aws

개발자 가이드

# Amazon Location Service

# Amazon Location Service: 개발자 가이드

# Table of Contents

# Amazon Location Service에 오신 것을 환영합니다

Amazon Location Service 개발자 가이드에 오신 것을 환영합니다.

다음 주제는 수행하려는 작업에 따라 설명서를 시작하는 데 도움이 될 수 있습니다.

Amazon Location 개요 보기

- Amazon Location의 개념에 대해 알아보십시오.
- Amazon Location Service 사용 방법 장에서 기능에 대해 더 자세히 알아보십시오.
- Amazon Location 데모 사이트에서 데모 앱을 참조하십시오.
- 이미 AWS 계정을(를) 가지고 있다면 Amazon Location Service 콘솔을 사용하여 기능을 직접 탐색할 수 있습니다.

개발자로서 Amazon Location 사용하기

- 빠른 시작을(를) 사용하여 첫 번째 앱을 빌드하십시오.
- Amazon Location Service 사용 방법 장에서 다양한 Amazon Location Service 기능이 어떻게 작동하는지 알아보십시오.
- Amazon Location Service를 사용한 개발 장에서 사용할 수 있는 SDK 및 도구를 참조하십시오.
- 자체 앱에서 사용할 수 있는 코드 예제 및 자습서를 참조하십시오. Amazon Location 데모 사이트 샘플 페이지를 방문하여 기능, 언어 또는 플랫폼별로 필터링할 수 있는 샘플을 찾을 수도 있습니다.
- Amazon Location API에 대한 자세한 내용은 API 참조 안내서를 참조하십시오.

# Amazon Location Service란 무엇입니까?

Amazon Location Service를 사용하면 맵, 관심 장소, 지오코딩, 라우팅, 지오펜스, 추적 등의 기능을 포함하는 위치 데이터 및 기능을 애플리케이션에 추가할 수 있습니다. Amazon Location은 신뢰할 수 있는 글로벌 공급자인 Esri, Grab, HERE의 고품질 데이터를 사용하여 위치 기반 서비스(LBS)를 제공합니다. 합리적인 가격의 데이터, 추적 및 지오펜싱 기능, 상태 모니터링을 위한 기본 제공 지표를 통해 정교한 위치 기반 애플리케이션을 구축할 수 있습니다.

Amazon Location을 사용하면 조직의 데이터에 대한 제어권을 유지할 수 있습니다. Amazon Location은 고객 메타데이터와 계정 정보를 제거하여 데이터 공급자에게 전송되는 모든 쿼리를 익명화합니다. 또한 시설, 자산 및 직원 위치와 같은 민감한 추적 및 지오펜싱 위치 정보는 AWS 계정을 전혀 벗어나지 않습니다. 이를 통해 제3자로부터 민감한 정보를 보호하고, 사용자 개인 정보를 보호하고, 애플리케이

션의 보안 위험을 줄일 수 있습니다. Amazon Location을 사용하면 Amazon 및 제3자는 귀하의 데이터를 판매하거나 광고에 사용할 권리가 없습니다.

아마존 로케이션은 아마존AWS CloudTrail, 아마존 CloudWatch EventBridge, AWS Identity and Access Management (IAM) 등의 서비스와 완벽하게 통합됩니다. Amazon Location은 데이터 통합을 통해 개발 워크플로를 간소화하고 내장된 모니터링, 보안 및 규정 준수 기능을 통해 앱을 프로덕션 단계까지 빠르게 추적합니다.

주요 내용, 제품 세부 정보 및 요금은 Amazon Location Service의 서비스 페이지를 참조하십시오.

# Amazon Location의 주요 기능

Amazon Location은 다음 기능을 제공합니다.

맵

Amazon Location Service Maps를 사용하면 위치 정보를 시각화할 수 있으며, 이는 다양한 위치 기반 서비스 기능의 토대입니다. Amazon Location Service는 글로벌 위치 데이터 공급자인 Esri, Grab, HERE에서 제공하는 다양한 스타일의 맵 타일과 오픈 데이터 맵을 제공합니다.

장소

Amazon Location Service Places를 사용하면 검색 기능을 애플리케이션에 통합하고, 주소를 위도 및 경도의 지리적 좌표로 변환(지오코딩)하고, 좌표를 거리 주소로 변환(역방향 지오코딩)할 수 있습니다. Amazon Location Service는 장소 기능을 지원하기 위해 Esri, Grab, HERE로부터 고품질 지리공간 데이터를 소싱합니다.

라우팅

Amazon Location Service Routes를 사용하면 up-to-date 도로 및 실시간 교통 정보를 기반으로 경로를 찾고 예상 이동 시간을 추정할 수 있습니다. 애플리케이션에서 두 위치 간의 이동 시간, 거리 및 방향을 요청할 수 있는 기능을 구축합니다. 경로 계획에 사용할 경로 매트릭스의 시간과 거리를 계산합니다.

지오펜싱

Amazon Location Service Geofences를 사용하면 디바이스가 지오펜스라는 정의된 지리적 경계에 들어가거나 나올 때 이를 감지하고 조치를 취하는 기능을 애플리케이션에 제공할 수 있습니다. 지

오펜스 침해가 EventBridge 감지되면 자동으로 진입 또는 퇴장 이벤트를 Amazon에 전송합니다. 이를 통해 대상에 대한 알림 전송과 같은 다운스트림 작업을 시작할 수 있습니다.

트래커

Amazon Location Service Trackers를 사용하면 추적 가능한 애플리케이션을 실행 중인 디바이스의 현재 위치와 과거 위치를 검색할 수 있습니다. 또한 추적기를 Amazon Location Service 지오펜스와 연결하여 디바이스의 위치 업데이트를 지오펜스와 비교하여 자동으로 평가할 수 있습니다. 추적기를 사용하면 위치 업데이트를 저장하거나 지오펜스와 비교하여 평가하기 전에 이동하지 않은 위치 업데이트를 필터링하여 비용을 절감할 수 있습니다.

추적기를 사용하면 추적되는 디바이스의 민감한 위치 정보가 AWS 계정을 떠나지 않습니다. 이를 통해 제3자로부터 민감한 정보를 보호하고, 사용자 개인 정보를 보호하고, 보안 위험을 줄일 수 있습니다.

# Amazon Location에서 사용할 수 있는 서비스

Amazon Location Service와 함께 다음 서비스를 사용하십시오.

통합 모니터링 및 관리

Amazon Location Service는 Amazon CloudWatchAWS CloudTrail, EventBridge Amazon과 통합되어 효율적인 모니터링 및 데이터 관리를 제공합니다.

- Amazon CloudWatch — 요청, 지연 시간, 장애 및 로그를 포함하여 서비스 사용 및 상태에 대한 메트릭을 볼 수 있습니다. 자세한 설명은 the section called “를 통한 모니터링 CloudWatch” 섹션을 참조하세요.
- AWS CloudTrail— 사용자, 역할 또는 AWS 서비스가 수행한 작업을 포함한 API 호출을 기록하고 모니터링합니다. 자세한 설명은 the section called “아마존 CloudTrail 로케이션과 함께 사용” 섹션을 참조하세요.
- Amazon EventBridge — AWS Lambda 함수를 사용하여 애플리케이션 및 워크플로의 다른 부분을 활성화할 수 있도록 이벤트 기반 애플리케이션 아키텍처를 활성화합니다. 자세한 설명은 the section called “다음과 같은 이벤트에 대응하기 EventBridge” 섹션을 참조하세요.

개발자 도구

Amazon Location Service는 개발자가 위치 지원 애플리케이션을 구축할 수 있는 다양한 도구를 제공합니다. 여기에는 표준 AWS SDK, 모바일 및 웹 SDK, 그리고 이를 다음과 같은 오픈 소스 라이브

러리와 결합하기 위한 샘플 코드가 포함됩니다. MapLibre Amazon Location Service 콘솔을 사용하여 리소스에 대해 알아보고 시각적이며 대화형 학습 도구를 사용해 보십시오.

# Amazon Location Service로 빠른 시작

Amazon Location Service를 시작하는 가장 효율적인 방법은 Amazon Location 콘솔을 사용하는 것입니다. 탐색 페이지를 사용하여 리소스를 생성 및 관리하고 Amazon Location 기능을 사용해 볼 수 있습니다.

> **Note**
>
> Amazon Location Service 콘솔을 사용하거나 이 자습서의 나머지 부분을 따르려면 먼저 AWS 계정 생성 및 Amazon Location에 대한 액세스 허용 등의 작업을 완료해야 합니다. Amazon Location Service 사용을 위한 사전 조건

Amazon Location API에 대해 배우기 시작하려면 다음 튜토리얼을 사용하여 대화형 맵을 표시하고 검색 기능을 사용하는 간단한 애플리케이션을 만들어 보세요. 자습서에는 세 가지 버전이 있습니다. 하나는 Kotlin을 사용하여 간단한 웹 페이지를 만드는 방법을 보여주고 JavaScript, 두 번째 버전은 Kotlin을 사용하는 Android 애플리케이션에 대해 동일하며, 세 번째 버전은 Swift를 사용하는 iOS 애플리케이션에 대해 동일합니다.

주제

- 웹 앱 생성
- Android 앱 생성
- iOS 앱 만들기

## 웹 앱 생성

이 섹션에서는 맵과 위치 검색 기능이 포함된 정적 웹 페이지를 생성합니다. 먼저 Amazon Location 리소스를 생성하고 애플리케이션을 위한 API 키를 생성합니다.

주제

- 앱을 위한 Amazon Location 리소스 생성
- 애플리케이션에 대한 인증 설정
- HTML애플리케이션에 맞게 만들기
- 애플리케이션에 대화형 맵 추가

- 애플리케이션에 검색 추가
- 최종 애플리케이션 보기
- 다음에 있는 것

## 앱을 위한 Amazon Location 리소스 생성

아직 리소스가 없는 경우 애플리케이션에서 사용할 Amazon Location 리소스를 생성해야 합니다. 여기서는 애플리케이션에 맵을 표시하는 맵 리소스와 맵에서 위치를 검색하기 위한 장소 색인을 생성합니다.

애플리케이션에 위치 리소스를 추가하려면

1. 사용하려는 맵 스타일을 선택합니다.

   a. Amazon Location 콘솔의 맵 페이지에서 맵 생성을 선택하여 맵 스타일을 미리 볼 수 있습니다.

   b. 새 맵 리소스의 이름과 설명을 추가합니다. 맵 리소스에 사용하는 이름을 기록해 둡니다. 이는 튜토리얼 뒷부분에서 스크립트 파일을 만들 때 필요합니다.

   c. 맵을 선택합니다.

   > **Note**
   >
   > 맵 스타일을 선택하면 사용할 맵 데이터 공급자도 선택됩니다. 애플리케이션이 배송 차량 또는 직원과 같이 비즈니스에 사용하는 자산을 추적하거나 라우팅하는 경우 지리적 위치 제공자로만 사용할 HERE 수 있습니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

   d. Amazon Location 이용 약관에 동의한 다음 맵 생성을 선택합니다. 선택한 맵과 상호 작용할 수 있습니다. 확대, 축소하거나 원하는 방향으로 이동할 수 있습니다.

   e. 새 맵 리소스에 표시된 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. 이 튜토리얼의 뒷부분에서 이를 사용하여 올바른 인증을 생성할 수 있습니다.

2. 사용하려는 장소 색인을 선택합니다.

   a. Amazon Location 콘솔의 장소 색인 페이지에서 장소 색인 생성을 선택합니다.

   b. 새 장소 색인 리소스의 이름과 설명을 추가합니다. 장소 색인 리소스에 사용하는 이름을 적어 둡니다. 이는 튜토리얼 뒷부분에서 스크립트 파일을 만들 때 필요합니다.

c. 데이터 공급자를 선택합니다.

> **Note**
>
> 대부분의 경우 이미 선택한 맵 공급자와 일치하는 데이터 공급자를 선택하세요. 이렇게 하면 검색 결과가 맵과 일치되게 할 수 있습니다.
> 애플리케이션이 배송 차량 또는 직원과 같이 비즈니스에 사용하는 자산을 추적하거나 라우팅하는 경우 지리적 위치 제공자로만 사용할 HERE 수 있습니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

d. 데이터 스토리지 옵션을 선택합니다. 이 튜토리얼에서는 결과가 저장되지 않으므로 아니오, 한 번만 사용합니다를 선택할 수 있습니다.

e. Amazon Location 이용 약관에 동의한 다음 장소 색인 생성을 선택합니다.

f. 새 장소 색인 리소스에 표시되는 내용을 기록해 두세요. ARN 이는 튜토리얼의 다음 섹션에서 올바른 인증을 만드는 데 사용합니다.

## 애플리케이션에 대한 인증 설정

이 자습서에서 만든 애플리케이션은 익명으로 사용되므로 사용자가 로그인하지 않아도 애플리케이션을 사용할 AWS 수 있습니다. 하지만 기본적으로 Amazon Location Service를 APIs 사용하려면 인증이 필요합니다. Amazon Cognito 또는 API 키를 사용하여 익명 사용자에게 인증 및 권한 부여를 제공할 수 있습니다. 이 자습서에서는 샘플 애플리케이션에서 사용할 API 키를 생성합니다.

> **Note**
>
> Amazon Location Service에서 API 키 또는 Amazon Cognito를 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. Amazon Location Service에 액세스 권한 부여

애플리케이션에 대한 인증을 설정하려면

1. Amazon 위치 콘솔로 이동하여 왼쪽 메뉴에서 API키를 선택합니다.
2. API 키 생성을 선택합니다.

> **Important**
>
> 생성하는 API 키는 이전 섹션에서 생성한 Amazon Location Service AWS 계정 리소스와 동일한 AWS 리전에 있어야 합니다.

3. API키 생성 페이지에서 다음 정보를 입력합니다.

   - 이름 - API 키의 이름 (예:)`MyWebAppKey`.
   - 리소스 – 이전 섹션에서 생성한 Amazon Location 맵과 장소 색인 리소스를 선택합니다. 리소스 추가를 선택하여 두 개 이상의 리소스를 추가할 수 있습니다. 이렇게 하면 API 키를 해당 리소스와 함께 사용할 수 있습니다.
   - 작업 - 이 API 키로 승인하려는 작업을 지정합니다. 자습서가 예상대로 작동하려면 최소한 geo: GetMap * 및 geo: SearchPlaceIndexfForPosition 를 선택해야 합니다.
   - 선택적으로 설명, 만료 시간 또는 태그를 키에 추가할 수 API 있습니다. 또한 리퍼러(예: `*.example.com`)를 추가하여 키가 특정 도메인에서만 사용되도록 제한할 수 있습니다. 즉, 튜토리얼은 해당 도메인에서만 작동합니다.

   > **Note**
   >
   > 만료 시간이나 리퍼러 (둘 다 설정하지 않는 경우) 둘 중 하나를 설정하여 API 키 사용을 보호하는 것이 좋습니다.

4. API키 생성을 선택하여 키를 생성합니다. API
5. Show API key를 선택하고 자습서에서 나중에 사용할 수 있도록 키 값을 복사합니다. 이는 `v1.public.`*`a1b2c3d4...`* 양식에 있습니다.

> **Important**
>
> 이 튜토리얼의 뒷부분에서 애플리케이션 코드를 작성할 때 이 키가 필요합니다.

## HTML애플리케이션에 맞게 만들기

이 자습서에서는 지도를 포함하는 정적 HTML 페이지를 만들어 사용자가 지도상의 특정 위치에 있는 내용을 찾을 수 있게 합니다. 앱은 세 개의 파일로 구성됩니다. 하나는 웹페이지용 HTML CSS 파일 및

파일이고, 다른 하나는 맵을 생성하고 사용자의 상호작용과 지도 이벤트에 응답하는 코드용 JavaScript (.js) 파일입니다.

먼저 애플리케이션에 사용할 HTML 및 CSS 프레임워크를 만들어 보겠습니다. 이 페이지는 맵 컨테이너를 보관하는  요소와 <pre>쿼리에 JSON 대한 응답을 표시하는 요소가 있는 간단한 페이지입니다.

퀵 스타트 애플리케이션을 만들려면 HTML

1. quickstart.html라는 파일을 새로 생성합니다.
2. 원하는 텍스트 편집기 또는 환경에서 파일을 편집합니다. 파일에 다음을 HTML 추가합니다.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quick start tutorial</title>

    <!-- Styles -->
    <link href="main.css" rel="stylesheet" />
  </head>

  <body>
    <header>
      <h1>Quick start tutorial</h1>
    </header>
    <main>
      <div id="map"></div>
      <aside>
        <h2>JSON Response</h2>
        <pre id="response"></pre>
      </aside>
    </main>
    <footer>This is a simple Amazon Location Service app. Pan and zoom. Click to
 see details about entities close to a point.</footer>

  </body>
</html>
```

여기에는 다음 단계에서 생성할 CSS 파일을 가리키는 포인터, 응용 프로그램의 일부 자리 표시자 요소 및 설명 HTML 텍스트가 있습니다.

이 튜토리얼의 후반부에서 사용할 두 개의 자리 표시자 요소가 있습니다. 첫 번째는 맵 컨트롤을 보유하는 <div id="map> 요소입니다. 두 번째는 맵에서의 검색 결과를 보여주는 <pre id="response"> 요소입니다.

3. 파일을 저장합니다.

이제 웹 CSS 페이지용으로 추가하십시오. 그러면 애플리케이션의 텍스트 및 자리 표시자 요소의 스타일이 설정됩니다.

퀵 스타트 CSS 애플리케이션용 앱을 만들려면

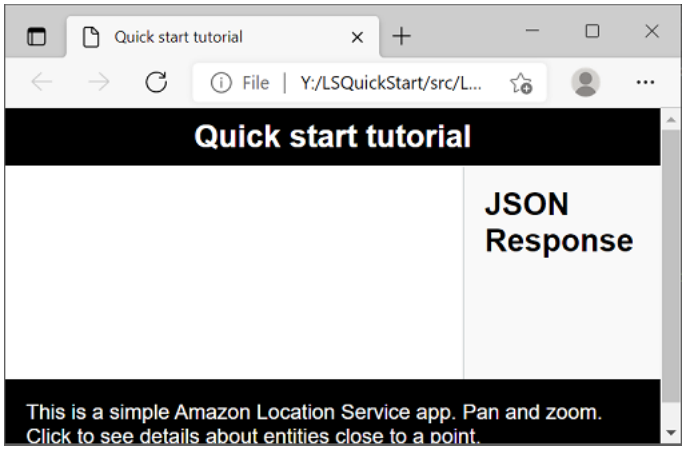
1. 이전 절차에서 만든 quickstart.html 파일과 동일한 폴더에 main.css라는 새 파일을 생성합니다.
2. 사용할 편집기에서 파일을 편집합니다. 다음 텍스트를 파일에 추가합니다.

```
* {
  box-sizing: border-box;
  font-family: Arial, Helvetica, sans-serif;
}

body {
  margin: 0;
}

header {
  background: #000000;
  padding: 0.5rem;
}

h1 {
  margin: 0;
  text-align: center;
  font-size: 1.5rem;
  color: #ffffff;
}

main {
  display: flex;
  min-height: calc(100vh - 94px);
}

#map {
```

```
  flex: 1;
}

aside {
  overflow-y: auto;
  flex: 0 0 30%;
  max-height: calc(100vh - 94px);
  box-shadow: 0 1px 1px 0 #001c244d, 1px 1px 1px 0 #001c2426, -1px 1px 1px 0
 #001c2426;
  background: #f9f9f9;
  padding: 1rem;
}

h2 {
  margin: 0;
}

pre {
  white-space: pre-wrap;
  font-family: monospace;
  color: #16191f;
}

footer {
  background: #000000;
  padding: 1rem;
  color: #ffffff;
}
```

이렇게 하면 맵이 다른 곳에서 사용하지 않는 공간을 채우도록 설정되고, 응답 영역이 앱 너비의 30% 를 차지하도록 설정되며, 제목과 설명 텍스트의 색상과 스타일이 설정됩니다.

3. 파일을 저장합니다.
4. 이제 브라우저에서 `quickstart.html` 파일을 보고 애플리케이션의 레이아웃을 볼 수 있습니다.




이제 애플리케이션에 맵 컨트롤을 추가합니다.

## 애플리케이션에 대화형 맵 추가

이제 프레임워크와 div 자리 표시자가 있으니 애플리케이션에 맵 컨트롤을 추가할 수 있습니다. 이 자습서에서는 MapLibre GL JS를 맵 컨트롤로 사용하여 Amazon Location Service에서 데이터를 가져옵니다. 또한 키를 사용하여 Amazon JavaScript 인증 도우미 Location으로 걸려오는 전화에 사용자 APIs API 키로 쉽게 서명할 수 있습니다.

애플리케이션에 대화형 맵을 추가하려면

1. 이전 섹션에서 생성한 `quickstart.html` 파일을 엽니다.
2. 필요한 라이브러리에 대한 참조와 생성할 스크립트 파일을 추가합니다. 변경해야 하는 내용은 **`green`**에 나와 있습니다.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quick start tutorial</title>

    <!-- Styles -->
    <link href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
 rel="stylesheet" />
    <link href="main.css" rel="stylesheet" />
  </head>

  <body>
    ...
```

```
    <footer>This is a simple Amazon Location Service app. Pan and zoom. Click to
 see details about entities close to a point.</footer>

    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-client@1.x/dist/
amazonLocationClient.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-utilities-auth-helper@1.x/
dist/amazonLocationAuthHelper.js"></script>

    <!-- JavaScript for the app -->
    <script src="main.js"></script>
  </body>
</html>
```

이것은 앱에 다음 종속 항목을 추가합니다.

- MapLibre GL JS. 이 라이브러리와 스타일시트에는 맵 타일을 표시하고 이동 및 확대/축소와 같은 대화형 기능을 포함하는 맵 컨트롤이 포함되어 있습니다. 컨트롤을 사용하면 맵에 직접 특징을 그리는 등 확장 기능도 사용할 수 있습니다.
- Amazon Location 클라이언트. 맵 데이터를 가져오고 맵에서 장소를 검색하는 데 필요한 Amazon Location 기능을 위한 인터페이스를 제공합니다. Amazon 로케이션 클라이언트는 AWS SDK JavaScript v3용 클라이언트를 기반으로 합니다.
- Amazon Location Authentication Helper. 이는 API 키 또는 Amazon Cognito를 사용하여 Amazon Location Service를 인증하는 데 유용한 기능을 제공합니다.

또한 이 단계에서는 다음에 생성할 `main.js`에 대한 참조도 추가합니다.

3. `quickstart.html` 파일을 저장합니다.
4. HTML및 파일과 같은 `main.js` 폴더에 호출된 새 CSS 파일을 만든 다음 편집용으로 엽니다.
5. 다음ㅍ 스크립트를 파일에 추가합니다. 안의 텍스트 *`red`* 이전에 생성한 API 키 값, 맵 리소스 이름, 장소 리소스 이름과 해당 지역의 지역 식별자 (예:`us-east-1`) 로 대체해야 합니다.

```
// Amazon Location Service resource names:
const mapName = "explore.map";
const placesName = "explore.place";
const region = "your_region";
const apiKey = "v1.public.a1b2c3d4...

// Initialize a map
```

```
async function initializeMap() {
  const mlglMap = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-77.03674, 38.891602], // Initial map centerpoint
    zoom: 16, // Initial map zoom
    style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=${apiKey}`, // Defines the appearance of the map and authenticates
 using an API key
  });

  // Add navigation control to the top left of the map
  mlglMap.addControl(new maplibregl.NavigationControl(), "top-left");

  return mlglMap;
}

async function main() {
  // Initialize map and Amazon Location SDK client:
  const map = await initializeMap();
}

main();
```

이 코드는 Amazon Location 리소스를 설정한 다음 MapLibre GL JS 맵 컨트롤을 구성 및 초기화하고 id와 함께 요소에 배치합니다.  map

initializeMap() 함수를 이해하는 것이 중요합니다. 애플리케이션에서 맵을 렌더링하는 데 사용되는 새 MapLibre 맵 컨트롤 (mlglMapmap로컬에서 호출되지만 나머지 코드에서는 호출) 을 생성합니다.

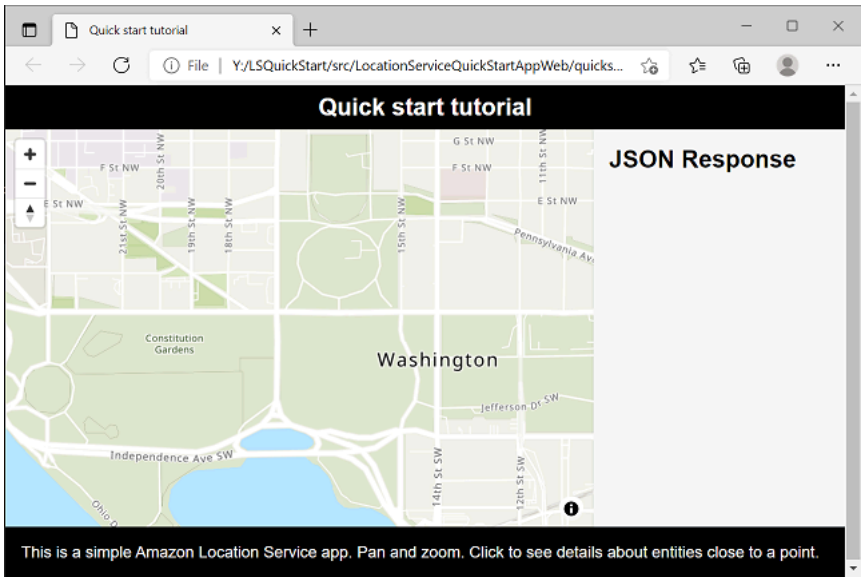
```
  // Initialize the map
  const mlglMap = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-77.03674, 38.891602], // Initial map centerpoint
    zoom: 16, // Initial map zoom
    style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=${apiKey}`, // Defines the appearance of the map and authenticates
 using an API key
  });
```

새 MapLibre 맵 컨트롤을 생성할 때 전달되는 매개변수는 맵 컨트롤의 초기 상태를 나타냅니다. 이제 다음 파라미터를 설정합니다.

- HTML우리의 HTML 맵 div 요소를 사용하는 컨테이너.
- 맵의 초기 중앙부터 워싱턴 DC의 특정 지점까지
- 16까지의 확대/축소 레벨(주변 지역 또는 블록 수준으로 확대/축소).
- 맵에 사용할 스타일로, 맵 타일과 맵을 URL 렌더링하는 데 필요한 기타 정보를 가져오는 데 사용할 수 있습니다. MapLibre 여기에는 인증을 위한 API 키도 URL 포함되어 있다는 점을 참고하세요.

6. JavaScript 파일을 저장하고 브라우저에서 엽니다. 이제 페이지에 이동 및 확대/축소 동작을 사용할 수 있는 맵이 생겼습니다.

> **Note**
>
> 이 앱을 사용하여 MapLibre 맵 컨트롤의 작동 방식을 확인할 수 있습니다. 드래그 작업을 사용하는 동안 Ctrl 또는 Shift를 사용하면 맵과 상호 작용하는 다른 방법을 확인할 수 있습니다. 이 모든 기능을 사용자 정의할 수 있습니다.




앱이 거의 완성되었습니다. 다음 섹션에서는 맵에서 위치를 선택하고 선택한 위치의 주소를 표시하는 방법에 대해서 살펴봅니다. 또한 페이지에 결과를 표시하여 JSON 전체 결과를 확인할 수 있습니다.

# 애플리케이션에 검색 추가

애플리케이션의 마지막 단계는 맵에 검색을 추가하는 것입니다. 이 경우 특정 위치에서 항목을 찾을 수 있는 역방향 지오코딩 검색을 추가합니다.

> **Note**
>
> 또한 Amazon Location Service는 이름이나 주소를 검색하여 맵에서 장소의 위치를 찾을 수 있는 기능을 제공합니다.

애플리케이션에 검색 기능을 추가하려면

1. 이전 섹션에서 생성한 `main.js` 파일을 엽니다.
2. 다음과 같이 `main` 함수를 수정합니다. 변경해야 하는 내용은 **`green`**에 나와 있습니다.

```
async function main() {
  // Create an authentication helper instance using an API key
  const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);


  // Initialize map and Amazon Location SDK client:
  const map = await initializeMap();

  const client = new amazonLocationClient.LocationClient({
    region,
    ...authHelper.getLocationClientConfig(), // Provides configuration required to
 make requests to Amazon Location
  });


  // On mouse click, display marker and get results:
  map.on("click", async function (e) {
    // Set up parameters for search call
    let params = {
      IndexName: placesName,
      Position: [e.lngLat.lng, e.lngLat.lat],
      Language: "en",
      MaxResults: "5",
    };

    // Set up command to search for results around clicked point
```

```
    const searchCommand = new
 amazonLocationClient.SearchPlaceIndexForPositionCommand(params);

    try {
      // Make request to search for results around clicked point
      const data = await client.send(searchCommand);

      // Write JSON response data to HTML
      document.querySelector("#response").textContent = JSON.stringify(data,
 undefined, 2);

      // Display place label in an alert box
      alert(data.Results[0].Place.Label);
    } catch (error) {
      // Write JSON response error to HTML
      document.querySelector("#response").textContent = JSON.stringify(error,
 undefined, 2);

      // Display error in an alert box
      alert("There was an error searching.");
    }
  });
}
```

이 코드는 Amazon Location 인증 도우미가 API 키를 사용하도록 설정하는 것으로 시작합니다.

```
const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);
```

그런 다음 해당 인증 도우미와 새 Amazon Location 클라이언트를 생성하는 데 사용하는 리전을 사용합니다.

```
const client = new amazonLocationClient.LocationClient({
    region,
    ...authHelper.getLocationClientConfig(),
  });
```

다음으로 코드는 사용자가 맵 컨트롤에서 지점을 선택하면 응답합니다. MapLibre 제공된 이벤트를 캡처하여 이 작업을 수행합니다. `click`

```
map.on("click", async function(e) {
    ...
```

```
  });
```

MapLibre `click`이벤트는 사용자가 선택한 위도와 경도를 포함하는 매개변수를 제공합니다 ()`e.lngLat`. `click` 이벤트 내에서 코드는 지정된 위도와 경도에 있는 엔티티를 찾기 위해 `searchPlaceIndexForPositionCommand`를 만듭니다.

```
    // Set up parameters for search call
    let params = {
      IndexName: placesName,
      Position: [e.lngLat.lng, e.lngLat.lat],
      Language: "en",
      MaxResults: "5"
    };

    // Set up command to search for results around clicked point
    const searchCommand = new
 amazonLocationClient.SearchPlaceIndexForPositionCommand(params);

    try {
      // Make request to search for results around clicked point
      const data = await client.send(searchCommand);
      ...
    });
```

여기서 `IndexName`는 이전에 생성한 장소 색인 리소스의 이름이고, `Position`는 검색할 위도와 경도이고, `Language`는 검색 결과에 사용할 기본 언어이고, `MaxResults`는 Amazon Location에 최대 5개의 결과만 반환하도록 지시합니다.

나머지 코드는 오류를 확인한 다음 `response`이라는 <pre> 요소에 검색 결과를 표시하고 알림 상자에 상위 결과를 표시합니다.

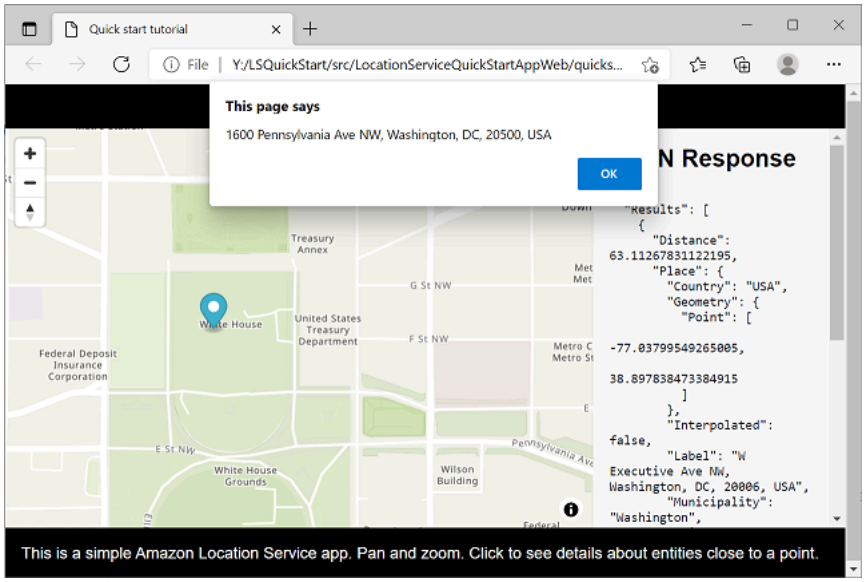
3. (선택 사항) 지금 `quickstart.html` 파일을 저장하고 브라우저에서 여는 경우 맵에서 위치를 선택하면 선택한 장소의 이름 또는 주소가 표시됩니다.
4. 애플리케이션의 마지막 단계는 이 MapLibre 기능을 사용하여 사용자가 선택한 지점에 마커를 추가하는 것입니다. `main` 함수를 다음과 같이 수정합니다. 변경해야 하는 내용은 **`green`**에 나와 있습니다.

```
async function main() {
  // Create an authentication helper instance using an API key
  const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);
```

```
 // Initialize map and Amazon Location SDK client
 const map = await initializeMap();
 const client = new amazonLocationClient.LocationClient({
   region,
   ...authHelper.getLocationClientConfig(), // Provides configuration required to
make requests to Amazon Location
 });

 // Variable to hold marker that will be rendered on click
 let marker;

 // On mouse click, display marker and get results:
 map.on("click", async function (e) {
   // Remove any existing marker
   if (marker) {
     marker.remove();
   }

   // Render a marker on clicked point
   marker = new maplibregl.Marker().setLngLat([e.lngLat.lng,
e.lngLat.lat]).addTo(map);

   // Set up parameters for search call
   let params = {
     IndexName: placesName,
     Position: [e.lngLat.lng, e.lngLat.lat],
     Language: "en",
     MaxResults: "5",
   };

   // Set up command to search for results around clicked point
   const searchCommand = new
amazonLocationClient.SearchPlaceIndexForPositionCommand(params);

...
```

이 코드는 사용자가 위치를 선택할 때마다 채워지는 marker 변수를 선언하여 선택한 위치를 표시합니다. .addTo(map);를 사용하여 마커를 맵에 추가하면 맵 컨트롤에서 마커를 자동으로 렌더링합니다. 또한 코드는 이전 마커를 확인하고 제거하여 화면에 한 번에 하나의 마커만 표시되도록 합니다.

5. `main.js` 파일을 저장하고 브라우저에서 `quickstart.html` 파일을 엽니다. 이전과 마찬가지로 맵을 이동 및 확대/축소할 수 있지만, 이제 위치를 선택하면 선택한 위치에 대한 세부 정보가 표시됩니다.




빠른 시작 애플리케이션을 완료했습니다. 이 자습서에서는 다음과 같은 정적 HTML 애플리케이션을 만드는 방법을 보여 주었습니다.

- 사용자가 상호 작용할 수 있는 맵을 만듭니다.
- 맵 이벤트(`click`)를 처리합니다.
- 특히 를 사용하여 특정 위치의 지도를 검색하기 위해 Amazon Location API Service를 `searchPlaceIndexForPosition` 호출합니다.
- MapLibre 맵 컨트롤을 사용하여 마커를 추가합니다.

## 최종 애플리케이션 보기

이 섹션에는 이 애플리케이션의 최종 소스 코드가 포함되어 있습니다. 에서 최종 프로젝트를 찾을 수도 GitHub 있습니다.

API키 대신 Amazon Cognito를 사용하는 애플리케이션 버전도 찾을 수 있습니다. GitHub

Overview

이 빠른 시작 튜토리얼에서 각 탭을 선택하면 파일의 최종 소스 코드를 볼 수 있습니다.

파일은 다음과 같습니다.

- quickstart.html — 맵과 검색 결과의 HTML 요소 홀더를 포함한 애플리케이션의 프레임워크입니다.
- main.css – 애플리케이션의 스타일시트입니다.
- main.js – 사용자를 인증하고, 맵을 생성하고, `click` 이벤트를 검색하는 애플리케이션용 스크립트입니다.

### quickstart.html

퀵 스타트 애플리케이션을 위한 HTML 프레임워크.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quick start tutorial</title>

    <!-- Styles -->
    <link href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
 rel="stylesheet" />
    <link href="main.css" rel="stylesheet" />
  </head>

  <body>
    ...
    <footer>This is a simple Amazon Location Service app. Pan and zoom. Click to see
 details about entities close to a point.</footer>

    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-client@1.x/dist/
amazonLocationClient.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-utilities-auth-helper@1.x/
dist/amazonLocationAuthHelper.js"></script>

    <!-- JavaScript for the app -->
    <script src="main.js"></script>
  </body>
</html>
```

### main.css

빠른 시작 애플리케이션의 스타일시트입니다.

```
* {
  box-sizing: border-box;
  font-family: Arial, Helvetica, sans-serif;
}

body {
  margin: 0;
}

header {
  background: #000000;
  padding: 0.5rem;
}

h1 {
  margin: 0;
  text-align: center;
  font-size: 1.5rem;
  color: #ffffff;
}

main {
  display: flex;
  min-height: calc(100vh - 94px);
}

#map {
  flex: 1;
}

aside {
  overflow-y: auto;
  flex: 0 0 30%;
  max-height: calc(100vh - 94px);
  box-shadow: 0 1px 1px 0 #001c244d, 1px 1px 1px 0 #001c2426, -1px 1px 1px 0
 #001c2426;
  background: #f9f9f9;
  padding: 1rem;
}

h2 {
  margin: 0;
}
```

```
pre {
  white-space: pre-wrap;
  font-family: monospace;
  color: #16191f;
}

footer {
  background: #000000;
  padding: 1rem;
  color: #ffffff;
}
```

main.js

빠른 시작 애플리케이션의 코드입니다. 텍스트는 다음과 같습니다.*red* 적절한 Amazon Location 객체 이름으로 대체해야 합니다.

```
// Amazon Location Service resource names:
const mapName = "explore.map";
const placesName = "explore.place";
const region = "your_region";
const apiKey = "v1.public.a1b2c3d4...

// Initialize a map
async function initializeMap() {
  // Initialize the map
  const mlglMap = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-77.03674, 38.891602], // Initial map centerpoint
    zoom: 16, // Initial map zoom
    style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=${apiKey}`, // Defines the appearance of the map and authenticates
 using an API key
  });

  // Add navigation control to the top left of the map
  mlglMap.addControl(new maplibregl.NavigationControl(), "top-left");

  return mlglMap;
}

async function main() {
```

```
 // Create an authentication helper instance using an API key
 const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);

 // Initialize map and Amazon Location SDK client
 const map = await initializeMap();
 const client = new amazonLocationClient.LocationClient({
   region,
   ...authHelper.getLocationClientConfig(), // Provides configuration required to
make requests to Amazon Location
 });

 // Variable to hold marker that will be rendered on click
 let marker;

 // On mouse click, display marker and get results:
 map.on("click", async function (e) {
   // Remove any existing marker
   if (marker) {
     marker.remove();
   }

   // Render a marker on clicked point
   marker = new maplibregl.Marker().setLngLat([e.lngLat.lng,
e.lngLat.lat]).addTo(map);

   // Set up parameters for search call
   let params = {
     IndexName: placesName,
     Position: [e.lngLat.lng, e.lngLat.lat],
     Language: "en",
     MaxResults: "5",
   };

   // Set up command to search for results around clicked point
   const searchCommand = new
amazonLocationClient.SearchPlaceIndexForPositionCommand(params);

   try {
     // Make request to search for results around clicked point
     const data = await client.send(searchCommand);

     // Write JSON response data to HTML
     document.querySelector("#response").textContent = JSON.stringify(data,
undefined, 2);
```

```
      // Display place label in an alert box
      alert(data.Results[0].Place.Label);
    } catch (error) {
      // Write JSON response error to HTML
      document.querySelector("#response").textContent = JSON.stringify(error,
 undefined, 2);

      // Display error in an alert box
      alert("There was an error searching.");
    }
  });
}

main();
```

## 다음에 있는 것

이제 빠른 시작 튜토리얼을 완료했으니 Amazon Location Service를 사용하여 애플리케이션을 구축하는 방법을 알 수 있습니다. Amazon Location을 최대한 활용하려면 다음 리소스를 확인하세요.

- Amazon Location Service의 개념에 대해 더 자세히 알아보기
- Amazon Location 특징 및 기능을 사용하는 방법에 대한 자세한 정보를 확인하세요.
- Amazon Location을 사용하는 코드 예시를 살펴보고 이 샘플을 확장하고 더 복잡한 애플리케이션을 구축하는 방법 알아보기

# Android 앱 생성

이 섹션에서는 지도, 위치 검색 기능, 포그라운드 추적 기능을 갖춘 Android 애플리케이션을 생성합니다. 먼저 Amazon 위치 리소스, Amazon Cognito 자격 증명 및 애플리케이션을 위한 API 키를 생성합니다.

주제

- 앱을 위한 Amazon Location 리소스 생성
- 애플리케이션에 대한 인증 설정
- 기본 Android 애플리케이션 만들기
- 애플리케이션에 대화형 맵 추가

- 애플리케이션에 역지오코딩 검색 추가
- 애플리케이션에 추적 추가
- 다음에 있는 것

## 앱을 위한 Amazon Location 리소스 생성

아직 리소스가 없는 경우 애플리케이션에서 사용할 Amazon Location 리소스를 생성해야 합니다. 여기서는 애플리케이션에 지도를 표시하는 맵 리소스, 지도에서 위치를 검색하는 장소 색인, 지도에서 객체를 추적하는 추적기를 생성합니다.

애플리케이션에 위치 리소스를 추가하려면

1. 사용하려는 맵 스타일을 선택합니다.

   a. Amazon Location 콘솔의 맵 페이지에서 맵 생성을 선택하여 맵 스타일을 미리 볼 수 있습니다.

   b. 새 맵 리소스의 이름과 설명을 추가합니다. 맵 리소스에 사용하는 이름을 기록해 둡니다. 이는 튜토리얼 뒷부분에서 스크립트 파일을 만들 때 필요합니다.

   c. 맵에는 HERE 맵 스타일을 선택하는 것이 좋습니다.

   > **Note**
   >
   > 맵 스타일을 선택하면 사용할 맵 데이터 공급자도 선택됩니다. 애플리케이션이 배송 차량이나 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 HERE만을 지리적 위치 제공업체로 사용할 수 있습니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

   d. Amazon Location 이용 약관에 동의한 다음 맵 생성을 선택합니다. 선택한 맵과 상호 작용할 수 있습니다. 확대, 축소하거나 원하는 방향으로 이동할 수 있습니다.

   e. 새 맵 리소스에 대해 표시되는 Amazon 리소스 이름(ARN)을 기록해 둡니다. 이 튜토리얼의 뒷부분에서 이를 사용하여 올바른 인증을 생성할 수 있습니다.

2. 사용하려는 장소 색인을 선택합니다.

   a. Amazon Location 콘솔의 장소 색인 페이지에서 장소 색인 생성을 선택합니다.

   b. 새 장소 색인 리소스의 이름과 설명을 추가합니다. 장소 색인 리소스에 사용하는 이름을 적어 둡니다. 이는 튜토리얼 뒷부분에서 스크립트 파일을 만들 때 필요합니다.

c. 데이터 공급자를 선택합니다.

> **Note**
>
> 대부분의 경우 이미 선택한 맵 공급자와 일치하는 데이터 공급자를 선택하세요. 이렇게 하면 검색 결과가 맵과 일치되게 할 수 있습니다.
> 애플리케이션이 배송 차량이나 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 HERE만을 지리적 위치 제공업체로 사용할 수 있습니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

d. 데이터 스토리지 옵션을 선택합니다. 이 튜토리얼에서는 결과가 저장되지 않으므로 아니오, 한 번만 사용합니다를 선택할 수 있습니다.

e. Amazon Location 이용 약관에 동의한 다음 장소 색인 생성을 선택합니다.

f. 새 장소 색인 리소스에 표시되는 ARN을 기록해 둡니다. 이는 튜토리얼의 다음 섹션에서 올바른 인증을 만드는 데 사용합니다.

3. Amazon 위치 콘솔을 사용하여 트래커를 만들려면

a. Amazon Location Service 콘솔을 엽니다.

b. 왼쪽 탐색 창에서 트래커를 선택합니다.

c. 트래커 생성을 선택합니다.

d. 필수 필드를 모두 입력합니다.

e. 포지션 필터링에서는 기본 설정인 을 사용하는 것이 좋습니다. TimeBased

f. 트래커 생성을 선택하여 완료하세요.

## 애플리케이션에 대한 인증 설정

이 자습서에서 만든 애플리케이션은 익명으로 사용되므로 사용자가 로그인하지 않아도 애플리케이션을 사용할 AWS 수 있습니다. 하지만 Amazon Location Service API를 사용하려면 인증이 필요합니다. API 키 또는 Amazon Cognito를 사용하여 익명 사용자에게 인증 및 권한 부여를 제공할 수 있습니다. 이 자습서에서는 Amazon Cognito와 API 키를 사용하여 애플리케이션을 인증합니다.

> **Note**
>
> Amazon Location Service에서의 Amazon Cognito 또는 API 키 사용에 대한 자세한 내용은 섹션 Amazon Location Service에 액세스 권한 부여을 참조하세요.

다음 자습서에서는 생성한 지도, 장소 색인, 추적기에 대한 인증을 설정하는 방법과 Amazon Location 에 대한 권한을 설정하는 방법을 보여줍니다.

인증 설정

1. Amazon 로케이션 콘솔로 이동하여 왼쪽 메뉴에서 API 키를 선택합니다.
2. 'API 키 생성'을 클릭합니다. API 키는 이전에 생성한 Amazon Location Service 리소스와 동일한 AWS 계정 및 지역에 있어야 한다는 점을 기억하십시오.
3. 'API 키 생성' 페이지에 필수 세부 정보를 입력합니다.

   - 이름: API 키의 이름을 입력합니다 (예:). `MyAppKey`
   - 리소스: 이전에 생성한 Amazon Location Service Map and Place 인덱스 리소스를 선택합니다. '리소스 추가'를 선택하여 여러 리소스를 추가할 수 있습니다. 이렇게 하면 API 키를 지정된 리소스와 함께 사용할 수 있습니다.
   - 조치: 이 API 키에 대해 승인된 작업을 지정합니다. 최소한 자습서가 의도한 대로 `geo:SearchPlaceIndexForPosition` 작동하도록 `geo:GetMap` 선택하고 확인하십시오.
   - 선택 사항으로 설명, 만료 시간, 태그 또는 리퍼러 등을 추가하여 키 사용을 특정 도메인으로 제한하여 자습서가 해당 도메인 내에서만 작동하도록 할 수 있습니다. `https://www.example.com`
4. API 키 생성을 클릭하여 API 키를 생성합니다.
5. `v1.public.a1b2c3d4`예를 들어 나중에 자습서에서 사용할 수 있도록 API 키 표시를 선택하고 키 값을 복사합니다.

추적을 위한 IAM 정책을 생성합니다.

1. 관리자 권한이 있는 사용자로 https://console.aws.amazon.com/iam/의 IAM 콘솔에 로그인합니다.
2. 탐색 창에서 정책을 선택합니다.
3. 콘텐츠 창에서 정책 생성을 선택합니다.
4. JSON 옵션을 선택한 다음 이 JSON 정책을 복사하여 JSON 텍스트 상자에 붙여넣습니다.

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
```

```
                "geo:GetMapTile",
                "geo:GetMapStyleDescriptor",
                "geo:GetMapSprites",
                "geo:GetMapGlyphs",
                "geo:SearchPlaceIndexForPosition",
                "geo:GetDevicePositionHistory",
                "geo:BatchUpdateDevicePosition"
            ],
            "Resource": [
                "arn:aws:geo:{Region}:{Account}:map/{MapName}",
                "arn:aws:geo:{Region}:{Account}:place-index/{IndexName}",
                "arn:aws:geo:{Region}:{Account}:tracker/{TrackerName}"
            ]
        }
    ]
}
```

다음은 추적에 대한 정책 예제입니다. 이 예제를 자체 정책에 사용하려면,, TrackerName자리 표시자를 바꾸십시오. `Region` `Account`

> **Note**
>
> 인증되지 않은 자격 증명 풀은 보안되지 않은 인터넷 사이트에 노출되기 위한 것이지만, 이는 기간 제한이 있는 표준 AWS 자격 증명으로 교환된다는 점에 유의하십시오. 인증되지 않은 자격 증명 풀과 관련된 IAM 역할의 범위를 적절하게 지정하는 것이 중요합니다. Amazon Location Service와 함께 Amazon Cognito에서 정책을 사용하고 적절하게 범위를 지정하는 방법에 대한 자세한 내용은 Amazon Location Service에 대한 액세스 권한 부여를 참조하십시오.

5. 검토 및 생성 페이지에서 정책 이름 필드에 이름을 입력합니다. 정책에서 부여한 권한을 검토한 다음 [Create Policy] 를 선택하여 작업 내용을 저장합니다.

새로운 정책이 관리형 정책 목록에 나타나며 연결 준비가 완료됩니다.

추적에 대한 인증을 설정하세요.

1. Amazon Cognito 콘솔에서 맵 애플리케이션에 대한 인증을 설정합니다.
2. 자격 증명 풀 페이지를 엽니다.

> **Note**
>
> 생성한 풀은 이전 섹션에서 생성한 Amazon Location Service 리소스와 동일한 AWS 계정 및 AWS 지역에 있어야 합니다.

3. 자격 증명 풀 생성을 선택합니다.
4. ID 풀 신뢰 구성 단계부터 시작합니다. 사용자 액세스 인증의 경우 게스트 액세스를 선택하고 다음을 누릅니다.
5. 권한 구성 페이지에서 기존 IAM 역할 사용을 선택하고 이전 단계에서 생성한 IAM 역할의 이름을 입력합니다. 준비가 되면 다음을 눌러 다음 단계로 넘어갑니다.
6. 속성 구성 페이지에서 자격 증명 풀의 이름을 입력합니다. 그런 다음 [다음] 을 누릅니다.
7. 검토 및 생성 페이지에서 제공되는 모든 정보를 검토한 다음 ID 풀 생성을 누릅니다.
8. ID 풀 페이지를 열고 방금 생성한 자격 증명 풀을 선택합니다. 그런 다음 나중에 사용할 내용을 IdentityPoolId 브라우저 스크립트에 복사하거나 기록해 둡니다.

## 기본 Android 애플리케이션 만들기

이 자습서에서는 지도를 포함하고 사용자가 지도상의 특정 위치에 있는 내용을 찾을 수 있도록 하는 Android 애플리케이션을 만듭니다.

먼저 Android 스튜디오의 새 프로젝트 마법사를 사용하여 빈 Kotlin 애플리케이션을 생성합니다.

빈 애플리케이션을 만들려면 () AndroidStudio

1. 시작 AndroidStudio. 메뉴를 열고 [파일], [새로 만들기], [새 프로젝트] 를 선택합니다.
2. 휴대폰 및 태블릿 탭에서 빈 활동을 선택한 후 다음을 선택합니다.
3. 애플리케이션의 이름, 패키지 이름 및 저장 위치를 선택합니다.
4. 언어 드롭다운 목록에서 Kotlin을 선택합니다.
5. 마침을 선택하여 빈 애플리케이션을 생성합니다.

## 애플리케이션에 대화형 맵 추가

이제 기본 애플리케이션을 만들었으므로 애플리케이션에 맵 컨트롤을 추가할 수 있습니다. 이 자습서에서는 맵 뷰를 관리하는 데 API 키를 사용합니다. 맵 컨트롤 자체는 API 키와 함께 MapLibre 네이티브 라이브러리의 일부이며 MapLibre, 맵 데이터는 Amazon Location에서 가져옵니다.

애플리케이션에 맵을 추가하려면 다음 작업을 수행해야 합니다.

- 프로젝트에 MapLibre 종속성을 추가합니다.
- compose를 사용하여 맵 뷰 코드를 설정합니다.
- 지도를 표시하는 코드를 작성하세요.

앱에 맵을 추가하려면 다음 절차를 따르세요.

1. 프로젝트에 MapLibre 종속 항목 추가

   a. 에서 AndroidStudio 보기 메뉴를 선택하고 도구, 창, 프로젝트를 선택합니다. 그러면 프로젝트 창이 열리고 프로젝트의 모든 파일에 접근할 수 있습니다.

   b. 프로젝트 창에서 gradle을 연 다음 트리 뷰에서 `libs.versions.toml` 파일을 엽니다. 그러면 편집할 `libs.versions.toml` 파일이 열립니다. 이제 `libs.versions.toml` 파일에 아래 버전과 라이브러리 데이터를 추가하세요.

   ```
   [versions]
    ...
    auth = "0.2.4"
    tracking = "0.2.4"

    [libraries]
    ...
    auth = { group = "software.amazon.location", name = "auth", version.ref =
    "auth" }
    tracking = { module = "software.amazon.location:tracking", version.ref =
    "tracking" }

    [plugins]
    ...
   ```

   c. `libs.versions.toml`파일 편집을 마친 후에는 프로젝트를 다시 AndroidStudio 동기화해야 합니다. `libs.versions.toml`편집 창 상단에 AndroidStudio 동기화하라는 메시지가 표시됩니다. 계속하기 전에 'Sync Now'를 선택하여 프로젝트를 동기화하세요.

   d. 프로젝트 창의 트리 뷰에서 Gradle Script를 열고 애플리케이션 모듈에 사용할 `build.gradle` 파일을 선택합니다. 그러면 편집할 `build.gradle` 파일이 열립니다.

   e. 파일 하단의 종속성 섹션에 다음 종속 항목을 추가합니다.

   ```
   dependencies {
   ```

```
    ...
    implementation(libs.org.maplibre.gl)
}
```

f. Gradle 종속 항목 추가를 완료한 후에는 Android 스튜디오에서 프로젝트를 다시 동기화해야 합니다. Android 스튜디오의 build.gradle 편집 창 상단에서 Sync Now를 선택하여 계속하기 전에 프로젝트를 동기화합니다.

2. 이제 compose를 사용하여 맵 뷰 코드를 설정해 보겠습니다. 다음 단계를 사용합니다.

a. 프로젝트 창에서 트리 뷰의 ***### ### App***, Java, 열고 ui 폴더로 이동한 다음 ui 폴더 안에 있는 뷰 디렉토리를 생성합니다.

b. 뷰 디렉터리 안에서 `MapLoadScreen.kt` 파일을 생성합니다.

c. 다음 코드를 `MapLoadScreen.kt` 파일에 추가합니다.

```
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.viewinterop.AndroidView
import org.maplibre.android.maps.OnMapReadyCallback

 @Composable
 fun MapLoadScreen(
     mapReadyCallback: OnMapReadyCallback,
 ) {
     Box(
         modifier = Modifier
             .fillMaxWidth()
             .fillMaxHeight(),
     ) {
         MapView(mapReadyCallback)
     }
 }

 @Composable
 fun MapView(mapReadyCallback: OnMapReadyCallback) {
     AndroidView(
         factory = { context ->
             val mapView = org.maplibre.android.maps.MapView(context)
             mapView.onCreate(null)
             mapView.getMapAsync(mapReadyCallback)
```

```
            mapView
        },
    )
}
```

3. 지도를 표시하는 코드를 작성하세요.

   a. 다음 코드를 `MainActivity.kt` 파일에 추가합니다.

```
// ...other imports
import org.maplibre.android.MapLibre
import org.maplibre.android.camera.CameraPosition
import org.maplibre.android.geometry.LatLng
import org.maplibre.android.maps.MapLibreMap
import org.maplibre.android.maps.OnMapReadyCallback
import org.maplibre.android.maps.Style

class MainActivity : ComponentActivity(), OnMapReadyCallback {
    private val region = "YOUR_AWS_REGION"
    private val mapName = "YOUR_AWS_MAP_NAME"
    private val apiKey = "YOUR_AWS_API_KEY"
    override fun onCreate(savedInstanceState: Bundle?) {
        MapLibre.getInstance(this)
        super.onCreate(savedInstanceState)
        setContent {
            TestMapAppTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MapLoadScreen(this)
                }
            }
        }
    }

    override fun onMapReady(map: MapLibreMap) {
        map.setStyle(
            Style.Builder()
                .fromUri(
                    "https://maps.geo.$region.amazonaws.com/maps/v0/maps/
$mapName/style-descriptor?key=$apiKey"
                ),
        ) {
```

```
            map.uiSettings.isAttributionEnabled = true
            map.uiSettings.isLogoEnabled = false
            map.uiSettings.attributionGravity = Gravity.BOTTOM or Gravity.END
            val initialPosition = LatLng(47.6160281982247,
-122.32642111977668)
            map.cameraPosition = CameraPosition.Builder()
                .target(initialPosition)
                .zoom(14.0)
                .build()
        }
    }
}
```

b. `MainActivity.kt` 파일을 저장합니다. 이제 애플리케이션을 구축할 수 있습니다. 앱을 실행하려면 에뮬레이션할 기기를 설정하거나 기기에서 앱을 사용해야 할 수 있습니다. AndroidStudio 이 앱을 사용하면 맵 컨트롤이 어떻게 작동하는지 확인할 수 있습니다. 맵에서 드래그한 다음 손가락을 집어 확대하여 패닝할 수 있습니다.

   다음 섹션에서는 지도에 마커를 추가하고 맵을 이동할 때 마커가 있는 위치의 주소를 표시합니다.

## 애플리케이션에 역지오코딩 검색 추가

이제 특정 위치에서 항목을 찾을 수 있는 애플리케이션에 역지오코딩 검색을 추가해 보겠습니다. Android 앱 사용을 단순화하기 위해 화면 중앙을 검색해 보겠습니다. 새 위치를 찾으려면 지도를 검색하려는 위치로 이동하세요. 지도 중앙에 마커를 배치하여 검색 위치를 표시합니다.

역방향 지오코딩 검색 추가는 두 부분으로 구성됩니다.

- 화면 중앙에 마커를 추가하여 검색 위치를 사용자에게 표시합니다.
- 결과를 표시할 텍스트 상자를 추가한 다음 마커 위치에 무엇이 있는지 검색하여 텍스트 상자에 표시합니다.

애플리케이션에 마커를 추가하려면

1. 이 이미지를 프로젝트의 `app/res/drawable` 폴더에 저장합니다 `red_marker.png` (에서 이미지에 액세스할 수도 있음) GitHub. 이미지를 직접 만들 수도 있습니다. 표시하지 않으려는 부분에 투명도가 있는.png 파일을 사용할 수도 있습니다.
2. MapLoadScreen.kt 파일에 다음 코드를 추가합니다.

```
// ...other imports
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.size
import androidx.compose.ui.Alignment
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.amazon.testmapapp.R

@Composable
fun MapLoadScreen(
    mapReadyCallback: OnMapReadyCallback,
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .fillMaxHeight(),
    ) {
        MapView(mapReadyCallback)
        Box(
            modifier = Modifier
                .align(Alignment.Center),
        ) {
            Image(
                painter = painterResource(id = R.drawable.red_marker),
                contentDescription = "marker",
                modifier = Modifier
                    .size(40.dp)
                    .align(Alignment.Center),
            )
        }
    }
}

@Composable
fun MapView(mapReadyCallback: OnMapReadyCallback) {
    AndroidView(
        factory = { context ->
            val mapView = org.maplibre.android.maps.MapView(context)
            mapView.onCreate(null)
            mapView.getMapAsync(mapReadyCallback)
            mapView
        },
```

```
    )
}
```

3. 앱을 빌드하고 실행하여 기능을 미리 살펴보세요.

이제 앱 화면에 마커가 생겼습니다. 이 경우에는 움직이지 않는 정적 이미지입니다. 맵 뷰의 중앙을 표시하는 데 사용되며, 여기서 검색할 위치입니다. 다음 절차에서는 해당 위치에 검색을 추가합니다.

특정 위치의 역방향 지오코딩 검색을 앱에 추가하려면

1. 프로젝트 창의 트리 뷰에서 Gradle to `libs.versions.toml` file을 엽니다. 그러면 편집할 `libs.versions.toml` 파일이 열립니다. 이제 `libs.versions.toml` 파일에 아래 버전과 라이브러리 데이터를 추가하세요.

```
[versions]
 ...
 okhttp = "4.12.0"

 [libraries]
 ...
 com-squareup-okhttp3 = { group = "com.squareup.okhttp3", name = "okhttp",
 version.ref = "okhttp" }

 [plugins]
 ...
```

2. `libs.versions.toml`파일 편집을 마친 후에는 프로젝트를 다시 AndroidStudio 동기화해야 합니다. `libs.versions.toml`편집 창 상단에 AndroidStudio 동기화하라는 메시지가 표시됩니다. 계속하기 전에 'Sync Now'를 선택하여 프로젝트를 동기화하세요.
3. 프로젝트 창의 트리 뷰에서 Gradle Script를 열고 애플리케이션 모듈에 사용할 `build.gradle` 파일을 선택합니다. 그러면 편집할 `build.gradle` 파일이 열립니다.
4. 파일 하단의 종속성 섹션에 다음 종속 항목을 추가합니다.

```
dependencies {
    ...
    implementation(libs.com.squareup.okhttp3)
 }
```

5. Gradle 종속성 편집을 마친 후에는 프로젝트를 다시 AndroidStudio 동기화해야 합니다. `build.gradle`편집 창 상단에 AndroidStudio 동기화하라는 메시지가 표시됩니다. SyncNow계속하기 전에 프로젝트 동기화를 선택하십시오.
6. 이제 트리 뷰에서 요청 디렉터리에 데이터를 추가하고 `ReverseGeocodeRequest.kt` 데이터 클래스를 생성합니다. 클래스에 다음 코드를 추가합니다.

```
import com.google.gson.annotations.SerializedName

data class ReverseGeocodeRequest(
    @SerializedName("Language")
    val language: String,
    @SerializedName("MaxResults")
    val maxResults: Int,
    @SerializedName("Position")
    val position: ListDouble
)
```

7. 이제 트리 뷰에서 응답 디렉터리에 데이터를 추가하고 `ReverseGeocodeResponse.kt` 데이터 클래스를 생성합니다. 그 안에 다음 코드를 추가합니다.

```
import com.google.gson.annotations.SerializedName

data class ReverseGeocodeResponse(
    @SerializedName("Results")
    val results: ListResult
)

data class Result(
    @SerializedName("Place")
    val place: Place
)

data class Place(
    @SerializedName("Label")
    val label: String
)
```

8. 이제 프로젝트 창에서 트리 뷰에서 App, Java, ### ### 열고 ui 폴더 안에 있는 ui 폴더로 이동하여 ViewModel 디렉토리를 생성합니다.
9. 뷰모델 디렉터리 내에서 파일을 생성합니다`MainViewModel.kt`.
10. 다음 코드를 `MainViewModel.kt` 파일에 추가합니다.

```
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.setValue
import androidx.lifecycle.ViewModel
import com.amazon.testmapapp.data.request.ReverseGeocodeRequest
import com.amazon.testmapapp.data.response.ReverseGeocodeResponse
import com.google.gson.Gson
import java.io.IOException
import okhttp3.Call
import okhttp3.Callback
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.Response
import org.maplibre.android.geometry.LatLng
import org.maplibre.android.maps.MapLibreMap

class MainViewModel : ViewModel() {
    var label by mutableStateOf("")
    var isLabelAdded: Boolean by mutableStateOf(false)
    var client = OkHttpClient()
    var mapLibreMap: MapLibreMap? = null

    fun reverseGeocode(latLng: LatLng, apiKey: String) {
        val region = "YOUR_AWS_REGION"
        val indexName = "YOUR_AWS_PLACE_INDEX"
        val url =
            "https://places.geo.${region}.amazonaws.com/places/v0/indexes/
${indexName}/search/position?key=${apiKey}"

        val requestBody = ReverseGeocodeRequest(
            language = "en",
            maxResults = 1,
            position = listOf(latLng.longitude, latLng.latitude)
        )
        val json = Gson().toJson(requestBody)

        val mediaType = "application/json".toMediaTypeOrNull()
        val request = Request.Builder()
            .url(url)
            .post(json.toRequestBody(mediaType))
            .build()
```

```
        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                e.printStackTrace()
            }

            override fun onResponse(call: Call, response: Response) {
                if (response.isSuccessful) {
                    val jsonResponse = response.body?.string()

                    val reverseGeocodeResponse =
                        Gson().fromJson(jsonResponse,
 ReverseGeocodeResponse::class.java)

                    val responseLabel =
 reverseGeocodeResponse.results.firstOrNull()?.place?.label

                    if (responseLabel != null) {
                        label = responseLabel
                        isLabelAdded = true
                    }
                }
            }
        })
    }
}
```

11. 파일이 아직 열리지 않은 경우 이전 절차에서와 같이 `MapLoadScreen.kt` 파일을 여세요. 다음 코드를 추가합니다. 그러면 해당 위치의 리버스 지오코딩 검색 결과를 볼 수 있는 Compose Text 뷰가 생성됩니다.

```
// ...other imports
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Text
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.testTag
```

```
import androidx.compose.ui.semantics.contentDescription
import androidx.compose.ui.semantics.semantics
import androidx.compose.ui.unit.sp
import com.amazon.testmapapp.ui.viewModel.MainViewModel

@Composable
fun MapLoadScreen(
    mapReadyCallback: OnMapReadyCallback,
    mainViewModel: MainViewModel,
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .fillMaxHeight(),
    ) {
        MapView(mapReadyCallback)
        Box(
            modifier = Modifier
                .align(Alignment.Center),
        ) {
            Image(
                painter = painterResource(id = R.drawable.red_marker),
                contentDescription = "marker",
                modifier = Modifier
                    .size(40.dp)
                    .align(Alignment.Center),
            )
        }
        if (mainViewModel.isLabelAdded) {
            Column(
                modifier = Modifier.fillMaxSize(),
                verticalArrangement = Arrangement.Bottom
            ) {
                Box(
                    modifier = Modifier
                        .fillMaxWidth()
                        .background(Color.White),
                ) {
                    Text(
                        text = mainViewModel.label,
                        modifier = Modifier
                            .padding(16.dp)
                            .align(Alignment.Center)
                            .testTag("label")
```

```
                            .semantics {
                                contentDescription = "label"
                            },
                        fontSize = 14.sp,
                    )
                }
                Spacer(modifier = Modifier.height(80.dp))
            }
        }
    }
}

@Composable
fun MapView(mapReadyCallback: OnMapReadyCallback) {
    AndroidView(
        factory = { context ->
            val mapView = org.maplibre.android.maps.MapView(context)
            mapView.onCreate(null)
            mapView.getMapAsync(mapReadyCallback)
            mapView
        },
    )
}
```

12. 앱의 Java에 있는 패키지 이름 폴더에서 AndroidStudio 파일을 엽니다. `MainActivity.kt` 다음과 같이 코드를 수정합니다.

```
// ...other imports
import androidx.activity.viewModels
import com.amazon.testmapapp.ui.viewModel.MainViewModel

 class MainActivity : ComponentActivity(), OnMapReadyCallback,
 MapLibreMap.OnCameraMoveStartedListener, MapLibreMap.OnCameraIdleListener {

     private val mainViewModel: MainViewModel by viewModels()
     private val region = "YOUR_AWS_REGION"
     private val mapName = "YOUR_AWS_MAP_NAME"
     private val apiKey = "YOUR_AWS_API_KEY"
     override fun onCreate(savedInstanceState: Bundle?) {
         MapLibre.getInstance(this)
         super.onCreate(savedInstanceState)
         setContent {
             TestMapAppTheme {
```

```
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MapLoadScreen(this, mainViewModel)
                }
            }
        }
    }

    override fun onMapReady(map: MapLibreMap) {
        map.setStyle(
            Style.Builder()
                .fromUri(
                    "https://maps.geo.$region.amazonaws.com/maps/v0/maps/$mapName/
style-descriptor?key=$apiKey"
                ),
        ) {
            map.uiSettings.isAttributionEnabled = true
            map.uiSettings.isLogoEnabled = false
            map.uiSettings.attributionGravity = Gravity.BOTTOM or Gravity.END
            val initialPosition = LatLng(47.6160281982247, -122.32642111977668)
            map.cameraPosition = CameraPosition.Builder()
                .target(initialPosition)
                .zoom(14.0)
                .build()

            map.addOnCameraMoveStartedListener(this)
            map.addOnCameraIdleListener(this)
            map.cameraPosition.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }
   override fun onCameraMoveStarted(p0: Int) {
       mainViewModel.label = ""
       mainViewModel.isLabelAdded = false
   }
```

```
    override fun onCameraIdle() {
        if (!mainViewModel.isLabelAdded) {
            mainViewModel.mapLibreMap?.cameraPosition?.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }
}
```

이 코드는 맵 뷰에서 작동합니다. 가상 카메라 위치는 맵 뷰를 정의합니다 MapLibre. 맵을 이동하는 것은 가상 카메라를 움직이는 것으로 생각할 수 있습니다.

- ViewModel 레이블 변수 포함: 이 변수는 텍스트 작성 뷰의 데이터를 설정합니다.
- onMapReady레이블 변수 포함: 이 변수는 작성 텍스트 보기에서 데이터를 설정합니다. ----sep----:이 함수는 두 개의 새 이벤트를 등록하도록 업데이트되었습니다.
- onCameraMove이벤트는 사용자가 맵을 이동할 때마다 발생합니다. 일반적으로 맵을 이동할 때는 사용자가 맵 이동을 완료할 때까지 검색을 숨기려고 합니다.
- 이 onCameraIdle이벤트는 사용자가 맵 이동을 일시 중지할 때 발생합니다. 이 이벤트는 리버스 지오코드 함수를 호출하여 지도 중앙을 검색합니다.
- `reverseGeocode(latLng: LatLng, apiKey: String)`: 이벤트에서 onCameraIdle 호출되는 이 함수는 지도 중앙에서 위치를 검색하고 레이블을 업데이트하여 결과를 표시합니다. 지도의 중심 (카메라가 바라보는 위치) 을 정의하는 카메라 타겟을 사용합니다.

13. 파일을 저장하고 앱을 빌드하고 실행하여 제대로 작동하는지 확인하세요.

검색 기능이 있는 퀵 스타트 애플리케이션이 완성되었습니다.

## 애플리케이션에 추적 추가

샘플 애플리케이션에 추적을 추가하려면 다음 단계를 따르십시오.

1. 프로젝트에 추적 및 인증 SDK 종속 항목을 추가하세요.
2. .xml 파일에 권한 및 서비스 항목을 포함하세요. AndroidManifest
3. compose를 사용하여 추적 시작/중지 버튼 코드를 설정합니다.

4. LocationTracker 객체를 만들고 추적을 시작하고 중지하기 위한 코드를 추가합니다.
5. Android 에뮬레이터로 테스트 경로를 만드세요.

1. 프로젝트에 추적 및 인증 SDK 종속 항목을 추가하세요.

   a. 프로젝트 창에서 gradle을 연 다음 트리 뷰에서 `libs.versions.toml` 파일을 엽니다. 그러면 편집할 `libs.versions.toml` 파일이 열립니다. 이제 `libs.versions.toml` 파일에 아래 버전과 라이브러리 데이터를 추가하세요.

   ```
   [versions]
    ...
    auth = "0.0.1"
    tracking = "0.0.1"

    [libraries]
    ...
    auth = { group = "software.amazon.location", name = "auth", version.ref =
    "auth" }
    tracking = { module = "software.amazon.location:tracking", version.ref =
    "tracking" }

    [plugins]
    ...
   ```

   b. `libs.versions.toml`파일 편집을 마친 후에는 프로젝트를 다시 AndroidStudio 동기화해야 합니다. `libs.versions.toml`편집 창 상단에 AndroidStudio 동기화하라는 메시지가 표시됩니다. 계속하기 전에 'Sync Now'를 선택하여 프로젝트를 동기화하세요.

   c. 프로젝트 창의 트리 뷰에서 Gradle Script를 열고 애플리케이션 모듈에 사용할 `build.gradle` 파일을 선택합니다. 그러면 편집할 `build.gradle` 파일이 열립니다.

   d. 파일 하단의 종속성 섹션에 다음 종속 항목을 추가합니다.

   ```
   dependencies {
       ...
       implementation(libs.auth)
       implementation(libs.tracking)
    }
   ```

   e. Gradle 종속성 편집을 마친 후에는 프로젝트를 다시 AndroidStudio 동기화해야 합니다. build.gradle 편집 창 상단에 동기화하라는 메시지가 표시됩니다. AndroidStudio SyncNow계속하기 전에 프로젝트 동기화를 선택하세요.

2. AndroidManifest.xml 파일에 권한 및 서비스 항목을 포함하세요.

   - 에 올바른 권한 및 서비스 항목을 포함하려면 다음 코드로 파일을 업데이트하십시오AndroidManifest.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AndroidQuickStartApp"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.AndroidQuickStartApp">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

3. compose로 추적 시작/중지 버튼 코드를 설정하세요.

   a. ic_pause 및 ic_play라는 드로어블 아래에 res에서 재생 및 일시 중지 이미지 두 개를 추가합니다. 에서 이미지에 액세스할 수도 있습니다. GitHub

b. 파일이 아직 열리지 않은 경우 이전 절차에서와 같이 MapLoadScreen.kt 파일을 여세요. 다음 코드를 추가합니다. 그러면 Compose Button 뷰가 생성되어 이를 클릭하여 추적을 시작하고 중지할 수 있습니다.

```
// ...other imports
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults

@Composable
fun MapLoadScreen(
    mapReadyCallback: OnMapReadyCallback,
    mainViewModel: MainViewModel,
    onStartStopTrackingClick: () -> Unit
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .fillMaxHeight(),
    ) {
        MapView(mapReadyCallback)
        Box(
            modifier = Modifier
                .align(Alignment.Center),
        ) {
            Image(
                painter = painterResource(id = R.drawable.red_marker),
                contentDescription = "marker",
                modifier = Modifier
                    .size(40.dp)
                    .align(Alignment.Center),
            )
        }
        if (mainViewModel.isLabelAdded) {
            Column(
                modifier = Modifier.fillMaxSize(),
                verticalArrangement = Arrangement.Bottom
            ) {
                Box(
                    modifier = Modifier
                        .fillMaxWidth()
                        .background(Color.White),
                ) {
                    Text(
```

```
                        text = mainViewModel.label,
                        modifier = Modifier
                            .padding(16.dp)
                            .align(Alignment.Center)
                            .testTag("label")
                            .semantics {
                                contentDescription = "label"
                            },
                        fontSize = 14.sp,
                    )
                }
                Spacer(modifier = Modifier.height(80.dp))
            }
        }
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(bottom = 16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Bottom,
        ) {
            Button(
                onClick = onStartStopTrackingClick,
                modifier = Modifier
                    .padding(horizontal = 16.dp)
            ) {
                Text(
                    text = if
(mainViewModel.isLocationTrackingForegroundActive) "Stop tracking" else "Start
tracking",
                    color = Color.Black
                )
                Spacer(modifier = Modifier.size(ButtonDefaults.IconSpacing))
                Image(
                    painter = painterResource(id = if
(mainViewModel.isLocationTrackingForegroundActive) R.drawable.ic_pause else
R.drawable.ic_play),
                    contentDescription = if
(mainViewModel.isLocationTrackingForegroundActive) "stop_tracking" else
"start_tracking"
                )
            }
        }
    }
```

```
}

@Composable
fun MapView(mapReadyCallback: OnMapReadyCallback) {
    AndroidView(
        factory = { context ->
            val mapView = org.maplibre.android.maps.MapView(context)
            mapView.onCreate(null)
            mapView.getMapAsync(mapReadyCallback)
            mapView
        },
    )
}
```

4. LocationTracker 객체를 만들고 추적을 시작하고 중지하기 위한 코드를 추가하세요.

   a. `MainViewModel.kt`파일 내에 다음 코드를 추가합니다.

   ```
   ...
   var isLocationTrackingForegroundActive: Boolean by mutableStateOf(false)
   var locationTracker: LocationTracker? = null
   ```

   b. 다음 코드를 `MainActivity.kt` 파일에 추가합니다.

   ```
   // ...other imports
   import software.amazon.location.auth.AuthHelper
   import software.amazon.location.auth.LocationCredentialsProvider
   import software.amazon.location.tracking.LocationTracker
   import software.amazon.location.tracking.aws.LocationTrackingCallback
   import software.amazon.location.tracking.config.LocationTrackerConfig
   import software.amazon.location.tracking.database.LocationEntry
   import software.amazon.location.tracking.filters.DistanceLocationFilter
   import software.amazon.location.tracking.filters.TimeLocationFilter
   import software.amazon.location.tracking.util.TrackingSdkLogLevel

   class MainActivity : ComponentActivity(), OnMapReadyCallback,
       MapLibreMap.OnCameraMoveStartedListener, MapLibreMap.OnCameraIdleListener {

       private val mainViewModel: MainViewModel by viewModels()
       private val poolId = "YOUR_AWS_IDENTITY_POOL_ID"
       private val trackerName = "YOUR_AWS_TRACKER_NAME"
       private val region = "YOUR_AWS_REGION"
       private val mapName = "YOUR_AWS_MAP_NAME"
   ```

```
    private val apiKey = "YOUR_AWS_API_KEY"
    private val coroutineScope = MainScope()
    private lateinit var locationCredentialsProvider:
LocationCredentialsProvider
    private lateinit var authHelper: AuthHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        MapLibre.getInstance(this)
        super.onCreate(savedInstanceState)
        setContent {
            TestMapAppTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MapLoadScreen(this, mainViewModel, onStartStopTrackingClick
= {
                        if (mainViewModel.isLocationTrackingForegroundActive) {
                            mainViewModel.isLocationTrackingForegroundActive =
false
                            mainViewModel.locationTracker?.stop()
                        } else {
                            if (checkLocationPermission(this))
return@MapLoadScreen
                            mainViewModel.isLocationTrackingForegroundActive =
true

mainViewModel.locationTracker?.start(locationTrackingCallback = object :
                                LocationTrackingCallback {
                                override fun
onLocationAvailabilityChanged(locationAvailable: Boolean) {
                                }

                                override fun onLocationReceived(location:
LocationEntry) {
                                }

                                override fun onUploadSkipped(entries:
LocationEntry) {
                                }

                                override fun onUploadStarted(entries:
ListLocationEntry) {
                                }
```

```
                                override fun onUploaded(entries:
ListLocationEntry) {
                                }

                            })
                        }
                    })
                }
            }
        }
        authenticateUser()
    }

    private fun authenticateUser() {
        coroutineScope.launch {
            authHelper = AuthHelper(applicationContext)
            locationCredentialsProvider =
authHelper.authenticateWithCognitoIdentityPool(
                poolId,
            )
            locationCredentialsProvider.let {
                val config = LocationTrackerConfig(
                    trackerName = trackerName,
                    logLevel = TrackingSdkLogLevel.DEBUG,
                    latency = 1000,
                    frequency = 5000,
                    waitForAccurateLocation = false,
                    minUpdateIntervalMillis = 5000,
                )
                mainViewModel.locationTracker = LocationTracker(
                    applicationContext,
                    it,
                    config,
                )

mainViewModel.locationTracker?.enableFilter(TimeLocationFilter())

mainViewModel.locationTracker?.enableFilter(DistanceLocationFilter())
            }
        }
    }
```

```
    private fun checkLocationPermission(context: Context) =
 ActivityCompat.checkSelfPermission(
        context,
        Manifest.permission.ACCESS_FINE_LOCATION,
    ) != PackageManager.PERMISSION_GRANTED &&
 ActivityCompat.checkSelfPermission(
        context,
        Manifest.permission.ACCESS_COARSE_LOCATION,
    ) != PackageManager.PERMISSION_GRANTED

    override fun onMapReady(map: MapLibreMap) {
        map.setStyle(
            Style.Builder()
                .fromUri(
                    "https://maps.geo.$region.amazonaws.com/maps/v0/maps/
$mapName/style-descriptor?key=$apiKey"
                ),
        ) {
            mainViewModel.mapLibreMap = map
            map.uiSettings.isAttributionEnabled = true
            map.uiSettings.isLogoEnabled = false
            map.uiSettings.attributionGravity = Gravity.BOTTOM or Gravity.END
            val initialPosition = LatLng(47.6160281982247, -122.32642111977668)
            map.cameraPosition = CameraPosition.Builder()
                .target(initialPosition)
                .zoom(14.0)
                .build()

            map.addOnCameraMoveStartedListener(this)
            map.addOnCameraIdleListener(this)
            map.cameraPosition.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }

    override fun onCameraMoveStarted(p0: Int) {
        mainViewModel.label = ""
        mainViewModel.isLabelAdded = false
```

```
    }

    override fun onCameraIdle() {
        if (!mainViewModel.isLabelAdded) {
            mainViewModel.mapLibreMap?.cameraPosition?.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }
}
```

위 코드는 를 사용하여 LocationTracker 객체를 만드는 방법과 를 사용하여 추적을 시작하고 중지하는 방법을 보여줍니다 LocationTracker. AuthHelper

- authenticateUser(): 이 메서드는 AuthHelper 및 LocationTracker 객체를 생성합니다.
- onStartStopTrackingClick: 이 콜백은 사용자가 추적 시작/중지 버튼을 클릭하면 트리거되며, 이 버튼을 클릭하면 Tracking SDK로 추적이 시작/중지됩니다.

5. Android 에뮬레이터로 테스트 경로를 생성하세요.

   a. Android 스튜디오를 사용하여 AVD를 실행하여 에뮬레이터를 엽니다.

   b. 에뮬레이터 툴바에서 더보기 (점 3개) 아이콘을 클릭하여 확장 컨트롤을 엽니다.

   c. 사이드바에서 위치를 선택하여 위치를 엽니다.

   d. GPX 데이터를 사용하거나 지도를 클릭하고 소스 및 목적지 데이터를 선택하여 경로를 생성합니다.

   e. 경로 재생을 클릭하여 시뮬레이션을 시작하고 GPS 경로 시뮬레이션을 시작합니다.

   f. 애플리케이션을 실행하고 시뮬레이션된 경로를 처리하는 방식을 관찰하여 애플리케이션을 테스트하십시오.

Android 퀵스타트 애플리케이션의 전체 데모입니다.

## 다음에 있는 것

이 애플리케이션의 소스 코드는 에서 사용할 수 있습니다 GitHub.

Amazon Location을 최대한 활용하려면 다음 리소스를 확인하세요.

- Amazon Location Service의 개념에 대해 더 자세히 알아보기
- Amazon Location 특징 및 기능을 사용하는 방법에 대한 자세한 정보를 확인하세요.
- Amazon Location을 사용하는 코드 예시를 살펴보고 이 샘플을 확장하고 더 복잡한 애플리케이션을 구축하는 방법 알아보기

# iOS 앱 만들기

이 섹션에서는 특정 위치를 검색하고 포그라운드에서 추적할 수 있는 iOS 애플리케이션을 생성합니다. 먼저 Amazon 위치 리소스와 애플리케이션을 위한 Amazon Cognito ID를 생성합니다.

주제

- 앱을 위한 Amazon Location 리소스 생성
- 애플리케이션에 대한 인증 설정
- 기본 iOS 애플리케이션 생성
- 초기 코드 설정
- 애플리케이션에 대화형 맵 추가
- 애플리케이션에 검색 추가
- 애플리케이션에 추적 추가
- 다음에 있는 것

## 앱을 위한 Amazon Location 리소스 생성

아직 리소스가 없는 경우 애플리케이션에서 사용할 Amazon Location 리소스를 생성해야 합니다. 애플리케이션에 지도를 표시하는 맵 리소스, 지도에서 위치를 검색하기 위한 장소 색인, 지도에서 객체를 추적하는 추적기를 생성합니다.

애플리케이션에 위치 리소스를 추가하려면

1. 사용하려는 맵 스타일을 선택합니다.

    a. Amazon Location 콘솔의 맵 페이지에서 맵 생성을 선택하여 맵 스타일을 미리 볼 수 있습니다.

b. 새 맵 리소스의 이름과 설명을 추가합니다. 맵 리소스에 사용하는 이름을 기록해 둡니다. 이는 튜토리얼 뒷부분에서 스크립트 파일을 만들 때 필요합니다.

c. 맵을 선택합니다.

> **Note**
>
> 맵 스타일을 선택하면 사용할 맵 데이터 공급자도 선택됩니다. 애플리케이션이 배송 차량이나 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 HERE만을 지리적 위치 제공업체로 사용할 수 있습니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

d. Amazon Location 이용 약관에 동의한 다음 맵 생성을 선택합니다. 선택한 맵과 상호 작용할 수 있습니다. 확대, 축소하거나 원하는 방향으로 이동할 수 있습니다.

e. 새 맵 리소스에 대해 표시되는 Amazon 리소스 이름(ARN)을 기록해 둡니다. 이 튜토리얼의 뒷부분에서 이를 사용하여 올바른 인증을 생성할 수 있습니다.

2. 사용하려는 장소 색인을 선택합니다.

a. Amazon Location 콘솔의 장소 색인 페이지에서 장소 색인 생성을 선택합니다.

b. 새 장소 색인 리소스의 이름과 설명을 추가합니다. 장소 색인 리소스에 사용하는 이름을 적어 둡니다. 이는 튜토리얼 뒷부분에서 스크립트 파일을 만들 때 필요합니다.

c. 데이터 공급자를 선택합니다.

> **Note**
>
> 대부분의 경우 이미 선택한 맵 공급자와 일치하는 데이터 공급자를 선택하세요. 이렇게 하면 검색 결과가 맵과 일치되게 할 수 있습니다.
> 애플리케이션이 배송 차량이나 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 HERE만을 지리적 위치 제공업체로 사용할 수 있습니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

d. 데이터 스토리지 옵션을 선택합니다. 이 튜토리얼에서는 결과가 저장되지 않으므로 아니오, 한 번만 사용합니다를 선택할 수 있습니다.

e. Amazon Location 이용 약관에 동의한 다음 장소 색인 생성을 선택합니다.

f. 새 장소 색인 리소스에 표시되는 ARN을 기록해 둡니다. 이는 튜토리얼의 다음 섹션에서 올바른 인증을 만드는 데 사용합니다.

3. Amazon 위치 콘솔을 사용하여 트래커를 만들려면

a. Amazon 위치 서비스 콘솔을 엽니다.
b. 왼쪽 탐색 창에서 트래커를 선택합니다.
c. 트래커 생성을 선택합니다.
d. 필수 필드를 모두 입력합니다.
e. 포지션 필터링에서는 기본 설정인 을 사용하는 것이 좋습니다. TimeBased
f. 트래커 생성을 선택하여 완료하세요.

## 애플리케이션에 대한 인증 설정

이 자습서에서 만든 애플리케이션은 익명으로 사용되므로 사용자가 로그인하지 않아도 애플리케이션을 사용할 AWS 수 있습니다. 하지만 Amazon Location Service API를 사용하려면 인증이 필요합니다. Amazon Cognito를 사용하여 익명 사용자에게 인증 및 권한 부여를 제공하게 됩니다. 이 자습서에서는 Amazon Cognito를 사용하여 애플리케이션을 인증합니다.

> **Note**
>
> Amazon Location Service와 함께 Amazon Cognito를 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. Amazon Location Service에 액세스 권한 부여

다음 자습서에서는 생성한 지도, 장소 색인, 추적기에 대한 인증을 설정하는 방법과 Amazon Location에 대한 권한을 설정하는 방법을 보여줍니다.

추적을 위한 IAM 정책을 생성하십시오.

1. 관리자 권한이 있는 사용자로 https://console.aws.amazon.com/iam/의 IAM 콘솔에 로그인합니다.
2. 탐색 창에서 정책을 선택합니다.
3. 콘텐츠 창에서 정책 생성을 선택합니다.
4. JSON 옵션을 선택한 다음 이 JSON 정책을 복사하여 JSON 텍스트 상자에 붙여넣습니다.

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
```

```
                    "geo:GetMapTile",
                    "geo:GetMapStyleDescriptor",
                    "geo:GetMapSprites",
                    "geo:GetMapGlyphs",
                    "geo:SearchPlaceIndexForPosition",
                    "geo:GetDevicePositionHistory",
                    "geo:BatchUpdateDevicePosition"
                ],
                "Resource": [
                    "arn:aws:geo:{Region}:{Account}:map/{MapName}",
                    "arn:aws:geo:{Region}:{Account}:place-index/{IndexName}",
                    "arn:aws:geo:{Region}:{Account}:tracker/{TrackerName}"
                ]
            }
        ]
    }
```

다음은 추적에 대한 정책 예제입니다. 이 예제를 자체 정책에 사용하려면, `Region` `Account``IndexName`, `MapName` 및 TrackerName자리 표시자를 바꾸십시오.

> **Note**
>
> 인증되지 않은 자격 증명 풀은 보안되지 않은 인터넷 사이트에 노출되기 위한 것이지만, 이러한 자격 증명 풀은 기간 제한이 있는 표준 자격 증명으로 교환된다는 점에 유의하십시오. AWS
>
> 인증되지 않은 자격 증명 풀과 관련된 IAM 역할의 범위를 적절하게 지정하는 것이 중요합니다. Amazon Location Service와 함께 Amazon Cognito에서 정책을 사용하고 적절하게 범위를 지정하는 방법에 대한 자세한 내용은 Amazon Location Service에 대한 액세스 권한 부여를 참조하십시오.

5. 검토 및 생성 페이지에서 정책 이름 필드에 이름을 입력합니다. 정책에서 부여한 권한을 검토한 다음 [Create Policy] 를 선택하여 작업 내용을 저장합니다.

새로운 정책이 관리형 정책 목록에 나타나며 연결 준비가 완료됩니다.

추적에 대한 인증을 설정하세요.

1. Amazon Cognito 콘솔에서 맵 애플리케이션에 대한 인증을 설정합니다.
2. 자격 증명 풀 페이지를 엽니다.

> **Note**
>
> 생성한 풀은 이전 섹션에서 생성한 Amazon Location Service 리소스와 동일한 AWS 계정 및 AWS 지역에 있어야 합니다.

3. 자격 증명 풀 생성을 선택합니다.
4. ID 풀 신뢰 구성 단계부터 시작합니다. 사용자 액세스 인증의 경우 게스트 액세스를 선택하고 다음을 누릅니다.
5. 권한 구성 페이지에서 기존 IAM 역할 사용을 선택하고 이전 단계에서 생성한 IAM 역할의 이름을 입력합니다. 준비가 되면 다음을 눌러 다음 단계로 넘어갑니다.
6. 속성 구성 페이지에서 자격 증명 풀의 이름을 입력합니다. 그런 다음 다음을 누릅니다.
7. 검토 및 생성 페이지에서 제공되는 모든 정보를 검토한 다음 ID 풀 생성을 누릅니다.
8. ID 풀 페이지를 열고 방금 생성한 자격 증명 풀을 선택합니다. 그런 다음 나중에 사용할 내용을 IdentityPoolId 브라우저 스크립트에 복사하거나 기록해 둡니다.

## 기본 iOS 애플리케이션 생성

이 자습서에서는 지도를 포함하는 iOS 애플리케이션을 만들어 사용자가 지도상의 특정 위치에 있는 내용을 찾을 수 있도록 합니다.

먼저 Xcode의 프로젝트 마법사를 사용하여 Swift 애플리케이션을 만들어 보겠습니다.

빈 애플리케이션 (Xcode) 을 만들려면

1. Xcode를 열고 메뉴에서 [파일], [새로 만들기], [새 프로젝트] 를 선택합니다.
2. iOS 탭에서 앱을 선택한 후 다음을 선택합니다.
3. 제품 이름, 조직 식별자를 입력하고 인터페이스 필드에 입력합니다`SwiftUI`. 선택을 완료하려면 다음을 선택합니다.
4. 프로젝트를 저장할 위치를 선택하고 생성 버튼을 눌러 빈 애플리케이션을 생성합니다.

기본 애플리케이션을 만든 후에는 샘플 앱에 필요한 패키지를 설치해야 합니다.

필수 종속성 설치

1. Xcode에서 프로젝트를 마우스 오른쪽 버튼으로 클릭하고 패키지 추가... 를 선택합니다. . 그러면 프로젝트에 패키지를 추가할 수 있는 패키지 창이 열립니다.

2. 패키지 창에서 다음 패키지를 추가합니다.

   - Maplibre 네이티브 패키지의 경우 다음 URL을 사용하세요: https://github.com/maplibre/.maplibre-gl-native-distribution URL에서 다음 패키지를 추가합니다. `maplibre-gl-native-distribution` 및. `Mapbox`
   - 아마존 위치 인증 iOS SDK의 경우 https://github.com/aws-geospatial/ amazon-location-mobile-auth -sdk-ios URL을 사용하십시오. URL에서 다음 패키지를 추가합니다. 및. `amazon-location-mobile-auth-sdk-ios` `AmazonLocationiOSAuthSDK`
   - 아마존 위치 추적 iOS SDK의 경우 https://github.com/aws-geospatial/ amazon-location-mobile-tracking -sdk-ios URL을 사용하십시오. URL에서 다음 패키지를 추가합니다. 및. `amazon-location-mobile-tracking-sdk-ios` `AmazonLocationiOSTrackingSDK`

## 초기 코드 설정

앱에서 위치 권한을 활성화하세요.

1. Xcode 프로젝트를 엽니다.
2. 프로젝트 `Info.plist` 파일을 찾습니다.
3. 앱의 요구 사항에 따라 위치 권한에 필요한 키를 추가합니다. 키는 다음과 같습니다.

   - `NSLocationWhenInUseUsageDescription`: 앱을 사용할 때 위치 액세스가 필요한 이유에 대한 설명.
   - `NSLocationAlwaysAndWhenInUseUsageDescription`: 앱에 지속적인 위치 액세스가 필요한 이유에 대한 설명.

이제 앱의 리소스 값을 구성해야 합니다. 라는 `Config.xcconfig` 새 파일을 추가하고 Amazon 콘솔에서 이전에 생성한 값을 입력합니다.

```
REGION =
INDEX_NAME =
MAP_NAME =
IDENTITY_POOL_ID =
TRACKER_NAME =
```

1. 왼쪽 네비게이터 섹션에서 프로젝트를 선택합니다.
2. 대상 섹션에서 앱을 선택하고 정보 탭을 클릭합니다.

3. 다음과 같은 값이 있는 정보 속성을 추가합니다.
4. 아래 내용이 포함된 `Config.swift` 파일을 추가합니다. 그러면 번들 정보 파일에서 구성 값을 읽을 수 있습니다.

```
import Foundation

enum Config {
    static let region = Bundle.main.object(forInfoDictionaryKey: "Region") as!
 String
    static let mapName = Bundle.main.object(forInfoDictionaryKey: "MapName") as!
 String
    static let indexName = Bundle.main.object(forInfoDictionaryKey: "IndexName")
 as! String
    static let identityPoolId = Bundle.main.object(forInfoDictionaryKey:
 "IdentityPoolId") as! String
    static let trackerName = Bundle.main.object(forInfoDictionaryKey:
 "TrackerName") as! String
}
```

5. 이름을 `ViewModel` 가진 새 폴더를 만들고 그 안에 `TrackingViewModel.swift` 파일을 추가합니다.

```
import SwiftUI
import AmazonLocationiOSAuthSDK
import MapLibre

final class TrackingViewModel : ObservableObject {
    @Published var trackingButtonText = NSLocalizedString("StartTrackingLabel",
 comment: "")
    @Published var trackingButtonColor = Color.blue
    @Published var trackingButtonIcon = "play.circle"
    @Published var region : String
    @Published var mapName : String
    @Published var indexName : String
    @Published var identityPoolId : String
    @Published var trackerName : String
    @Published var showAlert = false
    @Published var alertTitle = ""
    @Published var alertMessage = ""
    @Published var centerLabel = ""

    var clientIntialised: Bool
```

```
    var client: LocationTracker!
    var authHelper: AuthHelper
    var credentialsProvider: LocationCredentialsProvider?
    var mlnMapView: MLNMapView?
    var mapViewDelegate: MapViewDelegate?
    var lastGetTrackingTime: Date?
    var trackingActive: Bool

    init(region: String, mapName: String, indexName: String, identityPoolId:
 String, trackerName: String) {
        self.region = region
        self.mapName = mapName
        self.indexName = indexName
        self.identityPoolId = identityPoolId
        self.trackerName = trackerName
        self.authHelper = AuthHelper()
        self.trackingActive = false
        self.clientIntialised = false
    }

    func authWithCognito(identityPoolId: String?) {
        guard let identityPoolId =
 identityPoolId?.trimmingCharacters(in: .whitespacesAndNewlines)
        else {
            alertTitle = NSLocalizedString("Error", comment: "")
            alertMessage = NSLocalizedString("NotAllFieldsAreConfigured", comment:
 "")
            showAlert = true
            return
        }
        credentialsProvider =
 authHelper.authenticateWithCognitoUserPool(identityPoolId: identityPoolId)
        initializeClient()
    }

    func initializeClient() {
        client = LocationTracker(provider: credentialsProvider!, trackerName:
 trackerName)
        clientIntialised = true
    }
}
```

# 애플리케이션에 대화형 맵 추가

이제 애플리케이션에 맵 컨트롤을 추가해 보겠습니다. 이 자습서에서는 애플리케이션에서 맵 뷰를 관리하기 위해 AWS API를 사용합니다 MapLibre . 맵 컨트롤 자체는 MapLibre GL Native iOS 라이브러리의 일부입니다.

1. 다음 코드를 사용하여 Views 폴더 아래에 `MapView.swift` 파일을 추가합니다.

```
import SwiftUI
import MapLibre

struct MapView: UIViewRepresentable {
    var onMapViewAvailable: ((MLNMapView) -> Void)?
    var mlnMapView: MLNMapView?
    var trackingViewModel: TrackingViewModel

    func makeCoordinator() -> MapView.Coordinator {
        return Coordinator(self, trackingViewModel: trackingViewModel)
    }

    func makeUIView(context: Context) -> MLNMapView {
        let styleURL = URL(string: "https://maps.geo.
\(trackingViewModel.region).amazonaws.com/maps/v0/maps/
\(trackingViewModel.mapName)/style-descriptor")
        let mapView = MLNMapView(frame: .zero, styleURL: styleURL)
        mapView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
        mapView.setZoomLevel(15, animated: true)
        mapView.showsUserLocation = true
        mapView.userTrackingMode = .follow
        context.coordinator.mlnMapView = mapView
        mapView.delegate = context.coordinator

        mapView.logoView.isHidden = true
        context.coordinator.addCenterMarker()

        onMapViewAvailable?(mapView)
        trackingViewModel.mlnMapView = mapView
        return mapView
    }

    func updateUIView(_ uiView: MLNMapView, context: Context) {
    }
```

```
    class Coordinator: NSObject, MLNMapViewDelegate, MapViewDelegate {
        var control: MapView
        var mlnMapView: MLNMapView?
        var trackingViewModel: TrackingViewModel
        var centerMarker: MLNPointAnnotation?

        public init(_ control: MapView, trackingViewModel: TrackingViewModel) {
            self.control = control
            self.trackingViewModel = trackingViewModel
            super.init()
            self.trackingViewModel.mapViewDelegate = self
        }

        func mapViewDidFinishRenderingMap(_ mapView: MLNMapView, fullyRendered:
Bool) {
            if(fullyRendered) {
                mapView.accessibilityIdentifier = "MapView"
                mapView.isAccessibilityElement = false
            }
        }

        func addCenterMarker() {
            guard let mlnMapView = mlnMapView else {
                return
            }

            let centerCoordinate = mlnMapView.centerCoordinate
            let marker = MLNPointAnnotation()
            marker.coordinate = centerCoordinate
            marker.accessibilityLabel = "CenterMarker"
            mlnMapView.addAnnotation(marker)
            centerMarker = marker

            trackingViewModel.reverseGeocodeCenter(centerCoordinate:
centerCoordinate, marker: marker)
        }

        func mapView(_ mapView: MLNMapView, regionDidChangeAnimated animated: Bool)
{
            if let marker = centerMarker {
                                DispatchQueue.main.asyncAfter(deadline: .now() + 1.0)
{
                    mapView.deselectAnnotation(marker, animated: false)
                    marker.coordinate = mapView.centerCoordinate
```

```
                    let centerCoordinate = mapView.centerCoordinate
                    self.trackingViewModel.reverseGeocodeCenter(centerCoordinate:
 centerCoordinate, marker: marker)
                }
            }
        }
    }
}
```

2. ViewModel폴더 아래에 AWSSignatureV4Delegate 파일을 추가합니다. 이 파일은 맵을 렌더링하기 위한 모든 MapView http 요청과 함께 서명하는 데 사용됩니다.

```
import MapLibre
import AmazonLocationiOSAuthSDK

class AWSSignatureV4Delegate : NSObject, MLNOfflineStorageDelegate {
    private let awsSigner: AWSSigner

    init(credentialsProvider: LocationCredentialsProvider) {
        self.awsSigner = DENY LIST ERROR , serviceName: "geo")
        super.init()
    }

    func offlineStorage(_ storage: MLNOfflineStorage, urlForResourceOf kind:
 MLNResourceKind, with url: URL) -> URL {
        if url.host?.contains("amazonaws.com") != true {
            return url
        }
        let signedURL = awsSigner.signURL(url: url, expires: .hours(1))

        return signedURL
    }
}
```

3. Views 폴더에 UserLocationView.swift 파일을 추가합니다. 그러면 맵을 사용자 위치 중앙에 맞추는 버튼이 추가됩니다.

```
import SwiftUI

struct UserLocationView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
            Button(action: {
```

```
                trackingViewModel.locateMe()
            }) {
                Image(systemName: "scope")
                    .resizable()
                    .frame(width: 24, height: 24)
                    .padding(5)
                    .background(Color.white)
                    .foregroundColor(.blue)
                    .clipShape(RoundedRectangle(cornerRadius: 8))
                    .shadow(color: Color.black.opacity(0.3), radius: 3, x: 0, y: 2)
            }
            .accessibility(identifier: "LocateMeButton")
            .padding(.trailing, 10)
            .padding(.bottom, 10)
            .frame(maxWidth: .infinity, alignment: .trailing)
    }
}
```

4. 다음 코드를 사용하여 TrackingView.swift 파일을 추가합니다.

```
import SwiftUI

struct TrackingView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
        ZStack(alignment: .bottom) {
            MapView(trackingViewModel: trackingViewModel)
            VStack {
                UserLocationView(trackingViewModel: trackingViewModel)
            }
        }
        .onAppear() {
            if !trackingViewModel.identityPoolId.isEmpty {
                trackingViewModel.authWithCognito(identityPoolId:
 trackingViewModel.identityPoolId)
            }
        }
    }
}
```

이제 애플리케이션을 구축할 수 있습니다. 실행하려면 Xcode에서 에뮬레이션하도록 기기를 설정하거나 기기에서 앱을 사용해야 할 수 있습니다. 이 앱을 사용하면 맵 컨트롤이 어떻게 작동하는지 확인할

수 있습니다. 맵을 드래그하여 이동하고 손가락을 오므려 확대할 수 있습니다. 맵 컨트롤의 작동 방식을 직접 변경하여 애플리케이션의 요구 사항에 맞게 맵 컨트롤을 사용자 지정할 수 있습니다.

## 애플리케이션에 검색 추가

이제 특정 위치에서 항목을 찾을 수 있는 애플리케이션에 역지오코딩 검색을 추가할 것입니다. iOS 앱 사용을 단순화하기 위해 화면 중앙을 검색합니다. 새 위치를 찾으려면 지도를 검색하려는 위치로 이동하세요. 지도 중앙에 마커를 배치하여 검색 위치를 표시합니다.

1. 역지오코딩 검색과 관련된 TrackingViewModel `.swift` 파일에 다음 코드를 추가합니다.

```
func reverseGeocodeCenter(centerCoordinate: CLLocationCoordinate2D, marker:
 MLNPointAnnotation) {
    let position = [NSNumber(value: centerCoordinate.longitude), NSNumber(value:
 centerCoordinate.latitude)]
    searchPositionAPI(position: position, marker: marker)
}

func searchPositionAPI(position: [Double], marker: MLNPointAnnotation) {
    if let amazonClient = authHelper.getLocationClient() {
        Task {
            let searchRequest = SearchPlaceIndexForPositionInput(indexName:
 indexName, language: "en" , maxResults: 10, position: position)
            let searchResponse = try? await amazonClient.searchPosition(indexName:
 indexName, input: searchRequest)
            DispatchQueue.main.async {
                self.centerLabel = searchResponse?.results?.first?.place?.label ??
 ""
                self.mlnMapView?.selectAnnotation(marker, animated: true,
 completionHandler: {})
            }
        }
    }
}
```

2. 다음 코드로 TrackingView.swift 파일을 업데이트하면 맵뷰의 중앙 위치 주소가 표시됩니다.

```
import SwiftUI

struct TrackingView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
        ZStack(alignment: .bottom) {
```

```
            if trackingViewModel.mapSigningIntialised {
                MapView(trackingViewModel: trackingViewModel)
                VStack {
                    UserLocationView(trackingViewModel: trackingViewModel)
                    CenterAddressView(trackingViewModel: trackingViewModel)
                }
            }
            else {
                Text("Loading...")
            }
        }
        .onAppear() {
            if !trackingViewModel.identityPoolId.isEmpty {
                Task {
                    do {
                        try await trackingViewModel.authWithCognito(identityPoolId:
 trackingViewModel.identityPoolId)
                    }
                    catch {
                        print(error)
                    }
                }
            }
        }
    }
}
```

## 애플리케이션에 추적 추가

애플리케이션의 마지막 단계는 앱에 추적 기능을 추가하는 것입니다. 이 경우 앱에 추적 시작, 추적 중지, 추적 지점 가져오기 및 표시를 추가합니다.

1. 프로젝트에 `TrackingBottomView.swift` 파일을 추가합니다. 여기에는 사용자 위치 추적을 시작 및 중지하고 맵에 추적 지점을 표시하는 버튼이 있습니다.

```
import SwiftUI

struct TrackingBottomView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
           Button(action: {
               Task {
```

```
                    if(trackingViewModel.trackingButtonText ==
 NSLocalizedString("StartTrackingLabel", comment: "")) {
                        trackingViewModel.startTracking()
                    } else {
                        trackingViewModel.stopTracking()
                    }
                }
            }) {
                HStack {
                    Spacer()
                    Text("Tracking")
                        .foregroundColor(trackingViewModel.trackingButtonColor)
                        .background(.white)
                        .cornerRadius(15.0)

                    Image(systemName: trackingViewModel.trackingButtonIcon)
                        .resizable()
                        .frame(width: 24, height: 24)
                        .padding(5)
                        .background(.white)
                        .foregroundColor(trackingViewModel.trackingButtonColor)

                }
            }
            .accessibility(identifier: "TrackingButton")
            .background(.white)
            .clipShape(RoundedRectangle(cornerRadius: 8))
            .padding(.trailing, 10)
            .padding(.bottom, 40)
            .frame(width: 130, alignment: .trailing)
            .shadow(color: Color.black.opacity(0.3), radius: 3, x: 0, y: 2)
        }
}
```

2. 다음 코드로 TrackingView.swift 파일을 업데이트하세요.

```
import SwiftUI

struct TrackingView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
        ZStack(alignment: .bottom) {
            if trackingViewModel.mapSigningIntialised {
                MapView(trackingViewModel: trackingViewModel)
```

```
                VStack {
                    UserLocationView(trackingViewModel: trackingViewModel)
                    CenterAddressView(trackingViewModel: trackingViewModel)
                    TrackingBottomView(trackingViewModel: trackingViewModel)
                }
            }
            else {
                Text("Loading...")
            }
        }
        .onAppear() {
            if !trackingViewModel.identityPoolId.isEmpty {
                Task {
                    do {
                        try await trackingViewModel.authWithCognito(identityPoolId:
 trackingViewModel.identityPoolId)
                    }
                    catch {
                        print(error)
                    }
                }
            }
        }
    }
}
```

3. TrackingViewModel.swift파일에 다음 코드를 추가합니다. 이러한 함수는 추적 시작 및 중지를 담당합니다. 또한 사용자 위치 권한이 거부되면 오류 경고도 표시됩니다.

4. 포그라운드 트래킹을 구현하려면 다음 코드 예제를 복사하여 붙여넣습니다.

```
func showLocationDeniedRationale() {
        alertTitle = NSLocalizedString("locationManagerAlertTitle", comment: "")
        alertMessage = NSLocalizedString("locationManagerAlertText", comment: "")
        showAlert = true
    }

    // Required in info.plist: Privacy - Location When In Use Usage Description
    func startTracking() {
        do {
            print("Tracking Started...")
            if(client == nil) {
                initializeClient()
            }
```

```
            try client.startTracking()
            DispatchQueue.main.async { [self] in
                self.trackingButtonText = NSLocalizedString("StopTrackingLabel",
comment: "")
                self.trackingButtonColor = .red
                self.trackingButtonIcon = "pause.circle"
                trackingActive = true
            }
        } catch TrackingLocationError.permissionDenied {
            showLocationDeniedRationale()
        } catch {
            print("error in tracking")
        }
    }

    func stopTracking() {
        print("Tracking Stopped...")
        client.stopTracking()
        trackingButtonText = NSLocalizedString("StartTrackingLabel", comment: "")
        trackingButtonColor = .blue
        trackingButtonIcon = "play.circle"
        trackingActive = false
    }
```

> **Note**
>
> `startTracking`는 사용자의 위치 권한을 요청합니다. 애플리케이션은 사용 중 또는 Only Only Once 권한을 사용해야 합니다. 그렇지 않으면 애플리케이션에서 권한 거부 오류가 발생합니다.

추적 위치를 가져와 표시하려면 다음 절차를 따르십시오.

1. 사용자 기기에서 위치를 가져오려면 시작 및 종료 날짜와 시간을 제공해야 합니다. 단일 호출은 최대 100개의 추적 위치를 반환하지만, 추적 위치가 100개 이상인 경우 'NextToken' 값을 반환합니다. 지정된 시작 및 종료 시간에 대해 더 많은 추적 지점을 로드하려면 'NextToken'으로 후속 'getTrackerDevice위치' 호출을 호출해야 합니다.

```
func getTrackingPoints(nextToken: String? = nil) async throws {
        guard trackingActive else {
            return
```

```
        }
        // Initialize startTime to 24 hours ago from the current date and time.
        let startTime: Date = Date().addingTimeInterval(-86400)
        var endTime: Date = Date()
        if lastGetTrackingTime != nil {
            endTime = lastGetTrackingTime!
        }
        let result = try await client?.getTrackerDeviceLocation(nextToken:
nextToken, startTime: startTime, endTime: endTime)
        if let trackingData = result {

            lastGetTrackingTime = Date()
            let devicePositions = trackingData.devicePositions

            let positions = devicePositions!.sorted { (pos1:
LocationClientTypes.DevicePosition, pos2: LocationClientTypes.DevicePosition) ->
Bool in
                guard let date1 = pos1.sampleTime,
                      let date2 = pos2.sampleTime else {
                    return false
                }
                return date1 < date2
            }

            let trackingPoints = positions.compactMap { position ->
CLLocationCoordinate2D? in
                guard let latitude = position.position!.last, let longitude =
position.position!.first else {
                    return nil
                }
                return CLLocationCoordinate2D(latitude: latitude, longitude:
longitude)
            }
            DispatchQueue.main.async {
                self.mapViewDelegate!.drawTrackingPoints( trackingPoints:
trackingPoints)
            }
            if let nextToken = trackingData.nextToken {
                try await getTrackingPoints(nextToken: nextToken)
            }
        }
    }
```

2. 이제 파일의 코드를 다음 코드로 바꿉니다. `MapView.swift`

```
import SwiftUI
import MapLibre

struct MapView: UIViewRepresentable {
    var onMapViewAvailable: ((MLNMapView) -> Void)?
    var mlnMapView: MLNMapView?
    var trackingViewModel: TrackingViewModel

    func makeCoordinator() -> MapView.Coordinator {
        return Coordinator(self, trackingViewModel: trackingViewModel)
    }

    func makeUIView(context: Context) -> MLNMapView {
        let styleURL = URL(string: "https://maps.geo.
\(trackingViewModel.region).amazonaws.com/maps/v0/maps/
\(trackingViewModel.mapName)/style-descriptor")
        let mapView = MLNMapView(frame: .zero, styleURL: styleURL)
        mapView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
        mapView.setZoomLevel(15, animated: true)
        mapView.showsUserLocation = true
        mapView.userTrackingMode = .follow
        context.coordinator.mlnMapView = mapView
        mapView.delegate = context.coordinator

        mapView.logoView.isHidden = true
        context.coordinator.addCenterMarker()

        onMapViewAvailable?(mapView)
        trackingViewModel.mlnMapView = mapView
        return mapView
    }

    func updateUIView(_ uiView: MLNMapView, context: Context) {
    }

    class Coordinator: NSObject, MLNMapViewDelegate, MapViewDelegate {
        var control: MapView
        var mlnMapView: MLNMapView?
        var trackingViewModel: TrackingViewModel
        var centerMarker: MLNPointAnnotation?

        public init(_ control: MapView, trackingViewModel: TrackingViewModel) {
            self.control = control
```

```
            self.trackingViewModel = trackingViewModel
            super.init()
            self.trackingViewModel.mapViewDelegate = self
        }

        func mapViewDidFinishRenderingMap(_ mapView: MLNMapView, fullyRendered:
Bool) {
            if(fullyRendered) {
                mapView.accessibilityIdentifier = "MapView"
                mapView.isAccessibilityElement = false
            }
        }

        func addCenterMarker() {
            guard let mlnMapView = mlnMapView else {
                return
            }

            let centerCoordinate = mlnMapView.centerCoordinate
            let marker = MLNPointAnnotation()
            marker.coordinate = centerCoordinate
            marker.accessibilityLabel = "CenterMarker"
            mlnMapView.addAnnotation(marker)
            centerMarker = marker

            trackingViewModel.reverseGeocodeCenter(centerCoordinate:
centerCoordinate, marker: marker)
        }

        func mapView(_ mapView: MLNMapView, regionDidChangeAnimated animated: Bool)
{
            if let marker = centerMarker {
                DispatchQueue.main.asyncAfter(deadline: .now() + 1.0) {
                    mapView.deselectAnnotation(marker, animated: false)
                    marker.coordinate = mapView.centerCoordinate
                    let centerCoordinate = mapView.centerCoordinate
                    self.trackingViewModel.reverseGeocodeCenter(centerCoordinate:
centerCoordinate, marker: marker)
                }
            }
        }

        func mapView(_ mapView: MLNMapView, viewFor annotation: MLNAnnotation) ->
MLNAnnotationView? {
```

```
            guard let pointAnnotation = annotation as? MLNPointAnnotation else {
                return nil
            }

            let reuseIdentifier: String
            var color: UIColor = .black
            if pointAnnotation.accessibilityLabel == "Tracking" {
                reuseIdentifier = "TrackingAnnotation"
                color = UIColor(red: 0.00784313725, green: 0.50588235294, blue:
0.58039215686, alpha: 1)
            } else if pointAnnotation.accessibilityLabel == "LocationChange" {
                reuseIdentifier = "LocationChange"
                color = .gray
            } else {
                reuseIdentifier = "DefaultAnnotationView"
            }

            var annotationView =
mapView.dequeueReusableAnnotationView(withIdentifier: reuseIdentifier)

            if annotationView == nil {
                if reuseIdentifier != "DefaultAnnotationView" {
                    annotationView = MLNAnnotationView(annotation: annotation,
reuseIdentifier: reuseIdentifier)
                    //If point annotation is an uploaded Tracking point the radius
is 20 and color is blue, otherwise radius is 10 and color is gray
                    let radius = pointAnnotation.accessibilityLabel == "Tracking" ?
20:10
                    annotationView?.frame = CGRect(x: 0, y: 0, width: radius,
height: radius)
                    annotationView?.backgroundColor = color
                    annotationView?.layer.cornerRadius = 10

                    if pointAnnotation.accessibilityLabel == "Tracking" {
                        annotationView?.layer.borderColor = UIColor.white.cgColor
                        annotationView?.layer.borderWidth = 2.0
                        annotationView?.layer.shadowColor = UIColor.black.cgColor
                        annotationView?.layer.shadowOffset = CGSize(width: 0,
height: 2)
                        annotationView?.layer.shadowRadius = 3
                        annotationView?.layer.shadowOpacity = 0.2
                        annotationView?.clipsToBounds = false
                    }
                }
```

```
                else {
                    return nil
                }
            }

            return annotationView
        }

        func mapView(_ mapView: MLNMapView, didUpdate userLocation:
MLNUserLocation?) {
            if (userLocation?.location) != nil {
                if trackingViewModel.trackingActive {
                    let point = MLNPointAnnotation()
                    point.coordinate = (userLocation?.location!.coordinate)!
                    point.accessibilityLabel = "LocationChange"
                    mapView.addAnnotation(point)
                    Task {
                        do {
                            try await trackingViewModel.getTrackingPoints()
                        }
                        catch {
                            print(error)
                        }
                    }
                }
            }
        }

        func checkIfTrackingAnnotationExists(on mapView: MLNMapView, at
coordinates: CLLocationCoordinate2D) -> Bool {
            let existingAnnotation = mapView.annotations?.first(where: { annotation
in
                guard let annotation = annotation as? MLNPointAnnotation else
{ return false }
                return annotation.coordinate.latitude == coordinates.latitude &&
                annotation.coordinate.longitude == coordinates.longitude &&
annotation.accessibilityLabel == "Tracking" })
            return existingAnnotation != nil
        }

        public func drawTrackingPoints(trackingPoints: [CLLocationCoordinate2D]?) {
            guard let mapView = mlnMapView, let newTrackingPoints =
trackingPoints, !newTrackingPoints.isEmpty else {
                return
```

```
            }

            let uniqueCoordinates = newTrackingPoints.filter { coordinate in
                !checkIfTrackingAnnotationExists(on: mapView, at: coordinate)
            }

            let points = uniqueCoordinates.map { coordinate -> MLNPointAnnotation
 in
                let point = MLNPointAnnotation()
                point.coordinate = coordinate
                point.accessibilityLabel = "Tracking"
                return point
            }
            mapView.addAnnotations(points)
        }
    }
}

protocol MapViewDelegate: AnyObject {
    func drawTrackingPoints(trackingPoints: [CLLocationCoordinate2D]?)
}
```

문자열 값을 지역화하려면 다음 절차를 사용하십시오.

1. 라는 Localizable.xcstrings 새 파일을 만들고 추가합니다.
2. Localizable.xcstrings파일을 마우스 오른쪽 버튼으로 클릭하고 소스 코드로 엽니다.
3. 내용을 다음과 같이 바꾸십시오.

```
{
  "sourceLanguage" : "en",
  "strings" : {
    "Cancel" : {
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "Cancel"
          }
        }
      }
```

```
    },
    "Error" : {
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "Error"
          }
        }
      }
    },
    "Loading..." : {

    },
    "locationManagerAlertText" : {
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "Allow \\\"Quick Start App\\\" to use your location"
          }
        }
      }
    },
    "locationManagerAlertTitle" : {
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "We need your location to detect your location in map"
          }
        }
      }
    },
    "NotAllFieldsAreConfigured" : {
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "Not all the fields are configured"
```

```
        }
      }
    }
  },
  "OK" : {
    "extractionState" : "manual",
    "localizations" : {
      "en" : {
        "stringUnit" : {
          "state" : "translated",
          "value" : "OK"
        }
      }
    }
  },
  "StartTrackingLabel" : {
    "localizations" : {
      "en" : {
        "stringUnit" : {
          "state" : "translated",
          "value" : "Start Tracking"
        }
      }
    }
  },
  "StopTrackingLabel" : {
    "localizations" : {
      "en" : {
        "stringUnit" : {
          "state" : "translated",
          "value" : "Stop Tracking"
        }
      }
    }
  },
  "Tracking" : {

  }
},
"version" : "1.0"
}
```

4. 파일을 저장하고 앱을 빌드하고 실행하여 기능을 미리 보세요.

5. 위치 권한을 허용하고 추적 버튼을 탭합니다. 앱이 사용자 위치 업로드를 시작하고 Amazon 위치 추적기에 업로드합니다. 또한 지도에 사용자 위치 변경, 추적 지점, 현재 주소도 표시됩니다.

퀵스타트 신청이 완료되었습니다. 이 자습서에서는 다음과 같은 iOS 애플리케이션을 만드는 방법을 보여 주었습니다.

- 사용자가 상호 작용할 수 있는 맵을 만듭니다.
- 사용자가 맵 뷰를 변경하는 것과 관련된 여러 맵 이벤트를 처리합니다.
- 특히 Amazon 로케이션의 API를 사용하여 특정 위치의 지도를 검색하기 위해 Amazon Location Service searchByPosition API를 호출합니다.

## 다음에 있는 것

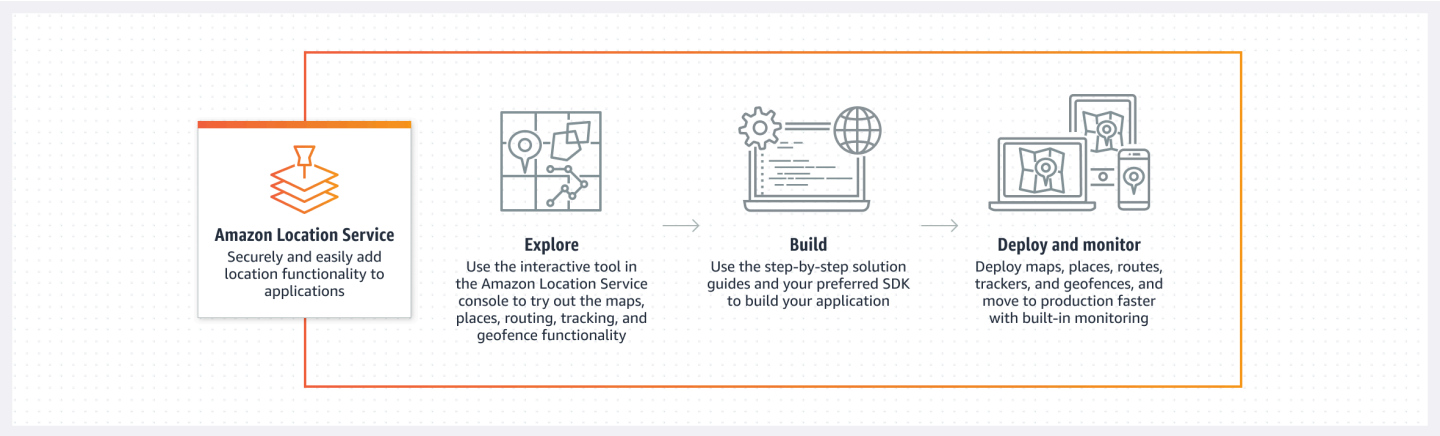
이 애플리케이션의 소스 코드는 에서 확인할 수 있습니다 GitHub.

Amazon Location을 최대한 활용하려면 다음 리소스를 확인하세요.

- Amazon Location Service의 개념에 대해 더 자세히 알아보기
- Amazon Location 특징 및 기능을 사용하는 방법에 대한 자세한 정보를 확인하세요.
- Amazon Location을 사용하는 코드 예시를 살펴보고 이 샘플을 확장하고 더 복잡한 애플리케이션을 구축하는 방법 알아보기

# Amazon Location Service 개념

Amazon Location Service를 사용하면 애플리케이션에 위치 데이터를 안전하게 추가할 수 있습니다. Amazon Location 콘솔에서 제공되는 시각적이고 대화형 도구를 사용하여 일부 기능을 살펴보세요. 탐색 도구를 사용하면 기본 맵을 조작하고, 관심 지점을 검색하고, 관심 영역 주변에 지오펜스를 그리고, 디바이스 위치를 트래커로 전송하는 것을 시뮬레이션할 수 있습니다.




구축할 준비가 되면 리소스를 생성하고 다양한 맵 스타일과 데이터 공급자 중에서 선택하세요. 그런 다음 이 안내서의 지침에 따라 개발 환경에 SDK 맞는 제품을 설치하고 Amazon APIs Location을 사용할 수 있습니다. 또한 Amazon CloudWatch 및 을 사용하여 모니터링을 통합할 수 AWS CloudTrail있습니다.

이 섹션의 항목에서는 Amazon Location 핵심 개념에 대한 개요를 제시하고 자체 애플리케이션에서 위치 관련 작업을 시작할 수 있도록 준비합니다.

주제

- Amazon Location 개요
- 맵
- 장소 검색
- 경로
- 지오펜스 및 트래커
- Amazon Location Service 사용에 대한 일반적인 사용 사례
- 데이터 공급자란 무엇입니까?
- Amazon Location 리전 및 엔드포인트
- Amazon Location Service Quotas

# Amazon Location 개요

Amazon Location Service는 AWS 리소스를 통해 위치 기반 기능 및 데이터 공급자에 대한 액세스를 제공합니다. Amazon Location은 필요한 기능 유형에 따라 다섯 가지 유형의 AWS 리소스를 제공합니다. 서로 다른 리소스를 함께 사용하여 완전한 위치 기반 애플리케이션을 만드세요. Amazon 로케이션 콘솔, Amazon 로케이션 또는 를 사용하여 이러한 리소스를 하나 이상 생성할 수 SDKs 있습니다. APIs

각 리소스는 사용할 기본 데이터 공급자(해당하는 경우)를 정의하고 해당 유형과 관련된 기능에 대한 액세스를 제공합니다.

예:

- Amazon Location Service Maps를 사용하면 맵 공급자로부터 맵을 선택하여 모바일 또는 웹 애플리케이션에서 사용할 수 있습니다.
- Amazon Location Service Places를 사용하면 관심 지점 검색, 텍스트 일부 작성, 지오코딩 및 역지오코딩에 사용할 데이터 공급자를 선택할 수 있습니다.
- Amazon Location Service Routes를 사용하면 데이터 공급자를 선택하고 up-to-date 도로 및 실시간 교통 정보를 기반으로 경로를 찾고 이동 시간을 추정할 수 있습니다.
- Amazon Location Service Geofences를 사용하면 관심 영역을 가상 경계로 정의할 수 있습니다. 그런 다음 이를 기준으로 위치를 평가하고 진입 및 종료 이벤트에 대한 알림을 받을 수 있습니다.
- Amazon Location Service Trackers는 디바이스로부터 위치 업데이트를 수신합니다. 트래커를 지오펜스 컬렉션에 연결하여 모든 위치 업데이트가 지오펜스에 대해 자동으로 평가되도록 할 수 있습니다.

IAM정책을 사용하여 Amazon Location 리소스에 대한 액세스를 관리하고 승인할 수 있습니다. 또한 리소스 수가 증가함에 따라 리소스를 리소스 그룹으로 구성하여 작업을 관리하고 자동화할 수 있습니다. AWS 리소스 관리에 대한 자세한 내용은 AWSResource Groups란? 을 참조하십시오. AWSResource Groups 사용 설명서에서.

위치는 일반적으로 글로벌 포지셔닝 시스템 () 서비스의 표준 좌표 참조 체계로 사용되는 세계 측지계 (WGS84) 를 따르는 위도 및 경도 좌표를 사용하여 정의됩니다. GPS

다음 섹션에서는 Amazon Location의 구성 요소가 작동하는 방식을 설명합니다.

# 맵

Amazon Location Service Map 리소스를 사용하면 맵의 기본 베이스맵 데이터에 액세스할 수 있습니다. 맵 리소스를 맵 렌더링 라이브러리와 함께 사용하여 애플리케이션에 대화형 맵을 추가할 수 있습니다. 애플리케이션에 필요에 따라 마커(또는 핀), 경로, 다각형 영역 등의 다른 기능을 맵에 추가할 수 있습니다.

> **Note**
>
> 실제로 맵 리소스를 사용하는 방법에 대한 자세한 내용은 애플리케이션에서 Amazon Location Map 사용 섹션을 참조하세요.

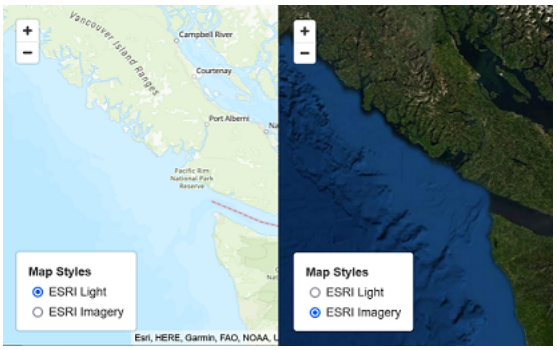
다음은 맵 리소스를 생성하고 사용하는 방법에 대한 개요입니다.


1. 데이터 공급자에서 맵 스타일을 선택하여 AWS 계정에서 맵 리소스를 생성합니다.
2. 그런 다음 개발 환경 및 애플리케이션에 SDK 맞는 것을 선택하여 설치할 수 있습니다. 사용 가능한 옵션에 대한 자세한 내용은 Amazon Location 액세스에 관한 항목을 참조하세요.
3. 응용 프로그램에 맵을 표시하려면 맵 리소스를 Amplify MapLibre, 또는 Tangram과 같은 렌더링 라이브러리와 결합하십시오. 자세한 내용은 이 가이드의 맵 사용하기를 참조하세요.
4. 그런 다음 Amazon 등의 CloudWatch 서비스와 Amazon Location을 사용하여 모니터링을 AWS CloudTrail 통합할 수 있습니다. 자세한 내용은 아마존을 통한 아마존 로케이션 서비스 모니터링 CloudWatch 및 을 참조하십시오AWS CloudTrail을 사용하여 로깅 및 모니터링.

## 맵 스타일

맵 리소스를 생성할 때 해당 리소스에 대한 맵 스타일을 선택해야 합니다. 맵 스타일은 렌더링된 맵의 모양을 정의합니다. 예를 들어, 다음 이미지는 Amazon Location의 서로 다른 맵 리소스에서 가져온 두 가지 스타일을 사용하는 동일한 데이터 공급자를 보여줍니다. 한 가지 스타일은 맵의 벡터 데이터를 기반으로 하는 일반적인 도로 스타일입니다. 다른 하나는 위성 이미지를 보여주는 래스터 데이터를 포함합니다. 맵을 확대하거나 축소하면 스타일이 변경될 수 있지만 일반적으로 스타일은 일관된 테마를 갖

습니다. 스타일 정보를 맵 렌더링 라이브러리로 전달하기 전에 스타일 정보의 일부 또는 전체를 재정의할 수 있습니다.




## 정치적 관점

Amazon Location Service의 특정 맵 스타일은 추가적인 정치적 견해를 지원합니다.

> **Note**
>
> 정치적 견해는 Amazon Location Service를 통해 액세스하는 맵, 이미지, 기타 데이터 및 타사 콘텐츠가 제공되는 국가 또는 리전의 매핑에 관한 법률을 포함하여 준거법을 준수하여 사용해야 합니다.

다음 지도 스타일은 인도 (IND) 정치 관점을 뒷받침합니다.

- Esri 맵 스타일:
  - Esri Navigation
  - Esri Light
  - Esri Street Map
  - Esri 다크 그레이 캔버스
  - Esri Light Gray Canvas
- Open Data 맵 스타일:
  - Open Data 스탠더드 라이트
  - Open Data 스탠더드 다크
  - Open Data 시각화 라이트
  - Open Data 시각화 다크

Amazon Location Service 콘솔에서는 표시된 스타일을 필터링하여 인도의 정치적 견해를 지원하는 스타일만 표시할 수 있습니다.

## 사용자 지정 계층

사용자 정의 레이어는 지도 스타일에서 활성화할 수 있는 추가 레이어입니다. 현재는 VectorEsriNavigation 맵 스타일만 `POI` 사용자 정의 레이어를 지원합니다.

`POI` 사용자 정의 레이어를 활성화하면 상점, 서비스, 레스토랑, 명소, 기타 관심 지점 등 다양한 장소가 지도에 추가됩니다. 기본적으로 사용자 정의 레이어는 `unset`입니다. 자세한 내용은 위치 MapConfigurationAPI참조를 참조하십시오.

## 맵 렌더링

애플리케이션에서 맵을 렌더링하려면 일반적으로 맵 렌더링 라이브러리를 사용합니다. 라이브러리에는 다음과 같은 몇 가지 일반적인 옵션을 사용할 수 있습니다.

- MapLibre— MapLibre 대화형 맵을 렌더링하기 위한 오픈 소스 라이브러리로, Amazon Location Service에서 맵을 렌더링할 때 가장 많이 사용하는 방법입니다. MapLibre 데이터 소스 (예: Amazon Location map 리소스) 에서 래스터 및 벡터 데이터를 렌더링하는 기능이 포함됩니다. MapLibre 확장하여 맵에 자체 데이터를 그릴 수 있습니다.
- Amplify – Amplify는 웹, iOS, Android 등을 위한 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. 애플리케이션이 Amplify를 사용하는 경우 Amazon Location 기능을 포함하도록 애플리케이션을 확장할 수 있습니다. Amplify에는 렌더링 맵을 비롯한 Amazon Location 기반 애플리케이션을 만들기 위한 라이브러리가 포함되어 있습니다. Amplify는 지도를 MapLibre 렌더링하는 데 사용하지만 Amazon Location Service에서만 사용할 수 있는 추가 기능을 제공하여 더욱 효율적으로 사용하고 검색 및 기타 기능도 추가합니다.
- Tangram — Tangram은 다음과 비슷한 대화형 맵을 렌더링하는 대체 오픈 소스 라이브러리입니다. MapLibre

맵 렌더링 라이브러리는 실행 시 Amazon Location Service에서 데이터를 가져와서 선택한 맵 리소스를 기반으로 맵 데이터를 렌더링합니다. 맵 리소스는 사용할 데이터 공급자와 맵 스타일을 정의합니다.

다음 이미지는 Amazon Location Service에서 맵 리소스를 맵 렌더링 라이브러리와 함께 사용하여 최종 맵을 생성하는 방법을 보여줍니다.


1. AWS Management Console 또는 를 사용하여 Amazon Location Service에서 맵 리소스를 생성합니다 AWS CLI. 이는 데이터 공급자와 사용하려는 맵 스타일을 정의합니다.
2. 애플리케이션에는 맵 렌더링 라이브러리가 포함되어 있습니다. 맵 렌더링 라이브러리에 사용할 맵 리소스의 이름을 지정합니다. 맵 렌더링 라이브러리는 Amazon Location에서 해당 맵 리소스에 대한 데이터 및 스타일 정보를 가져와 화면에 맵을 렌더링합니다.

## 맵 용어

맵 리소스

선택한 공급자의 맵 데이터에 액세스할 수 있습니다. 맵 리소스를 사용하여 맵 데이터가 포함된 맵 타일을 가져오고 스타일 설명자를 사용하여 형상이 맵에 렌더링되는 방식을 지정할 수 있습니다.

베이스맵

벡터 타일 레이어로 저장되는 지리적 컨텍스트를 맵에 제공합니다. 타일 레이어에는 시각적 참조를 위해 도로 이름, 건물, 토지 이용과 같은 지리적 컨텍스트가 포함됩니다.

벡터

벡터 데이터는 점, 선 및 다각형으로 구성된 형상 데이터입니다. 주로 도로, 위치, 지역을 맵에 저장하고 표시하는 데 사용됩니다. 벡터 모양은 맵의 마커 아이콘으로도 사용할 수 있습니다.

래스터

래스터 데이터는 일반적으로 색상으로 구성된 그리드로 이루어진 이미지 데이터입니다. 지형, 위성 이미지 또는 히트 맵과 같은 연속 데이터를 맵에 저장하고 표시하는 데 주로 사용됩니다. 래스터 이미지는 이미지나 아이콘으로도 사용할 수 있습니다.

맵 스타일

벡터 데이터에는 최종 맵을 만들기 위해 데이터 레이어를 그리는 방법에 대한 정보가 기본적으로 포함되어 있지 않습니다. 맵 스타일은 데이터의 색상 및 기타 스타일 정보를 정의하여 렌더링 시 모양을 정의합니다. 맵 리소스에는 맵의 스타일 정보가 포함됩니다.

Amazon Location Service는 Mapbox GL 스타일 사양에 따라 스타일을 제공합니다.

벡터 타일

벡터 모양을 사용하여 맵 데이터를 저장하는 타일 형식입니다. 이 데이터를 통해 최적의 성능을 위해 파일 크기를 작게 유지하면서 디스플레이 해상도에 맞게 조정하고 다양한 방법으로 기능을 선택적으로 렌더링할 수 있는 맵이 만들어집니다.

지원되는 벡터 파일 형식: Mapbox 벡터 타일 (MVT).

글리프 파일

인코딩된 유니코드 문자가 들어 있는 바이너리 파일입니다. 맵 렌더러에서 레이블을 표시하는 데 사용됩니다.

스프라이트 파일

위치 설명이 포함된 작은 래스터 이미지가 들어 있는 휴대용 네트워크 그래픽 (PNG) 이미지 파일입니다JSON. 맵 렌더러가 맵에 아이콘이나 텍스처를 렌더링하는 데 사용됩니다.

# 장소 검색

Amazon Location Service의 주요 기능 중 하나는 지리적 위치 정보를 검색하는 기능입니다. Amazon Location은 장소 색인 리소스를 통해 이 기능을 제공합니다.

> **Note**
>
> 실제로 장소 색인 리소스를 사용하여 검색하는 방법에 대한 자세한 내용은 Amazon Location을 사용하여 장소 및 지리적 위치 데이터 검색 섹션을 참조하세요.

장소 색인을 APIs 사용하여 다음을 검색할 수 있습니다.

- 레스토랑 및 랜드마크와 같은 관심 장소. 이름과 선택적인 위치를 기준으로 주위를 검색하고 관련성에 따라 정렬된 옵션 목록을 받을 수 있습니다.
- 해당 주소의 위도와 경도를 수신하는 거리 주소. 이를 지오코딩이라고 합니다.
- 관련 거리 주소 또는 위치에 대한 기타 정보를 수신하는 위도 및 경도 위치입니다. 이를 역지오코딩이라고 합니다.
- 일반적으로 사용자가 입력할 때 일부 또는 철자가 틀린 자유 형식 텍스트 쿼리입니다. 이를 자동 완성, 자동 제안 또는 퍼지 매칭이라고 합니다.

장소 색인에는 검색에 사용할 데이터 공급자가 포함됩니다.

> **Note**
>
> 정확한 위치를 포함한 맵 데이터 및 기타 지리적 위치 정보는 데이터 공급자마다 다를 수 있습니다. 가장 좋은 방법은 장소 색인, 맵 및 기타 Amazon Location 리소스에 동일한 데이터 공급자를 사용하는 것입니다. 예를 들어 장소 색인에서 반환된 장소가 맵 리소스에서 제공하는 동일한 장소의 위치와 일치하지 않는 경우, 맵에서 잘못된 위치로 보이는 곳에 마커를 배치할 수 있습니다.

다음은 장소 색인 리소스를 생성하고 사용하는 방법을 보여줍니다.


1. 먼저 데이터 공급자를 선택하여 AWS 계정에서 장소 색인 리소스를 생성합니다.
2. 그런 다음 개발 환경 및 애플리케이션에 SDK 맞는 것을 선택하여 설치할 수 있습니다. 사용 가능한 옵션에 대한 자세한 내용은 Amazon Location 액세스에 관한 항목을 참조하세요.
3. Amazon 로케이션 플레이스 사용을 시작하십시오APIs. 자세한 내용은 장소 검색 사용에 관한 항목을 참조하세요.
4. 그런 다음 Amazon CloudWatch 및 같은 서비스를 사용하여 모니터링을 통합할 수 AWS CloudTrail 있습니다. 자세한 내용은 the section called “를 통한 모니터링 CloudWatch” 및 을 참조하십시오the section called “아마존 CloudTrail 로케이션과 함께 사용”.

# 지오코딩 개념

Amazon Location 장소 색인은 검색할 텍스트를 지정할 수 있는 SearchPlaceIndexForText(이)라는 작업을 제공합니다. 예를 들어, 다음을 검색합니다.

- 장소 – **Paris** 검색 시 프랑스 내 도시 위치가 반환될 수 있습니다.
- 사업체 – **coffee shop** 검색 시 커피숍 이름과 위치를 포함한 커피숍 목록이 반환될 수 있습니다. 검색할 위치를 지정하거나 검색할 경계 상자를 지정하여 검색 결과의 관련성을 높일 수도 있습니다. 이 경우 워싱턴 주 시애틀 시내의 위치를 제공하면 해당 지역의 커피숍이 반환됩니다.
- 주소 – **1600 Pennsylvania Ave, Washington D.C.** 검색 시 미국 내 백악관(해당 주소)의 위치가 반환될 수 있습니다.

이러한 방식으로 텍스트를 검색하는 것을 일반적으로 지오코딩이라고 하며, 여기에는 주소 또는 장소의 지리적 위치를 찾는 작업이 포함됩니다.

Amazon Location Service는 SearchPlaceIndexForPosition(이)라는 역지오코딩 작업도 제공합니다. 이렇게 하면 지리적 위치를 가져와 해당 위치에 있는 주소, 회사 또는 기타 정보가 반환됩니다.

## 검색 결과

Amazon Location Service에서 검색 요청을 성공적으로 수행하면 하나 이상의 결과가 반환됩니다. 각 결과에는 결과의 이름 또는 설명인 레이블이 포함됩니다. 예를 들어, **coffee shop** 검색 시 레이블 `Hometown Cafe`과(와) 함께 “Hometown Cafe”라는 커피숍이 검색되었음을 알려주는 결과가 반환될 수 있습니다. 검색 결과에는 일반적으로 구조화된 주소(주소 번호, 단위, 도로명, 우편번호 등의 속성 포함)도 포함됩니다. 데이터 공급자에 따라 국가 및 시간대와 같은 다른 메타데이터도 포함됩니다.

업체 이름 또는 카테고리(예: **coffee shop**)를 검색하는 경우 반환된 모든 결과를 맵에 표시할 수 있습니다. 주소 검색의 경우 첫 번째 결과를 자동으로 사용할 수 있습니다. 관련성에 대한 자세한 내용은 다음 항목을 참조하세요.

## 다양한 결과 및 관련성

텍스트로 검색할 때 Amazon Location Service는 종종 하나 이상의 결과를 찾습니다. 예를 들어, **Paris** 검색 시 프랑스의 도시뿐만 아니라 텍사스의 도시도 반환될 수 있습니다. 결과는 데이터 공급자가 결정한 관련성을 기준으로 정렬됩니다.

> **Note**
>
> 결과는 모든 공급자로부터 관련성 순으로 반환됩니다. Esri 또는 Grab을 데이터 공급자로 선택하면 단일 요청 결과 간의 상대적 관련성을 이해하는 데 사용할 수 있는 관련성 값이 결과에 포함됩니다.

국가 이름, 검색할 위치 등의 추가 정보를 지정하면 결과 순서가 변경되거나, 결과 수가 줄어들거나, 반환되는 결과 집합이 변경될 수도 있습니다. 예를 들어, 검색 시 텍사스 내 위치한 **Paris**을(를) 검색하면 `Paris, France`보다 `Paris, Texas`이(가) 첫 번째 결과로 반환될 가능성이 더 높습니다.

대화형 애플리케이션에서는 관련성을 사용하여 상위 결과를 수락할지 여부를 결정하거나 반환된 여러 결과를 명확하게 구분하도록 사용자에게 요청할 수 있습니다. 첫 번째 결과의 관련성이 높으면 그냥 정답으로 받아들일 수도 있습니다. 관련성이 높은 결과가 여러 개 있거나 관련성이 높은 결과가 없는 경우, 결과를 나열하고 사용자가 최상의 결과를 선택하도록 하는 것이 좋습니다.

## 주소 결과

동일한 `SearchPlaceIndexForText` 작업을 사용하여 Amazon Location Service로 주소를 검색할 수 있습니다. 더 많은 정보를 제공할수록 반환된 주소가 제공된 주소와 일치할 가능성이 높아집니다. 예를 들어 **`123 Main St`**은(는) **`123 Main St, Anytown, California, 90210`**보다 정확한 결과를 찾을 가능성이 낮습니다.

주소에는 번지, 도로명, 도시, 리전, 우편번호 등과 같은 여러 속성이 있습니다. 이러한 속성은 장소 색인에서 최대한 많은 측면과 일치하는 주소를 찾는 데 사용됩니다. 더 많은 속성이 발견될수록 일치하는 항목의 관련성이 더 높아지며 반환될 가능성도 높아집니다.

> **Note**
>
> 주소 결과의 관련성은 결과가 입력과 얼마나 일치하는지에 따라 달라집니다. 이는 일치하는 속성의 수일 수도 있고 결과가 입력과 얼마나 밀접하게 일치하는지도 나타낼 수 있습니다. 예를 들어, **`123 Main St`** 입력은 `Maine St`이(가) 유일한 결과인 경우에 비해 데이터에서 `Main St`이(가) 발견될 때 관련성이 더 높습니다. `Maine St`은(는) 여전히 반환되지만 관련성 값은 더 낮을 가능성이 있습니다.

검색 결과에는 전체 주소(`123 Main St, Anytown, California, 90210`)에 대한 레이블뿐만 아니라 반환된 주소의 구조화된 개별 속성도 포함됩니다. 이는 예를 들어 데이터베이스의 주소 필드를 채

우거나 결과를 검토하여 찾은 위치의 도시, 리전 또는 우편 번호를 찾는 데 사용할 수 있기 때문에 유용합니다.

보간

장소 색인 데이터의 주소에는 정확한 주소 일치를 포함됩니다. 예를 들어, 다음 그림과 같이 거리, `9th street`이(가) 있고 하나의 블록에 `220`, `240`이라는 집 두 채가 있다고 가정해 보겠습니다.


데이터 공급자는 알려진 두 주소를 사용하여 지리적 위치 데이터를 생성합니다. 이 두 주소를 검색하면 찾을 수 있습니다. 데이터 공급자가 맵 데이터를 생성한 후 처음 두 주소 사이에 새 집이 추가되었다고 가정해 보겠습니다. 이 새 집에는 `230`(이)라는 주소가 지정됩니다. **`230 S 9th St`** 검색 시에도 데이터 공급자는 여전히 결과를 찾을 수 있습니다. 알려진 주소를 사용하는 대신 이미 알려진 주소 사이를 보간하고 이러한 주소를 바탕으로 새 주소의 위치를 추정합니다. 이 경우 230은 220과 240의 중간 지점(및 동일한 도로 쪽에 있음)에 있다고 가정하고 이를 기준으로 대략적인 위치를 반환할 수 있습니다.

> **Note**
>
> 데이터 공급자는 주기적으로 새 주소로 지리적 위치 데이터를 업데이트합니다. 이 경우 데이터 공급자 데이터에 `230 S 9th St`이(가) 추가되지만, 일반적으로 새 주소가 생성되었더라도 아직 데이터에 추가되지 않은 기간이 있습니다.

이 경우 데이터 공급자는 새 주소가 아직 데이터에 없기 때문에 새 주소가 세상에 존재하는지 여부를 알 수 없지만 보유한 정보를 바탕으로 가능한 최선의 답을 제공합니다. 이 결과를 보간이라고 하며, 데이터 공급자가 결과에서 반환할 수 있습니다. `interpolated`이(가) `false`을(를) 반환하는 경우 해당 주소는 알려진 주소입니다. `true`을(를) 반환하는 경우, 이는 대략적인 주소입니다. 반환되지 않은 경우, 데이터 공급자가 결과가 보간을 통해 나온 것인지 여부에 대한 정보를 제공하지 않은 것입니다.

> **Important**
>
> 데이터 공급자는 아예 존재하지 않는 주소에 대해서도 보간된 결과를 반환할 수 있습니다. 예를 들어 이 경우 **`232 S 9th St`** 입력 시, 공급자는 존재하지 않는 이 주소를 찾아 230에 가깝지만 240 쪽에 있는 위치를 반환합니다. 보간된 주소는 올바른 위치로 이동하는 데 유용하지만 알려진 주소가 아니라는 점을 염두에 두는 것이 좋습니다.

## 지오코드 결과 저장

장소 색인 리소스를 생성할 때는 데이터 저장소 옵션 (`IntendedUse`에서 호출API) 을 지정해야 합니다. 이는 일회용 또는 저장된 결과로 설정할 수 있습니다. 이것은 결과의 의도된 용도에 대해 묻는 질문입니다. 결과를 저장하려는 경우(캐싱 목적이라도) 일회용 옵션이 아닌 저장 옵션을 선택해야 합니다.

> **Note**
>
> 저장 옵션 (예, 레이블이 지정됨, 콘솔에 결과가 저장되거나 `storage` 에서 선택 `CreatePlaceIndex`API) 을 선택한 경우 Amazon Location Service는 결과를 저장하지 않습니다. 이는 결과를 저장할 계획임을 나타냅니다.

Amazon Location Service에 대한 쿼리 결과를 어떻게 사용할지 검토할 때는 적용되는 AWS 서비스 약관을 항상 숙지해야 합니다.

## 장소 용어

장소 색인 리소스

검색 쿼리를 지원하는 데이터 소스를 선택할 수 있습니다. 예를 들어 관심 장소, 주소 또는 좌표를 검색할 수 있습니다. 검색 쿼리가 장소 색인 리소스로 전송되면 해당 리소스에 구성된 데이터 소스를 사용하여 검색어가 처리됩니다.

지오코딩

지오코딩은 텍스트 입력을 받아 장소 색인에서 검색하고 위치와 함께 결과를 반환하는 프로세스입니다.

역방향 지오코딩

역지오코딩은 위치를 선택하고 장소 색인 내에서 해당 위치에 대한 정보(예: 주소, 도시, 회사 등)를 반환하는 프로세스입니다.

관련성

관련성은 결과가 입력과 얼마나 일치하는지를 나타냅니다. 이는 정확성의 척도는 아닙니다.

보간

보간은 알려진 주소 위치를 기준점으로 사용하여 알려지지 않은 주소를 찾는 과정입니다.

ISO3166개 국가 코드

Amazon Location Service Places는 국제 표준화 기구 (ISO) 3166 국가 코드를 사용하여 국가 또는 지역을 나타냅니다.

특정 국가 또는 지역의 코드를 찾으려면 온라인 검색 플랫폼을 사용하십시오. ISO

# 경로

이 섹션에서는 Amazon Location Service를 사용한 라우팅과 관련된 개념에 대한 개요를 제공합니다.

> **Note**
>
> 실제로 경로 리소스를 사용하는 방법에 대한 자세한 내용은 Amazon Location Service를 사용하여 경로 계산 섹션을 참조하세요.

# 경로 계산기 리소스

경로 계산기 리소스를 사용하면 선택한 데이터 제공자가 제공하는 up-to-date 도로망과 실시간 교통 정보를 기반으로 경로를 찾고 이동 시간을 추정할 수 있습니다.

경로를 APIs 사용하여 애플리케이션에서 두 위치 간 경로의 이동 시간, 거리 및 지오메트리를 요청할 수 있는 기능을 구축할 수 있습니다. 경로를 사용하여 매트릭스를 API 계산하기 위한 단일 요청으로 출발지와 목적지 사이의 이동 시간과 거리를 요청할 수도 있습니다.

다음은 경로 계산기 리소스를 생성하고 사용하는 방법을 보여 줍니다.


1. 먼저 데이터 공급자를 선택하여 AWS 계정에서 경로 계산기 리소스를 생성합니다.
2. 그런 다음 개발 환경 및 애플리케이션에 SDK 맞는 것을 선택하여 설치할 수 있습니다.
3. Amazon 위치 경로 사용을 시작하십시오APIs. 라우팅 APIs 사용 방법에 대한 자세한 내용은 의 주제를 참조하십시오Amazon Location Service를 사용하여 경로 계산.
4. 그런 다음 Amazon CloudWatch 및 같은 서비스를 사용하여 모니터링을 통합할 수 AWS CloudTrail 있습니다. 자세한 내용은 아마존을 통한 아마존 로케이션 서비스 모니터링 CloudWatch 및 을 참조하십시오AWS CloudTrail을 사용하여 로깅 및 모니터링.

## 경로 계산

Amazon Location 경로 계산기 리소스는 두 지리적 위치(출발지와 목적지) 간에 경로를 생성하는 데 사용할 수 있는 `CalculateRoute`(이)라는 작업을 제공합니다. 계산된 경로에는 맵에 경로를 그리기 위한 지오메트리와 경로의 전체 시간 및 거리가 포함됩니다.

### 웨이포인트 사용

경로 요청을 생성할 때 경로에 추가 웨이포인트를 추가할 수 있습니다. 이 지점은 출발지와 목적지 사이의 지점으로 경로를 따라 경유지 역할을 합니다. 경로는 지정된 각 웨이포인트를 통해 계산됩니다. 요청의 한 지점에서 다음 지점까지의 경로를 `Leg`(이)라고 합니다. 각 구간에는 경로의 해당 부분에 대한 거리, 시간 및 지오메트리가 포함됩니다.

> **Note**
>
> 웨이포인트는 요청에 명시된 순서대로 라우팅됩니다. 이는 최단 경로로는 재정렬되지 않습니다. 최단 경로 찾기에 대한 자세한 내용은 경로 계획 섹션을 참조하세요.

경로 계산을 위한 단일 요청에 최대 25개의 웨이포인트를 포함할 수 있습니다.

트래픽 및 출발 시간

Amazon Location Service는 경로를 계산할 때 트래픽을 고려합니다. 고려되는 트래픽은 지정한 시간을 기준으로 합니다. 지금 출발하도록 지정하거나 출발하고자 하는 특정 시간을 제공할 수 있습니다. 이렇게 하면 지정된 시간에 트래픽을 조정하여 경로 결과에 영향을 줍니다.

> **Note**
>
> 예를 들어, 출발 시간과 경로 응답 시간을 사용하여 도착 시간을 계산해 운전자의 도착 시간을 추정할 수 있습니다.

Amazon Location에서 트래픽을 고려하지 않으려면 출발 시간을 지정하지 말고 지금 출발도 지정하지 마세요. 그러면 해당 경로에 가장 적합한 교통 조건을 가정하는 경로가 계산됩니다.

이동 모드 옵션

Amazon Location Service를 사용하여 경로를 계산할 때 이동 모드를 설정할 수 있습니다. 기본 이동 모드는 자동차이지만 트럭 또는 도보 모드를 번갈아 선택할 수 있습니다.

자동차 또는 트럭 모드를 지정하는 경우 특정 추가 옵션도 지정할 수 있습니다.

자동차 모드의 경우 유료 도로나 페리를 피하도록 지정할 수 있습니다. 이렇게 하면 페리와 유료 도로를 피하려고 하지만, 그것이 목적지로 가는 유일한 방법인 경우에는 계속 그 경로를 따라 이동하게 됩니다.

트럭 모드의 경우에도 페리와 유료 도로를 피할 수 있지만, 트럭을 수용하지 않는 경로를 피하기 위해 트럭의 크기와 무게를 지정할 수도 있습니다.

# 경로 계획

Amazon Location Service를 사용하여 경로 계획 및 최적화 소프트웨어에 대한 입력을 생성할 수 있습니다. 출발 위치 세트와 도착 위치 세트 사이의 경로에 대해 이동 시간 및 이동 거리를 포함한 경로 결과를 생성할 수 있습니다. 이를 경로 매트릭스 생성이라고 합니다.

> **Note**
>
> 경로 계획 및 최적화 소프트웨어로 해결할 수 있는 다양한 시나리오가 있습니다. 예를 들어, 계획 소프트웨어는 지점 간 시간 및 거리 집합을 사용하여 각 지점에서 정지하는 최단 경로를 계산하여 단일 운전자에게 효율적인 경로를 제공할 수 있습니다. 또는 계획 소프트웨어를 사용하여 여러 트럭 간에 정류장을 나누어 전체 차량의 효율성을 높이거나 각 고객이 필요한 시간 내에 방문하도록 할 수 있습니다. Amazon Location은 계획 소프트웨어가 작업을 완료할 수 있도록 효율적인 방식으로 라우팅 기능을 제공합니다.

예를 들어 출발 위치 A와 B, 도착 위치 X와 Y가 주어지면 Amazon Location Service는 A에서 X, A에서 Y, B에서 X, B에서 Y까지, B에서 Y까지의 경로에 대한 이동 시간과 이동 거리를 반환합니다.

단일 경로를 계산할 때와 마찬가지로 다양한 교통 수단, 회피 및 교통 상황을 고려하여 경로를 계산할 수 있습니다. 예를 들어, 차량이 10.7m(35피트) 길이의 트럭이라고 지정하면 계산된 경로에서는 이러한 제한을 사용하여 이동 시간과 이동 거리를 결정합니다. 경로 매트릭스 계산에는 웨이포인트를 포함할 수 없습니다.

반환되는 결과 수와 계산된 경로 수는 출발 위치 수에 목적지 위치 수를 곱한 값입니다. 서비스에 대한 각 요청이 아니라 계산된 각 경로에 대해 요금이 부과되므로, 출발지가 10개이고 도착지가 10개인 경로 매트릭스는 100개의 경로로 청구됩니다.

## 경로 용어

경로 계산기 리소스

선택한 데이터 공급자로부터 가져온 교통 및 도로망 데이터를 사용하여 이동 시간, 거리를 추정하고 지도에 경로를 그릴 수 있는 AWS 리소스입니다.

경로 계산기 리소스를 사용하여 다양한 교통 수단, 우회 도로, 교통 상황에 따른 경로를 계산할 수 있습니다.

경로

경로에는 출발 위치, 웨이포인트 위치 및 목적지 위치에서 경로를 따라 이동할 때 사용되는 세부 정보가 포함됩니다.

경로 세부 정보의 예는 다음과 같습니다.

- 한 위치에서 다른 위치까지의 거리.
- 한 위치에서 다음 위치로 이동하는 데 걸리는 시간.
- 경로의 경로를 나타내는 LineString 지오메트리.

경로에 대한 자세한 내용은 Amazon Location Service Routes API 참조에서 CalculateRoute 작업에 대한 응답 구문을 참조하십시오.

경로 매트릭스

출발 위치 집합부터 목적지 위치 집합까지의 경로 목록입니다. 경로 계획 또는 최적화 소프트웨어에 입력하는 데 유용합니다.

경로 매트릭스 계산에 대한 자세한 내용은 Amazon Location Service Routes API 참조의 CalculateRouteMatrix 작업 구문을 참조하십시오.

**`LineString`** 지오메트리

Amazon Location 경로는 하나 이상의 구간(전체 경로 내에서 한 웨이포인트에서 다른 웨이포인트까지의 경로)으로 구성됩니다. 각 구간의 지오메트리는 `LineString`(으)로 표시되는 폴리라인입니다. `LineString`은(는) 순서가 지정된 위치 배열로, 맵에 경로를 표시하는 데 사용할 수 있습니다.

다음은 점 3개가 있는 `LineString`의 예시입니다.

```
[
    [-122.7565,49.0021],
    [-122.3394,47.6159],
    [-122.1082,45.8371]
]
```

웨이포인트

웨이포인트는 출발 위치와 목적지 위치 사이의 경로를 따라 정류장 역할을 하는 중간 위치입니다. 경로의 스탑오버 순서는 요청에 웨이포인트 위치를 제공하는 순서를 따릅니다.

구간

한 구간이란 한 위치에서 다른 위치로의 이동을 말합니다. 위치가 도로에 있지 않으면 가장 가까운 도로로 이동합니다. 경로의 구간 수는 총 위치 수보다 하나 적습니다.

웨이포인트가 없는 경로는 출발 위치에서 목적지까지의 단일 구간으로 구성됩니다. 웨이포인트 1개로 구성된 경로는 출발 지점에서 웨이포인트까지, 그리고 웨이포인트에서 목적지까지 총 2개의 구간으로 구성됩니다.

단계

단계는 구간의 하위 섹션입니다. 각 단계는 해당 구간의 해당 단계에 대한 요약 정보를 제공합니다.

# 지오펜스 및 트래커

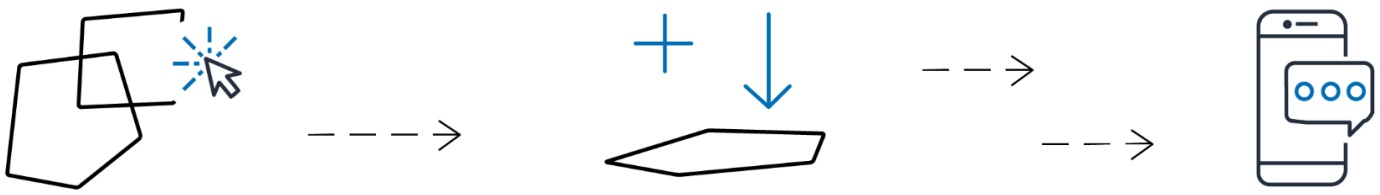
이 섹션에서는 Amazon Location Service 지오펜스 및 트래커를 사용한 작업에 대한 개요를 제공합니다. 지오펜스는 디바이스나 위치가 영역 안팎으로 이동할 때 알림을 받는 데 사용할 수 있는 다각형 경계입니다. 트래커 리소스는 디바이스가 이동할 때 위치를 저장하고 업데이트하는 데 사용됩니다.

> **Note**
>
> 지오펜스 및 트래커를 실제로 사용하는 방법에 대한 자세한 내용은 Amazon Location을 사용하여 관심 영역을 지오펜싱하기 섹션을 참조하세요.

## 지오펜스

지오펜스 수집 리소스를 사용하면 맵에 지오펜스(가상 경계)를 저장하고 관리할 수 있습니다. 지오펜스 수집 리소스를 기준으로 위치를 평가하고 위치 업데이트가 지오펜스 컬렉션의 지오펜스 경계를 넘을 때 알림을 받을 수 있습니다.

다음은 지오펜스 컬렉션 리소스를 생성하고 사용하는 방법을 보여 줍니다.



1. 계정에서 지오펜스 컬렉션 리소스를 생성하십시오. AWS

2. 해당 컬렉션에 지오펜스를 추가합니다. Amazon 로케이션 콘솔의 지오펜스 업로드 도구를 사용하거나 Amazon 로케이션 지오펜스를 사용하여 업로드할 수 있습니다. API 사용 가능한 옵션에 대한 자세한 내용은 Amazon Location 액세스를 참조하세요.

   지오펜스는 다각형이나 원으로 정의할 수 있습니다. 다각형을 사용하여 디바이스가 특정 영역에 진입하는 시점을 찾을 수 있습니다. 원을 사용하여 디바이스가 어떤 지점으로부터 특정 거리(반경) 내에 있는 시점을 찾을 수 있습니다.

3. 모든 지오펜스를 대상으로 위치 평가를 시작할 수 있습니다. 위치 업데이트가 하나 이상의 지오펜스의 경계를 넘어서는 경우, 지오펜스 수집 리소스는 Amazon에서 다음 지오펜스 이벤트 유형 중 하나를 내보냅니다. EventBridge

   - ENTER— 위치 업데이트가 해당 경계를 입력하여 해당 경계를 넘어서는 각 지오펜스에 대해 하나의 이벤트가 생성됩니다.
   - EXIT— 위치 업데이트가 지오펜스를 종료하여 경계를 넘어가는 각 지오펜스에 대해 하나의 이벤트가 생성됩니다.

   자세한 내용은 the section called “다음과 같은 이벤트에 대응하기 EventBridge” 단원을 참조하십시오. Amazon CloudWatch 및 같은 서비스를 사용하여 모니터링을 통합할 수도 AWS CloudTrail있습니다. 자세한 내용은 the section called “를 통한 모니터링 CloudWatch” 및 을 참조하십시오the section called “아마존 CloudTrail 로케이션과 함께 사용”.

예를 들어, 여러 트럭을 추적하고 있으며 한 트럭이 창고의 특정 구역 내에 들어오면 알림을 받고 싶은 경우를 들 수 있습니다. 각 창고 주변 영역에 지오펜스를 만들 수 있습니다. 그런 다음 트럭에서 업데이트된 위치를 보내면 Amazon Location Service를 사용하여 해당 위치를 평가하고 트럭이 지오펜스 영역 중 하나에 들어갔는지 (또는 빠져나왔는지) 확인할 수 있습니다.

> **Note**
>
> 평가 대상 지오펜스 컬렉션의 수를 기준으로 요금이 청구됩니다. 청구서는 각 컬렉션의 지오펜스 수에 영향을 받지 않습니다. 각 지오펜스 컬렉션에는 최대 50,000개의 지오펜스가 포함될 수 있으므로 가능한 경우 지오펜스를 더 적은 컬렉션으로 결합하여 지오펜스 평가 비용을 절감할 수 있습니다. 생성된 이벤트에는 컬렉션의 개별 지오펜스 ID와 컬렉션 ID가 포함됩니다.

## 지오펜스 이벤트

모니터링 중인 위치의 장소는 `DeviceId`라는 ID로 참조됩니다(그리고 해당 위치를 디바이스 위치라고 함). 평가할 디바이스 위치 목록을 지오펜스 컬렉션 리소스에 직접 보내거나 트래커를 사용할 수 있습니다. 트래커에 대한 자세한 내용은 다음 섹션을 참조하세요.

디바이스가 지오펜스에 들어오거나 EventBridge 나올 때만 이벤트 (Amazon을 통해) 를 수신하며, 모든 위치 변경에 대한 이벤트는 수신하지 않습니다. 즉, 일반적으로 이벤트를 수신하며 이벤트에 응답해야 하는 빈도는 매 디바이스 위치 업데이트 시에 비해 훨씬 적습니다.

> **Note**
>
> 특정 `DeviceID`에 대한 첫 번째 위치 평가에서는 해당 디바이스가 이전에는 지오펜스에 없었던 것으로 가정합니다. 따라서 첫 번째 업데이트는 컬렉션의 지오펜스 내부에 있으면 `ENTER` 이벤트를 생성하고 그렇지 않으면 이벤트를 생성하지 않습니다.

디바이스가 지오펜스에 진입했는지 아니면 종료했는지 계산하려면 Amazon Location Service에서 디바이스의 이전 위치 상태를 유지해야 합니다. 이 위치 상태는 30일 동안 저장됩니다. 디바이스가 업데이트되지 않은 지 30일이 지나면 새로운 위치 업데이트가 첫 번째 위치 업데이트로 간주됩니다.

## 트래커

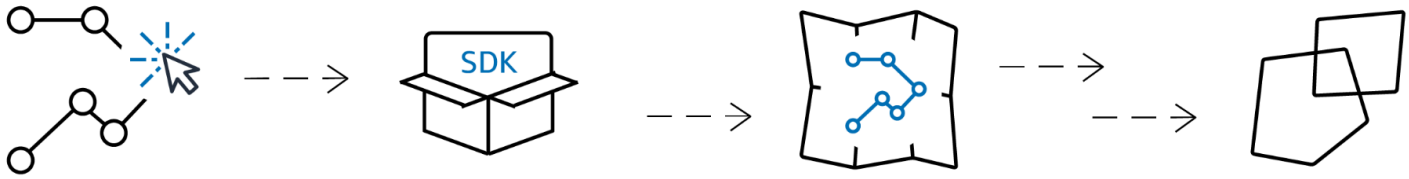
트래커는 디바이스 컬렉션에 대한 위치 업데이트를 저장합니다. 트래커를 사용하여 디바이스의 현재 위치 또는 위치 기록을 쿼리할 수 있습니다. 이는 업데이트를 저장하지만, 저장하기 전에 위치를 필터링하여 저장 공간과 시각적 노이즈를 줄입니다.

트래커 리소스에 저장된 각 위치 업데이트에는 위치 정확도 측정값과 저장하려는 위치 또는 디바이스에 대한 메타데이터 필드가 최대 3개까지 포함될 수 있습니다. 메타데이터는 키-값 쌍으로 저장되며 속도, 방향, 타이어 공기압 또는 엔진 온도와 같은 정보를 저장할 수 있습니다.

> **Note**
>
> 트래커 스토리지는 AWS 소유한 키로 자동으로 암호화됩니다. 관리하는 KMS 키를 사용하여 다른 암호화 계층을 추가하여 본인만 데이터에 액세스할 수 있도록 할 수 있습니다. 자세한 내용은 Amazon Location Service의 저장 데이터 암호화 단원을 참조하십시오.

트래커 위치 필터링 및 저장은 그 자체로도 유용하지만, 트래커는 지오펜스와 함께 사용할 때 특히 유용합니다. 트래커를 하나 이상의 지오펜스 컬렉션 리소스에 연결할 수 있으며 위치 업데이트는 해당 컬렉션의 지오펜스를 기준으로 자동으로 평가됩니다. 필터링을 적절히 사용하면 지오펜스 평가 비용도 크게 줄일 수 있습니다.

다음 다이어그램은 트래커 리소스를 생성하고 사용하는 방법을 보여줍니다.




1. 먼저 AWS 계정에 트래커 리소스를 생성합니다.
2. 다음으로 위치 업데이트를 트래커 리소스에 전송하는 방법을 결정합니다. 추적 기능을 모바일 애플리케이션에 통합하는 AWS SDKs데 사용합니다. 또는 를 사용하여 MQTT 추적의 step-by-step 지침을 MQTT 따라 사용할 수도 있습니다.
3. 이제 트래커 리소스를 사용하여 위치 기록을 기록하고 맵에 시각화할 수 있습니다.
4. 또한 트래커 리소스를 하나 이상의 지오펜스 컬렉션에 연결하여 트래커 리소스로 전송되는 모든 위치 업데이트가 링크된 모든 지오펜스 컬렉션의 모든 지오펜스에 대해 자동으로 평가되도록 할 수 있습니다. Amazon 위치 콘솔의 추적기 리소스 세부 정보 페이지에서 또는 Amazon 위치 API 추적기를 사용하여 리소스를 연결할 수 있습니다.
5. 그런 다음 Amazon CloudWatch 및 같은 서비스를 사용하여 모니터링을 통합할 수 AWS CloudTrail 있습니다. 자세한 내용은 the section called “를 통한 모니터링 CloudWatch” 및 을 참조하십시오the section called “아마존 CloudTrail 로케이션과 함께 사용”.

## 지오펜스와 함께 트래커 사용

트래커는 지오펜스와 함께 사용할 경우 추가 기능을 제공합니다. Amazon Location 콘솔 또는 를 통해 추적기를 지오펜스 컬렉션과 연결하여 추적기 위치를 자동으로 평가합니다. API 트래커가 업데이트된 위치를 수신할 때마다 컬렉션의 각 지오펜스와 비교하여 해당 위치가 평가되고 Amazon에서 적절한 `ENTER` `EXIT` AND 이벤트가 생성됩니다. EventBridge 또한 트래커에 필터링을 적용할 수 있으며 필터링에 따라 의미 있는 위치 업데이트만 평가하여 지오펜스 평가 비용을 줄일 수 있습니다.

일부 위치 업데이트를 이미 수신한 후 트래커를 지오펜스 컬렉션과 연결하면 연결 후 첫 번째 위치 업데이트가 지오펜스 평가의 초기 업데이트로 간주됩니다. 지오펜스 내에 있는 경우 `ENTER` 이벤트를 받게 됩니다. 지오펜스 내에 있지 않으면 이전 상태와 상관없이 `EXIT` 이벤트를 수신할 수 없습니다.

# 위치 필터링

트래커는 전송되는 위치를 자동으로 필터링할 수 있습니다. 일부 디바이스 위치 업데이트를 필터링하려는 데에는 몇 가지 이유가 있습니다. 매분마다 보고서를 보내는 시스템의 경우 디바이스를 시간별로 필터링하여 30초마다 위치를 저장하고 평가할 수 있습니다. 더 자주 모니터링하더라도 위치 업데이트를 필터링하여 하드웨어의 소음을 제거하는 것이 좋습니다. GPS GPS포지션 위치는 본질적으로 시끄럽습니다. 정확도가 100% 완벽하지는 않기 때문에 정지해 있는 디바이스라도 약간씩 움직이는 것처럼 보입니다. 낮은 속도에서는 이러한 지터가 시각적 혼란을 야기하고 디바이스가 지오펜스의 엣지 근처에 있으면 잘못된 진입 및 종료 이벤트가 발생할 수 있습니다.

위치 필터링은 트래커가 위치 업데이트를 수신할 때 작동하여 디바이스 경로의 시각적 노이즈(지터)를 줄이고, 잘못된 지오펜스 진입 및 종료 이벤트 수를 줄이며, 저장된 위치 업데이트 수와 트리거되는 지오펜스 평가 수를 줄여 비용을 관리하는 데 도움이 됩니다.

트래커는 비용을 관리하고 위치 업데이트의 지터를 줄이는 데 도움이 되는 세 가지 위치 필터링 옵션을 제공합니다.

- 정확도 기반 – 정확도 측정을 제공하는 모든 디바이스와 함께 사용할 수 있습니다. 대부분의 모바일 GPS 기기는 이 정보를 제공합니다. 각 위치 측정의 정확도는 GPS 위성 수신, 풍경, Wi-Fi 및 Bluetooth 장치의 근접성을 비롯한 여러 환경 요인의 영향을 받습니다. 대다수 모바일 디바이스를 포함한 디바이스는 대부분 측정과 함께 측정의 정확도 추정치를 제공할 수 있습니다. `AccuracyBased` 필터링을 사용하면 Amazon Location는 디바이스가 측정된 정확도보다 적게 움직인 경우 위치 업데이트를 무시합니다. 예를 들어 디바이스에서 두 번 연속 업데이트의 정확도 범위가 5m와 10m인 경우, 디바이스가 15m 미만으로 이동하면 Amazon Location은 두 번째 업데이트를 무시합니다. Amazon Location은 무시한 업데이트를 지오펜스와 비교하여 평가하거나 저장하지 않습니다.

  정확도가 제공되지 않으면 0으로 처리되어 측정값이 완전히 정확한 것으로 간주되며 업데이트에 필터링이 적용되지 않습니다.

  > **Note**
  >
  > 정확도 기반 필터링을 사용하여 모든 필터링을 제거할 수 있습니다. 정확도 기반 필터링을 선택했지만 모든 정확도 데이터를 0으로 재정의하거나 정확도를 완전히 생략하는 경우 Amazon Location은 업데이트를 필터링하지 않습니다.

  대부분의 시나리오에서 정확도 기반 필터링은 위치 업데이트를 필터링하는 데 적합하며, 불필요한 업데이트를 필터링하는 동시에 위치 추적의 균형을 제공하여 비용을 절감합니다.

- 거리 기반 – 디바이스가 정확도 측정을 제공하지 않지만 필터링을 활용하여 지터를 줄이고 비용을 관리하려는 경우에 사용합니다. `DistanceBased` 필터링은 디바이스가 30m (98.4피트) 미만으로 이동한 위치 업데이트를 무시합니다. `DistanceBased` 위치 필터링을 사용하는 경우 Amazon Location은 지오펜스에 대해 무시된 업데이트를 평가하거나 업데이트를 저장하지 않습니다.

  iOS 및 Android 디바이스의 평균 정확도를 포함하여 대부분의 모바일 디바이스의 정확도는 15m 이내입니다. 대부분의 애플리케이션에서 `DistanceBased` 필터링은 디바이스 궤적을 맵에 표시할 때 위치 부정확성의 영향을 줄이고, 디바이스가 지오펜스 경계 근처에 있을 때 여러 번의 연속적인 출입 이벤트로 인한 바운싱 효과를 줄일 수 있습니다. 또한 링크된 지오펜스를 기준으로 평가하거나 디바이스 위치를 검색하기 위한 호출을 줄여 애플리케이션 비용을 절감할 수 있습니다.

  거리 기반 필터링은 필터링하고 싶지만 디바이스에서 정확도 측정값을 제공하지 않는 경우 또는 정확도 기반 필터링보다 많은 수의 업데이트를 필터링하려는 경우에 유용합니다.

- 시간 기반 - (기본값) 디바이스가 위치 업데이트를 매우 자주(30초에 한 번 이상) 보내고 모든 업데이트를 저장하지 않고도 거의 실시간으로 지오펜스를 평가하려는 경우에 사용합니다. `TimeBased` 필터링에서는 모든 위치 업데이트가 연결된 지오펜스 컬렉션에 대해 평가되지만 모든 위치 업데이트가 저장되는 것은 아닙니다. 업데이트 빈도가 30초 이상인 경우 각 고유 디바이스 ID에 대해 30초당 하나의 업데이트만 저장됩니다.

  시간 기반 필터링은 더 적은 수의 위치를 저장하고 싶지만 모든 위치 업데이트를 관련 지오펜스 컬렉션과 비교하여 평가하려는 경우에 특히 유용합니다.

> **Note**
>
> 필터링 방법과 위치 업데이트 빈도를 결정할 때는 추적 애플리케이션 비용을 염두에 두세요. 모든 위치 업데이트에 대해 요금이 청구되며 연결된 각 지오펜스 컬렉션에 대한 위치 업데이트 평가 비용은 한 번 청구됩니다. 예를 들어, 시간 기반 필터링을 사용할 때 트래커가 두 개의 지오펜스 컬렉션에 연결된 경우 모든 위치 업데이트는 위치 업데이트 요청 1회와 지오펜스 컬렉션 평가 2회로 계산됩니다. 디바이스의 위치 업데이트를 5초마다 보고하고 시간 기반 필터링을 사용하는 경우 각 디바이스에 대해 시간당 720건의 위치 업데이트와 1,440회의 지오펜스 평가에 대한 요금이 청구됩니다.

# 지오펜스 용어

지오펜스 컬렉션

0개 이상의 지오펜스를 포함합니다. 요청 시 진입 및 종료 이벤트를 생성하여 지오펜스를 기준으로 디바이스 위치를 평가함으로써 지오펜스를 모니터링할 수 있습니다.

지오펜스

맵의 가상 경계를 정의하는 다각형 또는 원형 지오메트리입니다.

다각형 지오메트리

Amazon Location 지오펜스는 지리적 영역의 가상 경계이며 다각형 지오메트리 또는 원으로 표시됩니다.

원은 둘레에 거리가 있는 지점입니다. 디바이스가 특정 위치로부터 일정 거리 내에 있는지 알림을 받으려면 원을 사용하세요.

다각형은 하나 이상의 선형 고리로 구성된 배열입니다. 디바이스 알림에 대한 특정 경계를 정의하려는 경우 다각형을 사용하세요. 선형 고리는 네 개 이상의 꼭지점으로 구성된 배열로, 첫 번째 꼭지점과 마지막 꼭지점이 동일하여 닫힌 경계를 형성합니다. 각 꼭지점은 다음과 같은 형태의 2차원 점입니다.*[longitude, latitude]*, 여기서 경도와 위도의 단위는 도입니다. 꼭지점은 다각형을 중심으로 시계 반대 방향 순서로 나열되어야 합니다.

> **Note**
>
> Amazon Location Service는 고리가 두 개 이상 있는 다각형을 지원하지 않습니다. 여기에는 구멍, 섬 또는 다중 다각형이 포함됩니다. 또한 Amazon Location은 시계 방향으로 감겨 있거나 반이중선을 가로지르는 다각형을 지원하지 않습니다.

다음은 단일 선형 외부 고리의 예입니다.

```
[
  [
    [-5.716667, -15.933333],
    [-14.416667, -7.933333],
    [-12.316667, -37.066667],
```

```
    [-5.716667, -15.933333]
  ]
]
```

## 트래커 용어

트래커 리소스

기기로부터 위치 업데이트를 받는 AWS 리소스입니다. 트래커 리소스는 현재 및 과거 디바이스 위치와 같은 위치 쿼리를 지원합니다. 트래커 리소스를 지오펜스 컬렉션에 연결하면 링크된 지오펜스 컬렉션의 모든 지오펜스에 대한 위치 업데이트가 자동으로 평가됩니다.

위치 데이터 추적

트래커 리소스는 시간 경과에 따른 디바이스 정보를 저장합니다. 이 정보에는 일련의 위치 업데이트가 포함되며, 각 업데이트에는 위치, 시간 및 선택적 메타데이터가 포함됩니다. 메타데이터에는 위치의 정확도는 물론 추적 중인 차량의 속도, 방향, 타이어 공기압, 잔여 연료 또는 엔진 온도와 같은 각 위치에 대한 주요 정보를 추적하는 데 도움이 되는 최대 3개의 키-값 쌍이 포함될 수 있습니다. 트래커는 디바이스 위치 기록을 30일 동안 유지합니다.

위치 필터링

위치 필터링은 업데이트를 저장하거나 지오펜스에 대해 평가하기 전에 중요한 정보를 제공하지 않는 위치 업데이트를 필터링하여 비용을 제어하고 추적 애플리케이션의 품질을 개선하는 데 도움이 될 수 있습니다.

`AccuracyBased`, `DistanceBased`, 또는 `TimeBased` 필터링을 선택할 수 있습니다. 기본적으로 위치 필터링은 `TimeBased`(으)로 설정됩니다.

트래커 리소스를 생성하거나 업데이트할 때 위치 필터링을 구성할 수 있습니다.

RFC3339 타임스탬프 형식

Amazon Location Service Trackers는 날짜 및 시간에 대한 국제 표준화 기구 (ISO) 8601 형식을 따르는 RFC3339 형식을 사용합니다.

형식은 “-MM-:mm:SS.SSSZ+ 00:00”입니다. YYYY DDThh

- `YYYY-MM-DD` — 날짜 형식을 나타냅니다.
- `T` — 시간 값이 다음과 같음을 나타냅니다.
- `hh:mm:ss.sss` — 시간을 24시간 형식으로 나타냅니다.

- `Z`— 사용된 시간대가 임을 나타내며UTC, 시간대와의 편차가 뒤따를 수 있습니다. UTC
- `+00:00`— UTC 시간대와의 편차를 선택적으로 표시할 수 있습니다. 예를 들어, + 01:00 은 UTC + 1시간을 나타냅니다.

예

2020년 7월 2일 오후 12:15:20 에 시간대를 1시간 더 조정한 경우 UTC

```
2020-07-02T12:15:20.000Z+01:00
```

# Amazon Location Service 사용에 대한 일반적인 사용 사례

Amazon Location Service를 사용하면 자산 추적에서 위치 기반 마케팅에 이르기까지 다양한 애플리케이션을 구축할 수 있습니다. 일반적인 사용 사례는 다음과 같습니다.

사용자 참여 및 지오마케팅

위치 데이터를 사용하여 고객을 대상으로 마케팅에 대한 사용자 참여도를 향상시키는 솔루션을 구축합니다. 예를 들어 Amazon Location은 모바일 앱으로 커피를 주문한 고객이 근처에 있을 때 알림을 표시하는 이벤트를 트리거할 수 있습니다. 또한 소매업체가 대상 매장 근처에 있는 고객에게 할인 코드나 디지털 전단지를 보낼 수 있도록 지오타겟팅 기능을 구축할 수 있습니다.

자산 추적

기업이 제품, 직원, 인프라의 현재 및 과거 위치를 이해하는 데 도움이 되는 자산 추적 기능을 구축하세요. 자산 추적 기능을 사용하면 원격 인력 배치를 최적화하고 이동 중 배송을 보호하며 파견 효율성을 극대화하는 다양한 솔루션을 구축할 수 있습니다.

제공

위치 기능을 배송 애플리케이션에 통합하여 출발 위치, 배송 차량 및 목적지를 저장, 추적 및 조정합니다. 예를 들어 Amazon Location 기능이 내장된 음식 배달 애플리케이션에는 배달 기사가 근처에 있으면 레스토랑에 자동으로 알릴 수 있는 위치 추적 및 지오펜싱 기능이 있습니다. 이렇게 하면 대기 시간이 줄어들고 배달된 음식의 품질을 유지하는 데 도움이 됩니다.

이 항목에서는 Amazon Location으로 구축할 수 있는 애플리케이션의 아키텍처 및 단계에 대한 개요를 제공합니다.

항목

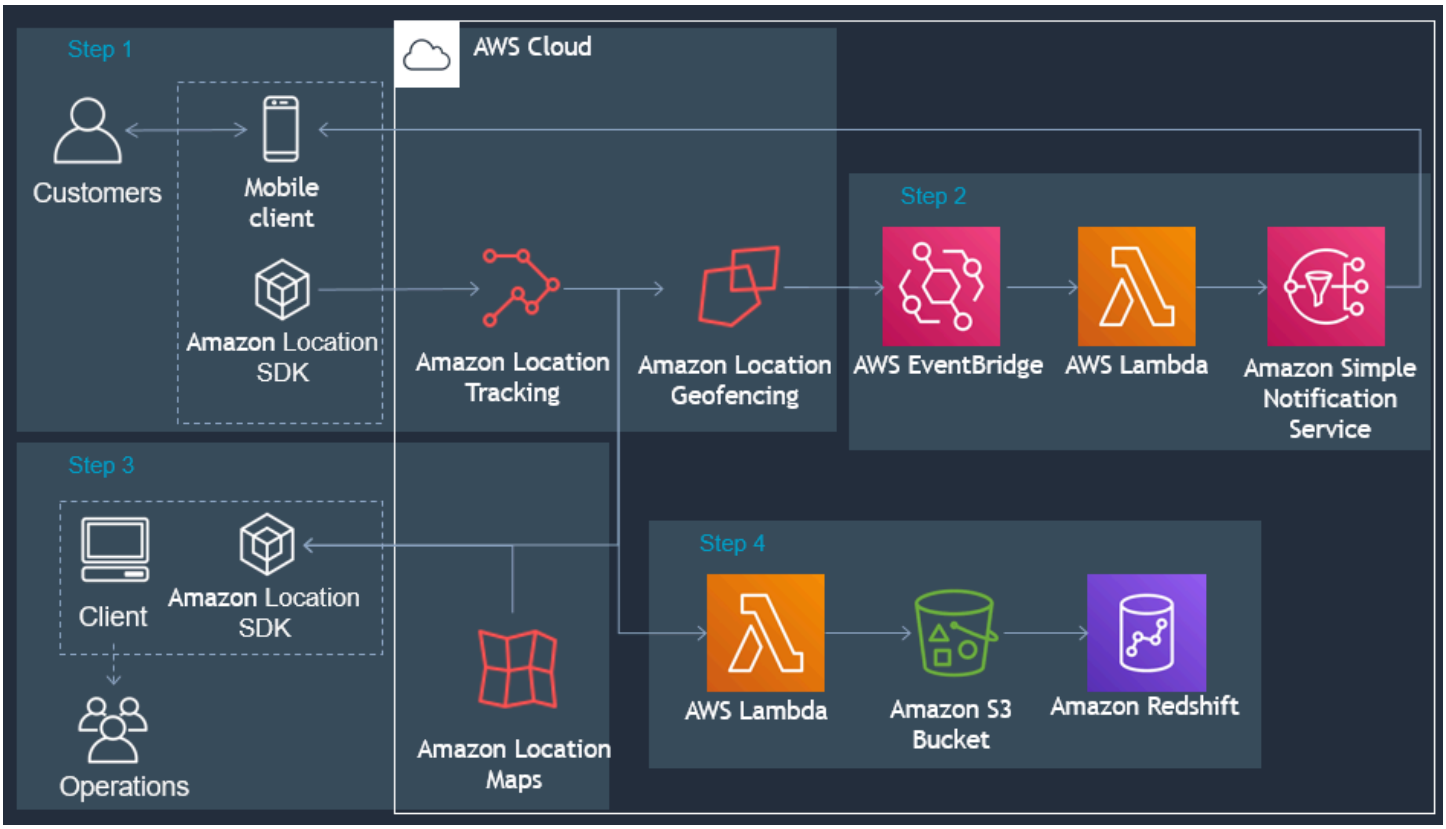


## 사용자 참여 및 지오마케팅 애플리케이션

다음은 Amazon Location을 사용한 사용자 참여 및 지오마케팅 애플리케이션 아키텍처를 보여주는 예입니다.

이 아키텍처로 다음을 수행할 수 있습니다.

- 대상의 근접성을 기반으로 이벤트를 시작하여 주변 고객에게 제안을 보내거나 최근에 시설을 떠난 고객의 참여를 유도할 수 있습니다(지오타겟팅이라고 함).
- 고객 디바이스 위치를 맵에 시각화하여 시간 경과에 따른 추세를 모니터링할 수 있습니다.
- 시간에 따라 분석할 수 있는 고객 디바이스 위치를 저장합니다.
- 위치 이력을 분석하여 최적화를 위한 추세와 기회를 파악합니다.



다음은 사용자 참여 및 지오마케팅 애플리케이션을 구축하는 데 필요한 단계에 대한 개요입니다.

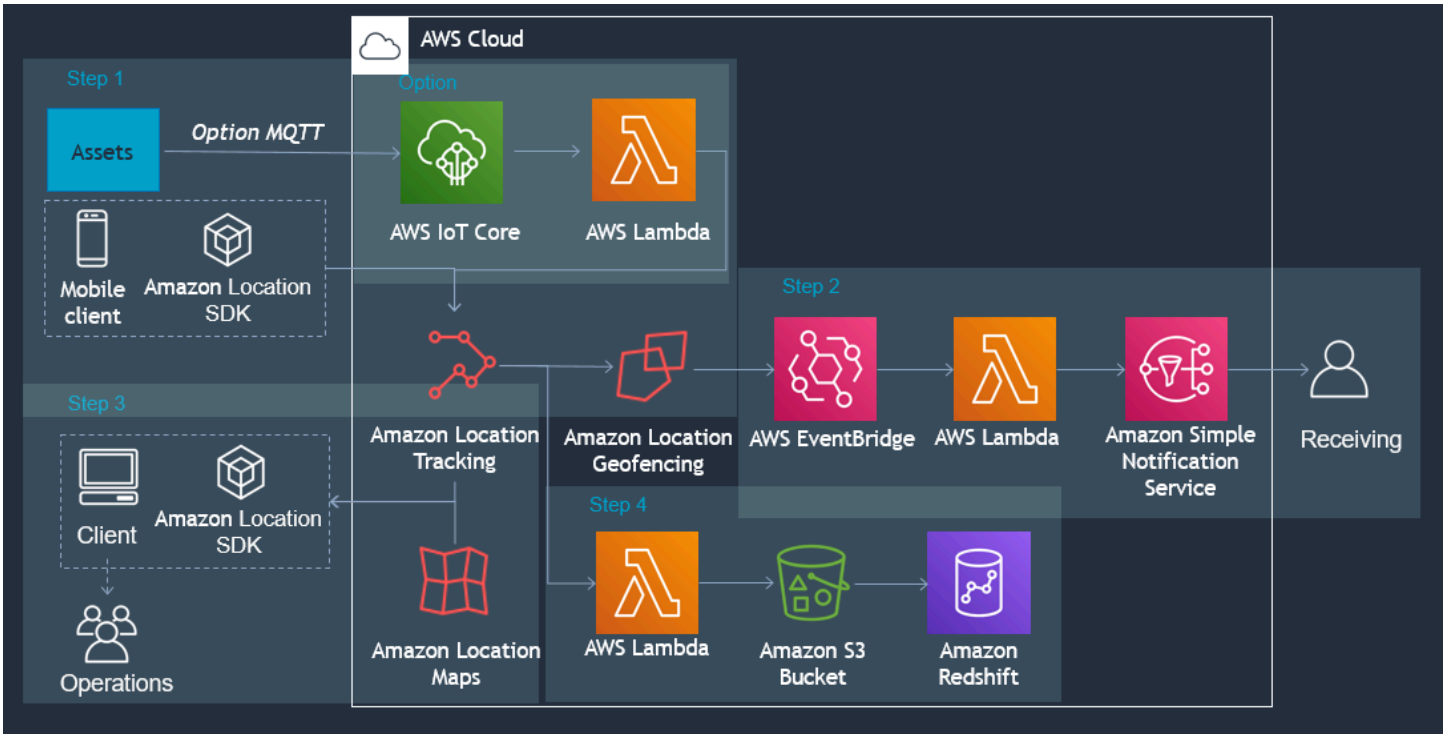
1. 지오펜스 컬렉션에서 지오펜스를 생성하고 트래커를 지오펜스에 연결하세요. 자세한 내용은 the section called “지오펜싱 및 추적” 단원을 참조하십시오.
2. 지오펜스 관심 영역에 들어오거나 나가는 고객에게 알림을 EventBridge 보내도록 Amazon을 구성합니다. 자세한 내용은 the section called “다음과 같은 이벤트에 대응하기 EventBridge” 단원을 참조하십시오.
3. 고객 위치와 지오펜스를 맵에 표시합니다. 자세한 내용은 맵 사용을 참조하세요.
4. 추가 분석을 위해 위치 데이터를 장기 저장소에 저장합니다.
5. 애플리케이션을 구축한 후에는 CloudWatch Amazon을 AWS CloudTrail 사용하여 애플리케이션을 관리할 수 있습니다. 자세한 내용은 the section called “를 통한 모니터링 CloudWatch” 및 the section called “아마존 CloudTrail 로케이션과 함께 사용” 단원을 참조하세요.

## 자산 추적 애플리케이션

다음은 Amazon Location을 사용하는 자산 추적 애플리케이션 아키텍처를 보여줍니다.

이 아키텍처로 다음을 수행할 수 있습니다.

- 맵에 자산 위치를 표시하여 큰 그림을 그릴 수 있습니다. 예를 들어, 운영 또는 계획 팀을 돕기 위해 과거 위치나 이벤트를 사용하여 히트 맵을 표시합니다.
- 자산 근접성을 기반으로 이벤트를 시작하여 수령 부서에 알림을 제공하여 발송물 도착을 준비하고 처리 시간을 단축할 수 있습니다.
- 백엔드 애플리케이션에서 작업을 시작하거나 시간에 따른 데이터 분석을 위해 자산 위치를 저장합니다.
- 위치 이력을 분석하여 최적화를 위한 추세와 기회를 파악합니다.




다음은 자산 추적 애플리케이션을 구축하는 데 필요한 단계에 대한 개요를 제공합니다.

1. 지오펜스 컬렉션에서 지오펜스를 생성하고 트래커를 지오펜스에 연결하세요. 자세한 내용은 the section called “지오펜싱 및 추적” 단원을 참조하십시오.
2. 알림을 보내거나 프로세스를 EventBridge 시작하도록 Amazon을 구성합니다. 자세한 내용은 the section called “다음과 같은 이벤트에 대응하기 EventBridge” 단원을 참조하십시오.
3. 추적된 자산과 활성 지오펜스를 맵에 표시합니다. 자세한 내용은 맵 사용을 참조하세요.
4. 추가 분석을 위해 위치 데이터를 장기 저장소에 저장합니다.
5. 애플리케이션을 구축한 후에는 CloudWatch Amazon을 AWS CloudTrail 사용하여 애플리케이션을 관리할 수 있습니다. 자세한 내용은 the section called “를 통한 모니터링 CloudWatch” 및 the section called “아마존 CloudTrail 로케이션과 함께 사용” 단원을 참조하세요.

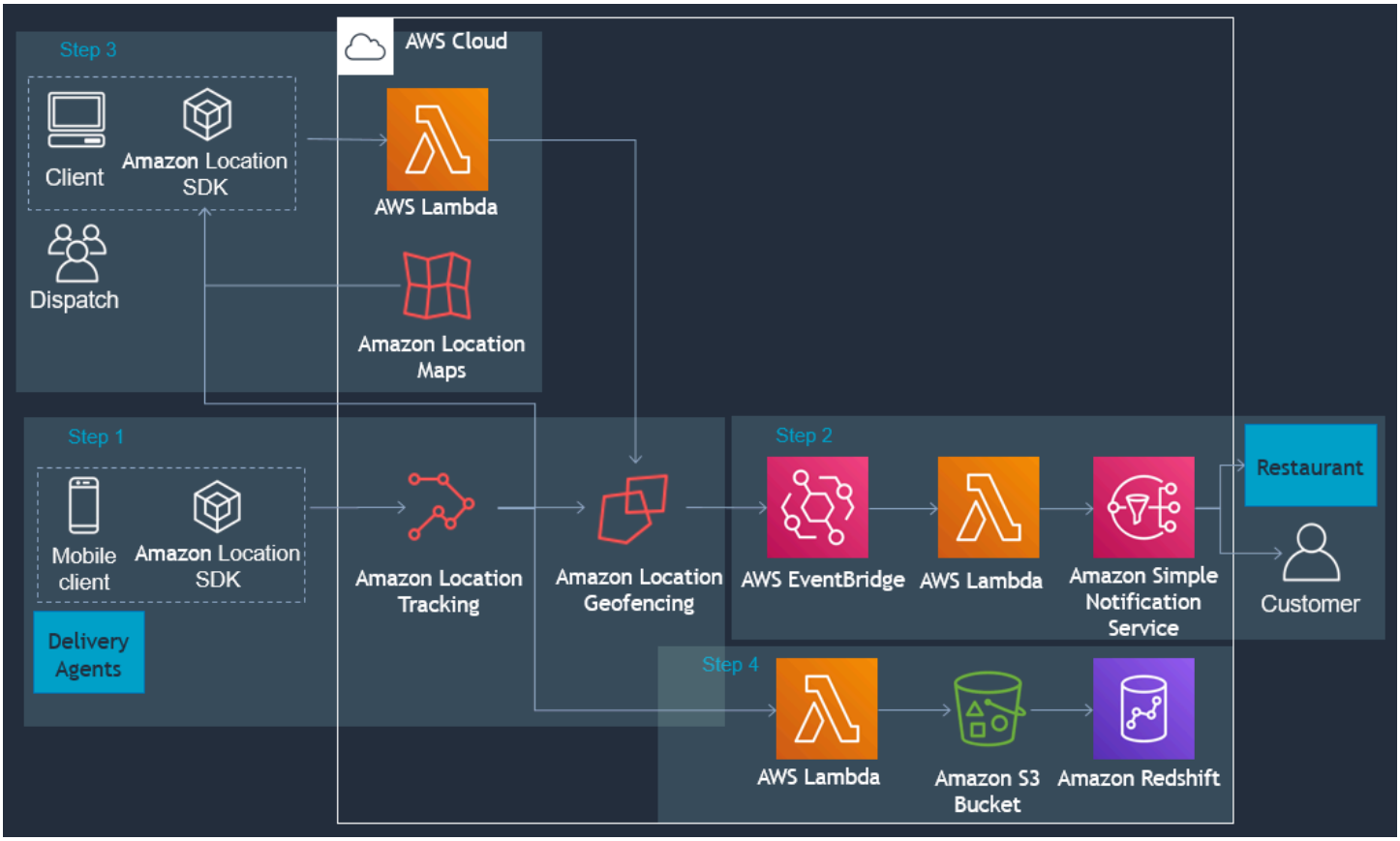
## 배송 애플리케이션

다음은 Amazon Location을 이용한 배송 애플리케이션 아키텍처의 예시입니다.

이 아키텍처로 다음을 수행할 수 있습니다.

- 배송원의 근접성을 기반으로 이벤트를 시작하여 픽업이 제시간에 준비되고 배송이 도착할 때 고객에게 알림을 보낼 수 있습니다.

- 운전자 위치와 픽업 및 하차 위치를 맵에 거의 실시간으로 표시하여 배송 팀에 큰 그림을 보여줍니다.
- 백엔드 애플리케이션에서 조치를 취하거나 시간이 지남에 따라 분석할 수 있도록 배송원의 위치를 저장합니다.
- 위치 이력을 분석하여 최적화를 위한 추세와 기회를 파악합니다.




다음은 배송 애플리케이션을 구축하는 데 필요한 단계에 대한 개요입니다.

1. 지오펜스 컬렉션을 생성하고 트래킹된 디바이스를 컬렉션에 연결하세요. 자세한 내용은 the section called “지오펜싱 및 추적” 단원을 참조하십시오.
2. 주문이 예약될 때 지오펜스를 자동으로 추가 및 제거하는 AWS Lambda 기능을 생성하십시오.
3. 알림을 보내거나 프로세스를 EventBridge 시작하도록 Amazon을 구성합니다. 자세한 내용은 the section called “다음과 같은 이벤트에 대응하기 EventBridge” 단원을 참조하십시오.
4. 추적된 자산과 활성 지오펜스를 맵에 표시합니다. 자세한 내용은 맵 사용을 참조하세요.
5. 추가 분석을 위해 위치 데이터를 장기 저장소에 저장합니다.

6. 애플리케이션을 구축한 후에는 CloudWatch Amazon을 AWS CloudTrail 사용하여 애플리케이션을 관리할 수 있습니다. 자세한 내용은 the section called “를 통한 모니터링 CloudWatch” 및 the section called “아마존 CloudTrail 로케이션과 함께 사용” 단원을 참조하세요.

# 데이터 공급자란 무엇입니까?

Amazon Location Service를 사용하면 타사 계약 또는 통합 없이 AWS 계정을 통해 여러 데이터 공급자의 지리적 위치 리소스에 액세스할 수 있습니다. 이를 통해 타사 계정, 보안 인증, 라이선스 및 청구를 관리할 필요 없이 애플리케이션 구축에만 집중할 수 있습니다.

다음 Amazon Location Service는 데이터 공급자를 사용합니다.

- 맵 – 맵 리소스를 생성할 때 다양한 맵 공급자의 스타일을 선택합니다. 맵 리소스를 사용하여 대화형 맵을 구축하여 데이터를 시각화할 수 있습니다.
- 장소 – 지오코딩, 역지오코딩, 검색을 위한 쿼리를 지원하는 장소 색인 리소스를 생성할 때 데이터 공급자를 선택합니다.
- 경로 – 경로 계산기 리소스를 생성할 때 다양한 지역 및 애플리케이션에서 경로 계산을 위한 쿼리를 지원하는 데이터 공급자를 선택합니다. Amazon Location Service는 선택한 데이터 공급자를 통해 up-to-date 도로망 데이터, 실시간 교통 데이터, 계획된 폐쇄 시간, 과거 교통 패턴을 기반으로 경로를 계산할 수 있도록 지원합니다.

각 공급자는 서로 다른 방법을 사용하여 데이터를 수집하고 관리합니다. 또한 그들은 전 세계 여러 리전에서 다양한 전문 지식을 보유하고 있을 수 있습니다. 이 섹션에서는 데이터 공급자에 대한 세부 정보를 제공합니다. 선호도에 따라 원하는 데이터 공급자를 선택할 수 있습니다.

Amazon Location Service 데이터 공급자를 사용할 때는 반드시 이용 약관을 읽어 보세요. 자세한 내용은 AWS서비스 약관을 참조하십시오. 또한 Amazon Location에서 개인 정보를 보호하는 방법에 대한 자세한 내용은 the section called “데이터 개인 정보 보호” 섹션을 참조하세요.

## 데이터 공급자의 적용 범위 및 기능

다음 표는 각 데이터 공급자의 적용 범위와 기능을 개괄적으로 보여줍니다.

| 데이터 공급자 | 지리적 범위 | 기능 적용 범위 | AWS 리전 |
| --- | --- | --- | --- |
| Esri | 전 세계 | 맵, 장소, 경로 | Amazon Location을 사용할 수 있는 모든 리전. |
| Grab | 동남아시아 | 맵, 장소, 경로 | 아시아 태평양(싱가포르), `ap-southeast-1` 만 해당. |
| HERE | 전 세계 | 맵, 장소, 경로 | Amazon Location을 사용할 수 있는 모든 리전. |
| Open Data(오픈 데이터) | 전 세계 | 맵 | Amazon Location을 사용할 수 있는 모든 리전. |

각 데이터 공급자의 특정 기능에 대한 자세한 내용은 데이터 공급자별 기능 섹션을 참조하세요.

각 데이터 공급자는 서로 다른 방식으로 데이터를 수집하고 생성합니다. 다음 항목에서 해당 서비스 범위에 대해 자세히 알아볼 수 있습니다.

- 적용 범위: Esri
- 적용 범위: Grab
- 적용 범위: HERE
- 적용 범위: Open Data

데이터에 문제가 발생하여 데이터 공급자에게 오류를 보고하려는 경우 다음 항목을 참조하세요.

- Esri에 오류 보고
- 데이터에 대한 오류 보고 GrabMaps
- 에 오류 신고 HERE
- Open Data에 대한 오류 신고 및 기여

## 맵 스타일

각 데이터 공급자는 제공하는 맵 데이터를 렌더링하기 위한 맵 스타일 세트를 제공합니다. 예를 들어 스타일에는 위성 이미지가 포함될 수도 있고 탐색을 위해 도로를 표시하도록 최적화될 수도 있습니다. 다음 항목에서 각 공급자의 스타일 목록 및 예를 찾을 수 있습니다.

- Esri 맵 스타일
- Grab 맵 스타일
- HERE맵 스타일
- Open Data 맵 스타일

## 각 데이터 공급자에 대한 추가 정보

다음 링크는 각 데이터 공급자에 대한 자세한 정보를 제공합니다.

- Esri
- GrabMaps
- HERE기술
- Open Data(오픈 데이터)

## Esri

Amazon Location Service는 Esri의 위치 서비스를 사용하여 AWS 고객이 맵을 사용하고, 지오코딩하고, 경로를 효과적으로 계산할 수 있도록 지원합니다. Esri의 위치 서비스는 지도 제작자, 지리학자, 인구 통계학자로 구성된 전문가 팀이 선별한 신뢰할 수 있는 고품질 ready-to-use 위치 데이터로 구축됩니다.

추가 내용을 알아보려면 Amazon Location Service 데이터 공급자의 Esri를 참조하세요.

주제

- Esri 맵 스타일
- 적용 범위: Esri
- 이용 약관 및 데이터 저작자 표시: Esri
- Esri에 오류 보고

# Esri 맵 스타일

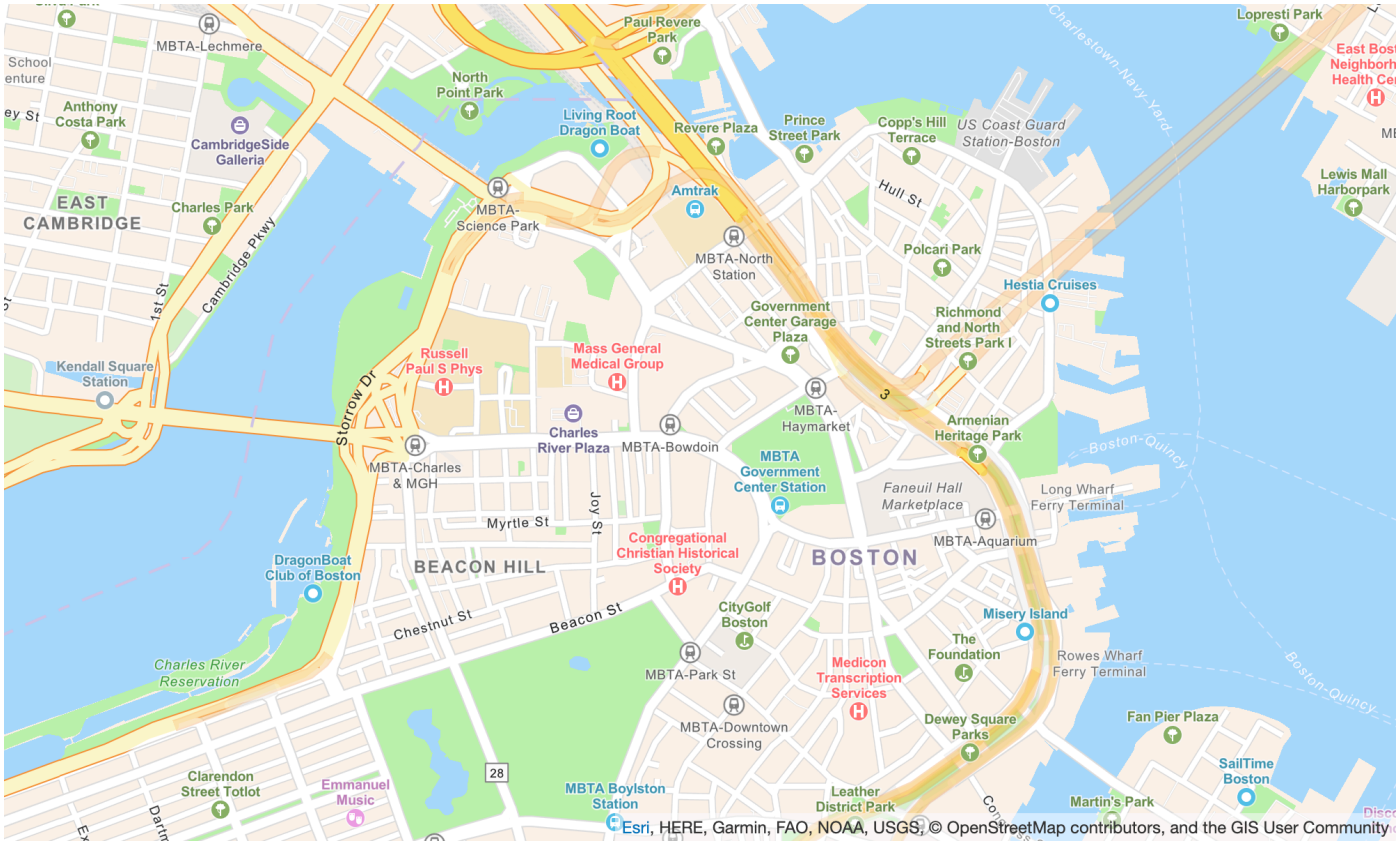
Amazon Location Service는 맵 리소스를 생성할 때 다음과 같은 Esri 맵 스타일을 지원합니다.

> **Note**
>
> 이 섹션에 나열되지 않은 Esri 맵 스타일은 지원되지 않습니다.

Esri 벡터 스타일은 대체 정치적 관점을(를) 지원합니다.

Esri Navigation

Esri Navigation




맵 스타일 이름: `VectorEsriNavigation`

이 맵은 모바일 디바이스에서 낮 동안 사용하도록 설계된 사용자 정의 내비게이션 맵 스타일로 상징화된 세계에 대한 상세한 베이스맵을 제공합니다.

이 포괄적인 스트리트 맵에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다. 이 맵의 벡터 타일 레이어는 World Street Map

및 기타 Esri 베이스맵에 사용된 것과 동일한 데이터 소스를 사용하여 구축됩니다. 추가 장소 데이터를 활용하도록 설정하여 `POI` 레이어를 활성화하세요. CustomLayers

자세한 내용은 Esri 웹사이트의 Esri World Navigation을 참고하세요.

> **Note**
>
> 위 그림의 `VectorEsriNavigation` 지도에는 `POI` 레이어가 활성화되어 있습니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Arial Italic
- Arial Regular
- Arial Bold
- Arial Unicode MS Bold
- Arial Unicode MS Regular

## Esri Imagery

Esri Imagery

맵 스타일 이름: `RasterEsriImagery`

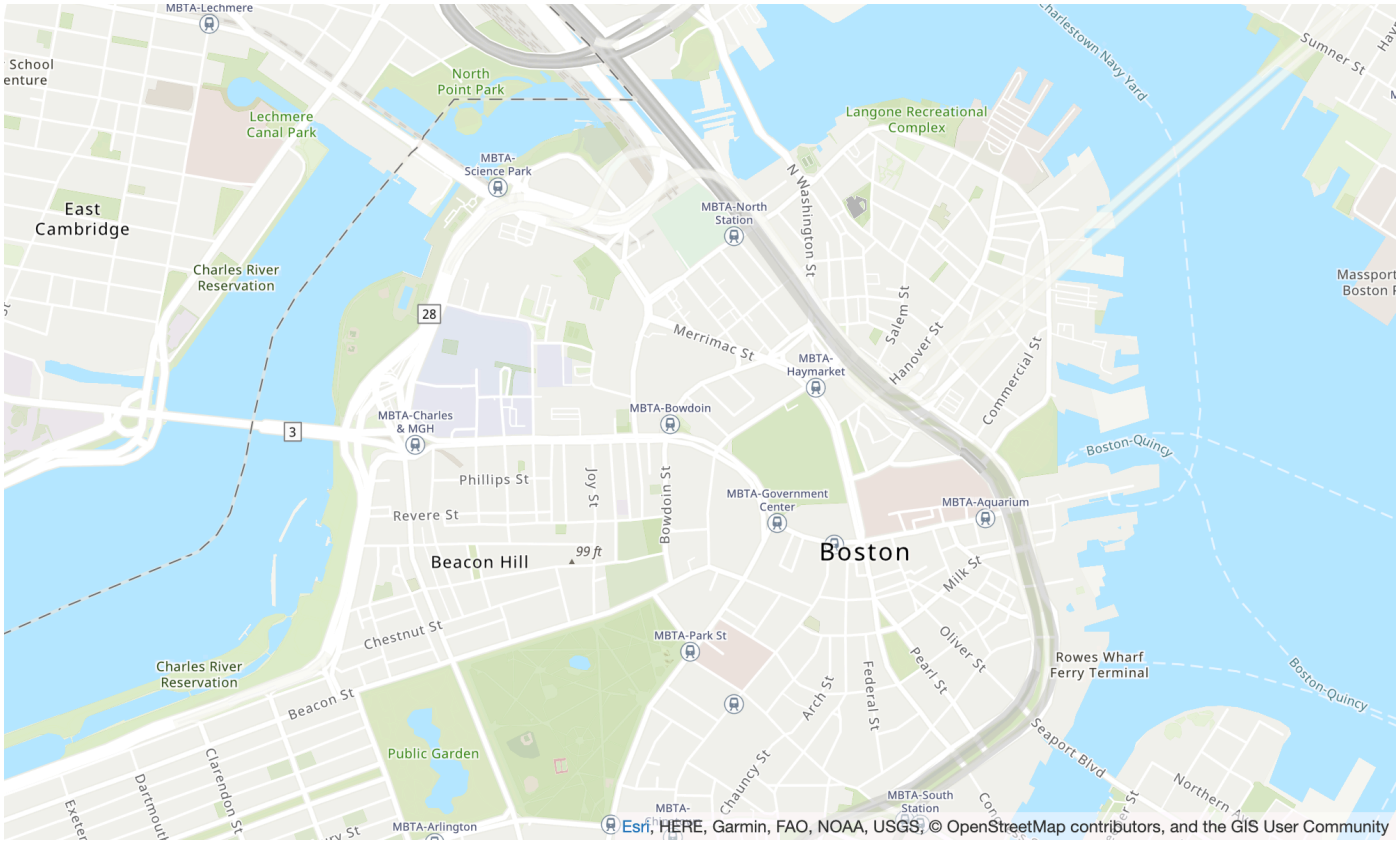
이 맵은 전 세계 여러 지역에서 1m 이상의 위성 및 항공 이미지를 제공하고 전 세계적으로는 저해상도 위성 이미지를 제공합니다.

맵에는 15m 소형 및 중간 규모의 이미지 (약 1:591 M에서 ~ 1:72 k) 와 2.5m 세계 SPOT 이미지 (약 1:288 k ~ ~ 1:72 k) 가 포함됩니다. 이 맵에는 Maxar에서 촬영한 미국 본토와 서유럽 일부 지역의 0.5m 해상도 이미지가 포함되어 있습니다. 이 맵에는 세계 여러 지역에서 1m 미만의 Maxar 이미지가 추가로 포함되어 있습니다. 전 세계 다른 지역에서는 사용자 커뮤니티가 다양한 해상도의 이미지를 제공했습니다. GIS 일부 커뮤니티에서는 ~1:280 스케일의 초고해상도 이미지(최저 0.03m)를 사용할 수 있습니다.

자세한 내용은 Esri 웹사이트의 Esri World Imagery를 참조하세요.

## Esri Light

### Esri Light




맵 스타일 이름: `VectorEsriTopographic`

이는 클래식 Esri 맵 스타일로 상징화된 세계의 상세한 베이스맵을 제공합니다. 여기에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다.

이 베이스맵은 미국 지질조사국 (), 미국 환경보호국 (), 미국 국립공원관리청 (USGS), 유엔 식량농업기구 (EPA), 캐나다 천연자원부 (NPS), Esri 등 여러 데이터 제공자의 다양한 신뢰할 수 있는 출처에서 작성되었습니다. FAO NRCAN HERE 일부 지역의 데이터는 기고자로부터 제공됩니다. OpenStreetMap 또한 데이터는 커뮤니티에서 GIS 제공합니다.

#### 글꼴

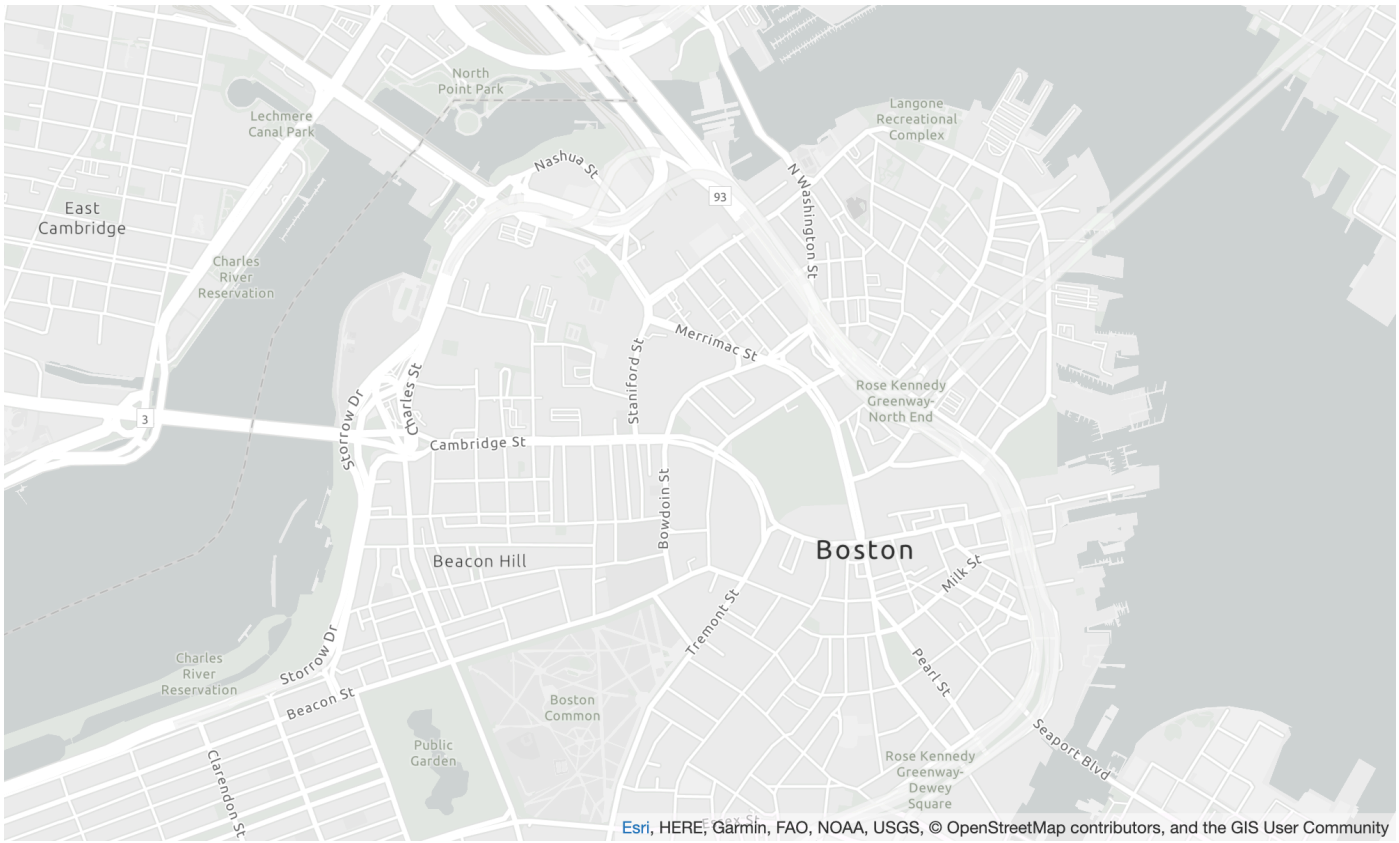
Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Noto Sans Italic

- Noto Sans Regular
- Noto Sans Bold
- Noto Serif Regular
- Roboto Condensed Light Italic

## Esri Light Gray Canvas

### Esri Light Gray Canvas




맵 스타일 이름: `VectorEsriLightGrayCanvas`

이 맵은 항목별 콘텐츠에 관심을 끌 수 있도록 설계된 최소한의 색상, 레이블 및 지형지물로 옅은 회색, 흐릿한 회색 배경 스타일로 상징되는 세계에 대한 상세한 베이스맵을 제공합니다.

이 벡터 타일 레이어는 라이트 그레이 캔버스 및 기타 Esri 베이스맵에 사용된 것과 동일한 데이터 소스를 사용하여 구축됩니다. 맵에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다.

자세한 내용은 Esri 웹사이트의 Esri Light Gray Canvas를 참고하세요.

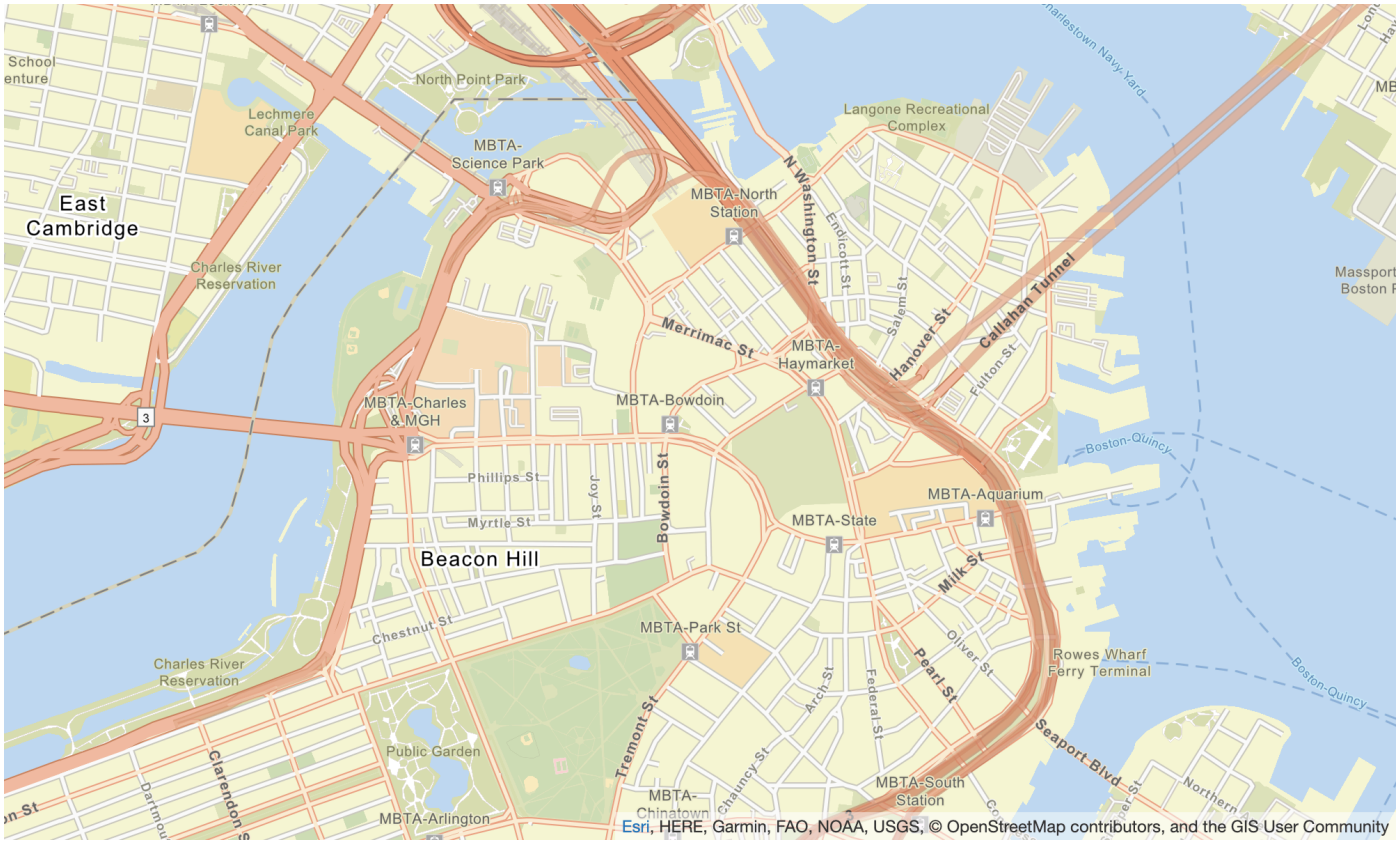
글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Ubuntu Italic
- Ubuntu Regular
- Ubuntu Light
- Ubuntu Bold

## Esri Street Map

Esri Street Map




맵 스타일 이름: `VectorEsriStreets`

이 맵은 모바일 디바이스에서 낮 동안 사용하도록 설계된 사용자 정의 내비게이션 맵 스타일로 상징화된 세계에 대한 상세한 베이스맵을 제공합니다.

이 포괄적인 스트리트 맵에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다. 여기에는 상점, 서비스, 레스토랑, 명소 및 기타 관심 지점 등과 같은 다양한 장소도 포함됩니다. 이 맵의 벡터 타일 레이어는 World Street Map 및 기타 Esri 베이스맵에 사용된 것과 동일한 데이터 소스를 사용하여 구축됩니다.

자세한 내용은 Esri 웹 사이트의 Esri World Street를 참고하세요.

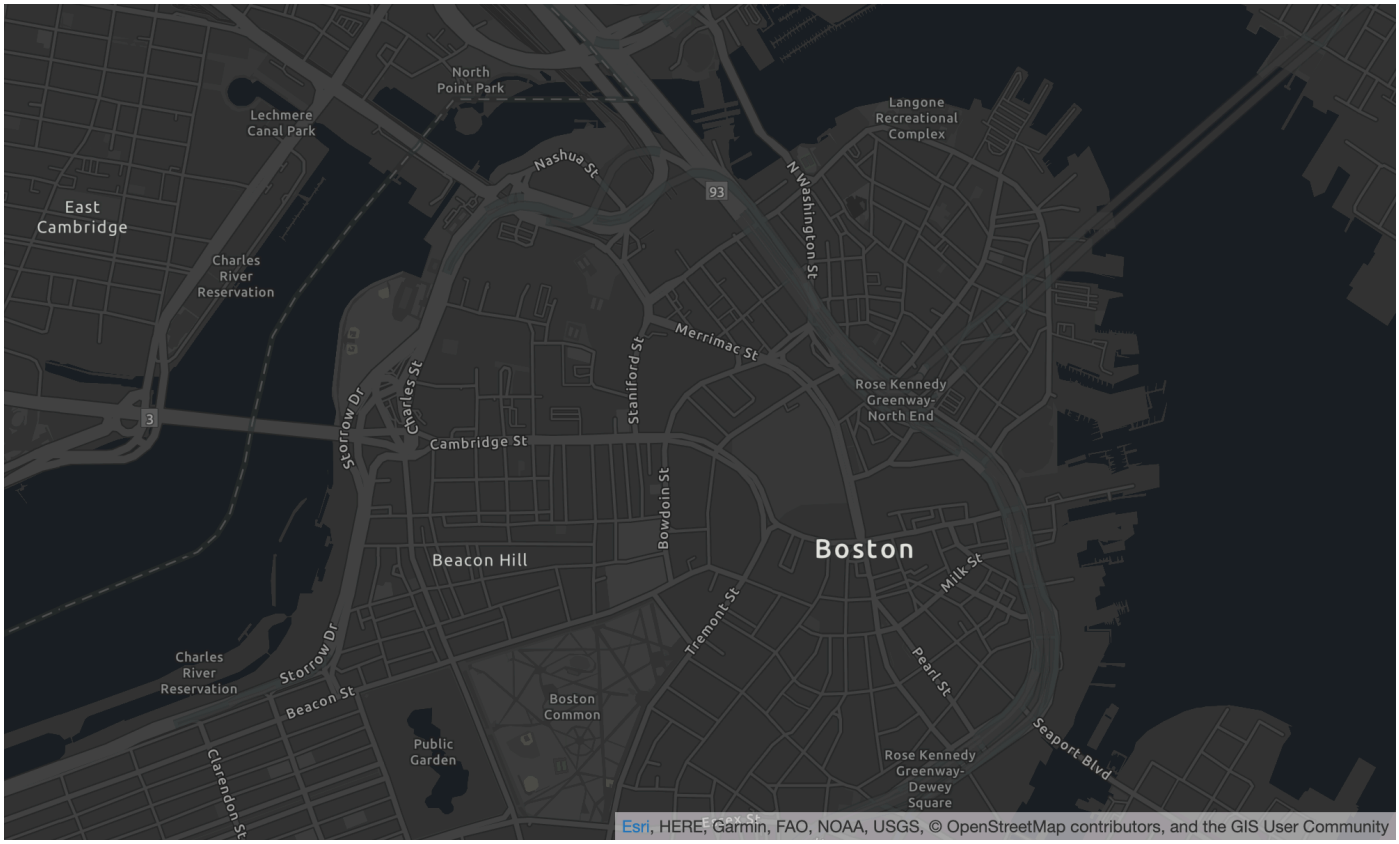
글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Arial Italic
- Arial Regular
- Arial Bold
- Arial Unicode MS Bold
- Arial Unicode MS Regular

Esri Dark Gray Canvas

Esri 다크 그레이 캔버스




맵 스타일 이름: `VectorEsriDarkGrayCanvas`

이 맵은 항목별 콘텐츠에 관심을 끌 수 있도록 설계된 최소한의 색상, 레이블 및 지형지물로 짙은 회색, 흐릿한 회색 배경 스타일로 상징되는 세계에 대한 상세한 벡터 베이스맵을 제공합니다.

이 맵에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다. 이 맵의 벡터 타일 레이어는 다크 그레이 캔버스 래스터 맵 및 기타 Esri 베이스맵에 사용된 것과 동일한 데이터 소스를 사용하여 구축됩니다.

자세한 내용은 Esri 웹사이트의 Esri 다크 그레이 캔버스를 참조하세요.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Ubuntu Medium Italic
- Ubuntu Medium
- Ubuntu Italic

- Ubuntu Regular
- Ubuntu Bold

### 적용 범위: Esri

Esri를 데이터 공급자로 사용하여 장소 색인 리소스를 생성할 때 지오코딩, 역지오코딩, 검색에 대한 쿼리를 지원하거나 경로 계산기 리소스를 생성할 때 경로 계산을 위한 쿼리를 지원할 수 있습니다.

Esri는 전 세계 여러 리전에서 다양한 수준의 데이터 품질을 제공합니다. 관심 리전의 적용 범위에 대한 추가 정보는 다음을 참조하세요.

- 지오코딩 적용 범위에 대한 Esri 세부정보
- 도로망 및 교통 범위에 대한 Esri 세부정보

### 이용 약관 및 데이터 저작자 표시: Esri

Esri의 데이터를 사용하기 전에 Esri 및 에 적용되는 라이선스 조건을 비롯한 모든 관련 법적 요구 사항을 준수할 수 있는지 확인하세요. AWS

AWS 요구 사항에 대한 자세한 내용은 AWS서비스 약관을 참고하세요.

Esri의 저작자 표시 지침에 대한 자세한 내용은 Esri의 데이터 저작자 표시 및 사용 약관을 참조하세요.

### Esri에 오류 보고

데이터에 문제가 발생하여 오류 및 불일치를 Esri에 보고하려면 Esri의 기술 지원 문서 방법: 베이스맵 및 지오코딩에 대한 피드백 제공을 팔로우하세요.

## GrabMaps

Grab은 수백만 명의 운전자 파트너와 고객을 보유하고 있는 동남아시아 최대 운송업체입니다. 자회사인 는 해당 국가/지역에서 자체 용도 및 기타 용도로 up-to-date 매핑 데이터를 생성합니다. GrabMaps Amazon Location Service는 GrabMaps '위치 서비스를 사용하여 AWS 고객이 효과적으로 지도를 사용하고, 지오코딩하고, 경로를 계산할 수 있도록 지원합니다. GrabMaps'위치 서비스는 특히 동남아시아 국가를 대상으로 신뢰할 수 있는 고품질 ready-to-use 위치 데이터를 제공하도록 구축되었습니다.

추가 기능에 대한 자세한 내용은 Amazon Location Service 데이터 공급자를 참조하십시오 GrabMaps.

> **Important**
>
> Grab은 동남아시아 지역에 대한 맵만 제공하며, 아시아 태평양(싱가포르) 리전(ap-southeast-1)에서만 사용할 수 있습니다. 자세한 내용은 대상 국가/리전 및 지역 섹션을 참조하세요.

주제



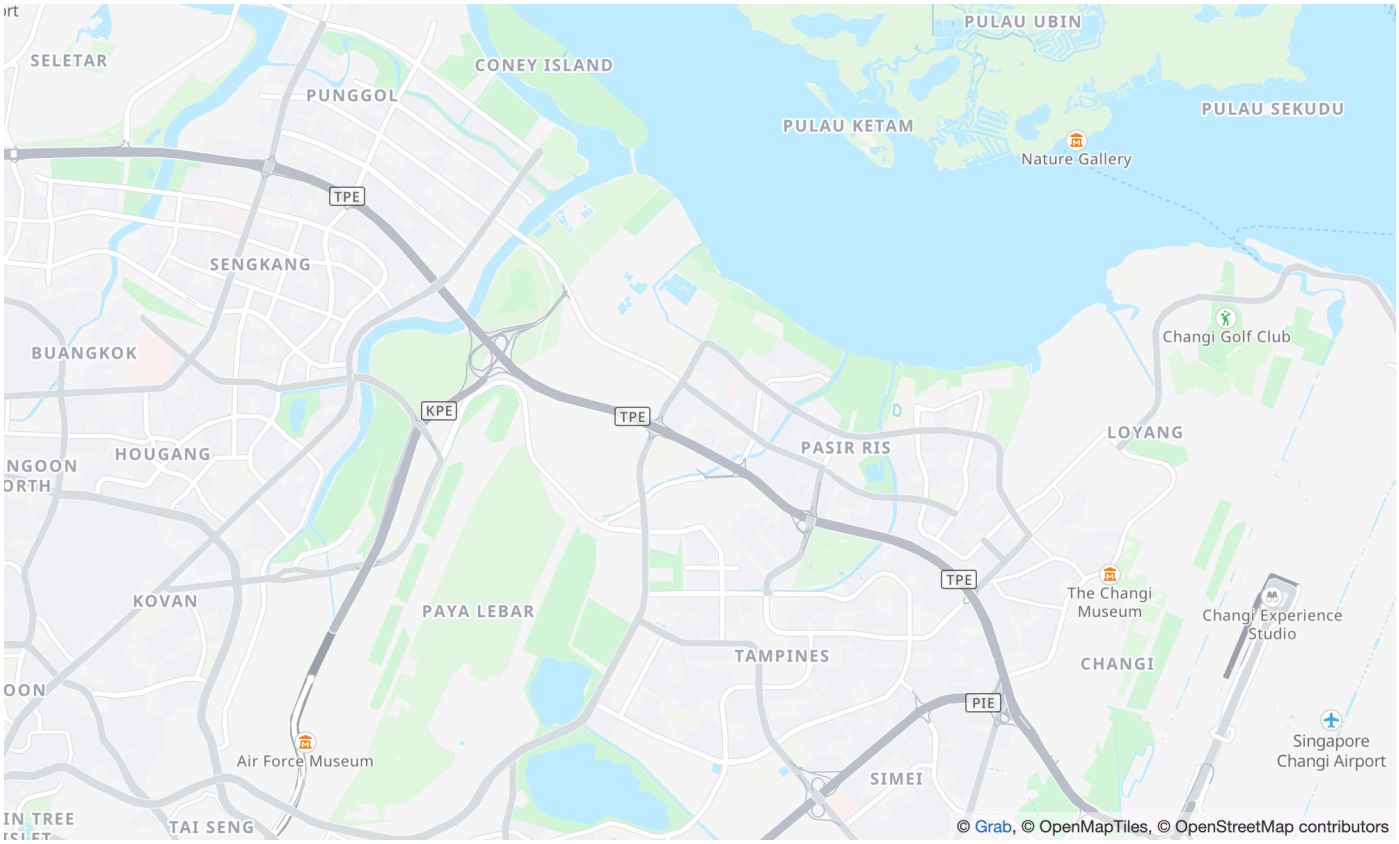
## Grab 맵 스타일

Amazon Location Service는 맵 리소스를 생성할 때 다음과 같은 Grab 맵 스타일을 지원합니다.

> **Note**
>
> 이 섹션에 나열되지 않은 Grab 맵 스타일은 현재 지원되지 않습니다.

Grab Standard Light Map

Grab 스탠더드 라이트 맵




맵 스타일 이름: `VectorGrabStandardLight`

상세한 토지 용도 색상, 지역 이름, 도로, 랜드마크 및 동남아시아를 포괄하는 관심 지점이 포함된 Grab의 표준 베이스맵입니다.

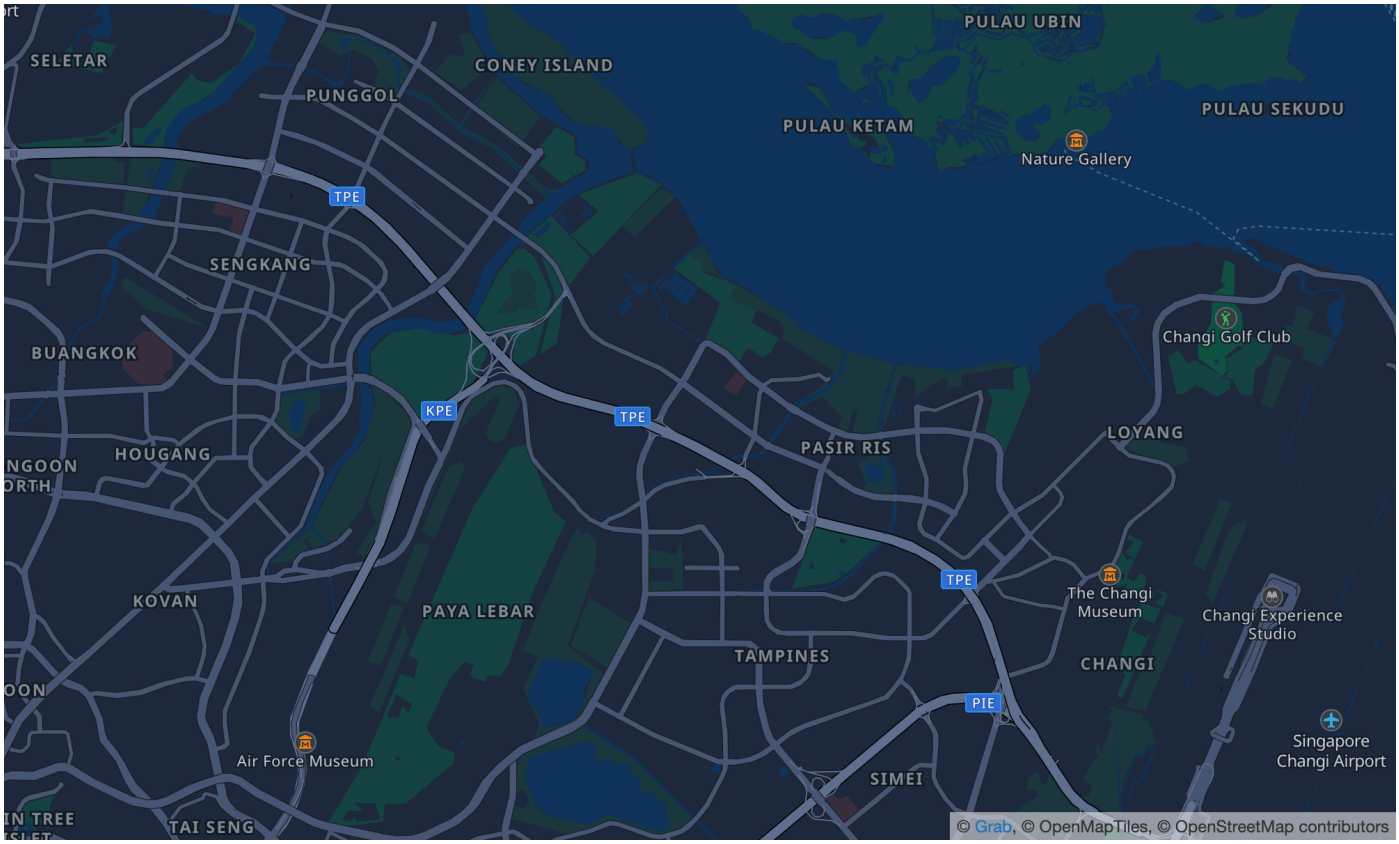
글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Noto Sans Regular
- Noto Sans Medium
- Noto Sans Bold

## Grab Standard Dark Map

Grab 스탠더드 다크 맵




맵 스타일 이름: `VectorGrabStandardDark`

상세한 토지 용도 색상, 지역 이름, 도로, 랜드마크 및 동남아시아를 포괄하는 관심 지점이 포함된 Grab의 표준 베이스맵의 다크 변형 버전입니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Noto Sans Regular
- Noto Sans Medium
- Noto Sans Bold

## 적용 범위: Grab

Grab을 데이터 공급자로 사용하여 장소 색인 리소스를 생성할 때 지오코딩, 역지오코딩 및 검색에 대한 쿼리를 지원하거나 경로 계산기 리소스를 생성할 때 경로 계산을 위한 쿼리를 지원할 수 있습니다.

## 대상 국가/리전 및 지역

Grab은 동남아시아 지역에 대한 맵만 제공하며, 아시아 태평양(싱가포르) 리전(ap-southeast-1)에서만 사용할 수 있습니다.

Grab은 다음 국가/리전에 대한 자세한 데이터를 제공합니다.

- 말레이시아
- 필리핀
- 태국
- 싱가포르
- 베트남
- 인도네시아
- 미얀마
- 캄보디아

> **Note**
>
> 이러한 영역 외부에서는 Grab을 데이터 공급자로 하여 작성된 Amazon Location Service 리소스가 어떠한 결과도 제공하지 않습니다. 여기에는 검색 결과 또는 경로가 포함됩니다.

Grab의 맵은 다음과 같은 경계 내에 있습니다.

- 남부 – 위도 -21.943045533438166
- 서쪽 – 경도 90.0
- 북쪽 – 위도 31.952162238024968
- 동쪽 – 경도 146.25

줌 레벨 1~4의 경우 Grab에는 글로벌 커버리지가 포함됩니다. 줌 레벨 5 이하의 경우 맵 타일은 이 경계 상자 내에서만 제공됩니다.

> **Note**
>
> 이 경계 상자 외부에서 Grab을 데이터 공급자로 사용하여 생성된 Amazon Location Service 맵 리소스는 맵 타일을 반환하지 않습니다. 애플리케이션에서 404 오류가 발생하지 않도록 하려

면 를 사용하여 맵의 범위를 설정합니다. MapLibre 에 설명된 대로 경계 상자를 사용하여 맵을 제한할 수 있습니다.

Grab 라우팅 이동 모드

라우팅의 경우 Grab은 앞서 열거한 모든 국가/리전에 대한 자동차 및 오토바이 경로를 제공합니다.

Grab은 트럭 라우팅을 지원하지 않습니다.

자전거 및 도보 경로의 경우 Grab은 다음 도시를 지원합니다.

- 싱가포르
- 자카르타
- 마닐라
- 클랑 밸리
- 방콕
- 호치민 시티
- 하노이

### 이용 약관 및 데이터 저작자 표시: Grab

Grab의 데이터를 사용할 때는 Grab 및 에 적용되는 라이선스 조건을 포함하여 모든 관련 법적 요구 사항을 준수해야 AWS합니다.

AWS 요구 사항에 대한 자세한 내용은 AWS서비스 약관을 참조하십시오.

GrabMaps'저작자 표시 가이드라인'에 대한 자세한 내용은 Grab의 데이터 어트리뷰션 및 이용 약관의 섹션 9.23을 참조하십시오.

### 데이터에 대한 오류 보고 GrabMaps

의 데이터에 문제가 발생하여 오류나 불일치를 보고하려면 AWS 기술 지원 부서에 문의하십시오. GrabMaps

## HERE기술

Amazon HERE Location Service는 기술의 위치 서비스를 사용하여 AWS 고객이 지도를 사용하고, 지오코딩하고, 경로를 효과적으로 계산할 수 있도록 지원합니다. HERE위치 데이터는 개방적이고 안전

하며 비공개인 위치 중심 플랫폼을 제공합니다. HERE위치 데이터를 선택하면 클라우드에 기본적으로 배포되는 정확하고 신선하며 강력한 데이터를 선택하는 것입니다. AWS

추가 기능 정보는 Amazon Location Service 데이터 공급자를 참조하십시오 HERE.

주제

- HERE맵 스타일
- 적용 범위: HERE
- 이용 약관 및 데이터 속성: HERE
- 에 오류 신고 HERE

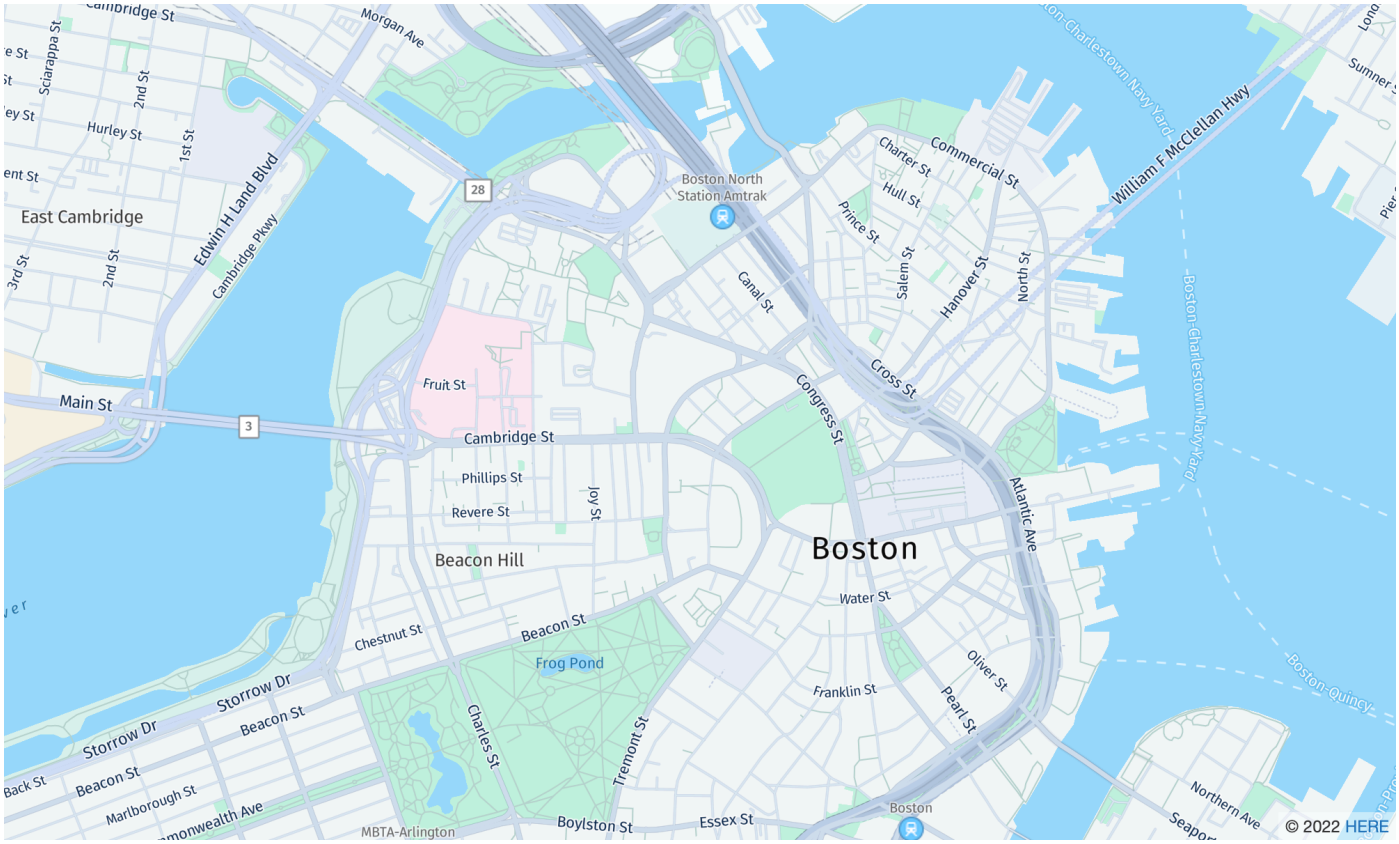
## HERE맵 스타일

Amazon Location Service는 HERE 맵 리소스를 생성할 때 다음과 같은 맵 스타일을 지원합니다.

> **Note**
>
> HERE이 섹션에 나열되지 않은 맵 스타일은 현재 지원되지 않습니다.

HERE Explore

HERE살펴보기




맵 스타일 이름: `VectorHereExplore`

HERE살펴보기

상세하고 중립적인 세계 베이스 맵입니다. 스트리트 맵에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다. 완전히 디자인된 일본 맵이 포함되어 있습니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Fira GO Italic
- Fira GO Regular
- Fira GO Bold
- 노토 산스 JP 라이트 CJK
- 노토 산스 JP 레귤러 CJK
- 모토 산스 JP 골드 CJK

### HERE Imagery

HERE이미지

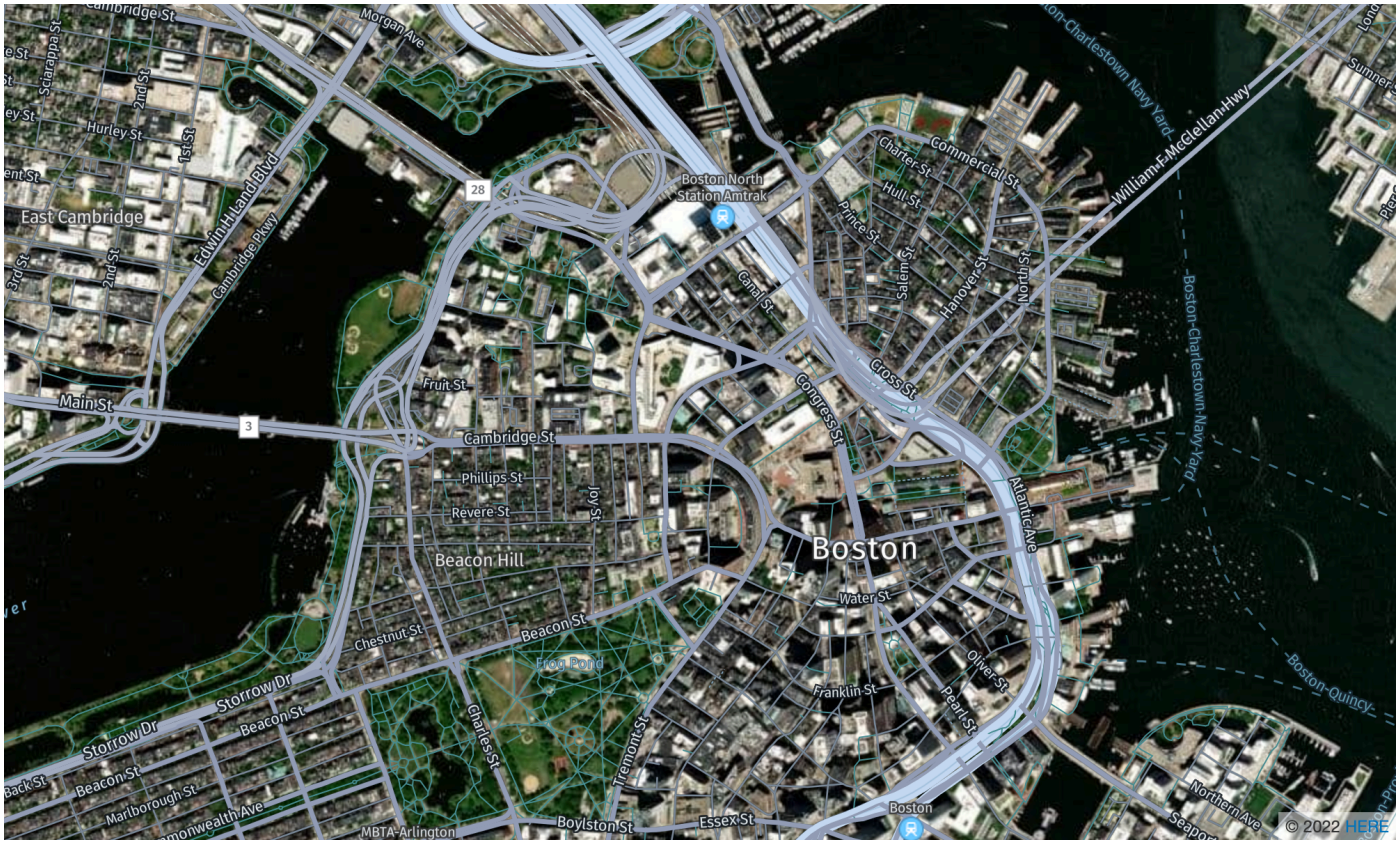

맵 스타일 이름: `RasterHereExploreSatellite`

HERE이미지

HERE이미지는 전 세계를 커버하는 고해상도 위성 이미지를 제공합니다.

### HERE Hybrid

HERE하이브리드




맵 스타일 이름: `HybridHereExploreSatellite`

HERE하이브리드

HERE하이브리드 스타일은 위성 이미지 위에 도로망, 거리 이름 및 도시 레이블을 표시합니다. 이 스타일은 두 개의 맵 타일, 즉 배경의 위성 이미지(래스터 타일)와 상단의 도로망 및 레이블(벡터 타일)을 오버레이합니다. 이 스타일은 맵을 렌더링하는 데 필요한 래스터 타일과 벡터 타일을 모두 자동으로 검색합니다.

> **Note**
>
> 하이브리드 스타일은 표시되는 지도를 렌더링할 때 벡터 및 래스터 타일을 모두 사용합니다. 즉, 벡터 또는 래스터 타일만 사용할 때보다 더 많은 타일이 검색됩니다. 요금에는 검색된 모든 타일이 포함됩니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Fira GO Italic
- Fira GO Regular
- Fira GO Bold
- 노토 산스 JP 라이트 CJK
- 노토 산스 JP 레귤러 CJK
- 모토 산스 JP 골드 CJK

### HERE Contrast (Berlin)

HERE콘트라스트 (베를린)


맵 스타일 이름: `VectorHereContrast`

HERE콘트라스트 (베를린)

3D와 2D 렌더링을 혼합한 세계의 상세한 베이스 맵입니다. 이 하이 콘트라스트 맵에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다.

글꼴

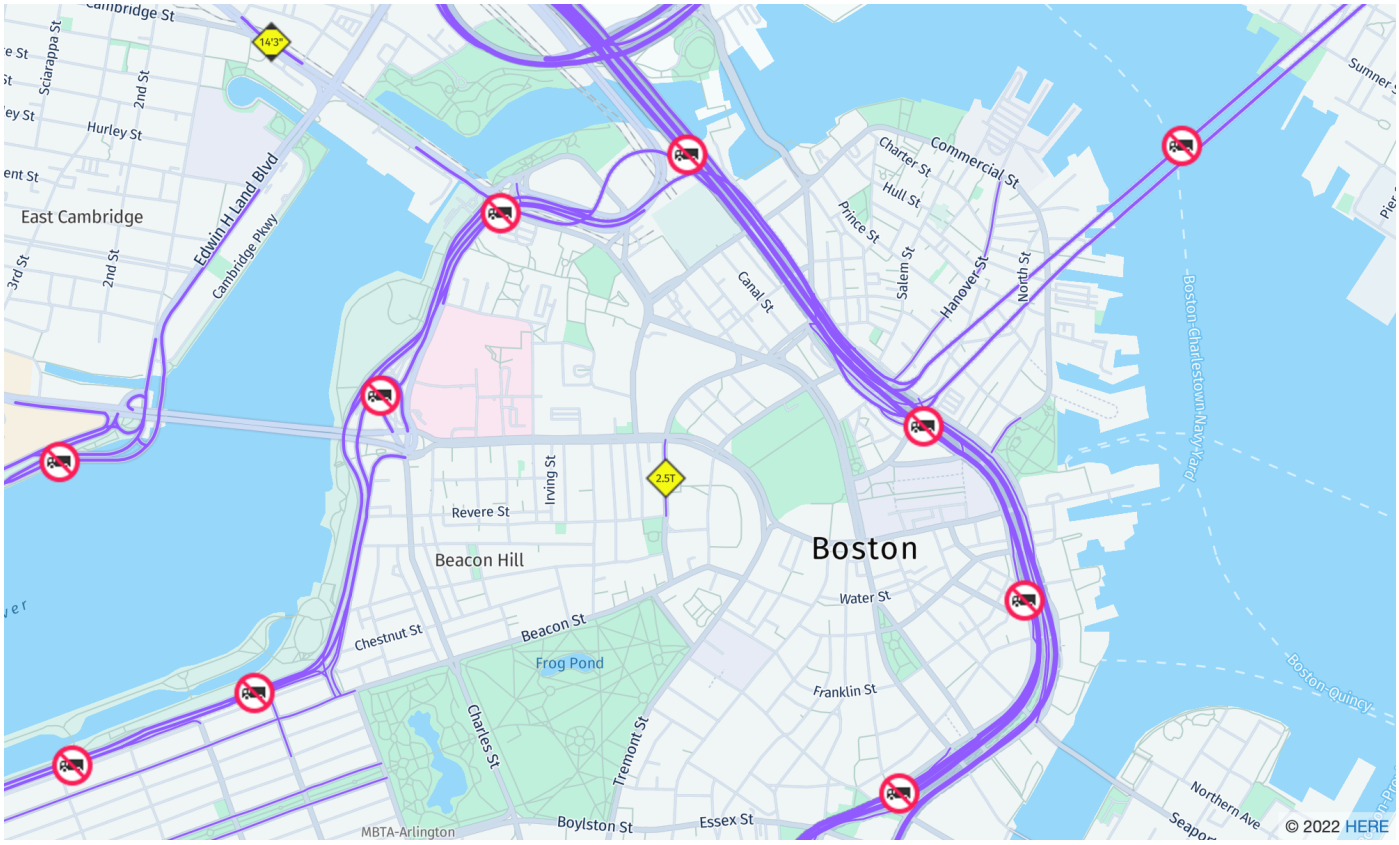
Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Fira GO Regular
- Fira GO Bold

> **Note**
>
> 이 스타일은 `VectorHereBerlin` (HERE베를린 지도) 에서 이름이 변경되었습니다. `VectorHereBerlin`더 이상 사용되지 않지만 이를 사용하는 애플리케이션에서는 계속 사용할 수 있습니다.

## HERE Explore Truck

HERE익스플로어 트럭




맵 스타일 이름: `VectorHereExploreTruck`

HERE익스플로어 트럭

상세하고 중립적인 세계 베이스 맵입니다. HEREExplore 스타일을 기반으로 구축된 스트리트 맵은 운송 및 물류 분야의 사용 사례를 지원하기 위해 심볼과 아이콘으로 트랙 제한 및 속성 (너비, 높이 등HAZMAT) 을 강조 표시합니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Fira GO Italic
- Fira GO Regular
- Fira GO Bold
- 노토 산스 JP 라이트 CJK
- 노토 산스 JP 레귤러 CJK
- 모토 산스 JP 골드 CJK

전 세계 여러 지역의 지도 데이터 품질에 대한 추가 정보는 지도 범위를 참조하십시오HERE.

## 적용 범위: HERE

장소 색인 리소스를 생성할 때 지오코딩, 리버스 지오코딩, 검색에 대한 쿼리를 지원하거나 경로 계산기 리소스를 생성할 때 경로 계산을 위한 쿼리를 지원하는 데이터 HERE 공급자로 사용할 수 있습니다.

HERE전 세계 여러 지역에서 다양한 수준의 데이터 품질을 제공합니다. 관심 있는 리전의 적용 범위에 대한 자세한 내용은 다음을 참조하세요.

- HEREHERE지오코딩 적용 범위
- HEREHERE자동차 라우팅 커버리지
- HEREHERE트럭 라우팅 커버리지

## 이용 약관 및 데이터 속성: HERE

HERE데이터를 사용하기 전에 HERE 및 AWS에 적용되는 라이선스 조건을 포함하여 해당하는 모든 법적 요구 사항을 준수할 수 있는지 확인하십시오. 라이선스 제한으로 인해 일본 내 위치에 대한 지오코딩 결과를 저장하는 HERE 데 사용할 수 없습니다.

AWS 요구 사항에 대한 자세한 내용은 AWSAWS서비스 약관을 참조하십시오.

HERE의 저작자 표시 가이드라인에 대한 자세한 내용은 위치 및 기타 콘텐츠에 적용되는 HERE Technologies의 공급업체 약관의 섹션 2를 참조하십시오.

### 에 오류 신고 HERE

지도 오류와 불일치를 HERE 신고하려면 로 이동하여 지도 오류 신고를 선택합니다. https://www.here.com/contact

## Open Data(오픈 데이터)

Amazon Location Service는 Open Data 공급자를 통해 오픈 소스 맵 데이터에 대한 액세스를 제공합니다. Open Data는 OpenStreetMap (OSM), 자연 지구 및 기타 공개 데이터 소스의 데이라이트 맵 분포를 기반으로 구축된 글로벌 베이스맵을 제공합니다. 제공되는 맵은 웹 및 모바일 환경에서 물류 및 배송, 데이터 시각화를 포함한 다양한 애플리케이션과 사용 사례를 지원하도록 설계되었습니다. 백만 개 이상의 맵 제작자가 있는 OSM 커뮤니티에서는 매일 수십만 개의 피처를 업데이트합니다. Amazon Location Service는 이러한 편집 내용을 정기적으로 통합합니다.

추가 내용을 알아보려면 Amazon Location Service 데이터 공급자의 Open Data를 참조하세요.

주제

- Open Data 맵 스타일
- 적용 범위: Open Data
- 이용 약관 및 데이터 저작자 표시: Open Data
- Open Data에 대한 오류 신고 및 기여

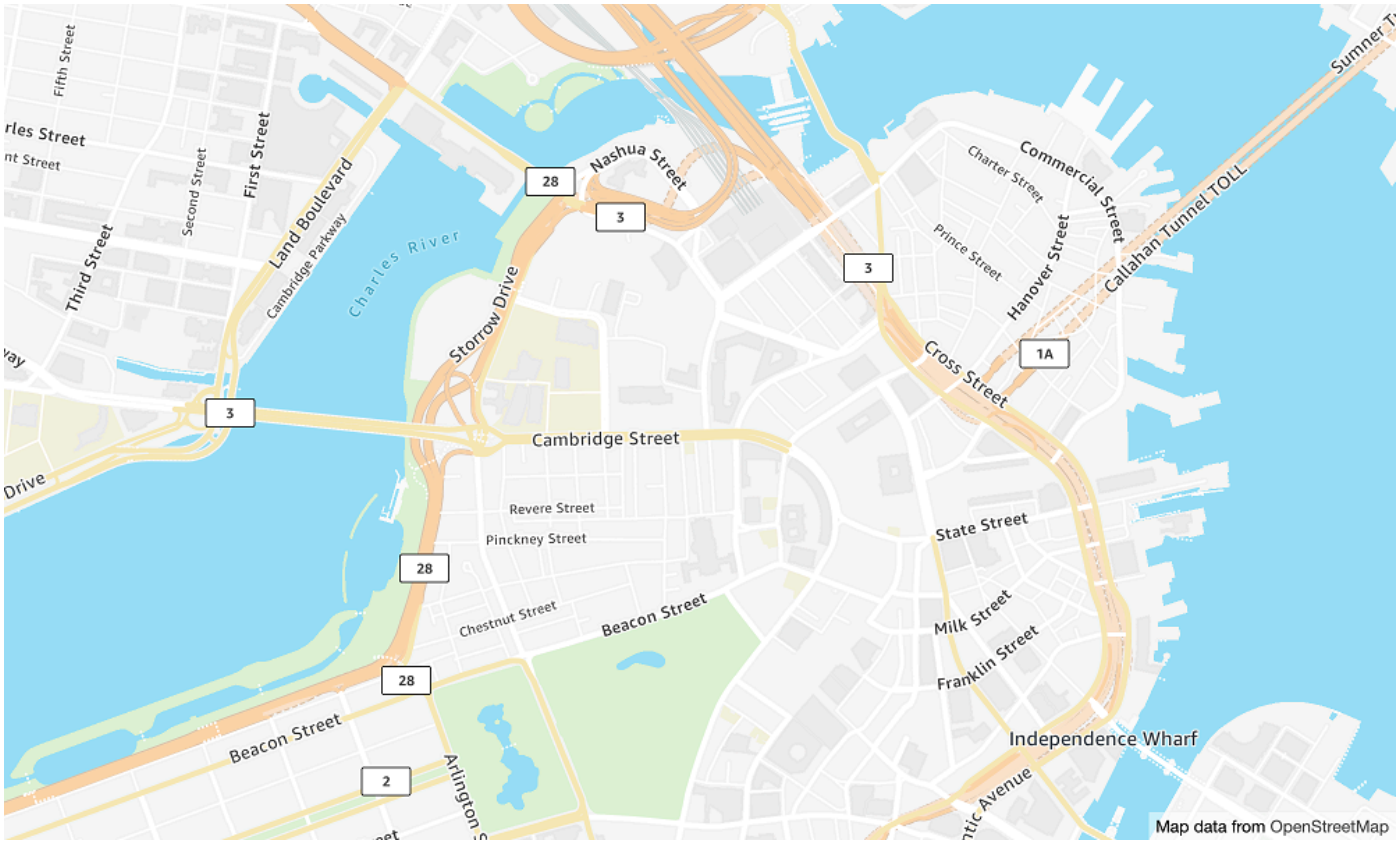
### Open Data 맵 스타일

Amazon Location Service는 맵 리소스를 생성할 때 다음과 같은 맵 스타일을 지원합니다.

Open Data 맵 스타일은 대체 정치적 관점을(를) 지원합니다.

Open Data Standard Light

Open Data 스탠더드 라이트




맵 스타일 이름: `VectorOpenDataStandardLight`

이는 웹사이트 및 모바일 애플리케이션 사용에 적합한 라이트 맵 스타일로 세계에 대한 상세한 베이스맵을 제공합니다. 여기에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다.

이 베이스맵은 OpenStreetMap (OSM) 기여자들로부터 컴파일된 OSM 데이라이트 맵 배포를 기반으로 합니다. OSM커뮤니티에는 매일 500,000개 이상의 기능을 업데이트하는 180만 명 이상의 기여자가 포함되어 있습니다. Amazon Location Service는 이러한 편집 내용을 정기적으로 통합합니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

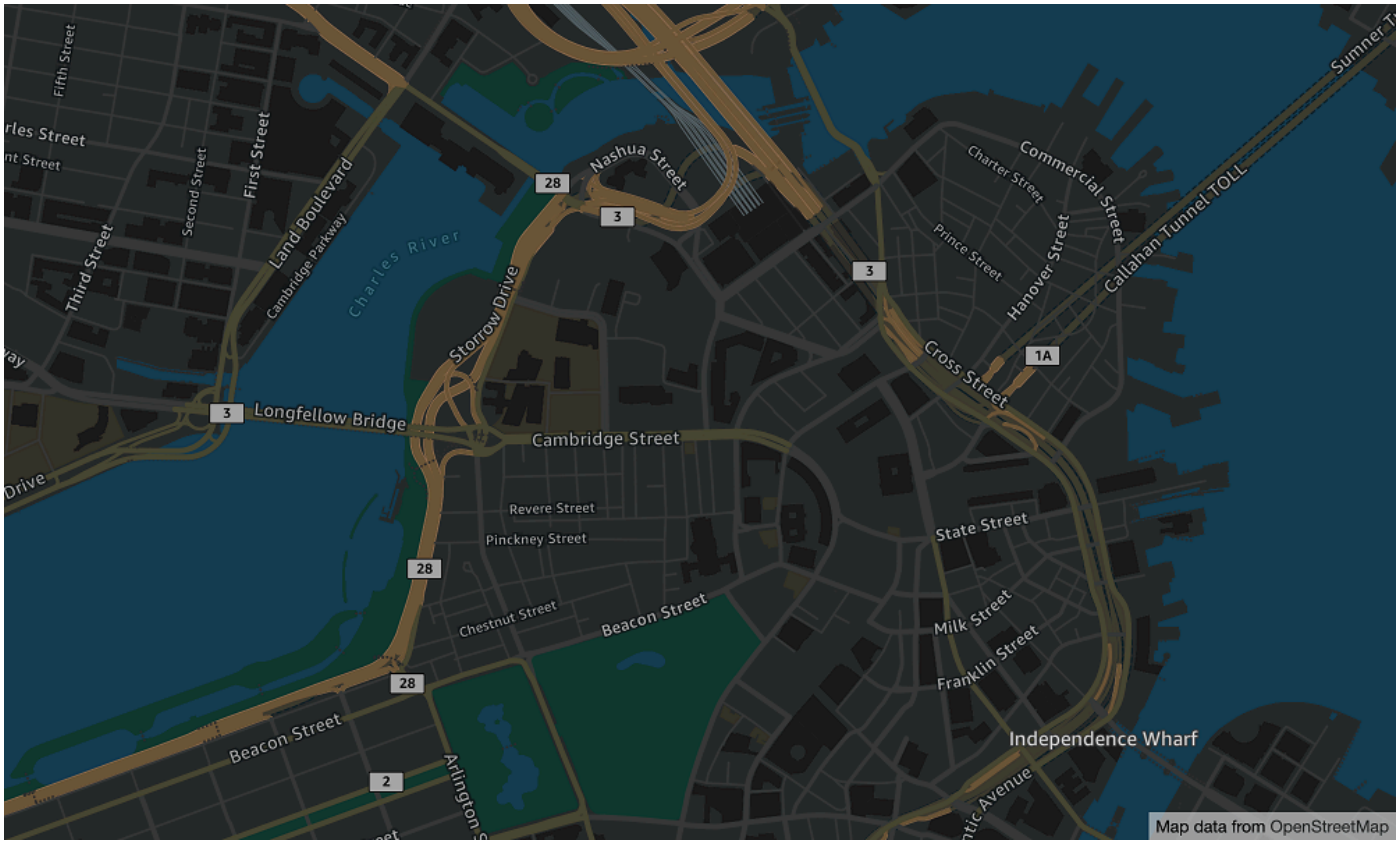
- Amazon Ember Bold, Noto Sans Bold
- Amazon Ember Condensed RC Bold, Noto Sans Bold
- Amazon Ember Condensed RC Regular, Noto Sans Regular

- Amazon Ember Medium, Noto Sans Medium
- Amazon Ember Regular Italic, Noto Sans Italic
- Amazon Ember Regular, Noto Sans Regular
- Amazon Ember Regular, Noto Sans Regular, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold, Noto Sans Bold, Noto Sans Arabic Condensed Bold
- Amazon Ember Bold, Noto Sans Bold, Noto Sans Arabic Bold
- Amazon Ember Regular Italic, Noto Sans Italic, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular, Noto Sans Regular, Noto Sans Arabic Condensed Regular
- Amazon Ember Medium, Noto Sans Medium, Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataStandardLight`에서 사용되는 글꼴은 대부분의 글리프에 `Amazon Ember`을(를) 사용하고, `Amazon Ember`에서 지원되지 않는 글리프에는 `Noto Sans`을(를) 사용하는 결합 글꼴입니다.

Open Data Standard Dark

Open Data 스탠더드 다크




맵 스타일 이름: `VectorOpenDataStandardDark`

다크 테마의 맵 스타일로 세계에 대한 자세한 베이스맵을 제공하며 웹사이트 및 모바일 애플리케이션 사용에 적합합니다. 여기에는 고속도로, 주요 도로, 보조 도로, 철도, 수변 지형, 도시, 공원, 랜드마크, 건물 바닥 면적 및 행정구역 경계가 포함됩니다.

이 베이스맵은 () 기여자들로부터 OpenStreetMap 컴파일된 OSM 데이라이트 맵 배포를 기반으로 합니다. OSM OSM커뮤니티에는 매일 500,000개 이상의 기능을 업데이트하는 180만 명 이상의 기여자가 포함되어 있습니다. Amazon Location Service는 이러한 편집 내용을 정기적으로 통합합니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

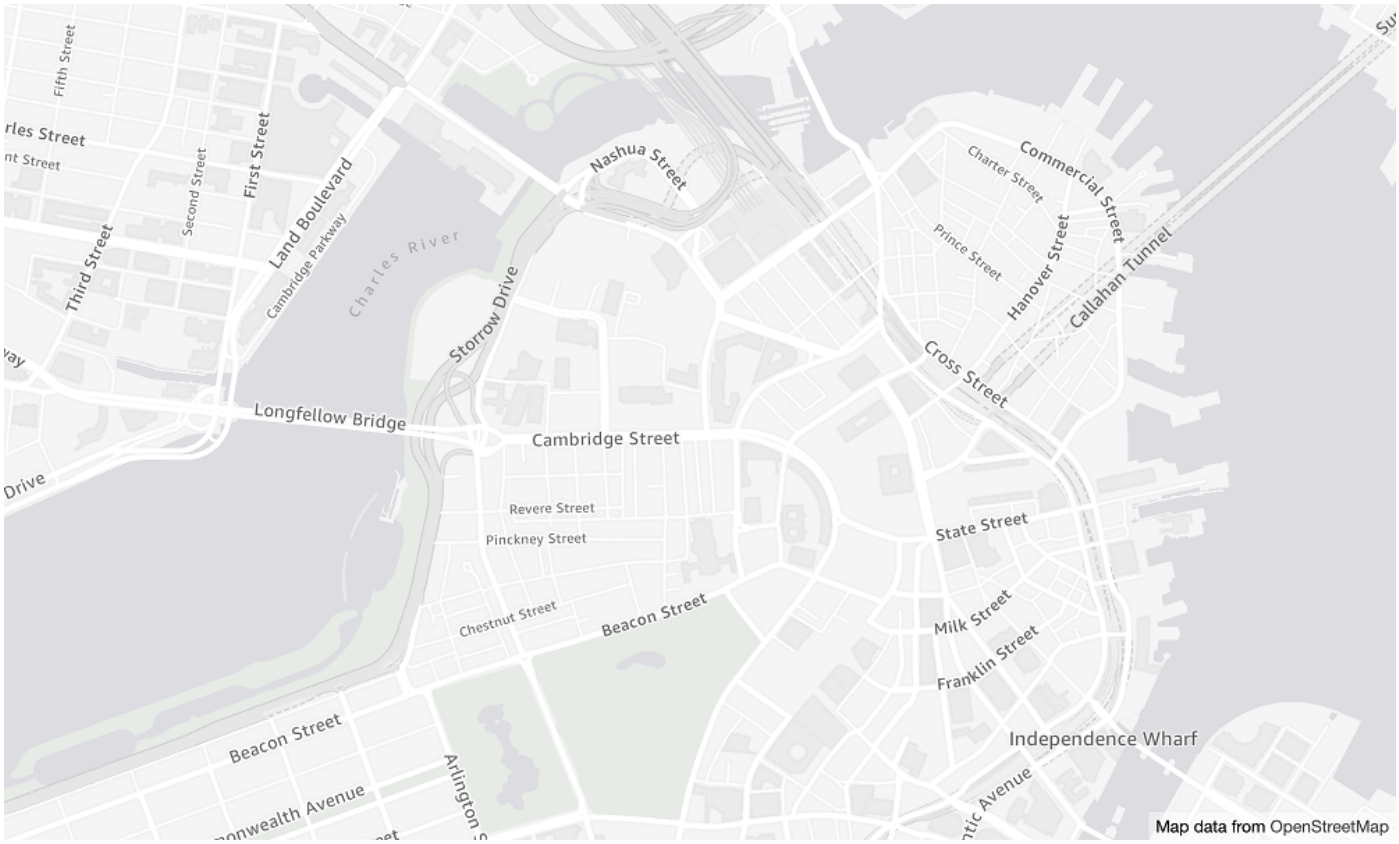
- Amazon Ember Bold, Noto Sans Bold
- Amazon Ember Condensed RC Bold, Noto Sans Bold
- Amazon Ember Condensed RC Regular, Noto Sans Regular

- Amazon Ember Medium, Noto Sans Medium
- Amazon Ember Regular Italic, Noto Sans Italic
- Amazon Ember Regular, Noto Sans Regular
- Amazon Ember Regular, Noto Sans Regular, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold, Noto Sans Bold, Noto Sans Arabic Condensed Bold
- Amazon Ember Bold, Noto Sans Bold, Noto Sans Arabic Bold
- Amazon Ember Regular Italic, Noto Sans Italic, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular, Noto Sans Regular, Noto Sans Arabic Condensed Regular
- Amazon Ember Medium, Noto Sans Medium, Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataStandardDark`에서 사용되는 글꼴은 대부분의 글리프에 `Amazon Ember`을(를) 사용하고, `Amazon Ember`에서 지원되지 않는 글리프에는 `Noto Sans`을(를) 사용하는 결합 글꼴입니다.

Open Data Visualization Light

Open Data 시각화 라이트




맵 스타일 이름: `VectorOpenDataVisualizationLight`

이는 차분한 색상의 밝은 테마 스타일로 기능이 더 적어 오버레이된 데이터를 이해하는 데 도움이 됩니다.

이 베이스맵은 () 기여자들로부터 OpenStreetMap 컴파일된 OSM 데이라이트 맵 배포를 기반으로 합니다. OSM OSM커뮤니티에는 매일 500,000개 이상의 기능을 업데이트하는 180만 명 이상의 기여자가 포함되어 있습니다. Amazon Location Service는 이러한 편집 내용을 정기적으로 통합합니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

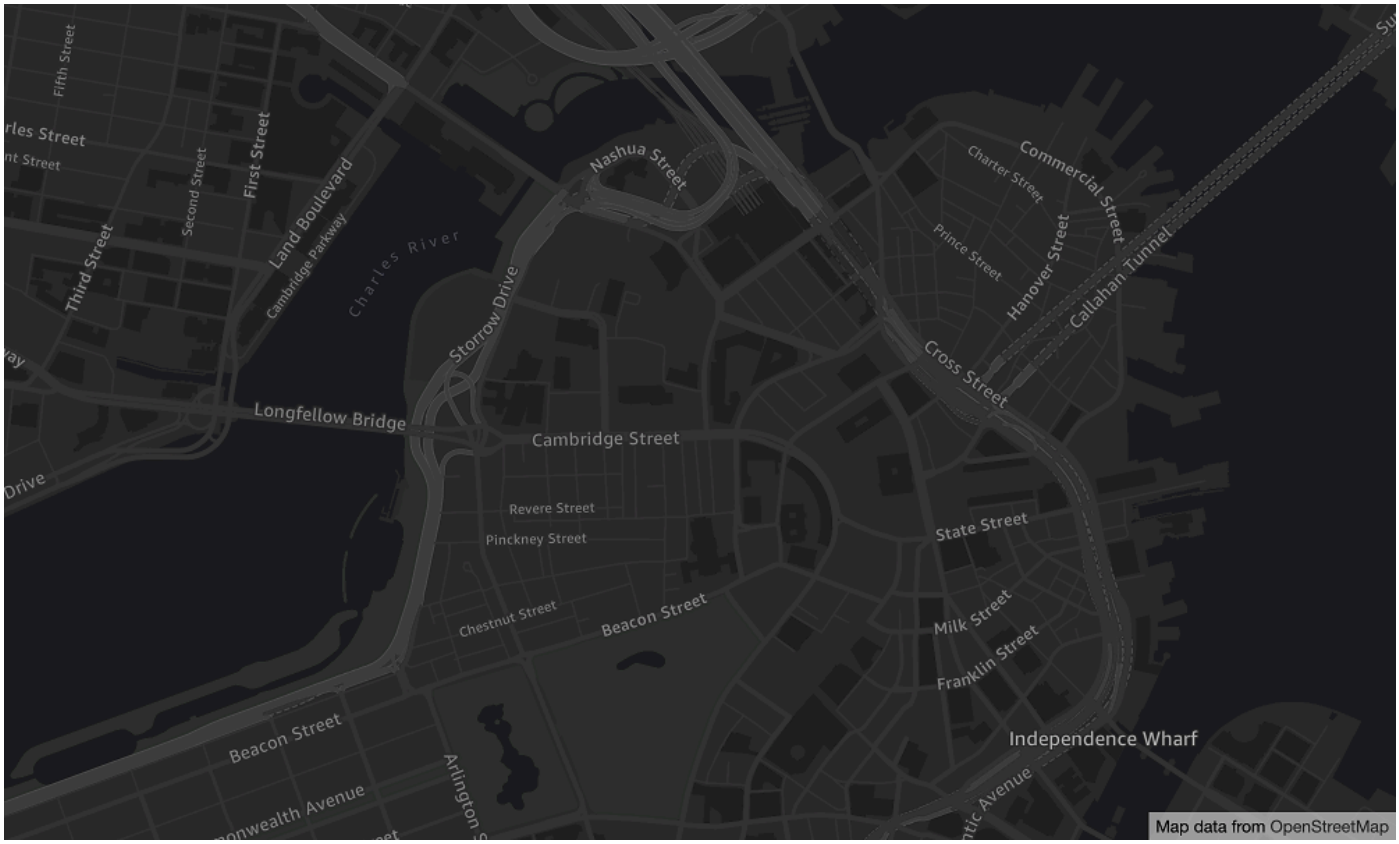
- Amazon Ember Bold, Noto Sans Bold
- Amazon Ember Condensed RC Bold, Noto Sans Bold
- Amazon Ember Condensed RC Regular, Noto Sans Regular
- Amazon Ember Medium, Noto Sans Medium

- Amazon Ember Regular Italic, Noto Sans Italic
- Amazon Ember Regular, Noto Sans Regular
- Amazon Ember Regular, Noto Sans Regular, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold, Noto Sans Bold, Noto Sans Arabic Condensed Bold
- Amazon Ember Bold, Noto Sans Bold, Noto Sans Arabic Bold
- Amazon Ember Regular Italic, Noto Sans Italic, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular, Noto Sans Regular, Noto Sans Arabic Condensed Regular
- Amazon Ember Medium, Noto Sans Medium, Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataVisualizationLight`에서 사용되는 글꼴은 대부분의 글리프에 `Amazon Ember`을(를) 사용하고, `Amazon Ember`에서 지원되지 않는 글리프에는 `Noto Sans`을(를) 사용하는 결합 글꼴입니다.

Open Data Visualization Dark

Open Data 시각화 다크




맵 스타일 이름: `VectorOpenDataVisualizationDark`

이 스타일은 차분한 색상의 어두운 테마 스타일로, 기능이 더 적어 오버레이된 데이터를 이해하는 데 도움이 됩니다.

이 베이스맵은 () 기여자들로부터 OpenStreetMap 컴파일된 OSM 데이라이트 맵 배포를 기반으로 합니다. OSM OSM커뮤니티에는 매일 500,000개 이상의 기능을 업데이트하는 180만 명 이상의 기여자가 포함되어 있습니다. Amazon Location Service는 이러한 편집 내용을 정기적으로 통합합니다.

글꼴

Amazon Location은 `GetMapGlyphs`을(를) 사용하여 글꼴을 제공합니다. 이 맵에 사용할 수 있는 글꼴 스택은 다음과 같습니다.

- Amazon Ember Bold, Noto Sans Bold
- Amazon Ember Condensed RC Bold, Noto Sans Bold
- Amazon Ember Condensed RC Regular, Noto Sans Regular
- Amazon Ember Medium, Noto Sans Medium

- Amazon Ember Regular Italic, Noto Sans Italic
- Amazon Ember Regular, Noto Sans Regular
- Amazon Ember Regular, Noto Sans Regular, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold, Noto Sans Bold, Noto Sans Arabic Condensed Bold
- Amazon Ember Bold, Noto Sans Bold, Noto Sans Arabic Bold
- Amazon Ember Regular Italic, Noto Sans Italic, Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular, Noto Sans Regular, Noto Sans Arabic Condensed Regular
- Amazon Ember Medium, Noto Sans Medium, Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataVisualizationDark`에서 사용되는 글꼴은 대부분의 글리프에 `Amazon Ember`을(를) 사용하고, `Amazon Ember`에서 지원되지 않는 글리프에는 `Noto Sans`을(를) 사용하는 결합 글꼴입니다.

## 적용 범위: Open Data

Open Data에는 Amazon Location Service 맵 리소스를 사용하여 렌더링하기 위한 글로벌 커버리지가 있는 맵이 포함되어 있습니다.

> **Note**
>
> Open Data는 Amazon Location Service 맵 리소스에만 사용할 수 있습니다. Open Data를 데이터 공급자로 사용하여 지오코딩, 역지오코딩 및 검색을 위한 쿼리를 지원하거나 경로 계산을 위한 쿼리를 지원할 수 없습니다.

## 이용 약관 및 데이터 저작자 표시: Open Data

오픈 데이터를 사용하기 전에 오픈 데이터에 적용되는 라이선스 조건을 포함하여 모든 관련 법적 요구 사항을 준수할 수 있는지 확인하십시오. AWS

AWS 요구 사항에 대한 자세한 내용은 AWS서비스 약관을 참조하십시오.

Open Data 저작자 표시 가이드라인에 대한 자세한 내용은 OpenStreetMap 의 저작권 및 라이선스 및 OpenStreetMap 라이선스/저작자 표시 가이드라인을 참조하십시오.

### Open Data에 대한 오류 신고 및 기여

OpenStreetMap (OSM) 및 Natural Earth는 커뮤니티 중심의 오픈 데이터 프로젝트입니다. 데이터에 문제가 발생하는 경우 오류를 보고하거나 수정 또는 제안을 직접 제공할 수 있습니다.

- 에서 OSM 오류를 보고하거나 제안 사항을 제공하려면 지도에 메모를 작성할 수 있습니다. 이는 기여자가 맵을 수정하는 데 도움이 되는 맵에 대한 설명입니다. OpenStreetMap 웹사이트를 통해 메모를 생성합니다. 메모에 대한 자세한 내용은 OpenStreetMap Wiki의 메모를 참조하십시오.
- 위치 추가 및 오류 수정을 포함하여 직접 기여하는 방법에 대한 자세한 내용은 OpenStreetMap Wiki의 Contribute 맵 데이터를 참조하십시오. OpenStreetMap
- Natural Earth의 데이터에 대한 수정 요청을 제출하려면 Natural Earth 웹사이트를 통해 문제를 제출하면 됩니다.

> **Note**
>
> 에서 오류를 빠르게 수정할 OpenStreetMap 수 있지만 Open Data 공급자가 사용하는 OSM 데이터의 데이라이트 맵 배포에 수정 내용이 표시되기까지는 시간이 걸릴 수 있습니다. 데이라이트 맵 배포 웹 사이트에서는 프로세스에 대한 자세한 정보를 제공합니다. 또한 Amazon Location Service는 Amazon Location Service에서 사용되는 맵 데이터를 대략 한 달에 한 번씩 업데이트합니다.

## 데이터 공급자별 기능

이 섹션에서는 Amazon Location Service에서 사용할 수 있는 기능을 데이터 공급자별로 분류하여 설명합니다.

다음 표에서는 기능에 대한 종합적인 개요를 제공합니다.

| 데이터 공급자 | 지리적 범위 | 기능 적용 범위 | AWS 리전 |
| --- | --- | --- | --- |
| Esri | 전 세계 | 맵, 장소, 경로 | Amazon Location을 사용할 수 있는 모든 리전. |

| 데이터 공급자 | 지리적 범위 | 기능 적용 범위 | AWS 리전 |
|---|---|---|---|
| Grab | 동남아시아 | 맵, 장소, 경로 | 아시아 태평양(싱가포르), `ap-southeast-1` 만 해당. |
| HERE | 전 세계 | 맵, 장소, 경로 | Amazon Location을 사용할 수 있는 모든 리전. |
| Open Data(오픈 데이터) | 전 세계 | 맵 | Amazon Location을 사용할 수 있는 모든 리전. |

다음 탭은 각 기능 영역의 세부 정보를 보여줍니다.

Map Features

다음 표는 데이터 제공자별 맵 기능을 보여줍니다. 맵 개념에 대한 자세한 내용은 맵 섹션을 참조하세요.

| 데이터 공급자 | 지원되는 맵 유형 | 벡터 확대/축소 수준 | 래스터 확대/축소 수준 |
|---|---|---|---|
| Esri | 벡터<br><br>래스터(이미지)<br><br>자세한 내용은 Esri 맵 스타일 섹션을 참조하세요. | 0-15 | 0~23 |
| Grab | 벡터<br><br>(동남아시아만 해당)<br><br>자세한 내용은 Grab 맵 스타일 섹션을 참조하세요. | 0~14 | 없음 |

| 데이터 공급자 | 지원되는 맵 유형 | 벡터 확대/축소 수준 | 래스터 확대/축소 수준 |
|---|---|---|---|
| HERE | 벡터<br><br>래스터(이미지)<br><br>하이브리드<br><br>자세한 내용은 HERE 맵 스타일 섹션을 참조하세요. | 1-17 | 0-19 |
| Open Data(오픈 데이터) | 벡터<br><br>자세한 내용은 Open Data 맵 스타일 섹션을 참조하세요. | 0-15 | 없음 |

> **Note**
>
> 확대/축소 수준은 각 공급자의 APIs 설정에 정의된 최대 및 최소 설정을 나타냅니다. 맵의 영역마다 최대값이 다를 수 있습니다. 예를 들어 바다 타일의 경우, 주요 도시 지역보다 세부 확대/축소 수준이 낮을 수 있습니다.
>
> MapLibre (및 기타 지도 렌더링 엔진) 를 사용하면 최소 및 최대 확대 수준을 설정할 수 있으며 특정 지역의 확대/축소 수준도 데이터 제공자의 확대/축소 수준을 적용하므로 이러한 불일치를 처리하기 위한 코드를 작성할 필요가 없습니다.

### Places and Search

다음 표에서는 데이터 공급자별 장소 및 검색 기능을 보여 줍니다. 장소 개념에 대한 자세한 내용은 장소 검색 섹션을 참조하세요.

| 데이터 공급자 | 지오코딩 | 역방향 지오코딩 | 자동 완성 | GetPlace |
|---|---|---|---|---|
| Esri | 다음을 제외한 모든 기능: | 다음을 제외한 모든 기능: | 모든 기능 | 모든 기능 |

| 데이터 공급자 | 지오코딩 | 역방향 지오코딩 | 자동 완성 | GetPlace |
|---|---|---|---|---|
| | PlaceId | TimeZone<br>PlaceId | | |
| Grab | 다음을 제외한 모든 기능:<br>단위 유형<br>카테고리는 지원되지 않음 | 모든 기능 | 모든 기능 | 다음을 제외한 모든 기능:<br>단위 유형<br>SubMunicipality |
| HERE | 다음을 제외한 모든 기능:<br>단위 숫자<br>단위 유형<br>관련성<br>필터링에 대한 추가 제한 사항 | 모든 기능 | 모든 기능 | 다음을 제외한 모든 기능:<br>단위 숫자<br>단위 유형<br>SubMunicipality |
| Open Data(오픈 데이터) | 지원되지 않음 | 지원되지 않음 | 지원되지 않음 | 지원 항목:<br>SubMunicipality |

## Route features

다음 표는 데이터 공급자별 경로 기능을 보여 줍니다. 경로에 대한 자세한 내용은 경로 개념을 참조하세요. 경로 매트릭스 제한 사항에 대한 자세한 설명은 출발 위치 및 도착 위치에 대한 제한 섹션을 참조하세요.

| 데이터 공급자 | 이동 모드 | 경로 계산 | 경로 매트릭스 |
| --- | --- | --- | --- |
| Esri | 자동차, 트럭, 도보 | 출발지와 목적지가 서로 400km 이내에 있어야 합니다. 총 이동 시간은 400분을 초과할 수 없습니다.<br><br>ArrivalTime 지원되지 않습니다. | 출발 및 목적지 위치는 최대 10개입니다.<br><br>한국에서는 지원되지 않습니다.<br><br>모든 출발 및 목적지 쌍은 서로 400km 이내에 있어야 합니다. |
| Grab | 자동차, 오토바이.<br><br>일부 도시에서 도보 및 자전거. | 거리 제한 없음. | 출발 및 목적지 위치는 최대 350개입니다. |
| HERE | 자동차, 트럭, 도보 | 거리 제한 없음. 출발 및 목적지 위치를 중심으로 원을 벗어난 10km 이상 경로는 계산되지 않습니다. | 출발 및 목적지 위치는 최대 350개입니다.<br><br>모든 출발 및 목적지 위치는 180km의 원 이내여야 합니다.<br><br>추가 제한 사항을 제외하고 더 긴 경로도 지원됩니다. |
| Open Data(오픈 데이터) | 지원되지 않음 | 지원되지 않음 | 지원되지 않음 |

## 데이터 공급자의 이용 약관 및 데이터 저작자 표시

데이터 공급자를 사용하기 전에 공급자 사용에 적용되는 라이선스 약관을 포함하여 해당하는 모든 법적 요구 사항을 준수할 수 있는지 확인하세요.

AWS 요구 사항에 대한 자세한 내용은 AWS서비스 약관을 참조하십시오.

애플리케이션 또는 설명서에 대한 Amazon Location 리소스와 함께 데이터 공급자를 사용하는 경우 사용하는 각 데이터 공급자에 대한 저작자 표시를 제공해야 합니다.

각 데이터 공급자의 규정 준수 및 저작자 표시에 대한 자세한 내용은 다음 항목을 참조하세요.

- Esri – 이용 약관 및 데이터 저작자 표시: Esri
- Grab – 이용 약관 및 데이터 저작자 표시: Grab
- HERE – 이용 약관 및 데이터 속성: HERE
- Open data – 이용 약관 및 데이터 저작자 표시: Open Data

# Amazon Location 리전 및 엔드포인트

Amazon 로케이션은 다음 AWS 지역에서 사용할 수 있습니다.

## 리전

| 리전 이름 | 지역 | 엔드포인트 | 프로토콜 | |
|---|---|---|---|---|
| 미국 동부(오하이오) | us-east-2 | geo.us-east-2.amazonaws.com | HTTPS | |
| 미국 동부(버지니아 북부) | us-east-1 | geo.us-east-1.amazonaws.com | HTTPS | |
| 미국 서부(오레곤) | us-west-2 | geo.us-west-2.amazonaws.com | HTTPS | |
| 아시아 태평양(뭄바이) | ap-south-1 | geo.ap-south-1.amazonaws.com | HTTPS | |
| 아시아 태평양(싱가포르) | ap-southeast-1 | geo.ap-southeast-1.amazonaws.com | HTTPS | |

| 리전 이름 | 지역 | 엔드포인트 | 프로토콜 |
|---|---|---|---|
| 아시아 태평양(시드니) | ap-southeast-2 | geo.ap-southeast-2.amazonaws.com | HTTPS |
| 아시아 태평양(도쿄) | ap-northeast-1 | geo.ap-northeast-1.amazonaws.com | HTTPS |
| 캐나다(중부) | ca-central-1 | geo.ca-central-1.amazonaws.com | HTTPS |
| 유럽(프랑크푸르트) | eu-central-1 | geo.eu-central-1.amazonaws.com | HTTPS |
| 유럽(아일랜드) | eu-west-1 | geo.eu-west-1.amazonaws.com | HTTPS |
| 유럽(런던) | eu-west-2 | geo.eu-west-2.amazonaws.com | HTTPS |
| 유럽(스톡홀름) | eu-north-1 | geo.eu-north-1.amazonaws.com | HTTPS |
| 남아메리카(상파울루) | sa-east-1 | geo.sa-east-1.amazonaws.com | HTTPS |
| AWS GovCloud (미국 서부) | us-gov-west-1 | geo.us-gov-west-1.amazonaws.com<br>geo-fips.us-gov-west-1.amazonaws.com | HTTPS<br>HTTPS |

> **Note**
>
> 이 표의 엔드포인트 사용 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.

## 엔드포인트

Amazon Location 리전 엔드포인트의 일반 구문은 다음과 같습니다.

```
protocol://service-code.geo.region-code.amazonaws.com
```

이 구문 내에서 Amazon Location은 다음 서비스 코드를 사용합니다.

| Service | 서비스 코드 |
| --- | --- |
| Amazon Location Maps | 맵 |
| Amazon Location Places | 장소 |
| Amazon Location Geofences | 지오펜싱 |
| Amazon Location Trackers | 추적 |
| Amazon Location Routes | 경로 |

예를 들어 미국 동부 (버지니아 북부) 용 Amazon 위치 지도의 지역별 엔드포인트는 https://입니다.*maps*.geo.*us-east-1*.amazonaws.com.

## API운영 엔드포인트

Amazon Location Service 컨트롤 플레인 엔드포인트의 구문은 다음과 같습니다.

```
protocol://cp.service-code.geo.region-code.amazonaws.com
```

Amazon Location Service의 컨트롤 플레인 작업은 다음과 같습니다.

| Service | 엔드포인트 | API오퍼레이션 |
| --- | --- | --- |
| Amazon Location Maps | https://cp.maps.geo.*region*.amazonaws.com | CreateMap<br>DeleteMap<br>DescribeMap |

| Service | 엔드포인트 | API오퍼레이션 |
| --- | --- | --- |
| | | ListMaps<br>UpdateMap |
| Amazon Location Places | https://cp.places.geo.*region*.amazonaws.com | CreatePlaceIndex<br>DeletePlaceIndex<br>DescribePlaceIndex<br>ListPlaceIndexes<br>UpdatePlaceIndex |
| Amazon Location Geofences | https://cp.geofencing.geo.*region*.amazonaws.com | CreateGeofenceCollection<br>DeleteGeofenceCollection<br>DescribeGeofenceCollection<br>ListGeofenceCollections<br>UpdateGeofenceCollection |
| Amazon Location Trackers | https://cp.tracking.geo.*region*.amazonaws.com | CreateTracker<br>DeleteTracker<br>DescribeTracker<br>UpdateTracker<br>ListTrackers<br>AssociateTrackerConsumer<br>DisassociateTrackerConsumer<br>ListTrackerConsumers |

| Service | 엔드포인트 | API오퍼레이션 |
| --- | --- | --- |
| Amazon Location Routes | https://cp.routes.geo.*region*.amazonaws.com | CreateRouteCalculator<br>DeleteRouteCalculator<br>DescribeRouteCalculator<br>ListRouteCalculators<br>UpdateRouteCalculator |
| Amazon Location Metadata | https://cp.metadata.geo.*region*.amazonaws.com | CreateKey<br>DeleteKey<br>DescribeKey<br>ListKeys<br>UpdateKey |

# Amazon Location Service Quotas

이 항목에서는 Amazon Location Service의 요율 한도 및 할당량에 대한 요약을 제공합니다.

> **Note**
>
> 더 높은 할당량이 필요한 경우, Service Quotas 콘솔을 사용하면 조정 가능한 할당량에 대한 할당량 증가를 요청할 수 있습니다. 할당량 증가를 요청할 때는 할당량 증가가 필요한 지역을 선택하세요. 대부분의 할당량은 지역별로 다르기 때문입니다. AWS

Service Quotas는 계정 및 AWS 지역별로 보유할 수 있는 최대 리소스 수입니다. AWS Amazon Location Service는 Service Quotas 를을 초과하는 추가 요청을 거부합니다.

요금 한도 (요율로 시작하는 할당량은... ) 는 초당 최대 요청 수로, 각 작업에 대해 정의된 초당 요청 수 중 초당 최대 요청 수의 80% 에 해당하는 버스트 비율을 가집니다. API Amazon Location Service는 작업 속도 제한을 초과하는 요청을 제한합니다.

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| API계정당 주요 리소스 | 지원되는 각 리전: 500개 | 아니요 | 계정당 보유할 수 있는 API 주요 리소스 (활성 또는 만료된 리소스) 의 최대 수 |
| 계정당 지오펜스 컬렉션 리소스 | 지원되는 각 리전: 1,500 | 예 | 계정당 생성할 수 있는 최대 지오펜스 컬렉션 리소스 수입니다. |
| 지오펜스 컬렉션당 지오펜스 | 지원되는 각 리전: 50,000 | 아니요 | 지오펜스 컬렉션당 생성할 수 있는 최대 지오펜스 수입니다. |
| 계정당 맵 리소스 | 지원되는 각 리전: 40 | 예 | 계정당 생성할 수 있는 최대 맵 리소스 수입니다. |
| 계정당 장소 색인 리소스 | 지원되는 각 리전: 40 | 예 | 계정당 생성할 수 있는 최대 장소 색인 리소스 수입니다. |
| 요청 비율 AssociateTrackerConsumer API | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 AssociateTrackerConsumer 요청 수입니다. 추가 요청은 제한됩니다. |
| BatchDeleteDevicePositionHistory API 요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 BatchDeleteDevicePositionHistory 요청 수입니다. 추가 요청은 제한됩니다. |
| BatchDeleteGeofence API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 BatchDeleteGeofence 요 |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| | | | 청 수입니다. 추가 요청은 제한됩니다. |
| BatchEvaluateGeofences API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 BatchEvaluateGeofences 요청 수입니다. 추가 요청은 제한됩니다. |
| BatchGetDevicePosition API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 BatchGetDevicePosition 요청 수입니다. 추가 요청은 제한됩니다. |
| BatchPutGeofence API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 BatchPutGeofence 요청 수입니다. 추가 요청은 제한됩니다. |
| BatchUpdateDevicePosition API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 BatchUpdateDevicePosition 요청 수입니다. 추가 요청은 제한됩니다. |
| CalculateRoute API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 CalculateRoute 요청 수입니다. 추가 요청은 제한됩니다. |
| CalculateRouteMatrix API요청 비율 | 지원되는 각 리전: 초당 5개 | 예 | 초당 만들 수 있는 최대 CalculateRouteMatrix 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| CreateGeofenceCollection API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 CreateGeofenceCollection 요청 수입니다. 추가 요청은 제한됩니다. |
| CreateKey API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 CreateKey 요청 수입니다. 추가 요청은 제한됩니다. |
| CreateMap API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 CreateMap 요청 수입니다. 추가 요청은 제한됩니다. |
| CreatePlaceIndex API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 CreatePlaceIndex 요청 수입니다. 추가 요청은 제한됩니다. |
| CreateRouteCalculator API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 CreateRouteCalculator 요청 수입니다. 추가 요청은 제한됩니다. |
| CreateTracker API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 CreateTracker 요청 수입니다. 추가 요청은 제한됩니다. |
| DeleteGeofenceCollection API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DeleteGeofenceCollection 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| DeleteKey API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DeleteKey 요청 수입니다. 추가 요청은 제한됩니다. |
| DeleteMap API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DeleteMap 요청 수입니다. 추가 요청은 제한됩니다. |
| DeletePlaceIndex API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DeletePlaceIndex 요청 수입니다. 추가 요청은 제한됩니다. |
| DeleteRouteCalculator API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DeleteRouteCalculator 요청 수입니다. 추가 요청은 제한됩니다. |
| DeleteTracker API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DeleteTracker 요청 수입니다. 추가 요청은 제한됩니다. |
| DescribeGeofenceCollection API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DescribeGeofenceCollection 요청 수입니다. 추가 요청은 제한됩니다. |
| DescribeKey API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DescribeKey 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| DescribeMap API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DescribeMap 요청 수입니다. 추가 요청은 제한됩니다. |
| DescribePlaceIndex API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DescribePlaceIndex 요청 수입니다. 추가 요청은 제한됩니다. |
| DescribeRouteCalculator API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DescribeRouteCalculator 요청 수입니다. 추가 요청은 제한됩니다. |
| DescribeTracker API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DescribeTracker 요청 수입니다. 추가 요청은 제한됩니다. |
| DisassociateTrackerConsumer API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 DisassociateTrackerConsumer 요청 수입니다. 추가 요청은 제한됩니다. |
| ForecastGeofenceEvents API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 ForecastGeofenceEvents 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| GetDevicePosition API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 GetDevicePosition 요청 수입니다. 추가 요청은 제한됩니다. |
| GetDevicePositionHistory API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 GetDevicePositionHistory 요청 수입니다. 추가 요청은 제한됩니다. |
| GetGeofence API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 GetGeofence 요청 수입니다. 추가 요청은 제한됩니다. |
| GetMapGlyphs API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 GetMapGlyphs 요청 수입니다. 추가 요청은 제한됩니다. |
| GetMapSprites API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 GetMapSprites 요청 수입니다. 추가 요청은 제한됩니다. |
| GetMapStyleDescriptor API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 GetMapStyleDescriptor 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| GetMapTile API요청 비율 | 지원되는 각 리전: 초당 500 | 예 | 초당 만들 수 있는 최대 GetMapTile 요청 수입니다. 추가 요청은 제한됩니다. |
| GetPlace API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 GetPlace 요청 수입니다. 추가 요청은 제한됩니다. |
| ListDevicePositions API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 ListDevicePositions 요청 수입니다. 추가 요청은 제한됩니다. |
| ListGeofenceCollections API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListGeofenceCollections 요청 수입니다. 추가 요청은 제한됩니다. |
| ListGeofences API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 ListGeofences 요청 수입니다. 추가 요청은 제한됩니다. |
| ListKeys API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListKeys 요청 수입니다. 추가 요청은 제한됩니다. |
| ListMaps API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListMaps 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| ListPlaceIndexes API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListPlaceIndexes 요청 수입니다. 추가 요청은 제한됩니다. |
| ListRouteCalculators API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListRouteCalculators 요청 수입니다. 추가 요청은 제한됩니다. |
| ListTagsForResource API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListTagsForResource 요청 수입니다. 추가 요청은 제한됩니다. |
| ListTrackerConsumers API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListTrackerConsumers 요청 수입니다. 추가 요청은 제한됩니다. |
| ListTrackers API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 ListTrackers 요청 수입니다. 추가 요청은 제한됩니다. |
| PutGeofence API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 PutGeofence 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| SearchPlaceIndexForPosition API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 SearchPlaceIndexForPosition 요청 수입니다. 추가 요청은 제한됩니다. |
| SearchPlaceIndexForSuggestions API 요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 SearchPlaceIndexForSuggestions 요청 수입니다. 추가 요청은 제한됩니다. |
| SearchPlaceIndexForText API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 SearchPlaceIndexForText 요청 수입니다. 추가 요청은 제한됩니다. |
| TagResource API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 TagResource 요청 수입니다. 추가 요청은 제한됩니다. |
| UntagResource API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 UntagResource 요청 수입니다. 추가 요청은 제한됩니다. |
| UpdateGeofenceCollection API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 UpdateGeofenceCollection 요청 수입니다. 추가 요청은 제한됩니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| UpdateKey API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 UpdateKey 요청 수입니다. 추가 요청은 제한됩니다. |
| UpdateMap API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 UpdateMap 요청 수입니다. 추가 요청은 제한됩니다. |
| UpdatePlaceIndex API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 UpdatePlaceIndex 요청 수입니다. 추가 요청은 제한됩니다. |
| UpdateRouteCalculator API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 UpdateRouteCalculator 요청 수입니다. 추가 요청은 제한됩니다. |
| UpdateTracker API요청 비율 | 지원되는 각 리전: 초당 10개 | 예 | 초당 만들 수 있는 최대 UpdateTracker 요청 수입니다. 추가 요청은 제한됩니다. |
| VerifyDevicePosition API요청 비율 | 지원되는 각 리전: 초당 50 | 예 | 초당 만들 수 있는 최대 VerifyDevicePosition 요청 수입니다. 추가 요청은 제한됩니다. |
| 계정당 경로 계산기 리소스 | 지원되는 각 리전: 40 | 예 | 계정별로 생성할 수 있는 최대 경로 계산기 리소스 수입니다. |

| 명칭 | 기본값 | 조정 가능 | 설명 |
| --- | --- | --- | --- |
| 트래커당 트래커 소비자 | 지원되는 각 리전: 5 | 아니요 | 트래커 리소스와 연결할 수 있는 지오펜스 컬렉션의 최대 수입니다. |
| 계정당 트래커 리소스 | 지원되는 각 리전: 500 | 예 | 계정당 생성할 수 있는 최대 트래커 리소스 수입니다. |

> **Note**
>
> Cloudwatch를 사용하면 할당량 대비 사용량을 모니터링할 수 있습니다. 자세한 내용은 할당량 대비 사용량을 모니터링하는 CloudWatch 데 사용 섹션을 참조하세요.

## Amazon Location Service 할당량 관리

Amazon Location Service는 중앙 위치에서 할당량을 보고 관리할 수 있는 AWS 서비스인 서비스 할당량과 통합되어 있습니다. 자세한 내용은 Service Quotas 사용 설명서의 Service Quotas는 무엇입니까?를 참조하세요.

Service Quotas를 사용하면 모든 Amazon Location Service 할당량의 값을 쉽게 찾을 수 있습니다.

AWS Management Console

콘솔을 사용하여 Amazon Location Service의 할당량을 보려면

1. 에서 Service Quotas 콘솔을 엽니다. https://console.aws.amazon.com/servicequotas/
2. 탐색 창에서 AWS 서비스를 선택합니다.
3. AWS 서비스 목록에서 Amazon Location을 검색하여 선택합니다.

   Service quotas 목록에서 서비스 할당량 이름, 적용된 값(제공된 경우), AWS 기본 할당량 및 할당량 값 조정 가능 여부를 확인할 수 있습니다.
4. 설명 등 서비스 할당량에 대한 추가 정보를 보려면 할당량 이름을 선택합니다.

5. (선택 사항) 할당량 증가를 요청하려면 증가시킬 할당량을 선택하고 할당량 증가 요청(Request quota increase)을 선택한 다음 필요한 정보를 입력하거나 선택한 다음 요청(Request)을 선택합니다.

콘솔을 사용하여 서비스 할당량에 대한 추가 작업을 수행하려면 Service Quotas 사용 설명서를 참조하세요. 할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

AWS CLI

AWS CLI을(를) 사용하여 Amazon Location Service의 할당량을 보려면

다음 명령을 실행하여 기본 Amazon Location 할당량을 확인합니다.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code geo \
    --output table
```

를 사용하여 서비스 할당량을 자세히 알아보려면 Service Quotas 명령 참조서를 참조하십시오. AWS CLI AWS CLI 할당량 증가를 요청하려면 AWS CLI 명령 참조에서 `request-service-quota-increase` 명령을 참조하세요.

# Amazon Location Service를 사용하여 개발자로 시작하기

Amazon Location Service를 사용하면 백엔드 웹 서비스, 웹 애플리케이션, 모바일 애플리케이션 등 다양한 폼 팩터 및 시스템에서 앱에 지리적 관련 기능을 제공할 수 있습니다. SDK, 라이브러리, 샘플 코드 등 애플리케이션을 구축하는 데 도움이 되는 다양한 도구가 제공됩니다.

이 섹션에서는 Amazon Location Service를 처음 사용하는 데 도움이 되는 정보와 링크를 제공합니다. 특히 다음 주제는 가장 도움이 될 수 있는 정보를 제공합니다.

- 시나리오 및 사용 사례 – 개발 시나리오 목록 및 Amazon Location Service가 이러한 시나리오를 완료하는 데 어떤 도움을 줄 수 있는지 설명합니다.
- Amazon 위치 SDK 및 도구 – Amazon Location으로 프로그래밍할 때 도움이 되는 소프트웨어 개발 키트(SDK) 및 라이브러리.
- Amazon 위치 서비스 API 참조 — AWS SDK와 함께 제공되는 핵심 Amazon 위치 API에 대한 참조입니다.
- 코드 예제 – 이 섹션에서는 애플리케이션을 시작하거나 기존 애플리케이션에 기능을 추가하는 데 도움이 되는 샘플을 제공합니다.
- 빠른 시작 자습서 – 이 자습서에서는 첫 번째 응용 프로그램을 만드는 방법을 보여 줍니다. 이 자습서에는 웹 애플리케이션 또는 Android 기반 모바일 애플리케이션을 만드는 데 사용할 수 있는 여러 버전이 있습니다.
- Amazon Location Service 개념 – 본 가이드의 이 섹션에서는 지도, 장소 검색, 경로, 지오펜스 및 트래커에 대한 섹션을 포함하여 Amazon Location의 기본 개념을 설명합니다.
- Amplify – Amplify는 AWS 클라우드를 사용하여 웹 및 모바일 애플리케이션을 만드는 데 필요한 많은 기능을 캡슐화한 완벽한 솔루션입니다. Amplify를 이미 사용하고 있거나 사용하기로 선택한 경우에는, Amplify에 Amazon Location Service를 사용하는 지리 라이브러리가 내장되어 있으므로 이를 사용할 수 있습니다. Amplify Geo를 시작하려면 여기의 설명서를 참조하세요.

## 시나리오 및 사용 사례

Amazon Location Service는 AWS 클라우드에서 실행되는 서비스입니다. 클라우드에 있는 자체 Amazon EC2 인스턴스에서 이를 호출할 수도 있지만, 대부분의 매핑 애플리케이션은 디바이스 또는 디바이스와 클라우드의 조합에서 실행됩니다. 다음은 몇 가지 일반적인 시나리오와 이러한 시나리오를 개발하는 방법을 열거한 것입니다.

- 차량 내 운전자의 경로를 최적화하는 데 도움이 되는 백엔드 애플리케이션입니다.

Amazon Location Service를 사용하여 플릿의 경로 최적화 프로그램에 대한 입력으로 경로 행렬을 계산하는 Amazon EC2에서 실행되는 애플리케이션을 작성할 수 있습니다. AWS 클라우드 AWS SDK를 사용하여 Amazon Location을 호출할 수 있습니다.

- 고객이 내 비즈니스 위치를 찾을 수 있게 해주는 웹 애플리케이션입니다.

  위치 기반 애플리케이션을 포함하여 Amazon EC2 인스턴스에서 실행되는 웹 사이트를 만들 수 있습니다. AWS SDK JavaScript form을 사용하여 장소 검색을 사용하여 위치를 검색하고 를 사용하여 결과를 지도에 표시하는 웹 애플리케이션을 개발할 수 있습니다. MapLibre Amazon Location SDK를 사용하여 위치 관련 프로그래밍을 더 쉽게 수행할 수 있습니다.

- 기존 iOS 또는 Android 애플리케이션에 위치 기능을 추가합니다.

  Swift (iOS) 또는 AWS Kotlin (Android) 용 SDK를 사용하여 Amazon Location을 호출하여 애플리케이션에 장소 검색 및 지도 기능을 추가할 수 있습니다. 지도를 렌더링하는 데 사용합니다 MapLibre . 다른 언어를 위한 추가 AWS SDK가 있습니다.

- 자산(기기 또는 차량)을 추적하고, 자산이 정의된 영역에 들어오거나 나가는 경우 업데이트를 받을 수 있습니다.

  기기를 추적하는 애플리케이션은 여러 부분으로 구성되어 있습니다.

  - 추적 중인 각 기기에는 이를 추적하기 위한 트래커 리소스가 생성되어 있어야 합니다. 예컨대 MQTT를 사용하여 Amazon Location Service에 위치 업데이트를 전송해야 합니다.
  - 지오펜스를 생성하여 자산의 진입 및 퇴장 이벤트를 받을 영역을 정의할 수 있습니다.
  - 자산이 지오펜스 영역에 들어오거나 나가는 경우 Amazon EC2를 사용하거나 이벤트에 AWS Lambda응답할 수 있습니다.
  - 이를 기반으로 웹 또는 디바이스 애플리케이션을 생성하여 자산 위치를 추적하고 지도에 표시할 수 있습니다.

다음 섹션에서는 Amazon Location Service의 각 측면에서 사용할 수 있는 도구 및 라이브러리에 대한 세부 정보를 제공합니다.

# Amazon Location Service를 사용하기 위한 SDK 및 도구

Amazon Location Service를 사용하는 데 도움이 되는 몇 가지 도구가 있습니다.

- AWS SDK — AWS 소프트웨어 개발 키트 (SDK) 는 널리 사용되는 여러 프로그래밍 언어로 제공되며, 선호하는 언어로 애플리케이션을 쉽게 빌드할 수 있도록 API, 코드 예제 및 설명서를 제공합니

다. AWS SDK에는 지도, 장소 검색, 경로, 지오펜스, 트래커에 대한 액세스를 비롯한 핵심 Amazon Location API 및 기능이 포함되어 있습니다. 다양한 애플리케이션 및 언어에 대해 Amazon Location Service와 함께 사용할 수 있는 SDK에 대해 자세히 알아보려면 언어별 SDK을 참조하세요.

- MapLibre— Amazon Location Service에서는 MapLibre렌더링 엔진을 사용하여 맵을 렌더링할 것을 권장합니다. MapLibre 웹 또는 모바일 애플리케이션에서 지도를 표시하기 위한 엔진입니다. MapLibre 또한 플러그인 모델이 있으며 일부 언어 및 플랫폼에서 검색 및 경로를 위한 사용자 인터페이스를 지원합니다. 사용 MapLibre 및 제공되는 기능에 대한 자세한 내용은 을 참조하십시오MapLibre.
- Amazon 위치 SDK – Amazon 위치 SDK는 Amazon Location Service를 사용하여 애플리케이션을 더 쉽게 개발할 수 있게 해주는 오픈 소스 라이브러리 세트입니다. 라이브러리는 모바일 및 웹 애플리케이션에 대한 인증, 모바일 애플리케이션의 위치 추적, Amazon Location 데이터 유형과 GeoJSON 간의 변환을 지원하는 기능과 SDK v3용 AWS Amazon Location 클라이언트의 호스팅된 패키지를 제공합니다. Amazon Location Service에 대한 자세한 내용은 Amazon Location SDK을 참조하세요.

## 언어별 SDK

다음 표는 웹, 모바일 또는 백엔드 애플리케이션 등 애플리케이션 유형별로 언어 및 프레임워크의 AWS SDK 및 MapLibre 버전 정보를 제공합니다.

> **SDK 버전**
>
> 프로젝트에 사용하는 SDK의 최신 빌드와 기타 AWS SDK를 사용하고 SDK를 최신 상태로 유지하는 것이 좋습니다. AWS SDK는 최신 특징과 기능, 보안 업데이트를 제공합니다. 예를 들어 AWS SDK의 최신 빌드를 찾으려면 AWS SDK의 브라우저 설치 주제 설명서를 참조하십시오. JavaScript JavaScript

Web frontend

웹 프런트엔드 애플리케이션 개발에 사용할 수 있는 AWS SDK 및 MapLibre 버전은 다음과 같습니다.

| 언어/프레임워크 | AWS SDK | 렌더링 프레임워크 |
| --- | --- | --- |
| 완전 지원 | | |

| 언어/프레임워크 | AWS SDK | 렌더링 프레임워크 |
| --- | --- | --- |
| JavaScript | https://aws.amazon.com//sdk-for-javascript | https://maplibre.org/projects/maplibre-gl-js/ |
| ReactJS | https://aws.amazon.com/sdk-for-javascript/ | https://github.com/maplibre/maplibre-react-native |
| TypeScript | https://aws.amazon.com/sdk-for-javascript/ | https://maplibre.org/projects/maplibre-gl-js/ |
| 일부 지원 | | |
| Flutter | https://docs.amplify.aws/start/q/integration/flutter/<br><br>Flutter는 아직 완전히 지원되지는 않지만 AWS Amplify를 통한 지원은 제한적입니다. | https://github.com/maplibre/flutter-maplibre-gl<br><br>MapLibre Flutter 라이브러리는 실험용으로 간주됩니다. |
| Node.js | https://aws.amazon.com//sdk-for-javascript | Node.js 는 MapLibre 지원되지 않습니다. |
| PHP | https://aws.amazon.com/sdk-for-php/ | PHP에 대한 MapLibre 지원은 없습니다. |

Mobile frontend

모바일 프런트엔드 애플리케이션 개발에 사용할 수 있는 AWS SDK 및 MapLibre 버전은 다음과 같습니다.

| 언어/프레임워크 | AWS SDK | 렌더링 프레임워크 |
| --- | --- | --- |
| 완전 지원 | | |
| Java | https://aws.amazon.com//sdk-for-java | https://maplibre.org/projects/maplibre-native/ |

| 언어/프레임워크 | AWS SDK | 렌더링 프레임워크 |
| --- | --- | --- |
| Kotlin | https://aws.amazon.com/sdk-for-kotlin/<br><br>안드로이드용 Amazon Location Service 모바일 인증 SDK: amazon-location-mobile-auth https://github.com/aws-geospatial/ -sdk-android<br><br>안드로이드용 Amazon Location Service 모바일 트래킹 SDK: amazon-location-mobile-tracking https://github.com/aws-geospatial/ -sdk-android | https://maplibre.org/projects/maplibre-native/<br><br>자바 기반과 마찬가지로 사용자 지정 바인딩이 필요합니다. MapLibre |
| ObjectiveC | https://github.com/aws-amplify/ aws-sdk-ios | https://maplibre.org/projects/maplibre-native/ |
| ReactNative | https://aws.amazon.com/sdk-for-javascript/ | https://github.com/maplibre/maplibre-react-native |
| Swift | https://aws.amazon.com/sdk-for-swift/<br><br>iOS용 Amazon Location Service 모바일 인증 SDK: amazon-location-mobile-auth https://github.com/aws-geospatial/ -sdk-ios<br><br>iOS용 Amazon Location Service 모바일 트래킹 SDK: amazon-location-mobile-tracking https://github.com/aws-geospatial/ -sdk-ios | https://maplibre.org/projects/maplibre-native/ |

| 언어/프레임워크 | AWS SDK | 렌더링 프레임워크 |
| --- | --- | --- |
| 일부 지원 | | |
| Flutter | https://docs.amplify.aws/start/q/integration/flutter/<br><br>Flutter는 아직 완전히 지원되지는 않지만 AWS Amplify를 통한 지원은 제한적입니다. | https://github.com/maplibre/flutter-maplibre-gl<br><br>MapLibre Flutter 라이브러리는 실험용으로 간주됩니다. |

Backend application

백엔드 애플리케이션 개발에 사용할 수 있는 AWS SDK는 다음과 같습니다. MapLibre 일반적으로 백엔드 애플리케이션에는 맵 렌더링이 필요하지 않기 때문에 여기에 나열되어 있지 않습니다.

| 언어 | AWS SDK |
| --- | --- |
| .NET | https://aws.amazon.com//sdk-for-net |
| C++ | https://aws.amazon.com/sdk-for-cpp/ |
| Go | https://aws.amazon.com/sdk-for-go/ |
| Java | https://aws.amazon.com/sdk-for-java/ |
| JavaScript | https://aws.amazon.com/sdk-for-javascript/ |
| Node.js | https://aws.amazon.com/sdk-for-javascript/ |
| TypeScript | https://aws.amazon.com/sdk-for-javascript/ |
| Kotlin | https://aws.amazon.com/sdk-for-kotlin/ |
| PHP | https://aws.amazon.com/sdk-for-php/ |
| Python | https://aws.amazon.com/sdk-for-python/ |
| Ruby | https://aws.amazon.com/sdk-for-ruby/ |

| 언어 | AWS SDK |
| --- | --- |
| Rust | https://aws.amazon.com/sdk-for-rust/<br><br>Rust용 AWS SDK는 개발자 프리뷰 단계에 있습니다. |

## Amazon 로케이션에서 MapLibre 도구 및 라이브러리 사용

Amazon Location을 사용하여 대화형 애플리케이션을 만드는 데 필요한 중요한 도구 중 하나는 다음과 같습니다 MapLibre. MapLibre주로 웹 또는 모바일 애플리케이션에서 지도를 표시하기 위한 렌더링 엔진입니다. 하지만 플러그인에 대한 지원도 포함하며 Amazon Location의 다른 측면과 함께 작업할 수 있는 기능도 제공합니다. 다음은 작업하려는 위치 지역을 기반으로 사용할 수 있는 도구를 설명합니다.

> **Note**
>
> Amazon Location의 모든 측면을 사용하려면 사용하려는 언어의AWS SDK를 설치하세요.

- 맵

  애플리케이션에 지도를 표시하려면 Amazon Location에서 제공하는 데이터를 사용하고 화면에 그릴 맵 렌더링 엔진이 필요합니다. 맵 렌더링 엔진은 맵을 이동 및 확대하거나 마커, 푸시핀 및 기타 주석을 맵에 추가하는 기능도 제공합니다.

  Amazon Location Service에서는 MapLibre렌더링 엔진을 사용하여 맵을 렌더링할 것을 권장합니다. MapLibre GL JS는 지도를 표시하는 엔진이며 JavaScript, MapLibre 네이티브는 iOS 또는 안드로이드용 지도를 제공합니다.

  MapLibre 또한 핵심 기능을 확장할 수 있는 플러그인 에코시스템을 갖추고 있습니다. 자세한 내용은 https://maplibre.org/ maplibre-gl-js-docs /plugins/를 참조하십시오.
- 장소 검색

  검색 사용자 인터페이스를 더 간단하게 만들려면 웹용 MapLibre 지오코더를 사용할 수 있습니다 (Android 애플리케이션은 Android Places 플러그인을 사용할 수 있음).

Maplibre용 Amazon 로케이션 지오코더 라이브러리를 사용하면 애플리케이션에서 Amazon Location을 사용하는 프로세스를 간소화할 수 있습니다. `amazon-location-for-maplibre-gl-geocoder` JavaScript

- 경로

  지도에 경로를 표시하려면 길찾기를 사용하십시오. MapLibre

- 지오펜스 및 트래커

  MapLibre 지오펜스 및 추적을 위한 특정 렌더링이나 도구는 없지만 렌더링 기능과 플러그인을 사용하여 지오펜스와 추적된 장치를 지도에 표시할 수 있습니다.

  추적 대상 디바이스는 MQTT를 사용하거나 Amazon Location Service에 업데이트를 수동으로 전송할 수 있습니다. AWS Lambda을 사용하여 지오펜스 이벤트에 응답할 수 있습니다.

Amazon Location Service에 추가 기능을 제공하는 데 사용할 수 있는 많은 오픈 소스 라이브러리(예: 공간 분석 기능을 제공하는 Turf)가 있습니다.

많은 라이브러리는 개방형 표준 GeoJSON 형식의 데이터를 사용합니다. Amazon Location Service는 애플리케이션에서 JavaScript GeoJSON을 사용할 수 있도록 지원하는 라이브러리를 제공합니다. 자세한 정보는 다음 섹션(Amazon Location SDK 및 라이브러리)을 참조하세요.

## Amazon 위치 MapLibre 지오코더 플러그인

Amazon Location MapLibre 지오코더 플러그인은 라이브러리를 사용하여 지도 렌더링 및 지오코딩을 수행할 때 Amazon Location 기능을 JavaScript 애플리케이션에 쉽게 통합할 수 있도록 설계되었습니다. maplibre-gl-geocoder

설치

다음 명령을 사용하여 모듈과 함께 사용할 수 있도록 NPM에서 Amazon Location MapLibre 지오코더 플러그인을 설치할 수 있습니다.

```
npm install @aws/amazon-location-for-maplibre-gl-geocoder
```

스크립트를 사용하여 브라우저에서 직접 사용할 수 있도록 HTML 파일로 가져올 수 있습니다.

```
<script src="https://www.unpkg.com/@aws/amazon-location-for-maplibre-gl-
geocoder@1">/script<
```

### 모듈과 함께 사용

이 코드는 Amazon 위치 지오코딩 기능을 JavaScript 갖춘 Maplibre GL 맵을 설정합니다. Amazon Cognito 자격 증명 풀을 통한 인증을 사용하여 Amazon 위치 리소스에 액세스합니다. 맵은 지정된 스타일과 중앙 좌표로 렌더링되며 맵에서 장소를 검색할 수 있습니다.

```
// Import MapLibre GL JS
import maplibregl from "maplibre-gl";
// Import from the AWS JavaScript SDK V3
import { LocationClient } from "@aws-sdk/client-location";
// Import the utility functions
import { withIdentityPoolId } from "@aws/amazon-location-utilities-auth-helper";
// Import the AmazonLocationWithMaplibreGeocoder
import { buildAmazonLocationMaplibreGeocoder, AmazonLocationMaplibreGeocoder } from
 "@aws/amazon-location-for-maplibre-gl-geocoder"

const identityPoolId = "Identity Pool ID";
const mapName = "Map Name";
const region = "Region"; // region containing the Amazon Location resource
const placeIndex = "PlaceIndexName" // Name of your places resource in your AWS
 Account.

// Create an authentication helper instance using credentials from Amazon Cognito
const authHelper = await withIdentityPoolId("Identity Pool ID");

const client = new LocationClient({
  region: "Region", // Region containing Amazon Location resources
  ...authHelper.getLocationClientConfig(), // Configures the client to use
 credentials obtained via Amazon Cognito
});

// Render the map
const map = new maplibregl.Map({
    container: "map",
    center: [-123.115898, 49.295868],
    zoom: 10,
    style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor`,
    ...authHelper.getMapAuthenticationOptions(),
});

// Gets an instance of the AmazonLocationMaplibreGeocoder Object.
```

```
const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client,
 placeIndex, {enableAll: true});

// Now we can add the Geocoder to the map.
map.addControl(amazonLocationMaplibreGeocoder.getPlacesGeocoder());
```

### 브라우저에서의 사용

이 예시에서는 Amazon 로케이션 클라이언트를 사용하여 Amazon Cognito를 사용하여 인증하도록 요청합니다.

> **Note**
>
> 이 예시 중 일부는 Amazon 로케이션 클라이언트를 사용합니다. Amazon 로케이션 클라이언트는 JavaScript V3용AWS SDK를 기반으로 하며, HTML 파일에 참조된 스크립트를 통해 Amazon 로케이션을 호출할 수 있습니다.

HTML 파일에 다음을 포함하십시오.

```
< Import the Amazon Location With Maplibre Geocoder>
<script src="https://www.unpkg.com/@aws/amazon-location-with-maplibre-geocoder@1"></
script>
<Import the Amazon Location Client>
<script src="https://www.unpkg.com/@aws/amazon-location-client@1"></script>
<!Import the utility library>
<script src="https://www.unpkg.com/@aws/amazon-location-utilities-auth-helper@1"></
script>
```

JavaScript 파일에 다음을 포함하십시오.

```
const identityPoolId = "Identity Pool ID";
const mapName = "Map Name";
const region = "Region"; // region containing Amazon Location resource

// Create an authentication helper instance using credentials from Amazon Cognito
const authHelper = await
 amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

// Render the map
```

```
const map = new maplibregl.Map({
  container: "map",
  center: [-123.115898, 49.295868],
  zoom: 10,
  style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor`,
  ...authHelper.getMapAuthenticationOptions(),
});

// Initialize the AmazonLocationMaplibreGeocoder object
const amazonLocationMaplibreGeocoderObject =
 amazonLocationMaplibreGeocoder.buildAmazonLocationMaplibreGeocoder(client,
 placesName, {enableAll: true});

// Use the AmazonLocationWithMaplibreGeocoder object to add a geocoder to the map.
map.addControl(amazonLocationMaplibreGeocoderObject.getPlacesGeocoder());
```

Amazon Location MapLibre 지오코더 플러그인에서 사용되는 함수와 명령은 다음과 같습니다.

- **buildAmazonLocationMaplibreGeocoder**

  이 클래스는 다른 모든 AmazonLocationMaplibreGeocder 호출의 진입점이 되는 인스턴스를 생성합니다.

```
const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client,
 placesIndex, {enableAll: true});
```

- **getPlacesGeocoder**

  맵에 직접 추가할 수 있는 바로 사용할 수 있는 iControl 객체를 반환합니다.

```
const geocoder = getPlacesGeocoder();

// Initialize map
let map = await initializeMap();

// Add the geocoder to the map.
map.addControl(geocoder);
```

# Amazon Location SDK 및 라이브러리

Amazon Location SDK는 Amazon Location 애플리케이션을 개발하는 데 유용한 기능을 제공하는 오픈 소스 라이브러리 세트입니다. 다음과 같은 기능이 포함됩니다.

- Amazon 로케이션 클라이언트 — AWS SDK v3의 Amazon 로케이션 객체는 웹 개발에서 쉽게 사용할 수 있도록 번들 및 패키징됩니다.
- 인증 — 인증 유틸리티는 Amazon Location Service용 웹 페이지 JavaScript, iOS 또는 Android 애플리케이션을 구축할 때 Amazon Cognito 또는 API 키를 사용하여 인증을 간소화합니다.
- 추적 — 모바일 추적 SDK는 iOS와 안드로이드에서 사용할 수 있습니다. 이 SDK를 사용하면 모바일 애플리케이션이 Amazon 위치 추적기와 더 쉽게 상호 작용할 수 있습니다.
- Amazon Location GeoJSON 함수 — GeoJSON 변환 유틸리티를 사용하면 업계 표준 GeoJSON 형식 데이터와 Amazon Location API 형식 간에 쉽게 변환할 수 있습니다.

주제



## Amazon Location SDK 사용을 시작하는 방법

Amazon Location SDK는 애플리케이션에서 Amazon Location Service를 더 간단하게 사용할 수 있는 기능 세트입니다. 이러한 함수를 애플리케이션에 설치하고 가져올 수 있습니다. JavaScript 다음 섹션에서는 Amazon Location 클라이언트, 인증 및 GeoJSON 도우미 라이브러리에 대해 설명합니다.

## Amazon Location 클라이언트

AWS SDK v3에서는 SDK가 서비스별로 구분됩니다. 필요한 부분만 설치할 수 있습니다. 예컨대 Amazon Location 클라이언트와 Amazon Cognito용 자격 증명 공급자를 설치하려면 다음 명령을 사용하세요.

```
npm install @aws-sdk/client-location
npm install @aws-sdk/credential-providers
```

JavaScript 웹 프론트엔드 애플리케이션에서 Amazon Location Service를 쉽게 사용할 수 있도록 Amazon Location 라이브러리 및 자격 증명 공급자의 호스팅 번들을 AWS 제공합니다. 번들 클라이언트를 사용하려면 다음과 같이 스크립트 태그의 HTML에 추가하세요:

```
<script src="https://unpkg.com/@aws/amazon-location-client@1.x/dist/
amazonLocationClient.js"></script>
```

> **Note**
>
> 패키지는 최신 상태로 유지되며 사용하기 쉽도록 이전 버전과 호환됩니다. 이 스크립트 태그 또는 NPM 설치를 사용하면 항상 최신 버전을 사용할 수 있습니다.

## JavaScript 인증 도우미

Amazon 위치 JavaScript 인증 도우미를 사용하면 애플리케이션에서 Amazon 위치 API를 호출할 때 더 간단하게 인증할 수 있습니다 JavaScript . 이 인증 도우미는 Amazon Cognito 또는 API 키를 인증 방법으로 사용할 때 특히 유용합니다. 이 라이브러리는 https://github.com/aws-geospatial/ amazon-location-utilities-auth -helper-js에서 GitHub 사용할 수 있는 오픈 소스 라이브러리입니다.

> **Note**
>
> 인증 도우미의 Amazon Cognito 지원은 Amazon Cognito의 페더레이션 자격 증명 기능을 지원하지 않습니다.

### 설치

웹팩과 같은 빌드 시스템을 사용하는 경우 로컬에 설치하거나 html에 태그가 포함된 미리 빌드된 JavaScript 번들을 포함하는 경우 라이브러리를 사용할 수 있습니다. `<script>`

- NPM을 사용하여 라이브러리를 설치하려면 다음 명령을 사용하세요:

  ```
  npm install @aws/amazon-location-utilities-auth-helper
  ```

- HTML 파일에서 다음 명령을 사용하여 스크립트를 로드합니다.

```
<script src="https://unpkg.com/@aws/amazon-location-utilities-auth-helper@1.x/dist/
amazonLocationAuthHelper.js"></script>
```

가져오기

JavaScript 애플리케이션에서 특정 함수를 사용하려면 해당 함수를 가져와야 합니다. 다음 코드는 기능 `withIdentityPoolId`을 애플리케이션으로 가져오는 데 사용됩니다.

```
import { withIdentityPoolId } from '@aws/amazon-location-utilities-auth-helper';
```

인증 기능

Amazon 위치 인증 도우미에는 `AuthHelper` 객체를 반환하는 다음 기능이 포함됩니다.

- `async withIdentityPoolId( identityPoolId: string): AuthHelper`— 이 함수는 Amazon Cognito와 함께 작동하도록 초기화된 AuthHelper 객체를 반환합니다.
- `async withAPIKey( API_KEY: string): AuthHelper`— 이 함수는 API 키와 함께 작동하도록 초기화된 AuthHelper 객체를 반환합니다.

`AuthHelper` 객체는 다음과 같은 기능을 제공합니다.

- `AuthHelper.getMapAuthenticationOptions()`— AuthHelper 객체의 이 함수는 MapLibre JS의 맵 옵션과 함께 사용할 수 `transformRequest` 있는 JavaScript 객체를 반환합니다. 자격 증명 풀로 초기화한 경우에만 제공됩니다.
- `AuthHelper.getLocationClientConfig()`— AuthHelper 객체의 이 함수는 a를 초기화하는 데 사용할 수 `credentials` 있는 JavaScript 객체를 반환합니다. LocationClient
- `AuthHelper.getCredentials()`— AuthHelper 객체의 이 함수는 Amazon Cognito의 내부 자격 증명을 반환합니다. 자격 증명 풀로 초기화한 경우에만 제공됩니다.

예: Amazon Cognito를 사용하여 MapLibre 맵 객체를 초기화하는 중 AuthHelper

```
import { withIdentityPoolId } from '@aws/amazon-location-utilities-auth-helper';

const authHelper = await withIdentityPoolId("identity-pool-id"); // use Cognito pool id
 for credentials

const map = new maplibregl.Map({
```

```
    container: "map", // HTML element ID of map element
    center: [-123.1187, 49.2819], // initial map center point
    zoom: 16, // initial map zoom
    style: https://maps.geo.region.amazonaws.com/maps/v0/maps/mapName/style-
descriptor', // Defines the appearance of the map
    ...authHelper.getMapAuthenticationOptions(), // Provides credential options
 required for requests to Amazon Location
});
```

예: API 키로 MapLibre 맵 객체 초기화 (**AuthHelper**이 경우에는 필요 없음)

```
const map = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-123.1187, 49.2819], // initial map center point
    zoom: 16, // initial map zoom
    style: https://maps.geo.region.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=api-key-id',
});
```

예: Amazon Cognito를 사용하여 JS용 AWS SDK에서 위치 클라이언트를 초기화하고 AuthHelper

이 예제에서는 AWS v3용 SDK를 사용합니다. JavaScript

```
import { withIdentityPoolId } from '@aws/amazon-location-utilities-auth-helper';

const authHelper = await withIdentityPoolId("identity-pool-id"); // use Cognito pool id
 for credentials

//initialize the Location client:
const client = new LocationClient({
    region: "region",
    ...authHelper.getLocationClientConfig() // sets up the Location client to use the
 Cognito pool defined above
});

//call a search function with the location client:
const result = await client.send(new SearchPlaceIndexForPositionCommand({
    IndexName: "place-index", // Place index resource to use
    Position: [-123.1187, 49.2819], // position to search near
    MaxResults: 10 // number of results to return
});
```

예: API 키를 사용하여 JS용 AWS SDK에서 위치 클라이언트를 초기화하고 AuthHelper

이 예제에서는 AWS v3용 SDK를 사용합니다. JavaScript

```
import { withAPIKey } from '@aws/amazon-location-utilities-auth-helper';

const authHelper = await withAPIKey("api-key-id"); // use API Key id for credentials

//initialize the Location client:
const client = new LocationClient({
    region: "region",
    ...authHelper.getLocationClientConfig() // sets up the Location client to use the
 API Key defined above
});

//call a search function with the location client:
const result = await client.send(new SearchPlaceIndexForPositionCommand({
    IndexName: "place-index", // Place index resource to use
    Position: [-123.1187, 49.2819], // position to search near
    MaxResults: 10 // number of results to return
});
```

## GeoJSON 변환 도우미

Amazon Location GeoJSON 변환 도우미는 Amazon Location Service 데이터 유형을 업계 표준 GeoJSON 형식으로 또는 그 형식에서 변환할 수 있는 도구를 제공합니다. GeoJSON은 예를 들어 MapLibre 지도에 지리 데이터를 렌더링하는 데 사용됩니다. 이 라이브러리는 https://github.com/aws-geospatial/ -js에서 GitHub 사용할 수 있는 오픈 소스 라이브러리입니다. amazon-location-utilities-datatypes

### 설치

웹팩과 같은 로컬 설치와 함께 라이브러리를 사용하거나 html에 <script> 태그가 포함된 사전 빌드된 JavaScript 번들을 포함하여 사용할 수 있습니다.

- NPM을 사용하여 라이브러리를 설치하려면 다음 명령을 사용하세요.

```
npm install @aws/amazon-location-utilities-datatypes
```

- HTML 파일에서 다음 명령을 사용하여 스크립트를 로드합니다.

```
<script src="https://unpkg.com/@aws/amazon-location-utilities-datatypes@1.x/dist/
amazonLocationDataConverter.js"></script>
```

가져오기

JavaScript 애플리케이션에서 특정 함수를 사용하려면 해당 함수를 가져와야 합니다. 다음 코드는 기능 `placeToFeatureCollection`을 애플리케이션으로 가져오는 데 사용됩니다.

```
import { placeToFeatureCollection } from '@aws/amazon-location-utilities-datatypes';
```

GeoJSON 변환 기능

Amazon Location GeoJSON 변환 도우미에는 다음과 같은 기능이 포함됩니다.

- `placeToFeatureCollection(place: GetPlaceResponse | searchPlaceIndexForPositionResponse | searchPlaceIndexForTextResponse, keepNull: boolean): Feature`— 이 함수는 장소 검색 기능의 응답을 1개 이상의 포인트 피처가 있는 FeatureCollection GeoJSON으로 변환합니다.
- `devicePositionToFeatureCollection(devicePositions: GetDevicePositionResponse | BatchGetDevicePositionResponse | GetDevicePositionHistoryResponse | ListDevicePositionsResponse, keepNull: boolean)`— 이 기능은 트래커 장치 위치 함수의 응답을 1개 이상의 Point 기능이 있는 FeatureCollection GeoJSON으로 변환합니다.
- `routeToFeatureCollection(legs: CalculateRouteResponse): FeatureCollection`— 이 함수는 경로 계산 함수의 응답을 단일 기능이 있는 FeatureCollection GeoJSON으로 변환합니다. MultiStringLine 경로의 각 구간은 의 LineString 항목으로 표시됩니다. MultiStringLine
- `geofenceToFeatureCollection(geofences: GetGeofenceResponse | PutGeofenceRequest | BatchPutGeofenceRequest | ListGeofencesResponse): FeatureCollection`— 이 함수는 지오펜스 함수 요청 또는 응답을 폴리곤 기능이 있는 GeoJSON으로 변환합니다. FeatureCollection 응답과 요청 모두에서 지오펜스를 변환할 수 있으므로 또는 를 사용하여 지오펜스를 업로드하기 전에 지도에 지오펜스를 표시할 수 있습니다. PutGeofence BatchPutGeofence

  이 기능은 원형 지오펜스를 근사 폴리곤이 있는 피처로 변환하지만, 필요한 경우 원형 지오펜스를 재현할 수 있는 “중심” 및 “반지름” 속성도 있습니다 (다음 기능 참조).
- `featureCollectionToGeofences(featureCollection: FeatureCollection): BatchPutGeofenceRequestEntry[]`— 이 함수는 폴리곤 피처가 있는 FeatureCollection GeoJSON을 객체 BatchPutGeofenceRequestEntry 배열로 변환하므로 결과를 사용하여 요청을 생성할 수 있습니다. BatchPutGeofence

의 피처에 “center” 및 “radius” 속성이 있는 경우, 해당 피처는 폴리곤의 지오메트리를 무시하고 원형 지오펜스 요청 항목으로 변환됩니다. FeatureCollection

예: 검색 결과를 포인트 레이어로 변환 MapLibre

이 예시에서는 JavaScript v3용 AWS SDK를 사용합니다.

```
import { placeToFeatureCollection } from '@aws/amazon-location-utility-datatypes';

...

let map; // map here is an initialized MapLibre instance

const client = new LocationClient(config);
const input = { your_input };
const command = new searchPlaceIndexForTextCommand(input);
const response = await client.send(command);

// calling utility function to convert the response to GeoJSON
const featureCollection = placeToFeatureCollection(response);
map.addSource("search-result", featureCollection);
map.addLayer({
    id: "search-result",
    type: "circle",
    source: "search-result",
    paint: {
        "circle-radius": 6,
        "circle-color": "#B42222",
    },
});
```

## 안드로이드 모바일 인증 SDK

이러한 유틸리티를 사용하면 Android 애플리케이션에서 Amazon Location Service API를 호출할 때 인증할 수 있습니다. 이는 Amazon Cognito 또는 API 키를 인증 방법으로 사용할 때 특히 유용합니다.

안드로이드 모바일 인증 SDK는 github: 안드로이드용 Amazon Location Service 모바일 인증 SDK에서 사용할 수 있습니다. 또한 모바일 인증 SDK와 SDK는 모두 Maven AWS 리포지토리에서 사용할 수 있습니다.AWS

### 설치

모바일 인증 SDK를 사용하려면 Android Studio의 `build.gradle` 파일에 다음과 같은 가져오기 문을 추가하세요.

```
implementation("software.amazon.location:auth:0.0.1")
implementation("com.amazonaws:aws-android-sdk-location:2.72.0")
```

### 인증 함수

인증 도우미 SDK에는 다음과 같은 기능이 있습니다.

- `authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider`: 이 함수는 API 키와 함께 작동하도록 `LocationCredentialsProvider` 초기화된 값을 반환합니다.
- `authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider`: 이 함수는 Amazon Cognito 자격 증명 풀과 함께 작동하도록 `LocationCredentialsProvider` 초기화된 값을 반환합니다.

### 사용량

코드에서 SDK를 사용하려면 다음 클래스를 가져오십시오.

```
import com.amazonaws.services.geo.AmazonLocationClient
import software.amazon.location.auth.AuthHelper
import software.amazon.location.auth.LocationCredentialsProvider
```

인증 도우미 인스턴스와 위치 클라이언트 제공자 인스턴스를 만들 때는 두 가지 옵션이 있습니다. Amazon Location API 키 또는 Amazon Cognito를 사용하여 인스턴스를 생성할 수 있습니다.

- Amazon Location API 키를 사용하여 인증 도우미 인스턴스를 생성하려면 다음과 같이 도우미 클래스를 선언하십시오.

  ```
  var authHelper = AuthHelper(applicationContext)
  var locationCredentialsProvider : LocationCredentialsProvider =
   authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key")
  ```

- Amazon Cognito를 사용하여 인증 도우미 인스턴스를 생성하려면 다음과 같이 도우미 클래스를 선언하십시오.

```
var authHelper = AuthHelper(applicationContext)
var locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id")
```

위치 자격 증명 공급자를 사용하여 Amazon Location 클라이언트 인스턴스를 생성하고 Amazon 위치 서비스를 호출할 수 있습니다. 다음 예제는 지정된 위도 및 경도 근처의 장소를 검색합니다.

```
var locationClient =
 authHelper.getLocationClient(locationCredentialsProvider.getCredentialsProvider())
var searchPlaceIndexForPositionRequest =
 SearchPlaceIndexForPositionRequest().withIndexName("My-Place-Index-
Name").withPosition(arrayListOf(30.405423, -97.718833))
var nearbyPlaces =
 locationClient.searchPlaceIndexForPosition(searchPlaceIndexForPositionRequest)
```

## iOS 모바일 인증 SDK

이러한 유틸리티를 사용하면 iOS 애플리케이션에서 Amazon Location Service API를 호출할 때 인증할 수 있습니다. 이는 Amazon Cognito 또는 API 키를 인증 방법으로 사용할 때 특히 유용합니다.

iOS 모바일 인증 SDK는 github: iOS용 Amazon Location Service 모바일 인증 SDK에서 사용할 수 있습니다.

### 설치

Xcode 프로젝트에 SDK를 설치합니다.

1. 파일로 이동한 다음 XCode 프로젝트에 패키지 종속성 추가를 선택합니다.
2. 검색 창에 패키지 URL: https://github.com/aws-geospatial/ amazon-location-mobile-auth -sdk-ios/를 입력하고 엔터 키를 누릅니다.
3. `amazon-location-mobile-auth-sdk-ios`패키지를 선택하고 Add Package (패키지 추가) 를 누릅니다.
4. `AmazonLocationiOSAuthSDK`패키지 제품을 선택하고 Add Package (패키지 추가) 를 누릅니다.

### 인증 기능

인증 도우미 SDK에는 다음과 같은 기능이 있습니다.

- `authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider`: 이 함수는 API 키와 함께 작동하도록 `LocationCredentialsProvider` 초기화된 값을 반환합니다.
- `authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider`: 이 함수는 Amazon Cognito 자격 증명 풀과 함께 작동하도록 `LocationCredentialsProvider` 초기화된 값을 반환합니다.

사용량

모바일 인증 SDK를 사용하려면 활동에 다음 명령문을 추가하십시오.

```
import AmazonLocationiOSAuthSDK
import AWSLocationXCF
```

인증 도우미 인스턴스와 위치 클라이언트 제공자 인스턴스를 만들 때는 두 가지 옵션이 있습니다. Amazon Location API 키 또는 Amazon Cognito를 사용하여 인스턴스를 생성할 수 있습니다.

- Amazon Location API 키를 사용하여 인증 도우미 인스턴스를 생성하려면 다음과 같이 도우미 클래스를 선언하십시오.

```
let authHelper = AuthHelper()
let locationCredentialsProvider = authHelper.authenticateWithAPIKey(apiKey: "My-Amazon-Location-API-Key", region: "account-region")
```

- Amazon Cognito를 사용하여 인증 도우미 인스턴스를 생성하려면 다음과 같이 도우미 클래스를 선언하십시오.

```
let authHelper = AuthHelper()
let locationCredentialsProvider =
 authHelper.authenticateWithCognitoUserPool(identityPoolId: "My-Amazon-Location-API-Key", region: "account-region")
```

위치 자격 증명 공급자를 사용하여 Amazon Location 클라이언트 인스턴스를 생성하고 Amazon 위치 서비스를 호출할 수 있습니다. 다음 예제는 지정된 위도 및 경도 근처의 장소를 검색합니다.

```
let locationClient = AWSLocation.default()
let searchPlaceIndexForPositionRequest =
 AWSLocationSearchPlaceIndexForPositionRequest()!
```

```
searchPlaceIndexForPositionRequest.indexName = "My-Place-Index-Name"
searchPlaceIndexForPositionRequest.position = [30.405423, -97.718833]
let nearbyPlaces = locationClient.searchPlaceIndex(forPosition:
 searchPlaceIndexForPositionRequest)
```

## 안드로이드 모바일 트래킹 SDK

Amazon Location 모바일 추적 SDK는 쉽게 인증하고, 디바이스 위치를 캡처하고, Amazon 위치 추적기로 위치 업데이트를 전송하는 데 도움이 되는 유틸리티를 제공합니다. SDK는 구성 가능한 업데이트 간격으로 위치 업데이트의 로컬 필터링을 지원합니다. 이렇게 하면 데이터 비용이 절감되고 Android 애플리케이션의 간헐적 연결이 최적화됩니다.

안드로이드 추적 SDK는 안드로이드용 Amazon 위치 모바일 추적 SDK에서 사용할 수 GitHub 있습니다. 또한 모바일 인증 SDK와 SDK는 모두 Maven AWS 리포지토리에서 사용할 수 있습니다.AWS Android 추적 SDK는 일반 SDK와 함께 작동하도록 설계되었습니다. AWS

이 섹션에서는 Amazon Location 모바일 추적 Android SDK에 대한 다음 주제를 다룹니다.

주제

- 설치
- 사용량
- 필터
- 안드로이드 모바일 SDK 추적 함수
- 예제

### 설치

SDK를 설치하려면 안드로이드 스튜디오에 있는 build.gradle 파일의 종속성 섹션에 다음 줄을 추가하십시오.

```
implementation("software.amazon.location:tracking:0.0.1")
implementation("software.amazon.location:auth:0.0.1")
implementation("com.amazonaws:aws-android-sdk-location:2.72.0")
```

### 사용량

이 절차는 SDK를 사용하여 객체를 인증하고 생성하는 방법을 보여줍니다. `LocationTracker`

> **Note**
>
> 이 절차에서는 섹션에 언급된 라이브러리를 가져온 것으로 가정합니다. 설치

1. 코드에서 다음 클래스를 가져오세요.

```
import software.amazon.location.tracking.LocationTracker
import software.amazon.location.tracking.config.LocationTrackerConfig
import software.amazon.location.tracking.util.TrackingSdkLogLevel
import com.amazonaws.services.geo.AmazonLocationClient
import software.amazon.location.auth.AuthHelper
import software.amazon.location.auth.LocationCredentialsProvider
```

2. `LocationTracker`객체를 만들려면 `LocationCredentialsProvider` 파라미터가 필요하므로 `AuthHelper` 다음으로 를 생성하십시오.

```
// Create an authentication helper using credentials from Cognito
val authHelper = AuthHelper(applicationContext)
val locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id")
```

3. 이제 `LocationCredentialsProvider` 및 `LocationTrackerConfig` 를 사용하여 `LocationTracker` 객체를 생성합니다.

```
val config = LocationTrackerConfig(
    trackerName = "MY-TRACKER-NAME",
    logLevel = TrackingSdkLogLevel.DEBUG,
    accuracy = Priority.PRIORITY_HIGH_ACCURACY,
    latency = 1000,
    frequency = 5000,
    waitForAccurateLocation = false,
    minUpdateIntervalMillis = 5000,
)
locationTracker = LocationTracker(
    applicationContext,
    locationCredentialsProvider,
    config,
)
```

필터

Amazon 위치 모바일 추적 Android SDK에는 세 개의 내장 위치 필터가 있습니다.

- `TimeLocationFilter`: 정의된 시간 간격을 기준으로 업로드할 현재 위치를 필터링합니다.
- `DistanceLocationFilter`: 지정된 거리 임계값을 기준으로 위치 업데이트를 필터링합니다.
- `AccuracyLocationFilter`: 마지막 업데이트 이후 이동한 거리를 현재 위치의 정확도와 비교하여 위치 업데이트를 필터링합니다.

이 예제에서는 생성 `LocationTracker` 시 필터를 추가합니다.

```
val config = LocationTrackerConfig(
    trackerName = "MY-TRACKER-NAME",
    logLevel = TrackingSdkLogLevel.DEBUG,
    accuracy = Priority.PRIORITY_HIGH_ACCURACY,
    latency = 1000,
    frequency = 5000,
    waitForAccurateLocation = false,
    minUpdateIntervalMillis = 5000,
    locationFilters = mutableListOf(TimeLocationFilter(), DistanceLocationFilter(),
 AccuracyLocationFilter())
)
locationTracker = LocationTracker(
    applicationContext,
    locationCredentialsProvider,
    config,
)
```

이 예제에서는 다음을 사용하여 `LocationTracker` 런타임에 필터를 활성화 및 비활성화합니다.

```
// To enable the filter
locationTracker?.enableFilter(TimeLocationFilter())

// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())
```

안드로이드 모바일 SDK 추적 함수

Android용 Amazon Location 모바일 추적 SDK에는 다음과 같은 기능이 포함되어 있습니다.

- 클래스: `LocationTracker`

```
constructor(context: Context,locationCredentialsProvider:
LocationCredentialsProvider,trackerName: String) 또는
constructor(context: Context,locationCredentialsProvider:
LocationCredentialsProvider,clientConfig: LocationTrackerConfig)
```

객체를 생성하는 이니셜라이저 함수입니다. `LocationTracker` 의 인스턴스가 `LocationCredentialsProvider` `trackerName` 필요하고 선택적으로 의 인스턴스가 필요합니다. `LocationTrackingConfig` 구성이 제공되지 않으면 기본값으로 초기화됩니다.

- 클래스: `LocationTracker`

  ```
  start(locationTrackingCallback: LocationTrackingCallback)
  ```

  사용자 위치에 액세스하여 Amazon 위치 추적기로 전송하는 프로세스를 시작합니다.

- 클래스: `LocationTracker`

  ```
  isTrackingInForeground()
  ```

  위치 추적이 현재 진행 중인지 확인합니다.

- 클래스: `LocationTracker`

  ```
  stop()
  ```

  사용자 위치 추적 프로세스를 중지합니다.

- 클래스: `LocationTracker`

  ```
  startTracking()
  ```

  사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 시작합니다.

- 클래스: `LocationTracker`

  ```
  startBackground(mode: BackgroundTrackingMode, serviceCallback:
  ServiceCallback)
  ```

  애플리케이션이 백그라운드에 있는 동안 사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 시작합니다. BackgroundTrackingMode 에는 다음과 같은 옵션이 있습니다.

  - `ACTIVE_TRACKING`: 이 옵션은 사용자의 위치 업데이트를 능동적으로 추적합니다.
  - `BATTERY_SAVER_TRACKING`: 이 옵션은 15분마다 사용자의 위치 업데이트를 추적합니다.

- 수업: `LocationTracker`

```
stopBackgroundService()
```

애플리케이션이 백그라운드에 있는 동안 사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 중지합니다.

- 클래스: `LocationTracker`

```
getTrackerDeviceLocation()
```

Amazon 위치 서비스에서 디바이스 위치를 검색합니다.

- 클래스: `LocationTracker`

```
getDeviceLocation(locationTrackingCallback: LocationTrackingCallback?)
```

통합 위치 제공자 클라이언트에서 현재 장치 위치를 검색하여 Amazon Location Tracker에 업로드합니다.

- 클래스: `LocationTracker`

```
uploadLocationUpdates(locationTrackingCallback:
LocationTrackingCallback?)
```

구성된 위치 필터를 기반으로 필터링한 후 디바이스 위치를 Amazon Location 서비스에 업로드합니다.

- 클래스: `LocationTracker`

```
enableFilter(filter: LocationFilter)
```

특정 위치 필터를 활성화합니다.

- 클래스: `LocationTracker`

```
checkFilterIsExistsAndUpdateValue(filter: LocationFilter)
```

특정 위치 필터를 비활성화합니다.

- 클래스: `LocationTrackerConfig`

```
LocationTrackerConfig( // Required var trackerName:
String, // Optional var locationFilters: MutableList =
mutableListOf( TimeLocationFilter(), DistanceLocationFilter(), ), var
logLevel: TrackingSdkLogLevel = TrackingSdkLogLevel.DEBUG, var accuracy:
```

Int = Priority.PRIORITY_HIGH_ACCURACY, var latency: Long = 1000, var frequency: Long = 1500, var waitForAccurateLocation: Boolean = false, var minUpdateIntervalMillis: Long = 1000, var persistentNotificationConfig: NotificationConfig = NotificationConfig())

이렇게 하면 사용자 정의 매개변수 LocationTrackerConfig 값으로 가 초기화됩니다. 매개변수 값이 제공되지 않으면 기본값으로 설정됩니다.

- 클래스: LocationFilter

  shouldUpload(currentLocation: LocationEntry, previousLocation: LocationEntry?): Boolean

  LocationFilter는 사용자가 사용자 지정 필터 구현을 위해 구현할 수 있는 프로토콜입니다. 이전 위치와 현재 위치를 비교하고 현재 위치를 업로드해야 하는지 여부를 반환하는 shouldUpload 함수를 구현해야 합니다.

예제

다음 코드 샘플은 모바일 추적 SDK 기능을 보여줍니다.

이 예제에서는 LocationTracker 를 사용하여 백그라운드에서 추적을 시작하고 중지합니다.

```
// For starting the location tracking
locationTracker?.startBackground(
BackgroundTrackingMode.ACTIVE_TRACKING,
object : ServiceCallback {
    override fun serviceStopped() {
        if (selectedTrackingMode == BackgroundTrackingMode.ACTIVE_TRACKING) {
            isLocationTrackingBackgroundActive = false
        } else {
            isLocationTrackingBatteryOptimizeActive = false
        }
    }
},
)

// For stopping the location tracking
locationTracker?.stopBackgroundService()
```

# iOS 모바일 트래킹 SDK

Amazon Location 모바일 추적 SDK는 쉽게 인증하고, 디바이스 위치를 캡처하고, Amazon 위치 추적기로 위치 업데이트를 전송하는 데 도움이 되는 유틸리티를 제공합니다. SDK는 구성 가능한 업데이트 간격으로 위치 업데이트의 로컬 필터링을 지원합니다. 이를 통해 데이터 비용이 절감되고 iOS 애플리케이션의 간헐적 연결이 최적화됩니다.

iOS 추적 SDK는 iOS용 Amazon 로케이션 모바일 추적 SDK에서 사용할 수 GitHub 있습니다.

이 섹션에서는 Amazon Location 모바일 추적 iOS SDK에 대한 다음 주제를 다룹니다.

주제



## 설치

iOS용 모바일 추적 SDK를 설치하려면 다음 절차를 따르십시오.

1. Xcode 프로젝트에서 파일로 이동하여 패키지 종속성 추가를 선택합니다.
2. 검색 창에 https://github.com/aws-geospatial/ amazon-location-mobile-tracking -sdk-ios/ URL을 입력하고 엔터 키를 누릅니다.
3. `amazon-location-mobile-tracking-sdk-ios`패키지를 선택하고 Add Package (패키지 추가) 를 클릭합니다.
4. `AmazonLocationiOSTrackingSDK`패키지 제품을 선택하고 Add Package (패키지 추가) 를 클릭합니다.

## 사용량

다음 절차는 Cognito의 자격 증명을 사용하여 인증 도우미를 만드는 방법을 보여줍니다.

1. 라이브러리를 설치한 후에는 설명 중 하나 또는 둘 모두를 파일에 추가해야 합니다. `info.plist`

```
Privacy - Location When In Use Usage Description
```

```
Privacy - Location Always and When In Use Usage Description
```

2. 다음으로, AuthHelper 클래스에서 파일을 가져오세요.

```
import AmazonLocationiOSAuthSDKimport AmazonLocationiOSTrackingSDK
```

3. 그런 다음 Amazon AuthHelper Cognito의 자격 증명을 사용하여 인증 도우미를 생성하여 객체를 생성하고 AWS SDK와 함께 사용합니다.

```
let authHelper = AuthHelper()
let locationCredentialsProvider =
 authHelper.authenticateWithCognitoUserPool(identityPoolId: "My-Cognito-Identity-
Pool-Id", region: "My-region") //example: us-east-1
let locationTracker = LocationTracker(provider: locationCredentialsProvider,
 trackerName: "My-tracker-name")

// Optionally you can set ClientConfig with your own values in either initialize or
 in a separate function
// let trackerConfig = LocationTrackerConfig(locationFilters:
 [TimeLocationFilter(), DistanceLocationFilter()],

trackingDistanceInterval: 30,
trackingTimeInterval: 30,
logLevel: .debug)

// locationTracker = LocationTracker(provider: credentialsProvider, trackerName:
 "My-tracker-name",config: trackerConfig)
// locationTracker.setConfig(config: trackerConfig)
```

필터

Amazon 위치 모바일 추적 iOS SDK에는 세 개의 내장 위치 필터가 있습니다.

- `TimeLocationFilter`: 정의된 시간 간격을 기준으로 업로드할 현재 위치를 필터링합니다.
- `DistanceLocationFilter`: 지정된 거리 임계값을 기준으로 위치 업데이트를 필터링합니다.
- `AccuracyLocationFilter`: 마지막 업데이트 이후 이동한 거리를 현재 위치의 정확도와 비교하여 위치 업데이트를 필터링합니다.

이 예제에서는 생성 `LocationTracker` 시 필터를 추가합니다.

```
val config = LocationTrackerConfig(
    trackerName = "MY-TRACKER-NAME",
    logLevel = TrackingSdkLogLevel.DEBUG,
    accuracy = Priority.PRIORITY_HIGH_ACCURACY,
    latency = 1000,
    frequency = 5000,
    waitForAccurateLocation = false,
    minUpdateIntervalMillis = 5000,
    locationFilters = mutableListOf(TimeLocationFilter(), DistanceLocationFilter(),
 AccuracyLocationFilter())
)

locationTracker = LocationTracker(
    applicationContext,
    locationCredentialsProvider,
    config,
)
```

이 예제에서는 다음을 사용하여 LocationTracker 런타임에 필터를 활성화 및 비활성화합니다.

```
// To enable the filter
locationTracker?.enableFilter(TimeLocationFilter())

// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())
```

### iOS 모바일 SDK 트래킹 기능

iOS용 Amazon 위치 모바일 추적 SDK에는 다음과 같은 기능이 포함되어 있습니다.

- 클래스: LocationTracker

  init(provider: LocationCredentialsProvider, trackerName: String, config: LocationTrackerConfig? = nil)

  객체를 생성하는 이니셜라이저 함수입니다. LocationTracker 의 인스턴스가 LocationCredentialsProvider trackerName 필요하고 선택적으로 의 인스턴스가 필요합니다. LocationTrackingConfig 구성이 제공되지 않으면 기본값으로 초기화됩니다.
- 클래스: LocationTracker

  setTrackerConfig(config: LocationTrackerConfig)

이렇게 하면 위치 추적기 초기화 후 언제든지 Tracker 구성이 적용되도록 설정됩니다.

- 클래스: `LocationTracker`

  `getTrackerConfig()`

  그러면 앱에서 사용하거나 수정할 위치 추적 구성을 가져옵니다.

  반환값: `LocationTrackerConfig`
- 클래스: `LocationTracker`

  `getDeviceId()`

  위치 추적기에서 생성된 기기 ID를 가져옵니다.

  반환값: `String?`
- 클래스: `LocationTracker`

  `startTracking()`

  사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 시작합니다.
- 클래스: `LocationTracker`

  `resumeTracking()`

  사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 재개합니다.
- 클래스: `LocationTracker`

  `stopTracking()`

  사용자 위치 추적 프로세스를 중지합니다.
- 클래스: `LocationTracker`

  `startBackgroundTracking(mode: BackgroundTrackingMode)`

  애플리케이션이 백그라운드에 있는 동안 사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 시작합니다. `BackgroundTrackingMode`에는 다음과 같은 옵션이 있습니다.
  - `Active`: 이 옵션은 위치 업데이트를 자동으로 일시 중지하지 않습니다.
  - `BatterySaving`: 이 옵션은 위치 업데이트를 자동으로 일시 중지합니다.
  - `None`: 이 옵션은 전체적으로 백그라운드 위치 업데이트를 비활성화합니다.

- 클래스: `LocationTracker`

  `resumeBackgroundTracking(mode: BackgroundTrackingMode)`

  애플리케이션이 백그라운드에 있는 동안 사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 재개합니다.

- 클래스: `LocationTracker`

  `stopBackgroundTracking()`

  애플리케이션이 백그라운드에 있는 동안 사용자 위치에 액세스하여 AWS 트래커로 전송하는 프로세스를 중지합니다.

- 클래스: `LocationTracker`

  `getTrackerDeviceLocation(nextToken: String?, startTime: Date? = nil, endTime: Date? = nil, completion: @escaping (Result<GetLocationResponse, Error>)`

  시작일과 종료일 및 시간 사이에 사용자 장치의 업로드된 추적 위치를 검색합니다.

  반품: `Void`

- 클래스: `LocationTrackerConfig`

  `init()`

  그러면 LocationTrackerConfig 기본값으로 가 초기화됩니다.

- 클래스: `LocationTrackerConfig`

  `init(locationFilters: [LocationFilter]? = nil, trackingDistanceInterval: Double? = nil, trackingTimeInterval: Double? = nil, trackingAccuracyLevel: Double? = nil, uploadFrequency: Double? = nil, desiredAccuracy: CLLocationAccuracy? = nil, activityType: CLActivityType? = nil, logLevel: LogLevel? = nil)`

  이렇게 하면 사용자 정의 매개변수 `LocationTrackerConfig` 값으로 가 초기화됩니다. 매개변수 값을 제공하지 않으면 기본값으로 설정됩니다.

- 클래스: `LocationFilter`

`shouldUpload(currentLocation: LocationEntity, previousLocation: LocationEntity?, trackerConfig: LocationTrackerConfig)`

`LocationFilter`는 사용자가 사용자 지정 필터 구현을 위해 구현할 수 있는 프로토콜입니다. 사용자는 이전 위치와 현재 위치를 비교하고 현재 위치를 업로드해야 하는 경우 반환하는 `shouldUpload` 함수를 구현해야 합니다.

예제

이 섹션에서는 iOS용 Amazon 위치 모바일 추적 SDK를 사용하는 예를 자세히 설명합니다.

> **Note**
>
> `info.plist`파일에 필요한 권한이 설정되어 있는지 확인하십시오. 이 권한은 사용량 섹션에 나열된 권한과 동일합니다.

다음 예제는 장치 위치를 추적하고 추적된 위치를 검색하는 기능을 보여줍니다.

```
Privacy - Location When In Use Usage Description
Privacy - Location Always and When In Use Usage Description
```

위치 추적 시작:

```
do {
    try locationTracker.startTracking()
    }
catch TrackingLocationError.permissionDenied {
        // Handle permissionDenied by showing the alert message or opening the app
 settings
        }
```

위치 추적 재개:

```
do {
    try locationTracker.resumeTracking()
    }
catch TrackingLocationError.permissionDenied {
```

```
    // Handle permissionDenied by showing the alert message or opening the app settings
    }
```

위치 추적 중지:

```
locationTracker.stopTracking()
```

백그라운드 추적 시작:

```
do {

 locationTracker.startBackgroundTracking(mode: .Active) // .Active, .BatterySaving, .None
    }
catch TrackingLocationError.permissionDenied {
   // Handle permissionDenied by showing the alert message or opening the app settings
   }
```

백그라운드 트래킹 재개:

```
do {
    locationTracker.resumeBackgroundTracking(mode: .Active)
    }
catch TrackingLocationError.permissionDenied {
    // Handle permissionDenied by showing the alert message or opening the app settings
    }
```

백그라운드 트래킹을 중지하려면:

```
locationTracker.stopBackgroundTracking()
```

트래커에서 기기의 추적된 위치를 검색하세요.

```
func getTrackingPoints(nextToken: String? = nil) {
let startTime: Date = Date().addingTimeInterval(-86400) // Yesterday's day date and
 time
let endTime: Date = Date()
locationTracker.getTrackerDeviceLocation(nextToken: nextToken, startTime: startTime,
 endTime: endTime, completion: { [weak self] result in
    switch result {
    case .success(let response):
```

```
        let positions = response.devicePositions
        // You can draw positions on map or use it further as per your requirement

        // If nextToken is available, recursively call to get more data
        if let nextToken = response.nextToken {
            self?.getTrackingPoints(nextToken: nextToken)
        }
    case .failure(let error):
        print(error)
    }
})
}
```

# Amazon Location API

Amazon Location Service는 프로그래밍 방식으로 위치 기능에 액세스할 수 있는 API 작업을 제공합니다. 여기에는 맵, 장소, 경로, 트래커, 지오펜스, 리소스 태그 지정을 위한 API가 포함됩니다. 사용 가능한 API 작업에 대한 자세한 내용은 Amazon Location Service API 참조를 참조하세요.

이 가이드의 코드 예시 챕터에서 샘플을 확인할 수 있습니다.

## AWS SDK와 함께 아마존 로케이션 사용

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

언어별로 Amazon Location Service와 함께 사용할 수 있는 SDK에 대한 자세한 내용을 알아보려면 이 가이드의 언어별 SDK 항목을 참조하세요.

> **SDK 버전**
>
> 프로젝트에서 사용하는 가장 최신 AWS SDK 빌드와 기타 SDK를 사용하고 SDK를 최신 상태로 유지하는 것이 좋습니다. AWS SDK는 최신 특징과 기능, 보안 업데이트를 제공합니다. 예를 들어 AWS SDK의 최신 빌드를 찾으려면 AWS SDK의 브라우저 설치 주제 설명서를 참조하십시오. JavaScript JavaScript

# Amazon Location API 오류 메시지 업데이트

2023년 8월 1일부터 Amazon Location 팀은 다음 테이블에 설명된 대로 API 오류 메시지를 변경하고 있습니다. 오류 코드는 변경되지 않습니다. 애플리케이션이 정확한 오류 메시지 문자열을 사용하는 경우 애플리케이션을 새 문자열로 업데이트해야 합니다. 질문이나 문제에 대한 도움이 필요하면 문의하세요. AWS Support

주제



## 장소

장소

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 500 | InternalServerException | Internal Server Exception | Internal server error. Try again later. |
| 404 | ResourceNotFoundException | resource <PlaceIndexName> not found, reason: <Reason><br><br>Resource '<PlaceIndexName>' not found<br><br>placeIdx<PlaceIndexName> not found, reason: <Reason> | Place index not found: <PlaceIndexName>. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| | | no place index with name '%s' found | |
| 404 | ResourceNotFoundException | place not found | Place not found: <PlaceId>. |
| 400 | ValidationException | PlaceIndex <PlaceIndexName> cannot be used for SearchPlaceIndexForSuggestions because it has IntendedUse <IntendedUse> | A place index with 'IntendedUse' set to Storage does not support 'SearchPlaceIndexForSuggestion' operation. |
| 400 | ValidationException | only one of 'BiasPosition' or 'FilterBBox' may be set | Only one of 'BiasPosition' or 'FilterBBox' may be set. |
| 400 | ValidationException | BiasPosition must have exactly 2 entries | 'BiasPosition' must have exactly 2 entries. |
| 400 | ValidationException | BiasPosition[0] must be between -180 and 180 | 'BiasPosition[0]' must be between -180 and 180. |
| 400 | ValidationException | BiasPosition[1] must be between -90 and 90 | 'BiasPosition[1]' must be between -90 and 90. |
| 400 | ValidationException | FilterBBox must have exactly 4 entries | 'FilterBBox' must have exactly 4 entries. |
| 400 | ValidationException | FilterBBox[0] must be between -180 and 180 | 'FilterBBox[0]' must be between -180 and 180. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | FilterBBox[1] must be between -90 and 90 | 'FilterBBox[1]' must be between -90 and 90. |
| 400 | ValidationException | FilterBBox[2] must be between -180 and 180 | 'FilterBBox[2]' must be between -180 and 180. |
| 400 | ValidationException | FilterBBox[3] must be between -90 and 90 | 'FilterBBox[3]' must be between -90 and 90. |
| 400 | ValidationException | FilterBBox must have more southwesterly point before more northeasterly point | 'FilterBBox' must have more southwesterly position before more northeasterly position. |
| 400 | ValidationException | Position must have exactly 2 entries | 'Position' must have exactly 2 entries. |
| 400 | ValidationException | Position[0] must be between -180 and 180 | 'Position[0]' must be between -180 and 180. |
| 400 | ValidationException | Position[1] must be between -90 and 90 | 'Position[1]' must be between -90 and 90. |
| 400 | ValidationException | Language is not a valid BCP 47 language tag | 'Language' must comply with the BCP 47 Language Tag standard, but was set to <GivenValue>. For more information, see https://wikipedia.org/wiki/IETF_language_tag. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | 'placeID' is invalid | 'PlaceId' must be a valid ID. |
| 400 | ValidationException | no customer account ID parameter found | 'RequesterAccountID' is a required field. |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |
| 400 | ValidationException | unsupported price plan '<PricingPlan>' | 'PricingPlan' must be set to RequestBasedUsage. |
| 400 | ValidationException | 'DataSource' must be one of: Here, Esri | 'DataSource' must be one of Esri, Grab, Here. |
| 400 | ValidationException | Grab is only supported in the ap-southeast-1 region | 'DataSource' Grab must only be used in following regions: ap-southeast-1. |
| 400 | ValidationException | 'IntendedUse' and 'PricingPlan' must both be provided to update either property | 'IntendedUse' and 'PricingPlan' must both be provided to update either attribute. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| 402 | ServiceQuotaExceededException | Place resources per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/ | Place index resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/. |
| 409 | ConflictException | Resource already exists | Place index already exists: <PlaceIndexName>. |

## 맵

맵

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| 500 | InternalServerException | Internal Server Exception<br><br>unable to find style template<br><br>Error fetching style<br><br>was not able to serialize the map style file | Internal server error. Try again later. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 404 | ResourceNotFoundException | Map not found | Map not found: <MapName>. |
| 404 | ResourceNotFoundException | Sprites are not supported for this resource | Sprite not found: <SpriteName>. |
| 400 | ValidationException | Resource name should be set | 'MapName' is a required field. |
| 400 | ValidationException | Must provide a valid number for start and end of Range | Font Unicode range start and end numbers must both be provided. |
| 400 | ValidationException | Start of range is an invalid number: <StartValue> | Start of font Unicode range must be a valid number. |
| 400 | ValidationException | End of range is an invalid number: <StartValue> | End of font Unicode range must be a valid number. |
| 400 | ValidationException | End of range must be exactly 255 higher from start of range, difference found: <Difference> | The difference between the start and end of the font Unicode range must be exactly 255. Difference found: <Difference>. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | Start of range must be a multiple of 256, found <StartValue> | Start of font Unicode range must be a multiple of 256, but was set to: <StartValue>. |
| 400 | ValidationException | Request font is empty | 'FontStack' is a required field. |
| 400 | ValidationException | Request font is not valid for the datasource <DataSource> | <FontStack> is not a supported font stack for data source <DataSource>. For more information about the list of supported font stacks, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapGlyphs.html. |
| 400 | ValidationException | Request font is not valid | <FontStack> is not a supported font stack for data source <DataSource>. For more information about the list of supported font stacks, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapGlyphs.html. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | DataSource is invalid: <DataSource> | 'DataSource' must be one of Esri, Grab, Here. |
| 400 | ValidationException | Request filename is empty | 'FileName' is a required field. |
| 400 | ValidationException | Request filename is not valid | <SpriteFile> is not a supported sprite file name. For more information about the list of supported sprite file names, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapSprites.html. |
| 400 | ValidationException | Filename is invalid: <FileName> | <SpriteFile> is not a supported sprite file name. For more information about the list of supported sprite file names, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapSprites.html. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | Filename is an invalid content type: <FileName> | <SpriteFile> is not a supported sprite file name. For more information about the list of supported sprite file names, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapSprites.html. |
| 400 | ValidationException | Filename is invalid: <FileName> | 'Filename' must not be empty. |
| 400 | ValidationException | y-coordinate part of 'Y' must be a valid integer | y- coordinate part of 'Y' must be an integer. |
| 400 | ValidationException | tile resolution part of 'Y' must be a valid integer followed by 'x' | Tile resolution part of 'Y' must be an integer followed by 'X'. |
| 400 | ValidationException | file type extension part of 'Y' must not be empty if a '.' is present | File type extension part of 'Y' must not be empty if a '.' is present. |
| 400 | ValidationException | 'Z' must be a valid integer | 'Z' must be an integer. |
| 400 | ValidationException | 'X' must be a valid integer | 'X' must be an integer. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | 'Z' must not be less than minimum zoom of style '<Style>' (<Minimum Value>) | 'Z' must not be less than minimum zoom of style <Style> (<MinimumValue>). |
| 400 | ValidationException | 'Z' must not be greater than maximum zoom of style '<Style' (<Maximum Value>) | 'Z' must not be greater than maximum zoom of style Style (<MaximumValue>). |
| 400 | ValidationException | 'Z' value not supported | 'Z' must be between 0 and 63. |
| 400 | ValidationException | tile resolution part of 'Y' must be omitted because '<Style>' is a vector style | Tile resolution part of 'Y' must be omitted for style <Style>. |
| 400 | ValidationException | tile resolution part of 'Y' must be at least 1 | Tile resolution part of 'Y' must be at least 1. |
| 400 | ValidationException | tile resolution part of 'Y' must not be greater than max resolution of style '<Style>' (<Maximum Resolution>) | Tile resolution part of 'Y' must not be greater than maximum resolution of style <Style> (max <MaxResolution>). |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | file type extension part of 'Y' must be one of <SupportedFileFormats> (or may be omitted) for style '<Style>' | File type extension part of 'Y' must be one of <SupportedFileFormats> (or may be omitted) for style <Style>. |
| 400 | ValidationException | file type extension part of 'Y' must be omitted for style '<Style>' | File type extension part of 'Y' must be omitted for style <Style>. |
| 400 | ValidationException | y-coordinate part of 'Y' must be an integer in the range 0..2^Zoom -1 (0..<MaxTileCoordinate>) | y-coordinate part of 'Y' must be an integer in the range 0..2^Zoom -1 (0..<MaxTileCoordinate>). |
| 400 | ValidationException | 'DataSource' must be one of: Here, Esri | 'DataSource' must be one of Esri, Grab, Here. |
| 400 | ValidationException | unsupported price plan '<PricingPlan>' | 'PricingPlan' must be set to RequestBasedUsage. |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| 400 | ValidationException | Unsupported Map Style: <Style> | <Style> is not a supported map style. For more information about list of supported map styles, see https://docs.aws.amazon.com/location/latest/APIReference/API_MapConfiguration.html. |
| 402 | ServiceQuotaExceededException | Map resources per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/ | Map resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/. |
| 409 | ConflictException | Resource already exists | Map already exists: <MapName>. |

## 트래커

트래커

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| 500 | InternalServerException | Internal Server Exception | Internal server error. Try again later. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| | | internal server error<br><br>unable to retrieve point from the storage<br><br>unable to verify tracker<br><br>Error processing List request | |
| 404 | ResourceNotFoundException | tracker not found: <TrackerName><br><br>Tracker with name <TrackerName> was not found | Tracker not found: <TrackerName>. |
| 404 | ResourceNotFoundException | association not found: TrackerName <TrackerName>; and ConsumerArn < ConsumerArn > | Association between tracker <TrackerName> and consumer <ConsumerArn> is not found. |
| 400 | ValidationException | 'ConsumerArn' must refer to a geofence collection resource | 'ConsumerArn' must refer to a geofence collection resource. |
| 400 | ValidationException | 'ConsumerArn' must refer to a resource in the same region as the tracker it is associated to | 'ConsumerArn' must refer to a resource in the same region as the tracker it is associated with. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | 'ConsumerArn' must refer to a resource in the same AWS account as the tracker is it associated to | 'ConsumerArn' must refer to a resource in the same AWS account as the tracker it is associated with. |
| 400 | ValidationException | 'DataSource' must be one of: Here, Esri | 'DataSource' must be one of Esri, Grab, Here. |
| 400 | ValidationException | Nothing to update. | At least one of the following fields must be set: 'Description', 'PositionFiltering' |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |
| 400 | ValidationException | request.TrackerName not found on request | 'TrackerName ' is a required field. |
| 400 | ValidationException | no deviceId parameter found | 'DeviceId' is a required field. |
| 400 | ValidationException | unsupported price plan '<PricingPlan>' | 'PricingPlan' must be set to RequestBasedUsage |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| 400 | ValidationException | provided start time is incorrect, should follow the format YYYY-MM-DDThh:mm:ss.sssZ“ | 'StartTimeInclusive' must follow the format YYYY-MM-DDThh:mm:ss.sssZ. |
| 400 | ValidationException | provided end time is incorrect, should follow the format YYYY-MM-DDThh:mm:ss.sssZ | 'EndTimeExclusive' must follow the format YYYY-MM-DDThh:mm:ss.sssZ. |
| 400 | ValidationException | end time must be after start time | 'EndTimeExclusive' must be after 'StartTimeInclusive'. |
| 400 | ValidationException | invalid key state | KMS key must be a symmetric Customer Master Key (CMK). Invalid state found. For more information about how key state affects the use of a KMS key, see https://docs.aws.amazon.com/kms/latest/developerguide/key-state.html. |
| 400 | ValidationException | key not found | Invalid KMS key. '<KmsKeyId>' <KmsKeyIdValue> not found. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | key is disabled | Symmetric Customer Master Key (CMK) must be enabled. |
| 400 | ValidationException | access denied | Symmetric Customer Master Key (CMK) must allow Amazon Location to create grants to its KMS key. |
| 402 | ServiceQuotaExceededException | Tracker <TrackerName> may not have more than <Max> consumer associations | Tracker resource may not have more than <Max> consumer associations. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/. |
| 402 | ServiceQuotaExceededException | Trackers per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/ | Tracking resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 409 | ConflictException | association already exists: TrackerName <TrackerName>; and ConsumerArn <ConsumerArn> | An association already exists between tracker <TrackerName> and consumer <ConsumerArn>. |
| 409 | ConflictException | Tracker already exists: <TrackerName> | Tracker already exists: <TrackerName>. |

## 경로

### 경로

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 500 | InternalServerException | Internal Server Exception | Internal server error. Try again later. |
| 404 | ResourceNotFoundException | Resource not found | Route calculator not found: <RouteCalculatorName>. |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |
| 400 | ValidationException | 'DataSource' must be one of: Here, Esri, Grab | 'DataSource' must be one of Esri, Grab, Here. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | <PricingPlan> pricing plan is not supported | 'PricingPlan' must be set to RequestBasedUsage |
| 400 | ValidationException | unsupported price plan '<PricingPlan>' | 'PricingPlan' must be set to RequestBasedUsage |
| 400 | ValidationException | Grab is only supported in the ap-southeast-1 region | 'DataSource' <DataSourceName> must only be used in following regions: ap-southeast-1. |
| 400 | ValidationException | PricingPlan must be 'RequestBasedUsage' | 'PricingPlan' must be set to RequestBasedUsage. |
| 400 | ValidationException | 'DeparturePositions[0][0]' must be between -180 and 180 | 'DeparturePositions[0][0]' must be between -180 and 180. |
| 400 | ValidationException | 'DeparturePositions[0][1]' must be between -90 and 90 | 'DeparturePositions[0][1]' must be between -90 and 90. |
| 400 | ValidationException | 'DestinationPositions[0][0]' must be between -180 and 180 | 'DestinationPositions[0][0]' must be between -180 and 180. |
| 400 | ValidationException | 'DestinationPositions[0][1]' must be between -90 and 90. | 'DestinationPositions[0][1]' must be between -90 and 90 |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | 'DepartNow' may not be true if 'DepartureTime' is set | Only one of 'DepartNow' or 'DepartureTime' may be set. |
| 400 | ValidationException | '<TravelModeOption>'' may not be set when 'TravelMode' has value <TravelModeOption> | '<TravelModeOption>' must not be set when 'TravelMode' has value <TravelModeOption>. |
| 400 | ValidationException | 'CarModeOptions' may not be set when 'TravelMode' has value Walking | 'CarModeOptions' must not be set when 'TravelMode' has value Walking. |
| 400 | ValidationException | 'TruckModeOptions' may not be set when 'TravelMode' has value Walking | 'TruckModeOptions' must not be set when 'TravelMode' has value Walking. |
| 400 | ValidationException | 'TruckModeOptions' may not be set when 'TravelMode' has value Car | 'TruckModeOptions' must not be set when 'TravelMode' has value Car. |
| 400 | ValidationException | 'CarModeOptions' may not be set when 'TravelMode' has value Truck | 'CarModeOptions' must not be set when 'TravelMode' has value Truck. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | At least one of [Height, Length, Width] must be set in 'TruckModeOptions.Dimensions' | At least one of the following attribute must be set in TruckModeOptions.Dimensions: Height, Length, Width. |
| 400 | ValidationException | At least one of [Total] must be set in 'TruckModeOptions.Weight' | At least one of the following attribute must be set in TruckModeOptions.Weight: Total. |
| 400 | ValidationException | 'DeparturePositions' count must be 10 or less with DataSource set to Esri | 'DeparturePositions' must have length at most 10 for 'DataSource' Esri. |
| 400 | ValidationException | 'DestinationPositions' count must be 10 or less with DataSource set to Esri | 'DestinationPositions' must have length at most 10 for 'DataSource' Esri. |
| 400 | ValidationException | 'DeparturePositions[0]' is more than 40km away from 'DestinationPositions[0]' | 'DeparturePositions[0]' must not be more than 40 km away from 'DestinationPositions[0]'. |
| 400 | ValidationException | 'DeparturePositions[0]' is more than 400km away from 'DestinationPositions[0]' | 'DeparturePositions[0]' must not be more than 400 km away from 'DestinationPositions[0]'. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | DeparturePositions [0] is contained within an unsupported region. Korea is not supported for CalculateRouteMatrix with the provider Esri. | DeparturePositions [0] is located in Korea, which is not supported when using CalculateRouteMatrix with data provider Esri. |
| 400 | ValidationException | '<HereTruckDimension>' must be between <Min> and <Max> <Unit> | 'HereTruckDimension' must be between <Min> and <Max> <Unit>. |
| 400 | ValidationException | 'WaypointPositions[0][0]' must be between -180 and 180 | 'WaypointPositions[0][0]' must be between -180 and 180. |
| 400 | ValidationException | 'WaypointPositions[0][1]' must be between -90 and 90 | 'WaypointPositions[0][1]' must be between -90 and 90. |
| 400 | ValidationException | 'WaypointPositions[1][0]' must be between -180 and 180 | 'WaypointPositions[1][0]' must be between -180 and 180. |
| 400 | ValidationException | 'WaypointPositions[1][1]' must be between -90 and 90 | 'WaypointPositions[1][1]' must be between -90 and 90. |
| 400 | ValidationException | No road segment could be matched for one or more coordinates within a radius (1km) | One or more provided positions are more than 1 km from the nearest road segment. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | Some positions in the request are unreachable | Some positions in the request are unreachable. |
| 400 | ValidationException | Total distance between all waypoints must be not be greater than 40km for DataSource Esri when using TravelMode Walking | Total distance between all route positions must not be greater than 40 km for 'DataSource' Esri and 'TravelMode' Walking. |
| 400 | ValidationException | Total distance between all waypoints must be not be greater than 400km for DataSource Esri | Total distance between all route positions must not be greater than 400 km for 'DataSource' Esri. |
| 400 | ValidationException | Following positions in the request are unreachable: <UnreachablePositions> | The following positions are unreachable: <UnreachablePositions>. |
| 400 | ValidationException | 'DepartureTime' contains a badly-formatted timestamp | 'DepartureTime' must follow the format YYYY-MM-DDThh:mm:ss.sssZ. |
| 400 | ValidationException | 'TravelMode' <TravelMode> is not supported by <DataProvider> | 'TravelMode' <TravelMode> not supported by data provider <DataProvider>. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | 'DeparturePositions' must be set | 'DeparturePositions' must not be empty. |
| 400 | ValidationException | 'DestinationPositions' must be set | 'DestinationPositions' must not be empty. |
| 400 | ValidationException | Some inputs in the request are invalid | Some inputs in the request are invalid. |
| 400 | ValidationException | No route found between position <FirstPosition> and position <SecondPosition> | No route found between position <FirstPosition> and position <SecondPosition>. |
| 400 | ValidationException | No route found | No route found. For more information, see https://developer.here.com/documentation/routing-api/dev_guide/topics/notice.html. |
| 400 | ValidationException | No route found | No route found. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 402 | ServiceQuotaExceededException | Route calculators per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/ | Route calculator resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/. |
| 409 | ConflictException | Resource already exists | Route calculator already exists: <RouteCalculatorName>. |

## 메타데이터

메타데이터

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 500 | InternalServerException | Internal Server Error<br><br>Error processing List request | Internal server error. Try again later. |
| 404 | ResourceNotFoundException | APIKey not found | Api key not found: <APIKeyName>. |
| 404 | ResourceNotFoundException | APIKeyID not found | ApiKeyId not found: <APIKeyID>. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | Either ExpireTime or NoExpiry must be provided | At least one of the following fields must be set: 'ExpireTime', 'NoExpiry'. |
| 400 | ValidationException | NoExpiry cannot be set to false if no ExpireTime is provided | 'ExpireTime' must be set when 'NoExpiry' has value false. |
| 400 | ValidationException | ExpireTime cannot be set if NoExpiry is true | 'ExpireTime' must not be set when 'NoExpiry' has value true. |
| 400 | ValidationException | Expire time '<ExpireTimeValue>' is not a valid time format | 'ExpireTime' must follow the format YYYY-MM-DDThh:mm:ss.sssZ. |
| 400 | ValidationException | Expire time '<ExpireTimeValue>' cannot be in the past when creating a key | 'ExpireTime' must not be in the past. |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | The API Key %s has been recently used and the requested update may impact current usage. Specify ForceUpdate=true to update the API Key configuration. | This update may cause some users to lose API access. Because this API Key has been used in the last 7 days, you must set 'ForceUpdate' to true to confirm this change. |
| 400 | ValidationException | Expire time '<ExpireTimeValue>' must not be more than 1 minute in the past | 'ExpireTime' must not be more than 1 minute in the past. |
| 400 | ValidationException | Description, ExpireTime, NoExpiry and Restrictions can't all be empty | At least one of the following fields must be set: 'Description', 'ExpireTime', 'NoExpiry', 'Restrictions'. |
| 400 | ValidationException | API Key expired | 'ApiKeyId' must not be expired. |
| 409 | ConflictException | API key named <APIKeyName> already exists | Api key already exists: <APIKeyName>. |

# 지오펜스

## 지오펜스

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| 500 | InternalServerException | internal server error<br><br>Internal server error<br><br>Unsupported geofence geometry encountered<br><br>geometry marshal error<br><br>geometry load error<br><br>unable to get geofence collection<br><br>unable to delete geofences<br><br>unable to retrieve geofence<br><br>Error processing List request | Internal server error. Try again later. |
| 404 | ResourceNotFoundException | collection not found: <GeofenceCollectionName><br><br><GeofenceCollectionName> geofence collection not found | Geofence Collection not found: <GeofenceCollectionName>. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
| --- | --- | --- | --- |
| | | Resource not found error<br><br>no geofence with given name found | |
| 400 | ValidationException | unsupported price plan '<PricingPlan>' | 'PricingPlan' must be set to RequestBasedUsage. |
| 400 | ValidationException | KMS key must be a symmetric CMK. Invalid usage type: <UsageType> | KMS key must be a symmetric Customer Master Key (CMK). Invalid usage type <UsageType>. For how to create a symmetric CMK, refer to https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html#create-symmetric-cmk. |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |
| 400 | ValidationException | PricingPlanDataSource cannot be updated without updating PricingPlan | 'PricingPlan' must be provided to update 'PricingPlanDataSource'. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | nothing to update | At least one of the following fields must be set: 'Description' |
| 400 | ValidationException | invalid key state | KMS key must be a symmetric Customer Master Key (CMK). Invalid state <InvalidState>. For more information about how key state affects the use of a KMS key, see https://docs.aws.amazon.com/kms/latest/developerguide/key-state.html. |
| 400 | ValidationException | key not found | Invalid KMS key. '<KmsKeyId>' <KmsKeyIdValue> not found. |
| 400 | ValidationException | key is disabled | Symmetric Customer Master Key (CMK) must be enabled. |
| 400 | ValidationException | access denied | Symmetric Customer Master Key (CMK) must allow Amazon Location to create grants to its KMS key. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | duplicate geofence ID in batch | 'GeofenceId' <DuplicatedGeofenceID> is duplicated in batch. |
| 400 | ValidationException | missing GeofenceId | 'GeofenceId' must not be empty. |
| 400 | ValidationException | Invalid token | 'NextToken' must be a valid token. |
| 400 | ValidationException | Expired token | 'NextToken' must not be expired. |
| 400 | ValidationException | Position[0] must be between -180 and 180 | 'Position[0]' must be between -180 and 180. |
| 400 | ValidationException | Position[1] must be between -90 and 90 | 'Position[1]' must be between -90 and 90. |
| 400 | ValidationException | radius must be less than or equal to 1000km | 'Geometry.Circle.Radius' must be less than or equal to 1000km. |
| 400 | ValidationException | no geofence with given name found | Geofence not found: <CollectionName>. |
| 400 | ValidationException | Geometry must contain either a Circle or Polygon, not both | Only one of 'Circle' or 'Polygon' may be set within 'Geometry'. |
| 400 | ValidationException | Geometry must contain a Polygon or a Circle | One of 'Polygon' or 'Circle' must be set within 'Geometry'. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | radius must be greater than 0m | 'Geometry.Circle.Radius' must be greater than 0m. |
| 400 | ValidationException | empty polygon | 'Geometry.Polygon' must not be empty. |
| 400 | ValidationException | empty polygon ring | 'Geometry.Polygon' must not be empty. |
| 400 | ValidationException | circle can not cross antimeridian | 'Geometry.Circle' must not cross antimeridian. Cut it in two such that neither part's representation crosses the antimeridian. |
| 400 | ValidationException | polygon can not cross antimeridian | 'Geometry.Polygon' must not cross antimeridian. Cut it in two such that neither part's representation crosses the antimeridian. |
| 400 | ValidationException | polygon can not have interior rings (holes), remove holes | 'Geometry.Polygon' must not have interior rings (holes). For more information about interior rings see https://www.rfc-editor.org/rfc/rfc7946.html#appendix-A.3. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | polygon ring is not closed | 'Geometry.Polygon' contains an open ring. Close the ring by ensuring the first and last positions are equal. |
| 400 | ValidationException | polygon ring has more than 1000 vertices | 'Geometry.Polygon' must not have more than 1000 vertices. |
| 400 | ValidationException | polygon ring has fewer than 4 positions | Number of vertices in 'Geometry.Polygon' must be greater or equal to 4. |
| 400 | ValidationException | invalid center | 'Geometry.Circle.Center' must be a valid position (longitude/latitude pair). |
| 400 | ValidationException | radius must be greater than 0m | 'Geometry.Circle.Radius' must be greater than 0 m. |
| 400 | ValidationException | longitude range should be between -180 and 180 degrees | Longitude must be between -180 and 180 degrees, but was set to <Provided Longitude>. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | latitude range should be between -90 and 90 degrees | Latitude must be between -90 and 90 degrees, but was set to <Provided Longitude>. |
| 400 | ValidationException | polygon exterior ring is expected to be counter clockwise | 'Geometry.Polygon' must be oriented counter-clockwise. |
| 400 | ValidationException | polygon interior ring should be clockwise oriented | 'Geometry.Polygon' must be oriented clockwise. |
| 400 | ValidationException | radius must be less than or equal to 1000km | 'Geometry.Circle.Radius' must be less than or equal to 1000 km. |
| 400 | ValidationException | timestamp.Parse() error | 'SampleTime' must follow the format YYYY-MM-DDThh:mm:ss.sssZ. |
| 400 | ValidationException | invalid input | 'SourceArn' must refer to a tracker resource. |
| 400 | ValidationException | arn: invalid prefix | 'SourceArn' must be a valid ARN. For more information, see https://docs.aws.amazon.com/general/latest/gr/AWS-arns-and-namespaces.html. |

| 오류 코드 | 예외 | 오래된 오류 메시지입니다. | 새 오류 메시지 |
|---|---|---|---|
| 400 | ValidationException | arn: not enough sections | 'SourceArn' must be a valid ARN. For more information, see https://docs.aws.amazon.com/general/latest/gr/AWS-arns-and-namespaces.html. |
| 400 | ValidationException | invalid resource part | 'SourceArn' must refer to a tracker resource. |
| 402 | ServiceQuotaExceededException | Geofence collections per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/ | Geofence collection resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/. |
| 409 | ConflictException | collection already exists: <GeofenceCollectionName> | Geofence Collection already exists: <GeofenceCollectionName>. |
| 409 | ConflictException | Resource conflict error | Geofence already exists: <GeofenceName>. |

# Amazon Location Service 작업을 위한 코드 예제 및 자습서

이 항목에서는 Amazon Location Service에 대해 배우는 데 도움이 되는 코드 예제, 자습서 및 블로그 게시물 목록을 보여줍니다. 각 코드 예제에는 작동 방식에 대한 설명이 포함되어 있습니다.

AWS 지리공간 GitHub 페이지, Amazon Location의AWS 샘플 GitHub 페이지 및 AWS 블로그 사이트에서 추가 샘플을 찾을 수 있습니다.

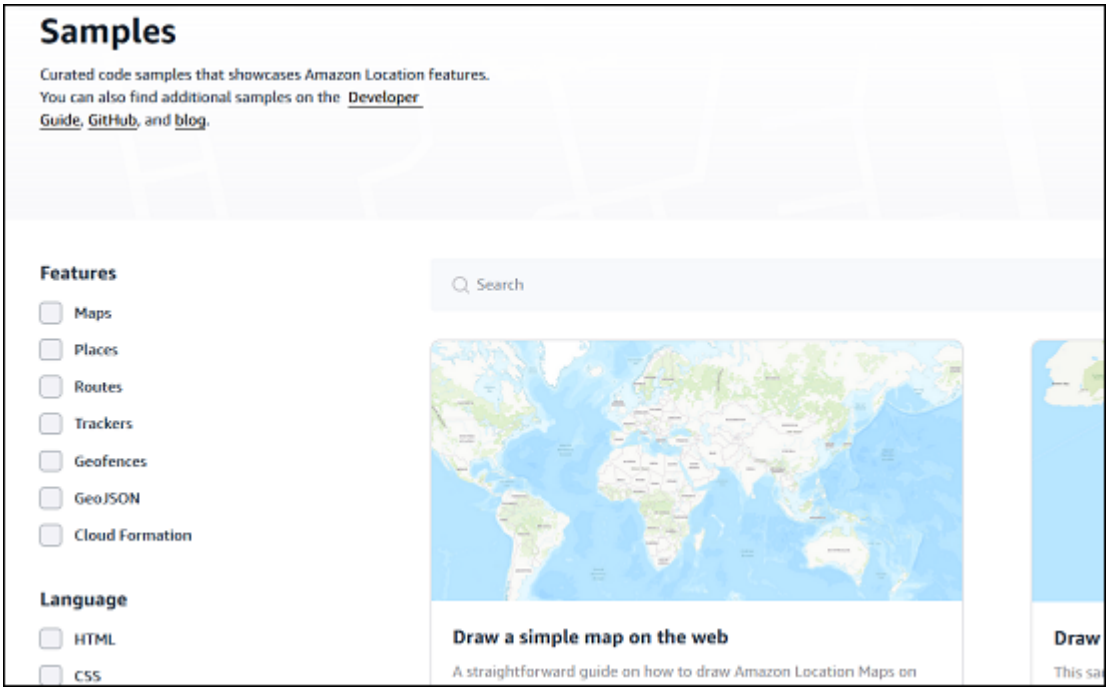
> **Note**
>
> AWS 지리공간 GitHub 페이지와 AWS 샘플 GitHub 페이지 간의 차이를 이해하는 것이 좋습니다.
>
> - 지리공간 GitHub — AWS 지리공간 GitHub 페이지에는 Amazon Location Service 팀에서 생성하고 유지 관리하는 샘플이 포함되어 있습니다.
> - 샘플 GitHub — Amazon AWS Location의 샘플 GitHub 페이지에는 Amazon Location용으로 생성되었지만 적극적으로 유지 관리되거나 유지 관리되지 않을 수 있는 샘플이 포함되어 있습니다.

빠른 시작 자습서는 대부분의 샘플에 유용한 필수 구성 요소를 완료하는 방법을 보여 주므로 다른 샘플을 사용하기 전에 시작하는 것이 좋습니다.

주제

- Amazon Location 데모 사이트
- 자습서: 빠른 시작
- 튜토리얼: 데이터베이스 강화
- 예제: 앱 탐색
- 예제: 맵 스타일 지정
- 예제: 마커 그리기
- 예제: 군집된 포인트 그리기
- 예제: 다각형 그리기
- 예제: 맵 언어 변경
- 블로그: 예상 배송 시간 알림
- 예: 스트림 포지션 업데이트
- 예: 지오펜싱 및 트래킹 모바일 애플리케이션

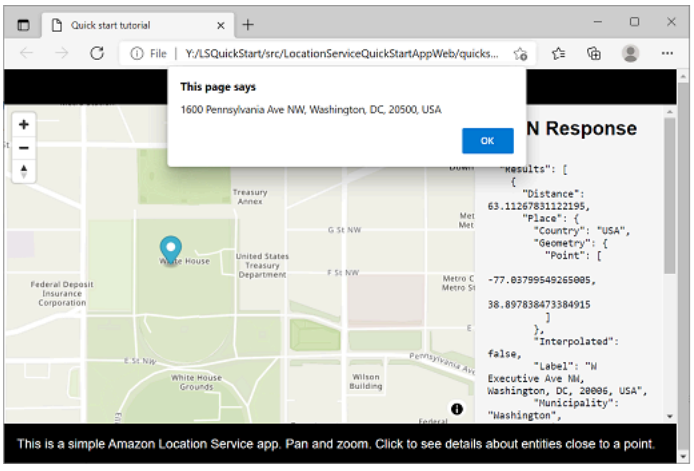
# Amazon Location 데모 사이트

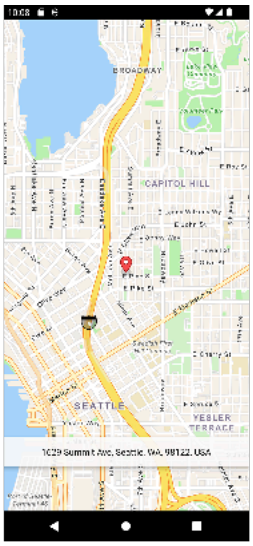



Amazon Location 데모 사이트에서 Amazon Location Service의 소스 코드가 포함된 데모를 확인할 수 있습니다. 이 사이트에는 호스팅된 웹 데모와 Android용 데모 앱이 포함되어 있습니다.

또한 사이트의 샘플 페이지에서 기능, 언어 및 플랫폼별로 필터링할 수 있는 다양한 샘플을 찾을 수 있습니다.

# 자습서: 빠른 시작

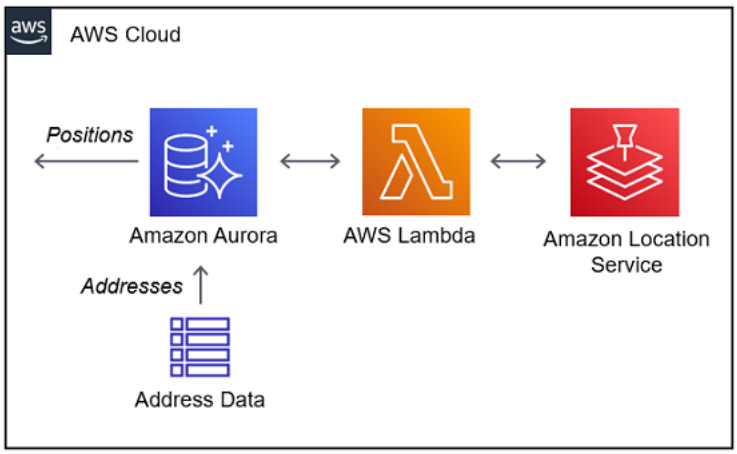


웹, iOS, Android 디바이스에서 사용할 수 있는 퀵 스타트 튜토리얼이 있습니다. 각 플랫폼에 대해 자습서에서는 애플리케이션에 대화형 맵을 추가하는 방법과 애플리케이션에서 Amazon Location Service API를 호출하는 방법을 보여줍니다. 이 튜토리얼은 정적 웹페이지에서, 안드로이드폰 애플리케이션은 Kotlin, iOS 애플리케이션은 JavaScript Swift에서 이용할 수 있습니다.

- JavaScript 정적 웹페이지 문서 링크의 경우: 웹 앱 생성
- 안드로이드용 Kotlin 애플리케이션 문서 링크: Amazon Location Service로 빠른 시작
- iOS 앱용 Swift 문서 링크: iOS 앱 만들기

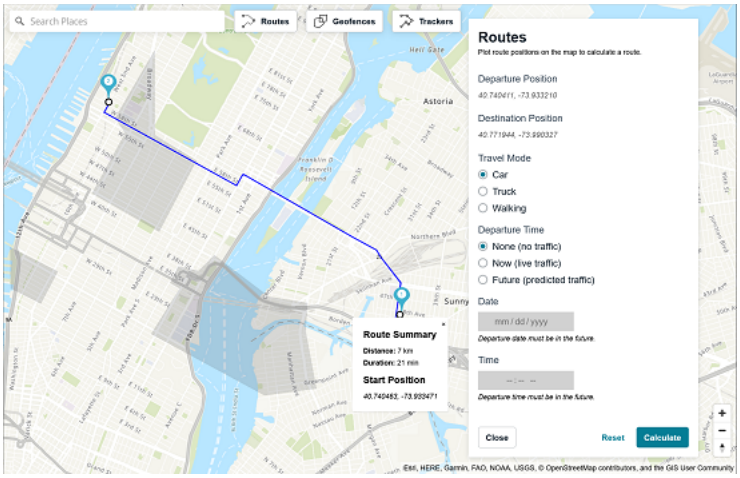
## 튜토리얼: 데이터베이스 강화




이 자습서에서는 Amazon Location Service를 사용하여 주소를 정규화하고 Amazon Aurora 데이터베이스의 레코드에 위도와 경도를 추가하는 방법을 보여줍니다. AWS Lambda 아마존 Aurora를 사용하고 있습니다. AWS Lambda

설명서 링크: Amazon Location Service용 Amazon Aurora PostgreSQL 사용자 정의 함수

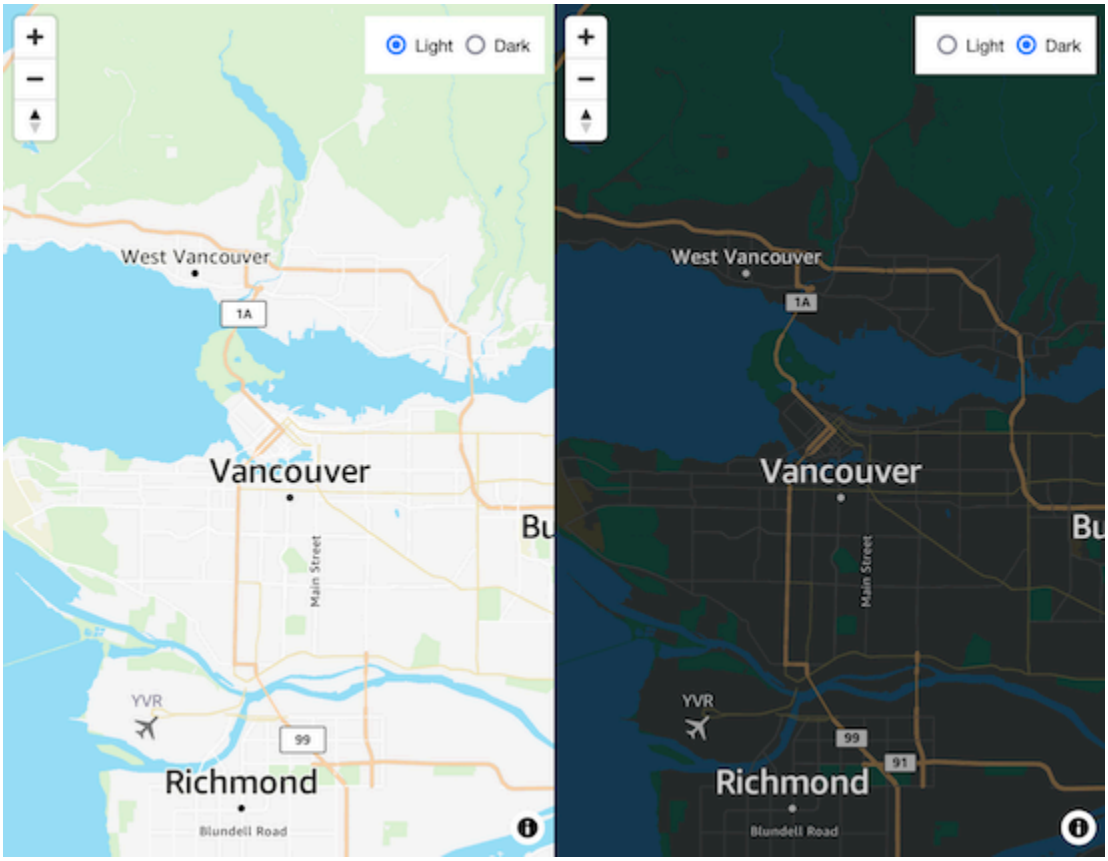
# 예제: 앱 탐색




Amazon Location Service의 기능을 익히는 가장 좋은 방법 중 하나는 Amazon Location 콘솔 내의 탐색 기능을 사용하는 것입니다. 이 전체 웹 애플리케이션 예제는 콘솔의 맵, 장소, 경로, 지오펜스, 트래커 기능을 모방하여 자신의 앱에서 이러한 기능을 다시 만드는 방법을 보여줍니다. Amplify, React 등을 사용합니다. JavaScript

샘플 GitHub 링크: 샘플 애플리케이션 살펴보기

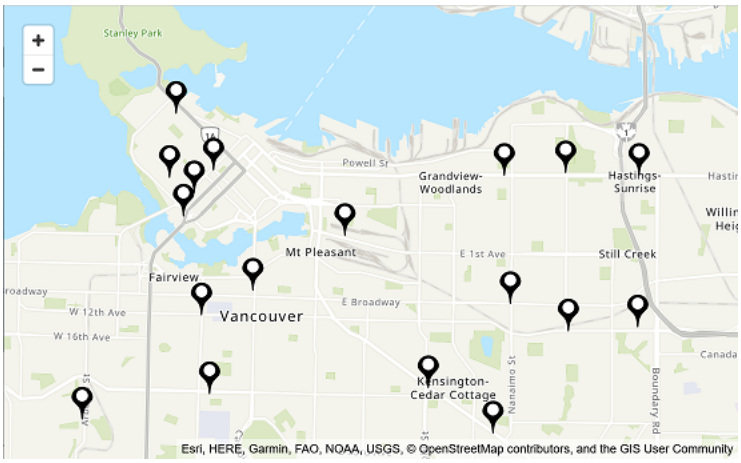
# 예제: 맵 스타일 지정




이 코드 예제는 in을 사용하여 MapLibre 위성 지도와 벡터 로드맵 간에 전환하는 방법을 보여줍니다 JavaScript. 사용자 MapLibre, Amazon 위치 인증 도우미 및 JavaScript.

지리공간 GitHub 링크: 스타일 전환 기능이 있는 대화형 지도

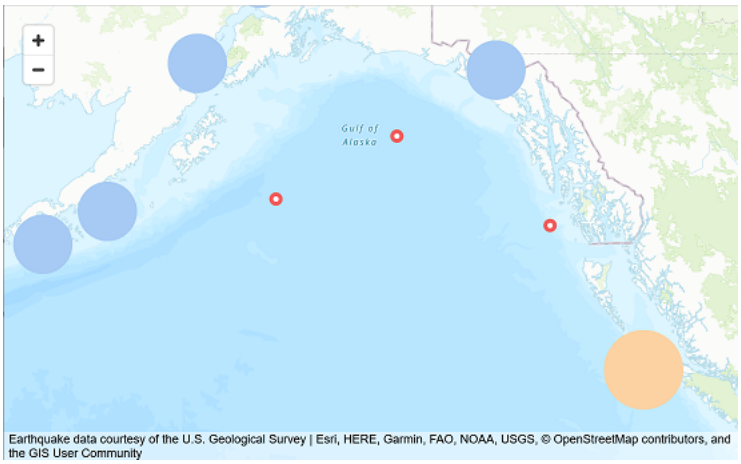
# 예제: 마커 그리기



이 코드 예제는 캐나다 BC 주 밴쿠버의 Amazon Locker 위치를 보여줍니다. 이는 포인트 위치에 마커를 그리는 방법을 보여줍니다. Node.js MapLibre, React, Amazon 위치 인증 도우미 등을 JavaScript 사용합니다.

지리공간 GitHub 링크: 지점에 마커가 있는 대화형 지도

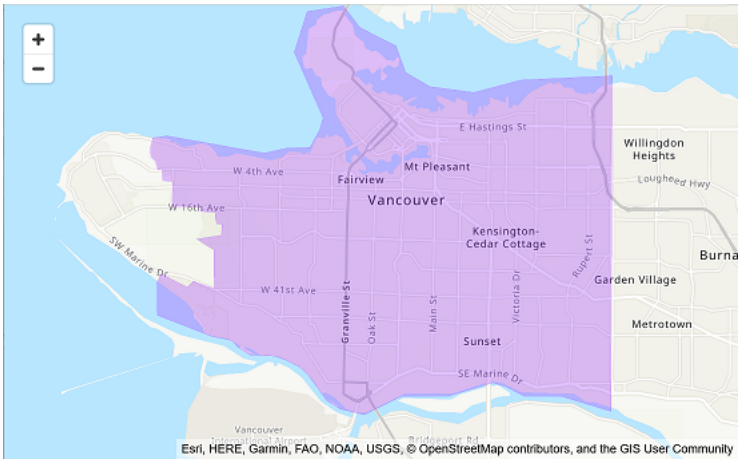
## 예제: 군집된 포인트 그리기




이 코드 예제는 USGS 지진 데이터를 사용하여 맵에서 서로 가까이 있을 때 서로 군집되는 포인트를 그리는 방법을 보여줍니다. 용도 MapLibre, Node.js, React, Amplify 및. JavaScript

샘플 GitHub 링크: 포인트 클러스터가 포함된 대화형 맵

## 예제: 다각형 그리기




이 코드 예제에서는 맵에 다각형을 그리는 방법을 보여줍니다. Node.js MapLibre, React, Amazon 위치 인증 도우미 등을 JavaScript 사용합니다.

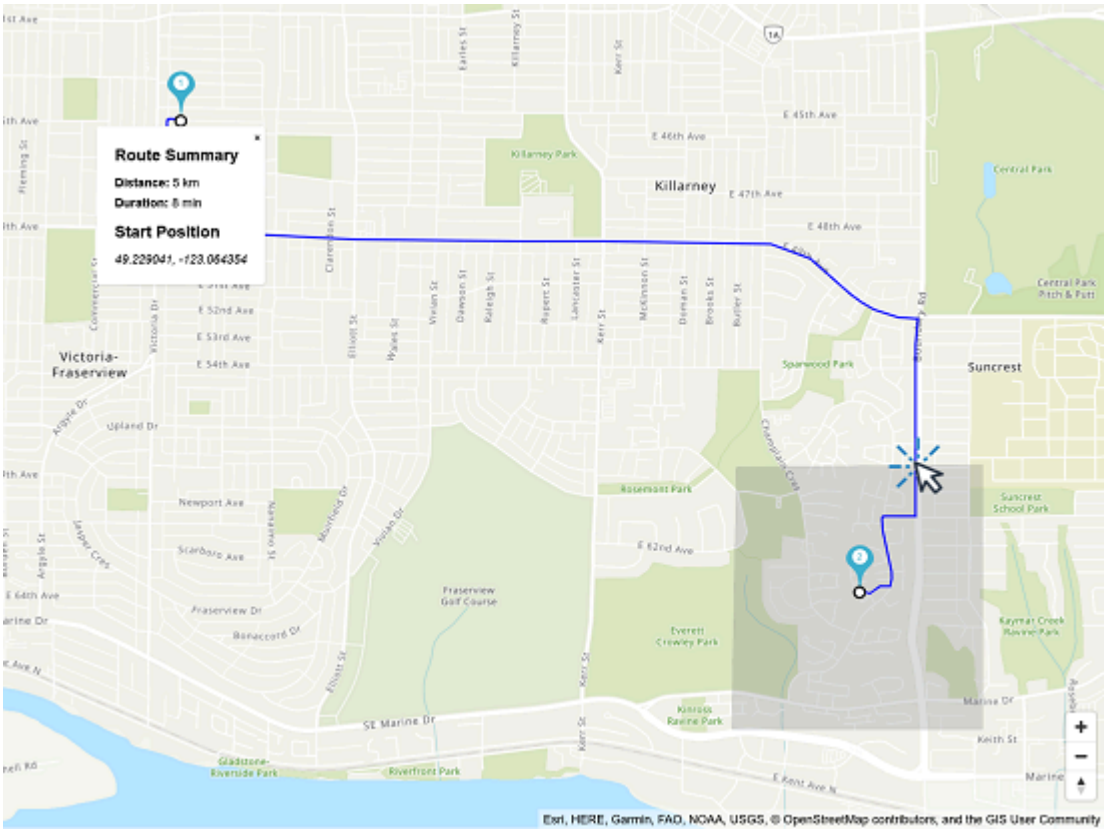
지리공간 GitHub 링크: 폴리곤이 포함된 대화형 맵

# 예제: 맵 언어 변경


이 코드 예제는 Amazon Location에서 맵의 표시 언어를 변경하는 방법을 보여줍니다. Amplify, React 등을 사용합니다. MapLibre

샘플 GitHub 링크: 맵 언어 변경 샘플

# 블로그: 예상 배송 시간 알림



이 블로그 게시물에서는 고객에게 예상 배송 시간을 알리는 다양한 방법을 보여줍니다. 경로를 사용하여 예상 운전 시간을 표시한 다음, 추적기와 지오펜스를 사용하여 운전자가 고객에게 가까워지면 이를 알리는 방법에 대해 설명합니다. Amplify, React, 아마존 EventBridge 및 아마존 심플 알림 서비스 (Amazon SNS) 를 사용합니다.

블로그 링크: 예상 도착 시간 및 근접 알림

## 예: 스트림 포지션 업데이트


Kinesis 스트림 투 트래커 앱: 이 샘플은 Kinesis 데이터 스트림을 사용하여 Amazon Location Service에 트래커 업데이트를 게시하는 방법을 보여줍니다. 이 샘플은 Python으로 작성된 배포 가능한 람다 애플리케이션으로, Kinesis 데이터 스트림과 통합하여 Kinesis 이벤트 및 배치 업데이트 디바이스 위치를 사용할 수 있습니다.

리포지토리 링크: 트래커 앱으로의 Amazon 로케이션 Amazon Kinesis Data Streams 스트림

추적 및 지오펜스에 대한 자세한 내용은 지오펜스 및 추적기 설명서를 참조하십시오. 개발자는 AWS의 서버리스 애플리케이션 리포지토리 설명서를 따르거나 Lambda 콘솔에서 직접 앱을 배포할 수 있습니다.

장치 위치 스트리밍 샘플 앱: 이 코드 예제는 장치 위치 데이터를 Kinesis Data Stream으로 스트리밍하는 방법과 지오펜스 알림의 작동 방식을 보여줍니다. 이 앱은 위에 나열된 Kinesis Stream to Tracker 샘플 앱에 따라 실행되며 Amazon Location Service에서 스트리밍된 트래커 위치를 업데이트할 수 있습니다.

리포지토리 링크: Amazon 로케이션 디바이스 포지션 스트리밍 샘플 앱

# 예: 지오펜싱 및 트래킹 모바일 애플리케이션

이 샘플 애플리케이션은 Lambda와 Amazon Location 기능의 조합을 사용하여 트래커와 지오펜스가 상호 작용하는 방식을 보여줍니다. AWS IoT iOS와 안드로이드용 튜토리얼이 있습니다.

튜토리얼 링크: 샘플 지오펜스 및 트래커 모바일 애플리케이션

# Amazon Location Service 사용 방법

Amazon Location Service 기능을 사용하여 지리 및 위치 관련 작업을 완료할 수 있습니다. 그런 다음 이러한 작업을 결합하여 지오마케팅, 배송, 자산 추적과 같은 더 복잡한 사용 사례를 처리할 수 있습니다.

애플리케이션에 위치 기능을 구축할 준비가 되면 목표와 성향에 따라 다음 방법을 사용하여 Amazon Location Service 기능을 사용합니다:

- 탐색 도구 — Amazon Location 리소스를 시험해 보고 싶다면 다음 도구는 API에 액세스하고 사용해 볼 수 있는 가장 빠른 방법이 될 수 있습니다:
  - Amazon Location 콘솔은 빠르게 액세스할 수 있는 다양한 도구를 제공합니다. Explore 페이지를 사용하여 리소스를 생성 및 관리하고 API를 사용해 볼 수 있습니다. 콘솔은 나중에 설명하는 다른 방법을 사용할 준비를 위해 리소스 (일반적으로 일회성 작업) 를 만드는 데도 유용합니다.
  - AWS명령줄 인터페이스(CLI)를 사용하면 터미널을 사용하여 리소스를 생성하고 Amazon Location API에 액세스할 수 있습니다. AWS CLI은 자격 증명을 사용하여 구성할 때 인증을 처리합니다.
  - Amazon Location Service API를 사용하여 작업을 수행하는 방법을 보여주는 코드 예시와 자습서를 볼 수 있습니다. 여기에는 콘솔의 Explore 페이지 기능 대부분을 모방한 예시가 포함됩니다.
- 플랫폼 SDK — 맵에서 데이터를 시각화하지 않는 경우 모든 AWS 표준 도구를 사용하여 AWS를 구축할 수 있습니다.
  - C++, Go, Java, .NET, Node.js, PHP, PHP JavaScript, Python, Ruby 등의 SDK를 사용할 수 있습니다.
- 프런트엔드 SDK 및 라이브러리 — Amazon Location을 사용하여 모바일 플랫폼에서 애플리케이션을 구축하거나 어떤 플랫폼에서든 지도에 데이터를 시각화하려는 경우 다음 옵션을 사용할 수 있습니다:
  - AWS Amplify라이브러리는 iOS, Android 및 JavaScript웹 애플리케이션 내에서 Amazon Location을 통합합니다.
  - MapLibre 라이브러리를 사용하면 Amazon Location을 사용하여 클라이언트측 맵을 iOS, Android 및 JavaScript웹 애플리케이션으로 렌더링할 수 있습니다.
  - Tangram ES 라이브러리를 사용하면 iOS 및 Android 웹 애플리케이션 내에서 OpenGL ES를 사용하여 벡터 데이터에서 2D 및 3D 맵을 렌더링할 수 있습니다. 웹 애플리케이션을 위한 JavaScriptTangram도 있습니다.
- 직접 HTTPS 요청 전송 — SDK를 사용할 수 없는 프로그래밍 언어로 작업하거나 AWS에 요청을 보내는 방법을 더 세밀하게 제어하려는 경우, 서명 버전 4 서명 프로세스로 인증된 직접 HTTPS 요청을

전송하여 Amazon Location에 액세스할 수 있습니다. 서명 버전 4 서명 프로세스에 대한 자세한 내용은 AWS 일반 참조를 참조하세요.

이 장에서는 위치 데이터를 사용하는 애플리케이션에서 흔히 발생하는 많은 작업에 대해 설명합니다. 일반 사용 사례 섹션에서는 이러한 서비스를 AWS 다른 서비스와 결합하여 더 복잡한 사용 사례를 달성하는 방법을 설명합니다.

주제

- Amazon Location Service 사용을 위한 사전 조건
- 애플리케이션에서 Amazon Location Map 사용
- Amazon Location을 사용하여 장소 및 지리적 위치 데이터 검색
- Amazon Location Service를 사용하여 경로 계산
- Amazon Location을 사용하여 관심 영역을 지오펜싱하기
- Amazon Location Service 리소스 태그 지정
- Amazon Location Service에 액세스 권한 부여
- Amazon Location Service 모니터링
- AWS CloudFormation을 사용하여 Amazon Location Service 리소스 생성하기

# Amazon Location Service 사용을 위한 사전 조건

이 섹션에서는 Amazon Location Service 사용을 위해 수행해야 하는 사항을 설명합니다. AWS 계정 이 있어야 하며, Amazon Location을 사용하려는 사용자에 대한 Amazon Location 액세스 권한을 설정해야 합니다.

## 가입하기 AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

## 관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

## Amazon Location Service에 대한 액세스 권한 부여

관리자가 아닌 사용자에게는 기본적으로 권한이 없습니다. 사용자가 Amazon Location에 액세스하려면 먼저 특정 권한이 포함된 IAM 정책을 추가하여 권한을 부여해야 합니다. 리소스에 대한 액세스 권한을 부여할 때는 최소 권한의 원칙을 따릅니다.

> **Note**
>
> 인증되지 않은 사용자에게 Amazon Location Service 기능에 대한 액세스 권한을 부여하는 방법(예: 웹 기반 애플리케이션)에 대한 자세한 내용은 Amazon Location Service에 액세스 권한 부여 항목을 참조하세요.

다음 예시 정책은 사용자에게 모든 Amazon Location 작업에 액세스할 수 있는 권한을 부여합니다. 더 많은 예제는 Amazon Location Service의 자격 증명 기반 정책 예제를 참조합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "geo:*"
```

```
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 내 AWS IAM Identity Center사용자 및 그룹:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.

- ID 제공자를 통해 IAM에서 관리되는 사용자:

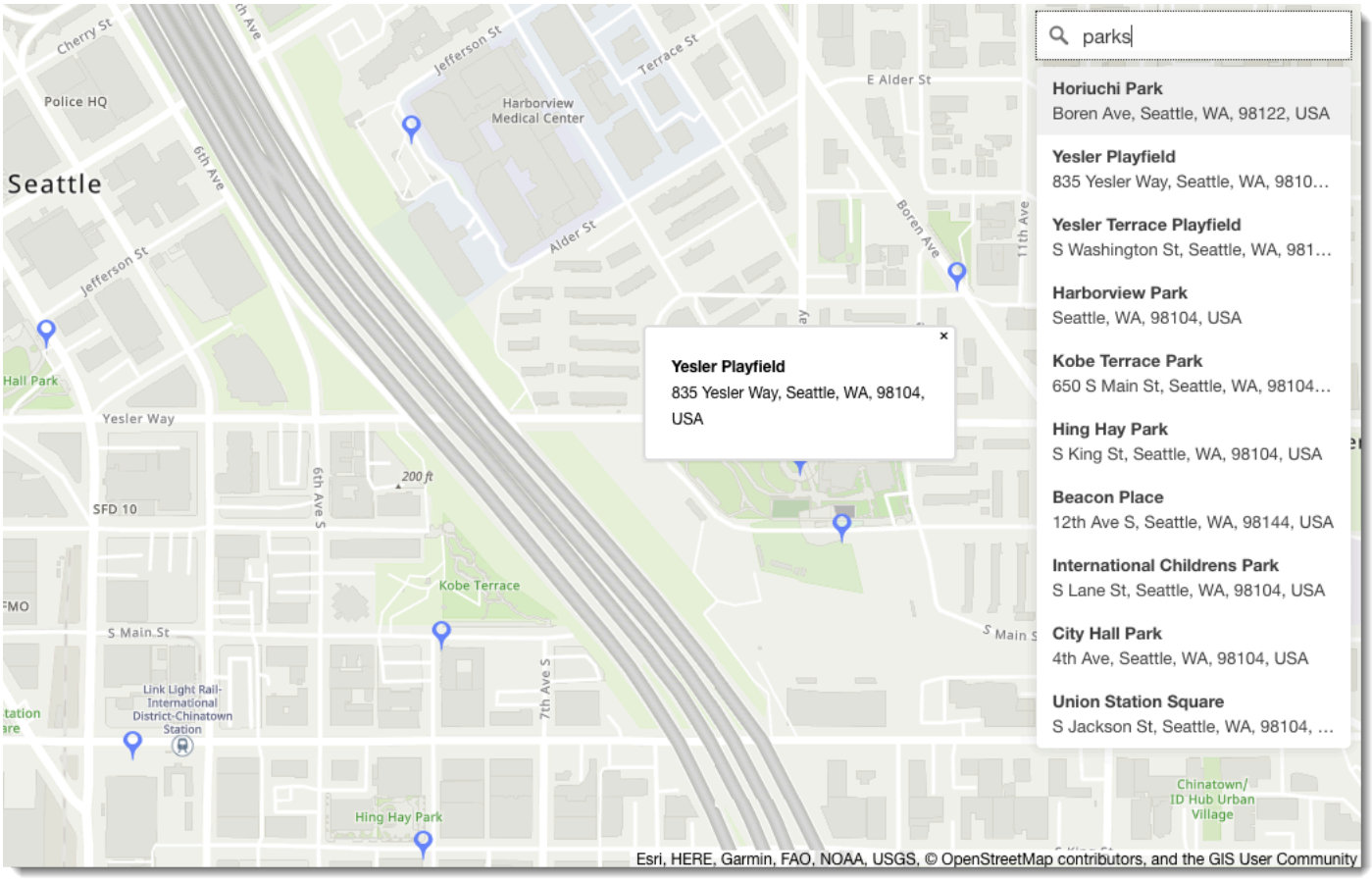
  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(연합)의 지침을 따르세요.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

Amazon Location Service를 사용하는 애플리케이션을 생성할 때 일부 사용자에게 인증되지 않은 액세스 권한이 필요할 수 있습니다. 이러한 사용 사례에 대해서는 Amazon Cognito를 사용한 인증되지 않은 액세스 활성화를 참조하세요.

# 애플리케이션에서 Amazon Location Map 사용

Amazon Location 맵은 비용 효율적이며 대화형입니다. 애플리케이션의 기존 맵을 교체하여 비용을 절감하거나 새 맵을 추가하여 스토어 위치와 같은 위치 기반 데이터를 시각적으로 표시할 수 있습니다.




Amazon Location Service를 사용하면 맵 리소스를 생성하고 구성하여 맵 작업에 사용할 데이터 공급자를 선택할 수 있습니다. 맵 리소스는 데이터 공급자와 맵을 렌더링하는 데 사용되는 스타일을 구성합니다.

리소스를 생성한 후 AWS SDK를 직접 사용하거나 사용자 환경에서 맵을 렌더링하기 위해 특별히 만들어진 라이브러리를 사용하여 요청을 보낼 수 있습니다.

> **Note**
>
> 맵 개념에 대한 개요는 맵 항목을 참고하세요.

주제

- 사전 조건
- 애플리케이션에 맵 표시하기
- 맵에 데이터 특성 그리기
- 를 사용하여 맵의 범위를 설정합니다. MapLibre

- 맵 리소스 관리

## 사전 조건

애플리케이션에 맵을 표시하기 전에 필수 단계를 따르세요.

주제

- 맵 리소스를 생성합니다.
- 요청 인증

### 맵 리소스를 생성합니다.

애플리케이션에서 맵을 사용하려면 맵에 사용할 맵 스타일과 데이터 공급자를 지정하는 맵 리소스가 있어야 합니다.

> **Note**
>
> 애플리케이션이 배송 차량 또는 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 Esri를 지리적 위치 제공업체로 사용해서는 안 됩니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

Amazon Location Service 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 맵 리소스를 생성할 수 있습니다.

Console

Amazon Location Service 콘솔을 사용하여 맵 리소스를 생성하려면

1. Amazon Location 콘솔의 맵 페이지에서 맵 생성을 선택하여 맵 스타일을 미리 볼 수 있습니다.
2. 새 맵 리소스의 이름과 설명을 추가합니다.
3. 맵 스타일을 선택합니다.

> **Note**
>
> 애플리케이션이 배송 차량 또는 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 Esri를 지리적 위치 제공업체로 사용해서는 안 됩니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

4. 사용할 정치적 관점를 선택합니다.
5. Amazon Location 이용 약관에 동의한 다음 맵 생성을 선택합니다. 선택한 맵과 상호 작용할 수 있습니다. 확대, 축소하거나 원하는 방향으로 이동할 수 있습니다.
6. 사용자가 스타일을 전환할 수 있도록 하려면(예: 위성 이미지와 벡터 스타일 사이를 전환할 수 있게 하려면) 각 스타일에 대한 맵 리소스를 생성해야 합니다.

   콘솔의 맵 홈페이지에서 사용하지 않으려는 맵 스타일이 포함된 리소스를 삭제할 수 있습니다.

API

Amazon Location API를 사용하여 맵 리소스를 만들려면

Amazon Location API에서 `CreateMap` 작업을 사용합니다.

다음은 맵 스타일을 *`ExampleMap`*사용하여 호출되는 맵 리소스를 생성하기 위한 API 요청입니다. *`VectorEsriStreets`*

```
POST /maps/v0/maps HTTP/1.1
Content-type: application/json

{
   "Configuration": {
      "Style": "VectorEsriStreets"
   },
   "MapName": "ExampleMap"
   }
}
```

> **Note**
>
> 애플리케이션이 배송 차량 또는 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 Esri를 지리적 위치 제공업체로 사용해서는 안 됩니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

AWS CLI

AWS CLI 명령을 사용하여 맵 리소스를 만들려면

`create-map` 명령을 사용합니다.

다음 예제에서는 using이라는 *ExampleMap*맵 리소스를 맵 스타일로 생성합니다. *VectorEsriStreets*

```
aws location \
  create-map \
  --configuration Style="VectorEsriStreets" \
  --map-name "ExampleMap"
```

> **Note**
>
> 애플리케이션이 배송 차량 또는 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 Esri를 지리적 위치 제공업체로 사용해서는 안 됩니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

## 요청 인증

맵 리소스를 생성하고 애플리케이션에 위치 기능을 빌드할 준비가 되면 요청을 인증할 방법을 선택해야 합니다.

> **Note**
>
> 대부분의 맵 프런트 엔드 애플리케이션에는 Amazon Location Service의 맵 또는 기타 기능에 대한 인증되지 않은 액세스가 필요합니다. 애플리케이션에 따라 AWS Signature v4를 사용하여 요청을 인증하거나 인증되지 않은 용도를 위해 Amazon Cognito 또는 Amazon Location API

키를 사용할 수 있습니다. 이러한 모든 옵션에 대한 자세한 내용은 Amazon Location Service에 액세스 권한 부여 단원을 참조하십시오.

## 애플리케이션에 맵 표시하기

이 섹션에서는 Amazon Location API를 사용할 때 맵 렌더링 도구를 사용하여 모바일 또는 웹 애플리케이션에서 맵을 표시하는 방법에 대한 튜토리얼을 제공합니다. 이 Amazon Location Service 사용 방법 주제에서 언급한 것처럼 Amazon Location을 사용하여 맵을 렌더링할 때 Amplify MapLibre, Tangram 등의 라이브러리를 선택할 수 있습니다.

애플리케이션에 맵을 표시하려면 다음 중 하나를 수행합니다.

- 웹 및 모바일 프런트 엔드 애플리케이션에서 지도를 표시하는 가장 직접적인 방법은 를 사용하는 것입니다. MapLibre MapLibre 자습서 또는 Quick Start 자습서를 따라 사용 MapLibre 방법을 배울 수 있습니다.
- 기존 AWS Amplify 개발자인 경우 Amplify Geo SDK를 사용하는 것이 좋습니다. 자세히 알아보려면 Amplify 튜토리얼을 따르세요.
- Tangram의 기존 사용자이고 Amazon Location Service로 이동하는 동안 계속해서 Tangram을 사용하여 맵을 렌더링하려면 Tangram 튜토리얼을 따르세요.

주제

- Amazon Location Service에서 MapLibre 라이브러리 사용하기
- Amazon Location Service로 Amplify 라이브러리 사용
- Amazon Location Service로 Tangram 사용

### Amazon Location Service에서 MapLibre 라이브러리 사용하기

다음 자습서는 Amazon Location에서 MapLibre 라이브러리를 사용하는 방법을 안내합니다.

주제

- Amazon Location Service와 함께 MapLibre GL JS 사용
- Amazon Location Service와 함께 안드로이드용 MapLibre 네이티브 SDK 사용
- Amazon Location Service와 함께 iOS용 MapLibre 네이티브 SDK 사용

### Amazon Location Service와 함께 MapLibre GL JS 사용

MapLibre GL JS를 사용하여 클라이언트측 맵을 웹 애플리케이션에 임베드할 수 있습니다.

MapLibre GL JS는 Amazon Location Service Maps API에서 제공하는 스타일 및 타일과 호환되는 오픈 소스 JavaScript 라이브러리입니다. MapLibre GL JS를 기본 HTML 또는 JavaScript 애플리케이션에 통합하여 사용자 지정이 가능하고 반응이 빠른 클라이언트측 지도를 내장할 수 있습니다.

이 자습서에서는 기본 HTML 및 JavaScript 애플리케이션 내에서 MapLibre GL JS를 Amazon Location과 통합하는 방법을 설명합니다. 이 튜토리얼에서 제공하는 것과 동일한 라이브러리 및 기법이 React와 Angular와 같은 프레임워크에도 적용됩니다.

이 자습서의 샘플 애플리케이션은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

#### 애플리케이션 빌드: 스캐폴딩

이 자습서에서는 HTML 페이지에 지도를 작성하는 JavaScript 데 사용하는 웹 애플리케이션을 생성합니다.

먼저 맵 컨테이너가 포함된 HTML 페이지(`index.html`)를 만듭니다.

- `map`의 `id`가 포함된 `div` 요소를 입력하여 맵의 크기를 맵 보기에 적용합니다. 크기는 뷰포트에서 상속됩니다.

```
<html>
  <head>
    <style>
      body {
        margin: 0;
      }

      #map {
        height: 100vh; /* 100% of viewport height */
      }
    </style>
  </head>
  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

애플리케이션 빌드: 종속성 추가

애플리케이션에 다음 종속 항목을 추가합니다.

- MapLibre GL JS (v3.x) 및 관련 CSS
- Amazon Location JavaScript 인증 도우미.

```
<!-- CSS dependencies -->
<link
  href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
  rel="stylesheet"
/>
<!-- JavaScript dependencies -->
<script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
<script src="https://unpkg.com/@aws/amazon-location-authentication-helper.js"></script>
<script>
  // application-specific code
</script>
```

이는 맵 컨테이너가 포함된 빈 페이지를 만듭니다.

애플리케이션 빌드: 구성

다음을 사용하여 애플리케이션을 구성하려면: JavaScript

1. 리소스의 이름과 식별자를 입력합니다.

   ```
   // Cognito Identity Pool ID
   const identityPoolId = "us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd";
   // Amazon Location Service Map name
   const mapName = "ExampleMap";
   ```

2. 맵 사용 - 2단계, 인증 설정에서 생성한 인증되지 않은 자격 증명 풀을 사용하여 보안 인증 정보 공급자를 인스턴스화합니다. 이를 `initializeMap`라는 함수에 넣습니다. 여기에는 다음 단계에서 추가된 다른 맵 초기화 코드도 포함됩니다.

   ```
   // extract the Region from the Identity Pool ID; this will be used for both Amazon
    Cognito and Amazon Location
   AWS.config.region = identityPoolId.split(":")[0];

   async function initializeMap() {
   ```

```
  // Create an authentication helper instance using credentials from Cognito
  const authHelper = await
 amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

  // ... more here, later
}
```

애플리케이션 빌드: 맵 초기화

페이지가 로드된 후 맵을 표시하려면 맵을 초기화해야 합니다. 초기 맵 위치를 조정하고, 컨트롤을 추가하고, 데이터를 오버레이할 수 있습니다.

```
async function initializeMap() {
  // Create an authentication helper instance using credentials from Cognito
  const authHelper = await amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

  // Initialize the map
  const map = new maplibregl.Map({
    container: "map",
    center: [-123.1187, 49.2819], // initial map centerpoint
    zoom: 10, // initial map zoom
    style: 'https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor',
    ...authHelper.getMapAuthenticationOptions(), // authentication, using cognito
  });

  map.addControl(new maplibregl.NavigationControl(), "top-left");
}

initializeMap();
```

> **Note**
>
> 애플리케이션 또는 문서에서 사용하는 각 데이터 공급자에 대한 워드마크 또는 텍스트 속성을 제공해야 합니다. 어트리뷰션 문자열은 스타일 디스크립터 응답의`sources.esri.attribution`,`sources.here.attribution`, 키 아래에 포함됩니다. `sources.grabmaptiles.attribution` MapLibre GL JS는 자동으로 어트리뷰션을 제공합니다. 데이터 공급자와 함께 Amazon Location 리소스를 사용할 때는 서비스 이용 약관을 반드시 읽어보세요.

애플리케이션 실행

이 샘플 애플리케이션을 로컬 웹 서버에서 사용하거나 브라우저에서 열어 실행할 수 있습니다.

로컬 웹 서버를 사용하려면 npx를 사용할 수 있습니다. npx는 Node.js 일부로 설치되기 때문입니다. `index.html`와 동일한 디렉터리 내에서 `npx serve`를 사용할 수 있습니다. 이는 `localhost:5000`에서 애플리케이션을 제공합니다.

> **Note**
>
> 인증되지 않은 Amazon Cognito 역할에 대해 생성한 정책에 `referer` 조건이 포함된 경우 `localhost:` URL을 사용한 테스트가 차단될 수 있습니다. 이 경우 정책에 포함된 URL을 제공하는 웹 서버로 테스트할 수 있습니다.

튜토리얼을 완료한 후 최종 애플리케이션은 다음 예시와 같습니다.

```
<!-- index.html -->
<html>
  <head>
    <link href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
 rel="stylesheet" />
    <style>
      body {
        margin: 0;
      }
      #map {
        height: 100vh;
      }
    </style>
  </head>

  <body>
    <!-- map container -->
    <div id="map" />
    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-authentication-helper.js"></
script>
    <script>
      // configuration
```

```
      const identityPoolId = "us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd"; //
 Cognito Identity Pool ID
      const mapName = "ExampleMap"; // Amazon Location Service Map Name

      // extract the region from the Identity Pool ID
      const region = identityPoolId.split(":")[0];

      async function initializeMap() {
        // Create an authentication helper instance using credentials from Cognito
        const authHelper = await
 amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

        // Initialize the map
        const map = new maplibregl.Map({
          container: "map",
          center: [-123.115898, 49.295868],
          zoom: 10,
          style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/
style-descriptor`,
          ...authHelper.getMapAuthenticationOptions(),
        });
        map.addControl(new maplibregl.NavigationControl(), "top-left");
      }

      initializeMap();
    </script>
  </body>
</html>
```

이 애플리케이션을 실행하면 선택한 맵 스타일을 사용하여 전체 화면 맵이 표시됩니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리에서 사용할 수 GitHub있습니다.

Amazon Location Service와 함께 안드로이드용 MapLibre 네이티브 SDK 사용

MapLibre네이티브 SDK를 사용하여 Android 애플리케이션에 대화형 맵을 임베드할 수 있습니다.

Android용 MapLibre 네이티브 SDK는 맵박스 네이티브 기반의 라이브러리로, Amazon Location Service Maps API에서 제공하는 스타일 및 타일과 호환됩니다. Android용 MapLibre Native SDK를 통합하여 Android 애플리케이션에 확장 가능하고 사용자 지정이 가능한 벡터 맵과 대화형 맵 뷰를 내장할 수 있습니다.

이 자습서에서는 Android용 MapLibre 네이티브 SDK를 Amazon Location과 통합하는 방법을 설명합니다. 이 자습서의 샘플 애플리케이션은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

애플리케이션 빌드: 초기화

애플리케이션을 초기화하려면

1. 빈 활동 템플릿에서 새 Android 스튜디오 프로젝트를 만듭니다.
2. 프로젝트 언어로 Kotlin이 선택되었는지 확인합니다.
3. API 14의 최소 SDK: Android 4.0 (Ice Cream Sandwich) 이상을 선택합니다.
4. 프로젝트 구조를 연 다음 파일 > 프로젝트 구조...로 이동하여 종속성 섹션을 선택합니다.
5. <All Modules>를 선택한 상태에서 + 버튼을 선택하여 새 라이브러리 종속성을 추가합니다.
6. AWS Android SDK 버전 2.20.0 이상을 추가합니다. 예: `com.amazonaws:aws-android-sdk-core:2.20.0`
7. Android용 MapLibre 네이티브 SDK 버전 9.4.0 이상을 추가합니다. 예: `org.maplibre.gl:android-sdk:9.4.0`
8. build.gradle 파일의 프로젝트 수준에서 다음과 같은 maven 저장소를 추가하여 안드로이드용 패키지에 액세스하세요. MapLibre

```
allprojects {
    repositories {
        // Retain your existing repositories
        google()
        jcenter()

        //  Declare the repositories for MapLibre
        mavenCentral()
    }
}
```

애플리케이션 빌드: 구성

리소스 및 AWS 리전을 사용하여 애플리케이션을 구성하려면

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
    <string name="identityPoolId">us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd</
string>
    <string name="mapName">ExampleMap</string>
    <string name="awsRegion">us-east-1</string>
</resources>
```

애플리케이션 빌드: 활동 레이아웃

편집 app/src/main/res/layout/activity_main.xml:

- 맵을 렌더링하는 MapView를 추가합니다. 이렇게 하면 맵의 초기 중심점도 설정됩니다.
- 속성을 표시하는 TextView를 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.mapbox.mapboxsdk.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:mapbox_cameraTargetLat="49.2819"
        app:mapbox_cameraTargetLng="-123.1187"
        app:mapbox_cameraZoom="12"
        app:mapbox_uiAttribution="false"
        app:mapbox_uiLogo="false" />

    <TextView
        android:id="@+id/attributionView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#80808080"
        android:padding="5sp"
        android:textColor="@android:color/black"
        android:textSize="10sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        tools:ignore="SmallSp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

> **Note**
>
> 애플리케이션 또는 문서에서 사용하는 각 데이터 공급자에 대한 워드마크 또는 텍스트 속성을 제공해야 합니다. 속성 문자열은 `sources.esri.attribution`, `sources.here.attribution`, `source.grabmaptiles.attribution` 키의 스타일 설명자 응답에 포함됩니다. 데이터 공급자와 함께 Amazon Location 리소스를 사용할 때는 서비스 이용 약관을 반드시 읽어보세요.

애플리케이션 빌드: 변환 요청

AWS 요청을 가로채는 이름이 `SigV4Interceptor`인 클래스를 만들고 Signature Version 4를 사용하여 서명합니다. 이는 기본 활동이 생성될 때 맵 리소스를 가져오는 데 사용되는 HTTP 클라이언트에 등록됩니다.

```
package aws.location.demo.okhttp

import com.amazonaws.DefaultRequest
import com.amazonaws.auth.AWS4Signer
import com.amazonaws.auth.AWSCredentialsProvider
import com.amazonaws.http.HttpMethodName
import com.amazonaws.util.IOUtils
import okhttp3.HttpUrl
import okhttp3.Interceptor
import okhttp3.Request
import okhttp3.Response
import okio.Buffer
import java.io.ByteArrayInputStream
import java.net.URI

class SigV4Interceptor(
    private val credentialsProvider: AWSCredentialsProvider,
    private val serviceName: String
) : Interceptor {
    override fun intercept(chain: Interceptor.Chain): Response {
        val originalRequest = chain.request()

        if (originalRequest.url().host().contains("amazonaws.com")) {
```

```
            val signer = if (originalRequest.url().encodedPath().contains("@")) {
                // the presence of "@" indicates that it doesn't need to be double URL-
encoded
                AWS4Signer(false)
            } else {
                AWS4Signer()
            }

            val awsRequest = toAWSRequest(originalRequest, serviceName)
            signer.setServiceName(serviceName)
            signer.sign(awsRequest, credentialsProvider.credentials)

            return chain.proceed(toSignedOkHttpRequest(awsRequest, originalRequest))
        }

        return chain.proceed(originalRequest)
    }

    companion object {
        fun toAWSRequest(request: Request, serviceName: String): DefaultRequest<Any> {
            // clone the request (AWS-style) so that it can be populated with
 credentials
            val dr = DefaultRequest<Any>(serviceName)

            // copy request info
            dr.httpMethod = HttpMethodName.valueOf(request.method())
            with(request.url()) {
                dr.resourcePath = uri().path
                dr.endpoint = URI.create("${scheme()}://${host()}")

                // copy parameters
                for (p in queryParameterNames()) {
                    if (p != "") {
                        dr.addParameter(p, queryParameter(p))
                    }
                }
            }

            // copy headers
            for (h in request.headers().names()) {
                dr.addHeader(h, request.header(h))
            }

            // copy the request body
```

```
            val bodyBytes = request.body()?.let { body ->
                val buffer = Buffer()
                body.writeTo(buffer)
                IOUtils.toByteArray(buffer.inputStream())
            }

            dr.content = ByteArrayInputStream(bodyBytes ?: ByteArray(0))

            return dr
        }

        fun toSignedOkHttpRequest(
            awsRequest: DefaultRequest<Any>,
            originalRequest: Request
        ): Request {
            // copy signed request back into an OkHttp Request
            val builder = Request.Builder()

            // copy headers from the signed request
            for ((k, v) in awsRequest.headers) {
                builder.addHeader(k, v)
            }

            // start building an HttpUrl
            val urlBuilder = HttpUrl.Builder()
                .host(awsRequest.endpoint.host)
                .scheme(awsRequest.endpoint.scheme)
                .encodedPath(awsRequest.resourcePath)

            // copy parameters from the signed request
            for ((k, v) in awsRequest.parameters) {
                urlBuilder.addQueryParameter(k, v)
            }

            return builder.url(urlBuilder.build())
                .method(originalRequest.method(), originalRequest.body())
                .build()
        }
    }
}
```

애플리케이션 빌드: 기본 활동

기본 활동은 사용자에게 표시될 보기를 초기화하는 역할을 담당합니다. 여기에는 다음이 포함됩니다.

- Amazon Cognito `CredentialsProvider` 인스턴트화.
- Signature Version 4 인터셉터 등록.
- 맵 스타일 설명자를 가리키고 적절한 속성을 표시하여 맵을 구성합니다.

`MainActivity`는 또한 맵 보기에 수명 주기 이벤트를 전달하여 호출 사이에도 활성 뷰포트를 유지할 수 있도록 합니다.

```
package aws.location.demo.maplibre

import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import aws.location.demo.okhttp.SigV4Interceptor
import com.amazonaws.auth.CognitoCachingCredentialsProvider
import com.amazonaws.regions.Regions
import com.mapbox.mapboxsdk.Mapbox
import com.mapbox.mapboxsdk.maps.MapView
import com.mapbox.mapboxsdk.maps.Style
import com.mapbox.mapboxsdk.module.http.HttpRequestUtil
import okhttp3.OkHttpClient

private const val SERVICE_NAME = "geo"

class MainActivity : AppCompatActivity() {
    private var mapView: MapView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // configuration
        val identityPoolId = getString(R.string.identityPoolId)
        val region = getString(R.string.awsRegion)
        val mapName = getString(R.string.mapName)

        // Credential initialization
        val credentialProvider = CognitoCachingCredentialsProvider(
            applicationContext,
            identityPoolId,
            Regions.fromName(identityPoolId.split(":").first())
        )

        // initialize MapLibre
```

```
        Mapbox.getInstance(this, null)
        HttpRequestUtil.setOkHttpClient(
            OkHttpClient.Builder()
                .addInterceptor(SigV4Interceptor(credentialProvider, SERVICE_NAME))
                .build()
        )

        // initialize the view
        setContentView(R.layout.activity_main)

        // initialize the map view
        mapView = findViewById(R.id.mapView)
        mapView?.onCreate(savedInstanceState)
        mapView?.getMapAsync { map ->
            map.setStyle(
                Style.Builder()
                    .fromUri("https://maps.geo.${region}.amazonaws.com/maps/v0/maps/
${mapName}/style-descriptor")
            ) { style ->
                findViewById<TextView>(R.id.attributionView).text =
 style.sources.first()?.attribution
            }
        }
    }

    override fun onStart() {
        super.onStart()
        mapView?.onStart()
    }

    override fun onResume() {
        super.onResume()
        mapView?.onResume()
    }

    override fun onPause() {
        super.onPause()
        mapView?.onPause()
    }

    override fun onStop() {
        super.onStop()
        mapView?.onStop()
    }
```

```
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        mapView?.onSaveInstanceState(outState)
    }

    override fun onLowMemory() {
        super.onLowMemory()
        mapView?.onLowMemory()
    }

    override fun onDestroy() {
        super.onDestroy()
        mapView?.onDestroy()
    }
}
```

이 애플리케이션을 실행하면 선택한 스타일로 전체 화면 맵이 표시됩니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

Amazon Location Service와 함께 iOS용 MapLibre 네이티브 SDK 사용

iOS용 MapLibre 네이티브 SDK를 사용하여 클라이언트측 맵을 iOS 애플리케이션에 임베드할 수 있습니다.

iOS용 MapLibre 네이티브 SDK는 맵박스 GL 네이티브를 기반으로 하는 라이브러리로, Amazon Location Service Maps API에서 제공하는 스타일 및 타일과 호환됩니다. iOS용 MapLibre Native SDK를 통합하여 확장 가능하고 사용자 정의 가능한 벡터 맵이 있는 대화형 맵 뷰를 iOS 애플리케이션에 임베드할 수 있습니다.

이 자습서에서는 iOS용 MapLibre 네이티브 SDK를 Amazon Location과 통합하는 방법을 설명합니다. 이 자습서의 샘플 애플리케이션은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

애플리케이션 빌드: 초기화

애플리케이션을 초기화하려면

1. 앱 템플릿에서 새 Xcode 프로젝트를 생성합니다
2. 인터페이스로 SwiftUI를 선택합니다.
3. 수명 주기로 SwiftUI 애플리케이션을 선택합니다.

4. 해당 언어로 Swift를 선택합니다.

Swift 패키지를 사용하여 MapLibre 종속성 추가

Xcode 프로젝트에 패키지 종속성을 추가하려면

1. 파일 > Swift 패키지 > 패키지 종속성 추가로 이동합니다.
2. 다음 리포지토리 URL을 입력합니다. **`https://github.com/maplibre/maplibre-gl-native-distribution`**

> **Note**
>
> Swift 패키지에 대한 자세한 내용은 Apple.com에서 Adding Package Dependencies to Your App 항목을 참조하세요.

3. 터미널에 CocoaPods 다음을 설치하세요.

```
sudo gem install cocoapods
```

4. 애플리케이션의 프로젝트 디렉토리로 이동하여 CocoaPods 패키지 관리자를 사용하여 Podfile을 초기화하십시오.

```
pod init
```

5. Podfile을 열어 `AWSCore`을 종속성으로 추가합니다.

```
platform :ios, '12.0'

target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
end
```

6. 종속성 다운로드 및 설치:

```
pod install --repo-update
```

7. 다음을 생성한 Xcode 워크스페이스를 엽니다. CocoaPods

```
xed .
```

애플리케이션 빌드: 구성

Info.plist에 다음 키와 값을 추가하여 애플리케이션을 구성합니다.

| 키 | 값 |
| --- | --- |
| AWSRegion | us-east-1 |
| IdentityPoolId | us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd |
| MapName | ExampleMap |

애플리케이션 빌드: 레이아웃 ContentView

맵을 렌더링하려면 `ContentView.swift`을 편집합니다.

- 맵을 렌더링하는 `MapView`를 추가합니다.
- 속성을 표시하는 `TextField`를 추가합니다.

이렇게 하면 맵의 초기 중심점도 설정됩니다.

```
import SwiftUI

struct ContentView: View {
    @State private var attribution = ""

    var body: some View {
        MapView(attribution: $attribution)
            .centerCoordinate(.init(latitude: 49.2819, longitude: -123.1187))
            .zoomLevel(12)
            .edgesIgnoringSafeArea(.all)
            .overlay(
                TextField("", text: $attribution)
                    .disabled(true)
                    .font(.system(size: 12, weight: .light, design: .default))
```

```
                    .foregroundColor(.black)
                    .background(Color.init(Color.RGBColorSpace.sRGB, white: 0.5,
 opacity: 0.5))
                    .cornerRadius(1),
                alignment: .bottomTrailing)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

> **Note**
>
> 애플리케이션 또는 문서에서 사용하는 각 데이터 공급자에 대한 워드마크 또는 텍스트 속성을 제공해야 합니다. 속성 문자열은 `sources.esri.attribution`, `sources.here.attribution`, `source.grabmaptiles.attribution` 키의 스타일 설명자 응답에 포함됩니다. 데이터 공급자와 함께 Amazon Location 리소스를 사용할 때는 서비스 이용 약관을 반드시 읽어보세요.

애플리케이션 빌드: 변환 요청

다음 클래스 정의가 포함된 이름이 `AWSSignatureV4Delegate.swift`인 새 Swift 파일을 생성하여 AWS 요청을 가로채고 Signature Version 4를 사용하여 서명합니다. 이 클래스의 인스턴스는 맵 보기에서 URL 재작성도 담당하는 오프라인 스토리지 대리자로 할당됩니다.

```
import AWSCore
import Mapbox

class AWSSignatureV4Delegate : NSObject, MGLOfflineStorageDelegate {
    private let region: AWSRegionType
    private let identityPoolId: String
    private let credentialsProvider: AWSCredentialsProvider

    init(region: AWSRegionType, identityPoolId: String) {
        self.region = region
        self.identityPoolId = identityPoolId
```

```
        self.credentialsProvider = AWSCognitoCredentialsProvider(regionType: region,
 identityPoolId: identityPoolId)
        super.init()
    }

    class func doubleEncode(path: String) -> String? {
        return path.addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)?
            .addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)
    }

    func offlineStorage(_ storage: MGLOfflineStorage, urlForResourceOf kind:
 MGLResourceKind, with url: URL) -> URL {
        if url.host?.contains("amazonaws.com") != true {
            // not an AWS URL
            return url
        }

        // URL-encode spaces, etc.
        let keyPath = String(url.path.dropFirst())
        guard let percentEncodedKeyPath =
 keyPath.addingPercentEncoding(withAllowedCharacters: .urlPathAllowed) else {
            print("Invalid characters in path '\(keyPath)'; unsafe to sign")
            return url
        }

        let endpoint = AWSEndpoint(region: region, serviceName: "geo", url: url)
        let requestHeaders: [String: String] = ["host": endpoint!.hostName]

        // sign the URL
        let task = AWSSignatureV4Signer
            .generateQueryStringForSignatureV4(
                withCredentialProvider: credentialsProvider,
                httpMethod: .GET,
                expireDuration: 60,
                endpoint: endpoint!,
                // workaround for https://github.com/aws-amplify/aws-sdk-ios/
issues/3215
                keyPath: AWSSignatureV4Delegate.doubleEncode(path:
 percentEncodedKeyPath),
                requestHeaders: requestHeaders,
                requestParameters: .none,
                signBody: true)
        task.waitUntilFinished()
```

```
        if let error = task.error as NSError? {
            print("Error occurred: \(error)")
        }

        if let result = task.result {
            var urlComponents = URLComponents(url: (result as URL),
 resolvingAgainstBaseURL: false)!
            // re-use the original path; workaround for https://github.com/aws-amplify/
aws-sdk-ios/issues/3215
            urlComponents.path = url.path

            // have Mapbox GL fetch the signed URL
            return (urlComponents.url)!
        }

        // fall back to an unsigned URL
        return url
    }
}
```

애플리케이션 빌드: 맵 보기

맵 보기는 AWSSignatureV4Delegate 인스턴스를 초기화하고, 리소스를 가져오고 맵을 렌더링하는 기본 MGLMapView 구성을 담당합니다. 또한 스타일 설명자 소스에서 속성 문자열을 다시 ContentView로 전파되는 것도 처리합니다.

다음 MapView.swift 정의를 포함하는 struct라는 이름을 가진 새 Swift 파일을 생성합니다.

```
import SwiftUI
import AWSCore
import Mapbox

struct MapView: UIViewRepresentable {
    @Binding var attribution: String

    private var mapView: MGLMapView
    private var signingDelegate: MGLOfflineStorageDelegate

    init(attribution: Binding<String>) {
        let regionName = Bundle.main.object(forInfoDictionaryKey: "AWSRegion") as!
 String
        let identityPoolId = Bundle.main.object(forInfoDictionaryKey: "IdentityPoolId")
 as! String
```

```
        let mapName = Bundle.main.object(forInfoDictionaryKey: "MapName") as! String

        let region = (regionName as NSString).aws_regionTypeValue()

        // MGLOfflineStorage doesn't take ownership, so this needs to be a member here
        signingDelegate = AWSSignatureV4Delegate(region: region, identityPoolId:
 identityPoolId)

        // register a delegate that will handle SigV4 signing
        MGLOfflineStorage.shared.delegate = signingDelegate

        mapView = MGLMapView(
            frame: .zero,
            styleURL: URL(string: "https://maps.geo.\(regionName).amazonaws.com/maps/
v0/maps/\(mapName)/style-descriptor"))

        _attribution = attribution
    }

    func makeCoordinator() -> Coordinator {
        Coordinator($attribution)
    }

    class Coordinator: NSObject, MGLMapViewDelegate {
        var attribution: Binding<String>

        init(_ attribution: Binding<String>) {
            self.attribution = attribution
        }

        func mapView(_ mapView: MGLMapView, didFinishLoading style: MGLStyle) {
            let source = style.sources.first as? MGLVectorTileSource
            let attribution = source?.attributionInfos.first
            self.attribution.wrappedValue = attribution?.title.string ?? ""
        }
    }

    // MARK: - UIViewRepresentable protocol

    func makeUIView(context: UIViewRepresentableContext<MapView>) -> MGLMapView {
        mapView.delegate = context.coordinator

        mapView.logoView.isHidden = true
        mapView.attributionButton.isHidden = true
```

```
        return mapView
    }

    func updateUIView(_ uiView: MGLMapView, context:
 UIViewRepresentableContext<MapView>) {
    }

    // MARK: - MGLMapView proxy

    func centerCoordinate(_ centerCoordinate: CLLocationCoordinate2D) -> MapView {
        mapView.centerCoordinate = centerCoordinate
        return self
    }

    func zoomLevel(_ zoomLevel: Double) -> MapView {
        mapView.zoomLevel = zoomLevel
        return self
    }
}
```

이 애플리케이션을 실행하면 선택한 스타일로 전체 화면 맵이 표시됩니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

## Amazon Location Service로 Amplify 라이브러리 사용

다음 튜토리얼에서는 Amazon Location으로 AWS Amplify를 사용하는 방법을 안내합니다. Amplify는 MapLibre GL JS를 사용하여 기반 애플리케이션에서 맵을 렌더링합니다. JavaScript

Amplify는 Amazon Location Service로 구동되는 Amplify Geo를 비롯한 다양한 서비스 카테고리에 대한 인터페이스를 제공하는 오픈 소스 클라이언트 라이브러리 세트입니다. AWS Amplify JavaScript Geo 라이브러리에 대해 자세히 알아보십시오.

> **Note**
>
> 이 튜토리얼에서는 맵 사용 - 애플리케이션에 맵 추가의 단계를 이미 수행했다고 가정합니다.

### 애플리케이션 빌드: 스캐폴딩

이 자습서에서는 HTML 페이지에 맵을 작성하는 JavaScript 데 사용하는 웹 애플리케이션을 생성합니다.

먼저 맵 컨테이너가 포함된 HTML 페이지(index.html)를 만듭니다.

- map의 id가 포함된 div 요소를 입력하여 맵의 크기를 맵 보기에 적용합니다. 크기는 뷰포트에서 상속됩니다.

```
<html>
  <head>
    <style>
      body { margin: 0; }
      #map { height: 100vh; } /* 100% of viewport height */
    </style>
  </head>

  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

애플리케이션 빌드: 종속성 추가

애플리케이션에 다음 종속 항목을 추가합니다.

- AWS Amplify 맵 및 지리 라이브러리.
- AWS Amplify 코어 라이브러리.
- AWS Amplify 인증 라이브러리.
- AWS Amplify 스타일시트.

```
<!-- CSS dependencies -->
    <link href="https://cdn.amplify.aws/packages/maplibre-
gl/1.15.2/maplibre-gl.css" rel="stylesheet" integrity="sha384-
DrPVD9GufrxGb7kWwRv0CywpXTmfvbKOZ5i5pN7urmIThew0zXKTME+gutUgtpeD"
 crossorigin="anonymous" referrerpolicy="no-referrer"></link>

<!-- JavaScript dependencies -->
    <script src="https://cdn.amplify.aws/packages/maplibre-gl/1.15.2/maplibre-gl.js"
 integrity="sha384-rwYfkmAOpciZS2bDuwZ/Xa/Gog6jXem8D/whm3wnsZSVFemDDlprcUXHnDDUcrNU"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
```

```
    <script src="https://cdn.amplify.aws/packages/core/4.3.0/aws-amplify-core.min.js"
 integrity="sha384-7Oh+5w0l7XGyYvSqbKi2Q7SA5K640V5nyW2/LEbevDQEV1HMJqJLA1A00z2hu8fJ"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/auth/4.3.8/aws-amplify-auth.min.js"
 integrity="sha384-jfkXCEfYyVmDXYKlgWNwv54xRaZgk14m7sjeb2jLVBtUXCD2p+WU8YZ2mPZ9Xbdw"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/geo/1.1.0/aws-amplify-geo.min.js"
 integrity="sha384-TFMTyWuCbiptXTzvOgzJbV8TPUupG1rA1AVrznAhCSpXTIdGw82bGd8RTk5rr3nP"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/maplibre-gl-js-
amplify/1.1.0/maplibre-gl-js-amplify.umd.min.js" integrity="sha384-7/
RxWonKW1nM9zCKiwU9x6bkQTjldosg0D1vZYm0Zj+K/vUSnA3sOMhlRRWAtHPi" crossorigin="anonymous"
 referrerpolicy="no-referrer"></script>
<script>
  // application-specific code
</script>
```

이는 맵 컨테이너가 포함된 빈 페이지를 만듭니다.

애플리케이션 빌드: 구성

다음을 사용하여 애플리케이션을 구성하려면 JavaScript:

1. 맵 사용 - 2단계, 인증 설정에 생성한 인증되지 않은 자격 증명 풀의 식별자를 입력합니다.

   ```
   // Cognito Identity Pool ID
   const identityPoolId = "region:identityPoolID"; // for example: us-
   east-1:123example-1234-5678
   // extract the Region from the Identity Pool ID
   const region = identityPoolId.split(":")[0];
   ```

2. 자격 증명 풀 및 맵 리소스(여기서는 기본 이름 AWS Amplify으로 표시됨)를 포함하여 생성한 리소스를 사용하도록 `explore.map`를 구성합니다.

   ```
   // Configure Amplify
   const { Amplify } = aws_amplify_core;
   const { createMap } = AmplifyMapLibre;

   Amplify.configure({
     Auth: {
       identityPoolId,
       region,
     },
   ```

```
  geo: {
    AmazonLocationService: {
      maps: {
        items: {
          "explore.map": {
            style: "Default style"
          },
        },
        default: "explore.map",
      },
      region,
    },
  }
});
```

애플리케이션 빌드: 맵 초기화

페이지가 로드된 후 맵을 표시하려면 맵을 초기화해야 합니다. 초기 맵 위치를 조정하고, 컨트롤을 추가하고, 데이터를 오버레이할 수 있습니다.

```
async function initializeMap() {
  const map = await createMap(
    {
      container: "map",
      center: [-123.1187, 49.2819],
      zoom: 10,
      hash: true,
    }
  );

  map.addControl(new maplibregl.NavigationControl(), "top-left");
}

initializeMap();
```

> **Note**
>
> 애플리케이션 또는 문서에서 사용하는 각 데이터 공급자에 대한 워드마크 또는 텍스트 속성을 제공해야 합니다. 속성 문자열은 `sources.esri.attribution`, `sources.here.attribution`, `sources.grabmaptiles.attribution` 키의 스타일

설명자 응답에 포함됩니다. Amplify는 자동으로 속성을 제공합니다. 데이터 공급자와 함께 Amazon Location 리소스를 사용할 때는 서비스 이용 약관을 반드시 읽어보세요.

애플리케이션 실행

이 샘플 애플리케이션을 로컬 웹 서버에서 사용하거나 브라우저에서 열어 실행할 수 있습니다.

로컬 웹 서버를 사용하려면 Node.js 일부로 설치된 npx 또는 원하는 다른 웹 서버를 사용할 수 있습니다. npx를 사용하려면 `index.html`과 동일한 디렉터리 내에서 `npx serve`를 입력합니다. 이는 `localhost:5000`에서 애플리케이션을 제공합니다.

> **Note**
>
> 인증되지 않은 Amazon Cognito 역할에 대해 생성한 정책에 `referer` 조건이 포함된 경우 `localhost:` URL을 사용한 테스트가 차단될 수 있습니다. 이 경우 정책에 포함된 URL을 제공하는 웹 서버로 테스트할 수 있습니다.

튜토리얼을 완료한 후 최종 애플리케이션은 다음 예시와 같습니다.

```
<html>
  <head>
    <!-- CSS dependencies -->
    <link href="https://cdn.amplify.aws/packages/maplibre-
gl/1.15.2/maplibre-gl.css" rel="stylesheet" integrity="sha384-
DrPVD9GufrxGb7kWwRv0CywpXTmfvbKOZ5i5pN7urmIThew0zXKTME+gutUgtpeD"
 crossorigin="anonymous" referrerpolicy="no-referrer"></link>

    <!-- JavaScript dependencies -->
    <script src="https://cdn.amplify.aws/packages/maplibre-gl/1.15.2/maplibre-gl.js"
 integrity="sha384-rwYfkmAOpciZS2bDuwZ/Xa/Gog6jXem8D/whm3wnsZSVFemDDlprcUXHnDDUcrNU"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/core/4.3.0/aws-amplify-core.min.js"
 integrity="sha384-7Oh+5w0l7XGyYvSqbKi2Q7SA5K640V5nyW2/LEbevDQEV1HMJqJLA1A00z2hu8fJ"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/auth/4.3.8/aws-amplify-auth.min.js"
 integrity="sha384-jfkXCEfYyVmDXYKlgWNwv54xRaZgk14m7sjeb2jLVBtUXCD2p+WU8YZ2mPZ9Xbdw"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/geo/1.1.0/aws-amplify-geo.min.js"
 integrity="sha384-TFMTyWuCbiptXTzvOgzJbV8TPUupG1rA1AVrznAhCSpXTIdGw82bGd8RTk5rr3nP"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
```

```
    <script src="https://cdn.amplify.aws/packages/maplibre-gl-js-
amplify/1.1.0/maplibre-gl-js-amplify.umd.min.js" integrity="sha384-7/
RxWonKW1nM9zCKiwU9x6bkQTjldosg0D1vZYm0Zj+K/vUSnA3sOMhlRRWAtHPi" crossorigin="anonymous"
 referrerpolicy="no-referrer"></script>

    <style>
      body { margin: 0; }
      #map { height: 100vh; }
    </style>
  </head>

  <body>
    <div id="map" />
    <script type="module">
      // Cognito Identity Pool ID
      const identityPoolId = "region:identityPoolId"; // for example: us-
east-1:123example-1234-5678
      // extract the Region from the Identity Pool ID
      const region = identityPoolId.split(":")[0];

      // Configure Amplify
      const { Amplify } = aws_amplify_core;
      const { createMap } = AmplifyMapLibre;

      Amplify.configure({
        Auth: {
          identityPoolId,
          region,
        },
        geo: {
          AmazonLocationService: {
            maps: {
              items: {
                "explore.map": {
                  style: "Default style"
                },
              },
              default: "explore.map",
            },
            region,
          },
        }
      });
```

```
      async function initializeMap() {
        const map = await createMap(
          {
            container: "map",
            center: [-123.1187, 49.2819],
            zoom: 10,
            hash: true,
          }
        );

        map.addControl(new maplibregl.NavigationControl(), "top-left");
      }

      initializeMap();
    </script>
  </body>
</html>
```

이 애플리케이션을 실행하면 선택한 맵 스타일을 사용하여 전체 화면 맵이 표시됩니다. 이 샘플은 Amazon Location Service 콘솔의 맵 리소스 페이지에 있는 맵 임베드 탭에도 설명되어 있습니다.

이 튜토리얼을 완료한 후 AWS Amplify 설명서의 맵 표시 항목으로 이동하여 맵에 마커를 표시하는 방법을 비롯한 자세한 내용을 알아보세요.

## Amazon Location Service로 Tangram 사용

이 섹션에서는 Tangram을 Amazon Location과 통합하는 방법에 대한 다음 튜토리얼을 제공합니다.

> **Important**
>
> 다음 튜토리얼의 Tangram 스타일은 `VectorHereContrast` 스타일로 구성된 Amazon Location 맵 리소스와만 호환됩니다.

다음은 스타일을 *`TangramExampleMap`*사용하여 호출되는 새 맵 리소스를 생성하는 AWS CLI 명령의 *`VectorHereContrast`*예입니다.

```
aws --region us-east-1 \
  location \
  create-map \
  --map-name "TangramExampleMap" \
  --configuration "Style=VectorHereContrast"
```

> **Note**
>
> 요금은 사용량에 따라 결정됩니다. 다른 AWS 서비스 사용 시 요금이 부과될 수 있습니다. 자세한 정보는 Amazon Location Service 가격을 참조하세요.

주제

- Amazon Location Service로 Tangram 사용
- Amazon Location Service로 Android용 Tangram ES 사용
- Amazon Location Service로 iOS용 Tangram ES 사용

## Amazon Location Service로 Tangram 사용

Tangram은 벡터 타일에서 2D 및 3D 맵을 실시간으로 렌더링하도록 설계된 유연한 매핑 엔진입니다. Mapzen 설계 스타일 및 Amazon Location Service Maps API에서 제공하는 HERE 타일과 함께 사용할 수 있습니다. 이 가이드에서는 기본 HTML/ JavaScript 애플리케이션 내에서 Tangram을 Amazon Location과 통합하는 방법을 설명합니다. 하지만 React 및 Angular와 같은 프레임워크를 사용할 때도 동일한 라이브러리와 기술이 적용됩니다.

Tangram은 모바일 친화적인 대화형 지도를 위한 오픈 소스 라이브러리인 Leaflet을 기반으로 구축되었습니다. JavaScript 즉, 많은 Leaflet 호환 플러그인 및 컨트롤이 Tangram에서도 작동합니다.

Tilezen 스키마와 함께 작동하도록 구축된 Tangram 스타일은 HERE의 맵을 사용할 때 Amazon Location과 대부분 호환됩니다. 다음이 포함됩니다.

- Bubble Wrap – 모든 기능을 갖춘 길 찾기 스타일로, 관심 지점을 표시하는 유용한 아이콘이 포함되어 있습니다.
- Cinnabar – 클래식한 디자인으로 일반 매핑 애플리케이션에 적합합니다.
- Refill – Stamen Design의 획기적인 Toner 스타일에서 영감을 받아 데이터 시각화 오버레이를 위해 디자인된 미니멀한 맵 스타일입니다.
- Tron – TRON의 시각적 언어를 활용한 스케일 변환에 대한 탐구
- Walkabout – 야외 활동에 초점을 맞춘 스타일로 하이킹이나 야외 활동에 안성맞춤입니다.

이 가이드에서는 버블 랩이라는 탱그램 스타일을 사용하여 기본 HTML/ JavaScript 애플리케이션 내에서 Tangram을 Amazon Location과 통합하는 방법을 설명합니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

다른 Tangram 스타일은 지형 정보를 인코딩하는 래스터 타일을 사용하는 것이 가장 좋지만, Amazon Location에서는 아직 이 기능을 지원하지 않습니다.

> **Important**
>
> 다음 튜토리얼의 Tangram 스타일은 `VectorHereContrast` 스타일로 구성된 Amazon Location 맵 리소스와만 호환됩니다.

애플리케이션 빌드: 스캐폴딩

애플리케이션은 웹 애플리케이션에 맵을 구축하는 JavaScript 데 사용되는 HTML 페이지입니다. HTML 페이지(`index.html`)를 만들고 맵의 컨테이너를 생성합니다.

- 맵의 `id`가 포함된 `div` 요소를 입력하여 맵의 크기를 맵 보기에 적용합니다.
- 크기는 뷰포트에서 상속됩니다.

```
<html>
  <head>
    <style>
      body {
        margin: 0;
      }

      #map {
        height: 100vh; /* 100% of viewport height */
      }
    </style>
  </head>
  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

애플리케이션 빌드: 종속성 추가

다음 종속성을 추가합니다.

- Leaflet 및 관련 CSS.

- Tangram.
- AWS SDK를 위한 것입니다. JavaScript

```
<!-- CSS dependencies -->
<link
  rel="stylesheet"
  href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
  integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/
sMZMZ19scR4PsZChSR7A=="
  crossorigin=""
/>
<!-- JavaScript dependencies -->
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
<script src="https://unpkg.com/tangram"></script>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.784.0.min.js"></script>
<script>
  // application-specific code
</script>
```

이는 필수 사전 조건이 포함된 빈 페이지를 만듭니다. 다음 단계는 애플리케이션용 JavaScript 코드를 작성하는 과정을 안내합니다.

애플리케이션 빌드: 구성

리소스 및 보안 인증 정보로 애플리케이션을 구성하려면

1. 리소스의 이름과 식별자를 입력합니다.

   ```
   // Cognito Identity Pool ID
   const identityPoolId = "us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd";
   // Amazon Location Service map name; must be HERE-backed
   const mapName = "TangramExampleMap";
   ```

2. 맵 사용 - 2단계, 인증 설정에서 생성한 인증되지 않은 자격 증명 풀을 사용하여 보안 인증 정보 공급자를 인스턴스화합니다. 이는 일반적인 AWS SDK 워크플로 외부의 보안 인증 정보를 사용하기 때문에 세션은 1시간 후에 만료됩니다.

   ```
   // extract the region from the Identity Pool ID; this will be used for both Amazon
    Cognito and Amazon Location
   AWS.config.region = identityPoolId.split(":", 1)[0];
   ```

```
// instantiate a Cognito-backed credential provider
const credentials = new AWS.CognitoIdentityCredentials({
  IdentityPoolId: identityPoolId,
});
```

3. Tangram을 사용하면 타일을 가져오는 데 사용되는 URL을 재정의할 수 있지만, 서명을 받을 수 있도록 요청을 가로채는 기능은 포함되지 않습니다.

   이 문제를 해결하려면 서비스 작업자가 처리할 가상 호스트 이름 `amazon.location`을 사용하여 Amazon Location을 가리키도록 `sources.mapzen.url`을 재정의합니다. 다음은 Bubble Wrap을 사용한 장면 구성의 예시입니다.

```
const scene = {
  import: [
    // Bubble Wrap style
    "https://www.nextzen.org/carto/bubble-wrap-style/10/bubble-wrap-style.zip",
    "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/label-7.zip",
    "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-usa.zip",
    "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-international.zip",
  ],
  // override values beneath the `sources` key in the style above
  sources: {
    mapzen: {
      // point at Amazon Location using a synthetic URL, which will be handled by
 the service
      // worker
      url: `https://amazon.location/${mapName}/{z}/{x}/{y}`,
    },
    // effectively disable raster tiles containing encoded normals
    normals: {
      max_zoom: 0,
    },
    "normals-elevation": {
      max_zoom: 0,
    },
  },
};
```

애플리케이션 빌드: 변환 요청

서비스 작업자를 등록하고 초기화하려면 맵을 초기화하기 전에 호출할 `registerServiceWorker` 함수를 만듭니다. 이렇게 하면 제공된 JavaScript 코드가 서비스 워커 `index.html` 제어라는 `sw.js` 별도의 파일에 등록됩니다.

보안 인증 정보는 Amazon Cognito에서 로드되고 리전과 함께 서비스 작업자로 전달되어 Signature Version 4의 타일 요청에 서명하기 위한 정보를 제공합니다.

```
/**
 * Register a service worker that will rewrite and sign requests using Signature
 Version 4.
 */
async function registerServiceWorker() {
  if ("serviceWorker" in navigator) {
    try {
      const reg = await navigator.serviceWorker.register("./sw.js");

      // refresh credentials from Amazon Cognito
      await credentials.refreshPromise();

      await reg.active.ready;

      if (navigator.serviceWorker.controller == null) {
        // trigger a navigate event to active the controller for this page
        window.location.reload();
      }

      // pass credentials to the service worker
      reg.active.postMessage({
        credentials: {
          accessKeyId: credentials.accessKeyId,
          secretAccessKey: credentials.secretAccessKey,
          sessionToken: credentials.sessionToken,
        },
        region: AWS.config.region,
      });
    } catch (error) {
      console.error("Service worker registration failed:", error);
    }
  } else {
    console.warn("Service worker support is required for this example");
  }
```

```
}
```

sw.js의 서비스 작업자 구현은 message이벤트를 수신하여 보안 인증 정보 및 리전 구성 변경 사항을 수집합니다. 또한 fetch 이벤트를 수신하여 프록시 서버 역할을 합니다. amazon.location 가상 호스트 이름을 대상으로 하는 fetch이벤트는 적절한 Amazon Location API를 대상으로 다시 작성되고 Amplify Core의 Signer를 사용하여 서명됩니다.

```
// sw.js
self.importScripts(
  "https://unpkg.com/@aws-amplify/core@3.7.0/dist/aws-amplify-core.min.js"
);

const { Signer } = aws_amplify_core;

let credentials;
let region;

self.addEventListener("install", (event) => {
  // install immediately
  event.waitUntil(self.skipWaiting());
});

self.addEventListener("activate", (event) => {
  // control clients ASAP
  event.waitUntil(self.clients.claim());
});

self.addEventListener("message", (event) => {
  const {
    data: { credentials: newCredentials, region: newRegion },
  } = event;

  if (newCredentials != null) {
    credentials = newCredentials;
  }

  if (newRegion != null) {
    region = newRegion;
  }
});

async function signedFetch(request) {
  const url = new URL(request.url);
```

```
  const path = url.pathname.slice(1).split("/");

  // update URL to point to Amazon Location
  url.pathname = `/maps/v0/maps/${path[0]}/tiles/${path.slice(1).join("/")}`;
  url.host = `maps.geo.${region}.amazonaws.com`;
  // strip params (Tangram generates an empty api_key param)
  url.search = "";

  const signed = Signer.signUrl(url.toString(), {
    access_key: credentials.accessKeyId,
    secret_key: credentials.secretAccessKey,
    session_token: credentials.sessionToken,
  });

  return fetch(signed);
}

self.addEventListener("fetch", (event) => {
  const { request } = event;

  // match the synthetic hostname we're telling Tangram to use
  if (request.url.includes("amazon.location")) {
    return event.respondWith(signedFetch(request));
  }

  // fetch normally
  return event.respondWith(fetch(request));
});
```

보안 인증 정보를 자동으로 갱신하여 만료되기 전에 서비스 작업자에게 보내려면 `index.html` 내에서 다음 함수를 사용하세요.

```
async function refreshCredentials() {
  await credentials.refreshPromise();

  if ("serviceWorker" in navigator) {
    const controller = navigator.serviceWorker.controller;

    controller.postMessage({
      credentials: {
        accessKeyId: credentials.accessKeyId,
        secretAccessKey: credentials.secretAccessKey,
        sessionToken: credentials.sessionToken,
```

```
      },
    });
  } else {
    console.warn("Service worker support is required for this example.");
  }

  // schedule the next credential refresh when they're about to expire
  setTimeout(refreshCredentials, credentials.expireTime - new Date());
}
```

애플리케이션 빌드: 맵 초기화

페이지가 로드된 후 맵을 표시하려면 맵을 초기화해야 합니다. 초기 맵 위치를 조정하고, 컨트롤을 추가하고, 데이터를 오버레이할 수 있습니다.

> **Note**
>
> 애플리케이션 또는 문서에서 사용하는 각 데이터 공급자에 대한 워드마크 또는 텍스트 속성을 제공해야 합니다. 속성 문자열은 `sources.esri.attribution`, `sources.here.attribution`, `source.grabmaptiles.attribution` 키의 스타일 설명자 응답에 포함됩니다.
>
> Tangram은 이러한 리소스를 요청하지 않고 HERE의 맵과만 호환되므로 “© 2020 HERE”를 사용하세요. 데이터 공급자와 함께 Amazon Location 리소스를 사용할 때는 서비스 이용 약관을 반드시 읽어보세요.

```
/**
 * Initialize a map.
 */
async function initializeMap() {
  // register the service worker to handle requests to https://amazon.location
  await registerServiceWorker();

  // Initialize the map
  const map = L.map("map").setView([49.2819, -123.1187], 10);
  Tangram.leafletLayer({
    scene,
  }).addTo(map);
  map.attributionControl.setPrefix("");
  map.attributionControl.addAttribution("© 2020 HERE");
}
```

```
initializeMap();
```

애플리케이션 실행

이 샘플을 실행하기 위해 다음을 수행할 수 있습니다.

- HTTPS를 지원하는 호스트를 사용합니다.
- 로컬 웹 서버를 사용하여 서비스 작업자 보안 제한 사항을 준수합니다.

로컬 웹 서버를 사용하려면 npx를 사용할 수 있습니다. npx는 Node.js 일부로 설치되기 때문입니다. `index.html` 및 `sw.js`와 동일한 디렉터리 내에서 `npx serve`를 사용할 수 있습니다. 이는 localhost:5000에서 애플리케이션을 제공합니다.

`index.html` 파일은 다음과 같습니다.

```
<!-- index.html -->
<html>
  <head>
    <link
      rel="stylesheet"
      href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
      integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/
keqq/sMZMZ19scR4PsZChSR7A=="
      crossorigin=""
    />
    <style>
      body {
        margin: 0;
      }

      #map {
        height: 100vh;
      }
    </style>
  </head>

  <body>
    <div id="map" />
    <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
    <script src="https://unpkg.com/tangram"></script>
    <script src="https://sdk.amazonaws.com/js/aws-sdk-2.784.0.min.js"></script>
```

```
    <script>
      // configuration
      // Cognito Identity Pool ID
      const identityPoolId = "<Identity Pool ID>";
      // Amazon Location Service Map name; must be HERE-backed
      const mapName = "<Map name>";

      AWS.config.region = identityPoolId.split(":")[0];

      // instantiate a credential provider
      credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: identityPoolId,
      });

      const scene = {
        import: [
          // Bubble Wrap style
          "https://www.nextzen.org/carto/bubble-wrap-style/10/bubble-wrap-style.zip",
          "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/label-7.zip",
          "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-usa.zip",
          "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-international.zip",
        ],
        // override values beneath the `sources` key in the style above
        sources: {
          mapzen: {
            // point at Amazon Location using a synthetic URL, which will be handled by
 the service
            // worker
            url: `https://amazon.location/${mapName}/{z}/{x}/{y}`,
          },
          // effectively disable raster tiles containing encoded normals
          normals: {
            max_zoom: 0,
          },
          "normals-elevation": {
            max_zoom: 0,
          },
        },
      };

      /**
```

```
       * Register a service worker that will rewrite and sign requests using Signature
Version 4.
       */
      async function registerServiceWorker() {
        if ("serviceWorker" in navigator) {
          try {
            const reg = await navigator.serviceWorker.register("./sw.js");

            // refresh credentials from Amazon Cognito
            await credentials.refreshPromise();

            await reg.active.ready;

            if (navigator.serviceWorker.controller == null) {
              // trigger a navigate event to active the controller for this page
              window.location.reload();
            }

            // pass credentials to the service worker
            reg.active.postMessage({
              credentials: {
                accessKeyId: credentials.accessKeyId,
                secretAccessKey: credentials.secretAccessKey,
                sessionToken: credentials.sessionToken,
              },
              region: AWS.config.region,
            });
          } catch (error) {
            console.error("Service worker registration failed:", error);
          }
        } else {
          console.warn("Service Worker support is required for this example");
        }
      }

      /**
       * Initialize a map.
       */
      async function initializeMap() {
        // register the service worker to handle requests to https://amazon.location
        await registerServiceWorker();

        // Initialize the map
        const map = L.map("map").setView([49.2819, -123.1187], 10);
```

```
        Tangram.leafletLayer({
          scene,
        }).addTo(map);
        map.attributionControl.setPrefix("");
        map.attributionControl.addAttribution("© 2020 HERE");
      }

      initializeMap();
    </script>
  </body>
</html>
```

sw.js 파일은 다음과 같습니다.

```
// sw.js
self.importScripts(
  "https://unpkg.com/@aws-amplify/core@3.7.0/dist/aws-amplify-core.min.js"
);

const { Signer } = aws_amplify_core;

let credentials;
let region;

self.addEventListener("install", (event) => {
  // install immediately
  event.waitUntil(self.skipWaiting());
});

self.addEventListener("activate", (event) => {
  // control clients ASAP
  event.waitUntil(self.clients.claim());
});

self.addEventListener("message", (event) => {
  const {
    data: { credentials: newCredentials, region: newRegion },
  } = event;

  if (newCredentials != null) {
    credentials = newCredentials;
  }
```

```
  if (newRegion != null) {
    region = newRegion;
  }
});

async function signedFetch(request) {
  const url = new URL(request.url);
  const path = url.pathname.slice(1).split("/");

  // update URL to point to Amazon Location
  url.pathname = `/maps/v0/maps/${path[0]}/tiles/${path.slice(1).join("/")}`;
  url.host = `maps.geo.${region}.amazonaws.com`;
  // strip params (Tangram generates an empty api_key param)
  url.search = "";

  const signed = Signer.signUrl(url.toString(), {
    access_key: credentials.accessKeyId,
    secret_key: credentials.secretAccessKey,
    session_token: credentials.sessionToken,
  });

  return fetch(signed);
}

self.addEventListener("fetch", (event) => {
  const { request } = event;

  // match the synthetic hostname we're telling Tangram to use
  if (request.url.includes("amazon.location")) {
    return event.respondWith(signedFetch(request));
  }

  // fetch normally
  return event.respondWith(fetch(request));
});
```

이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

### Amazon Location Service로 Android용 Tangram ES 사용

Tangram ES는 OpenGL ES를 사용하여 벡터 데이터에서 2D 및 3D 맵을 렌더링하기 위한 C++ 라이브러리입니다. 이것은 Tangram의 네이티브 버전입니다.

Tilezen 스키마와 함께 작동하도록 구축된 Tangram 스타일은 HERE의 맵을 사용할 때 Amazon Location과 대부분 호환됩니다. 다음이 포함됩니다.

- Bubble Wrap – 모든 기능을 갖춘 길 찾기 스타일로, 관심 지점을 표시하는 유용한 아이콘이 포함되어 있습니다.
- Cinnabar – 클래식한 디자인으로 일반 매핑 애플리케이션에 적합합니다.
- Refill – Stamen Design의 획기적인 Toner 스타일에서 영감을 받아 데이터 시각화 오버레이를 위해 디자인된 미니멀한 맵 스타일입니다.
- Tron – TRON의 시각적 언어를 활용한 스케일 변환에 대한 탐구
- Walkabout – 야외 활동에 초점을 맞춘 스타일로 하이킹이나 야외 활동에 안성맞춤입니다.

이 가이드에서는 Cinnabar라는 Tangram 스타일을 사용하여 Android용 Tangram ES를 Amazon Location과 통합하는 방법을 설명합니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

다른 Tangram 스타일은 지형 정보를 인코딩하는 래스터 타일을 사용하는 것이 가장 좋지만, Amazon Location에서는 아직 이 기능을 지원하지 않습니다.

> **Important**
>
> 다음 튜토리얼의 Tangram 스타일은 `VectorHereContrast` 스타일로 구성된 Amazon Location 맵 리소스와만 호환됩니다.

애플리케이션 빌드: 초기화

애플리케이션을 초기화하려면

1. 빈 활동 템플릿에서 새 Android 스튜디오 프로젝트를 만듭니다.
2. 프로젝트 언어로 Kotlin이 선택되었는지 확인합니다.
3. API 16의 최소 SDK: Android 4.1(Jelly Bean) 이상을 선택합니다.
4. 프로젝트 구조를 열어 파일, 프로젝트 구조...를 선택한 다음 종속성 섹션을 선택합니다.
5. <All Modules>를 선택한 상태에서 + 버튼을 선택하여 새 라이브러리 종속성을 추가합니다.
6. AWS Android SDK 버전 2.19.1 이상을 추가합니다. 예: `com.amazonaws:aws-android-sdk-core:2.19.1`

7. Tangram 버전 0.13.0 이상을 추가합니다. 예를 들면 `com.mapzen.tangram:tangram:0.13.0`입니다.

> **Note**
>
> Tangram을 검색하면 `com.mapzen.tangram:tangram:0.13.0`이 “not found”라는 메시지를 만들지만 확인을 선택하면 추가할 수 있습니다.

애플리케이션 빌드: 구성

리소스 및 AWS 리전을 사용하여 애플리케이션을 구성하려면

1. `app/src/main/res/values/configuration.xml` 생성.
2. 리소스의 이름과 식별자는 물론 리소스가 생성된 AWS 리전도 입력합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="identityPoolId">us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd</
string>
    <string name="mapName">TangramExampleMap</string>
    <string name="awsRegion">us-east-1</string>
    <string name="sceneUrl">https://www.nextzen.org/carto/cinnabar-style/9/cinnabar-
style.zip</string>
    <string name="attribution">© 2020 HERE</string>
</resources>
```

애플리케이션 빌드: 활동 레이아웃

편집 `app/src/main/res/layout/activity_main.xml`:

- 맵을 렌더링하는 `MapView`를 추가합니다. 이렇게 하면 맵의 초기 중심점도 설정됩니다.
- 속성을 표시하는 `TextView`를 추가합니다.

이렇게 하면 맵의 초기 중심점도 설정됩니다.

> **Note**
>
> 애플리케이션 또는 문서에서 사용하는 각 데이터 공급자에 대한 워드마크 또는 텍스트 속성을 제공해야 합니다. 속성 문자열은 `sources.esri.attribution`, `sources.here.attribution`, `source.grabmaptiles.attribution` 키의 스타일 설명자 응답에 포함됩니다.
>
> Tangram은 이러한 리소스를 요청하지 않고 HERE의 맵과만 호환되므로 "© 2020 HERE"를 사용하세요. 데이터 공급자와 함께 Amazon Location 리소스를 사용할 때는 서비스 이용 약관을 반드시 읽어보세요.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.mapzen.tangram.MapView
        android:id="@+id/map"
        android:layout_height="match_parent"
        android:layout_width="match_parent" />

    <TextView
        android:id="@+id/attributionView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#80808080"
        android:padding="5sp"
        android:textColor="@android:color/black"
        android:textSize="10sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        tools:ignore="SmallSp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

애플리케이션 빌드: 변환 요청

AWS 요청을 가로채는 이름이 SigV4Interceptor인 클래스를 만들고 Signature Version 4를 사용하여 요청에 서명합니다. 이는 기본 활동이 생성될 때 맵 리소스를 가져오는 데 사용되는 HTTP 클라이언트에 등록됩니다.

```
package aws.location.demo.okhttp

import com.amazonaws.DefaultRequest
import com.amazonaws.auth.AWS4Signer
import com.amazonaws.auth.AWSCredentialsProvider
import com.amazonaws.http.HttpMethodName
import com.amazonaws.util.IOUtils
import okhttp3.HttpUrl
import okhttp3.Interceptor
import okhttp3.Request
import okhttp3.Response
import okio.Buffer
import java.io.ByteArrayInputStream
import java.net.URI

class SigV4Interceptor(
    private val credentialsProvider: AWSCredentialsProvider,
    private val serviceName: String
) : Interceptor {
    override fun intercept(chain: Interceptor.Chain): Response {
        val originalRequest = chain.request()

        if (originalRequest.url().host().contains("amazonaws.com")) {
            val signer = if (originalRequest.url().encodedPath().contains("@")) {
                // the presence of "@" indicates that it doesn't need to be double URL-
encoded
                AWS4Signer(false)
            } else {
                AWS4Signer()
            }

            val awsRequest = toAWSRequest(originalRequest, serviceName)
            signer.setServiceName(serviceName)
            signer.sign(awsRequest, credentialsProvider.credentials)

            return chain.proceed(toSignedOkHttpRequest(awsRequest, originalRequest))
        }
```

```
        return chain.proceed(originalRequest)
    }

    companion object {
        fun toAWSRequest(request: Request, serviceName: String): DefaultRequest<Any> {
            // clone the request (AWS-style) so that it can be populated with
credentials
            val dr = DefaultRequest<Any>(serviceName)

            // copy request info
            dr.httpMethod = HttpMethodName.valueOf(request.method())
            with(request.url()) {
                dr.resourcePath = uri().path
                dr.endpoint = URI.create("${scheme()}://${host()}")

                // copy parameters
                for (p in queryParameterNames()) {
                    if (p != "") {
                        dr.addParameter(p, queryParameter(p))
                    }
                }
            }

            // copy headers
            for (h in request.headers().names()) {
                dr.addHeader(h, request.header(h))
            }

            // copy the request body
            val bodyBytes = request.body()?.let { body ->
                val buffer = Buffer()
                body.writeTo(buffer)
                IOUtils.toByteArray(buffer.inputStream())
            }

            dr.content = ByteArrayInputStream(bodyBytes ?: ByteArray(0))

            return dr
        }

        fun toSignedOkHttpRequest(
            awsRequest: DefaultRequest<Any>,
            originalRequest: Request
```

```
        ): Request {
            // copy signed request back into an OkHttp Request
            val builder = Request.Builder()

            // copy headers from the signed request
            for ((k, v) in awsRequest.headers) {
                builder.addHeader(k, v)
            }

            // start building an HttpUrl
            val urlBuilder = HttpUrl.Builder()
                .host(awsRequest.endpoint.host)
                .scheme(awsRequest.endpoint.scheme)
                .encodedPath(awsRequest.resourcePath)

            // copy parameters from the signed request
            for ((k, v) in awsRequest.parameters) {
                urlBuilder.addQueryParameter(k, v)
            }

            return builder.url(urlBuilder.build())
                .method(originalRequest.method(), originalRequest.body())
                .build()
        }
    }
}
```

애플리케이션 빌드: 기본 활동

기본 활동은 사용자에게 표시될 보기를 초기화하는 역할을 담당합니다. 여기에는 다음이 포함됩니다.

- Amazon Cognito `CredentialsProvider` 인스턴트화.
- Signature Version 4 인터셉터 등록.
- 맵 스타일을 가리키고 타일 URL을 재정의하고 적절한 속성을 표시하여 맵을 구성합니다.

`MainActivity`은 수명 주기 이벤트를 맵 보기로 전달하는 역할도 합니다.

```
package aws.location.demo.tangram

import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
```

```
import aws.location.demo.okhttp.SigV4Interceptor
import com.amazonaws.auth.CognitoCachingCredentialsProvider
import com.amazonaws.regions.Regions
import com.mapzen.tangram.*
import com.mapzen.tangram.networking.DefaultHttpHandler
import com.mapzen.tangram.networking.HttpHandler

private const val SERVICE_NAME = "geo"

class MainActivity : AppCompatActivity(), MapView.MapReadyCallback {
    private var mapView: MapView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        mapView = findViewById(R.id.map)

        mapView?.getMapAsync(this, getHttpHandler())
        findViewById<TextView>(R.id.attributionView).text =
 getString(R.string.attribution)
    }

    override fun onMapReady(mapController: MapController?) {
        val sceneUpdates = arrayListOf(
            SceneUpdate(
                "sources.mapzen.url",
                "https://maps.geo.${getString(R.string.awsRegion)}.amazonaws.com/maps/
v0/maps/${
                    getString(
                        R.string.mapName
                    )
                }/tiles/{z}/{x}/{y}"
            )
        )

        mapController?.let { map ->
            map.updateCameraPosition(
                CameraUpdateFactory.newLngLatZoom(
                    LngLat(-123.1187, 49.2819),
                    12F
                )
            )
```

```
            map.loadSceneFileAsync(
                getString(R.string.sceneUrl),
                sceneUpdates
            )
        }
    }

    private fun getHttpHandler(): HttpHandler {
        val builder = DefaultHttpHandler.getClientBuilder()

        val credentialsProvider = CognitoCachingCredentialsProvider(
            applicationContext,
            getString(R.string.identityPoolId),
            Regions.US_EAST_1
        )

        return DefaultHttpHandler(
            builder.addInterceptor(
                SigV4Interceptor(
                    credentialsProvider,
                    SERVICE_NAME
                )
            )
        )
    }

    override fun onResume() {
        super.onResume()
        mapView?.onResume()
    }

    override fun onPause() {
        super.onPause()
        mapView?.onPause()
    }

    override fun onLowMemory() {
        super.onLowMemory()
        mapView?.onLowMemory()
    }

    override fun onDestroy() {
        super.onDestroy()
        mapView?.onDestroy()
```

```
    }
}
```

이 애플리케이션을 실행하면 선택한 스타일로 전체 화면 맵이 표시됩니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

Amazon Location Service로 iOS용 Tangram ES 사용

Tangram ES는 OpenGL ES를 사용하여 벡터 데이터에서 2D 및 3D 맵을 렌더링하기 위한 C++ 라이브러리입니다. 이것은 Tangram의 네이티브 버전입니다.

Tilezen 스키마와 함께 작동하도록 구축된 Tangram 스타일은 HERE의 맵을 사용할 때 Amazon Location과 대부분 호환됩니다. 다음이 포함됩니다.

- Bubble Wrap – 모든 기능을 갖춘 길 찾기 스타일로, 관심 지점을 표시하는 유용한 아이콘이 포함되어 있습니다.
- Cinnabar – 클래식한 디자인으로 일반 매핑 애플리케이션에 적합합니다.
- Refill – Stamen Design의 획기적인 Toner 스타일에서 영감을 받아 데이터 시각화 오버레이를 위해 디자인된 미니멀한 맵 스타일입니다.
- Tron – TRON의 시각적 언어를 활용한 스케일 변환에 대한 탐구
- Walkabout – 야외 활동에 초점을 맞춘 스타일로 하이킹이나 야외 활동에 안성맞춤입니다.

이 가이드에서는 Cinnabar라는 Tangram 스타일을 사용하여 iOS용 Tangram ES를 Amazon Location과 통합하는 방법을 설명합니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

다른 Tangram 스타일은 지형 정보를 인코딩하는 래스터 타일을 사용하는 것이 가장 좋지만, Amazon Location에서는 아직 이 기능을 지원하지 않습니다.

> **Important**
>
> 다음 튜토리얼의 Tangram 스타일은 `VectorHereContrast` 스타일로 구성된 Amazon Location 맵 리소스와만 호환됩니다.

애플리케이션 빌드: 초기화

애플리케이션을 초기화하려면,

1. 앱 템플릿에서 새 Xcode 프로젝트를 생성합니다
2. 인터페이스로 SwiftUI를 선택합니다.
3. 수명 주기로 SwiftUI 애플리케이션을 선택합니다.
4. 해당 언어로 Swift를 선택합니다.

애플리케이션 빌드: 종속성 추가

종속성을 추가하려면 다음과 같은 종속성 관리자를 사용할 수 있습니다. CocoaPods

1. 터미널에 다음을 설치합니다. CocoaPods

```
sudo gem install cocoapods
```

2. 애플리케이션의 프로젝트 디렉토리로 이동하여 CocoaPods 패키지 관리자를 사용하여 Podfile을 초기화하십시오.

```
pod init
```

3. Podfile을 열고 `AWSCore` 및 `Tangram-es`를 종속성으로 추가합니다.

```
platform :ios, '12.0'

target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
  pod 'Tangram-es'
end
```

4. 종속성 다운로드 및 설치:

```
pod install --repo-update
```

5. 다음을 생성한 Xcode 워크스페이스를 엽니다. CocoaPods

```
xed .
```

애플리케이션 빌드: 구성

Info.plist에 다음 키와 값을 추가하여 애플리케이션을 구성하고 원격 측정을 비활성화합니다.

| 키 | 값 |
| --- | --- |
| AWSRegion | us-east-1 |
| IdentityPoolId | us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd |
| MapName | ExampleMap |
| SceneURL | https://www.nextzen.org/carto/cinnabar-style/9/cinnabar-style.zip |

애플리케이션 빌드: 레이아웃 ContentView

맵을 렌더링하려면 `ContentView.swift`을 편집합니다.

- 맵을 렌더링하는 `MapView`를 추가합니다.
- 속성을 표시하는 `TextField`를 추가합니다.

이렇게 하면 맵의 초기 중심점도 설정됩니다.

> **Note**
>
> 애플리케이션 또는 문서에서 사용하는 각 데이터 공급자에 대한 워드마크 또는 텍스트 속성을 제공해야 합니다. 속성 문자열은 `sources.esri.attribution`, `sources.here.attribution`, `source.grabmaptiles.attribution` 키의 스타일 설명자 응답에 포함됩니다. 데이터 공급자와 함께 Amazon Location 리소스를 사용할 때는 서비스 이용 약관을 반드시 읽어보세요.

```
import SwiftUI
import TangramMap

struct ContentView: View {
```

```
    var body: some View {
        MapView()
            .cameraPosition(TGCameraPosition(
                                center: CLLocationCoordinate2DMake(49.2819, -123.1187),
                                zoom: 10,
                                bearing: 0,
                                pitch: 0))
            .edgesIgnoringSafeArea(.all)
            .overlay(
                Text("© 2020 HERE")
                    .disabled(true)
                    .font(.system(size: 12, weight: .light, design: .default))
                    .foregroundColor(.black)
                    .background(Color.init(Color.RGBColorSpace.sRGB, white: 0.5,
 opacity: 0.5))
                    .cornerRadius(1),
                alignment: .bottomTrailing)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

애플리케이션 빌드: 변환 요청

다음 클래스 정의가 포함된 `AWSSignatureV4URLHandler.swift`라는 이름의 새 Swift 파일을 만들어 AWS 요청을 가로채고 Signature Version 4를 사용하여 서명합니다. 이는 Tangram `MapView` 내에서 URL 핸들러로 등록됩니다.

```
import AWSCore
import TangramMap

class AWSSignatureV4URLHandler: TGDefaultURLHandler {
    private let region: AWSRegionType
    private let identityPoolId: String
    private let credentialsProvider: AWSCredentialsProvider

    init(region: AWSRegionType, identityPoolId: String) {
        self.region = region
        self.identityPoolId = identityPoolId
```

```
        self.credentialsProvider = AWSCognitoCredentialsProvider(regionType: region,
identityPoolId: identityPoolId)
        super.init()
    }

    override func downloadRequestAsync(_ url: URL, completionHandler: @escaping
TGDownloadCompletionHandler) -> UInt {
        if url.host?.contains("amazonaws.com") != true {
            // not an AWS URL
            return super.downloadRequestAsync(url, completionHandler:
completionHandler)
        }

        // URL-encode spaces, etc.
        let keyPath = String(url.path.dropFirst())
        guard let keyPathSafe =
keyPath.addingPercentEncoding(withAllowedCharacters: .urlPathAllowed) else {
            print("Invalid characters in path '\(keyPath)'; unsafe to sign")
            return super.downloadRequestAsync(url, completionHandler:
completionHandler)
        }

        // sign the URL
        let endpoint = AWSEndpoint(region: region, serviceName: "geo", url: url)
        let requestHeaders: [String: String] = ["host": endpoint!.hostName]
        let task = AWSSignatureV4Signer
            .generateQueryStringForSignatureV4(
                withCredentialProvider: credentialsProvider,
                httpMethod: .GET,
                expireDuration: 60,
                endpoint: endpoint!,
                keyPath: keyPathSafe,
                requestHeaders: requestHeaders,
                requestParameters: .none,
                signBody: true)
        task.waitUntilFinished()

        if let error = task.error as NSError? {
            print("Error occurred: \(error)")
        }

        if let result = task.result {
            // have Tangram fetch the signed URL
```

```
            return super.downloadRequestAsync(result as URL, completionHandler:
 completionHandler)
        }

        // fall back to an unsigned URL
        return super.downloadRequestAsync(url, completionHandler: completionHandler)
    }
}
```

애플리케이션 빌드: 맵 보기

맵 보기는 AWSSignatureV4Delegate 인스턴스를 초기화하고, 리소스를 가져오고 맵을 렌더링하는 기본 MGLMapView 구성을 담당합니다. 또한 스타일 설명자 소스에서 속성 문자열을 다시 ContentView로 전파되는 것도 처리합니다.

다음 MapView.swift 정의를 포함하는 struct라는 이름을 가진 새 Swift 파일을 생성합니다.

```
import AWSCore
import TangramMap
import SwiftUI

struct MapView: UIViewRepresentable {
    private let mapView: TGMapView

    init() {
        let regionName = Bundle.main.object(forInfoDictionaryKey: "AWSRegion") as!
 String
        let identityPoolId = Bundle.main.object(forInfoDictionaryKey: "IdentityPoolId")
 as! String
        let mapName = Bundle.main.object(forInfoDictionaryKey: "MapName") as! String
        let sceneURL = URL(string: Bundle.main.object(forInfoDictionaryKey: "SceneURL")
 as! String)!

        let region = (regionName as NSString).aws_regionTypeValue()

        // rewrite tile URLs to point at AWS resources
        let sceneUpdates = [
            TGSceneUpdate(path: "sources.mapzen.url",
                          value: "https://maps.geo.\(regionName).amazonaws.com/maps/v0/
maps/\(mapName)/tiles/{z}/{x}/{y}")]

        // instantiate a TGURLHandler that will sign AWS requests
```

```
        let urlHandler = AWSSignatureV4URLHandler(region: region, identityPoolId:
 identityPoolId)

        // instantiate the map view and attach the URL handler
        mapView = TGMapView(frame: .zero, urlHandler: urlHandler)

        // load the map style and apply scene updates (properties modified at runtime)
        mapView.loadScene(from: sceneURL, with: sceneUpdates)
    }

    func cameraPosition(_ cameraPosition: TGCameraPosition) -> MapView {
        mapView.cameraPosition = cameraPosition

        return self
    }

    // MARK: - UIViewRepresentable protocol

    func makeUIView(context: Context) -> TGMapView {
        return mapView
    }

    func updateUIView(_ uiView: TGMapView, context: Context) {
    }
}
```

이 애플리케이션을 실행하면 선택한 스타일로 전체 화면 맵이 표시됩니다. 이 샘플은 의 Amazon Location Service 샘플 리포지토리의 일부로 제공됩니다 GitHub.

## 맵에 데이터 특성 그리기

Amplify MapLibre 또는 Tangram을 사용하여 맵을 렌더링하는 응용 프로그램을 만든 후 자연스러운 다음 단계는 맵 위에 피처를 그리는 것입니다. 예를 들어, 고객 위치를 맵에 마커로 렌더링할 수 있습니다.

일반적으로 장소 검색 기능을 사용하여 데이터에서 위치를 찾은 다음 Amplify 또는 Tangram의 기능을 사용하여 위치를 렌더링할 수 있습니다. MapLibre

지도에 다양한 유형의 객체를 렌더링하는 샘플을 보려면 다음 MapLibre 샘플을 참조하십시오.

- 예제: 마커 그리기
- 예제: 군집된 포인트 그리기
- 예제: 다각형 그리기

추가 샘플 및 튜토리얼은 Amazon Location Service 작업을 위한 코드 예제 및 자습서 항목을 참조하세요.

## 를 사용하여 맵의 범위를 설정합니다. MapLibre

사용자가 전 세계 곳곳을 이동하거나 확대/축소하지 못하도록 하고 싶은 경우가 있습니다. MapLibre의 맵 컨트롤을 사용하는 경우 옵션을 사용하여 맵 컨트롤의 범위 또는 범위를 제한하고 및 `maxBounds` 옵션으로 확대/축소를 제한할 수 있습니다. `minZoom` `maxZoom`

다음 코드 예시는 특정 경계(이 경우 Grab 데이터 소스의 범위)로 이동을 제한하도록 맵 컨트롤을 초기화하는 방법을 보여줍니다.

> **Note**
>
> 이 샘플은 튜토리얼의 JavaScript 상황에 맞게 제공되며 작동 가능합니다. 웹 앱 생성

```
// Set bounds to Grab data provider region
var bounds = [
        [90.0, -21.943045533438166], // Southwest coordinates
        [146.25, 31.952162238024968] // Northeast coordinates
    ];

var mlglMap = new maplibregl.Map(
    {
    container: 'map',
    style: mapName,
    maxBounds: bounds // Sets bounds as max
    transformRequest,
    }
);
```

마찬가지로 맵의 최소 및 최대 확대/축소 수준을 설정할 수 있습니다. 두 값 모두 0에서 24 사이일 수 있지만 기본값은 최소 확대/축소는 0, 최대 확대/축소는 22입니다(데이터 공급자가 모든 확대/축소 수준에서 데이터를 제공하지 않을 수도 있습니다. 대부분의 맵 라이브러리는 이를 자동으로 처리합니다). 다음 예제에서는 MapLibre 맵 컨트롤의 `minZoom` 및 `maxZoom` 옵션을 초기화합니다.

```
// Set the minimum and maximum zoom levels
var mlglMap = new maplibregl.Map(
    {
    container: 'map',
```

```
    style: mapName,
    maxZoom: 12,
    minZoom: 5,
    transformRequest,
    }
);
```

> **Tip**
>
> 또한 MapLibre 맵 컨트롤을 사용하면 초기화 과정 대신 런타임에 및 함수를 사용하여 이러한 옵션을 설정할 수 있습니다. `get... set...` 예를 들어 런타임 시 `getMaxBounds` 및 `setMaxBounds`를 사용하여 맵 경계를 변경할 수 있습니다.

## 맵 리소스 관리

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 맵 리소스를 관리할 수 있습니다.

### 맵 리소스 목록

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 맵 리소스 목록을 볼 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 기존 맵 리소스 목록을 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 맵을 선택합니다.
3. 내 맵에서 맵 리소스 목록을 확인합니다.

API

Amazon Location Maps API에서 `ListMaps` 작업을 사용합니다.

다음은 AWS 계정의 맵 리소스 목록을 가져오기 위한 API 요청입니다.

```
POST /maps/v0/list-maps
```

다음은 ListMaps에 대한 응답의 예입니다:

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-30T01:38:36Z,
         "DataSource": "Esri",
         "Description": "string",
         "MapName": "ExampleMap",
         "UpdateTime": 2020-10-30T01:38:36Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

list-map 명령을 사용합니다.

다음 예는 AWS 계정의 맵 리소스 목록을 가져오는 AWS CLI입니다.

```
aws location list-maps
```

## 맵 리소스 세부 정보 가져오기

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정의 모든 맵 리소스에 대한 세부 정보를 얻을 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 맵 리소스의 세부 정보를 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 맵을 선택합니다.
3. 내 맵에서 대상 맵 리소스의 이름 링크를 선택합니다.

API

Amazon Location Maps API에서 DescribeMap 작업을 사용합니다.

다음 예시는 에 대한 맵 리소스 세부정보를 가져오기 위한 API *ExampleMap*요청입니다.

```
GET /maps/v0/maps/ExampleMap
```

다음은 DescribeMap에 대한 응답의 예입니다:

```
{
   "Configuration": {
      "Style": "VectorEsriNavigation"
   },
   "CreateTime": 2020-10-30T01:38:36Z,
   "DataSource": "Esri",
   "Description": "string",
   "MapArn": "arn:aws:geo:us-west-2:123456789012:maps/ExampleMap",
   "MapName": "ExampleMap",
   "Tags": {
      "Tag1" : "Value1"
   },
   "UpdateTime": 2020-10-30T01:40:36Z
}
```

CLI

describe-map 명령을 사용합니다.

다음 예제는 AWS CLI 에 대한 맵 리소스 세부 정보를 가져오는 예제입니다 *ExampleMap*.

```
aws location describe-map \
    --map-name "ExampleMap"
```

## 맵 리소스 삭제

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정에서 맵 리소스를 삭제할 수 있습니다.

> **Warning**
>
> 이 작업은 리소스를 영구적으로 삭제합니다.

Console

Amazon Location 콘솔을 사용하여 기존 맵 리소스를 삭제하려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 맵을 선택합니다.
3. 내 맵 목록에서 대상 맵을 선택합니다.
4. 맵 삭제를 선택합니다.

API

Amazon Location Maps API에서 `DeleteMap` 작업을 사용합니다.

다음은 맵 리소스를 삭제하기 위한 API *ExampleMap*요청입니다.

```
DELETE /maps/v0/maps/ExampleMap
```

다음은 `DeleteMap`에 대한 성공적인 응답의 예입니다.

```
HTTP/1.1 200
```

CLI

`delete-map` 명령을 사용합니다.

다음 예시는 맵 리소스를 삭제하는 AWS CLI *ExampleMap*명령입니다.

```
aws location delete-map \
    --map-name "ExampleMap"
```

# Amazon Location을 사용하여 장소 및 지리적 위치 데이터 검색

Amazon Location에는 선택한 제공자의 지리적 위치 또는 장소 데이터를 검색하는 기능이 포함됩니다. 여러 종류의 검색이 가능합니다.

- 지오코딩 - 지오코딩은 텍스트 입력을 기반으로 주소, 지역, 회사 이름 또는 기타 관심 지역을 검색하는 프로세스입니다. 검색된 결과의 세부 정보와 위치(위도 및 경도)를 반환합니다.

- 역방향 지오코딩 - 역방향 지오코딩을 사용하면 지정된 위치 근처의 장소를 찾을 수 있습니다.
- 자동 완성 - 자동 완성은 사용자가 쿼리를 입력할 때 자동으로 제안을 하는 프로세스입니다. 예를 들어 사용자가 **Par**을 입력하면 제안 중 하나가 `Paris, France`일 수 있습니다.

Amazon Location에서는 장소 색인 리소스를 생성하고 구성하여 장소 검색 작업에 사용할 데이터 제공자를 선택할 수 있습니다.

리소스를 생성한 후에는 선호하는 언어인 Amplify 또는 REST API 엔드포인트의 AWS SDK를 사용하여 요청을 보낼 수 있습니다. 응답 데이터를 사용하여 지도에 위치를 표시하고, 위치 데이터를 강화하고, 위치를 사람이 읽을 수 있는 텍스트로 변환할 수 있습니다.

> **Note**
>
> 장소 검색 개념에 대한 개요는 장소 검색을 참조하세요.

주제



## 필수 조건

지오코딩, 역방향 지오코딩 또는 장소 검색을 시작하기 전에 필수 단계를 따르세요.

주제

## 장소 색인 리소스 생성

먼저 계정에서 장소 색인 리소스를 생성하세요. AWS

장소 색인 리소스를 생성할 때 지오코딩, 역방향 지오코딩, 검색에 대한 쿼리를 지원하는 데 사용할 수 있는 데이터 제공자를 선택할 수 있습니다.

1. Esri - 관심 지역의 Esri 범위에 대한 자세한 내용은 Esri 설명서의 Esri 지오코딩 범위를 참조하세요.
2. HERE 기술 - 관심 지역의 HERE 범위에 대한 자세한 내용은 HERE 설명서의 HERE 지오코딩 범위를 참조하세요.
3. Grab - Grab은 동남아시아에 대한 데이터만 제공합니다. Grab 범위에 대한 자세한 내용은 본 가이드의 대상 국가/리전 및 지역을 참조하세요.

Amazon 위치 서비스 콘솔 AWS CLI, 또는 Amazon 위치 API를 사용하여 이 작업을 수행할 수 있습니다.

Console

Amazon Location Service 콘솔을 사용하여 장소 색인 리소스를 생성하려면

1. https://console.aws.amazon.com/location/에서 Amazon Location Service 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 장소 색인을 선택합니다.
3. 장소 색인 생성을 선택합니다.
4. 다음 입력란을 작성합니다.
   - 이름 – 장소 색인 리소스의 이름을 입력합니다. 예를 들면 *ExamplePlaceIndex*다음과 같습니다. 최대 100자입니다. 유효한 항목에는 영숫자 문자, 하이픈, 마침표 및 밑줄이 포함됩니다.
   - 설명 – 선택적 설명을 입력합니다.
5. 데이터 공급자에서 장소 색인 리소스와 함께 사용할 데이터 공급자를 선택합니다.

   > **Note**
   >
   > 애플리케이션이 배송 차량 또는 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 Esri를 지리적 위치 제공업체로 사용해서는 안 됩니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

6. 데이터 스토리지 옵션에서 장소 색인 리소스의 검색 결과를 저장할지 여부를 지정합니다.

7. (선택 사항) 태그 아래에 태그 키 및 값을 입력합니다. 그러면 새 장소 색인 리소스에 태그가 추가됩니다. 자세한 내용을 알아보려면 리소스 태그 지정을 참조하세요.
8. 장소 색인 생성을 선택합니다.

API

Amazon Location API를 사용하여 장소 색인 리소스를 만들려면

Amazon Location Places API의 `CreatePlaceIndex` 작업을 사용합니다.

다음은 데이터 제공자 *Esri#* *ExamplePlaceIndex*사용하여 호출되는 장소 색인 리소스를 생성하기 위한 API 요청입니다.

```
POST /places/v0/indexes
Content-type: application/json

{
   "DataSource": "Esri",
   "DataSourceConfiguration": {
      "IntendedUse": "SingleUse"
   },
   "Description": "string",
   "IndexName": "ExamplePlaceIndex",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

AWS CLI 명령을 사용하여 장소 색인 리소스를 만들려면

`create-place-index` 명령을 사용합니다.

다음 예시에서는 *Esri#* 데이터 공급자로 *ExamplePlaceIndex*사용하여 라는 장소 색인 리소스를 생성합니다.

```
aws location \
  create-place-index \
  --data-source "Esri" \
  --description "Example place index" \
```

```
  --index-name "ExamplePlaceIndex" \
  --tags Tag1=Value1
```

> **Note**
>
> 청구는 사용량에 따라 달라집니다. 다른 AWS 서비스 사용 시 요금이 부과될 수 있습니다. 자세한 정보는 Amazon Location Service 가격을 참조하세요.

## 요청 인증

장소 색인 리소스를 생성하고 애플리케이션에 위치 기능을 구축할 준비가 되었으면 요청을 인증할 방법을 선택하세요.

- 서비스에 액세스하는 방법을 알아보려면 Amazon Location Service 액세스를 참조하세요.
- 익명 사용자가 있는 웹 사이트가 있는 경우 API 키 또는 Amazon Cognito를 사용하는 것이 좋습니다.

  예

  다음 예는 권한 부여를 위한 API 키 사용, AWS JavaScript SDK v3 및 Amazon Location을 사용하는 방법을 보여줍니다. JavaScript 인증 도우미

```
import { LocationClient, SearchPlaceIndexForTextCommand } from "@aws-sdk/client-
location";
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";

const apiKey = "v1.public.your-api-key-value"; // API key

// Create an authentication helper instance using an API key
const authHelper = await withAPIKey(apiKey);

const client = new LocationClient({
  region: "<region>", // region containing Cognito pool
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  IndexName: "ExamplePlaceIndex",
  Text: "Anyplace",
```

```
  BiasPosition: [-123.4567, 45.6789]
};

const command = new SearchPlaceIndexForTextCommand(input);

const response = await client.send(command);
```

# 지오코딩

지오코딩은 주소, 지역, 업체명, 관심 장소 등의 텍스트를 지리적 좌표 세트로 변환하는 프로세스입니다. 장소 색인 리소스를 사용하여 지오코딩 요청을 제출하고 지오코딩에서 검색된 데이터를 통합하여 웹 또는 모바일 애플리케이션용 지도에 데이터를 표시할 수 있습니다.

이 섹션에서는 간단한 지오코딩 요청을 보내는 방법, 그리고 선택적 사양과 함께 지오코딩 요청을 보내는 방법을 안내합니다.

## 지오코딩

주소를 좌표 세트로 변환하는 SearchPlaceIndexForText 작업을 사용하여 간단한 지오코딩 요청을 제출할 수 있습니다. 단순 요청에는 다음과 같은 필수 파라미터가 포함됩니다.

- `Text` – 좌표 세트로 변환할 주소, 이름, 도시 또는 지역. 예를 들어, 문자열 `Any  Town`.

페이지당 최대 결과 수를 지정하려면 다음과 같은 선택적 파라미터를 사용하세요.

- `MaxResults` – 쿼리 응답에 반환되는 최대 결과 수를 제한합니다.

AWS CLI 또는 Amazon 위치 API를 사용할 수 있습니다.

API

다음은 장소 색인 리소스에서 *Any  Town###* 주소 *ExamplePlaceIndex*, 이름, 도시 또는 지역을 SearchPlaceIndexForText 검색하라는 요청입니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
    "Text": "Any Town",
```

```
    "MaxResults": 10
}
```

AWS CLI

다음 예제는 장소 색인 리소스에서 *Any Town###* 주소, 이름, 도시 또는 지역을 검색하는 search-place-index-for-text 명령입니다. *ExamplePlaceIndex*

```
aws location \
    search-place-index-for-text \
        --index-name ExamplePlaceIndex \
        --text "Any Town" \
        --max-results 10
```

## 위치 주변 지오코드

지오코딩할 때 다음과 같은 선택적 파라미터를 사용하여 지정된 위치 근처를 지오코딩할 수 있습니다.

- `BiasPosition` – 주변에서 검색하려는 위치. 이렇게 하면 주어진 위치와 가장 가까운 결과를 검색하여 검색 범위를 좁힐 수 있습니다. `[longitude, latitude]`과 같이 정의됩니다.

다음 예는 위치 색인 리소스에서 위치 [*-123.4567*, *45.6789*] 근처의 *Any Town*이라는 주소, 이름, 도시 또는 지역을 검색하기 위한 SearchPlaceIndexForText 요청입니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "BiasPosition": [-123.4567,45.6789]
}
```

## 경계 상자 내의 지오코드

다음과 같은 선택적 파라미터를 사용하여 경계 상자 내에서 지오코딩하여 결과 범위를 지정된 경계 내의 좌표로 좁힐 수 있습니다.

- `FilterBBox` – 상자 경계 내의 좌표로 결과를 필터링하도록 지정하는 경계 상자. `[LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]`과 같이 정의됩니다.

> **Note**
>
> 요청은 `FilterBBox` 및 `BiasPosition` 파라미터를 모두 포함할 수 없습니다. 요청에서 두 파라미터를 모두 지정하면 `ValidationException` 오류가 반환됩니다.

다음 예는 경계 상자 내에서 *Any Town*이라는 주소, 이름, 도시 또는 지역을 검색하기 위한 `SearchPlaceIndexForText` 요청입니다. 경계 상자는 다음과 같습니다.

- *### #### ### -124.1450###*.
- *### #### ### 41.7045###*.
- *### #### ### -124.1387###*.
- *### #### ### 41.7096###*.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "FilterBBox":  [
        -124.1450,41.7045,
        -124.1387,41.7096
    ]
}
```

### 국가 내 지오코드

다음과 같은 선택적 파라미터를 사용하여 지정한 하나 이상의 국가 내에서 지오코딩할 수 있습니다.

- `FilterCountries` – 지오코딩하려는 국가 또는 지역. ISO 3166 3문자 국가 코드를 사용하여 요청 한 번으로 최대 100개의 국가를 정의할 수 있습니다. 예를 들어, 오스트레일리아의 경우 AUS을 사용합니다.

다음 예는 독일과 프랑스의 *Any Town*이라는 주소, 이름, 도시 또는 지역을 검색하기 위한 `SearchPlaceIndexForText` 요청입니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
```

```
Content-type: application/json

{
   "Text": "Any Town",
   "FilterCountries": ["DEU","FRA"]
}
```

## 카테고리별 필터링

다음과 같은 선택적 파라미터를 사용하여 지오코드 요청에서 반환되는 카테고리를 필터링할 수 있습니다.

- `FilterCategories` – 쿼리에 반환하려는 결과의 카테고리. 단일 요청에 최대 5개의 카테고리를 지정할 수 있습니다. 카테고리 섹션에서 Amazon Location Service 카테고리 목록을 찾을 수 있습니다. 예를 들어, 쿼리에 반환 호텔만 지정하도록 `Hotel`을 지정할 수 있습니다.

다음 예는 미국의 *`Hometown Coffee`*라는 커피숍을 검색하기 위한 `SearchPlaceIndexForText` 요청입니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Hometown Coffee",
   "FilterCategories": ["Coffee Shop"],
   "FilterCountries": ["USA"]
}
```

카테고리 필터링에 대한 자세한 내용은 장소 카테고리 및 필터링 결과을 참조하세요.

## 기본 언어의 지오코드

선택적 `Language` 파라미터를 사용하여 검색 결과에 대한 언어 기본 설정을 할 수 있습니다. 예를 들어, **`100 Main St, Anytown, USA`**에 대한 검색 결과가 기본적으로 `100 Main St, Any Town, USA`로 반환될 수 있습니다. 그러나 `fr`로서 `Language`을 선택하면 결과가 대신 `100 Rue Principale, Any Town, États-Unis`으로 반환될 수 있습니다.

- `Language` – 쿼리 결과를 렌더링하는 데 사용할 언어 코드입니다. 값은 유효한 BCP 47 언어 코드여야 합니다. 예를 들어, 영어의 경우`en`.

> **Note**
>
> Language이 지정되지 않았거나 또는 결과에 지정된 언어가 지원되지 않는 경우, 해당 결과에 대한 파트너의 기본 언어가 사용됩니다.

다음 예는 de으로 지정된 기본 언어로 **Any Town**라는 장소를 검색하기 위한 SearchPlaceIndexforText 요청입니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json
{
   "Text": "Any Town",
   "Language": "de"
}
```

## 응답의 예

### Example

다음은 Amazon Location Places API에서 SearchPlaceIndexForText 작업을 호출할 때의 응답의 예입니다. 결과에는 관련 장소와 요청 요약이 포함됩니다. 파트너로 Esri 또는 Here를 선택하면 두 개의 응답이 표시됩니다.

### Example request

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Amazon",
   "MaxResults": 1,
   "FilterCountries": ["USA"],
   "BiasPosition": [-112.10, 46.32]
}
```

### Example response (Esri)

```
{
    "Results": [
        {
```

```
            "Place": {
                "Country": "USA",
                "Geometry": {
                    "Point": [
                        -112.10667999999998,
                        46.319090000000074
                    ]
                },
                "Interpolated": false,
                "Label": "Amazon, MT, USA",
                "Municipality": "Amazon",
                "Region": "Montana",
                "SubRegion": "Jefferson County"
            },
            "Distance": 523.4619749879726,
            "Relevance": 1
        }
    ],
    "Summary": {
        "BiasPosition": [
            -112.1,
            46.32
        ],
        "DataSource": "Esri",
        "FilterCountries": [
            "USA"
        ],
        "MaxResults": 1,
        "ResultBBox": [
            -112.10667999999998,
            46.319090000000074,
            -112.10667999999998,
            46.319090000000074
        ],
        "Text": "Amazon"
    }
}
```


Example response (HERE)


```
{
    "Summary": {
        "Text": "Amazon",
```

```
        "BiasPosition": [
            -112.1,
            46.32
        ],
        "FilterCountries": [
            "USA"
        ],
        "MaxResults": 1,
        "ResultBBox": [
            -112.10668,
            46.31909,
            -112.10668,
            46.31909
        ],
        "DataSource": "Here"
    },
    "Results": [
        {
            "Place": {
                "Label": "Amazon, Jefferson City, MT, United States",
                "Geometry": {
                    "Point": [
                        -112.10668,
                        46.31909
                    ]
                },
                "Neighborhood": "Amazon",
                "Municipality": "Jefferson City",
                "SubRegion": "Jefferson",
                "Region": "Montana",
                "Country": "USA",
                "Interpolated": false,
                "TimeZone": {
                    "Name": "America/Denver",
                    "Offset": -25200
                }
            },
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9a
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ",            "Distance":
 523.4619749905755
        }
    ]
```

```
}
```

# 역방향 지오코딩

역방향 지오코딩은 일련의 좌표를 주소, 지역, 업체명, 관심 장소 등 의미 있는 텍스트로 변환하는 프로세스입니다. 장소 색인 리소스를 사용하여 역방향 지오코딩 요청을 제출하고 역방향 지오코딩에서 검색된 데이터를 통합하여 웹 또는 모바일 애플리케이션용 지도에 데이터를 표시할 수 있습니다.

이 섹션에서는 간단한 역방향 지오코딩 요청을 보내는 방법을 안내합니다.

## 역방향 지오코딩

`SearchPlaceIndexForPosition` 작업을 사용하여 좌표 세트를 역방향 지오코딩하고 이를 의미 있는 주소, 관심 장소 또는 주소가 없는 일반적인 위치로 변환해 달라는 간단한 요청을 제출할 수 있습니다. 단순 요청에는 다음과 같은 필수 파라미터가 포함됩니다.

- `Position` – 주소, 관심 장소 또는 일반 위치로 변환하려는 좌표 세트. 형식 `[longitude,latitude]`을 사용하여 정의됩니다.

페이지당 최대 결과 수를 지정하려면 다음과 같은 선택적 파라미터를 추가합니다.

- `MaxResults` – 쿼리 응답에 반환되는 최대 결과 수를 제한합니다.

쿼리 결과에 사용할 기본 언어를 지정하려면 다음과 같은 선택적 파라미터를 사용하세요.

- `Language` – 결과를 렌더링하는 데 사용되는 언어 코드입니다. 값은 유효한 BCP 47 언어 코드여야 합니다. 예를 들어, 영어의 경우`en`.

> **Note**
>
> `Language`이 지정되지 않았거나 또는 결과에 지정된 언어가 지원되지 않는 경우, 해당 결과에 대한 파트너의 기본 언어가 사용됩니다.

AWS CLI 또는 Amazon 위치 API를 사용할 수 있습니다.

API

*### ## ## ##### ## [122.3394 ExamplePlaceIndex, 47.6159] ### ## ## ##, # # ## ## #### ### #### ## SearchPlaceIndexForPosition #####.*

```
POST /places/v0/indexes/ExamplePlaceIndex/search/position
Content-type: application/json

{
   "Position": [-122.3394,47.6159],
   "MaxResults": 5,
   "Language": "de"
}
```

AWS CLI

*## ### ## ## ##### ## ## ## ExamplePlaceIndex, ## ## ## ## ### #### ### #### search-place-index-for-position ##### [122.3394, 47.6159].*

```
aws location \
    search-place-index-for-position \
        --index-name ExamplePlaceIndex \
        --position -122.3394 47.6159 \
        --max-results 5 \
        --language de
```

## 응답의 예

### Example

다음은 Amazon Location Places API에서 SearchPlaceIndexForPosition 작업을 호출할 때의 응답의 예입니다. 결과는 관련 장소와 요청 요약을 반환합니다. 파트너로 Esri 또는 Here를 선택하면 두 개의 응답이 표시됩니다.

### Example request

```
POST /places/v0/indexes/ExamplePlaceIndex/search/position
Content-type: application/json

{
   "Position": [-122.3394,47.6159],
```

```
   "MaxResults": 1
}
```

Example response (Esri)


```
{
    "Results": [
        {
            "Place": {
                "AddressNumber": "2111",
                "Country": "USA",
                "Geometry": {
                    "Point": [
                        -122.33937999999995,
                        47.61591000000004
                    ]
                },
                "Interpolated": false,
                "Label": "The Spheres, 2111 7th Ave, Seattle, WA, 98121, USA",
                "Municipality": "Seattle",
                "Neighborhood": "Belltown",
                "PostalCode": "98121",
                "Region": "Washington",
                "SubRegion": "King County"
            },
            "Distance": 1.8685861313438727
        }
    ],
    "Summary": {
        "DataSource": "Esri",
        "MaxResults": 1,
        "Position": [
            -122.3394,
            47.6159
        ]
    }
}
```


Example response (HERE)


```
{
    "Summary": {
        "Position": [
```

```
            -122.3394,
            47.6159
        ],
        "MaxResults": 1,
        "DataSource": "Here"
    },
    "Results": [
        {
            "Place": {
                "Label": "2111 7th Ave, Seattle, WA 98121-5114, United States",
                "Geometry": {
                    "Point": [
                        -122.33938,
                        47.61591
                    ]
                },
                "AddressNumber": "2111",
                "Street": "7th Ave",
                "Neighborhood": "Belltown",
                "Municipality": "Seattle",
                "SubRegion": "King",
                "Region": "Washington",
                "Country": "USA",
                "PostalCode": "98121-5114",
                "Interpolated": false,
                "TimeZone": {
                    "Name": "America/Los_Angeles",
                    "Offset": -28800
                }
            },
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9a
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ",
            "Distance": 1.868586125090601
        }
    ]
}
```

# 자동 완성

자동 완성은 최종 사용자가 검색 쿼리를 입력할 때 반응형 피드백을 제공합니다. 이는 부분적이거나 철자가 잘못된 자유 형식 텍스트를 기반으로 주소 및 관심 지역에 대한 제안을 제공합니다. 장소 색인 리소스를 사용하여 자동 완성 제안을 요청하고 결과 제안을 애플리케이션에 표시할 수 있습니다.

Amazon Location은 자동 완성 제안 저장을 지원하지 않습니다. 자동 완성 호출에 사용되는 장소 색인이 저장된 지오코드와 함께 사용하도록 구성된 경우 오류가 반환됩니다. 저장된 지오코드 및 제안 쿼리를 사용하려면 여러 장소 색인을 생성하고 구성하세요.

이 섹션에서는 자동 완성 요청을 보내는 방법을 설명합니다. 가장 기본적인 형태의 요청부터 시작하여 자동 완성 검색 결과의 관련성을 높이는 데 사용할 수 있는 선택적 파라미터를 보여줍니다.

## 자동 완성 사용

SearchPlaceIndexForSuggestions 작업을 사용하여 자동 완성 제안에 대한 간단한 요청을 제출할 수 있습니다. 가장 간단한 형태의 요청에는 필수 파라미터가 하나뿐인 쿼리 Text이 있습니다.

- Text – 장소 제안을 생성하는 데 사용할 자유 형식의 부분 텍스트입니다. 예를 들어, 문자열 eiffel tow.

반환되는 결과 수를 제한하려면 선택적 MaxResults 파라미터를 추가합니다.

- MaxResults – 쿼리 응답에 반환되는 결과 수를 제한합니다.

Amazon Location API 또는 AWS CLI을 사용할 수 있습니다.

API

*## ### ## ## ##### ## ## ## kamp# #### ## 5## ## #### #### SearchPlaceIndexForSuggestions #####. ExamplePlaceIndex*

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "MaxResults": 5
}
```

AWS CLI

다음 예시는 장소 이름 *kamp#* 일부를 기준으로 최대 *5##* 추천 검색어를 장소 색인 리소스에서 검색하는 search-place-index-for-suggestions 명령입니다. *ExamplePlaceIndex*

```
aws location \
              search-place-index-for-suggestions \
              --index-name ExamplePlaceIndex \
              --text kamp \
              --max-results 5
```

SearchPlaceIndexForSuggestions을 호출하면 각 장소의 이름과 ID가 포함된 장소 목록이 생성됩니다. 이러한 결과를 사용하여 텍스트 상자 아래에 선택 항목의 드롭다운 목록을 제공하는 등 사용자가 입력할 때 검색할 수 있는 내용을 제안할 수 있습니다. 예를 들어, 사용자가 *kamp*를 입력한 결과에 따른 제안 결과는 다음과 같습니다.

```
{
    "Summary": {
        "Text": "kamp",
        "MaxResults": 5,
        "DataSource": "Esri"
    },
    "Results": [
        {
            "Text": "Kampuchea",
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ"
        },
        {
            "Text": "Kampoul, Kabul, AFG",
            "PlaceId":
 "AQAAAIAAA1mxl_-9ffzXD07rBgo9fh6E01Pd1YKvuT5rz2qBDxqBkhTlgkeiOPR2s5sa3YBLxUqQI8bhymsYcu9R-
DkX3L9QSi3CB5LhNPu160iSFJo6H8S1CrxO3QsJALhrr9mdbg0R4R4YDywkhkeBlnbn7g5C5LI_wYx873WeQZuilwtsGm8j
UeXcb_bg"
        },
        {
            "Text": "Kampala, UGA",
            "PlaceId":
 "AQAAAIAAzZfZt3qMruKGObyhP6MM0pqy2L8SULlVWT7a3ertLBRS6Q5n7I4s9D7E0nRHADAj7mL7kvX1Q8HD-
```

```
mpuiATXNJ1Ix4_V_1B15zHe8jlYKMWvXbgbO8cMpgR2fqYqZMR1x-
dfBOO8OoqujKZldvPIDK1kNe3GwcaqvvMWWPMeaGd203brFynubAe-MmFF-Gjz-WBMfUy9og6MV7bkk6NGCA"
        },
        {
            "Text": "Kampar, Riau, IDN",
            "PlaceId": "AQAAAIAAvbXXx-
srOi111tHOkPdao0GF7WQ_KaZ444SEnevycp6Gtf_2JWgPfCE5bIQCYwya1uZQpX2a8YJoFm2K7Col4fLu7IK0yYOLhZx4k
        },
        {
            "Text": "Kampung Pasir Gudang Baru, Johor, MYS",
            "PlaceId":
 "AQAAAIAA4HLQHdjUDcaaXLE9wtNIT1cjQYLgkBnMoG2eNN0AaQ8PJoWabLRXmmPUaAj8MAD6vT0i6zqaun5Mixyj7vnYX
        }
    ]
}
```

다음 섹션에서는 이러한 결과의 PlaceID을 사용하는 방법을 설명합니다.

## 자동완성 결과 사용

SearchPlaceIndexForSuggestions을 호출하면 각 장소의 이름과 ID가 포함된 장소 목록이 생성됩니다. 이러한 결과를 사용하여 텍스트 상자 아래에 선택 항목의 드롭다운 목록을 제공하는 등 사용자가 입력할 때 검색할 수 있는 내용을 제안할 수 있습니다. 사용자가 결과 중 하나를 선택하면 선택한 ID로 GetPlace작업을 호출하여 위치, 주소 또는 기타 세부 정보를 포함하여 해당 장소의 세부 정보를 반환할 수 있습니다.

> **Note**
>
> PlaceId은 원래의 검색 요청 및 GetPlace의 호출에서 다음 항목이 모두 동일한 경우에만 유효합니다.
>
> - 고객 AWS 계정
> - AWS 리전
> - 장소 색인 리소스에 지정된 데이터 제공자

일반적으로 GetPlace을 Amazon Location API와 함께 사용합니다. 다음 예는 이전 섹션의 제안 중 하나를 찾기 위한 GetPlace 요청입니다. 이 예는 장소 이름의 일부만 표시한 *kamp*를 기반으로 합니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/
places/AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
```

```
o3nqdlJZAdgcT2oWi1w9pS4wXX0k301vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

## 위치 주변 자동 완성

SearchPlaceIndexForSuggestions을 사용하여 자동 완성 장소 제안을 검색할 때 다음과 같은 선택적 파라미터를 추가하여 지역적으로 더 관련성이 높은 제안을 받을 수 있습니다.

- `BiasPosition` – 주변에서 검색하려는 위치. `[longitude, latitude]`과 같이 정의됩니다.

*## #### SearchPlaceIndexForSuggestions ### #### ## [32.5827, 0.3169] ### ## ## ### #### ## ### ## ## ##### ExamplePlaceIndex#####.*

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "BiasPosition": [32.5827,0.3169]
}
```

[-96.7977, 32.7776]과 같이 다른 항목 `BiasPosition`가 선택되면 동일한 항목 `Text`에 대해 반환되는 제안 내용이 달라질 수 있습니다.

## 경계 박스 내 자동 완성

다음과 같은 선택적 파라미터를 추가하여 지정된 경계 내에 있는 장소에 대한 제안만 받도록 자동 완성 검색 범위를 좁힐 수 있습니다.

- `FilterBBox` – 상자 경계 내의 좌표로 결과를 필터링하도록 지정하는 경계 상자. `[LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]`과 같이 정의됩니다.

> **Note**
>
> 요청은 `FilterBBox` 및 `BiasPosition` 파라미터를 모두 포함할 수 없습니다. 요청에서 두 파라미터를 모두 지정하면 `ValidationException` 오류가 반환됩니다.

다음 예시에서는 SearchPlaceIndexForSuggestions 요청을 사용하여 장소 색인 리소스에서 *ExamplePlaceIndex*부분 검색어와 일치하며 경계 상자 내에 *###* 장소 제안을 검색합니다.

- 경계 상자 남서쪽 모서리의 경도는 *32.5020*입니다.
- 경계 상자 남서쪽 모서리의 위도는 *0.2678*입니다.
- 경계 상자의 북동쪽 모서리 경도는 *32.6129*입니다.
- 경계 상자 북동쪽 모서리의 위도는 *0.3502*입니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterBBox": [
        32.5020, 0.2678,
        32.6129, 0.3502
    ]
}
```

[*-97.9651*, *32.0640*, *-95.1196*, *34.0436*]과 같이 다른 항목`FilterBBox`가 선택되면 동일한 `Text` 항목에 대해 반환되는 제안 내용이 달라집니다.

## 특정 국가 내 자동 완성

다음과 같은 선택적 파라미터를 추가하여 특정 국가 또는 국가 집합 내에 위치한 장소에 대한 제안만 받도록 자동 완성 검색 범위를 좁힐 수 있습니다.

- `FilterCountries` – 장소 제안을 검색하려는 국가. ISO 3166 3문자 국가 코드를 사용하여 요청 한 번에 최대 100개의 국가를 지정할 수 있습니다. 예를 들어, 오스트레일리아의 경우 `AUS`을 사용합니다.

다음 예시에서는 `SearchPlaceIndexForSuggestions` 요청을 사용하여 장소 색인 리소스에서 *ExamplePlaceIndex*부분 검색어 *Kamp#* 일치하며 우간다, 케냐 또는 탄자니아에 포함된 추천 장소를 검색합니다.

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterCountries": ["UGA", "KEN", "TZA"]
```

```
}
```

["##"]과 같이 다른 항목 FilterCountries가 선택된 경우 동일한 항목 Text에 대해 반환되는 제안 내용이 달라집니다.

## 응답의 예

다음은 텍스트 *kamp*를 사용한 SearchPlaceIndexForSuggestions 작업에 대한 자동 완성 제안 응답의 예입니다.


```
{
    "Summary": {
        "Text": "kamp",
        "MaxResults": 5,
        "DataSource": "Esri"
    },
    "Results": [
        {
            "Text": "Kampuchea",
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ"
        },
        {
            "Text": "Kampoul, Kabul, AFG",
            "PlaceId":
 "AQAAAIAAA1mxl_-9ffzXD07rBgo9fh6E01Pd1YKvuT5rz2qBDxqBkhTlgkeiOPR2s5sa3YBLxUqQI8bhymsYcu9R-
DkX3L9QSi3CB5LhNPu160iSFJo6H8S1CrxO3QsJALhrr9mdbg0R4R4YDywkhkeBlnbn7g5C5LI_wYx873WeQZuilwtsGm8j
UeXcb_bg"
        },
        {
            "Text": "Kampala, UGA",
            "PlaceId":
 "AQAAAIAAzZfZt3qMruKGObyhP6MM0pqy2L8SULlVWT7a3ertLBRS6Q5n7I4s9D7E0nRHADAj7mL7kvX1Q8HD-
mpuiATXNJ1Ix4_V_1B15zHe8jlYKMWvXbgbO8cMpgR2fqYqZMR1x-
dfBOO8OoqujKZldvPIDK1kNe3GwcaqvvMWWPMeaGd203brFynubAe-MmFF-Gjz-WBMfUy9og6MV7bkk6NGCA"
        },
        {
            "Text": "Kampar, Riau, IDN",
            "PlaceId": "AQAAAIAAvbXXx-
srOi111tHOkPdao0GF7WQ_KaZ444SEnevycp6Gtf_2JWgPfCE5bIQCYwya1uZQpX2a8YJoFm2K7Col4fLu7IK0yYOLhZx4k
        },
        {
```

```
            "Text": "Kampung Pasir Gudang Baru, Johor, MYS",
            "PlaceId":
 "AQAAAIAA4HLQHdjUDcaaXLE9wtNIT1cjQYLgkBnMoG2eNN0AaQ8PJoWabLRXmmPUaAj8MAD6vT0i6zqaun5Mixyj7vnYX
        }
    ]
}
```

## 장소 ID 사용

장소를 검색하면 결과 목록이 반환됩니다. 대부분의 결과에는 해당 결과에 대한 PlaceId이 포함됩니다. GetPlace 작업에서 PlaceId을 사용하여 해당 장소에 대한 정보(이름, 주소, 위치 또는 기타 세부 정보 포함)를 반환할 수 있습니다.

> **Note**
>
> 를 SearchPlaceIndexForSuggestions사용하면 모든 데이터 소스로 만든 모든 장소 색인에 대한 PlaceId 결과가 반환됩니다. SearchPlaceIndexForText를 SearchPlaceIndexForPosition사용하면 사용된 데이터 소스가 PlaceId HERE인 경우에만 a 를 반환합니다.

PlaceId 각각은 자신이 참조하는 장소를 고유하게 정의하지만, 시간이 지남에 따라 그리고 상황에 따라 한 장소에 여러 개의 PlaceId가 있을 수 있습니다. 다음 규칙은 PlaceId의 고유성과 수명을 설명합니다.

- PlaceId회신되는 호출은 사용자 AWS 계정, AWS 지역, PlaceIndex 리소스의 데이터 공급자별로 다릅니다. GetPlace이 세 가지 속성이 를 생성한 원래 호출과 일치하는 경우에만 결과를 찾을 수 PlaceId 있습니다.
- 장소에 대한 데이터가 변경되면 해당 장소에 대한 PlaceId도 변경됩니다. 예를 들어, 추천하는 사업체가 위치를 옮기거나 이름을 변경하는 경우.
- 반복되는 검색 호출에서 반환되는 PlaceId은 백엔드 서비스가 업데이트할 때 변경될 수 있습니다. 이전 PlaceId은 계속 검색되지만 새로 검색을 호출하면 다른 ID가 반환될 수 있습니다.

PlaceId은 문자열입니다. PlaceId의 길이에는 특별한 제한이 없습니다. 다음은 유효한 PlaceId의 예입니다.

```
AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

데이터가 변경된 장소(예: 영업이 중단된 사업장)에 대해 `PlaceId`로 `GetPlace`을 호출하면 `404`, `ResourceNotFound` 오류가 발생합니다. a를 `GetPlace` 사용하여 호출하면 유효하지 않거나 다른 호출과 같이 컨텍스트에 맞지 `PlaceId` 않는 것으로 호출하면 AWS 계정`400`, `ValidationException` 오류가 반환됩니다.

PlaceID는 후속 요청에서 사용할 수 있지만 PlaceID는 영구적인 식별자가 아니므로 연속적인 API 호출 간에 ID가 변경될 수 있습니다. 각 데이터 공급자에 대한 다음 PlaceID 동작을 참조하십시오.

- Esri: 장소 ID는 최소한 분기마다 변경됩니다. 이러한 변경이 적용되는 일반적인 기간은 3월, 6월, 9월, 12월입니다. 장소 ID는 일반적인 분기별 변경 시에도 변경될 수 있지만 그 빈도는 훨씬 줄어들 것입니다.
- HERE: 데이터를 최신 상태로 유지하려면 데이터를 일주일 이상 캐시하지 않는 것이 좋습니다. 릴리스 이후 공개되는 ID 변경은 주당 약 1~2회, 1% 미만이라고 가정할 수 있습니다.
- Grab: 다음과 같은 상황에서는 장소 ID가 만료되거나 유효하지 않게 될 수 있습니다.
  - 데이터 작업: Grab Map Ops는 현실 세계에서 폐쇄되었거나, 중복된 POI로 감지되거나, 정보가 올바르지 않은 경우 등 근거 정보를 기반으로 Grab POI 데이터베이스에서 POI를 제거할 수 있습니다. Grab은 매주 데이터를 Waypoint 환경과 동기화합니다.
  - 보간 POI: 보간 POI는 요청을 처리할 때 실시간으로 생성되는 임시 POI이며, 응답 시 필드에서 파생된 것으로 표시됩니다. `place.result_type` 보간된 POI의 정보는 최소 30일 동안 보관됩니다. 즉, 30일 이내에 장소 세부정보 API에서 장소 ID로 POI 세부정보를 얻을 수 있습니다. 30일이 지나면 보간된 POI (장소 ID 및 세부정보 모두) 가 만료되어 장소 세부정보 API에서 액세스할 수 없게 될 수 있습니다.

## 장소 카테고리 및 필터링 결과

장소가 분류됩니다. 예컨대, 비즈니스를 검색하면 해당 비즈니스가 `Restaurant`일 수 있습니다. 주소 검색 결과도 주소, 도로 또는 교차로와 일치하는지 여부에 따라 분류될 수 있습니다.

Amazon Location Service는 대체로 장소를 장소 유형으로 분류합니다. 관심 지역은 관심 지역 유형으로 더 분류됩니다.

> **Note**
>
> 모든 결과에 카테고리가 있는 것은 아닙니다.

카테고리를 사용하여 지오코딩 검색을 필터링할 수 있습니다.

## 필터링 결과

`SearchPlaceIndexForText`을 사용하는 경우 사용하려는 카테고리별로 반환되는 결과를 필터링할 수 있습니다. 예:

- “Hometown Coffee”라는 장소를 검색하고 커피숍으로 분류된 결과만 반환하려는 경우, `SearchPlaceIndexForText`을 호출하여 `FilterCategories` 파라미터에 관심 지역 카테고리 `Coffee Shop`을 포함시키면 됩니다.
- “123 Main St, WA, Anytown, 98123, USA”를 검색할 때는 결과를 주소로만 필터링하여, 예를 들면, 우편번호가 일치하는 항목이 검색되지 않도록 할 수 있습니다. `FilterCategories` 파라미터에 장소 유형 `AddressType`를 포함하여 주소만 필터링할 수 있습니다.

> **Note**
>
> 모든 데이터 공급자가 필터링을 지원하거나 동일한 방식으로 지원하는 것은 아닙니다. 자세한 내용은 데이터 공급자별 필터링 제한 섹션을 참조하세요.

다음 섹션에는 필터링할 수 있는 카테고리가 열거되어 있습니다.

## 카테고리

다음 목록은 Amazon Location Service가 분류 및 필터링에 사용하는 카테고리를 보여줍니다. 이러한 카테고리는 언어 파라미터가 다른 언어로 설정된 것과 상관없이 모든 언어에서 사용됩니다.

> **Note**
>
> Amazon Location Service는 데이터 제공자 카테고리를 이 카테고리 세트에 매핑합니다. 데이터 제공자가 Amazon Location Service 카테고리 목록에 포함되지 않은 카테고리에 장소를 등록하는 경우, 제공자 카테고리는 추가 카테고리로 결과에 포함됩니다.

장소 유형 – 이러한 유형은 결과를 찾는 데 사용된 일치 유형을 나타내는 데 사용됩니다.

- `AddressType` – 결과가 주소와 일치했을 때 반환됩니다.
- `StreetType` – 결과가 도로와 일치했을 때 반환됩니다.
- `IntersectionType` – 결과가 두 거리의 교차로와 일치했을 때 반환됩니다.
- `PointOfInterestType` – 결과가 사업체 또는 시민 소재지와 같은 관심 지점과 일치할 때 반환됩니다.
- `CountryType` – 결과가 국가 또는 주요 지역과 일치했을 때 반환됩니다.
- `RegionType` – 결과가 국가 내 지역(예: 주 또는 도) 과 일치했을 때 반환됩니다.
- `SubRegionType` – 결과가 카운티 또는 대도시 지역과 같은 국가 내의 하위 지역과 일치했을 때 반환됩니다.
- `MunicipalityType` – 결과가 도시 또는 마을과 일치했을 때 반환됩니다.
- `NeighborhoodType` – 결과가 도시 내 이웃 또는 지역과 일치했을 때 반환됩니다.
- `PostalCodeType` – 결과가 우편 번호와 일치했을 때 반환됩니다.

관심 지역 카테고리 – 이 카테고리는 관심 지역 검색 결과의 비즈니스 유형이나 위치를 나타내는 데 사용됩니다.

- `Airport`
- `Amusement Park`
- `Aquarium`
- `Art Gallery`
- `ATM`
- `Bakery`
- `Bank`
- `Bar`
- `Beauty Salon`
- `Bus Station`
- `Car Dealer`
- `Car Rental`
- `Car Repair`
- `Car Wash`

- Cemetery
- Cinema
- City Hall
- Clothing Store
- Coffee Shop
- Consumer Electronics Store
- Convenience Store
- Court House
- Dentist
- Embassy
- Fire Station
- Fitness Center
- Gas Station
- Government Office
- Grocery
- Higher Education
- Hospital
- Hotel
- Laundry
- Library
- Liquor Store
- Lodging
- Market
- Medical Clinic
- Motel
- Museum
- Nightlife
- Nursing Home
- Park
- Parking

- Pet Store
- Pharmacy
- Plumbing
- Police Station
- Post Office
- Religious Place
- Restaurant
- School
- Shopping Mall
- Sports Center
- Storage
- Taxi Stand
- Tourist Attraction
- Train Station
- Veterinary Care
- Zoo

## 데이터 공급자별 필터링 제한

모든 제공자에 동일한 필터링 기능이 있는 것은 아닙니다. 다음 표에서는 차이점을 설명합니다.

| 제공자 | 필터를 지원하는 API | 필터링이 지원되는 카테고리 | 반환 값 |
| --- | --- | --- | --- |
| Esri | SearchPlaceIndexForText , SearchPlaceIndexForSuggestions | 장소 유형 및 관심 지역 카테고리별로 필터링할 수 있습니다. | 카테고리는 SearchPlaceIndexForText ,SearchPlaceIndexForPosition 및 GetPlace에 의해 반환됩니다. |

| 제공자 | 필터를 지원하는 API | 필터링이 지원되는 카테고리 | 반환 값 |
| --- | --- | --- | --- |
| Here | `SearchPlaceIndexForText` , `SearchPlaceIndexForSuggestions` | 장소 유형별로만 필터링할 수 있습니다. | 카테고리는 `SearchPlaceIndexForText` 및 `SearchPlaceIndexForSuggestions` , `SearchPlaceIndexForPosition` , 그리고 `GetPlace`에 의해 반환됩니다. |
| Grab | 지원되지 않음 | 지원되지 않음 | 지원되지 않음 |
| Open Data(오픈 데이터) | 해당 사항 없음(장소 검색은 지원되지 않음) | 해당 사항 없음 | 해당 사항 없음 |

## Amazon Location Service용 Amazon Aurora PostgreSQL 사용자 정의 함수

Amazon Location Service를 사용하면 데이터베이스 테이블에 저장된 좌표와 주소를 사용하여 지리 공간 데이터를 정리하고 강화할 수 있습니다.

예:

- 지오코딩을 사용하여 주소를 좌표로 변환하여 데이터베이스 테이블에 저장된 주소의 데이터 공백을 정규화하고 간격을 메울 수 있습니다.
- 주소를 지오코딩하여 위치를 확보하고, 지정된 영역에 행을 표시하는 함수와 같은 데이터베이스 공간 함수와 함께 좌표를 사용할 수 있습니다.
- 보강된 데이터를 사용하여 자동 보고를 생성할 수 있습니다. 예를 들어 지정된 리전의 모든 디바이스를 설명하는 자동 보고서를 생성하거나 위치 업데이트를 전송할 때 고장률이 높은 리전을 설명하는 기계 학습용 자동 보고서를 생성할 수 있습니다.

이 튜토리얼에서는 Amazon Location Service를 사용하여 Amazon Aurora PostgreSQL 데이터베이스 테이블에 저장된 주소를 포맷하고 강화하는 방법을 보여줍니다.

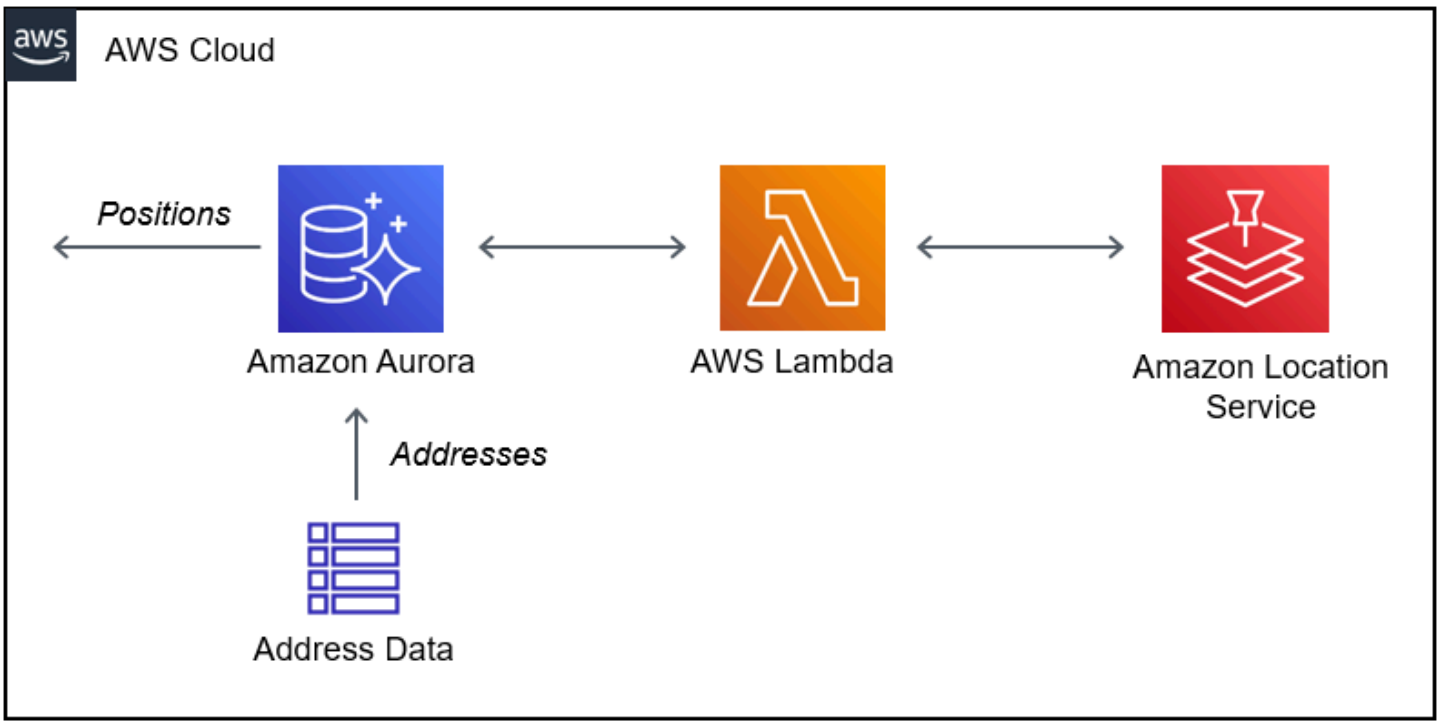
- Amazon Aurora PostgreSQL – MySQL 및 PostgreSQL과 호환되는 종합 관리형 관계형 데이터베이스 엔진으로, 기존 애플리케이션을 거의 변경하지 않고도 MySQL의 처리량을 최대 5배, PostgreSQL의 처리량을 최대 3배 출력합니다. 자세한 내용은 Amazon Aurora 사용 설명서의 Amazon Aurora란 무엇인가요?를 참조하세요.

> **Important**
>
> 이 튜토리얼의 결과 애플리케이션은 지오코딩 결과를 저장하는 장소 색인을 사용합니다. 지오코딩 결과 저장에 적용되는 요금에 대한 자세한 내용은 Amazon Location Service 요금을 참조하세요.

샘플 코드는 AWS CloudFormation템플릿이 포함된 Amazon Location Service 샘플 리포지토리에서 사용할 수 있습니다. GitHub

주제

## 개요




아키텍처에는 다음과 같은 통합이 포함됩니다.

- 이 솔루션은 Amazon Location 장소 색인 리소스를 사용하여 `SearchPlaceIndexForText` 작업을 사용한 지오코딩 쿼리를 지원합니다.
- AWS Lambda는 IAM 정책에서 AWS Lambda가 Amazon Location 지오코딩 작업인 `SearchPlaceIndexForText`을 호출할 수 있도록 권한을 부여할 때 주소를 지오코딩하는 Python Lambda를 사용합니다.
- SQL 사용자 정의 함수를 사용하여 지오코딩 Lambda 함수를 호출하는 권한을 Amazon Aurora PostgreSQL에 부여합니다.

## 사전 조건

시작하려면 다음 사전 요구 사항이 필요합니다.

- Amazon Aurora PostgreSQL 클러스터. 자세한 내용을 알아보려면 Amazon Aurora 사용 설명서의 Amazon Aurora DB 클러스터 생성을 참조하세요.

> **Note**
>
> Amazon Aurora 클러스터를 공개적으로 사용할 수 없는 경우, AWS 계정의 Virtual Private Cloud(VPC)에서 AWS Lambda에 연결하도록 Amazon Aurora도 구성해야 합니다. 자세한 설명은 Amazon Aurora PostgreSQL 액세스 권한을 AWS Lambda에 부여 섹션을 참조하세요.

- Amazon Aurora PostgreSQL 클러스터에 연결하기 위한 SQL 개발자 도구.

## 빠른 시작

이 튜토리얼의 단계를 수행하는 대신 빠른 스택을 실행하여 Amazon Location 작업 `SearchPlaceIndexForText`을 지원하는 AWS Lambda 함수를 배포할 수 있습니다. 그러면 Amazon Aurora가 AWS Lambda를 호출할 수 있도록 AWS 계정이 자동으로 구성됩니다.

AWS 계정을 구성한 후에는 다음을 수행해야 합니다.

- Amazon Aurora에 Lambda 기능을 추가합니다. Amazon Aurora PostgreSQL 액세스 권한을 AWS Lambda에 부여의 Amazon Aurora DB 클러스터에 IAM 역할 추가를 참조하세요.
- 사용자 정의 함수를 데이터베이스에 로드합니다. AWS Lambda 함수 호출 섹션을 참조하세요.


## 장소 색인 리소스 만들기

먼저 지오코딩 쿼리를 지원하는 장소 색인 리소스를 만드는 것부터 시작합니다.

1. https://console.aws.amazon.com/location/에서 Amazon Location Service 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 장소 색인을 선택합니다.
3. 다음 입력란을 작성합니다.
   - 이름 – 장소 색인 리소스의 이름을 입력합니다. 예를 들면 다음과 같습니다 *AuroraPlaceIndex*. 최대 100자입니다. 유효한 항목에는 영숫자 문자, 하이픈, 마침표 및 밑줄이 포함됩니다.

- 설명 – 선택적 설명을 입력합니다. 예를 들어, *Place index for Amazon Aurora*가 있습니다.

4. 데이터 공급자에서 장소 색인 리소스와 함께 사용할 데이터 공급자를 선택합니다. 선호하는 항목이 없는 경우, *Esri*로 시작하는 것이 좋습니다.
5. 데이터 스토리지 옵션에서 예, 결과가 저장됩니다를 지정합니다. 이는 지오코딩 결과를 데이터베이스에 저장하려는 것임을 나타냅니다.
6. (선택 사항) 태그 아래에 태그 키 및 값을 입력합니다. 그러면 새 장소 색인 리소스에 태그가 추가됩니다. 자세한 내용을 알아보려면 리소스 태그 지정을 참조하세요.
7. 장소 색인 생성을 선택합니다.

## 지오코딩을 위한 AWS Lambda 함수 생성

Amazon Aurora PostgreSQL과 Amazon Location Service 간에 연결을 생성하려면 데이터베이스 엔진의 요청을 처리하는 AWS Lambda 함수가 필요합니다. 이 함수는 Lambda 사용자 정의 함수 이벤트를 변환하고 Amazon Location 작업 `SearchPlaceIndexForText`을 호출합니다.

AWS Lambda 콘솔, AWS Command Line Interface 또는 AWS Lambda API를 사용하여 함수를 생성할 수 있습니다.

콘솔을 사용하여 Lambda 사용자 정의 함수를 생성하려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 왼쪽 탐색에서 함수를 선택합니다.
3. 함수 생성을 선택하고 새로 작성이 선택되어 있는지 확인합니다.
4. 다음 입력란을 작성합니다.

   - 함수 이름 – 함수에 고유한 이름을 입력합니다. 유효한 항목에는 공백 없는 영숫자 문자, 하이픈 및 밑줄이 포함됩니다. 예를 들어, *AuroraGeocoder*.
   - 런타임 - *Python 3.8*을 선택합니다.

5. 함수 생성을 선택합니다.
6. Code 탭을 선택하여 편집기를 엽니다.
7. `lambda_function.py`의 자리 표시자 코드를 다음과 같이 덮어씁니다.

```
from os import environ

import boto3
```

```
from botocore.config import Config

# load the place index name from the environment, falling back to a default
PLACE_INDEX_NAME = environ.get("PLACE_INDEX_NAME", "AuroraPlaceIndex")

location = boto3.client("location", config=Config(user_agent="Amazon Aurora
 PostgreSQL"))

"""
This Lambda function receives a payload from Amazon Aurora and translates it to
an Amazon Location `SearchPlaceIndex` call and returns the results as-is, to be
post-processed by a PL/pgSQL function.
"""
def lambda_handler(event, context):
    kwargs = {}

    if event.get("biasPosition") is not None:
        kwargs["BiasPosition"] = event["biasPosition"]

    if event.get("filterBBox") is not None:
        kwargs["FilterBBox"] = event["filterBBox"]

    if event.get("filterCountries") is not None:
        kwargs["FilterCountries"] = event["filterCountries"]

    if event.get("maxResults") is not None:
        kwargs["MaxResults"] = event["maxResults"]

    return location.search_place_index_for_text(
        IndexName=PLACE_INDEX_NAME,
        Text=event["text"],
        **kwargs)["Results"]
```

8. 장소 색인의 이름을 다른 이름으로 지정한 경우 *AuroraPlaceIndex*, 리소스 이름을 PLACE_INDEX_NAME 지정할 환경 변수를 생성하세요.

   - 구성 탭에서 환경 변수를 선택합니다.
   - 편집을 선택한 다음 환경 변수 추가를 선택합니다.
   - 키에 PLACE_INDEX_NAME를 입력합니다.
   - 값에 장소 색인 리소스의 이름을 입력합니다.

9. [배포]를 선택하여 업데이트된 함수를 저장합니다.

10. 테스트 드롭다운 메뉴에서 테스트 이벤트 구성을 선택합니다.
11. 새로운 테스트 이벤트 생성을 선택하세요.
12. 다음 테스트 이벤트를 입력합니다.

```
{
  "text": "Baker Beach",
  "biasPosition": [-122.483, 37.790],
  "filterCountries": ["USA"]
}
```

13. Lambda 함수를 테스트하려면 테스트를 선택합니다.
14. 구성 탭을 선택합니다.
15. 일반 구성에서 권한을 선택합니다.
16. 실행 역할에서 하이퍼링크된 역할 이름을 선택하여 Lambda 함수에 Amazon Location Service 권한을 부여합니다.
17. 권한 탭에서 권한 추가 드롭 다운을 선택한 다음, 인라인 정책 생성을 선택합니다.
18. JSON 탭을 선택합니다.
19. 다음 IAM 정책을 추가합니다.

    - 다음 정책은 장소 색인 `SearchPlaceIndexForText` 리소스에 보낼 수 있는 권한을 *AuroraPlaceIndex*부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "geo:SearchPlaceIndexForText",
      "Resource": "arn:aws:geo:<Region>:<AccountId>:place-index/AuroraPlaceIndex"
    }
  ]
}
```

20. 정책 검토를 선택합니다.
21. 정책 이름을 입력합니다. 예를 들어, *AuroraPlaceIndexReadOnly*.
22. 정책 생성을 선택합니다.

## Amazon Aurora PostgreSQL 액세스 권한을 AWS Lambda에 부여

Amazon Aurora PostgreSQL이 AWS Lambda 함수를 호출하려면 먼저 액세스 권한을 부여해야 합니다.

Amazon Aurora PostgreSQL 클러스터에 공개적으로 액세스할 수 없는 경우, Amazon Aurora가 Lambda 함수를 호출하도록 하려면 먼저 AWS Lambda에 대한 VPC 엔드포인트를 생성해야 합니다.

AWS Lambda에 대한 VPC 엔드포인트 생성

> **Note**
>
> 이 단계는 Amazon Aurora PostgreSQL 클러스터에 공개적으로 액세스할 수 없는 경우에만 필요합니다.

1. Amazon Virtual Private Cloud Console을 엽니다.
2. 왼쪽 탐색에서 엔드포인트를 선택합니다.
3. Create endpoint(엔드포인트 생성)을 선택합니다.
4. 서비스 이름 필터에 “lambda”를 입력한 다음 `com.amazonaws.<region>.lambda`를 선택합니다.
5. Aurora 클러스터를 포함하는 VPC를 선택합니다.
6. 각 가용 영역마다 서브넷을 하나씩 선택합니다.
7. 보안 그룹 필터에서 “default” 또는 Aurora 클러스터가 속한 보안 그룹의 이름을 입력한 다음 보안 그룹을 선택합니다.
8. Create endpoint(엔드포인트 생성)을 선택합니다.

AWS Lambda 함수 호출 권한을 부여하는 IAM 정책을 생성합니다.

1. IAM 콘솔(IAM console)을 엽니다.
2. 왼쪽 탐색에서 액세스 관리를 확장하여 정책을 선택합니다.
3. 정책 생성을 선택합니다.
4. JSON 탭에 다음 정책을 입력합니다.

   - 다음은 `AuroraGeocoder` AWS Lambda 함수를 호출하는 Amazon Aurora PostgreSQL 권한을 제공하는 IAM 정책의 예제입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:<Region>:<AccountId>:function:AuroraGeocoder"
            ]
        }
    ]
}
```

5. (선택 사항) 다음: 태그를 선택하여 선택적 태그를 추가합니다.
6. 다음: 검토를 선택합니다.
7. 정책을 검토하고 정책에 대한 다음 세부 정보를 입력합니다.
   - 이름 - 영숫자와 '+=, .@-_' 문자를 사용합니다. 최대 128자입니다. 예를 들어, *AuroraGeocoderInvoke*.
   - 설명 – 선택적 설명을 입력합니다. 영숫자와 '+=, .@-_' 문자를 사용합니다. 최대 1000자입니다.
8. 정책 생성을 선택합니다. 정책을 IAM 역할에 연결하는 데 사용하는 이 정책의 ARN을 기록해 두십시오.

### IAM 역할을 생성하여 Amazon Relational Database Service (RDS)에 권한을 부여하기

IAM 역할을 생성하면 Amazon Aurora PostgreSQL이 사용자 대신 Lambda 함수에 액세스하는 역할을 맡을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하십시오.

다음 예제는 다음과 같은 역할을 생성하는 AWS CLI 명령입니다 *AuroraGeocoderInvokeRole*.

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
```

```
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

IAM 정책을 IAM 역할에 연결합니다.

IAM 역할이 있는 경우 생성한 IAM 정책을 연결합니다.

다음 예제는 정책을 *AuroraGeocoderInvoke*역할에 *AuroraGeocoderInvokeRole*연결하는 AWS CLI 명령입니다.

```
aws iam attach-role-policy  --policy-arn AuroraGeocoderInvoke  --role-
name AuroraGeocoderInvokeRole
```

IAM 역할을 Amazon Aurora DB 클러스터에 추가

다음 예제는 라는 Amazon Aurora PostgreSQL DB 클러스터에 IAM 역할을 추가하는 AWS CLI 명령입니다. *MyAuroraCluster*

```
aws rds add-role-to-db-cluster \
--db-cluster-identifier MyAuroraCluster \
--feature-name Lambda \
--role-arn AuroraGeocoderInvokeRole   \
--region your-region
```

## AWS Lambda 함수 호출

지오코딩 Lambda 함수를 호출할 권한을 Amazon Aurora PostgreSQL에 부여한 후에는 지오코딩 AWS Lambda 함수를 호출하는 Amazon Aurora PostgreSQL 사용자 정의 함수를 생성할 수 있습니다. 자세한 내용은 Amazon Aurora 사용 설명서의 Amazon Aurora PostgreSQL DB 클러스터에서 AWS Lambda 함수 호출을 참조하십시오.

필요한 PostgreSQL 확장을 설치합니다.

필수 PostgreSQL 확장 `aws_lambda` 및 `aws _commons` 확장을 설치하려면 Amazon Aurora 사용 설명서의 Lambda 함수 사용에 대한 개요를 참조하세요.

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
```

필요한 PostGIS 확장을 설치합니다.

PostGIS는 공간 정보를 저장하고 관리하기 위해 PostgreSQL을 확장한 것입니다. 자세한 내용은 Amazon 관계형 데이터베이스 서비스 사용 설명서의 PostGIS 확장 작업을 참조하세요.

```
CREATE EXTENSION IF NOT EXISTS postgis;
```

Lambda 함수를 호출하는 SQL 사용자 정의 함수 생성

SQL 편집기에서 새 사용자 정의 함수를 `f_SearchPlaceIndexForText` 생성하여 함수를 호출합니다. *AuroraGeocoder*

```
CREATE OR REPLACE FUNCTION f_SearchPlaceIndexForText(
  text text,
  bias_position geometry(Point, 4326) DEFAULT NULL,
  filter_bbox box2d DEFAULT NULL,
  filter_countries text[] DEFAULT NULL,
  max_results int DEFAULT 1
)
 RETURNS TABLE (
   label text,
   address_number text,
   street text,
   municipality text,
   postal_code text,
   sub_region text,
   region text,
   country text,
   geom geometry(Point, 4326)
 )
 LANGUAGE plpgsql
 IMMUTABLE
AS $function$
begin
    RETURN QUERY
    WITH results AS (
      SELECT json_array_elements(payload) rsp
      FROM aws_lambda.invoke(
        aws_commons.create_lambda_function_arn('AuroraGeocoder'),
        json_build_object(
          'text', text,
          'biasPosition',
          CASE WHEN bias_position IS NOT NULL THEN
            array_to_json(ARRAY[ST_X(bias_position), ST_Y(bias_position)])
          END,
```

```
          'filterBBox',
          CASE WHEN filter_bbox IS NOT NULL THEN
            array_to_json(ARRAY[ST_XMin(filter_bbox), ST_YMin(filter_bbox),
 ST_XMax(filter_bbox), ST_YMax(filter_bbox)])
          END,
          'filterCountries', filter_countries,
          'maxResults', max_results
        )
      )
    )
    SELECT
      rsp->'Place'->>'Label' AS label,
      rsp->'Place'->>'AddressNumber' AS address_number,
      rsp->'Place'->>'Street' AS street,
      rsp->'Place'->>'Municipality' AS municipality,
      rsp->'Place'->>'PostalCode' AS postal_code,
      rsp->'Place'->>'SubRegion' AS sub_region,
      rsp->'Place'->>'Region' AS region,
      rsp->'Place'->>'Country' AS country,
      ST_GeomFromGeoJSON(
        json_build_object(
          'type', 'Point',
            'coordinates', rsp->'Place'->'Geometry'->'Point'
        )
      ) geom
    FROM results;
end;
$function$;
```

SQL 함수를 호출하여 Aurora에서 지오코딩하기

SQL 문을 실행하면 Lambda *AuroraGeocoder*함수가 호출됩니다. 이 함수는 데이터베이스의 데이터베이스 테이블에서 주소 레코드를 가져와 장소 인덱스 Amazon Aurora PostgreSQL 리소스를 사용하여 지오코딩합니다.

> **Note**
>
> Amazon Aurora PostgreSQL은 SQL 사용자 정의 함수를 호출할 때마다 Lambda 함수를 호출합니다.
> 50개 행을 지오코딩하는 경우 Amazon Aurora PostgreSQL은 Lambda 함수를 50번 호출합니다. 각 행마다 한 번 호출합니다.

다음 f_SearchPlaceIndexForText SQL 함수는 *AuroraGeocoder*Lambda 함수를 통해 아마존 로케이션의 SearchPlaceIndexForText API에 요청을 보냅니다. 이 함수는 ST_AsText(geom)가 텍스트로 변환되는 PostGIS 지오메트리인 geom 열을 반환합니다.

```
SELECT *, ST_AsText(geom)
FROM f_SearchPlaceIndexForText('Vancouver, BC');
```

기본적으로 반환에는 하나의 행이 포함됩니다. MaxResults 한도까지 추가 행을 요청하려면 BiasPosition을 제공하고 결과를 캐나다로 제한하면서 다음 SQL 문을 실행하세요.

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', ST_MakePoint(-123.113, 49.260), null,
 '{"CAN"}', 5);
```

경계 상자를 사용하여 결과를 필터링하려면 Box2D를 filter_bbox로 전달합니다.

- FilterBBox – 경계 상자 내의 위치를 반환하여 결과를 필터링합니다. 이는 선택 가능한 파라미터입니다.

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', null, 'BOX(-139.06 48.30, -114.03
 60.00)'::box2d, '{"CAN"}', 5);
```

PostGIS 유형과 함수에 대한 자세한 내용은 PostGIS 참조를 참조하세요.

## 주소 데이터를 포함하는 데이터베이스 강화

다음 데이터가 다음 열로 구분된 데이터베이스 테이블이 주어지면 Amazon Location 작업 SearchPlaceIndexForText을 사용하여 형식화된 주소를 생성하는 동시에 정규화 및 지오코딩 수행할 수 있습니다.

- id
- address
- city
- state
- zip

```
WITH source_data AS (
  SELECT
    id,
    address || ', ' || city || ', ' || state || ', ' || zip AS formatted_address
  FROM addresses
),
geocoded_data AS (
  SELECT
    *,
    (f_SearchPlaceIndexForText(formatted_address)).*
  FROM source_data
)
SELECT
  id,
  formatted_address,
  label normalized_address,
  ST_Y(geom) latitude,
  ST_X(geom) longitude
FROM geocoded_data
-- limit the number of rows that will be geocoded; remove this to geocode the entire
 table
LIMIT 1;
```

다음 예시는 결과 데이터테이블 행 하나를 보여줍니다.

```
 id |      formatted_address         |            normalized_address              |
 latitude     |     longitude
----+--------------------------------+--------------------------------------------
+------------------+-------------------
 42 | 123 Anytown Ave N, Seattle, WA | 123 Anytown Ave N, Seattle, WA, 12345, USA |
 47.6223000127926 | -122.336745971039
(1 row)
```

데이터베이스 테이블 업데이트 및 열 채우기

다음 예시에서는 테이블을 업데이트하고 SearchPlaceIndexForText 쿼리의 결과로 열을 채웁니다.

```
WITH source_data AS (
  -- select rows that have not been geocoded and created a formatted address for each
  SELECT
    id,
```

```
    address || ', ' || city || ', ' || state || ', ' || zip AS formatted_address
  FROM addresses
  WHERE label IS NULL
  -- limit the number of rows that will be geocoded; remove this to geocode the entire
 table
  LIMIT 1
),
geocoded_data AS (
  -- geocode each row and keep it linked to the source's ID
  SELECT
    id,
    (f_SearchPlaceIndexForText(formatted_address)).*
  FROM source_data
)
UPDATE addresses
-- populate columns
SET
  normalized_address = geocoded_data.label,
  latitude = ST_Y(geocoded_data.geom),
  longitude = ST_X(geocoded_data.geom)
FROM geocoded_data
-- ensure that rows match
WHERE addresses.id = geocoded_data.id;
```

### 다음 단계

샘플 코드는 AWS CloudFormation템플릿이 포함된 Amazon Location Service 샘플 리포지토리에서 사용할 수 있습니다. GitHub

## 장소 색인 리소스 관리

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 장소 색인 리소스를 관리할 수 있습니다.

### 장소 색인 리소스를 나열합니다.

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 장소 색인 리소스 목록을 볼 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 장소 색인 자원 목록을 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 장소 색인을 선택합니다.
3. 내 장소 색인에서 장소 색인 리소스 목록을 확인합니다.

API

Amazon Location Places API의 `ListPlaceIndexes` 작업을 사용합니다.

다음 예시는 AWS 계정의 장소 색인 리소스 목록을 가져오는 API 요청입니다.

```
POST /places/v0/list-indexes
```

다음은 `ListPlaceIndexes`에 대한 응답의 예입니다:

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-30T01:38:36Z,
         "DataSource": "Esri",
         "Description": "string",
         "IndexName": "ExamplePlaceIndex",
         "UpdateTime": 2020-10-30T01:40:36Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

`list-place-indexes` 명령을 사용합니다.

다음 예시는 AWS 계정의 장소 색인 리소스 목록을 가져오는 AWS CLI입니다.

```
aws location list-place-indexes
```

## 장소 색인 리소스 세부 정보 가져오기

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정의 모든 장소 색인 리소스에 대한 세부 정보를 확인할 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 장소 색인 리소스의 세부 정보를 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 장소 색인을 선택합니다.
3. 내 장소 색인에서 대상 장소 색인 리소스의 이름 링크를 선택합니다.

API

Amazon Location Place API의 DescribePlaceIndex 작업을 사용합니다.

다음은 장소 색인 리소스 세부 정보를 가져오기 위한 API *ExamplePlaceIndex*요청입니다.

```
GET /places/v0/indexes/ExamplePlaceIndex
```

다음은 DescribePlaceIndex에 대한 응답의 예입니다:

```
{
   "CreateTime": 2020-10-30T01:38:36Z,
   "DataSource": "Esri",
   "DataSourceConfiguration": {
      "IntendedUse": "SingleUse"
   },
   "Description": "string",
   "IndexArn": "arn:aws:geo:us-west-2:123456789012:place-indexes/ExamplePlaceIndex",
   "IndexName": "ExamplePlaceIndex",
   "Tags": {
      "string" : "string"
   },
   "UpdateTime": 2020-10-30T01:40:36Z
}
```

CLI

describe-place-index 명령을 사용합니다.

다음은 장소 색인 리소스 세부 정보를 AWS CLI 가져오는 예제입니다 *ExamplePlaceIndex*.

```
aws location describe-place-index \
    --index-name "ExamplePlaceIndex"
```

# 장소 색인 리소스 삭제

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정에서 장소 색인 리소스를 삭제할 수 있습니다.

## Console

Amazon Location 콘솔을 사용하여 장소 색인 리소스를 삭제하려면

> ⚠ Warning
> 이 작업은 리소스를 영구적으로 삭제합니다.

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 장소 색인을 선택합니다.
3. 내 장소 색인에서 대상 장소 색인 리소스를 선택합니다.
4. 장소 색인 삭제를 선택합니다.

## API

Amazon Location Places API의 `DeletePlaceIndex` 작업을 사용합니다.

다음은 장소 색인 리소스를 삭제하기 위한 API *ExamplePlaceIndex*요청입니다.

```
DELETE /places/v0/indexes/ExamplePlaceIndex
```

다음은 `DeletePlaceIndex`에 대한 성공적인 응답의 예입니다.

```
HTTP/1.1 200
```

## CLI

`delete-place-index` 명령을 사용합니다.

다음 예시는 장소 색인 리소스를 삭제하는 AWS CLI *ExamplePlaceIndex*명령입니다.

```
aws location delete-place-index \
    --index-name "ExamplePlaceIndex"
```

# Amazon Location Service를 사용하여 경로 계산

Amazon Location에서는 경로 계산기 리소스를 생성하고 구성하여 경로를 계산할 데이터 공급자를 선택할 수 있습니다.

경로 계산기 리소스를 사용하여 AWS SDK 또는 REST API 엔드포인트를 사용해 특정 파라미터가 지정된 경로를 계산할 수 있습니다. 이 경로 계산기 리소스를 사용하여 다양한 교통 수단, 회피, 교통 상황에 따라 출발지, 도착지 및 최대 23개 중간 지점 간 경로를 계산할 수 있습니다.

또한 경로 계산기 리소스를 사용하여 경로 매트릭스를 계산하여 경로 계획 알고리즘 또는 제품에 대한 입력을 생성할 수 있습니다. 출발 위치 세트와 도착 위치 세트 사이의 이동 시간과 이동 거리를 계산합니다. 경로 계획 소프트웨어는 이러한 시간 및 거리 데이터를 사용하여 경로 또는 경로 세트를 최적화할 수 있습니다. 예를 들어 여러 배송 경로를 계획하고 있고 각 경유지에 가장 적합한 경로와 시간을 찾으려는 경우가 있습니다. 다양한 교통 수단, 회피, 교통 상황에 따라 경로 매트릭스를 계산할 수 있습니다.

> **Note**
>
> 경로 개념에 대한 개요는 경로 항목을 참조하세요.

주제



## 사전 조건

경로 계산을 시작하기 전에 사전 필수 단계를 따르세요.

주제

- 요청 인증

## 경로 계산기 리소스 생성

경로를 계산하려면 먼저 AWS 계정에 경로 계산기 리소스를 생성해야 합니다.

경로 계산기 리소스를 생성할 때 다음 사용 가능한 데이터 공급자 중에서 선택할 수 있습니다.

1. Esri – 관심 리전의 Esri 커버리지에 대한 추가 정보를 알아보려면 도로 네트워크 및 교통량 커버리지에 대한 Esri 세부 정보를 참조하세요.
2. HERE Technologies - 관심 리전의 HERE 적용 범위에 대한 자세한 내용은 HERE car routing coverage(HERE 자동차 경로 적용 범위) 및 HERE truck routing coverage(HERE 트럭 경로 적용 범위)를 참조하세요.
3. Grab – Grab의 적용 범위에 대한 자세한 내용은 대상 국가/리전 및 지역 항목을 참조하세요.

> **Note**
>
> 애플리케이션이 배송 차량 또는 직원 등 업무에서 사용하는 자산을 추적하거나 라우팅하는 경우 Esri를 지리적 위치 제공업체로 사용해서는 안 됩니다. 자세한 내용은 AWS 서비스 약관의 섹션 82를 참조하세요.

이 작업을 위해 Amazon Location Service 콘솔, AWS CLI 또는 Amazon Location API를 사용할 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 경로 계산기 리소스를 만들려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 경로 계산기를 선택합니다.
3. 경로 계산기 생성을 선택합니다.
4. 다음 입력란을 작성합니다.
   - 이름 - 경로 계산기 리소스의 이름을 입력합니다. 예를 들어, `ExampleCalculator`. 최대 100자입니다. 유효한 항목에는 영숫자 문자, 하이픈, 마침표 및 밑줄이 포함됩니다.
   - 설명 – 선택적 설명을 입력합니다.

5. 데이터 공급자의 경우 경로 계산기로 사용할 데이터 공급자를 선택합니다.
6. (선택 사항) 태그 아래에 태그 키 및 값을 입력합니다. 그러면 새 경로 계산기 리소스에 태그가 추가됩니다. 자세한 내용을 알아보려면 리소스 태그 지정을 참조하세요.
7. 경로 계산기 생성을 선택합니다.

API

Amazon Location API를 사용하여 경로 계산기 리소스를 만들려면

Amazon Location Places API의 `CreateRouteCalculator` 작업을 사용합니다.

다음은 데이터 공급자 *`Esri`*# *`ExampleCalculator`*사용하여 호출되는 경로 계산기 리소스를 생성하기 위한 API 요청입니다.

```
POST /routes/v0/calculators
Content-type: application/json

{
   "CalculatorName": "ExampleCalculator",
   "DataSource": "Esri",
   "Description": "string",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

AWS CLI 명령을 사용하여 경로 계산기 리소스를 만들려면

`create-route-calculator` 명령을 사용합니다.

다음 예시에서는 *`Esri`*# 데이터 공급자로 *`ExampleCalculator`*사용하여 라는 경로 계산기 리소스를 생성합니다.

```
aws location \
  create-route-calculator \
  --calculator-name "ExampleCalculator" \
  --data-source "Esri" \
  --tags Tag1=Value1
```

> **Note**
>
> 청구는 사용량에 따라 달라집니다. 다른 AWS 서비스 사용 시 요금이 부과될 수 있습니다. 자세한 정보는 Amazon Location Service 가격을 참조하세요.

## 요청 인증

경로 계산기 리소스를 생성하고 애플리케이션에 위치 기능을 빌드할 준비가 되었으면 요청을 인증할 방법을 선택합니다

- 서비스에 액세스하는 방법을 알아보려면 Amazon Location Service 액세스를 참조하세요.
- 익명 사용자가 있는 웹 사이트가 있는 경우 API 키 또는 Amazon Cognito를 사용하는 것이 좋습니다.

  예

  다음 예는 권한 부여를 위한 API 키 사용, AWS JavaScript SDK v3 및 Amazon Location을 사용하는 방법을 보여줍니다. JavaScript 인증 도우미

```
import { LocationClient, CalculateRouteCommand } from "@aws-sdk/client-location";
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";

const apiKey = "v1.public.your-api-key-value"; // API key

// Create an authentication helper instance using an API key
const authHelper = await withAPIKey(apiKey);

const client = new LocationClient({
  region: "<region>", // region containing Cognito pool
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  CalculatorName: "ExampleCalculator",
  DeparturePosition: [-123.4567, 45.6789],
  DestinationPosition: [-123.123, 45.234],
};

const command = new CalculateRouteCommand(input);
```

```
const response = await client.send(command);
```

# 경로 계산

Amazon Location Service를 사용하면 다양한 교통 수단, 회피, 교통 상황에 따라 경로를 따라 최대 23개의 중간 지점이 있는 출발지와 도착지 사이의 경로를 계산할 수 있습니다.

> **Note**
>
> 먼저 경로 계산기 리소스를 생성하고 Amazon Location에 대한 요청에 대한 인증을 설정해야 합니다. 자세한 설명은 사전 조건 섹션을 참조하세요.

## 경로 계산 시작

`CalculateRoute` 작업을 사용하여 간단한 요청을 제출합니다. 단순 요청에는 다음과 같은 필수 필드가 포함됩니다.

- `DeparturePosition` – 경로를 계산할 시작 위치입니다. `[longitude, latitude]`과 같이 정의됩니다.
- `DestinationPosition` – 경로를 계산할 끝 위치입니다. `[longitude, latitude]`과 같이 정의됩니다.

> **Note**
>
> 도로에 있지 않은 출발 위치 또는 도착 위치를 지정하는 경우 Amazon Location은 해당 위치를 가장 가까운 도로로 이동합니다.

요청에 중간 지점, 출발 시간, 이동 수단을 지정할 수도 있습니다.

AWS CLI 또는 Amazon Location API를 사용할 수 있습니다.

API

다음은 경로 계산기 리소스를 사용한 `CalculateRoute` 요청입니다. *`ExampleCalculator`* 요청은 출발 위치 [*`-122.7565`*, *`49.0021`*]에서 도착 위치 [*`-122.3394`*, *`47.6159`*]까지의 경로를 계산하도록 지정합니다.

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159]
}
```

AWS CLI

다음 예제는 경로 계산기 리소스를 사용하는 `calculate-route` *ExampleCalculator*명령입니다. 요청은 출발 위치 [*-122.7565*, *49.0021*]에서 도착 위치 [*-122.3394*, *47.6159*]까지의 경로를 계산하도록 지정합니다.

```
aws location \
    calculate-route \
        --calculator-name ExampleCalculator \
        --departure-position -122.7565 49.0021 \
        --destination-position -122.3394 47.6159
```

기본적으로 응답은 킬로미터 단위로 `Distance`을 반환합니다. 다음 선택적 파라미터를 사용하여 측정 단위를 마일로 변경할 수 있습니다.

- `DistanceUnit` – 거리 결과에 사용할 단위 체계를 지정합니다.

Example

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "DistanceUnit": "Miles"
}
```

## 중간 지점 설정

경로를 계산할 때 중간 지점 위치를 사용하여 출발 위치와 도착 위치 사이의 중간 경유지를 최대 23개까지 지정할 수 있습니다.

- `WaypointPositions` – 출발 위치와 도착 위치 사이의 경로를 따라 포함시킬 중간 위치의 정렬된 목록을 지정합니다.

> **Note**
>
> 도로에 없는 중간 지점 위치를 지정하는 경우 Amazon Location은 해당 위치를 가장 가까운 도로로 이동합니다.

Example

다음 `CalculateRoute` 요청은 중간 지점 2개가 있는 경로를 계산합니다.

- 출발 위치는 [-122.7565, 49.0021]이고 도착 위치는 [-122.3394, 47.6159] 입니다.
- 요청 파라미터 `WaypointPositions`의 경우:
  - 첫 번째 경유지 위치는 [*-122.1884, 48.0936*]입니다.
  - 두 번째 경유지 위치는 [*-122.3493, 47.6205*]입니다.
- 이 두 중간 지점 사이에 레그 라인스트링 지오메트리를 포함하려면 다음 선택적 파라미터를 *true*로 설정합니다.
  - `IncludeLegGeometry` – 응답에 한 쌍의 위치 사이에 있는 각 경로의 지오메트리를 포함합니다.

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "WaypointPositions":[
        [-122.1884,48.0936],
        [-122.3493,47.6205]
    ],
   "IncludeLegGeometry": true
}
```

## 응답의 예

다음은 `IncludeLegGeometry`를 *true*로 설정하고 Amazon Location Routes API에서 `CalculateRoute` 작업을 호출할 때 해당 응답이 포함된 요청 예시입니다. 여기에는 응답에 한 쌍의 위치 사이의 각 경로의 라인스트링 지오메트리를 포함합니다.

## Example request

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "IncludeLegGeometry": true
}
```

## Example response


```
{
   "Legs": [
      {
         "Distance": 178.5,
         "DurationSeconds": 6480,
         "EndPosition": [-122.3394,47.6159],
         "Geometry": {
            "LineString": [
               [-122.7565,49.0021],
               [-122.3394,47.6159]
            ]
         },
         "StartPosition": [-122.7565,49.0021],
         "Steps": [
            {
               "Distance": 178.5,
               "DurationSeconds": 6480,
               "EndPosition": [-122.3394,47.6159],
               "GeometryOffset": 0,
               "StartPosition": [-122.7565,49.0021]
            }
         ]
      }
   ],
   "Summary": {
      "DataSource": "Esri",
      "Distance": 178.5,
      "DistanceUnit": "Kilometers",
      "DurationSeconds": 6480,
      "RouteBBox": [
        -122.7565,49.0021,
```

```
            -122.3394,47.6159
        ]
       }
}
```

## 경로 매트릭스를 사용한 경로 계획

Amazon Location Service를 사용하여 경로 계획 및 최적화 소프트웨어에 대한 입력을 생성할 수 있습니다. 출발 위치 세트와 도착 위치 세트 사이의 경로에 대해 이동 시간 및 이동 거리를 포함한 경로 결과를 생성할 수 있습니다.

예를 들어 출발 위치 A와 B, 도착 위치 X와 Y가 주어지면 Amazon Location Service는 A에서 X, A에서 Y, B에서 X, B에서 Y까지, B에서 Y까지의 경로에 대한 이동 시간과 이동 거리를 반환합니다.

다양한 교통 수단, 회피, 교통 상황을 고려하여 경로를 계산할 수 있습니다. 예를 들어, 차량이 10.7m(35피트) 길이의 트럭이라고 지정하면 계산된 경로에서는 이러한 제한을 사용하여 이동 시간과 이동 거리를 결정합니다.

반환되는 결과 수와 계산된 경로 수는 출발 위치 수에 목적지 위치 수를 곱한 값입니다. 서비스에 대한 각 요청이 아니라 계산된 각 경로에 대해 요금이 부과되므로, 출발지가 10개이고 도착지가 10개인 경로 매트릭스는 100개의 경로로 청구됩니다.

### 경로 매트릭스 계산

출발 위치 세트와 도착 위치 세트 사이의 경로 매트릭스를 계산할 수 있습니다. 경로 결과에는 이동 시간과 이동 거리가 포함됩니다.

사전 조건

- 먼저 경로 계산기 리소스를 생성하고 Amazon Location에 대한 요청에 대한 인증을 설정해야 합니다. 자세한 설명은 사전 조건 섹션을 참조하세요.

`CalculateRouteMatrix` 작업을 사용하여 요청을 제출합니다. 최소 요청에는 다음과 같은 필수 필드가 포함됩니다.

- `DeparturePositions` – 경로를 계산할 시작 위치 세트. `[longitude, latitude]`의 배열로 정의됩니다.
- `DestinationPositions` – 경로를 계산할 최종 위치 세트. `[longitude, latitude]`의 배열로 정의됩니다.

> **Note**
>
> 도로에 있지 않은 출발 위치 또는 도착 위치를 지정하는 경우 Amazon Location은 해당 위치를 가장 가까운 도로로 이동합니다.

요청에 출발 시간과 이동 수단을 선택적으로 지정할 수 있습니다.

AWS CLI 또는 Amazon Location API를 사용할 수 있습니다.

API

다음 예제는 경로 계산기 리소스를 사용한 CalculateRouteMatrix *ExampleCalculator*요청입니다. 요청은 출발 위치 [*-122.7565*, *49.0021*] 및 [*-122.2014*, *47.6101*]에서 도착 위치 [*-122.3394*, *47.6159*] 및 [*-122.4813*, *48.7511*]까지의 경로 매트릭스를 계산하도록 지정합니다.

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route-matrix
Content-type: application/json
{
   "DeparturePositions": [
       [-122.7565,49.0021],
       [-122.2014,47.6101]
   ],
   "DestinationPositions": [
       [-122.3394, 47.6159],
       [-122.4813,48.7511]
   ]
}
```

AWS CLI

다음 예제는 경로 계산기 리소스를 사용하는 calculate-route-matrix *ExampleCalculator*명령입니다. 요청은 출발 위치 [*-122.7565*, *49.0021*] 및 [*-122.2014*, *47.6101*]에서 도착 위치 [*-122.3394*, *47.6159*] 및 [*-122.4813*, *48.7511*]까지의 경로 매트릭스를 계산하도록 지정합니다.

```
aws location \
    calculate-route-matrix \
        --calculator-name ExampleCalculator \
        --departure-positions "[[-122.7565,49.0021],[-122.2014,47.6101]]" \
```

```
    --destination-positions "[[-122.3394,47.6159],[-122.4813,48.7511]]"
```

기본적으로 응답은 킬로미터 단위로 `Distance`을 반환합니다. 다음 선택적 파라미터를 사용하여 측정 단위를 마일로 변경할 수 있습니다.

- `DistanceUnit` – 거리 결과에 사용할 단위 체계를 지정합니다.

Example

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route-matrix
Content-type: application/json
{
   "DeparturePositions": [
       [-122.7565,49.0021],
       [-122.2014,47.6101]
   ],
   "DestinationPositions": [
       [-122.3394, 47.6159],
       [-122.4813,48.7511]
   ],
   "DistanceUnit": "Miles"
}
```

## 출발 위치 및 도착 위치에 대한 제한

경로 매트릭스를 계산할 때 출발 위치 및 도착 위치에 제한이 있습니다. 이러한 제한은 `RouteCalculator` 리소스에서 사용하는 공급자에 따라 다릅니다.

| 제한 사항 | Esri | Grab | HERE |
|---|---|---|---|
| 위치 개수 | 최대 10개의 출발 위치와 10개의 도착 위치. | 최대 350개의 출발 위치와 350개의 도착 위치. | 최대 350개의 출발 위치와 350개의 도착 위치.<br><br>장거리 경로의 경우 추가 제한 사항이 적용됩니다. 섹션을 참조하세요. |

| 제한 사항 | Esri | Grab | HERE |
| --- | --- | --- | --- |
| 위치 간 거리 | 출발 및 도착 위치 쌍은 서로 400km 이내에 있어야 합니다(도보 경로의 경우 40km). | | 모든 출발 및 도착 위치는 직경 180km의 원 내에 있어야 합니다.<br><br>장거리 경로의 경우 추가 제한 사항이 적용됩니다. 섹션을 참조하세요. |
| 경로 길이 | 해당 경로의 총 이동 시간이 400분을 넘으면 경로가 완료되지 않습니다. | | 출발지와 도착지를 중심으로 한 원을 10km 이상 벗어나는 경로는 계산되지 않습니다.<br><br>장거리 경로의 경우 추가 제한 사항이 적용됩니다. 섹션을 참조하세요. |
| 리전 | 한국에서는 경로 매트릭스 계산이 지원되지 않습니다. | 동남아시아에서 사용 가능합니다. 지원되는 국가/리전 목록 및 자세한 내용은 대상 국가/리전 및 지역 항목을 참조하세요. | 추가 제한 사항은 없습니다. |

장거리 경로 계획

경로 결과 매트릭스를 계산하는 것은 효율적인 경로 계획에 유용하지만 계산에는 다소 시간이 걸릴 수 있습니다. 모든 Amazon Location Service 데이터 공급자는 계산할 수 있는 경로 수 또는 경로 거리를 제한합니다. 예를 들어, HERE로 350개의 출발 위치와 도착 위치 사이에 경로를 생성할 수 있지만, 이러한 위치는 180km의 원 안에 있어야 합니다. 장거리 경로를 계획하고 싶다면 어떻게 해야 할까요?

HERE를 데이터 공급자로 사용한 `RouteCalculator`를 사용하여 더 적은 수의 경로에 대해 길이 제한이 없는 경로 매트릭스를 계산할 수 있습니다. 이렇게 해도 `CalculateRouteMatrix` API 호출 방

식이 바뀌지는 않으며, Amazon Location에서는 요구 사항을 충족하면 단순히 더 긴 경로를 허용합니다.

더 긴 길이의 경로 계산에 대한 요구 사항은 다음과 같습니다.

- `RouteCalculator`는 HERE 데이터 공급자를 사용해야 합니다.
- 출발 위치의 수는 15개를 초과할 수 없습니다.
- 계산할 총 경로 수는 100개를 초과할 수 없습니다.
- 경로가 1,000km를 초과할 때 통행료를 피할 수 있는 트럭 경로의 경우 장거리 경로가 허용되지 않습니다. 이 조합은 계산 속도가 느리고 호출 시간이 초과될 수 있습니다. CalculateRoute작업을 통해 이러한 경로를 개별적으로 계산할 수 있습니다.

호출이 이러한 요구 사항을 충족하지 않는 경우(예: 단일 호출로 150개의 경로 계산을 요청하는 경우) `CalculateRouteMatrix`는 더 짧은 경로 규칙만 허용하도록 되돌아갑니다. 그런 다음 위치가 180km의 원 이내이면 경로를 계산할 수 있습니다.

장거리 경로를 계산할 때는 다음 사항을 염두에 두세요 .

- 경로가 길면 계산 시간이 더 오래 걸릴 수 있으며, Amazon Location API의 최대 시간보다 더 오래 걸릴 수 있습니다. 특정 경로에서 시간 초과가 자주 발생하는 경우 각 `CalculateRouteMatrix` 호출에서 더 적은 수의 경로를 시도할 수 있습니다.
- `CalculateRouteMatrix` 요청에 도착 위치 또는 출발 위치를 더 추가하면 작업이 더 제한된 모드로 전환될 수 있으며, 생성할 경로가 적을 때 문제 없이 계산할 수 있는 경로에 오류가 발생할 수 있습니다. 이 경우 도착 또는 출발 위치의 수를 줄이고 여러 번 요청하여 필요한 전체 경로 계산 세트를 얻으세요.

## 응답의 예

다음은 Amazon Location Routes API에서 `CalculateRouteMatrix` 작업을 호출할 때 해당 응답이 포함된 요청의 예시입니다.

Example request

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route-matrix
Content-type: application/json
{
   "DeparturePositions": [
       [-122.7565,49.0021],
```

```
        [-122.2014,47.6101]
    ],
    "DestinationPositions": [
        [-122.3394, 47.6159],
        [-122.4813,48.7511]
    ]
}
```

## Example response

```
{
    "RouteMatrix": [
        [
            {
                "Distance": 178.764,
                "DurationSeconds": 7565
            },
            {
                "Distance": 39.795,
                "DurationSeconds": 1955
            }
        ],
        [
            {
                "Distance": 15.31,
                "DurationSeconds": 1217
            },
            {
                "Distance": 142.506,
                "DurationSeconds": 6279
            }
        ]
    ],
    "Summary": {
        "DataSource": "Here",
        "RouteCount": 4,
        "ErrorCount": 0,
        "DistanceUnit": "Kilometers"
    }
}
```

# 도로에 위치하지 않은 위치

CalculateRoute 또는 CalculateRouteMatrix를 사용하는 경우, 도로에 있지 않은 출발 위치, 도착 위치 또는 중간 지점 위치를 지정하는 경우 Amazon Location은 위치를 인근 도로로 이동합니다.

다음 CalculateRoute 요청은 도로에 있지 않은 출발 위치 및 도착 위치를 지정합니다.

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-123.128014, 49.298472],
   "DestinationPosition": [-123.134701, 49. 294315]
}
```

결과 응답은 인근 도로로 이동된 위치를 반환합니다.

```
{
   "Legs": [
      {
         "StartPosition": [-123.12815, 49.29717],
         "EndPosition":   [-123.13375, 49.2926],
         "Distance": 4.223,
         "DurationSeconds": 697,
         "Steps": [
            {
               "StartPosition": [ -123.12815, 49.29717 ],
               "EndPosition":   [ -123.12806, 49.29707 ],
               "Distance": 0.013,
               "DurationSeconds": 8
            },
            {
               "StartPosition": [ -123.12806, 49.29707 ],
               "EndPosition":   [ -123.1288, 49.29659 ],
               "Distance": 0.082,
               "DurationSeconds": 36
            },
            {
               "StartPosition": [ -123.1288, 49.29659 ],
               "EndPosition":   [ -123.12021, 49.29853 ],
               "Distance": 0.742,
               "DurationSeconds": 128
            },
            {
```

```
            "StartPosition": [ -123.12021, 49.29853 ],
            "EndPosition":   [ -123.1201, 49.29959 ],
            "Distance": 0.131,
            "DurationSeconds": 26
         },
         {
            "StartPosition": [ -123.1201, 49.29959 ],
            "EndPosition":   [ -123.13562, 49.30681 ],
            "Distance": 1.47,
            "DurationSeconds": 238
         },
         {
            "StartPosition": [ -123.13562, 49.30681 ],
            "EndPosition":   [ -123.13693, 49.30615 ],
            "Distance": 0.121,
            "DurationSeconds": 28
         },
         {
            "StartPosition": [ -123.13693, 49.30615 ],
            "EndPosition":   [ -123.13598, 49.29755 ],
            "Distance": 0.97,
            "DurationSeconds": 156
         },
         {
            "StartPosition": [ -123.13598, 49.29755 ],
            "EndPosition":   [ -123.13688, 49.29717 ],
            "Distance": 0.085,
            "DurationSeconds": 15
         },
         {
            "StartPosition": [ -123.13688, 49.29717 ],
            "EndPosition":   [ -123.13375, 49.2926 ],
            "Distance": 0.609,
            "DurationSeconds": 62
         }
      ]
   }
],
"Summary": {
   "RouteBBox": [ -123.13693, 49.2926, -123.1201, 49.30681 ],
   "DataSource": "Here",
   "Distance": 4.223,
   "DurationSeconds": 697,
   "DistanceUnit": "Kilometers"
```

```
    }
}
```

## 출발 시간

기본적으로 CalculateRoute 또는 CalculateRouteMatrix를 호출할 때 요청에 출발 시간을 제공하지 않으면 계산된 경로에 최적의 교통 상황이 반영됩니다.

다음 옵션 중 하나를 사용하여 선택한 데이터 공급자의 실시간 및 예측 교통 상황을 사용하도록 특정 출발 시간을 설정할 수 있습니다.

- DepartNow – *true*로 설정하면 실시간 교통 상황을 사용하여 가장 빠른 이동 경로를 계산합니다.
- DepartureTime – 제공된 경우 요청된 시간에 대한 예측 및 알려진 교통 상황을 사용합니다. YYYY-MM-DDThh:mm:ss.sssZ 형식으로 정의됩니다.

### Example

다음 CalculateRoute 요청은 출발 시간을 2024년 7월 2일 12:15:20 UTC로 설정합니다.

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "WaypointPositions":[
        [-122.1884,48.0936],
        [-122.3493,47.6205]
    ]
    "IncludeLegGeometry": true,
    "DepartureTime": 2024-07-02T12:15:20.000Z,
}
```

## 이동 수단

CalculateRoute 또는 CalculateRouteMatrix를 사용할 때 이동 수단을 설정할 수 있습니다. 이동 수단은 이동 속도와 도로 호환성에 영향을 줍니다. 기본 이동 수단은 자동차이지만 다음 선택적 파라미터를 사용하여 경로를 따라 이동하는 동안 이용할 이동 수단을 지정할 수 있습니다.

- TravelMode – 경로 계산 시 교통 모드를 지정합니다. 예: *Bicycle*, *Car*, *Motorcycle*, *Truck* 또는 *Walking*.

제한 사항

- Walking을 이동 수단을 지정하고 데이터 공급자가 Esri인 경우 출발지와 도착지가 40km 이내여야 합니다.
- Bicycle 또는 Motorcycle은 Grab을 데이터 공급자로 사용하는 경우에만 사용할 수 있습니다.
- Grab은 특정 도시의 Bicycle 경로 및 Walking 경로만 제공합니다. 자세한 설명은 대상 국가/리전 및 지역 섹션을 참조하세요.
- Truck은 Grab을 데이터 공급자로 사용할 때는 사용할 수 없습니다.

추가 기본 설정

*Car*의 TravelMode를 지정하는 경우 다음 선택적 파라미터를 사용하여 추가 경로 기본 설정을 지정할 수 있습니다.

- CarModeOptions – 자동차로 이동할 때의 경로 기본 설정을 지정합니다(예: *AvoidFerries* 또는 *AvoidTolls*).

*Truck*의 TravelMode를 지정하는 경우 다음 선택적 파라미터를 사용하여 추가 경로 기본 설정을 지정할 수 있습니다.

- TruckModeOptions – *TruckDimensions* 및 *TruckWeight*을 수용할 수 있는 경로를 지정하는 것 외에도 트럭으로 이동할 때 경로 기본 설정(예: *AvoidFerries* 또는 *AvoidTolls*)을 지정합니다.

Example

다음 CalculateRoute 요청은 *Truck*을 이동 수단으로 지정합니다. 추가 경로 제한 사항에는 페리를 이용하는 경로를 피하고 트럭 크기와 중량을 수용할 수 없는 도로는 피하는 것이 포함됩니다.

```
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "DepartNow": true,
   "TravelMode": "Truck",
   "TruckModeOptions": {
      "AvoidFerries": true,
      "AvoidTolls": false,
      "Dimensions": {
```

```
            "Height": 4.5,
            "Length": 15.5,
            "Unit": "Meters",
            "Width": 4.5
        },
        "Weight": {
            "Total": 7500,
            "Unit": "Pounds"
        }
    }
}
```

## 경로 계산기 리소스 관리

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 경로 계산기 리소스를 관리할 수 있습니다.

### 경로 계산기 리소스 나열

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 경로 계산기 목록을 볼 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 경로 계산기 목록을 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 경로 계산기를 선택합니다.
3. 내 경로 계산기에서 경로 계산기 세부 정보를 확인하십시오.

API

Amazon Location Routes API에서 `ListRouteCalculators` 작업을 사용하십시오.

다음은 AWS 계정의 경로 계산기 목록을 가져오기 위한 API 요청입니다.

```
POST /routes/v0/list-calculators
```

다음은 `ListRouteCalculators`에 대한 응답의 예입니다:

```
{
       "Entries": [
          {
             "CalculatorName": "ExampleCalculator",
             "CreateTime": 2020-09-30T22:59:34.142Z,
             "DataSource": "Esri",
             "Description": "string",
             "UpdateTime": 2020-09-30T23:59:34.142Z
          }
       ],
       "NextToken": "1234-5678-9012"
    }
```

CLI

`list-route-calculators` 명령을 사용합니다.

다음 예는 AWS 계정의 경로 계산기 목록을 가져오기 위한 AWS CLI입니다.

```
aws location list-route-calculators
```

## 경로 계산기 세부 정보 가져오기

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정의 모든 경로 계산기 리소스에 대한 세부 정보를 얻을 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 경로 계산기의 세부 정보를 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 경로 계산기를 선택합니다.
3. 내 경로 계산기에서 대상 경로 계산기의 이름 링크를 선택합니다.

API

Amazon Location Routes API에서 `DescribeRouteCalculator` 작업을 사용하십시오.

다음 예는 경로 계산기 세부 정보를 가져오기 위한 API *ExampleCalculator*요청입니다.

```
GET /routes/v0/calculators/ExampleCalculator
```

다음은 DescribeRouteCalculator에 대한 응답의 예입니다:

```
{
       "CalculatorArn": "arn:aws:geo:us-west-2:123456789012:route-
calculator/ExampleCalculator",
       "CalculatorName": "ExampleCalculator",
       "CreateTime": 2020-09-30T22:59:34.142Z,
       "DataSource": "Esri",
       "Description": "string",
       "Tags": {
          "Tag1" : "Value1"
       },
       "UpdateTime": 2020-09-30T23:59:34.142Z
    }
```

CLI

`describe-route-calculator` 명령을 사용합니다.

다음 예는 경로 계산기 세부 정보를 AWS CLI 가져오는 예입니다 *ExampleCalculator*.

```
aws location describe-route-calculator \
        --calculator-name "ExampleCalculator"
```

## 경로 계산기 삭제

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정에서 경로 계산기를 삭제할 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 경로 계산기를 삭제하려면

> **Warning**
> 이 작업은 리소스를 영구적으로 삭제합니다.

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 경로 계산기를 선택합니다.
3. 내 경로 계산기에서 대상 경로 계산기를 선택합니다.
4. 경로 계산기 삭제를 선택합니다.

API

Amazon Location Routes API에서 `DeleteRouteCalculator` 작업을 사용하십시오.

다음은 지오펜스 컬렉션을 삭제하기 위한 API 요청입니다. *`ExampleCalculator`*

```
DELETE /routes/v0/calculators/ExampleCalculator
```

다음은 `DeleteRouteCalculator`에 대한 응답의 예입니다:

```
HTTP/1.1 200
```

CLI

`delete-route-calculator` 명령을 사용합니다.

다음 예시는 지오펜스 AWS CLI 컬렉션을 삭제하는 명령입니다. *`ExampleCalculator`*

```
aws location delete-route-calculator \
        --calculator-name "ExampleCalculator"
```

# Amazon Location을 사용하여 관심 영역을 지오펜싱하기

지오펜싱 애플리케이션은 이전에 등록된 관심 영역을 기준으로 추적 대상 디바이스의 위치를 평가합니다. 이를 통해 위치 업데이트를 기반으로 조치를 취할 수 있습니다. 예를 들어, 모바일 앱으로 커피를 주문한 고객이 매장 근처에 있을 때 알림을 보내는 이벤트를 시작할 수 있습니다.

> **Note**
>
> 지오펜싱 및 트래커 개념에 대한 개요는 지오펜스 및 트래커 항목을 참조하세요.

이 가이드 섹션에서는 Amazon Location Service를 사용하여 지오펜싱 애플리케이션을 생성하는 step-by-step 방법에 대한 지침을 제공합니다.

단계 개요

1. 관심 영역 주변에 지오펜스를 추가하고 지오펜스 컬렉션 리소스에 저장합니다.
2. 대상 디바이스 추적을 시작하고 디바이스 위치 기록을 트래커 리소스에 저장합니다.
3. 트래커 리소스를 지오펜스 컬렉션 리소스에 연결하여 디바이스 위치 업데이트가 모든 지오펜스에 대해 자동으로 평가되도록 합니다.
4. Amazon Location Tracker를 사용하여 디바이스의 위치 기록을 보관하고 싶지 않다면 지오펜스 컬렉션 리소스를 기준으로 디바이스 위치를 직접 평가할 수 있습니다.

지오펜싱 솔루션을 구현한 후에는 지오펜스 컬렉션 리소스에서 다음과 같은 이벤트가 발생합니다.

- `ENTER` — 추적된 디바이스가 지오펜스 컬렉션 내의 지오펜스에 진입합니다.
- `EXIT` — 추적된 디바이스는 지오펜스 컬렉션 내의 지오펜스를 나갑니다.

EventBridge Amazon을 사용하여 이벤트를 다른 곳으로 라우팅하여 이벤트에 대응할 수 있습니다.

각 APIs 디바이스에서 Amazon Location Service를 통해 업데이트를 전송하는 대신 를 사용하여 디바이스 업데이트를 MQTT 전송할 수 있습니다.

다음 주제에서는 이러한 단계와 대안을 자세히 설명합니다.

주제

- 지오펜싱 및 트래킹 모바일 애플리케이션 샘플

## 지오펜스 추가

지오펜스는 관심 영역을 정의하는 닫힌 경계를 형성하는 지점과 꼭지점을 포함합니다. 지오펜스 컬렉션은 하나 이상의 지오펜스를 저장하고 관리합니다.

Amazon Location 지오펜스 컬렉션은 Geo (7946) 라는 표준 지리공간 데이터 형식을 사용하여 정의된 지오펜스를 저장합니다. JSON RFC geojson.io와 같은 도구를 무료로 사용하여 지오펜스를 그래픽으로 그리고 출력 Geo 파일을 저장할 수 있습니다. JSON

> **Note**
>
> Amazon Location은 구멍이 있는 다각형, 여러 개의 다각형, 시계 방향 다각형, 반대 자오선을 가로지르는 지오펜스를 지원하지 않습니다.

### 지오펜스 컬렉션 생성

Amazon Location 콘솔, AWS CLI또는 Amazon Location을 사용하여 지오펜스를 저장하고 관리할 지오펜스 컬렉션을 생성합니다. APIs

Console

Amazon Location 콘솔을 사용하여 지오펜스 컬렉션을 만들려면

1. 에서 Amazon Location Service 콘솔을 엽니다 https://console.aws.amazon.com/location/.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 지오펜스 컬렉션 생성을 선택합니다.
4. 다음 입력란을 작성합니다.
   - 이름 – 고유한 이름을 입력합니다. 예: *ExampleGeofenceCollection*. 최대 100자까지 입력할 수 있습니다. 유효한 항목에는 영숫자 문자, 하이픈, 마침표 및 밑줄이 포함됩니다.
   - 설명 – 리소스를 구분할 수 있도록 선택적 설명을 입력합니다.
5. 대상이 되는 EventBridge 규칙에서는 선택적 EventBridge 규칙을 생성하여 CloudWatch 지오펜스 이벤트에 대한 대응을 시작할 수 있습니다. 이를 통해 Amazon Location은 Amazon CloudWatch Logs에 이벤트를 게시할 수 있습니다.

6. (선택 사항) 태그 아래에 태그 키 및 값을 입력합니다. 이렇게 하면 새 지오펜스 컬렉션에 태그가 추가됩니다. 자세한 내용은 Amazon Location Service 리소스 태그 지정 단원을 참조하십시오.
7. (선택 사항) 고객 관리형 키 암호화에서 고객 관리형 키 추가를 선택할 수 있습니다. 이렇게 하면 기본 AWS 소유 암호화를 통해 생성, 소유 및 관리하는 대칭적인 고객 관리 키가 추가됩니다. 자세한 내용은 저장 데이터 암호화를 참조하세요.
8. 지오펜스 컬렉션 생성을 선택합니다.

API

Amazon 로케이션을 사용하여 지오펜스 컬렉션을 만들려면 APIs

Amazon 로케이션 APIs 지오펜스에서 `CreateGeofenceCollection` 오퍼레이션을 사용하십시오.

다음 예시에서는 API 요청을 사용하여 라는 지오펜스 컬렉션을 생성합니다.*ExampleGeofenceCollection*. 지오펜스 컬렉션은 고객 데이터를 암호화하는 고객 관리 AWS KMS 키와 연결됩니다.

```
POST /geofencing/v0/collections
Content-type: application/json

{
   "CollectionName": "ExampleGeofenceCollection",
   "Description": "Geofence collection 1 for shopping center",
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

AWS CLI 명령을 사용하여 지오펜스 컬렉션을 만들려면

`create-geofence-collection` 명령을 사용합니다.

다음 예에서는 a를 AWS CLI 사용하여 라는 지오펜스 컬렉션을 만듭니다.*ExampleGeofenceCollection*. 지오펜스 컬렉션은 고객 데이터를 암호화하는 고객 관리 AWS KMS 키와 연결됩니다.

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --kms-key-id "1234abcd-12ab-34cd-56ef-1234567890ab" \
  --tags Tag1=Value1
```

> **Note**
>
> 청구는 사용량에 따라 달라집니다. 다른 AWS 서비스 사용 시 요금이 부과될 수 있습니다. 자세한 정보는 Amazon Location Service 가격을 참조하세요.

## 지오펜스 그리기

이제 지오펜스 컬렉션을 만들었으니 지오펜스를 정의할 수 있습니다. 지오펜스는 다각형 또는 원으로 정의됩니다. 다각형 지오펜스를 그리려면 geojson.io와 같은 JSON 지리 편집 도구를 사용할 수 있습니다.

지오펜스를 원으로 만들려면 원의 중심점과 반경을 정의해야 합니다. 예를 들어, 디바이스가 특정 위치로부터 50미터 이내에 올 때마다 알림을 받도록 지오펜스를 만들려면 해당 위치의 위도와 경도를 사용하고 반경을 50미터로 지정합니다.

Amazon Location APIs Service를 사용하면 키-값 쌍의 형태로 지오펜스에 메타데이터를 추가할 수도 있습니다. 이는 유형과 같은 지오펜스에 대한 정보 또는 애플리케이션에 대한 기타 정보를 저장하는 데 유용할 수 있습니다. 아마존을 통한 아마존 로케이션 서비스 이벤트에 대응하기 EventBridge 시 이 메타데이터를 사용할 수 있습니다.

## 다각형 지오펜스 추가

이 섹션에서는 다각형 지오펜스 생성 방법에 대해 설명합니다.

Geo 도구를 사용하여 지오펜스를 그리세요. JSON

이제 지오펜스 컬렉션을 만들었으니 geojson.io와 같은 지리 편집 도구를 사용하여 지오펜스를 정의할 수 있습니다. JSON

JSON지오 파일을 만들려면

1. 지리 JSON 편집 도구를 엽니다. 예: geojson.io.
2. 다각형 그리기 아이콘을 선택하고 관심 영역을 그립니다.
3. 저장을 선택한 다음 드롭다운 메뉴에서 JSONGeo를 선택합니다.

### 지오펜스 컬렉션에 JSON 지오펜스 추가

생성된 지리 JSON 파일을 사용하여 Amazon Location Service 콘솔, 또는 Amazon 로케이션을 사용하여 지오펜스를 업로드할 AWS CLI수 있습니다. APIs

#### Console

Amazon Location Service 콘솔을 사용하여 지오펜스 컬렉션에 지오펜스를 추가하려면

1. 에서 Amazon Location Service 콘솔을 엽니다 https://console.aws.amazon.com/location/.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 지오펜스 컬렉션 목록에서 대상 지오펜스 컬렉션의 이름 링크를 선택합니다.
4. 지오펜스에서 지오펜스 생성을 선택합니다.
5. 지오펜스 추가 창에서 지리를 JSON 창으로 끌어다 놓습니다.
6. 지오펜스 추가를 선택합니다.

#### API

Amazon 위치를 사용하여 지오펜스를 추가하려면 APIs

Amazon 로케이션 APIs 지오펜스에서 `PutGeofence` 오퍼레이션을 사용하십시오.

다음 예시에서는 API 요청을 사용하여 ID가 지정된 지오펜스를 추가합니다.*GEOFENCE-EXAMPLE1* 라는 지오펜스 컬렉션에 *ExampleGeofenceCollection*. 또한 Type 키와 값이 있는 단일 지오펜스 메타데이터 속성을 지정합니다. `loadingArea`

```
PUT /geofencing/v0/collections/ExampleGeofenceCollection/geofence/GEOFENCE-EXAMPLE1
    Content-type: application/json

    {
       "GeofenceProperties": {
          "Type" : "loadingArea"
       },
```

```
       "Geometry": {
          "Polygon": [
            [
                [-5.716667, -15.933333],
                [-14.416667, -7.933333],
                [-12.316667, -37.066667],
                [-5.716667, -15.933333]
            ]
          ]
       }
    }
```

또는 BatchPutGeofence 작업을 사용하여 둘 이상의 지오펜스를 추가할 수 있습니다.

```
POST /geofencing/v0/collections/ExampleGeofenceCollection/put-geofences
    Content-type: application/json

    {
       "Entries": [
          {
             "GeofenceProperties": {
                "Type" : "loadingArea"
             },
             "GeofenceId": "GEOFENCE-EXAMPLE1",
             "Geometry": {
                "Polygon": [
                   [
                      [-5.716667, -15.933333],
                      [-14.416667, -7.933333],
                      [-12.316667, -37.066667],
                      [-5.716667, -15.933333]
                   ]
                ]
             }
          }
       ]
    }
```

AWS CLI

명령을 사용하여 지오펜스 컬렉션에 지오펜스를 추가하려면 AWS CLI

put-geofence 명령을 사용합니다.

다음 예제에서는 AWS CLI a를 사용하여 라는 지오펜스 컬렉션에 지오펜스를 추가합니다.*ExampleGeofenceCollection*.

```
$ aws location \
        put-geofence \
            --collection-name ExampleGeofenceCollection \
            --geofence-id ExampleGeofenceTriangle \
            --geofence-properties '{"Type": "loadingArea"}' \
            --geometry 'Polygon=[[[-5.716667, -15.933333],[-14.416667, -7.933333],
[-12.316667, -37.066667],[-5.716667, -15.933333]]]'
    {
        "CreateTime": "2020-11-11T00:16:14.487000+00:00",
        "GeofenceId": "ExampleGeofenceTriangle",
        "UpdateTime": "2020-11-11T00:19:59.894000+00:00"
    }
```

## 원형 지오펜스 추가

이 섹션에서는 원형 지오펜스를 만드는 방법에 대해 설명합니다. 원의 중심으로 삼고자 하는 지점의 위도와 경도, 원의 반경(미터)을 알아야 합니다. Amazon Location APIs 또는 를 사용하여 원형 지오펜스를 생성할 수 있습니다. AWS CLI

API

Amazon 로케이션을 사용하여 원형 지오펜스를 추가하려면 APIs

Amazon 로케이션 APIs 지오펜스에서 PutGeofence 오퍼레이션을 사용하십시오.

다음 예시에서는 API 요청을 사용하여 ID가 지정된 지오펜스를 추가합니다.*GEOFENCE-EXAMPLE2* 라는 지오펜스 컬렉션에 *ExampleGeofenceCollection*:

```
PUT /geofencing/v0/collections/ExampleGeofenceCollection/geofence/GEOFENCE-EXAMPLE2
    Content-type: application/json

    {
       "Geometry": {
          "Circle": {
            "Center": [-5.716667, -15.933333],
            "Radius": 50
          }
       }
```

```
    }
```

AWS CLI

명령을 사용하여 지오펜스 컬렉션에 원형 지오펜스 추가하기 AWS CLI

put-geofence 명령을 사용합니다.

다음 예제에서는 AWS CLI a를 사용하여 라는 지오펜스 컬렉션에 지오펜스를 추가합니다.*ExampleGeofenceCollection*.

```
$ aws location \
        put-geofence \
            --collection-name ExampleGeofenceCollection \
            --geofence-id ExampleGeofenceCircle \
            --geometry 'Circle={Center=[-5.716667, -15.933333], Radius=50}'
```

> **Note**
>
> 다음 예제와 같이 복잡한 지오메트리를 자체 파일에 넣을 JSON 수도 있습니다.
>
> ```
> $ aws location \
>         put-geofence \
>             --collection-name ExampleGeofenceCollection \
>             --geofence-id ExampleGeofenceCircle \
>             --geometry file:circle.json
> ```
>
> 이 예제의 circle.json 파일에는 원 지오메트리가 포함되어 JSON 있습니다.
>
> ```
> {
>     "Circle": {
>         "Center": [-74.006975, 40.717127],
>         "Radius": 287.7897969218057
>     }
> }
> ```

## 추적 시작

이 섹션에서는 디바이스 위치를 포착하는 추적 애플리케이션을 빌드하는 방법을 안내합니다.

## 트래커 생성

디바이스의 위치 업데이트를 저장하고 처리하는 트래커 리소스를 생성합니다. Amazon 로케이션 서비스 콘솔 AWS CLI, 또는 아마존 로케이션을 사용할 수 APIs 있습니다.

트래커 리소스에 저장된 각 위치 업데이트에는 위치 정확도 측정값과 저장하려는 위치 또는 디바이스에 대한 메타데이터 필드를 최대 3개까지 포함할 수 있습니다. 메타데이터는 키-값 쌍으로 저장되며 속도, 방향, 타이어 공기압 또는 엔진 온도와 같은 정보를 저장할 수 있습니다.

트래커는 위치 업데이트가를 수신하면 이를 필터링합니다. 이렇게 하면 디바이스 경로의 시각적 노이즈(지터라고 함)가 줄어들고 잘못된 지오펜스 출입 이벤트 수가 줄어듭니다. 또한 지오펜스 평가 시작 횟수를 줄여 비용을 관리하는 데 도움이 됩니다.

트래커는 비용을 관리하고 위치 업데이트의 지터를 줄이는 데 도움이 되는 세 가지 위치 필터링 옵션을 제공합니다.

- 정확도 기반 – 정확도 측정을 제공하는 모든 디바이스와 함께 사용할 수 있습니다. 대부분의 모바일 디바이스는 이 정보를 제공합니다. 각 위치 측정의 정확도는 GPS 위성 수신, 풍경, Wi-Fi 및 Bluetooth 장치의 근접성 등 여러 환경 요인의 영향을 받습니다. 대다수 모바일 디바이스를 포함한 디바이스는 대부분 측정과 함께 측정의 정확도 추정치를 제공할 수 있습니다. `AccuracyBased` 필터링을 사용하면 Amazon Location는 디바이스가 측정된 정확도보다 적게 움직인 경우 위치 업데이트를 무시합니다. 예를 들어 디바이스에서 두 번 연속 업데이트의 정확도 범위가 5m와 10m인 경우, 디바이스가 15m 미만으로 이동하면 Amazon Location은 두 번째 업데이트를 무시합니다. Amazon Location은 무시한 업데이트를 지오펜스와 비교하여 평가하거나 저장하지 않습니다.

  정확도가 제공되지 않으면 0으로 처리되며, 측정값은 완전히 정확한 것으로 간주됩니다.

  > **Note**
  >
  > 정확도 기반 필터링을 사용하여 모든 필터링을 제거할 수도 있습니다. 정확도 기반 필터링을 선택했지만 모든 정확도 데이터를 0으로 재정의하거나 정확도를 완전히 생략하는 경우 Amazon Location은 업데이트를 필터링하지 않습니다.

- 거리 기반 – 디바이스가 정확도 측정을 제공하지 않지만 필터링을 활용하여 지터를 줄이고 비용을 관리하려는 경우에 사용합니다. `DistanceBased` 필터링은 디바이스가 30m (98.4피트) 미만으로 이동한 위치 업데이트를 무시합니다. `DistanceBased` 위치 필터링을 사용하는 경우 Amazon Location은 지오펜스에 대해 무시된 업데이트를 평가하거나 업데이트를 저장하지 않습니다.

  iOS 및 Android 디바이스의 평균 정확도를 포함하여 대부분의 모바일 디바이스의 정확도는 15m 이내입니다. 대부분의 애플리케이션에서 `DistanceBased` 필터링은 디바이스 궤적을 맵에 표시할 때

위치 부정확성의 영향을 줄이고, 디바이스가 지오펜스 경계 근처에 있을 때 여러 번의 연속적인 출입 이벤트로 인한 바운싱 효과를 줄일 수 있습니다. 또한 링크된 지오펜스를 기준으로 평가하거나 디바이스 위치를 검색하기 위한 호출을 줄여 애플리케이션 비용을 절감할 수 있습니다.

- 시간 기반 - (기본값) 디바이스가 위치 업데이트를 매우 자주(30초에 한 번 이상) 보내고 모든 업데이트를 저장하지 않고도 거의 실시간으로 지오펜스를 평가하려는 경우에 사용합니다. `TimeBased` 필터링에서는 모든 위치 업데이트가 연결된 지오펜스 컬렉션에 대해 평가되지만 모든 위치 업데이트가 저장되는 것은 아닙니다. 업데이트 빈도가 30초 이상인 경우 각 고유 디바이스 ID에 대해 30초당 하나의 업데이트만 저장됩니다.

> **Note**
>
> 필터링 방법과 위치 업데이트 빈도를 결정할 때는 추적 애플리케이션 비용을 염두에 두세요. 모든 위치 업데이트에 대해 요금이 청구되며 연결된 각 지오펜스 컬렉션에 대한 위치 업데이트 평가 비용은 한 번 청구됩니다. 예를 들어, 시간 기반 필터링을 사용할 때 트래커가 두 개의 지오펜스 컬렉션에 연결된 경우 모든 위치 업데이트는 위치 업데이트 요청 1회와 지오펜스 컬렉션 평가 2회로 계산됩니다. 디바이스의 위치 업데이트를 5초마다 보고하고 시간 기반 필터링을 사용하는 경우 각 디바이스에 대해 시간당 720건의 위치 업데이트와 1,440회의 지오펜스 평가에 대한 요금이 청구됩니다.
>
> 청구서는 각 컬렉션의 지오펜스 수에 영향을 받지 않습니다. 각 지오펜스 컬렉션에는 최대 50,000개의 지오펜스가 포함될 수 있으므로 가능한 경우 지오펜스를 더 적은 컬렉션으로 결합하여 지오펜스 평가 비용을 절감할 수 있습니다.

기본적으로 추적되는 장치가 연결된 지오펜스에 들어오거나 나갈 때마다 EventBridge 이벤트가 발생합니다. 자세한 내용은 트래커를 지오펜스 컬렉션에 연결 단원을 참조하십시오.

트래커 리소스에 대해 필터링된 모든 위치 업데이트에 대해 이벤트를 활성화할 수 있습니다. 자세한 내용은 트래커의 업데이트 이벤트를 활성화합니다. 단원을 참조하십시오.

> **Note**
>
> 자체 AWS KMS 고객 관리 키를 사용하여 데이터를 암호화하려는 경우 Bounding Polygon Query 기능은 기본적으로 비활성화됩니다. 이는 이 바운딩 폴리곤 쿼리 기능을 사용하면 기기 위치 표현이 관리 키를 사용하여 암호화되지 않기 때문입니다. AWS KMS 하지만 정확한 디바이스 위치는 여전히 관리형 키를 사용하여 암호화됩니다.

트래커를 만들거나 업데이트할 때 `KmsKeyEnableGeospatialQueries` 파라미터를 true로 설정하여 경계 다각형 쿼리 기능을 옵트인하도록 선택할 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 트래커를 만들려면

1. 에서 Amazon Location Service 콘솔을 엽니다 https://console.aws.amazon.com/location/.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 트래커 생성을 선택합니다.
4. 다음 필드를 입력합니다.
   - 이름 – 고유한 이름을 입력합니다. 예: *ExampleTracker*. 최대 100자까지 입력할 수 있습니다. 유효한 항목에는 영숫자 문자, 하이픈, 마침표 및 밑줄이 포함됩니다.
   - 설명 – 선택적 설명을 입력합니다.
5. 위치 필터링에서 트래커 리소스를 사용하려는 방식에 가장 적합한 옵션을 선택합니다. 위치 필터링을 설정하지 않은 경우 기본 설정은 `TimeBased`입니다. 자세한 내용은 이 안내서와 `PositionFiltering` Amazon Location Service 트래커 API 레퍼런스를 참조하십시오트래커.
6. (선택 사항) 태그 아래에 태그 키 및 값을 입력합니다. 이렇게 하면 새 지오펜스 컬렉션에 태그가 추가됩니다. 자세한 내용을 알아보려면 리소스 태그 지정을 참조하세요.
7. (선택 사항) 고객 관리형 키 암호화에서 고객 관리형 키 추가를 선택할 수 있습니다. 이렇게 하면 기본 AWS 소유 암호화를 통해 생성, 소유 및 관리하는 대칭형 고객 관리 키가 추가됩니다. 자세한 내용은 저장 데이터 암호화를 참조하세요.
8. (선택 사항) 에서 KmsKeyEnableGeospatialQueries지리공간 쿼리를 활성화하도록 선택할 수 있습니다. 이렇게 하면 고객 관리 키를 사용하여 데이터를 암호화하면서 경계 다각형 쿼리 기능을 사용할 수 있습니다. AWS KMS

> **Note**
>
> 경계 다각형 쿼리 기능을 사용하는 경우 디바이스 위치 표현은 AWS KMS 관리형 키를 사용하여 암호화되지 않습니다. 하지만 정확한 디바이스 위치는 여전히 관리형 키를 사용하여 암호화됩니다.

9. (선택 사항) EventBridge 구성에서 필터링된 위치 업데이트에 대한 EventBridge 이벤트를 활성화하도록 선택할 수 있습니다. 그러면 이 트래커에 있는 디바이스의 위치 업데이트가 위치 필터링 평가를 충족할 때마다 이벤트가 전송됩니다.
10. 트래커 생성을 선택합니다.

API

Amazon 위치를 사용하여 트래커를 만들려면 APIs

Amazon 위치 APIs 추적기에서 `CreateTracker` 작업을 사용하십시오.

다음 예시에서는 API 요청을 사용하여 라는 추적기를 생성합니다.*`ExampleTracker`*. 트래커 리소스는 고객 데이터를 암호화하기 위한 고객 관리 AWS KMS 키와 연결되어 있으며, 포지션 업데이트는 지원하지 않습니다. EventBridge

```
POST /tracking/v0/trackers
Content-type: application/json

{

   "TrackerName": "ExampleTracker",
   "Description": "string",
   "KmsKeyEnableGeospatialQueries": false,
   "EventBridgeEnabled": false,
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "PositionFiltering": "AccuracyBased",
   "Tags": {
      "string" : "string"
   }
}
```

**`KmsKeyEnableGeospatialQueries`**가 활성화된 상태로 트래커 생성

다음 예시에서는 파라미터 `KmsKeyEnableGeospatialQueries`가 true로 설정되어 있습니다. 이렇게 하면 바운딩 폴리곤 쿼리 기능을 사용하는 동시에 고객 관리 키를 사용하여 데이터를 암호화할 수 있습니다. AWS KMS

경계 다각형 쿼리 기능 사용에 대한 자세한 내용은 ??? 항목을 참조하세요.

> **Note**
>
> 경계 다각형 쿼리 기능을 사용하는 경우 장치 위치 표현은 관리 키를 사용하여 암호화되지 않습니다. AWS KMS 하지만 정확한 디바이스 위치는 여전히 관리형 키를 사용하여 암호화됩니다.

```
POST /tracking/v0/trackers
Content-type: application/json

{

   "TrackerName": "ExampleTracker",
   "Description": "string",
   "KmsKeyEnableGeospatialQueries": true,
   "EventBridgeEnabled": false,
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "PositionFiltering": "AccuracyBased",
   "Tags": {
      "string" : "string"
   }
}
```

AWS CLI

AWS CLI 명령을 사용하여 트래커를 만들려면

`create-tracker` 명령을 사용합니다.

다음 예시에서는 를 AWS CLI 사용하여 라는 트래커를 생성합니다.*ExampleTracker*. 트래커 리소스는 고객 데이터를 암호화하기 위한 고객 관리 AWS KMS 키와 연결되어 있으며, 포지션 업데이트는 지원하지 않습니다. EventBridge

```
aws location \
  create-tracker \
  --tracker-name "ExampleTracker" \
  --position-filtering "AccuracyBased" \
  --event-bridge-enabled false \
  --kms-key-enable-geospatial-queries false \
  --kms-key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
```

**`KmsKeyEnableGeospatialQueries`**가 활성화된 상태로 트래커 생성

다음 예시에서는 파라미터 `KmsKeyEnableGeospatialQueries`가 true로 설정되어 있습니다. 이렇게 하면 바운딩 폴리곤 쿼리 기능을 사용하는 동시에 고객 관리 키를 사용하여 데이터를 암호화할 수 있습니다. AWS KMS

경계 다각형 쿼리 기능 사용에 대한 자세한 내용은 ??? 항목을 참조하세요.

> **Note**
>
> 경계 다각형 쿼리 기능을 사용하는 경우 장치 위치 표현은 관리 키를 사용하여 암호화되지 않습니다. AWS KMS 하지만 정확한 디바이스 위치는 여전히 관리형 키를 사용하여 암호화됩니다.

```
aws location \
  create-tracker \
  --tracker-name "ExampleTracker" \
  --position-filtering "AccuracyBased" \
  --event-bridge-enabled false \
  --kms-key-enable-geospatial-queries true \
  --kms-key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
```

> **Note**
>
> 청구는 사용량에 따라 달라집니다. 다른 AWS 서비스 사용 시 요금이 부과될 수 있습니다. 자세한 정보는 Amazon Location Service 가격을 참조하세요.

트래커 편집을 선택하여 트래커가 생성된 후 설명, 위치 필터링 및 EventBridge 구성을 편집할 수 있습니다.

## 요청 인증

트래커 리소스를 생성하고 지오펜스를 기준으로 디바이스 위치를 평가할 준비가 되었으면 요청을 인증할 방법을 선택합니다.

- 서비스에 액세스하는 방법을 알아보려면 Amazon Location Service 액세스를 참조하세요.

- 인증되지 않은 요청이 있는 디바이스 위치를 게시하려면 Amazon Cognito를 사용하는 것이 좋습니다.

  예

  다음 예시에서는 권한 부여를 위한 Amazon Cognito 자격 증명 풀 사용, AWS JavaScript SDKv3 사용, Amazon 로케이션을 보여줍니다. JavaScript 인증 도우미

```
import { LocationClient, BatchUpdateDevicePositionCommand } from "@aws-sdk/client-
location";
import { withIdentityPoolId } from "@aws/amazon-location-utilities-auth-helper";

// Unauthenticated identity pool you created
const identityPoolId = "us-east-1:1234abcd-5678-9012-abcd-sample-id";

// Create an authentication helper instance using credentials from Cognito
const authHelper = await withIdentityPoolId(identityPoolId);

const client = new LocationClient({
  region: "us-east-1", // The region containing both the identity pool and tracker
 resource
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  TrackerName: "ExampleTracker",
  Updates: [
    {
      DeviceId: "ExampleDevice-1",
      Position: [-123.4567, 45.6789],
      SampleTime: new Date("2020-10-02T19:09:07.327Z"),
    },
    {
      DeviceId: "ExampleDevice-2",
      Position: [-123.123, 45.123],
      SampleTime: new Date("2020-10-02T19:10:32Z"),
    },
  ],
};

const command = new BatchUpdateDevicePositionCommand(input);
```

```
// Send device position updates
const response = await client.send(command);
```

## 디바이스 위치로 트래커를 업데이트합니다.

디바이스를 추적하기 위해 디바이스 위치 업데이트를 트래커에 게시할 수 있습니다. 나중에 트래커 리소스에서 이러한 디바이스 위치 또는 디바이스 위치 기록을 검색할 수 있습니다.

각 위치 업데이트에는 디바이스 ID, 타임스탬프, 위치가 포함되어야 합니다. 필요에 따라 정확도 및 최대 3개의 키-값 쌍을 비롯한 기타 메타데이터를 포함시킬 수 있습니다.

트래커가 하나 이상의 지오펜스 컬렉션에 연결된 경우, 트래커에 지정한 필터링 규칙에 따라 해당 지오펜스를 기준으로 업데이트가 평가됩니다. 디바이스가 영역 내부에서 외부로 또는 그 반대로 이동하여 지오펜스 영역을 침해하는 경우, 내부로 이벤트가 수신됩니다. EventBridge 이러한 `ENTER` 이벤트 또는 `EXIT` 이벤트에는 디바이스 ID, 타임스탬프, 관련 메타데이터를 비롯한 위치 업데이트 세부 정보가 포함됩니다.

> **Note**
>
> 위치 필터링에 대한 자세한 내용은 트래커 생성 섹션을 참조하세요.
> 지오펜스 이벤트에 대한 자세한 내용은 아마존을 통한 아마존 로케이션 서비스 이벤트에 대응하기 EventBridge 단원을 참조하세요.

다음 방법 중 하나를 사용하여 디바이스 업데이트를 전송합니다.

- AWS IoT Core 리소스에 MQTT업데이트를 전송하고 이를 트래커 리소스에 연결합니다.
- Amazon 위치 추적기를 사용하거나 API AWS CLI, 또는 Amazon 위치를 사용하여 위치 APIs 업데이트를 전송하십시오. 를 사용하여 iOS 또는 Android APIs 애플리케이션에서 를 호출할 수 있습니다. AWS SDKs

API

Amazon 위치를 사용하여 포지션 업데이트를 보내려면 APIs

Amazon 위치 APIs 추적기에서 `BatchUpdateDevicePosition` 작업을 사용하십시오.

다음 예시에서는 API 요청을 사용하여 기기 위치 업데이트를 게시합니다.*`ExampleDevice`* 트래커에 *`ExampleTracker`*.

```
POST /tracking/v0/trackers/ExampleTracker/positions
Content-type: application/json
{
 "Updates": [
    {
    "DeviceId": "1",
    "Position": [
   -123.12245146162303, 49.27521118043802
    ],
    "SampleTime": "2022-10-24T19:09:07.327Z",
     "PositionProperties": {
            "name" : "device1"
        },
        "Accuracy": {
           "Horizontal": 10
        }
    },

    {
    "DeviceId": "2",
    "Position": [
   -123.1230104928471, 49.27752402723152
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
    {
    "DeviceId": "3",
    "Position": [
    -123.12325592118916, 49.27340530543111
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
    {
    "DeviceId": "4",
    "Position": [
    -123.11958813096311, 49.27774641063121
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
    {
    "DeviceId": "5",
    "Position": [
    -123.1277418058896, 49.2765989015285
```

```
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
    {
    "DeviceId": "6",
    "Position": [
   -123.11964267059481, 49.274188155916534
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    }
    ]
}
```

AWS CLI

AWS CLI 명령을 사용하여 위치 업데이트를 전송하려면

batch-update-device-position 명령을 사용합니다.

다음 예제에서는 AWS CLI a를 사용하여 기기 위치 업데이트를 게시합니다.*ExampleDevice-1* 그리고 *ExampleDevice-2* 트래커에 *ExampleTracker*.

```
aws location batch-update-device-position \
--tracker-name ExampleTracker \
--updates '[{"DeviceId":"ExampleDevice-1","Position":
[-123.123,47.123],"SampleTime":"2021-11-30T21:47:25.149Z"},
{"DeviceId":"ExampleDevice-2","Position":
[-123.123,47.123],"SampleTime":"2021-11-30T21:47:25.149Z","Accuracy":
{"Horizontal":10.30},"PositionProperties":{"field1":"value1","field2":"value2"}}]'
```

## 트래커에서 디바이스의 위치 기록 가져오기

Amazon Location 트래커 리소스는 추적한 모든 디바이스의 위치 기록을 30일 동안 유지합니다. 트래커 리소스에서 모든 관련 메타데이터를 포함한 디바이스 위치 기록을 검색할 수 있습니다. 다음 예에서는 AWS CLI, 또는 Amazon 위치를 사용합니다APIs.

API

Amazon Location을 사용하여 추적기에서 장치 위치 기록을 가져오려면 APIs

Amazon 위치 APIs 추적기에서 GetDevicePositionHistory 작업을 사용하십시오.

다음 예시에서는 API URI 요청을 사용하여 기기 위치 기록을 가져오는 예시입니다.*ExampleDevice* 라는 트래커에서 *ExampleTracker* 19:05:07(포함) 에서 시작하여 19:20:07 (제외) 에서 끝납니다2020-10-02.

```
POST /tracking/v0/trackers/ExampleTracker/devices/ExampleDevice/list-positions
Content-type: application/json
{
  "StartTimeInclusive": "2020-10-02T19:05:07.327Z",
  "EndTimeExclusive": "2020-10-02T19:20:07.327Z"
}
```

AWS CLI

AWS CLI 명령을 사용하여 트래커에서 기기 위치 기록을 가져오려면

get-device-position-history 명령을 사용합니다.

다음 예시에서는 AWS CLI a를 사용하여 다음 기기의 위치 기록을 가져옵니다.*ExampleDevice* 라는 트래커에서 *ExampleTracker* 19:05:07(포함) 에서 시작하여 19:20:07 (제외) 에서 끝납니다2020-10-02.

```
aws location \
    get-device-position-history \
        --device-id "ExampleDevice" \
        --start-time-inclusive "2020-10-02T19:05:07.327Z" \
        --end-time-exclusive "2020-10-02T19:20:07.327Z" \
        --tracker-name "ExampleTracker"
```

## 디바이스 위치를 나열합니다.

를 사용하거나 Amazon Location을 사용하여 트래커의 디바이스 위치 APIs 목록을 볼 수 ListDevicePositions API 있습니다. AWS CLI를 호출하면 해당 ListDevicePositions API 트래커와 연결된 모든 디바이스의 최신 위치 목록이 반환됩니다. 기본적으로 지정된 트래커에 대한 결과 페이지당 최신 기기 위치 100개가 API 반환됩니다. 특정 리전 내의 디바이스만 반환하려면 FilterGeometry 파라미터를 사용하여 경계 다각형 쿼리를 생성하세요. 이렇게 하면 ListDevicePositions 호출할 때 폴리곤 안에 있는 장치만 반환됩니다.

> **Note**
>
> 자체 AWS KMS 고객 관리 키를 사용하여 데이터를 암호화하려는 경우 바운딩 폴리곤 쿼리 기능은 기본적으로 비활성화됩니다. 이 기능을 사용하면 기기 위치 표현이 관리 키를 사용하여 암호화되지 않기 때문입니다. AWS KMS 하지만 정확한 디바이스 위치는 여전히 관리형 키를 사용하여 암호화됩니다.
> 경계 다각형 쿼리 기능을 선택할 수 있습니다. 트래커를 만들거나 업데이트할 때 `KmsKeyEnableGeospatialQueries` 파라미터를 true로 설정하면 됩니다.

API

Amazon 위치 APIs 추적기에서 `ListDevicePositions` 작업을 사용하십시오.

다음은 선택적 파라미터를 사용하여 다각형 영역의 디바이스 위치 목록을 가져오는 API 요청입니다. `FilterGeometry` 이 예시에서는 `Polygon` 배열로 정의된 영역에 있는 3개의 디바이스 위치를 반환합니다.

```
POST /tracking/v0/trackers/TrackerName/list-positions HTTP/1.1
Content-type: application/json

{
   "FilterGeometry": {
        "Polygon": [
          [
            [
              -123.12003339442259,
              49.27425121147397
            ],
            [
              -123.1176984148229,
              49.277063620879744
            ],
            [
              -123.12389509145294,
              49.277954183760926
            ],
            [
              -123.12755921328647,
              49.27554025235713
            ],
            [
```

```
              -123.12330236586217,
              49.27211836076236
            ],
            [
              -123.12003339442259,
              49.27425121147397
            ]
          ]
        ]
    },
   "MaxResults": 3,
   "NextToken": "1234-5678-9012"
}
```

다음은 ListDevicePositions에 대한 응답의 예입니다.

```
{
    "Entries": [
        {
            "DeviceId": "1",
            "SampleTime": "2022-10-24T19:09:07.327Z",
            "Position": [
                -123.12245146162303,
                49.27521118043802
            ],
            "Accuracy": {
                "Horizontal": 10
            },
            "PositionProperties": {
                "name": "device1"
            }
        },
        {
            "DeviceId": "3",
            "SampleTime": "2022-10-02T19:09:07.327Z",
            "Position": [
                -123.12325592118916,
                49.27340530543111
            ]
        },
        {
            "DeviceId": "2",
            "SampleTime": "2022-10-02T19:09:07.327Z",
```

```
            "Position": [
                -123.1230104928471,
                49.27752402723152
            ]
        }
    ],
   "NextToken": "1234-5678-9012"
}
```

CLI

`list-trackers` 명령을 사용합니다.

다음 예제는 다각형 영역의 장치 목록을 가져오는 것입니다. AWS CLI

```
aws location list-device-positions TODO: add arguments add props for filter geo
```

## 트래커를 지오펜스 컬렉션에 연결

이제 지오펜스 컬렉션과 트래커를 만들었으니, 이들을 서로 연결하여 위치 업데이트가 모든 지오펜스에 대해 자동으로 평가되도록 할 수 있습니다. 모든 위치 업데이트를 평가하고 싶지 않거나 일부 위치를 트래커 리소스에 저장하지 않는 경우 필요한 경우에만 디바이스 위치를 지오펜스와 비교하여 평가할 수 있습니다.

지오펜스를 기준으로 디바이스 위치를 평가하면 이벤트가 생성됩니다. 이러한 이벤트에 작업을 설정할 수 있습니다. 지오펜스 이벤트에 설정할 수 있는 작업에 대한 자세한 내용은 Amazon을 통한 Amazon Location Service 이벤트 대응을 참조하십시오. EventBridge

Amazon Location 이벤트는 이벤트를 생성하는 디바이스 위치 업데이트의 속성과 입장 또는 퇴장이 발생한 지오펜스의 일부 속성을 포함합니다. 지오펜스 이벤트에 포함된 데이터에 대한 자세한 내용은 Amazon Location Service의 아마존 EventBridge 이벤트 예제 항목을 참조하세요.

다음 예시는 콘솔 AWS CLI, 또는 Amazon Location을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결합니다. APIs

Console

Amazon Location Service 콘솔을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결하려면

1. 에서 Amazon Location Service 콘솔을 엽니다 https://console.aws.amazon.com/location/.

2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 디바이스 트래커에서 대상 트래커의 이름 링크를 선택합니다.
4. 연결된 지오펜스 컬렉션에서 지오펜스 컬렉션 연결을 선택합니다.
5. 연결된 지오펜스 컬렉션 창의 드롭다운 메뉴에서 지오펜스 컬렉션을 선택합니다.
6. 연결을 선택합니다.

트래커 리소스를 연결하면 해당 리소스에 활성 상태가 할당됩니다.

API

Amazon Location을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결하려면 APIs

Amazon 위치 APIs 추적기에서 AsssociateTrackerConsumer 작업을 사용하십시오.

다음 예시에서는 다음과 관련된 API 요청을 사용합니다.*ExampleTracker* Amazon 리소스 이름 () ARN 을 사용하는 지오펜스 컬렉션을 사용합니다.

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json

{
   "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection"
}
```

AWS CLI

명령을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결하는 방법 AWS CLI

associate-tracker-consumer 명령을 사용합니다.

다음 예제에서는 a를 AWS CLI 사용하여 라는 지오펜스 컬렉션을 만듭니다.*ExampleGeofenceCollection*.

```
aws location \
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection" \
        --tracker-name "ExampleTracker"
```

# 지오펜스를 기준으로 디바이스 위치 평가하기

지오펜스를 기준으로 위치를 평가하여 지오펜스 이벤트를 생성하는 두 가지 방법이 있습니다.

- 트래커와 지오펜스 컬렉션을 연결할 수 있습니다. 자세한 내용은 트래커를 지오펜스 컬렉션에 연결 섹션을 참조하세요.
- 를 사용하여 하나 이상의 위치를 평가하도록 지오펜스 컬렉션 리소스에 직접 요청할 수 있습니다. BatchEvaluateGeofencesAPI

또한 지오펜스에 들어오거나, 종료되거나, 지오펜스 내에서 유휴 상태로 남아 있는 장치에 대해 들어오는 지오펜스 이벤트를 예측할 수 있습니다. 를 사용하여 이벤트를 예측할 수 있습니다. ForecastGeofenceEventsAPI

디바이스 위치 기록을 추적하거나 맵에 위치를 표시하려면 트래커를 지오펜스 컬렉션과 연결합니다. 또는 모든 위치 업데이트를 평가하고 싶지 않거나 위치 데이터를 트래커 리소스에 저장하지 않을 수 있습니다. 두 가지 상황 중 하나에 해당하는 경우 지오펜스 컬렉션에 직접 요청하여 지오펜스와 비교하여 하나 이상의 디바이스 위치를 평가할 수 있습니다.

지오펜스와 비교하여 디바이스 위치를 평가하면 이벤트가 생성됩니다. 이러한 이벤트에 반응하여 다른 AWS 서비스로 라우팅할 수 있습니다. 지오펜스 이벤트를 수신할 때 취할 수 있는 조치에 대한 자세한 내용은 Amazon을 통한 Amazon Location Service 이벤트에 대응하기를 참조하십시오. EventBridge

Amazon Location 이벤트는 시간, 위치, 정확도, 키-값 메타데이터 등 이벤트를 생성하는 디바이스 위치 업데이트의 속성과 입장 또는 퇴장이 발생한 지오펜스의 일부 속성을 포함합니다. 지오펜스 이벤트에 포함된 데이터에 대한 자세한 내용은 Amazon Location Service의 아마존 EventBridge 이벤트 예제 항목을 참조하세요.

다음 예에서는 AWS CLI, 또는 Amazon 위치를 사용합니다APIs.

API

Amazon Location을 사용하여 지오펜스의 위치를 기준으로 디바이스 위치를 평가하려면 APIs

Amazon 로케이션 APIs 지오펜스에서 `BatchEvaluateGeofences` 오퍼레이션을 사용하십시오.

다음 예시에서는 API 요청을 사용하여 디바이스의 위치를 평가합니다.*`ExampleDevice`* 관련 지오펜스 컬렉션에 *`ExampleGeofenceCollection`*. 이 값을 자체 지오펜스 및 장치로 바꾸십시오. IDs

```
POST /geofencing/v0/collections/ExampleGeofenceCollection/positions HTTP/1.1
```

```
Content-type: application/json

{
   "DevicePositionUpdates": [
      {
         "DeviceId": "ExampleDevice",
         "Position": [-123.123, 47.123],
         "SampleTime": "2021-11-30T21:47:25.149Z",
         "Accuracy": {
            "Horizontal": 10.30
         },
         "PositionProperties": {
            "field1": "value1",
            "field2": "value2"
         }
      }
   ]
}
```

AWS CLI

명령을 사용하여 지오펜스의 위치를 기준으로 장치 위치를 평가하려면 AWS CLI

batch-evaluate-geofences 명령을 사용합니다.

다음 예제에서는 a를 AWS CLI 사용하여 위치를 평가합니다.*ExampleDevice* 관련 지오펜스 컬렉션에 대해 *ExampleGeofenceCollection*. 이 값을 자체 지오펜스 및 장치로 바꾸십시오. IDs

```
aws location \
    batch-evaluate-geofences \
        --collection-name ExampleGeofenceCollection \
        --device-position-updates '[{"DeviceId":"ExampleDevice","Position":
[-123.123,47.123],"SampleTime":"2021-11-30T21:47:25.149Z","Accuracy":
{"Horizontal":10.30},"PositionProperties":{"field1":"value1","field2":"value2"}}]'
```

지오펜스와 비교하여 디바이스 위치를 평가하면 이벤트가 생성됩니다. 일반적으로 를 사용하여 Amazon EventBridge이벤트에 대응할 수 있지만 이 프로세스를 사용하면 이벤트가 발생한 후에만 이벤트에 대응할 수 있습니다. 디바이스가 지오펜스에 들어오고 나가는 시점을 예측해야 하는 경우(예: 디바이스가 국경을 넘어서 다른 규정이 적용되는 경우) 를 사용하여 미래의 지오펜스 이벤트를 ForecastGeofenceEventsAPI예측할 수 있습니다.

는 장치 time-to-breach, 근접성, 속도, 위치 등의 기준을 ForecastGeofenceEventsAPI사용하여 이벤트를 예측합니다. 지오펜스 이벤트가 발생할 것으로 예상되는 시간을 알려주는 `ForecastedBreachTime` a가 API 반환됩니다.

다음 예시에서는 Amazon 로케이션을 사용합니다APIs.

API

Amazon 위치를 사용하여 지오펜스 이벤트를 예측하려면 APIs

Amazon 로케이션 APIs 지오펜스에서 `ForecastGeofenceEvents` 오퍼레이션을 사용하십시오.

다음 예시에서는 API 요청을 사용하여 특정 지역의 지오펜스 이벤트를 예측합니다.*`ExampleDevice`* 를 기준으로 *`ExampleGeofence`*. 이 값을 자체 지오펜스 및 장치로 바꾸십시오. IDs

```
POST /geofencing/v0/collections/CollectionName/forecast-geofence-events HTTP/1.1
Content-type: application/json

{
   "DeviceState": {
      "Position": [ number ],
      "Speed": number
   },
   "DistanceUnit": "string",
   "MaxResults": number,
   "NextToken": "string",
   "SpeedUnit": "string",
   "TimeHorizonMinutes": number
}
```

## 장치 위치를 확인하세요.

장치 위치의 무결성을 확인하려면 를 사용하십시오 VerifyDevicePositionAPI. 그러면 장치의 셀 신호, Wi-Fi 액세스 포인트, Ipv4 주소, 프록시 사용 여부 등의 속성을 평가하여 장치 위치의 무결성에 대한 정보가 API 반환됩니다.

### 사전 조건

나열된 APIs 항목을 장치 확인에 사용할 수 있으려면 먼저 다음 사전 요구 사항을 충족해야 합니다.

- 확인하려는 하나 또는 여러 개의 장치에 대한 추적기를 만들었습니다. 자세한 내용은 추적 시작 단원을 참조하십시오.

다음 예시는 Amazon 로케이션에 대한 요청을 보여줍니다 VerifyDevicePositionAPI.

API

Amazon 위치를 사용하여 기기 위치를 확인하려면 APIs

Amazon 위치 추적의 `VerifyDevicePosition` 작업을 사용하십시오APIs.

다음 예는 디바이스 위치의 무결성을 평가하기 위한 API 요청을 보여줍니다. 이 값을 자체 기기로 바꾸십시오IDs.

```
POST /tracking/v0/trackers/TrackerName/positions/verify HTTP/1.1
Content-type: application/json

{
   "DeviceState": {
      "Accuracy": {
         "Horizontal": number
      },
      "CellSignals": {
         "LteCellDetails": [
            {
               "CellId": number,
               "LocalId": {
                  "Earfcn": number,
                  "Pci": number
               },
               "Mcc": number,
               "Mnc": number,
               "NetworkMeasurements": [
                  {
                     "CellId": number,
                     "Earfcn": number,
                     "Pci": number,
                     "Rsrp": number,
                     "Rsrq": number
                  }
               ],
               "NrCapable": boolean,
               "Rsrp": number,
```

```
                "Rsrq": number,
                "Tac": number,
                "TimingAdvance": number
             }
          ]
       },
       "DeviceId": "ExampleDevice",
       "Ipv4Address": "string",
       "Position": [ number ],
       "SampleTime": "string",
       "WiFiAccessPoints": [
          {
             "MacAddress": "string",
             "Rss": number
          }
       ]
    },
    "DistanceUnit": "string"
}
```

> **Note**
>
> SDKIntegrity는 장치 확인과 관련된 향상된 기능을 제공하며 요청 시 사용할 수 있습니다. 에 액세스하려면 영업 지원 SDK 부서에 문의하십시오.

## 아마존을 통한 아마존 로케이션 서비스 이벤트에 대응하기 EventBridge

EventBridge Amazon은 Amazon Location과 같은 AWS 서비스의 데이터를 사용하여 애플리케이션을 효율적으로 연결하는 서버리스 이벤트 버스입니다. EventBridge Amazon Location에서 이벤트를 수신하고 해당 데이터를 다음과 같은 대상으로 AWS Lambda 라우팅합니다. 데이터를 전송할 대상을 결정하는 라우팅 규칙을 설정하여 실시간으로 대응하는 애플리케이션 아키텍처를 구축할 수 있습니다.

기본적으로 지오펜스 이벤트 (ENTER및 디바이스가 지오펜스 영역에 들어오거나 나가는 EXIT 이벤트)만 전송 대상으로 전송됩니다. EventBridge 트래커 리소스에 대해 필터링된 모든 위치 업데이트 이벤트를 활성화할 수도 있습니다. 자세한 설명은 트래커의 업데이트 이벤트를 활성화합니다. 섹션을 참조하세요.

자세한 내용은 Amazon EventBridge 사용 설명서의 이벤트 및 이벤트 패턴을 참조하십시오.

주제

- 트래커의 업데이트 이벤트를 활성화합니다.
- Amazon Location의 이벤트 규칙 생성
- Amazon Location Service의 아마존 EventBridge 이벤트 예제

## 트래커의 업데이트 이벤트를 활성화합니다.

기본적으로 Amazon Location은 EXIT 지오펜스 이벤트만 ENTER 전송합니다. EventBridge 트래커에 대해 필터링된 모든 포지션 UPDATE 이벤트를 전송하도록 활성화할 수 있습니다. EventBridge 트래커를 생성하거나 업데이트할 때 이 작업을 수행할 수 있습니다.

예를 들어 를 사용하여 기존 트래커를 업데이트하려면 다음 명령어를 사용할 수 있습니다 (대신 트래커 리소스 이름 사용 *MyTracker*). AWS CLI

```
aws location update-tracker --tracker-name MyTracker --event-bridge-enabled
```

트래커의 포지션 이벤트를 끄려면 API 또는 Amazon Location Service 콘솔을 사용해야 합니다.

## Amazon Location의 이벤트 규칙 생성

Amazon Location 이벤트에 대한 응답으로 취해진 조치를 EventBridge 구성하기 위해 이벤트 버스당 최대 300개의 규칙을 생성할 수 있습니다.

예를 들어, 지오펜스 경계 내에서 전화가 감지되면 푸시 알림이 전송되는 지오펜스 이벤트 규칙을 생성할 수 있습니다.

Amazon Location 이벤트에 대한 규칙을 만들려면

다음 값을 사용하여 Amazon Location 이벤트를 기반으로 EventBridge 규칙을 생성합니다.

- 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
- 이벤트 패턴 상자에 다음 패턴을 추가합니다:

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"]
}
```

다음 패턴을 사용하여 트래커 위치 업데이트 규칙을 생성할 수 있습니다:

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Device Position Event"]
}
```

`detail` 태그를 추가하여 `ENTER` 또는 `EXIT` 이벤트만 선택적으로 지정할 수도 있습니다 (규칙이 트래커 위치 업데이트에 대한 규칙인 경우 한개의 `EventType`만 존재하므로 필터링할 필요가 없음):

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"]
  }
}
```

또한 위치 또는 지오펜스의 속성을 기준으로 선택적으로 필터링할 수도 있습니다:

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"],
    "GeofenceProperties": {
      "Type": "LoadingDock"
    },
    "PositionProperties": {
      "VehicleType": "Truck"
    }
  }
}
```

- 대상 선택에서 Amazon Location Service로부터 이벤트가 수신될 때 수행할 대상 작업을 선택합니다.

  예를 들어, 이벤트 발생 시 Amazon Simple Notification Service (SNS) 주제를 사용하여 이메일 또는 텍스트 메시지를 보낼 수 있습니다. 먼저 Amazon SNS 콘솔을 사용하여 Amazon SNS 주제를 생성해야 합니다. 자세한 내용은 사용자 알림을 위한 Amazon SNS 사용을 참조하십시오.

> **Warning**
>
> 이벤트 규칙이 성공적으로 적용되었는지 확인하는 것이 가장 좋습니다. 그렇지 않으면 자동 작업이 예상대로 시작되지 않을 수 있습니다. 이벤트 규칙을 확인하려면 이벤트 규칙의 조건을 시작하십시오. 예를 들어, 장치가 지오펜스 영역에 진입하는 것을 시뮬레이션해 보십시오.

`detail-type` 섹션을 제외하여 Amazon Location에서 모든 이벤트를 캡처할 수도 있습니다. 예:

```
{
  "source": [
    "aws.geo"
  ]
}
```

> **Note**
>
> 동일한 이벤트가 두 번 이상 전송될 수 있습니다. 이벤트 ID를 사용하여 수신한 이벤트의 중복을 제거할 수 있습니다.

## Amazon Location Service의 아마존 EventBridge 이벤트 예제

다음은 `BatchUpdateDevicePosition` 호출로 시작된 지오펜스 진입 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "aa11aa22-33a-4a4a-aaa5-example",
  "detail-type": "Location Geofence Event",
  "source": "aws.geo",
  "account": "636103698109",
  "time": "2020-11-10T23:43:37Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:geo:eu-west-1:0123456789101:geofence-collection/GeofenceEvents-
GeofenceCollection_EXAMPLE",
    "arn:aws:geo:eu-west-1:0123456789101:tracker/Tracker_EXAMPLE"
  ],
  "detail": {
    "EventType": "ENTER",
```

```
    "GeofenceId": "polygon_14",
    "DeviceId": "Device1-EXAMPLE",
    "SampleTime": "2020-11-10T23:43:37.531Z",
    "Position": [
      -123.12390073297821,
      49.23433613216247
    ],
    "Accuracy": {
      "Horizontal": 15.3
    },
    "GeofenceProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    },
    "PositionProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    }
  }
}
```

다음은 BatchUpdateDevicePosition 호출로 시작된 지오펜스 퇴장 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "aa11aa22-33a-4a4a-aaa5-example",
  "detail-type": "Location Geofence Event",
  "source": "aws.geo",
  "account": "123456789012",
  "time": "2020-11-10T23:41:44Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:geo:eu-west-1:0123456789101:geofence-collection/GeofenceEvents-
GeofenceCollection_EXAMPLE",
    "arn:aws:geo:eu-west-1:0123456789101:tracker/Tracker_EXAMPLE"
  ],
  "detail": {
    "EventType": "EXIT",
    "GeofenceId": "polygon_10",
    "DeviceId": "Device1-EXAMPLE",
    "SampleTime": "2020-11-10T23:41:43.826Z",
    "Position": [
      -123.08569321875426,
```

```
      49.23766166742559
    ],
    "Accuracy": {
      "Horizontal": 15.3
    },
    "GeofenceProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    },
    "PositionProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    }
  }
}
```

다음은 BatchUpdateDevicePosition 호출로 시작된 위치 업데이트 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "aa11aa22-33a-4a4a-aaa5-example",
  "detail-type": "Location Device Position Event",
  "source": "aws.geo",
  "account": "123456789012",
  "time": "2020-11-10T23:41:44Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:geo:eu-west-1:0123456789101:tracker/Tracker_EXAMPLE"
  ],
  "detail": {
    "EventType": "UPDATE",
    "TrackerName": "tracker_2",
    "DeviceId": "Device1-EXAMPLE",
    "SampleTime": "2020-11-10T23:41:43.826Z",
    "ReceivedTime": "2020-11-10T23:41:39.235Z",
    "Position": [
      -123.08569321875426,
      49.23766166742559
    ],
    "Accuracy": {
      "Horizontal": 15.3
    },
    "PositionProperties": {
```

```
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    }
  }
}
```

## Amazon Location AWS IoT Service를 MQTT 이용한 추적

MQTT제약이 있는 디바이스용으로 설계된 가볍고 널리 채택되는 메시징 프로토콜입니다. AWS IoT Core MQTT프로토콜 및 MQTT over WebSocket Secure (WSS) 프로토콜을 사용하는 장치 연결을 지원합니다.

AWS IoT Core는 디바이스를 AWS 에 연결하고 디바이스 간에 메시지를 보내고 받을 수 있도록 지원합니다. AWS IoT Core 규칙 엔진은 디바이스의 메시지 주제에 대한 쿼리를 저장하고 Amazon Location Service와 같은 다른 AWS 서비스에 메시지를 보내기 위한 작업을 정의할 수 있도록 합니다. 자신의 위치를 좌표로 인식하는 디바이스는 규칙 엔진을 통해 Amazon Location으로 위치를 전달할 수 있습니다.

> **Note**
>
> 기기는 예를 들어 내장 기능을 통해 자신의 위치를 알 수 있습니다. GPS AWS IoT 또한 타사 장치 위치 추적을 지원합니다. 자세한 내용은 AWS IoT 코어 개발자 가이드의 AWS IoT 코어 디바이스 위치를 참조하세요.

다음 안내에서는 규칙을 사용한 AWS IoT Core 추적에 대해 설명합니다. Amazon Location으로 전송하기 전에 처리해야 하는 경우 디바이스 정보를 자체 AWS Lambda 기능으로 전송할 수도 있습니다. Lambda를 사용하여 디바이스 위치를 처리하는 방법에 대한 자세한 내용은 와 함께 사용 AWS Lambda MQTT 항목을 참조하세요.

주제

- 전제 조건
- 규칙 만들기 AWS IoT Core
- 콘솔에서 AWS IoT Core 규칙을 테스트하세요.
- 와 함께 사용 AWS Lambda MQTT

## 전제 조건

추적을 시작하기 전에 다음과 같은 전제 조건을 완료해야 합니다.

- 디바이스 위치 데이터를 전송할 트래커 리소스를 생성합니다.
- 트래커에 대한 AWS IoT Core 액세스 권한을 부여하는 IAM역할을 생성하십시오.

  이러한 단계를 따를 때는 다음 정책을 사용하여 트래커에 대한 액세스 권한을 부여하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
}
```

## 규칙 만들기 AWS IoT Core

다음으로, 디바이스의 위치 텔레메트리를 Amazon Location Service로 전달하는 AWS IoT Core 규칙을 생성합니다. 규칙 생성에 대한 자세한 내용은 AWS IoT Core 개발자 가이드에서 다음을 참조하세요.

- 새 AWS IoT 규칙 생성에 대한 정보를 제공하는 규칙 생성
- Amazon Location에 게시하기 위한 규칙 생성과 관련된 정보를 위한 Location 작업

## 콘솔에서 AWS IoT Core 규칙을 테스트하세요.

현재 위치가 포함된 원격 분석을 게시하는 장치가 없는 경우 AWS IoT Core 콘솔을 사용하여 규칙을 테스트할 수 있습니다. 콘솔에는 샘플 메시지를 게시하여 솔루션 결과를 확인할 수 있는 테스트 클라이언트가 있습니다.

1. 에서 AWS IoT Core https://console.aws.amazon.com/iot/콘솔에 로그인합니다.
2. 왼쪽 탐색 영역에서 테스트를 확장하고 MQTT 테스트 클라이언트를 선택합니다.
3. 주제에 게시에서 주제 이름을 다음과 같이 설정합니다.*iot/topic* (또는 AWS IoT Core 규칙에 설정한 주제 이름 (다른 경우) 를 입력하고 메시지 페이로드에 다음을 제공하십시오.

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. 테스트 메시지를 보내려면 주제 게시를 선택합니다.
5. Amazon Location Service에서 메시지를 수신했는지 확인하려면 다음 AWS CLI 명령을 사용하세요. 설정 중에 수정한 경우, 트래커 이름을 사용한 이름으로 바꿉니다.

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

## 와 함께 사용 AWS Lambda MQTT

추적을 위해 Amazon Location으로 디바이스 위치 데이터를 전송할 때 더 이상 를 사용하지 않아도 되지만, 경우에 AWS Lambda 따라서는 Lambda를 계속 사용하고 싶을 수도 있습니다. 예를 들어, Amazon Location으로 전송하기 전에 디바이스 위치 데이터를 직접 처리하는 경우가 있습니다. 다음 주제에서는 Lambda를 사용하여 메시지를 트래커로 보내기 전에 메시지를 처리하는 방법을 설명합니다. 이 패턴에 대한 자세한 내용은 참조 아키텍처를 참조하세요.

주제

- 전제 조건
- Lambda 함수 생성
- 규칙 만들기 AWS IoT Core
- 콘솔에서 AWS IoT Core 규칙을 테스트하세요.

### 전제 조건

추적을 시작하려면 먼저 트래커 리소스를 만들어야 합니다. 트래커 리소스를 생성하려면 Amazon Location 콘솔 AWS CLI, 또는 Amazon Location을 사용할 수 APIs 있습니다.

다음 예시에서는 Amazon Location Service 콘솔을 사용하여 트래커 리소스를 생성합니다.

1. 에서 Amazon Location Service 콘솔을 엽니다 https://console.aws.amazon.com/location/.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 트래커 생성을 선택합니다.
4. 다음 입력란을 작성합니다.
   - 이름 – 최대 100자의 입력할 수 있는 고유한 이름을 입력합니다. 영숫자, 하이픈, 밑줄을 입력할 수 있습니다. 예: *MyTracker*.
   - 설명 – 선택적 설명을 입력합니다. 예: *Tracker for storing AWS IoT Core device positions*.
   - 위치 필터링 – 위치 업데이트에 사용할 필터링을 선택합니다. 예를 들어, 정확도 기반 필터링이 있습니다.
5. 트래커 생성을 선택합니다.

### Lambda 함수 생성

Amazon Location Service 간에 AWS IoT Core 연결을 생성하려면 에서 전달한 AWS IoT Core메시지를 처리하는 AWS Lambda 함수가 필요합니다. 이 함수는 모든 위치 데이터를 추출하여 Amazon Location Service에 맞게 형식을 지정한 다음 Amazon 위치 추적기를 API 통해 제출합니다. AWS Lambda 콘솔을 통해 이 함수를 만들거나 AWS Command Line Interface (AWS CLI) 또는 를 사용할 수 있습니다. AWS Lambda APIs

콘솔을 사용하여 Amazon Location에 위치 업데이트를 게시하는 Lambda 함수를 생성하려면

1. 에서 AWS Lambda 콘솔을 엽니다 https://console.aws.amazon.com/lambda/.
2. 왼쪽 탐색에서 함수를 선택합니다.
3. 함수 생성을 선택하고 새로 작성이 선택되어 있는지 확인합니다.
4. 다음 입력란을 작성합니다.
   - 함수 이름 – 함수에 고유한 이름을 입력합니다. 유효한 항목에는 공백 없는 영숫자 문자, 하이픈 및 밑줄이 포함됩니다. 예: *MyLambda*.
   - 런타임 — 선택 *Python 3.8*.
5. 함수 생성(Create function)을 선택합니다.
6. Code 탭을 선택하여 편집기를 엽니다.
7. `lambda_function.py`의 플레이스홀더 코드를 다음으로 덮어쓰고, `TRACKER_NAME`에 할당된 값을 사전 조건으로 생성한 트래커 이름으로 바꿉니다.

```
from datetime import datetime
import json
import os

import boto3

# Update this to match the name of your Tracker resource
TRACKER_NAME = "MyTracker"

"""
This Lambda function receives a payload from AWS IoT Core and publishes device
 updates to
Amazon Location Service via the BatchUpdateDevicePosition API.

Parameter 'event' is the payload delivered from AWS IoT Core.

In this sample, we assume that the payload has a single top-level key 'payload' and
 a nested key
'location' with keys 'lat' and 'long'. We also assume that the name of the device
 is nested in
the payload as 'deviceid'. Finally, the timestamp of the payload is present as
 'timestamp'. For
example:

>>> event
{ 'payload': { 'deviceid': 'thing123', 'timestamp': 1604940328,
  'location': { 'lat': 49.2819, 'long': -123.1187 },
  'accuracy': {'Horizontal': 20.5 },
  'positionProperties': {'field1':'value1','field2':'value2'} }
}

If your data doesn't match this schema, you can either use the AWS IoT Core rules
 engine to
format the data before delivering it to this Lambda function, or you can modify the
 code below to
match it.
"""
def lambda_handler(event, context):
  update = {
      "DeviceId": event["payload"]["deviceid"],
      "SampleTime": datetime.fromtimestamp(event["payload"]
["timestamp"]).strftime("%Y-%m-%dT%H:%M:%SZ"),
      "Position": [
```

```
        event["payload"]["location"]["long"],
        event["payload"]["location"]["lat"]
        ]
    }
  if "accuracy" in event["payload"]:
      update["Accuracy"] = event["payload"]['accuracy']
  if "positionProperties" in event["payload"]:
      update["PositionProperties"] = event["payload"]['positionProperties']

  client = boto3.client("location")
  response = client.batch_update_device_position(TrackerName=TRACKER_NAME,
 Updates=[update])

  return {
    "statusCode": 200,
    "body": json.dumps(response)
  }
```

8. [배포]를 선택하여 업데이트된 함수를 저장합니다.
9. 구성 탭을 선택합니다.
10. 권한 섹션에서 하이퍼링크된 역할 이름을 선택하여 Lambda 함수에 Amazon Location Service 권한을 부여합니다.
11. 역할의 요약 페이지에서 권한 추가를 선택한 다음 드롭다운 목록에서 인라인 정책 생성을 선택합니다.
12. JSON탭을 선택하고 다음 문서로 정책을 덮어씁니다. 이를 통해 Lambda 함수는 모든 리전의 모든 트래커 리소스에서 관리하는 디바이스 위치를 업데이트할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
}
```

13. 정책 검토를 선택합니다.
14. 정책 이름을 입력합니다. 예: *AmazonLocationTrackerWriteOnly*.

15. 정책 생성을 선택합니다.

필요에 따라 이 함수 코드를 수정하여 자체 디바이스 메시지 스키마에 맞게 조정할 수 있습니다.

규칙 만들기 AWS IoT Core

다음으로, 디바이스의 위치 텔레메트리를 AWS Lambda 함수로 전달하여 Amazon Location Service로 변환하여 게시하는 AWS IoT Core 규칙을 생성합니다. 제공된 예시 규칙은 필요한 디바이스 페이로드 변환이 Lambda 함수에서 처리된다고 가정합니다. AWS IoT Core 콘솔, AWS Command Line Interface (AWS CLI) 또는 를 통해 이 규칙을 생성할 수 있습니다. AWS IoT Core APIs

> **Note**
>
> AWS IoT 콘솔은 Lambda 함수 호출을 허용하는 AWS IoT Core 데 필요한 권한을 처리하지만, SDK OR에서 AWS CLI 규칙을 생성하는 경우 권한을 부여할 정책을 구성해야 합니다. AWS IoT

콘솔을 사용하여 AWS IoT Core 생성하려면

1. 에서 AWS IoT Core 콘솔에 https://console.aws.amazon.com/iot/로그인합니다.
2. 왼쪽 탐색에서 [작업]를 확장하고 [규칙]를 선택합니다.
3. 규칙 생성을 선택하여 새 규칙 마법사를 시작합니다.
4. 규칙의 이름과 설명을 입력합니다.
5. 규칙 쿼리 문의 경우, 하나 이상의 디바이스에서 위치가 포함된 원격 측정을 게시하는 주제를 참조하도록 FROM 속성을 업데이트합니다. 솔루션을 테스트하는 경우에는 수정할 필요가 없습니다.

   ```
   SELECT * FROM 'iot/topic'
   ```

6. 하나 이상의 작업 설정 에서 작업 추가를 선택합니다.
7. 메시지를 Lambda 함수로 전송를 선택합니다.
8. [Configure action]을 선택합니다.
9. 목록에서 Lambda 함수 를 찾아 선택합니다.
10. 작업 추가를 선택합니다.
11. Create rule을 선택합니다.

콘솔에서 AWS IoT Core 규칙을 테스트하세요.

현재 위치가 포함된 원격 분석을 게시하는 장치가 없는 경우 AWS IoT Core 콘솔을 사용하여 규칙과 이 솔루션을 테스트할 수 있습니다. 콘솔에는 샘플 메시지를 게시하여 솔루션 결과를 확인할 수 있는 테스트 클라이언트가 있습니다.

1. 에서 AWS IoT Core https://console.aws.amazon.com/iot/콘솔에 로그인합니다.
2. 왼쪽 탐색 영역에서 테스트를 확장하고 MQTT 테스트 클라이언트를 선택합니다.
3. 주제에 게시에서 주제 이름을 다음과 같이 설정합니다.*iot/topic* (또는 AWS IoT Core 규칙에 설정한 주제 이름 (다른 경우) 를 입력하고 메시지 페이로드에 다음을 제공하십시오. 타임스탬프를 교체하십시오.*1604940328* 최근 30일 이내의 유효한 타임스탬프로 표시합니다 (30일이 지난 타임스탬프는 무시됨).

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. 테스트 메시지를 보내려면 주제 게시를 선택합니다.
5. Amazon Location Service에서 메시지를 수신했는지 확인하려면 다음 AWS CLI 명령을 사용하세요. 설정 중에 수정한 경우 트래커 이름과 디바이스 ID를 사용했던 것으로 바꾸세요.

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

## 지오펜스 컬렉션 리소스 관리

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 지오펜스 컬렉션을 관리합니다.

### 지오펜스 컬렉션 리소스를 나열합니다.

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 지오펜스 컬렉션 목록을 볼 수 있습니다.

### Console

Amazon Location 콘솔을 사용하여 지오펜스 컬렉션 목록을 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 내 지오펜스 컬렉션에서 지오펜스 컬렉션 목록을 볼 수 있습니다.

### API

Amazon Location 지오펜스 API에서 `ListGeofenceCollections` 작업을 사용합니다.

다음 예시는 AWS 계정의 지오펜스 컬렉션 목록을 가져오기 위한 API 요청입니다.

```
POST /geofencing/v0/list-collections
```

다음은 `ListGeofenceCollections`에 대한 응답의 예입니다:

```
{
    "Entries": [
    {
        "CollectionName": "ExampleCollection",
        "CreateTime": 2020-09-30T22:59:34.142Z,
        "Description": "string",
        "UpdateTime": 2020-09-30T23:59:34.142Z
    },
    "NextToken": "1234-5678-9012"
}
```

### CLI

`list-geofence-collections` 명령을 사용합니다.

다음 예시는 AWS CLI 계정의 지오펜스 컬렉션 목록을 가져오는 AWS입니다.

```
aws location list-geofence-collections
```

## 지오펜스 컬렉션 세부 정보 가져오기

Amazon Location 콘솔, AWS CLI또는 Amazon Location API를 사용하여 AWS 계정의 모든 지오펜스 컬렉션 리소스에 대한 세부 정보를 확인할 수 있습니다.

### Console

Amazon Location 콘솔을 사용하여 지오펜스 컬렉션의 세부 정보를 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 내 지오펜스 컬렉션에서 대상 지오펜스 컬렉션의 이름 링크를 선택합니다.

### API

Amazon Location 지오펜스 API에서 `DescribeGeofenceCollection` 작업을 사용합니다.

다음 예시는 지오펜스 수집 세부 정보를 가져오기 위한 API 요청입니다. *ExampleCollection*

```
GET /geofencing/v0/collections/ExampleCollection
```

다음은 DescribeGeofenceCollection에 대한 응답의 예입니다:

```
{
    "CollectionArn": "arn:aws:geo:us-west-2:123456789012:geofence-collection/
GeofenceCollection",
    "CollectionName": "ExampleCollection",
    "CreateTime": 2020-09-30T22:59:34.142Z,
    "Description": "string",
    "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Tags": {
        "Tag1" : "Value1"
    },
    "UpdateTime": 2020-09-30T23:59:34.142Z
}
```

### CLI

`describe-geofence-collection` 명령을 사용합니다.

다음 예제는 지오펜스 AWS CLI 컬렉션 세부 정보를 가져오는 예입니다. *ExampleCollection*

```
aws location describe-geofence-collection \
    --collection-name "ExampleCollection"
```

## 지오펜스 컬렉션 삭제

Amazon Location 콘솔, AWS 또는 Amazon Location API를 사용하여 AWS CLI 계정에서 지오펜스 컬렉션을 삭제할 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 지오펜스 컬렉션을 삭제하려면

> ⚠ Warning
> 이 작업은 리소스를 영구적으로 삭제합니다.

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 내 지오펜스 컬렉션에서 대상 지오펜스 컬렉션을 선택합니다.
4. 지오펜스 컬렉션 삭제를 선택합니다.

API

Amazon Location API에서 `DeleteGeofenceCollection` 작업을 사용합니다.

다음은 지오펜스 컬렉션을 삭제하기 위한 API 요청입니다. *ExampleCollection*

```
DELETE /geofencing/v0/collections/ExampleCollection
```

다음은 DeleteGeofenceCollection에 대한 응답의 예입니다:

```
HTTP/1.1 200
```

CLI

`delete-geofence-collection` 명령을 사용합니다.

다음 예시는 지오펜스 AWS CLI 컬렉션을 삭제하는 명령입니다. *ExampleCollection*

```
aws location delete-geofence-collection \
    --collection-name "ExampleCollection"
```

## 저장된 지오펜스 나열

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 지정된 지오펜스 컬렉션에 저장된 지오펜스를 나열할 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 지오펜스 목록을 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 내 지오펜스 컬렉션에서 대상 지오펜스 컬렉션의 이름 링크를 선택합니다.
4. 지오펜스의 지오펜스 컬렉션에서 지오펜스 확인하기

API

Amazon Location 지오펜스 API에서 `ListGeofences` 작업을 사용합니다.

다음은 지오펜스 컬렉션에 저장된 지오펜스 목록을 가져오기 위한 API 요청입니다. *ExampleCollection*

```
POST /geofencing/v0/collections/ExampleCollection/list-geofences
```

다음은 `ListGeofences`에 대한 응답의 예입니다:

```
{
   "Entries": [
      {
         "CreateTime": 2020-09-30T22:59:34.142Z,
         "GeofenceId": "geofence-1",
         "Geometry": {
             "Polygon": [
                 [-5.716667, -15.933333,
```

```
                [-14.416667, -7.933333],
                [-12.316667, -37.066667],
                [-5.716667, -15.933333]
            ]
        },
        "Status": "ACTIVE",
        "UpdateTime": 2020-09-30T23:59:34.142Z
    }
  ],
  "NextToken": "1234-5678-9012"
}
```

CLI

`list-geofences` 명령을 사용합니다.

다음 예제는 지오펜스 AWS CLI 컬렉션에 저장된 지오펜스 목록을 가져오는 것입니다. *ExampleCollection*

```
aws location list-geofences \
    --collection-name "ExampleCollection"
```

## 지오펜스 세부 정보 가져오기

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하는 지오펜스 컬렉션에서 생성 시간, 업데이트 시간, 지오메트리, 상태 등 특정 지오펜스의 세부 정보를 가져올 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 지오펜스의 상태를 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 내 지오펜스 컬렉션에서 대상 지오펜스 컬렉션의 이름 링크를 선택합니다.
4. 지오펜스에서 지오펜스의 상태를 볼 수 있습니다.

API

Amazon Location 지오펜스 API에서 `GetGeofence` 작업을 사용합니다.

다음 예시는 지오펜스 컬렉션에서 지오펜스 세부 정보를 가져오기 위한 API 요청입니다. *ExampleCollection*

```
GET /geofencing/v0/collections/ExampleCollection/geofences/ExampleGeofence1
```

다음은 GetGeofence에 대한 응답의 예입니다:

```
{
   "CreateTime": 2020-09-30T22:59:34.142Z,
   "GeofenceId": "ExampleGeofence1",
   "Geometry": {
      "Polygon": [
          [-1,-1],
          [1,-1],
          [0,1],
          [-1,-1]
      ]
   },
   "Status": "ACTIVE",
   "UpdateTime": 2020-09-30T23:59:34.142Z
}
```

CLI

get-geofence 명령을 사용합니다.

다음 예제는 지오펜스 컬렉션 AWS CLI 세부 정보를 가져오는 예입니다. *ExampleCollection*

```
aws location get-geofence \
    --collection-name "ExampleCollection" \
    --geofence-id "ExampleGeofence1"
```

## 지오펜스 삭제

Amazon Location 콘솔, AWS CLI또는 Amazon Location API를 사용하여 지오펜스 컬렉션에서 지오펜스를 삭제할 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 지오펜스를 삭제하려면

> **Warning**
> 이 작업은 리소스를 영구적으로 삭제합니다.

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 지오펜스 컬렉션을 선택합니다.
3. 내 지오펜스 컬렉션에서 대상 지오펜스 컬렉션의 이름 링크를 선택합니다.
4. 지오펜스에서 대상 지오펜스를 선택합니다.
5. 지오펜스 삭제를 선택합니다.

API

Amazon Location 지오펜스 API에서 BatchDeleteGeofence 작업을 사용합니다.

다음은 지오펜스 컬렉션에서 지오펜스를 삭제하기 위한 API 요청입니다. *ExampleCollection*

```
POST /geofencing/v0/collections/ExampleCollection/delete-geofences
Content-type: application/json

{
   "GeofenceIds": [ "ExampleGeofence11" ]
}
```

다음은 BatchDeleteGeofence에 대한 성공적인 응답의 예입니다.

```
HTTP/1.1 200
```

CLI

batch-delete-geofence 명령을 사용합니다.

다음 예시는 지오펜스 컬렉션에서 지오펜스를 삭제하는 AWS CLI 명령입니다. *ExampleCollection*

```
aws location batch-delete-geofence \
```

```
    --collection-name "ExampleCollection" \
    --geofence-ids "ExampleGeofence11"
```

# 트래커 리소스 관리

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 트래커를 관리할 수 있습니다.

## 트래커 목록

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 트래커 목록을 볼 수 있습니다:

Console

Amazon Location 콘솔을 사용하여 기존 트래커 목록을 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 내 트래커에서 트래커 리소스 목록을 확인합니다.

API

Amazon Location Trackers API에서 `ListTrackers` 작업을 사용합니다.

다음 예는 AWS 계정의 트래커 목록을 가져오기 위한 API 요청입니다.

```
POST /tracking/v0/list-trackers
```

다음은 `ListTrackers`에 대한 응답의 예입니다:

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-02T19:09:07.327Z,
         "Description": "string",
         "TrackerName": "ExampleTracker",
         "UpdateTime": 2020-10-02T19:10:07.327Z
      }
```

```
    ],
    "NextToken": "1234-5678-9012"
}
```

CLI

list-trackers 명령을 사용합니다.

다음 예는 AWS 계정의 트래커 목록을 가져오는 AWS CLI입니다.

```
aws location list-trackers
```

## 지오펜스 컬렉션에서 트래커 연결 끊기

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 지오펜스 컬렉션에서 트래커의 연결을 끊을 수 있습니다:

Console

Amazon Location 콘솔을 사용하여 관련 지오펜스 컬렉션에서 트래커를 분리하려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 내 트래커에서 대상 트래커의 이름 링크를 선택합니다.
4. 연결된 지오펜스 컬렉션에서 연결 상태의 지오펜스 컬렉션을 선택합니다.
5. 링크 해제를 선택합니다.

API

Amazon Location Trackers APIs에서 DisassociateTrackerConsumer 작업을 사용합니다.

다음 예는 관련 지오펜스 컬렉션에서 트래커를 분리하기 위한 API 요청입니다.

```
DELETE /tracking/v0/trackers/ExampleTracker/consumers/arn:aws:geo:us-
west-2:123456789012:geofence-collection/ExampleCollection
```

다음은 DisassociateTrackerConsumer에 대한 응답의 예입니다:

```
HTTP/1.1 200
```

CLI

disassociate-tracker-consumer 명령을 사용합니다.

다음 예시는 관련 지오펜스 컬렉션에서 트래커를 분리하는 AWS CLI 명령입니다.

```
aws location disassociate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-collection/
ExampleCollection" \
    --tracker-name "ExampleTracker"
```

## 트래커 세부 정보 가져오기

Amazon Location 콘솔AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정의 모든 트래커에 대한 세부 정보를 얻을 수 있습니다.

Console

Amazon Location 콘솔을 사용하여 트래커 세부 정보를 보려면

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 내 트래커에서 대상 트래커의 이름 링크를 선택합니다.
4. 정보에서 트래커 세부 정보를 확인합니다.

API

Amazon Location Tracker API에서 DescribeTracker 작업을 사용합니다.

다음 예시는 트래커 세부 정보를 가져오기 위한 API *ExampleTracker*요청입니다.

```
GET /tracking/v0/trackers/ExampleTracker
```

다음은 DescribeTracker에 대한 응답의 예입니다:

```
{
```

```
   "CreateTime": 2020-10-02T19:09:07.327Z,
   "Description": "string",
   "EventBridgeEnabled": false,
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "PositionFiltering": "TimeBased",
   "Tags": {
      "Tag1" : "Value1"
   },
   "TrackerArn": "arn:aws:geo:us-west-2:123456789012:tracker/ExampleTracker",
   "TrackerName": "ExampleTracker",
   "UpdateTime": 2020-10-02T19:10:07.327Z
}
```

CLI

describe-tracker 명령을 사용합니다.

다음 예시는 트래커 세부 정보를 가져오는 AWS CLI 명령어입니다 *ExampleTracker*.

```
aws location describe-tracker \
    --tracker-name "ExampleTracker"
```

## 트래커 삭제

Amazon 로케이션 콘솔AWS CLI 또는 Amazon Location API를 사용하여 AWS 계정에서 트래커를 삭제할 수 있습니다:

Console

Amazon Location 콘솔을 사용하여 기존 맵 리소스를 삭제하려면

> **Warning**
>
> 이 작업은 리소스를 영구적으로 삭제합니다. 트래커 리소스를 사용 중인 경우 오류가 발생할 수 있습니다. 대상 리소스가 애플리케이션의 종속 항목이 아닌지 확인하세요.

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.

3. 내 트래커에서 대상 트래커를 선택합니다.
4. 트래커 삭제를 선택합니다.

API

Amazon Location Tracker API에서 DeleteTracker 작업을 사용합니다.

다음은 트래커 삭제를 위한 API *ExampleTracker*요청입니다.

```
DELETE /tracking/v0/trackers/ExampleTracker
```

다음은 DeleteTracker에 대한 응답의 예입니다:

```
HTTP/1.1 200
```

CLI

delete-tracker 명령을 사용합니다.

다음 예시는 트래커를 삭제하는 AWS CLI *ExampleTracker*명령입니다.

```
aws location delete-tracker \
    --tracker-name "ExampleTracker"
```

## 지오펜싱 및 트래킹 모바일 애플리케이션 샘플

이 주제에서는 모바일 애플리케이션에서 Amazon Location 지오펜스 및 트래커를 사용하는 주요 기능을 시연하기 위해 고안된 자습서를 다룹니다. 애플리케이션은 Lambda와 Amazon Location 기능의 조합을 사용하여 트래커와 지오펜스가 상호 작용하는 방식을 보여줍니다. AWS IoT 두 개의 자습서를 사용할 수 있습니다.

- Android용 샘플 트래킹 및 지오펜싱 애플리케이션으로, /tree/main/에서 프로젝트 파일을 복제할 수 있습니다. GitHub https://github.com/aws-geospatial/ amazon-location-samples-android tracking-with-geofence-notifications
- iOS용 샘플 트래킹 및 지오펜싱 애플리케이션. /tree/main/에서 GitHub 프로젝트 파일을 복제할 수 있습니다. https://github.com/aws-geospatial/ amazon-location-samples-ios tracking-with-geofence-notifications

# Android용 샘플 트래킹 및 지오펜스 애플리케이션

이 주제에서는 모바일 애플리케이션에서 Amazon Location 지오펜스 및 트래커를 사용하는 주요 기능을 시연하기 위해 설계된 Android 자습서를 다룹니다. 애플리케이션은 Lambda와 Amazon Location 기능의 조합을 사용하여 트래커와 geonfence가 상호 작용하는 방식을 보여줍니다. AWS IoT

주제

- 앱을 위한 Amazon 위치 리소스 생성
- 지오펜스 컬렉션 만들기
- 트래커를 지오펜스 컬렉션에 연결
- AWS Lambda를 MQTT와 함께 사용하기
- 샘플 앱 코드를 설정합니다.
- 샘플 앱 사용

앱을 위한 Amazon 위치 리소스 생성

시작하려면 필요한 Amazon 위치 리소스를 생성해야 합니다. 이러한 리소스는 애플리케이션의 기능과 제공된 코드 스니펫을 실행하는 데 필수적입니다.

> **Note**
>
> 계정을 만들지 않았다면 AWS AWS 계정 관리 사용 설명서의 지침을 따르세요.

시작하려면 다음 절차를 사용하여 Amazon Cognito 자격 증명 풀 ID를 생성해야 합니다.

1. Amazon Cognito 콘솔을 열고 왼쪽 메뉴에서 자격 증명 풀을 선택한 다음 자격 증명 풀 생성을 선택합니다.
2. 게스트 액세스가 선택되어 있는지 확인하고 다음을 눌러 계속하십시오.
3. 다음으로 새 IAM 역할을 만들거나 기존 IAM 역할을 사용합니다.
4. 자격 증명 풀 이름을 입력하고 자격 증명 풀이 다음 절차에서 생성할 맵 및 추적기의 Amazon Location (geo) 리소스에 액세스할 수 있는지 확인합니다.

다음으로 AWS Amazon Location 콘솔에서 맵을 생성하고 스타일을 지정해야 합니다. 다음 절차를 사용하십시오.

1. Amazon 위치 콘솔의 맵 섹션으로 이동하여 맵 생성을 선택합니다.
2. 새 맵에 이름과 설명을 입력합니다. 튜토리얼 뒷부분에서 사용할 수 있도록 지정한 이름을 기록해 두십시오.
3. 맵 스타일을 선택할 때는 맵 데이터 공급자를 고려해 보세요. 자세한 내용은 AWS 서비스 약관의 섹션 82을 참조하십시오.
4. Amazon 위치 이용 약관에 동의한 다음 맵 생성을 선택하여 맵 생성 프로세스를 완료합니다.

다음으로 Amazon Location 콘솔에서 트래커를 생성해야 합니다. 다음 절차를 사용하십시오.

1. Amazon 위치 콘솔에서 지도 섹션을 엽니다.
2. 트래커 생성을 선택합니다.
3. 필수 필드를 입력합니다. 트래커의 이름은 이 튜토리얼 전체에서 참조되므로 기록해 두십시오.
4. 위치 필터링 필드에서 트래커 리소스를 사용하려는 방식에 가장 적합한 옵션을 선택합니다. 포지션 필터링을 설정하지 않은 경우 기본 설정은 입니다`TimeBased`. 자세한 내용은 트래커 및 PositionFilteringAmazon 위치 API 참조를 참조하십시오.
5. 트래커 생성을 선택하여 트래커 생성을 완료하십시오.

지오펜스 컬렉션 만들기

이제 지오펜스 컬렉션을 만들어 볼까요? 콘솔, API 또는 CLI를 사용할 수 있습니다. 다음 절차는 각 옵션을 안내합니다.

- Amazon 로케이션 콘솔을 사용하여 지오펜스 컬렉션을 생성합니다.

  1. Amazon 로케이션 콘솔의 지오펜스 컬렉션 섹션을 엽니다.
  2. 지오펜스 컬렉션 생성을 선택합니다.
  3. 컬렉션의 이름과 설명을 입력합니다.
  4. 대상으로 사용하는 EventBridge 규칙에서 선택적 EventBridge 규칙을 생성하여 Amazon CloudWatch 지오펜스 이벤트에 대한 대응을 시작할 수 있습니다. 이렇게 하면 Amazon Location에서 이벤트를 게시할 수 Amazon CloudWatch Logs있습니다.
  5. 지오펜스 컬렉션 생성을 눌러 컬렉션 생성을 완료합니다.
- Amazon 로케이션 API를 사용하여 지오펜스 컬렉션을 생성하십시오.

  Amazon 로케이션 지오펜스 API에서 CreateGeofenceCollection작업을 사용하십시오. 다음 예시에서는 API 요청을 사용하여 라는 지오펜스 컬렉션을 생성합니다. *GEOCOLLECTION_NAME*

```
POST /geofencing/v0/collections
Content-type: application/json

{
   "CollectionName": "GEOCOLLECTION_NAME",
   "Description": "Geofence collection 1 for shopping center",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

- CLI AWS 명령을 사용하여 지오펜스 컬렉션을 생성합니다.

  `create-geofence-collection` 명령을 사용합니다. 다음 예제에서는 AWS CLI를 사용하여 라는 지오펜스 컬렉션을 생성합니다. *GEOCOLLECTION_NAME* AWS CLI 사용에 대한 자세한 내용은 AWS 명령줄 인터페이스 설명서를 참조하십시오.

  ```
  aws location \
    create-geofence-collection \
    --collection-name "ExampleGeofenceCollection" \
    --description "Shopping center geofence collection" \
    --tags Tag1=Value1
  ```

트래커를 지오펜스 컬렉션에 연결

트래커를 지오펜스 컬렉션에 연결하려면 콘솔, API 또는 CLI를 사용할 수 있습니다. 다음 절차는 각 옵션을 안내합니다.

Amazon Location Service 콘솔을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결합니다.

1. Amazon 로케이션 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 디바이스 트래커에서 대상 트래커의 이름 링크를 선택합니다.
4. 연결된 지오펜스 컬렉션에서 지오펜스 컬렉션 연결을 선택합니다.
5. 연결된 지오펜스 컬렉션 창의 드롭다운 메뉴에서 지오펜스 컬렉션을 선택합니다.
6. 연결을 선택합니다.
7. 트래커 리소스를 연결하면 해당 리소스에 활성 상태가 할당됩니다.

Amazon Location API를 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결합니다.

Amazon Location Trackers API에서 `AsssociateTrackerConsumer` 작업을 사용합니다. 다음 예시에서는 Amazon 리소스 이름 (ARN) 을 사용하여 지오펜스 컬렉션과 연결하는 API 요청을 사용합니다. `ExampleTracker`

```
        POST /tracking/v0/trackers/ExampleTracker/consumers
        Content-type: application/json
        {
           "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME"
            }
```

CLI AWS 명령을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결합니다.

`associate-tracker-consumer` 명령을 사용합니다. 다음 예제에서는 AWS CLI를 사용하여 라는 지오펜스 컬렉션을 생성합니다. *GOECOLLECTION_NAME*

```
aws location \
associate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME" \
    --tracker-name "ExampleTracker"
```

AWS Lambda를 MQTT와 함께 사용하기

Amazon Location 간에 AWS IoT 연결을 생성하려면 이벤트에 의해 전달된 메시지를 처리하는 Lambda 함수가 필요합니다. EventBridge CloudWatch 이 함수는 모든 위치 데이터를 추출하여 Amazon 위치에 맞게 형식을 지정한 다음 Amazon 위치 추적기 API를 통해 제출합니다.

다음 절차는 Lambda 콘솔을 통해 이 함수를 생성하는 방법을 보여줍니다.

1. 콘솔을 엽니다.
2. 왼쪽 탐색에서 함수를 선택합니다.
3. 그런 다음 [함수 생성] 을 선택하고 [처음부터 작성] 옵션이 선택되었는지 확인합니다.
4. 함수 이름을 제공하고 [런타임] 옵션으로 Node.js 16.x를 선택합니다.

5. 함수 생성을 선택합니다.
6. 코드 탭을 열어 편집기에 액세스합니다.
7. index.js파일의 플레이스홀더 코드를 다음과 같이 덮어씁니다.

```
            const AWS = require('aws-sdk')
            const iot = new AWS.Iot();
            exports.handler =  function(event) {
              console.log("event===>>>", JSON.stringify(event));
              var param = {
                endpointType: "iot:Data-ATS"
              };
              iot.describeEndpoint(param, function(err, data) {
                if (err) {
                  console.log("error===>>>", err, err.stack); // an error occurred
                } else {
                  var endp = data['endpointAddress'];
                  const iotdata = new AWS.IotData({endpoint: endp});
                  const trackerEvent = event["detail"]["EventType"];
                  const src = event["source"];
                  const time = event["time"];
                  const gfId = event["detail"]["GeofenceId"];
                  const resources = event["resources"][0];
                  const splitResources = resources.split(".");
                  const geofenceCollection = splitResources[splitResources.length -
1];
                  const coordinates = event["detail"]["Position"];

                  const deviceId = event["detail"]["DeviceId"];
                  console.log("deviceId===>>>", deviceId);
                  const msg =  {
                      "trackerEventType" : trackerEvent,
                      "source" : src,
                      "eventTime" : time,
                      "geofenceId" : gfId,
                      "coordinates": coordinates,
                      "geofenceCollection": geofenceCollection
                    };
                  const params = {
                    topic: `${deviceId}/tracker`,
                    payload: JSON.stringify(msg),
                    qos: 0
                  };
```

```
                    iotdata.publish(params, function(err, data) {
                        if (err) {
                          console.log("error===>>>", err, err.stack); // an error
occurred
                        } else {
                          console.log("Ladmbda triggered===>>>", trackerEvent);  //
successful response
                        }
                    });
                }
              });
            }
```

8. Deploy를 눌러 업데이트된 함수를 저장합니다.
9. 다음으로 구성 탭을 엽니다.
10. 트리거 섹션에서 트리거 추가 버튼을 누릅니다.
11. 소스 필드에서 EventBridge (CloudWatch 이벤트) 를 선택합니다.
12. 기존 규칙 옵션을 선택합니다.
13. 규칙 이름을 입력합니다 (예:)`AmazonLocationMonitor-GEOFENCECOLLECTION_NAME`.
14. 추가 버튼을 누릅니다.
15. 이렇게 하면 권한 탭의 리소스 기반 정책 설명도 첨부됩니다.

이제 다음 절차를 사용하여 AWS IoT MQTT 테스트 클라이언트를 설정해 보겠습니다.

1. https://console.aws.amazon.com/iot/ 을 엽니다.
2. 왼쪽 탐색 창에서 MQTT 테스트 클라이언트를 선택합니다.
3. MQTT 연결을 구성할 수 있는 MQTT 테스트 클라이언트라는 제목의 섹션이 표시됩니다.
4. 필요한 설정을 구성한 후 Connect 버튼을 클릭하여 제공된 매개변수를 사용하여 MQTT 브로커에 대한 연결을 설정합니다.
5. 엔드포인트를 기록하십시오. 이 엔드포인트는 자습서 뒷부분에서 사용됩니다.

   테스트 클라이언트에 연결되면 MQTT 테스트 클라이언트 인터페이스에 제공된 각 입력 필드를 사용하여 MQTT 주제를 구독하거나 주제에 메시지를 게시할 수 있습니다. 다음으로 정책을 생성해 보겠습니다. AWS IoT
6. 왼쪽 메뉴의 관리에서 보안 옵션을 펼치고 정책을 클릭합니다.
7. 정책 생성 버튼을 클릭합니다.

8. 정책 이름을 입력합니다.
9. 정책 문서에서 JSON 탭을 선택합니다.
10. 아래 표시된 정책을 복사하여 붙여넣으세요. 단, 모든 요소는 반드시 다음과 같이 업데이트해야 합니다*REGION*. *ACCOUNT_ID*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:REGION:ACCOUNT_ID:client/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topicfilter/${cognito-
identity.amazonaws.com:sub}/*",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-identity.amazonaws.com:sub}/
tracker"
      ],
      "Effect": "Allow"
    }
  ]
}
```

11. Create 버튼을 선택하여 완료하십시오.

이전 절차를 완료했으면 이제 게스트 역할에 대한 권한을 다음과 같이 업데이트합니다.

1. Amazon Cognito로 이동하여 자격 증명 풀을 엽니다. 그런 다음 사용자 액세스로 이동하여 게스트 역할을 선택합니다.
2. 권한 정책을 클릭하여 편집을 활성화합니다.

```
{
    'Version': '2012-10-17',
    'Statement': [
```

```
        {
            'Action': [
                'geo:GetMap*',
                'geo:BatchUpdateDevicePosition',
                'geo:BatchEvaluateGeofences',
                'iot:Subscribe',
                'iot:Publish',
                'iot:Connect',
                'iot:Receive',
                'iot:AttachPrincipalPolicy',
                'iot:AttachPolicy',
                'iot:DetachPrincipalPolicy',
                'iot:DetachPolicy'
            ],
            'Resource': [
                'arn:aws:geo:us-east-1:{USER_ID}:map/{MAP_NAME}',
                'arn:aws:geo:us-east-1:{USER_ID}:tracker/{TRACKER_NAME}',
                'arn:aws:geo:us-east-1:{USER_ID}:geofence-collection/
{GEOFENCE_COLLECTION_NAME}',
                'arn:aws:iot:us-east-1:{USER_ID}:client/${cognito-
identity.amazonaws.com:sub}',
                'arn:aws:iot:us-east-1:{USER_ID}:topic/${cognito-
identity.amazonaws.com:sub}',
                'arn:aws:iot:us-east-1:{USER_ID}:topicfilter/${cognito-
identity.amazonaws.com:sub}/*',
                'arn:aws:iot:us-east-1:{USER_ID}:topic/${cognito-
identity.amazonaws.com:sub}/tracker'
            ],
            'Effect': 'Allow'
        },
        {
            'Condition': {
                'StringEquals': {
                    'cognito-identity.amazonaws.com:sub': '${cognito-
identity.amazonaws.com:sub}'
                }
            },
            'Action': [
                'iot:AttachPolicy',
                'iot:DetachPolicy',
                'iot:AttachPrincipalPolicy',
                'iot:DetachPrincipalPolicy'
            ],
            'Resource': [
```

```
                '*'
            ],
            'Effect': 'Allow'
        }
    ]
}
```

3. 위의 정책 변경으로 이제 필요한 모든 AWS 리소스가 애플리케이션에 맞게 구성되었습니다.

샘플 앱 코드를 설정합니다.

1. https://github.com/aws-geospatial/ amazon-location-samples-android /tree/main/ 리포지토리를 로컬 컴퓨터에 tracking-with-geofence-notifications 복제합니다.
2. Android 스튜디오에서 AmazonSampleSDKApp 프로젝트를 엽니다.
3. Android 기기 또는 에뮬레이터에서 앱을 빌드하고 실행합니다.

샘플 앱 사용

샘플을 사용하려면 다음 절차를 따르십시오.

- custom.properties다음을 생성하십시오.

  custom.properties파일을 구성하려면 다음 단계를 따르세요.

  1. 원하는 텍스트 편집기 또는 IDE를 엽니다.
  2. 새 파일을 만듭니다.
  3. custom.properties 이름으로 파일을 저장합니다.
  4. 를 다음 코드 custom.properties 샘플로 업데이트하고,MQTT_END_POINT, POLICY_NAMEGEOFENCE_COLLECTION_NAME, 를 리소스 TOPIC_TRACKER 이름으로 바꾸십시오.

     ```
     MQTT_END_POINT=YOUR_END_POINT.us-east-1.amazonaws.com
     POLICY_NAME=YOUR_POLICY
     GEOFENCE_COLLECTION_NAME=YOUR_GEOFENCE
     TOPIC_TRACKER=YOUR_TRACKER
     ```

  5. 프로젝트를 정리하고 다시 빌드하세요. 그런 다음 프로젝트를 실행할 수 있습니다.
- 로그인:

애플리케이션에 로그인하려면 아래 단계를 따르세요.

1. 로그인 버튼을 누릅니다.
2. ID 풀 ID, 트래커 이름, 맵 이름을 입력합니다.
3. 로그인을 다시 눌러 완료하세요.

- 필터 관리:

  구성 화면을 열고 다음을 수행합니다.

  1. 스위치 UI를 사용하여 필터를 켜거나 끕니다.
  2. 필요한 경우 시간 및 거리 필터를 업데이트하세요.

- 추적 작업:

  추적 화면을 열고 다음을 수행합니다.

  - 각 버튼을 눌러 포그라운드, 백그라운드 또는 배터리 절약 모드에서 추적을 시작하고 중지할 수 있습니다.

## iOS용 샘플 트래킹 및 지오펜싱 애플리케이션

이 주제에서는 모바일 애플리케이션에서 Amazon Location 지오펜스 및 트래커를 사용하는 주요 기능을 시연하기 위해 설계된 iOS 자습서를 다룹니다. 애플리케이션은 Lambda와 Amazon Location 기능의 조합을 사용하여 트래커와 geonfence가 상호 작용하는 방식을 보여줍니다. AWS IoT

주제

- 앱을 위한 Amazon 위치 리소스 생성
- 지오펜스 컬렉션 만들기
- 트래커를 지오펜스 컬렉션에 연결
- AWS Lambda를 MQTT와 함께 사용하기
- 샘플 앱 코드 설정
- 샘플 앱 사용

### 앱을 위한 Amazon 위치 리소스 생성

시작하려면 필요한 Amazon 위치 리소스를 생성해야 합니다. 이러한 리소스는 애플리케이션의 기능과 제공된 코드 스니펫을 실행하는 데 필수적입니다.

> **Note**
>
> 계정을 만들지 않았다면 AWS AWS 계정 관리 사용 설명서의 지침을 따르세요.

시작하려면 다음 절차를 사용하여 Amazon Cognito 자격 증명 풀 ID를 생성해야 합니다.

1. Amazon Cognito 콘솔을 열고 왼쪽 메뉴에서 자격 증명 풀을 선택한 다음 자격 증명 풀 생성을 선택합니다.
2. 게스트 액세스가 선택되어 있는지 확인하고 다음을 눌러 계속하십시오.
3. 다음으로 새 IAM 역할을 만들거나 기존 IAM 역할을 사용합니다.
4. 자격 증명 풀 이름을 입력하고 자격 증명 풀이 다음 절차에서 생성할 맵 및 추적기의 Amazon Location (`geo`) 리소스에 액세스할 수 있는지 확인합니다.

다음으로 AWS Amazon Location 콘솔에서 맵을 생성하고 스타일을 지정해야 합니다. 다음 절차를 사용하십시오.

1. Amazon 위치 콘솔의 맵 섹션으로 이동하여 맵 생성을 선택합니다.
2. 새 맵에 이름과 설명을 입력합니다. 튜토리얼 뒷부분에서 사용할 수 있도록 지정한 이름을 기록해 두십시오.
3. 맵 스타일을 선택할 때는 맵 데이터 공급자를 고려해 보세요. 자세한 내용은 AWS 서비스 약관의 섹션 82을 참조하십시오.
4. Amazon 위치 이용 약관에 동의한 다음 맵 생성을 선택하여 맵 생성 프로세스를 완료합니다.

다음으로 Amazon Location 콘솔에서 트래커를 생성해야 합니다. 다음 절차를 사용하십시오.

1. Amazon 위치 콘솔에서 지도 섹션을 엽니다.
2. 트래커 생성을 선택합니다.
3. 필수 필드를 입력합니다. 트래커의 이름은 이 튜토리얼 전체에서 참조되므로 기록해 두십시오.
4. 위치 필터링 필드에서 트래커 리소스를 사용하려는 방식에 가장 적합한 옵션을 선택합니다. 포지션 필터링을 설정하지 않은 경우 기본 설정은 입니다`TimeBased`. 자세한 내용은 Amazon 위치 API PositionFiltering참조에서 추적 시작을 참조하십시오.
5. 트래커 생성을 선택하여 트래커 생성을 완료하십시오.

지오펜스 컬렉션 만들기

이제 지오펜스 컬렉션을 만들어 볼까요? 콘솔, API 또는 CLI를 사용할 수 있습니다. 다음 절차는 각 옵션을 안내합니다.

- Amazon 로케이션 콘솔을 사용하여 지오펜스 컬렉션을 생성합니다.

  1. Amazon 로케이션 콘솔의 지오펜스 컬렉션 섹션을 엽니다.
  2. 지오펜스 컬렉션 생성을 선택합니다.
  3. 컬렉션의 이름과 설명을 입력합니다.
  4. 대상으로 사용하는 EventBridge 규칙에서 선택적 EventBridge 규칙을 생성하여 Amazon CloudWatch 지오펜스 이벤트에 대한 대응을 시작할 수 있습니다. 이렇게 하면 Amazon Location에서 이벤트를 게시할 수 Amazon CloudWatch Logs있습니다.
  5. 지오펜스 컬렉션 생성을 눌러 컬렉션 생성을 완료합니다.

- Amazon 로케이션 API를 사용하여 지오펜스 컬렉션을 생성하십시오.

  Amazon 위치 지오펜스 API에서 CreateGeofenceCollection작업을 사용하십시오. 다음 예시에서는 API 요청을 사용하여 라는 지오펜스 컬렉션을 생성합니다. *GEOCOLLECTION_NAME*

```
POST /geofencing/v0/collections
Content-type: application/json

{
   "CollectionName": "GEOCOLLECTION_NAME",
   "Description": "Geofence collection 1 for shopping center",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

- CLI AWS 명령을 사용하여 지오펜스 컬렉션을 생성합니다.

  `create-geofence-collection` 명령을 사용합니다. 다음 예제에서는 AWS CLI를 사용하여 라는 지오펜스 컬렉션을 생성합니다. *GEOCOLLECTION_NAME* AWS CLI 사용에 대한 자세한 내용은 AWS 명령줄 인터페이스 설명서를 참조하십시오.

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
```

```
  --tags Tag1=Value1
```

트래커를 지오펜스 컬렉션에 연결

트래커를 지오펜스 컬렉션에 연결하려면 콘솔, API 또는 CLI를 사용할 수 있습니다. 다음 절차는 각 옵션을 안내합니다.

Amazon Location Service 콘솔을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결합니다.

1. Amazon 로케이션 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 트래커를 선택합니다.
3. 디바이스 트래커에서 대상 트래커의 이름 링크를 선택합니다.
4. 연결된 지오펜스 컬렉션에서 지오펜스 컬렉션 연결을 선택합니다.
5. 연결된 지오펜스 컬렉션 창의 드롭다운 메뉴에서 지오펜스 컬렉션을 선택합니다.
6. 연결을 선택합니다.
7. 트래커 리소스를 연결하면 해당 리소스에 활성 상태가 할당됩니다.

Amazon Location API를 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결합니다.

Amazon Location Trackers API에서 `AsssociateTrackerConsumer` 작업을 사용합니다. 다음 예시에서는 Amazon 리소스 이름 (ARN) 을 사용하여 지오펜스 컬렉션과 연결하는 ExampleTracker API 요청을 사용합니다.

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json
    {
       "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GEOCOLLECTION_NAME"
    }
```

AWS CLI 명령을 사용하여 트래커 리소스를 지오펜스 컬렉션에 연결합니다.

`associate-tracker-consumer` 명령을 사용합니다. 다음 예제는 AWS CLI를 사용하여 라는 지오펜스 컬렉션을 생성합니다. *GEOCOLLECTION_NAME*

```
aws location \
```

```
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GEOCOLLECTION_NAME" \
        --tracker-name "ExampleTracker"
```

AWS Lambda를 MQTT와 함께 사용하기

Amazon Location 간에 AWS IoT 연결을 생성하려면 이벤트에 의해 전달된 메시지를 처리하는 Lambda 함수가 필요합니다. EventBridge CloudWatch 이 함수는 모든 위치 데이터를 추출하여 Amazon 위치에 맞게 형식을 지정한 다음 Amazon 위치 추적기 API를 통해 제출합니다.

다음 절차는 Lambda 콘솔을 통해 이 함수를 생성하는 방법을 보여줍니다.

1. 콘솔을 엽니다.
2. 왼쪽 탐색에서 함수를 선택합니다.
3. 그런 다음 [함수 생성] 을 선택하고 [처음부터 작성] 옵션이 선택되었는지 확인합니다.
4. 함수 이름을 제공하고 [런타임] 옵션으로 Node.js 16.x를 선택합니다.
5. 함수 생성을 선택합니다.
6. 코드 탭을 열어 편집기에 액세스합니다.
7. `index.js`파일의 플레이스홀더 코드를 다음과 같이 덮어씁니다.

```
const AWS = require('aws-sdk')
const iot = new AWS.Iot();
exports.handler =  function(event) {
  console.log("event===>>>", JSON.stringify(event));
  var param = {
    endpointType: "iot:Data-ATS"
  };
  iot.describeEndpoint(param, function(err, data) {
    if (err) {
      console.log("error===>>>", err, err.stack); // an error occurred
    } else {
      var endp = data['endpointAddress'];
      const iotdata = new AWS.IotData({endpoint: endp});
      const trackerEvent = event["detail"]["EventType"];
      const src = event["source"];
      const time = event["time"];
      const gfId = event["detail"]["GeofenceId"];
      const resources = event["resources"][0];
```

```
                const splitResources = resources.split(".");
                const geofenceCollection = splitResources[splitResources.length -
1];
                const coordinates = event["detail"]["Position"];

                const deviceId = event["detail"]["DeviceId"];
                console.log("deviceId===>>>", deviceId);
                const msg =  {
                    "trackerEventType" : trackerEvent,
                    "source" : src,
                    "eventTime" : time,
                    "geofenceId" : gfId,
                    "coordinates": coordinates,
                    "geofenceCollection": geofenceCollection
                  };
                const params = {
                  topic: `${deviceId}/tracker`,
                  payload: JSON.stringify(msg),
                  qos: 0
                };
                iotdata.publish(params, function(err, data) {
                    if (err) {
                      console.log("error===>>>", err, err.stack); // an error
occurred
                    } else {
                      console.log("Ladmbda triggered===>>>", trackerEvent);  //
successful response
                    }
                  });
              }
            });
          }
```

8. Deploy를 눌러 업데이트된 함수를 저장합니다.
9. 다음으로 구성 탭을 엽니다.
10. 트리거 섹션에서 트리거 추가 버튼을 누릅니다.
11. 소스 필드에서 EventBridge (CloudWatch 이벤트) 를 선택합니다.
12. 기존 규칙 옵션을 선택합니다.
13. 규칙 이름을 입력합니다 (예:)AmazonLocationMonitor-GEOFENCECOLLECTION_NAME.
14. 추가 버튼을 누릅니다.
15. 이렇게 하면 권한 탭의 리소스 기반 정책 설명도 첨부됩니다.

이제 AWS IoT MQTT 테스트 클라이언트를 설정하려면 다음 절차를 사용하십시오.

1. https://console.aws.amazon.com/iot/ 을 엽니다.
2. 왼쪽 탐색 창에서 MQTT 테스트 클라이언트를 선택합니다.
3. MQTT 연결을 구성할 수 있는 MQTT 테스트 클라이언트라는 제목의 섹션이 표시됩니다.
4. 필요한 설정을 구성한 후 Connect 버튼을 클릭하여 제공된 매개변수를 사용하여 MQTT 브로커에 대한 연결을 설정합니다.
5. 엔드포인트를 기록하십시오. 이 엔드포인트는 자습서 뒷부분에서 사용됩니다.

   테스트 클라이언트에 연결되면 MQTT 테스트 클라이언트 인터페이스에 제공된 각 입력 필드를 사용하여 MQTT 주제를 구독하거나 주제에 메시지를 게시할 수 있습니다. 다음으로 정책을 생성해 보겠습니다. AWS IoT
6. 왼쪽 메뉴의 관리에서 보안 옵션을 펼치고 정책을 클릭합니다.
7. 정책 생성 버튼을 클릭합니다.
8. 정책 이름을 입력합니다.
9. 정책 문서에서 JSON 탭을 선택합니다.
10. 아래 표시된 정책을 복사하여 붙여넣으세요. 단, 모든 요소는 반드시 다음과 같이 업데이트해야 합니다*REGION*. *ACCOUNT_ID*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:REGION:ACCOUNT_ID:client/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topicfilter/${cognito-
identity.amazonaws.com:sub}/*",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-identity.amazonaws.com:sub}/
tracker"
```

```
      ],
      "Effect": "Allow"
    }
  ]
}
```

11. Create 버튼을 선택하여 완료하십시오.

### 샘플 앱 코드 설정

샘플 코드를 설정하려면 다음 도구가 설치되어 있어야 합니다.

- Git
- XCode 15.3 이상
- iOS 시뮬레이터 16 이상

다음 절차를 사용하여 샘플 앱 코드를 설정하십시오.

1. 다음 URL (https://github.com/aws-geospatial/ amazon-location-samples-ios tracking-with-geofence-notifications /tree/main/) 에서 git 리포지토리를 복제합니다.
2. `AWSLocationSampleApp.xcodeproj` 프로젝트 파일을 엽니다.
3. 패키지 확인 프로세스가 완료될 때까지 기다리세요.
4. 프로젝트 탐색 메뉴에서 이름을 `ConfigTemplate.xcconfig` `Config.xcconfig` 바꾸고 다음 값을 입력합니다.

```
IDENTITY_POOL_ID = `YOUR_IDENTITY_POOL_ID`
MAP_NAME = `YOUR_MAP_NAME`
TRACKER_NAME = `YOUR_TRACKER_NAME`
WEBSOCKET_URL = `YOUR_MQTT_TEST_CLIENT_ENDPOINT`
GEOFENCE_ARN = `YOUR_GEOFENCE_COLLECTION_NAME`
```

### 샘플 앱 사용

샘플 코드를 설정한 후 이제 iOS 시뮬레이터 또는 물리적 기기에서 앱을 실행할 수 있습니다.

1. 앱을 빌드하고 실행합니다.
2. 앱에서 위치 및 알림 권한을 요청합니다. 허용해야 합니다.

3. Cognito 구성 버튼을 누릅니다.
4. 구성을 저장합니다.
5. 이제 시간, 거리, 정확도에 대한 필터 옵션을 볼 수 있습니다. 필요에 따라 사용하세요.
6. 앱의 추적 탭으로 이동하면 지도와 추적 시작 버튼이 표시됩니다.
7. 시뮬레이터에 앱을 설치한 경우 위치 변경을 시뮬레이션하는 것이 좋습니다. 이 작업은 위치 메뉴 옵션의 기능에서 수행할 수 있습니다. 예를 들어 기능, 위치, 프리웨이 드라이브를 차례로 선택합니다.
8. 추적 시작 버튼을 누릅니다. 지도에 트래킹 포인트가 보일 것입니다.
9. 앱은 백그라운드에서 위치도 추적합니다. 따라서 앱을 백그라운드로 이동하면 백그라운드 모드에서 추적을 계속할 수 있도록 권한을 요청합니다.
10. 추적 중지 버튼을 눌러 추적을 중지할 수 있습니다.

# Amazon Location Service 리소스 태그 지정

Amazon Location에서 리소스 태그 지정 기능을 사용하여 목적, 소유자, 환경 또는 기준에 따라 리소스를 분류할 수 있습니다. 리소스에 태그를 지정하면 리소스를 관리, 식별, 정리, 검색 및 필터링하는 데 도움이 됩니다.

에를 들어 AWS Resource Groups을 통해 하나 이상의 태그 또는 태그의 일부를 기반으로 AWS 리소스 그룹을 만들 수 있습니다. AWS CloudFormation 스택에 리소스가 있는지 여부에 따라 그룹을 만들 수도 있습니다. 리소스 그룹 및 Tag Editor를 사용하면 여러 서비스, 리소스 및 리전으로 구성된 애플리케이션의 데이터를 한 곳에 통합하여 볼 수 있습니다. 일반적인 태그 지정 전략에 대한 자세한 내용은 AWS 일반 참조를 참조하세요.

각 태그는 사용자가 정의하는 키와 값으로 구성되는 레이블입니다.

- 태그 키 – 태그 값을 분류하는 일반적인 레이블입니다. 예를 들어 `CostCenter`입니다.
- 태그 값 – 태그 키 카테고리에 대한 선택적 설명입니다. 예를 들어 `MobileAssetTrackingResourcesProd`입니다.

이 주제는 태그 지정 제한을 검토하여 태그 지정을 시작할 수 있도록 도와줍니다. 또한 비용 할당 보고서를 사용하여 태그를 생성하고 태그를 사용하여 각 활성 태그의 AWS 비용을 추적하는 방법도 보여줍니다.

주제

## 태그 지정 제한

태그에 적용되는 기본 제한은 다음과 같습니다.

- 리소스당 최대 태그 – 50개
- 각 리소스에 대해 각 태그 키는 고유하며 하나의 값만 가질 수 있습니다.

> **Note**
>
> 태그 키가 기존 태그와 동일한 새 태그를 추가하는 경우 새 태그가 기존 태그를 덮어씁니다.

- 최대 키 길이 - UTF-8 형식의 유니코드 문자 128자
- 최대 값 길이 - UTF-8 형식의 유니코드 문자 256자
- 서비스에서 허용되는 문자는 UTF-8로 표현할 수 있는 문자, 숫자 및 공백과 특수 문자 + - = . _ : / @ 입니다.
- 태그 키와 값은 대/소문자를 구분합니다.
- `aws:` 접두사는 AWS용으로 예약되어 있습니다. 태그에 이 접두사가 있는 태그 키가 있는 경우 태그의 키 또는 값을 편집하거나 삭제할 수 없습니다. `aws:` 접두사가 지정된 태그는 리소스당 태그 수 제한에 포함되지 않습니다.

## 리소스에 태그를 지정할 수 있는 권한을 부여합니다.

IAM 정책을 사용하여 Amazon Location 리소스에 대한 액세스를 제어하고 생성 시 리소스에 태그를 지정할 수 있는 권한을 부여할 수 있습니다. 리소스 생성 권한을 부여하는 것 외에도 정책에는 태그 지정 작업을 허용하는 `Action` 권한이 포함될 수 있습니다.

- `geo:TagResource` – 사용자가 지정된 Amazon Location 리소스에 하나 이상의 태그를 할당할 수 있습니다.

- `geo:UntagResource` – 사용자가 지정된 Amazon Location 리소스에서 하나 이상의 태그를 제거할 수 있습니다.
- `geo:ListTagsForResource` – 사용자가 Amazon Location 리소스에 할당된 모든 태그를 나열할 수 있습니다.

다음은 사용자가 지오펜스 컬렉션을 생성하고 리소스에 태그를 지정할 수 있도록 허용하는 정책 예시입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowTaggingForGeofenceCollectionOnCreation",
            "Effect": "Allow",
            "Action": [
                "geo:CreateGeofenceCollection",
                "geo:TagResource"
            ],
            "Resource": "arn:aws:geo:region:accountID:geofence-collection/*"
    ]
}
```

## Amazon Location Service 리소스에 태그 추가

Amazon Location 콘솔, AWS CLI 또는 Amazon Location API를 사용하여 리소스를 생성할 때 태그를 추가할 수 있습니다.

- 맵 리소스를 생성합니다.
- 장소 색인 리소스 만들기
- 경로 계산기 리소스 생성
- 지오펜스 컬렉션 생성
- 트래커 리소스 생성

기존 리소스에 태그를 지정하려면 태그를 편집하거나 삭제하세요.

1. https://console.aws.amazon.com/location/에서 Amazon Location 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 태그를 지정하려는 리소스를 선택합니다. 예를 들어 맵이 있습니다.

3. 목록에서 리소스를 선택합니다.
4. 태그 관리를 선택하여 태그를 추가, 편집 또는 삭제합니다.

## 태그별로 리소스 비용 추적하기

비용 할당용 태그를 사용하여 AWS 비용을 자세히 추적할 수 있습니다. 비용 할당 태그를 활성화하고 나면 AWS가 비용 할당 태그를 사용하여 비용 할당 보고서에서 리소스 청구를 구성합니다. 이를 통해 사용 비용을 분류하고 추적할 수 있습니다.

활성화할 수 있는 비용 할당 태그에는 다음 두 가지 유형이 있습니다.

- AWS 생성 - 이 태그는 AWS에서 생성합니다. AWS태그에는 `aws:` 접두사(예:`aws:createdBy`)를 사용합니다.
- 사용자 정의 - 직접 만든 사용자 지정 태그입니다. 사용자 정의 태그에는 `user:` 접두사(예: `user:CostCenter`)를 사용합니다.

각 태그 유형을 개별적으로 활성화해야 합니다. 태그를 활성화하면 월별 비용 할당 보고서를 AWS Cost Explorer 활성화하거나 볼 수 있습니다.

AWS-generated tags

AWS에서 생성된 태그 활성화 방법

1. https://console.aws.amazon.com/billing/에서 Billing and Cost Management 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 비용 할당 태그를 선택합니다.
3. AWS에서 생성된 비용 할당 태그 탭에서 활성화할 태그 키를 선택합니다.
4. 활성화를 선택합니다.

User-defined tags

사용자 정의된 태그 활성화 방법

1. https://console.aws.amazon.com/billing/에서 Billing and Cost Management 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 비용 할당 태그를 선택합니다.
3. 사용자 정의 비용 할당 태그 탭에서 활성화하려는 태그 키를 선택합니다.
4. 활성화를 선택합니다.

태그를 활성화하면 AWS가 리소스 사용량 및 비용에 대한 월별 비용 할당 보고서를 생성합니다. 이 비용 할당 보고서에는 태그가 지정된 리소스와 태그가 없는 리소스를 포함하여 각 결제 기간의 모든 AWS 비용이 포함되어 있습니다. 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 비용 할당 태그 사용을 참조하세요.

### 태그를 사용하여 Amazon Location Service 리소스에 대한 액세스 제어

AWS Identity and Access Management(IAM) 정책은 태그 기반 조건을 지원하므로 특정 태그 키 및 값에 따라 리소스에 대한 권한을 관리할 수 있습니다. 예를 들어 IAM 역할 정책에는 태그를 기반으로 개발, 테스트, 프로덕션 등 특정 환경에 대한 액세스를 제한하는 조건이 포함될 수 있습니다.

자세한 내용은 태그 기반 리소스 액세스 제어 항목을 참조하세요.

### 자세히 알아보기

에 대한 자세한 내용:

- 태그 지정 모범 사례는 AWS 일반 참조의 AWS 리소스 태그 지정을 참조하세요.
- 태그를 사용하여 AWS 리소스에 대한 액세스는 AWS Identity and Access Management 사용 설명서의 태그를 사용하여 AWS 리소스에 대한 액세스 제어를 참조하세요.

## Amazon Location Service에 액세스 권한 부여

Amazon Location Service를 사용하려면 사용자는 Amazon Location을 구성하는 리소스 및 API에 대한 액세스 권한을 받아야 합니다. 리소스에 대한 액세스를 허용하는 데 사용할 수 있는 세 가지 전략이 있습니다.

- IAM 사용 – AWS IAM Identity Center 또는 AWS Identity and Access Management(IAM)로 인증된 사용자에게 액세스 권한을 부여하려면 원하는 리소스에 대한 액세스를 허용하는 IAM 정책을 생성합니다. IAM 및 Amazon Location Service에 대한 자세한 정보는 Amazon Location Service를 위한 ID 및 액세스 관리 항목을 참조하세요.
- API 키 사용 – 인증되지 않은 사용자에게 액세스 권한을 부여하려면 Amazon Location Service 리소스에 대한 읽기 전용 액세스 권한을 부여하는 API 키를 생성할 수 있습니다. 이는 모든 사용자를 인증하고 싶지 않은 경우에 유용합니다. 웹 애플리케이션을 예로 들 수 있습니다. API 키에 대한 자세한 내용은 API 키를 사용하여 인증되지 않은 게스트의 애플리케이션 액세스 허용하기 단원을 참조하세요.
- Amazon Cognito 사용 – API 키의 대안은 Amazon Cognito를 사용하여 익명 액세스를 허용하는 것입니다. Amazon Cognito를 사용하면 인증되지 않은 사용자가 수행할 수 있는 작업을 정의하는 정책을

통해 보다 다양한 권한을 부여할 수 있습니다. Amazon Cognito 사용에 대한 자세한 내용은 Amazon Cognito를 사용하여 미인증 게스트의 애플리케이션 액세스 허용하기을 참조하십시오.

> **Note**
>
> 또한 Amazon Cognito를 사용하여 자체 인증 프로세스를 사용하거나 Amazon Cognito 페더레이션 ID를 사용하여 여러 인증 방법을 결합할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 페더레이션 ID 시작하기를 참조하세요.

주제

- API 키를 사용하여 인증되지 않은 게스트의 애플리케이션 액세스 허용하기
- Amazon Cognito를 사용하여 미인증 게스트의 애플리케이션 액세스 허용하기

## API 키를 사용하여 인증되지 않은 게스트의 애플리케이션 액세스 허용하기

애플리케이션에서 Amazon Location Service API를 호출할 때에는 일반적으로 API 호출 권한이 있는 인증된 사용자로서 이 호출을 수행합니다. 그러나 경우에 따라 애플리케이션의 모든 사용자를 인증하고 싶지는 않습니다. 예를 들어, 웹 사이트를 사용하는 모든 사람이 로그인 여부에 관계없이 비즈니스 위치를 보여주는 웹 애플리케이션을 사용할 수 있도록 하고 싶을 수 있습니다. 이 경우 한 가지 대안은 API 키를 사용하여 API를 호출하는 것입니다.

API 키는 사용자의 AWS 계정특정 Amazon Location Service 리소스 및 해당 리소스에서 수행할 수 있는 특정 작업과 관련된 키 값입니다. 애플리케이션의 API 키를 사용하여 해당 리소스에 대해 Amazon Location API를 인증되지 않은 상태로 호출할 수 있습니다. 예를 들어, API 키를 맵 리소스인 MyMap 및 `GetMap*` 작업에 연결하는 경우, 해당 API 키를 사용하는 애플리케이션은 해당 리소스로 생성된 맵을 볼 수 있게 되며 계정의 다른 사용량과 마찬가지로 사용자 계정에 요금이 부과됩니다. 동일한 API 키는 맵 리소스를 변경하거나 업데이트할 권한을 부여하지 않으며 해당 리소스만 사용할 수 있습니다.

> **Note**
>
> API 키는 지도, 장소, 경로 리소스에만 사용할 수 있으며 이러한 리소스를 수정하거나 생성할 수 없습니다. 애플리케이션에서 인증되지 않은 사용자를 위해 다른 리소스 또는 작업에 액세스해야 하는 경우 Amazon Cognito를 사용하여 API 키와 함께 또는 API 키 대신 액세스를 제공할 수 있습니다. 자세한 설명은 Amazon Cognito를 사용하여 미인증 게스트의 애플리케이션 액세스 허용하기 섹션을 참조하세요.

API 키에는 하나 이상의 사용자 AWS 계정리소스에 대한 액세스를 제공하는 일반 텍스트 값이 포함됩니다. 누군가 API 키를 복사해도 동일한 리소스에 액세스할 수 있습니다. 이를 방지하려면 키를 생성할 때 API 키를 사용할 수 있는 도메인을 지정할 수 있습니다. 이러한 도메인을 리퍼러라고 합니다. 필요한 경우 API 키의 만료 시간을 설정하여 단기 API 키를 생성할 수도 있습니다.

주제



## Amazon Cognito와 비교한 API 키

API 키와 Amazon Cognito는 유사한 시나리오에서 비슷한 방식으로 사용되는데, 둘 중 하나를 다른 것보다 더 사용하는 이유는 무엇입니까? 다음 표는 이 둘 사이의 차이점 중 일부를 강조합니다.

- API 키는 지도, 장소 및 경로 리소스와 특정 작업에만 사용할 수 있습니다. Amazon Cognito는 대부분의 Amazon Location Service API에 대한 액세스를 인증하는 데 사용할 수 있습니다.
- API 키를 사용한 맵 요청의 성능은 일반적으로 Amazon Cognito를 사용하는 유사한 시나리오보다 빠릅니다. 인증이 간단하면 짧은 시간 내에 동일한 맵 타일을 다시 가져올 때 서비스 및 캐시된 요청으로의 왕복 횟수가 줄어듭니다.
- Amazon Cognito를 사용하면 Amazon Cognito의 페더레이션 ID를 사용하여 자체 인증 프로세스를 사용하거나 여러 인증 방법을 결합할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 페더레이션 ID 시작하기를 참조하세요.

## API 키 만들기

API 키를 생성하여 하나 이상의 사용자 AWS 계정리소스에 연결할 수 있습니다.

Amazon 위치 서비스 콘솔 AWS CLI, 또는 Amazon 위치 API를 사용하여 API 키를 생성할 수 있습니다.

### Console

Amazon Location Service 콘솔을 사용하여 API 키를 생성하려면

1. Amazon Location 콘솔의 왼쪽 메뉴에서 API 키를 선택합니다.
2. API 키 페이지에서 API 키 생성을 선택합니다.
3. API 키 생성 페이지에서 다음 정보를 입력합니다.
   - 이름 – API 키의 이름(예:`MyWebAppKey`).
   - 설명 – API 키에 대한 선택적 설명.
   - 리소스 – 드롭다운에서 이 API 키로 액세스 권한을 부여할 Amazon Location 리소스를 선택합니다. 리소스 추가를 선택하여 두 개 이상의 리소스를 추가할 수 있습니다.
   - 작업 – 이 API 키로 승인하려는 작업을 지정합니다. 선택한 각 리소스 유형과 일치하는 작업을 하나 이상 선택해야 합니다. 예를 들어, 장소 리소스를 선택한 경우 장소 작업에 있는 선택 항목 중 하나 이상을 선택해야 합니다.
   - 만료 시간 – 선택적으로 API 키의 만료 날짜 및 시간을 추가할 수 있습니다. 자세한 설명은 API 키 수명 관리 섹션을 참조하세요.
   - 리퍼러 – 선택적으로 API 키를 사용할 수 있는 도메인을 하나 이상 추가할 수 있습니다. 예를 들어`example.com`, 웹 사이트에서 실행되는 애플리케이션을 허용하는 것이 API 키인 경우 `*.example.com/`를 허용된 리퍼러로 지정할 수 있습니다.
   - 태그 – 선택적으로 API 키에 태그를 추가할 수 있습니다.
4. API 키 생성을 선택하여 API 키를 생성합니다.
5. API 키의 세부 정보 페이지에서, 생성한 API 키에 대한 정보를 볼 수 있습니다. Amazon 위치 API를 호출할 때 사용하는 키 값을 보려면 API 키 표시를 선택합니다. 키 값의 형식은 `v1.public.`*`a1b2c3d4...`*과 같습니다. API 키를 사용하여 맵을 렌더링하는 방법에 대한 자세한 내용은 API 키를 사용하여 맵 렌더링을 참조하세요.

### API

Amazon Location API를 사용하여 API 키를 생성하려면

Amazon Location API에서 `CreateKey` 작업을 사용합니다.

다음은 만료일 없이 *`ExampleKey`*호출된 API 키를 생성하고 단일 맵 리소스에 액세스할 수 있도록 하기 위한 API 요청입니다.

```
POST /metadata/v0/keys HTTP/1.1
```

```
Content-type: application/json

{
   "KeyName": "ExampleKey"
   "Restrictions": {
      "AllowActions": [
         "geo:GetMap*"
      ],
      "AllowResources": [
         "arn:aws:geo:region:map/mapname"
      ]
   },
   "NoExpiry": true
   }
}
```

응답에는 애플리케이션의 리소스에 액세스할 때 사용할 API 키 값이 포함됩니다. 키 값의 형식은 v1.public.*a1b2c3d4...*과 같습니다. API 키를 사용하여 맵을 렌더링하는 방법에 대한 자세한 내용은 API 키를 사용하여 맵 렌더링을 참조하세요.

DescribeKey API를 사용하여 나중에 키의 키 값을 찾을 수도 있습니다.

AWS CLI

AWS CLI 명령을 사용하여 API 키를 만들려면

create-key 명령을 사용합니다.

다음 예시에서는 만료일이 없는 *ExampleKey*호출된 API 키와 단일 맵 리소스에 대한 액세스 권한을 생성합니다.

```
aws location \
  create-key \
  --key-name ExampleKey \
  --restrictions '{"AllowActions":["geo:GetMap*"],"AllowResources":
["arn:aws:geo:region:map/mapname"]}' \
  --no-expiry
```

응답에는 애플리케이션의 리소스에 액세스할 때 사용할 API 키 값이 포함됩니다. 키 값의 형식은 v1.public.*a1b2c3d4...*과 같습니다. API 키를 사용하여 맵을 렌더링하는 방법에 대한 자세한 내용은 API 키를 사용하여 맵 렌더링을 참조하세요. create-key에 대한 응답은 다음과 같습니다.

```
{
```

```
    "Key": "v1.public.a1b2c3d4...",
    "KeyArn": "arn:aws:geo:region:accountId:api-key/ExampleKey",
    "KeyName": "ExampleKey",
    "CreateTime": "2023-02-06T22:33:15.693Z"
}
```

`describe-key`을 사용하여 나중에 키 값을 찾을 수도 있습니다. 다음 예시는 이름이 지정된 API 키를 `describe-key` 호출하는 방법을 보여줍니다 *ExampleKey*.

```
aws location describe-key \
    --key-name ExampleKey
```

## API 키를 사용하여 Amazon Location API를 호출합니다.

API 키를 생성한 후 키 값을 사용하여 애플리케이션의 Amazon Location API를 호출할 수 있습니다.

API 키를 지원하는 API에는 API 키 값을 취하는 추가 파라미터가 있습니다. 예를 들어, `GetPlace` API를 호출하는 경우 다음과 같이 키 파라미터를 입력할 수 있습니다.

```
GET /places/v0/indexes/IndexName/places/PlaceId?key=KeyValue
```

이 값을 입력하면 평소처럼 AWS Sig v4로 API 호출을 인증할 필요가 없습니다.

JavaScript 개발자의 경우 Amazon Location을 사용하여 API 키로 API 작업을 인증하는 JavaScript 인증 도우미 데 도움을 받을 수 있습니다.

모바일 개발자의 경우 다음 Amazon Location 모바일 인증 SDK를 사용할 수 있습니다.

- iOS용 Amazon Location Service 모바일 인증 SDK
- 안드로이드용 Amazon Location Service 모바일 인증 SDK

AWS CLI 사용자의 경우 `--key` 파라미터를 사용할 때 Sig v4로 서명하지 않도록 `--no-sign-request` 파라미터도 사용해야 합니다.

> **Note**
>
> Amazon Location Service에 대한 호출에 `key` 및 AWS Sig v4 서명을 모두 포함하는 경우 API 키만 사용됩니다.

## API 키를 사용하여 맵 렌더링

API 키 값을 사용하여 애플리케이션에서 맵을 렌더링할 수 있습니다. MapLibre 이는 사용자가 직접 호출하는 다른 Amazon Location API의 API 키를 사용하는 것과는 약간 다릅니다. 사용자를 MapLibre 대신하여 호출하기 때문입니다.

다음 샘플 코드는 MapLibre GL JS 맵 컨트롤을 사용하여 API 키를 사용하여 간단한 웹 페이지에 맵을 렌더링하는 방법을 보여줍니다. *# ### ### ##### v1.public# #####. your-api-key-value*, *us-east-1*, 그리고 다음과 일치하는 *ExampleMap*값을 가진 문자열 AWS 계정

```
<!-- index.html -->
<html>
  <head>
    <link href="https://unpkg.com/maplibre-gl@1.14.0/dist/maplibre-gl.css"
 rel="stylesheet" />
    <style>
      body { margin: 0; }
      #map { height: 100vh; }
    </style>
  </head>
  <body>
    <!-- Map container -->
    <div id="map" />
    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@1.14.0/dist/maplibre-gl.js"></script>
    <script>
      const apiKey = "v1.public.your-api-key-value"; // API key
      const region = "us-east-1"; // Region
      const mapName = "ExampleMap"; // Map name
      // URL for style descriptor
      const styleUrl = `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/
${mapName}/style-descriptor?key=${apiKey}`;
      // Initialize the map
      const map = new maplibregl.Map({
        container: "map",
        style: styleUrl,
        center: [-123.1187, 49.2819],
        zoom: 11,
      });
      map.addControl(new maplibregl.NavigationControl(), "top-left");
    </script>
  </body>
```

```
</html>
```

### API 키 수명 관리

무기한으로 작동하는 API 키를 생성할 수 있습니다. 하지만 임시 API 키를 생성하거나 정기적으로 API 키를 교체하거나 기존 API 키를 취소하려는 경우에는 API 키 만료를 사용할 수 있습니다.

새 API 키를 만들거나 기존 API 키를 업데이트할 때 해당 API 키의 만료 시간을 설정할 수 있습니다.

- API 키가 만료 시간에 도달하면 키가 자동으로 비활성화됩니다. 비활성 키는 더 이상 맵 요청에 사용할 수 없습니다.
- API 키를 비활성화한 후 90일이 지나면 삭제할 수 있습니다.
- 아직 삭제하지 않은 비활성 키가 있는 경우 만료 시간을 미래 시간으로 업데이트하여 복원할 수 있습니다.
- 영구 키를 만들려면 만료 시간을 제거하면 됩니다.
- 지난 7일 이내에 사용된 API 키를 비활성화하려고 하면 변경을 확인하라는 메시지가 표시됩니다. Amazon Location Service API 또는 를 사용하는 경우 `ForceUpdate` 파라미터를 true로 설정하지 않으면 오류가 발생합니다. AWS CLI

## Amazon Cognito를 사용하여 미인증 게스트의 애플리케이션 액세스 허용하기

프런트엔드 SDK와 직접 HTTPS 요청 모두에서 직접 사용 AWS Identity and Access Management (IAM) 하는 대신 Amazon Cognito 인증을 사용할 수 있습니다.

다음과 같은 이유로 이 인증 형식을 사용하는 것이 좋습니다.

- 인증되지 않은 사용자 – 익명 사용자가 있는 웹 사이트가 있는 경우 Amazon Cognito 자격 증명 풀을 사용할 수 있습니다. 자세한 내용은 the section called “Amazon Cognito 사용” 섹션을 참조하세요.
- 자체 인증 – 자체 인증 프로세스를 사용하거나 여러 인증 방법을 결합하려는 경우 Amazon Cognito 페더레이션 ID를 사용할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 페더레이션 ID 시작하기를 참조하세요.

Amazon Cognito는 웹 및 모바일 앱에 대한 인증, 권한 부여 및 사용자 관리를 제공합니다. Amazon Location과 함께 Amazon Cognito 비인증 자격 증명 풀을 애플리케이션에서 범위가 축소된 임시 자격 증명을 검색하는 방법으로 사용할 수 있습니다. AWS

자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀 시작하기를 참조하세요.

> **Note**
>
> 모바일 개발자를 위해 Amazon Location은 iOS와 Android용 모바일 인증 SDK를 제공합니다. 자세한 내용은 다음 github 리포지토리를 참조하십시오.
>
> - iOS용 Amazon Location Service 모바일 인증 SDK
> - 안드로이드용 Amazon Location Service 모바일 인증 SDK

## Amazon Cognito 자격 증명 풀 생성

Amazon Cognito 자격 증명 풀을 생성하여 인증되지 않은 게스트가 Amazon Cognito 콘솔, 또는 AWS CLIAmazon Cognito API를 통해 애플리케이션에 액세스하도록 허용할 수 있습니다.

> **Important**
>
> 생성한 풀은 사용 중인 Amazon Location Service AWS 계정 리소스와 동일한 AWS 리전에 있어야 합니다.

인증되지 않은 자격 증명 역할과 관련된 IAM 정책을 다음 작업에 사용할 수 있습니다.

- `geo:GetMap*`
- `geo:SearchPlaceIndex*`
- `geo:GetPlace`
- `geo:CalculateRoute*`
- `geo:GetGeofence`
- `geo:ListGeofences`
- `geo:PutGeofence`
- `geo:BatchDeleteGeofence`
- `geo:BatchPutGeofence`
- `geo:BatchEvaluateGeofences`
- `geo:GetDevicePosition*`
- `geo:ListDevicePositions`

- `geo:BatchDeleteDevicePositionHistory`
- `geo:BatchGetDevicePosition`
- `geo:BatchUpdateDevicePosition`

다른 Amazon Location 작업을 포함해도 효과가 없으며 인증되지 않은 ID는 해당 작업을 호출할 수 없습니다.

Example

Amazon Cognito 콘솔을 사용하여 자격 증명 풀을 만들려면

1. Amazon Cognito 콘솔로 이동합니다.
2. 자격 증명 풀 관리를 선택합니다.
3. 새 자격 증명 풀 생성을 선택한 다음 자격 증명 풀의 이름을 입력합니다.
4. 축소 가능한 인증되지 않은 자격 증명 섹션에서 인증되지 않은 자격 증명에 대한 액세스 활성화를 선택합니다.
5. 풀 생성을 선택합니다.
6. 자격 증명 풀에서 사용하려는 IAM 역할을 선택합니다.
7. 세부 정보 보기를 확장합니다.
8. 인증되지 않은 ID에서 역할 이름을 입력합니다.
9. 정책 문서 보기 섹션을 확장한 다음, 정책을 추가하기 위해 편집을 선택합니다.
10. 정책을 추가하여 리소스에 대한 액세스 권한을 부여합니다.

    다음은 맵, 장소, 트래커 및 경로에 대한 정책 예시입니다. 자체 정책에 대한 예를 사용하려면 *##* 및 *accountID* 플레이스홀더를 교체하세요:

    Maps policy example

    다음 정책은 이라는 *ExampleMap*맵 리소스에 대한 읽기 전용 액세스 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MapsReadOnly",
      "Effect": "Allow",
      "Action": [
```

```
        "geo:GetMapStyleDescriptor",
        "geo:GetMapGlyphs",
        "geo:GetMapSprites",
        "geo:GetMapTile"
      ],
      "Resource": "arn:aws:geo:region:accountID:map/ExampleMap"
    }
  ]
}
```

일치하는 aws:refererIAM 조건을 추가하면 리소스에 대한 브라우저 액세스를 URL 또는 URL 접두사 목록으로 제한할 수 있습니다. 다음 예시에서는 example.com 웹사이트에서 RasterEsriImagery 이름이 지정된 맵 리소스에만 액세스할 수 있도록 허용합니다.

> **Warning**
>
> aws:referer은 액세스를 제한할 수는 있지만 보안 메커니즘은 아닙니다. 공개적으로 알려진 참조자 헤더 값을 포함하는 것은 위험합니다. 권한이 없는 사용자가 수정된 브라우저나 사용자 지정 브라우저를 사용하여 원하는 aws:referer 값을 제공할 수 있습니다. 따라서 권한이 없는 당사자가 직접 AWS 요청하는 것을 방지하는 데 aws:referer 사용해서는 안 됩니다. 이러한 값은 고객이 Amazon S3에 저장된 콘텐츠 등의 디지털 콘텐츠를 권한이 없는 타사 사이트에서 참조하지 못하도록 보호하기 위해서만 사용하세요. 자세한 내용은 AWS:referer를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "geo:GetMap*",
            "Resource": "arn:aws:geo:us-west-2:111122223333:map/
RasterEsriImagery",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
```

```
            }
        }
    ]
}
```

Tangram을 사용하여 맵을 표시하는 경우 Maps API에서 반환되는 스타일 설명자, 글리프 또는 스프라이트는 사용되지 않습니다. 대신 스타일 규칙 및 필수 에셋이 포함된 .zip 파일을 가리키도록 구성됩니다. 다음 정책은 `GetMapTile` 작업을 위해 이름이 지정된 맵 리소스에 *`ExampleMap`*대한 읽기 전용 액세스 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MapsReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:GetMapTile"
      ],
      "Resource": "arn:aws:geo:region:accountID:map/ExampleMap"
    }
  ]
}
```

### Places policy example

다음 정책은 텍스트 또는 위치로 장소를 *`ExamplePlaceIndex`*검색하도록 이름이 지정된 장소 색인 리소스에 대한 읽기 전용 액세스 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PlacesReadOnly",
      "Effect": "Allow",
      "Action": [
          "geo:SearchPlaceIndex*",
          "geo:GetPlace"
      ],
      "Resource": "arn:aws:geo:region:accountID:place-index/ExamplePlaceIndex"
    }
  ]
```

```
}
```

일치하는 aws:refererIAM 조건을 추가하면 리소스에 대한 브라우저 액세스를 URL 또는 URL 접두사 목록으로 제한할 수 있습니다. 다음 예시에서는 다음을 제외한 모든 참조 *ExamplePlaceIndex*웹사이트에서 이름이 지정된 장소 색인 리소스에 대한 액세스를 거부합니다. example.com

> **Warning**
>
> aws:referer은 액세스를 제한할 수는 있지만 보안 메커니즘은 아닙니다. 공개적으로 알려진 참조자 헤더 값을 포함하는 것은 위험합니다. 권한이 없는 사용자가 수정된 브라우저나 사용자 지정 브라우저를 사용하여 원하는 aws:referer 값을 제공할 수 있습니다. 따라서 권한이 없는 당사자가 직접 AWS 요청하는 것을 방지하는 데 aws:referer 사용해서는 안 됩니다. 이러한 값은 고객이 Amazon S3에 저장된 콘텐츠 등의 디지털 콘텐츠를 권한이 없는 타사 사이트에서 참조하지 못하도록 보호하기 위해서만 사용하세요. 자세한 내용은 AWS:referer를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "geo:*",
            "Resource": "arn:aws:geo:us-west-2:111122223333:place-
index/ExamplePlaceIndex",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
            }
        }
    ]
}
```

### Trackers policy example

다음 정책은 기기 위치를 *ExampleTracker*업데이트하기 위해 이름이 지정된 트래커 리소스에 대한 액세스 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePosition",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": "arn:aws:geo:region:accountID:tracker/ExampleTracker"
    }
  ]
}
```

일치하는 aws:refererIAM 조건을 추가하면 리소스에 대한 브라우저 액세스를 URL 또는 URL 접두사 목록으로 제한할 수 있습니다. 다음 예시는 다음을 제외한 example.com 모든 참조 웹 *ExampleTracker*사이트에서 이름이 지정된 트래커 리소스에 대한 액세스를 거부합니다.

> **Warning**
>
> aws:referer은 액세스를 제한할 수는 있지만 보안 메커니즘은 아닙니다. 공개적으로 알려진 참조자 헤더 값을 포함하는 것은 위험합니다. 권한이 없는 사용자가 수정된 브라우저나 사용자 지정 브라우저를 사용하여 원하는 aws:referer 값을 제공할 수 있습니다. 따라서 승인되지 않은 당사자가 직접 AWS 요청하는 것을 방지하는 데 aws:referer 사용해서는 안 됩니다. 이러한 값은 고객이 Amazon S3에 저장된 콘텐츠 등의 디지털 콘텐츠를 권한이 없는 타사 사이트에서 참조하지 못하도록 보호하기 위해서만 사용하세요. 자세한 내용은 AWS:referer를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "geo:GetDevice*",
            "Resource": "arn:aws:geo:us-
west-2:111122223333:tracker/ExampleTracker",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
            }
        }
    ]
}
```

### Routes policy example

다음 정책은 경로를 계산하기 위해 이름이 지정된 경로 계산기 리소스에 *ExampleCalculator*대한 액세스 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:region:accountID:route-
calculator/ExampleCalculator"
    }
  ]
}
```

일치하는 [aws:refererIAM 조건을](#) 추가하면 리소스에 대한 브라우저 액세스를 URL 또는 URL 접두사 목록으로 제한할 수 있습니다. 다음 예시에서는 다음을 제외한 `example.com` 모든 참조 웹 *ExampleCalculator*사이트에서 이름이 지정된 경로 계산기에 대한 액세스를 거부합니다.

> **Warning**
>
> `aws:referer`은 액세스를 제한할 수는 있지만 보안 메커니즘은 아닙니다. 공개적으로 알려진 참조자 헤더 값을 포함하는 것은 위험합니다. 권한이 없는 사용자가 수정된 브라우저나 사용자 지정 브라우저를 사용하여 원하는 `aws:referer` 값을 제공할 수 있습니다. 따라서 권한이 없는 당사자가 직접 AWS 요청하는 것을 방지하는 데 `aws:referer` 사용해서는 안 됩니다. 이러한 값은 고객이 Amazon S3에 저장된 콘텐츠 등의 디지털 콘텐츠를 권한이 없는 타사 사이트에서 참조하지 못하도록 보호하기 위해서만 사용하세요. 자세한 내용은 AWS:referer를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "geo:*",
            "Resource": "arn:aws:geo:us-west-2:111122223333:route-
calculator/ExampleCalculator",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> 인증되지 않은 자격 증명 풀은 보안되지 않은 인터넷 사이트에 노출되기 위한 것이지만, 이러한 자격 증명 풀은 시간이 제한된 표준 자격 증명으로 교환된다는 점에 유의하십시오. AWS

인증되지 않은 자격 증명 풀과 관련된 IAM 역할의 범위를 적절하게 지정하는 것이 중요합니다.

11. 허용를 선택하여 자격 증명 풀을 생성하세요.

결과 자격 증명 풀은 `<region>:<GUID>.` 문법을 따릅니다.

예:

```
us-east-1:1sample4-5678-90ef-aaaa-1234abcd56ef
```

Amazon Location과 관련된 추가 정책 예는 the section called “자격 증명 기반 정책 예시”을 참조하세요.

## 에서 Amazon Cognito 자격 증명 풀 사용 JavaScript

다음 예시에서는 생성한 인증되지 않은 자격 증명 풀을 자격 증명으로 교환한 다음 이 자격 증명을 사용하여 맵 리소스의 스타일 설명자를 가져옵니다. *ExampleMap*

```
const AWS = require("aws-sdk");

const credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: "<identity pool ID>" // for example, us-east-1:1sample4-5678-90ef-
aaaa-1234abcd56ef
});

const client = new AWS.Location({
   credentials,
   region: AWS.config.region || "<region>"
});

console.log(await client.getMapStyleDescriptor("ExampleMap").promise());
```

> **Note**
>
> 인증되지 않은 ID에서 검색한 자격 증명은 1시간 동안 유효합니다.

다음은 자격 증명이 만료되기 전에 자격 증명을 자동으로 갱신하는 함수의 예입니다.

```
async function refreshCredentials() {
  await credentials.refreshPromise();
  // schedule the next credential refresh when they're about to expire
  setTimeout(refreshCredentials, credentials.expireTime - new Date());
}
```

이 작업을 간소화하기 위해 Amazon Location JavaScript 인증 도우미을 사용할 수 있습니다. 이는 자격 증명을 가져오고 새로 고치는 작업을 모두 대신합니다. 이 예시에서는 v3용 SDK를 사용합니다. AWS JavaScript

```
import { LocationClient, GetMapStyleDescriptorCommand } from "@aws-sdk/client-
location";
import { withIdentityPoolId } from "@aws/amazon-location-utilities-auth-helper";

const identityPoolId = "<identity pool ID>"; // for example, us-
east-1:1sample4-5678-90ef-aaaa-1234abcd56ef

// Create an authentication helper instance using credentials from Cognito
const authHelper = await withIdentityPoolId(identityPoolId);

const client = new LocationClient({
  region: "<region>", // The region containing both the identity pool and tracker
 resource
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  MapName: "ExampleMap",
};

const command = new GetMapStyleDescriptorCommand(input);

console.log(await client.send(command));
```

## 다음 단계

- 역할을 수정하려면 IAM 콘솔로 이동하세요.
- 자격 증명 풀을 관리하려면 Amazon Cognito 콘솔로 이동하세요.

# Amazon Location Service 모니터링

Amazon Location Service를 사용할 때 다음을 사용하여 시간 경과에 따른 사용량과 리소스를 모니터링할 수 있습니다:

- 아마존 CloudWatch. Amazon Location Service 리소스를 모니터링하고 통계가 포함된 지표를 거의 실시간으로 제공합니다.
- AWS CloudTrail. Amazon Location Service API에 대한 모든 호출의 이벤트 추적을 제공합니다.

이 섹션에서는 이러한 서비스 사용에 대한 정보를 제공합니다.

주제

- 아마존을 통한 아마존 로케이션 서비스 모니터링 CloudWatch
- AWS CloudTrail을 사용하여 로깅 및 모니터링

## 아마존을 통한 아마존 로케이션 서비스 모니터링 CloudWatch

Amazon은 실행 중인 AWS 리소스와 애플리케이션을 거의 AWS 실시간으로 CloudWatch 모니터링합니다. 원시 데이터를 수집하고 지표를 거의 실시간으로 의미 있는 통계로 처리하는 를 사용하여 CloudWatch Amazon Location 리소스를 모니터링할 수 있습니다. 최대 15개월 동안의 기록 정보를 보거나 Amazon CloudWatch 콘솔에서 확인할 지표를 검색하여 Amazon Location 리소스에 대한 더 자세한 관점을 확인할 수 있습니다. 또한 임계값 지정에 의한 경보를 설정하여 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취할 수 있습니다.

자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

주제

- 아마존으로 내보낸 Amazon Location Service 지표 CloudWatch
- Amazon Location Service 지표 보기
- Amazon Location Service 지표에 대한 CloudWatch 경보 생성
- 할당량 대비 사용량을 모니터링하는 CloudWatch 데 사용
- CloudWatch Amazon Location Service의 지표 예제

## 아마존으로 내보낸 Amazon Location Service 지표 CloudWatch

지표는 내보내는 CloudWatch 시간이 순차적으로 정렬된 데이터 포인트입니다. 차원이란 지표를 식별하는 이름/값 쌍을 말합니다. 자세한 내용은 Amazon 사용 CloudWatch 설명서의 CloudWatch 지표 및 CloudWatch차원 사용을 참조하십시오.

다음은 Amazon Location Service가 `AWS/Location` 네임스페이스로 내보내는 CloudWatch 지표입니다.

| 지표 | 설명 |
| --- | --- |
| `CallCount` | 지정된 API 엔드포인트에 대한 호출 횟수.<br><br>유효한 차원: Amazon Location Service API 이름<br><br>유효한 통계: Sum<br><br>단위: 개 |
| `ErrorCount` | 지정된 API 엔드포인트에 대한 호출의 오류 응답 횟수.<br><br>유효한 차원: Amazon Location Service API 이름<br><br>유효한 통계: Sum<br><br>단위: 개 |
| `SuccessCount` | 지정된 API 엔드포인트에 대한 성공적인 호출 횟수.<br><br>유효한 차원: Amazon Location Service API 이름<br><br>유효한 통계: Sum<br><br>단위: 개 |
| `CallLatency` | 지정된 API 엔드포인트로 호출이 이루어졌을 때 작업이 응답을 처리하고 반환하는 데 걸리는 시간.<br><br>유효한 차원: Amazon Location Service API 이름<br><br>유효한 통계: Average |

| 지표 | 설명 |
| --- | --- |
| | 단위: 밀리초 |

## Amazon Location Service 지표 보기

Amazon CloudWatch 콘솔에서 또는 Amazon CloudWatch API를 사용하여 Amazon Location Service의 지표를 볼 수 있습니다.

콘솔을 사용하여 지표를 보려면 CloudWatch

Example

1. https://console.aws.amazon.com/cloudwatch/ 에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. 모든 지표 탭에서 Amazon Location 네임스페이스를 선택합니다.
4. 확인하려는 지표 유형을 선택합니다.
5. 지표를 선택하여 차트에 추가합니다.

자세한 내용은 Amazon 사용 CloudWatch 설명서의 사용 가능한 지표 보기를 참조하십시오.

## Amazon Location Service 지표에 대한 CloudWatch 경보 생성

를 CloudWatch 사용하여 Amazon Location Service 지표에 경보를 설정할 수 있습니다. 예를 들어, 오류 수가 급증할 때마다 이메일을 CloudWatch 보내도록 경보를 생성할 수 있습니다.

다음 주제에서는 CloudWatch를 사용하여 경보를 설정하는 방법에 대한 상위 수준의 개요를 제공합니다. 자세한 지침은 Amazon 사용 CloudWatch 설명서의 알람 사용을 참조하십시오.

콘솔을 사용하여 알람을 설정하려면 CloudWatch

Example

1. https://console.aws.amazon.com/cloudwatch/ 에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보를 선택합니다.
3. 경보 생성을 선택합니다.
4. 지표 선택(Select metric)을 선택합니다.

5. 모든 지표 탭에서 Amazon Location 네임스페이스를 선택합니다.
6. 지표 카테고리를 선택합니다.
7. 경보를 만들려는 지표가 있는 행을 찾은 다음 이 행 옆의 확인란을 선택합니다.
8. 지표 선택(Select metric)을 선택합니다.
9. 지표에서 값을 입력합니다.
10. 경보 조건을 지정합니다.
11. 다음을 선택합니다.
12. 경보 조건이 충족될 때 알림을 보내려는 경우:
    - 경보 상태 트리거에서 알림 전송 여부를 묻는 경보 상태를 선택합니다.
    - SNS 주제 선택에서 새 주제 생성을 선택하여 Amazon Simple Notification Service (Amazon SNS) 주제를 새로 생성합니다. 주제 이름과 알림을 보낼 이메일을 입력합니다.
    - 알림 수신처에서 알림을 보낼 추가 이메일 주소를 입력합니다.
    - 알림 추가를 선택합니다. 이 목록은 향후 경보를 위해 필드에 저장되고 표시됩니다.
13. 완료되면 다음을 선택합니다.
14. 경보의 이름과 설명을 입력하고 다음을 선택합니다.
15. 알람 세부 정보를 확인한 후 다음을 선택합니다.

> **Note**
>
> 새 Amazon SNS 주제를 생성할 때에는, 이메일 주소를 확인해야 알림을 보낼 수 있습니다. 이메일이 확인되지 않은 경우, 상태 변경으로 경보가 시작될 때 알림이 수신되지 않습니다.

CloudWatch 콘솔을 사용하여 경보를 설정하는 방법에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 이메일을 보내는 경보 생성을 참조하십시오.

## 할당량 대비 사용량을 모니터링하는 CloudWatch 데 사용

Amazon CloudWatch 경보를 생성하여 지정된 할당량 사용률이 구성 가능한 임계값을 초과할 때 알림을 받을 수 있습니다. 이를 통해 할당량 한도에 근접했을 때를 인지하고 사용량을 조정하여 비용 초과가 발생하지 않도록 하거나 필요한 경우 할당량 증가를 요청할 수 있습니다. 할당량을 모니터링하는 CloudWatch 데 사용하는 방법에 대한 자세한 내용은 Amazon User Guide의 서비스 할당량 시각화 및 경보 설정을 참조하십시오. CloudWatch

## CloudWatch Amazon Location Service의 지표 예제

GetMetricDataAPI를 사용하여 Amazon 로케이션의 메트릭을 검색할 수 있습니다.

- 예를 들어, 수치가 떨어질 때를 대비하여 이를 모니터링하고 CallCount 경보를 설정할 수 있습니다.

  SendDeviceLocation의 CallCount 지표를 모니터링하면 추적된 자산을 한눈에 파악할 수 있습니다. CallCount이 하락했다면 트럭 등 추적 대상 자산이 현재 위치 전송을 중단했다는 뜻입니다. 이에 대한 경보를 설정하면 문제 발생을 알리는 데 도움이 될 수 있습니다.
- 또 다른 예로, 수치가 급증할 때를 대비하여 이를 모니터링하고 ErrorCount 경보를 설정할 수 있습니다.

  지오펜스를 기준으로 기기 위치를 평가하려면 트래커를 지오펜스 컬렉션과 연결해야 합니다. 지속적인 위치 업데이트가 필요한 디바이스 플릿이 있는 경우, BatchEvaluateGeofence 또는 BatchPutDevicePosition의 CallCount이 0으로 떨어지면 업데이트가 더 이상 진행되지 않는다는 의미입니다.

다음은 맵 리소스 생성에 대한 GetMetricData지표와 맵 리소스 ErrorCount 생성에 대한 예제 출력입니다. CallCount

```
{
  "StartTime": 1518867432,
  "EndTime": 1518868032,
  "MetricDataQueries": [
    {
      "Id": "m1",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Location",
          "MetricName": "CallCount",
          "Dimensions": [
            {
              "Name": "SendDeviceLocation",
              "Value": "100"
            }
          ]
        },
        "Period": 300,
        "Stat": "SampleCount",
        "Unit": "Count"
```

```
      }
    },
    {
      "Id": "m2",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Location",
          "MetricName": "ErrorCount",
          "Dimensions": [
            {
              "Name": "AssociateTrackerConsumer",
              "Value": "0"
            }
          ]
        },
        "Period": 1,
        "Stat": "SampleCount",
        "Unit": "Count"
      }
    }
  ]
}
```

## AWS CloudTrail을 사용하여 로깅 및 모니터링

AWS CloudTrail사용자, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스입니다. CloudTrail 모든 API 호출을 이벤트로 기록합니다. Amazon Location Service와 함께 사용하여 API 호출을 CloudTrail 모니터링할 수 있습니다. 여기에는 Amazon 위치 서비스 콘솔에서의 호출과 Amazon 위치 서비스 API 작업에 대한 AWS SDK 호출이 포함됩니다.

트레일을 생성하면 Amazon Location Service의 CloudTrail 이벤트를 포함하여 S3 버킷으로 이벤트를 지속적으로 전송할 수 있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 가장 최근 이벤트를 계속 볼 수 있습니다. 에서 수집한 CloudTrail 정보를 사용하여 Amazon Location Service에 이루어진 요청, 요청이 이루어진 IP 주소, 요청한 사람, 요청 시기 및 추가 세부 정보를 확인할 수 있습니다.

에 대한 CloudTrail 자세한 내용은 AWS CloudTrail사용 설명서를 참조하십시오.

주제

- 아마존 위치 서비스 정보 CloudTrail
- Amazon Location Service 로그 파일 항목 이해

## 아마존 위치 서비스 정보 CloudTrail

CloudTrail 계정을 만들면 AWS 계정에서 활성화됩니다. Amazon Location Service에서 활동이 발생하면 해당 활동이 CloudTrail 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 이벤트 기록으로 CloudTrail 이벤트 보기를 참조하십시오.

Amazon Location Service의 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려는 경우, 추적을 생성합니다. 트레일을 사용하면 CloudTrail S3 버킷에 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- 추적 생성 개요
- CloudTrail 지원되는 서비스 및 통합
- 에 대한 Amazon SNS 알림 구성 CloudTrail
- 여러 지역에서 CloudTrail 로그 파일 수신 및 여러 계정으로부터 CloudTrail 로그 파일 수신

모든 Amazon Location Service 작업은 Amazon 위치 서비스 API 참조에 의해 CloudTrail 기록되고 문서화됩니다. 예를 들어`CreateTracker`, 에 대한 호출 `UpdateTracker` 및 `DescribeTracker` 작업은 CloudTrail 로그 파일에 항목을 생성합니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 다음 중 어떤 자격 증명 정보를 사용하여 요청이 수행되었는지 여부를 확인할 수 있습니다:

- 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명 사용.
- 역할 또는 연합된 사용자에 대한 임시 보안 자격 증명 사용.
- 다른 AWS 서비스 사용.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

## Amazon Location Service 로그 파일 항목 이해

트레일은 지정한 S3 버킷 또는 Amazon CloudWatch Logs에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. 자세한 내용은 AWS CloudTrail사용 설명서의 CloudTrail 로그 파일 작업을 참조하십시오.

CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함되어 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다.

> **Note**
>
> CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 추적이 아니므로 특정 순서로 표시되지 않습니다. 작업 순서를 정하려면 `eventTime`을 사용하세요.

다음 예제는 트래커 리소스를 생성하는 `CreateTracker` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "123456789012",
        "arn": "arn:aws:geo:us-east-1:123456789012:tracker/ExampleTracker"
        "accountId": "123456789012",
        "accessKeyId": "123456789012",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "123456789012",
                "arn": "arn:aws:geo:us-east-1:123456789012:tracker/ExampleTracker",
                "accountId": "123456789012",
                "userName": "exampleUser",
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-10-22T16:36:07Z"
            }
        }
    },
    "eventTime": "2020-10-22T17:43:30Z",
    "eventSource": "geo.amazonaws.com",
    "eventName": "CreateTracker",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0/24—TEST-NET-1",
```

```
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.864
 Linux/4.14.193-110.317.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.8+10-LTS java/11.0.8
 kotlin/1.3.72 vendor/Amazon.com_Inc. exec-env/AWS_Lambda_java11",
    "requestParameters": {
        "TrackerName": "ExampleTracker",
        "Description": "Resource description"
    },
    "responseElements": {
        "TrackerName": "ExampleTracker",
        "Description": "Resource description"
        "TrackerArn": "arn:partition:service:region:account-id:resource-id",
        "CreateTime": "2020-10-22T17:43:30.521Z"
    },
    "requestID": "557ec619-0674-429d-8e2c-eba0d3f34413",
    "eventID": "3192bc9c-3d3d-4976-bbef-ac590fa34f2c",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012",
}
```

다음은 트래커 리소스의 세부 정보를 반환하는 DescribeTracker 작업에 대한 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "123456789012",
        "arn": "arn:partition:service:region:account-id:resource-id",
        "accountId": "123456789012",
        "accessKeyId": "123456789012",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "123456789012",
                "arn": "arn:partition:service:region:account-id:resource-id",
                "accountId": "123456789012",
                "userName": "exampleUser",
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-10-22T16:36:07Z"
```

```
            }
        }
    },
    "eventTime": "2020-10-22T17:43:33Z",
    "eventSource": "geo.amazonaws.com",
    "eventName": "DescribeTracker",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0/24–TEST-NET-1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.864
 Linux/4.14.193-110.317.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.8+10-LTS java/11.0.8
 kotlin/1.3.72 vendor/Amazon.com_Inc. exec-env/AWS_Lambda_java11",
    "requestParameters": {
        "TrackerName": "ExampleTracker"
    },
    "responseElements": null,
    "requestID": "997d5f93-cfef-429a-bbed-daab417ceab4",
    "eventID": "d9e0eebe-173c-477d-b0c9-d1d8292da103",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012",
}
```

# AWS CloudFormation을 사용하여 Amazon Location Service 리소스 생성하기

Amazon Location Service는 리소스 및 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있도록 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스인 AWS CloudFormation과 통합됩니다. Amazon Location 리소스 등 필요한 모든 AWS 리소스를 설명하는 템플릿을 생성하고, AWS CloudFormation는 그러한 리소스를 프로비저닝하고 구성합니다.

AWS CloudFormation을 사용할 때 템플릿을 재사용하여 Amazon Location 리소스를 일관되고 반복적으로 설정할 수 있습니다. 리소스를 한 번 설명한 후 여러 AWS 계정 및 리전에서 동일한 리소스를 반복적으로 프로비저닝할 수 있습니다.

## Amazon Location 및 AWS CloudFormation 템플릿

Amazon Location 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 AWS CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이 템플릿은 AWS CloudFormation 스택에서 프로비저닝할 리소스에 대해 설명합니다. JSON 또는 YAML에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하면 AWS CloudFormation 템플릿을 시작

하는 데 도움이 됩니다. 자세한 내용은 AWS CloudFormation 사용 설명서에서 AWS CloudFormation Designer란 무엇입니까?를 참조하세요.

Amazon Location은 AWS CloudFormation에서 다음과 같은 리소스 유형의 생성을 지원합니다:

- AWS::Location::Map
- AWS::Location::PlaceIndex
- AWS::Location::RouteCalculator
- AWS::Location::Tracker
- AWS::Location::TrackerConsumer
- AWS::Location::GeofenceCollection

Amazon Location 리소스에 대한 JSON 및 YAML 템플릿의 예를 포함한 자세한 내용은 AWS CloudFormation 사용 설명서의 Amazon Location Service 리소스 유형 참조를 참조하세요.

## AWS CloudFormation에 대해 자세히 알아보기

AWS CloudFormation에 대한 자세한 내용은 다음 리소스를 참조하십시오.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation API 참조
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

# Amazon Location Service의 보안

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 귀사 간의 공동 책임입니다. AWS 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다 AWS 클라우드. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. Amazon Location Service에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내 서비스 규정 준수 참조하십시오.
- 클라우드에서의 보안 — 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon Location을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 항목에서는 보안 및 규정 준수 목적에 맞게 Amazon Location을 구성하는 방법을 보여줍니다. 또한 Amazon Location 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

# Amazon Location Service의 데이터 보호

AWS 공동 책임 모델 Amazon Location Service의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 은 (는) 모두를 실행하는 글로벌 인프라를 보호할 책임이 AWS 클라우드있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시를 참조하십시오FAQ. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 공동 책임 모델 및AWS 보안 GDPR 블로그의 블로그 게시물을 참조하십시오.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 개별 사용자에게 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM) 를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정마다 다단계 인증 (MFA) 을 사용하십시오.
- SSLTLS/를 사용하여 AWS 리소스와 통신하세요. TLS1.2가 필요하고 TLS 1.3을 권장합니다.
- API를 사용하여 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API an을 AWS 통해 액세스할 때 FIPS 140-3개의 검증된 암호화 모듈이 필요한 경우 엔드포인트를 사용하십시오. FIPS 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준 () 140-3을 참조하십시오. FIPS

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon Location 또는 다른 AWS 서비스 콘솔을 사용하여 작업하는 경우 API AWS CLI, 또는 등이 포함됩니다 AWS SDKs. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL a를 제공하는 경우 해당 서버에 대한 요청을 URL 검증하기 위해 자격 증명 정보를 에 포함하지 않는 것이 좋습니다.

## 데이터 개인 정보 보호

Amazon Location Service를 사용하면 조직의 데이터에 대한 통제권을 유지할 수 있습니다. Amazon Location은 고객 메타데이터와 계정 정보를 제거하여 데이터 공급자에게 전송되는 모든 쿼리를 익명화합니다.

Amazon Location은 추적 및 지오펜싱에 데이터 공급자를 사용하지 않습니다. 즉, 민감한 데이터는 AWS 계정에 그대로 남아 있습니다. 이를 통해 시설, 자산 및 직원 위치와 같은 민감한 위치 정보를 제3자로부터 보호하고, 사용자 개인 정보를 보호하고, 애플리케이션의 보안 위험을 줄일 수 있습니다.

자세한 내용은 AWS 데이터 프라이버시를 참조하십시오FAQ.

## Amazon Location에서의 데이터 보존

다음 특성은 Amazon Location에서 서비스에 대한 데이터를 수집하고 저장하는 방식과 관련이 있습니다.

- Amazon Location Service 추적기 — APIs 추적기를 사용하여 개체의 위치를 추적하면 해당 좌표를 저장할 수 있습니다. 디바이스 위치는 서비스에 의해 삭제되기 전에 30일 동안 저장됩니다.
- Amazon Location Service 지오펜스 — 지오펜스를 APIs 사용하여 관심 영역을 정의하면 서비스는 사용자가 제공한 지오메트리를 저장합니다. 이는 명시적으로 삭제해야 합니다.

> **Note**
>
> 계정을 삭제하면 AWS 계정 내 모든 리소스가 삭제됩니다. 자세한 내용은 AWS 데이터 프라이버시를 참조하십시오FAQ.

## Amazon Location Service의 저장 데이터 암호화

Amazon Location Service는 기본적으로 암호화를 제공하여 저장된 민감한 고객 데이터를 AWS 자체 암호화 키를 사용하여 보호합니다.

- AWS 소유 키 — Amazon Location은 기본적으로 이러한 키를 사용하여 개인 식별 데이터를 자동으로 암호화합니다. AWS 소유 키를 확인, 관리 또는 사용하거나 사용 여부를 감사할 수 없습니다. 하지만 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS 소유 키를 참조하세요.

기본적으로 저장 데이터를 암호화하면 민감한 데이터 보호와 관련된 운영 오버헤드와 복잡성을 줄이는 데 도움이 됩니다. 동시에 엄격한 암호화 규정 준수 및 규제 요구 사항을 충족하는 안전한 애플리케이션을 구축할 수 있습니다.

이 암호화 계층을 비활성화하거나 다른 암호화 유형을 선택할 수는 없지만, 트래커 및 지오펜스 수집 리소스를 생성할 때 고객 관리 키를 선택하여 기존 AWS 소유 암호화 키 위에 두 번째 암호화 계층을 추가할 수 있습니다.

- 고객 관리형 키 — Amazon Location은 사용자가 생성하고 소유하고 관리하는 대칭형 고객 관리 키를 사용하여 기존 AWS 소유 암호화에 두 번째 암호화 계층을 추가할 수 있도록 지원합니다. 이 암호화 계층을 완전히 제어할 수 있으므로 다음과 같은 작업을 수행할 수 있습니다.
  - 키 정책 수립 및 유지
  - IAM정책 및 보조금의 수립 및 유지
  - 키 정책 활성화 및 비활성화
  - 키 암호화 자료 교체
  - 태그 추가
  - 키 별칭 생성
  - 키 삭제 일정 수립

  자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 관리 키를 참조하세요.

다음 표에는 Amazon Location에서 개인 식별 데이터를 암호화하는 방법이 요약되어 있습니다.

| 데이터 유형 | AWS 소유 키 암호화 | 고객 관리 키 암호화(선택 사항) |
|---|---|---|
| `Position`<br><br>디바이스 위치 세부 정보가 포함된 포인트 지오메트리입니다. | 활성화됨 | 활성화됨 |
| `PositionProperties`<br><br>위치 업데이트와 연결된 키-값 쌍 세트입니다. | 활성화됨 | 활성화됨 |
| `GeofenceGeometry` | 활성화됨 | 활성화됨 |

| 데이터 유형 | AWS 소유 키 암호화 | 고객 관리 키 암호화(선택 사항) |
| --- | --- | --- |
| 지오펜스 영역을 나타내는 다각형 지오펜스 지오메트리입니다. | | |
| `DeviceId`<br><br>트래커 리소스에 디바이스 위치 업데이트를 업로드할 때 지정되는 디바이스 식별자입니다. | 활성화됨 | 지원되지 않음 |
| `GeofenceId`<br><br>지오펜스 지오메트리 또는 특정 지오펜스 컬렉션의 지오펜스 배치를 저장할 때 지정되는 식별자입니다. | 활성화됨 | 지원되지 않음 |

> **Note**
>
> Amazon Location은 AWS 소유 키를 사용하여 저장된 데이터를 자동으로 암호화하여 개인 식별 데이터를 무료로 보호합니다.
> 하지만 고객 관리 키 사용에는 AWS KMS 요금이 부과됩니다. 요금에 대한 자세한 내용은 AWS Key Management Service 요금을 참조하십시오.

에 대한 자세한 내용은 AWS KMSWhat is AWS Key Management Service? 를 참조하십시오.

## Amazon Location Service에서 지원금을 사용하는 방법 AWS KMS

Amazon Location에서 고객 관리형 키를 사용하려면 권한 부여가 필요합니다.

고객 관리 키로 암호화된 트래커 리소스 또는 지오펜스 컬렉션을 생성하면 Amazon Location에서 CreateGrant요청을 전송하여 사용자를 대신하여 허가를 생성합니다. AWS KMS권한 AWS KMS 부여는 Amazon Location에서 고객 계정의 KMS 키에 대한 액세스 권한을 부여하는 데 사용됩니다.

Amazon Location은 다음 내부 작업에 대해 고객 관리형 키를 사용하기 위한 권한 부여가 필요합니다.

- 트래커 또는 지오펜스 컬렉션을 생성할 때 입력한 대칭 고객 관리 KMS 키 ID가 유효한지 DescribeKey AWS KMS 확인하라는 요청을 보내십시오.
- 고객 관리 키로 암호화된 데이터 키를 AWS KMS 생성해 GenerateDataKeyWithoutPlaintext달라는 요청을 보내세요.
- 암호화된 데이터 키를 해독하여 AWS KMS 데이터를 암호화하는 데 사용할 수 있도록 암호 해독 요청을 보내십시오.

언제든지 권한 부여에 대한 액세스 권한을 취소하거나 고객 관리형 키에 대한 서비스 액세스를 제거할 수 있습니다. 그렇게 하면 Amazon Location은 고객 관리형 키로 암호화된 데이터에 액세스할 수 없게 되며, 이는 해당 데이터에 의존하는 작업에 영향을 미칩니다. 예를 들어 Amazon Location에서 액세스할 수 없는 암호화된 트래커에서 디바이스 위치를 가져오려고 시도하면 작업에서 `AccessDeniedException` 오류가 반환됩니다.

## 고객 관리형 키 생성

또는 를 사용하여 대칭 고객 관리 키를 생성할 수 있습니다. AWS Management Console AWS KMS APIs

대칭형 고객 관리형 키를 생성하려면

AWS Key Management Service 개발자 안내서의 대칭형 고객 관리형 키 생성 단계를 따르세요.

키 정책

키 정책은 고객 관리형 키에 대한 액세스를 제어합니다. 모든 고객 관리형 키에는 키를 사용할 수 있는 사람과 키를 사용하는 방법을 결정하는 문장이 포함된 정확히 하나의 키 정책이 있어야 합니다. 고객 관리형 키를 생성할 때 키 정책을 지정할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 관리형 키에 대한 액세스 관리를 참조하십시오.

Amazon Location 리소스와 함께 고객 관리형 키를 사용하려면 키 정책에서 다음 API 작업을 허용해야 합니다.

- `kms:CreateGrant` - 고객 관리형 키에 권한 부여를 추가합니다. 지정된 KMS 키에 대한 제어 액세스 권한을 부여하여 Amazon Location에서 요구하는 작업을 허용할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 권한 부여 사용을 참조하세요.

  이를 통해 Amazon Location은 다음을 수행할 수 있습니다.

  - 데이터 키는 암호화에 바로 사용되지 않기 때문에 `GenerateDataKeyWithoutPlainText`을(를) 호출하여 암호화된 데이터 키를 생성하여 저장합니다.
  - 저장된 암호화된 데이터 키를 사용하여 암호화된 데이터에 액세스하려면 `Decrypt`를 호출합니다.
  - 사용 중지하는 보안 주체를 설정하여 `RetireGrant`에 대한 서비스를 허용합니다.
- `kms:DescribeKey` - Amazon Location에서 키의 유효성을 확인할 수 있도록 고객 관리형 키 세부 정보를 제공합니다.

다음은 Amazon Location에 추가할 수 있는 정책 설명 예시입니다.

```
 "Statement" : [
   {
     "Sid" : "Allow access to principals authorized to use Amazon Location",
     "Effect" : "Allow",
     "Principal" : {
       "AWS" : "*"
     },
     "Action" : [
       "kms:DescribeKey",
       "kms:CreateGrant"
     ],
     "Resource" : "*",
     "Condition" : {
       "StringEquals" : {
         "kms:ViaService" : "geo.region.amazonaws.com",
         "kms:CallerAccount" : "111122223333"
       }
   },
   {
     "Sid": "Allow access for key administrators",
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::111122223333:root"
      },
     "Action" : [
       "kms:*"
      ],
     "Resource": "arn:aws:kms:region:111122223333:key/key_ID"
   },
   {
     "Sid" : "Allow read-only access to key metadata to the account",
     "Effect" : "Allow",
```

```
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:root"
      },
      "Action" : [
        "kms:Describe*",
        "kms:Get*",
        "kms:List*",
        "kms:RevokeGrant"
      ],
      "Resource" : "*"
    }
  ]
```

정책에서 사용 권한을 지정하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.

키 액세스 문제 해결에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.

## Amazon Location에 대한 고객 관리형 키 지정

고객 관리 키를 다음 리소스의 2차 계층 암호화로 지정할 수 있습니다.

- 트래커 리소스
- 지오펜스 컬렉션

리소스를 생성할 때 Amazon Location에서 리소스에 저장된 식별 가능한 개인 데이터를 암호화하는 데 사용하는 KMSID를 입력하여 데이터 키를 지정할 수 있습니다.

- KMSID — AWS KMS 고객 관리 키의 키 식별자입니다. 키 ID, 키ARN, 별칭 이름 또는 별칭을 ARN 입력합니다.

## Amazon Location Service 암호화 컨텍스트

암호화 컨텍스트는 데이터에 대한 추가 컨텍스트 정보를 포함하는 선택적 키-값 페어 세트입니다.

AWS KMS 암호화 컨텍스트를 추가 인증 데이터로 사용하여 인증된 암호화를 지원합니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하면 암호화 컨텍스트를 암호화된 데이터에 AWS KMS 바인딩합니다. 데이터를 해독하려면 요청에 동일한 암호화 컨텍스트를 포함합니다.

Amazon Location Service 암호화 컨텍스트

Amazon Location은 모든 AWS KMS 암호화 작업에서 동일한 암호화 컨텍스트를 사용합니다. 여기서 키는 Amazon 리소스 이름 () 이고 값은 리소스 Amazon 리소스 이름 (ARN) 입니다. `aws:geo:arn`

Example

```
"encryptionContext": {
    "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/SAMPLE-
GeofenceCollection"
}
```

모니터링을 위한 암호화 컨텍스트 사용

대칭형 고객 관리 키를 사용하여 트래커 또는 지오펜스 컬렉션을 암호화하는 경우 감사 기록 및 로그의 암호화 컨텍스트를 사용하여 고객 관리 키가 사용되는 방식을 식별할 수도 있습니다. 암호화 컨텍스트는 AWS CloudTrail 또는 Amazon Logs에서 생성한 CloudWatch 로그에도 나타납니다.

암호화 컨텍스트를 사용하여 고객 관리형 키에 대한 액세스 제어

키 정책 및 IAM 정책의 암호화 컨텍스트를 사용하여 대칭 고객 관리 키에 대한 `conditions` 액세스를 제어할 수 있습니다. 권한 부여에 암호화 컨텍스트 제약 조건을 사용할 수도 있습니다.

Amazon Location은 권한 부여 시 암호화 컨텍스트 제약 조건을 사용하여 계정 또는 리전의 고객 관리형 키에 대한 액세스를 제어합니다. 권한 부여 제약 조건에 따라 권한 부여가 허용하는 작업은 지정된 암호화 컨텍스트를 사용해야 합니다.

Example

다음은 특정 암호화 컨텍스트에서 고객 관리형 키에 대한 액세스 권한을 부여하는 키 정책 설명의 예입니다. 이 정책 설명의 조건에 따라 권한 부여에는 암호화 컨텍스트를 지정하는 암호화 컨텍스트 제약 조건이 있어야 합니다.

```
{
    "Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:DescribeKey",
     "Resource": "*"
},
{
     "Sid": "Enable CreateGrant",
     "Effect": "Allow",
```

```
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
    },
    "Action": "kms:CreateGrant",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:EncryptionContext:aws:geo:arn": "arn:aws:geo:us-
west-2:111122223333:tracker/SAMPLE-Tracker"
         }
    }
}
```

## Amazon Location Service의 암호화 키 모니터링

Amazon Location Service 리소스와 함께 AWS KMS 고객 관리형 키를 사용하면 Amazon Location에서 보내는 요청을 추적하는 데 Amazon CloudWatch Logs를 사용할 AWS CloudTrail수 AWS KMS있습니다.

다음은 고객 관리 키로 암호화된 데이터에 `DescribeKey` 액세스하기 위해 Amazon Location에서 호출하는 `CreateGrant` `GenerateDataKeyWithoutPlainTextDecrypt`,, 및 모니터링 KMS 작업을 위한 AWS CloudTrail 이벤트입니다.

CreateGrant

AWS KMS 고객 관리 키를 사용하여 트래커 또는 지오펜스 수집 리소스를 암호화하면 Amazon Location에서 사용자를 대신하여 계정의 KMS 키에 `CreateGrant` 액세스하라는 요청을 보냅니다. AWS Amazon Location에서 생성하는 권한 부여는 AWS KMS 고객 관리형 키와 연결된 리소스에만 적용됩니다. 또한 Amazon Location은 리소스를 삭제하면 `RetireGrant` 작업을 사용하여 권한 부여를 제거합니다.

다음 예제 이벤트는 `CreateGrant` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
```

```
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "retiringPrincipal": "geo.region.amazonaws.com",
        "operations": [
            "GenerateDataKeyWithoutPlaintext",
            "Decrypt",
            "DescribeKey"
        ],
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "granteePrincipal": "geo.region.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

### GenerateDataKeyWithoutPlainText

트래커 또는 지오펜스 수집 리소스의 AWS KMS 고객 관리 키를 활성화하면 Amazon Location 에서 고유한 테이블 키를 생성합니다. 리소스의 AWS KMS고객 관리 키를 AWS KMS 지정하는 `GenerateDataKeyWithoutPlainText` 요청을 보냅니다.

다음 예제 이벤트는 `GenerateDataKeyWithoutPlainText` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/
SAMPLE-GeofenceCollection"
        },
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
```

```
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "57f5dbee-16da-413e-979f-2c4c6663475e"
}
```

### Decrypt

암호화된 트래커 또는 지오펜스 컬렉션에 액세스하면 Amazon Location에서 `Decrypt` 작업을 호출하여 저장된 암호화 데이터 키를 사용하여 암호화된 데이터에 액세스합니다.

다음 예제 이벤트는 `Decrypt` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:10:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/
SAMPLE-GeofenceCollection"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
```

```
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

### DescribeKey

Amazon Location은 `DescribeKey` 작업을 사용하여 트래커 또는 지오펜스 컬렉션과 관련된 AWS KMS 고객 관리 키가 계정 및 리전에 존재하는지 확인합니다.

다음 예제 이벤트는 `DescribeKey` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
```

```
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```


## 자세히 알아보기

다음 리소스에서 키에 대한 추가 정보를 확인할 수 있습니다.

- AWS Key Management Service 기본 개념에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.
- 의 보안 모범 사례에 대한 AWS Key Management Service 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하십시오.

## Amazon Location Service의 전송 중 데이터 암호화

Amazon Location은 Transport Layer Security (TLS) 1.2 암호화 프로토콜을 사용하여 모든 네트워크 간 데이터를 자동으로 암호화하여 서비스를 오가는 전송 데이터를 보호합니다. Amazon Location Service로 전송된 직접 HTTPS APIs 요청은 AWS서명 버전 4 알고리즘을 사용하여 서명되어 보안 연결을 설정합니다.

# Amazon Location Service를 위한 ID 및 액세스 관리

AWS Identity and Access Management (IAM) 는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와줍니다. IAM관리자는 Amazon Location 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 를 받을 수 있는 사용자를 관리합니다. IAM추가 비용 없이 사용할 AWS 서비스 수 있습니다.

주제

- 고객
- ID를 통한 인증
- 정책을 사용한 액세스 관리
- Amazon Location Service와 함께 작동하는 방식 IAM
- Amazon Location Service가 인증되지 않은 사용자를 처리하는 방법
- Amazon Location Service의 자격 증명 기반 정책 예제
- Amazon Location Service 자격 증명 및 액세스 문제 해결

### 고객

Amazon Location에서 수행하는 작업에 따라 AWS Identity and Access Management (IAM) 사용 방법이 다릅니다.

서비스 사용자 – Amazon Location Service 를 사용하여 작업을 수행하는 경우 관리자는 필요한 보안 인증과 권한을 제공합니다. 더 많은 Amazon Location 기능을 사용하여 작업을 수행할수록 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon Location의 기능에 액세스할 수 없다면 Amazon Location Service 자격 증명 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 – 회사에서 Amazon Location 리소스를 담당하고 있다면 아마도 Amazon Location에 대한 전체 액세스 권한을 갖고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon Location 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 서비스 사용자의 권한을 변경해 달라는 요청을 제출해야 합니다. 이 페이지의 정보를 검토하여 의 기본 개념을 IAM 이해하십시오. 회사에서 Amazon Location을 사용하는 방법에 대해 자세히 알아보려면 IAM 을 참조하십시오Amazon Location Service와 함께 작동하는 방식 IAM.

IAM관리자 — IAM 관리자인 경우 Amazon Location에 대한 액세스를 관리하는 정책을 작성하는 방법에 대한 세부 정보를 알고 싶을 수 있습니다. 에서 IAM 사용할 수 있는 Amazon Location ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon Location Service의 자격 증명 기반 정책 예제

## ID를 통한 인증

인증은 ID 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM사용자로서 또는 역할을 위임하여 인증 (로그인 AWS) 을 받아야 합니다. AWS 계정 루트 사용자 IAM

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAMID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 페더레이션 ID로 로그인하는 경우 관리자는 이전에 역할을 사용하여 ID 페더레이션을 설정했습니다. IAM 페더레이션을 AWS 사용하여 액세스하는 경우 간접적으로 역할을 수임하는 것입니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호로 서명할 수 있는 소프트웨어 개발 키트 (SDKCLI) 와 명령줄 인터페이스 () 가 AWS 제공됩니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 사용 IAM설명서의 AWS API요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, 계정 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 AWS 것을 권장합니다. 자세한 내용은 사용 설명서의 다단계 인증 및 사용 AWS IAM Identity Center 설명서의 다단계 인증 사용 (MFA) 을 IAM 참조하십시오. AWS

### AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 작업에 루트 사용자를 사용하지 않

을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 작업의 전체 목록은 사용 설명서의 루트 사용자 자격 증명이 필요한 작업을 참조하십시오. IAM

## 페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 AWS 서비스 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 액세스하도록 하는 것입니다.

페더레이션 ID는 기업 사용자 디렉토리, 웹 ID 공급자, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 자격 증명을 사용하여 액세스하는 AWS 서비스 모든 사용자를 말합니다. AWS Directory Service페더레이션 ID에 AWS 계정액세스하면 이들이 역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center(을)를 사용하는 것이 좋습니다. IAMIdentity Center에서 사용자 및 그룹을 만들거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 AWS 계정 모든 사용자 및 애플리케이션에서 사용할 수 있습니다. ID 센터에 대한 자세한 내용은 IAM ID 센터란 IAM 무엇입니까? 를 참조하십시오. AWS IAM Identity Center 사용 설명서에서

## IAM 사용자 및 그룹

IAM사용자란 단일 사용자 또는 애플리케이션에 대한 특정 권한을 가진 사용자 내의 ID입니다. AWS 계정 가능하면 암호 및 액세스 키와 같은 장기 자격 증명을 가진 IAM 사용자를 만드는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 특정 사용 사례에서 IAM 사용자의 장기 자격 증명이 필요한 경우에는 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 사용 설명서의 장기 자격 증명이 필요한 사용 사례에 대한 정기적인 액세스 키 IAM 교체를 참조하십시오.

IAM그룹은 IAM 사용자 컬렉션을 지정하는 ID입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어 이름을 지정한 IAMAdmins그룹을 만들고 해당 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세히 알아보려면 사용 설명서의 역할 대신 IAM 사용자를 만드는 시기를 참조하십시오. IAM

## IAM역할

IAM역할은 특정 권한을 AWS 계정 가진 사용자 내의 ID입니다. IAM사용자와 비슷하지만 특정인과 관련이 있는 것은 아닙니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을

맡을 수 있습니다. AWS CLI or AWS API 작업을 호출하거나 사용자 지정을 사용하여 역할을 수임할 수 URL 있습니다. 역할 사용 방법에 대한 자세한 내용은 사용 IAM설명서의 IAM역할 사용을 참조하십시오.

IAM임시 자격 증명이 있는 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션을 위한 역할에 대한 자세한 내용은 IAM사용 설명서의 타사 ID 제공자를 위한 역할 생성을 참조하십시오. IAMIdentity Center를 사용하는 경우 권한 집합을 구성합니다. ID가 인증된 후 액세스할 수 있는 대상을 제어하기 위해 IAM Identity Center는 권한 집합을 역할의 상관 관계와 연결합니다. IAM 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할이 역할을 맡아 특정 작업에 대해 일시적으로 다른 권한을 부여받을 수 있습니다. IAM
- 계정 간 액세스 - IAM 역할을 사용하여 다른 계정의 사용자 (신뢰할 수 있는 사용자) 가 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 하지만 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 계정 간 액세스에 대한 역할과 리소스 기반 정책 간의 차이점을 알아보려면 사용 설명서의 교차 계정 리소스 액세스를 참조하십시오. IAM IAM
- 서비스 간 액세스 — 일부는 다른 기능을 AWS 서비스 사용합니다. AWS 서비스예를 들어, 서비스를 호출하면 해당 서비스가 Amazon에서 애플리케이션을 EC2 실행하거나 Amazon S3에 객체를 저장하는 것이 일반적입니다. 서비스는직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS전화를 거는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하라는 요청과 결합하여 사용합니다. AWS 서비스 FAS요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS요청 시 적용되는 정책 세부 정보는 전달 액세스 세션을 참조하십시오.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 간주하는 IAM 역할입니다. IAM관리자는 내부에서 IAM 서비스 역할을 만들고, 수정하고, 삭제할 수 있습니다. 자세한 내용은 사용 설명서의 역할 만들기를 참조하여 권한을 위임하십시오IAM. AWS 서비스
  - 서비스 연결 역할 - 서비스 연결 역할은 에 연결된 서비스 역할 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자

에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

- Amazon에서 실행 중인 애플리케이션 EC2 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS API 요청을 보내는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS CLI EC2인스턴스 내에 액세스 키를 저장하는 것보다 이 방법이 더 좋습니다. EC2인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 만들어야 합니다. 인스턴스 프로필에는 역할이 포함되며, 이를 통해 EC2 인스턴스에서 실행 중인 프로그램이 임시 자격 증명을 얻을 수 있습니다. 자세한 내용은 사용 설명서의 IAM역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 IAM 참조하십시오.

IAM역할을 사용할지 IAM 사용자를 사용할지 알아보려면 사용 설명서의 IAM 역할 생성 시기 (사용자 대신) 를 IAM참조하십시오.

## 정책을 사용한 액세스 관리

정책을 만들고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON정책 문서의 구조 및 내용에 대한 자세한 내용은 IAM사용 설명서의 JSON 정책 개요를 참조하십시오.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. IAM관리자는 IAM 정책을 생성하여 필요한 리소스에서 작업을 수행할 수 있는 권한을 사용자에게 부여할 수 있습니다. 그러면 관리자가 역할에 IAM 정책을 추가할 수 있으며, 사용자는 역할을 수임할 수 있습니다.

IAM정책은 작업을 수행하는 데 사용하는 방법에 관계없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 에서 역할 정보를 가져올 수 AWS API 있습니다.

### 보안 인증 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 사용 설명서의 IAM정책 생성을 참조하십시오. IAM

보안 인증 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책과 인라인 정책 중에서 선택하는 방법을 알아보려면 IAM사용 설명서의 관리형 정책과 인라인 정책 중 선택을 참조하십시오.

### 리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 IAM 정책에서는 AWS 관리형 정책을 사용할 수 없습니다.

### 액세스 제어 목록 () ACLs

액세스 제어 목록 (ACLs) 은 리소스에 액세스할 수 있는 권한을 가진 주체 (계정 구성원, 사용자 또는 역할) 를 제어합니다. ACLs정책 문서 형식을 사용하지는 않지만 리소스 기반 정책과 JSON 비슷합니다.

지원하는 서비스의 VPC 예로는 Amazon S3와 Amazon이 ACLs 있습니다. AWS WAF자세한 내용은 Amazon 심플 스토리지 서비스 개발자 안내서의 액세스 제어 목록 (ACL) 개요를 참조하십시오. ACLs

### 기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 - 권한 경계는 ID 기반 정책이 IAM 엔티티 (IAM사용자 또는 역할) 에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 사용 IAM설명서의 IAM 엔티티의 권한 경계를 참조하십시오.
- 서비스 제어 정책 (SCPs) - SCPs 조직 또는 OU (조직 구성 단위) 에 대한 최대 권한을 지정하는 JSON AWS Organizations정책입니다. AWS Organizations 기업이 소유한 여러 AWS 계정 개를 그

룹화하고 중앙에서 관리하는 서비스입니다. 조직의 모든 기능을 사용하도록 설정하면 일부 또는 모든 계정에 서비스 제어 정책 (SCPs) 을 적용할 수 있습니다. 각 항목을 포함하여 구성원 계정의 엔티티에 대한 권한을 SCP AWS 계정 루트 사용자제한합니다. Organizations 및 SCPs 에 대한 자세한 내용은 AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하십시오.

- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 보안 인증 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 IAM사용 설명서의 세션 정책을 참조하십시오.

### 여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련된 경우 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM사용 설명서의 정책 평가 로직을 참조하십시오.

## Amazon Location Service와 함께 작동하는 방식 IAM

Amazon IAM Location을 사용하여 액세스를 관리하기 전에 Amazon Location에서 사용할 수 있는 IAM 기능에 대해 알아보십시오.

IAMAmazon Location Service와 함께 사용할 수 있는 기능

| IAM기능: | Amazon Location 지원 |
|---|---|
| ID 기반 정책 | 예 |
| 리소스 기반 정책 | 아니요 |
| 정책 작업 | 예 |
| 정책 리소스 | 예 |
| 정책 조건 키(서비스별) | 예 |
| ACLs | 아니요 |
| ABAC(정책의 태그) | 예 |

| IAM기능: | Amazon Location 지원 |
| --- | --- |
| 임시 보안 인증 | 예 |
| 보안 주체 권한 | 아니요 |
| 서비스 역할 | 아니요 |
| 서비스 연결 역할 | 아니요 |

Amazon Location 및 기타 AWS 서비스가 대부분의 IAM 기능과 어떻게 작동하는지 자세히 알아보려면 IAM사용 IAM 설명서에서 함께 작동하는AWS 서비스를 참조하십시오.

## Amazon Location의 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 사용 설명서의 IAM정책 생성을 참조하십시오. IAM

IAMID 기반 정책을 사용하면 허용 또는 거부된 작업 및 리소스는 물론 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 사용 IAM설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

Amazon Location의 자격 증명 기반 정책 예

Amazon Location 자격 증명 기반 정책 예제를 보려면 Amazon Location Service의 자격 증명 기반 정책 예제 섹션을 참조하세요.

## Amazon Location 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스

의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

계정 간 액세스를 활성화하려면 다른 계정의 전체 계정 또는 IAM 엔티티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM사용 설명서의 계정 간 리소스 액세스를 참조하십시오. IAM

## Amazon Location의 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

정책 `Action` 요소는 JSON 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 작업이 없는 권한 전용 작업과 같은 몇 가지 예외가 있습니다. API 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

Amazon Location 작업 목록을 보려면 서비스 승인 참조의 Amazon Location Service에서 정의한 작업을 참조하세요.

Amazon Location의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
geo
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "geo:action1",
      "geo:action2"
         ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Get(이)라는 단어로 시작하는 모든 작업을 지정하려면 다음 태스크를 포함합니다.

```
"Action": "geo:Get*"
```

Amazon Location 자격 증명 기반 정책 예제를 보려면 Amazon Location Service의 자격 증명 기반 정책 예제 섹션을 참조하세요.

## Amazon Location의 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

ResourceJSON정책 요소는 작업이 적용되는 하나 또는 여러 개의 객체를 지정합니다. 문장에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. Amazon 리소스 이름 (ARN) 을 사용하여 리소스를 지정하는 것이 가장 좋습니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon 위치 리소스 유형 및 해당 ARNs 유형의 목록을 보려면 서비스 권한 부여 참조의 Amazon Location Service에서 정의한 리소스를 참조하십시오. 각 리소스에 어떤 작업을 지정할 수 있는지 알아보려면 Amazon Location Service에서 정의하는 작업을 참조하십시오. ARN

Amazon Location 자격 증명 기반 정책 예제를 보려면 Amazon Location Service의 자격 증명 기반 정책 예제 섹션을 참조하세요.

## Amazon Location에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어 리소스에 IAM 사용자 이름이 태그가 지정된 경우에만 리소스에 대한 액세스 권한을 IAM 사용자에게 부여할 수 있습니다. 자세한 내용은 IAM사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하십시오.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM사용 설명서의AWS 글로벌 조건 컨텍스트 키를 참조하십시오.

Amazon Location 조건 키 목록을 보려면 서비스 승인 참조의 Amazon Location Service에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Location Service에서 정의한 작업을 참조하세요.

Amazon Location은 정책 설명에서 특정 지오펜스 또는 디바이스에 대한 액세스를 허용하거나 거부할 수 있는 조건 키를 지원합니다. 다음 조건 키를 사용할 수 있습니다.

- 지오펜스 작업과 함께 `geo:GeofenceIds`을(를) 사용합니다. 유형은 `ArrayOfString`입니다.
- 트래커 작업과 함께 `geo:DeviceIds`을(를) 사용합니다. 유형은 `ArrayOfString`입니다.

`geo:GeofenceIds`IAM정책에서 다음 작업을 함께 사용할 수 있습니다.

- `BatchDeleteGeofences`
- `BatchPutGeofences`
- `GetGeofence`
- `PutGeofence`

다음 작업을 IAM 정책과 함께 `geo:DeviceIds` 사용할 수 있습니다.

- `BatchDeleteDevicePositionHistory`
- `BatchGetDevicePosition`
- `BatchUpdateDevicePosition`
- `GetDevicePosition`
- `GetDevicePositionHistory`

> **Note**
>
> 이러한 조건 키는 `BatchEvaluateGeofencesListGeofences`, 또는 `ListDevicePosition` 작업과 함께 사용할 수 없습니다.

Amazon Location 자격 증명 기반 정책 예제를 보려면 Amazon Location Service의 자격 증명 기반 정책 예제 섹션을 참조하세요.

## ACLs아마존 위치에서

지원ACLs: 아니요

액세스 제어 목록 (ACLs) 은 리소스에 액세스할 수 있는 권한을 가진 주체 (계정 구성원, 사용자 또는 역할) 를 제어합니다. ACLs정책 문서 형식을 사용하지는 않지만 리소스 기반 정책과 JSON 비슷합니다.

## ABAC아마존 위치 포함

지원 ABAC (정책의 태그): 예

속성 기반 액세스 제어 (ABAC) 는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM엔티티 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. 의 ABAC 첫 번째 단계는 엔티티와 리소스에 태그를 지정하는 것입니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC빠르게 성장하는 환경에서 유용하며 정책 관리가 복잡해지는 상황에도 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

에 대한 자세한 내용은 What is ABAC? 를 참조하십시오. ABAC IAM사용 설명서에서. 설정 ABAC 단계가 포함된 자습서를 보려면 사용 IAM설명서의 속성 기반 액세스 제어 사용 (ABAC) 을 참조하십시오.

Amazon Location 리소스 태그 지정에 대한 자세한 내용은 Amazon Location Service 리소스 태그 지정 섹션을 참조하세요.

리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예시는 태그를 기반으로 리소스에 대한 액세스 제어에서 확인할 수 있습니다.

## Amazon Location에서 임시 보안 인증 사용

임시 자격 증명 지원: 예

임시 자격 증명을 사용하여 로그인하면 작동하지 AWS 서비스 않는 것도 있습니다. 임시 자격 증명을 사용하는 AWS 서비스 방법을 비롯한 추가 정보는 IAM사용 설명서의 AWS 서비스 해당 자격 증명을 참조하십시오. IAM

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하는 경우 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 Single Sign-On (SSO) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM사용 설명서의 역할 전환 (콘솔) 을 참조하십시오.

AWS CLI 또는 를 사용하여 임시 자격 증명을 수동으로 생성할 수 AWS API 있습니다. 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 내용은 의 임시 보안 자격 증명을 참조하십시오. IAM

## Amazon Location에 대한 교차 서비스 보안 주체 권한

정방향 액세스 세션 지원 (FAS): 아니요

에서 IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 사용자는 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS를 호출하는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. AWS 서비스 FAS요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS요청 시 적용되는 정책 세부 정보는 전달 액세스 세션을 참조하십시오.

### Amazon Location의 서비스 역할

서비스 역할 지원: 아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 간주하는 IAM역할입니다. IAM관리자는 내부에서 IAM 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 사용 설명서의 역할 만들기를 참조하여 권한을 위임하십시오IAM. AWS 서비스

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon Location 기능이 중단될 수 있습니다. Amazon Location에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

### Amazon Location의 서비스 연결 역할

서비스 링크 역할 지원: 아니요

서비스 연결 역할은 에 연결된 서비스 역할 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

서비스 연결 역할을 만들거나 관리하는 방법에 대한 자세한 내용은 함께 작동하는 AWS 서비스를 참조하십시오. IAM 서비스 연결 역할 열에서 Yes(이)가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

## Amazon Location Service가 인증되지 않은 사용자를 처리하는 방법

웹 또는 모바일 애플리케이션에 지도를 표시하는 것을 포함하여 Amazon Location Service를 사용하기 위한 많은 시나리오에서는 로그인하지 않은 사용자에게 액세스를 허용해야 IAM 합니다. 이러한 인증되지 않은 시나리오의 경우, 두 가지 옵션이 있습니다.

- APIKI 사용 — 인증되지 않은 사용자에게 액세스 권한을 부여하려면 Amazon Location Service 리소스에 대한 읽기 전용 액세스 권한을 부여하는 API 키를 생성할 수 있습니다. 이는 모든 사용자를 인증하고 싶지 않은 경우에 유용합니다. 웹 애플리케이션을 예로 들 수 있습니다. APIKI에 대한 자세한 내용은 을 참조하십시오. API 키를 사용하여 인증되지 않은 게스트의 애플리케이션 액세스 허용하기
- Amazon Cognito 사용 — API 키의 대안은 Amazon Cognito를 사용하여 익명 액세스를 허용하는 것입니다. Amazon Cognito를 사용하면 인증되지 않은 사용자가 수행할 수 있는 작업을 정의하는 IAM

정책을 통해 보다 풍부한 권한 부여를 생성할 수 있습니다. Amazon Cognito 사용에 대한 자세한 내용은 Amazon Cognito를 사용하여 미인증 게스트의 애플리케이션 액세스 허용하기을 참조하세요.

인증되지 않은 사용자에게 액세스를 제공하는 방법에 대한 개요는 Amazon Location Service에 액세스 권한 부여 섹션을 참조하세요.

## Amazon Location Service의 자격 증명 기반 정책 예제

기본적으로 사용자와 역할에는 Amazon Location 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface ()AWS CLI또는 를 사용하여 작업을 수행할 수 없습니다. AWS API IAM관리자는 IAM 정책을 생성하여 필요한 리소스에서 작업을 수행할 수 있는 권한을 사용자에게 부여할 수 있습니다. 그러면 관리자가 역할에 IAM 정책을 추가할 수 있으며, 사용자는 역할을 수임할 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 만드는 방법을 알아보려면 사용 IAM설명서에서 IAM 정책 생성을 참조하십시오.

각 리소스 유형의 형식을 비롯하여 Amazon Location에서 정의한 작업 및 리소스 유형에 ARNs 대한 자세한 내용은 서비스 인증 참조의 Amazon Location Service용 작업, 리소스 및 조건 키를 참조하십시오.

주제

- 정책 모범 사례
- Amazon Location 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 정책에서 Amazon Location Service 리소스 사용
- 디바이스 위치 업데이트 권한
- 트래커 리소스에 대한 읽기 전용 정책
- 지오펜스 생성 정책
- 지오펜스에 대한 읽기 전용 정책
- 맵 리소스를 렌더링하기 위한 권한
- 검색 작업을 허용하는 권한
- 경로 계산기에 대한 읽기 전용 정책
- 조건 키를 기반으로 리소스 액세스 제어
- 태그를 기반으로 리소스에 대한 액세스 제어

## 정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 Amazon Location 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하고 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하십시오. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 내용은 IAM사용 설명서의 AWS 관리형 정책 또는 작업 기능에 대한AWS 관리형 정책을 참조하십시오.
- 최소 권한 적용 — IAM 정책으로 권한을 설정하는 경우 작업 수행에 필요한 권한만 부여하십시오. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. 를 사용하여 권한을 IAM 적용하는 방법에 대한 자세한 내용은 사용 설명서의 정책 및 권한을 참조하십시오. IAM IAM
- IAM정책의 조건을 사용하여 액세스를 추가로 제한 - 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, 를 사용하여 모든 요청을 전송하도록 지정하는 정책 조건을 작성할 수 SSL 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 내용은 IAM사용 설명서의 IAMJSON정책 요소: 조건을 참조하십시오.
- IAMAccess Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장합니다. IAM Access Analyzer는 새 정책과 기존 정책을 검증하여 정책이 IAM 정책 언어 (JSON) 및 IAM 모범 사례를 준수하는지 확인합니다. IAMAccess Analyzer는 안전하고 기능적인 정책을 작성하는 데 도움이 되는 100개 이상의 정책 검사와 실행 가능한 권장 사항을 제공합니다. 자세한 내용은 사용 설명서의 IAMAccess Analyzer 정책 검증을 참조하십시오. IAM
- 다단계 인증 필요 (MFA) - 사용자 또는 루트 IAM 사용자가 필요한 시나리오가 있는 경우 보안을 강화하려면 이 기능을 MFA 켜십시오. AWS 계정 API작업 호출 MFA 시기를 요구하려면 정책에 MFA 조건을 추가하세요. 자세한 내용은 IAM사용 설명서의 MFA -보호된 API 액세스 구성을 참조하십시오.

의 모범 사례에 IAM 대한 자세한 내용은 IAM사용 설명서의 보안 모범 사례를 참조하십시오. IAM

## Amazon Location 콘솔 사용

Amazon Location Service 콘솔에 액세스하려면 최소한의 권한 집합이 있어야 합니다. 이러한 권한을 통해 내 Amazon Location 리소스를 나열하고 세부 정보를 볼 수 있어야 AWS 계정합니다. 최소 필수

권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔터티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 에만 전화를 거는 사용자에게 최소 콘솔 권한을 허용할 필요는 AWS API 없습니다. 대신 수행하려는 작업과 일치하는 API 작업에만 액세스를 허용하세요.

사용자와 역할이 Amazon Location 콘솔을 사용할 수 있도록 하려면 엔터티에 다음 정책을 연결합니다. 자세한 내용은 사용 설명서의 IAM사용자에게 권한 추가를 참조하십시오.

다음 정책은 Amazon Location Service 콘솔에 대한 액세스 권한을 부여하여 AWS 계정의 Amazon Location 리소스에 대한 세부 정보를 생성, 삭제, 나열 및 확인할 수 있도록 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GeoPowerUser",
      "Effect": "Allow",
      "Action": [
        "geo:*"
      ],
      "Resource": "*"
    }
  ]
}
```

읽기 전용 권한을 부여하여 읽기 전용 액세스를 용이하게 할 수도 있습니다. 읽기 전용 권한을 사용하는 경우 사용자가 리소스 생성 또는 삭제와 같은 쓰기 작업을 시도하면 오류 메시지가 표시됩니다. 예시는 the section called “트래커의 읽기 전용 정책”에서 확인하세요.

## 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제에서는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 OR를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 AWS CLI 권한이 포함됩니다. AWS API

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

## 정책에서 Amazon Location Service 리소스 사용

Amazon Location Service는 리소스에 다음과 같은 접두사를 사용합니다.

Amazon Location 리소스 접두사

| Resource | 리소스 접두사 |
| --- | --- |
| 맵 리소스 | `map` |
| 장소 리소스 | `place-index` |
| 경로 리소스 | `route-calculator` |
| 트래킹 리소스 | `tracker` |

| Resource | 리소스 접두사 |
| --- | --- |
| 지오펜스 컬렉션 리소스 | geofence-collection |

다음 ARN 구문을 사용하십시오.

```
arn:Partition:geo:Region:Account:ResourcePrefix/ResourceName
```

형식에 대한 자세한 내용은 Amazon 리소스 이름 (ARNs) 및 AWS 서비스 네임스페이스를 참조하십시오. ARNs

예제

- 다음을 사용하여 지정된 맵 ARN 리소스에 대한 액세스를 허용하십시오.

```
"Resource": "arn:aws:geo:us-west-2:account-id:map/map-resource-name"
```

- 특정 계정에 속하는 모든 map 리소스에 대한 액세스를 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:geo:us-west-2:account-id:map/*"
```

- 리소스 생성 작업과 같은 일부 Amazon Location 작업은 특정 리소스에서 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

```
"Resource": "*"
```

Amazon 위치 리소스 유형 및 해당 ARNs 유형의 목록을 보려면 서비스 권한 부여 참조의 Amazon Location Service에서 정의한 리소스를 참조하십시오. 각 리소스에 어떤 작업을 지정할 수 있는지 알아보려면 Amazon Location Service에서 정의하는 작업을 참조하십시오. ARN

## 디바이스 위치 업데이트 권한

여러 트래커의 디바이스 위치를 업데이트하려면 사용자에게 하나 이상의 트래커 리소스에 대한 액세스 권한을 부여해야 합니다. 또한 사용자가 디바이스 위치를 일괄 업데이트할 수 있도록 허용해야 합니다.

이 예제에서는 액세스 권한을 부여하는 것 외에도 *Tracker1* 그리고 *Tracker2* 다음 정책은 리소스에 대한 geo:BatchUpdateDevicePosition 작업 사용 권한을 부여합니다.*Tracker1* 그리고 *Tracker2* 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": [
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker1",
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker2"
      ]
    }
  ]
}
```

사용자가 특정 디바이스의 디바이스 위치만 업데이트할 수 있도록 제한하려면 해당 디바이스 ID에 조건 키를 추가하면 됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": [
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker1",
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker2"
      ],
      "Condition":{
        "ForAllValues:StringLike":{
          "geo:DeviceIds":[
            "deviceId"
          ]
```

```
        }
      }
    }
  ]
}
```

## 트래커 리소스에 대한 읽기 전용 정책

AWS 계정의 모든 트래커 리소스에 대한 읽기 전용 정책을 만들려면 모든 트래커 리소스에 대한 액세스 권한을 부여해야 합니다. 또한 사용자에게 여러 디바이스의 위치를 가져오고, 단일 디바이스에서 디바이스 위치를 가져오고, 위치 기록을 가져오는 작업에 대한 액세스 권한을 부여할 수도 있습니다.

이 예에서 다음 정책은 다음 작업에 대한 권한을 부여합니다.

- 여러 디바이스의 위치를 검색할 `geo:BatchGetDevicePosition` 권한.
- 단일 디바이스의 위치를 검색할 `geo:GetDevicePosition` 권한.
- 한 디바이스의 위치 기록을 검색할 `geo:GetDevicePositionHistory` 권한.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchGetDevicePosition",
        "geo:GetDevicePosition",
        "geo:GetDevicePositionHistory"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:tracker/*"
    }
  ]
}
```

## 지오펜스 생성 정책

사용자가 지오펜스를 생성할 수 있도록 허용하는 정책을 만들려면 사용자가 지오펜스 컬렉션에 하나 이상의 지오펜스를 만들 수 있도록 허용하는 특정 작업에 대한 액세스 권한을 부여해야 합니다.

아래 정책은 다음과 같은 작업에 권한을 부여합니다.*Collection*:

- 여러 지오펜스를 생성하는 geo:BatchPutGeofence.
- 단일 지오펜스를 생성하는 geo:PutGeofence.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateGeofences",
      "Effect": "Allow",
      "Action": [
        "geo:BatchPutGeofence",
        "geo:PutGeofence"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:geofence-collection/Collection"
    }
  ]
}
```

## 지오펜스에 대한 읽기 전용 정책

AWS 계정의 지오펜스 컬렉션에 저장된 지오펜스에 대한 읽기 전용 정책을 생성하려면 지오펜스를 저장하는 지오펜스 컬렉션에서 읽기 작업에 대한 액세스 권한을 부여해야 합니다.

아래 정책은 다음과 같은 작업에 권한을 부여합니다.*Collection*:

- 지정된 지오펜스 컬렉션의 지오펜스를 나열할 geo:ListGeofences 권한.
- 지오펜스 컬렉션에서 지오펜스를 검색할 geo:GetGeofence 권한.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetGeofences",
      "Effect": "Allow",
      "Action": [
        "geo:ListGeofences",
        "geo:GetGeofence"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:geofence-collection/Collection"
    }
```

```
  ]
}
```

## 맵 리소스를 렌더링하기 위한 권한

맵을 렌더링할 수 있는 충분한 권한을 부여하려면 맵 타일, 스프라이트, 글리프, 스타일 설명자에 대한 액세스 권한을 부여해야 합니다.

- `geo:GetMapTile`은(는) 맵에서 지형지물을 선택적으로 렌더링하는 데 사용되는 맵 타일을 검색합니다.
- `geo:GetMapSprites`PNG스프라이트 시트와 그 안의 오프셋을 설명하는 해당 JSON 문서를 검색합니다.
- `geo:GetMapGlyphs`은(는) 텍스트를 표시하는 데 사용되는 글리프를 검색합니다.
- `geo:GetMapStyleDescriptor`은(는) 렌더링 규칙이 포함된 맵의 스타일 설명자를 검색합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetTiles",
      "Effect": "Allow",
      "Action": [
        "geo:GetMapTile",
        "geo:GetMapSprites",
        "geo:GetMapGlyphs",
        "geo:GetMapStyleDescriptor"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:map/Map"
    }
  ]
}
```

## 검색 작업을 허용하는 권한

검색 작업을 허용하는 정책을 만들려면 먼저 계정의 플레이스 인덱스 리소스에 대한 액세스 권한을 부여해야 합니다. AWS 또한 사용자가 지오코딩을 통해 텍스트를 사용하여 검색하고 역지오코딩으로 위치를 사용하여 검색할 수 있는 작업에 대한 액세스 권한을 부여할 수도 있습니다.

이 예시에서는 액세스 권한을 부여하는 것 외에도 *PlaceIndex*다음 정책은 다음 작업에 대한 권한도 부여합니다.

- geo:SearchPlaceIndexForPosition(으)로 특정 위치 근처의 장소나 관심 장소를 검색할 수 있습니다.
- geo:SearchPlaceIndexForText(으)로 자유 형식 텍스트를 사용하여 주소, 이름, 도시 또는 리전을 검색할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Search",
      "Effect": "Allow",
      "Action": [
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:place-index/PlaceIndex"
    }
  ]
}
```

### 경로 계산기에 대한 읽기 전용 정책

사용자가 경로 계산기 리소스에 액세스하여 경로를 계산하도록 허용하는 읽기 전용 정책을 만들 수 있습니다.

이 예에서는 액세스 권한을 부여하는 것 외에도 *ExampleCalculator*다음 정책은 다음 작업에 권한을 부여합니다.

- geo:CalculateRoute은(는) 출발 위치, 목적지 위치, 웨이포인트 위치 목록을 고려하여 경로를 계산합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
```

```
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:us-west-2:accountID:route-calculator/ExampleCalculator"
    }
  ]
}
```

## 조건 키를 기반으로 리소스 액세스 제어

지오펜스 또는 장치 위치를 사용할 수 있는 액세스 권한을 부여하는 IAM 정책을 생성할 때 조건 연산자를 사용하여 사용자가 액세스할 수 있는 지오펜스 또는 장치를 보다 정밀하게 제어할 수 있습니다. 정책의 `Condition` 요소에 지오펜스 ID 또는 디바이스 ID를 포함하여 이를 수행할 수 있습니다.

다음 예제 정책에서는 사용자가 특정 디바이스의 디바이스 위치를 업데이트하도록 허용하는 정책을 생성하는 방법을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": [
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker"
      ],
      "Condition":{
        "ForAllValues:StringLike":{
          "geo:DeviceIds":[
            "deviceId"
          ]
        }
      }
    }
  ]
}
```

## 태그를 기반으로 리소스에 대한 액세스 제어

Amazon Location 리소스 사용에 대한 액세스 권한을 부여하는 IAM 정책을 생성할 때 속성 기반 액세스 제어를 사용하여 사용자가 수정, 사용 또는 삭제할 수 있는 리소스를 더 잘 제어할 수 있습니다. 정책의 `Condition` 요소에 태그 정보를 포함하여 리소스 태그를 기반으로 액세스를 제어하면 됩니다.

다음 예제 정책은 사용자가 지오펜스를 생성하도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이렇게 하면 라는 지오펜스 컬렉션에 하나 이상의 지오펜스를 생성할 수 있는 다음 작업에 대한 권한이 부여됩니다.*`Collection`*:

- 여러 지오펜스를 생성하는 `geo:BatchPutGeofence`.
- 단일 지오펜스를 생성하는 `geo:PutGeofence`.

하지만 이 정책은 다음과 같은 경우에만 `Condition` 요소를 사용하여 권한을 부여합니다.*`Collection`* `Owner`, 태그는 해당 사용자의 사용자 이름 값을 가집니다.

- 예를 들어, 라는 사용자가 Amazon 위치를 `richard-roe` 보려고 시도하는 경우 *`Collection`*,*`Collection`* `Owner=richard-roe`또는 `owner=richard-roe` 태그가 지정되어야 합니다. 그렇지 않으면 사용자는 액세스가 거부됩니다.

> **Note**
>
> 조건 키 `Owner`은(는) 조건 키 이름이 대소문자를 구분하지 않기 때문에 `Owner` 및 `owner` 모두와 일치합니다. 자세한 내용은 IAM사용 설명서의 IAM JSON 정책 요소: 조건을 참조하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateGeofencesIfOwner",
      "Effect": "Allow",
      "Action": [
        "geo:BatchPutGeofence",
        "geo:PutGeofence"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:geofence-collection/Collection",
      "Condition": {
```

```
                "StringEquals": {"geo:ResourceTag/Owner": "${aws:username}"}
        }
      }
    ]
}
```

태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한을 정의하는 방법에 대한 자습서는 사용 AWS Identity and Access Management 설명서를 참조하십시오.

## Amazon Location Service 자격 증명 및 액세스 문제 해결

다음 정보를 사용하면 Amazon Location 및 작업 시 발생할 수 있는 일반적인 문제를 진단하고 해결하는 데 도움이 IAM 됩니다.

주제

- Amazon Location에서 작업을 수행할 권한이 없음
- 저는 IAM을 수행할 권한이 없습니다. PassRole
- 외부 사용자가 내 Amazon Location 리소스에 액세스할 AWS 계정 수 있도록 허용하고 싶습니다.

### Amazon Location에서 작업을 수행할 권한이 없음

작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 권한이 없는 경우 발생합니다. geo:*GetWidget*

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 geo:GetWidget on resource: my-example-widget
```

이 경우 geo:*GetWidget*작업을 사용하여 *my-example-widget*리소스에 액세스할 수 있도록 mateojackson사용자 정책을 업데이트해야 합니다.

도움이 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

### 저는 IAM을 수행할 권한이 없습니다. PassRole

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon Location에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 Amazon Location에서 콘솔을 사용하여 작업을 `marymajor` 수행하려고 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하십시오. 관리자는 로그인 자격 증명을 제공한 사람입니다.

## 외부 사용자가 내 Amazon Location 리소스에 액세스할 AWS 계정 수 있도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록 (ACLs) 을 지원하는 서비스의 경우 해당 정책을 사용하여 사용자에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- Amazon Location에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon Location Service와 함께 작동하는 방식 IAM 섹션을 참조하세요.
- 소유한 리소스에 대한 액세스 권한을 AWS 계정 부여하는 방법을 알아보려면 사용 설명서에서 소유한 다른 IAM AWS 계정 사용자의 액세스 권한 제공을 IAM 참조하십시오.
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM사용 설명서의 제3자가 AWS 계정 소유한 리소스에 대한 액세스 제공을 참조하십시오. AWS 계정
- ID 페더레이션을 통해 액세스를 제공하는 방법을 알아보려면 사용 설명서의 외부 인증된 사용자에게 액세스 제공 (ID 페더레이션) 을 IAM 참조하십시오.
- 계정 간 액세스에 대한 역할 사용과 리소스 기반 정책의 차이점을 알아보려면 사용 설명서의 계정 간 리소스 액세스를 참조하십시오. IAM IAM

# Amazon Location Service의 사고 대응

AWS에서는 보안을 가장 중요하게 생각합니다. AWS 클라우드 공동 책임 모델의 일환으로 가장 보안에 민감한 조직의 요구 사항을 충족하는 데이터 센터 및 네트워크 아키텍처를 AWS 관리합니다. AWS 고객은 클라우드에서 보안을 유지할 책임이 있습니다. 즉, 액세스 가능한 AWS 도구 및 기능을 통해 구현하기로 선택한 보안을 제어할 수 있습니다.

클라우드에서 실행되는 애플리케이션의 목표를 충족하는 보안 기준을 설정하면 대응할 수 있는 편차를 감지할 수 있습니다. 보안 사고 대응은 복잡한 주제일 수 있으므로 IR (사고 대응) 과 선택이 기업 목표에 미치는 영향을 더 잘 이해할 수 있도록 AWS보안 사고 대응 가이드, AWS보안 모범 사례 백서, AWS클라우드 채택 프레임워크 (AWSCAF) 와 같은 리소스를 검토하는 것이 좋습니다.

## Amazon Location Service의 로깅 및 모니터링

로깅 및 모니터링은 사고 대응의 중요한 부분입니다. 이를 통해 편차를 탐지하고 조사하고 대응할 수 있는 보안 기준을 설정할 수 있습니다. Amazon Location Service의 로깅 및 모니터링을 구현하면 프로젝트와 리소스의 안정성, 가용성 및 성능을 유지할 수 있습니다.

AWS 사고 대응을 위한 데이터를 기록하고 수집하는 데 도움이 되는 몇 가지 도구를 제공합니다.

AWS CloudTrail

Amazon Location Service는 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 기록을 제공하는 AWS 서비스인 와 통합됩니다. 여기에는 Amazon 위치 서비스 콘솔에서의 작업과 Amazon 위치 API 작업에 대한 프로그래밍 방식 호출이 포함됩니다. 이러한 작업 기록을 이벤트라고 합니다. 자세한 내용은 Amazon Location Service를 사용한 로깅 및 모니터링을 참조하십시오 AWS CloudTrail.

아마존 CloudWatch

CloudWatch Amazon을 사용하여 Amazon Location Service 계정과 관련된 지표를 수집하고 분석할 수 있습니다. 지표가 특정 조건을 충족하고 지정된 임계값에 도달하면 CloudWatch 경보를 활성화하여 알림을 받을 수 있습니다. 경보를 생성하면 사용자가 정의한 Amazon 단순 알림 서비스에 알림을 CloudWatch 보냅니다. 자세한 내용은 Amazon을 통한 Amazon 위치 모니터링 서비스를 참조하십시오 CloudWatch.

AWS Health 대시보드

AWS Health 대시보드를 사용하여 Amazon Location Service 서비스의 상태를 확인할 수 있습니다. 또한 AWS 환경에 영향을 미칠 수 있는 모든 이벤트 또는 문제에 대한 기간별 데이터를 모니터링하고 볼 수 있습니다. 자세한 내용은 AWS Health 사용 설명서를 참조하십시오.

# Amazon Location Service에 대한 규정 준수 확인

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의AWS 서비스 범위별, 규정 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 적합한 애플리케이션을 만드는 AWS HIPAA 데 사용할 수 있는 방법을 설명합니다.

  > **Note**
  >
  > 모든 AWS 서비스 사람이 자격이 있는 것은 아닙니다. HIPAA 자세한 내용은 HIPAA적격 서비스 참조를 참조하십시오.

- AWS 규정AWS 준수 리소스 — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (국립 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (), 국제 표준화 기구 ()) 를 포함한PCI) 전반의 보안 제어에 대한 지침을 매핑합니다. ISO
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.
- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수

프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하는 PCI DSS 등 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.

- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

## Amazon Location Service의 복원성

AWS 글로벌 인프라는 가용 영역을 중심으로 구축됩니다 AWS 리전 . AWS 리전 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다. 이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

가용 영역에 대한 AWS 리전 자세한 내용은 글로벌 인프라를 참조하십시오AWS .

Amazon Location은 AWS 글로벌 인프라 외에도 데이터 복원력 및 백업 요구 사항을 지원하는 데 도움이 되는 여러 기능을 제공합니다.

## Amazon Location Service의 인프라 보안

관리형 서비스인 Amazon Location Service는 AWS 글로벌 네트워크 보안의 보호를 받습니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 통화를 사용하여 네트워크를 통해 Amazon Location에 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안 (TLS). TLS1.2가 필요하고 TLS 1.3을 권장합니다.
- (임시 디피-헬만) 또는 (타원 곡선 임시 디피-헬만PFS) 와 같이 완벽한 순방향 기밀성 DHE () 을 갖춘 암호 제품군. ECDHE Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 액세스 키 ID와 보안 주체와 연결된 비밀 액세스 키를 사용하여 요청에 서명해야 합니다. IAM 또는 AWS Security Token Service(AWS STS)을(를) 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

# Amazon Location의 구성 및 취약성 분석

구성 및 IT 제어는 귀하와 당사 고객 간의 AWS 공동 책임입니다. 자세한 내용은 AWS 공동 책임 모델을 참조하십시오.

# 교차 서비스 혼동된 대리자 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS서비스 간 사칭은 대리인 문제로 혼란스러운 결과를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

Amazon Location Service는 사용자를 대신하여 다른 AWS 서비스를 호출하는 서비스 역할을 하지 않으므로 이 경우 이러한 보호 기능을 추가할 필요가 없습니다. 혼동된 대리자 문제에 대해 자세히 알아보려면, AWS Identity and Access Management 사용 설명서의 혼동된 대리자 문제를 참조하세요.

# Amazon Location Service의 보안 모범 사례

Amazon Location Service는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주세요.

## Amazon Location Service의 탐지 보안 모범 사례

다음과 같은 Amazon Location Service 모범 사례는 보안 사고를 탐지하는 데 도움이 됩니다.

AWS모니터링 도구 구현

모니터링은 사고 대응에 매우 중요하며 Amazon Location Service 리소스 및 솔루션의 안정성과 보안을 유지합니다. 사용 가능한 여러 도구 및 서비스 중에서 모니터링 도구를 AWS 구현하여 리소스 및 기타 AWS 서비스를 모니터링할 수 있습니다.

예를 들어 CloudWatch Amazon에서는 Amazon Location Service의 지표를 모니터링하고 지표가 사용자가 설정한 특정 조건을 충족하고 사용자가 정의한 임계값에 도달하면 알림을 받도록 경보

를 설정할 수 있습니다. 경보를 생성할 때 Amazon Simple Notification Service를 사용하여 알림을 CloudWatch 보내도록 설정할 수 있습니다. 자세한 내용은 the section called “로깅 및 모니터링” 단원을 참조하십시오.

AWS로깅 도구를 활성화합니다.

로깅은 Amazon Location Service에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 기록을 제공합니다. 작업에 대한 데이터를 AWS CloudTrail 수집하여 비정상적인 API 활동을 탐지하는 등의 로깅 도구를 구현할 수 있습니다.

트레일을 만들 때 이벤트를 CloudTrail 기록하도록 구성할 수 있습니다. 이벤트는 리소스에서 또는 리소스 내에서 수행되는 리소스 작업의 기록으로서 Amazon Location에 대한 요청, 요청이 이루어진 IP 주소, 요청한 사람, 요청이 이루어진 시간 및 추가 데이터가 포함됩니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 트레일에 대한 데이터 이벤트 로깅을 참조하십시오.

## Amazon Location Service의 예방적 보안 모범 사례

다음과 같은 Amazon Location Service 모범 사례를 통해 보안 사고를 예방할 수 있습니다.

보안 연결 사용

전송 중에 민감한 정보를 안전하게 유지하기 위해 항상 암호화된 연결(예: `https://`(으)로 시작하는 연결)을 사용하세요.

리소스에 대한 최소 권한 액세스 구현

Amazon Location 리소스에 대한 사용자 지정 정책을 생성하는 경우 작업을 수행하는 데 필요한 권한만 부여하세요. 최소한의 권한으로 시작하고 필요에 따라 추가 권한을 부여하는 것이 좋습니다. 오류나 악의적인 공격으로 인해 발생할 수 있는 위험과 영향을 줄이려면 최소 권한 액세스를 구현하는 것이 필수적입니다. 자세한 내용은 the section called “ID 및 액세스 관리” 단원을 참조하십시오.

전 세계적으로 고유한 디바이스로 IDs 사용 IDs

장치에는 다음 규칙을 사용하십시오. IDs

- 장치는 IDs 고유해야 합니다.
- 다른 시스템의 외부 키로 사용될 수 있으므로 장치를 비밀로 IDs 설정해서는 안 됩니다.
- 기기는 전화, 기기, 이메일 주소 등 개인 식별이 가능한 정보 (PII) 를 IDs 포함해서는 안 됩니다. IDs

- 기기는 예측할 수 IDs 없어야 합니다. 와 같은 불투명 식별자를 사용하는 UUIDs 것이 좋습니다.

기기 위치 PII 속성에 포함하지 마세요.

장치 업데이트를 보낼 때 (예: 사용 DevicePositionUpdate) 전화번호나 이메일 주소와 같은 개인 식별 정보 (PII) 를 에 포함하지 마십시오. `PositionProperties`

# Amazon Location Service의 모범 사례

이 항목에서는 Amazon Location Service를 사용하는 데 도움이 되는 모범 사례를 제공합니다. 이러한 모범 사례는 Amazon Location Service를 최대한 활용하는 데 도움이 될 수 있지만 완전한 솔루션은 아닙니다. 환경에 해당하는 권장 사항만 따라야 합니다.

주제

- 보안
- 리소스 관리
- 비용 및 청구 관리
- 할당량 및 사용량

## 보안

보안 위험을 관리하거나 방지하려면 다음 모범 사례를 고려하세요.

- 자격 증명 연동 및 IAM 역할을 사용하여 Amazon Location 리소스에 대한 액세스를 관리, 제어 또는 제한할 수 있습니다. 자세한 내용은 IAM사용 설명서의 IAM모범 사례를 참조하십시오.
- 최소 권한 원칙에 따라 Amazon Location Service 리소스에 필요한 최소 액세스 권한만 부여하세요. 자세한 내용은 the section called “정책을 사용한 액세스 관리” 단원을 참조하십시오.
- 웹 애플리케이션에 사용되는 Amazon Location Service 리소스의 경우 `aws:referer` IAM 조건을 사용하여 액세스를 제한하고 허용 목록에 포함된 사이트 이외의 사이트에서의 사용을 제한하십시오.
- 모니터링 및 로깅 도구를 사용하여 리소스 액세스 및 사용을 추적하세요. 자세한 내용은 사용 설명서의 트레일에 대한 데이터 이벤트 로깅을 참조하십시오the section called “로깅 및 모니터링”. AWS CloudTrail
- 보안 연결(예: `https://`로 시작)을 사용하여 보안을 강화하고 서버와 브라우저 간에 데이터가 전송되는 동안 공격으로부터 사용자를 보호하세요.

탐지 및 예방 보안 모범 사례에 대한 자세한 내용은 the section called “보안 모범 사례”의 항목을 참조하세요.

## 리소스 관리

Amazon Location Service에서 위치 리소스를 효과적으로 관리하려면 다음 모범 사례를 고려하세요.

- 예상 사용자 기반에서 중심이 되는 리전별 엔드포인트를 사용하여 사용자 경험을 개선하세요. 리전 엔드포인트에 대한 자세한 내용은 Amazon Location 리전 및 엔드포인트 섹션을 참조하세요.
- 맵 리소스 및 장소 색인 리소스와 같이 데이터 공급자를 사용하는 리소스의 경우 특정 데이터 공급자의 사용 약관을 준수해야 합니다. 자세한 정보는 데이터 공급자를 참조하세요.
- 맵, 장소 색인 또는 경로의 각 구성마다 하나의 리소스를 보유하여 리소스 생성을 최소화합니다. 리전 내에서는 일반적으로 데이터 공급자 또는 맵 스타일당 하나의 리소스만 필요합니다. 대부분의 애플리케이션은 기존 리소스를 사용하며 런타임 시 리소스를 생성하지 않습니다.
- 맵 리소스 및 경로 계산기와 같은 단일 애플리케이션에서 서로 다른 리소스를 사용하는 경우 각 리소스에서 동일한 데이터 공급자를 사용하여 데이터가 일치하는지 확인하세요. 예를 들어, 경로 계산기로 생성한 경로 지오메트리는 맵 리소스를 사용하여 그린 맵의 거리와 일치합니다.

## 비용 및 청구 관리

비용 및 청구 관리에 도움이 되도록 다음 모범 사례를 고려하세요.

- CloudWatchAmazon과 같은 모니터링 도구를 사용하여 리소스 사용량을 추적하십시오. 사용량이 지정된 한도를 초과하려 할 때 알려주는 알림을 설정할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 예상 AWS 요금을 모니터링하기 위한 결제 경보 생성을 참조하십시오.

## 할당량 및 사용량

사용량의 기본 한도를 설정하는 할당량을 AWS 계정 포함합니다. 사용량이 한도에 가까워지면 알림을 받도록 알람을 설정하고 필요할 때 할당량 상향을 요청할 수 있습니다. 할당량 사용 방법에 대한 자세한 내용은 다음 항목을 참조하세요.

- Amazon Location Service Quotas
- 할당량 대비 사용량을 모니터링하는 CloudWatch 데 사용
- Amazon 사용 설명서에서 서비스 할당량을 시각화하고 경보를 설정합니다. CloudWatch

한도 초과에 가까워지면 미리 경고를 표시하는 알람을 생성할 수 있습니다. Amazon Location을 사용하는 각 위치에서 각 할당량에 대해 경보를 설정하는 것이 좋습니다. AWS 리전 예를 들어 `SearchPlaceIndexForText` 작업 사용을 모니터링하여 현재 할당량의 80%를 초과할 경우 알람을 생성할 수 있습니다.

할당량에 대한 알람이 표시되면 수행할 작업을 결정해야 합니다. 고객 기반이 늘어났기 때문에 추가 리소스를 사용하고 있을 수 있습니다. 이 경우 해당 지역의 API 통화 할당량을 50% 늘리는 등 할당량 증가를 요청할 수 있습니다. 또는 Amazon Location에 불필요한 추가 호출을 발생시키는 서비스 오류가 있을 수도 있습니다. 이 경우에는 서비스에서 문제를 해결해 보는 것이 좋습니다.

# 문서 이력

다음 표에서는 Amazon Location Service 관련 문서를 소개합니다. 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하시면 됩니다.

| 변경 사항 | 설명 | 날짜 |
| --- | --- | --- |
| Amazon Location Service는 다음을 위한 새로운 SDK를 출시했습니다. JavaScript | 웹 프런트 엔드에서 Amazon Location 애플리케이션을 더 쉽게 개발할 수 있도록 Amazon Location은 JavaScript v3용 AWS SDK를 지원하는 새로운 오픈 소스 SDK를 추가하여 인증을 간소화하고 GeoJSON을 사용합니다. 자세한 내용은 Amazon Location SDK를 참조하세요. | 2023년 7월 6일 |
| Amazon Location Service, API 키 정식 출시 | Amazon Location이 장소 및 경로에 대한 지원을 추가하고 API 키 기능을 정식 출시합니다. 자세한 내용은 API 키 사용을 참조하세요. | 2023년 7월 6일 |
| Amazon Location Service는 포지션 업데이트를 위한 Amazon EventBridge 이벤트를 추가합니다. | Amazon Location은 트래커 위치 업데이트 이벤트 전송을 위한 지원을 추가합니다 EventBridge. 트래커의 이벤트를 활성화하는 방법을 비롯한 자세한 내용은 이벤트 대응을 참조하십시오. EventBridge | 2023년 7월 6일 |
| Amazon Location, 지오펜스에 메타데이터 추가 | 이제 Amazon Location API를 사용하여 지오펜스에 메타데이터 속성을 추가할 수 있습니다. 이러한 정보는 지오펜스와 함께 저장되며 Amazon의 지오펜 | 2023년 6월 15일 |

| | | |
|---|---|---|
| | 스와 관련된 이벤트에 포함됩니다. EventBridge 자세한 내용은 지오펜스 그리기 및 이벤트 규칙 생성을 참조하세요. | |
| Amazon Location, 장소의 카테고리 추가 | Amazon Location은 장소 검색 결과에 카테고리를 추가하고 카테고리별로 결과를 필터링합니다. 자세한 내용은 카테고리 및 필터링을 참조하세요. | 2023년 6월 15일 |
| Amazon Location, 정치적 견해 도입 | Amazon Location이 특정 맵 스타일에 정치적 견해를 추가합니다. 자세한 내용은 정치적 견해를 참조하세요. | 2023년 5월 23일 |
| Amazon Location, 새로운 데모 및 샘플 사이트 소개 | Amazon Location이 Amazon Location 데모 및 샘플에 액세스할 수 있는 새로운 웹사이트를 발표합니다. 자세한 내용은 Amazon Location 데모 사이트를 참조하세요. | 2023년 5월 3일 |
| 아마존 로케이션은 더 긴 경로를 소개합니다. CalculateRouteMatrix | 이제 Amazon Location에서는 HERE 데이터 공급자를 통해 생성한 경로 매트릭스 경로에 대해 무제한 길이의 경로를 허용합니다. 자세한 내용은 장거리 경로 계획을 참조하세요. | 2023년 4월 24일 |
| Amazon Location 설명서에 데이터 공급자별 기능 차이 추가 | Amazon Location 설명서가 맵, 장소 검색, 라우팅에서 각 데이터 공급자 간의 차이점에 관한 정보로 업데이트되었습니다. 자세한 내용은 데이터 공급자별 기능을 참조하세요. | 2023년 3월 30일 |

| | | |
|---|---|---|
| Amazon Location 오픈 데이터 맵 정식 출시 | Amazon Location Service 데이터 공급자의 일반 사용 가능 여부 및 스타일 (일광 지도 기준 OpenStreetMap) 자세한 내용은 오픈 데이터를 참조하십시오. | 2023년 3월 7일 |
| Amazon Location, 미리 보기에 새로운 인증 방법 추가 | Amazon Location Service는 미리 보기 모드에서 익명 사용자를 위한 새로운 인증 방법으로 API 키를 추가합니다. 자세한 내용은 API 키를 사용한 인증되지 않은 게스트의 애플리케이션 액세스 허용을 참조하세요. | 2023년 2월 23일 |
| Amazon Location 설명서, 최신 IAM 모범 사례로 업데이트 | Amazon Location Service 설명서가 최신 AWS Identity and Access Management 모범 사례를 충족하도록 업데이트되었습니다. 자세한 내용을 알아보려면 Amazon Location Service에서의 보안을 참조하세요. | 2023년 1월 26일 |
| Amazon Location GrabMaps Service가 동남아시아의 데이터 공급자로 추가 | Amazon Location은 동남아시아에서 데이터 GrabMaps 공급자로 소개되었습니다. 자세한 내용은 을 참조하십시오 GrabMaps. | 2023년 1월 10일 |
| Amazon Location Service 오픈 데이터 맵 미리 보기 | OpenStreetMap의 일광 지도를 기반으로 하는 공개 미리보기에 새로운 Amazon 위치 데이터 제공업체 및 스타일이 추가되었습니다. 자세한 내용은 오픈 데이터(미리보기)를 참조하세요. | 2022년 12월 15일 |

| | | |
|---|---|---|
| 새로운 HERE 위성 이미지 스타일 | HERE를 데이터 공급자로 사용하는 맵에 HERE 위성 이미지와 HERE 하이브리드 맵 스타일이라는 두 가지 새로운 맵 스타일이 추가되었습니다. 자세한 내용은 HERE 맵 스타일을 확인하세요. | 2022년 10월 25일 |
| 주소 단위 | Amazon Location Service는 이제 주소 내의 단위를 지원합니다 (예: "123 Main St, Apartment 3B, Anytown, USA"). | 2022년 9월 20일 |
| ID로 장소 가져오기 | Amazon Location Service에는 이제 `GetPlace` 작업을 사용하여 `SearchPlaceIndexForSuggestions` 작업에서 제안한 정확한 위치를 찾는 지원이 포함됩니다. 자동 완성 사용을 참조하세요. | 2022년 9월 20일 |
| IAM 정책을 위한 추가 조건 키 | Amazon Location Service는 이제 IAM 정책에서 특정 지오펜스 또는 디바이스에 대한 액세스를 설정할 수 있는 추가 조건 키를 지원합니다. 조건 키를 참조하세요. | 2022년 8월 23일 |
| 원형 지오펜스 | Amazon Location Service는 이제 중앙점과 반지름이 있는 원으로 정의된 지오펜스를 지원하여 디바이스가 특정 위치 내에 있을 때 이벤트를 수신할 수 있습니다. 원형 지오펜스 추가를 참조하세요. | 2022년 8월 11일 |

| | | |
|---|---|---|
| 통합 API 레퍼런스 | Amazon Location Service에 이제 각 하위 서비스에 대한 별도의 가이드가 아닌 단일 API 참조 가이드가 생겼습니다. API에 대한 자세한 내용은 Amazon Location API를 참조하세요. | 2022년 7월 7일 |
| Service Quotas 통합 | Amazon Location은 이제 Service Quotas와 통합됩니다. 이를 통해 AWS Management Console 또는 AWS CLI를 사용하여 할당량을 보고 관리할 수 있습니다. | 2022년 7월 6일 |
| 개념 설명서 챕터 업데이트 | Amazon Location 개념 챕터가 Amazon Location 사용자를 위한 추가 정보로 업데이트되었습니다. | 2022년 4월 22일 |
| 새로운 Android 빠른 시작 튜토리얼 | 개발자가 빠르게 시작하고 실행할 수 있도록 Kotlin을 사용한 Android 개발을 위한 새로운 빠른 시작 튜토리얼이 추가되었습니다. | 2022년 4월 15일 |
| 새로운 HERE 맵 스타일 | HERE를 데이터 공급자로 사용하는 맵에 두 가지 새로운 맵 스타일이 추가되었습니다. 자세한 내용은 HERE 맵 스타일을 확인하세요. | 2022년 3월 15일 |
| 추가된 코드 예제와 튜토리얼로 문서 재구성 | 이 개발자 가이드는 새로운 빠른 시작 및 코드 예제 챕터를 포함하여 주제를 더 쉽게 찾을 수 있도록 재구성되었습니다. | 2022년 2월 25일 |

| | | |
|---|---|---|
| 트래커를 위한 정확도 기반 위치 필터링 | 이제 트래커 리소스를 생성할 때 정확도 기반 필터를 사용할 수 있습니다. | 2021년 12월 7일 |
| 장소 색인 자동 완성 | 이제 장소 색인을 검색할 때 자동 완성을 사용할 수 있습니다. | 2021년 12월 6일 |
| 맵 사용을 위한 새로운 Amplify 튜토리얼 | 웹 애플리케이션에서 맵을 표시하기 위해 AWS Amplify를 사용하는 방법을 보여주는 새로운 튜토리얼을 사용할 수 있습니다. 이 튜토리얼은 Amazon Location Service에서 Amplify 라이브러리 사용에서 확인할 수 있습니다. | 2021년 11월 24일 |
| 쿼리 확장 배치 | Amazon Location Service는 이제 지오코딩 또는 역지오코딩을 수행할 때 결과에 사용할 선호 언어 설정을 지원하고, 결과에 시간대 및 기타 정보를 추가합니다. 지오코딩 및 리버스 지오코딩에 대한 자세한 내용은 지오코딩, 리버스 지오코딩 및 검색을 참조하세요. | 2021년 11월 16일 |
| 트래커 위치 필터링 | Amazon Location Service가 비용 관리에 도움이 되는 새로운 위치 필터링 기능을 트래커에 추가합니다. 이 기능은 업데이트를 저장하거나 지오펜스를 기준으로 평가하기 전에 디바이스의 일부 위치 업데이트를 필터링합니다. 위치 필터링에 대한 자세한 내용은 트래커를 참조하세요. | 2021년 10월 5일 |

| | | |
|---|---|---|
| 업데이트 작업 | Amazon Location Service API 참조에 UpdateMap,, UpdatePlaceIndexUpdateRouteCalculator UpdateGeofenceCollection, 및 작업이 추가되었습니다 UpdateTracker. | 2021년 7월 19일 |
| 튜토리얼 업데이트: Amazon Aurora PostgreSQL 사용자 정의 함수 | Amazon Location에서 Amazon Aurora PostgreSQL 사용자 정의 함수를 사용하여 지리 공간 데이터를 검증, 정리 및 강화하는 방법에 대한 새 튜토리얼이 추가되었습니다. | 2021년 7월 19일 |
| AWS CloudFormation 리소스 | Amazon Location은 이제 AWS CloudFormation 리소스에서AWS::Location::Map ,,, AWS::Location::PlaceIndex AWS::Location::RouteCalculator AWS::Location::Tracker AWS::Location::TrackerConsumer , 및 같은 리소스 유형을 생성할 수 있도록 지원합니다AWS::Location::GeofenceCollection . | 2021년 6월 7일 |
| 리소스에 태그 지정 | 이제 Amazon Location 리소스에 태그를 추가하여 리소스를 관리, 식별, 구성, 검색 및 필터링할 수 있습니다. | 2021년 6월 1일 |

| | | |
|---|---|---|
| 정식 출시 | Amazon Location Service 개발자 문서 정식 출시: 리전, 엔드포인트 및 Service Quotas 가 업데이트되었습니다. | 2021년 6월 1일 |
| Esri Imagery | Amazon Location은 이제 Esri 맵 스타일인 Esri Imagery 사용을 지원합니다. 자세한 내용은 Esri 웹사이트의 Esri World Imagery를 참조하십시오. | 2021년 6월 1일 |
| 경로 계산 | 이제 Amazon Location 경로 계산기를 사용하여 선택한 데이터 공급자의 up-to-date 도로망과 실시간 교통 정보를 기반으로 경로를 계산하고 이동 시간을 추정할 수 있습니다. | 2021년 6월 1일 |
| 저장 데이터에 대한 AWS KMS 고객 관리형 키 암호화 | Amazon Location은 이제 기존 AWS 소유 암호화에 두 번째 암호화 계층을 추가하기 위해 생성하고 소유하고 관리하는 대칭 고객 관리형 키를 사용할 수 있도록 지원합니다. | 2021년 6월 1일 |
| 공개 미리 보기 릴리스 | 공개 미리보기 문서의 최초 릴리스 | 2020년 12월 16일 |
| 튜토리얼 업데이트: 맵 표시 | Android 및 iOS용 맵 MapLibre 표시에 대한 튜토리얼이 MapLibre 네이티브 SDK를 사용하도록 업데이트되었습니다. | 2020년 3월 17일 |

# AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하십시오.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.