

AWS Secrets Manager와 Parameter Store를 통한 민감 데이터 관리

시스템 구성 요소

1.AWS Secrets Manager

- 민감 정보(데이터베이스 자격 증명, API 키 등)를 암호화하여 저장.
- KMS를 활용한 데이터 암호화.

2.AWS Parameter Store

- 비민감 데이터(환경 변수, 설정값) 관리.
- 계층 구조를 통해 설정 데이터를 논리적으로 분리.

3.AWS Lambda

- 서버리스 방식으로 데이터 접근 로직 구현.
- 특정 조건에서만 Secrets Manager/Parameter Store 데이터를 호출.

4.AWS KMS

- 데이터 암호화 및 암호화 키 관리.
- Secrets Manager와 통합하여 민감 정보를 안전하게 보호.

5.AWS CloudWatch Logs

- Lambda 실행 기록 및 데이터 접근 로깅.
- 보안 모니터링 및 감사 용도로 활용.

6.IAM (Identity and Access Management)

- 접근 권한 구성.
- 역할 기반 접근 제어(Role-Based Access Control, RBAC) 설정.

- **민감 데이터 관리의 핵심 요소**

- **4-1. 암호화**

- 데이터는 반드시 **암호화된 상태**로 저장되고, 전송 시에도 보호되어야 합니다.
- AWS KMS(Key Management Service)는 데이터 암호화와 키 관리를 자동화하여 데이터를 안전하게 보호.

- **4-2. 접근 제어**

- 모든 민감 데이터는 최소 권한 원칙(Principle of Least Privilege)에 따라 접근 권한이 설정되어야 합니다.
- IAM(Role-Based Access Control)을 사용하여 특정 역할만 민감 데이터에 접근하도록 제한.

- **4-3. 키 로테이션**

- 비밀번호, API 키 등 민감 데이터가 주기적으로 변경되어야 합니다.
- AWS Secrets Manager는 자동 키 로테이션 기능을 제공하여 이러한 과정을 자동화.

- **4-4. 감사와 모니터링**

- **CloudTrail**과 **CloudWatch** 같은 도구를 사용해 민감 데이터 접근 기록을 남기고 비정상적인 활동을 모니터링.
- 이를 통해 보안 사고 발생 시 원인을 빠르게 파악 가능.

1단계: 설계 및 초기 환경 설정

IAM 역할 및 정책 생성

Lambda 함수가 AWS Secrets Manager와 Systems Manager Parameter Store에 안전하게 접근할 수 있도록 **IAM 역할**과 **정책**을 설정합니다.

LambdaSecretsRole [정보](#)

Allows Lambda functions to call AWS services on your behalf.


[삭제](#)

요약

생성 날짜

December 16, 2024, 16:24 (UTC+09:00)

마지막 활동

 4시간 전

ARN

 arn:aws:iam::891376949960:role/LambdaSecretsRole

최대 세션 지속 시간

1시간

[편집](#)[권한](#)[신뢰 관계](#)[태그](#)[마지막 액세스](#)[세션 취소](#)

권한 정책 (4) [정보](#)

최대 10개의 관리형 정책을 연결할 수 있습니다.

[시뮬레이션](#)[삭제](#)[권한 추가](#)

검색

필터링 기준 유형

모든 유형

< 1 >



<input type="checkbox"/>	정책 이름	유형	연결된 엔터티
<input type="checkbox"/>	  AmazonSSMFullAccess	AWS 관리형	2
<input type="checkbox"/>	  AmazonSSMReadOnlyAccess	AWS 관리형	2
<input type="checkbox"/>	  AWSLambdaBasicExecutionRole	AWS 관리형	1
<input type="checkbox"/>	  SecretsManagerReadWrite	AWS 관리형	2

1.IAM 역할 생성

- AWS Management Console에서 IAM 서비스로 이동합니다.
- Lambda 서비스에 사용할 새 IAM 역할을 생성합니다.

2.권한 정책 추가

- AmazonSSMFullAccess**

- SSM Parameter Store와 관련된 모든 권한을 허용합니다. (쓰기 및 읽기 권한 포함)

- AWSLambdaBasicExecutionRole**

- Lambda 함수의 실행을 위한 **CloudWatch Logs** 접근 권한만 제공합니다.

- SecretsManagerReadWrite**

- AWS Secrets Manager의 **읽기/쓰기 권한**을 모두 허용합니다.특정 Secret에 대한 제한이 없습니다.

MyCustom의 권한 수정 정보

서비스, 작업, 리소스 및 조건을 선택하여 권한을 추가합니다. JSON 편집기를 사용하여 권한 설명문을 작성합니다.

정책 편집기

```
1 ▼ {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "secretsmanager:GetSecretValue"
8       ],
9       "Resource": "arn:aws:secretsmanager:us-east-1:891376949960:secret:prod/db-credentials-XgQDpc"
10    },
11    {
12      "Effect": "Allow",
13      "Action": [
14        "ssm:GetParameter"
15      ],
16      "Resource": "arn:aws:ssm:us-east-1:891376949960:parameter/prod/config/api-url"
17    },
18    {
19      "Effect": "Allow",
20      "Action": [
21        "logs:CreateLogGroup",
22        "logs:CreateLogStream",
23        "logs:PutLogEvents"
24      ],
25      "Resource": "arn:aws:logs:us-east-1:891376949960:log-group:/aws/lambda/myLamda:*"
26    }
27  ]
28 }
```

AWS Secrets Manager 설정

Secrets Manager는 민감한 데이터를 안전하게 저장하고 관리하는 서비스입니다. 데이터는 **AWS KMS(Key Management Service)**를 사용해 암호화됩니다.

☰ AWS Secrets Manager > 보안 암호

보안 암호

🔍 이름, 설명, 태그 키, 태그 값, 보유 서비스 또는 기본 리전을 기준으로 보안 암호 필터링

보안 암호 이름	설
prod/db-credentials	-

생성 시 화면

- 1단계 보안 암호 유형 선택
- 2단계 보안 암호 구성
- 3단계 - 선택 사항 교체 구성
- 4단계 검토

보안 암호 유형 선택

보안 암호 유형 정보

- ☒ Amazon RDS 데이터베이스에 대한 자격 증명
- ☐ Amazon DocumentDB 데이터베이스에 대한 자격 증명
- ☐ Amazon Redshift 데이터 웨어하우스용 보안 인증
- ☐ 기타 데이터베이스에 대한 자격 증명
- ☐ 다른 유형의 보안 암호
API 키, OAuth 토큰, 기타.

자격 증명 정보

사용자 이름

암호

☐ 암호 표시

암호화 키 정보

Secrets Manager가 생성하는 KMS 키 또는 사용자가 생성한 고객 관리형 KMS 키를 사용하여 암호화할 수 있습니다.

aws/secretsmanager

새 키 추가

데이터베이스 정보

🔍 인스턴스 검색

< 1 >

DB 인스턴스	DB 엔진	상태	생성 날짜(UTC)
database-1	mysql	available	2024년 12월 16일 7시 54분...

1.비밀(Secret) 생성

- AWS Management Console에서 Secrets Manager로 이동합니다.
- 새로운 비밀 생성**을 선택하고, 다음과 같은 민감 정보를 저장합니다:
 - 데이터베이스 자격 증명:{ "username": "admin", "password": "mypassword" }

2.암호화 설정

- 기본적으로 AWS KMS 키를 사용해 데이터가 암호화됩니다.
- 필요에 따라 **고유한 KMS 키**를 생성하거나 기존 KMS 키를 사용합니다. 본 실습에서는 기존 KMS 키를 사용했습니다.

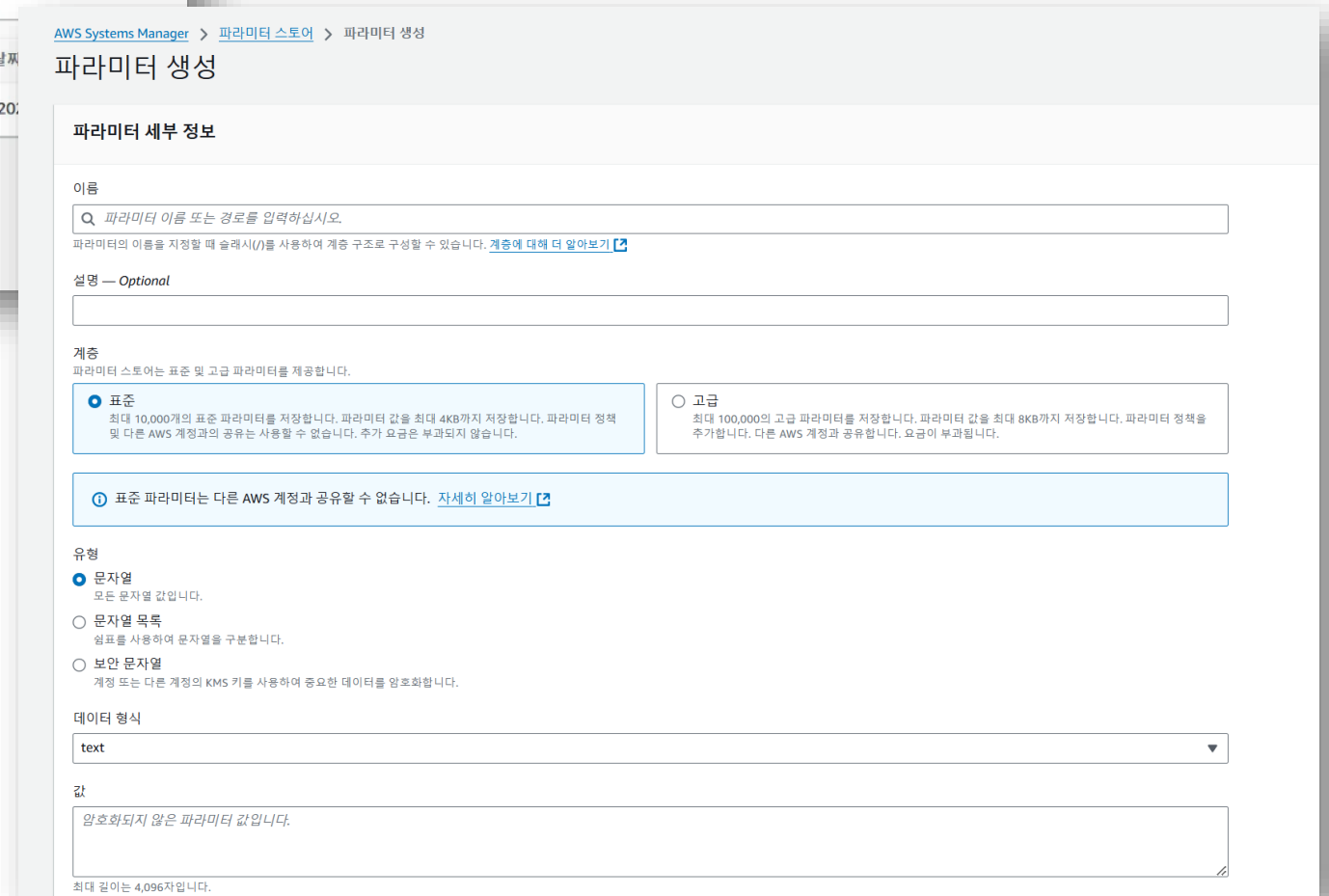
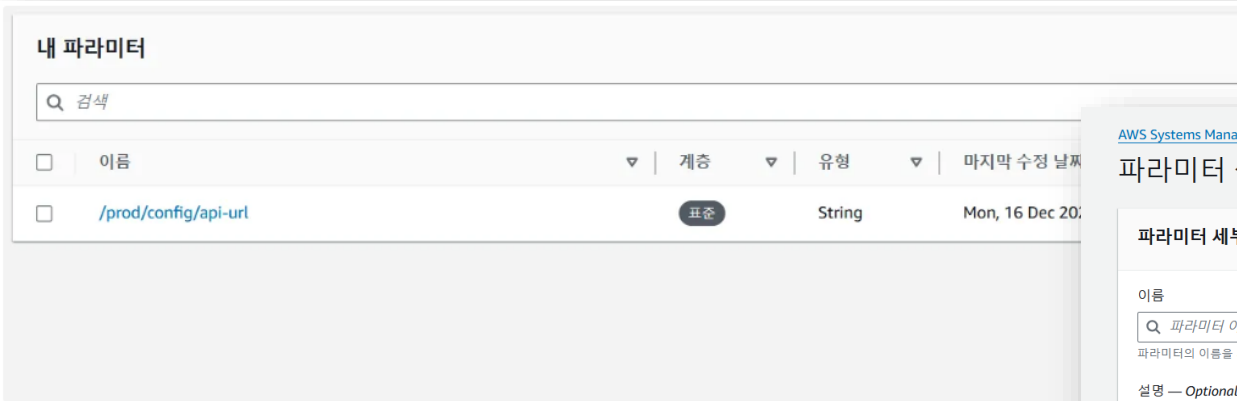
3.리소스 이름 지정 규칙

비밀의 이름은 접근 및 관리가 쉽도록 명명 규칙을 따릅니다:

- prod/db-credentials

AWS Parameter Store 설정

Parameter Store는 비민감 데이터(환경 변수 및 설정값)를 저장하는 서비스입니다. 계층적 구조를 사용해 설정값을 관리합니다.



1.파라미터 생성

AWS Management Console에서 Systems Manager → Parameter Store로 이동합니다.

- 새 파라미터 생성.
- 파라미터 유형:
 - String**: 단일 문자열 값.
 - StringList**: 쉼표로 구분된 문자열 목록.
 - SecureString**: KMS를 사용한 암호화된 문자열.

2.비민감 데이터

일반적인 설정 값들을 계층 구조로 저장합니다. 해당 프로젝트에서는 아래의 파라미터를 저장합니다.

- /prod/config/api-url: 개발 환경의 API 엔드포인트 URL.

3.KMS 암호화 (옵션)

- 민감도가 낮은 데이터라면 String을 사용해 저장하고,
- 추가 보안이 필요할 경우 **SecureString**으로 저장해 KMS 키로 암호화합니다.

2단계: 민감 데이터 저장 및 관리

Secrets Manager: Secrets Manager에 민감한 데이터를 저장하고 안전하게 관리합니다.

•데이터베이스 자격 증명

- Secret Name: prod/db-credentials

데이터 구조

```
{  
  "username": "admin",  
  "password": "mypassword",  
  "engine": "mysql"  
  "host": "database-1.c7kg62g660d6.us-east-1.rds.amazonaws.com",  
  "port": "3306",  
  "dbInstanceIdentifier": "database-1"  
}
```

보안 암호 세부 정보



작업 ▼

암호화 키

aws/secretsmanager

보안 암호 이름

prod/db-credentials

보안 암호 ARN

arn:aws:secretsmanager:us-east-1:891376949960:secret:prod/db-credentials-XgQDpc

보안 암호 설명

-

개요

교체

버전

복제

태그

보안 암호 값 [정보](#)

닫기

편집

보안 암호 값을 검색하고 확인합니다.

키/값

일반 텍스트

보안 암호 키

보안 암호 값

username

admin

password

MYPASSWORD

engine

mysql

host

database-1.c7kg62g660d6.us-east-1.rds.amazonaws.com

port

3306

dbInstanceIdentifier

database-1

Parameter Store

Parameter Store에는 환경 변수 및 일반 설정 값을 저장합니다.

•개발 환경 URL


- Parameter Name: /prod/config/api-url
- 값: https://localhost:3000

[AWS Systems Manager](#) > [파라미터 스토어](#) > [/prod/config/api-url](#) > 개요

/prod/config/api-url

[개요](#) | [기록](#) | [태그](#)

파라미터 세부 정보

이름	설명
/prod/config/api-url	-
ARN	데이터 형식
 arn:aws:ssm:us-east-1:891376949960:parameter/prod/config/api-url	text
계층	마지막으로 수정한 사용자
<div>Standard</div>	arn:aws:iam::891376949960:root
유형	마지막 수정 날짜
String	Mon, 16 Dec 2024 08:16:43 GMT
값	버전
http://localhost:3000	1

3단계: Lambda 함수 개발

Lambda 함수 생성

1.Lambda 함수의 IAM 역할

- 미리 만들어 둔
LambdaSecretsRole
역할을 사용해서
Lambda 생성

2.Secrets Manager와 Parameter Store 연동

2. Lambda 함수에서
AWS SDK를 사용해
비밀 값과 파라미터
를 가져옴.

Lambda > 함수 > 함수 생성

함수 생성 정보

다음 옵션 중 하나를 선택하여 함수를 생성합니다.

☒ 새로 작성

간단한 Hello World 예제는 시작하십시오.

☐ 블루프린트 사용

샘플 코드 및 구축 Lambda 애플리케이션을 위한 구성 사전 설정을 일반적인 사용 사례를 살펴봅니다.

☐ 컨테이너 이미지

함수에 대해 배포할 컨테이너 이미지를 선택합니다.

기본 정보

함수 이름

함수의 용도를 설명하는 이름을 입력합니다.

myLambda1

함수 이름은 1~64자여야 하고, 리전에 고유해야 하며, 공백을 포함할 수 없습니다. 유효한 문자는 a~z, A~Z, 0~9, 하이픈(-), 밑줄(_)입니다.

런타임 정보

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.13

아키텍처 정보

함수 코드에 대해 원하는 명령 세트 아키텍처를 선택합니다.

☒ x86_64

☐ arm64

권한 정보

기본적으로 Lambda는 Amazon CloudWatch Logs에 로그를 업로드하는 권한을 가진 실행 역할을 생성합니다. 이 기본 역할은 나중에 트리거를 추가할 때 사용자 지정할 수 있습니다.

▼ 기본 실행 역할 변경

실행 역할

함수에 대한 권한을 정의하는 역할을 선택합니다. 사용자 지정 역할을 생성하려면 IAM 콘솔로 이동하십시오.

☐ 기본 Lambda 권한을 가진 새 역할 생성

☒ 기존 역할 사용

☐ AWS 정책 템플릿에서 새 역할 생성

기존 역할

생성한 기존 역할 중에 이 Lambda 함수와 함께 사용할 역할을 선택합니다. 이 역할에는 Amazon CloudWatch Logs에 로그를 업로드할 수 있는 권한이 있어야 합니다.

LambdaSecretsRole

IAM 콘솔에서 LambdaSecretsRole 역할을 확인합니다.

함수 생성 [정보](#)

다음 옵션 중 하나를 선택하여 함수를 생성합니다.

☒ 새로 작성

간단한 Hello World 예제는 시작하십시오.

☐ 블루프린트 사용

샘플 코드 및 구축 Lambda 애플리케이션을 위한 구성 사전 설정을 일반적인 사용 사례를 살펴봅니다.

☐ 컨테이너 이미지

함수에 대해 배포할 컨테이너 이미지를 선택합니다.

기본 정보

함수 이름

함수의 용도를 설명하는 이름을 입력합니다.

myLamda1

함수 이름은 1~64자여야 하고, 리전에 고유해야 하며, 공백을 포함할 수 없습니다. 유효한 문자는 a~z, A~Z, 0~9, 하이픈(-), 밑줄(_)입니다.

런타임 [정보](#)

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.13



아키텍처 [정보](#)

함수 코드에 대해 원하는 명령 세트 아키텍처를 선택합니다.

☒ x86_64

☐ arm64

권한 [정보](#)

기본적으로 Lambda는 Amazon CloudWatch Logs에 로그를 업로드하는 권한을 가진 실행 역할을 생성합니다. 이 기본 역할은 나중에 트리거를 추가할 때 사용자 지정할 수 있습니다.

▼ 기본 실행 역할 변경

실행 역할

함수에 대한 권한을 정의하는 역할을 선택합니다. 사용자 지정 역할을 생성하려면 [IAM 콘솔](#)으로 이동하십시오.

☐ 기본 Lambda 권한을 가진 새 역할 생성

☒ 기존 역할 사용

☐ AWS 정책 템플릿에서 새 역할 생성

기존 역할

생성한 기존 역할 중에 이 Lambda 함수와 함께 사용할 역할을 선택합니다. 이 역할에는 Amazon CloudWatch Logs에 로그를 업로드할 수 있는 권한이 있어야 합니다.

LambdaSecretsRole



IAM 콘솔에서 [LambdaSecretsRole](#) 역할을 확인합니다.

Lambda 코드 개발

파이썬으로 작성

```
import boto3
import logging
from botocore.exceptions import ClientError

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    # AWS 클라이언트 초기화
    secrets_manager = boto3.client('secretsmanager')
    ssm = boto3.client('ssm')

    try:
        # Secrets Manager에서 데이터 호출
        secret_name = "prod/db-credentials"
        secret_response = secrets_manager.get_secret_value(SecretId=secret_name)
        secret = secret_response['SecretString']
        logger.info(f"Fetched secret: {secret}")
    except ClientError as e:
        logger.error(f"Failed to fetch secret {secret_name}: {e}")
        return {
            "statusCode": 500,
            "body": f"Error fetching secret: {e}"
        }
```

```
try:
    # Parameter Store에서 데이터 호출
    parameter_name = "/prod/config/api-url"
    parameter_response = ssm.get_parameter(Name=parameter_name, WithDecryption=False)
    parameter = parameter_response['Parameter']['Value']
    logger.info(f"Fetched parameter: {parameter}")
except ClientError as e:
    logger.error(f"Failed to fetch parameter {parameter_name}: {e}")
    return {
        "statusCode": 500,
        "body": f"Error fetching parameter: {e}"
    }

# 결과 반환
return {
    "statusCode": 200,
    "body": {
        "secret": secret,
        "parameter": parameter
    }
}
```

- 로깅:

- logger.info()와 logger.error()를 사용하여 실행 기록 및 에러를 로그에 남깁니다.이 로그는 **CloudWatch Logs**에 자동으로 전송됩니다.

- AWS 클라이언트:

- boto3 라이브러리를 사용하여 **Secrets Manager**와 **SSM Parameter Store**에 접근합니다.

- 에러 처리:

- try-except 블록으로 오류를 처리하며, 발생한 오류를 로깅하고 반환합니다.

- Lambda 실행 결과:

- 성공 시 secret과 parameter를 반환합니다.

- 환경 변수 설정

- Lambda 함수 환경 변수에 SECRET_NAME과 PARAMETER_NAME을 추가.
 - secret_name: prod/db-credentials
 - parameter_name: /prod/config/api-url

- 패키지 종속성 관리

- boto3 패키지 사용. Lambda의 Python 런타임 환경에 포함되어 있음.

[Test 실행]

1. 테스트 이벤트 생성

- 상단 메뉴에서 "**테스트**" 버튼을 클릭합니다.
- 팝업 창에서 새 테스트 이벤트를 생성합니다:
 - 이벤트 템플릿**: Lambda가 처리할 이벤트 구조를 선택합니다. 예를 들어, AWS 서비스와의 통합 여부에 따라 템플릿을 선택합니다.
 - 이벤트 이름**: 테스트 이벤트의 이름을 지정합니다. 예: test-event-1
 - 이벤트 JSON**: Lambda 함수가 처리할 입력 데이터를 JSON 형식으로 작성합니다.

```
{ "key1": "value1", "key2": "value2", "key3": "value3" }
```

2. 테스트 이벤트 저장

- 작성된 이벤트를 저장합니다.

3. 테스트 실행

- "**테스트**" 버튼을 클릭하여 테스트를 실행합니다.
- 실행 결과는 화면 아래 "**Execution results**" 섹션에 표시됩니다:
 - Status**: 성공(Success) 또는 실패(Error).
 - Log output**: 로그 메시지 확인.

[Test 결과]

Status: Succeeded

Test Event Name: TestEvent

Response:

```
{
  "statusCode": 200,
  "body": {
    "secret": "{\"username\":\"admin\",\"password\":\"MYPASSWORD\",\"engine\":\"mysql\",\"host\":\"database-1.c7kg62g660d6.us-east-1.rds.amazonaws.com\",\"port\":3306,\"dbInstanceIdentifier\":\"database-1\"}\",",
    "parameter": "http://localhost:3000"
  }
}
```

Function Logs:

START RequestId: 42de3ecd-7f74-4066-a2e7-40a7e7540dbf Version: \$LATEST

[INFO] 2024-12-17T08:41:47.456Z 42de3ecd-7f74-4066-a2e7-40a7e7540dbf Found credentials in environment variables.

[INFO] 2024-12-17T08:41:50.041Z 42de3ecd-7f74-4066-a2e7-40a7e7540dbf Fetched secret: {"username":"admin",
"password":"MYPASSWORD","engine":"mysql","host":"database-1.c7kg62g660d6.us-east-1.rds.amazonaws.com","port":3306,
"dbInstanceIdentifier":"database-1"}

[INFO] 2024-12-17T08:41:50.534Z 42de3ecd-7f74-4066-a2e7-40a7e7540dbf Fetched parameter: http://localhost:3000

END RequestId: 42de3ecd-7f74-4066-a2e7-40a7e7540dbf

REPORT RequestId: 42de3ecd-7f74-4066-a2e7-40a7e7540dbf Duration: 3377.13 ms Billed Duration: 3378 ms Memory
Size: 128 MB Max Memory Used: 81 MB Init Duration: 375.09 ms

Request ID: 42de3ecd-7f74-4066-a2e7-40a7e7540dbf

성공적인 실행:

•Lambda 함수는
성공적으로
실행되었으며,
예상된 데이터를
반환했습니다.

4단계: 보안 및 로깅 설정

AWSLambdaBasicExecutionRole 을 포함한 IAM 역할을 사용해 클라우드 워치에 자동으로 로그를 생성 했습니다

CloudWatch > 로그 그룹 > /aws/lambda/myLambda

CloudWatch

로그 그룹

ARN

arn:aws:logs:us-east-1:891376949960:log-group:/aws/lambda/myLambda:

생성 시간

1일 전

부존

만기 없음

저장된 바이트

6.48 KB

구독 필터

0

기여자 인사이트 규칙

-

KMS 키 ID

-

이상 탐지

구성

인감한 데이터 개수

-

필드 인덱스

구성

트랜스포머

구성

로그 스트림

태그

이상 탐지

지표 필터

구독 필터

기여자 인사이트

데이터 보호

필드 인덱스 - 신규

트랜스포머 - 신규

로그 스트림 (10)

로그 스트림 필터링 또는 접두어 검색 시도

☐ 정확히 일치 ☐ 만드된 항목 표시 [정보](#)

☐ 로그 스트림

마지막 이벤트 시간

<input type="checkbox"/>	2024/12/17/[LATEST]28181a2cc8704397a8c23dbfcd058388	2024-12-17 14:58:54 (UTC)
<input type="checkbox"/>	2024/12/17/[LATEST]03fc3885b59a4551977f5281b9537be9	2024-12-17 14:57:59 (UTC)
<input type="checkbox"/>	2024/12/17/[LATEST]0220a0fc0c93418ab4bb8ee99b877228	2024-12-17 08:41:50 (UTC)
<input type="checkbox"/>	2024/12/17/[LATEST]6c870fd7aace472b83d8611bb69edd81	2024-12-17 04:36:50 (UTC)
<input type="checkbox"/>	2024/12/17/[LATEST]3fafc98f61c74918bf96a9b98472f4f4	2024-12-17 04:35:22 (UTC)
<input type="checkbox"/>	2024/12/17/[LATEST]e81037350afe4b3d97f049355761ef02	2024-12-17 04:28:29 (UTC)
<input type="checkbox"/>	2024/12/17/[LATEST]d3fa57b313654b73a4bd8bd410a52c60	2024-12-17 04:24:25 (UTC)
<input type="checkbox"/>	2024/12/17/[LATEST]979b6fbaedc4f71ada64c7dcc9f866c	2024-12-17 04:23:23 (UTC)
<input type="checkbox"/>	2024/12/16/[LATEST]9e868967ea59470485b1f50eae4f2324	2024-12-16 08:35:21 (UTC)
<input type="checkbox"/>	2024/12/16/[LATEST]1e8bc74ea59b4c57b17e2e610c70efb1	2024-12-16 08:33:44 (UTC)

현재 상태

•여러 로그 스트림이 시간별로 나열되어 있으므로, Lambda 함수가 **정상적으로 실행**되고 있다는 의미입니다.

•**로그 스트림의 ID**는 실행된 Lambda 함수에 대한 고유 식별자입니다. 여러 개의 로그 스트림이 있다는 것은 여러 번 Lambda 함수가 호출되었음을 나타냅니다.

•**마지막 이벤트 시간**은 각 로그 스트림의 가장 최근 실행 시각을 보여주며, 이는 Lambda 함수가 마지막으로 실행된 시점을 확인할 수 있게 해줍니다.

<에러를 일으키기 위해 임의로 제작한 Lamda code>

```
import boto3
import logging
import json
import time
import random
from botocore.exceptions import ClientError

# Logger 설정
logger = logging.getLogger()
logger.setLevel(logging.INFO)

# CloudWatch Logs 클라이언트 초기화
logs_client = boto3.client('logs')

# CloudWatch Log Group과 Stream 이름 설정
LOG_GROUP_NAME = "TestLogGroup"
LOG_STREAM_NAME = "TestLogStream"

def create_log_group_and_stream():
    """CloudWatch Logs의 Log Group과 Log Stream 생성"""
    # 로그 그룹 생성
    try:
        logs_client.create_log_group(logGroupName=LOG_GROUP_NAME)
        logger.info(f"Log group {LOG_GROUP_NAME} created.")
    except logs_client.exceptions.ResourceAlreadyExistsException:
        logger.info(f"Log group {LOG_GROUP_NAME} already exists.")

    # 로그 스트림 생성
    try:
        logs_client.create_log_stream(logGroupName=LOG_GROUP_NAME, logStreamName=LOG_STREAM_NAME)
        logger.info(f"Log stream {LOG_STREAM_NAME} created.")
    except logs_client.exceptions.ResourceAlreadyExistsException:
        logger.info(f"Log stream {LOG_STREAM_NAME} already exists.")
```

```
def put_log_event(message, timestamp):
    """CloudWatch Logs에 로그 이벤트 추가"""
    try:
        logs_client.put_log_events(
            logGroupName=LOG_GROUP_NAME,
            logStreamName=LOG_STREAM_NAME,
            logEvents=[
                {
                    'timestamp': timestamp,
                    'message': message
                }
            ]
        )
        logger.info(f"Log sent: {message}")
    except ClientError as e:
        logger.error(f"Failed to send log: {e}")
```

```

# 비정상 로그 추가 (CloudWatch에 감지될 수 있는 패턴)
for i in range(3):
    timestamp = int(time.time() * 1000)
    put_log_event(f"ERROR: Simulated critical error log {i}", timestamp)
    time.sleep(0.2) # 로그 생성 간 간격 추가

# 의도적으로 예외 발생
try:
    raise Exception("Simulated Critical Error")
except Exception as e:
    logger.error(f"Critical error occurred: {e}")
    timestamp = int(time.time() * 1000)
    put_log_event(f"ERROR: Critical error - {e}", timestamp)
    raise # 재발생시켜 Lambda 함수 상태가 실패로 기록되도록 함

# 결과 반환
return {
    "statusCode": 200,
    "body": json.dumps({
        "message": "Function executed successfully",
        "secret": secret,
        "parameter": parameter
    })
}

```

```

try:
    # Parameter Store에서 데이터 호출
    parameter_name = "/prod/config/api-url"
    parameter_response = ssm.get_parameter(Name=parameter_name, WithDecryption=False)
    parameter = parameter_response['Parameter']['Value']
    logger.info(f"Fetched parameter: {parameter}")
    # 정상 로그 기록
    timestamp = int(time.time() * 1000)
    put_log_event(f"INFO: Fetched parameter successfully - {parameter_name}", timestamp)
except ClientError as e:
    logger.error(f"Failed to fetch parameter {parameter_name}: {e}")
    # 비정상 로그 기록
    timestamp = int(time.time() * 1000)
    put_log_event(f"ERROR: Failed to fetch parameter - {parameter_name}, Error: {e}")
    return {
        "statusCode": 500,
        "body": f"Error fetching parameter: {e}"
    }

```

```

def lambda_handler(event, context):
    # AWS 클라이언트 초기화
    secrets_manager = boto3.client('secretsmanager')
    ssm = boto3.client('ssm')

    # 로그 그룹 및 스트림 생성
    create_log_group_and_stream()

    try:
        # Secrets Manager에서 데이터 호출
        secret_name = "prod/db-credentials"
        secret_response = secrets_manager.get_secret_value(SecretId=secret_name)
        secret = secret_response['SecretString']
        logger.info(f"Fetched secret: {secret}")
        # 정상 로그 기록
        timestamp = int(time.time() * 1000)
        put_log_event(f"INFO: Fetched secret successfully - {secret_name}", timestamp)
    except ClientError as e:
        logger.error(f"Failed to fetch secret {secret_name}: {e}")
        # 비정상 로그 기록
        timestamp = int(time.time() * 1000)
        put_log_event(f"ERROR: Failed to fetch secret - {secret_name}, Error: {e}", timestamp)
    return {
        "statusCode": 500,
        "body": f"Error fetching secret: {e}"
    }

```

```
Status: Failed
Test Event Name: TestEvent

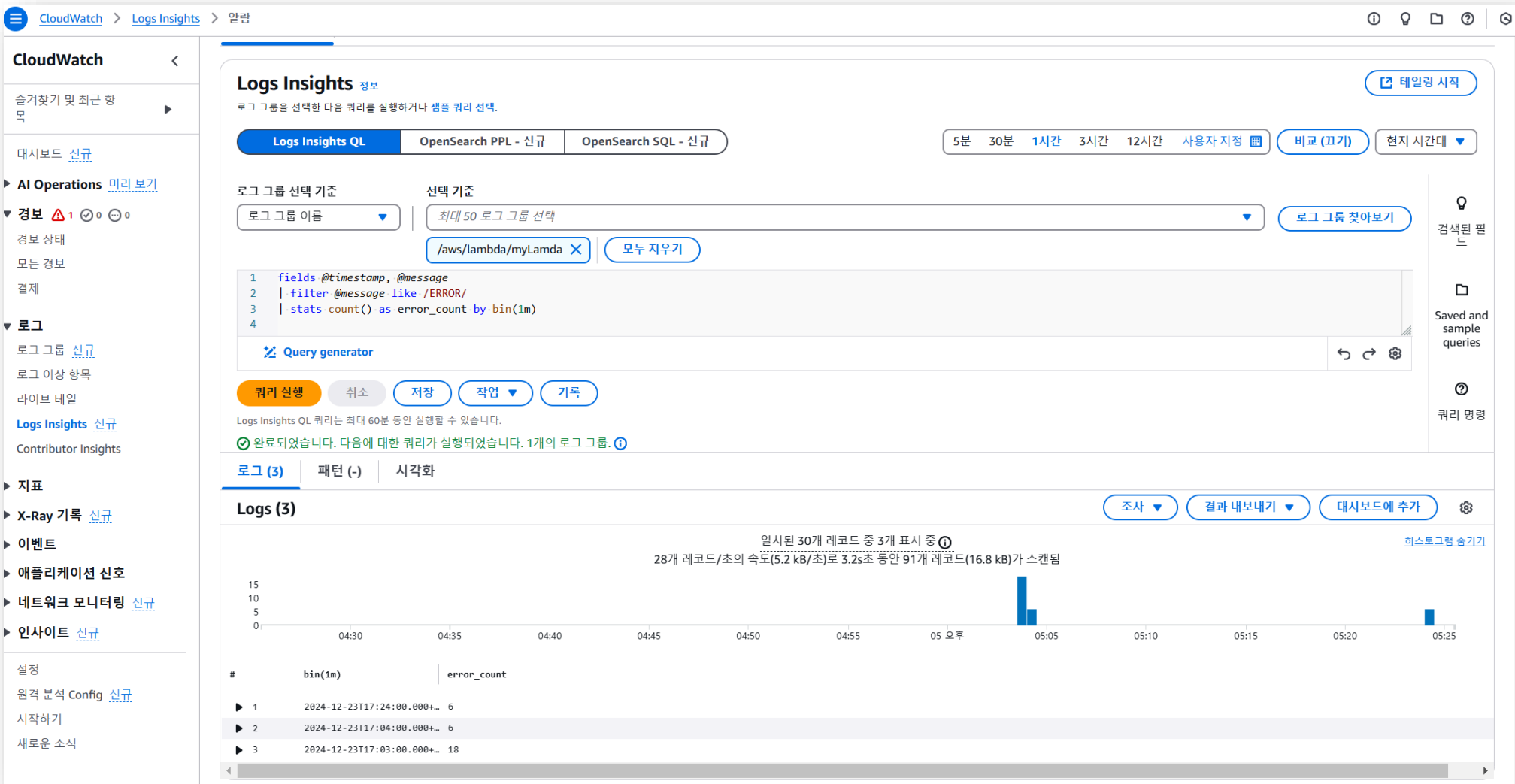
Response:
{
  "errorMessage": "Simulated Critical Error",
  "errorType": "Exception",
  "requestId": "dd861706-f4d9-4387-b6c1-22a68a364f14",
  "stackTrace": [
    "   File \"/var/task/lambda_function.py", line 106, in lambda_handler\n    raise Exception(\"Simulated Critical Error\")\n"
  ]
}

Function Logs:
[INFO] 2024-12-23T08:24:05.176Z          Found credentials in environment variables.
START RequestId: dd861706-f4d9-4387-b6c1-22a68a364f14 Version: $LATEST
[INFO] 2024-12-23T08:24:06.445Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log group TestLogGroup already exists.
[INFO] 2024-12-23T08:24:06.983Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log stream TestLogStream already exists.
[INFO] 2024-12-23T08:24:07.502Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Fetched secret: {"username":"admin","password":"MYPASSWORD","engine":"mysql",
"host":"database-1.c7kg62g660d6.us-east-1.rds.amazonaws.com","port":3306,"dbInstanceIdentifier":"database-1"}
[INFO] 2024-12-23T08:24:08.012Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log sent: INFO: Fetched secret successfully - prod/db-credentials
[INFO] 2024-12-23T08:24:08.506Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Fetched parameter: http://localhost:3000
[INFO] 2024-12-23T08:24:08.523Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log sent: INFO: Fetched parameter successfully - /prod/config/api-url
[INFO] 2024-12-23T08:24:08.563Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log sent: ERROR: Simulated critical error log 0
[INFO] 2024-12-23T08:24:08.783Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log sent: ERROR: Simulated critical error log 1
[INFO] 2024-12-23T08:24:09.002Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log sent: ERROR: Simulated critical error log 2
[ERROR] 2024-12-23T08:24:09.203Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Critical error occurred: Simulated Critical Error
[INFO] 2024-12-23T08:24:09.215Z          dd861706-f4d9-4387-b6c1-22a68a364f14    Log sent: ERROR: Critical error - Simulated Critical Error
LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the function code. However, in rare cases, a Lambda runtime update can cause unexpected function behavior. For functions using managed runtimes, runtime updates can be triggered by a function change, or can be applied automatically. To determine if the runtime has been updated, check the runtime version in the INIT_START log entry. If this error correlates with a change in the runtime version, you may be able to mitigate this error by temporarily rolling back to the previous runtime version. For more information, see https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtime-versions.html.
```

실행시키면 에러가 뜨는 것을 확인할 수 있습니다
이 에러를 탐지하기 위해서 아래 내용들을 설정했습니다.

1. Logs Insights 쿼리
2. 로그그룹 이상탐지
3. 경고 설정

Logs Insights 쿼리



ERROR라고 뜬 로그들을 카운트해서
합산하여 시간별로 막대 그래프로 보여주는 쿼리

경보 설정

- 1 단계
- 패턴 정의
- 2 단계
- 지표 할당
- 3 단계
- Review and confirm

패턴 정의

필터 패턴 생성

지표 필터를 사용하여 CloudWatch Logs로 전송되는 로그 그룹에서 이벤트를 모니터링할 수 있습니다. 특정 항목 모니터링 및 집계하거나 로그 이벤트에서 값을 추출하고 결과를 특정 지표에 연결할 수 있습니다. [패턴 구문에 대해 자세히 알아보십시오.](#)

패턴 필터링

로그 이벤트에서 일치될 항목 또는 패턴을 지정하여 지표를 생성합니다.

Q ERROR X

☐ 변환된 로그에 지표 필터 활성화
활성화하면 변환된 로그에 지표 필터가 적용됩니다. 비활성화하면 원래 로그에 지표 필터가 적용됩니다.

패턴 테스트

테스트할 로그 데이터 선택

로그 데이터 사용자 지정 ▼

로그 이벤트 메시지

필터 패턴을 사용하여 테스트할 로그 데이터를 입력합니다. 줄 바꿈을 사용하여 로그 이벤트를 구분하십시오.

[83078518-fcc1-4d30-9573-8b9737671438] BENCHMARK : Running Start Crawl for Crawler TestCrawler2
[83078518-fcc1-4d30-9573-8b9737671438] BENCHMARK : Classification complete, writing results to database mygluedatabase
[83078518-fcc1-4d30-9573-8b9737671438] INFO : Crawler configured with SchemaChangePolicy {"UpdateBehavior":"UPDATE_IN_DATABASE","DeleteBehavior":"DEPRECATE_IN_DATABASE"}.
[83078518-fcc1-4d30-9573-8b9737671438] INFO : Created table gluetest in database mygluedatabase

패턴 테스트

결과

위의 로그 이벤트 메시지를 선택하고 '패턴 테스트'를 클릭하여 결과를 확인하십시오.

ERROR를 카운트하는 지표를 만들어서 해당 지표를 넣어 경보를 제작하였습니다.

찾아보기

다중 소스 쿼리

그래프로 표시된 지표

옵션

소스

지표 (1)

N. Virginia ▼

모두 > ... > 차원을 포함하지 않은 지표

모든 지표, 측정기준

☐

지표 이름 1/1 ▲

경보

☐

ErrorCount

1개의 경보



로그그룹 이상탐지

/aws/lambda/myLamda

작업 ▼ | Logs Insights에서 보기 | 테일링 시작 | 로그 그룹 검색

▶ 로그 그룹 세부 정보

로그 스트림 | 태그 | 이상 탐지 | 지표 필터 | 구독 필터 | 기여자 인사이트 | 데이터 보호 | 필드 인덱스 - 신규 | 트랜스포머 - 신규

이상 탐지 (2) 정보

경보 생성 | 작업 ▼ | Logs Insights에서 보기

최신 50개의 이상 항목이 1분마다 자동으로 업데이트됨

Q 우선순위, 패턴 또는 키워드로 이상 항목 필터링

이상 항목 | 억제됨 | < 1 > | ⚙

<input type="checkbox"/>	Inspect	이상 항목 ▼	우선 순위 ▼	로그 패턴 ▼	이상 로그 추세	마지막 탐지 시간 ▼	최초	정확한가요?
<input type="checkbox"/>		Unexpected pattern detected with severity ERROR	중간	[ERROR] Exception: Simulated Critical Error Traceback (most recent call last): File Token-1 , line Number-2 , in lambda_handler raise Exception("Simulated Critical Error")		27분 전	13초	
<input type="checkbox"/>		Unexpected pattern detected with severity ERROR	중간	[ERROR] Token-1UUID-2 Critical error occurred: Simulated Critical Error		27분 전	13초	

비슷하게 로그그룹에 이상탐지를 설정해서 에러가 나면 이상이 탐지되도록 설정했습니다.

해당 프로젝트는 애플리케이션의 구성 데이터를 관리하고
감사 및 로그 모니터링을 좀 더 손쉽게 하며

다중 환경에서 민감 데이터를 처리할 수 있고 (예: /dev/, /test/ 등의 경로)
데이터베이스 자격증명을 하드코딩 하지 않고 사용할 수 있습니다.

또한 이 코드는 DevOps 환경에서 중요한 데이터를 안전하게 관리하며,
CI/CD 파이프라인에서 동적으로 호출될 수 있으며,

CloudWatch 대시보드와 Logs Insights를 활용하여 로그 데이터를 분석하고,
이를 기반으로 시스템 동작을 시각화할 수 있습니다.

가장 다양한 형태로 응용이 가능하도록 만들기 위해서 프로젝트를
최대한 기본이 되는 형태로 제작했습니다.