Logs는 대상에 리소스 기반 정책을 지원하는 데, 이 정책을 사용하여 계정 구독에서 활성화를 수행할 수 있습니다. 자세한 내용은 1단계: 대상 생성 단원을 참조하십시오. PutDestination API를 사용하여 대상을 생성하고, PutDestinationPolicy API를 사용하여 대상에 리소스 정책을 추가할 수 있습니다. 다음 예제는 계정 ID가 111122223333인 또 다른 AWS 계정이 대상 `arn:aws:logs:us-east-1:123456789012:destination:testDestination`에 대한 로그 그룹을 구독할 수 있도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
```

```
        "AWS" : "111122223333"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:123456789012:destination:testDestination"
    }
  ]
}
```

정책 요소 지정: 작업, 효과, 보안 주체

각 CloudWatch Logs 리소스에 대해 서비스는 API 작업 세트를 정의합니다. 이러한 API 작업에 대한 권한을 부여하기 위해 CloudWatch Logs에서는 정책에서 지정할 수 있는 작업을 정의합니다. 일부 API 작업에서는 API 작업을 수행하기 위해 복수의 작업에 대한 권한이 필요할 수 있습니다. 리소스 및 API 작업에 대한 자세한 내용은 CloudWatch Logs 리소스 및 작업 및 CloudWatch Logs 권한 참조 섹션을 참조하세요.

다음은 기본 정책 요소입니다.

- 리소스 – Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. 자세한 정보는 CloudWatch Logs 리소스 및 작업 섹션을 참조하세요.
- 작업 – 작업 키워드를 사용하여 허용 또는 거부하려는 리소스 작업을 식별합니다. 예를 들어, `logs.DescribeLogGroups` 권한은 사용자에게 `DescribeLogGroups` 작업 수행 권한을 허용합니다.
- Effect - 사용자가 특정 작업을 요청할 때 허용할지 아니면 거부할지 그 결과를 지정합니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 다른 정책에서 액세스 권한을 부여하는 경우라도 사용자가 해당 리소스에 액세스할 수 없도록 하기 위해 리소스에 대한 권한을 명시적으로 거부할 수도 있습니다.
- 보안 주체 – ID 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우, 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당). CloudWatch Logs는 대상에 대한 리소스 기반 정책을 지원합니다.

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

모든 CloudWatch Logs API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 CloudWatch Logs 권한 참조을 참조하세요.

정책에서 조건 지정

권한을 부여할 때 액세스 정책 언어를 사용하여 조건이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건을 참조하십시오.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. 각 AWS 서비스에서 지원하는 컨텍스트 키 목록과 AWS전체 정책 키 목록은 AWS 서비스 및 전역 조건 컨텍스트 키에 대한 작업, 리소스 AWS 및 조건 키를 참조하세요.

> **Note**
>
> 태그를 사용하여 로그 그룹 및 대상을 비롯한 CloudWatch Logs 리소스에 대한 액세스를 제어할 수 있습니다. 로그 그룹과 로그 스트림 간의 계층적 관계로 인해 로그 스트림에 대한 액세스는 로그 그룹 수준에서 제어됩니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 Amazon Web Services 리소스에 대한 액세스 제어를 참조하세요.

CloudWatch Logs에 대한 ID 기반 정책(IAM 정책) 사용

이 주제에서는 계정 관리자가 IAM ID(사용자, 그룹, 역할)에 권한 정책을 연결할 수 있는 ID 기반 정책의 예제를 제공합니다.

> **Important**
>
> CloudWatch Logs 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명이 나온 소개 주제 부분을 먼저 읽는 것이 좋습니다. 자세한 내용은 CloudWatch Logs 리소스에 대한 액세스 권한 관리 개요 단원을 참조하십시오.

이 주제에서는 다음을 다룹니다.

- CloudWatch 콘솔을 사용하는 데 필요한 권한
- AWS CloudWatch Logs에 대한 관리형(미리 정의된) 정책
- 고객 관리형 정책 예제

다음은 권한 정책의 예제입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
    ],
      "Resource": [
        "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

이 정책에는 로그 그룹 및 로그 스트림을 생성하고 로그 스트림에 로그 이벤트를 업로드하며 로그 스트림에 대한 세부 사항을 나열할 수 있는 권한을 부여하는 명령문이 하나 포함되어 있습니다.

Resource 값의 와일드카드 문자(*)는 CloudWatch Logs 리소스에서 나열된 작업에 대한 권한을 부여합니다.

- 권한을 특정 로그 그룹 작업(예: ,)으로 제한하려면 와일드카드를 로그 그룹 ARN으로 바꿉니다. CreateLogGroup DescribeLogStreamsarn:aws:logs:*us-west-2*:*123456789012*:log-group:*MyLogGroup*
- 권한을 특정 로그 스트림 작업(예: ,)으로 제한하려면 와일드카드를 로그 스트림 ARN —arn:aws:logs:*us-west-2*:*123456789012*:log-group:*MyLogGroup*:*(모든 스트림과 일치) 또는 arn:aws:logs:*us-west-2*:*123456789012*:log-group:*MyLogGroup*:log-stream:*MyStream* (특정 스트림)으로 바꿉니다. CreateLogStream PutLogEvents

각 API에 필요한 리소스 유형에 대한 서비스 권한 부여 참조를 참조하세요.

CloudWatch 콘솔을 사용하는 데 필요한 권한

사용자가 CloudWatch 콘솔에서 CloudWatch Logs로 작업하려면 해당 사용자에게 AWS 계정의 다른 AWS 리소스를 설명할 수 있는 최소 권한 집합이 있어야 합니다. CloudWatch 콘솔에서 CloudWatch Logs를 사용하려면 다음 서비스의 권한이 있어야 합니다.

- CloudWatch
- CloudWatch Logs
- OpenSearch Service
- IAM
- Kinesis
- Lambda
- Amazon S3

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔은 해당 IAM 정책에 연결된 사용자에 대해 의도대로 작동하지 않습니다. 해당 사용자가 CloudWatch 콘솔을 계속 사용할 수 있도록 하려면 AWS CloudWatch Logs에 대한 관리형(미리 정의된) 정책 단원에 설명된 대로 `CloudWatchReadOnlyAccess` 관리형 정책도 사용자에게 연결해야 합니다.

AWS CLI 또는 CloudWatch Logs API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다.

CloudWatch 콘솔을 사용하여 로그 구독을 관리하지 않는 사용자가 콘솔에서 작업하는 데 필요한 권한의 전체 집합은 다음과 같습니다.

- cloudwatch:GetMetricData
- cloudwatch:ListMetrics
- logs:CancelExportTask
- logs:CreateExportTask
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:DeleteLogGroup
- logs:DeleteLogStream
- logs:DeleteMetricFilter

- logs:DeleteQueryDefinition
- logs:DeleteRetentionPolicy
- logs:DeleteSubscriptionFilter
- logs:DescribeExportTasks
- logs:DescribeLogGroups
- logs:DescribeLogStreams
- logs:DescribeMetricFilters
- logs:DescribeQueryDefinitions
- logs:DescribeQueries
- logs:DescribeSubscriptionFilters
- logs:FilterLogEvents
- logs:GetLogEvents
- logs:GetLogGroupFields
- logs:GetLogRecord
- logs:GetQueryResults
- logs:PutMetricFilter
- logs:PutQueryDefinition
- logs:PutRetentionPolicy
- logs:StartQuery
- logs:StopQuery
- logs:PutSubscriptionFilter
- logs:TestMetricFilter

사용자가 콘솔을 사용하여 로그 구독도 관리할 경우 다음 권한도 필요합니다.

- es:DescribeElasticsearchDomain
- es:ListDomainNames
- iam:AttachRolePolicy
- iam:CreateRole
- iam:GetPolicy

- iam:GetPolicyVersion
- iam:GetRole
- iam:ListAttachedRolePolicies
- iam:ListRoles
- kinesis:DescribeStreams
- kinesis:ListStreams
- lambda:AddPermission
- lambda:CreateFunction
- lambda:GetFunctionConfiguration
- lambda:ListAliases
- lambda:ListFunctions
- lambda:ListVersionsByFunction
- lambda:RemovePermission
- s3:ListBuckets

AWS CloudWatch Logs에 대한 관리형(미리 정의된) 정책

AWS 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 관리형 정책은 필요한 권한을 사용자가 조사할 필요가 없도록 일반 사용 사례에 필요한 권한을 부여합니다. 자세한 내용은 IAM 사용자 안내서의 AWS 관리형 정책을 참조하세요.

계정의 사용자 및 역할에 연결할 수 있는 다음 AWS 관리형 정책은 CloudWatch Logs에 고유합니다.

- CloudWatchLogsFullAccess – CloudWatch Logs에 대한 전체 액세스 권한을 부여합니다.
- CloudWatchLogsReadOnlyAccess – CloudWatch Logs에 대한 읽기 전용 액세스 권한을 부여합니다.

CloudWatchLogsFullAccess

CloudWatchLogsFullAccess 정책은 CloudWatch Logs에 대한 전체 액세스 권한을 부여합니다. 이 정책에는 `cloudwatch:GenerateQuery` 및 `cloudwatch:GenerateQueryResultsSummary` 권한이 포함되어 있으므로 이 정책을 사용하는 사용자는 자연어 프롬프트에서 CloudWatch Logs Insights 쿼리 문자열을 생성할 수 있습니다. 정책의 전체 내용을 보려면 AWS 관리형 정책 참조 가이드의 CloudWatchLogsFullAccess를 참조하세요.

CloudWatchLogsReadOnlyAccess

CloudWatchLogsReadOnlyAccess 정책은 CloudWatch Logs에 대한 읽기 전용 액세스 권한을 부여합니다. 이 정책은 `cloudwatch:GenerateQuery` 및 `cloudwatch:GenerateQueryResultsSummary` 권한을 포함하고 있으므로 이 정책은 자연 언어 프롬프트에서 CloudWatch Logs Insights을 생성할 수 있습니다. 정책의 전체 내용을 보려면 AWS 관리형 정책 참조 가이드의 CloudWatchLogsReadOnlyAccess를 참조하세요.

CloudWatchOpenSearchDashboardsFullAccess

CloudWatchOpenSearchDashboardsFullAccess 정책은 OpenSearch Service와의 통합을 생성, 관리 및 삭제하고 해당 통합에서 판매 로그 대시보드를 생성 및 관리할 수 있는 액세스 권한을 부여합니다.

정책의 전체 내용을 보려면 AWS 관리형 정책 참조 가이드의 CloudWatchOpenSearchDashboardsFullAccess를 참조하세요.

CloudWatchOpenSearchDashboardAccess

CloudWatchOpenSearchDashboardAccess 정책은 Amazon OpenSearch Service 분석으로 생성된 판매 로그 대시보드를 볼 수 있는 액세스 권한을 부여합니다.

정책의 전체 내용을 보려면 AWS 관리형 정책 참조 가이드의 CloudWatchOpenSearchDashboardAccess를 참조하세요.

CloudWatchLogsCrossAccountSharingConfiguration

CloudWatchLogsCrossAccountSharingConfiguration 정책은 계정 간에 CloudWatch Logs 리소스를 공유하기 위해 Observability Access Manager 링크를 생성 및 관리하고 볼 수 있는 액세스 권한을 부여합니다. 자세한 내용은 CloudWatch 크로스 계정 관찰성을 참조하세요.

정책의 전체 내용을 보려면 AWS 관리형 정책 참조 가이드의 CloudWatchLogsCrossAccountSharingConfiguration를 참조하세요.

CloudWatchLogsAPIKeyAccess

CloudWatchLogsAPIKeyAccess 정책은 CloudWatch Logs API 키 인증 및 암호화된 로그 수집을 활성화합니다. 이 정책은 보유자 토큰을 사용하여 인증하고 로그 이벤트를 CloudWatch Logs에 쓸 수 있는 권한을 부여하며, 로그가 암호화될 때 데이터 키를 해독하고 생성할 수 있는 추가 AWS KMS 권한을 부여합니다.

이 정책은 다음 권한을 부여합니다.

- `logs` - 보안 주체가 API 키 보유자 토큰을 통해 인증하고 CloudWatch Logs 스트림에 로그 이벤트를 쓸 수 있도록 허용합니다.
- `kms` - 보안 주체가 AWS KMS 키 메타데이터를 읽고, 암호화를 위한 데이터 키를 생성하고, 데이터를 복호화할 수 있도록 허용합니다. 이러한 권한은 서비스가 고객 관리형 AWS KMS 키를 사용하여 로그 데이터를 암호화할 수 있도록 하여 암호화된 CloudWatch Logs를 지원합니다. 액세스는 CloudWatch Logs 서비스를 통해 호출된 작업으로 제한됩니다.

최신 버전의 JSON 정책 문서를 포함하여 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 CloudWatchLogsAPIKeyAccess를 참조하세요.

AWS 관리형 정책에 대한 CloudWatch Logs 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 CloudWatch Logs의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 CloudWatch Logs 문서 기록 페이지에서 RSS 피드를 구독합니다.

변경	설명	Date
CloudWatchLogsAPIKeyAccess – 새 정책입니다.	CloudWatch Logs에 새로운 관리형 정책 CloudWatchLogsAPIKeyAccess가 추가되었습니다. 이 정책은 CloudWatch Logs API 키 인증 및 암호화된 로그 수집을 활성화하여 보유자 토큰을 사용하여 인증하고 CloudWatch Logs에 로그 이벤트를 쓸 수 있는 권한을 부여합니다.	2026년 2월 17일
CloudWatchLogsFullAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsFullAccess에 대한 권한을 추가했습니다.	2025년 12월 2일

변경	설명	Date
	원격 분석 파이프라인 및 S3 테이블 통합에 대한 읽기 전용 액세스를 허용할 수 있도록 관찰성 관리 작업에 대한 권한이 추가되었습니다.	
CloudWatchLogsReadOnlyAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsReadOnlyAccess에 대한 권한을 추가했습니다. 원격 분석 파이프라인 및 S3 테이블 통합에 대한 읽기 전용 액세스를 허용할 수 있도록 관찰성 관리 작업에 대한 권한이 추가되었습니다.	2025년 12월 2일
CloudWatchLogsFullAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsFullAccess에 대한 권한을 추가했습니다. `cloudwatch:GenerateQueryResultsSummary` 에 대한 권한이 쿼리 결과의 자연어 요약을 생성할 수 있도록 추가되었습니다.	2025년 5월 20일

변경	설명	Date
CloudWatchLogsReadOnlyAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsReadOnlyAccess에 대한 권한을 추가했습니다. `cloudwatch:GenerateQueryResultsSummary` 에 대한 권한이 쿼리 결과의 자연어 요약을 생성할 수 있도록 추가되었습니다.	2025년 5월 20일
CloudWatchLogsFullAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsFullAccess에 대한 권한을 추가했습니다. 일부 기능에 대해 OpenSearch Service와의 CloudWatch Logs 통합을 활성화하기 위해 Amazon OpenSearch Service 및 IAM에 대한 권한이 추가되었습니다.	2024년 12월 1일
CloudWatchOpenSearchDashboardsFullAccess – 새 IAM 정책.	CloudWatch Logs에서 새 IAM 정책인 CloudWatchOpenSearchDashboardsFullAccess를 추가했습니다. 이 정책은 OpenSearch Service와의 통합을 생성, 관리 및 삭제하고 해당 통합에서 판매 로그 대시보드를 생성, 관리 및 삭제할 수 있는 액세스 권한을 부여합니다.	2024년 12월 1일

변경	설명	Date
CloudWatchOpenSearchDashboardAccess – 새 IAM 정책	CloudWatch Logs에서 새 IAM 정책인 CloudWatchOpenSearchDashboardAccess를 추가했습니다. 이 정책은 Amazon OpenSearch Service로 구동되는 판매 로그 대시보드를 볼 수 있는 액세스 권한을 부여합니다.	2024년 12월 1일
CloudWatchLogsFullAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsFullAccess에 대한 권한을 추가했습니다. `cloudwatch:GenerateQuery` 권한이 추가되었으므로 이 정책이 적용되는 사용자는 자연어 프롬프트에서 CloudWatch Logs Insights 쿼리 문자열을 생성할 수 있습니다.	2023년 11월 27일
CloudWatchLogsReadOnlyAccess - 기존 정책에 대한 업데이트	CloudWatch에서 CloudWatchLogsReadOnlyAccess에 대한 권한을 추가했습니다. `cloudwatch:GenerateQuery` 권한이 추가되었으므로 이 정책이 적용되는 사용자는 자연어 프롬프트에서 CloudWatch Logs Insights 쿼리 문자열을 생성할 수 있습니다.	2023년 11월 27일

변경	설명	Date
CloudWatchLogsReadOnlyAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsReadOnlyAccess에 대한 권한을 추가했습니다. 이 정책이 있는 사용자가 콘솔을 사용하여 CloudWatch Logs Live Tail 세션을 시작하고 중지할 수 있도록 `logs:StartLiveTail` 및 `logs:StopLiveTail` 권한이 추가되었습니다. 자세한 내용은 Live Tail을 사용하여 거의 실시간으로 로그 보기를 참조하세요.	2023년 6월 6일
CloudWatchLogsCrossAccountSharingConfiguration - 새 정책	CloudWatch Logs에서 CloudWatch Logs 로그 그룹을 공유하는 CloudWatch 교차 계정 관측성 링크를 관리할 수 있는 새 정책을 추가했습니다. 자세한 내용은 CloudWatch 크로스 계정 관찰성을 참조하세요.	2022년 11월 27일

변경	설명	Date
CloudWatchLogsReadOnlyAccess - 기존 정책에 대한 업데이트	CloudWatch Logs에서 CloudWatchLogsReadOnlyAccess에 대한 권한을 추가했습니다. 이 정책이 적용되는 사용자가 콘솔을 사용하여 CloudWatch 크로스 계정 관찰성에서 소스 계정의 공유된 데이터를 볼 수 있도록 여기에는 `oam:ListSinks` 및 `oam:ListAttachedLinks` 권한이 포함되었습니다.	2022년 11월 27일

고객 관리형 정책 예제

CloudWatch Logs 작업 및 리소스에 대한 권한을 허용하는 고유의 사용자 지정 IAM 정책을 생성할 수도 있습니다. 해당 권한이 필요한 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

이 섹션에서는 다양한 CloudWatch Logs 작업에 대한 권한을 부여하는 사용자 정책의 예제를 제공합니다. 이러한 정책은 CloudWatch Logs API, AWS SDK 또는 AWS CLI를 사용하는 경우에 유효합니다.

예제

- 예제 1: CloudWatch Logs에 대한 전체 액세스 허용
- 예제 2: CloudWatch Logs에 대한 읽기 전용 액세스 허용
- 예제 3: 하나의 로그 그룹/로그 스트림에 대한 액세스 허용

예제 1: CloudWatch Logs에 대한 전체 액세스 허용

다음 정책은 사용자가 모든 CloudWatch Logs 작업에 액세스할 수 있도록 허용합니다.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

예제 2: CloudWatch Logs에 대한 읽기 전용 액세스 허용

AWS 는 CloudWatch Logs 데이터에 대한 읽기 전용 액세스를 활성화하는 CloudWatchLogsReadOnlyAccess 정책을 제공합니다. CloudWatch 이 정책에는 다음 권한이 포함되어 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:Describe*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "logs:StartLiveTail",
                "logs:StopLiveTail",
                "cloudwatch:GenerateQuery"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

예제 3: 하나의 로그 그룹/로그 스트림에 대한 액세스 허용

CloudWatch Logs에는 서로 다른 ARN 형식의 두 가지 리소스 유형이 있습니다.

- 로그 그룹: `arn:aws:logs:region:account:log-group:LogGroupName`
- 로그 스트림: `arn:aws:logs:region:account:log-group:LogGroupName:log-stream:StreamName`

IAM 정책을 작성할 때 사용하는 ARN 형식은 API가 권한을 부여하는 리소스 유형과 일치해야 합니다. 각 API에 필요한 리소스 유형에 대한 서비스 권한 부여 참조를 참조하세요.

예제 3a: 특정 log-group-level 작업에 대한 액세스 허용

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
      "Action": [
        "logs:DeleteLogGroup",
        "logs:PutRetentionPolicy",
        "logs:PutSubscriptionFilter",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName"
      }
   ]
}
```

이러한 작업은 `log-group` 리소스 유형에 대한 권한을 부여합니다. 표준 ARN 형식(제외:*)이 지원됩니다.

예제 3b: 특정 log-stream-level 작업에 대한 액세스 허용

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
```

```
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName:*"
      }
   ]
}
```

이러한 작업은 log-stream 리소스 유형에 대한 권한을 부여합니다. :* 접미사는 로그 그룹 내의 모든 로그 스트림을 일치시키거나를 사용하여 특정 스트림을 지정하는 데 필요합니다:log-stream:*StreamName*.

예제 3c: 로그 그룹 및 로그 스트림 작업 모두에 대한 통합 정책

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
        "Action": [
          "logs:DeleteLogGroup",
          "logs:PutRetentionPolicy",
          "logs:DescribeLogStreams",
          "logs:FilterLogEvents"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName"
      },
      {
        "Action": [
          "logs:CreateLogStream",
          "logs:PutLogEvents",
          "logs:GetLogEvents"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:logs:us-west-2:123456789012:log-
group:SampleLogGroupName:*"
      }
   ]
}
```

로그 그룹 수준의 제어를 위한 태깅 및 IAM 정책 사용

사용자에게 특정 로그 그룹에 대한 액세스 권한을 부여함과 동시에 다른 로그 그룹에 대한 액세스를 방지할 수 있습니다. 이렇게 하려면 로그 그룹에 태그를 지정하고 해당 태그를 참조하는 IAM 정책을 사용

하세요. 로그 그룹에 태그를 적용하려면 `logs:TagResource` 또는 `logs:TagLogGroup` 권한이 있어야 합니다. 이는 로그 그룹을 생성할 때 로그 그룹에 태그를 할당하거나 나중에 할당하는 경우 모두에 적용됩니다.

로그 그룹의 태깅에 대한 자세한 내용은 Amazon CloudWatch Logs의 로그 그룹에 태그 지정 섹션을 참조하세요.

로그 그룹에 태그를 지정할 때 특정 태그가 있는 로그 그룹에만 액세스를 허용할 수 있도록 사용자에게 IAM 정책에 대한 권한을 부여할 수 있습니다. 예를 들어, 다음 정책 명령문은 태그 키 `Team`에 대한 값이 `Green`인 로그 그룹에만 액세스하도록 권한을 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/Team": "Green"
                }
            }
        }
    ]
}
```

StopQuery 및 StopLiveTail API 작업은 기존 방식으로 AWS 리소스와 상호 작용하지 않습니다. 즉, 어떤 식으로든 데이터를 반환하거나 데이터를 넣거나 리소스를 수정하지 않습니다. 대신, 리소스로 분류되지 않는 지정된 Live Tail 세션 또는 지정된 CloudWatch 로그 인사이트 쿼리에서만 작동합니다. 따라서 이러한 작업에 대해 IAM 정책에서 `Resource` 필드를 지정하는 경우 다음 예제와 같이 `Resource` 필드 값을 `*`로 설정해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
        [ {
            "Effect": "Allow",
            "Action": [
                "logs:StopQuery",
                "logs:StopLiveTail"
            ],
            "Resource": "*"
            }
        ]
}
```

IAM 정책 설명 사용에 대한 자세한 내용은 IAM User Guide의 Controlling Access Using Policies를 참조하세요.

CloudWatch Logs 권한 참조

IAM ID에 연결할 수 있는 액세스 관리 및 쓰기 권한 정책(ID 기반 정책)을 설정할 때 다음 표를 참조로 사용할 수 있습니다. 표에는 각 CloudWatch Logs API 작업과 이 작업을 수행할 수 있는 권한을 부여할 수 있는 작업이 나와 있습니다. 정책의 `Action` 필드에 작업을 지정합니다. `Resource` 필드에서는 로그 그룹 또는 로그 스트림의 ARN을 지정하거나 *를 지정하여 모든 CloudWatch Logs 리소스를 대표할 수 있습니다.

CloudWatch Logs 정책에서 AWS전체 조건 키를 사용하여 조건을 표시할 수 있습니다. AWS전체 키의 전체 목록은 AWS IAM 사용 설명서의 전역 및 IAM 조건 컨텍스트 키를 참조하세요.

> **Note**
>
> 작업을 지정하려면 `logs:` 접두사 다음에 API 작업 이름을 사용합니다. 예:`logs:CreateLogGroup`,`logs:CreateLogStream` 또는 `logs:*`(모든 CloudWatch Logs 작업의 경우).

CloudWatch Logs API 작업 및 작업에 필요한 권한

CloudWatch Logs API 작업	필요한 권한(API 작업)
CancelExportTask	`logs:CancelExportTask` 보류 또는 실행 중인 내보내기 작업을 취소하는 데 필요합니다.
CreateExportTask	`logs:CreateExportTask` 로그 그룹에서 Amazon S3 버킷으로 데이터를 내보내는 데 필요합니다.
CreateLogGroup	`logs:CreateLogGroup` 새 보안 그룹을 생성하는 데 필요합니다.
CreateLogStream	`logs:CreateLogStream` 로그 그룹에서 로그 스트림을 새로 생성하는 데 필요합니다.
DeleteDestination	`logs:DeleteDestination` 로그 대상을 삭제하고 이에 대한 모든 구독 필터를 비활성화하는 데 필요합니다.
DeleteLogGroup	`logs:DeleteLogGroup` 로그 그룹과 보관되는 모든 연관 로그 이벤트를 삭제하는 데 필요합니다.
DeleteLogStream	`logs:DeleteLogStream`

CloudWatch Logs API 작업	필요한 권한(API 작업)
	로그 스트림과 보관되는 모든 연관 로그 이벤트를 삭제하는 데 필요합니다.
DeleteMetricFilter	`logs:DeleteMetricFilter` 로그 그룹과 연관된 지표 필터를 삭제하는 데 필요합니다.
DeleteQueryDefinition	`logs:DeleteQueryDefinition` CloudWatch Logs Insights에서 저장된 쿼리 정의를 삭제하는 데 필요합니다.
DeleteResourcePolicy	`logs:DeleteResourcePolicy` CloudWatch Logs 리소스 정책을 삭제하는 데 필요합니다.
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` 로그 그룹의 보존 정책을 삭제하는 데 필요합니다.
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` 로그 그룹과 연관된 구독 필터를 삭제하는 데 필요합니다.
DescribeDestinations	`logs:DescribeDestinations` 계정과 연결된 모든 대상을 확인하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
DescribeExportTasks	`logs:DescribeExportTasks` 계정과 연관된 모든 내보내기 작업을 확인하는 데 필요합니다.
DescribeLogGroups	`logs:DescribeLogGroups` 계정과 연관된 모든 로그 그룹을 확인하는 데 필요합니다.
DescribeLogStreams	`logs:DescribeLogStreams` 로그 그룹과 연관된 모든 로그 스트림을 확인하는 데 필요합니다.
DescribeMetricFilters	`logs:DescribeMetricFilters` 로그 그룹과 연관된 모든 지표를 보는 데 필요합니다.
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` CloudWatch Logs Insights에 저장된 쿼리 정의의 목록을 확인하는 데 필요합니다.
DescribeQueries	`logs:DescribeQueries` 예약되었거나 실행 중이거나 최근에 실행된 CloudWatch Logs Insights 쿼리 목록을 확인하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
DescribeResourcePolicies	`logs:DescribeResourcePolicies` CloudWatch Logs 리소스 정책 목록을 확인하는 데 필요합니다.
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` 로그 그룹과 연관된 모든 구독 필터를 확인하는 데 필요합니다.
FilterLogEvents	`logs:FilterLogEvents` 로그 그룹 필터 패턴에 따라 로그 이벤트를 정렬하는 데 필요합니다.
GetLogEvents	`logs:GetLogEvents` 로그 스트림에서 로그 이벤트를 검색하는 데 필요합니다.
GetLogGroupFields	`logs:GetLogGroupFields` 로그 그룹의 로그 이벤트에 포함된 필드 목록을 검색하는 데 필요합니다.
GetLogRecord	`logs:GetLogRecord` 단일 로그 이벤트에서 세부 정보를 검색하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
GetLogObject	`logs:GetLogRecord` PutOpenTelemetryLogs API를 통해 수집된 로그 이벤트의 많은 부분을 가져오는 데 필요합니다.
GetQueryResults	`logs:GetQueryResults` CloudWatch Logs Insights 쿼리 결과를 검색하는 데 필요합니다.
ListEntitiesForLogGroup (CloudWatch 콘솔 전용 권한)	`logs:ListEntitiesForLogGroup` 로그 그룹과 연관된 엔터티를 찾는 데 필요합니다. CloudWatch 콘솔 내 관련 로그를 살펴보는 데 필요합니다.
ListLogGroupsForEntity (CloudWatch 콘솔 전용 권한)	`logs:ListLogGroupsForEntity` 엔터티와 연결된 로그 그룹을 찾는 데 필요합니다. CloudWatch 콘솔 내 관련 로그를 살펴보는 데 필요합니다.
ListTagsLogGroup	`logs:ListTagsLogGroup` 로그 그룹과 연결된 태그를 나열하는 데 필요합니다.
ListLogGroups	`logs:DescribeLogGroups` 계정과 연관된 모든 로그 그룹을 확인하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
ProcessWithPipeline(권한만 해당)	`logs:ProcessWithPipeline` 저장하기 전에 파이프라인 변환기를 통해 로그 이벤트를 처리하고 변환하는 데 필요합니다. 파이프라인 구성에서 CloudWatch Logs 소스 역할로 전달된 IAM 역할에 대해이 권한을 부여해야 합니다. 리소스는 로그 그룹 ARN입니다.
PutDestination	`logs:PutDestination` 대상 로그 스트림(예: Kinesis 스트림)을 생성하거나 업데이트하는 데 필요합니다.
PutDestinationPolicy	`logs:PutDestinationPolicy` 기존 로그 대상과 연관된 액세스 정책을 생성 또는 업데이트하는 데 필요합니다.
PutLogEvents	`logs:PutLogEvents` 로그 스트림에서 로그 이벤트 배치를 업로드하는 데 필요합니다.
PutMetricFilter	`logs:PutMetricFilter` 지표 필터를 생성 또는 업데이트하고 이를 로그 그룹과 연관시키는 데 필요합니다.
PutQueryDefinition	`logs:PutQueryDefinition` 파라미터가 있는 저장된 쿼리를 포함하여 CloudWatch Logs Insights에 쿼리를 저장하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
PutResourcePolicy	`logs:PutResourcePolicy` CloudWatch Logs 리소스 정책을 생성하는 데 필요합니다.
PutRetentionPolicy	`logs:PutRetentionPolicy` 로그 그룹에 로그 이벤트를 유지하는 일수(보존 일수)를 설정하는 데 필요합니다.
PutSubscriptionFilter	`logs:PutSubscriptionFilter` 구독 필터를 생성 또는 업데이트하고 이를 로그 그룹과 연관시키는 데 필요합니다.
StartQuery	`logs:StartQuery` CloudWatch Logs Insights 쿼리를 시작하는 데 필요합니다. 파라미터로 저장된 쿼리를 실행하려면 도 필요합니다`logs:DescribeQueryDefinitions` .
StopQuery	`logs:StopQuery` 진행 중인 CloudWatch Logs Insights 쿼리를 중지하는 데 필요합니다.
TagLogGroup	`logs:TagLogGroup` 로그 그룹 태그를 추가하거나 업데이트하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
TestMetricFilter	`logs:TestMetricFilter` 로그 이벤트 메시지 샘플을 기준으로 필터 패턴을 테스트하는 데 필요합니다.

CloudWatch Logs에 서비스 연결 역할 사용

Amazon CloudWatch Logs는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 CloudWatch Logs에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 CloudWatch Logs에서 사전 정의하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 CloudWatch Logs를 더 효율적으로 설정할 수 있습니다. CloudWatch Logs에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, CloudWatch Logs만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함됩니다. 이 권한 정책은 다른 어떤 IAM 엔터티에도 연결할 수 없습니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

CloudWatch Logs에 대한 서비스 연결 역할 권한

CloudWatch Logs에서는 AWSServiceRoleForLogDelivery인 서비스 연결 역할을 사용합니다. CloudWatch Logs는 이 서비스 연결 역할을 사용하여 Firehose에 로그를 직접 씁니다. 자세한 내용은 AWS 서비스에서 로깅 활성화 단원을 참조하십시오.

AWSServiceRoleForLogDelivery 서비스 연결 역할은 역할을 위임하기 위해 다음 서비스를 신뢰합니다.

- `logs.amazonaws.com`

역할 권한 정책인를 AWSServiceRoleForLogDeliveryPolicy사용하면 CloudWatch Logs가 지정된 리소스에서 다음 작업을 완료할 수 있습니다. 전체 JSON 정책 문서는 AWS 관리형 정책 참조 안내서의 AWSServiceRoleForLogDeliveryPolicy를 참조하세요.

- 작업: 값이 인 LogDeliveryEnabled 키가 있는 태그가 있는 모든 Firehose 전송 스트림firehose:ListTagsForDeliveryStream에서 firehose:PutRecordfirehose:PutRecordBatch, 및 . true 이 태그는 구독을 생성하여 Firehose에 로그를 전송할 때 Firehose 전송 스트림에 자동으로 연결됩니다.
- 작업: Firehose를 통해 요청이 이루어진 경우에만 모든 AWS KMS 키kms:Decrypt에 대해 kms:GenerateDataKey 및가 적용됩니다(로 설정된 kms:ViaService 조건 키에 의해 적용됨firehose.*.amazonaws.com). 이러한 권한을 통해 CloudWatch Logs는 고객 관리형 키를 사용한 서버 측 암호화(SSE-CMK)를 사용하는 Firehose 전송 스트림에 로그를 전송할 수 있습니다. 또한 고객의 AWS KMS 키 정책은 서비스 연결 역할에 대한 액세스 권한을 독립적으로 부여해야 합니다.

IAM 엔터티가 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 권한을 구성해야 합니다. 이 엔터티는 사용자, 그룹 또는 역할일 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

CloudWatch Logs에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. 로그를 AWS Management Console AWS CLI, 또는 AWS API의 Firehose 스트림으로 직접 전송하도록 설정하면 CloudWatch Logs가 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. Firehose 스트림으로 직접 전송되도록 로그를 다시 설정하면 CloudWatch Logs는 서비스 연결 역할을 자동으로 다시 생성합니다.

CloudWatch Logs에 대한 서비스 연결 역할 편집

CloudWatch Logs는 생성한 후 AWSServiceRoleForLogDelivery 또는 다른 서비스 연결 역할의 편집을 허용하지 않습니다. 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

CloudWatch Logs에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 이렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

> **Note**
>
> 리소스를 삭제하려 할 때 CloudWatch Logs 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

AWSServiceRoleForLogDelivery 서비스 연결 역할에서 사용되는 CloudWatch Logs 리소스를 에서 제거하려면 다음을 수행합니다.

- Firehose 스트림으로 직접 로그 전송을 중지합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 AWSServiceRoleForLogDelivery 서비스 연결 역할을 삭제합니다. 자세한 내용은 서비스 연결 역할 삭제를 참조하세요.

CloudWatch Logs 서비스 연결 역할을 지원하는 리전

CloudWatch Logs는 서비스를 사용할 수 있는 모든 AWS 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 CloudWatch Logs 리전 및 엔드포인트를 참조하세요.

AWS 서비스 연결 역할에 대한 CloudWatch Logs 업데이트

이 AWS 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 CloudWatch Logs의 서비스 연결 역할 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 CloudWatch Logs 문서 기록 페이지에서 RSS 피드를 구독합니다.

변경	설명	Date
AWSServiceRoleForLogDelivery 서비스 연결 역할 정책 – 기존 정책에 대한 업데이트	CloudWatch Logs가 AWSServiceRoleForLogDelivery 서비스 연결 역할과 연결된 IAM 정책에 `kms:GenerateDataKey` 및 `kms:Decrypt` 권한을 추가했습니다. 이러한 권한은 Firehose를 통해서만 사	2026년 5월 15일

변경	설명	Date
	용할 수 있도록 `kms:ViaService` 조건 키로 범위가 지정됩니다. 이 변경 사항을 통해 CloudWatch Logs는 고객 관리형 키를 사용한 서버 측 암호화(SSE-CMK)를 사용하는 Firehose 전송 스트림에 로그를 전송할 수 있습니다.	
AWSServiceRoleForLogDelivery 서비스 연결 역할 정책 – 기존 정책에 대한 업데이트	CloudWatch Logs는 AWSServiceRoleForLogDelivery 서비스 연결 역할과 연결된 IAM 정책에서 권한을 변경했습니다. 변경 내용은 다음과 같습니다. • `firehose:ResourceTag/LogDeliveryEnabled": "true"` 조건 키가 `aws:ResourceTag/LogDeliveryEnabled": "true"`로 변경되었습니다.	2021년 7월 15일
CloudWatch Logs 변경 내용 추적 시작	CloudWatch Logs가 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 6월 10일

Amazon CloudWatch Logs 규정 준수 검증

새로운 AWS 가입 환경은 규제 워크로드용으로 설계되지 않았습니다. 새로운 AWS 가입 환경을 사용하고 있지만 규제 대상 워크로드 AWS 에를 사용하려는 경우 AWS 환경에 가입 AWS (고급)하거나 고급 기능을 활성화할 수 있습니다.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 제공 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

Amazon CloudWatch Logs의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며, 이러한 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

Amazon CloudWatch Logs의 인프라 보안

관리형 서비스인 Amazon CloudWatch Logs는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 CloudWatch Logs에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.

- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

인터페이스 VPC 엔드포인트에서 CloudWatch Logs 사용

Amazon Virtual Private Cloud(VPC)를 사용하여 AWS 리소스를 호스팅하는 경우 VPC와 CloudWatch Logs 간에 프라이빗 연결을 설정할 수 있습니다. 이 연결을 사용하여 인터넷을 통하지 않고 CloudWatch Logs에 로그를 보낼 수 있습니다. CloudWatch Logs는 모든 리전에서 IPv4 VPC 엔드포인트를 지원하고 모든 리전에서 IPv6 엔드포인트를 지원합니다.

Amazon VPC는 정의한 가상 네트워크에서 AWS 리소스를 시작하는 데 사용할 수 있는 AWS 서비스입니다. VPC가 있으면 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등 네트워크 설정을 제어할 수 있습니다. VPC를 CloudWatch Logs에 연결하려면 CloudWatch Logs용 인터페이스 VPC 엔드포인트에 대해 알아보세요. 이 유형의 엔드포인트를 사용하여 VPC를 AWS 서비스에 연결할 수 있습니다. 이 엔드포인트를 이용하면 인터넷 게이트웨이나 NAT(네트워크 주소 변환) 인스턴스 또는 VPN 연결 없이도 CloudWatch Logs에 안정적이고 확장할 수 있게 연결됩니다. 자세한 내용은 VPC 사용 설명서의 VPC란 무엇입니까? 섹션을 참조하세요.

인터페이스 VPC 엔드포인트는 프라이빗 IP 주소가 있는 탄력적 네트워크 인터페이스를 사용하여 AWS 서비스 간의 프라이빗 통신을 지원하는 AWS 기술인 AWS PrivateLink로 구동됩니다. 자세한 내용은 New – AWS PrivateLink for AWS Services를 참조하세요.

다음은 Amazon VPC 사용자를 위한 단계들입니다. 자세한 내용은 Amazon VPC 사용 설명서의 시작하기를 참조하세요.

가용성

CloudWatch Logs는 현재 AWS 리전을 포함한 모든 AWS GovCloud (US) 리전에서 VPC 엔드포인트를 지원합니다.

CloudWatch Logs에 대한 VPC 엔드포인트 생성

VPC에서 CloudWatch Logs를 사용하려면 CloudWatch Logs에 대한 인터페이스 VPC 엔드포인트를 생성합니다. 선택할 서비스 이름은 com.amazonaws.***Region***.logs입니다. FIPS 엔드포인트에 연결하려면 선택할 서비스는 입니다`com.amazonaws.Region.logs-fips`. CloudWatch Logs에 대한 설정을 변경할 필요가 없습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

StartLiveTail 및 GetLogObject와 같은 일부 CloudWatch Logs APIs는 다른 엔드포인트 및 VPC 엔드포인트에서 호스팅됩니다`stream-logs.Region.amazonaws.com`. 이러한 APIs에 대한 인터페이스 VPC 엔드포인트를 생성하기 위해 선택할 서비스는 com.amazonaws.***Region***.stream-logs입니다. FIPS 엔드포인트에 연결하려면 선택할 서비스는 입니다`com.amazonaws.Region.stream-logs-fips`.

VPC와 CloudWatch Logs 간 연결 테스트

엔드포인트를 생성한 후에 연결을 테스트할 수 있습니다.

VPC와 CloudWatch Logs 엔드포인트 간 연결을 테스트하려면

1. VPC에 있는 Amazon EC2 인스턴스에 연결합니다. 연결에 대한 자세한 내용은 Amazon EC2 설명서의 Linux 인스턴스에 연결 또는 Windows 인스턴스에 연결을 참조하세요.
2. 인스턴스에서 AWS CLI 를 사용하여 기존 로그 그룹 중 하나에 로그 항목을 생성합니다.

   먼저 로그 이벤트 JSON 파일을 생성합니다. 타임스탬프는 1970년 1월 1일 1:1970 00:00:00 UTC 이후 경과된 시간(밀리초)으로 지정되어야 합니다.

   ```
   [
     {
       "timestamp": 1533854071310,
       "message": "VPC Connection Test"
     }
   ]
   ```

   그런 다음 `put-log-events` 명령을 사용하여 로그 항목을 생성합니다.

   ```
   aws logs put-log-events --log-group-name LogGroupName --log-stream-
   name LogStreamName --log-events file://JSONFileName
   ```

   명령이 성공해 VPC 엔드포인트를 사용할 수 있는 상태가 되었다면 명령에 대한 응답에 `nextSequenceToken` 명령이 포함됩니다.

CloudWatch Logs VPC 엔드포인트에 대한 액세스 제어

VPC 엔드포인트 정책은 엔드포인트를 만들거나 수정 시 엔드포인트에 연결하는 IAM 리소스 정책입니다. 엔드포인트를 만들 때 정책을 추가하지 않으면 서비스에 대한 모든 액세스를 허용하는 기본 정책이

추가됩니다. 엔드포인트 정책은 IAM 정책 또는 서비스별 정책을 재정의하거나 대체하지 않습니다. 이는 엔드포인트에서 지정된 서비스로의 액세스를 제어하기 위한 별도의 정책입니다.

엔드포인트 정책은 JSON 형식으로 작성해야 합니다.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

다음은 CloudWatch Logs에 대한 엔드포인트 정책의 예입니다. 이 정책에서는 사용자가 VPC를 통해 CloudWatch Logs에 연결하여 로그 스트림을 생성하고 CloudWatch Logs에 로그를 전송할 수 있도록 허용하고, 다른 CloudWatch Logs 작업을 수행하지 못하게 금지합니다.

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

CloudWatch Logs에 대한 VPC 엔드포인트 정책을 수정하려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. CloudWatch Logs에 대한 엔드포인트를 아직 생성하지 않았다면 엔드포인트 생성을 선택합니다. 그런 다음 com.amazonaws.***Region***.logs를 선택하고 엔드포인트 생성을 선택합니다.
4. com.amazonaws.***Region***.logs 엔드포인트를 선택하고 화면 하단의 정책 탭을 선택합니다.
5. 정책 편집을 선택하고 정책을 변경합니다.

VPC 컨텍스트 키에 대한 지원

CloudWatch Logs는 특정 VPC 또는 특정 VPC 엔드포인트에 대한 액세스를 제한할 수 있는 `aws:SourceVpc` 및 `aws:SourceVpce` 컨텍스트 키를 지원합니다. 이러한 키는 사용자가 VPC 엔드포인트를 사용하고 있을 때만 작동합니다. 자세한 내용은 IAM 사용 설명서의 일부 서비스에 사용 가능한 키를 참조하세요.

에서 CloudWatch Logs API 및 콘솔 작업 로깅 AWS CloudTrail

Amazon CloudWatch Logs는 사용자, 역할, 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 CloudWatch Logs에 대한 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 CloudWatch Logs 콘솔로부터의 호출과 CloudWatch Logs API 작업에 대한 코드 호출이 포함됩니다. CloudTrail이 수집한 정보를 사용하여 CloudWatch Logs에 수행된 요청, 요청이 수행된 IP 주소, 요청이 수행된 시간, 추가 세부 정보를 확인할 수 있습니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부.
- IAM Identity Center 사용자를 대신하여 요청이 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화되며 CloudTrail 이벤트 기록에 자동으로 액세스할 수 있습니다. CloudTrail 이벤트 기록은 지난 90일 간 AWS 리전의 관리 이벤트에 대해 보기, 검색 및 다운로드가 가능하고, 수정이 불가능한 레코드를 제공합니다. 자세한 설명은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 기록 작업을 참조하세요. 이벤트 기록 보기는 CloudTrail 요금이 부과되지 않습니다.

AWS 계정 지난 90일 동안의 이벤트를 지속적으로 기록하려면 추적 또는 CloudTrail Lake 이벤트 데이터 스토어를 생성합니다.

CloudTrail 추적

CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 를 사용하여 생성된 모든 추적 AWS Management Console 은 다중 리전입니다. AWS CLI를 사용하여 단일 리전 또는 다중 리전 추적을 생성할 수 있습니다. 계정의 모든 AWS 리전 에서 활동을 캡처하므로 다중 리전 추적을 생성하는 것이 좋습니다. 단일 리전 추적을 생성하는 경우 추적의 AWS 리전에 로깅된 이벤트만 볼 수 있습니다. 추적에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS 계정에 대한 추적 생성 및 조직에 대한 추적 생성을 참조하세요.

CloudTrail에서 추적을 생성하여 진행 중인 관리 이벤트의 사본 하나를 Amazon S3 버킷으로 무료로 전송할 수는 있지만, Amazon S3 스토리지 요금이 부과됩니다. CloudTrail 요금에 관한 자세한

내용은 AWS CloudTrail 요금을 참조하세요. Amazon S3 요금에 관한 자세한 내용은 Amazon S3 요금을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어

CloudTrail Lake를 사용하면 이벤트에 대해 SQL 기반 쿼리를 실행할 수 있습니다. CloudTrail Lake는 행 기반 JSON 형식의 기존 이벤트를 Apache ORC 형식으로 변환합니다. ORC는 빠른 데이터 검색에 최적화된 열 기반 스토리지 형식입니다. 이벤트는 이벤트 데이터 스토어로 집계되며, 이벤트 데이터 스토어는 고급 이벤트 선택기를 적용하여 선택한 기준을 기반으로 하는 변경 불가능한 이벤트 컬렉션입니다. 이벤트 데이터 스토어에 적용하는 선택기는 어떤 이벤트가 지속되고 쿼리에 사용 가능한지를 제어합니다. CloudTrail Lake에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail Lake 작업을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어 및 쿼리에는 비용이 발생합니다. 이벤트 데이터 스토어를 생성할 때 이벤트 데이터 스토어에 사용할 요금 옵션을 선택합니다. 요금 옵션에 따라 이벤트 모으기 및 저장 비용과 이벤트 데이터 스토어의 기본 및 최대 보존 기간이 결정됩니다. CloudTrail 요금에 관한 자세한 내용은 AWS CloudTrail 요금을 참조하세요.

CloudWatch Logs에서는 CloudTrail 로그 파일에 다음 작업을 이벤트로 로그할 수 있습니다.

- AssociateKmsKey
- CancelExportTask
- CreateDelivery
- CreateExportTask
- CreateLogAnomalyDetector
- CreateLogGroup
- CreateLogStream
- DeleteAccountPolicy
- DeleteDataProtectionPolicy
- DeleteDelivery
- DeleteDeliveryDestination
- DeleteDeliveryDestinationPolicy
- DeleteDeliverySource
- DeleteDestination

- DeleteIndexPolicy
- DeleteIntegration
- DeleteLogAnomalyDetector
- DeleteLogGroup
- DeleteLogStream
- DeleteMetricFilter
- DeleteQueryDefinition
- DeleteResourcePolicy
- DeleteRetentionPolicy
- DeleteSubscriptionFilter
- DeleteTransformer
- DescribeAccountPolicies
- DescribeConfigurationTemplates
- DescribeDeliveries
- DescribeDeliveryDestinations
- DescribeDeliverySources
- DescribeDestinations
- DescribeExportTasks
- DescribeFieldIndexes
- DescribeIndexPolicies
- DescribeLogGroups
- DescribeLogStreams
- DescribeMetricFilters
- DescribeQueries
- DescribeQueryDefinitions
- DescribeResourcePolicies
- DescribeSubscriptionFilters
- DisassociateKmsKey

- FilterLogEvents
- GetDataProtectionPolicy
- GetDelivery
- GetDeliveryDestination
- GetDeliveryDestinationPolicy
- GetDeliverySource
- GetIntegration
- GetLogAnomalyDetector
- GetLogEvents
- GetLogGroupFields
- GetLogRecord
- GetQueryResults
- GetTransformer
- ListAnomalies
- ListIntegrations
- ListLogAnomalyDetectors
- ListLogGroups
- ListLogGroupsForQuery
- ListTagsForResource
- ListTagsLogGroup
- PutAccountPolicy
- PutDataProtectionPolicy
- PutDeliveryDestination
- PutDeliveryDestinationPolicy
- PutDeliverySource
- PutDestination
- PutDestinationPolicy
- PutIndexPolicy
- PutIntegration

- PutMetricFilter
- PutQueryDefinition
- PutResourcePolicy
- PutRetentionPolicy
- PutSubscriptionFilter
- PutTransformer
- StartLiveTail
- StartQuery
- StopQuery
- TagResource
- TestMetricFilter
- TestTransformer
- UntagResource
- UpdateAnomaly
- UpdateDeliveryConfiguration
- UpdateLogAnomalyDetector

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 자격 증명 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

CloudTrail에서의 쿼리 생성 정보

쿼리 생성기 콘솔 이벤트에 대한 CloudTrail 로깅도 지원됩니다. 쿼리 생성기는 현재 CloudWatch Logs Insights와 CloudWatch Metric Insights에서 지원됩니다. 이러한 CloudTrail 이벤트에서 `eventSource`는 `monitoring.amazonaws.com`입니다.

다음 예제에서는 CloudWatch Logs Insights의 GenerateQuery 작업을 시연하는 CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
                "accountId": "123456789012",
                "userName": "SAMPLE_NAME"
            },
            "attributes": {
                "creationDate": "2020-04-08T21:43:24Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:30Z",
    "eventSource": "monitoring.amazonaws.com",
    "eventName": "GenerateQuery",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "exampleUserAgent",
    "requestParameters": {
        "query_ask": "***",
        "query_type": "LogsInsights",
        "logs_insights": {
            "fields": "***",
            "log_group_names": ["yourloggroup"]
        },
        "include_description": true
    },
    "responseElements": null,
    "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
    "eventID": "52723fd9-4a54-478c-ac55-1234567890",
```

```
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

아래 로그 파일 항목은 사용자가 CloudWatch Logs CreateExportTask 작업을 호출했음을 보여줍니다.

```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
```

```
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```

CloudWatch 지표를 사용한 모니터링

이 섹션의 테이블을 사용하여 Amazon CloudWatch Logs가 Amazon CloudWatch에 1분마다 보내는 지표를 검토할 수 있습니다.

CloudWatch Logs 지표

AWS/Logs 네임스페이스에는 다음과 같은 지표가 포함됩니다.

지표	설명
CallCount	계정에서 수행된 지정된 API 작업 수입니다. CallCount 는 CloudWatch Logs 서비스 사용량 지표입니다. 자세한 정보는 CloudWatch Logs 서비스 사용량 지표 섹션을 참조하세요. 적합한 차원: 클래스, 리소스, 서비스, 유형 유효한 통계: Sum 단위: 없음
DeliveryErrors	데이터를 구독 대상으로 전달할 때 CloudWatch Logs로 오류가 수신된 로그 이벤트 수입니다. 대상 서비스가 스로틀링 예외 또는 재시도 가능한 서비스 예외(예: HTTP 5xx)와 같은 재시도 가능한 오류를 반환하는 경우 CloudWatch Logs는 최대 24시간 동안 계속해서 전송을 다시 시도합니다. AccessDeniedException 또는 ResourceNotFoundException 과 같이 다시 시도할 수 없는 오류인 경우 CloudWatch Logs는 다시 전송을 시도하지 않습니다. 유효한 차원: LogGroupName, DestinationType, FilterName, PolicyLevel 유효한 통계: Sum 단위: 없음
DeliveryThrottling	데이터를 구독 대상으로 전달할 때 CloudWatch Logs에 스로틀링이 발생한 로그 이벤트 수입니다.

지표	설명
	대상 서비스가 스로틀링 예외 또는 재시도 가능한 서비스 예외(예: HTTP 5xx)와 같은 재시도 가능한 오류를 반환하는 경우 CloudWatch Logs는 최대 24시간 동안 계속해서 전송을 다시 시도합니다. `AccessDeniedException` 또는 `ResourceNotFoundException` 과 같이 다시 시도할 수 없는 오류인 경우 CloudWatch Logs는 다시 전송을 시도하지 않습니다. 유효한 차원: LogGroupName, DestinationType, FilterName, PolicyLevel 유효한 통계: Sum 단위: 없음
`EMFDisabledErrors`	로그 그룹과 일치하는 METRIC_EXTRACTION_POLICY 유형의 활성 계정 정책으로 인해 무시된 EMF 형식 로그 이벤트 수입니다. 지표 추출 정책에 대한 자세한 내용은 PutAccountPolicy를 참조하세요. 유효한 차원: `LogGroupName` 유효한 통계: Sum 단위: 없음
`EMFParsingErrors`	임베디드 지표 형식 로그를 처리하는 동안 발생한 구문 분석 오류 수입니다. 이러한 오류는 로그가 임베디드 지표 형식으로 식별되지만 올바른 형식을 따르지 않을 때 발생합니다. 임베디드 지표 형식에 대한 자세한 정보는 사양: 임베디드 지표 형식을 참조하세요. 유효한 차원: `LogGroupName` 유효한 통계: Sum 단위: 없음

지표	설명
EMFValidationErrors	임베디드 지표 형식 로그를 처리하는 동안 발생한 유효성 검사 오류 수입니다. 이러한 오류는 임베디드 지표 형식 로그 내의 지표 정의가 임베디드 지표 형식 및 MetricDatum 사양을 준수하지 않을 때 발생합니다. CloudWatch 임베디드 지표 형식에 대한 자세한 내용은 사양: 임베디드 지표 형식을 참조하세요. 데이터 유형 MetricDatum 에 대한 자세한 내용은 Amazon CloudWatch API Reference의 MetricDatum을 참조하세요. **Note** 특정 유효성 검사 오류로 인해 EMF 로그 내의 여러 지표가 게시되지 않을 수 있습니다. 예를 들어, 유효하지 않은 네임스페이스로 설정된 모든 지표는 삭제됩니다. 유효한 차원: LogGroupName 유효한 통계: Sum 단위: 없음
ErrorCount	계정에서 수행되어 오류가 발생한 API 작업 수입니다. ErrorCount 는 CloudWatch Logs 서비스 사용량 지표입니다. 자세한 정보는 CloudWatch Logs 서비스 사용량 지표 섹션을 참조하세요. 적합한 차원: 클래스, 리소스, 서비스, 유형 유효한 통계: Sum 단위: 없음
ForwardedBytes	구독 대상으로 전달되는 압축 바이트 단위의 로그 이벤트 볼륨입니다. 유효한 차원: LogGroupName, DestinationType, FilterName 유효한 통계: Sum 단위: 바이트

지표	설명
ForwardedLogEvents	구독 대상으로 전달되는 로그 이벤트 수입니다. 유효한 차원: LogGroupName, DestinationType, FilterName, PolicyLevel 유효한 통계: Sum 단위: 없음
IncomingBytes	CloudWatch Logs로 업로드되는 비압축 바이트 단위의 로그 이벤트 볼륨. LogGroupName 차원과 함께 사용하면 로그 그룹으로 업로드되는 비압축 바이트 단위의 로그 이벤트 볼륨이 됩니다. 유효한 차원: LogGroupName 유효한 통계: Sum 단위: 바이트
IncomingLogEvents	CloudWatch Logs로 업로드되는 로그 이벤트 수. LogGroupName 차원과 함께 사용하면 로그 그룹으로 업로드되는 로그 이벤트 수가 됩니다. 유효한 차원: LogGroupName 유효한 통계: Sum 단위: 없음
LogEventsWithFindings	CloudWatch Logs 데이터 보호 기능을 사용하여 감사하는 데이터 문자열과 일치하는 로그 이벤트 수입니다. 자세한 내용은 마스킹 처리를 통해 민감한 로그 데이터를 보호하도록 지원 단원을 참조하십시오. 유효한 차원: 없음 유효한 통계: Sum 단위: 없음

지표	설명
ThrottleCount	사용 할당량으로 인해 제한된 계정에서 수행된 API 작업 수. ThrottleCount 는 CloudWatch Logs 서비스 사용량 지표입니다. 자세한 정보는 CloudWatch Logs 서비스 사용량 지표 섹션을 참조하세요. 적합한 차원: 클래스, 리소스, 서비스, 유형 유효한 통계: Sum 단위: 없음

CloudWatch Logs 지표의 차원

다음 표에는 대부분의 CloudWatch Logs 지표에 사용할 수 있는 차원이 나와 있습니다.

차원	설명
LogGroupName	지표를 표시할 CloudWatch Logs 로그 그룹의 이름입니다.
DestinationType	CloudWatch Logs 데이터에 대한 구독 대상으로, AWS Lambda, Amazon Kinesis Data Streams 또는 Amazon Data Firehose일 수 있습니다.
FilterName	데이터를 로그 그룹에서 대상으로 전달하는 구독 필터의 이름입니다. 구독 필터 이름은 CloudWatch에서 ASCII로 자동 변환되며, 지원되지 않는 문자는 모두 물음표(?)로 바뀝니다.

구독 필터 지표 차원

다음 표에는 계정 수준 구독 필터와 관련된 지표의 차원이 나와 있습니다.

차원	설명
PolicyLevel	정책이 적용되는 수준입니다. 현재 이 차원에 대해 유효한 값은 AccountPolicy 뿐입니다.

차원	설명
DestinationType	CloudWatch Logs 데이터에 대한 구독 대상으로, AWS Lambda, Amazon Kinesis Data Streams 또는 Amazon Data Firehose일 수 있습니다.
FilterName	데이터를 로그 그룹에서 대상으로 전달하는 구독 필터의 이름입니다. 구독 필터 이름은 CloudWatch에서 ASCII로 자동 변환되며, 지원되지 않는 문자는 모두 물음표(?)로 바뀝니다.

로그 트랜스포머 지표 및 차원

CloudWatch Logs는 다음과 같은 로그 트랜스포머 지표를 AWS/Logs 네임스페이스의 CloudWatch에 게시합니다.

지표	설명
TransformationErrors	지정된 트랜스포머로 로그 이벤트를 변환하는 동안 발생한 오류 수입니다. 단위: 없음 유효한 통계: Sum
TransformedBytes	압축되지 않은 바이트 단위의 변환된 로그 이벤트 출력 볼륨입니다. 단위: 바이트 유효한 통계: Sum
TransformedLogEvents	변환된 로그 이벤트 수입니다. 단위: 없음 유효한 통계: Sum

다음 차원은 트랜스포머 지표에서 사용됩니다.

차원	설명
LogGroupname	이 차원은 로그 그룹 수준 트랜스포머에만 사용됩니다.
PolicyLevel	이 차원은 계정 수준 트랜스포머에만 사용됩니다. 현재 이 차원에 대해 유효한 값은 AccountPolicy 뿐입니다.

중앙 집중화 지표 및 차원

여러 계정 및 리전에서 중앙 집중식 모니터링을 위해 CloudWatch Logs Centralization을 사용하여 중앙 위치에 로그 데이터와 지표를 통합할 수 있습니다. 자세한 내용은 교차 계정 교차 리전 로그 중앙 집중화 단원을 참조하십시오.

CloudWatch Logs는 다음과 같은 중앙 집중화 지표를 AWS/Logs 네임스페이스의 CloudWatch에 게시합니다. 이러한 지표는 CloudWatch Logs 중앙 집중화 규칙을 사용할 때 소스 계정에서 대상 계정으로 로그 데이터의 복제를 모니터링하는 데 도움이 됩니다.

> **Note**
>
> CloudWatch는 중앙 집중화 규칙을 생성한 직후 중앙 집중화 지표 보고를 시작합니다. 지표는 최선을 다해 게시되며 중앙 집중화 규칙이 생성된 후 도착하는 새 로그 이벤트만 추적합니다.

지표	설명	게시 위치
IncomingCopiedBytes	대상 계정에 복제된 압축되지 않은 바이트 단위의 로그 데이터 볼륨입니다. 이 지표는 중앙 집중화 규칙이 생성된 후 도착하는 새 로그 이벤트에만 적용됩니다. 유효한 차원: SourceRegion, SourceAccount 유효한 통계: Sum 단위: 바이트	대상 계정

지표	설명	게시 위치
IncomingCopiedLogEvents	대상 계정으로 복제된 로그 이벤트 수입니다. 이 지표는 중앙 집중화 규칙이 생성된 후 도착하는 새 로그 이벤트에만 적용됩니다. 유효한 차원: SourceRegion, SourceAccount 유효한 통계: Sum 단위: 없음	대상 계정
OutgoingCopiedBytes	소스 계정에서 대상 계정으로 전송된 압축되지 않은 바이트 단위의 로그 데이터 볼륨입니다. 이 지표는 중앙 집중화 규칙이 생성된 후 대상으로 복사되는 새 로그 이벤트에만 적용됩니다. 유효한 차원: DestinationRegion 유효한 통계: Sum 단위: 바이트	소스 계정
OutgoingCopiedLogEvents	소스 계정에서 대상 계정으로 전송된 로그 이벤트 수입니다. 이 지표는 중앙 집중화 규칙이 생성된 후 대상으로 복사되는 새 로그 이벤트에만 적용됩니다. 유효한 차원: DestinationRegion 유효한 통계: Sum 단위: 없음	소스 계정

지표	설명	게시 위치
CentralizationError	로그 데이터를 복제하는 동안 발생한 오류 수입니다. KMS 키 권한 문제, 로그 그룹 할당량 제한 또는 로그 계층 불일치와 같은 다양한 이유로 오류가 발생할 수 있습니다. 복제에 실패한 특정 로그 그룹 또는 로그 스트림과 그 실패 이유를 식별하려면 ErrorType 차원을 모니터링합니다. 유효한 차원: ErrorType 유효한 통계: Sum 단위: 없음	대상 계정
CentralizationThrottled	중앙 집중화 처리가 조절된 횟수입니다. 제한을 적용해도 중앙 집중화 처리 속도가 느려지지만 로그 데이터가 복제되는 것을 막지는 않습니다. 유효한 차원: DestinationRegion 유효한 통계: Sum 단위: 없음	소스 계정

다음 차원은 중앙 집중화 지표에 사용됩니다.

차원	설명
SourceRegion	소스 로그 데이터가 시작된 AWS 리전입니다.
SourceAccount	소스 로그 데이터가 시작된 AWS 계정 ID입니다.
DestinationRegion	로그 데이터가 복제되는 AWS 리전입니다.
ErrorType	중앙 집중화 중에 발생하는 오류 유형입니다. 가능한 값은 다음과 같습니다.

차원	설명
	• LogGroupQuotaExceeded : 대상 계정이 로그 그룹 할당량에 도달했습니다. • InvalidKMS : KMS 키가 누락되었거나, 삭제되었거나, 활성화되지 않았거나, 대칭 대신 비대칭입니다. • AccessDenied : 로그 데이터를 복제하려고 할 때 액세스가 거부되었습니다. 이는 잘못된 KMS 키 권한, 누락된 IAM 권한 또는 제한적인 리소스 정책과 같은 권한 부족으로 인해 발생할 수 있습니다. • LogTierMismatch : 소스 및 대상 로그 그룹의 로그 계층이 일치하지 않습니다. • InvalidLogStream : 로그 스트림 이름이 최대 길이를 초과하는 경우와 같이 로그 스트림을 생성하거나 로그 스트림에 쓸 때 잘못된 파라미터가 발생했습니다. • InvalidLogGroup : 대상 계정에서 로그 그룹을 생성하거나 구성할 때 잘못된 파라미터가 발생했습니다. • DestinationEncryptionMismatch : 대상 로그 그룹이 중앙 집중화 규칙이 지정하는 것과 다른 KMS 암호화 구성으로 이미 존재합니다.

CloudWatch Logs 서비스 사용량 지표

CloudWatch Logs는 CloudWatch에 CloudWatch Logs API 작업의 사용량을 추적하는 지표를 전송합니다. 이러한 지표는 AWS 서비스 할당량에 해당합니다. 이러한 지표를 추적하면 할당량을 사전 예방적으로 관리하는 데 도움이 될 수 있습니다. 자세한 내용은 Service Quotas 통합 및 사용량 지표를 참조하세요.

예를 들어, ThrottleCount 지표를 추적하거나 해당 지표에 대해 경보를 설정할 수 있습니다. 이 지표의 값이 증가하면 스로틀링되는 API 작업에 대한 할당량 증가를 요청하는 것이 좋습니다. CloudWatch Logs 서비스 할당량에 대한 자세한 내용은 CloudWatch Logs 할당량 섹션을 참조하세요.

CloudWatch Logs는 AWS/Usage 및 AWS/Logs 네임스페이스 둘 다에서 서비스 할당량 사용 지표를 1분마다 게시합니다.

다음 표에는 CloudWatch Logs가 게시한 서비스 사용량 지표가 나와 있습니다. 이러한 지표에는 지정된 단위가 없습니다. 지표에 가장 유용한 통계는 SUM이며 1분 동안의 총 작업 수를 나타냅니다.

이러한 각 지표는 Service, Class, Type 및 Resource 차원 모두에 대해 값과 함께 게시됩니다. 또한 해당 지표는 Account Metrics라는 단일 차원으로 게시됩니다. Account Metrics 차원을 사용하여 계정의 모든 API 작업에 대한 지표 합계를 확인할 수 있습니다. 다른 차원을 사용하고 특정 API에 대한 지표를 찾을 수 있도록 Resource 차원에 대한 API 작업의 이름을 지정합니다.

Metrics

지표	설명
CallCount	계정에서 수행된 지정된 작업 수입니다. CallCount 는 AWS/Usage 및 AWS/Logs 네임스페이스 둘 다에서 게시됩니다.
ErrorCount	계정에서 수행되어 오류가 발생한 API 작업 수 ErrorCount 는 AWS/Logs에서만 게시됩니다.
ThrottleCount	사용 할당량으로 인해 스로틀링된 계정에서 수행된 API 작업 수입니다. ThrottleCount 는 AWS/Logs에서만 게시됩니다.

측정기준

차원	설명
Account metrics	모든 CloudWatch Logs API의 지표 합계를 가져오려면 이 차원을 사용하세요. 특정 API의 지표를 보려면 이 표에 나열된 다른 차원을 사용하고 API 이름을 Resource의 값으로 지정하세요.
Service	리소스가 포함된 AWS 서비스의 이름입니다. CloudWatch Logs 사용량 지표의 경우 이 차원 값은 Logs입니다.

차원	설명
Class	추적 중인 리소스의 클래스입니다. CloudWatch Logs API 사용량 지표는 이 차원을 None의 값으로 사용합니다.
Type	추적 중인 리소스의 유형입니다. 현재 Service 차원이 Logs인 경우 Type에 대해 유효한 값은 API뿐입니다.
Resource	API 작업의 이름입니다. 유효한 값에는 작업에 나열된 모든 API 작업 이름이 포함됩니다. 예: PutLogEvents

CloudWatch Logs 할당량

이 섹션의 표를 사용하여 Amazon CloudWatch Logs의 AWS 계정에 대한 제한이라고도 하는 기본 서비스 할당량을 검토할 수 있습니다. 전부는 아니지만 서비스 할당량 대부분은 Service Quotas 콘솔의 Amazon CloudWatch Logs 네임스페이스 아래에 나열됩니다.

> **Note**
>
> 이러한 할당량에 대한 할당량 증가를 요청하려면 이 섹션에 있는 절차를 참조하세요.

이름	기본값	조정 가능	설명
활성 내보내기 작업	지원되는 각 리전: 1	아니요	계정당 활성(실행 중 또는 보류 중) 내보내기 작업 수
초당 트랜잭션의 AssociateKmsKey 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 associate-kms-key 직접 호출 수
초당 트랜잭션의 AssociateSourceToS3TableIntegration 스로틀 제한	지원되는 각 리전: 5	아니요	계정당/리전당 초당 최대 AssociateSourceToS3TableIntegration 호출 수
버스트에서 초당 트랜잭션의 AssociateSourceToS3TableIntegration 스로틀 제한	지원되는 각 리전: 10	아니요	버스트에서 계정당/리전당 초당 최대 AssociateSourceToS3TableIntegration 호출 수
배치 크기	지원되는 각 리전: 1메가바이트	아니요	put-log-events 요청의 최대 배치 크기(MB)

이름	기본값	조정 가능	설명
초당 트랜잭션의 CancelExportTask 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 cancel-export-task 직접 호출 수
초당 트랜잭션의 CancelImportTask 스로틀 제한	지원되는 각 리전: 1	아니요	계정별/리전별 초당 최대 CancelImportTask 호출 수
버스트에서 초당 트랜잭션의 CancelImportTask 스로틀 제한	지원되는 각 리전: 1	아니요	버스트의 계정/리전당 초당 최대 CancelImportTask 호출 수
초당 트랜잭션의 CreateExportTask 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 create-export-task 직접 호출 수
초당 트랜잭션의 CreateImportTask 스로틀 제한	지원되는 각 리전: 1	아니요	계정당/리전당 초당 최대 CreateImportTask 호출 수
버스트에서 초당 트랜잭션의 CreateImportTask 스로틀 제한	지원되는 각 리전: 1	아니요	버스트의 계정/리전당 초당 최대 CreateImportTask 호출 수
초당 트랜잭션의 CreateLogGroup 스로틀 제한	지원되는 각 리전: 초당 10	예	계정별/리전별 초당 최대 create-log-group 직접 호출 수
초당 트랜잭션의 CreateLogStream 스로틀 제한	지원되는 각 리전: 초당 50개	예	계정별/리전별 초당 최대 create-log-stream 직접 호출 수

이름	기본값	조정 가능	설명
데이터 보관	지원되는 각 리전: 5GB	아니요	무료 데이터 보관의 최대 크기(GB)
초당 트랜잭션의 DeleteDataProtectionPolicy 스로틀 제한	지원되는 각 리전: 5	아니요	계정당/리전당 초당 최대 delete-data-protection-policy 호출 수
초당 트랜잭션의 DeleteDestination 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 delete-destination 직접 호출 수
초당 트랜잭션의 DeleteLogGroup 스로틀 제한	지원되는 각 리전: 초당 10	예	계정별/리전별 초당 최대 delete-log-group 직접 호출 수
초당 트랜잭션의 DeleteLogStream 스로틀 제한	지원되는 각 리전: 초당 15	예	계정별/리전별 초당 최대 delete-log-stream 직접 호출 수
초당 트랜잭션의 DeleteMetricFilter 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 delete-metric-filter 직접 호출 수
초당 트랜잭션의 DeleteRetentionPolicy 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 delete-retention-policy 직접 호출 수
초당 트랜잭션의 DeleteSubscriptionFilter 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 delete-subscription-filter 직접 호출 수

이름	기본값	조정 가능	설명
초당 트랜잭션의 DeleteTransformer 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 delete-transformer 직접 호출 수
초당 트랜잭션의 DescribeDestinations 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 describe-destinations 직접 호출 수
초당 트랜잭션의 DescribeExportTasks 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 describe-export-tasks 직접 호출 수
초당 트랜잭션의 DescribeImportTaskBatches 스로틀 제한	지원되는 각 리전: 10	아니요	계정당/리전당 초당 최대 DescribeImportTaskBatches 호출 수
버스트에서 초당 트랜잭션의 DescribeImportTaskBatches 스로틀 제한	지원되는 각 리전: 10	아니요	버스트에서 계정당/리전당 초당 최대 DescribeImportTaskBatches 호출 수
초당 트랜잭션의 DescribeImportTasks 스로틀 제한	지원되는 각 리전: 10	아니요	계정당/리전당 초당 최대 DescribeImportTasks 호출 수
버스트에서 초당 트랜잭션의 DescribeImportTasks 스로틀 제한	지원되는 각 리전: 10	아니요	버스트의 계정/리전당 초당 최대 DescribeImportTasks 호출 수
초당 트랜잭션의 DescribeLogGroups 스로틀 제한	지원되는 각 리전: 초당 10	예	계정별/리전별 초당 최대 describe-log-groups 직접 호출 수

이름	기본값	조정 가능	설명
초당 트랜잭션의 DescribeLogStreams 스로틀 제한	지원되는 각 리전: 초당 25	예	계정별/리전별 초당 최대 describe-log-streams 직접 호출 수
초당 트랜잭션의 DescribeMetricFilters 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 describe-metric-filters 직접 호출 수
초당 트랜잭션의 DescribeSubscriptionFilters 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 describe-subscription-filters 직접 호출 수
초당 트랜잭션의 DisassociateSourceFromS3TableIntegration 스로틀 제한	지원되는 각 리전: 5	아니요	계정별/리전별 초당 최대 DisassociateSourceFromS3TableIntegration 호출 수
버스트에서 초당 트랜잭션의 DisassociateSourceFromS3TableIntegration 스로틀 제한	지원되는 각 리전: 10	아니요	버스트에서 계정/리전당 초당 최대 DisassociateSourceFromS3TableIntegration 호출 수
이벤트 크기	지원되는 각 리전: 1,024KB	아니요	최대 로그 이벤트 크기(KB)

이름	기본값	조정 가능	설명
초당 트랜잭션의 FilterLogEvents 스로틀 제한	us-east-1: 초당 25개 ap-northeast-3: 초당 5개 ap-southeast-3: 초당 5개 ca-west-1: 초당 5개 eu-central-1: 초당 5개 il-central-1: 초당 5개 각각의 지원되는 다른 리전: 초당 10개	아니요	계정별/리전별 초당 최대 filter-log-events 직접 호출 수
초당 트랜잭션의 GetDataProtectionPolicy 스로틀 제한	지원되는 각 리전: 5	아니요	계정별/리전별 초당 최대 get-data-protection-policy 호출 수

이름	기본값	조정 가능	설명
초당 트랜잭션의 GetLogEvents 스로틀 제한	us-west-2: 초당 10개 ap-northeast-3: 초당 10개 ap-southeast-3: 초당 10개 ca-west-1: 초당 10개 eu-central-1: 초당 10개 eu-west-1: 초당 10개 eu-west-3: 초당 30개 il-central-1: 초당 10개 각각의 지원되는 다른 리전: 초당 25개	아니요	계정별/리전별 초당 최대 get-log-events 직접 호출 수
초당 트랜잭션의 GetQueryResults 스로틀 제한	지원되는 각 리전: 10	아니요	계정별/리전별 초당 최대 get-query-results 직접 호출 수

이름	기본값	조정 가능	설명
초당 트랜잭션의 GetTransformer 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 get-transformer 직접 호출 수
초당 트랜잭션의 ListSourcesForS3TableIntegration 스로틀 제한	지원되는 각 리전: 5	아니요	계정당/리전당 초당 최대 ListSourcesForS3TableIntegration 호출 수
버스트에서 초당 트랜잭션의 ListSourcesForS3TableIntegration 스로틀 제한	지원되는 각 리전: 10	아니요	버스트에서 계정당/리전당 초당 최대 ListSourcesForS3TableIntegration 호출 수
초당 트랜잭션의 ListTagsForResource 스로틀 제한	지원되는 각 리전: 초당 30개	아니요	계정별/리전별 초당 최대 list-tags-for-resource 직접 호출 수
초당 트랜잭션의 ListTagsLogGroup 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 list-tags-log-group 직접 호출 수
Live Tail 동시 세션 제한	지원되는 각 리전: 15	예	계정당 최대 동시 활성 Live Tail 세션 수
로그 그룹	지원되는 각 리전: 1,000,000	예	한 계정이 보유할 수 있는 최대 로그 그룹의 수
Live Tail 세션당 스캔된 로그 그룹	지원되는 각 리전: 10	아니요	Live Tail 세션당 스캔할 수 있는 최대 로그 그룹 수
로그 그룹당 지표 필터	지원되는 각 리전: 100	아니요	로그 그룹당 지표 필터 수

이름	기본값	조정 가능	설명
초당 트랜잭션의 PutBearerTokenAuthentication 스로틀 제한	지원되는 각 리전: 5	아니요	계정별/리전별 초당 최대 put-bearer-token-authentication 호출 수
버스트에서 초당 트랜잭션의 PutBearerTokenAuthentication 스로틀 제한	지원되는 각 리전: 10	아니요	버스트의 계정/리전당 초당 최대 put-bearer-token-authentication 호출 수
초당 트랜잭션의 PutDataProtectionPolicy 스로틀 제한	지원되는 각 리전: 5	아니요	계정별/리전별 초당 최대 put-data-protection-policy 호출 수
초당 트랜잭션의 PutDestination 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 put-destination 직접 호출 수
초당 트랜잭션의 PutDestinationPolicy 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 put-destination-policy 직접 호출 수
초당 트랜잭션의 PutLogEvents 스로틀 제한	지원되는 각 리전: 초당 5,000	예	계정별/리전별 초당 최대 put-log-events 직접 호출 수
초당 트랜잭션의 PutMetricFilter 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 put-metric-filter 직접 호출 수
초당 트랜잭션의 PutRetentionPolicy 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 put-retention-policy 직접 호출 수

이름	기본값	조정 가능	설명
초당 트랜잭션의 PutSubscriptionFilter 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 put-subscription-filter 직접 호출 수
초당 트랜잭션의 PutTransformer 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 put-transformer 직접 호출 수
리소스 정책	지원되는 각 리전: 10	아니요	계정별/리전별 초당 최대 리소스 정책 수
초당 트랜잭션의 StartLiveTail 스로틀 제한	지원되는 각 리전: 5	아니요	계정별/리전별 초당 최대 start-live-tail 직접 호출 수. 이 제한은 콘솔과 API 모두에 독립적으로 적용됩니다.
초당 트랜잭션의 StartQuery 스로틀 제한	지원되는 각 리전: 10	아니요	계정별/리전별 초당 최대 start-query 직접 호출 수
로그 그룹당 구독 필터	지원되는 각 리전: 5	아니요	로그 그룹당 구독 필터의 수
초당 트랜잭션의 TagLogGroup 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 tag-log-group 직접 호출 수
초당 트랜잭션의 TagResource 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 tag-resource 직접 호출 수

이름	기본값	조정 가능	설명
초당 트랜잭션의 TestMetricFilter 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 test-metric-filter 직접 호출 수
초당 트랜잭션의 TestTransformer 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 test-transformer 직접 호출 수
초당 트랜잭션의 UntagLogGroup 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 untag-log-group 직접 호출 수
초당 트랜잭션의 UntagResource 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	계정별/리전별 초당 최대 untag-resource 직접 호출 수
로그 이상 탐지기	지원되는 각 리전: 500	예	활성 로그 이상 탐지기의 최대 수

CloudWatch Logs 서비스 할당량 관리

CloudWatch Logs는 중앙 위치에서 할당량을 보고 관리할 수 있는 AWS 서비스인 Service Quotas와 통합되었습니다. 자세한 내용은 Service Quotas 사용 설명서의 Service Quotas는 무엇입니까?를 참조하세요.

Service Quotas를 사용하면 CloudWatch Logs 서비스 할당량의 값을 쉽게 찾을 수 있습니다.

AWS Management Console

콘솔을 사용하여 CloudWatch Logs 서비스 할당량을 보려면

1. https://console.aws.amazon.com/servicequotas/에서 Service Quotas 콘솔을 엽니다.
2. 탐색 창에서 AWS 서비스를 선택합니다.

3. AWS 서비스 목록에서 Amazon CloudWatch Logs를 검색하여 선택합니다.

   서비스 할당량 목록에서 서비스 할당량 이름, 적용된 값(제공된 경우), AWS 기본 할당량 및 할당량 값 조정 가능 여부를 확인할 수 있습니다.
4. 설명 등 서비스 할당량에 대한 추가 정보를 보려면 할당량 이름을 선택합니다.
5. (선택 사항) 할당량 증가를 요청하려면 증가시킬 할당량을 선택하고 할당량 증가 요청을 선택한 다음, 필요한 정보를 입력하거나 선택한 다음 요청을 선택합니다.

콘솔을 사용하여 서비스 할당량에 대한 추가 작업을 수행하려면 Service Quotas 사용 설명서를 참조하세요. 할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

AWS CLI

를 사용하여 CloudWatch Logs 서비스 할당량을 보려면 AWS CLI

다음 명령을 실행하여 기본 CloudWatch Logs 할당량을 확인합니다.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code logs \
    --output table
```

를 사용하여 서비스 할당량에 대해 자세히 알아보려면 Service Quotas AWS CLI 명령 참조를 AWS CLI참조하세요. 할당량 증가를 요청하려면 AWS CLI 명령 참조에서 `request-service-quota-increase` 명령을 참조하세요.

문서 기록

다음 표에는 2018년 6월부터 적용되는 CloudWatch Logs 사용 설명서의 각 릴리스에서 변경된 중요 사항이 나와 있습니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
CloudWatch Logs Insights에 대한 새로운 견적 쿼리 명령	이제 CloudWatch Logs Insights의 `estimate` 명령을 사용하여 쿼리를 실행하지 않고 선택한 로그 그룹 및 시간 범위에서 쿼리가 스캔할 예상 바이트를 반환할 수 있습니다. `estimate` 명령에는 CloudWatch Logs Insights 쿼리 요금이 발생하지 않습니다. 자세한 내용은 CloudWatch Logs Insights 언어 쿼리 구문을 참조하세요.	2026년 7월 22일
AWSServiceRoleForLogDelivery 서비스 연결 역할 정책 업데이트	CloudWatch Logs가 AWSServiceRoleForLogDelivery 서비스 연결 역할 정책에 `kms:GenerateDataKey` 및 `kms:Decrypt` 권한을 추가했습니다. 이러한 권한을 통해 고객 관리형 키(SSE-CMK)를 사용한 서버 측 암호화를 사용하는 Firehose 전송 스트림으로 로그를 전송할 수 있습니다. 자세한 내용은 AWS 서비스 연결 역할에 대한 CloudWatch Logs 업데이트를 참조하세요.	2026년 5월 15일

CloudWatch Logs Insights에 대한 파라미터를 사용하여 저장된 쿼리	이제 CloudWatch Logs Insights에서 명명된 파라미터를 사용하여 재사용 가능한 쿼리 템플릿을 생성할 수 있습니다. 저장된 쿼리를 거의 동일하게 여러 개 유지 관리하는 대신 단일 템플릿을 정의하고 런타임 시 다른 값을 전달합니다. 자세한 내용은 파라미터와 함께 저장된 쿼리 사용을 참조하세요.	2026년 3월 26일
새 CloudWatch Logs 관리형 정책 CloudWatchLogsAPIKeyAccess	CloudWatch Logs는 CloudWatch Logs API 키 인증 및 암호화된 로그 수집을 활성화`CloudWatchLogsAPIKeyAccess` 하는 새로운 관리형 정책을 릴리스했습니다. 자세한 내용은 AWS 관리형 정책에 대한 CloudWatch Logs 업데이트를 참조하세요.	2026년 2월 17일
CloudWatchLogsReadOnlyAccess 정책 업데이트	원격 측정 파이프라인 및 S3 테이블 통합에 대한 읽기 전용 액세스를 허용하기 위해 관찰성 관리 작업에 대한 권한이 CloudWatchLogsReadOnlyAccess에 추가되었습니다.	2025년 12월 2일
업데이트된 CloudWatchLogsFullAccess 정책	원격 측정 파이프라인 및 S3 테이블 통합에 대한 읽기 전용 액세스를 허용하기 위해 관찰성 관리 작업에 대한 권한이 CloudWatchLogsFullAccess에 추가되었습니다.	2025년 12월 2일

CloudWatch Logs의 새로운 로그 관리 기능	CloudWatch는 서로 다른 도구 및 데이터 스토어에서 동일한 데이터의 여러 복사본을 저장하고 유지 관리하지 않고도 기본 제공 거버넌스 기능을 사용하여 로그 관리를 단일 서비스로 통합합니다. 자세한 내용은 로그 관리를 참조하세요.	2025년 12월 2일
CloudWatch Logs의 새로운 데이터 수집 및 정규화 기능	CloudWatch는 AWS Organizations 및 타사 소스용 사전 구축된 커넥터와의 통합을 통해 계정 및 AWS 리전 간에 AWS 판매된 로그를 자동으로 수집합니다. 자세한 내용은 Transform logs during ingestion을 참조하세요.	2025년 12월 2일
CloudWatch Logs Insights의 데이터 소스 기능을 기반으로 하는 새로운 분석 및 쿼리	CloudWatch Insights는 쿼리를 위한 패싯을 도입합니다. 패싯은 쿼리를 실행하지 않고 데이터를 대화식으로 필터링하고 그룹화할 수 있으므로 로그를 분석하는 데 유용합니다. 패싯은 로그 그룹의 필터링, 집계 및 분석을 지원하는 로그의 필드(예: ServiceName 또는 StatusCode)입니다. 자세한 내용은 패싯을 사용하여 로그 그룹화 및 탐색을 참조하세요.	2025년 12월 2일

새로운 S3 테이블 통합	CloudWatch와 S3 Tables 통합을 사용하면 Amazon Athena, Amazon Redshift 및 Apache Iceberg 호환 스토어에 대한 연결을 지원하는 타사 도구와 같은 분석 엔진을 사용하여 CloudWatch에 수집된 로그 데이터에 액세스할 수 있습니다. 자세한 내용은 S3 Tables 통합을 통한 액세스 로그를 참조하세요.	2025년 12월 2일
CloudWatch Logs에서 교차 계정 교차 리전 로그 중앙 집중화 지원 추가	CloudWatch Logs는 CloudWatch Logs가 분석을 위해 AWS Organizations 멤버 계정에서 중앙 계정으로 로그 데이터를 수집할 수 있도록 교차 계정 교차 리전 로그 중앙 집중화에 대한 지원을 추가했습니다. 자세한 내용은 Cross-account cross-Region log centralization를 참조하세요.	2025년 9월 17일
CloudWatch Logs 이상 탐지 업데이트된 예제 정책	CloudWatch Logs는 `aws:SourceAccount` 및 `aws:SourceArn` 에 대한 조건을 추가하여 범위를 줄이고 보안을 개선하도록 KMS 예제 정책을 업데이트했습니다. 자세한 내용은 AWS KMS를 사용하여 이상 탐지기 및 그 결과 암호화를 참조하세요.	2025년 7월 14일

CloudWatch Logs Insights에서 Amazon VPC Route Server 로그에 대한 지원 추가	CloudWatch Logs Insights에서 Amazon VPC Route Server 로그에 대한 지원을 추가합니다. 자세한 내용은 Enable logging from AWS services을 참조하세요. Amazon VPC Route Server의 새 로그 소스`EVENT_LOGS` 은 PutDeliverySource에 문서화되어 있습니다.	2025년 6월 12일
CloudWatch Logs Insights에서 AWS PCS 로그에 대한 지원 추가	CloudWatch Logs Insights는 AWS PCS 로그에 대한 지원을 추가합니다. 자세한 내용은 Enable logging from AWS services을 참조하세요. AWS PCS`PCS_SCHEDULER_LOGS` 및에 대한 새 로그 소스`PCS_JOBCOMP_LOGS` 는 PutDeliverySource에 문서화되어 있습니다.	2025년 6월 12일
CloudWatch Logs Insights에서 AWS Entity Resolution 로그에 대한 지원 추가	CloudWatch Logs Insights는 AWS Entity Resolution 로그에 대한 지원을 추가합니다. 자세한 내용은 Enable logging from AWS services을 참조하세요. 에 대한 새 로그 소스 AWS Entity Resolution`WORKFLOW_LOGS` 는 PutDeliverySource에 문서화되어 있습니다.	2025년 5월 22일

자연어 요약을 지원하는 CloudWatch Logs 관리형 정책 업데이트	CloudWatchLogsFullAccess 및 CloudWatchLogsReadOnlyAccess에 `cloudwatch:GenerateQueryResultsSummary` 에 대한 권한이 추가되어 쿼리 결과에 대한 자연어 요약을 생성할 수 있게 되었습니다. 정책의 내용을 보려면 AWS 관리형 정책 참조 가이드의 CloudWatchLogsFullAccess 및 CloudWatchLogsReadOnlyAccess를 참조하세요.	2025년 5월 20일
CloudWatch Logs Insights에서 CloudWatch Logs Insights 쿼리 결과에서 자연어 요약 생성에 대한 지원 추가	CloudWatch Logs Insights에 자연어 요약에 대한 지원이 추가됨 이 기능은 현재 미국 동부(버지니아 북부)에서 이용할 수 있으며, 사람이 읽을 수 있는 쿼리 결과 요약을 생성합니다. 자세한 내용은 Generate natural language summaries from CloudWatch Logs Insights query results을 참조하세요.	2025년 5월 20일

CloudWatchOpenSearchDashboardAccess 및 CloudWatchOpenSearchDashboardsFullAccess는 새 IAM 정책입니다.	CloudWatch Logs에는 두 가지 새 IAM 정책인 CloudWatchOpenSearchDashboardsFullAccess 및 CloudWatchOpenSearchDashboardAccess가 추가되었습니다. CloudWatchOpenSearchDashboardsFullAccess는 OpenSearch Service와의 통합을 생성 및 관리할 수 있는 권한을 부여합니다. CloudWatchOpenSearchDashboardAccess는 이러한 통합에서 생성된 판매 로그 대시보드를 확인할 수 있는 액세스 권한을 부여합니다.	2024년 12월 1일
CloudWatchLogsFullAccess 정책 업데이트됨	CloudWatch Logs는 CloudWatchLogsFullAccess 정책에 Amazon OpenSearch Service 및 IAM에 대한 권한을 추가하여 일부 기능에 대해 CloudWatch Logs를 OpenSearch Service와 통합할 수 있도록 했습니다.	2024년 12월 1일
CloudWatch Logs Insights에서 쿼리 구문에 새 구조 유형 추가	CloudWatch Logs Insights에서 `unnest` 명령과 두 개의 JSON 함수를 추가하여 JSON 문자열을 맵 및 목록으로 연산할 수 있습니다. 자세한 내용은 Structure types을 참조하세요.	2024년 11월 21일

CloudWatch Logs는 로그 수집 중 로그 변환 지원	수집 시 로그 이벤트를 수정할 수 있는 로그 트랜스포머를 생성하여 다양한 형식과 다양한 소스의 로그를 일관되고 컨텍스트가 풍부한 형식으로 정규화할 수 있습니다. 자세한 내용은 Transform logs during ingestion을 참조하세요.	2024년 11월 20일
CloudWatch Logs Insights에서 필드 인덱싱 추가	CloudWatch Logs Insights에서 로그의 필드 인덱싱에 대한 지원을 추가했습니다. 그런 다음 CloudWatch Log Insights 쿼리에서 필드 인덱스를 사용하면 쿼리는 인덱싱된 필드를 포함하지 않는 것으로 알려진 로그 이벤트 처리를 건너뛰려고 시도합니다. 자세한 내용은 쿼리 성능을 향상시키고 스캔 볼륨을 줄이기 위해 필드 인덱스 생성을 참조하세요.	2024년 11월 20일
CloudWatch Logs Insights 자연어 쿼리 생성 지원 정식 출시	CloudWatch Logs Insights는 자연어를 사용한 쿼리 생성과 업데이트를 지원합니다. 자세한 내용은 Use natural language to generate and update CloudWatch Logs Insights queries를 참조하세요.	2024년 6월 20일

CloudWatchLogsReadOnlyAccess 정책 업데이트됨	CloudWatch Logs에서 CloudWatchLogsReadOnlyAccess에 `cloudwatch:GenerateQuery` 권한을 추가했습니다. 이 정책이 적용되는 사용자는 자연어 프롬프트에서 CloudWatch Logs Insights 쿼리 문자열을 생성할 수 있습니다.	2023년 11월 26일
CloudWatchLogsFullAccess 정책 업데이트됨	CloudWatch Logs에서 CloudWatchLogsFullAccess에 `cloudwatch:GenerateQuery` 권한을 추가했습니다. 이 정책이 적용되는 사용자는 자연어 프롬프트에서 CloudWatch Logs Insights 쿼리 문자열을 생성할 수 있습니다.	2023년 11월 26일
CloudWatch Logs에서 로그 패턴 분석 추가	CloudWatch Logs는 이제 사용자가 CloudWatch Logs Insights 쿼리를 수행할 때마다 로그 이벤트에서 패턴을 스캔합니다. 자세한 내용은 패턴 분석을 참조하세요.	2023년 11월 26일
CloudWatch Logs에서 로그 이상 탐지 추가	로그 그룹의 로그 이상 탐지기를 생성할 수 있습니다. 이상 탐지기는 로그 그룹에 수집된 로그 이벤트를 스캔하여 로그 데이터에서 이상을 찾습니다. 자세한 내용은 로그 이상 탐지를 참조하세요.	2023년 11월 26일

CloudWatch Logs에서 비교 기능 추가	이제 CloudWatch Logs Insights를 사용하여 시간 경과에 따른 로그 이벤트의 변경 사항을 비교할 수 있습니다. 자세한 내용은 이전 시간 범위와 (diff) 비교를 참조하세요.	2023년 11월 26일
CloudWatch Logs에서 새 로그 클래스 추가	CloudWatch Logs는 두 가지 로그 그룹 클래스를 지원하므로 자주 액세스하지 않는 로그에 비용 효율적인 옵션을 사용할 수 있으며, 실시간 모니터링이나 기타 기능이 필요한 로그에는 전체 기능을 사용할 수 있습니다. 자세한 내용은 로그 클래스를 참조하세요.	2023년 11월 26일
CloudWatch Logs Insights 자연어 쿼리 생성 지원	CloudWatch Logs Insights는 자연어를 사용한 쿼리 생성과 업데이트를 지원합니다. 자세한 내용은 Use natural language to generate and update CloudWatch Logs Insights queries를 참조하세요.	2023년 11월 26일
CloudWatch Logs에서 Live Tail에 대한 정규 표현식 필터 패턴 구문 지원 추가	이제 Live Tail 필터 패턴 내에서 유연한 정규식을 사용하여 필요에 맞게 검색 및 매칭 작업을 추가로 사용자 지정할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 필터 패턴 구문을 참조하세요.	2023년 11월 13일

CloudWatch Logs, 지표 필터, 구독 필터 및 필터 로그 이벤트에 대한 정규식 필터 패턴 구문 지원 추가	이제 필터 패턴 내에서 유연한 정규식을 사용하여 필요에 맞게 검색 및 매칭 작업을 추가로 사용자 지정할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 필터 패턴 구문을 참조하세요.	2023년 9월 5일
CloudWatch Logs Insights에서 패턴 명령 추가	이제 CloudWatch Logs Insights 쿼리에서 패턴을 사용하여 로그 데이터를 패턴으로 자동 클러스터링할 수 있습니다. 패턴은 로그 필드 간에 반복되는 공유 텍스트 구조입니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 패턴을 참조하세요.	2023년 7월 17일
CloudWatch Logs Insights에서 dedup 명령 추가	이제 CloudWatch Logs Insights 쿼리에서 dedup을 사용하여 지정한 필드의 특정 값을 기준으로 중복 결과를 제거할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 dedup을 참조하세요.	2023년 6월 20일

계정 수준 데이터 보호 정책	이제 계정 수준에서 데이터 보호 정책을 설정할 수 있습니다. 이러한 계정 수준 정책은 계정의 모든 로그 그룹에 있는 로그 이벤트의 민감한 정보를 감사 및 마스킹 처리할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 마스킹으로 민감한 로그 데이터 보호 지원을 참조하세요.	2023년 6월 8일
Live Tail 기능 추가됨	CloudWatch Logs에 Live Tail 기능이 추가되어 문제 해결에 도움이 되도록 모은 로그를 스캔할 수 있습니다. 필요에 따라 지정된 항을 기준으로 표시된 로그 이벤트 스트림을 필터링하고 지정된 항이 있는 로그 이벤트를 강조 표시할 수도 있습니다. 자세한 내용은 Live Tail을 사용하여 거의 실시간으로 로그 보기를 참조하세요.	2023년 6월 6일
CloudWatchLogsReadOnlyAccess 정책 업데이트됨	CloudWatch Logs에서 CloudWatchLogsReadOnlyAccess에 대한 권한을 추가했습니다. 이 정책이 있는 사용자가 콘솔을 사용하여 CloudWatch Logs Live Tail 세션을 시작하고 중지할 수 있도록 `logs:StartLiveTail` 및 `logs:StopLiveTail` 권한이 추가되었습니다. 자세한 내용은 Live Tail을 사용하여 거의 실시간으로 로그 보기를 참조하세요.	2023년 6월 6일

CloudWatch Logs Insights 릴리스됨	CloudWatch Logs Insights를 사용하면 로그 데이터를 대화식으로 검색해 분석할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 CloudWatch Logs Insights에서 로그 데이터 분석을 참조하세요.	2018년 11월 27일
Amazon VPC 엔드포인트에 대한 지원	이제 VPC와 CloudWatch Logs 간에 프라이빗 연결을 설정할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 인터페이스 VPC 엔드포인트에서 CloudWatch Logs 사용을 참조하세요.	2018년 6월 28일

아래 표에 Amazon CloudWatch Logs 사용자 가이드의 주요 변경 사항이 설명되어 있습니다.

변경	설명	릴리스 날짜
인터페이스 VPC 엔드포인트	일부 리전에서는 인터페이스 VPC 엔드포인트를 사용하여 Amazon 네트워크에서 Amazon VPC와 CloudWatch Logs 간의 트래픽을 유지할 수 있습니다. 자세한 내용은 인터페이스 VPC 엔드포인트에서 CloudWatch Logs 사용을 참조하세요.	2018년 3월 7일
Route 53 DNS 쿼리 로그	Route 53을 사용하여 CloudWatch Logs가 수신하는 DNS 쿼리에 대한 로그를 저장할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 Amazon CloudWatch Logs란 무엇입니까? 또는 DNS 쿼리 로깅을 참조하세요.	2017년 9월 7일
로그 그룹 태그 지정	태그를 사용하여 로그 그룹을 분류할 수 있습니다. 자세한 정보는 Amazon CloudWatch Logs의 로그 그룹에 태그 지정 섹션을 참조하세요.	2016년 12월 13일

변경	설명	릴리스 날짜
콘솔 개선 사항	지표 그래프부터 관련 로그 그룹까지 검색할 수 있습니다. 자세한 정보는 지표에서 로그로 피벗 적용 섹션을 참조하세요.	2016년 11월 7일
콘솔 활용도 개선 사항	보다 손쉬운 검색, 필터링 및 문제 해결을 위해 경험을 개선했습니다. 예를 들어, 날짜 및 시간 범위로 로드 데이터를 필터링할 수 있습니다. 자세한 정보는 CloudWatch Logs로 전송된 로그 데이터 보기 섹션을 참조하세요.	2016년 8월 29일
Amazon CloudWatch Logs 및 새 CloudWatch Logs 지표에 대한 AWS CloudTrail 지원 추가	CloudWatch Logs에 대한 AWS CloudTrail 지원이 추가되었습니다. 자세한 내용은 에서 CloudWatch Logs API 및 콘솔 작업 로깅 AWS CloudTrail 단원을 참조하십시오.	2016년 3월 10일
Amazon S3로 CloudWatch Logs 내보내기에 대한 지원이 추가됨	Amazon S3에 CloudWatch Logs 데이터 내보내기에 대한 지원이 추가됨 자세한 정보는 Amazon S3로 로그 데이터 내보내기 섹션을 참조하세요.	2015년 12월 7일
Amazon CloudWatch Logs에 AWS CloudTrail 로깅된 이벤트에 대한 지원 추가	CloudWatch에서 경보를 생성하고, CloudTrail에서 포착된 특정 API 활동에 대한 알림을 수신하며, 이러한 알림을 사용하여 문제를 해결할 수 있습니다.	2014년 11월 10일

변경	설명	릴리스 날짜
Amazon CloudWatch Logs에 대한 지원이 추가됨	Amazon CloudWatch Logs을 사용하여 Amazon Elastic Compute Cloud (Amazon EC2) 인스턴스나 다른 원본에서 시스템, 애플리케이션 및 사용자 지정 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. 그런 다음 Amazon CloudWatch 콘솔, 의 Amazon CloudWatch Logs 명령 AWS CLI또는 CloudWatch Logs SDK를 사용하여 CloudWatch Logs에서 연결된 로그 데이터를 검색할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs란 무엇입니까? 단원을 참조하십시오.	2014년 7월 10일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.