AWS 솔루션

AWS 솔루션 구성체

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

AWS 솔루션 구성체: AWS 솔루션

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

Amazon의 상표 및 트레이드 드레스는 Amazon이 제공하지 않는 제품 또는 서비스와 관련하여 고객에게 혼동을 유발할 수 있는 방식 또는 Amazon을 폄하하거나 평판에 악영향을 주는 방식으로 사용될 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계 여부에 관계없이 해당 소유자의 자산입니다.

Table of Contents

개요 1
- AWS 솔루션 구성란 무엇입니까? 1
- AWS 솔루션 구문을 사용하는 이유는 무엇입니까? 1

시작하기 3
- 사전 조건 3
- AWS CDK 설치 4
- AWS 솔루션 구성 작업 4
- 이동장면 - 1부 4
  - Hello 구조 5
  - 애플리케이션 디렉토리 생성 및 AWS CDK 초기화 5
  - 프로젝트 기반 종속성 업데이트 6
  - Lambda 핸들 코드 9
  - AWS CDK 및 AWS 솔루션 구축 종속성 설치 10
  - 스택에 Amazon API 게이트웨이/AWS 람다 패턴 추가 12
  - cdk 배포 18
  - 스택 출력 18
  - 앱 테스트 18
- 이동장면 - 부품 2 19
  - Lambda 코드 히트 20
  - 새 종속성 설치 22
  - 리소스를 정의합니다. 23
  - 변경 사항 검토 36
  - cdk 배포 37
  - 스택 출력 38
  - 앱 테스트 38
- 예제 사용 사례 39
  - AWS 정적 S3 웹 사이트 39
  - AWS 단순 서버를 사용하지 않는 이미지 처리기 40
  - AWS 서버리스 웹 앱 40

API 참조 41
- 모듈 41
- 모듈 컨텐츠 41
- AWS-아피가티웨이-다이나모DB 42
  - 개요 42

이니셜 라이저 43
패턴 구성 43
패턴 속성 44
기본 설정 45
아키텍처 46
GitHub 46
아피가테이웨이 IoT 46
개요 47
이니셜 라이저 47
소품 패턴 구성 48
패턴 등록 정보 49
기본 설정 49
아키텍처 52
예제: 52
GitHub 54
aws-apigateway-키네시스스트림 54
개요 55
이니셜 라이저 55
패턴 구성 55
패턴 속성 57
샘플 API 사용 57
기본 설정 58
아키텍처 59
GitHub 59
aws-아피가티웨이 람다 60
개요 60
이니셜 라이저 61
패턴 구성 61
패턴 속성 62
기본 설정 62
아키텍처 63
GitHub 63
아피가테이웨이 세이지메이크렌드포인트 63
개요 64
이니셜 라이저 65
PropingTemplate 65

패턴 속성 67
샘플 API 사용 57
기본 설정 68
아키텍처 68
GitHub 69
aws-apigateway-sqs 69
개요 69
이니셜 라이저 70
소품 패턴 구성 70
패턴 속성 72
API 사용 57
기본 설정 73
아키텍처 74
GitHub 74
클라우드 프론트 어피가트웨이 74
개요 75
초기화 프로그램 76
소품 패턴 구성 76
패턴 속성 77
기본 설정 77
아키텍처 78
GitHub 78
aws-클라우드 프론트 - 아피 가티 웨이 - 람다 78
개요 79
이니셜 라이저 80
패턴 구성 80
패턴 속성 81
기본 설정 82
아키텍처 83
GitHub 83
AWS 클라우드 프런트 미디어 스토어 83
개요 84
이니셜 라이저 84
패턴 구성 소품 85
패턴 특성 85
기본 설정 86

아키텍처 87
GitHub 88
AWS-클라우드프런트 S3 88
개요 88
이니셜 라이저 89
패턴 구성 89
패턴 속성 90
기본 설정 90
아키텍처 91
GitHub 91
aws-코그니토-아피가테이웨이 - 람다 92
개요 75
이니셜 라이저 93
패턴 구성 94
패턴 속성 95
기본 설정 96
아키텍처 97
GitHub 97
AWS-Dynamodb-stream-lambda-lambda-stream 97
개요 98
이니셜 라이저 99
패턴 구성 소품 99
패턴 속성 100
Lambda 함수 100
기본 설정 100
아키텍처 101
GitHub 101
aws-다이나모드-스트림-람다-탄성 검색-키바나 102
개요 102
이니셜 라이저 103
소품 패턴 구성 103
패턴 속성 104
Lambda 함수 105
기본 설정 105
아키텍처 107
GitHub 107

AWS-이벤트-규칙-키네시스파이어호스-s3 107
개요 108
이니셜 라이저 109
패턴 구성 109
패턴 속성 110
기본 설정 111
아키텍처 112
GitHub 112
AWS-이벤트-규칙-키네시스스트림 112
개요 113
이니셜 라이저 113
Props 패턴 구성 114
패턴 속성 114
기본 설정 115
아키텍처 116
GitHub 116
aws-이벤트-규칙 - 람다 116
개요 117
이니셜 라이저 118
패턴 구성 118
패턴 속성 119
기본 설정 119
아키텍처 120
GitHub 120
aws-이벤트-규칙-sns 120
개요 121
이니셜 라이저 122
패턴 구성 122
패턴 속성 123
기본 설정 123
아키텍처 124
GitHub 124
AWS-이벤트-규칙-sqs 124
개요 125
이니셜 라이저 126
패턴 구성 126

패턴 속성 128
기본 설정 128
아키텍처 129
GitHub 129
AWS-이벤트-규칙-스텝 기능 129
개요 130
이니셜 라이저 131
패턴 구성 131
패턴 속성 132
기본 설정 132
아키텍처 133
GitHub 133
AWS-이오트키네시스파이어호스-3 133
개요 134
이니셜 라이저 135
패턴 구성 135
패턴 속성 136
기본 설정 137
아키텍처 138
GitHub 138
aws-iot-람다 138
개요 139
이니셜 라이저 140
패턴 구성 140
패턴 속성 141
기본 설정 141
아키텍처 142
GitHub 142
AWS-Iot-람다-다이나모DB 142
개요 143
이니셜 라이저 144
패턴 구성 144
패턴 속성 145
기본 설정 145
아키텍처 146
GitHub 147

AWS-키네시스파이어호스-3 147
개요 147
이니셜 라이저 148
Props 패턴 구성 148
패턴 속성 149
기본 설정 150
아키텍처 151
GitHub 151
AWS 키네시스파이어호스-S3 및 키네시스분석 151
개요 152
이니셜 라이저 153
패턴 구성 153
패턴 속성 154
기본 설정 155
아키텍처 156
GitHub 156
AWS-키네시스스트림-접착제 작업 156
개요 157
이니셜 라이저 158
소품 패턴 구성 159
싱크데이터스토어 160
싱크스토어 유형 161
기본 설정 161
아키텍처 162
GitHub 163
AWS-키네시스스트림-키네시스파이어호스-3 163
개요 163
이니셜 라이저 164
패턴 구성 164
패턴 속성 165
기본 설정 166
아키텍처 167
GitHub 167
AWS-키네시스스트림-람다 168
개요 168
이니셜 라이저 169

Propction 구성 169
패턴 속성 170
기본 설정 171
아키텍처 172
GitHub 172
aws-lambda-dynamodb 172
개요 173
이니셜 라이저 173
패턴 구성 174
패턴 속성 176
기본 설정 177
아키텍처 178
GitHub 178
aws-람다 탄성검색-키바나 178
개요 179
이니셜 라이저 180
ProptoPool 180
패턴 속성 181
Lambda 함수 182
기본 설정 182
아키텍처 183
GitHub 184
aws-람다-s3 184
개요 184
이니셜 라이저 185
패턴 185
패턴 188
기본 설정 188
아키텍처 189
GitHub 189
AWS-람다-ssm문자열 매개 변수 189
개요 190
이니셜 라이저 191
Prop 구성 191
패턴 속성 194
기본 설정 194

아키텍처 195
GitHub 195
람다-세이지메이크렌드포인트 196
개요 196
이니셜 라이저 197
패턴 구성 197
패턴 속성 200
기본 설정 201
아키텍처 202
GitHub 202
aws-람다 비밀 관리자 203
개요 203
이니셜 라이저 204
패턴 구성 204
패턴 속성 207
기본 설정 207
아키텍처 208
GitHub 208
aws-람다-sns 208
개요 209
이니셜 라이저 210
패턴 구성 210
패턴 등록 정보 212
기본 설정 213
아키텍처 214
GitHub 214
람다 스퀘어 214
개요 215
이니셜 라이저 215
패턴 구성 216
패턴 속성 219
기본 설정 219
아키텍처 220
GitHub 221
람다 스퀘어 람다 221
개요 221

이니셜 라이저 222
패턴 구성 222
패턴 속성 224
기본 설정 225
아키텍처 226
GitHub 226
aws-람다 단계 함수 226
개요 227
이니셜 라이저 228
패턴 구성 228
패턴 속성 229
기본 설정 229
아키텍처 231
GitHub 231
aws-s3-람다 231
개요 232
이니셜 라이저 233
패턴 구성 233
패턴 속성 234
기본 설정 234
아키텍처 235
GitHub 235
AWS-s3-sqs 236
개요 236
이니셜 라이저 237
패턴 구성 237
패턴 속성 239
기본 설정 239
아키텍처 240
GitHub 241
AWS-s3-스텝 기능 241
개요 241
이니셜 라이저 242
Prop 패턴 구성 243
패턴 속성 244
기본 설정 244

아키텍처 246
GitHub 246
람다 246
개요 247
이니셜 라이저 247
패턴 구성 248
패턴 속성 249
기본 설정 249
아키텍처 250
GitHub 250
AWS-SNS-스퀘어 250
개요 251
이니셜 라이저 251
패턴 구성 252
패턴 속성 253
기본 설정 254
아키텍처 255
GitHub 255
aws-sqs-람다 255
개요 256
이니셜 라이저 256
패턴 구성 257
패턴 속성 258
기본 설정 258
아키텍처 259
GitHub 259
core 260
AWS CDK 구조의 기본 속성 260
기본 속성 재정의 260
특성 재지정 경고 261
문서 수정 262
고지 사항 267
..... cclxviii

AWS 솔루션 구조

게시 날짜: 2021년 5월(문서 수정)

AWS 솔루션 구성란 무엇입니까?

AWS 솔루션 구조 (구문) 는AWS Cloud Development Kit (AWS CDK)는 예측 가능하고 반복 가능한 인프라를 만들기 위해 코드에서 솔루션을 신속하게 정의하기 위한 다중 서비스, 잘 설계된 패턴을 제공합니다. 목표는 개발자가 아키텍처에 대한 패턴 기반 정의를 사용하여 모든 규모의 솔루션을 빌드할 수 있는 환경을 가속화하는 것입니다.

AWS 솔루션 구문을 사용하여 익숙한 프로그래밍 언어로 솔루션을 정의합니다. AWS 솔루션 구문은 현재 TypeScript, JavaScript, 파이썬 및 자바를 지원합니다.

AWS 솔루션 구성 패턴의 전체 카탈로그를 찾아보려면여기를 클릭하십시오..

AWS 솔루션 구문을 사용하는 이유는 무엇입니까?

클라우드 프로바이더의 혁신 속도에 따라 모범 사례를 알고 이해하며 솔루션 전반에서 올바르게 구현되도록 보장하는 것은 매우 어려울 수 있습니다. 구성을 사용하면 확장 가능하고 안전한 방식으로 클라우드 서비스를 사용하여 일반적인 작업을 수행하는 사전 구축되고 잘 설계된 패턴과 사용 사례를 결합할 수 있습니다. Constructs는 최신 프로그래밍 언어에 대한 라이브러리를 제공하므로 솔루션을 위해 잘 설계된 클라우드 인프라를 구축하는 작업에 기존 개발 기술과 친숙한 도구를 적용할 수 있습니다.

AWS 솔루션 구성의 다른 이점은 다음과 같습니다.

- AWS Cloud Development Kit (AWS CDK) 오픈 소스 소프트웨어 개발 프레임워크를 기반으로 합니다.
- 솔루션 인프라를 정의할 때 논리 (if 문, for-loop 등) 를 사용합니다.
- 객체 지향 기술을 사용하여 시스템의 모델을 만듭니다.
- 상위 수준 추상화를 정의하고 공유하며 팀, 회사 또는 커뮤니티에 게시합니다.
- 솔루션을 논리적 모듈로 구성하십시오.
- 솔루션을 라이브러리로 공유하고 다시 사용할 수 있습니다.
- 업계 표준 프로토콜을 사용하여 인프라 코드를 테스트합니다.
- 기존 코드 검토 워크플로를 사용합니다.

AWS 솔루션 구축의 목적은 AWS에서 솔루션 목표를 달성하기 위해 잘 설계된 일반적인 패턴을 통합할 때 필요한 복잡성과 접착제 논리를 줄이는 것입니다.

AWS 솔루션 구성 시작하기

이 항목에서는 AWS CDK (Cloud Development Kit), AWS 솔루션 구성을 설치 및 구성하고 AWS 솔루션 구축 패턴을 사용하여 첫 번째 AWS CDK 앱을 만드는 방법에 대해 설명합니다.

> **Note**
>
> AWS 솔루션 구문은 AWS CDK 버전 이상 1.46.0에서 지원됩니다.

> **Tip**
>
> 더 깊이 파고 싶습니까? 사용해보기CDK 워크샵에서 실제 프로젝트에 대한 심층적 인 투어를 제공합니다.

> **Tip**
>
> AWS CDK (AWS Cloud Development Kit) 시작하기에 대한 자세한 내용은AWS CDK 개발자 안내서.

Prerequisites

AWS 솔루션 구문은 AWS CDK를 기반으로 구축되므로 TypeScript 스크립트 또는 JavaScript 이외의 언어로 작업하는 경우에도 Node.js (>= 10.3.0) 를 설치해야 합니다. 이 때문에AWS CDK및 AWS 솔루션 구문은 TypeScript 터로 개발되었으며 Node.js 에서 실행됩니다. 지원되는 다른 언어에 대한 바인딩은이 백엔드 및 도구 집합을 사용합니다.

자격 증명 및 리전 지정에 설명된 대로 AWS CDK CLI를 사용하려면 자격 증명과 AWS 리전을 제공해야 합니다.

기타 전제 조건은 다음과 같이 개발 언어에 따라 다릅니다.

언어	사전 조건
P	Python 3.6

언어	사전 조건
TS T t	TypeScript >= 2.7
J	Java 1.8

AWS CDK 설치

AWS CDK를 설치 및 구성하려면 AWS CDK 개발자 가이드 (AWS CDK 설치.

AWS 솔루션 구성 작업

AWS 솔루션 구성으로 작업할 때 새 앱을 만드는 일반적인 워크플로는 AWS CDK와 동일한 접근 방식을 따릅니다.

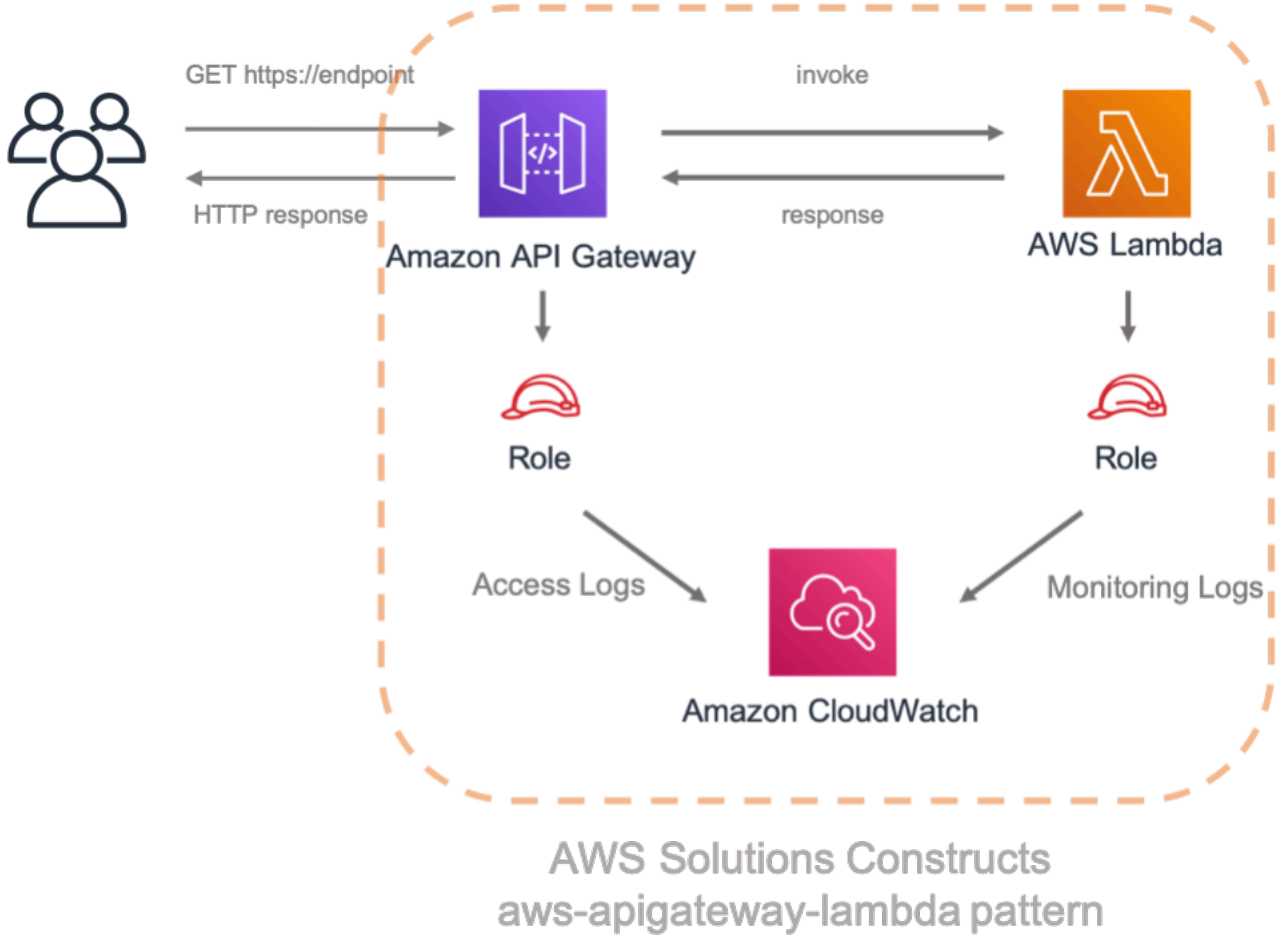
1. 앱 디렉토리를 만듭니다.
2. 앱을 초기화합니다.
3. AWS 솔루션 구성 패턴 종속성을 추가합니다.
4. 앱에 추가 코드를 추가합니다.
5. 필요한 경우 앱을 컴파일합니다.
6. 앱에 정의된 리소스를 배포합니다.
7. 앱을 테스트합니다.

문제가 있으면 수정, 컴파일 (필요한 경우), 배포 및 다시 테스트하십시오.

이동장면 - 1부

> **Note**
> AWS 솔루션 구조는 AWS CDK 버전 1.46.0 이상에서 지원됩니다.

이 자습서에서는 프로젝트 초기부터 결과 AWS CloudFormation 템플릿 배포에 이르기까지 AWS 솔루션 구조의 패턴을 사용하는 간단한 “Hello Constructs” AWS CDK 앱을 만들고 배포하는 방법을 안내합니다. Hello 구성 응용 프로그램은 다음과 같은 간단한 솔루션을 만듭니다:



Hello 구조

패턴 기반 개발을 사용하여 첫 번째 AWS CDK 앱 구축을 시작해 보겠습니다.

> **Note**
>
> 이것은의 샘플 수정입니다`Hello CDK!`(사용)CDK 워크숍. AWS CDK를 처음 사용하는 경우 이 워크숍부터 실습 연습과 CDK를 활용하여 실제 프로젝트를 구축하는 방법을 권장합니다.

애플리케이션 디렉토리 생성 및 AWS CDK 초기화

CDK 앱의 디렉토리를 생성한 다음 해당 디렉토리에 AWS CDK 앱을 생성합니다.

TypeScript

```
mkdir hello-constructs
cd hello-constructs
cdk init --language typescript
```

Python

```
mkdir hello-constructs
cd hello-constructs
cdk init --language python
```

> **Tip**
> 이제 좋아하는 IDE에서 프로젝트를 열고 탐색 할 수있는 좋은 시간입니다. 프로젝트 구조에 대한 자세한 내용은 해당 링크를 선택합니다.
>
> - TypeScript
> - Python

프로젝트 기반 종속성 업데이트

> **Warning**
> 적절한 기능을 보장하기 위해 AWS 솔루션 구성과 AWS CDK 패키지는 프로젝트 내에서 동일한 버전 번호를 사용해야 합니다. 예를 들어 AWS 솔루션 구성 v.1.52.0을 사용하는 경우 AWS CDK v.1.52.0도 사용해야 합니다.

> **Tip**
>
> AWS 솔루션 구성의 최신 버전을 메모하고 해당 버전 번호를`VERSION_NUMBER`자리 표시자를 참조하십시오 (AWS 솔루션 구성 및 AWS CDK 패키지 모두에 대해). 구성 라이브러리의 모든 공개 릴리스를 확인하려면여기를 클릭하십시오..

TypeScript

편집`package.json`파일에 다음 정보가 포함됩니다.

```
"devDependencies": {
  "@aws-cdk/assert": "VERSION_NUMBER",
  "@types/jest": "^24.0.22",
  "@types/node": "10.17.5",
  "jest": "^24.9.0",
  "ts-jest": "^24.1.0",
  "aws-cdk": "VERSION_NUMBER",
  "ts-node": "^8.1.0",
  "typescript": "~3.7.2"
},
"dependencies": {
  "@aws-cdk/core": "VERSION_NUMBER",
  "source-map-support": "^0.5.16"
}
```

Python

편집`setup.py`파일에 다음 정보가 포함됩니다.

```
install_requires=[
    "aws-cdk.core==VERSION_NUMBER",
],
```

프로젝트 기반 종속성을 설치하십시오.

TypeScript

```
npm install
```

Python

```
source .venv/bin/activate
pip install -r requirements.txt
```

앱을 빌드하고 실행하고 빈 스택을 생성하는지 확인합니다.

TypeScript

```
npm run build
cdk synth
```

Python

```
cdk synth
```

다음과 같이 스택이 표시되어야 합니다.`CDK-VERSION`는 CDK의 버전입니다. (출력은 여기에 표시된 것과 약간 다를 수 있습니다.)

TypeScript

```
Resources:
  CDKMetadata:
    Type: AWS::CDK::Metadata
    Properties:
          Modules: aws-cdk=CDK-VERSION,@aws-cdk/core=VERSION_NUMBER,@aws-cdk/cx-
api=VERSION_NUMBER,jsii-runtime=node.js/10.17.0
```

Python

```
Resources:
  CDKMetadata:
    Type: AWS::CDK::Metadata
    Properties:
          Modules: aws-cdk=CDK-VERSION,@aws-cdk/core=VERSION_NUMBER,@aws-cdk/cx-
api=VERSION_NUMBER,jsii-runtime=Python/3.7.7
```

Lambda 핸들 코드

AWS Lambda 핸들러 코드부터 시작하겠습니다.

디렉터리 생성`lambda`프로젝트 트리의 루트에 추가합니다.

TypeScript

이라는 파일을 추가합니다.`lambda/hello.js`다음의 콘텐츠가 포함됩니다.

```
exports.handler = async function(event) {
  console.log("request:", JSON.stringify(event, null, 2));
  return {
    statusCode: 200,
    headers: { "Content-Type": "text/plain" },
    body: `Hello, AWS Solutions Constructs! You've hit ${event.path}\n`
  };
};
```

Python

이라는 파일을 추가합니다.`lambda/hello.py`다음의 콘텐츠가 포함됩니다.

```
import json
```

```
def handler(event, context):
    print('request: {}'.format(json.dumps(event)))
    return {
        'statusCode': 200,
        'headers': {
            'Content-Type': 'text/plain'
        },
        'body': 'Hello, CDK! You have hit {}\n'.format(event['path'])
    }
```

이것은 텍스트를 반환하는 간단한 Lambda 함수입니다 “안녕하세요, 구조! [URL 경로] 를 눌렀습니다.” 함수의 출력에는 HTTP 상태 코드와 HTTP 헤더도 포함됩니다. API Gateway 에서 사용자에게 HTTP 응답을 공식화하는 데 사용됩니다.

이 Lambda JavaScript 제공됩니다. 선택한 언어로 Lambda 함수를 작성하는 방법에 대한 자세한 내용은AWS Lambda 설명서.

AWS CDK 및 AWS 솔루션 구축 종속성 설치

AWS 솔루션 구문은 광범위한 구조 라이브러리와 함께 제공됩니다. 라이브러리는 잘 설계된 각 패턴에 대해 하나씩 모듈로 나뉩니다. 예를 들어 AWS Lambda 함수에 Amazon API Gateway 레스트 API를 정의하려는 경우`aws-apigateway-lambda`패턴 라이브러리를 생성합니다.

또한 AWS CDK에서 AWS Lambda 및 Amazon API Gateway 구성 라이브러리를 추가해야 합니다.

AWS Lambda 모듈과 모든 종속성을 프로젝트에 설치합니다.

> **Note**
>
> AWS 솔루션 구문과 AWS CDK 모두에 사용할 올바른 일치 버전을`VERSION_NUMBER`각 명령에 대한 자리 표시자 필드입니다. 패키지 간에 버전이 일치하지 않으면 오류가 발생할 수 있습니다.

TypeScript

```
npm install -s @aws-cdk/aws-lambda@VERSION_NUMBER
```

Python

```
pip install aws_cdk.aws_lambda==VERSION_NUMBER
```

그런 다음 Amazon API Gateway 모듈과 모든 종속성을 프로젝트에 설치하십시오.

TypeScript

```
npm install -s @aws-cdk/aws-apigateway@VERSION_NUMBER
```

Python

```
pip install aws_cdk.aws_apigateway==VERSION_NUMBER
```

마지막으로, AWS 솔루션 구문을 설치합니다.`aws-apigateway-lambda`모듈과 모든 종속성을 프로젝트에 추가합니다.

TypeScript

```
npm install -s @aws-solutions-constructs/aws-apigateway-lambda@VERSION_NUMBER
```

Python

```
pip install aws_solutions_constructs.aws_apigateway_lambda==VERSION_NUMBER
```

스택에 Amazon API 게이트웨이/AWS 람다 패턴 추가

이제 AWS Lambda 프록시를 사용하여 Amazon API Gateway 구현하기 위한 AWS 솔루션 구성 패턴을 정의해 보겠습니다.

TypeScript

파일을 편집합니다.`lib/hello-constructs.ts`에서 다음을 사용합니다.

```
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-
constructs/aws-apigateway-lambda';

export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here
    const api_lambda_props: ApiGatewayToLambdaProps = {
      lambdaFunctionProps: {
        code: lambda.Code.fromAsset('lambda'),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'hello.handler'
      },
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };

    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}
```

Python

파일을 편집합니다.`hello_constructs/hello_constructs_stack.py`에서 다음을 사용합니다.

```
from aws_cdk import (
    aws_lambda as _lambda,
    aws_apigateway as apigw,
    core,
)

from aws_solutions_constructs import (
    aws_apigateway_lambda as apigw_lambda
)

class HelloConstructsStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hello.handler',
            ),
            api_gateway_props=apigw.RestApiProps(
                default_method_options=apigw.MethodOptions(
                    authorization_type=apigw.AuthorizationType.NONE
                )
            )
        )
```

그게 다야 AWS Lambda 함수에 대한 모든 요청을 프록시하는 API Gateway 정의하기 위해 필요한 모든 작업입니다. 새로운 스택을 원래 스택과 비교해 보겠습니다.

TypeScript

```
npm run build
cdk diff
```

Python

```
cdk diff
```

출력은 다음과 같아야 합니다.

```
Stack HelloConstructsStack
IAM Statement Changes
#######################################################################################
#   # Resource                    # Effect # Action                      # Principal
                # Condition                   #
#######################################################################################
# + # ${LambdaFunction.Arn}       # Allow  # lambda:InvokeFunction       #
 Service:apigateway.amazonaw # "ArnLike": {                 #
#   #                             #        #                             # s.com
                #   "AWS:SourceArn": "arn:${AW #
#   #                             #        #                             #
                # S::Partition}:execute-api:${ #
#   #                             #        #                             #
                # AWS::Region}:${AWS::AccountI #
#   #                             #        #                             #
                # d}:${RestApi0C43BF4B}/${Rest #
#   #                             #        #                             #
                # Api/DeploymentStage.prod}/*/ #
#   #                             #        #                             #
                # {proxy+}"                   #
#   #                             #        #                             #
                # }                           #
# + # ${LambdaFunction.Arn}       # Allow  # lambda:InvokeFunction       #
 Service:apigateway.amazonaw # "ArnLike": {                 #
#   #                             #        #                             # s.com
                #   "AWS:SourceArn": "arn:${AW #
#   #                             #        #                             #
                # S::Partition}:execute-api:${ #
#   #                             #        #                             #
                # AWS::Region}:${AWS::AccountI #
#   #                             #        #                             #
                # d}:${RestApi0C43BF4B}/test-i #
```

```
#   #                               #        #                              #
                # nvoke-stage/*/{proxy+}"      #
#   #                               #        #                              #
                # }                            #
# + # ${LambdaFunction.Arn}       # Allow  # lambda:InvokeFunction       #
 Service:apigateway.amazonaw # "ArnLike": {                 #
#   #                               #        #                              # s.com
                #   "AWS:SourceArn": "arn:${AW #
#   #                               #        #                              #
                # S::Partition}:execute-api:${ #
#   #                               #        #                              #
                # AWS::Region}:${AWS::AccountI #
#   #                               #        #                              #
                # d}:${RestApi0C43BF4B}/${Rest #
#   #                               #        #                              #
                # Api/DeploymentStage.prod}/*/ #
#   #                               #        #                              #
                # "                            #
#   #                               #        #                              #
                # }                            #
# + # ${LambdaFunction.Arn}       # Allow  # lambda:InvokeFunction       #
 Service:apigateway.amazonaw # "ArnLike": {                 #
#   #                               #        #                              # s.com
                #   "AWS:SourceArn": "arn:${AW #
#   #                               #        #                              #
                # S::Partition}:execute-api:${ #
#   #                               #        #                              #
                # AWS::Region}:${AWS::AccountI #
#   #                               #        #                              #
                # d}:${RestApi0C43BF4B}/test-i #
#   #                               #        #                              #
                # nvoke-stage/*/"              #
#   #                               #        #                              #
                # }                            #
##################################################################################
# + # ${LambdaFunctionServiceRole # Allow  # sts:AssumeRole              #
 Service:lambda.amazonaws.co #                               #
#   # .Arn}                         #        #                              # m
                #                              #
##################################################################################
# + # ${LambdaRestApiCloudWatchRo # Allow  # sts:AssumeRole              #
 Service:apigateway.amazonaw #                               #
#   # le.Arn}                       #        #                              # s.com
                #                              #
```

```
##########################################################################################
# + # arn:aws:logs:${AWS::Region} # Allow  # logs:CreateLogGroup         # AWS:
${LambdaRestApiCloudWat #                                #
#   # :${AWS::AccountId}:*        #        # logs:CreateLogStream        # chRole}
                #                                #
#   #                             #        # logs:DescribeLogGroups      #
                #                                #
#   #                             #        # logs:DescribeLogStreams     #
                #                                #
#   #                             #        # logs:FilterLogEvents        #
                #                                #
#   #                             #        # logs:GetLogEvents           #
                #                                #
#   #                             #        # logs:PutLogEvents           #
                #                                #
##########################################################################################
# + # arn:aws:logs:${AWS::Region} # Allow  # logs:CreateLogGroup         # AWS:
${LambdaFunctionService #                                #
#   # :${AWS::AccountId}:log-grou #        # logs:CreateLogStream        # Role}
                #                                #
#   # p:/aws/lambda/*             #        # logs:PutLogEvents           #
                #                                #
##########################################################################################
(NOTE: There may be security-related changes not in this list. See https://github.com/
aws/aws-cdk/issues/1299)

Parameters
[+] Parameter AssetParameters/
ba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340a/S3Bucket
 AssetParametersba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340aS3Bucket9780A3B
 {"Type":"String","Description":"S3 bucket for asset
 \"ba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340a\""}
[+] Parameter AssetParameters/
ba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340a/S3VersionKey
 AssetParametersba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340aS3VersionKey37F
 {"Type":"String","Description":"S3 key for asset version
 \"ba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340a\""}
[+] Parameter AssetParameters/
ba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340a/ArtifactHash
 AssetParametersba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340aArtifactHash801
 {"Type":"String","Description":"Artifact hash for asset
 \"ba91444ebd644d9419e8cfee417f3aaa728507dd428788a2fc40574646c4340a\""}

Conditions
```

```
[+] Condition CDKMetadataAvailable: {"Fn::Or":[{"Fn::Or":[{"Fn::Equals":
[{"Ref":"AWS::Region"},"ap-east-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"ap-
northeast-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"ap-northeast-2"]},{"Fn::Equals":
[{"Ref":"AWS::Region"},"ap-south-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"ap-
southeast-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"ap-southeast-2"]},{"Fn::Equals":
[{"Ref":"AWS::Region"},"ca-central-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"cn-
north-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"cn-northwest-1"]},
{"Fn::Equals":[{"Ref":"AWS::Region"},"eu-central-1"]}]},{"Fn::Or":[{"Fn::Equals":
[{"Ref":"AWS::Region"},"eu-north-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"eu-
west-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"eu-west-2"]},{"Fn::Equals":
[{"Ref":"AWS::Region"},"eu-west-3"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"me-
south-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"sa-east-1"]},{"Fn::Equals":
[{"Ref":"AWS::Region"},"us-east-1"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"us-
east-2"]},{"Fn::Equals":[{"Ref":"AWS::Region"},"us-west-1"]},{"Fn::Equals":
[{"Ref":"AWS::Region"},"us-west-2"]}]}]}

Resources
[+] AWS::Logs::LogGroup ApiGatewayToLambda/ApiAccessLogGroup
 ApiGatewayToLambdaApiAccessLogGroupE2B41502
[+] AWS::IAM::Role LambdaFunctionServiceRole LambdaFunctionServiceRole0C4CDE0B
[+] AWS::Lambda::Function LambdaFunction LambdaFunctionBF21E41F
[+] AWS::ApiGateway::RestApi RestApi RestApi0C43BF4B
[+] AWS::ApiGateway::Deployment RestApi/Deployment
 RestApiDeployment180EC503d2c6df3c8dc8b7193b98c1a0bff4e677
[+] AWS::ApiGateway::Stage RestApi/DeploymentStage.prod
 RestApiDeploymentStageprod3855DE66
[+] AWS::ApiGateway::Resource RestApi/Default/{proxy+} RestApiproxyC95856DD
[+] AWS::Lambda::Permission RestApi/Default/{proxy+}/ANY/
ApiPermission.HelloConstructsStackRestApiFDB18C2E.ANY..{proxy+}
 RestApiproxyANYApiPermissionHelloConstructsStackRestApiFDB18C2EANYproxyE43D39B3
[+] AWS::Lambda::Permission RestApi/Default/{proxy+}/ANY/
ApiPermission.Test.HelloConstructsStackRestApiFDB18C2E.ANY..{proxy+}
 RestApiproxyANYApiPermissionTestHelloConstructsStackRestApiFDB18C2EANYproxy0B23CDC7
[+] AWS::ApiGateway::Method RestApi/Default/{proxy+}/ANY RestApiproxyANY1786B242
[+] AWS::Lambda::Permission RestApi/Default/ANY/
ApiPermission.HelloConstructsStackRestApiFDB18C2E.ANY..
 RestApiANYApiPermissionHelloConstructsStackRestApiFDB18C2EANY5684C1E6
[+] AWS::Lambda::Permission RestApi/Default/ANY/
ApiPermission.Test.HelloConstructsStackRestApiFDB18C2E.ANY..
 RestApiANYApiPermissionTestHelloConstructsStackRestApiFDB18C2EANY81DBDF56
[+] AWS::ApiGateway::Method RestApi/Default/ANY RestApiANYA7C1DC94
[+] AWS::ApiGateway::UsagePlan RestApi/UsagePlan RestApiUsagePlan6E1C537A
[+] AWS::Logs::LogGroup ApiAccessLogGroup ApiAccessLogGroupCEA70788
[+] AWS::IAM::Role LambdaRestApiCloudWatchRole LambdaRestApiCloudWatchRoleF339D4E6
```

```
[+] AWS::ApiGateway::Account LambdaRestApiAccount LambdaRestApiAccount

Outputs
[+] Output RestApi/Endpoint RestApiEndpoint0551178A: {"Value":{"Fn::Join":["",
["https://",{"Ref":"RestApi0C43BF4B"},".execute-api.",{"Ref":"AWS::Region"},".",
{"Ref":"AWS::URLSuffix"},"/",{"Ref":"RestApiDeploymentStageprod3855DE66"},"/"]]}}
```

그건 좋은 일이야 AWS 솔루션 구조에서 잘 설계된 패턴 하나를 사용한 이 간단한 예제는 스택에 21개의 새로운 리소스를 추가했습니다.

cdk 배포

> **Tip**
>
> Lambda 함수가 포함된 첫 번째 AWS CDK 앱을 배포하려면 먼저 AWS 환경을 부트스트랩해야 합니다. 이렇게 하면 AWS CDK가 자산을 포함하는 스택을 배포하는 데 사용하는 스테이징 버킷이 생성됩니다. AWS CDK를 사용하여 자산을 배포하는 것이 처음인 경우`cdk bootstrap`를 사용하여 CDK 툴킷 스택을 AWS 환경에 배포할 수 있습니다.

좋아, 배포할 준비가 되셨나요?

```
cdk deploy
```

스택 출력

배포가 완료되면 다음 줄을 확인할 수 있습니다.

```
Outputs:
HelloConstructsStack.RestApiEndpoint0551178A = https://xxxxxxxxxx.execute-api.us-
east-1.amazonaws.com/prod/
```

이는 AWS 솔루션 구성 패턴에 의해 자동으로 추가되는 스택 출력이며 API Gateway 엔드포인트의 URL을 포함합니다.

앱 테스트

이 끝점을 사용하려고합시다.`curl`. URL을 복사하고 실행하십시오 (접두사와 지역이 다를 수 있음).

```
curl https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/prod/
```

출력은 다음과 같아야 합니다.

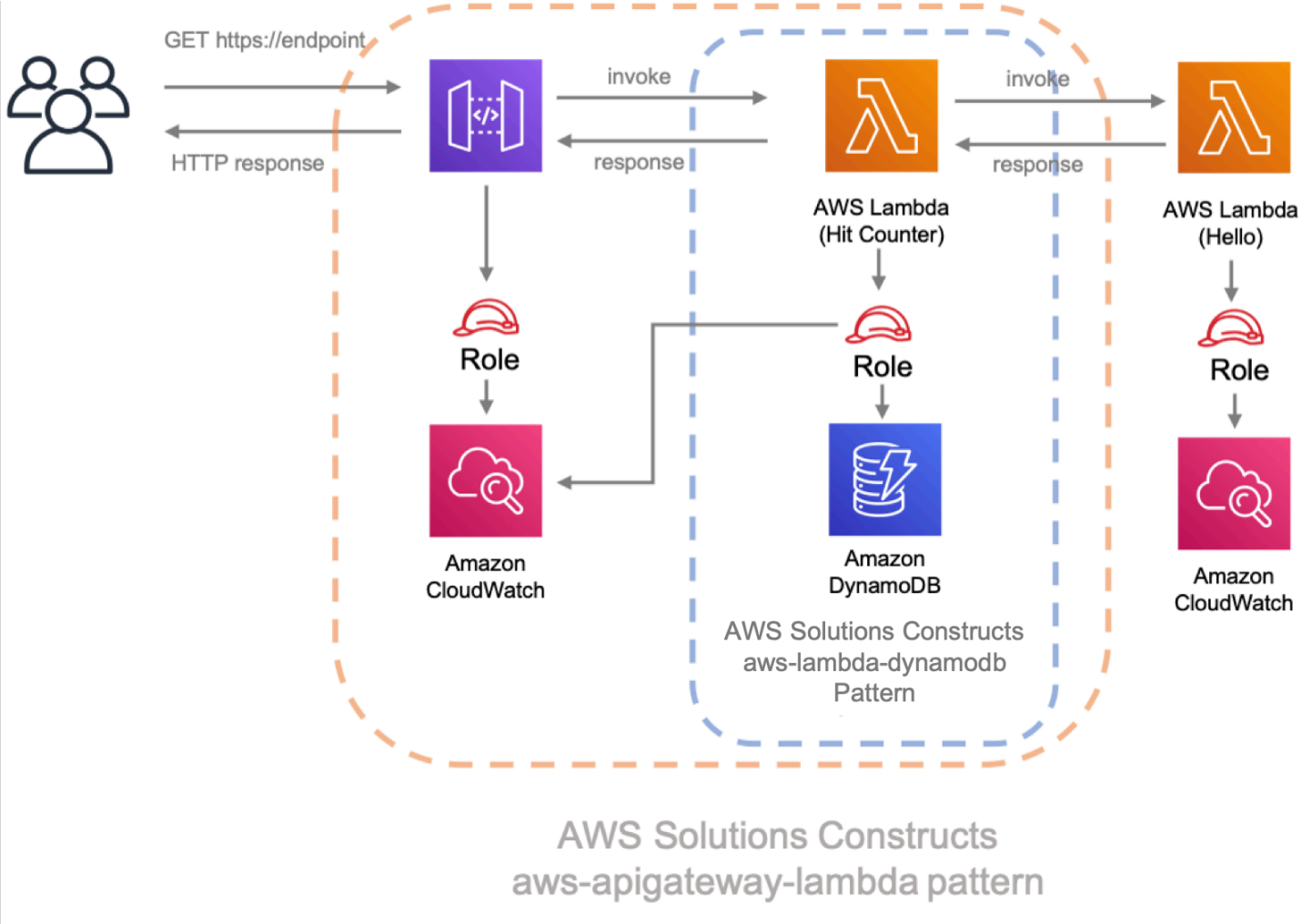
```
Hello, AWS Solutions Constructs! You've hit /
```

이것이 당신이받은 출력 인 경우, 앱이 작동합니다!

이동장면 - 부품 2

> **Note**
>
> AWS 솔루션 구성은 AWS CDK 버전

이 자습서에서는 에서 만든 “Hello Constructs” 앱을 수정하는 방법을 안내합니다.파트. 수정하면 AWS Lambda를 사용하여 AWS 솔루션 구문에서 DynamoDB 패턴에 사이트 히트 카운터가 추가됩니다. Hello Constructs 앱을 수정하면 다음과 같은 해결책이 생성됩니다.



Lambda 코드 히트

먼저 히트 카운터 AWS Lambda 함수에 대한 코드를 작성해 보겠습니다. 이 함수는 다음을 수행합니다.

- 는 Amazon DynamoDB 테이블의 API 경로와 관련된 카운터를 증가시키고
- 다운스트림 Hello AWS Lambda 함수를 호출합니다.
- 를 호출하고 최종 사용자에게 응답을 반환합니다.

TypeScript

이라는 파일을 추가합니다.`lambda/hitcounter.js`다음의 콘텐츠가 포함된 것들과 변경됩니다.

```
const { DynamoDB, Lambda } = require('aws-sdk');
```

```
exports.handler = async function(event) {
  console.log("request:", JSON.stringify(event, undefined, 2));

  // create AWS SDK clients
  const dynamo = new DynamoDB();
  const lambda = new Lambda();

  // update dynamo entry for "path" with hits++
  await dynamo.updateItem({
    TableName: process.env.DDB_TABLE_NAME,
    Key: { path: { S: event.path } },
    UpdateExpression: 'ADD hits :incr',
    ExpressionAttributeValues: { ':incr': { N: '1' } }
  }).promise();

  // call downstream function and capture response
  const resp = await lambda.invoke({
    FunctionName: process.env.DOWNSTREAM_FUNCTION_NAME,
    Payload: JSON.stringify(event)
  }).promise();

  console.log('downstream response:', JSON.stringify(resp, undefined, 2));

  // return response back to upstream caller
  return JSON.parse(resp.Payload);
};
```

Python

이라는 파일을 추가합니다.`lambda/hitcounter.py`다음의 콘텐츠가 포함된 것들과 변경됩니다.

```
import json
import os
import boto3

ddb = boto3.resource('dynamodb')
table = ddb.Table(os.environ['DDB_TABLE_NAME'])
_lambda = boto3.client('lambda')


def handler(event, context):
```

```
    print('request: {}'.format(json.dumps(event)))
    table.update_item(
        Key={'path': event['path']},
        UpdateExpression='ADD hits :incr',
        ExpressionAttributeValues={':incr': 1}
    )

    resp = _lambda.invoke(
        FunctionName=os.environ['DOWNSTREAM_FUNCTION_NAME'],
        Payload=json.dumps(event),
    )

    body = resp['Payload'].read()

    print('downstream response: {}'.format(body))
    return json.loads(body)
```

새 종속성 설치

> **Note**
>
> AWS 솔루션 구문과 AWS CDK 모두에 사용할 올바른 일치 버전을`VERSION_NUMBER`각 명령에 대한 자리 표시자 필드 이 연습의 첫 번째 부분에서 종속성에 사용된 버전 번호와 동일해야 합니다. 패키지 간에 버전이 일치하지 않으면 오류가 발생할 수 있습니다.

평소와 같이 솔루션 업데이트에 필요한 종속성을 먼저 설치해야합니다. 먼저 DynamoDB 구성 라이브러리를 설치해야 합니다.

TypeScript

```
npm install -s @aws-cdk/aws-dynamodb@VERSION_NUMBER
```

Python

```
pip install aws_cdk.aws_dynamodb==VERSION_NUMBER
```

마지막으로, AWS 솔루션 구문을 설치합니다.`aws-lambda-dynamodb`모듈과 모든 종속성을 프로젝트에 추가합니다.

TypeScript

```
npm install -s @aws-solutions-constructs/aws-lambda-dynamodb@VERSION_NUMBER
```

Python

```
pip install aws_solutions_constructs.aws_lambda_dynamodb==VERSION_NUMBER
```

리소스를 정의합니다.

이제 새로운 아키텍처를 수용하기 위해 스택 코드를 업데이트해 보겠습니다.

첫째, 우리는 우리의 새로운 종속성을 가져 와서 외부의 “안녕하세요”기능을 이동하려고합니다`aws-apigateway-lambda`우리가 파트 1에서 만든 패턴.

TypeScript

파일을 편집합니다.`lib/hello-constructs.ts`다음의 것들과 변경됩니다.

```
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import * as dynamodb from '@aws-cdk/aws-dynamodb';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-
constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/
aws-lambda-dynamodb';

export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
```

```
    super(scope, id, props);

    // The code that defines your stack goes here

    const helloFunc = new lambda.Function(this, 'HelloHandler', {
      runtime: lambda.Runtime.NODEJS_12_X,
      code: lambda.Code.fromAsset('lambda'),
      handler: 'hello.handler'
    });

    const api_lambda_props: ApiGatewayToLambdaProps = {
      lambdaFunctionProps: {
        code: lambda.Code.fromAsset('lambda'),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'hello.handler'
      },
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };

    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}
```

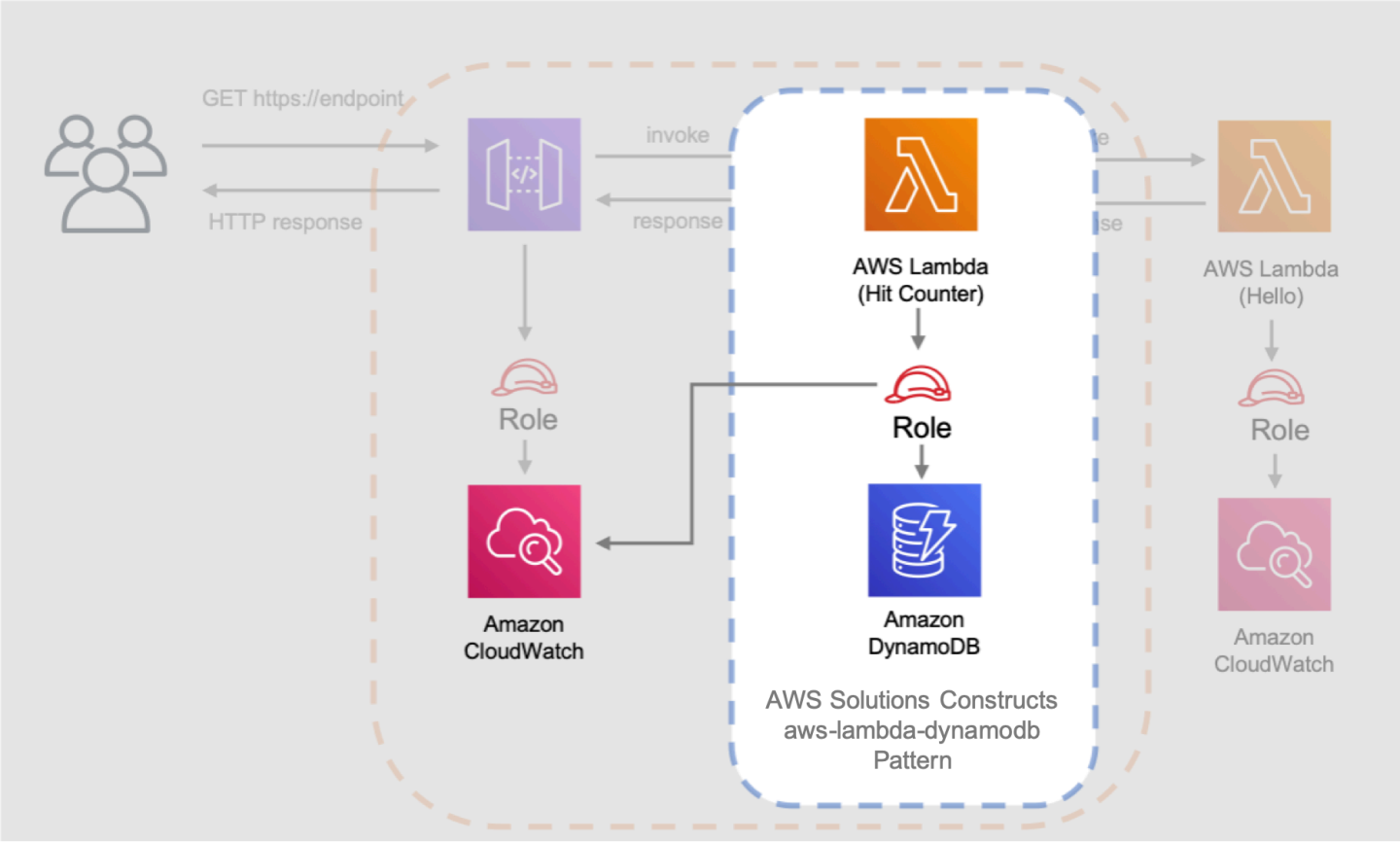
Python

파일을 편집합니다.`hello_constructs/hello_constructs_stack.py`다음의 것들과 변경됩니다.

```
from aws_cdk import (
    aws_lambda as _lambda,
    aws_apigateway as apigw,
    aws_dynamodb as ddb,
    core,
)

from aws_solutions_constructs import (
    aws_apigateway_lambda as apigw_lambda,
```

```
    aws_lambda_dynamodb as lambda_ddb
)

class HelloConstructsStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

        self._handler = _lambda.Function(
          self, 'HelloHandler',
          runtime=_lambda.Runtime.PYTHON_3_7,
          handler='hello.handler',
          code=_lambda.Code.asset('lambda'),
        )

        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hello.handler',
            ),
            api_gateway_props=apigw.RestApiProps(
                default_method_options=apigw.MethodOptions(
                    authorization_type=apigw.AuthorizationType.NONE
                )
            )
        )
```

다음으로, 우리는 추가 할 것입니다`aws-lambda-dynamodb`패턴을 사용하여 업데이트 된 아키텍처에 대한 히트 카운터 서비스를 구축 할 수 있습니다.



AWS Solutions Constructs
aws-apigateway-lambda pattern

다음 업데이트는 아래의 속성을 정의`aws-lambda-dynamodb`패턴을 사용하여 히트 카운터 처리기를 사용하여 AWS Lambda 함수를 정의합니다. 또한 Amazon DynamoDB 테이블은`Hits`의 파티션 키`path`.

TypeScript

파일을 편집합니다.`lib/hello-constructs.ts`다음의 것들과 변경됩니다.

```
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import * as dynamodb from '@aws-cdk/aws-dynamodb';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-
constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/
aws-lambda-dynamodb';
```

```
export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here

    const helloFunc = new lambda.Function(this, 'HelloHandler', {
      runtime: lambda.Runtime.NODEJS_12_X,
      code: lambda.Code.fromAsset('lambda'),
      handler: 'hello.handler'
    });

    // hit counter, aws-lambda-dynamodb pattern
    const lambda_ddb_props: LambdaToDynamoDBProps = {
      lambdaFunctionProps: {
          code: lambda.Code.asset(`lambda`),
          runtime: lambda.Runtime.NODEJS_12_X,
          handler: 'hitcounter.handler',
          environment: {
              DOWNSTREAM_FUNCTION_NAME: helloFunc.functionName
          }
      },
      dynamoTableProps: {
          tableName: 'Hits',
          partitionKey: { name: 'path', type: dynamodb.AttributeType.STRING }
      }
    };

    const hitcounter = new LambdaToDynamoDB(this, 'LambdaToDynamoDB',
 lambda_ddb_props);

    const api_lambda_props: ApiGatewayToLambdaProps = {
      lambdaFunctionProps: {
        code: lambda.Code.fromAsset('lambda'),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'hello.handler'
      },
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };
```

```
    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}
```

Python

파일을 편집합니다.`hello_constructs/hello_constructs_stack.py`다음의 것들과 변경됩니다.

```
from aws_cdk import (
    aws_lambda as _lambda,
    aws_apigateway as apigw,
    aws_dynamodb as ddb,
    core,
)

from aws_solutions_constructs import (
    aws_apigateway_lambda as apigw_lambda,
    aws_lambda_dynamodb as lambda_ddb
)

class HelloConstructsStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

        self.hello_func = _lambda.Function(
          self, 'HelloHandler',
          runtime=_lambda.Runtime.PYTHON_3_7,
          handler='hello.handler',
          code=_lambda.Code.asset('lambda'),
        )

        #  hit counter, aws-lambda-dynamodb pattern
        self.hit_counter = lambda_ddb.LambdaToDynamoDB(
            self, 'LambdaToDynamoDB',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
```

```
                code=_lambda.Code.asset('lambda'),
                handler='hitcounter.handler',
                environment={
                    'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name
                }
            ),
            dynamo_table_props=ddb.TableProps(
                table_name='Hits',
                partition_key={
                    'name': 'path',
                    'type': ddb.AttributeType.STRING
                }
            )
        )

        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hello.handler',
            ),
            api_gateway_props=apigw.RestApiProps(
                default_method_options=apigw.MethodOptions(
                    authorization_type=apigw.AuthorizationType.NONE
                )
            )
        )
```

다음으로, 우리는에서 생성 된 히트 카운터 함수를 부여해야합니다`aws-lambda-dynamodb`패턴은 우리의 Hello 함수를 호출 할 수있는 권한 위에 추가되었습니다.

TypeScript

파일을 편집합니다.`lib/hello-constructs.ts`다음의 것들과 변경됩니다.

```
  import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import * as dynamodb from '@aws-cdk/aws-dynamodb';
```

```
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-
constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/
aws-lambda-dynamodb';

export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here

    // hello function responding to http requests
    const helloFunc = new lambda.Function(this, 'HelloHandler', {
      runtime: lambda.Runtime.NODEJS_12_X,
      code: lambda.Code.fromAsset('lambda'),
      handler: 'hello.handler'
    });

    // hit counter, aws-lambda-dynamodb pattern
    const lambda_ddb_props: LambdaToDynamoDBProps = {
      lambdaFunctionProps: {
          code: lambda.Code.asset(`lambda`),
          runtime: lambda.Runtime.NODEJS_12_X,
          handler: 'hitcounter.handler',
          environment: {
              DOWNSTREAM_FUNCTION_NAME: helloFunc.functionName
          }
      },
      dynamoTableProps: {
          tableName: 'Hits',
          partitionKey: { name: 'path', type: dynamodb.AttributeType.STRING }
      }
    };

    const hitcounter = new LambdaToDynamoDB(this, 'LambdaToDynamoDB',
 lambda_ddb_props);

    // grant the hitcounter lambda role invoke permissions to the hello function
    helloFunc.grantInvoke(hitcounter.lambdaFunction);

    const api_lambda_props: ApiGatewayToLambdaProps = {
      lambdaFunctionProps: {
        code: lambda.Code.fromAsset('lambda'),
        runtime: lambda.Runtime.NODEJS_12_X,
```

```
        handler: 'hello.handler'
      },
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };

    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}
```

Python

파일을 편집합니다.`hello_constructs/hello_constructs_stack.py`다음의 것들과 변경됩니다.

```
from aws_cdk import (
    aws_lambda as _lambda,
    aws_apigateway as apigw,
    aws_dynamodb as ddb,
    core,
)

from aws_solutions_constructs import (
    aws_apigateway_lambda as apigw_lambda,
    aws_lambda_dynamodb as lambda_ddb
)

class HelloConstructsStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

        self.hello_func = _lambda.Function(
          self, 'HelloHandler',
          runtime=_lambda.Runtime.PYTHON_3_7,
          handler='hello.handler',
```

```
            code=_lambda.Code.asset('lambda'),
        )

        #  hit counter, aws-lambda-dynamodb pattern
        self.hit_counter = lambda_ddb.LambdaToDynamoDB(
            self, 'LambdaToDynamoDB',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hitcounter.handler',
                environment={
                    'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name
                }
            ),
            dynamo_table_props=ddb.TableProps(
                table_name='Hits',
                partition_key={
                    'name': 'path',
                    'type': ddb.AttributeType.STRING
                }
            )
        )

        # grant the hitcounter lambda role invoke permissions to the hello function
        self.hello_func.grant_invoke(self.hit_counter.lambda_function)

        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hello.handler',
            ),
            api_gateway_props=apigw.RestApiProps(
                default_method_options=apigw.MethodOptions(
                    authorization_type=apigw.AuthorizationType.NONE
                )
            )
        )
```

마지막으로 원본을 업데이트해야합니다.`aws-apigateway-lambda`패턴을 사용하여 프로비저닝된 새로운 히트 카운터 함수를 활용할 수 있습니다.`aws-lambda-dynamodb`패턴을 사용합니다.

TypeScript

파일을 편집합니다.`lib/hello-constructs.ts`다음의 것들과 변경됩니다.

```
  import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import * as api from '@aws-cdk/aws-apigateway';
import * as dynamodb from '@aws-cdk/aws-dynamodb';
import { ApiGatewayToLambda, ApiGatewayToLambdaProps } from '@aws-solutions-
constructs/aws-apigateway-lambda';
import { LambdaToDynamoDB, LambdaToDynamoDBProps } from '@aws-solutions-constructs/
aws-lambda-dynamodb';

export class HelloConstructsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here

    // hello function responding to http requests
    const helloFunc = new lambda.Function(this, 'HelloHandler', {
      runtime: lambda.Runtime.NODEJS_12_X,
      code: lambda.Code.fromAsset('lambda'),
      handler: 'hello.handler'
    });

    // hit counter, aws-lambda-dynamodb pattern
    const lambda_ddb_props: LambdaToDynamoDBProps = {
      lambdaFunctionProps: {
          code: lambda.Code.asset(`lambda`),
          runtime: lambda.Runtime.NODEJS_12_X,
          handler: 'hitcounter.handler',
          environment: {
              DOWNSTREAM_FUNCTION_NAME: helloFunc.functionName
          }
      },
      dynamoTableProps: {
          tableName: 'Hits',
          partitionKey: { name: 'path', type: dynamodb.AttributeType.STRING }
      }
```

```
    };

    const hitcounter = new LambdaToDynamoDB(this, 'LambdaToDynamoDB',
 lambda_ddb_props);

    // grant the hitcounter lambda role invoke permissions to the hello function
    helloFunc.grantInvoke(hitcounter.lambdaFunction);

    const api_lambda_props: ApiGatewayToLambdaProps = {
      existingLambdaObj: hitcounter.lambdaFunction,
      apiGatewayProps: {
        defaultMethodOptions: {
          authorizationType: api.AuthorizationType.NONE
        }
      }
    };

    new ApiGatewayToLambda(this, 'ApiGatewayToLambda', api_lambda_props);
  }
}
```

Python

파일을 편집합니다.`hello_constructs/hello_constructs_stack.py`다음의 것들과 변경됩니다.

```
from aws_cdk import (
    aws_lambda as _lambda,
    aws_apigateway as apigw,
    aws_dynamodb as ddb,
    core,
)

from aws_solutions_constructs import (
    aws_apigateway_lambda as apigw_lambda,
    aws_lambda_dynamodb as lambda_ddb
)

class HelloConstructsStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
```

```
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here

        self.hello_func = _lambda.Function(
          self, 'HelloHandler',
          runtime=_lambda.Runtime.PYTHON_3_7,
          handler='hello.handler',
          code=_lambda.Code.asset('lambda'),
        )

        #  hit counter, aws-lambda-dynamodb pattern
        self.hit_counter = lambda_ddb.LambdaToDynamoDB(
            self, 'LambdaToDynamoDB',
            lambda_function_props=_lambda.FunctionProps(
                runtime=_lambda.Runtime.PYTHON_3_7,
                code=_lambda.Code.asset('lambda'),
                handler='hitcounter.handler',
                environment={
                    'DOWNSTREAM_FUNCTION_NAME': self.hello_func.function_name
                }
            ),
            dynamo_table_props=ddb.TableProps(
                table_name='Hits',
                partition_key={
                    'name': 'path',
                    'type': ddb.AttributeType.STRING
                }
            )
        )

        # grant the hitcounter lambda role invoke permissions to the hello function
        self.hello_func.grant_invoke(self.hit_counter.lambda_function)

        apigw_lambda.ApiGatewayToLambda(
            self, 'ApiGatewayToLambda',
            existing_lambda_obj=self.hit_counter.lambda_function,
            api_gateway_props=apigw.RestApiProps(
                default_method_options=apigw.MethodOptions(
                    authorization_type=apigw.AuthorizationType.NONE
                )
            )
        )
```

변경 사항 검토

프로젝트를 빌드하고 배포 할 때 발생할 리소스의 변경 사항을 검토해 보겠습니다.

```
npm run build
cdk diff
```

출력은 다음과 같아야 합니다.

```
Stack HelloConstructsStack
IAM Statement Changes
###########################################################################################
#   # Resource                          # Effect # Action                            #
 Principal                          # Condition #
###########################################################################################
# + # ${HelloHandler.Arn}               # Allow  # lambda:InvokeFunction              #
 AWS:${LambdaFunctionServiceRole}   #           #
###########################################################################################
# + # ${HelloHandler/ServiceRole.Arn}   # Allow  # sts:AssumeRole                     #
 Service:lambda.amazonaws.com       #           #
###########################################################################################
# + # ${LambdaToDynamoDB/DynamoTable.Ar # Allow  # dynamodb:BatchGetItem              #
 AWS:${LambdaFunctionServiceRole}   #           #
#   # n}                                #        # dynamodb:BatchWriteItem           #
                                    #           #
#   #                                 #        # dynamodb:DeleteItem               #
                                    #           #
#   #                                 #        # dynamodb:GetItem                  #
                                    #           #
#   #                                 #        # dynamodb:GetRecords               #
                                    #           #
#   #                                 #        # dynamodb:GetShardIterator         #
                                    #           #
#   #                                 #        # dynamodb:PutItem                  #
                                    #           #
#   #                                 #        # dynamodb:Query                    #
                                    #           #
```

```
#   #                                   #        # dynamodb:Scan                   #
                                  #           #
#   #                                   #        # dynamodb:UpdateItem             #
                                  #           #
##########################################################################################
IAM Policy Changes
##########################################################################################
#   # Resource                     # Managed Policy ARN
                             #
##########################################################################################
# + # ${HelloHandler/ServiceRole} # arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole #
##########################################################################################
(NOTE: There may be security-related changes not in this list. See https://github.com/
aws/aws-cdk/issues/1299)

Resources
[+] AWS::IAM::Role HelloHandler/ServiceRole HelloHandlerServiceRole11EF7C63
[+] AWS::Lambda::Function HelloHandler HelloHandler2E4FBA4D
[+] AWS::DynamoDB::Table LambdaToDynamoDB/DynamoTable
 LambdaToDynamoDBDynamoTable53C1442D
[+] AWS::IAM::Policy LambdaFunctionServiceRole/DefaultPolicy
 LambdaFunctionServiceRoleDefaultPolicy126C8897
[~] AWS::Lambda::Function LambdaFunction LambdaFunctionBF21E41F
 ## [+] Environment
 #   ## {"Variables":{"DOWNSTREAM_FUNCTION_NAME":
{"Ref":"HelloHandler2E4FBA4D"},"DDB_TABLE_NAME":
{"Ref":"LambdaToDynamoDBDynamoTable53C1442D"}}}
 ## [~] Handler
 #   ## [-] hello.handler
 #   ## [+] hitcounter.handler
 ## [~] DependsOn
     ## @@ -1,3 +1,4 @@
        [ ] [
        [+]   "LambdaFunctionServiceRoleDefaultPolicy126C8897",
        [ ]   "LambdaFunctionServiceRole0C4CDE0B"
        [ ] ]
```

cdk 배포

좋아, 배포할 준비가 되셨나요?

```
cdk deploy
```

스택 출력

배포가 완료되면 다음 줄을 확인할 수 있습니다.

```
Outputs:
HelloConstructsStack.RestApiEndpoint0551178A = https://xxxxxxxxxx.execute-api.us-
east-1.amazonaws.com/prod/
```

앱 테스트

이 끝점을 컬로 치려고 노력합시다. URL을 복사하고 실행하십시오 (접두사와 지역이 다를 수 있음).

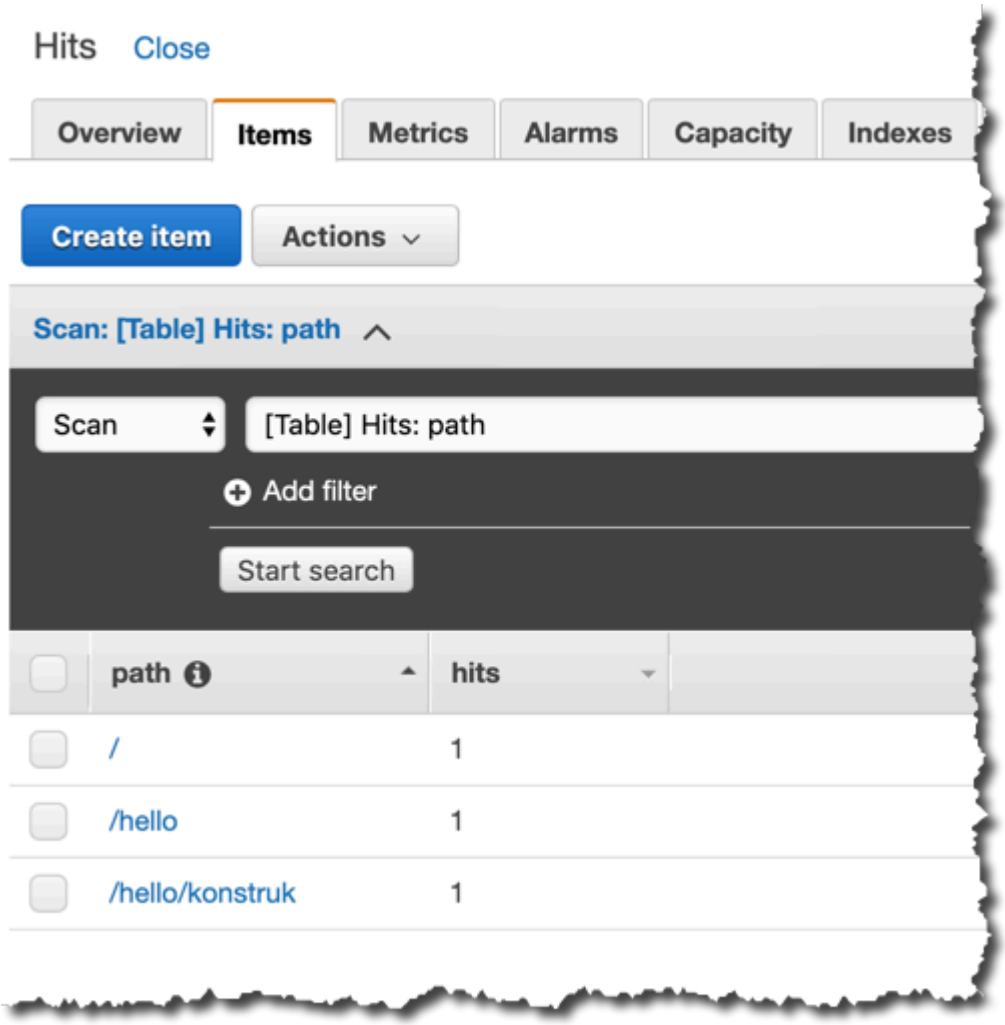
```
curl https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/prod/
```

출력은 다음과 같아야 합니다.

```
Hello, AWS Solutions Constructs! You've hit /
```

이제 검토해 보겠습니다.`Hits`Amazon DynamoDB DB 테이블

1. DynamoDB 콘솔로 이동합니다.
2. 테이블을 생성한 리전에 있는지 확인합니다.
3. Select테이블탐색 창에서Hits테이블을 생성합니다.
4. 테이블을 열고 “항목”을 선택하십시오.
5. 각 경로에 대해 얼마나 많은 히트 곡을 볼 수 있습니다.



6. 새 경로를 누르고 항목 보기를 새로 고치십시오. 새 항목이 표시되어야 합니다.`hits`하나의 카운트.

이것이 당신이받은 출력 인 경우, 앱이 작동합니다!

예제 사용 사례

이 라이브러리에는 구조 아키텍처 패턴의 사용을 보여주기 위해 기능적 유스 케이스 구현의 컬렉션이 포함되어 있습니다. 이들은 아키텍처 패턴과 같은 방식으로 사용할 수 있으며 이러한 패턴의 추가 "상위 수준"추상화로 개념화 될 수 있습니다. 다음과 같은 사용 사례가 기능적 예제로 제공됩니다.

AWS 정적 S3 웹 사이트

이 유스 케이스 패턴 (`aws-s3-static-website`) 는 Amazon CloudFront 배포, Amazon S3 버킷 및 AWS 람다 기반 사용자 지정 리소스를 구현하여 Wild Rydes 데모 웹 사이트의 정적 웹 사이트 콘텐츠를 복사합니다 (`aws-serverless-web-app`구현입니다).

소스 코드 (AWS-s3-정적 웹 사이트)

https://github.com/awslabs/aws-solutions-constructs/tree/master/source/use_cases/aws-s3-static-website

AWS 단순 서버를 사용하지 않는 이미지 처리기

이 유스 케이스 패턴 (`aws-serverless-image-handler`) 는 Amazon CloudFront 배포, Amazon API Gateway REST API, AWS Lambda 함수 및 배포 계정 내의 하나 이상의 Amazon S3 버킷에서 이미지 콘텐츠를 제공하기 위한 기능 이미지 처리기 API를 프로비저닝하는 데 필요한 권한/논리를 구현합니다.

소스 코드 (AWS 서버리스 이미지 처리기)

https://github.com/awslabs/aws-solutions-constructs/tree/master/source/use_cases/aws-serverless-image-handler

AWS 서버리스 웹 앱

이 유스 케이스 패턴 (`aws-serverless-web-app`) 는 사용자가 Wild Rydes 함대에서 유니콘 라이드를 요청할 수 있는 간단한 서버를 사용하지 않는 웹 애플리케이션을 구현합니다. 응용 프로그램은 사용자에게 선택하고자하는 위치를 나타내는 HTML 기반 사용자 인터페이스를 제공하고 요청을 제출하고 근처의 유니콘을 디스패치하기 위해 RESTful 웹 서비스와 백엔드에서 인터페이스합니다. 또한 이 응용 프로그램은 사용자가 서비스에 등록하고 승차 요청 전에 로그인 할 수있는 기능을 제공합니다.

소스 코드 (aws-서버-웹 응용 프로그램)

https://github.com/awslabs/aws-solutions-constructs/tree/master/source/use_cases/aws-serverless-web-app

API 참조

AWS 솔루션 구조 (구조) 는 AWS CDK (Cloud Development Kit) 의 오픈 소스 확장으로, 예측 가능하고 반복 가능한 인프라를 생성하기 위해 코드에서 솔루션을 신속하게 정의하기 위한 다중 서비스, 잘 설계된 패턴을 제공합니다. Construct의 목표는 개발자가 아키텍처에 대한 패턴 기반 정의를 사용하여 모든 크기의 솔루션을 빌드할 수 있는 환경을 가속화하는 것입니다.

구성에 정의된 패턴은 잘 설계된 모범 사례를 기반으로 기본 구성이 있는 AWS CDK 구조의 상위 수준의 다중 서비스 추상화입니다. 라이브러리는 각 아키텍처 패턴 모델을 생성하는 객체 지향 기술을 사용하여 논리 모듈로 구성된다.

CDK는 다음 언어로 제공됩니다.

- JavaScript, TypeScript (Node.js ≥ 10.3.0)
- 파이썬 (파이썬 ≥ 3.6)
- 자바 (자바 1.8 이상)

Modules

AWS 솔루션 구문은 여러 모듈로 구성되어 있습니다. 그들은 다음과 같이 명명됩니다:

- AWS-xxx: 표시된 서비스에 대한 잘 설계된 패턴 패키지. 이 패키지에는 지정된 패턴을 구성하는 여러 AWS CDK 서비스 모듈을 포함하는 구문이 포함됩니다.
- xxx: 시작되지 않는 패키지”aws-“는 패턴 라이브러리 내에서 사용되는 서비스의 모범 사례 기본값을 구성하는 데 사용되는 핵심 모듈을 구성합니다.

모듈 컨텐츠

모듈에는 다음 유형이 포함됩니다.

- 패턴-이 라이브러리의 모든 상위 수준의 다중 서비스 구조입니다.
- 기타 유형- 패턴을 지원하기 위해 존재하는 모든 비 구성 클래스, 인터페이스, 구조체 및 열거 형.

패턴은 생성자에서 (입력) 속성 집합을 사용합니다. 속성 집합 (및 필요한 속성 집합) 은 패턴의 문서 페이지에서 볼 수 있습니다.

패턴의 문서 페이지는 또한 호출 할 수있는 메소드와 인스턴스화 된 후 패턴에 대한 정보를 검색하는 데 사용할 수있는 속성을 나열합니다.

AWS-아피가티웨이-다이나모DB

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 참조하십시오. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_apigateway_dynamodb`
타이프 스크립트	`@aws-solutions-constructs/aws-apigateway-dynamodb`
Java	`software.amazon.awsconstructs.services.apigatewaydynamodb`

Overview

이 AWS 솔루션 구성은 Amazon DynamoDB 테이블에 연결된 Amazon API 게이트웨이 REST API를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { ApiGatewayToDynamoDBProps, ApiGatewayToDynamoDB } from "@aws-solutions-constructs/aws-apigateway-dynamodb";
```

```
new ApiGatewayToDynamoDB(this, 'test-api-gateway-dynamodb-default', {});
```

Initializer

```
new ApiGatewayToDynamoDB(scope: Construct, id: string, props:
 ApiGatewayToDynamoDBProps);
```

파라미터

- scopeConstruct
- idstring
- propsApiGatewayToDynamoDBProps

패턴 구성

이름	유형	설명
다이나모터프로프	dynamodb.TableProps	DynamoDB 테이블의 기본 소품을 재정의할 수 있는 선택적 사용자가 제공한 소품
어피게이트웨이 소품?	api.RestApiProps	API Gateway 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
허용만들기 작업	boolean	DynamoDB 테이블에 생성 작업을 위한 API Gateway 메소드를 배포할지 여부입니다.
요청 템플릿 작성	string	만들기 메서드에 대한 API Gateway 요청 템플릿, 허용만들기작업이 true로 설정된 경우 필요합니다.

이름	유형	설명
허용읽기 작업	`boolean`	DynamoDB 테이블에 읽기 작업을 위한 API Gateway 메소드를 배포할지 여부입니다.
허용업데이트 작업	`boolean`	DynamoDB 테이블에 업데이트 작업을 위한 API Gateway 메서드를 배포할지 여부입니다.
업데이트 요청 템플릿	`string`	업데이트 메서드에 대한 API Gateway 요청 템플릿. 허용업데이트 작업이 true로 설정된 경우 필요합니다.
허용삭제 작업	`boolean`	DynamoDB 테이블에서 삭제 작업을 위한 API Gateway 메서드를 배포할지 여부입니다.
로그그룹Props?	`logs.LogGroupProps`	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품입니다.

패턴 속성

이름	유형	설명
ApiGateway	`api.RestApi`	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
아피게이트웨이클라우드워치역할	`iam.Role`	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.

이름	유형	설명
어피게이트웨이로그 그룹	logs.LogGroup	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.
에이피게이트웨이의 역할	iam.Role	API Gateway REST API의 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
다이나모터블	dynamodb.Table	패턴에 의해 생성된 DynamoDB 테이블의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

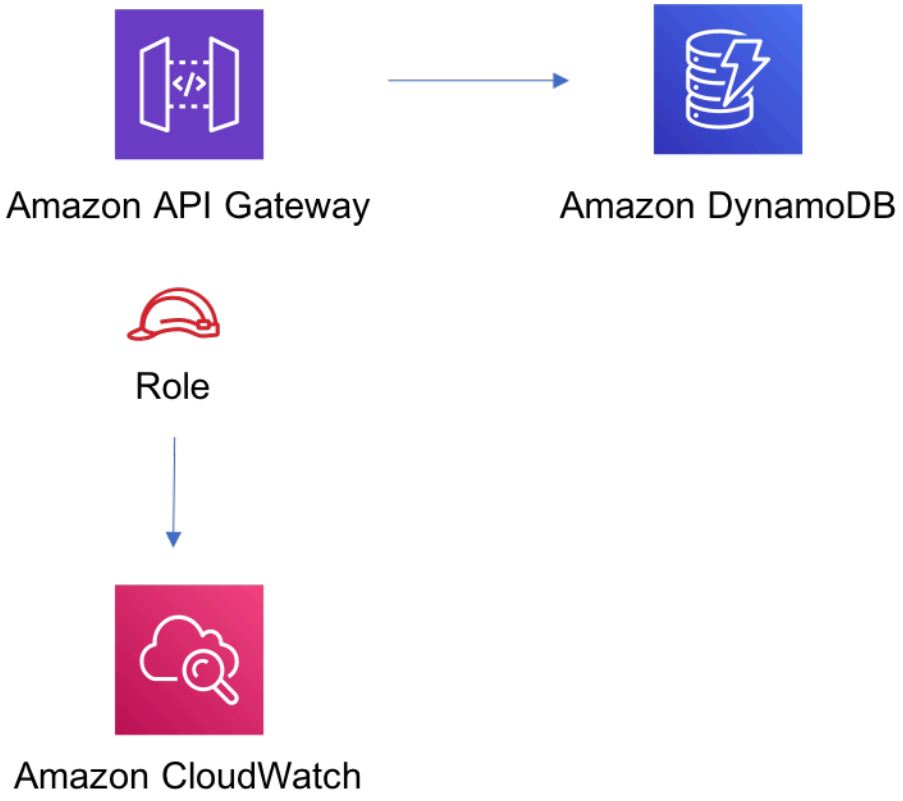
Amazon API Gateway

- 엣지 최적화 API 엔드포인트 배포
- API Gateway 에 대한 CloudWatch 로깅
- API Gateway 대한 최소 권한 액세스 IAM 역할 구성
- 모든 API 메소드에 대한 기본 권한 부여 유형을 IAM으로 설정
- X-Ray 추적 사용

Amazon DynamoDB 테이블

- DynamoDB 테이블의 결제 모드를 온 디맨드 (요청당 지불) 로 설정
- AWS 관리형 KMS 키를 사용하여 DynamoDB 테이블에 대한 서버 측 암호화 활성화
- DynamoDB 테이블에 대해 'id'라는 파티션 키를 생성합니다.
- CloudFormation 스택을 삭제할 때 테이블 유지
- 지속적인 백업 및 지정 시간 복구

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws-apigateway-Dynamodb

아피가테이웨이 IoT

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_apigateway_iot`
타이프 스크립트	`@aws-solutions-constructs/aws-apigateway-iot`
Java	`software.amazon.awsconstructs.services.apigatewayiot`

Overview

이 AWS 솔루션 구성은 AWS IoT 패턴에 연결된 Amazon API Gateway REST API를 구현합니다.

이 구조는 API Gateway AWS IoT 간에 확장 가능한 HTTPS 프록시를 생성합니다. 이는 MQTT 또는 MQTT/WebSocket 프로토콜을 지원하지 않는 레거시 디바이스가 AWS IoT 플랫폼과 상호 작용하도록 허용하려는 경우에 유용합니다.

이 구현은 지정된 MQTT 항목에 쓰기 전용 메시지를 게시할 수 있게 하며, 장치 레지스트리에서 허용된 항목에 대한 HTTPS 장치의 섀도 업데이트를 지원합니다. 메시지를 프록싱하기 위해 Lambda 함수를 포함하지 않으며 대신 JSON 메시지와 이진 메시지를 모두 지원하는 AWS IoT 통합 직접 API Gateway 사용합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { ApiGatewayToIot } from '@aws-solutions-constructs/aws-apigateway-iot';

new ApiGatewayToIot(this, 'ApiGatewayToIotPattern', {
    iotEndpoint: 'a1234567890123-ats'
});
```

Initializer

```
new ApiGatewayToIot(scope: Construct, id: string, props: ApiGatewayToIotProps);
```

파라미터

- scopeConstruct
- idstring
- propsApiGatewayToIotProps

소품 패턴 구성

이름	유형	설명
이오텐드포인트	string	API Gateway 통합하기 위한 AWS IoT 엔드포인트 하위 도메인 (예: a1234567890123-ats).
아피가트웨이만들기API키?	boolean	다음의 경우true로 설정하면 API 키가 생성되어 UsagePlan 연결됩니다. RESTAPI에 액세스하는 동안 사용자는`X-API 키'헤더를 지정해야합니다. 기본값 설정false.
어피그레이트웨이 실행역할?	iam.Role	API Gateway 에서 AWS IoT 에 액세스하는 데 사용하는 IAM 역할입니다. 지정하지 않으면 모든 주제와 사물에 대한 와일드카드 ('*') 액세스 권한이 있는 기본 역할이 만들어집니다.
아피가트웨이 소품?	api.restApiProps	API Gateway REST API의 기본 소품을 재정의하는 선택적 사용자 제공 소품.

이름	유형	설명
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품.

패턴 등록 정보

이름	유형	설명
에이피가트웨이	api.RestApi	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
APIGateway클라우드워치역할	iam.Role	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
에이피가트웨이로그 그룹	logs.LogGroup	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.
에이피게이트 웨이 역할	iam.Role	API Gateway REST API의 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon API Gateway

- 엣지 최적화 API 엔드포인트 배포
- 다음을 사용하여 API 리소스를 만듭니다.POSTIoT 주제에 메시지를 게시하는 방법

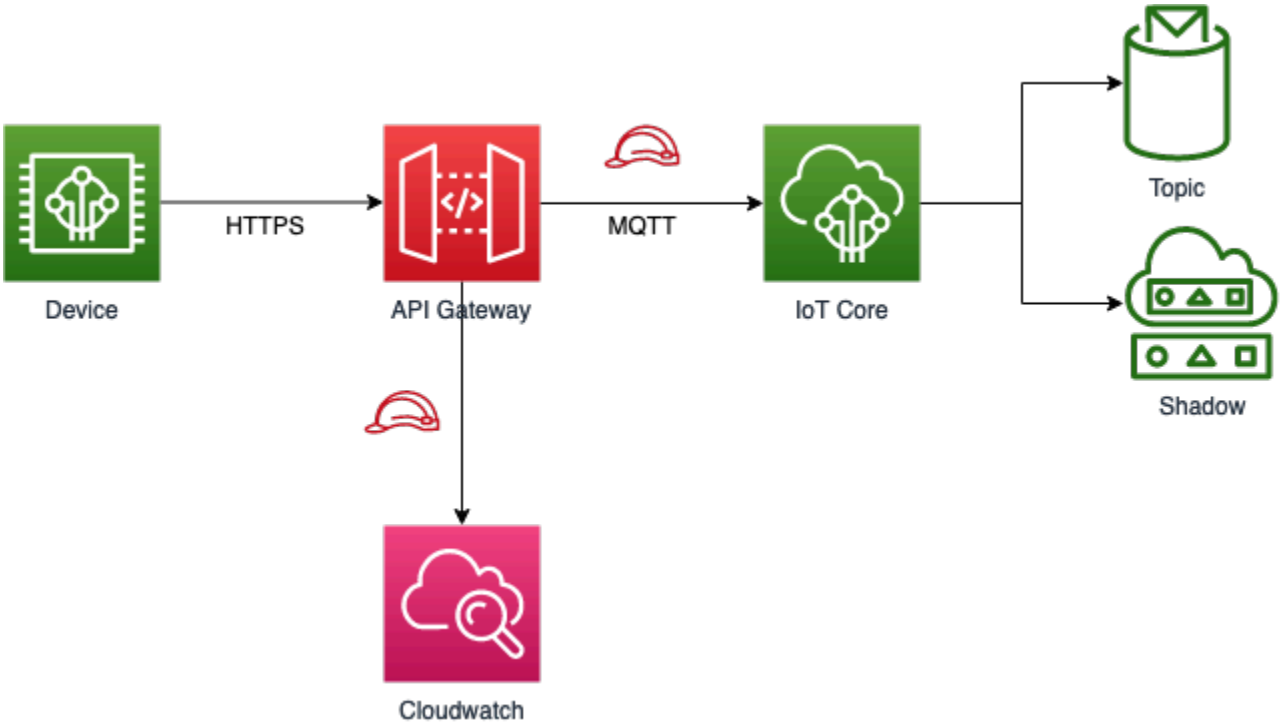
- 다음을 사용하여 API 리소스를 만듭니다.POST메시지를 게시하는 방법ThingShadow및NamedShadows
- API Gateway 대한 CloudWatch 로깅 활성화
- 모든 주제 및 사물에 액세스할 수 있는 API Gateway 대한 IAM 역할 구성
- 모든 API 메소드의 기본 권한 부여 유형을 IAM으로 설정
- X-Ray 추적 설정
- UsagePlan 생성하고prodstage

다음은 구성을 배포한 후 API Gateway 에서 제공하는 다양한 리소스 및 메서드에 대한 설명입니다. 단원을 참조하십시오.예제:를 사용하여 이러한 엔드포인트를 쉽게 테스트하는 방법에 대한 자세한 정보는curl.

방법	리소스	쿼리 매개 변수	반환 코드	설명
POST	/message/<topics>	qos	200/403/500	이 엔드 포인트를 호출하면 게시 할 주제 (예:/message/device/foo `).
POST	/shadow/<thingName>	없음	200/403/500	이 경로는 사물의 그림자 문서를 업데이트 할 수 있습니다.thingName 명명되지 않은 (클래식) 섀도우 유형 사용 몸은 포함하는 표준 그림자 stucture 준수한다state노드 및 연관된desired및reported노드. 단원을 참조하십시오.섀도우 업데이트단원의

방법	리소스	쿼리 매개 변수	반환 코드	설명
				예제를 참조하십시오.
POST	/shadow/<thingName>/<shadowName>	없음	200/403/500	이 경로는 사물의 명명 된 그림자 문서를 업데이트 할 수 있습니다.thingName 및shadowName 명명된 그림자 유형을 사용합니다. 몸은 포함하는 표준 그림자 stucture 준수한다state노드 및 연관된desired및reported노드. 단원을 참조하십시오.명명된 섀도우 업데이트단원의 예제를 참조하십시오.

Architecture



Examples

다음 예제에서는`API_KEY`인증 유형을 사용할 수 있습니다. IAM 권한 부여에는 SIGv4 토큰도 지정해야 하기 때문에`apiGatewayCreateApiKey`속성이 구성 소품의`true`를 사용하여 스택을 배포하는 경우 그렇지 않으면 아래 예제가 작동하지 않습니다.

메시지 게시

다음을 수행할 수 있습니다.`curl`을 클릭하여 HTTPS API를 사용하여 다른 MQTT 주제에 메시지를 게시합니다. 아래의 예는에 메시지를 게시합니다`device/foo`주제를 참조하십시오.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/message/device/
foo -H "x-api-key: <api-key>" -H "Content-Type: application/json" -d '{"Hello":
 "World"}'
```

참고: 를 대체합니다.`stage-id`,`region`, 및`api-key`매개 변수를 배포 값으로 바꿉니다.

URL에서 주제 이름을 연결할 수 있으며 API는 게시할 수 있는 하위 주제를 최대 7개까지 허용합니다. 예를 들어, 아래의 예는 주제에 메시지를 게시합니다`device/foo/bar/abc/xyz`.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/message/device/
foo/bar/abc/xyz -H "x-api-key: <api-key>" -H "Content-Type: application/json" -d
 '{"Hello": "World"}'
```

섀도우 업데이트

주어진 것과 관련된 섀도우 문서를 업데이트하려면 사물 이름을 사용하여 섀도우 상태 요청을 실행할 수 있습니다. 사물 그림자를 업데이트하는 방법에 대한 다음 예제를 참조하십시오.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/shadow/device1 -
H "x-api-key: <api-key>" -H "Content-Type: application/json" -d '{"state": {"desired":
 { "Hello": "World" }}}'
```

명명된 섀도우 업데이트

주어진 사물의 이름이 지정된 섀도우와 연관된 섀도우 문서를 업데이트하려면 사물 이름과 섀도우 이름을 사용하여 섀도우 상태 요청을 실행할 수 있습니다. 명명된 섀도우를 업데이트하는 방법은 다음 예제를 참조하십시오.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/shadow/device1/
shadow1 -H "x-api-key: <api-key>" -H "Content-Type: application/json" -d '{"state":
 {"desired": { "Hello": "World" }}}'
```

이진 페이로드 전송

바이너리 페이로드를 프록시 API로 AWS IoT 서비스로 전송할 수 있습니다. 다음 예제에서는 의 내용을 보냅니다.`README.md`이 모듈과 관련된 파일 (이진 데이터로 처리됨) 을`device/foo`를 사용하여`application/octet-stream`콘텐츠 유형.

```
curl -XPOST https://<stage-id>.execute-api.<region>.amazonaws.com/prod/message/device/
foo/bar/baz/qux -H "x-api-key: <api-key>" -H "Content-Type: application/octet-stream"
 --data-binary @README.md
```

참고: 이 프로젝트의 디렉토리에있는 동안이 명령을 실행하십시오. 그런 다음 파일 시스템에서 다른 유형의 바이너리 파일 전송을 테스트 할 수 있습니다.

GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/aws-apigateway - IoT

aws-apigateway-키네시스스트림

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 사용해 보세요. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_apigateway_kinesisstreams`
타이프 스크립트	`@aws-solutions-constructs/aws-apigateway-kinesisstreams`
Java	`software.amazon.awsconstructs.services.apigatewaykinesisstreams`

Overview

이 패턴은 Amazon Kinesis 데이터 스트림에 연결된 Amazon API 게이트웨이 REST API를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { ApiGatewayToKinesisStreams, ApiGatewayToKinesisStreamsProps } from '@aws-solutions-constructs/aws-apigateway-kinesisstreams';

new ApiGatewayToKinesisStreams(this, 'test-apigw-kinesis', {});
```

Initializer

```
new ApiGatewayToKinesisStreams(scope: Construct, id: string, props:
 ApiGatewayToKinesisStreamsProps);
```

파라미터

- scope`Construct`
- id`string`
- props`ApiGatewayToKinesisStreamsProps`

패턴 구성

이름	유형	설명
아피가트웨이 소품?	`api.RestApiProps`	API Gateway REST API의 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
입력코드 요청 템플릿?	`string`	PutRecord 작업에 대한 API Gateway 요청 템플릿입니다.

이름	유형	설명
		제공되지 않으면 기본 값이 사용됩니다.
입력 코드 요청 모델?	`api.ModelOptions`	PutRecord 작업에 대한 API Gateway 요청 모델입니다. 제공되지 않으면 기본 항목이 생성됩니다.
푸트코드요청 템플릿?	`string`	PutRecords 작업에 대한 API Gateway 요청 템플릿입니다. 제공되지 않으면 기본 값이 사용됩니다.
입력 코드 요청 모델?	`api.ModelOptions`	PutRecords 작업에 대한 API Gateway 요청 모델입니다. 제공되지 않으면 기본 항목이 생성됩니다.
기존스트리모브?	`kinesis.Stream`	Kinesis Stream의 기존 인스턴스로, 이 인스턴스와`kinesisStreamProps` 를 사용해 오류가 발생합니다.
키네시스스트림프롭스?	`kinesis.StreamProps`	Kinesis 스트림의 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
로그그룹Props?	`logs.LogGroupProps`	선택적 사용자가 제공한 소품을 CloudWatch Logs 로그 그룹에 대한 기본 소품을 무시합니다.

패턴 속성

이름	유형	설명
ApiGateway	`api.RestApi`	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
에이피게이트 웨이의 역할	`iam.Role`	API Gateway REST API의 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
APIGateway클라우드워치역할	`iam.Role`	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
에이피가트웨이로그 그룹	`logs.LogGroup`	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.
키네시스스트림	`kinesis.Stream`	패턴에 의해 생성된 Kinesis 스트림의 인스턴스를 반환합니다.

샘플 API 사용

방법	요청 경로	요청 본문	대기열 작업	설명
POST	`/record`	`{ "data": "Hello World!",`	`kinesis:PutRecord`	단일 데이터 레코드를 스트림에 씁니다.

방법	요청 경로	요청 본문	대기열 작업	설명
		"partitionKey": "pk001" }		
POST	/records	{ "records": [{ "data": "abc", "partitionKey": "pk001" }, { "data": "xyz", "partitionKey": "pk001" }] }	kinesis:PutRecords	단일 호출로 스트림에 여러 데이터 레코드를 기록합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon API Gateway

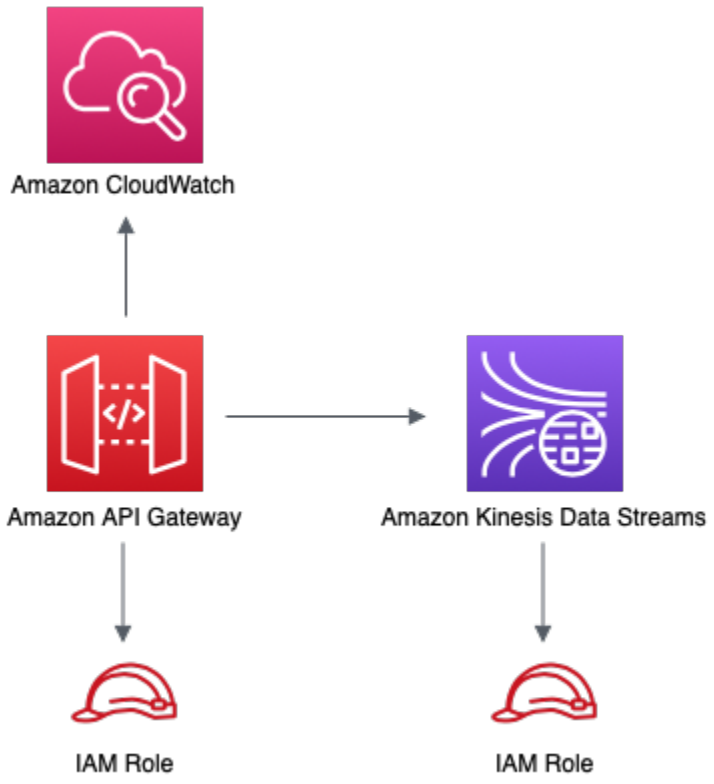
- 엣지 최적화 API 엔드포인트를 배포합니다.
- API Gateway 대해 CloudWatch 로깅을 사용하도록 설정합니다.
- API Gateway 대한 최소 권한 액세스 IAM 역할을 구성합니다.
- 모든 API 메소드의 기본 권한 부여 유형을 IAM으로 설정합니다.

- X-Ray 추적을 사용합니다.
- Kinesis 에 데이터를 전달하기 전에 요청 본문의 유효성을 검사합니다

Amazon Kinesis Data Stream

- Kinesis 스트림에 대한 최소 권한 액세스 IAM 역할을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 Kinesis 스트림에 대한 서버 측 암호화를 활성화합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws-apigateway - 키네시스스트림

aws-아피가티웨이 람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_apigateway_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-apigateway-lambda`
Java	`software.amazon.awsconstructs.services.apigatewaylambda`

Overview

이 AWS 솔루션 구성은 AWS Lambda 함수에 연결된 Amazon API Gateway REST API를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { ApiGatewayToLambda } from '@aws-solutions-constructs/aws-apigateway-lambda';

new ApiGatewayToLambda(this, 'ApiGatewayToLambdaPattern', {
    lambdaFunctionProps: {
      runtime: lambda.Runtime.NODEJS_14_X,
      // This assumes a handler function in lib/lambda/index.js
      code: lambda.Code.fromAsset(`${__dirname}/lambda`),
      handler: 'index.handler'
```

```
    }
});
```

Initializer

```
new ApiGatewayToLambda(scope: Construct, id: string, props: ApiGatewayToLambdaProps);
```

파라미터

- scopeConstruct
- idstring
- propsApiGatewayToLambdaProps

패턴 구성

이름	유형	설명
이미 람다오브?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 사용하면 오류가 발생할 수 있습니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 가 제공될 예정입니다.
아피가트웨이 소품?	api.LambdaRestApiProps	API의 기본 소품을 재정의하는 선택적 사용자 제공 소품.
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품입니다.

패턴 속성

이름	유형	설명
APIGateway클라우드워치역할	`iam.Role`	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
에이피가트웨이로그 그룹	`logs.LogGroup`	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.
람다함수	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
ApiGateway	`api.LambdaRestApi`	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

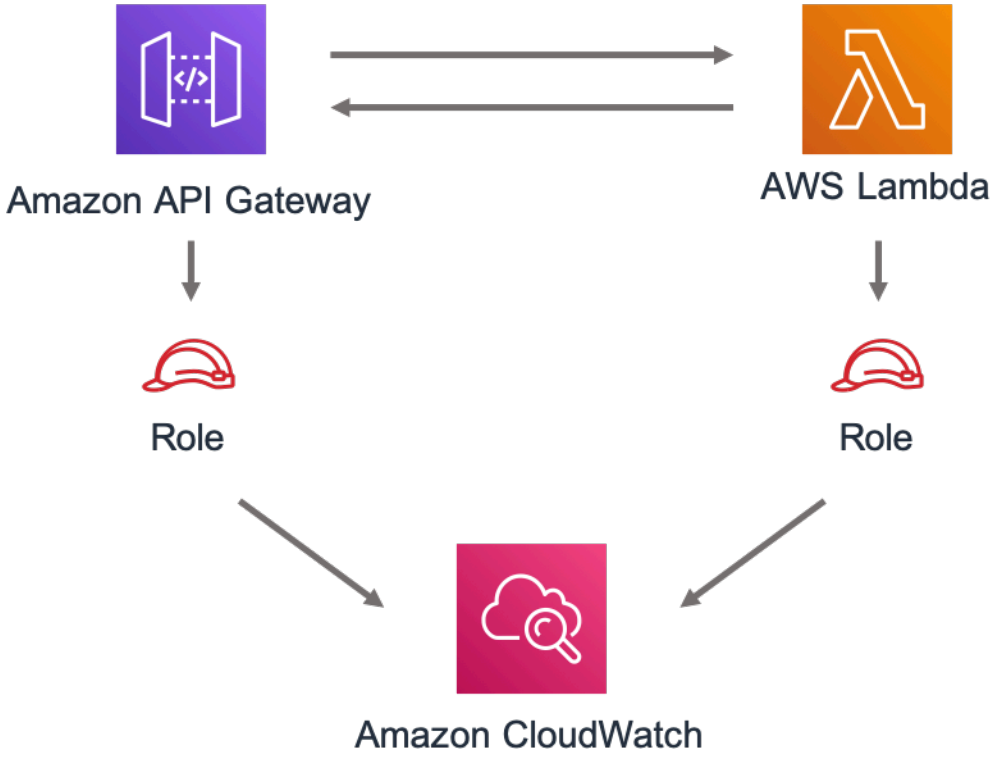
Amazon API Gateway

- 엣지 최적화 API 엔드포인트 배포
- API Gateway 대한 CloudWatch 로깅
- API Gateway 대한 최소 권한 액세스 IAM 역할 구성
- 모든 API 메소드의 기본 권한 부여 유형을 IAM으로 설정
- X-Ray 추적하기
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

AWS Lambda 함수

- Lambda 용 제한된 권한 액세스 IAM 역할 구성
- NodeJS Lambda 함수에 대한 연결 유지로 연결 재사용 활성화
- X-Ray 추적하기

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws-apigateway - 람다

아피가테이웨이 세이지메이크렌드포인트

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 사용해 보세요. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_apigateway_sagemakerendpoint`
타이프 스크립트	`@aws-solutions-constructs/aws-apigateway-sagemakerendpoint`
Java	`software.amazon.awsconstructs.services.apigatewaysagemakerendpoint`

Overview

이 AWS 솔루션 구성은 Amazon SageMaker 엔드포인트에 연결된 Amazon API 게이트웨이 REST API를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { ApiGatewayToSageMakerEndpoint, ApiGatewayToSageMakerEndpointProps } from
 '@aws-solutions-constructs/aws-apigateway-sagemakerendpoint';

// Below is an example VTL (Velocity Template Language) mapping template for mapping
 the Api GET request to the Sagemaker POST request
const requestTemplate =
`{
    "instances": [
#set( $user_id = $input.params("user_id") )
#set( $items = $input.params("items") )
```

```
#foreach( $item in $items.split(",") )
    {"in0": [$user_id], "in1": [$item]}#if( $foreach.hasNext ),#end
    $esc.newline
#end
    ]
}`;

// Replace 'my-endpoint' with your Sagemaker Inference Endpoint
new ApiGatewayToSageMakerEndpoint(this, 'test-apigw-sagemakerendpoint', {
    endpointName: 'my-endpoint',
    resourcePath: '{user_id}',
    requestMappingTemplate: requestTemplate
});
```

Initializer

```
new ApiGatewayToSageMakerEndpoint(scope: Construct, id: string, props:
 ApiGatewayToSageMakerEndpointProps);
```

파라미터

- scopeConstruct
- idstring
- propsApiGatewayToSageMakerEndpointProps

PropingTemplate

이름	유형	설명
어피게이트웨이 소품?	api.RestApiProps	API Gateway REST API의 기본 소품을 재정의하는 선택적 사용자 제공 소품.
어피게이트웨이 실행역할?	iam.Role	API Gateway 에서 SageMaker 엔드포인트를 호출하기 위해 사용하는 IAM 역할 지정되지

이름	유형	설명
		않은 경우 기본 역할이 생성되어`endpointName` .
EndpointName	`string`	배포된 SageMaker 추론 끝점의 이름입니다.
ResourceName	`string`	GET 메서드를 사용할 수 있습니다 선택적 리소스 이름입니다.
resourcePath	`string`	GET 메서드의 리소스 경로입니다. 여기에 정의 된 변수는에서 참조 할 수 있습니다`requestMappingTemplate` .
RequestMappingTemplate	`string`	REST API에서 수신된 GET 요청을 SageMaker 끝점에서 예상되는 POST 요청으로 변환하는 매핑 템플릿.
ResponseMappingTemplate	`string`	SageMaker 엔드포인트로부터 수신된 응답을 변환하는 선택적 매핑 템플릿입니다.
로그그룹Props?	`logs.LogGroupProps`	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 prop입니다.

패턴 속성

이름	유형	설명
ApiGateway way	`api.LambdaRestApi`	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
에이피게이트웨이 역할	`iam.Role`	API Gateway REST API의 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
아피게이트웨이클라우드워치 역할	`iam.Role`	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
어피게이트웨이로그 그룹	`logs.LogGroup`	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.

샘플 API 사용

참고: 각 SageMaker 끝점은 고유하며 API의 응답은 배포된 모델에 따라 달라집니다. 아래의 예는에서 샘플을 가정이 블로그 게시물. 구현 방법에 대한 참조는 다음을 참조하십시오.인테.apigateway-세이지메이크렌드포인트-덮어 쓰기.ts.

방법	요청 경로	쿼리 문자열	SageMaker 액션	설명
GET	`/321`	`items=101,131,162`	`sagemaker:InvokeEndpoint`	특정 사용자 및 항목에 대한 예측을 검색합니다.

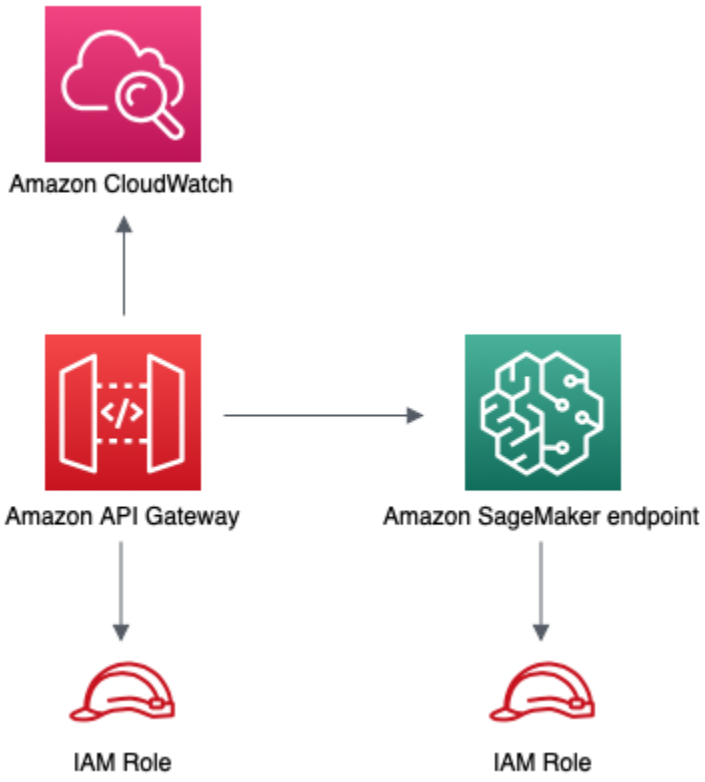
기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon API Gateway

- 엣지 최적화 API 엔드포인트 배포
- API Gateway 에 대한 CloudWatch 로깅
- API Gateway 대한 최소 권한 액세스 IAM 역할 구성
- 모든 API 메소드의 기본 권한 부여 유형을 IAM으로 설정
- X-Ray 추적
- 데이터를 SageMaker 에 전달하기 전에 요청 매개 변수의 유효성

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/aws-apigateway-세이지메이크 렌드 포인트

aws-apigateway-sqs

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 검색합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_apigateway_sqs`
타이프 스크립트	`@aws-solutions-constructs/aws-apigateway-sqs`
Java	`software.amazon.awsconstructs.services.apigatewaysqs`

Overview

이 AWS 솔루션 구성은 Amazon SQS 대기열에 연결된 Amazon API 게이트웨이 REST API를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { ApiGatewayToSqs, ApiGatewayToSqsProps } from "@aws-solutions-constructs/aws-apigateway-sqs";

new ApiGatewayToSqs(this, 'ApiGatewayToSqsPattern', {});
```

Initializer

```
new ApiGatewayToSqs(scope: Construct, id: string, props: ApiGatewayToSqsProps);
```

파라미터

- scopeConstruct
- idstring
- propsApiGatewayToSqsProps

소품 패턴 구성

이름	유형	설명
어피게이트웨이 소품?	api.RestApiProps	API Gateway 기본 소품을 재정의하는 선택적 사용자 제공 소품.
대기열 소품?	sqs.QueueProps	대기열의 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
배포데드 레터 큐?	boolean	배달 못한 편지 대기열로 사용할 보조 대기열을 배포할지 여부입니다. 기본값은 true입니다.

이름	유형	설명
maxReceiveCount	`number`	배달 못한 편지 대기열로 이동되기 전에 메시지가 대기열에서 빼지 못한 횟수입니다.
만들기작업 허용하시겠습니까?	`boolean`	큐에 만들기 작업을위한 API Gateway 메소드를 배포할지 여부 (즉, SQS:sendMessage).
요청 템플릿을 작성하시겠습니까?	`string`	Create 메서드에 대한 기본 API Gateway 요청 템플릿 재정의 (`allowCreateOperation` 다음의 경우 이 로 설정됩니다.`true`.
읽기 작업이 허용됩니까?	`boolean`	대기열에 읽기 작업을위한 API Gateway 메소드를 배포할지 여부 (즉, SQ:ReceiveMessage).
읽기 요청 템플릿	`string`	Read 메서드에 대한 기본 API Gateway 요청 템플릿을 재정의합니다 (`allowReadOperation` 다음의 경우 이로 설정됩니다.`true`.
삭제 작업을 허용하시겠습니까?	`boolean`	대기열에 삭제 작업을위한 API Gateway 메소드를 배포할지 여부 (즉, SQS:deleteMessage).
삭제 요청 템플릿?	`string`	Delete 메서드에 대한 기본 API Gateway 요청 템플릿 재정의 (`allowDeleteOperation` 다음의 경우 이 로 설정됩니다.`true`.

이름	유형	설명
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품입니다.

패턴 속성

이름	유형	설명
ApiGateway	api.RestApi	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
아피게이트웨이클라우드워치역할	iam.Role	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
어피게이트웨이로그 그룹	logs.LogGroup	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.
에이피게이트웨이의 역할	iam.Role	API Gateway REST API의 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
데드 레터 큐?	sqs.Queue	패턴에 의해 생성 된 데드 레터 큐의 인스턴스를 돌려줍니다 (배포 된 경우).
분류: SQqueue	sqs.Queue	패턴에 의해 생성된 SQS 대기열의 인스턴스를 반환합니다.

API 사용

방법	요청 경로	요청 본문	대기열 작업	설명
GET	`/`		`sqs::ReceiveMessage`	대기열에서 메시지를 검색합니다.
POST	`/`	`{ "data": "Hello World!" }`	`sqs::SendMessage`	대기열로 메시지를 전달합니다.
DELETE	`/message?receiptHandle=[value]`		`sqs::DeleteMessage`	대기열에서 지정된 메시지를 삭제합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

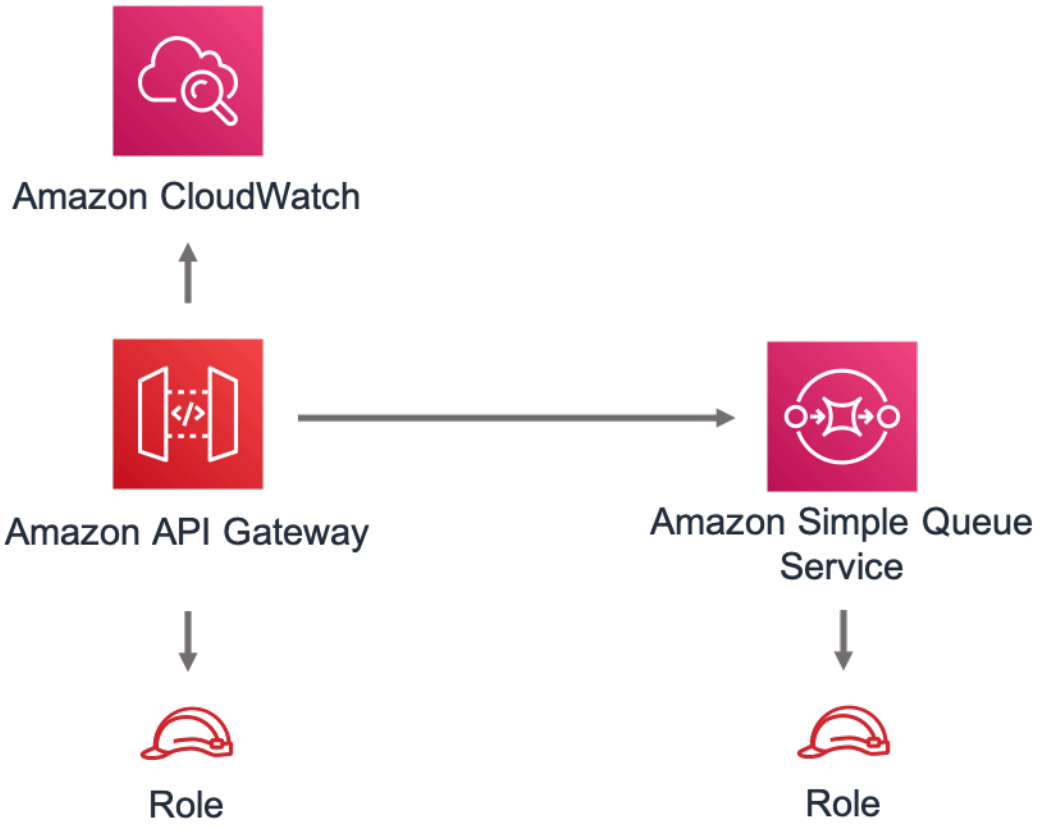
Amazon API Gateway

- 엣지 최적화 API 엔드포인트 배포
- API Gateway 대한 CloudWatch 로깅 사용
- API Gateway 대한 최소 권한 액세스 IAM 역할 구성
- 모든 API 메소드에 대한 기본 권한 부여 유형을 IAM으로 설정
- X-Ray 추적

Amazon SQS 대기열

- 소스 SQS 대기열에 대한 SQS 배달 못한 편지 대기열 배포
- AWS 관리형 KMS 키를 사용하여 소스 SQS 대기열에 대해 서버 측 암호화를 활성화합니다.
- 전송 중인 데이터의 암호화 강제 시행

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws-apigateway - 스퀘어

클라우드 프론트 어피가트웨이

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_cloudfront_apigateway`
타이프 스크립트	`@aws-solutions-constructs/aws-cloudfront-apigateway`
Java	`software.amazon.awsconstructs.services.cloudfrontapigateway`

Overview

이 AWS 솔루션 구성은 Amazon API 게이트웨이 REST API 앞에 Amazon CloudFront 배포를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import * as api from '@aws-cdk/aws-apigateway';
import * as lambda from "@aws-cdk/aws-lambda";
import { CloudFrontToApiGateway } from '@aws-solutions-constructs/aws-cloudfront-
apigateway';

const lambdaProps: lambda.FunctionProps = {
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    runtime: lambda.Runtime.NODEJS_12_X,
    handler: 'index.handler'
};

const lambdafunction = new lambda.Function(this, 'LambdaFunction', lambdaProps);

const apiGatewayProps: api.LambdaRestApiProps = {
        handler: lambdafunction,
        endpointConfiguration: {
            types: [api.EndpointType.REGIONAL]
        },
```

```
        defaultMethodOptions: {
            authorizationType: api.AuthorizationType.NONE
        }
};

const apiGateway = new api.LambdaRestApi(this, 'LambdaRestApi', apiGatewayProps);

new CloudFrontToApiGateway(this, 'test-cloudfront-apigateway', {
    existingApiGatewayObj: apiGateway
});
```

Initializer

```
new CloudFrontToApiGateway(scope: Construct, id: string, props:
 CloudFrontToApiGatewayProps);
```

파라미터

- scope`Construct`
- id`string`
- props`CloudFrontToApiGatewayProps`

소품 패턴 구성

이름	유형	설명
기존가피가테와요비	`api.RestApi`	CloudFront 와 함께 선두될 지역별 API Gateway
클라우드프런트배포소품?	`cloudfront.DistributionProps`	선택적 사용자가 CloudFront 배포에 대한 기본 소품을 재정의하는 소품을 제공했습니다.
보안 헤더를 삽입하시겠습니까?	`boolean`	CloudFront 모든 응답에서 모범 사례 HTTP 보안 헤더의 자

이름	유형	설명
		동 삽입을 켜거나 끌 수 있는 선택적 사용자 제공 소품

패턴 속성

이름	유형	설명
ApigGateway	`api.RestApi`	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
클라우드 프론트로깅 버킷?	`s3.Bucket`	CloudFront 웹 배포의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.
클라우드프런트웹배포	`cloudfront.CloudFrontWebDistribution`	패턴에 의해 생성된 CloudFront 웹 배포의 인스턴스를 반환합니다.
엣지람다기능 버전?	`lambda.Version`	패턴에 의해 생성 된 Lambda edge 함수 버전의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon CloudFront

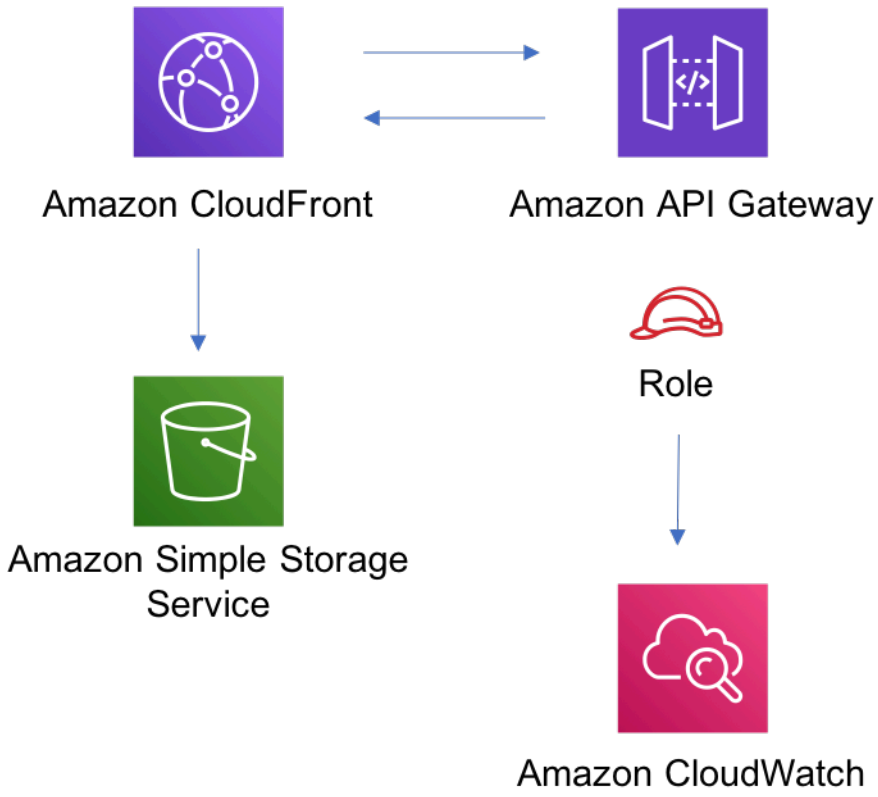
- CloudFront 웹 배포에 대한 액세스 로깅 구성
- CloudFront 웹 배포의 모든 응답에서 모범 사례 HTTP 보안 헤더의 자동 삽입 사용

Amazon API Gateway

- 사용자 제공 API Gateway 객체는 그대로 사용됩니다.

- X-Ray 추적 사용

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구조/AWS - 클라우드 프론트 어피 게이트 웨이

aws-클라우드 프론트 - 아피 가티 웨이 - 람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 참조하십시오. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_cloudfront_apigateway_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-cloudfront-apigateway-lambda`
Java	`software.amazon.awsconstructs.services.cloudfrontapigatewaylambda`

Overview

이 AWS 솔루션 구성은 Amazon API 게이트웨이 람다 지원 REST API 앞에 Amazon CloudFront 배포를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { CloudFrontToApiGatewayToLambda } from '@aws-solutions-constructs/aws-cloudfront-apigateway-lambda';

new CloudFrontToApiGatewayToLambda(this, 'test-cloudfront-apigateway-lambda', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
```

```
});
```

Initializer

```
new CloudFrontToApiGatewayToLambda(scope: Construct, id: string, props:
 CloudFrontToApiGatewayToLambdaProps);
```

파라미터

- scope`Construct`
- id`string`
- props`CloudFrontToApiGatewayToLambdaProps`

패턴 구성

이름	유형	설명
람다오브 기존인가요?	`lambda.Function`	Lambda 함수 객체의 기존 인스턴스, 이 및`lambdaFunctionProps` 오류가 발생합니다.
람다기능소품?	`lambda.FunctionProps`	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.`existingLambdaObj` 제공될 예정입니다.
어피게이트웨이 소품?	`api.LambdaRestApiProps`	API Gateway 기본 소품을 재정의하는 선택적 사용자 제공 소품

이름	유형	설명
클라우드프런트배포소품?	cloudfront.DistributionProps	선택적 사용자가 CloudFront 배포에 대한 기본 소품을 재정의하는 소품을 제공했습니다.
보안 헤더를 삽입하시겠습니까?	boolean	CloudFront 모든 응답에서 모범 사례 HTTP 보안 헤더의 자동 삽입을 켜거나 끌 수 있는 선택적 사용자 제공 소품
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품입니다.

패턴 속성

이름	유형	설명
ApiGateway	api.RestApi	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
아피게이트웨이클라우드워치역할	iam.Role	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
어피게이트웨이로그 그룹	logs.LogGroup	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.
클라우드 프론트로깅 버킷?	s3.Bucket	CloudFront 웹 배포의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.

이름	유형	설명
클라우드프런트웹배포	cloudfront.CloudFrontWebDistribution	패턴에 의해 생성된 CloudFront 웹 배포의 인스턴스를 반환합니다.
엣지람다기능 버전?	lambda.Version	패턴에 의해 생성 된 Lambda edge 함수 버전의 인스턴스를 돌려줍니다.
람다함수	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon CloudFront

- CloudFront 웹 배포에 대한 액세스 로깅 구성
- CloudFront 웹 배포의 모든 응답에서 모범 사례 HTTP 보안 헤더의 자동 삽입 사용

Amazon API Gateway

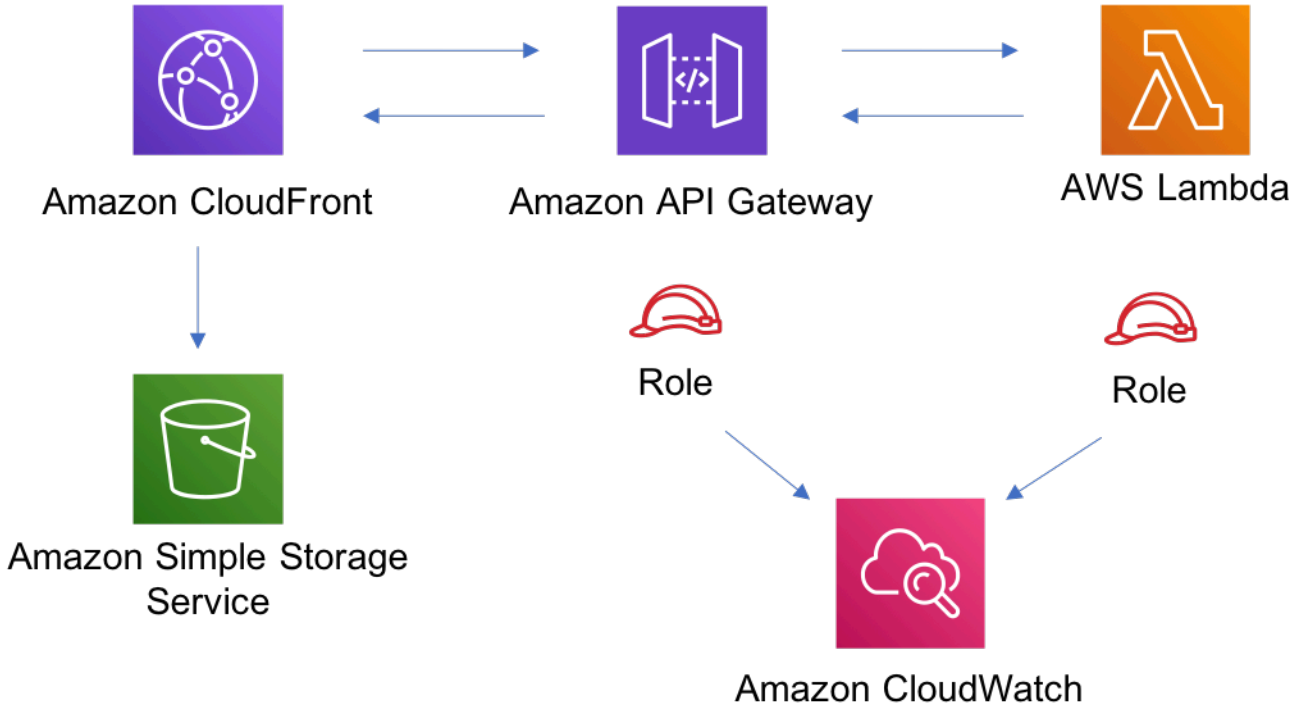
- 지역 API 끝점 배포
- API Gateway 대한 CloudWatch 로깅 활성화
- API Gateway 대한 최소 권한 액세스 IAM 역할 구성
- 모든 API 메소드의 기본 권한 부여 유형을 IAM으로 설정
- X-Ray 추적 설정

AWS Lambda 함수

- Lambda 용 제한된 권한 액세스 IAM 역할을 구성하려면,
- NodeJS Lambda 함수에 대한 연결 유지로 연결 재사용 활성화
- X-Ray 추적 설정
- 환경 변수를 설정합니다.

- `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws - 클라우드 프론트 어피 그웨이 - 람다

AWS 클라우드 프런트 미디어 스토어

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 설정합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_cloudfront_mediastore`
타이프 스크립트	`@aws-solutions-constructs/aws-cloudfront-mediastore`
Java	`software.amazon.awsconstructs.services.cloudfrontmediastore`

Overview

이 AWS 솔루션 구성은 AWS Elemental MediaStore 컨테이너에 연결된 Amazon CloudFront 배포를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { CloudFrontToMediaStore } from '@aws-solutions-constructs/aws-cloudfront-mediastore';

new CloudFrontToMediaStore(this, 'test-cloudfront-mediastore-default', {});
```

Initializer

```
new CloudFrontToMediaStore(scope: Construct, id: string, props: CloudFrontToMediaStoreProps);
```

파라미터

- scopeConstruct
- idstring
- propsCloudFrontToMediaStoreProps

패턴 구성 소품

이름	유형	설명
기존미디어스토어콘테이너Robj?	mediastore.CfnContainer	기본 MediaStore 컨테이너를 재정의하는 선택적 사용자 제공 MediaStore 컨테이너입니다.
미디어저장소컨테이너Props?	mediastore.CfnContainerProps	MediaStore 컨테이너의 기본 소품을 재정의하는 선택적 사용자 제공 소품.
클라우드프런트배포소품?	cloudfront.DistributionProps | any	CloudFront 배포에 대한 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
보안 헤더를 삽입하시겠습니까?	boolean	CloudFront 모든 응답에서 모범 사례 HTTP 보안 헤더의 자동 삽입을 켜거나 끄는 선택적 사용자 제공 소품.

패턴 특성

이름	유형	설명
클라우드프런트웹배포	cloudfront.CloudFrontWebDistribution	패턴에 의해 생성된 CloudFront 웹 배포의 인스턴스를 반환합니다.

이름	유형	설명
미디어스토어 컨테이너	`mediastore.CfnContainer`	패턴에 의해 생성 된 MediaStore 컨테이너의 인스턴스를 돌려줍니다.
클라우드 프런트로깅 버킷	`s3.Bucket`	CloudFront 웹 배포의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.
클라우드프론트오리진 요청 정책	`cloudfront.OriginRequestPolicy`	CloudFront 웹 배포에 대한 패턴으로 생성된 CloudFront 오리진 요청 정책의 인스턴스를 반환합니다.
클라우드프론트오리진 액세스 권한?	`cloudfront.OriginAccessIdentity`	CloudFront 웹 배포에 대한 패턴으로 생성된 CloudFront 원본 액세스 ID의 인스턴스를 반환합니다.
엣지람다기능버전	`lambda.Version`	패턴에 의해 생성 된 Lambda edge 함수 버전의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon CloudFront

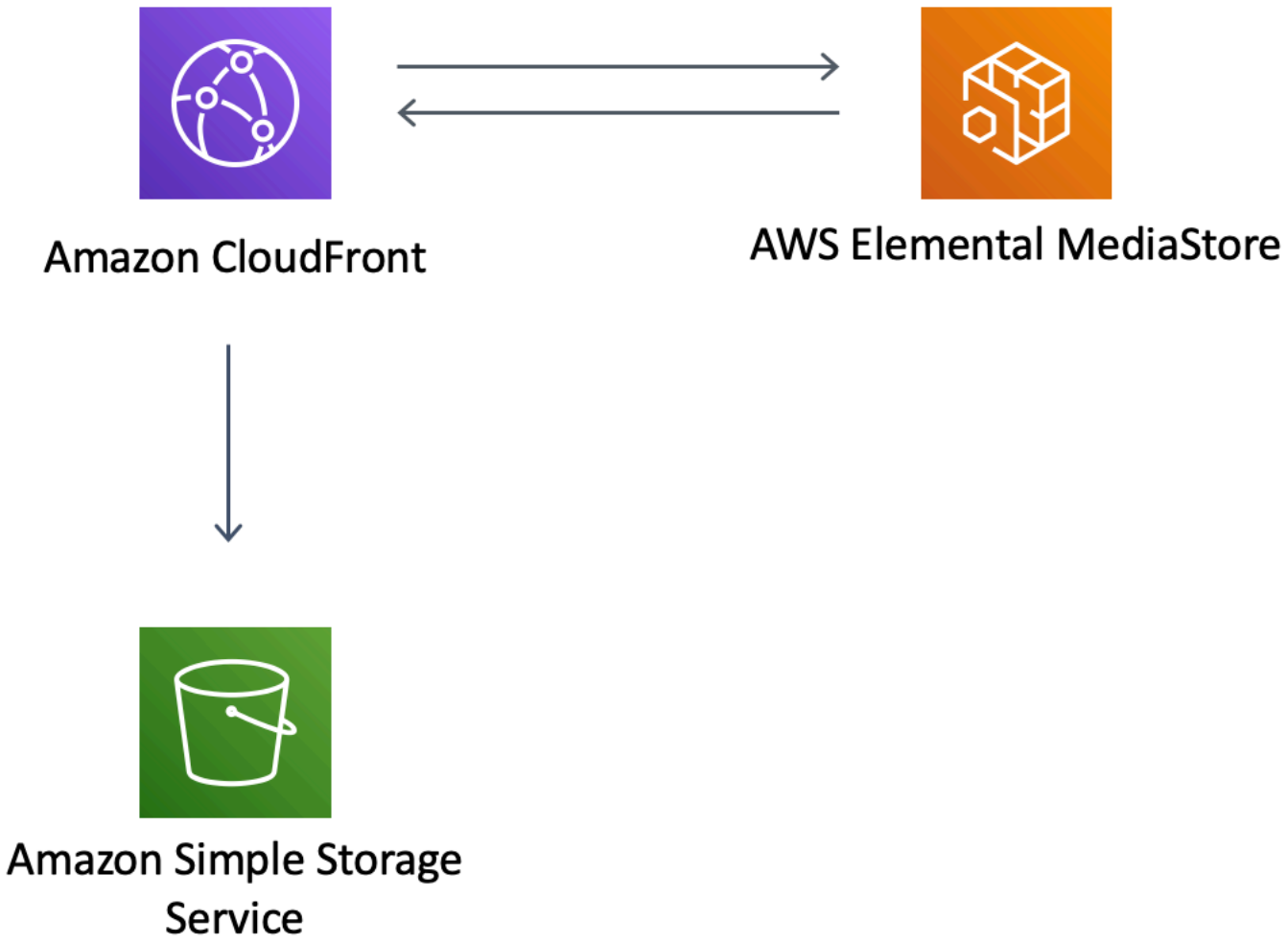
- CloudFront 웹 배포에 대한 액세스 로깅 구성
- AWS Elemental MediaStore 컨테이너에 대해 CloudFront 오리진 요청 정책 활성화
- Set`User-Agent`CloudFront 원본 액세스 ID를 사용하는 사용자 지정 헤더
- CloudFront 웹 배포의 모든 응답에 모범 사례 HTTP 보안 헤더 자동 삽입 활성화

AWS Elemental MediaStore

- 리소스를 보존하도록 삭제 정책 설정

- CloudFormation 스택 이름으로 컨테이너 이름 설정
- 기본 설정컨테이너 Cross-Origin Resource Sharing (
- 기본 설정객체 수명 주기 정책
- 기본 설정컨테이너 정책만 허용하도록`aws:UserAgent`CloudFront 원본 액세스 ID
- 기본 설정지표 정책
- 액세스 로그 활성화

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/AWS - 클라우드 프론트 미디어 스토어

AWS-클라우드프런트 S3

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_cloudfront_s3`
타이프 스크립트	`@aws-solutions-constructs/aws-cloudfront-s3`
Java	`software.amazon.awsconstructs.services.cloudfronts3`

Overview

이 AWS 솔루션 구성은 Amazon S3 버킷 앞에 Amazon CloudFront 배포를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { CloudFrontToS3 } from '@aws-solutions-constructs/aws-cloudfront-s3';

new CloudFrontToS3(this, 'test-cloudfront-s3', {});
```

Initializer

```
new CloudFrontToS3(scope: Construct, id: string, props: CloudFrontToS3Props);
```

파라미터

- scopeConstruct
- idstring
- propsCloudFrontToS3Props

패턴 구성

이름	유형	설명
버켓토비 기존에?	s3.Bucket	S3 버킷 객체의 기존 인스턴스입니다. 이것이 제공되는 경우bucketProps 는 오류입니다.
버킷 소품?	s3.BucketProps	버킷의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingBucketObj 제공될 것입니다.
클라우드프런트배포소품을 사용하시겠습니까?	cloudfront.DistributionProps	선택적 사용자가 CloudFront 배포에 대한 기본 소품을 재정의하는 소품을 제공했습니다.

이름	유형	설명
보안 헤더를 삽입하시겠습니까?	boolean	CloudFront 모든 응답에서 모범 사례 HTTP 보안 헤더의 자동 삽입을 켜거나 끌 수 있는 선택적 사용자 제공 소품

패턴 속성

이름	유형	설명
클라우드프런트웹배포	cloudfront.CloudFrontWebDistribution	패턴에 의해 생성된 CloudFront 웹 배포의 인스턴스를 반환합니다.
S3Bucket?	s3.Bucket	패턴에 의해 생성된 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	s3.Bucket	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.
엣지람다 기능 버전?	lambda.Version	패턴에 의해 생성 된 Lambda edge 함수 버전의 인스턴스를 돌려줍니다.
클라우드 프론트로깅 버킷?	s3.Bucket	CloudFront 웹 배포에 대한 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon CloudFront

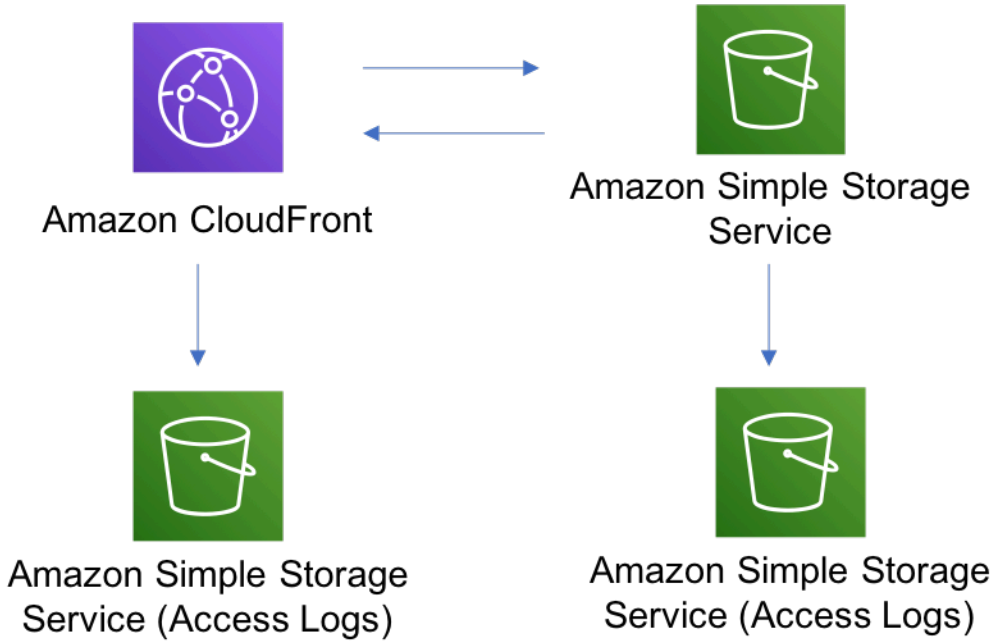
- CloudFront 웹 배포에 대한 액세스 로깅 구성

- CloudFront 웹 배포의 모든 응답에서 모범 사례 HTTP 보안 헤더의 자동 삽입 사용

Amazon S3 버킷

- S3 버킷에 대한 액세스 로깅 구성
- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화 사용
- S3 버킷의 버전 관리 켜기
- S3 버킷에 대한 공용 액세스 허용 안 함
- CloudFormation 스택을 삭제할 때 S3 버킷 유지
- 전송 중인 데이터의 암호화 강제 시행
- 90일 후에 비최신 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	[@aws -솔루션 - 구성/AWS - 클라우드 프런트 S3]

aws-코그니토-아피가테이웨이 - 람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_cognito_apigateway_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-cognito-apigateway-lambda`
Java	`software.amazon.awsconstructs.services.cognitoapigatewaylambda`

Overview

이 AWS 솔루션 구성은 Amazon API 게이트웨이 람다 지원 REST API를 보호하는 Amazon Cognito를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { CognitoToApiGatewayToLambda } from '@aws-solutions-constructs/aws-cognito-apigateway-lambda';

new CognitoToApiGatewayToLambda(this, 'test-cognito-apigateway-lambda', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
```

```
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

API에서 리소스와 메소드를 정의하는 경우 (예:`proxy = false`) 를 호출해야 합니다.`addAuthorizers()`API가 완전히 정의 된 후 메서드입니다. 이렇게하면 API의 모든 메소드가 보호됩니다.

다음은 TypeScript 터의 예입니다.

```
import { CognitoToApiGatewayToLambda } from '@aws-solutions-constructs/aws-cognito-
apigateway-lambda';

const construct = new CognitoToApiGatewayToLambda(this, 'test-cognito-apigateway-
lambda', {
    lambdaFunctionProps: {
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        runtime: lambda.Runtime.NODEJS_12_X,
        handler: 'index.handler'
    },
    apiGatewayProps: {
      proxy: false
    }
});

const resource = construct.apiGateway.root.addResource('foobar');
resource.addMethod('POST');

// Mandatory to call this method to Apply the Cognito Authorizers on all API methods
construct.addAuthorizers();
```

Initializer

```
new CognitoToApiGatewayToLambda(scope: Construct, id: string, props:
 CognitoToApiGatewayToLambdaProps);
```

파라미터

- scopeConstruct
- idstring
- propsCognitoToApiGatewayToLambdaProps

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 호출하면 오류가 발생합니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 가 제공될 예정입니다.
아피가트웨이 소품?	api.LambdaRestApiProps	API Gateway 기본 소품을 재정의하는 선택적 사용자 제공 소품
Cognitouser풀Props?	cognito.UserPoolProps	Cognito 사용자 풀의 기본 소품을 재정의하는 선택적 사용자 제공 소품
Cognitouser풀클라이언트Props?	cognito.UserPoolClientProps	Cognito 사용자 풀 클라이언트에 대한 기본 소품을 재정의하는 선택적 사용자 제공 prop
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹에 대한 기본 소품을 무시하기 위

이름	유형	설명
		한 선택적 사용자 제공 소품입니다.

패턴 속성

이름	유형	설명
ApiGateway	`api.RestApi`	패턴에 의해 생성 된 API Gateway REST API의 인스턴스를 돌려줍니다.
람다함수	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
userPool	`cognito.UserPool`	패턴에 의해 생성된 Cognito 사용자 풀의 인스턴스를 반환합니다.
UserPool	`cognito.UserPoolClient`	패턴에 의해 생성된 Cognito 사용자 풀 클라이언트의 인스턴스를 반환합니다.
APIGateway클라우드워치역할	`iam.Role`	API Gateway REST API에서 CloudWatch 로의 액세스 로깅을 활성화하는 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
에이피가트웨이로그 그룹	`logs.LogGroup`	API Gateway REST API 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.

이름	유형	설명
어피가트웨이 인가자	`api.CfnAuthorizer`	패턴에 의해 생성 된 API Gateway 인가자의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon Cognito

- 사용자 풀에 대한 암호 정책 설정
- 사용자 풀에 고급 보안 모드 적용

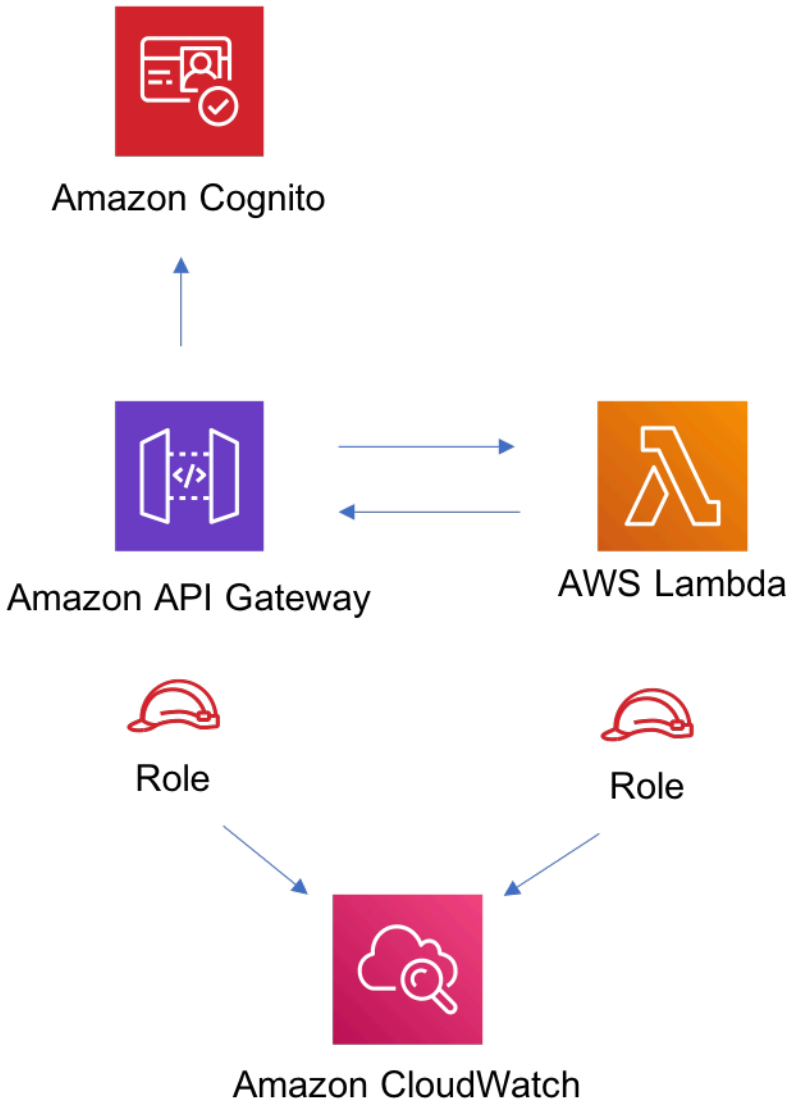
Amazon API Gateway

- 엣지 최적화 API 엔드포인트 배포
- API Gateway 대한 CloudWatch 로깅 설정
- API Gateway 대한 최소 권한 액세스 IAM 역할 구성
- 모든 API 메소드에 대한 기본 권한 부여 유형을 IAM으로 설정
- X-Ray 추적하기

AWS Lambda 함수

- Lambda 용 제한된 권한 액세스 IAM 역할 구성
- NodeJS Lambda 함수에 대한 연결 유지로 연결 재사용 사용
- X-Ray 추적하기
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws - 인식 - 아피 가티 웨이 - 람다

AWS-Dynamodb-stream-lambda-lambda-stream

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_dynamodb_stream_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-dynamodb-stream-lambda`
Java	`software.amazon.awsconstructs.services.dynamodbstreamlambda`

Overview

이 AWS 솔루션 구성은 최소한의 권한으로 AWS Lambda 함수를 호출하기 위한 스트림이 있는 Amazon DynamoDB 테이블 패턴을 구현합니다.

다음은 최소한의 배포 가능한 패턴 정의입니다.

```
import { DynamoDBStreamToLambdaProps, DynamoDBStreamToLambda} from '@aws-solutions-constructs/aws-dynamodb-stream-lambda';

new DynamoDBStreamToLambda(this, 'test-dynamodb-stream-lambda', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    },
```

```
});
```

Initializer

```
new DynamoDBStreamToLambda(scope: Construct, id: string, props:
 DynamoDBStreamToLambdaProps);
```

파라미터

- scope`Construct`
- id`string`
- props`DynamoDBStreamToLambdaProps`

패턴 구성 소품

이름	유형	설명
이미 람다오브?	`lambda.Function`	Lambda 함수 객체의 기존 인스턴스, 이 및`lambdaFunctionProps` 오류가 발생합니다.
람다기능소품?	`lambda.FunctionProps`	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.`existingLambdaObj` 제공될 예정입니다.
다이나모테이블소품?	`dynamodb.TableProps`	DynamoDB 테이블의 기본 소품을 무시하기 위한 선택적 사용자가 제공한 소품
기존 테이블오브J?	`dynamodb.Table`	DynamoDB 테이블 객체의 기존 인스턴스로, 이 인스턴스

이름	유형	설명
		와dynamoTableProps 오류가 발생합니다.
다이나모이벤트소품?	aws-lambda-event-sources.DynamoEventSourceProps	DynamoDB 이벤트 소스의 기본 소품을 재정의할 수 있는 선택적 사용자 제공 소품

패턴 속성

이름	유형	설명
다이나모터블	dynamodb.Table	패턴에 의해 생성된 DynamoDB 테이블의 인스턴스를 반환합니다.
lambdaFunction	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.

Lambda 함수

이 패턴을 사용하려면 DynamoDB 스트림에서 Elasticsearch 서비스에 데이터를 게시할 수 있는 Lambda 함수가 필요합니다. 샘플 함수가 제공됩니다.여기에서.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

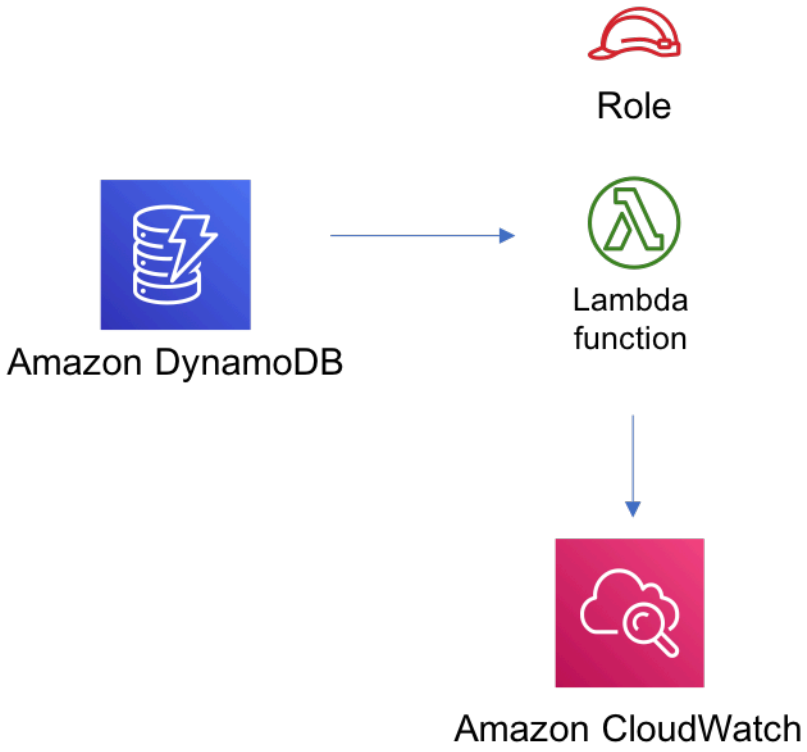
Amazon DynamoDB 테이블

- DynamoDB 테이블의 결제 모드를 온 디맨드 (요청당 지불) 로 설정합니다.
- AWS 관리형 KMS 키를 사용하여 DynamoDB 테이블에 대한 서버 측 암호화 활성화
- DynamoDB 테이블에 대해 'id'라는 파티션 키를 생성합니다.
- CloudFormation 스택을 삭제할 때 테이블 유지
- 지속적인 백업 및 지정 시간 복구

AWS Lambda 함수

- Lambda 용 제한된 권한 액세스 IAM 역할 구성
- NodeJS Lambda 함수에 대한 연결 유지로 연결 재사용 활성화
- X-Ray 추적 사용
- 실패 처리 기능 사용: 함수 오류 시 이분절 사용, 기본 최대 레코드 사용 기간 (24시간) 설정, 기본 최대 재시도 횟수 (500) 설정, 실패 시 SQS 데드 레터 큐를 대상으로 배포
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - 동적 DB - 스트림 - 람다

aws-다이나모드-스트림-람다-탄성 검색-키바나

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_dynamodb_stream_lambda_elasticsearch_kibana`
타이프 스크립트	`@aws-solutions-constructs/aws-dynamodb-stream-lambda-elasticsearch-kibana`
Java	`software.amazon.awsconstructs.services.dynamodbstreamlambdaelasticsearchkibana`

Overview

이 AWS 솔루션 구성은 스트림, AWS Lambda 함수 및 최소 권한이 있는 Amazon Elasticsearch Service 포함하는 Amazon DynamoDB 테이블을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { DynamoDBStreamToLambdaToElasticSearchAndKibana,
 DynamoDBStreamToLambdaToElasticSearchAndKibanaProps } from '@aws-solutions-constructs/
aws-dynamodb-stream-lambda-elasticsearch-kibana';
import { Aws } from "@aws-cdk/core";

const props: DynamoDBStreamToLambdaToElasticSearchAndKibanaProps = {
```

```
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    },
    domainName: 'test-domain',
    // TODO: Ensure the Cognito domain name is globally unique
    cognitoDomainName: 'globallyuniquedomain' + Aws.ACCOUNT_ID;
};

new DynamoDBStreamToLambdaToElasticSearchAndKibana(this, 'test-dynamodb-stream-lambda-
elasticsearch-kibana', props);
```

Initializer

```
new DynamoDBStreamToLambdaToElasticSearchAndKibana(scope: Construct, id: string, props:
 DynamoDBStreamToLambdaToElasticSearchAndKibanaProps);
```

파라미터

- scope`Construct`
- id`string`
- props`DynamoDBStreamToLambdaToElasticSearchAndKibanaProps`

소품 패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	`lambda.Function`	Lambda 함수 객체의 기존 인스턴스, 이 및`lambdaFunctionProps` 오류가 발생합니다.
람다기능소품?	`lambda.FunctionProps`	Lambda 함수의 기본 속성을 재정의하는 선택적 사용

이름	유형	설명
		자 제공 속성입니다. 의 경우 무시됩니다.existingLambdaObj 가 제공될 예정입니다.
다이나모테이블소품?	dynamodb.TableProps	DynamoDB 테이블의 기본 소품을 재정의할 수 있는 선택적 사용자가 제공한 소품
기존 테이블오브J?	dynamodb.Table	DynamoDB 테이블 객체의 기존 인스턴스로,dynamoTableProps 오류가 발생합니다.
다이나모이벤트소품?	aws-lambda-event-sources.DynamoEventSourceProps	DynamoDB 이벤트 소스의 기본 소품을 재정의할 수 있는 선택적 사용자 제공 소품
이도메인Props?	elasticsearch.CfnDomainProps	Amazon Elasticsearch Service 기본 소품을 무시하기 위한 선택적 사용자가 제공한 소품
domainName	string	Cognito 및 Amazon Elasticsearch Service 도메인 이름
클라우드왓찰암스 만들기	boolean	권장 CloudWatch 경보를 생성할지 여부입니다.

패턴 속성

이름	유형	설명
클라우드왓찰암즈?	cloudwatch.Alarm[]	패턴에 의해 생성된 하나 이상의 CloudWatch 경보 목록을 반환합니다.

이름	유형	설명
다이나모터블	dynamodb.Table	패턴에 의해 생성된 DynamoDB 테이블의 인스턴스를 반환합니다.
엘라스틱검색 도메인	elasticsearch.CfnDomain	패턴에 의해 생성된 Elasticsearch 도메인의 인스턴스를 반환합니다.
IdentityPool	cognito.CfnIdentityPool	패턴에 의해 생성된 Cognito 자격 증명 풀의 인스턴스를 반환합니다.
람다함수	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
userPool	cognito.UserPool	패턴에 의해 생성된 Cognito 사용자 풀의 인스턴스를 반환합니다.
UserPool	cognito.UserPoolClient	패턴에 의해 생성된 Cognito 사용자 풀 클라이언트의 인스턴스를 반환합니다.

Lambda 함수

이 패턴을 사용하려면 DynamoDB 스트림에서 Elasticsearch 서비스에 데이터를 게시할 수 있는 Lambda 함수가 필요합니다. 샘플 함수가 제공됩니다.여기에서.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon DynamoDB 테이블

- DynamoDB 테이블의 결제 모드를 온 디맨드 (요청당 지불) 로 설정
- AWS 관리형 KMS 키를 사용하여 DynamoDB 테이블에 대한 서버 측 암호화 활성화

- DynamoDB 테이블에 대해 'id'라는 파티션 키를 생성합니다.
- CloudFormation 스택을 삭제할 때 테이블 유지
- 지속적인 백업 및 지정 시간 복구

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할 구성
- NodeJS Lambda 함수에 대한 연결 유지로 연결 재사용 사용
- X-Ray 추적 활성화
- 실패 처리 기능 사용: 함수 오류 시 이분절 사용, 기본 최대 레코드 사용 기간 (24시간) 설정, 기본 최대 재시도 횟수 (500) 설정, 실패 시 SQS 데드 레터 큐를 대상으로 배포
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

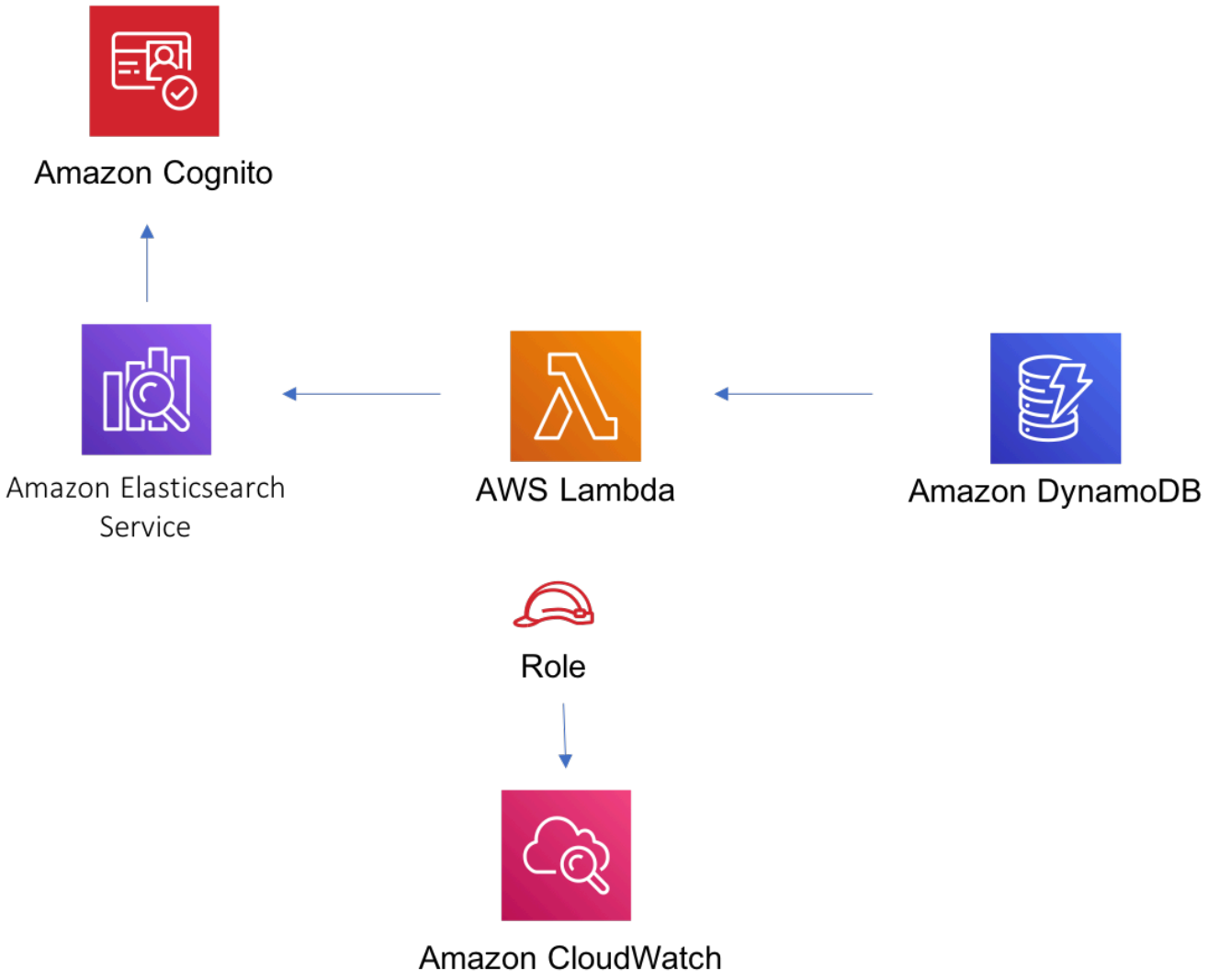
Amazon Cognito

- 사용자 풀에 대한 암호 정책 설정
- 사용자 풀에 고급 보안 모드 적용

Amazon Elasticsearch Service

- 엘라스틱 검색 도메인에 대한 CloudWatch 경보 모범 사례 배포
- Cognito 사용자 풀로 Kibana 대시보드 액세스 보호
- AWS 관리형 KMS 키를 사용하여 Elasticsearch 도메인에 대한 서버 측 암호화 활성화
- ElasticSearch 도메인에 대해 노드 간 암호화를 활성화합니다.
- Amazon ES 도메인에 대한 클러스터 구성

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - 동적 DB - 스트림 - 람다 - 탄성 검색 - 키바나

AWS-이벤트-규칙-키네시스파이어호스-s3

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_events_rule_kinesisfirehose_s3`
타이프 스크립트	`@aws-solutions-constructs/aws-events-rule-kinesisfirehose-s3`
Java	`software.amazon.awsconstructs.services.eventsrulekinesisfirehoses3`

Overview

이 AWS 솔루션 구성은 Amazon CloudWatch Events 규칙을 구현하여 Amazon S3 버킷에 연결된 Amazon Kinesis Data Firehose 전송 스트림으로 데이터를 전송합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import * as cdk from '@aws-cdk/core';
import { EventsRuleToKinesisFirehoseToS3, EventsRuleToKinesisFirehoseToS3Props } from
 '@aws-solutions-constructs/aws-events-rule-kinesisfirehose-s3';

const eventsRuleToKinesisFirehoseToS3Props: EventsRuleToKinesisFirehoseToS3Props = {
    eventRuleProps: {
    schedule: events.Schedule.rate(cdk.Duration.minutes(5))
    }
};
```

```
new EventsRuleToKinesisFirehoseToS3(this, 'test-events-rule-firehose-s3',
 eventsRuleToKinesisFirehoseToS3Props);
```

Initializer

```
new EventsRuleToKinesisFirehoseToS3(scope: Construct, id: string, props:
 EventsRuleToKinesisFirehoseToS3Props);
```

파라미터

- scope`Construct`
- id`string`
- props`EventsRuleToKinesisFirehoseToS3Props`

패턴 구성

이름	유형	설명
이벤트루프로프	`events.RuleProps`	CloudWatch 이벤트 규칙의 기본 속성을 재정의하는 사용자 제공 속성입니다.
키네시스파이어호스프로프스?	`aws-kinesisfirehose.CfnDeliveryStreamProps`	Kinesis 파이어호스 배달 스트림의 기본 소품을 무시하기 위한 선택적 사용자가 제공한 소품입니다.
버켓토비 기존에?	`s3.IBucket`	S3 버킷 객체의 기존 인스턴스입니다. 이것이 제공되는 경우`bucketProps` 는 오류입니다.
버킷 소품?	`s3.BucketProps`	S3 버킷의 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.

이름	유형	설명
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품.

패턴 속성

이름	유형	설명
이벤트 규칙	events.Rule	패턴에 의해 생성된 Events 규칙의 인스턴스를 반환합니다.
키네시스파이어호스	kinesisfirehose.CfnDeliveryStream	패턴에 의해 생성된 Kinesis 파이어호스 전달 스트림의 인스턴스를 반환합니다.
S3Bucket	s3.Bucket	패턴으로 생성한 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	s3.Bucket	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.
이벤트 역할?	iam.Role	CloudWatch 이벤트 규칙에 대해 구조에서 생성한 역할의 인스턴스를 반환합니다.
키네시스파이어호스롤	iam.Role	Kinesis Firehose 전송 스트림의 패턴에 의해 생성된 IAM 역할의 인스턴스를 반환합니다.
키네시스파이어호스셀로그그룹	logs.LogGroup	Kinesis Firehose 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon CloudWatch Events 규칙

- 이벤트 규칙에 대한 최소 권한 액세스 IAM 역할을 Kinesis Firehose 전송 스트림에 게시하도록 구성합니다.

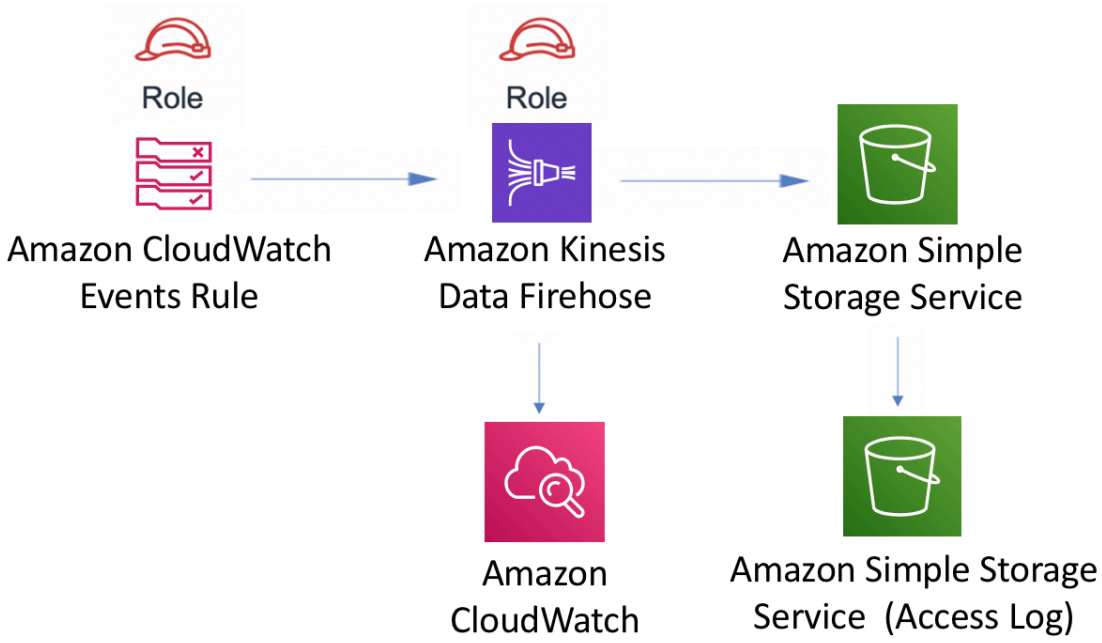
Amazon Kinesis Firehose

- Kinesis 파이어호스에 대해 CloudWatch 로깅을 활성화합니다
- Amazon Kinesis Firehose 에 대한 최소 권한 액세스 IAM 역할을 구성합니다.

Amazon S3 버킷

- 버킷에 대한 액세스 로깅을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 버킷에 대한 서버 측 암호화를 활성화합니다.
- 버킷의 버전 관리를 설정합니다.
- 버킷에 대한 퍼블릭 액세스를 허용하지 않습니다.
- CloudFormation 스택을 삭제할 때 버킷을 유지합니다.
- 90일 후에 비최신 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - 이벤트 - 규칙 - 키네시스파이어 호스-s3

AWS-이벤트-규칙-키네시스스트림

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_events_rule_kinesisstream`
타이프 스크립트	`@aws-solutions-constructs/aws-events-rule-kinesisstreams`
Java	`software.amazon.awsconstructs.services.eventsrulekinesisstream`

Overview

이 AWS 솔루션 구성은 Amazon CloudWatch Events 규칙을 구현하여 Amazon Kinesis 데이터 스트림으로 데이터를 전송합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import * as cdk from '@aws-cdk/core';
import {EventsRuleToKinesisStreams, EventsRuleToKinesisStreamsProps} from "@aws-solutions-constructs/aws-events-rule-kinesisstreams";

const props: EventsRuleToKinesisStreamsProps = {
    eventRuleProps: {
      schedule: events.Schedule.rate(Duration.minutes(5)),
    }
};

new EventsRuleToKinesisStreams(this, 'test-events-rule-kinesis-stream', props);
```

Initializer

```
new EventsRuleToKinesisStreams(scope: Construct, id: string, props:
 EventsRuleToKinesisStreamsProps);
```

파라미터

- scopeConstruct
- idstring
- propsEventsRuleToKinesisStreamsProps

Props 패턴 구성

이름	유형	설명
이벤트루프로프	events.RuleProps	CloudWatch 이벤트 규칙의 기본 속성을 재정의하는 사용자 제공 속성입니다.
기존스트리모브?	kinesis.Stream	Kinesis 스트림의 기존 인스턴스. 이 인스턴스와kinesisStreamProps 오류가 발생합니다.
키네시스스트림프롭스?	kinesis.StreamProps	Kinesis 스트림의 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
클라우드왓찰암스 만들기	boolean	권장 CloudWatch 경보를 생성할지 여부입니다.

패턴 속성

이름	유형	설명
이벤트 규칙	events.Rule	패턴에 의해 생성된 Events 규칙의 인스턴스를 반환합니다.

이름	유형	설명
키네시스스트림	`kinesis.Stream`	패턴에 의해 생성된 Kinesis 스트림의 인스턴스를 반환합니다.
이벤트 역할?	`iam.Role`	CloudWatch 이벤트 규칙에 대해 구조에서 생성한 역할의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon CloudWatch Events

- Kinesis 데이터 스트림에 게시할 이벤트 규칙에 대한 최소 권한 액세스 IAM 역할을 구성합니다.

Amazon Kinesis Stream

- AWS 관리형 KMS 키를 사용하여 Kinesis 데이터 스트림에 대한 서버 측 암호화를 활성화합니다.

Architecture

GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - 이벤트 - 규칙 - 운동 스트림

aws-이벤트-규칙 - 람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_events_rule_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-events-rule-lambda`
Java	`software.amazon.awsconstructs.services.eventsrulelambda`

Overview

이 AWS 솔루션 구조는 AWS 이벤트 규칙과 AWS Lambda 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
const { EventsRuleToLambdaProps, EventsRuleToLambda } from '@aws-solutions-constructs/
aws-events-rule-lambda';

const props: EventsRuleToLambdaProps = {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    },
    eventRuleProps: {
      schedule: events.Schedule.rate(Duration.minutes(5))
    }
};

new EventsRuleToLambda(this, 'test-events-rule-lambda', props);
```

Initializer

```
new EventsRuleToLambda(scope: Construct, id: string, props: EventsRuleToLambdaProps);
```

파라미터

- scopeConstruct
- idstring
- propsEventsRuleToLambdaProps

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 오류가 발생합니다.
람다기능소프	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 제공될 예정입니다.
이벤트루프로프	events.RuleProps	사용자가 제공 한 EventRuleProps 기본값을 무시합니다.

패턴 속성

이름	유형	설명
이벤트 규칙	`events.Rule`	패턴에 의해 생성된 Events 규칙의 인스턴스를 반환합니다.
Lambda함수	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

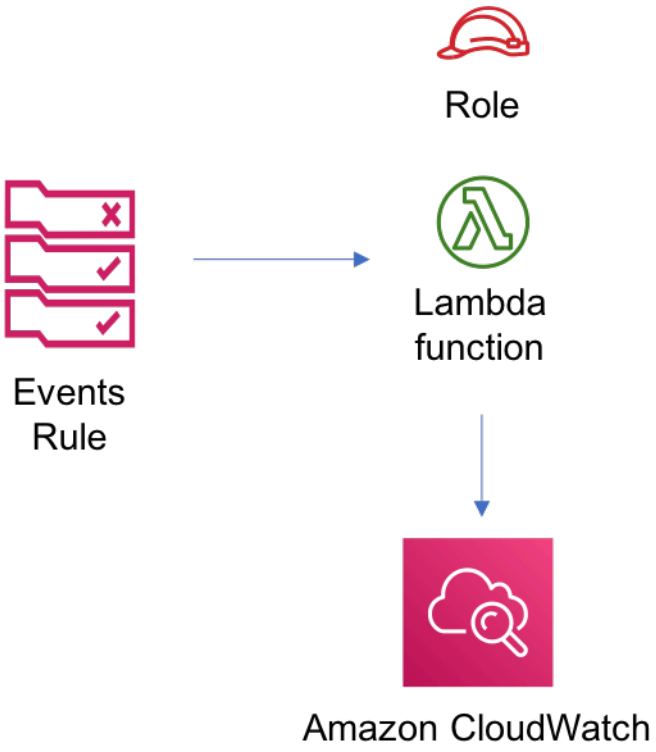
Amazon CloudWatch Events 규칙

- CloudWatch 이벤트에 Lambda 함수를 트리거하는 최소 권한 부여

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성하려면,
- NodeJS Lambda 함수에 대한 연결 유지로 연결 재사용 사용
- X-Ray 추적 설정
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - 이벤트 - 규칙 - 람다

aws-이벤트-규칙-sns

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_events_rule_sns`
타이프 스크립트	`@aws-solutions-constructs/aws-events-rule-sns`
Java	`software.amazon.awsconstructs.services.eventsrulesns`

Overview

이 패턴은 Amazon CloudWatch Events 규칙을 Amazon SNS 주제에 연결합니다.

다음은 최소한의 배포 가능한 패턴 정의입니다.

```
import { Duration } from '@aws-cdk/core';
import * as events from '@aws-cdk/aws-events';
import * as iam from '@aws-cdk/aws-iam';
import { EventsRuleToSnsProps, EventsRuleToSns } from "@aws-solutions-constructs/aws-events-rule-sns";

const props: EventsRuleToSnsProps = {
    eventRuleProps: {
      schedule: events.Schedule.rate(Duration.minutes(5)),
    }
};

const constructStack = new EventsRuleToSns(this, 'test-construct', props);

// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement({
    actions: ["kms:Encrypt", "kms:Decrypt"],
    effect: iam.Effect.ALLOW,
    principals: [ new iam.AccountRootPrincipal() ],
    resources: [ "*" ]
```

```
});

constructStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

Initializer

```
new EventsRuleToSNS(scope: Construct, id: string, props: EventsRuleToSNSProps);
```

파라미터

- scopeConstruct
- idstring
- propsEventsRuleToSnsProps

패턴 구성

이름	유형	설명
이벤트루프로프	events.RuleProps	CloudWatch 이벤트 규칙의 기본 속성을 재정의하는 사용자 제공 속성입니다.
기존토픽코비?	sns.Topic	SNS Topic 객체의 기존 인스턴스로,topicProps 오류가 발생합니다.
주제소품?	sns.TopicProps	SNS 주제에 대한 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingTopicObj 제공될 예정입니다.
고객 관리 키를 사용하여 암호화를 활성화하시겠습니까?	boolean	이 CDK 앱에서 관리하거나 가져온 고객 관리 암호화 키를 사용할지 여부 암호화 키

이름	유형	설명
		를 가져오는 경우 암호화 키를encryptionKey 이 구문에 대한 속성입니다.
encryptionKey	kms.Key	기본 암호화 키 대신 사용할 선택적 기존 암호화 키입니다.
암호화 키 소품?	kms.KeyProps	암호화 키의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.

패턴 속성

이름	유형	설명
이벤트 규칙	events.Rule	패턴에 의해 생성된 Events 규칙의 인스턴스를 반환합니다.
snsTopic	sns.Topic	패턴에 의해 생성된 SNS 주제의 인스턴스를 반환합니다.
encryptionKey	kms.Key	패턴에 의해 생성 된 암호화 키의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon CloudWatch Events

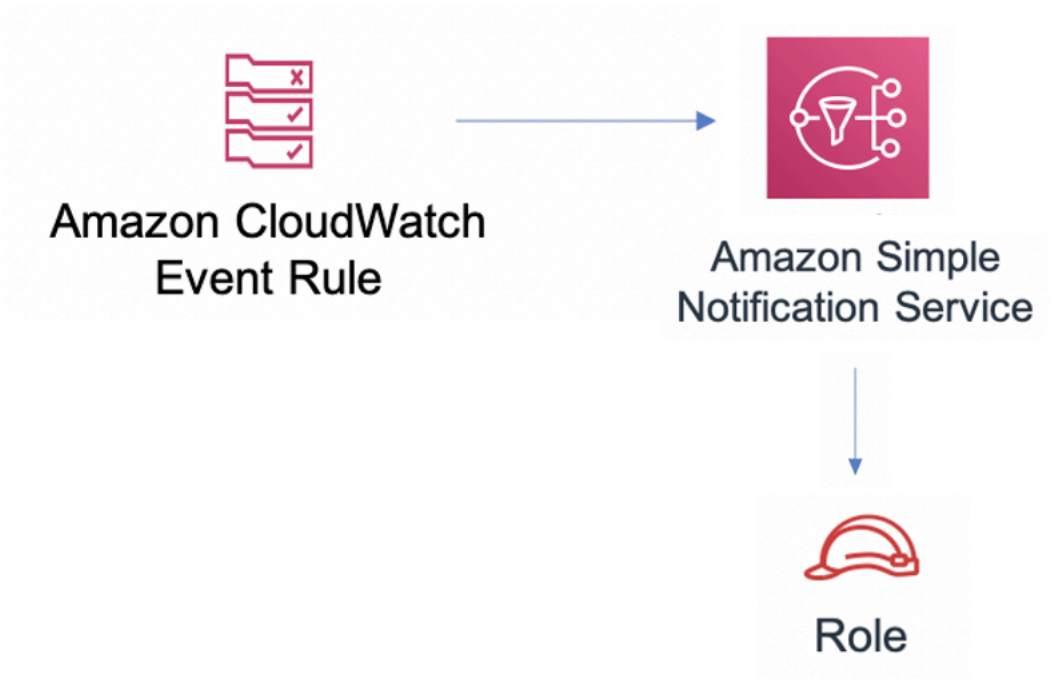
- CloudWatch 이벤트에 SNS 주제에 게시할 최소 권한을 부여합니다.

Amazon SNS 주제

- SNS 주제에 대한 최소 권한 액세스 권한을 구성합니다.

- 고객 관리형 AWS KMS 키를 사용하여 SNS 주제에 대해 서버 측 암호화를 활성화합니다.
- 전송 중인 데이터의 암호화를 강제 시행

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

@aws -솔루션 - 구성/AWS - 이벤트 - 규칙 - sns

AWS-이벤트-규칙-sqs

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 구성합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_events_rule_sqs`
타이프 스크립트	`@aws-solutions-constructs/aws-events-rule-sqs`
Java	`software.amazon.awsconstructs.services.eventsrulesqs`

Overview

이 패턴은 Amazon SQS 대기열에 연결된 Amazon CloudWatch Events 규칙을 구현합니다.

다음은 최소한의 배포 가능한 패턴 정의입니다.

```
import { Duration } from '@aws-cdk/core';
import * as events from '@aws-cdk/aws-events';
import * as iam from '@aws-cdk/aws-iam';
import { EventsRuleToSqsProps, EventsRuleToSqs } from "@aws-solutions-constructs/aws-events-rule-sqs";

const props: EventsRuleToSqsProps = {
    eventRuleProps: {
      schedule: events.Schedule.rate(Duration.minutes(5))
    }
};

const constructStack = new EventsRuleToSqs(this, 'test-construct', props);
```

```
// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement({
    actions: ["kms:Encrypt", "kms:Decrypt"],
    effect: iam.Effect.ALLOW,
    principals: [ new iam.AccountRootPrincipal() ],
    resources: [ "*" ]
});

constructStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

Initializer

```
new EventsRuleToSqs(scope: Construct, id: string, props: EventsRuleToSqsProps);
```

파라미터

- scopeConstruct
- idstring
- propsEventsRuleToSqsProps

패턴 구성

이름	유형	설명
이벤트루프로프	events.RuleProps	CloudWatch 이벤트 규칙의 기본 속성을 재정의하는 사용자 제공 속성입니다.
대기열에 있는 Obj?	sqs.Queue	기본 대기열 대신 사용할 기존 SQS 대기열 (선택 사항) 이 두 가지를 모두 제공queueProps 오류가 발생합니다.
대기열 소품?	sqs.QueueProps	SQS 대기열의 기본 속성을 재정의하는 선택적 사용자 제

이름	유형	설명
		공 속성입니다. 이 인 경우에는 무시됩니다.`existingQueueObj` 이 제공될 예정입니다.
대기열 제거 기능을 활성화하시겠습니까?	`boolean`	SQS 대기열을 비울 수 있도록 Lambda 함수에 추가 권한을 부여할지 여부 기본값은 `false`입니다.
배포데드 레터 큐?	`boolean`	배달 못한 편지 대기열로 사용되는 보조 대기열을 생성할지 여부를 지정합니다. 기본값은 `true`입니다.
데드레터 큐프로프?	`sqs.QueueProps`	데드 레터 큐의 기본 소품을 재정의하는 선택적 사용자가 제공 한 소품. 에만 사용됩니다.`deployDeadLetterQueue` 속성이 true로 설정됩니다.
maxReceiveCount?	`number`	배달 못한 편지 대기열로 이동되기 전에 메시지가 대기열에서 빼지 못한 횟수입니다. 기본값은 `15`입니다.
고객 관리 키를 사용하여 암호화를 활성화하시겠습니까?	`boolean`	이 CDK 앱에서 관리하거나 가져온 고객 관리 암호화 키를 사용할지 여부 암호화 키를 가져오는 경우 암호화 키를`encryptionKey` 이 구문에 대한 속성입니다.
encryptionKey	`kms.Key`	기본 암호화 키 대신 사용할 선택적 기존 암호화 키입니다.

이름	유형	설명
암호화 키 소품?	`kms.KeyProps`	암호화 키의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.

패턴 속성

이름	유형	설명
이벤트 규칙	`events.Rule`	패턴에 의해 생성된 Events 규칙의 인스턴스를 반환합니다.
분류: SQqueue	`sqs.Queue`	패턴에 의해 생성된 SQS 대기열의 인스턴스를 반환합니다.
encryptionKey	`kms.Key`	패턴에 의해 생성 된 암호화 키의 인스턴스를 돌려줍니다.
데드 레터 큐?	`sqs.Queue`	패턴에 의해 생성 된 데드 레터 큐의 인스턴스를 돌려줍니다 (배포 된 경우).

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

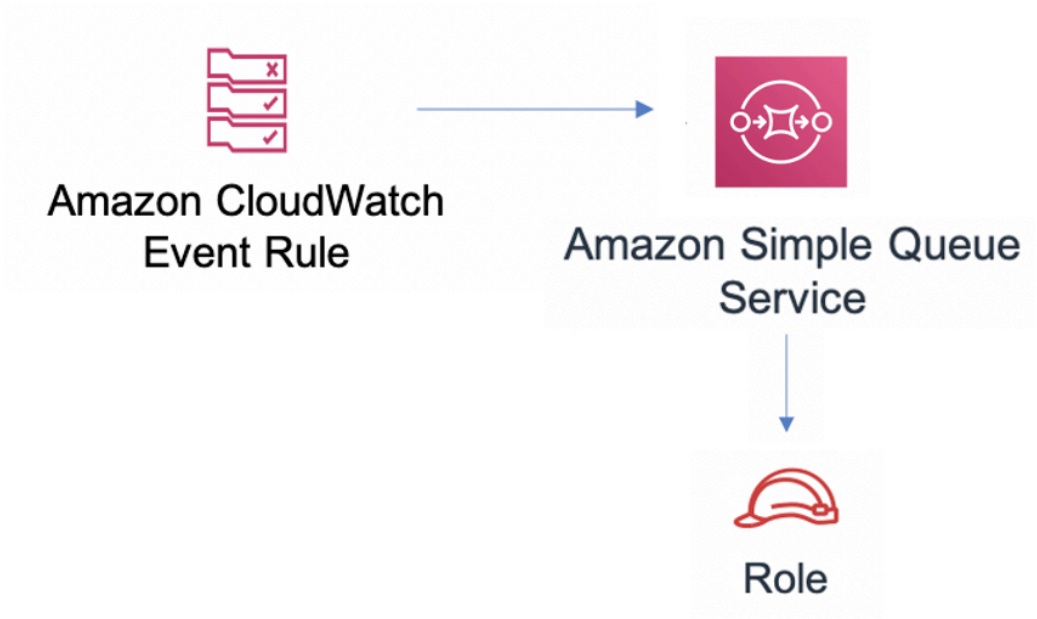
Amazon CloudWatch Events

- CloudWatch 이벤트에 최소 권한을 부여하여 SQS 대기열에 게시할 수 있습니다.

Amazon SQS 대기열

- 소스 대기열에 대한 배달 못한 편지 대기열을 배포합니다.
- 고객 관리형 AWS KMS 키를 사용하여 소스 대기열에 대한 서버 측 암호화를 활성화합니다.
- 전송 중인 데이터의 암호화를 강제 시행.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

@aws -솔루션 - 구성/AWS - 이벤트 - 규칙 - sqs

AWS-이벤트-규칙-스텝 기능

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 반환합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_events_rule_step_function`
타이프 스크립트	`@aws-solutions-constructs/aws-events-rule-step-function`
Java	`software.amazon.awsconstructs.services.eventsrulestepfunction`

Overview

이 AWS 솔루션 구성은 AWS 이벤트 규칙과 AWS 스텝 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { EventsRuleToStepFunction, EventsRuleToStepFunctionProps } from '@aws-solutions-constructs/aws-events-rule-step-function';

const startState = new stepfunctions.Pass(this, 'StartState');

const props: EventsRuleToStepFunctionProps = {
    stateMachineProps: {
      definition: startState
    },
    eventRuleProps: {
      schedule: events.Schedule.rate(Duration.minutes(5))
    }
};

new EventsRuleToStepFunction(this, 'test-events-rule-step-function-stack', props);
```

Initializer

```
new EventsRuleToStepFunction(scope: Construct, id: string, props:
 EventsRuleToStepFunctionProps);
```

파라미터

- scopeConstruct
- idstring
- propsEventsRuleToStepFunctionProps

패턴 구성

이름	유형	설명
상태 머신소품	sfn.StateMachineProps	선택적 사용자가 SFN.State Machine의 기본 소품을 무시하기 위해 소품을 제공했습니다.
이벤트루프로프	events.RuleProps	사용자가 제공 한 EventRuleProps 기본값을 무시합니다.
클라우드왓찰암스 만들기	boolean	권장 CloudWatch 경보를 생성할지 여부입니다.
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품입니다.

패턴 속성

이름	유형	설명
CloudwatchAms?	`cloudwatch.Alarm[]`	패턴에 의해 생성된 하나 이상의 CloudWatch 경보 목록을 반환합니다.
이벤트 규칙	`events.Rule`	패턴에 의해 생성된 Events 규칙의 인스턴스를 반환합니다.
스테이트머신	`sfn.StateMachine`	패턴에 의해 생성 된 상태 머신의 인스턴스를 돌려줍니다.
상태시스템로그 그룹	`logs.LogGroup`	상태 시스템의 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

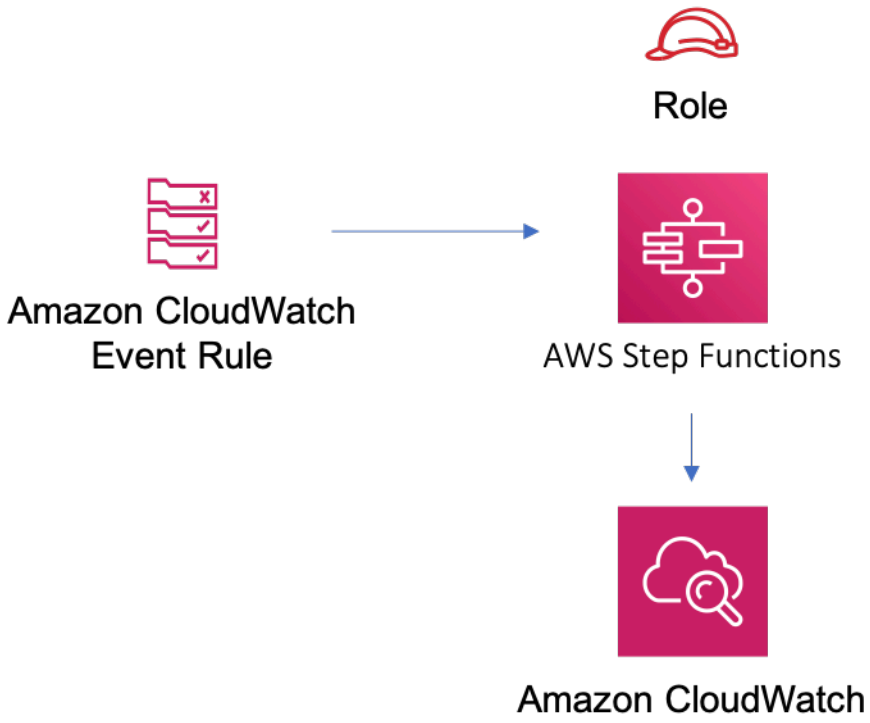
Amazon CloudWatch Events

- CloudWatch 이벤트에 Lambda 함수를 트리거하는 최소 권한 부여

AWS Step Fun

- API Gateway 에 대한 CloudWatch 로깅
- 단계 기능에 대한 모범 사례 CloudWatch 경보 배포

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - 이벤트 - 규칙 - 단계 함수

AWS-이오트키네시스파이어호스-3

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_iot_kinesisfirehose_s3`
타이프 스크립트	`@aws-solutions-constructs/aws-iot-kinesisfirehose-s3`
Java	`software.amazon.awsconstructs.services.iotkinesisfirehoses3`

Overview

이 AWS 솔루션 구성체는 AWS IoT MQTT 주제 규칙을 구현하여 Amazon S3 버킷에 연결된 Amazon Kinesis Data Firehose 전송 스트림에 데이터를 전송합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { IotToKinesisFirehoseToS3Props, IotToKinesisFirehoseToS3 } from '@aws-solutions-constructs/aws-iot-kinesisfirehose-s3';

const props: IotToKinesisFirehoseToS3Props = {
    iotTopicRuleProps: {
        topicRulePayload: {
            ruleDisabled: false,
            description: "Persistent storage of connected vehicle telematics data",
            sql: "SELECT * FROM 'connectedcar/telemetry/#'",
            actions: []
        }
    }
};

new IotToKinesisFirehoseToS3(this, 'test-iot-firehose-s3', props);
```

Initializer

```
new IotToKinesisFirehoseToS3(scope: Construct, id: string, props:
 IotToKinesisFirehoseToS3Props);
```

파라미터

- scopeConstruct
- idstring
- propsIotToKinesisFirehoseToS3Props

패턴 구성

이름	유형	설명
이오토피크루프롭스	iot.CfnTopicRuleProps	사용자가 기본값을 재정의하기 위해 CFNTopyCrulePropsㄹ를 제공했습니다.
키네시스파이어호스프로프스?	kinesisfirehose.CfnDeliveryStreamProps	Kinesis 파이어호스 배달 스트림의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품
버켓토비 기존에?	s3.Bucket	S3 버킷 객체의 기존 인스턴스로,bucketProps 오류가 발생합니다.
버킷 소품?	s3.BucketProps	사용자가 S3 버킷의 기본 소품을 재정의하는 소품을 제공했습니다. 이것이 제공되는 경우bucketProps 는 오류입니다.
로그그룹Props?	logs.LogGroupProps	선택적 사용자 제공 소품을 사용하여 CloudWatch Logs 로그

이름	유형	설명
		그룹의 기본 소품을 재정의합니다.

패턴 속성

이름	유형	설명
IOT액션역할	`iam.Role`	IoT 규칙의 패턴에 의해 생성된 IAM 역할의 인스턴스를 반환합니다.
아이오토항목 규칙	`iot.CfnTopicRule`	패턴에 의해 생성된 IoT 주제 규칙의 인스턴스를 반환합니다.
키네시스파이어호스	`kinesisfirehose.CfnDeliveryStream`	패턴에 의해 생성된 Kinesis 파이어호스 전달 스트림의 인스턴스를 반환합니다.
키네시스파이어호스셀로그그룹	`logs.LogGroup`	Kinesis Firehose 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.
키네시스파이어호스롤	`iam.Role`	Kinesis Firehose 전송 스트림의 패턴에 의해 생성된 IAM 역할의 인스턴스를 반환합니다.
S3Bucket?	`s3.Bucket`	패턴에 의해 생성된 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	`s3.Bucket`	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon IoT 규칙

- Amazon IoT 에 대한 최소 권한 액세스 IAM 역할 구성

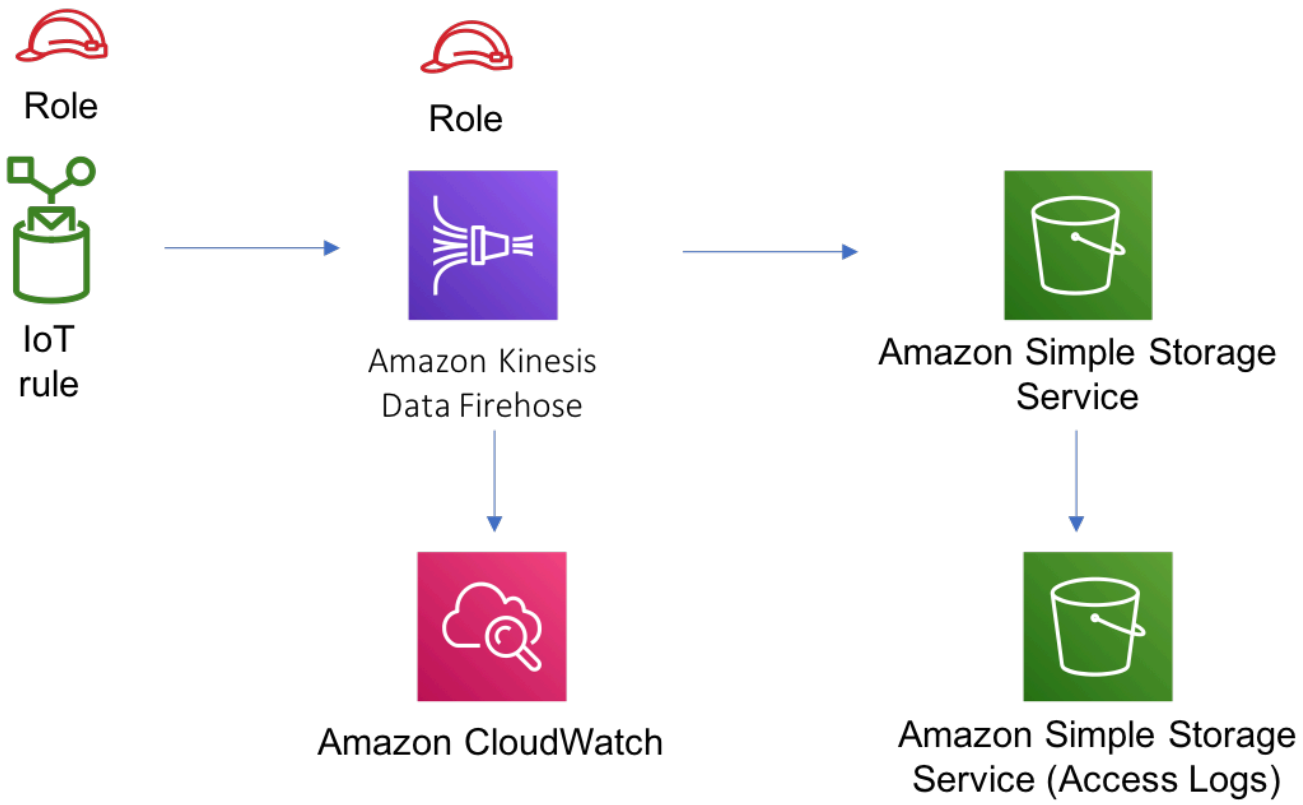
Amazon Kinesis Firehose

- Kinesis 파이어호스에 대한 CloudWatch 로깅 활성화
- Amazon Kinesis Firehose 대한 최소 권한 액세스 IAM 역할 구성

Amazon S3 버킷

- S3 버킷에 대한 액세스 로깅 구성
- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화 활성화
- S3 버킷에 대한 버전 관리 켜기
- S3 버킷에 대한 공용 액세스 허용 안 함
- CloudFormation 스택을 삭제할 때 S3 버킷 유지
- 90일 후에 비최신 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS-Iot-키네시스파이어호스-s3

aws-iot-람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_iot_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-iot-lambda`
Java	`software.amazon.awsconstructs.services.iotlambda`

Overview

이 AWS 솔루션은 패턴을 구축하여 AWS IoT MQTT 주제 규칙과 AWS Lambda 함수 패턴을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { IotToLambdaProps, IotToLambda } from '@aws-solutions-constructs/aws-iot-
lambda';

const props: IotToLambdaProps = {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    },
    iotTopicRuleProps: {
        topicRulePayload: {
            ruleDisabled: false,
            description: "Processing of DTC messages from the AWS Connected Vehicle
 Solution.",
            sql: "SELECT * FROM 'connectedcar/dtc/#'",
            actions: []
        }
```

```
    }
};

new IotToLambda(this, 'test-iot-lambda-integration', props);
```

Initializer

```
new IotToLambda(scope: Construct, id: string, props: IotToLambdaProps);
```

파라미터

- scope`Construct`
- id`string`
- props`IotToLambdaProps`

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	`lambda.Function`	Lambda 함수 객체의 기존 인스턴스, 이 및`lambdaFunctionProps` 오류가 발생합니다.
람다기능소품?	`lambda.FunctionProps`	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.`existingLambdaObj` 제공될 예정입니다.
이오토피크루프롭스?	`iot.CfnTopicRuleProps`	사용자가 기본값을 재정의하기 위해 CFNTopyCruleProps를 제공했습니다.

패턴 속성

이름	유형	설명
아이오토항목 규칙	`iot.CfnTopicRule`	패턴에 의해 생성된 IoT 주제 규칙의 인스턴스를 반환합니다.
람다함수	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

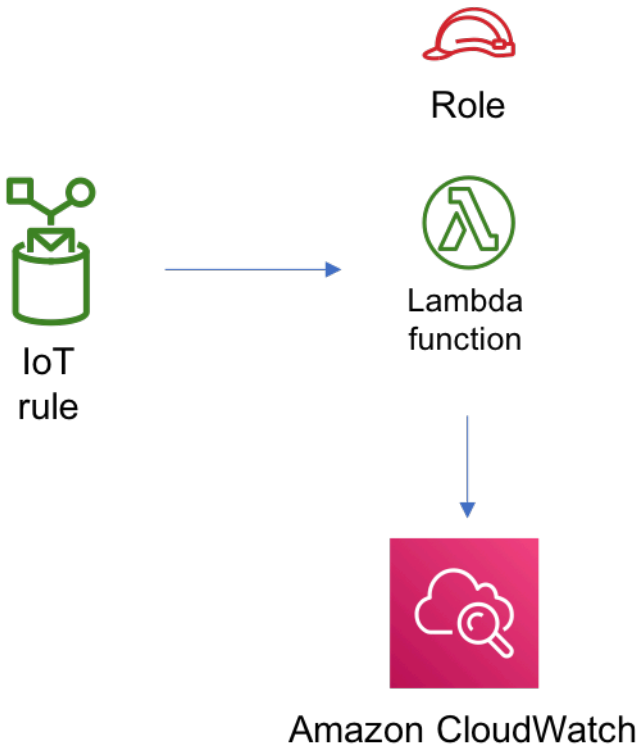
아마존 IoT 규칙

- Amazon IoT 에 대한 최소 권한 액세스 IAM 역할을 구성합니다.

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 사용합니다.
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - Iot-람다

AWS-Iot-람다-다이나모DB

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 참조하십시오. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_iot_lambda_dynamodb`
타이프 스크립트	`@aws-solutions-constructs/aws-iot-lambda-dynamodb`
Java	`software.amazon.awsconstructs.services.iotlambdadynamodb`

Overview

이 AWS 솔루션은 최소 권한이 있는 AWS IoT 주제 규칙, AWS Lambda 함수 및 Amazon DynamoDB 테이블을 구현하는 패턴을 구성합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { IotToLambdaToDynamoDBProps,  IotToLambdaToDynamoDB } from '@aws-solutions-constructs/aws-iot-lambda-dynamodb';

const props: IotToLambdaToDynamoDBProps = {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    },
    iotTopicRuleProps: {
        topicRulePayload: {
            ruleDisabled: false,
            description: "Processing of DTC messages from the AWS Connected Vehicle Solution.",
            sql: "SELECT * FROM 'connectedcar/dtc/#'",
            actions: []
        }
```

```
    }
};

new IotToLambdaToDynamoDB(this, 'test-iot-lambda-dynamodb-stack', props);
```

Initializer

```
new IotToLambdaToDynamoDB(scope: Construct, id: string, props:
 IotToLambdaToDynamoDBProps);
```

파라미터

- scopeConstruct
- idstring
- propsIotToLambdaToDynamoDBProps

패턴 구성

이름	유형	설명
이미 람다오브?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 사용하면 오류가 발생할 수 있습니다.
람다기능소프	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우 무시됩니다.existingLambdaObj 제공될 예정입니다.
이오토피크루프롭스	iot.CfnTopicRuleProps	기본 소품을 재정의하는 소품을 제공함

이름	유형	설명
다이나모테이블소품?	dynamodb.TableProps	DynamoDB 테이블의 기본 소품을 무시하기 위한 선택적 사용자가 제공한 소품
테이블 사용 권한?	string	Lambda 함수에 부여할 선택적 테이블 권한입니다. 다음 옵션 중 하나를 지정할 수 있습니다.`All`,`Read`,`ReadWrite` 또는`Write`.

패턴 속성

이름	유형	설명
다이나모터블	dynamodb.Table	패턴에 의해 생성된 DynamoDB 테이블의 인스턴스를 반환합니다.
아이오토항목 규칙	iot.CfnTopicRule	패턴에 의해 생성된 IoT 주제 규칙의 인스턴스를 반환합니다.
람다함수	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon IoT 규칙

- Amazon IoT 에 대한 최소 권한 액세스 IAM 역할을 구성합니다.

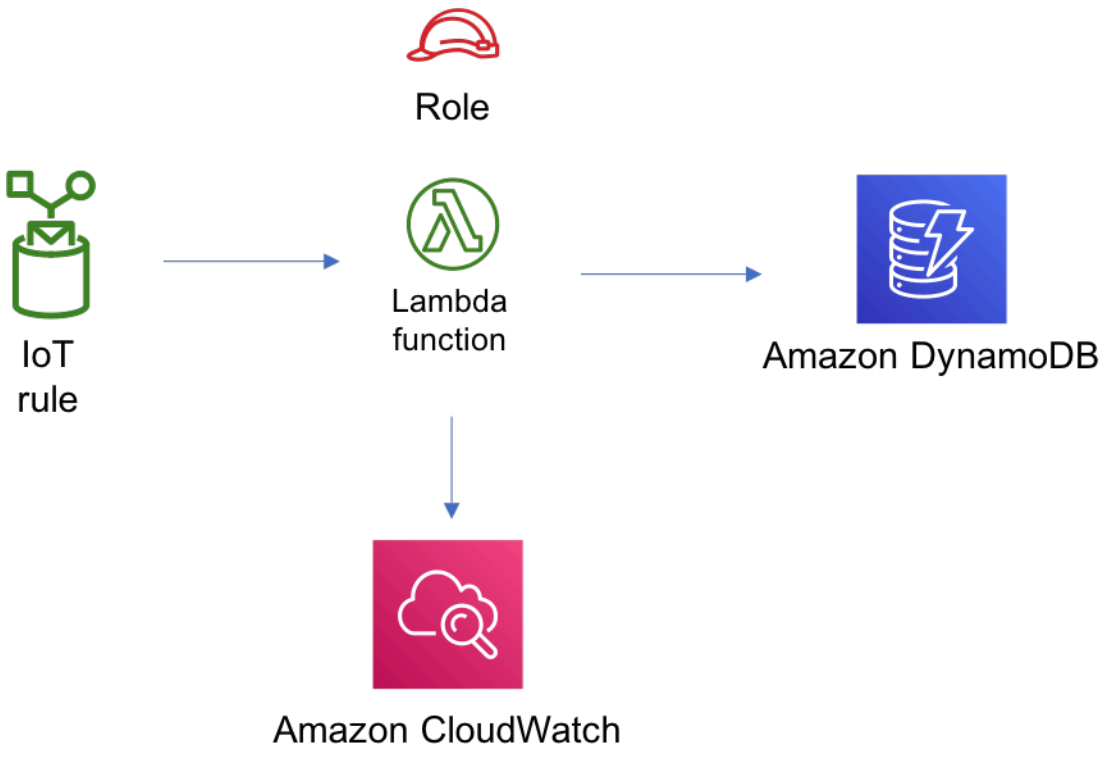
AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon DynamoDB 테이블

- DynamoDB 테이블의 결제 모드를 온 디맨드 (요청당 지불) 로 설정합니다.
- AWS 관리형 KMS 키를 사용하여 DynamoDB 테이블에 대한 서버 측 암호화를 활성화합니다.
- DynamoDB 테이블에 대해 'id'라는 파티션 키를 생성합니다.
- CloudFormation 스택을 삭제할 때 테이블을 유지합니다.
- 지속적인 백업 및 지정 시간 복구를 지원합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/AWS-Iot-람다 - 다이나모 DB

AWS-키네시스파이어호스-3

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws-kinesis-firehose-s3`
타이프 스크립트	`@aws-solutions-constructs/aws-kinesisfirehose-s3`
Java	`software.amazon.awsconstructs.services.kinesisfirehoses3`

Overview

AWS 솔루션 구성체는 Amazon S3 버킷에 연결된 Amazon Kinesis Data Firehose 전송 스트림을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { KinesisFirehoseToS3 } from '@aws-solutions-constructs/aws-kinesisfirehose-s3';

new KinesisFirehoseToS3(this, 'test-firehose-s3', {});
```

Initializer

```
new KinesisFirehoseToS3(scope: Construct, id: string, props: KinesisFirehoseToS3Props);
```

파라미터

- scope`Construct`
- id`string`
- props`KinesisFirehoseToS3Props`

Props 패턴 구성

이름	유형	설명
버킷 소품?	`s3.BucketProps`	선택적 사용자가 S3 버킷의 기본 소품을 재정의하는 소품을 제공했습니다.
버켓토비 기존에?	`s3.IBucket`	S3 버킷의 선택적 기존 인스턴스. 이것이 제공되는 경우`bucketProps` 는 오류입니다.
기존로깅버켓트BJ?	`s3.IBucket`	패턴에 의해 생성된 S3 버킷에 대한 S3 버킷 로깅 기존 인스턴스 (선택 사항)
키네시스파이어호스프로프스?	`kinesisfirehose.CfnDeliveryStreamProps` | any	Kinesis 파이어호스 배달 스트림의 기본 소품을 무시하기 위

이름	유형	설명
		한 선택적 사용자가 제공한 소품입니다.
로그그룹Props?	logs.LogGroupProps	선택적 사용자가 CloudWatchLogs 로그 그룹에 대한 기본 소품을 재정의하는 소품을 제공했습니다.

패턴 속성

이름	유형	설명
키네시스파이어호스	kinesisfirehose.CfnDeliveryStream	구문에서 만든 키네시스파이어호스.cfn배달 스트림의 인스턴스를 반환합니다.
키네시스파이어호스셀로그그룹	logs.LogGroup	Kinesis Data Firehose 전달 스트림에 대한 구문에서 생성된 Logs.logGroup의 인스턴스를 반환합니다.
키네시스파이어호스롤	iam.Role	Kinesis Data Firehose 전송 스트림용 구문에서 생성한 IAM.role 인스턴스를 반환합니다.
S3Bucket?	s3.Bucket	구조에 의해 생성 된 S3.bucket의 인스턴스를 돌려줍니다.
s3로깅 버킷?	s3.Bucket	구조에 의해 생성된 S3.bucket의 인스턴스를 기본 버킷의 로깅 버킷으로 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

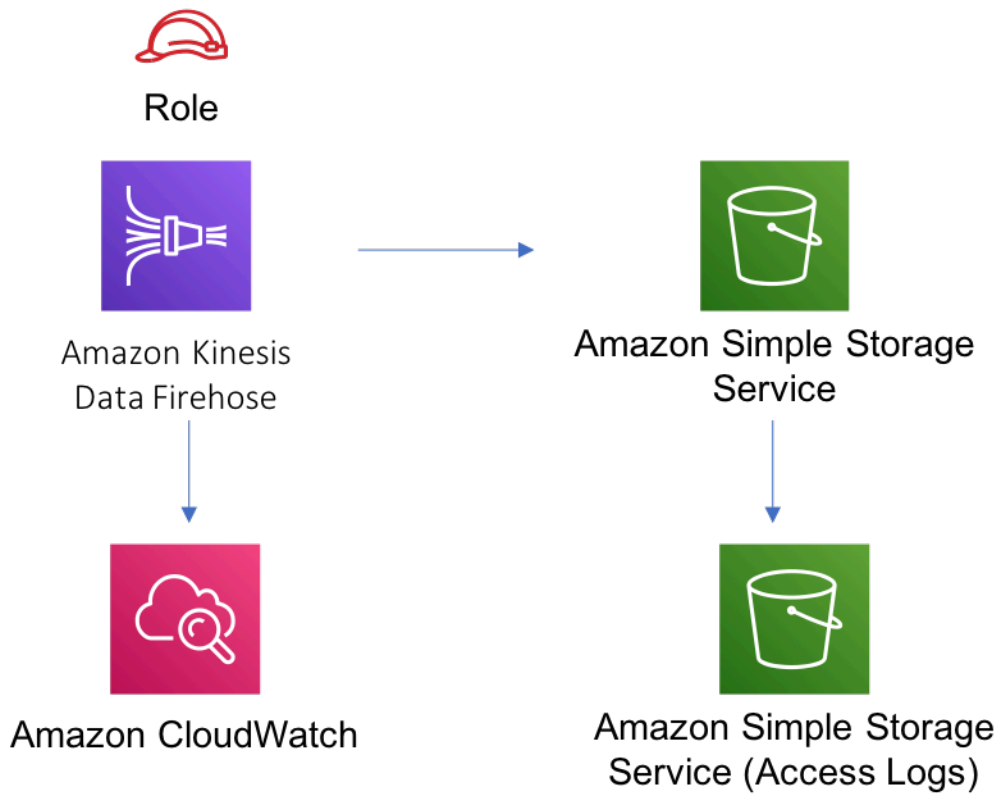
Amazon Kinesis Firehose

- Kinesis 파이어호스에 대한 CloudWatch 로깅 활성화
- Amazon Kinesis Firehose 대한 최소 권한 액세스 IAM 역할 구성

Amazon S3 버킷

- S3 버킷에 대한 액세스 로깅 구성
- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화 활성화
- S3 버킷에 대한 버전 관리 켜기
- S3 버킷에 대한 공용 액세스 허용 안 함
- CloudFormation 스택을 삭제할 때 S3 버킷 유지
- 전송 중인 데이터의 암호화 강제 시행
- 90일 후에 비최신 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/AWS-키네시스파이어호스-s3

AWS 키네시스파이어호스-S3 및 키네시스분석

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_kinesisfirehose_s3_and_kinesisanalytics`
타이프 스크립트	`@aws-solutions-constructs/aws-kinesisfirehose-s3-and-kinesisanalytics`
Java	`software.amazon.awsconstructs.services.kinesisfirehose.s3kinesisanalytics`

Overview

이 AWS 솔루션 구성체는 Amazon S3 버킷과 Amazon Kinesis 애널리틱스 애플리케이션에 연결된 Amazon Kinesis 파이어호스 전송 스트림을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { KinesisFirehoseToAnalyticsAndS3 } from '@aws-solutions-constructs/aws-
kinesisfirehose-s3-and-kinesisanalytics';

new KinesisFirehoseToAnalyticsAndS3(this, 'FirehoseToS3AndAnalyticsPattern', {
    kinesisAnalyticsProps: {
        inputs: [{
            inputSchema: {
                recordColumns: [{
                    name: 'ticker_symbol',
                    sqlType: 'VARCHAR(4)',
                    mapping: '$.ticker_symbol'
                }, {
                    name: 'sector',
                    sqlType: 'VARCHAR(16)',
                    mapping: '$.sector'
                }, {
                    name: 'change',
                    sqlType: 'REAL',
```

```
                    mapping: '$.change'
                }, {
                    name: 'price',
                    sqlType: 'REAL',
                    mapping: '$.price'
                }],
                recordFormat: {
                    recordFormatType: 'JSON'
                },
                recordEncoding: 'UTF-8'
            },
            namePrefix: 'SOURCE_SQL_STREAM'
        }]
    }
});
```

Initializer

```
new KinesisFirehoseToAnalyticsAndS3(scope: Construct, id: string, props:
 KinesisFirehoseToAnalyticsAndS3Props);
```

파라미터

- scope`Construct`
- id`string`
- props`KinesisFirehoseToAnalyticsAndS3Props`

패턴 구성

이름	유형	설명
키네시스파이어호스프로프스?	`kinesisFirehose.CfnDeliveryStreamProps`	Kinesis Firehose 전달 스트림의 기본 소품을 무시하기 위한 사용자 제공 소품 (선택 사항)
키네해석스프로프?	`kinesisAnalytics.CfnApplicationProps`	Kinesis Analytics 응용 프로그램의 기본 소품을 재정의하는

이름	유형	설명
		선택적 사용자 제공 소품입니다.
버켓토비 기존에?	`s3.IBucket`	S3 버킷 객체의 기존 인스턴스입니다. 이것이 제공되는 경우`bucketProps` 는 오류입니다. |
버킷 소품?	`s3.BucketProps`	버킷의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.`existingBucketObj` 가 제공되는지 확인합니다.
로그그룹Props?	`logs.LogGroupProps`	CloudWatch Logs 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품.

패턴 속성

이름	유형	설명
키네시스분석	`kinesisAnalytics.CfnApplication`	패턴에 의해 생성된 Kinesis Analytics 애플리케이션의 인스턴스를 반환합니다.
키네시스파이어호스	`kinesisfirehose.CfnDeliveryStream`	패턴에 의해 생성된 Kinesis 파이어호스 전달 스트림의 인스턴스를 반환합니다.
키네시스파이어호스셀로그그룹	`logs.LogGroup`	Kinesis Firehose 액세스 로그가 전송되는 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.

이름	유형	설명
키네시스파이어호스롤	`iam.Role`	Kinesis Firehose 전송 스트림의 패턴에 의해 생성된 IAM 역할의 인스턴스를 반환합니다.
S3Bucket?	`s3.Bucket`	패턴에 의해 생성된 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	`s3.Bucket`	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon Kinesis Firehose

- Kinesis 파이어호스에 대한 CloudWatch 로깅 활성화
- Amazon Kinesis Firehose 대한 최소 권한 액세스 IAM 역할 구성

Amazon S3 버킷

- S3 버킷에 대한 액세스 로깅 구성
- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화 활성화
- S3 버킷의 버전 관리 켜기
- S3 버킷에 대한 공용 액세스 허용 안 함
- CloudFormation 스택을 삭제할 때 S3 버킷 유지
- 전송 중인 데이터의 암호화 강제 시행
- 90일 후에 비최신 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

Amazon Kinesis Data Analytics

- Amazon Kinesis Analytics 스에 대한 최소 권한 액세스 IAM 역할 구성

Architecture

GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구조/AWS-키네시스파이어호스-S3 및 키네시스분석

AWS-키네시스스트림-접착제 작업

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_kinesis_streams_gluejob`
타이프 스크립트	`@aws-solutions-constructs/aws-kinesisstreams-gluejob`
Java	`software.amazon.awsconstructs.services.kinesisstreamsgluejob`

Overview

이 AWS 솔루션 구성 요소는 Amazon Kinesis 데이터 스트림을 배포하고 상호 작용 및 보안을 위해 적절한 리소스/속성을 사용하여 사용자 지정 ETL 변환을 수행하도록 AWS Glue Job 구성합니다. 또한 AWS Glue Job 용 Python 스크립트를 업로드할 수 있는 Amazon S3 버킷을 생성합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import * as glue from '@aws-cdk/aws-glue';
import * as s3assets from '@aws-cdk/aws-s3-assets';
import { KinesisstreamsToGluejob } from '@aws-solutions-constructs/aws-kinesisstreams-
gluejob';

const fieldSchema: glue.CfnTable.ColumnProperty[] = [
    {
        name: 'id',
        type: 'int',
        comment: 'Identifier for the record',
    },
    {
        name: 'name',
        type: 'string',
        comment: 'Name for the record',
    },
    {
```

```
        name: 'address',
        type: 'string',
        comment: 'Address for the record',
    },
    {
        name: 'value',
        type: 'int',
        comment: 'Value for the record',
    },
];

const customEtlJob = new KinesisstreamsToGluejob(this, 'CustomETL', {
    glueJobProps: {
        command: {
            name: 'gluestreaming',
            pythonVersion: '3',
            scriptLocation: new s3assets.Asset(this, 'ScriptLocation', {
                path: `${__dirname}/../etl/transform.py`,
            }).s3ObjectUrl,
        },
    },
    fieldSchema: fieldSchema,
});
```

Initializer

```
new KinesisstreamsToGluejob(scope: Construct, id: string, props:
 KinesisstreamsToGluejobProps);
```

파라미터

- scope`Construct`
- id`string`
- props`KinesisstreamsToGluejobProps`

소품 패턴 구성

이름	유형	설명
키네시스스트림프롭스?	`kinesis.StreamProps`	Amazon Kinesis 데이터 스트림의 기본 소품을 무시하기 위한 사용자 제공 소품 (선택 사항)
기존스트리모브?	`kinesis.Stream`	Kinesis 스트림의 기존 인스턴스. 이 인스턴스와`kinesisStreamProps` 오류가 발생합니다.
글루조프롭스?	`cfnJob.CfnJobProps`	AWS Glub 작업에 대한 기본 소품을 재정의하는 사용자 제공 소품입니다.
기존 글루 작업?	`cfnJob.CfnJob`	AWS Glue le Job 의 기존 인스턴스로, 이 인스턴스와`glueJobProps` 오류가 발생합니다.
기존 데이터베이스?	`CfnDatabase`	이 구문과 함께 사용할 기존 AWS Glue b 데이터베이스 이것이 설정되면`databaseProps` 이 값은 무시됩니다.
데이터베이스Props?	`CfnDatabaseProps`	AWS Glub 데이터베이스를 생성하는 데 사용되는 기본 소품을 재정의하는 사용자 제공 소품입니다.
기존 테이블?	`CfnTable`	AWS Glue 테이블의 기존 인스턴스입니다. 이것이 설정되면`tableProps` 및`fieldSchema` 이 매개변수는 무시됩니다.

이름	유형	설명
테이블소품?	`CfnTableProps`	AWS Glub 테이블을 생성하는 데 사용되는 기본 소품을 재정의하는 사용자 제공 소품입니다.
필드스키마?	`CfnTable.ColumnProperty[]`	AWS Glue le 테이블을 생성하기 위한 사용자 제공 스키마 구조
데이터 저장소 출력	`SinkDataStoreProps`	AWS Glue b 작업의 출력을 저장하는 Amazon S3 버킷에 대한 사용자가 제공한 소품입니다. 현재 Amazon S3 만 출력 데이터스토어 유형으로 지원합니다.

SinkDataStoreProps

이름	유형	설명
기존3출력 버킷?	`Bucket`	데이터를 기록해야하는 S3 버킷의 기존 인스턴스입니다. 이 두 가지를 모두 제공`outputBucketProps` 오류가 발생합니다.
출력버킷소품	`BucketProps`	AWS Glue e의 출력을 저장하는 데 사용되는 Amazon S3 버킷을 만들 수 있는 사용자 제공 버킷 속성입니다.
데이터스토어 유형	`SinkStoreType`	싱크 데이터 저장소 유형입니다.

SinkStoreType

S3, DynamoDB, DocumentDB, RDS 또는 Redshift 포함할 수 있는 데이터 저장소 유형의 열거입니다. 현재 구문 구현은 S3 만 지원하지만 나중에 다른 출력 유형을 추가 할 수 있습니다.

이름	유형	설명
S3	`string`	S3 스토리지 유형

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon Kinesis Streams

- Amazon Kinesis 데이터 스트림에 대한 최소 권한 액세스 IAM 역할을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 Amazon Kinesis 스트림에 대한 서버 측 암호화를 활성화합니다.
- 아마존 Kinesis 스트림에 대한 모범 사례 아마존 CloudWatch 경보를 배포합니다.

GGlue y Job

- CloudWatch, Job 북마크 및 S3에 대한 암호화를 구성하는 AWS Glue 보안 구성을 생성합니다. CloudWatch 및 Job 북마크는 AWS AWS Glue 서비스용으로 생성된 AWS 관리형 KMS 키를 사용하여 암호화됩니다. S3 버킷은 SSE-S3 암호화 모드로 구성됩니다.
- Amazon Kinesis Data Streams es에서 AWS Glue e가 읽을 수 있도록 허용하는 서비스 역할 정책을 구성합니다.

GGlue e데이터베이스

- AWS Glue 데이터베이스를 생성합니다. AWS Glue 테이블이 데이터베이스에 추가됩니다. 이 표에서는 Amazon Kinesis 데이터 스트림에서 버퍼링된 레코드에 대한 스키마를 정의합니다.

GGlue e테이블

- AWS Glue 테이블을 만듭니다. 테이블 스키마 정의는 Amazon Kinesis 데이터 스트림에서 버퍼링된 레코드의 JSON 구조를 기반으로 합니다.

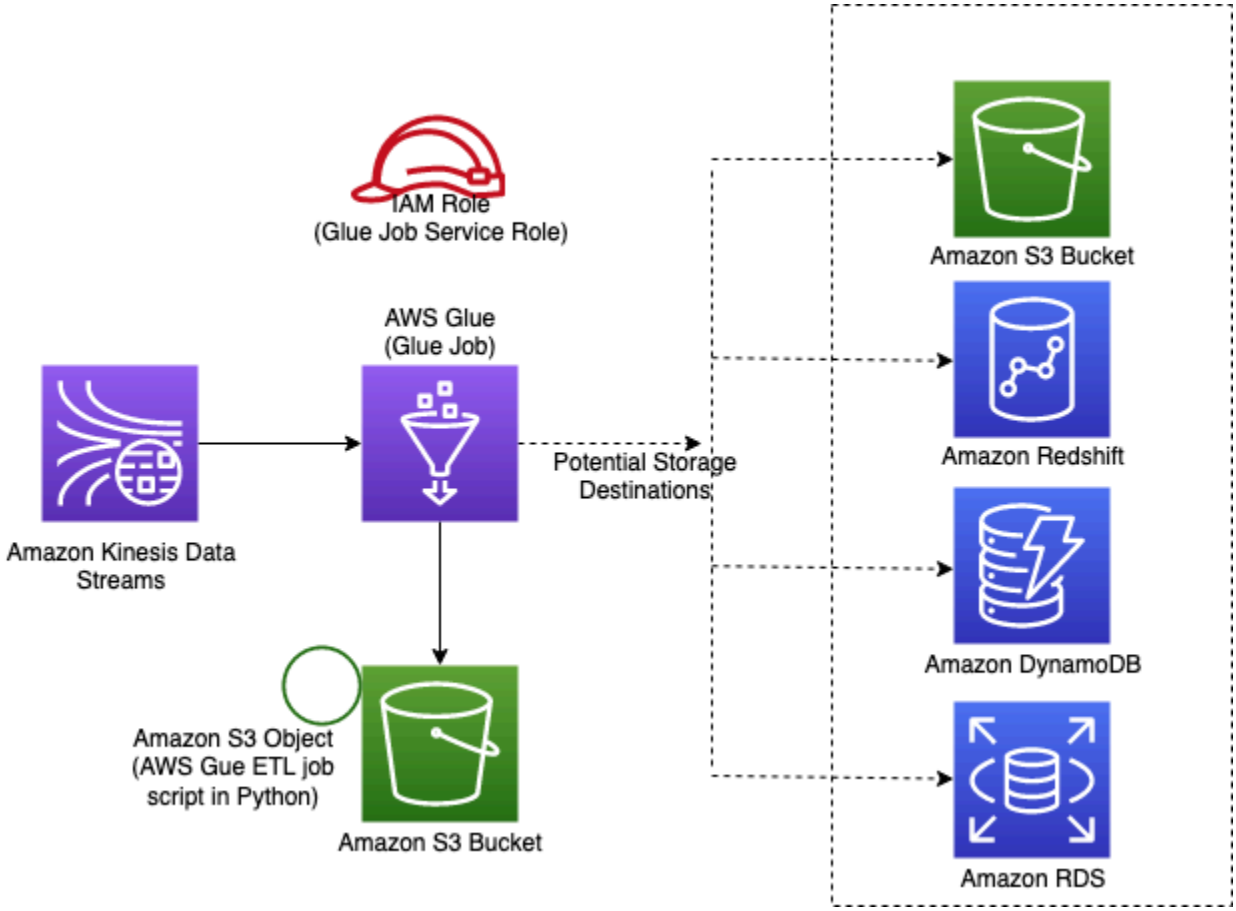
IAM 역할

- 1) Amazon S3 버킷 위치에서 ETL 스크립트를 읽고, 2) Amazon Kinesis 데이터 스트림에서 레코드를 읽고, 3) Amazon Glue 루 작업을 실행할 수 있는 권한을 가진 작업 실행 역할입니다.

Output S3 Bucket

- ETL 변환의 출력을 저장할 Amazon S3 버킷입니다. 이 버킷은 생성된 AWS Glue b 작업에 인수로 전달되므로 ETL 스크립트에서 데이터를 쓸 수 있습니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/AWS - 키네시스스트림-접착제 작업

AWS-키네시스스트림-키네시스파이어호스-3

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 참조하십시오. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_kinesisstreams_kinesisfirehose_s3`
타이프 스크립트	`@aws-solutions-constructs/aws-kinesis-streams-kinesis-firehose-s3`
Java	`software.amazon.awsconstructs.services.kinesisstreamskinesisfirehoses3`

Overview

이 AWS 솔루션 구조는 Amazon S3 버킷에 연결된 Amazon Kinesis 데이터 파이어호스 (KDF) 전송 스트림에 연결된 Amazon Kinesis 데이터 스트림 (KDS) 을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { KinesisStreamsToKinesisFirehoseToS3 } from '@aws-solutions-constructs/aws-kinesisstreams-kinesisfirehose-s3';

new KinesisStreamsToKinesisFirehoseToS3(this, 'test-stream-firehose-s3', {});
```

Initializer

```
new KinesisStreamsToKinesisFirehoseToS3(scope: Construct, id: string, props:
 KinesisStreams...ToS3Props);
```

파라미터

- scope`Construct`
- id`string`
- props`KinesisStreams...ToS3Props`

패턴 구성

이름	유형	설명
버킷 소품?	`s3.BucketProps`	선택적 사용자가 S3 버킷의 기본 소품을 재정의하는 소품을 제공했습니다.
클라우드왓찰암스 만들기	`boolean`	권장 CloudWatch 경보를 생성할지 여부를 선택할 수 있습니다.
버켓토비 기존에?	`s3.IBucket`	S3 버킷 객체의 선택적 기존 인스턴스입니다. 이것이 제공되는 경우`bucketProps` 는 오류입니다.

이름	유형	설명
기존로깅버켓트BJ?	s3.IBucket	패턴에 의해 생성된 S3 버킷에 대해 S3 버킷 객체를 로깅하는 선택적 기존 인스턴스입니다.
기존스트리모브?	kinesis.Stream	Kinesis Stream의 기존 인스턴스로, 이 인스턴스와kinesisStreamProps 를 사용하면 오류가 발생할 수 있습니다.
키네시스파이어호스프로프스?	aws-kinesisfirehose.CfnDeliveryStreamProps | any	Kinesis 파이어호스 배달 스트림의 기본 소품을 무시하기 위한 선택적 사용자가 제공한 소품입니다.
키네시스스트림프롭스?	kinesis.StreamProps	선택적 사용자가 Kinesis 스트림의 기본 소품을 무시하기 위해 prop을 제공했습니다.
로그그룹Props?	logs.LogGroupProps	선택적 사용자가 CloudWatchLogs 로그 그룹에 대한 기본 소품을 재정의하는 소품을 제공했습니다.

패턴 속성

이름	유형	설명
CloudwatchAms?	cloudwatch.Alarm[]	구조에서 만든 CloudWatch.alarm 인스턴스의 목록을 반환합니다.
키네시스파이어호스	kinesisfirehose.CfnDeliveryStream	구문에서 만든 키네시스파이어호스.cfn배달 스트림의 인스턴스를 반환합니다.

이름	유형	설명
키네시스파이어호스셀로그그룹	`logs.LogGroup`	Kinesis Data Firehose 전달 스트림에 대한 구문에서 생성된 Logs.logGroup의 인스턴스를 반환합니다.
키네시스파이어호스롤	`iam.Role`	Kinesis Data Firehose 전송 스트림용 구문에서 생성한 IAM.role 인스턴스를 반환합니다.
키네시스스트림롤	`iam.Role`	Kinesis 스트림에 대한 구문에서 생성된 IAM.role 인스턴스를 반환합니다.
S3Bucket?	`s3.Bucket`	구조에 의해 생성 된 S3.bucket의 인스턴스를 돌려줍니다.
s3로깅 버킷?	`s3.Bucket`	구조에 의해 생성된 S3.bucket의 인스턴스를 기본 버킷의 로깅 버킷으로 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon Kinesis Stream

- Kinesis 스트림에 대한 최소 권한 액세스 IAM 역할 구성
- AWS 관리형 KMS 키를 사용하여 Kinesis 스트림에 대한 서버 측 암호화 활성화
- Kinesis 스트림에 대한 CloudWatch 경보 모범 사례 배포

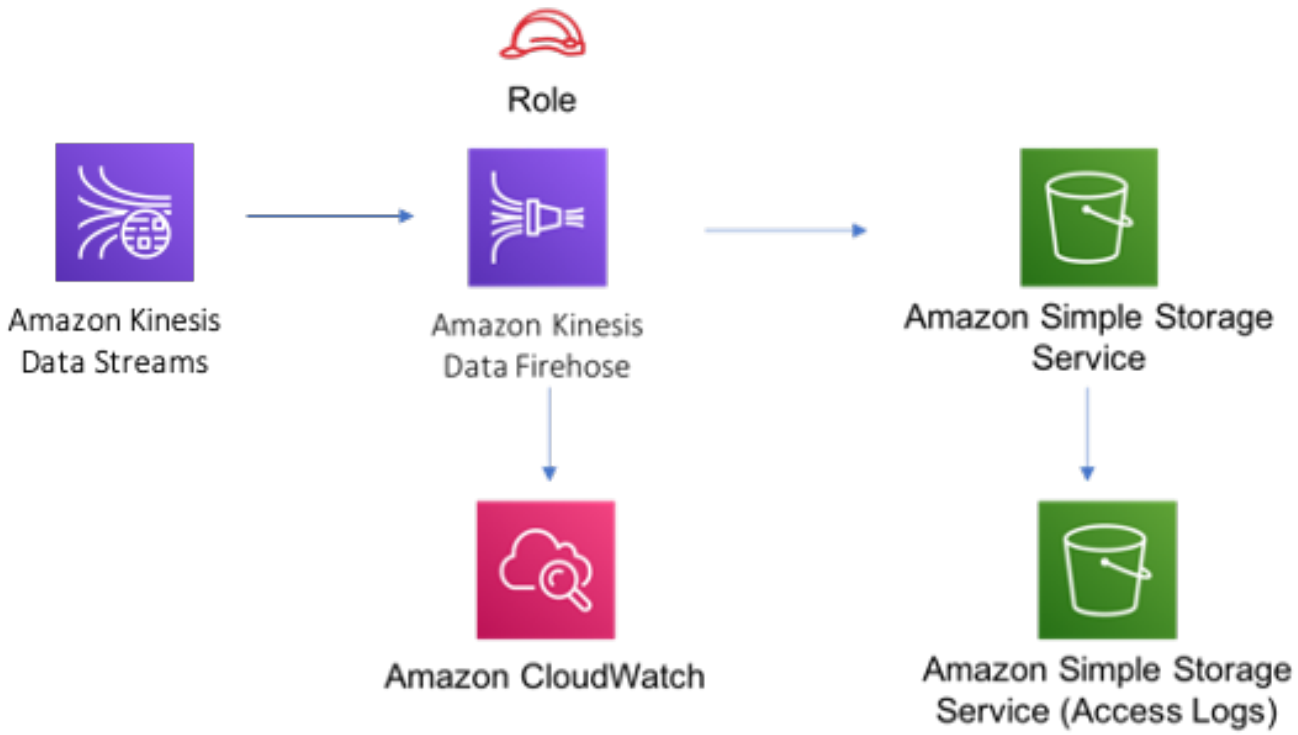
Amazon Kinesis Firehose

- Kinesis 파이어호스에 대한 CloudWatch 로깅 활성화
- Amazon Kinesis Firehose 대한 최소 권한 액세스 IAM 역할 구성

Amazon S3 버킷

- S3 버킷에 대한 액세스 로깅 구성
- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화 활성화
- 전송 중인 데이터의 암호화 강제 시행
- 버킷 버전 관리 사용
- S3 버킷에 대한 공용 액세스 허용 안 함
- CloudFormation 스택을 삭제할 때 S3 버킷 유지
- 90일 후에 최신 버전이 아닌 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙 적용

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS-키네시스스트림-키네시스파이어호스-s3

AWS-키네시스스트림-람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 사용합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws-kinesis-streams-lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-kinesisstreams-lambda`
Java	`software.amazon.awsconstructs.services.kinesisstreamslambda`

Overview

이 AWS 솔루션 구성 요소는 상호 작용 및 보안에 적합한 리소스/속성과 함께 Kinesis Stream 및 Lambda 함수를 배포합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { KinesisStreamsToLambda } from '@aws-solutions-constructs/aws-kinesisstreams-
lambda';

new KinesisStreamsToLambda(this, 'KinesisToLambdaPattern', {
    kinesisEventSourceProps: {
        startingPosition: lambda.StartingPosition.TRIM_HORIZON,
        batchSize: 1
```

```
    },
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

Initializer

```
new KinesisStreamsToLambda(scope: Construct, id: string, props:
 KinesisStreamsToLambdaProps);
```

파라미터

- scope`Construct`
- id`string`
- props`KinesisStreamsToLambdaProps`

Propction 구성

이름	유형	설명
람다오브즈 기존인가요?	`lambda.Function`	Lambda 함수 객체의 기존 인스턴스, 이 및`lambdaFunctionProps` 오류가 발생합니다.
람다기능소품?	`lambda.FunctionProps`	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.`existingLambdaObj` 가 제공될 예정입니다.

이름	유형	설명
키네시스스트림프롭스?	kinesis.StreamProps	Kinesis 스트림의 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
기존스트리모브?	kinesis.Stream	Kinesis 스트림의 기존 인스턴스. 이 인스턴스와kinesisStreamProps 오류가 발생합니다.
키네시세븐스소품?	aws-lambda-event-sources.KinesisEventSourceProps	Lambda 이벤트 소스 매핑에 대한 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
클라우드왓찰암스 만들기	boolean	권장 CloudWatch 경보를 생성할지 여부입니다.

패턴 속성

이름	유형	설명
키네시스스트림	kinesis.Stream	패턴에 의해 생성된 Kinesis 스트림의 인스턴스를 반환합니다.
LambDA	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
키네시스스트림롤	iam.Role	Kinesis 스트림의 패턴에 의해 생성된 IAM 역할의 인스턴스를 반환합니다.

이름	유형	설명
클라우드왓찰암즈?	`cloudwatch.Alarm[]`	패턴에 의해 생성된 하나 이상의 CloudWatch 경보 목록을 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

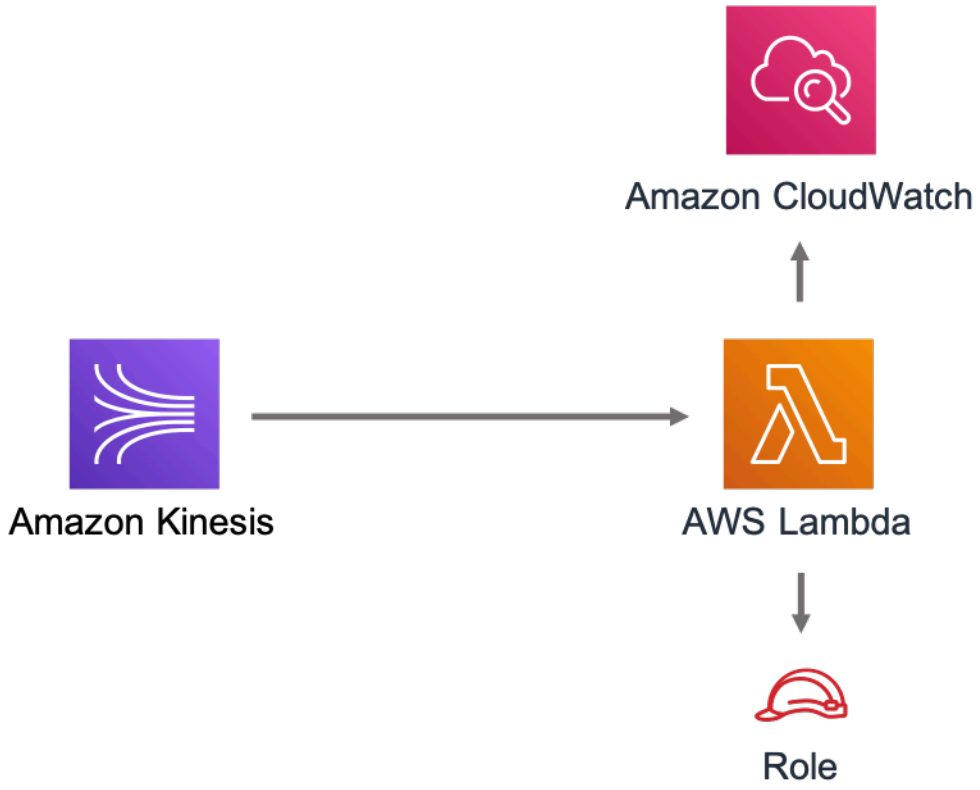
Amazon Kinesis Stream

- Kinesis 스트림에 대한 최소 권한 액세스 IAM 역할을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 Kinesis 스트림에 대한 서버 측 암호화를 활성화합니다.
- Kinesis 스트림에 대한 모범 사례 CloudWatch 경보를 배포합니다.

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 실패 처리 기능 사용: 함수 오류 시 이분절 사용, 기본 최대 레코드 사용 기간 (24시간) 설정, 기본 최대 재시도 횟수 (500) 설정, 실패 시 SQS 데드 레터 큐를 대상으로 배포합니다.
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - 키네시스 스트림 - 람다

aws-lambda-dynamodb

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_dynamodb`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-dynamodb`
Java	`software.amazon.awsconstructs.services.lambdadynamodb`

Overview

이 AWS 솔루션 구성은 최소 권한 권한을 가진 AWS Lambda 함수와 Amazon DynamoDB 테이블을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { LambdaToDynamoDBProps,  LambdaToDynamoDB } from '@aws-solutions-constructs/
aws-lambda-dynamodb';

const props: LambdaToDynamoDBProps = {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
};

new LambdaToDynamoDB(this, 'test-lambda-dynamodb-stack', props);
```

Initializer

```
new LambdaToDynamoDB(scope: Construct, id: string, props: LambdaToDynamoDBProps);
```

파라미터

- scopeConstruct
- idstring
- propsLambdaToDynamoDBProps

패턴 구성

이름	유형	설명
람다오브 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 사용하면 오류가 발생할 수 있습니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 가 제공될 예정입니다.
다이나모테이블소품?	dynamodb.TableProps	DynamoDB 테이블의 기본 소품을 무시하기 위한 선택적 사용자가 제공한 소품
기존 테이블오브J?	dynamodb.Table	DynamoDB 테이블 객체의 기존 인스턴스로, 이 인스턴스와dynamoTableProps 를 사용하면 오류가 발생할 수 있습니다.
테이블 사용 권한?	string	Lambda 함수에 부여할 선택적 테이블 권한입니다. 다음 옵션 중 하나를 지정할 수 있습니

이름	유형	설명
		다.All,Read,ReadWrite 또는Write.
테이블 환경변수 이름?	string	Lambda 함수에 대해 설정된 DynamoDB 테이블 환경 변수의 선택적 이름입니다.
기존VPC?	ec2.IVpc	이 패턴을 배포해야 하는 선택 사항인 기존 VPC 입니다. VPC PC에 배포되면 Lambda 함수는 VPC의 ENI를 사용하여 네트워크 리소스에 액세스하고 게이트웨이 엔드포인트는 Amazon DynamoDB 용 VPC에 생성됩니다. 기존 VPC 가 제공되면deployVpc 속성일 수 없습니다.true. 이를 사용합니다.ec2.IVpc를 사용하여 클라이언트가 스택 외부에 있는 VPC를 제공할 수 있도록ec2.Vpc.fromLookup() 메서드를 사용합니다.
vPCProps?	ec2.VpcProps	새 VPC 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.enableDnsHostnames ,enableDnsSupport ,natGateways , 및subnetConfiguration 는 패턴에 의해 설정되므로 여기에 제공된 속성에 대한 모든 값이 오버라이드됩니다. 다음의 경우,deployVpc 가 아닙니다.true이 속성은 무시됩니다.

이름	유형	설명
VPC를 배포하시겠습니까?	`boolean`	새 VPC 생성할지 여부`vpcProps`이 패턴을 배포 할 수 있습니다. 이 값을 true로 설정하면 패턴을 실행하기 위해 최소한의 대부분의 프라이빗 VPC 가 배포됩니다. • CDK 프로그램에서 사용하는 각 가용 영역에 하나의 격리된 서브넷 • `enableDnsHostnames` 및`enableDns Support` 는 둘 다`true` 이 속성이`true`을 선택한 다음`existingVpc` 를 지정할 수 없습니다. 기본값은 `false`입니다.

패턴 속성

이름	유형	설명
다이나모터블	`dynamodb.Table`	패턴에 의해 생성된 DynamoDB 테이블의 인스턴스를 반환합니다.
lambdaFunction	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
vPC?	`ec2.IVpc`	패턴에서 사용하는 VPC 인터페이스를 반환합니다 (있는 경우). 패턴에 의해 생성된 VPC

이름	유형	설명
		또는 패턴 생성자에 제공된 VPC일 수 있습니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

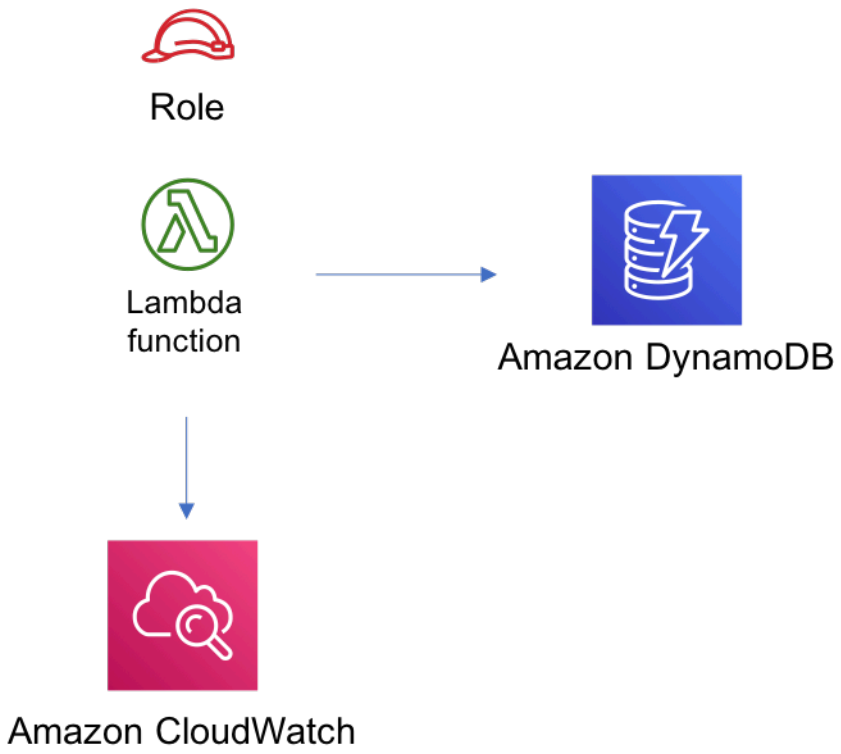
AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `DDB_TABLE_NAME` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon DynamoDB 테이블

- DynamoDB 테이블의 결제 모드를 온 디맨드 (요청당 지불) 로 설정합니다.
- AWS 관리형 KMS 키를 사용하여 DynamoDB 테이블에 대한 서버 측 암호화를 활성화합니다.
- DynamoDB 테이블에 대해 'id'라는 파티션 키를 생성합니다.
- CloudFormation 스택을 삭제할 때 테이블을 유지합니다.
- 지속적인 백업 및 지정 시간 복구를 지원합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws - 람다 - 동적 DB

aws-람다 탄성검색-키바나

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_elasticsearch_kibana`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-elasticsearch-kibana`
Java	`software.amazon.awsconstructs.services.lambdaelasticsearchkibana`

Overview

이 AWS 솔루션 구조는 최소 권한이 있는 AWS Lambda 함수와 Amazon Elasticsearch Service 도메인을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { LambdaToElasticSearchAndKibana } from '@aws-solutions-constructs/aws-lambda-elasticsearch-kibana';
import { Aws } from "@aws-cdk/core";

const lambdaProps: lambda.FunctionProps = {
    runtime: lambda.Runtime.NODEJS_14_X,
    // This assumes a handler function in lib/lambda/index.js
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler'
};

new LambdaToElasticSearchAndKibana(this, 'test-lambda-elasticsearch-kibana', {
    lambdaFunctionProps: lambdaProps,
    domainName: 'test-domain',
    // TODO: Ensure the Cognito domain name is globally unique
    cognitoDomainName: 'globallyuniquedomain' + Aws.ACCOUNT_ID;
});
```

Initializer

```
new LambdaToElasticSearchAndKibana(scope: Construct, id: string, props:
 LambdaToElasticSearchAndKibanaProps);
```

파라미터

- scopeConstruct
- idstring
- propsLambdaToElasticSearchAndKibanaProps

ProptoPool

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 오류가 발생할 수 있습니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 제공될 예정입니다.
이도메인Props?	elasticsearch.CfnDomainProps	Amazon Elasticsearch Service 기본 소품을 무시하기 위해 사용자가 제공한 소품 (옵션)
domainName	string	Cognito 및 Amazon Elasticsearch Service 도메인 이름
코그니트도메인 이름?	string	선택 사항인 Cognito 도메인 이름입니다. 제공된 경우 Cognito

이름	유형	설명
		도메인에 사용되며domainName 는 Elasticsearch 도메인에 사용됩니다.
클라우드왓찰암스 만들기	boolean	권장 CloudWatch 경보를 생성할지 여부입니다.
도메인엔드포인트변수 이름?	string	Lambda 함수에 대해 설정된 ElasticSearch 도메인 끝점 환경 변수의 선택적 이름입니다.

패턴 속성

이름	유형	설명
클라우드왓찰암스?	cloudwatch.Alarm[]	패턴에 의해 생성된 하나 이상의 CloudWatch 경보 목록을 반환합니다.
엘라스틱검색 도메인	elasticsearch.CfnDomain	패턴에 의해 생성된 Elasticsearch 도메인의 인스턴스를 반환합니다.
탄력적 검색도메인 역할	iam.Role	Elasticsearch 도메인의 패턴으로 생성된 IAM 역할의 인스턴스를 반환합니다.
IdentityPool	cognito.CfnIdentityPool	패턴에 의해 생성된 Cognito 자격 증명 풀의 인스턴스를 반환합니다.
람다함수	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.

이름	유형	설명
userPool	cognito.UserPool	패턴에 의해 생성된 Cognito 사용자 풀의 인스턴스를 반환합니다.
UserPool클라이언트	cognito.UserPoolClient	패턴에 의해 생성된 Cognito 사용자 풀 클라이언트의 인스턴스를 반환합니다.

Lambda 함수

이 패턴을 사용하려면 DynamoDB 스트림에서 Elasticsearch 서비스에 데이터를 게시할 수 있는 Lambda 함수가 필요합니다. 샘플 함수가 제공됩니다.여기에서.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `DOMAIN_ENDPOINT` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon Cognito

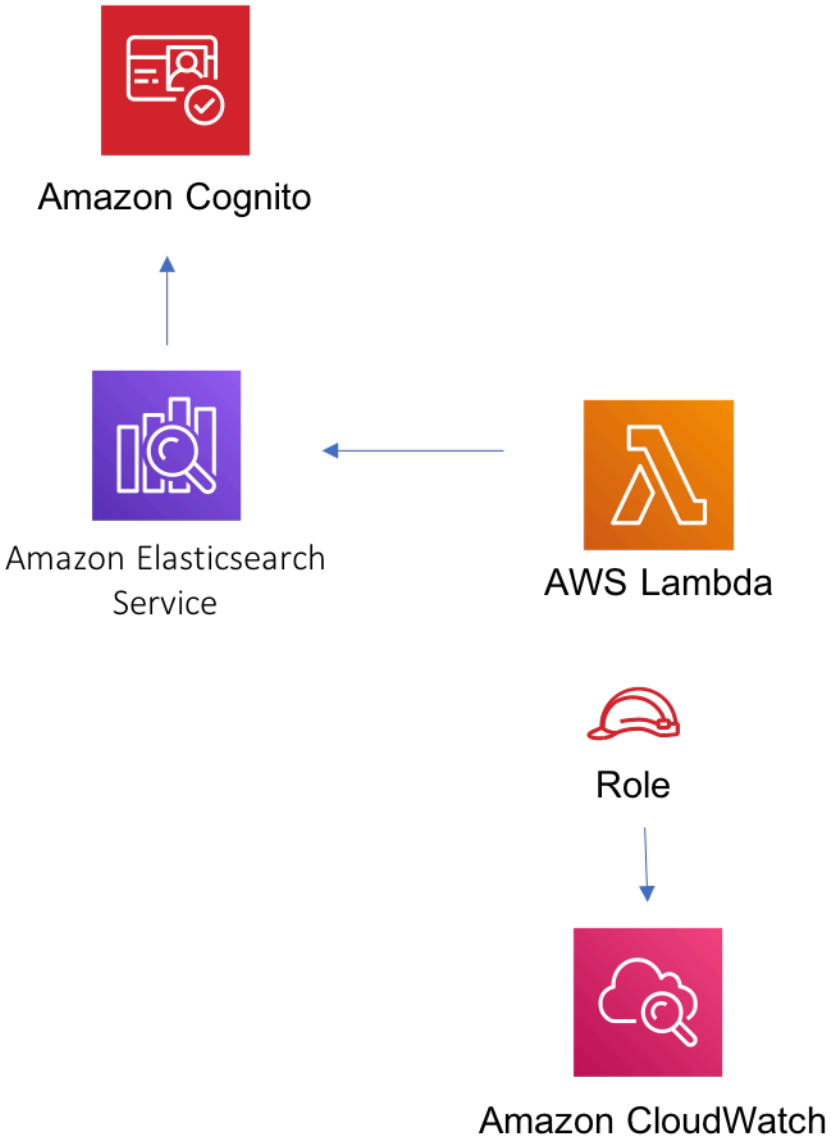
- 사용자 풀에 대한 암호 정책을 설정합니다.
- 사용자 풀에 고급 보안 모드를 적용합니다.

Amazon Elasticsearch Service

- 엘라스틱 검색 도메인에 대한 CloudWatch 경보를 배포합니다.

- Cognito 사용자 풀을 사용하여 Kibana 대시보드 액세스를 보호합니다.
- AWS 관리형 KMS 키를 사용하여 Elasticsearch 도메인에 대한 서버 측 암호화를 활성화합니다.
- Elasticsearch 도메인에 노드 간 암호화를 활성화합니다.
- Amazon ES 도메인에 대해 클러스터를 구성합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/aws - 람다 - 탄성 검색 - 키바나

aws-람다-s3

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 활성화합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_s3`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-s3`
Java	`software.amazon.awsconstructs.services.lambdas3`

Overview

이 AWS 솔루션 구조는 Amazon S3 버킷에 연결된 AWS Lambda 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { LambdaToS3 } from '@aws-solutions-constructs/aws-lambda-s3';

new LambdaToS3(this, 'LambdaToS3Pattern', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

Initializer

```
new LambdaToS3(scope: Construct, id: string, props: LambdaToS3Props);
```

파라미터

- scopeConstruct
- idstring
- propsLambdaToS3Props

패턴

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 에서 오류가 발생할 수 있습니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingL

이름	유형	설명
		`ambdaObj` 제공될 예정입니다.
버켓토비 기존에?	`s3.IBucket`	S3 버킷 객체의 기존 인스턴스입니다. 이것이 제공되는 경우`bucketProps` 는 오류입니다.
버킷 소품?	`s3.BucketProps`	버킷의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.`existingBucketObj` 제공될 예정입니다.
버킷 사용 권한?	`string[]`	Lambda 함수에 부여할 선택적 버킷 권한입니다. 다음 중 한 개 이상을 지정할 수 있습니다.`Delete`,`Put`,`Read`,`ReadWrite` ,`Write`.
기존VPC?	`ec2.IVpc`	이 패턴을 배포해야 하는 선택 사항인 기존 VPC 입니다. VPC PC에 배포되면 Lambda 함수는 VPC의 ENI를 사용하여 네트워크 리소스에 액세스하고 인터페이스 엔드포인트는 Amazon SQS 용 VPC에 생성됩니다. 기존 VPC 가 제공되면`deployVpc` 속성은`true`. 이 사용`ec2.IVpc`를 사용하여 클라이언트가 스택 외부에 있는 VPC를 제공할 수 있도록`ec2.Vpc.fromLookup()` 메서드를 호출합니다.

이름	유형	설명
VPC를 배포하시겠습니까?	boolean	새 VPC 생성하려면vpcProps이 패턴을 배포 할 수 있습니다. 이 를 로 설정합니다.true는 패턴을 실행하기 위해 최소한의 대부분의 프라이빗 VPC 배포합니다. • CDK 프로그램에서 사용하는 각 가용 영역에 하나의 격리된 서브넷입니다. • enableDnsHostnames 및enableDns Support 는 둘 다true. 이 속성이true을 선택한 다음existingVpc 를 지정할 수 없습니다. 기본값은 false입니다.
vPCProps?	ec2.VpcProps	새 VPC 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.enableDns Hostnames ,enableDns Support ,natGatewa ys 및subnetCon figuration 는 패턴에 의해 설정되므로 여기에 제공된 속성에 대한 모든 값이 오버라이드됩니다. 다음의 경우,deployVpc 이 (가)true이 속성은 무시됩니다.
버켓텐바이론변수 이름?	string	Lambda 함수에 대한 S3 버킷 환경 변수 세트의 선택적 이름입니다.

패턴

이름	유형	설명
람다함수	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
S3Bucket?	`s3.Bucket`	패턴에 의해 생성된 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	`s3.Bucket`	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.
VPC VPC	`ec2.IVpc`	패턴에 사용된 VPC 인스턴스 (있는 경우) 를 반환합니다. 패턴에 의해 생성된 VPC 또는 패턴 생성자에 제공된 VPC일 수 있습니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적
- 환경 변수를 설정합니다.
  - `S3_BUCKET_NAME` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon S3 버킷

- S3 버킷에 대한 액세스 로깅을 구성합니다.

- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화를 활성화합니다.
- S3 버킷의 버전 관리를 켭니다.
- S3 버킷에 대한 공용 액세스를 허용하지 않습니다.
- CloudFormation 스택을 삭제할 때 S3 버킷을 유지합니다.
- 전송 중인 데이터의 암호화를 실행합니다.
- 90일 후에 최신 버전이 아닌 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

Architecture

GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws - 람다 - S3

AWS-람다-ssm문자열 매개 변수

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_ssm_string_parameter`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-ssmstringparameter`
Java	`software.amazon.awsconstructs.services.lambdassmstringparameter`

Overview

이 AWS 솔루션 구문은 권한이 가장 적은 AWS Lambda 함수와 AWS 시스템 관리자 매개 변수 저장소 문자열 매개 변수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
const { LambdaToSsmstringparameterProps,  LambdaToSsmstringparameter } from '@aws-solutions-constructs/aws-lambda-ssmstringparameter';

const props: LambdaToSsmstringparameterProps = {
    lambdaFunctionProps: {
      runtime: lambda.Runtime.NODEJS_14_X,
      // This assumes a handler function in lib/lambda/index.js
      code: lambda.Code.fromAsset(`${__dirname}/lambda`),
      handler: 'index.handler'
    },
    stringParameterProps: { stringValue: "test-string-value" }
};

new LambdaToSsmstringparameter(this, 'test-lambda-ssmstringparameter-stack', props);
```

Initializer

```
new LambdaToSsmstringparameter(scope: Construct, id: string, props:
 LambdaToSsmstringparameterProps);
```

파라미터

- scopeConstruct
- idstring
- propsLambdaToSsmstringparameterProps

Prop 구성

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 사용하면 오류가 발생할 수 있습니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 다음의 경우 무시됩니다.existingLambdaObj 가 제공됩니다.
기존 문자열 매개 변수BJ?	ssm.StringParameter	SSM String 매개 변수 객체의 기존 인스턴스로,stringParameterProps 를 사용하면 오류가 발생할 수 있습니다.
문자열 매개 변수소품?	ssm.StringParameterProps	선택적 사용자가 SSM String 매개 변수의 기본 소품을 재정의하는 소품을 제공했습니다. 다음의 경우,existingStringParameterObj 이 설

이름	유형	설명
		정되지 않은 경우stringParameterProps 은 필수입니다. 지원되는 유일한ssm.StringParameterProps.type 확장하는 데STRING다른 값이 제공되면 재정의됩니다.
문자열매개 변수변환변수 이름	string	Lambda 함수에 대해 설정된 SSM 문자열 매개 변수 환경 변수의 선택적 이름입니다.
기존VPC?	ec2.IVpc	이 패턴을 배포해야 하는 선택 사항인 기존 VPC 입니다. VPC PC에 배포되면 Lambda 함수는 VPC의 ENI를 사용하여 네트워크 리소스에 액세스하고 인터페이스 엔드포인트는 AWS Systems Manager 파라미터용 VPC에 생성됩니다. 기존 VPC 가 제공되면deployVpc 속성은true. 이를 사용합니다.ec2.IVpc를 사용하여 클라이언트가 스택 외부에 있는 VPC를 제공할 수 있도록ec2.Vpc.fromLookup() 메서드를 사용합니다.

이름	유형	설명
vPCProps?	ec2.VpcProps	새 VPC 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.enableDnsHostnames ,enableDnsSupport ,natGateways 및subnetConfiguration 는 패턴에 의해 설정되므로 여기에 제공된 속성에 대한 모든 값이 오버라이드됩니다. 다음의 경우,deployVpc 이 (가)true이 속성은 무시됩니다.
VPC를 배포하시겠습니까?	boolean	를 기반으로 새 VPC 생성할지 여부vpcProps이 패턴을 배포 할 수 있습니다. 이 로 설정true는 패턴을 실행하기 위해 최소한의 대부분의 프라이빗 VPC 배포합니다. • CDK 프로그램에서 사용하는 각 가용 영역에 하나의 격리된 서브넷입니다. • enableDnsHostnames 및enableDnsSupport 는 둘 다true. 이 속성이true을 선택한 다음existingVpc 를 지정할 수 없습니다. 기본값은false입니다.

이름	유형	설명
문자열 매개 변수사용 권한?	`string`	Lambda 함수에 부여할 선택적 SSM 문자열 매개 변수 권한입니다. 다음 중 하나를 지정할 수 있습니다.`Read`,`ReadWrite` .

패턴 속성

이름	유형	설명
람다함수	`lambda.Function`	의 인스턴스를 반환`lambda.Function` 구성에 의해 작성되었습니다.
String파라미터	`ssm.StringParameter`	의 인스턴스를 반환`ssm.StringParameter` 구성에 의해 작성되었습니다.
VPC?	`ec2.IVpc`	패턴에서 사용하는 VPC 인터페이스를 반환합니다 (있는 경우). 패턴에 의해 생성된 VPC 또는 패턴 생성자에 제공된 VPC일 수 있습니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

AWS Lambda 함수

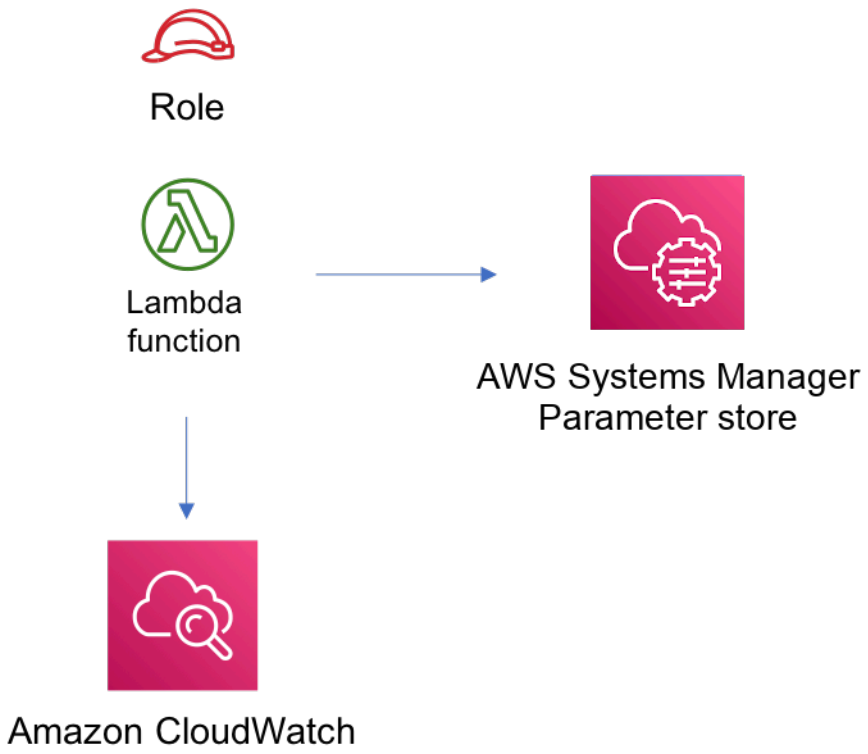
- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.

- `SSM_STRING_PARAMETER_NAME` (default)
- `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon AWS Systems Manager 파라미터 스토어 문자열

- 연결된 AWS Lambda 함수에 대한 읽기 전용 액세스를 활성화합니다.
- 제공된 값으로 새 SSM 문자열 매개 변수를 만듭니다.
- CloudFormation 스택을 삭제할 때 SSM 문자열 매개 변수를 유지합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws-람다 - ssm문자열 매개 변수

람다-세이지메이크렌드포인트

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_sagemakerendpoint`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-sagemakerendpoint`
Java	`software.amazon.awsconstructs.services.lambdasagemakerendpoint`

Overview

이 AWS 솔루션 구성은 Amazon 세이지메이커 엔드포인트에 연결된 AWS Lambda 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { Duration } from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';
import {
  LambdaToSagemakerEndpoint,
  LambdaToSagemakerEndpointProps,
} from '@aws-solutions-constructs/aws-lambda-sagemakerendpoint';
```

```
const constructProps: LambdaToSagemakerEndpointProps = {
  modelProps: {
    primaryContainer: {
      image: '{{AccountId}}.dkr.ecr.{{region}}.amazonaws.com/linear-learner:latest',
      modelDataUrl: 's3://{{bucket-name}}/{{prefix}}/model.tar.gz',
    },
  },
  lambdaFunctionProps: {
    runtime: lambda.Runtime.PYTHON_3_8,
    // This assumes a handler function in lib/lambda/index.py
    code: lambda.Code.fromAsset(`${__dirname}/lambda`),
    handler: 'index.handler',
    timeout: Duration.minutes(5),
    memorySize: 128,
  },
};

new LambdaToSagemakerEndpoint(this, 'LambdaToSagemakerEndpointPattern',
 constructProps);
```

Initializer

```
new LambdaToSagemakerEndpoint(scope: Construct, id: string, props:
 LambdaToSagemakerEndpointProps);
```

파라미터

- scope`Construct`
- id`string`
- props`LambdaToSagemakerEndpointProps`

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	`lambda.Function`	Lambda 함수 객체의 기존 인스턴스, 이 및`lambdaFun`

이름	유형	설명
		ctionProps 오류가 발생합니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.
기존메이크렌드포인트저장되시겠습니까?	sagemaker.CfnEndpoint	사용할 기존 세이지메이커 엔포인트 (선택 사항) 입니다. 이 두 가지를 모두 제공endpointProps 오류가 발생합니다.
모델소품?	sagemaker.CfnModelProps | any	Sagemaker 모델의 기본 속성을 재정의하는 사용자 제공 속성입니다. 최소modelProps.primaryContainer 를 제공해야만 모델을 만들 수 있습니다. 기본적으로 패턴은 최소 필수 권한을 가진 역할을 만들지만 클라이언트는modelProps.executionRoleArn .
끝점구성 소품?	sagemaker.CfnEndpointConfigProps	Sagemaker 끝점 구성의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.
엔드포인트 Props?	sagemaker.CfnEndpointProps	Sagemaker 끝점의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.

이름	유형	설명
기존VPC?	`ec2.IVpc`	이 구문을 배포해야 하는 선택 사항인 기존 VPC 입니다. VPC 배포되면 Lambda 함수와 Sagemaker 엔드포인트는 VPC의 ENI를 사용하여 네트워크 리소스에 액세스합니다. 인터페이스 엔드포인트는 Amazon 세이지메이커 런타임용 VPC 및 Amazon S3 VPC 엔드포인트에 생성됩니다. 기존 VPC 가 제공되면`deployVpc` 속성은 일 수 없습니다.`true`.
vPCProps?	`ec2.VpcProps`	새 VPC 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.`enableDnsHostnames` ,`enableDnsSupport` ,`natGateways` 및`subnetConfiguration` 는 구조에 의해 설정되므로 여기에 제공된 속성에 대한 모든 값이 overrriden됩니다. 다음의 경우,`deployVpc` 이 (가)`true`, 다음이 속성은 무시됩니다.

이름	유형	설명
VPC를 배포하시겠습니까?	`boolean`	를 기반으로 새 VPC 생성할 지 여부`vpcProps`이 패턴을 배포 할 수 있습니다. 이 로 설정`true`는 패턴을 실행하기 위해 최소한의 대부분의 프라이빗 VPC 배포합니다. • CDK 프로그램에서 사용하는 각 가용 영역에 하나의 격리된 서브넷입니다. • `enableDnsHostnames` 및`enableDns Support` 는 둘 다`true`. 이 속성을 로 설정한 경우`true`을 선택한 다음`existingVpc` 를 지정할 수 없습니다. 기본값은 `false`입니다.
이름변경 내용을 저장하시겠습니까?	`string`	Lambda 함수에 대한 SageMaker 끝점 환경 변수 세트의 선택적 이름입니다.

패턴 속성

이름	유형	설명
람다함수	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.

이름	유형	설명
세이지메이크렌드포인트	sagemaker.CfnEndpoint	패턴에 의해 생성 된 세이지메이커 끝점의 인스턴스를 돌려줍니다.
저장하시겠습니까?	sagemaker.CfnEndpointConfig	패턴에 의해 생성된 SageMaker EndpointConfig 인스턴스를 반환합니다.existingSagemakerEndpointObj 는 제공되지 않습니다.
세이지메이커 모델?	sagemaker.CfnModel	만약 패턴에 의해 생성 된 세이지메이커 모델의 인스턴스를 돌려줍니다existingSagemakerEndpointObj 는 제공되지 않습니다.
VPC?	ec2.IVpc	패턴에 의해 생성된 VPC 인스턴스를 반환합니다.deployVpc 확장하는 데true, 또는existingVpc 가 제공될 예정입니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

AWS Lambda 함수

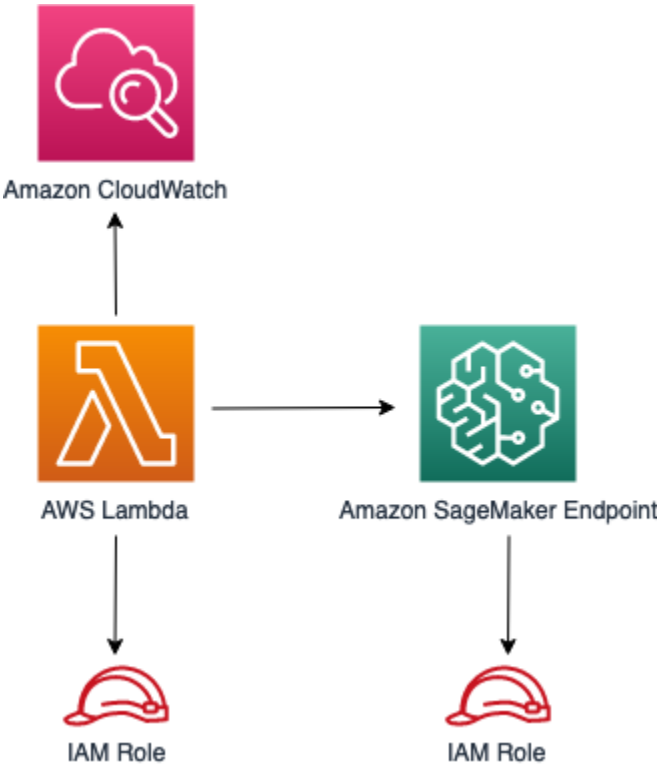
- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- 함수가 추론을 위해 Sagemaker 끝점을 호출하도록 허용합니다.
- Sagemaker 엔드포인트가 배포되는 VPC 리소스에 액세스하도록 함수를 구성합니다.
- X-Ray 추적 사용
- 환경 변수를 설정합니다.

- `SAGEMAKER_ENDPOINT_NAME` (default)
- `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon SageMaker 엔드포인트

- Sagemaker 리소스를 생성할 수 있는 제한된 권한을 구성합니다.
- 세이지메이커 모델, 엔드포인트 구성 및 엔드포인트를 배포합니다.
- VPC 배포할 Sagemaker 엔드포인트를 구성합니다.
- S3 VPC 엔드포인트 및 세이지메이커 런타임 VPC 인터페이스를 배포합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	[@aws -솔루션 - 구성/aws-람다 - 세이지메이크 렌드 포인트]

aws-람다 비밀 관리자

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_secretsmanager`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-secretsmanager`
Java	`software.amazon.awsconstructs.services.lambdasecretsmanager`

Overview

이 AWS 솔루션 구성 요소는 최소한의 권한으로 AWS Lambda 함수 및 AWS Secrets Manager 암호를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
const { LambdaToSecretsmanagerProps,  LambdaToSecretsmanager } from '@aws-solutions-constructs/aws-lambda-secretsmanager';

const props: LambdaToSecretsmanagerProps = {
    lambdaFunctionProps: {
      runtime: lambda.Runtime.NODEJS_14_X,
```

```
      // This assumes a handler function in lib/lambda/index.js
      code: lambda.Code.fromAsset(`${__dirname}/lambda`),
      handler: 'index.handler'
    },
};

new LambdaToSecretsmanager(this, 'test-lambda-secretsmanager-stack', props);
```

Initializer

```
new LambdaToSecretsmanager(scope: Construct, id: string, props:
 LambdaToSecretsmanagerProps);
```

파라미터

- scope`Construct`
- id`string`
- props`LambdaToSecretsmanagerProps`

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	`lambda.Function`	Lambda 함수 객체의 기존 인스턴스, 이 및`lambdaFunctionProps` 를 사용하면 오류가 발생할 수 있습니다.
람다기능소품?	`lambda.FunctionProps`	사용자가 Lambda 함수의 기본 소품을 재정의하는 소품을 제공했습니다.
비밀 소품?	`secretsmanager.SecretProps`	선택적 사용자가 Secrets Manager 기본 소품을 재정의하는 소품을 제공했습니다.

이름	유형	설명
기존비밀정보비?	secretsmanager.Secret	Secrets Manager 비밀 개체의 기존 인스턴스, 이것이 설정되면secretProps 는 무시됩니다.
부여된 사이트 액세스?	boolean	Lambda 함수의 비밀에 대한 선택적 쓰기 액세스 (기본적으로 읽기 전용).
비밀번호변수이름?	string	Lambda 함수에 대해 설정된 Secrets Manager 보안 환경 변수의 선택적 이름입니다.
기존VPC?	ec2.IVpc	이 패턴을 배포해야 하는 선택 사항인 기존 VPC 입니다. VPC PC에 배포되면 Lambda 함수는 VPC의 ENI를 사용하여 네트워크 리소스에 액세스하며 AWS Secrets Manager 용 VPC에 인터페이스 엔드포인트가 생성됩니다. 기존 VPC 가 제공되면deployVpc 속성일 수 없습니다.true. 이 사용ec2.IVpc를 사용하여 클라이언트가 스택 외부에 있는 VPC를 제공할 수 있도록ec2.Vpc.fromLookup() 메서드를 사용합니다.

이름	유형	설명
vPCProps?	ec2.VpcProps	새 VPC 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.enableDnsHostnames ,enableDnsSupport ,natGateways , 및subnetConfiguration 는 패턴에 의해 설정되므로 여기에 제공된 속성에 대한 모든 값이 오버라이드됩니다. 다음의 경우,deployVpc 가 아닙니다.true이 속성은 무시됩니다.
VPC를 배포하시겠습니까?	boolean	를 기반으로 새 VPC 생성할지 여부vpcProps이 패턴을 배포할 수 있습니다. 이 설정을 로 설정합니다.true는 패턴을 실행하기 위해 최소한의 대부분의 프라이빗 VPC 배포합니다. • CDK 프로그램에서 사용하는 각 가용 영역에서 하나의 격리된 서브넷 • enableDnsHostnames 및enableDnsSupport 는 둘 다true 이 속성이true을 선택한 다음existingVpc 를 지정할 수 없습니다. 기본값은 false입니다.

패턴 속성

이름	유형	설명
람다함수	`lambda.Function`	의 인스턴스를 반환`lambda.Function` 구성에 의해 작성되었습니다.
암호	`secretsmanager.Secret`	의 인스턴스를 반환`secretsmanager.Secret` 구성에 의해 작성되었습니다.
VPC?	`ec2.IVpc`	패턴에서 사용하는 VPC 인터페이스를 반환합니다 (있는 경우). 패턴에 의해 생성된 VPC 또는 패턴 생성자에 제공된 VPC일 수 있습니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

AWS Lambda 함수

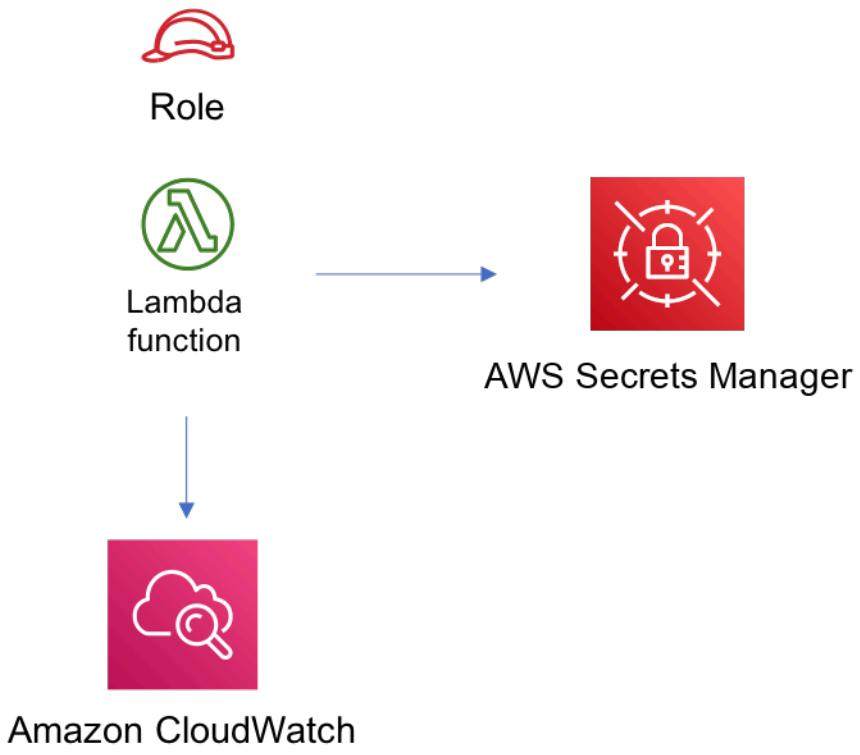
- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - (기본값) CDK에 의해 반환되는 비밀의 ARN 포함하는 SECRET_ARNSecretArnproperty
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon Secrets Manager 비밀

- 연결된 AWS Lambda 함수에 대한 읽기 전용 액세스 활성화
- 계정 및 지역에 대한 기본 KMS 키를 사용하여 서버 측 암호화 사용

- 새 보안 암호를 생성합니다.
  - (기본값) 임의의 이름
  - (기본값) 난수 값
- CloudFormation 스택을 삭제할 때 비밀 유지

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws - 람다 - 비밀 관리자

aws-람다-sns

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_sns`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-sns`
Java	`software.amazon.awsconstructs.services.lambdasns`

Overview

이 AWS 솔루션 구성은 Amazon SNS 주제에 연결된 AWS Lambda 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { LambdaToSns, LambdaToSnsProps } from "@aws-solutions-constructs/aws-lambda-sns";

new LambdaToSns(this, 'test-lambda-sns', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

Initializer

```
new LambdaToSns(scope: Construct, id: string, props: LambdaToSnsProps);
```

파라미터

- scopeConstruct
- idstring
- propsLambdaToSnsProps

패턴 구성

이름	유형	설명
람다오브 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 초과하면 오류가 발생합니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우 무시됩니다.existingLambdaObj 가 제공될 예정입니다.
기존토픽코비?	sns.Topic	SNS Topic 객체의 기존 인스턴스로,topicProps 를 초과하면 오류가 발생합니다.
주제소품?	sns.TopicProps	SNS 주제의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.
기존VPC?	ec2.IVpc	이 패턴을 배포해야 하는 선택 사항인 기존 VPC 입니다.

이름	유형	설명
		VPC PC에 배포되면 Lambda 함수는 VPC의 ENI를 사용하여 네트워크 리소스에 액세스하고 인터페이스 엔드포인트는 Amazon SQS 용 VPC에 생성됩니다. 기존 VPC 가 제공되면`deployVpc` 속성은 될 수 없습니다.`true`. 이 사용`ec2.IVpc`를 사용하여 클라이언트가 스택 외부에 있는 VPC를 제공할 수 있도록`ec2.Vpc.fromLookup()` 메서드를 사용합니다.
VPC를 배포하시겠습니까?	`boolean`	를 기반으로 새 VPC 생성할지 여부`vpcProps`이 패턴을 배포할 수 있습니다. 이를 로 설정합니다.`true`는 패턴을 실행하기 위해 최소한의 대부분의 프라이빗 VPC 배포합니다. • 각 가용 영역에서 CDK 프로그램에서 사용하는 하나의 격리된 서브넷입니다. • `enableDnsHostnames` 및`enableDnsSupport` 는 둘 다`true`. 이 속성이`true`을 선택한 다음`existingVpc` 를 지정할 수 없습니다. 기본값은 `false`입니다.

이름	유형	설명
vPCProps?	`ec2.VpcProps`	새 VPC 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.`enableDnsHostnames` ,`enableDnsSupport` ,`natGateways` 및`subnetConfiguration` 는 패턴에 의해 설정되므로 여기에 제공된 속성에 대한 모든 값이 오버라이드됩니다. 다음의 경우,`deployVpc` 가 아닙니다.`true`이 속성은 무시됩니다.
주제환경변수이름?	`string`	Lambda 함수에 대한 SNS 주제 ARN 환경 변수 세트의 선택적 이름입니다.
항목이름환경변수 이름?	`string`	Lambda 함수에 대한 SNS 주제 이름 환경 변수 세트의 선택적 이름입니다.

패턴 등록 정보

이름	유형	설명
LambDop	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
snsTopic	`sns.Topic`	패턴에 의해 생성된 SNS 주제의 인스턴스를 반환합니다.
vPC?	`ec2.IVpc`	패턴에 사용된 VPC 인스턴스 (있는 경우) 를 반환합니다. 패턴에 의해 생성된 VPC 또는 패

이름	유형	설명
		턴 생성자에 제공된 VPC일 수 있습니다.

기본 설정

재정의가없는 구성의 기본 구현은 다음 기본값을 설정합니다.

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `SNS_TOPIC_NAME` (default)
  - `SNS_TOPIC_ARN` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon SNS 주제

- SNS 주제에 대한 최소 권한 액세스 권한을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 서버 측 암호화가 활성화됩니다.
- 전송 중인 데이터의 암호화 강제 시행

Architecture

GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/aws - 람다 - SNS

람다 스퀘어

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_sqs`

언어	패키지
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-sqs`
Java	`software.amazon.awsconstructs.services.lambdasqs`

Overview

이 AWS 솔루션 구성은 Amazon SQS 대기열에 연결된 AWS Lambda 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { LambdaToSqs, LambdaToSqsProps } from "@aws-solutions-constructs/aws-lambda-sqs";

new LambdaToSqs(this, 'LambdaToSqsPattern', {
   lambdaFunctionProps: {
       runtime: lambda.Runtime.NODEJS_14_X,
       // This assumes a handler function in lib/lambda/index.js
       code: lambda.Code.fromAsset(`${__dirname}/lambda`),
       handler: 'index.handler'
   }
});
```

Initializer

```
new LambdaToSqs(scope: Construct, id: string, props: LambdaToSqsProps);
```

파라미터

- scope`Construct`
- id`string`

- propsLambdaToSqsProps

패턴 구성

이름	유형	설명
람다오브 기존인가요?	lambda.Function	기본 함수 대신 사용할 선택적 기존 Lambda 함수입니다. 이 두 가지를 모두 제공lambdaFunctionProps 를 사용하면 오류가 발생할 수 있습니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.
대기열에 있는 Obj?	sqs.Queue	기본 대기열 대신 사용할 기존 SQS 대기열 선택 사항입니다. 이 두 가지를 모두 제공queueProps 를 사용하면 오류가 발생할 수 있습니다.
대기열 소품?	sqs.QueueProps	SQS 대기열의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.
대기열 제거를 활성화하시겠습니까?	boolean	SQS 대기열을 비울 수 있도록 Lambda 함수에 추가 권한을 부여할지 여부 기본값은 false입니다.
배포데드 레터 큐?	boolean	배달 못한 편지 대기열로 사용될 보조 대기열을 생성할지 여부. 기본값은 true입니다.
데드레터 대기열Props?	sqs.QueueProps	데드 레터 큐의 기본 소품을 재정의하는 선택적 사용자 제

이름	유형	설명
		공 소품. 이 경우에만 사용됩니다.`deployDeadLetterQueue` 속성이 true로 설정됩니다.
maxReceiveCount?	`number`	배달 못한 편지 대기열로 이동되기 전에 메시지가 대기열에서 성공적으로 대기열에서 뺄 수 있는 횟수입니다. 기본값은 15입니다.
기존VPC?	`ec2.IVpc`	이 패턴을 배포해야 하는 선택 사항인 기존 VPC 입니다. VPC PC에 배포되면 Lambda 함수는 VPC의 ENI를 사용하여 네트워크 리소스에 액세스하고 인터페이스 엔드포인트는 Amazon SQS 용 VPC에 생성됩니다. 기존 VPC 가 제공되면`deployVpc` 속성일 수 없습니다.`true`. 한`ec2.IVpc`는 클라이언트가 스택 외부에 있는 VPC를`ec2.Vpc.fromLookup()` 메서드를 호출합니다.

이름	유형	설명
VPC를 배포하시겠습니까?	boolean	를 기반으로 새 VPC 생성할지 여부vpcProps이 패턴을 배포할 수 있습니다. 이 설정을 로 설정합니다.true는 패턴을 실행하기 위해 최소한의 대부분의 프라이빗 VPC 배포합니다. • CDK 프로그램에서 사용하는 각 가용 영역에 격리된 서브넷 1개 • enableDnsHostnames 및enableDns Support 는 둘 다true 이 속성이true을 선택한 다음existingVpc 를 지정할 수 없습니다. 기본값은 false입니다.
vPCProps?	ec2.VpcProps	새 VPC 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.enableDns Hostnames ,enableDns Support ,natGateways , 및subnetConfiguratio n 는 패턴에 의해 설정되므로 여기에 제공된 속성의 모든 값이 재정의됩니다. 다음의 경우,deployVpc 가 아닙니다.true, 다음이 속성은 무시됩니다.
대기열 환경변수 이름?	string	Lambda 함수에 대한 SQS 대기열 URL 환경 변수 세트의 선택적 이름입니다.

패턴 속성

이름	유형	설명
데드 레터 큐?	`sqs.Queue`	패턴에 의해 생성 된 데드 레터 큐의 인스턴스를 돌려줍니다 (배포 된 경우).
Lambdafunction	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
SQqueue	`sqs.Queue`	패턴에 의해 생성된 SQS 대기열의 인스턴스를 반환합니다.
VPC?	`ec2.IVpc`	패턴에 의해 생성되거나 사용된 VPC 인스턴스 (있는 경우)를 반환합니다. 패턴에 의해 생성된 VPC 또는 패턴 생성자에 제공된 VPC일 수 있습니다.

기본 설정

재정의가없는 구성의 기본 구현은 다음 기본값을 설정합니다.

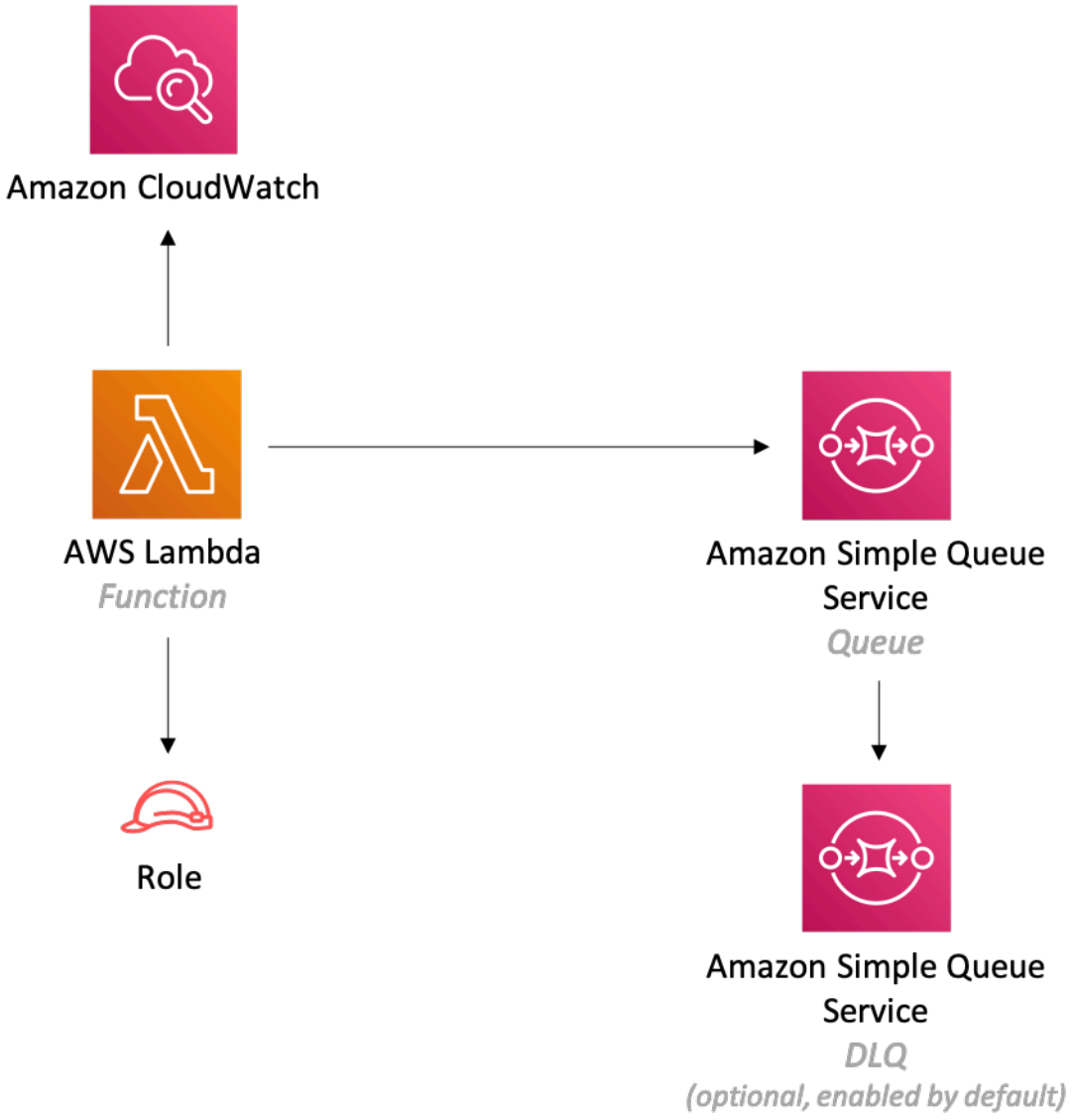
AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- 함수에서 대기열에만 메시지를 보낼 수 있습니다 (제거는`enableQueuePurge`속성입니다).
- X-Ray 추적 활성화
- 환경 변수를 설정합니다.
  - `SQS_QUEUE_URL`
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon SQS 대기열

- 소스 SQS 대기열에 대한 SQS 배달 못한 편지 대기열을 배포합니다.
- AWS 관리형 KMS 키를 사용하여 소스 SQS 대기열에 대한 서버 측 암호화를 활성화합니다.
- 전송 중인 데이터의 암호화를 적용합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/aws - 람다 - sqs

람다 스퀘어 람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_sqs_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-sqs-lambda`
Java	`software.amazon.awsconstructs.services.lambdasqslambda`

Overview

이 AWS 솔루션은 (1) 메시지를 대기열로 전송하도록 구성된 AWS Lambda 함수, (2) Amazon SQS 대기열, (3) 대기열에서 메시지를 소비하도록 구성된 AWS Lambda 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { LambdaToSqsToLambda, LambdaToSqsToLambdaProps } from "@aws-solutions-
constructs/aws-lambda-sqs-lambda";

new LambdaToSqsToLambda(this, 'LambdaToSqsToLambdaPattern', {
    producerLambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/producer-function/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda/producer-function`),
        handler: 'index.handler'
    },
    consumerLambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/consumer-function/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda/consumer-function`),
        handler: 'index.handler'
    }
});
```

Initializer

```
new LambdaToSqsToLambda(scope: Construct, id: string, props: LambdaToSqsToLambdaProps);
```

파라미터

- scope`Construct`
- id`string`
- props`LambdaToSqsToLambdaProps`

패턴 구성

이름	유형	설명
기존프로듀서람다오브?	`lambda.Function`	대기열에 메시지를 보내는 기본 함수 대신 사용할 선택적 기존 Lambda 함수입니다. 이 두

이름	유형	설명
		가지를 모두 제공`producerLambdaFunctionProps` 오류가 발생합니다.
프로듀서람다기능Props?	`lambda.FunctionProps`	생성자 Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.
대기열에 있는 Obj?	`sqs.Queue`	기본 대기열 대신 사용할 기존 SQS 대기열 선택 사항입니다. 이 두 가지를 모두 제공`queueProps` 오류가 발생합니다.
대기열 소품?	`sqs.QueueProps`	SQS 대기열의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 이 두 가지를 모두 제공`existingQueueObj` 오류가 발생합니다.
배포데드 레터 큐?	`boolean`	배달 못한 편지 대기열로 사용할 보조 대기열을 생성할지 여부를 지정합니다. 기본값은 `true`입니다.
데드레터 대기열Props?	`sqs.QueueProps`	데드 레터 큐의 기본 소품을 재정의하는 선택적 사용자 제공 소품. 이 인 경우에만 사용됩니다.`deployDeadLetterQueue` 속성은`true`.
maxReceiveCount?	`number`	배달 못한 편지 대기열로 이동되기 전에 메시지가 대기열에서 빼지 못한 횟수입니다. 기본값은 15입니다.

이름	유형	설명
기존소비자람다오브?	`lambda.Function`	대기열에서 메시지를 수신/소비하기 위한 기본 함수 대신 사용할 선택적 기존 Lambda 함수입니다. 이 두 가지를 모두 제공`consumerLambdaFunctionProps` 오류가 발생합니다.
소비자람다기능Props?	`lambda.FunctionProps`	소비자 Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.
대기열 환경변수 이름?	`string`	생성자 Lambda 함수에 대해 설정된 SQS 대기열 URL 환경변수의 선택적 이름입니다.

패턴 속성

이름	유형	설명
소비자람다함수	`lambda.Function`	패턴에 의해 생성 된 소비자 Lambda 함수의 인스턴스를 돌려줍니다.
데드 레터 큐?	`sqs.Queue`	패턴에 의해 생성 된 데드 레터 큐의 인스턴스를 돌려줍니다.
프로듀서람다함수	`lambda.Function`	패턴에 의해 생성 된 프로듀서 Lambda 함수의 인스턴스를 돌려줍니다.
분류: SQqueue	`sqs.Queue`	패턴에 의해 생성된 SQS 대기열의 인스턴스를 반환합니다.

기본 설정

재정의 된 속성이없는 이 Construct의 기본 구현은 다음 기본값을 준수합니다.

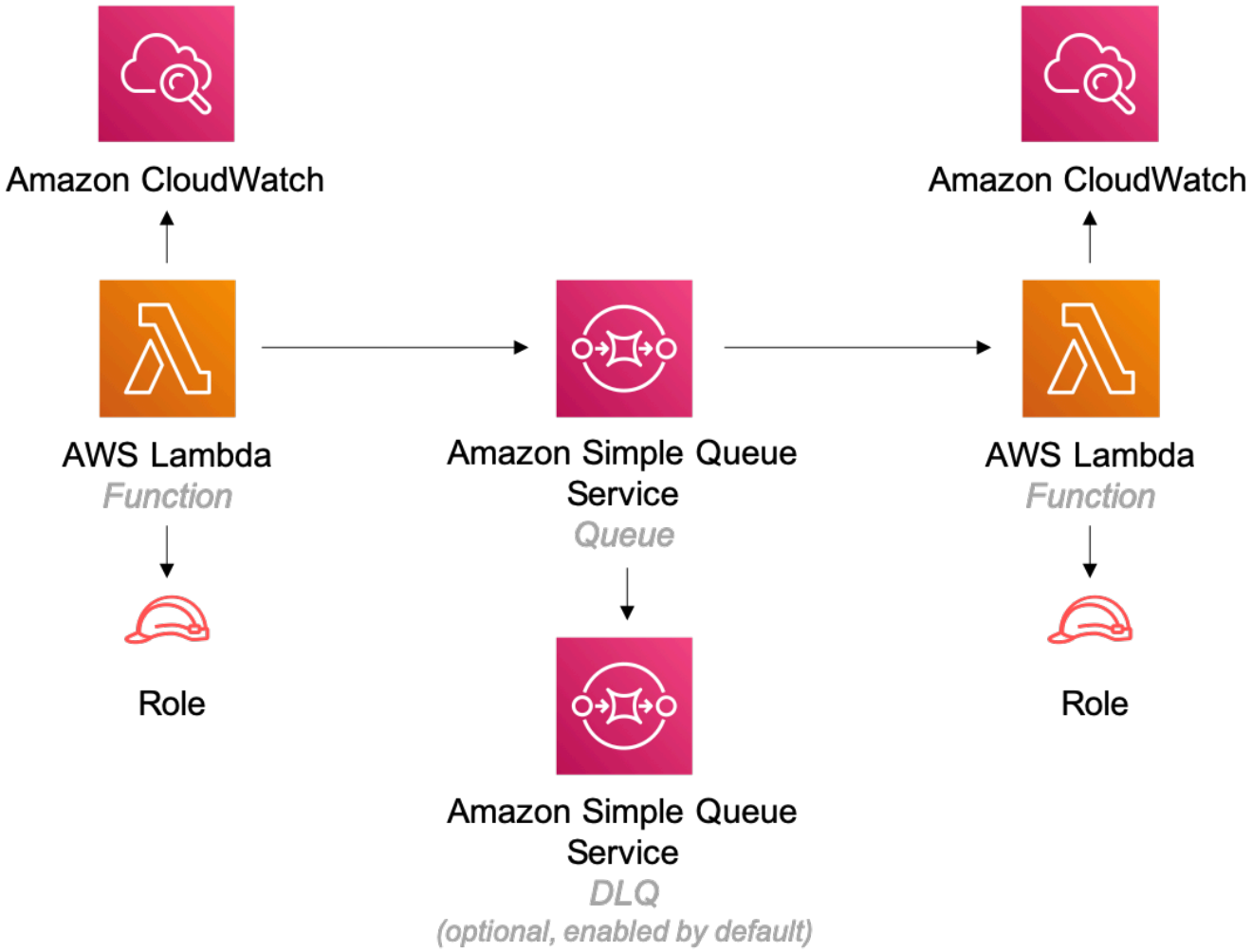
AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적 사용
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Amazon SQS 대기열

- 기본 대기열에 대한 배달 못한 편지 대기열을 배포합니다.
- AWS 관리형 KMS 키를 사용하여 기본 대기열에 대해 서버 측 암호화를 활성화합니다.
- 전송 중인 데이터의 암호화 강제 시행

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws - 람다 - sqs-람다

aws-람다 단계 함수

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_lambda_step_function`
타이프 스크립트	`@aws-solutions-constructs/aws-lambda-step-function`
Java	`software.amazon.awsconstructs.services.lambdastepfunction`

Overview

이 AWS 솔루션 구성은 AWS 스텝 함수에 연결된 AWS Lambda 함수를 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { LambdaToStepFunction } from '@aws-solutions-constructs/aws-lambda-step-
function';
import * as stepfunctions from '@aws-cdk/aws-stepfunctions';

const startState = new stepfunctions.Pass(this, 'StartState');

new LambdaToStepFunction(this, 'LambdaToStepFunctionPattern', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
```

```
    },
    stateMachineProps: {
      definition: startState
    }
});
```

Initializer

```
new LambdaToStepFunction(scope: Construct, id: string, props:
 LambdaToStepFunctionProps);
```

파라미터

- scopeConstruct
- idstring
- propsLambdaToStepFunctionProps

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 오류가 발생합니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 제공될 예정입니다.
상태 머신소품	sfn.StateMachineProps	사용자가 SFN.상태 시스템에 대한 소품을 제공했습니다.

이름	유형	설명
클라우드왓찰암스 만들기	boolean	권장 CloudWatch 경보를 생성할지 여부입니다.
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 로그 그룹의 기본 소품을 무시하기 위한 선택적 사용자 제공 소품입니다.
상태시스템환경변수이름	string	생산자 Lambda 함수에 대한 Step Functions 상태 시스템 환경 변수 세트의 선택적 이름입니다.

패턴 속성

이름	유형	설명
클라우드왓찰암즈?	cloudwatch.Alarm[]	패턴에 의해 생성된 하나 이상의 CloudWatch 경보 목록을 반환합니다.
람다함수	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
스테이트머신	sfn.StateMachine	패턴에 의해 생성 된 상태 머신의 인스턴스를 돌려줍니다.
상태시스템로그 그룹	logs.LogGroup	상태 시스템의 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

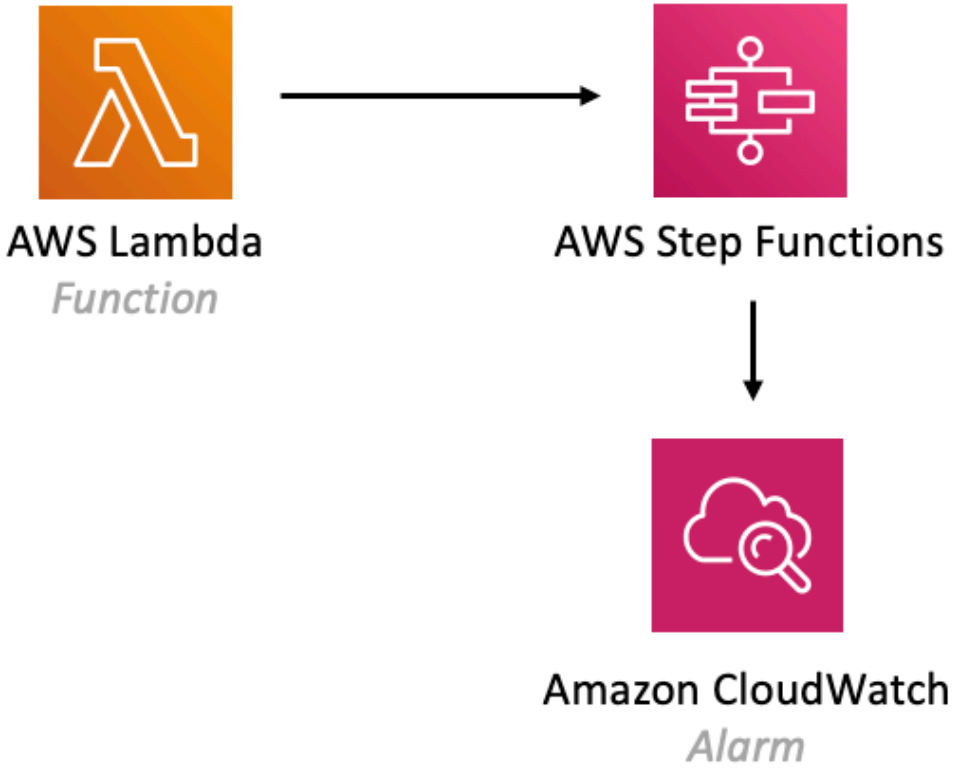
AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `STATE_MACHINE_ARN` (default)
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

AWS Step Functions

- AWS Step Functions 상태 시스템에 대한 모범 사례 CloudWatch 경보를 배포합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/aws - 람다 - 단계 함수

aws-s3-람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_s3_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-s3-lambda`
Java	`software.amazon.awsconstructs.services.s3lambda`

Overview

이 AWS 솔루션 구조는 AWS Lambda 함수에 연결된 Amazon S3 버킷을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { S3ToLambdaProps, S3ToLambda } from '@aws-solutions-constructs/aws-s3-lambda';

new S3ToLambda(this, 'test-s3-lambda', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    },
});
```

Initializer

```
new S3ToLambda(scope: Construct, id: string, props: S3ToLambdaProps);
```

파라미터

- scopeConstruct
- idstring
- propsS3ToLambdaProps

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 오류가 발생합니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 제공될 예정입니다.
버켓토비 기존에?	s3.Bucket	S3 버킷 객체의 기존 인스턴스입니다. 이것이 제공되는 경우bucketProps 는 오류입니다.
버킷 소품?	s3.BucketProps	버킷의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니

이름	유형	설명
		다.existingBucketObj 제공될 예정입니다.
S3이벤트 소품?	S3EventSourceProps	S3EventsOurceProps의 기본 소품을 재정의하는 선택적 사용자 제공 소품

패턴 속성

이름	유형	설명
람다함수	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
S3Bucket?	s3.Bucket	패턴에 의해 생성된 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	s3.Bucket	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon S3 버킷

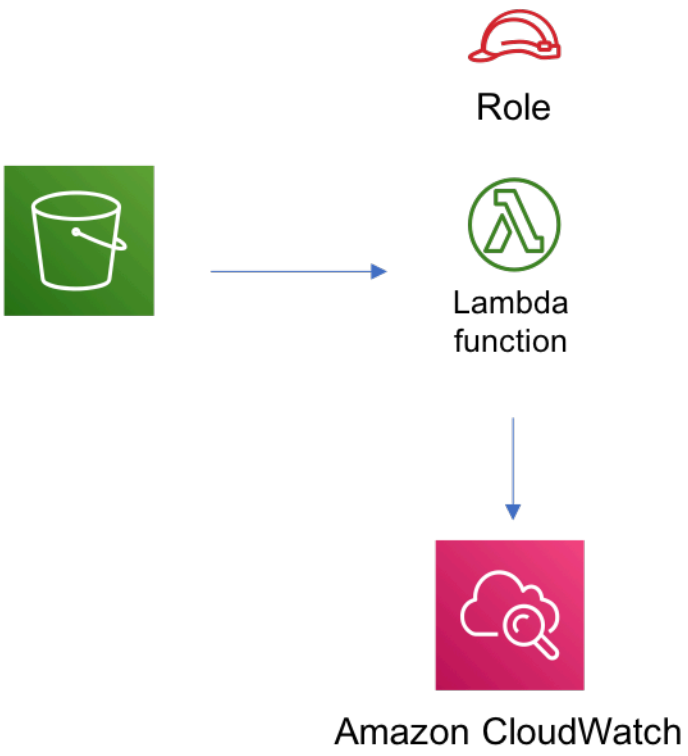
- S3 버킷에 대한 액세스 로깅을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화를 활성화합니다.
- S3 버킷의 버전 관리를 켭니다.
- S3 버킷에 대한 공용 액세스를 허용하지 않습니다.
- CloudFormation 스택을 삭제할 때 S3 버킷을 유지합니다.
- 전송 중인 데이터의 암호화를 강제 시행.

- 90일 후에 최신 버전이 아닌 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws-s3-람다

AWS-s3-sqs

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_s3_sqs`
타이프 스크립트	`@aws-solutions-constructs/aws-s3-sqs`
Java	`software.amazon.awsconstructs.services.s3sqs`

Overview

이 AWS 솔루션 구조는 Amazon SQS 대기열에 알림을 보내도록 구성된 Amazon S3 버킷을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { S3ToSqs } from "@aws-solutions-constructs/aws-s3-sqs";

new S3ToSqs(stack, 'S3ToSQSPattern', {});
```

Initializer

```
new S3ToSqs(scope: Construct, id: string, props: S3ToSqsProps);
```

파라미터

- scopeConstruct
- idstring
- propsS3ToSqsProps

패턴 구성

이름	유형	설명
버켓토비 기존에?	s3.Bucket	S3 버킷 객체의 기존 인스턴스입니다. 이것이 제공되는 경우bucketProps 는 오류입니다.
버킷 소품?	s3.BucketProps	S3 버킷의 기본 소품을 재정의하는 선택적 사용자 제공 소품입니다.
S3이벤트 유형?	s3.EventType[]	알림을 트리거할 S3 이벤트 유형입니다. 기본값은 s3.EventType.OBJECT_CREATED 입니다.
S3이벤트 필터?	s3.NotificationKeyFilter[]	S3 객체 키 필터 규칙은 이 이벤트를 트리거하는 객체를 결정합니다. 지정하지 않으면 필터 규칙이 적용되지 않습니다.
대기열에 있는 OBJ?	sqs.Queue	기본 대기열 대신 사용할 기존 SQS 대기열 (선택 사항) 이 두 가지를 모두 제공queueProp

이름	유형	설명
		s 를 초과하면 오류가 발생합니다. SQS 대기열이 암호화되면 암호화에 사용되는 KMS 키는 고객이 관리하는 CMK여야 합니다.
대기열 소품?	sqs.QueueProps	SQS 대기열의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 이 인 경우에는 무시됩니다.existingQueueObj 이 제공될 예정입니다.
데드레터 대기열Props?	sqs.QueueProps	데드 레터 큐의 기본 소품을 재정의하는 선택적 사용자가 제공 한 소품. 에만 사용됩니다.deployDeadLetterQueue 속성이 true로 설정됩니다.
배포데드 레터 큐?	boolean	배달 못한 편지 대기열로 사용될 보조 대기열을 생성할지 여부. 기본값은 true입니다.
maxReceiveCount?	number	배달 못한 편지 대기열로 이동되기 전에 메시지가 대기열에서 빠질 수 있는 횟수입니다. 기본값은 15입니다.
고객 관리 키를 사용하여 암호화를 활성화하시겠습니까?	boolean	이 CDK 앱에서 관리하거나 가져온 KMS 키를 사용할지 여부 암호화 키를 가져오는 경우 암호화 키를encryptionKey 이 구문에 대한 속성입니다.

이름	유형	설명
encryptionKey	kms.Key	기본 암호화 키 대신 사용할 선택적 기존 암호화 키입니다.
암호화 키 소품?	kms.KeyProps	암호화 키의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.

패턴 속성

이름	유형	설명
분류: SQqueue	sqs.Queue	패턴에 의해 생성된 SQS 대기열의 인스턴스를 반환합니다.
데드 레터 큐?	sqs.Queue	패턴에 의해 생성 된 데드 레터 큐의 인스턴스를 돌려줍니다 (배포 된 경우).
encryptionKey	kms.IKey	패턴에 의해 생성 된 암호화 키의 인스턴스를 돌려줍니다.
S3Bucket?	s3.Bucket	패턴에 의해 생성된 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	s3.Bucket	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon S3 버킷

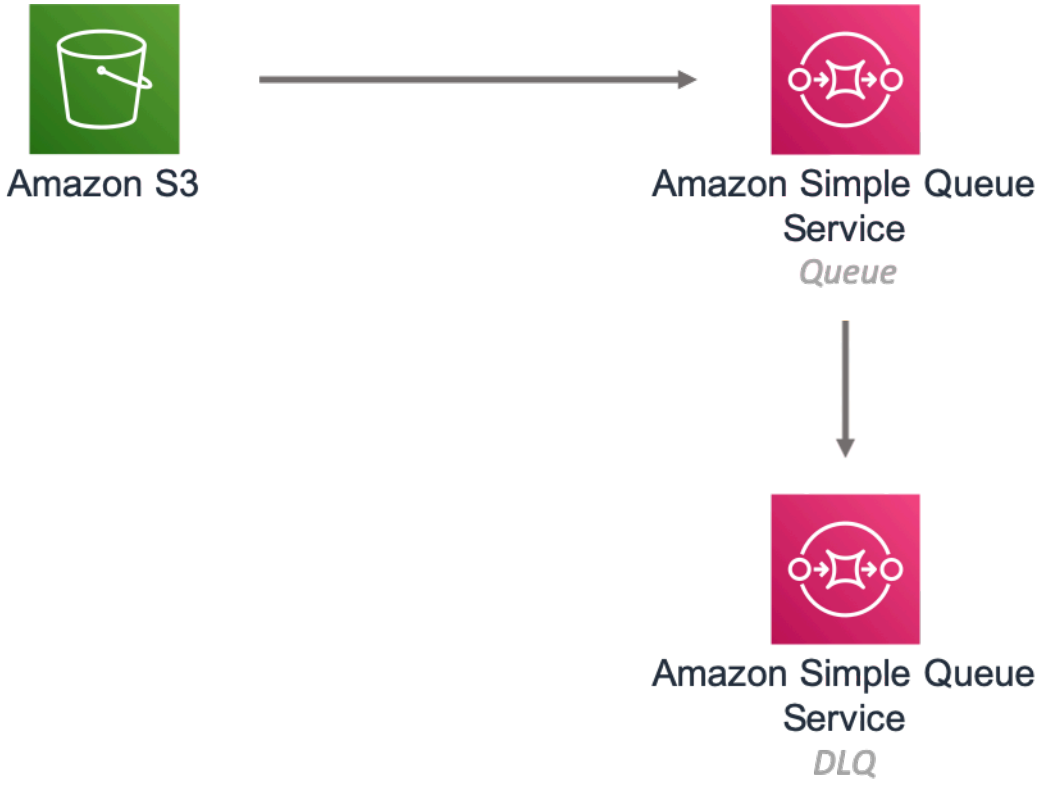
- S3 버킷에 대한 액세스 로깅 구성

- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화 활성화
- S3 버킷에 대한 버전 관리 켜기
- S3 버킷에 대한 공용 액세스 허용 안 함
- CloudFormation 스택을 삭제할 때 S3 버킷 유지
- 전송 중인 데이터의 암호화 강제 시행
- 90일 후에 비최신 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

Amazon SQS 대기열

- SQS 대기열에 대한 최소 권한 액세스 권한 구성
- 소스 SQS 대기열에 대한 SQS 배달 못한 편지 대기열 배포
- 고객 관리 KMS 키를 사용하여 SQS 대기열을 위한 서버 측 암호화 활성화
- 전송 중인 데이터의 암호화 강제 시행

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.	
	@aws -솔루션 - 구성/AWS-s3-sqs

AWS-s3-스텝 기능

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 반환합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_s3_step_function`
타이프 스크립트	`@aws-solutions-constructs/aws-s3-step-function`
Java	`software.amazon.awsconstructs.services.s3stepfunction`

Overview

이 AWS 솔루션 구성은 AWS 단계 함수에 연결된 Amazon S3 버킷을 구현합니다.

> **Note**
>
> 이 구조는 아마존 EventBridge (Amazon CloudWatch Events) 를 사용하여 AWS Step Functions 트리거합니다. EventBridge 는 유연성이 뛰어나지만 S3 이벤트 알림으로 Step Functions 트리거하면 지연 시간이 줄어들고 비용 효율성이 높아집니다. 비용 및/또는 대기 시간이 문제인 경우`aws-s3-lambda`및`aws-lambda-stepfunctions`이 구조 대신.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { S3ToStepFunction, S3ToStepFunctionProps } from '@aws-solutions-constructs/aws-s3-step-function';
import * as stepfunctions from '@aws-cdk/aws-stepfunctions';

const startState = new stepfunctions.Pass(this, 'StartState');

new S3ToStepFunction(this, 'test-s3-step-function-stack', {
    stateMachineProps: {
      definition: startState
    }
});
```

Initializer

```
new S3ToStepFunction(scope: Construct, id: string, props: S3ToStepFunctionProps);
```

파라미터

- scope`Construct`
- id`string`
- props`S3ToStepFunctionProps`

Prop 패턴 구성

이름	유형	설명
버켓토비 기존에?	s3.IBucket	S3 버킷 객체의 기존 인스턴스입니다. 이것이 제공되는 경우bucketProps 는 오류입니다.
버킷 소품?	s3.BucketProps	버킷의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 이면 무시됩니다.existingBucketObj 가 제공될 것입니다.
상태 머신소품	sfn.StateMachineProps	선택적 사용자가 SFN.State Machine의 기본 소품을 무시하기 위해 소품을 제공했습니다.
이벤트 소품?	events.RuleProps	선택적 사용자가 기본값을 재정의하기 위해 EventRule Props를 제공했습니다.
배포클라우드트레일?	boolean	Amazon S3 API 이벤트를 기록하기 위해 AWS CloudTrail 에 트레일을 배포할지 여부 기본값은 true입니다.
클라우드왓찰암스 만들기	boolean	권장 CloudWatch 경보를 생성할지 여부입니다.
로그그룹Props?	logs.LogGroupProps	CloudWatch Logs 로그 그룹의 기본 소품을 재정의하기 위한 선택적 사용자 제공 prop입니다.

패턴 속성

이름	유형	설명
CloudTrail?	`cloudtrail.Trail`	패턴에 의해 생성 된 Cloudtrail 트레일의 인스턴스를 돌려줍니다.
클라우드 트레일버킷?	`s3.Bucket`	Cloudtrail 데이터를 저장하기 위한 패턴으로 생성된 버킷의 인스턴스를 반환합니다.
클라우드 트레일로깅 버킷?	`s3.Bucket`	Cloudtrail 추적에서 사용하는 기본 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.
CloudWatCalarms?	`cloudwatch.Alarm[]`	패턴에 의해 생성된 하나 이상의 CloudWatch 경보 목록을 반환합니다.
S3Bucket?	`s3.Bucket`	패턴에 의해 생성된 S3 버킷의 인스턴스를 반환합니다.
s3로깅 버킷?	`s3.Bucket`	S3 버킷의 패턴으로 생성된 로깅 버킷의 인스턴스를 반환합니다.
스테이트머신	`sfn.StateMachine`	패턴에 의해 생성 된 상태 머신의 인스턴스를 돌려줍니다.
상태시스템로그 그룹	`logs.LogGroup`	상태 시스템의 패턴으로 생성된 로그 그룹의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon S3 버킷

- S3 버킷에 대한 액세스 로깅을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 S3 버킷에 대한 서버 측 암호화를 활성화합니다.
- S3 버킷의 버전 관리를 켭니다.
- S3 버킷에 대한 공용 액세스를 허용하지 않습니다.
- CloudFormation 스택을 삭제할 때 S3 버킷을 유지합니다.
- 전송 중인 데이터의 암호화를 적용합니다.
- 90일 후에 비최신 객체 버전을 Glacier 스토리지로 이동하는 수명 주기 규칙을 적용합니다.

AWS CloudTrail

- AWS CloudTrail에서 트레일을 구성하여 구성에 의해 생성된 버킷과 관련된 Amazon S3 API 이벤트를 기록합니다.

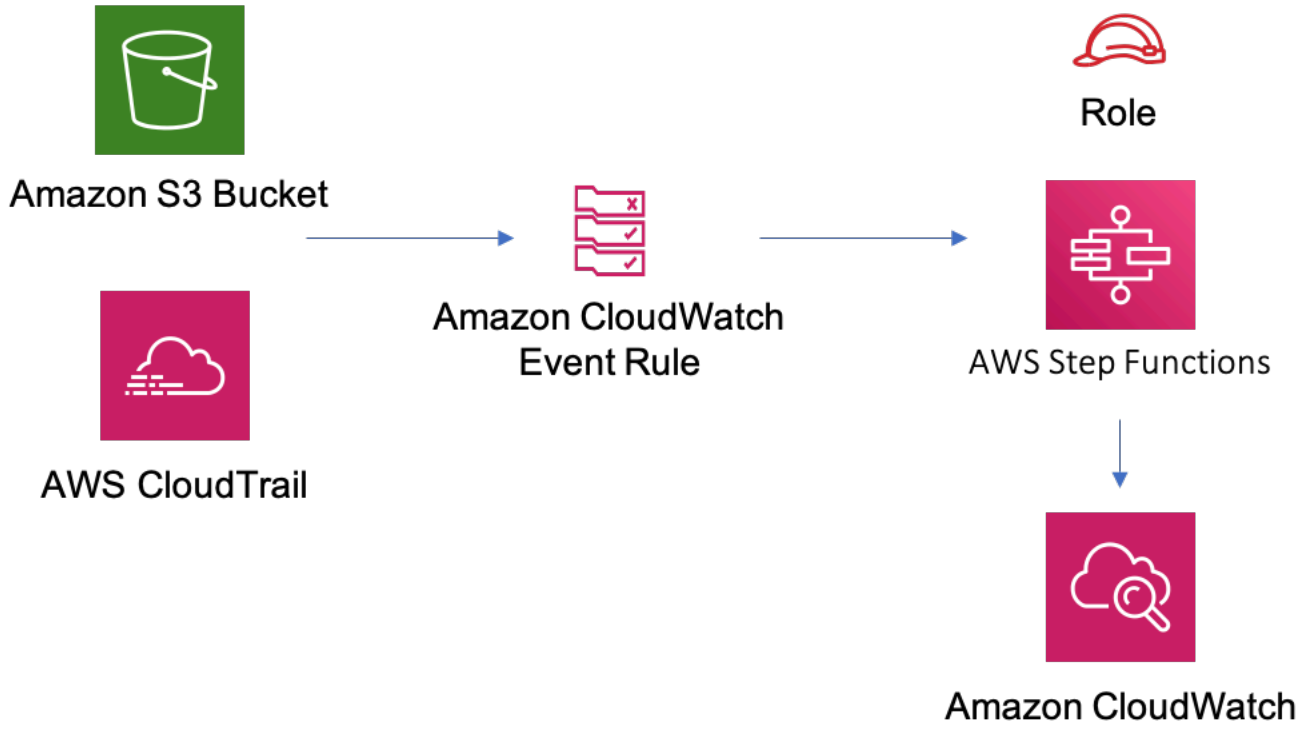
Amazon CloudWatch Events

- CloudWatch 이벤트에 최소 권한을 부여하여 Lambda 함수를 트리거합니다.

AWS Step Fun

- API Gateway 대해 CloudWatch 로깅을 사용하도록 설정합니다.
- 단계 기능에 대한 모범 사례 CloudWatch 경보를 배포합니다.

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS-s3 단계 함수

람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_sns_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-sns-lambda`
Java	`software.amazon.awsconstructs.services.snslambda`

Overview

이 AWS 솔루션 구성은 AWS Lambda 함수에 연결된 Amazon SNS 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { SnsToLambda, SnsToLambdaProps } from "@aws-solutions-constructs/aws-sns-
lambda";

new SnsToLambda(this, 'test-sns-lambda', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

Initializer

```
new SnsToLambda(scope: Construct, id: string, props: SnsToLambdaProps);
```

파라미터

- scopeConstruct
- idstring
- propsSnsToLambdaProps

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 사용하면 오류가 발생할 수 있습니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 의 경우에는 무시됩니다.existingLambdaObj 제공될 예정입니다.
기존토픽코비?	sns.Topic	SNS Topic 객체의 기존 인스턴스로,topicProps 를 사용하면 오류가 발생할 수 있습니다.
주제소품?	sns.TopicProps	SNS 주제의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.

패턴 속성

이름	유형	설명
Lambda함수	`lambda.Function`	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
snsTopic	`sns.Topic`	패턴에 의해 생성된 SNS 주제의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

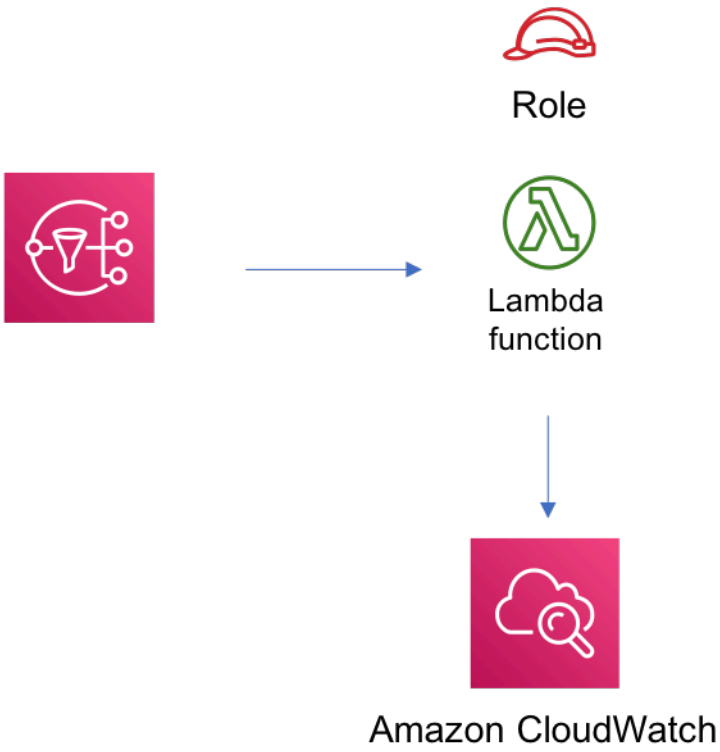
Amazon SNS 주제

- SNS 주제에 대한 최소 권한 액세스 권한을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 서버 측 암호화가 활성화됩니다.
- 전송 중인 데이터의 암호화 강제 시행

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws - SNS - 람다

AWS-SNS-스퀘어

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전 관리모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_sns_sqs`
타이프 스크립트	`@aws-solutions-constructs/aws-sns-sqs`
Java	`software.amazon.awsconstructs.services.snssqs`

Overview

Amazon SQS 대기열에 연결된 Amazon SNS 주제를 구현하는 AWS 솔루션 구성입니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
import { SnsToSqs, SnsToSqsProps } from "@aws-solutions-constructs/aws-sns-sqs";
import * as iam from '@aws-cdk/aws-iam';

const snsToSqsStack = new SnsToSqs(this, 'SnsToSqsPattern', {});

// Grant yourself permissions to use the Customer Managed KMS Key
const policyStatement = new iam.PolicyStatement({
    actions: ["kms:Encrypt", "kms:Decrypt"],
    effect: iam.Effect.ALLOW,
    principals: [ new iam.AccountRootPrincipal() ],
    resources: [ "*" ]
});

snsToSqsStack.encryptionKey?.addToResourcePolicy(policyStatement);
```

Initializer

```
new SnsToSqs(scope: Construct, id: string, props: SnsToSqsProps);
```

파라미터

- scopeConstruct
- idstring
- propsSnsToSqsProps

패턴 구성

이름	유형	설명
기존토픽코비?	sns.Topic	SNS Topic 객체의 기존 인스턴스로,topicProps 오류가 발생합니다.
주제소품?	sns.TopicProps	SNS 주제에 대한 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 이 인 경우에는 무시됩니다.existingTopicObj 제공될 예정입니다.
대기열에 있는 Obj?	sqs.Queue	기본 대기열 대신 사용할 기존 SQS 대기열 (선택 사항) 이 두 가지를 모두 제공queueProps 오류가 발생합니다.
대기열 소품?	sqs.QueueProps	SQS 대기열의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 이 인 경우에는 무시됩니다.existingQueueObj 제공될 예정입니다.
배포데드 레터 큐?	boolean	배달 못한 편지 대기열로 사용할 보조 대기열을 생성할지

이름	유형	설명
		여부를 지정합니다. 기본값은 `true`입니다.
데드레터 대기열Props?	`sqs.QueueProps`	데드 레터 큐의 기본 소품을 재정의하는 선택적 사용자 제공 소품. 에만 사용됩니다.`deployDeadLetterQueue` 속성이 true로 설정됩니다.
maxReceiveCount?	`number`	배달 못한 편지 대기열로 이동되기 전에 메시지가 대기열에서 빼지 못할 수 있는 횟수입니다. 기본값은 `15`입니다.
고객 관리 키를 사용하여 암호화를 활성화하시겠습니까?	`boolean`	이 CDK 앱에서 관리하거나 가져온 고객 관리 암호화 키를 사용할지 여부 암호화 키를 가져오는 경우 암호화 키를`encryptionKey` 이 구문에 대한 속성입니다.
encryptionKey?	`kms.Key`	기본 암호화 키 대신 사용할 선택적 기존 암호화 키입니다.
암호화 키 소품?	`kms.KeyProps`	암호화 키의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다.

패턴 속성

이름	유형	설명
snsTopic	`sns.Topic`	패턴에 의해 생성된 SNS 주제의 인스턴스를 반환합니다.

이름	유형	설명
encryptionKey	`kms.Key`	패턴에 의해 생성 된 암호화 키의 인스턴스를 돌려줍니다.
분류: SQqueue	`sqs.Queue`	패턴에 의해 생성된 SQS 대기열의 인스턴스를 반환합니다.
데드 레터 큐?	`sqs.Queue`	패턴에 의해 생성 된 데드 레터 큐의 인스턴스를 돌려줍니다 (배포 된 경우).

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

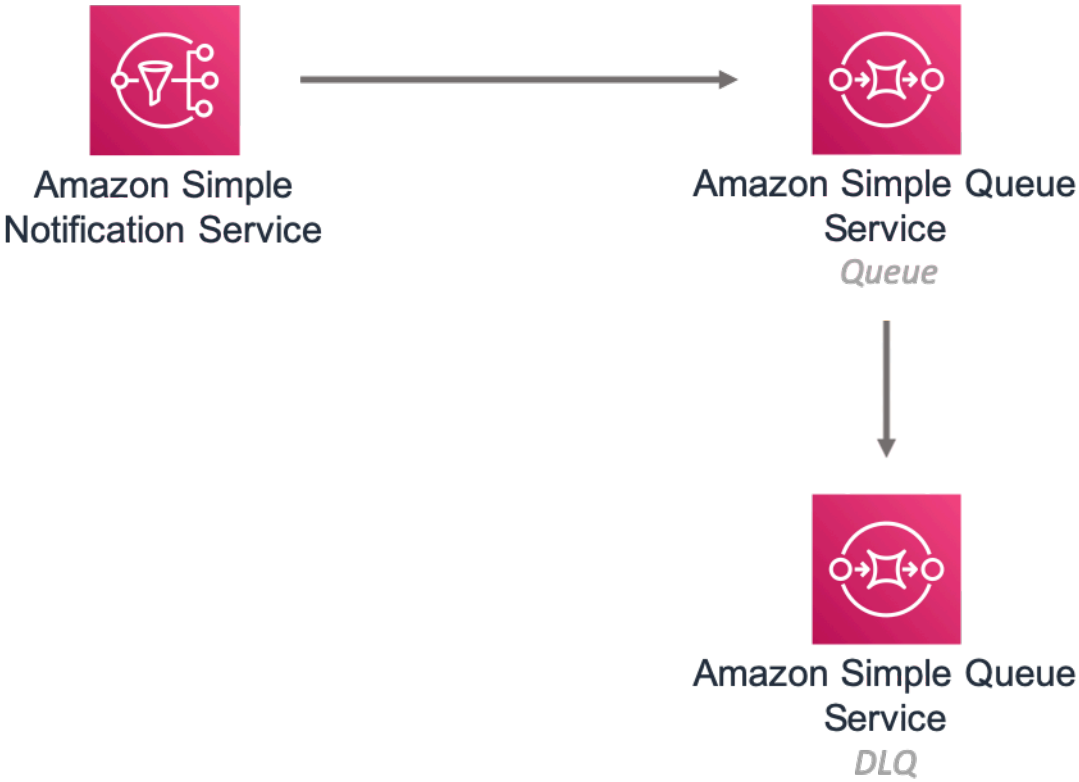
Amazon SNS 주제

- SNS 주제에 대한 최소 권한 액세스 권한을 구성합니다.
- AWS 관리형 KMS 키를 사용하여 서버 측 암호화를 활성화합니다.
- 전송 중인 데이터의 암호화를 강제 시행

Amazon SQS 대기열

- SQS 대기열에 대한 최소 권한 액세스 권한을 구성합니다.
- 소스 SQS 대기열에 대한 배달 못한 편지 대기열을 배포합니다.
- 고객 관리 KMS 키를 사용하여 SQS 대기열에 대한 서버 측 암호화를 활성화합니다.
- 전송 중인 데이터의 암호화를 강제 시행

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/AWS - SNS - 스퀘어

aws-sqs-람다

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델을 선택합니다. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

참고: 적절한 기능을 보장하려면 프로젝트의 AWS 솔루션 구성 패키지와 AWS CDK 패키지가 동일한 버전이어야 합니다.

언어	패키지
Python	`aws_solutions_constructs.aws_sqs_lambda`
타이프 스크립트	`@aws-solutions-constructs/aws-sqs-lambda`
Java	`software.amazon.awsconstructs.services.sqslambda`

Overview

이 AWS 솔루션 구성체는 AWS Lambda 함수에 연결된 Amazon SQS 대기열을 구현합니다.

다음은 TypeScript 터의 최소 배포 가능한 패턴 정의입니다.

```
const { SqsToLambda } = require('@aws-solutions-constructs/aws-sqs-lambda');

new SqsToLambda(stack, 'SqsToLambdaPattern', {
    lambdaFunctionProps: {
        runtime: lambda.Runtime.NODEJS_14_X,
        // This assumes a handler function in lib/lambda/index.js
        code: lambda.Code.fromAsset(`${__dirname}/lambda`),
        handler: 'index.handler'
    }
});
```

Initializer

```
new SqsToLambda(scope: Construct, id: string, props: SqsToLambdaProps);
```

파라미터

- scopeConstruct
- idstring
- propsSqsToLambdaProps

패턴 구성

이름	유형	설명
람다오브즈 기존인가요?	lambda.Function	Lambda 함수 객체의 기존 인스턴스, 이 및lambdaFunctionProps 를 초과하면 오류가 발생합니다.
람다기능소품?	lambda.FunctionProps	Lambda 함수의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 이 인 경우에는 무시됩니다.existingLambdaObj 이 제공될 예정입니다.
대기열에 있는 Obj?	sqs.Queue	기본 대기열 대신 사용할 기존 SQS 대기열 (선택 사항) 이 두 가지를 모두 제공queueProps 를 초과하면 오류가 발생합니다.
대기열 소품?	sqs.QueueProps	SQS 대기열의 기본 속성을 재정의하는 선택적 사용자 제공 속성입니다. 이 인 경우에는 무시됩니다.existingQueueObj 이 제공될 예정입니다.
배포데드 레터 큐?	boolean	배달 못한 편지 대기열로 사용할 보조 대기열을 생성할지 여부를 지정합니다. 기본값은 true입니다.

이름	유형	설명
데드레터 대기열Props?	sqs.QueueProps	데드 레터 큐의 기본 소품을 재정의하는 선택적 사용자 제공 소품. 에만 사용됩니다.deployDeadLetterQueue 속성이 true로 설정됩니다.
maxReceiveCount?	number	배달 못한 편지 대기열로 이동되기 전에 메시지가 대기열에서 빼지 못한 횟수입니다. 기본값은 15입니다.

패턴 속성

이름	유형	설명
데드 레터 큐?	sqs.Queue	패턴에 의해 생성 된 데드 레터 큐의 인스턴스를 돌려줍니다 (배포 된 경우).
람다함수	lambda.Function	패턴에 의해 생성 된 Lambda 함수의 인스턴스를 돌려줍니다.
분류: SQqueue	sqs.Queue	패턴에 의해 생성된 SQS 대기열의 인스턴스를 반환합니다.

기본 설정

재정의없이이 패턴을 즉시 구현하면 다음과 같은 기본값이 설정됩니다.

Amazon SQS 대기열

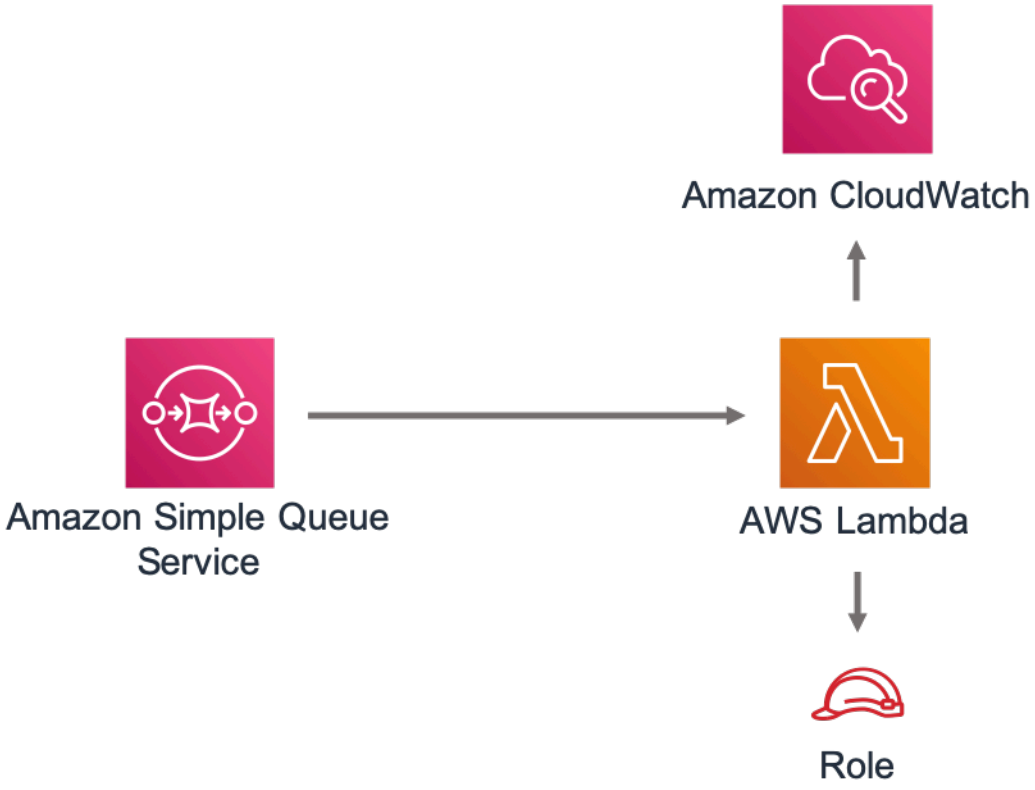
- 소스 SQS 대기열에 대한 SQS 배달 못한 편지 대기열을 배포합니다.
- AWS 관리형 KMS 키를 사용하여 소스 SQS 대기열로 서버 측 암호화를 활성화합니다.

- 전송 중인 데이터의 암호화를 강제 시행.

AWS Lambda 함수

- Lambda 함수에 대한 제한된 권한 액세스 IAM 역할을 구성합니다.
- NodeJS Lambda 함수에 대한 연결 유지와 연결을 재사용 할 수 있습니다.
- X-Ray 추적을 활성화합니다.
- 환경 변수를 설정합니다.
  - `AWS_NODEJS_CONNECTION_REUSE_ENABLED`(노드 10.x 이상 함수의 경우)

Architecture



GitHub

이 패턴의 코드를 보려면 문제 및 끌어오기 요청을 작성/조회하는 등의 작업을 수행합니다.

	@aws -솔루션 - 구성/aws-sqs-람다

core

STABILITY EXPERIMENTAL

모든 클래스는 활발히 개발 중이며 향후 버전에서 이전 버전과 호환되지 않는 변경 또는 제거 될 수 있습니다. 이들은 적용되지 않습니다의미 체계 버전모델. 즉, 이 패키지를 사용할 수도 있지만 이 패키지의 최신 버전으로 업그레이드할 때 소스 코드를 업데이트해야 할 수도 있습니다.

핵심 라이브러리에는 AWS 솔루션 구성의 기본 구성 요소가 포함되어 있습니다. 나머지 AWS 솔루션 구조에서 사용되는 핵심 클래스를 정의합니다.

AWS CDK 구조의 기본 속성

코어 라이브러리는 AWS 솔루션 구조에서 사용하는 AWS CDK 구조의 기본 속성을 설정합니다.

예를 들어, 다음은 AWS 솔루션 구조 구조에 의해 생성된 S3 버킷 구조의 기본 속성 조각입니다. 기본적으로 서버 측 암호화, 버킷 버전 관리를 설정하고 모든 공용 액세스를 차단하며 S3 액세스 로깅을 설정합니다.

```
{
  encryption: s3.BucketEncryption.S3_MANAGED,
  versioned: true,
  blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
  removalPolicy: RemovalPolicy.RETAIN,
  serverAccessLogsBucket: loggingBucket
}
```

기본 속성 재정의

코어 라이브러리에 의해 설정된 기본 속성은 사용자 제공 속성에 의해 재정의 될 수 있습니다. 예를 들어 사용자는 특정 요구 사항을 충족하기 위해 Amazon S3 퍼블릭 액세스 차단 속성을 재정의할 수 있습니다.

```
const stack = new cdk.Stack();

const props: CloudFrontToS3Props = {
  bucketProps: {
    blockPublicAccess: {
```

```
        blockPublicAcls: false,
        blockPublicPolicy: true,
        ignorePublicAcls: false,
        restrictPublicBuckets: true
      }
    }
  };

  new CloudFrontToS3(stack, 'test-cloudfront-s3', props);

  expect(stack).toHaveResource("AWS::S3::Bucket", {
    PublicAccessBlockConfiguration: {
      BlockPublicAcls: false,
      BlockPublicPolicy: true,
      IgnorePublicAcls: false,
      RestrictPublicBuckets: true
    },
  });
```

특성 재지정 경고

코어 라이브러리의 기본 속성이 사용자가 제공 한 속성에 의해 재정의되면 구문은 변경 사항을 강조 표시하는 하나 이상의 경고 메시지를 콘솔에 내 보냅니다. 이러한 메시지는 사용자에게 상황 인식을 제공하고 보안 위험을 초래할 수 있는 의도하지 않은 재정의를 방지하기 위한 것입니다. 이러한 메시지는 배포/빌드 관련 명령이 실행될 때마다 나타납니다.`cdk deploy`,`cdk synth`,`npm test`등.

예제 메시지`AWS_CONSTRUCTS_WARNING: An override has been provided for the property: BillingMode. Default value: 'PAY_PER_REQUEST'. You provided: 'PROVISIONED'.`

재지정 경고 전환

무시 경고 메시지는 기본적으로 활성화되어 있지만`overrideWarningsEnabled`shell 변수를 지정합니다.

- 명시적으로끄기경고 무시, 실행`export overrideWarningsEnabled=false`.
- 명시적으로켜기경고 무시, 실행`export overrideWarningsEnabled=true`.
- 기본 값으로 되돌리려면`unset overrideWarningsEnabled`.

문서 수정

AWS 솔루션 구성 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

update-history-change	update-history-description	update-history-date
업데이트 내용	AWS-람다-ssm문자열 매개 변수 패턴이 추가되었습니다. 기타 마이너 콘텐츠 업데이트.	2021년 5월 27일
업데이트 내용	aws-람다 - 비밀 관리자 패턴을 추가했습니다. 기타 마이너 콘텐츠 업데이트.	2021년 5월 12일
업데이트 내용	* - 람다 패턴을 선택하기 위해 속성이 업데이트됩니다. 기타 마이너 콘텐츠 업데이트.	2021년 4월 17일
업데이트 내용	Python 사용자를 위한 연습 문제 및 Lambda 함수를 포함하는 구문에 대한 업데이트된 속성 예제 문제를 수정했습니다.	2021년 3월 30일
업데이트 내용	패턴 소품 및 선택 패턴의 기본 설정에 대한 사소한 수정/업데이트.	2021년 3월 8일
업데이트 내용	보행시선 컨텐츠에 대한 사소한 수정/업데이트.	2021년 3월 4일
업데이트 내용	추가됨`aws-lambda-sagemakerendpoint` 패턴과 선택한 Kinesis 파이어호스 패턴의 속성을 업데이트했습니다.	2021년 2월 24일
업데이트 내용	추가됨`aws-kinesisstreams-gluejob` 패턴	2021년 2월 17일

	과 파이썬 사용자를위한 업데이트 된 연습 단계.	
업데이트 내용	에 대한 등록 정보 업데이트`aws-cloudfront-*` 패턴.	2021년 2월 9일
업데이트 내용	각 패턴에 대해 GitHub 에 대한 링크가 추가되었습니다.	2021년 2월 5일
업데이트 내용	선택 패턴의 특성이 업데이트되었습니다.	2021년 2월 1일
업데이트 내용	선택한 패턴의 특성 및 기본 설정에 대한 문서가 업데이트되었습니다.	2021년 1월 4일
업데이트 내용	AWS-클라우드 프런트 미디어 스토어 및 AWS-s3-sqs라는 새로운 패턴이 추가되었습니다.	2020년 12월 2일
업데이트 내용	aws-람다 - 세이지메이커 패턴을 제거했습니다.	2020년 11월 2일
업데이트 내용	AWS-이벤트-규칙-키네시스스트림, aws-이벤트-규칙-키네시스파이어호스-S3, aws-람다 세이지메이커와 같은 새로운 패턴이 추가되었습니다.	2020년 10월 27일
업데이트 내용	aws-events-rule-sns 및 aws-events-rule-sqs 패턴의 주요 변화를 반영하도록 업데이트되었습니다. 클래스 및 인터페이스 이름이 파스칼 케이스로 변경되었습니다.	2020년 10월 22일

업데이트 내용	aws-apigatewa-세이지메이크렌드포인트 및 AWS-키네시스스트림-키니시스파이어호스-S3 패턴 추가, 기존 컨텐츠에 대한 기타 사소한 업데이트	2020년 10월 20일
업데이트 내용	aws-apigateway IoT 패턴 추가, 기존 콘텐츠에 대한 기타 사소한 업데이트.	2020년 10월 7일
업데이트 내용	모든 패턴에 대한 배포 가능한 최소 패턴 코드 조각과 모범 사례 기본값을 업데이트했습니다.	2020년 10월 5일
업데이트 내용	주요 변경 사항을 반영하기 위해 aws-kinesisstream-람다 패턴의 속성이 업데이트되었습니다.	2020년 9월 14일
업데이트 내용	보행시선의 두 번째 부분에 대한 사소한 수정.	2020년 9월 10일
업데이트 내용	aws-apigateway - 키네시스스트림, aws-이벤트-규칙-sns 및 aws-이벤트-규칙-sqs 패턴이 추가되었습니다.	2020년 9월 10일
업데이트 내용	aws-sns-sqs 패턴 추가, 모든 SNS 패턴 업데이트, 사소한 인쇄상의 수정.	2020년 9월 2일
업데이트 내용	aws-sqs-람다 패턴의 고정 모듈 이름.	2020년 8월 31일

업데이트 내용	aws-Dynamodb-스트림 람다-탄성 검색-키바나 패턴에 대한 파이썬 모듈 이름이 수정되었습니다.	2020년 8월 31일
업데이트 내용	Lambda 패턴에 대한 기본값 업데이트; 기타 사소한 업데이트.	2020년 8월 27일
업데이트 내용	S3 패턴의 공용 속성이 업데이트되었습니다. DynamoDB 패턴의 기본값이 업데이트되었습니다.	2020년 8월 10일
업데이트 내용	전송 중 암호화의 기본 적용을 강조하기 위해 여러 패턴을 업데이트했습니다.	2020년 8월 4일
업데이트 내용	aws-lambda-sqs-lambda 패턴 추가, 시작 안내서의 향상된 구성 지침, 공용 속성을 통해 추가 리소스를 사용할 수 있도록 모든 패턴을 업데이트했습니다.	2020년 7월 27일
업데이트 내용	aws-lambda-sqs 패턴 추가; 기타 사소한 업데이트.	2020년 7월 20일
업데이트 내용	관련 패턴에서 DeployLambda 및 DeployBucket 속성을 제거했습니다. 기타 사소한 업데이트입니다.	2020년 7월 9일
업데이트 내용	aws-lambda 단계 기능 패턴을 추가하고 사소한 입력 오류를 수정했습니다.	2020년 7월 7일
업데이트 내용	추가된 내용: 속성을 사용하여 DynamoDB 패턴을 선택합니다.	2020년 6월 25일

업데이트 내용	끊어진 링크에 대한 몇 가지 텍스트 수정 및 수정.	2020년 6월 23일
최초 릴리스	AWS 솔루션 구조는 공개적으로 사용할 수 있습니다.	2020년 6월 22일

Notices

고객은 이 문서의 정보를 독립적으로 평가할 책임이 있습니다. 이 문서: (a) 정보 제공 목적으로만 작성되며, (b) 현재 AWS 제품 제공 및 관행을 나타내며, 이는 통지 없이 변경될 수 있으며, (c) AWS와 그 계열사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정이나 보증도 작성하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 “있는 그대로” 제공됩니다. 고객에 대한 AWS의 책임 및 채무는 AWS 계약에 준거합니다. 본 문서는 AWS와 고객 간의 어떠한 계약도 구성하지 않으며 이를 변경하지도 않습니다.

© 2020 Amazon Web Services, Inc. 또는 계열사. All rights reserved.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.