aws

開發人員指南

Amazon Location Service

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Location Service: 開發人員指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

歡迎 1
什麼是 Amazon Location Service？ 1
主要功能 2
相關服務 3
快速入門 4
建立網路應用程式 4
建立 資源 5
設定驗證 6
創建 HTML 7
添加地圖 10
新增搜尋 14
最終申請 18
下一步是什麼 23
創建一個安卓應用 23
為您的應用程式建立 Amazon 位置資源 23
設定驗證 25
建立應用程式 28
新增地圖 28
新增搜尋 32
新增追蹤 41
下一步是什麼 50
建立 iOS 應用程式 50
建立 資源 50
設定驗證 52
建立應用程式 54
初始代碼 55
新增地圖 57
新增搜尋 62
新增追蹤 63
下一步是什麼 75
Amazon 位置概念 76
概觀 77
地圖 77
地圖樣式 78

政治觀點 79
自訂 Layer 80
地圖渲染 80
地圖術語 81
地點搜尋 82
地理編碼概念 83
搜尋結果 83
多重結果和相關性 84
地址結果 84
儲存地理編碼結果 86
地方術語 86
路由 87
路線計算器資源 87
計算路線 88
規劃路線 89
路線術語 90
地理圍欄和跟踪器 91
地理圍欄 91
追蹤器 93
地理圍欄術語 96
追蹤器術語 97
常用案例 98
用戶參與和地理營銷應用 99
資產追蹤應用 100
交付應用 102
資料提供者 103
數據提供商的覆蓋範圍和 103
地圖樣式 104
更多詳細資訊 105
埃斯里 105
GrabMaps 112
HERE技術 117
開啟資料 124
資料提供者的功能 131
使用條款和數據歸因 135
區域與端點 135

區域 136
端點 137
API作業端點 138
Service Quotas 139
管理您的 Amazon 定位服務配額 149
與 Amazon 位置開發 151
案例和使用案例 151
軟體開發套件和工具 152
依語言分類的 SDK 153
MapLibre 156
Amazon 位置 SDK 161
Amazon 位置 API 183
使用 Amazon 位置與 AWS SDK 184
錯誤訊息更新 184
程式碼範例 216
Amazon 位置演示站點 218
教學課程：快速入門 218
教學課程：資料庫豐富 219
範例：探索應用程式 219
範例：設定地圖型式 220
範例：繪製標記 221
範例：繪製叢集點 221
範例：繪製多邊形 222
範例：變更地圖語言 222
部落格：預估交貨時間通知 223
範例：串流位置更新 224
範例：地理圍欄和追蹤行動應用程式 224
如何使用 Amazon 位置 225
帳戶先決條 226
註冊一個 AWS 帳戶 226
建立具有管理權限的使用者 226
授予對 Amazon 定 Location Service 的訪問 228
使用地圖 229
必要條件 230
顯示地圖 233
在地圖上繪製 286

設定地圖的實際範圍 286
管理地圖資源 287
地點搜尋 291
必要條件 291
地理编码 295
反向地理編碼 302
自動完成 305
使用地點 ID 311
類別和篩選 312
教學課程：資料庫豐富 317
管理地點索引資源 331
計算路線 334
必要條件 335
計算路線 338
路線規劃 342
不位於道路上的位置 347
出發時間 349
旅行模式 350
管理路由資源 351
地理圍欄和跟踪 354
第 1 步：添加地理圍欄 356
步驟 2：開始追蹤 362
步驟 3：將跟踪器鏈接到地理圍欄集合 375
步驟 4：根據地理圍欄評估設備位置 376
驗證裝置位置 379
對事件做出反應 EventBridge 381
使用 AWS IoT 和追蹤 MQTT 386
管理地理圍欄資源 393
管理追蹤器資源 400
地理圍欄和跟踪移動應用程序示例 405
標記您的 資源 422
限制 422
授予標記權限 423
將標籤新增至資源 424
按標籤追蹤成本 424
使用標籤控制對 資源的存取權 425

進一步了解 425
授予訪問 Amazon 位置 426
使用 API 金鑰 426
使用 Amazon Cognito 432
監控 Amazon Location Service 442
使用監控 CloudWatch 442
使 CloudTrail 用 Amazon 位置 447
用AWS CloudFormation來建立資源 451
Amazon 位置和AWS CloudFormation模板 451
進一步了解 AWS CloudFormation 452
安全 453
資料保護 453
資料隱私 454
資料保留 454
靜態資料加密 455
傳輸中資料加密 466
身分和存取權管理 467
物件 467
使用身分驗證 468
使用政策管理存取權 470
Amazon Location Service 如何與 IAM 472
Amazon 定 Location Service 如何與未經身份驗證的用戶 479
身分型政策範例 479
故障診斷 490
事件反應 492
記錄和監控 492
法規遵循驗證 493
恢復能力 494
基礎架構安全 494
組態與漏洞分析 495
預防混淆代理人 495
安全最佳實務 495
Detective 最佳做法 495
預防性最佳做法 496
最佳實務 496
安全 497

資源管理 497
帳單與成本管理 498
配額和用量 498
文件歷史紀錄 499
AWS 詞彙表 506
..... dvii

歡迎來到 Amazon 定 Location Service

歡迎使用 Amazon 定 Location Service 開發人員指南。

下列主題可協助您根據您嘗試執行的動作，開始使用文件。

獲取 Amazon 位置的概述

- 了解 Amazon 位置中的概念。
- 深入了解如何使用 Amazon 定 Location Service本章中的功能。
- 在 Amazon 位置演示站點中查看演示應用程序。
- 如果您已經擁有AWS 帳戶，則可以使用 Amazon 定 Location Service 主控台第一手探索功能。

使用 Amazon 位置作為開發人員

- 使用建置您的第一個應用程式快速入門。
- 請參閱如何使用 Amazon 定 Location Service本章了解各種 Amazon 定 Location Service 功能的運作方式。
- 請參閱與 Amazon 位置開發本章中提供給您的 SDK 和工具。
- 請參閱您可以在自己的應用程式中使用的程式碼範例和教學課程。您也可以造訪 Amazon Location 示範網站範例頁面，尋找可依功能、語言或平台篩選的範例。
- 在 API 參考指南中取得有關 Amazon 定位 API 的資訊。

什麼是 Amazon Location Service？

Amazon Location Service 可讓您將位置資料和功能新增至應用程式，其中包括地圖、興趣點、地理編碼、路由、地理圍欄和追蹤等功能。Amazon 位置使用來自全球受信任供應商 Esri、Grab 和 HERE 的高品質資料，提供基於位置的服務 (LBS)。借助經濟實惠的數據、跟踪和地理圍欄功能以及用於健康監控的內置指標，您可以構建複雜的啟用位置的應用程序。

使用 Amazon 位置，您可以保留組織資料的控制權。Amazon Location 會移除客戶中繼資料和帳戶資訊，以匿名方式傳送給資料提供者的所有查詢。此外，敏感的追蹤和地理圍欄位置資訊 (例如設施、資產和人員位置) 根本不會離開您的AWS帳戶。這可協助您保護敏感資訊不受第三方侵害、保護使用者隱私，並降低應用程式的安全風險。使用 Amazon 位置，Amazon 和第三方無權出售您的數據或將其用於廣告。

Amazon 位置與 AmazonAWS CloudTrail，Amazon CloudWatch 和AWS Identity and Access Management（IAM）等服務完全集成。 EventBridgeAmazon Location 透過資料整合來簡化您的開發工作流程，並透過內建的監控、安全性和合規功能快速追蹤要生產的應用程式。

如需重點資訊、產品詳細資訊和定價，請參閱 Amazon 定位服務的服務頁面。

Amazon 位置的主要功能

Amazon 位置提供以下功能：

地圖

Amazon 定 Location Service 地圖可讓您以視覺化方式呈現位置資訊，並且是許多基於位置的服務功能的基礎。Amazon 定 Location Service 提供來自全球位置資料供應商 Esri、Grab 和 HERE 的不同風格的地圖磚，以及開放資料地圖。

地方

Amazon Location Service Places 可讓您將搜尋功能整合到應用程式中、將地址轉換為經緯度 (地理編碼) 的地理座標，以及將座標轉換為街道地址 (反向地理編碼)。Amazon 定 Location Service 會從 Esri、Grab 和 HERE 取得高品質的地理空間資料，以支援地點功能。

路由

Amazon 定 Location Service 路線可讓您根據道路和即時交通資訊來尋找 up-to-date 路線並預估行駛時間。構建功能，使您的應用程序可以請求任意兩個位置之間的旅行時間，距離和方向。計算路線規劃中要使用的路線矩陣的時間和距離。

地理圍欄

Amazon 定 Location Service Geofences 可讓您的應用程式在裝置進入或離開已解除的地理區域邊界 (稱為地理圍欄) 時偵測並採取行動。檢測到地理圍欄違規 EventBridge 時，自動將進入或退出事件發送到 Amazon。這可讓您啟動下游動作，例如將通知傳送至目標。

追蹤器

Amazon 定 Location Service 追蹤器可讓您擷取執行已啟用追蹤應用程式之裝置的目前和歷史位置。您也可以將追蹤器與 Amazon 定 Location Service 圍欄連結，以針對您的地理圍欄自動評估裝

置的位置更新。跟踪器可以通過過過濾在存儲或評估地理圍欄之前未移動的位置更新來幫助您降低成本。

當您使用追蹤器時，您所追蹤裝置上的敏感位置資訊不會離開您的AWS帳戶。這有助於保護敏感資訊不受第三方侵害、保護使用者隱私，並降低安全風險。

您可以在 Amazon 位置使用的服務

使用以下服務以及 Amazon 定 Location Service。

整合式監控與管理

Amazon 定 Location Service 與 Amazon 和 Amazon 集成 CloudWatchAWS CloudTrail，以實現有效 EventBridge 的監控和數據管理：

- Amazon CloudWatch — 檢視有關服務使用情況和運作狀態的指標，包括請求、延遲、故障和日誌。如需詳細資訊，請參閱 the section called “使用監控 CloudWatch”。
- AWS CloudTrail— 記錄和監控您的 API 呼叫，其中包括使用者、角色或 AWS 服務採取的動作。如需詳細資訊，請參閱 the section called “使 CloudTrail 用 Amazon 位置”。
- Amazon EventBridge — 啟用事件驅動的應用程式架構，以便您可以使用AWS Lambda函數啟動應用程式和工作流程的其他部分。如需詳細資訊，請參閱 the section called “對事件做出反應 EventBridge”。

開發人員工具

Amazon 定 Location Service 為開發人員提供各種工具，以建立具有位置功能的應用程式。其中包括標準 AWS SDK、行動和 Web SDK，以及將它們與開放原始碼程式庫結合的範例程式碼，例如. MapLibre 使用 Amazon 定 Location Service 主控台來了解資源，並開始使用視覺化和互動式學習工具。

快速入門 Amazon Location Service

開始使用 Amazon 定 Location Service 的最有效方法是使用 Amazon 定位主控台。您可以使用「探索」頁面建立和管理資源，並嘗試使用 Amazon 位置功能。

> **Note**
>
> 若要使用 Amazon 定 Location Service 主控台或遵循本教學的其餘部分，您必須先完成使用 Amazon 定 Location Service 的先決條件，包括建立 AWS 帳戶，以及允許存取 Amazon 位置。

若要開始學習 Amazon Location API，請使用以下教學建立顯示互動式地圖並使用搜尋功能的簡單應用程式。該教程有三個版本：一個向您展示瞭如何使用創建簡單的網頁 JavaScript，第二個版本對於使用 Kotlin 的 Android 應用程序顯示相同的內容，第三個版本顯示了使用 Swift 的 iOS 應用程序相同。

主題

- 建立網路應用程式
- 創建一個安卓應用
- 建立 iOS 應用程式

建立網路應用程式

在本節中，您將創建一個靜態網頁，其中包含地圖和在某個位置進行搜索的功能。首先，您將創建您的 Amazon 位置資源，並為您的應用程序創建一個API密鑰。

主題

- 為您的應用程式建立 Amazon 位置資源
- 為您的應用程式設定驗證
- HTML為您的應用程式建立
- 將交互式地圖添加到您的應用程序
- 新增搜尋到您的應用程式
- 查看最終申請
- 下一步是什麼

為您的應用程式建立 Amazon 位置資源

如果您還沒有這些資源，則必須建立應用程式將使用的 Amazon 位置資源。在這裡，您可以創建一個地圖資源以在應用程序中顯示地圖，並創建地圖索引以在地圖上搜索位置。

將位置資源新增至您的應用程式

1. 選擇您要使用的地圖樣式。

   a. 在 Amazon 位置主控台的「地圖」頁面上，選擇「建立地圖」以預覽地圖樣式。

   b. 為新地圖資源新增「名稱」和「描述」。記下您用於地圖資源的名稱。稍後在自學課程中建立腳本檔案時，您將需要它。

   c. 選擇地圖。

   > **Note**
   >
   > 選擇地圖型式也會選擇您將使用的地圖資料提供者。如果您的應用程式正在追蹤或路由您在業務中使用的資產 (例如送貨車輛或員工)，您只能用HERE作地理位置提供者。如需詳細資訊，請參閱AWS 服務條款第 82 節。

   d. 同意 Amazon 位置條款與條件，然後選擇「建立地圖」。您可以與選擇的地圖互動：放大、縮小或向任何方向平移。

   e. 記下為您的新地圖資源顯示的 Amazon 資源名稱 (ARN)。您將在本教程稍後使用它來創建正確的身份驗證。

2. 選擇您要使用的位置索引。

   a. 在「放置索引」頁面上的 Amazon 位置主控台中，選擇「建立位置索引」。

   b. 新增位置索引資源的「名稱」和「說明」。記下您用於放置索引資源的名稱。稍後在自學課程中建立腳本檔案時，您將需要它。

   c. 選擇資料提供者。

   > **Note**
   >
   > 在大多數情況下，請選擇與您已選擇的地圖提供者相符的資料提供者。這有助於確保搜索將匹配地圖。
   > 如果您的應用程式正在追蹤或路由您在業務中使用的資產 (例如送貨車輛或員工)，您只能用HERE作地理位置提供者。如需詳細資訊，請參閱AWS 服務條款第 82 節。

d. 選擇數據存儲選項。在本自學課程中，不會儲存結果，因此您可以選擇「否，僅限單次使用」。

e. 同意 Amazon 地點條款與條件，然後選擇建立地點索引。

f. 記下ARN新地點索引資源所顯示的內容。您將在本教程的下一部分中使用它來創建正確的身份驗證。

為您的應用程式設定驗證

您在本教程中創建的應用程序具有匿名用法，這意味著您的用戶不需 AWS 要登錄即可使用該應用程序。但是，根據預設，Amazon 定 Location Service APIs 需要身份驗證才能使用。您可以使用 Amazon Cognito 或API金鑰為匿名使用者提供身份驗證和授權。在本教學課程中，您將建立在範例應用程式中使用的API金鑰。

> **Note**
>
> 如需將API金鑰或 Amazon Cognito 與 Amazon 定 Location Service 搭配使用的詳細資訊，請參閱授予對 Amazon 定 Location Service 的訪問。

若要為您的應用程式設定驗證

1. 轉到 Amazon 位置控制台，然後從左側菜單中選擇API鍵。
2. 選擇建立API金鑰。

> **Important**
>
> 您建立的API金鑰必須 AWS 帳戶 與您在上一節中建立的 Amazon 定 Location Service 資源所在的 AWS 區域相同。

3. 一個創建API鍵頁面，填寫以下信息。

   - 名稱 — API 金鑰的名稱，例如`MyWebAppKey`。
   - 資源 — 選擇您在上一節中建立的 Amazon 位置圖和放置索引資源。您可以選擇 [新增資源] 來新增多個資源。這將允許API密鑰與這些資源一起使用。
   - 動作 — 指定您要使用此API金鑰授權的動作。您必須至少選取 geo: GetMap * 和 geo:，這SearchPlaceIndexfForPosition樣教學課程才能如預期般運作。

- 您可以選擇性地將「描述」、「到期時間」或「標籤」新增至API金鑰。您還可以添加引用者（例如*.example.com），將密鑰限制為僅從特定域中使用。這意味著該教程將僅適用於該域。

> **Note**
>
> 建議您通過設置到期時間或API引用者（如果不是兩者）來保護您的密鑰使用情況。

4. 選擇建立API金鑰以建立API金鑰。
5. 選擇 [顯示API金鑰]，然後複製機碼值以供稍後在自學課程中使用。它將以形式出現`v1.public.a1b2c3d4...`。

> **Important**
>
> 在本教程稍後為您的應用程序編寫代碼時，您將需要此密鑰。

HTML為您的應用程式建立

在本教程中，您將創建一個嵌入地圖的靜態HTML頁面，並允許用戶查找地圖上的某個位置。該應用程序將由三個文件組成：網頁的CSS文件和文件，以及一個 JavaScript（.js）文件，用於創建地圖並響應用戶的交互和地圖事件的代碼。HTML

首先，讓我們創建將用於應用程序的HTML和CSS框架。這將是一個簡單的頁面，其中包含一個`<div>`元素來保存地圖容器和一個`<pre>`元素來顯示您的查詢的JSON響應。

若要HTML為您的快速入門應用程式建立

1. 建立稱為 `quickstart.html` 的新檔案。
2. 在您選擇的文字編輯器或環境中編輯檔案。將以下內容添加HTML到文件中。

```
<!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>
```

這HTML有一個指向您將在下一步中創建的文CSS件的指針，一些應用程序的佔位符元素以及一些解釋性文本。

您將在本教學課程稍後使用兩個預留位置元素。第一個是 元素，這將持有地圖控件。第二個是<pre id="response">元素，這將顯示在地圖上搜索的結果。

3. 儲存您的檔案。

現在添CSS加網頁。這將設置應用程序的文本和佔位符元素的樣式。

若要CSS為您的快速入門應用程式建立

1. 在與先前程序中建立的 quickstart.html 檔案相同的資料夾中建立名main.css為的新檔案。
2. 在您要使用的任何編輯器中編輯檔案。將下列文字新增至檔案。

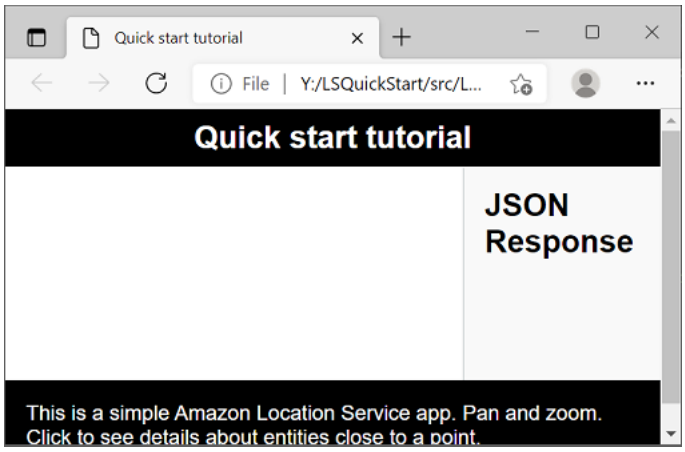
```
* {
  box-sizing: border-box;
  font-family: Arial, Helvetica, sans-serif;
}

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;
}
```

這將地圖設置為填充其他任何東西未使用的空間，將我們的響應區域設置為佔用應用程序寬度的30%，並為標題和說明性文本設置顏色和樣式。

3. 儲存檔案。
4. 現在，您可以在瀏覽器中查看`quickstart.html`文件以查看應用程序的佈局。



接下來，您將地圖控制項新增至應用程式。

將交互式地圖添加到您的應用程序

現在您有架構和 div 預留位置，您可以將地圖控制項新增至應用程式。本教程使用 MapLibre GL JS 作為地圖控制項，從 Amazon 定 Location Service 獲取數據。您也可以使用API金鑰JavaScript 驗證助手來協助您簽署 Amazon 位置APIs的呼叫。

若要將互動式地圖加入至您的應用程式

1. 開啟您在上一節中建立的`quickstart.html`檔案。
2. 將參考新增至所需程式庫，以及您要建立的指令碼檔案。您需要進行的變更顯示在中**`green`**。

```
<!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>
```

這將以下依賴項添加到您的應用程序中：

- MapLibre GL JS. 此物件庫和樣式表包括顯示地圖框並包含互動性 (例如平移和縮放) 的地圖控制項。該控制還允許擴展，例如在地圖上繪製自己的圖徵。
- Amazon 位置客戶端。這為取得地圖資料以及在地圖上搜尋地點所需的 Amazon 位置功能提供界面。Amazon 位置客戶端是基 AWS SDK於 JavaScript v3。
- Amazon 位置身份驗證助手 這提供了有用的功能，可用於使用API金鑰或 Amazon Cognito 驗證 Amazon 定 Location Service。

此步驟也會新增參照`main.js`，您將在下一步建立該參照。

3. 儲存 `quickstart.html` 檔案。
4. 在HTML與和檔案相同的資料夾`main.js`中建立一個名為的新檔CSS案，然後將其開啟以進行編輯。
5. 將以下腳本添加到您的文件中。中的文字 *`red`* 應該替換為API鍵值，映射資源名稱和您之前創建的位置資源名稱，以及您所在地區的區域標識符（例如`us-east-1`）。

```
// 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();
```

此程式碼會設定 Amazon 位置資源，然後設定並初始化 MapLibre GL JS 地圖控制項，並將其放置在具有 id 的項 目中。map

該initializeMap()功能是重要的是要理解。它創建一個新的 MapLibre 地圖控制項（在mlglMap本地調用，但map在代碼的其餘部分調用），用於在應用程序中渲染地圖。

```
  // 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
```

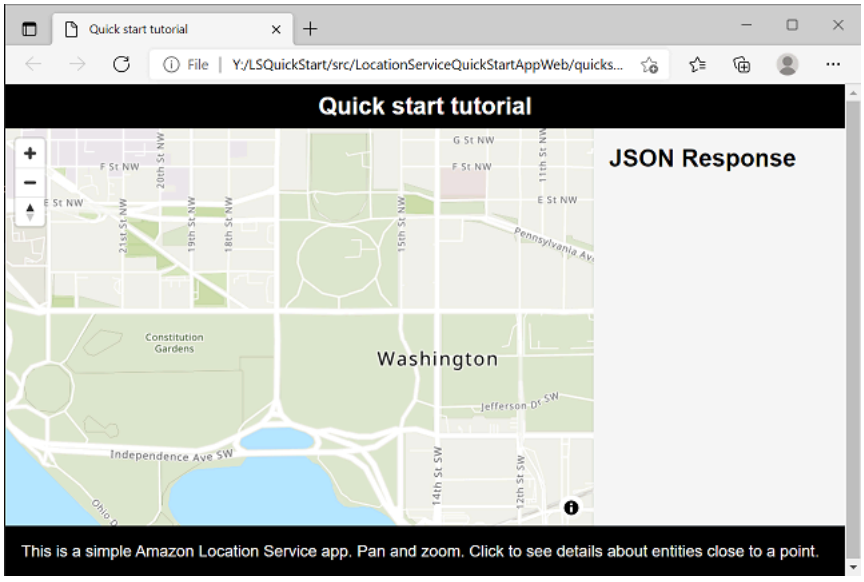
```
});
```

當您建立新的 MapLibre 地圖控制項時，您傳遞的參數指示地圖控制項的初始狀態。在這裡，我們設置以下參數。

- HTML容器，它使用在我們的地圖 div 元素HTML。
- 地圖的初始中心到華盛頓特區的一個點。
- 縮放層級為 16 (縮放至鄰近或圖塊層級)。
- 用於地圖的樣式，可 MapLibre 用於獲取地圖框和其他用於渲染地圖的信息。URL請注意，這URL包括用於驗證的API密鑰。

6. 儲存 JavaScript 檔案，然後使用瀏覽器開啟檔案。現在，您的頁面上有一個地圖，您可以在其中使用平移和縮放動作。

> **Note**
>
> 您可以使用此應用程序來查看 MapLibre 地圖控件的行為。您可以在使用拖動操作時嘗試使用 Ctrl 或 Shift，以查看與地圖進行交互的其他方式。所有這些功能都是可定制的。



您的應用程式即將完成。在下一部分中，您將處理在地圖上選擇位置，並顯示所選位置的地址。您還將在頁面JSON上顯示結果，以查看完整結果。

新增搜尋到您的應用程式

您的應用程序的最後一步是在地圖上添加搜索。在這種情況下，您將添加反向地理編碼搜索，您可以在其中找到某個位置的項目。

> **Note**
>
> Amazon 定 Location Service 還提供按名稱或地址進行搜索的功能，以便在地圖上查找地點的位置。

若要新增搜尋功能至您的應用程式

1. 開啟您在上一節中建立的`main.js`檔案。
2. 修改`main`函數，如圖所示。您需要進行的變更顯示在中**`green`**。

```
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.");
    }
  });
}
```

此程式碼首先設定 Amazon 位置身份驗證協助程式以使用您的API金鑰。

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

然後，它會使用該身份驗證協助程式，以及您用來建立新 Amazon 位置用戶端的區域。

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

接下來，代碼響應用戶在地圖控件上選擇一個位置。它通過捕獲 MapLibre 提供的事件來做到這一點`click`。

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

此 MapLibre `click`事件提供的參數包括使用者選擇的緯度和經度 (`e.lngLat`)。在`click`事件中，程式碼會建立`searchPlaceIndexForPositionCommand`以尋找指定緯度和經度的實體。

```
    // 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);
      ...
    });
```

這裡`IndexName`是您先前建立的位置索引資源的名稱，`Position`也就是要搜尋的緯度和經度，`Language`是結果的偏好語言，並`MaxResults`告訴 Amazon Location 最多只傳回五個結果。

剩餘的程式碼會檢查是否有錯誤，然後在呼叫的`<pre>`元素中顯示搜尋結果`response`，並在警告方塊中顯示最上層的結果。

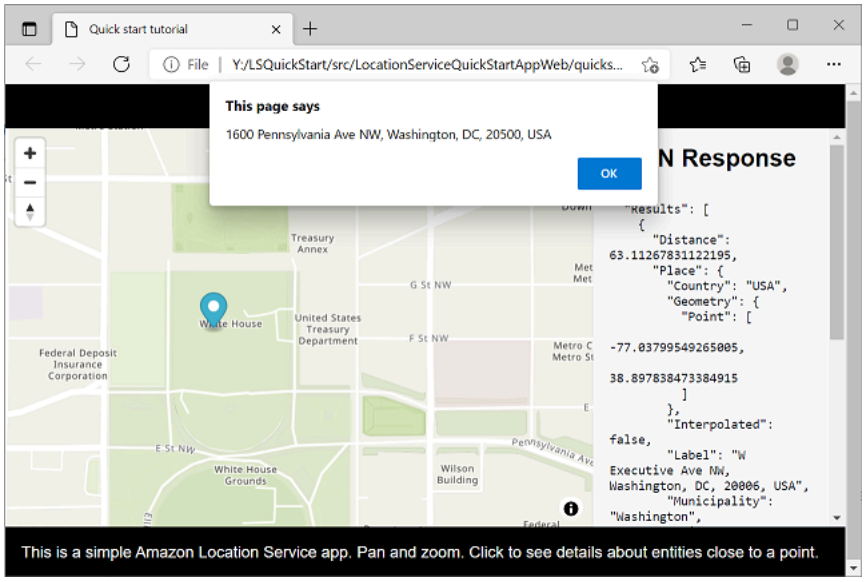
3. (選擇性) 如果您現在在瀏覽器中儲存並開啟`quickstart.html`檔案，在地圖上選擇一個位置將會顯示所選地點的名稱或地址。
4. 應用程序中的最後一步是使用該 MapLibre 功能在用戶選擇的位置添加標記。修改`main`功能，如下所示。您需要進行的變更顯示在中**`green`**。

```
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);

...
```

此代碼聲明一個marker變量，該變量在用戶每次選擇一個位置時填充，顯示他們選擇的位置。該標記是由地圖控件自動呈現，一旦它被添加到地圖與.addTo(map);。程式碼也會檢查先前的標記，並將其移除，以便一次螢幕上只有 1 個標記。

5. 儲存main.js檔案，然後在瀏覽器中開啟quickstart.html檔案。您可以像以前一樣在地圖上平移和縮放，但是現在，如果您選擇位置，您將看到有關所選位置的詳細信息。



您的快速入門應用程式已完成。本教程向您展示瞭如何創建靜態HTML應用程序：

- 建立使用者可以與之互動的地圖。
- 處理地圖事件（`click`）。
- 撥打 Amazon 定 Location ServiceAPI，專門用於搜索某個位置的地圖，使用`searchPlaceIndexForPosition`.
- 使用地 MapLibre 圖控制項來加入標記。

查看最終申請

本節包含此應用程式的最終原始程式碼。您也可以在上找到最終專案 GitHub。

您也可以找到使用 Amazon Cognito 而非開啟API GitHub金鑰的應用程式版本。

Overview

在此快速入門教學課程中，選取每個頁籤以檢視檔案的最終原始程式碼。

這些文件是：

- quickstart.html — 應用程式的架構，包括地圖和搜尋結果的HTML元素持有者。
- main.css — 應用程式的樣式表。
- main.js — 用於驗證使用者、建立地圖以及搜尋事件的應用程式指令碼。`click`

quickstart.html

快速啟動應用程式的HTML架構。

```
<!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

快速啟動應用程式的樣式表。

```
* {
  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

快速啟動應用程式的程式碼。中的文字 *red* 應該以適當的 Amazon 位置物件名稱取代。

```
// 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();
```

下一步是什麼

您已完成快速入門教學課程，並且應該瞭解如何使用 Amazon 定 Location Service 來建置應用程式。要充分利用 Amazon 位置，您可以查看以下資源：

- 深入了解 Amazon 定 Location Service 的概念
- 取得有關如何使用 Amazon 定位功能和功能的詳細資訊
- 了解如何使用 Amazon Location 查看程式碼範例，以擴展此範例並建置更複雜的應用程式

創建一個安卓應用

在本節中，您將創建一個帶有地圖的 Android 應用程序，可以在一個位置進行搜索並在前台跟踪。首先，您將為您的應用程式建立您的 Amazon 位置資源、Amazon Cognito 身分和 API 金鑰。

主題

- 為您的應用程式建立 Amazon 位置資源
- 為您的應用程式設定驗證
- 創建基本的安卓應用程序
- 將交互式地圖添加到您的應用程序
- 將反向地理編碼搜尋新增至應用程式
- 新增追蹤至您的應用程式
- 下一步是什麼

為您的應用程式建立 Amazon 位置資源

如果您還沒有這些資源，則必須建立應用程式將使用的 Amazon 位置資源。在這裡，您可以創建一個地圖資源以在應用程序中顯示地圖，用於在地圖上搜索位置的位置索引，以及用於在地圖上跟踪對象的跟踪器。

將位置資源新增至您的應用程式

1. 選擇您要使用的地圖樣式。

   a. 在 Amazon 位置主控台的「地圖」頁面上，選擇「建立地圖」以預覽地圖樣式。

   b. 為新地圖資源新增「名稱」和「描述」。記下您用於地圖資源的名稱。稍後在自學課程中建立腳本檔案時，您將需要它。

   c. 我們建議您為您的地圖選擇 HERE 地圖。

      > **Note**
      >
      > 選擇地圖型式也會選擇您將使用的地圖資料提供者。如果您的應用程式正在追蹤或路由您在業務中使用的資產，例如送貨車輛或員工，您只能使用 HERE 作為您的地理位置供應商。如需詳細資訊，請參閱AWS 服務條款第 82 節。

   d. 同意 Amazon 位置條款與條件，然後選擇「建立地圖」。您可以與選擇的地圖互動：放大、縮小或向任何方向平移。

   e. 記下為您的新地圖資源顯示的 Amazon 資源名稱（ARN）。您將在本教程稍後使用它來創建正確的身份驗證。

2. 選擇您要使用的位置索引。

   a. 在「放置索引」頁面上的 Amazon 位置主控台中，選擇「建立位置索引」。

   b. 新增位置索引資源的「名稱」和「說明」。記下您用於放置索引資源的名稱。稍後在自學課程中建立腳本檔案時，您將需要它。

   c. 選擇資料提供者。

      > **Note**
      >
      > 在大多數情況下，請選擇與您已選擇的地圖提供者相符的資料提供者。這有助於確保搜索將匹配地圖。
      > 如果您的應用程式正在追蹤或路由您在業務中使用的資產，例如送貨車輛或員工，您只能使用 HERE 作為您的地理位置供應商。如需詳細資訊，請參閱AWS 服務條款第 82 節。

   d. 選擇數據存儲選項。在本自學課程中，不會儲存結果，因此您可以選擇「否，僅限單次使用」。

   e. 同意 Amazon 地點條款與條件，然後選擇建立地點索引。

f. 記下針對新位置索引資源顯示的 ARN。您將在本教程的下一部分中使用它來創建正確的身份驗證。

3. 使用 Amazon 位置控制台創建跟踪器。

a. 打開 Amazon 定 Location Service 控制台。

b. 在左側導覽窗格中，選擇「追蹤器」。

c. 選擇建立追蹤器。

d. 填寫所有必填欄位。

e. 在「位置篩選」下，我們建議您使用預設設定：TimeBased。

f. 選擇創建跟踪器以完成。

為您的應用程式設定驗證

您在本教程中創建的應用程序具有匿名用法，這意味著您的用戶不需 AWS 要登錄即可使用該應用程序。但是，Amazon 定 Location Service API 需要身份驗證才能使用。您可以使用 API 金鑰或 Amazon Cognito 為匿名使用者提供身份驗證和授權。本教學將使用 Amazon Cognito 和 API 金鑰來驗證您的應用程式。

> **Note**
>
> 如需將 Amazon Cognito 或 API 金鑰與 Amazon 定 Location Service 搭配使用的詳細資訊，請參閱授予對 Amazon 定 Location Service 的訪問。

以下教學課程說明如何設定地圖、位置索引和您在其中建立的追蹤器的身份驗證，以及如何設定 Amazon 位置的許可。

設定驗證

1. 導覽至 Amazon 位置主控台，然後從左側功能表中選取 API 金鑰。
2. 點擊「創建 API 密鑰」。請記住，API 金鑰必須與先前建立的 Amazon 定位服務資源位於相同的 AWS 帳戶和區域。
3. 在「創建 API 密鑰」頁面上填寫所需的詳細信息：

- 名稱：提供 API 金鑰的名稱，例如`MyAppKey`。

- 資源：選擇先前建立的 Amazon 定 Location Service 地圖和放置索引資源。您可以通過選擇「添加資源」添加多個資源。這允許 API 密鑰與指定的資源一起使用。
- 動作：指定此 API 金鑰的授權動作。至少選取`geo:GetMap`和`geo:SearchPlaceIndexForPosition`以確保自學課程如預期般運作。
- 選用您可以新增說明、到期時間、標籤或參照者，例如`https://www.example.com`，將金鑰的使用量限制在特定網域，讓教學課程只能在該網域內運作。

4. 按一下「建立 API 金鑰」以產生 API 金鑰。
5. 選取 [顯示 API 金鑰] 並複製金鑰值，以`v1.public.a1b2c3d4`供稍後在教學課程中使用。

建立 IAM 政策以進行追蹤

1. 以具有管理員許可的使用者身分登入 IAM 主控台 (https://console.aws.amazon.com/iam/)。
2. 在導覽窗格上選擇 Policies (政策)。
3. 在內容窗格中，選擇 Create policy (建立政策)。
4. 選擇 JSON 選項，然後將此 JSON 政策複製並貼到 JSON 文字方塊中。

```
{
        "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}"
                ]
            }
        ]
```

```
    }
```

這是追蹤的政策範例。若要將範例用於您自己的原則，請取代RegionAccount、和TrackerName預留位置。

> **Note**
>
> 雖然未驗證身分集區的目的是在不安全的網際網路網站上曝光，但請注意，這些身分集區會被交換成標準且有時間限制的 AWS 登入資料。
> 適當範圍與未驗證身分集區相關聯的 IAM 角色非常重要。如需有關在 Amazon Cognito 與 Amazon 定 Location Service 務搭配使用和適當設定政策範圍的詳細資訊，請參閱授與 Amazon 定 Location Service 的存取權。

5. 在「檢閱並建立」頁面上，輸入原則名稱欄位的名稱。檢閱原則所授與的權限，然後選擇 [建立原則] 儲存工作。

新的政策會出現在受管政策清單中，並且已準備好連接。

設定追蹤的驗證

1. 在 Amazon Cognito 主控台中為您的地圖應用程式設定身份驗證。
2. 開啟 [識別集區] 頁面。

> **Note**
>
> 您建立的集 AWS 區必須與您在上一節中建立的 Amazon 定 Location Service 資源位於相同的 AWS 帳戶和區域。

3. 選擇 [建立識別集區]。
4. 從 [設定識別集區信任] 步驟開始。對於使用者存取驗證，請選取來賓存取，然後按下一步。
5. 在 [設定權限] 頁面上，選取 [使用現有的 IAM 角色]，然後輸入您在上一步中建立的 IAM 角色名稱。準備就緒後，按下一步繼續進行下一步。
6. 在 [設定內容] 頁面上，提供身分識別集區的名稱。然後按下一頁.
7. 在 [檢閱並建立] 頁面上，檢閱存在的所有資訊，然後按 [建立身分集區]。
8. 開啟 [身分識別集區] 頁面，然後選取您剛建立的身分集區。然後複製或寫下 IdentityPoolId 您稍後將在瀏覽器腳本中使用的內容。

創建基本的安卓應用程序

在本教程中，您將創建一個嵌入地圖的 Android 應用程序，並允許用戶查找地圖上的位置。

首先，使用 Android 工作室的新項目嚮導創建一個空的 Kotlin 應用程序。

若要建立空白應用程式 (AndroidStudio)

1. 開始 AndroidStudio。打開菜單，然後選擇文件，新建，新建項目。
2. 從 [電話和平板電腦] 索引標籤中，選取 [空白活動]，然後選擇 [下一步
3. 為您的應用程式選擇「名稱」、「Pack age 名稱」和「儲存位置」。
4. 在語言的下拉列表中，選擇 Kotlin。
5. 選擇「完成」以建立空白應用程式。

將交互式地圖添加到您的應用程序

現在您已經建立了基本的應用程式，您可以將地圖控制項新增至您的應用程式。本教學課程使用 API 金鑰來管理地圖檢視。地圖控件本身是本MapLibre 機庫的一部分，使用 API 密鑰和 MapLibre，地圖數據來自 Amazon 位置。

要將映射添加到您的應用程序，您必須執行以下操作：

- 將 MapLibre 依賴項添加到您的項目中。
- 使用撰寫設定地圖檢視程式碼。
- 編寫代碼以顯示地圖。

使用以下步驟將地圖添加到您的應用程序：

1. 將 MapLibre 依賴項添加到您的項目

   a. 在中 AndroidStudio，選取「檢視」功能表，然後選擇「工具視窗」、「專案」。這將打開項目窗口，該窗口使您可以訪問項目中的所有文件。

   b. 在「項目」窗口中，打開 gradle，然後在樹視圖中打開`libs.versions.toml`文件。這將打開`libs.versions.toml`文件進行編輯。現在在`libs.versions.toml`文件中添加以下版本和庫數據。

```
[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. 完成編輯`libs.versions.toml`檔案後， AndroidStudio 必須重新同步專案。在編`libs.versions.toml`輯視窗頂端， AndroidStudio 會提示您同步。選擇「立即同步」以同步您的項目，然後再繼續。

d. 在「項目」窗口中，在樹視圖中打開 Gradle 腳本，然後選擇應用程序模塊的`build.gradle`文件。這將打開`build.gradle`文件進行編輯。

e. 在檔案底部的 [相依性] 區段中，新增下列相依性。

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

f. 完成添加搖籃依賴關係後，Android 工作室必須重新同步該項目。在編輯窗口的頂部，Android Studio 中，選擇立即同步以同步您的項目，然後再繼續。

2. 現在，您將使用撰寫設置地圖視圖代碼。使用下列步驟：

a. 從項目窗口中，在樹視圖中打開 App，Java，**#####**，然後轉到 ui 文件夾，在 ui 文件夾中創建一個視圖目錄。

b. 裡面的視圖目錄創建一個`MapLoadScreen.kt`文件。

c. 將以下代碼添加到您的`MapLoadScreen.kt`文件中。

```
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. 編寫代碼以顯示地圖。

   a. 將以下代碼添加到您的MainActivity.kt文件中。

```
// ...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. 儲存 `MainActivity.kt` 檔案。您現在可以建置應用程式。要運行它，您可能必須設置一個設備來模擬它 AndroidStudio，或在設備上使用該應用程序。使用此應用程序查看地圖控件的行為。您可以通過在地圖上拖動並捏住以進行縮放來進行平移。

在下一節中，您將在地圖上添加一個標記，並在移動地圖時顯示標記所在位置的地址。

將反向地理編碼搜尋新增至應用程式

現在，您將向應用程序添加反向地理編碼搜索，您可以在其中找到某個位置的項目。為了簡化使用 Android 應用程序，我們將搜索屏幕的中心。若要尋找新位置，請將地圖移至您要搜尋的位置。我們將在地圖的中心放置一個標記，以顯示我們正在搜索的位置。

新增反向地理編碼搜尋將由兩部分組成。

- 在屏幕的中心添加一個標記，以向用戶顯示我們正在搜索的位置。
- 為結果添加文本框，然後搜索標記位置的內容並在文本框中顯示它。

若要將標記新增至您的應用程式

1. 將此圖像保存到app/res/drawable文件夾中的項目中red_marker.png (您也可以從中訪問圖像GitHub。或者，您也可以建立映像檔。您也可以對不想顯示的零件使用具有透明度的 .png 檔案。
2. 將以下代碼添加到您的 MapLoadScreen.kt 文件中。

```
// ...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. 構建並運行您的應用程序以預覽功能。

您的應用程序現在在屏幕上有一個標記。在這種情況下，它是不會移動的靜態圖像。它用於顯示地圖視圖的中心，這是我們將搜索的地方。在以下過程中，我們將在該位置添加搜索。

將某個位置的反向地理編碼搜尋新增至您的應用程式

1. 在「項目」窗口中，在樹視圖中打開 gradle 到`libs.versions.toml`文件。這將打開`libs.versions.toml`文件進行編輯。現在在`libs.versions.toml`文件中添加以下版本和庫數據。

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

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

```
[plugins]
...
```

2. 完成編輯libs.versions.toml檔案後， AndroidStudio 必須重新同步專案。在編libs.versions.toml輯視窗頂端， AndroidStudio 會提示您同步。選擇「立即同步」以同步您的項目，然後再繼續。
3. 在「項目」窗口中，在樹視圖中打開 Gradle 腳本，然後選擇應用程序模塊的build.gradle文件。這將打開build.gradle文件進行編輯。
4. 在檔案底部的 [相依性] 區段中，新增下列相依性。

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

5. 完成編輯 Gradle 依賴關係後， AndroidStudio 必須重新同步項目。在編build.gradle輯視窗頂端， AndroidStudio 會提示您同步。選取SyncNow以同步您的專案，然後再繼續。
6. 現在，在樹視圖中將數據添加到請求目錄中，並創建ReverseGeocodeRequest.kt數據類。將以下代碼添加到類中。

```
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. 現在，在樹視圖中將數據添加到響應目錄中，並創建ReverseGeocodeResponse.kt數據類。在其中添加以下代碼。

```
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. 現在，從項目窗口，打開應用程序，Java，####在樹視圖，並轉到 ui 文件夾，裡面 ui 文件夾創建視圖模型目錄。
9. 裡面視圖模型目錄創建MainViewModel.kt文件。
10. 將以下代碼添加到您的MainViewModel.kt文件中。

```
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. 如果尚未開啟，請開啟MapLoadScreen.kt檔案，如先前程序所示。新增以下程式碼。這將創建一個撰寫文本視圖，您將在其中看到位置的反向地理編碼搜索結果。

```
// ...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. 在應用程式中的 java 下的套件名稱資料夾中 AndroidStudio，開啟`MainActivity.kt`檔案。如圖所示修改代碼。

```
// ...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
                )
            }
        }
    }
}
```

此代碼適用於地圖視圖。中的虛擬相機位置定義了地圖視圖 MapLibre。移動地圖可以被認為是移動該虛擬攝像機。

- ViewModel 具有標籤變數：此變數會在撰寫文字檢視中設定資料。
- onMapReady：此功能已更新以註冊兩個新事件。
- 每當用戶移動地圖時都會發生該onCameraMove事件。一般來說，移動地圖時，我們要隱藏搜索，直到用戶完成移動地圖。
- 當用戶暫停移動地圖發生該onCameraIdle事件。此事件會呼叫我們的反向地理編碼功能，在地圖中央進行搜尋。

- `reverseGeocode(latLng: LatLng, apiKey: String)`：此功能（在事件 onCameraIdle中調用）在地圖的中心搜索位置並更新標籤以顯示結果。它使用相機目標，該目標定義了地圖的中心（相機正在查看的位置）。

13. 保存您的文件，並構建和運行您的應用程序，以查看其是否有效。

具有搜索功能的快速啟動應用程序已完成。

新增追蹤至您的應用程式

若要將追蹤新增至範例應用程式，請依照下列步驟執行：

1. 將跟踪和身份驗證 SDK 依賴項添加到您的項目中。
2. 在 AndroidManifest .xml 檔案中包含權限和服務項目。
3. 使用撰寫設定開始/停止追蹤按鈕程式碼。
4. 添加用於創建 LocationTracker 對象的代碼，並開始和停止跟踪。
5. 使用安卓模擬器創建一個測試路線。

1. 將跟踪和身份驗證 SDK 依賴項添加到您的項目中。

   a. 在「項目」窗口中，打開 gradle，然後在樹視圖中打開`libs.versions.toml`文件。這將打開`libs.versions.toml`文件進行編輯。現在在`libs.versions.toml`文件中添加以下版本和庫數據。

```
[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. 完成編輯libs.versions.toml檔案後， AndroidStudio 必須重新同步專案。在編libs.versions.toml輯視窗頂端， AndroidStudio 會提示您同步。選擇「立即同步」以同步您的項目，然後再繼續。

c. 在「項目」窗口中，在樹視圖中打開 Gradle 腳本，然後選擇應用程序模塊的build.gradle文件。這將打開build.gradle文件進行編輯。

d. 在檔案底部的 [相依性] 區段中，新增下列相依性。

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

e. 完成編輯 Gradle 依賴關係後， AndroidStudio 必須重新同步項目。在構建 .gradle 編輯窗口的頂部， AndroidStudio 提示您同步。選取SyncNow以同步您的專案，然後再繼續。

2. 在 AndroidManifest .xml 檔案中包含權限和服務項目。

- 若要在您的中包含正確的權限和服務項目AndroidManifest.xml file，請使用下列程式碼更新檔案：

```
<?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. 使用撰寫設定開始/停止追蹤按鈕程式碼。

   a. 添加兩個圖像播放和暫停在分辨率下可繪製命名為 ic_pause 和 ic_ play。您也可以從中存取影像GitHub。

   b. 如果尚未開啟，請開啟MapLoadScreen.kt檔案，如先前程序所示。新增以下程式碼。這將創建一個撰寫按鈕視圖，我們可以點擊它開始和停止跟踪。

```
// ...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. 添加用於創建 LocationTracker 對象的代碼，並開始和停止跟踪。

   a. 在文件中添加以下代`MainViewModel.kt`碼。

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

   b. 將以下代碼添加到您的`MainActivity.kt`文件中。

```
// ...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
                )
            }
        }
    }
}
```

上面的代碼演示了如何創建一個`LocationTracker`對象，以`AuthHelper`及如何啟動和停止跟踪 LocationTracker。

- `authenticateUser()`：此方法創建 AuthHelper 和 LocationTracker 對象。
- `onStartStopTrackingClick`：當用戶單擊開始/停止跟踪按鈕時觸發此回調，該按鈕將使用 Tracking SDK 開始/停止跟踪。

5. 使用安卓模擬器創建測試路線。

   a. 通過使用安卓工作室啟動 AVD 打開模擬器。

   b. 通過單擊模擬器工具欄中的更多（三點）圖標打開擴展控件。

   c. 從側邊欄中選擇「位置」以打開「位置」。

d. 使用 GPX 數據創建路線，或通過單擊地圖並選擇源和目標數據來創建路線。
e. 通過單擊播放路線開始模擬開始模擬 GPS 路線。
f. 通過運行您的應用程序並觀察它如何處理模擬路由來測試應用程序。

這是 Android 快速入門應用程序的完整演示。

下一步是什麼

此應用程式的原始程式碼可在上取得GitHub。

要充分利用 Amazon 位置，您可以查看以下資源：

- 深入了解 Amazon 定 Location Service 的概念
- 取得有關如何使用 Amazon 定位功能和功能的詳細資訊
- 了解如何使用 Amazon Location 查看程式碼範例，以擴展此範例並建置更複雜的應用程式

建立 iOS 應用程式

在本節中，您將創建一個 iOS 應用程序，以便在某個位置進行搜索並在前台進行跟踪。首先，您將創建您的 Amazon 位置資源，並為您的應用程序創建一個 Amazon Cognito 身份。

主題

- 為您的應用程式建立 Amazon 位置資源
- 為您的應用程式設定驗證
- 建立基本 iOS 應用程式
- 設定初始程式碼
- 將交互式地圖添加到您的應用程序
- 新增搜尋到您的應用程式
- 新增追蹤至您的應用程式
- 下一步是什麼

為您的應用程式建立 Amazon 位置資源

如果您還沒有這些資源，則必須建立應用程式將使用的 Amazon 位置資源。您將創建一個地圖資源以在應用程序中顯示地圖，用於在地圖上搜索位置的位置索引，以及一個跟踪器來跟踪地圖上的對象。

將位置資源新增至您的應用程式

1. 選擇您要使用的地圖樣式。

   a. 在 Amazon 位置主控台的「地圖」頁面上，選擇「建立地圖」以預覽地圖樣式。

   b. 為新地圖資源新增「名稱」和「描述」。記下您用於地圖資源的名稱。稍後在自學課程中建立腳本檔案時，您將需要它。

   c. 選擇地圖。

   > **Note**
   >
   > 選擇地圖型式也會選擇您將使用的地圖資料提供者。如果您的應用程式正在追蹤或路由您在業務中使用的資產，例如送貨車輛或員工，您只能使用 HERE 作為您的地理位置供應商。如需詳細資訊，請參閱AWS 服務條款第 82 節。

   d. 同意 Amazon 位置條款與條件，然後選擇「建立地圖」。您可以與選擇的地圖互動：放大、縮小或向任何方向平移。

   e. 記下為您的新地圖資源顯示的 Amazon 資源名稱（ARN）。您將在本教程稍後使用它來創建正確的身份驗證。

2. 選擇您要使用的位置索引。

   a. 在「放置索引」頁面上的 Amazon 位置主控台中，選擇「建立位置索引」。

   b. 新增位置索引資源的「名稱」和「說明」。記下您用於放置索引資源的名稱。稍後在自學課程中建立腳本檔案時，您將需要它。

   c. 選擇資料提供者。

   > **Note**
   >
   > 在大多數情況下，請選擇與您已選擇的地圖提供者相符的資料提供者。這有助於確保搜索將匹配地圖。
   > 如果您的應用程式正在追蹤或路由您在業務中使用的資產，例如送貨車輛或員工，您只能使用 HERE 作為您的地理位置供應商。如需詳細資訊，請參閱AWS 服務條款第 82 節。

   d. 選擇數據存儲選項。在本自學課程中，不會儲存結果，因此您可以選擇「否，僅限單次使用」。

   e. 同意 Amazon 地點條款與條件，然後選擇建立地點索引。

f. 記下針對新位置索引資源顯示的 ARN。您將在本教程的下一部分中使用它來創建正確的身份驗證。

3. 使用 Amazon 位置控制台創建跟踪器。

a. 打開 Amazon 定 Location Service 控制台。

b. 在左側導覽窗格中，選擇「追蹤器」。

c. 選擇建立追蹤器。

d. 填寫所有必填欄位。

e. 在「位置篩選」下，我們建議您使用預設設定：TimeBased。

f. 選擇創建跟踪器以完成。

為您的應用程式設定驗證

您在本教程中創建的應用程序具有匿名用法，這意味著您的用戶不需 AWS 要登錄即可使用該應用程序。但是，Amazon 定 Location Service API 需要身份驗證才能使用。您將使用 Amazon Cognito 為匿名使用者提供身份驗證和授權。本教學將使用 Amazon Cognito 來驗證您的應用程式。

> **Note**
>
> 如需將 Amazon Cognito 與 Amazon 定 Location Service 搭配使用的詳細資訊，請參閱授予對 Amazon 定 Location Service 的訪問。

以下教學課程說明如何設定地圖、位置索引和您在其中建立的追蹤器的身份驗證，以及如何設定 Amazon 位置的許可。

建立 IAM 政策以進行追蹤

1. 以具有管理員許可的使用者身分登入 IAM 主控台 (https://console.aws.amazon.com/iam/)。
2. 在導覽窗格上選擇 Policies (政策)。
3. 在內容窗格中，選擇 Create policy (建立政策)。
4. 選擇 JSON 選項，然後將此 JSON 政策複製並貼到 JSON 文字方塊中。

```
{
        "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}"
                ]
            }
        ]
    }
```

這是追蹤的政策範例。若要將範例用於您自己的原則，請取代`Region`、`AccountIndexName`、`MapName`和TrackerName預留位置。

> **Note**
>
> 雖然未驗證身分集區的目的是在不安全的網際網路網站上曝光，但請注意，這些身分集區會被交換成標準、有時間限制 AWS 的認證。
> 適當範圍與未驗證身分集區相關聯的 IAM 角色非常重要。如需有關在 Amazon Cognito 與 Amazon 定 Location Service 務搭配使用和適當設定政策範圍的詳細資訊，請參閱授與 Amazon 定 Location Service 的存取權限。

5. 在「檢閱並建立」頁面上，輸入原則名稱欄位的名稱。檢閱原則所授與的權限，然後選擇 [建立原則] 儲存工作。

新的政策會出現在受管政策清單中，並且已準備好連接。

設定追蹤的驗證

1. 在 Amazon Cognito 主控台中為您的地圖應用程式設定身份驗證。
2. 開啟 [識別集區] 頁面。

> **Note**
>
> 您建立的集 AWS 區必須與您在上一節中建立的 Amazon 定 Location Service 資源位於相同的 AWS 帳戶和區域。

3. 選擇 [建立識別集區]。
4. 從 [設定識別集區信任] 步驟開始。對於使用者存取驗證，請選取來賓存取，然後按下一步。
5. 在 [設定權限] 頁面上，選取 [使用現有的 IAM 角色]，然後輸入您在上一步中建立的 IAM 角色名稱。準備就緒後，按下一步繼續進行下一步。
6. 在 [設定內容] 頁面上，提供身分識別集區的名稱。然後按下一頁.
7. 在 [檢閱並建立] 頁面上，檢閱存在的所有資訊，然後按 [建立身分集區]。
8. 開啟 [身分識別集區] 頁面，然後選取您剛建立的身分集區。然後複製或寫下 IdentityPoolId 您稍後將在瀏覽器腳本中使用的內容。

建立基本 iOS 應用程式

在本教程中，您將創建嵌入地圖的 iOS 應用程序，並允許用戶查找地圖上的位置。

首先，讓我們使用 Xcode 的項目嚮導創建一個 Swift 應用程序。

要創建一個空的應用程序（Xcode）

1. 打開 Xcode，然後從菜單中選擇文件，新建，新建項目。
2. 從 iOS 索引標籤中，選取 [應用程式]，然後選擇 [下一步]。
3. 在「介面」欄位中輸入「產品名稱」、「組織識別碼」`SwiftUI`。選擇「下一步」以完成選取。
4. 選擇一個位置，您將保存您的項目，然後按創建按鈕創建空的應用程序。

建立基本應用程式之後，您將需要為範例應用程式安裝所需的套件。

安裝必要的依賴

1. 在 Xcode 中，右鍵單擊該項目，然後選擇添加包... 。這將打開「數据包」窗口，您可以在其中將軟件包添加到項目中。
2. 在「封裝」視窗中，新增下列封裝：

- 對於地圖本地軟件包，請使用以下網址：https://github.com/maplibre/ maplibre-gl-native-distribution。從 URL 中，新增下列套件：`maplibre-gl-native-distribution`、和`Mapbox`。
- 對於 Amazon 位置身份驗證 iOS 開amazon-location-mobile-auth發套件，請使用以下網址：https://github.com/aws-geospatial/ 從 URL 中，新增下列套件：`amazon-location-mobile-auth-sdk-ios`、和`AmazonLocationiOSAuthSDK`。
- 對於 Amazon 位置跟踪 iOS 開amazon-location-mobile-tracking發套件，請使用以下網址：https://github.com/aws-geospatial/ 從 URL 中，新增下列套件：`amazon-location-mobile-tracking-sdk-ios`、和`AmazonLocationiOSTrackingSDK`。

設定初始程式碼

在您的應用中啟用位置權限

1. 打開你的 Xcode 項目。
2. 找到專案的`Info.plist`檔案。
3. 根據您的應用程序的要求添加必要的位置權限密鑰。以下是關鍵字：

   - `NSLocationWhenInUseUsageDescription`：說明您的應用程序在使用時為什麼需要位置訪問權限。
   - `NSLocationAlwaysAndWhenInUseUsageDescription`：說明您的應用程式需要持續存取位置的原因。

現在，您需要在應用程序中配置資源值。新增名為的新檔案，`Config.xcconfig`並填寫您先前在 Amazon 主控台中建立的值。

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

1. 從左側導覽器區段中，選取專案。
2. 在「目標」部分下，選擇您的應用程序，然後單擊「信息」選項卡。
3. 使用如下所示的值添加信息屬性：

4. 添加包含以下內容的Config.swift文件，該文件將從 Bundle 信息文件中讀取配置值。

```
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. 使用名稱創建一個新文件夾ViewModel並在其中添加TrackingViewModel.swift文件。

```
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
    }
}
```

將交互式地圖添加到您的應用程序

現在，您將地圖控件添加到您的應用程序。本教學課程使用 MapLibre 和 AWS API 來管理應用程式中的地圖檢視。地圖控制項本身就是 MapLibre GL 原生 iOS 程式庫的一部分。

1. 使用以下代碼在 Views **MapView.swift** 文件夾下添加文件：

```
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. 在AWSSignatureV4Delegate文件ViewModel夾下添加文件。該文件用於與所有的 MapView http 請求來呈現地圖簽名：

```
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. 在視圖**UserLocationView.swift**文件夾下添加文件。這增加了一個按鈕，其中心地圖到用戶的位置

```
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. 使用以下代碼添加TrackingView.swift文件：

```
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)
            }
        }
    }
}
```

您現在可以建置應用程式。要運行它，您可能必須設置一個設備才能在 Xcode 中模擬它，或者在設備上使用該應用程序。使用此應用程序查看地圖控件的行為。您可以通過在地圖上拖動並捏縮放來平移。您可以自行變更地圖控制項的運作方式，以根據應用程式的需求自訂它。

新增搜尋到您的應用程式

現在，您將向應用程序添加反向地理編碼搜索，您可以在其中找到某個位置的項目。為了簡化 iOS 應用程序的使用，我們將搜索屏幕的中心。若要尋找新位置，請將地圖移至您要搜尋的位置。我們將在地圖的中心放置一個標記，以顯示我們正在搜索的位置。

1. 在與反向地理編碼搜索相關的 TrackingViewModel .swift` 文件中添加以下代碼

```
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. 使用下面的代碼更新`TrackingView.swift`文件，這將顯示 mapview 的居中位置的地址

```
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)
                    }
                }
            }
        }
    }
}
```

新增追蹤至您的應用程式

您的應用程序的最後一步是將跟踪功能添加到您的應用程序。在這種情況下，您將在應用程序上添加開始跟踪，停止跟踪，獲取和顯示跟踪點。

1. 在專TrackingBottomView.swift案中加入檔案。其中有一個按鈕，可以啟動和停止跟踪用戶位置，並在地圖上顯示跟踪點。

```
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. 使用以下代碼更新TrackingView.swift文件

```
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. 在TrackingViewModel.swift文件中添加以下代碼。這些功能負責啟動和停止跟踪。如果拒絕使用者位置權限，它也會顯示錯誤警示。

4. 要實現前景跟踪副本，請粘貼以下代碼示例：

```
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將要求使用者的位置權限。應用程式必須使用「使用中時」或「僅限一次」權限。否則，應用程序將拋出權限被拒絕的錯誤。

若要取得並顯示追蹤位置，請遵循下列步驟：

1. 要從用戶的設備獲取位置，您需要提供開始和結束日期和時間。單一呼叫最多可傳回 100 個追蹤位置，但如果追蹤位置超過 100 個，則會傳回「nextToken」值。您將需要使用 `nextToken '調用後續的 getTrackerDevice「位置」調用，以便在給定的開始和結束時間加載更多跟踪點。

```
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. 現在用下面的代碼替換MapView.swift文件中的代碼：

```
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]?)
}
```

若要本地化字串值，請使用下列程序。

1. 建立並新增名為的新檔案Localizable.xcstrings。
2. 右鍵單擊該Localizable.xcstrings文件，然後將其作為源代碼打開。
3. 以下列項目取代其內容：

```
{
  "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. 保存文件，並構建和運行您的應用程序以預覽功能。

5. 允許位置權限，然後點擊跟踪按鈕。該應用程序將開始上傳用戶位置並將其上傳到 Amazon 位置跟踪器。它還將在地圖上顯示用戶位置更改，跟踪點和當前地址。

您的快速啟動應用程序已完成。本教程向您展示瞭如何創建一個 iOS 應用程序：

- 建立使用者可以與之互動的地圖。
- 處理與更改地圖視圖的用戶相關聯的幾個地圖事件。
- 呼叫 Amazon 定 Location Service API，特別是使用 Amazon 位置的 searchByPosition API 在某個位置搜索地圖。

下一步是什麼

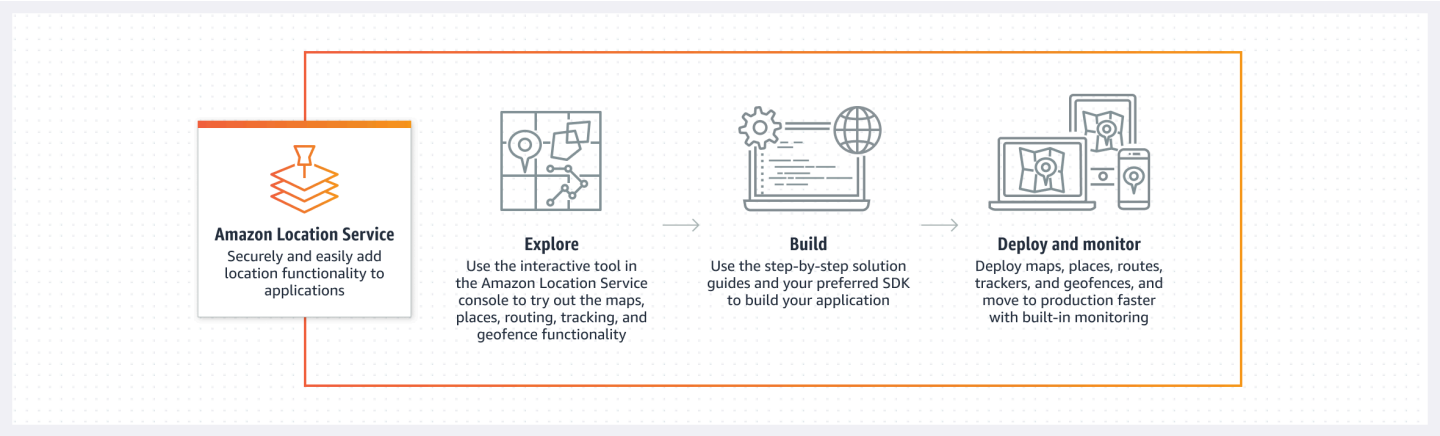
此應用程式的原始程式碼可在上取得GitHub。

要充分利用 Amazon 位置，您可以查看以下資源：

- 深入了解 Amazon 定 Location Service 的概念
- 取得有關如何使用 Amazon 定位功能和功能的詳細資訊
- 了解如何使用 Amazon Location 查看程式碼範例，以擴展此範例並建置更複雜的應用程式

Amazon Location Service 概念

使用 Amazon 定 Location Service，您可以將位置資料安全地新增至應用程式。使用 Amazon 位置主控台提供的視覺化和互動式工具來探索某些功能。使用探索工具，您可以操作默認地圖，搜索興趣點，在感興趣的區域周圍繪製地理圍欄，以及模擬將設備位置發送到跟踪器。



當您準備好進行建置時，請建立資源，並從各種地圖樣式和資料提供者中進行選擇。然後，您可以安裝與您SDK的開發環境相匹配的，並按照本指南中的說明APIs使用 Amazon 位置。此外，您還可以使用 Amazon CloudWatch 和 AWS CloudTrail.

本節中的主題為您提供 Amazon Location 核心概念的概觀，並讓您準備好開始在自己的應用程式中使用位置。

主題

- Amazon 位置概述
- 地圖
- 地點搜尋
- 路由
- 地理圍欄和跟踪器
- 使用 Amazon 定 Location Service 的常見使用案例
- 什麼是資料提供者？
- Amazon 位置區域和端點
- Amazon 定 Location Service 配額

Amazon 位置概述

Amazon 定 Location Service 可透過資源存取以位置為基礎的功能和 AWS 資料提供者。Amazon 位置提供五種類型的 AWS 資源，具體取決於您需要的功能類型。一起使用不同的資源來建立完整的基於位置的應用程式。您可以使用 Amazon 位置主控台、Amazon 位置APIs或SDKs.

每個資源都會定義要使用的基礎資料提供者 (如果適用)，並可存取與其類型相關的功能。

例如：

- Amazon 定 Location Service 地圖可讓您從地圖供應商選擇要在行動或 Web 應用程式上使用的地圖。
- Amazon 定 Location Service 可讓您選擇用於搜尋興趣點、完成部分文字、地理編碼和反向地理編碼的資料提供者。
- Amazon 定 Location Service 路線可讓您選擇資料供應商，並根據道路和即時交通資訊尋找 up-to-date 路線並預估行駛時間。
- Amazon 定 Location Service 圍欄可讓您將感興趣的區域定義為虛擬邊界。然後，您可以針對它們評估位置，並獲取進入和退出事件的通知。
- Amazon 定 Location Service 追蹤器會從您的裝置接收位置更新。您可以將跟踪器鏈接到地理圍欄集合，以便根據地理圍欄自動評估所有位置更新。

您可以使用IAM政策來管理和授權對 Amazon 位置資源的存取。您也可以將資源組織到資源群組中，以便隨著資源數量增加而管理和自動化工作。如需有關管理 AWS 資源的詳細資訊，請參閱什麼是 AWS Resource Groups？ 在《Res AWSource Groups 使用指南》中。

位置是透過使用遵循世界大地測量系統 (WGS84) 的緯度和經度座標來定義，通常用作「全球定位系統」(GPS) 服務的標準座標參考系統。

以下各節說明 Amazon 位置元件的運作方式。

地圖

Amazon Location Service 對應資源可讓您存取地圖的基礎底圖資料。您可以將 Map 資源與地圖彩現程式庫搭配使用，以將互動式地圖加入至您的應用程式。您可以根據應用程式的需要，將其他功能加入至地圖，例如標記 (或圖釘)、佈線和多邊形區域。

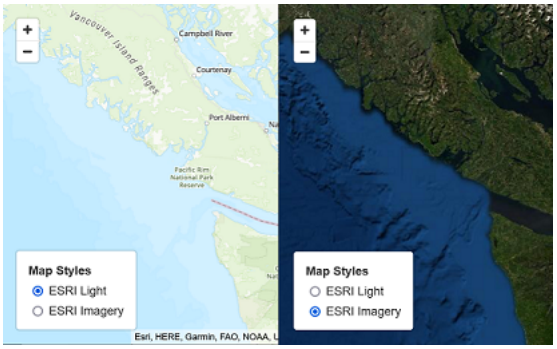
> **Note**
>
> 若要取得有關如何在實踐中使用地圖資源的資訊，請參閱在您的應用程式中使用 Amazon 位置地圖。

以下是如何建立和使用地圖資源的概觀：

1. 您可以透過從資料提供者中選取地圖樣式，在 AWS 帳戶中建立地圖資源。
2. 然後，您可以選取並安裝符SDK合您開發環境和應用程式的。如需有關可用選項的詳細資訊，請參閱有關存取 Amazon 位置的主題。
3. 若要在應用程式中顯示地圖，請將地圖資源與彩現資源庫 (例如「Amplify」 MapLibre、或「七巧板」) 結合使用。如需詳細資訊，請參閱本指南中的使用地圖。
4. 然後，您可以使用 Amazon CloudWatch 和 AWS CloudTrail Amazon 位置等服務來集成監控。如需詳細資訊，請參閱 監控 Amazon Location Service 與 Amazon CloudWatch 及 記錄和監控 AWS CloudTrail。

地圖樣式

當您建立地圖資源時，您必須為該資源選擇地圖樣式。地圖型式定義彩現貼圖的外觀。例如，下圖顯示了 Amazon Location 中不同地圖資源中具有兩種不同樣式的相同資料提供者。一種型式是基於地圖中的向量資料的典型道路型式。另一個包括顯示衛星圖像的光柵數據。型式可能會隨著您在地圖上拉近或縮小而變更，但通常型式具有一致的主題圖。在將部分或所有型式資訊傳遞至地圖彩現資源庫之前，可以取代部分或所有型式資訊。



政治觀點

Amazon 定 Location Service 中的某些地圖樣式支援額外的政治觀點。

> **Note**
>
> 政治觀點必須遵守適用法律，包括您透過 Amazon 定 Location Service 存取之地圖、影像和其他資料和第三方內容之國家或地區對應的相關法律。

下列地圖樣式支援印度 (IND) 政治觀點。

- 埃斯里地圖樣式：
  - 埃斯里導航
  - 埃斯里燈
  - 伊斯里街地圖
  - 深灰色帆布
  - 淺灰色帆布
- 開啟資料對映樣式：
  - 開放式資料標準燈
  - 開放數據標準黑暗
  - 開放式資料視覺化燈
  - 開放數據可視化黑暗

在 Amazon 定 Location Service 主控台中，您可以篩選顯示的樣式，只顯示支援印度政治觀點的樣式。

自訂 Layer

自訂圖層是您可以為地圖型式啟用的其他圖層。目前只有地 VectorEsriNavigation 圖樣式支援POI自訂圖層。

當您啟用POI自定義圖層時，它將為您的地圖添加一組更豐富的地方，例如商店，服務，餐廳，景點和其他景點。依預設，自訂圖層為unset。若要取得更多資訊，請參閱MapConfiguration位置API參考中的。

地圖渲染

若要在應用程式中彩現地圖，您通常會使用地圖彩現資源庫。程式庫有幾個常見的選項可供使用：

- MapLibre— MapLibre 是專門用於渲染交互式地圖的開源庫，是從 Amazon Location Service 渲染地圖的首選方法。 MapLibre 包括從資料來源 (例如 Amazon 位置地圖資源) 呈現點陣式和向量資料的功能。您可以擴展 MapLibre 以在地圖上繪製自己的數據。
- Amplify-Amplify 是一個開源框架，用於構建適用於 Web，iOS，Android 等應用程序的應用程序。如果您的應用程式使用 Amplify，您可以將其擴充以包含 Amazon 位置功能。Amplify 包含專門用於建立 Amazon 位置型應用程式的程式庫，包括轉譯地圖。Amplify 使用 MapLibre 來呈現地圖，但提供 Amazon 定 Location Service 專用的其他功能，以提高使用效率，以及新增搜尋和其他功能。
- 七巧板 — 七巧板是彩現互動式地圖的替代開放原始碼資源庫，類似於。 MapLibre

地圖轉譯程式庫會在執行時期從 Amazon Location Service 提取資料，並根據您選取的地圖資源轉譯地圖資料。map 資源定義將使用的資料提供者和地圖樣式。

下圖顯示了如何在 Amazon Location Service 中使用地圖資源以及地圖渲染庫來創建最終地圖。

1. 您在 Amazon 定 Location Service 中創建地圖資源，使用 AWS Management Console 或 AWS CLI. 這會定義您要使用的資料提供者和地圖型式。
2. 您的應用程式包含地圖彩現資源庫。您可以為地圖渲染資源庫提供要使用的地圖資源的名稱。地圖轉譯程式庫會從 Amazon Location 擷取該地圖資源的資料和樣式資訊，並在螢幕上呈現地圖。

地圖術語

地圖資源

可讓您從選取的提供者存取地圖資料。使用 map 資源擷取包含地圖資料的地圖框，並使用樣式描述元來指定圖徵在地圖上呈現的方式。

底圖

為地圖提供地圖的地理關聯，該地圖儲存為向量圖框圖層。並排圖層包括地理環境 (例如街道名稱、建築物和土地用途)，以供視覺參考。

向量

向量資料是由點、線和多邊形組成的形狀資料。它通常用於在地圖上存儲和顯示道路，位置和區域。矢量形狀也可以用作地圖上的標記的圖標。

光柵

光柵數據是由網格組成的圖像數據，通常由顏色組成。它通常用於在地圖上存儲和顯示連續數據的表示，例如地形，衛星圖像或熱圖。點陣式影像也可以用作影像或圖示。

地圖樣式

向量資料本質上並不包含有關如何繪製資料圖層以建立最終地圖的資訊。地圖樣式會定義資料的顏色和其他型式資訊，以定義彩現時的外觀。地圖資源包括地圖的樣式資訊。

Amazon 定 Location Service 提供符合 Mapbox GL 樣式規格的樣式。

矢量, 瓦片

使用向量形狀儲存地圖資料的拼貼格式。此資料產生的地圖可以根據顯示解析度進行調整，並以多種方式選擇性地彩現圖徵，同時保持較小的檔案大小以獲得最佳效能。

支持的矢量文件格式：Mapbox 矢量瓷磚（MVT）。

字形文件

包含編碼的 Unicode 字符的二進制文件。由地圖渲染器用於顯示標籤。

雪碧文件

一種可攜式網路圖形 (PNG) 影像檔，其中包含小型點陣式影像，並在JSON檔案中具有位置描述。由貼圖渲染器用於彩現地圖上的圖示或材質。

地點搜尋

Amazon 定 Location Service 的一個關鍵功能是搜索地理位置信息的能力。Amazon 位置通過地點索引資源提供此功能。

> **Note**
>
> 如需如何在實務中使用放置索引資源進行搜尋的相關資訊，請參閱使用 Amazon 位置搜尋地點和地理位置資料。

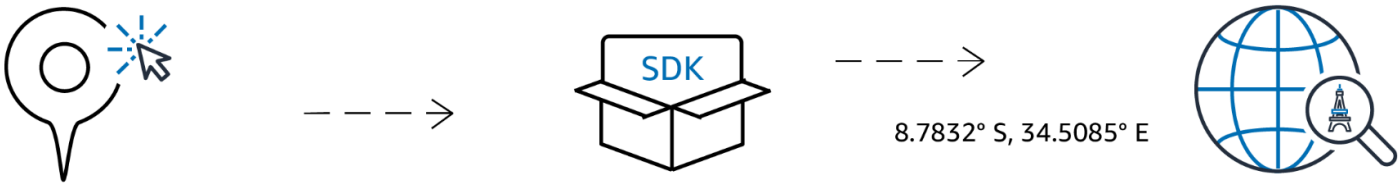
您可以使用地點索引APIs來搜索：

- 景點，例如餐廳和地標。按名稱和可選位置進行搜索，並接收按相關性排序的選項列表。
- 街道地址，接收該地址的緯度和經度。這就是所謂的地理編碼。
- 緯度和經度位置，接收相關的街道地址或有關該位置的其他資訊。這就是所謂的反向地理編碼。
- 部分或拼錯的自由格式文字查詢，通常在使用者鍵入時輸入。這被稱為自動完成，自動建議或模糊匹配。

place 索引包括用於搜尋的資料提供者。

> **Note**
>
> 地圖資料和其他地理位置資訊 (包括確切位置) 可能會因資料提供者而異。最佳做法是為您的地點索引、地圖和其他 Amazon 位置資源使用相同的資料提供者。例如，如果地點索引返回的位置與地圖資源提供的相同位置的位置不匹配，則可以在地圖上看似錯誤位置的位置放置標記。

以下說明如何建立和使用放置索引資源：



1. 首先，您可以透過選取資料提供者在 AWS 帳戶中建立位置索引資源。
2. 然後，您可以選取並安裝符SDK合您開發環境和應用程式的。如需有關可用選項的詳細資訊，請參閱有關存取 Amazon 位置的主題。
3. 開始使用 Amazon 位置APIs。如需詳細資訊，請參閱有關使用「地點」搜尋的主題。
4. 然後，您可以使用 Amazon CloudWatch 和 AWS CloudTrail. 如需詳細資訊，請參閱 the section called “使用監控 CloudWatch” 及 the section called “使 CloudTrail 用 Amazon 位置”。

地理編碼概念

Amazon 位置位置索引提供一個名為的動作 `SearchPlaceIndexForText`，可讓您指定要搜尋的文字。例如，您可以搜尋：

- 地點-搜索**`Paris`**可以返回法國城市的位置。
- 企業 — 搜索**`coffee shop`**可以返回咖啡店的列表，包括他們的名稱和位置。您也可以指定要搜尋的位置，或指定要在其中搜尋的邊界方框，以使結果更具相關性。在這種情況下，在華盛頓州西雅圖市中心提供一個位置，將返回該地區的咖啡店。
- 地址-搜索**`1600 Pennsylvania Ave, Washington D.C.`**可以返回白宮在美國（位於該地址）的位置。

以這種方式搜尋文字通常稱為地理編碼，其中包括尋找地址或地點的地理位置。

Amazon 定 Location Service 也提供名為`SearchPlaceIndexForPosition`的反向地理編碼動作。這需要一個地理位置，並返回有關該位置的地址，業務或其他信息。

搜尋結果

當您在 Amazon 定 Location Service 中成功提出搜尋請求時，會傳回一或多個結果。每個結果都包含一個標籤，即結果的名稱或描述。例如，搜尋時**`coffee shop`**，可能會傳回含有標籤的結果`Hometown Cafe`，告訴您找到了一家名為「家鄉咖啡廳」的咖啡店。搜尋結果通常也會包含結構化

位址 (包含地址編號、單位、街道和郵遞區號等屬性)。視資料提供者而定，它也會包含其他中繼資料，例如國家/地區和時區。

對於商家名稱或類別（例如**`coffee shop`**）進行搜索，您可能希望在地圖上顯示所有返回的結果。對於地址搜索，您可能只想自動使用第一個結果。如需相關性的資訊，請參閱下一個主題。

多重結果和相關性

透過文字搜尋時，Amazon 定 Location Service 通常會找到不只一個結果。例如，搜尋**`Paris`**可能會傳回法國的城市，但也會傳回德克薩斯州的城市。結果依據資料提供者決定的相關性排序。

> **Note**
>
> 結果會以所有提供者的相關性順序傳回。如果您選擇 Esri 或 Grab 做為資料提供者，則結果會包含相關性值，您可以使用這個值瞭解單一要求結果之間的相關性。

指定其他資訊 (例如國家/地區名稱或要搜尋的位置) 可以變更結果的順序、減少結果數目，甚至變更傳回的結果集。例如，在德克薩斯州搜尋某個位置以進行搜尋，傳回`Paris, Texas`的第一個結果比較**`Paris`**有可能`Paris, France`。

在交互式應用程序中，您可以使用相關性來幫助決定是否接受最高結果，還是要求用戶在多個返回結果之間消除歧義。如果第一個結果具有很高的相關性，您可能只接受它作為正確答案。如果有多個高相關性結果或沒有高相關性結果，您可能需要列出結果並讓使用者選擇最佳結果。

地址結果

您可以使用相同的`SearchPlaceIndexForText`動作搜尋 Amazon 定 Location Service 的地址。您提供的資訊越多，傳回的地址就越有可能符合指定的地址。例如，不**`123 Main St`**太可能找到正確的結果**`123 Main St, Anytown, California, 90210`**。

地址有多個屬性，例如街道號碼、街道、城市、地區和郵遞區號等。這些屬性用於在 place 索引中尋找與盡可能多方面相符的位址。找到的屬性越多，相符項目就越相關，傳回的可能性就越大。

> **Note**
>
> 地址結果的相關性取決於結果與輸入相符的程度。這可能是匹配的屬性的數量，也可以是結果與輸入匹配的程度。例如，在資料中找到時`Main St`，的輸入**`123 Main St`**會具有較高的相關性，而不`Maine St`是唯一的結果。 `Maine St`仍會傳回，但可能具有較低的相關性值。

搜尋結果包括完整位址 (123 Main St, Anytown, California, 90210) 的標籤，以及傳回地址的個別結構化屬性。這很有用，因為您可以使用它來填充數據庫中的地址字段，或者檢查結果並查找找到的位置的城市，地區或郵政編碼。

插值

位置索引資料中的位址包含完全相符的位址。例如，假設有一條街道，9th street而一個街區有 2 棟房屋，220並且240，如下圖所示。

資料提供者會使用這兩個已知位址建立地理位置資料。您可以搜索這兩個地址，並找到它們。在資料提供者建立地圖資料之後，讓我們假設在前兩個位址之間加入新房屋。這個新房子給了地址230。如果您搜尋**230 S 9th St**，資料提供者仍會找到結果。而不是使用已知的地址，它會插入已知的地址之間，並從這些地址估計新地址的位置。在這種情況下，可能會假設 230 在 220 和 240 之間（並在街道的同一側）中間，並根據此返回一個大致位置。

> **Note**
>
> 資料提供者會定期使用新地址更新其地理位置資料。在這種情況下，230 S 9th St會被添加到數據提供者數據，但通常會有一個時間段，當一個新的地址已經創建，但尚未添加到數據。

在這種情況下，數據提供者無法判斷新地址是否存在於世界上，因為它尚未存在於數據中，而是從它擁有的信息中提供了最佳答案。此結果稱為內插，並且可以由資料提供者在結果中傳回。如

果interpolated返回false，它是一個已知的地址。如果它返回true，它是一個近似的地址。如果沒有返回，則數據提供者不提供有關結果是否來自插值的信息。

> **Important**
>
> 對於根本不存在的地址，數據提供者也可能返回插值結果。例如，在這種情況下，如果您輸入**232 S 9th St**，提供者會找到這個不存在的地址，並返回接近 230 的位置，但在 240 側。插值地址對於將您帶到正確的位置非常有用，但最好記住它們不是已知的地址。

儲存地理編碼結果

當您建立位置索引資源時，必須指定 [資料儲存] 選項 (IntendedUse在中呼叫API)。可以設置為單次使用或存儲的結果。這是詢問您對結果的預期用途。如果要存儲結果（甚至出於緩存目的），則必須選擇存儲選項，而不是單次使用選項。

> **Note**
>
> 當您選擇儲存的選項 (標示為是，結果將儲存在主控台中，或storage在中選擇CreatePlaceIndexAPI) 時，Amazon 定 Location Service 不會為您儲存結果。這表示您計劃存儲結果。

在查看如何使用查詢到 Amazon 定 Location Service 的結果時，您應始終了解適用的AWS 服務條款。

地方術語

放置索引資源

可讓您選擇資料來源以支援搜尋查詢。例如，您可以搜尋地標、地址或座標。將搜尋查詢傳送至 place 索引資源時，會使用資源配置的資料來源來完成該查詢。

地理编码

地理編碼是採取文本輸入，在地點索引中搜索它並返回結果與位置的過程。

反向地理編碼

反向地理編碼是指取得位置並從位置索引中傳回該位置相關資訊的程序，例如該位置的地址、城市或商家。

相關性

相關性是結果與輸入匹配的程度。這不是衡量正確性的衡量標準。

插值

插補是使用已知位址位置做為導引點來尋找未知位址的程序。

ISO國家代碼

Amazon 定 Location Service 地點使用國際標準化組織 (ISO) 3166 個國家/地區代碼來代表國家或地區。

若要尋找特定國家或地區的代碼，請使用ISO線上瀏覽平台。

路由

本節提供有關使用 Amazon 定 Location Service 進行路由的概念概觀。

> **Note**
>
> 如需如何在實務中使用路由資源的相關資訊，請參閱使用 Amazon 定 Location Service 計算路線。

路線計算器資源

路線計算器資源使您可以根據所選數據提供商的 up-to-date 道路網絡和實時交通信息查找路線並估算旅行時間。

您可以使用「路APIs由」建立功能，讓您的應用程式要求任意兩個位置之間路線的行駛時間、距離和幾何圖形。您還可以在單個請求中使用路線API來請求一組出發地和目的地之間的行駛時間和距離，以計算矩陣。

以下說明如何建立和使用路線計算機資源：

1. 首先，您可以透過選取資料提供者，在 AWS 帳戶中建立路由計算機資源。
2. 然後，您可以選取並安裝符SDK合您開發環境和應用程式的。
3. 開始使用 Amazon 定位路線APIs。有關如何使用路由的詳細資訊APIs，請參閱（詳見）主題使用 Amazon 定 Location Service 計算路線。
4. 然後，您可以使用 Amazon CloudWatch 和 AWS CloudTrail. 如需詳細資訊，請參閱 監控 Amazon Location Service 與 Amazon CloudWatch 及 記錄和監控 AWS CloudTrail。

計算路線

Amazon Location 路線計算器資源提供了一`CalculateRoute`個名為的動作，可用於在兩個地理位置 (出發地和目的地) 之間建立路線。計算出的路線包括用於在地圖上繪製路線的幾何圖形，以及佈線的總時間和距離。

使用航點

創建路由請求時，可以向路線添加其他航點。這些是出發地和目的地之間的點，可作為路線上的停靠點。路線將通過指定的每個航點計算。從請求中的一個點到下一個點的路由稱為 a `Leg`。每個段都包括路線中該部分的距離、時間和幾何圖形。

> **Note**
>
> 航點按請求中給出的順序進行路由。它們不會針對最短路徑重新排序。請參閱〈〉一規劃路線節，以取得有關尋找最短路徑的資訊。

您可以在單個請求中包含多達 25 個航點以計算路線。

交通和出發時間

計算路線時，Amazon 定 Location Service 會將流量納入考量。它會考慮的流量是根據您指定的時間而定。您可以指定立即出發，也可以提供您要離開的特定時間，這將通過在指定時間調整交通情況來影響路線結果。

> **Note**
>
> 您可以使用出發時間和路線響應時間計算到達時間，例如估計司機的到達時間。

如果您希望 Amazon Location 不考慮流量，請不要指定出發時間，也不要指定立即出發。這將計算假設路線的最佳交通狀況的路線。

旅行模式選項

您可以在使用 Amazon 定 Location Service 計算路線時設定行駛模式。預設行駛模式為汽車，但您可以選擇貨車或步行。

如果您指定汽車或卡車模式，您也可以指定其他選項。

對於汽車模式，您可以指定要避開收費公路或渡輪。這將試圖避開渡輪和收費公路，但是如果它們是到達目的地的唯一途徑，那麼它們仍然會沿著它們行駛。

對於卡車模式，您也可以避開渡輪和收費公路，但此外，您可以指定卡車的大小和重量，以避免無法容納卡車的路線。

規劃路線

您可以使用 Amazon 定 Location Service 為您的路線規劃和優化軟體建立輸入。您可以為一組出發位置和一組目的地位置之間的路線創建路線結果，包括旅行時間和行駛距離。這就是所謂的創建路由矩陣。

> **Note**
>
> 路線規劃和優化軟件可以解決許多不同的情況。例如，規劃軟體可以使用點之間的一組時間和距離來計算在每個點停止的最短路徑，從而為單一駕駛員提供有效的路徑。或者，規劃軟體可用於在多輛卡車之間分隔停靠點，提供整個車隊的效率，或確保在所需的時間範圍內拜訪每位客戶。Amazon 位置以有效的方式提供路由功能，讓規劃軟體能夠完成其任務。

例如，指定出發位置 A 和 B，以及目的地位置 X 和 Y，Amazon 定 Location Service 將返回從 A 到 X、A 到 Y、B 到 X 的路線的旅行時間和旅行距離，以及從 B 返回 Y 的路線的旅行時間和旅行距離。

與計算單條路線一樣，您可以計算具有不同交通方式，避免和交通狀況的路線。例如，您可以指定車輛是 35 英呎長的卡車，計算的路線將使用這些限制來確定行駛時間和行駛距離。您不能在路由矩陣計算中包含航點。

傳回的結果數 (以及計算的路線) 是出發位置數乘以目的地位置數目。您需按計算出的每條路線收費，而不是每個服務的請求，因此包含 10 個出發地和 10 個目的地的路線矩陣將以 100 條路線計費。

路線術語

路線計算器資源

一種 AWS 資源，可讓您在地圖上估算行駛時間、距離和繪製路線，其中包含來自您選擇的資料提供者的交通和道路網路資料。

使用路線計算器資源，您可以計算不同交通方式，彎路和交通狀況的路線。

路由

路線包含從出發位置、航點位置和目的地位置沿著路徑行駛時使用的詳細資訊。

路線中的詳細資訊範例包括：

- 從一個位置到另一個位置的距離。
- 從一個位置移動到下一個位置所需的時間。
- 表示路線路徑的 LineString 幾何圖形。

如需有關路由的詳細資訊，請參閱 Amazon 定 Location Service 路由API參考中的 CalculateRoute 作業回應語法。

路由矩陣

從一組出發位置到一組目的地位置的路線列表。有用的輸入到路線規劃或優化軟件。

如需有關計算路由矩陣的詳細資訊，請參閱 Amazon 定 Location Service 路由API參考中的 CalculateRouteMatrix 操作語法。

`LineString`幾何

Amazon 位置路線由一條或多條路線組成（整條路線中從一個航點到另一個航點的路線）。每條段的幾何圖形都是一條聚合線，表示為 a `LineString`。A `LineString` 是位置的有序數組，可用於在地圖上繪製路線。

以下是一個`LineString`帶有三點的例子：

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

Waypoint

航點是中間位置，充當起飛位置和目的地位置之間的路線上的停靠點。路線上的中途停留順序遵循您在請求中提供航點位置的順序。

腿

單腿是從一個位置到另一個位置的旅程。如果這些位置不在道路上，它們會移動到最近的道路上。路線中的段數小於位置總數之一。

沒有航路點的路線由從出發位置到目的地的單程組成。具有 1 個航點的路線由 2 條腿組成，從出發位置到航點，然後從航點到目的地。

步驟

一個步驟是一條腿的子部分。每個步驟都會提供該步驟的摘要資訊。

地理圍欄和跟踪器

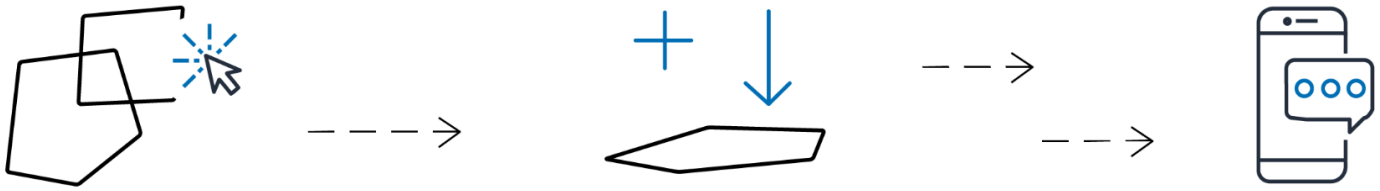
本節提供使用 Amazon 定 Location Service 圍欄和追蹤器的概念和概觀。地理圍欄是多邊形邊界，當裝置或位置移入和移出區域時，您可以使用這些邊界來收到通知。追蹤器資源用於儲存和更新裝置移動時的位置。

> **Note**
>
> 有關如何在實踐中使用地理圍欄和跟踪器的信息，請參閱。使用 Amazon 位置對感興趣的區域進行地理圍欄

地理圍欄

地理圍欄收集資源可讓您儲存和管理地理圍欄 — 地圖上的虛擬邊界。您可以根據地理圍欄收集資源評估位置，並在位置更新超過地理圍欄集合中任何地理圍欄的邊界時收到通知。

以下說明如何建立和使用地理圍欄集合資源：



1. 在您的帳戶中建立地理圍欄集合資 AWS 源。
2. 將地理圍欄添加到該集合。您可以使用 Amazon 位置主控台上的地理圍欄上傳工具，或使用 Amazon 位置地理柵欄來完成此操作。API如需有關可用選項的詳細資訊，請參閱存取 Amazon 位置。

   地理圍欄可以通過多邊形或圓來定義。使用多邊形來查找設備何時進入特定區域。使用圓來尋找裝置在某個點的特定距離 (半徑) 內的時間。
3. 您可以開始針對所有地理圍欄評估位置。當位置更新跨越一個或多個地理圍欄的邊界時，您的地理圍欄收集資源會在 Amazon 上發出以下地理圍欄事件類型之一： EventBridge
   - ENTER— 每個地理圍欄會產生一個事件，其中位置更新通過輸入它的邊界。
   - EXIT— 為每個地理圍欄產生一個事件，其中位置更新通過退出其邊界。

   如需詳細資訊，請參閱the section called “對事件做出反應 EventBridge”。您還可以使用 Amazon CloudWatch 和 AWS CloudTrail. 如需詳細資訊，請參閱 the section called “使用監控 CloudWatch” 及 the section called “使 CloudTrail 用 Amazon 位置”。

例如，如果您正在跟踪卡車車隊，並且希望在卡車進入任何倉庫的特定區域時收到通知。您可以為每個倉庫周圍的區域建立地理圍欄。然後，當卡車向您發送更新的位置時，您可以使用 Amazon 定 Location Service 評估這些位置，並查看卡車是否已進入（或退出）其中一個地理圍欄區域。

> **Note**
>
> 您的費用是依據您評估的地理圍欄集合數量計費。您的帳單不受每個集合中的地理圍欄數量的影響。由於每個地理圍欄集合可能包含多達 50,000 個地理圍欄，因此您可能希望將地理圍欄合併為更少的集合，以減少地理圍欄評估的成本。產生的事件將包含集合中個別地理圍欄的 ID，以及集合的 ID。

地理圍欄事件

您正在監視的位置會以稱為 a 的 ID 參照 `DeviceId` (位置稱為裝置位置)。您可以將設備位置列表直接發送到地理圍欄收集資源進行評估，也可以使用跟踪器。有關追蹤器的更多資訊，請參閱下一節。

只有當設備進入或退出地理圍欄時，您才會收到事件（通過 Amazon EventBridge），而不是每次更改位置。這表示您通常會收到事件，而且必須比每次裝置位置更新少得多回應事件。

> **Note**
>
> 對於特定的第一個位置評估DeviceID，假定該設備以前不在任何地理圍欄中。因此，如果在集合中的地理圍欄中，第一次更新將生成一個ENTER事件，如果沒有，則不會產生事件。

為了計算裝置是否已進入或退出地理圍欄，Amazon 定 Location Service 必須保留該裝置的先前位置狀態。此職位狀態會儲存 30 天。在未更新裝置的 30 天後，新的位置更新會被視為第一次更新位置。

追蹤器

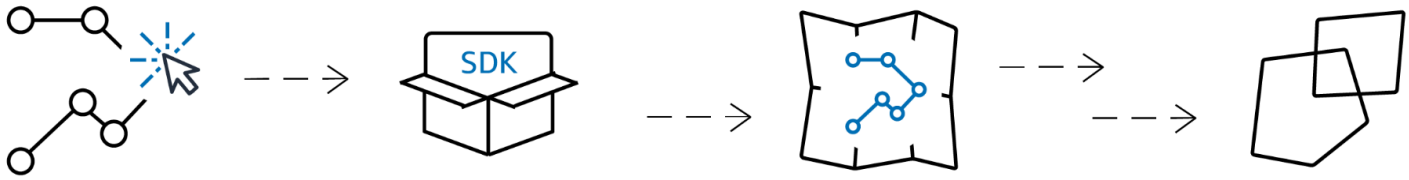
追蹤器會儲存裝置集合的位置更新。跟踪器可用於查詢設備的當前位置或位置歷史記錄。它存儲更新，但通過在存儲之前過濾位置來減少存儲空間和視覺噪聲。

儲存在追蹤器資源中的每個位置更新，都可以包含一個定位準確度的度量，以及最多 3 個與您要儲存的位置或裝置有關的中繼資料欄位。元數據存儲為鍵值對，並可以存儲諸如速度，方向，輪胎壓力或引擎溫度等信息。

> **Note**
>
> 追蹤器儲存空間使用 AWS 擁有的金鑰自動加密。您可以使用自己管理的KMS金鑰新增另一層加密，以確保只有您可以存取您的資料。如需詳細資訊，請參閱Amazon 定 Location Service 的靜態資料加密。

跟踪器位置過濾和存儲自己很有用，但跟踪器在與地理圍欄配對時特別有用。您可以將跟踪器鏈接到一個或多個地理圍欄收集資源，並根據這些集合中的地理圍欄自動評估位置更新。正確使用過濾也可以大大降低地理圍欄評估的成本。

下圖顯示了如何創建和使用跟踪器資源：



1. 首先，您在 AWS 帳戶中創建一個跟踪器資源。
2. 接下來，決定如何將位置更新發送到跟踪器資源。用AWS SDKs於將追蹤功能整合到您的行動應用程式中。或者，您可以使用MQTT按照 step-by-step 指示進行跟踪。MQTT
3. 現在，您可以使用跟踪器資源記錄位置歷史記錄並在地圖上將其視覺化。
4. 您還可以將跟踪器資源鏈接到一個或多個地理圍欄集合，以便發送到跟踪器資源的每個位置更新都會根據所有鏈接的地理圍欄集合中的所有地理圍欄自動評估。您可以在 Amazon 位置主控台的追蹤器資源詳細資訊頁面上連結資源，或使用 Amazon 位置追蹤API器來連結資源。
5. 然後，您可以使用 Amazon CloudWatch 和 AWS CloudTrail. 如需詳細資訊，請參閱 the section called “使用監控 CloudWatch” 及 the section called “使 CloudTrail 用 Amazon 位置”。

使用帶有地理圍欄的跟踪器

跟踪器在與地理圍欄配對時提供額外的功能。您可以透過 Amazon Location 主控台或將追蹤器與地理圍欄集合建立關聯API，以自動評估追蹤器位置。每次跟踪器收到更新的位置時，該位置將根據集合中的每個地理圍欄進行評估，並在 Amazon 中生成適當的ENTER和EXIT事件。 EventBridge您也可以將篩選套用至追蹤器，根據篩選條件，您可以僅評估有意義的位置更新，以降低地理圍欄評估的成本。

如果您在收到某些位置更新之後將追蹤器與地理圍欄集合產生關聯，則在關聯之後的第一個位置更新會被視為地理圍欄評估的初始更新。如果它在地理圍欄內，您將收到一個ENTER事件。如果它不在任何地理圍欄內，則無論以前的狀態如何，都不會收到EXIT事件。

位置篩選

追蹤器可以自動篩選傳送給他們的倉位。您可能需要過濾掉某些設備位置更新的原因有幾個。如果您的系統每分鐘左右都會傳送報告，您可能想要依時間篩選裝置，以便每 30 秒儲存和評估位置。即使您更頻繁地監視，也可能需要篩選位置更新以清除硬體的GPS雜訊。GPS位置位置本質上是嘈雜的。它們的準確性並非 100% 完美，因此即使是靜止的設備似乎也略微移動。在低速時，這種抖動會導致視覺混亂，如果設備靠近地理圍欄的邊緣，則可能會導致錯誤的進入和退出事件。

位置篩選可在追蹤器接收位置更新時運作，減少裝置路徑中的視覺雜訊 (抖動)、減少錯誤的地理圍欄進入和退出事件的數量，並透過減少儲存的位置更新和觸發地理圍欄評估的次數來協助管理成本。

追蹤器提供三種位置篩選選項，協助管理成本並減少位置更新中的抖動。

- 基於準確性 — 與任何提供精度測量的設備一起使用。大多數GPS和移動設備提供此信息。每個位置測量的準確性都受到許多環境因素的影響，包括GPS衛星接收、景觀以及 wifi 和藍牙設備的距離。大多數設備（包括大多數移動設備）都可以提供測量精度的估計以及測量結果。透AccuracyBased過篩選功能，如果裝置的移動速度低於測量的準確度，Amazon Location 會忽略位置更新。例如，如果裝置的連續兩次更新精確度範圍為 5 m 和 10 m，則 Amazon Location 會在裝置移動小於 15 公尺時忽略第二次更新。Amazon 位置既不會針對地理圍欄評估被忽略的更新，也不會存儲它們。

  如果未提供精度，則將其視為零，並且測量被認為是完全準確的，並且不會對更新應用任何濾波。

  > **Note**
  >
  > 您可以使用基於精確度的過濾來刪除所有過濾。如果您選取精確度篩選，但將所有準確度資料覆寫為零，或完全忽略準確度，則 Amazon Location 不會篩選出任何更新。

  在大多數情況下，基於準確的篩選是篩選職位更新的理想選擇，可在過濾不需要的更新的同時取得追蹤位置的平衡，進而降低成本。

- 以距離為基礎 — 當您的裝置未提供精確度量測，但您仍希望利用濾波功能來減少抖動並管理成本時使用。 DistanceBased篩選會忽略裝置移動小於 30 公尺 (98.4 英呎) 的位置更新。當您使用DistanceBased位置篩選時，Amazon Location 既不會針對地理圍欄評估這些被忽略的更新，也不會儲存更新。

  大多數移動設備的準確度（包括 iOS 和 Android 設備的平均準確度）在 15 米內。在大多數應用程式中，DistanceBased篩選可以減少在地圖上顯示裝置軌跡時，位置不準確的影響，以及當裝置靠近地理圍欄邊界時，多個連續進入和退出事件的彈跳效果。它還可以通過減少對鏈接地理圍欄進行評估或檢索設備位置的調用來幫助降低應用程序的成本。

  如果您想要進行篩選，但您的裝置並未提供精確度量測，或者您想要篩選出大量更多的更新，而不是以精確度為基礎，則基於距離的篩選非常有用。

- 以時間為基礎 — (預設) 當您的裝置非常頻繁地傳送位置更新 (每 30 秒超過一次)，且您希望在不儲存每次更新的情況下達到近乎即時的地理圍欄評估時使用。在TimeBased過濾中，每個位置更新都會根據鏈接的地理圍欄集合進行評估，但不會存儲每個位置更新。若您的更新頻率超過 30 秒，則每個唯一裝置 ID 每 30 秒只會存放一次更新。

當您想要儲存較少的職位，但希望針對關聯的地理圍欄集合評估每個位置更新時，以時間為基礎的篩選特別有用。

> **Note**
>
> 在決定篩選方法和職位更新頻率時，請注意追蹤應用程式的成本。您需支付每次位置更新的費用，以及針對每個連結的地理圍欄集合評估位置更新一次的費用。例如，使用基於時間的過濾時，如果您的跟踪器鏈接到兩個地理圍欄集合，則每個位置更新將計為一個位置更新請求和兩個地理圍欄集合評估。如果您每 5 秒報告裝置的位置更新一次，並使用基於時間的篩選，則每部裝置每小時需支付 720 個位置更新和 1,440 次地理圍欄評估的費用。

地理圍欄術語

地理圍欄合集

包含零個或多個地理圍欄。它能夠通過在要求時發出入口和退出事件來評估設備位置對其地理圍欄進行地理圍欄監視。

地理圍欄

定義地圖上虛擬邊界的多邊形或圓幾何圖形。

多邊形幾何

Amazon 位置地理圍欄是地理區域的虛擬邊界，表示為多邊形幾何圖形或圓形。

圓是一個周圍有一定距離的點。當您想要通知裝置是否在某個位置的特定距離內時，請使用圓圈。

多邊形是由 1 個或多個線性環組成的陣列。當您要定義裝置通知的特定邊界時，請使用多邊形。線性環是由四個或更多頂點組成的陣列，其中第一個頂點和最後一個頂點相同以形成封閉邊界。每個頂點都是形式的二維點 *`[longitude, latitude]`*，其中經度和緯度的單位為度。頂點必須圍繞多邊形以逆時鐘順序列出。

> **Note**
>
> Amazon 定 Location Service 不支援具有多個環的多邊形。這包括孔、孤立物件或多重多邊形。Amazon 位置也不支援順時針纏繞或穿過反經絡的多邊形。

以下是單線性外環的範例：

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

追蹤器術語

追蹤器資源

從裝置接收位置更新的 AWS 資源。跟踪器資源為位置查詢提供支持，例如當前和歷史設備位置。將跟踪器資源鏈接到地理圍欄集合會自動根據鏈接的地理圍欄集合中的所有地理圍欄評估位置更新。

追蹤職位資料

跟踪器資源隨著時間的推移存儲有關您設備的信息。這些資訊包括一系列的位置更新，其中每個更新都包含位置、時間和選用的中繼資料。元數據可以包括位置的準確性和最多三個鍵值對，以幫助您跟踪有關每個位置的關鍵信息，例如速度，方向，輪胎壓力，剩餘燃料或您正在跟踪的車輛的引擎溫度。追蹤器會維護裝置位置記錄 30 天。

位置篩選

位置篩選可以透過篩選掉在儲存或根據地理柵欄評估更新之前，不會提供有價值資訊的職位更新，協助您控制成本並改善追蹤應用程式的品質。

您可以選擇`AccuracyBased`、`Distance Based`或`TimeBased`篩選。依預設，位置篩選設定為`TimeBased`。

您可以在建立或更新追蹤器資源時設定職位篩選。

RFC3339 時間戳記格式

Amazon Location Service 追蹤器使用 RFC3339 格式，日期和時間遵循國際標準化組織 (ISO) 8601 格式。

格式為「YYYY-毫米 —: 毫米:SSS+00:00」: DDThh

- YYYY-MM-DD— 代表日期格式。
- T— 指示時間值將遵循。
- hh:mm:ss.sss— 代表 24 小時格式的時間。
- Z— 表示所使用的時區為UTC，之後可能會出現與UTC時區的偏差。
- +00:00— 選擇性地指示與UTC時區的偏差。例如，+ 01：00 表示 UTC + 1 小時。

範例

對於 2020 年 7 月 2 日下午 12 時 15 分 20 分，並將時區額外調整 1 小UTC時。

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

使用 Amazon 定 Location Service 的常見使用案例

Amazon 定 Location Service 可讓您建立各種應用程式，從資產追蹤到基於位置的行銷。以下是常見的使用案例：

用戶參與度和地理營銷

使用位置資料建立解決方案，以改善使用者與行銷以鎖定目標客戶的互動。例如，Amazon Location 可以觸發事件，當客戶在行動應用程式上訂購咖啡時就在附近時提示通知。此外，您可以構建地理定位功能，以便零售商可以向目標商店附近的客戶發送 discount 代碼或數字傳單。

資產追蹤

建置資產追蹤功能，協助企業瞭解其產品、人員和基礎架構的目前和歷史位置。透過資產追蹤功能，您可以建置多種解決方案，以最佳化遠端人員配置、確保途中託運安全，以及最大化派遣效率。

送貨

將位置功能集成到交付應用程序中，以存儲，跟踪和協調出發地點，交付車輛及其目的地。例如，內建 Amazon Location 功能的食品外送應用程式具有位置追蹤和地理圍欄功能，可在送貨司機在附近時自動通知餐廳。這樣可以減少等待時間，並有助於維持外送食物的質量。

本主題為您提供可使用 Amazon 位置建置之應用程式的架構和步驟概觀。

主題

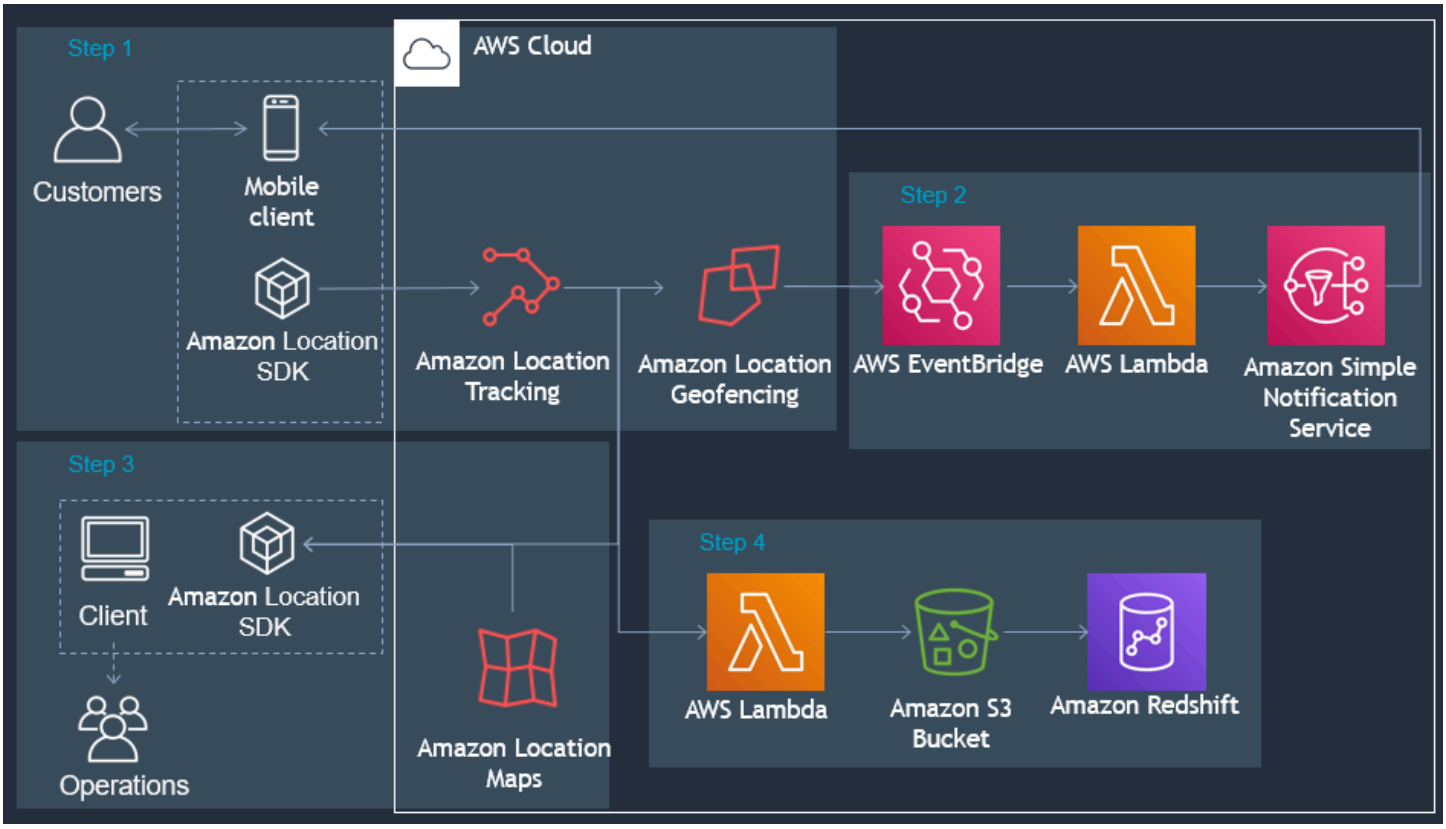
- 用戶參與和地理營銷應用
- 資產追蹤應用
- 交付應用

用戶參與和地理營銷應用

以下是使用 Amazon 位置的使用者參與和地理行銷應用程式架構的圖例：

使用此架構，您可以：

- 根據目標的接近程度啟動事件，以便您可以將其發送給附近的客戶，或吸引最近離開您機構的人（稱為地理定位）。
- 在地圖上可視化客戶設備位置，以監控隨時間推移的趨勢。
- 儲存可隨時間分析的客戶裝置位置。
- 分析位置歷史記錄以確定趨勢和優化機會。



以下是構建用戶參與和地理營銷應用程序所需步驟的概述：

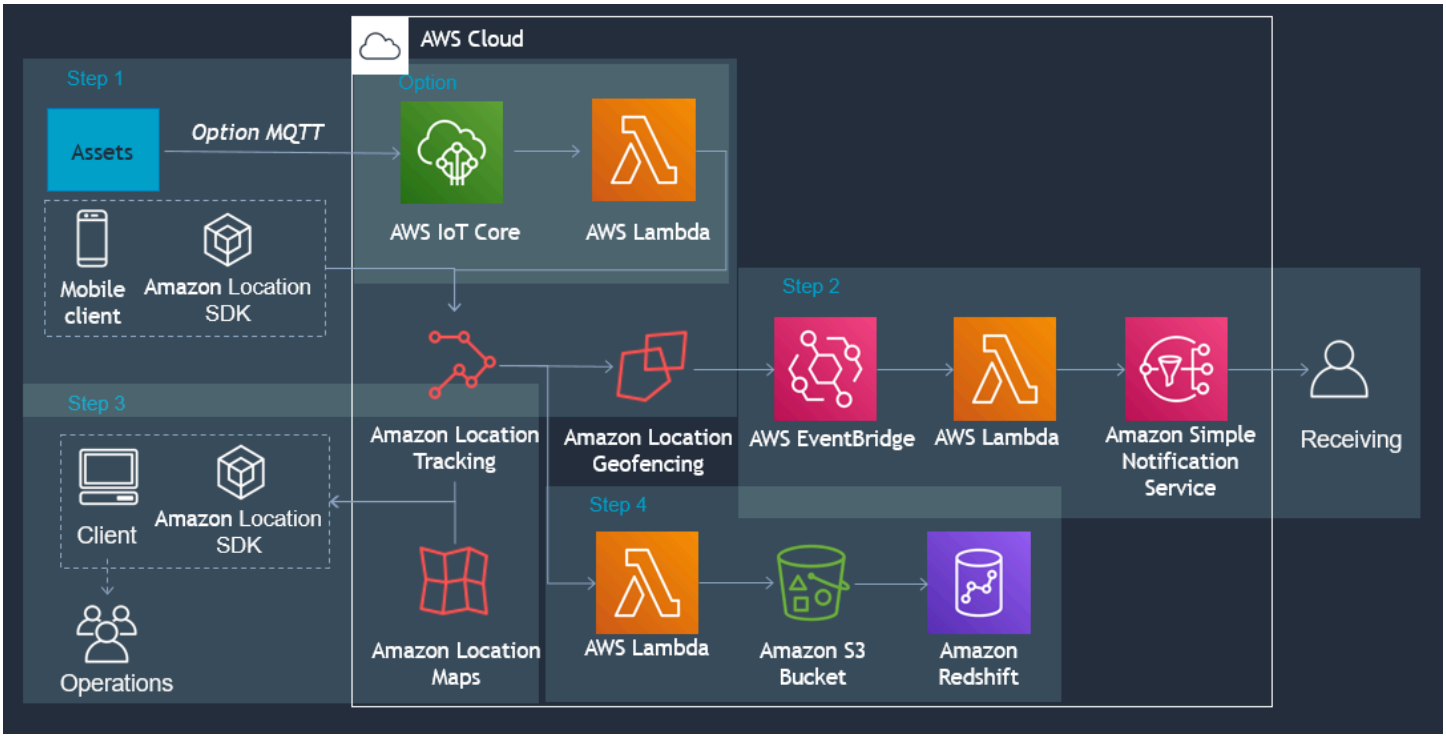
1. 在地理圍欄集合中創建您的地理圍欄並將跟踪器鏈接到它們。如需詳細資訊，請參閱the section called “地理圍欄和跟踪”。
2. 設定 Amazon EventBridge 以傳送通知給進入或離開感興趣地理圍欄區域的客戶。如需詳細資訊，請參閱the section called “對事件做出反應 EventBridge”。
3. 在地圖上顯示客戶位置和地理圍欄。如需詳細資訊，請參閱使用地圖。
4. 將位置數據保存到長期存儲中以供進一步分析。
5. 一旦你建立了你的應用程序，你可以使用 Amazon CloudWatch 和 AWS CloudTrail 管理你的應用程序。如需詳細資訊，請參閱 the section called “使用監控 CloudWatch” 和 the section called “使 CloudTrail 用 Amazon 位置”。

資產追蹤應用

以下是使用 Amazon 位置的資產追蹤應用程式架構圖例：

使用此架構，您可以：

- 在地圖上顯示資產位置以說明大圖。例如，使用歷史位置或事件顯示熱圖，以協助作業或規劃團隊。
- 根據資產接近程度來啟動事件，以向收貨部門提供通知，以準備出貨抵達並縮短處理時間。
- 儲存資產位置以在後端應用程式中啟動動作，或隨時間分析資料。
- 分析位置歷史記錄以確定趨勢和優化機會。



以下提供建置資產追蹤應用程式所需步驟的概觀：

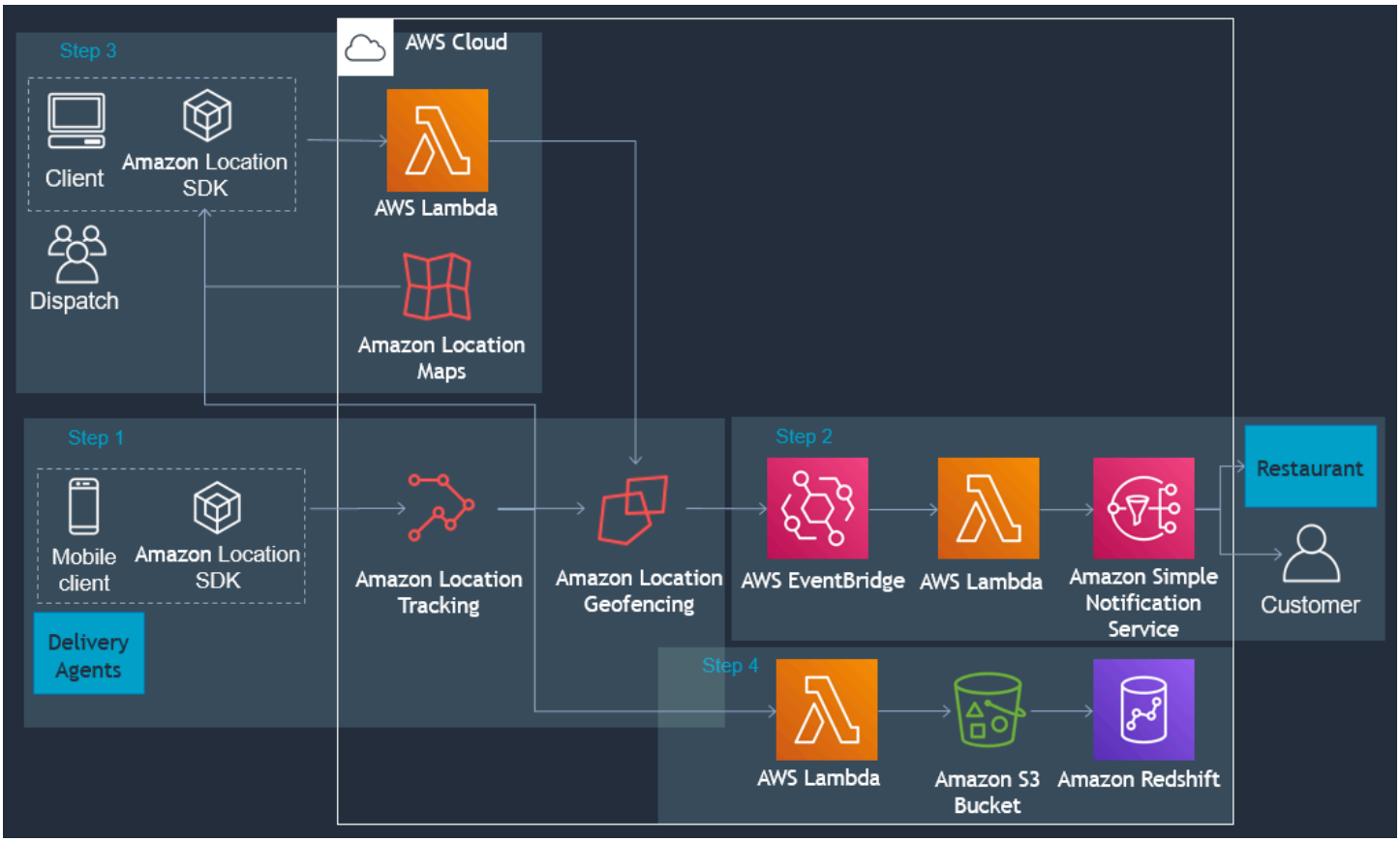
1. 在地理圍欄集合中創建您的地理圍欄並將跟踪器鏈接到它們。如需詳細資訊，請參閱the section called “地理圍欄和跟踪”。
2. EventBridge 將 Amazon 設定為傳送通知或啟動程序。如需詳細資訊，請參閱the section called “對事件做出反應 EventBridge”。
3. 在地圖上顯示您追蹤的資產和作用中的地理圍欄。如需詳細資訊，請參閱使用地圖。
4. 將位置數據保存在長期存儲中以供進一步分析。
5. 一旦你建立了你的應用程序，你可以使用 Amazon CloudWatch 和 AWS CloudTrail 管理你的應用程序。如需詳細資訊，請參閱 the section called “使用監控 CloudWatch” 和 the section called “使 CloudTrail 用 Amazon 位置”。

交付應用

以下是使用 Amazon 位置的交付應用程式架構圖例。

使用此架構，您可以：

- 根據傳送代理程式的接近程度來啟動事件，以便及時收件，並在客戶遞送到達時收到通知。
- 在地圖上以近乎實時的方式顯示司機位置，以及接送和下降位置，以顯示調度團隊的大局。
- 儲存傳遞代理程式的位置，以便您可以在後端應用程式中對其執行動作，或隨著時間的推移對其進行分析。
- 分析位置歷史記錄以確定趨勢和優化機會。



以下是建置傳遞應用程式所需步驟的概觀：

1. 建立您的地理圍欄集合，並將追蹤的裝置連結至收藏。若要取得更多資訊，請參閱the section called “地理圍欄和跟踪”。
2. 創建一個 AWS Lambda 功能以在預訂訂單時自動添加和刪除地理圍欄。

3. EventBridge 將 Amazon 設定為傳送通知或啟動程序。如需詳細資訊，請參閱the section called “對事件做出反應 EventBridge”。
4. 在地圖上顯示追蹤的資產和作用中的地理圍欄。如需詳細資訊，請參閱使用地圖。
5. 將位置數據保存到長期存儲中以供進一步分析。
6. 一旦你建立了你的應用程序，你可以使用 Amazon CloudWatch 和 AWS CloudTrail 管理你的應用程序。如需詳細資訊，請參閱 the section called “使用監控 CloudWatch” 和 the section called “使 CloudTrail 用 Amazon 位置”。

什麼是資料提供者？

使用 Amazon 定 Location Service，透過您的 AWS 帳戶存取來自多個資料供應商的地理位置資源，而不需要第三方合約或整合。這可協助您專注於建置應用程式，而不必管理第三方帳戶、憑證、授權和帳單。

下列 Amazon 定位服務使用資料提供者。

- 地圖 — 建立地圖資源時，從不同的地圖提供者中選擇樣式。您可以使用地圖資源建立互動式地圖以視覺化資料。
- 地點 — 建立位置索引資源時，請選擇資料提供者，以支援地理編碼、反向地理編碼和搜尋的查詢。
- 路由 — 當您建立路線計算器資源時，選擇資料提供者以支援查詢不同地理位置和應用程式中的路線計算。透過您選擇的資料供應商，Amazon Location Service 可讓您根據 up-to-date 道路網路資料、即時交通資料、計劃關閉和歷史流量模式來計算路線。

每個供應商都會使用不同的方式來收集和策劃其資料。他們也可能在世界上不同地區擁有不同的專業知識。本節提供有關我們數據提供商的詳細信息。您可以根據自己的喜好選擇任何數據提供商。

使用 Amazon 定位服務資料供應商時，請務必閱讀條款。如需詳細資訊，請參閱AWS服務條款。如需 Amazon 位置如何保護您隱私權的詳細資訊，請參閱the section called “資料隱私”本節。

數據提供商的覆蓋範圍和

下表顯示每個資料提供者的高層級涵蓋範圍和功能。

資料提供者	地理覆蓋	功能覆蓋	AWS 區域
埃斯里	全球服務	地圖，地方，路線	提供 Amazon 位置服務的所有區域。
抓斗	東南亞	地圖，地方，路線	亞太區域 (新加坡) `ap-southeast-1` 、僅限。
HERE	全球服務	地圖，地方，路線	提供 Amazon 位置服務的所有區域。
開啟資料	全球服務	地圖	提供 Amazon 位置服務的所有區域。

如需有關每個資料提供者特定功能的詳細資訊，請參閱資料提供者的功能。

每個資料提供者都會以不同的方式收集和產生資料。您可以在下列主題中深入瞭解其涵蓋範圍：

- 覆蓋範圍：埃斯里
- 覆蓋範圍：抓斗
- 覆蓋範圍:HERE
- 涵蓋範圍：開放數據

如果您遇到資料問題，並想要向資料提供者報告錯誤，請參閱下列主題：

- 向 Esri 報告錯誤
- GrabMaps 資料錯誤報告
- 錯誤報告給 HERE
- 錯誤報告和貢獻開放資料

地圖樣式

每個資料提供者都提供一組地圖型式來彩現其提供的地圖資料。例如，型式可能包含衛星影像，或者可能會進行最佳化以展示用於導覽的道路。您可以在以下主題中找到每個提供者的樣式清單和範例。

- Esri 地圖樣式
- 抓取地圖樣式
- HERE地圖型式
- 開啟資料對映型式

每個資料提供者的詳細資訊

下列連結提供有關每個資料提供者的詳細資訊。

- 埃斯里
- GrabMaps
- HERE技術
- 開啟資料

埃斯里

Amazon 定 Location Service 使用 Esri 的定位服務，協助 AWS 客戶有效地使用地圖、地理編碼和計算路線。Esri 的定位服務採用高質量，權威性和 ready-to-use 位置數據構建，由製圖師，地理學家和人口統計學家的專家團隊策劃。

如需其他功能資訊，請參閱 Amazon 定 Location Service 資料提供者上的 Esri。

主題

- Esri 地圖樣式
- 覆蓋範圍：埃斯里
- 使用條款和數據歸因：Esri
- 向 Esri 報告錯誤

Esri 地圖樣式

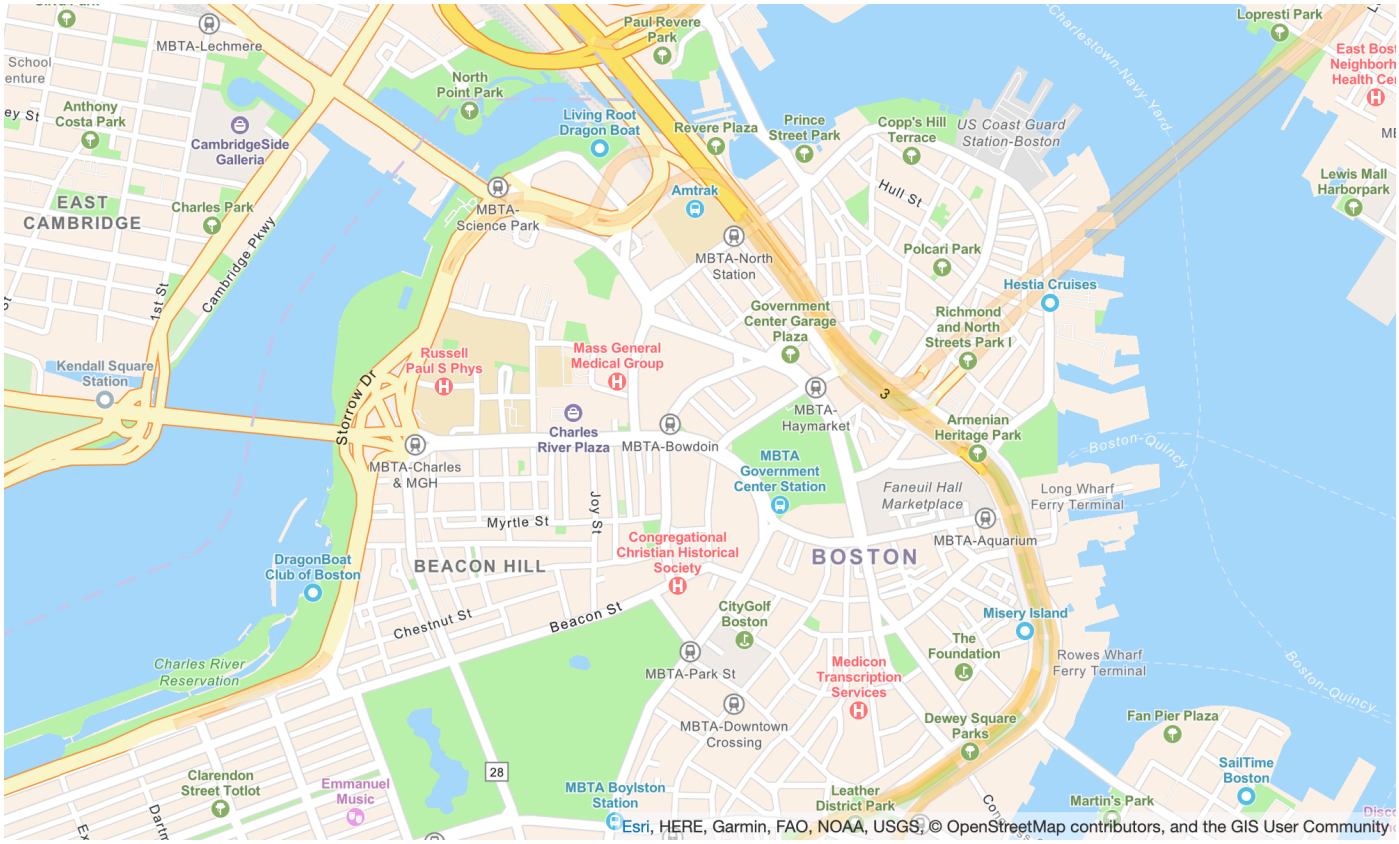
Amazon 定 Location Service 在建立地圖資源時支援以下 Esri 地圖樣式。

> **Note**
>
> 不支援此區段中未列出的 Esri 地圖樣式。

Esri 向量樣式支援替代政治觀點。

Esri Navigation

埃斯里導航



地圖樣式名稱：`VectorEsriNavigation`

這張地圖為世界提供了詳細的底圖，象徵著自定義導航地圖樣式，專為日間在移動設備中使用而設計。

這張綜合街道地圖包括高速公路、主要道路、次要道路、鐵路、水景、城市、公園、地標、建築足跡和行政邊界。此地圖中的向量圖框圖層是使用與世界街道地圖和其他 Esri 基準地圖相同的資料來源建立的。透過在中設定`POI`圖層來啟用圖層，CustomLayers以利用其他地點資料。

如需詳細資訊，請參閱 Esri 網站上的 Esri 世界導航。

> **Note**
>
> 上`VectorEsriNavigation`圖的地圖已啟用圖`POI`層。

字型

Amazon 位置使用GetMapGlyphs. 以下是此地圖的可用字體堆疊：

- 宋體斜體
- 宋體定期
- 宋體粗體
- 宋體萬國碼 MS 粗體
- 宋體國際碼 MS 常規

Esri Imagery

埃斯里圖像

地圖樣式名稱：RasterEsriImagery

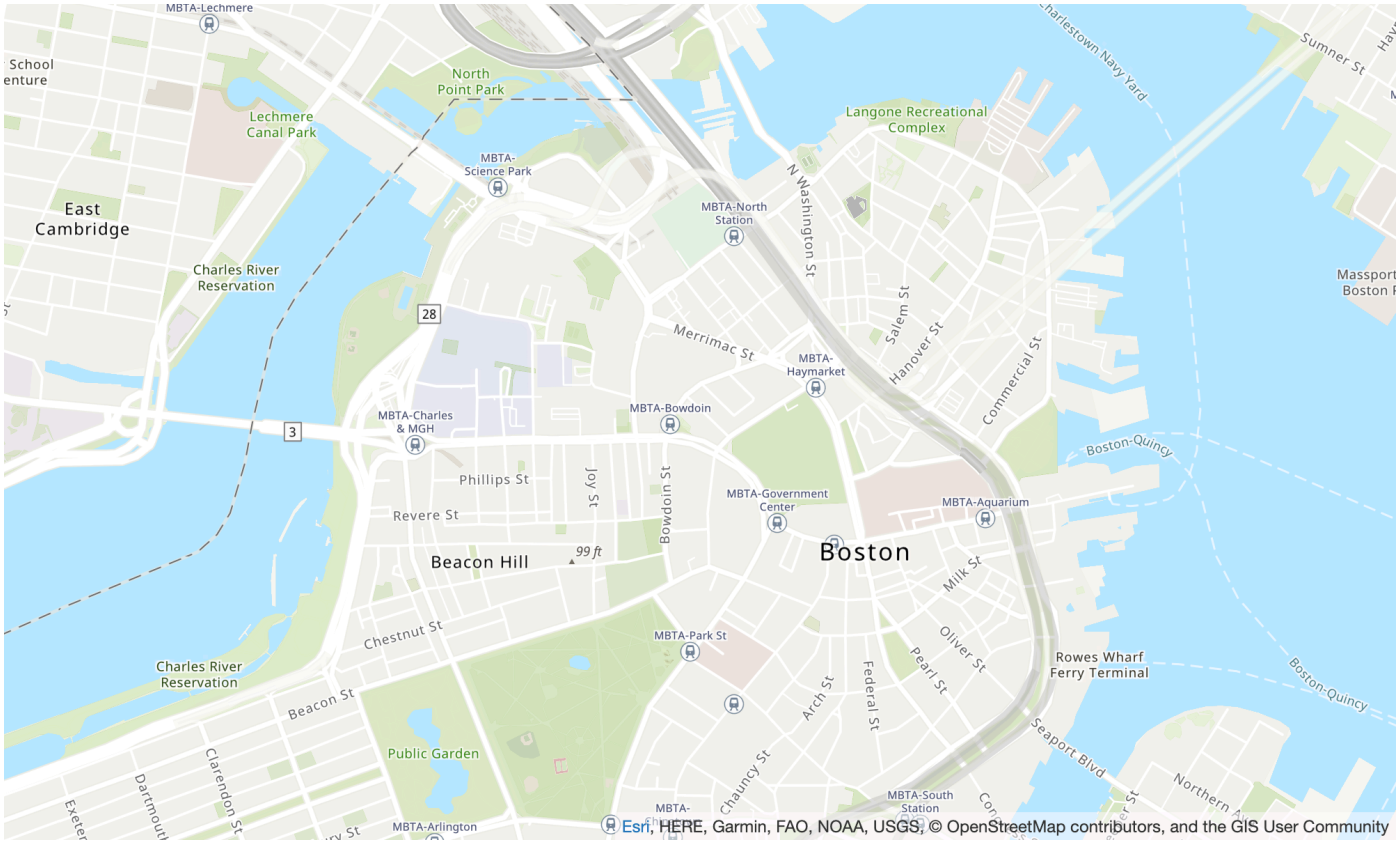
這張地圖在世界許多地方提供了一米或更好的衛星和航空圖像，以及全球較低分辨率的衛星圖像。

該地圖包括 15 米的圖像中小尺度（一 1：591 米下降到一 1：72 千）和 2.5 米的圖SPOT像（一 1：288 千到一 1：72 千克）為世界。該地圖具有美國大陸和 Maxar 西歐部分地區的 0.5 米分辨率圖像。這張地圖具有世界許多地方的其他 Maxar 次計圖像。在世界其他地區，GIS用戶社區以不同的分辨率貢獻了圖像。在某些社區中，可以使用低至 0.03 米的高分辨率圖像（低至 0.03 米）一 1:280 的比例。

有關更多信息，請參閱 Esri 網站上的 Esri 世界圖像。

Esri Light

埃斯里燈



地圖樣式名稱：`VectorEsriTopographic`

這為世界提供了一個詳細的底圖，象徵著經典的 Esri 地圖樣式。這包括高速公路，主要道路，次要道路，鐵路，水景，城市，公園，地標，建築物足跡和行政邊界。

該底圖是從多個數據提供商的各種權威來源編製而成，包括美國地質調查局（USGS），美國環境保護局（EPA），美國國家公園管理局（NPS），聯合國糧食和農業組織（FAO），加拿大自然資源部（NRCAN）和 Esri。HERE特定區域的資料來源於 OpenStreetMap 貢獻者。此外，數據由GIS社區提供。

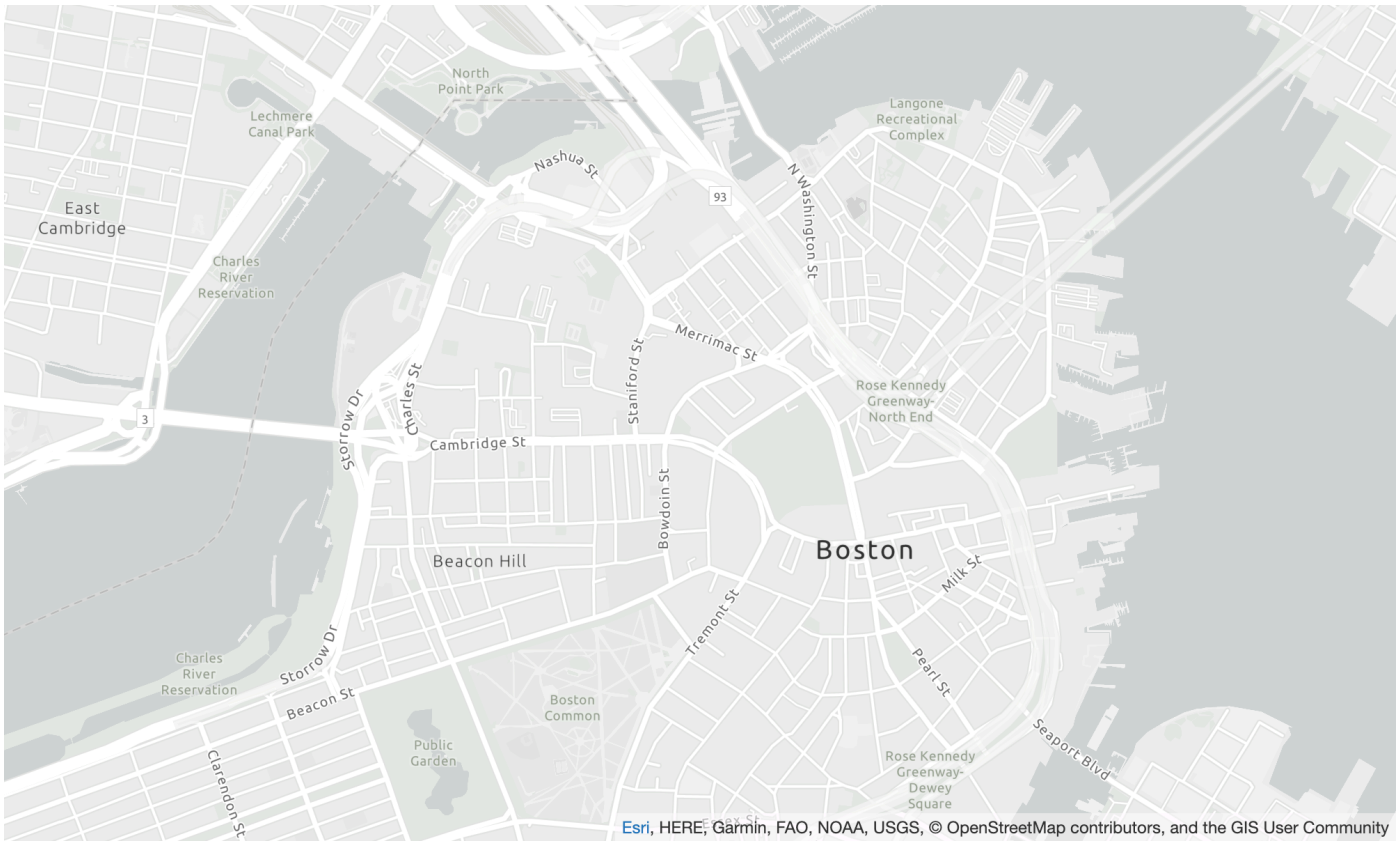
字型

Amazon 位置使用[GetMapGlyphs](#). 以下是此地圖的可用字體堆疊：

- 諾托無斜體
- 諾托三世常規
- 諾托無大膽
- 諾托襯線常規
- 機器人濃縮光斜體

Esri Light Gray Canvas

淺灰色帆布



地圖樣式名稱：`VectorEsriLightGrayCanvas`

這張地圖為世界提供了詳細的底圖，象徵著淺灰色，中性背景樣式，具有最少的顏色，標籤和功能，旨在吸引人們注意您的主題內容。

此向量拼貼圖層是使用淺灰色畫布和其他 Esri 底圖所使用的相同資料來源建立的。該地圖包括高速公路，主要道路，次要道路，鐵路，水景，城市，公園，地標，建築物足跡和行政邊界。

如需詳細資訊，請參閱 Esri 網站上的 Esri 淺灰色畫布。

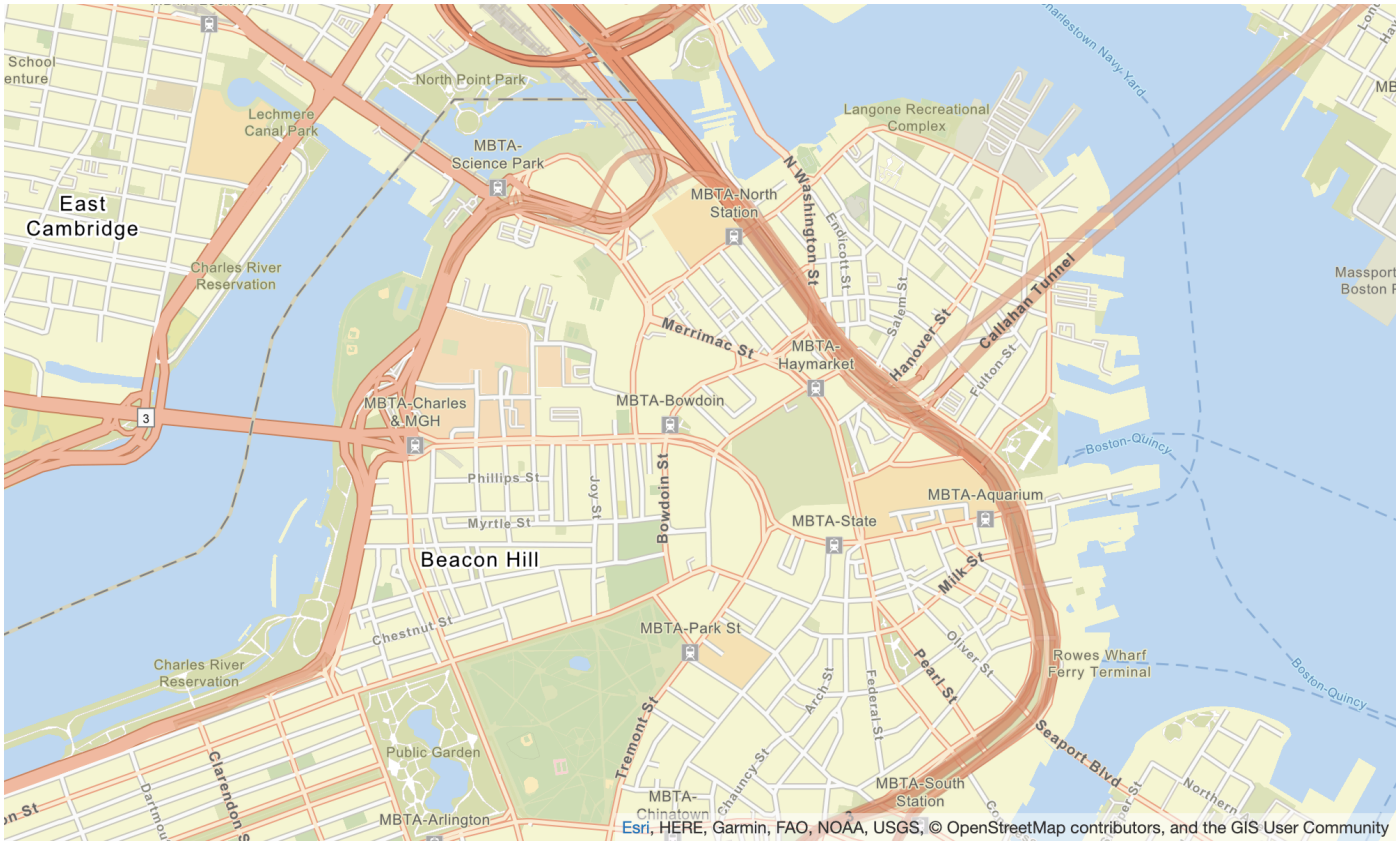
字型

Amazon 位置使用`GetMapGlyphs`. 以下是此地圖的可用字體堆疊：

- 斜體
- 普通
- Ubuntu 的光
- 大膽

Esri Street Map

伊斯里街地圖



地圖樣式名稱：`VectorEsriStreets`

這張地圖為世界提供了詳細的底圖，象徵著自定義導航地圖樣式，專為日間在移動設備中使用而設計。

這張綜合街道地圖包括高速公路、主要道路、次要道路、鐵路、水景、城市、公園、地標、建築足跡和行政邊界。它還包括一系列更豐富的地方，例如商店，服務，餐館，景點和其他景點。此地圖中的向量圖框圖層是使用與世界街道地圖和其他 Esri 基準地圖相同的資料來源建立的。

如需詳細資訊，請參閱 Esri 網站上的 Esri 世界街道。

字型

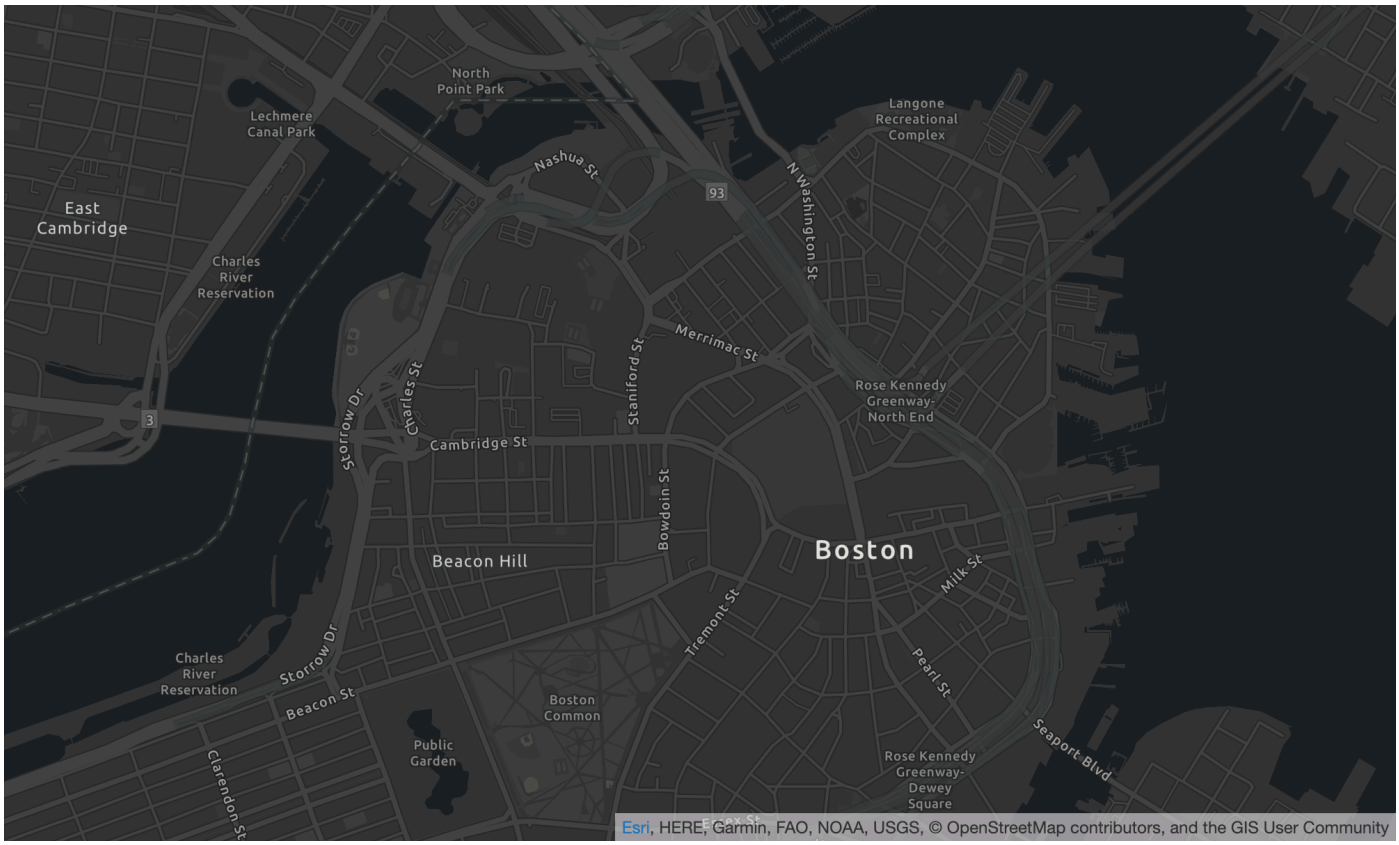
Amazon 位置使用`GetMapGlyphs`. 以下是此地圖的可用字體堆疊：

- 宋體斜體
- 宋體定期
- 宋體粗體

- 宋體萬國碼 MS 粗體
- 宋體國際碼 MS 常規

Esri Dark Gray Canvas

深灰色帆布



地圖樣式名稱：`VectorEsriDarkGrayCanvas`

這張地圖為世界提供了詳細的向量底圖，象徵著深灰色，中性背景樣式，具有最少的顏色，標籤和功能，旨在吸引人們注意您的主題內容。

此地圖包括高速公路、主要道路、次要道路、鐵路、水景、城市、公園、地標、建築物足跡和行政邊界。此地圖中的向量拼貼圖層是使用與深灰色 Canvas 點陣圖和其他 Esri 基準圖所使用的相同資料來源建立的。

如需詳細資訊，請參閱 Esri 網站上的 Esri 深灰色畫布。

字型

Amazon 位置使用`GetMapGlyphs`. 以下是此地圖的可用字體堆疊：

- 中等斜體
- 中型
- 斜體
- 普通
- 大膽

覆蓋範圍：埃斯里

您可以使用 Esri 做為資料提供者，以支援在建立位置索引資源時進行地理編碼、反向地理編碼和搜尋的查詢，或在建立路線計算器資源時支援查詢以計算路由。

Esri 在全球不同地區提供不同層級的資料品質。如需您感興趣地區涵蓋範圍的其他資訊，請參閱：

- Esri 地理編碼覆蓋範圍的詳細信息
- Esri 有關街道網絡和流量覆蓋的詳細信息

使用條款和數據歸因：Esri

在您使用 Esri 資料之前，請務必遵守所有適用的法律要求，包括適用於 Esri 和的授權條款。 AWS

如需有關 AWS 需求的詳細資訊，請參閱AWS服務條款。

有關 Esri 歸因準則的詳細資訊，請參閱 Esri 的資料歸因和使用條款。

向 Esri 報告錯誤

如果您遇到資料問題，並希望向 Esri 報告錯誤和差異，請參閱 Esri 關於 HOW TO 的技術支援文章：提供有關底圖和地理編碼的意見反應。

GrabMaps

Grab 是東南亞最大的送貨機構，擁有數百萬的司機合作夥伴和客戶。他們的子公司會在這些國家/地區建立地 up-to-date圖資料 GrabMaps，以供自己使用，以及其他國家/地區。Amazon 定 Location Service 使用 GrabMaps「定位服務」協助 AWS 客戶有效地使用地圖、地理編碼和計算路線。GrabMaps「定位服務旨在提供高質量，權威和 ready-to-use 位置數據，特別是針對東南亞國家/地區。

如需其他功能的相關資訊，請參閱 Amazon GrabMaps定 Location Service 資料提供者上的資訊

> **Important**
>
> Grab 僅提供東南亞的地圖，並且僅在亞太區域（新加坡）區域（ap-southeast-1）提供。如需詳細資訊，請參閱所涵蓋的國家 / 地區。

主題

- 抓取地圖樣式
- 覆蓋範圍：抓斗
- 所涵蓋的國家 / 地區
- 使用條款和數據歸因：Grab
- GrabMaps 資料錯誤報告

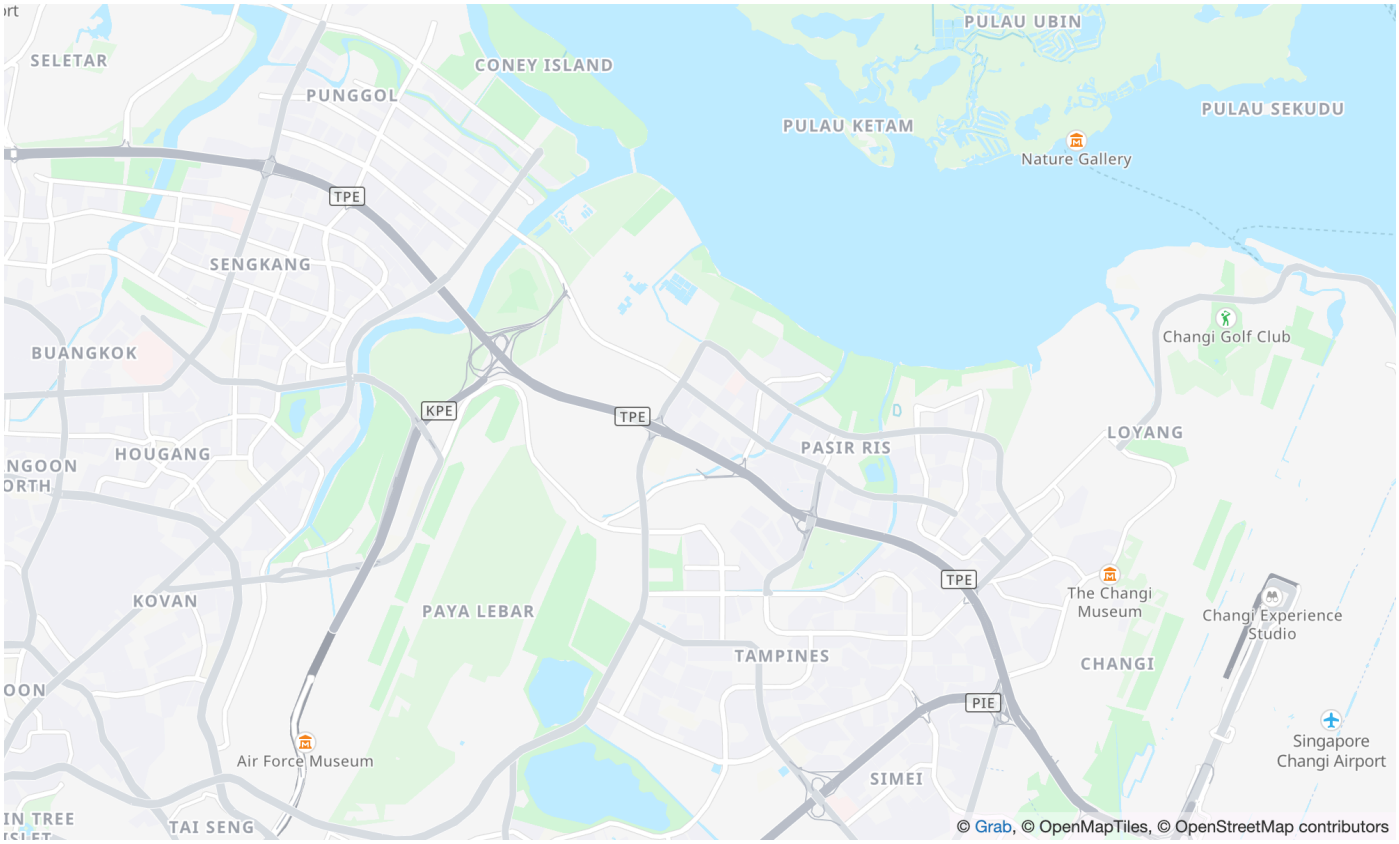
抓取地圖樣式

Amazon 定 Location Service 在創建地圖資源時支持以下 Grab 地圖樣式：

> **Note**
>
> 目前不支援此區段中未列出的抓取地圖樣式。

Grab Standard Light Map

抓取標準光地圖



地圖樣式名稱：`VectorGrabStandardLight`

Grab 的標準底圖，其中包含詳細的土地用途顏色，區域名稱，道路，地標和東南亞景點。

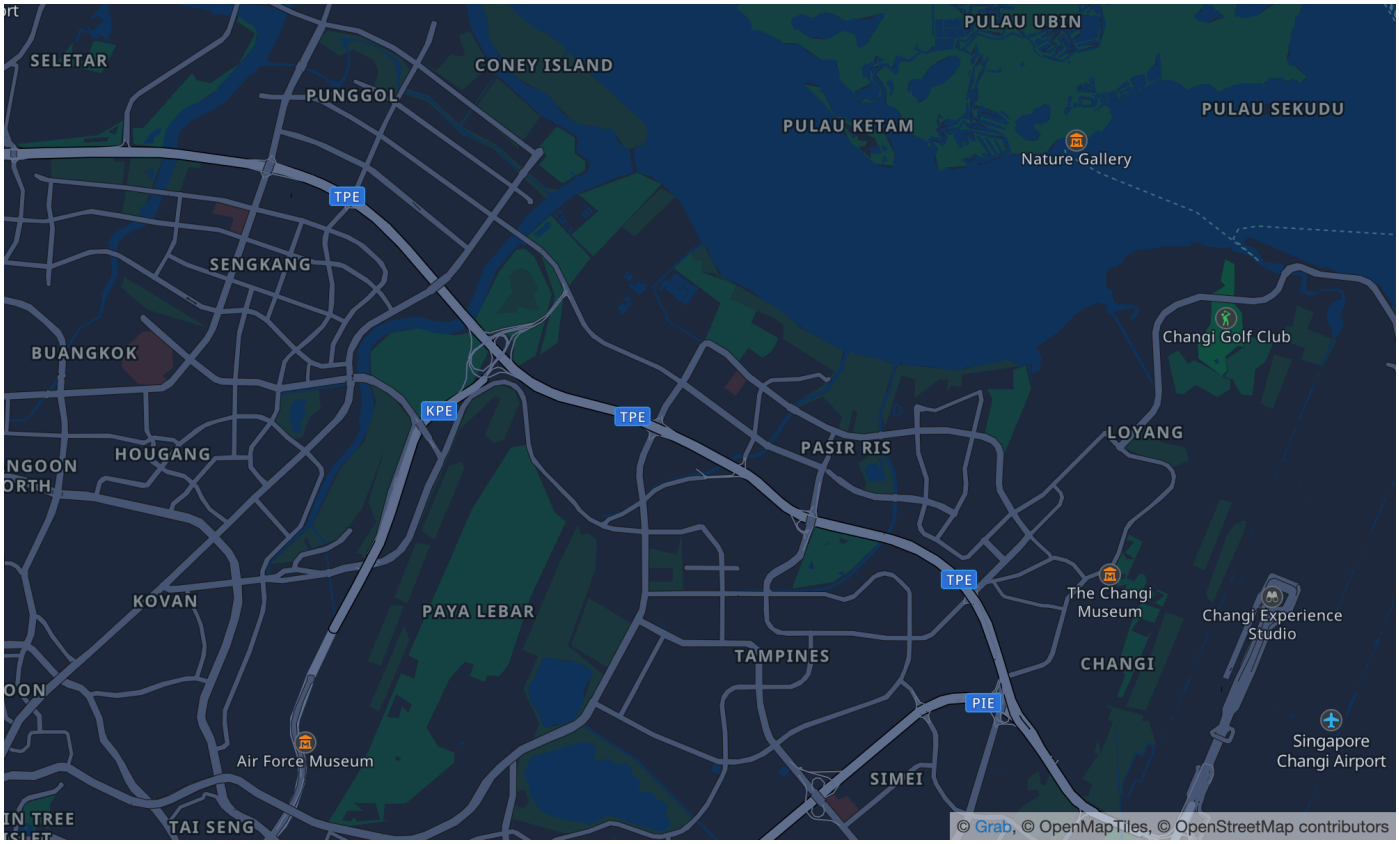
字型

Amazon 位置使用[GetMapGlyphs](#). 以下是此地圖的可用字體堆疊：

- 諾托三世常規
- 諾托三世中
- 諾托無大膽

Grab Standard Dark Map

抓取標準黑暗地圖



地圖樣式名稱：`VectorGrabStandardDark`

Grab 的標準底圖的黑暗變化，具有詳細的土地使用顏色，區域名稱，道路，地標和東南亞的景點。

字型

Amazon 位置使用`GetMapGlyphs`. 以下是此地圖的可用字體堆疊：

- 諾托三世常規
- 諾托三世中
- 諾托無大膽

覆蓋範圍：抓斗

您可以使用 Grab 做為資料提供者，以支援在建立位置索引資源時進行地理編碼、反向地理編碼和搜尋的查詢，或在建立路線計算器資源時支援查詢以計算路由。

所涵蓋的國家 / 地區

Grab 僅提供東南亞的地圖，並且僅在亞太區域（新加坡）區域（`ap-southeast-1`）提供。

Grab 提供下列國家/地區的詳細資料：

- 馬來西亞
- 菲律賓
- 泰國
- 新加坡
- 越南
- 印尼
- 緬甸
- 柬埔寨

> **Note**
>
> 在這些區域之外，以 Grab 身為資料提供者建立的 Amazon 定 Location Service 資源將不會提供任何結果。這包括搜尋結果或路線。

Grab 的地圖在以下邊界內：

- 南部 — 北緯線
- 西 — 經度
- 北緯 — 北緯
- 東 — 經度

對於縮放等級 1—4，Grab 包含全域覆蓋範圍。對於縮放等級 5 及以下，僅在此邊界方塊內提供地圖框。

> **Note**
>
> 在此邊界方塊之外，以 Grab 做為資料提供者建立的 Amazon 定 Location Service 地圖資源將不會傳回地圖磚。若要避免在應用程式中看到 404 錯誤，您可以使用邊界方框來限制地圖，如中所述使用以下方式設定地圖的範圍 MapLibre。

抓取路由旅行模式

對於路線，Grab 為所有先前列出的國家/地區提供汽車和摩托車路線。

Grab 不支援卡車路由。

對於自行車和步行路線，Grab 支持以下城市：。

- 新加坡
- 雅加達
- 馬尼拉
- 巴生谷
- 曼谷
- 胡志明市
- 河內

使用條款和數據歸因：Grab

使用 Grab 的數據時，您必須遵守所有適用的法律要求，包括適用於 Grab 和 AWS.

如需有關 AWS 需求的詳細資訊，請參閱AWS服務條款。

有關 GrabMaps「歸因指南」的信息，請參閱 Grab 的數據屬性和使用條款第 9.23 節。

GrabMaps 資料錯誤報告

如果您遇到來自資料的問題 GrabMaps，並且想要報告錯誤或差異，請聯絡 AWS 技術支援。

HERE技術

Amazon 定 Location Service 使用HERE技術的定位服務，協助 AWS 客戶有效地使用地圖、地理編碼和計算路線。HERE的位置資料提供以位置為中心的開放、安全且私密的平台。透過選取HERE位置資料，您可以選取原生部署在 AWS 雲端上的準確、新鮮且可靠的資料。

如需其他功能資訊，請參閱 Amazon 定 Location Service 資料提供者HERE上的資訊。

主題

- HERE地圖型式
- 覆蓋範圍:HERE
- 使用條款和數據歸因：HERE
- 錯誤報告給 HERE

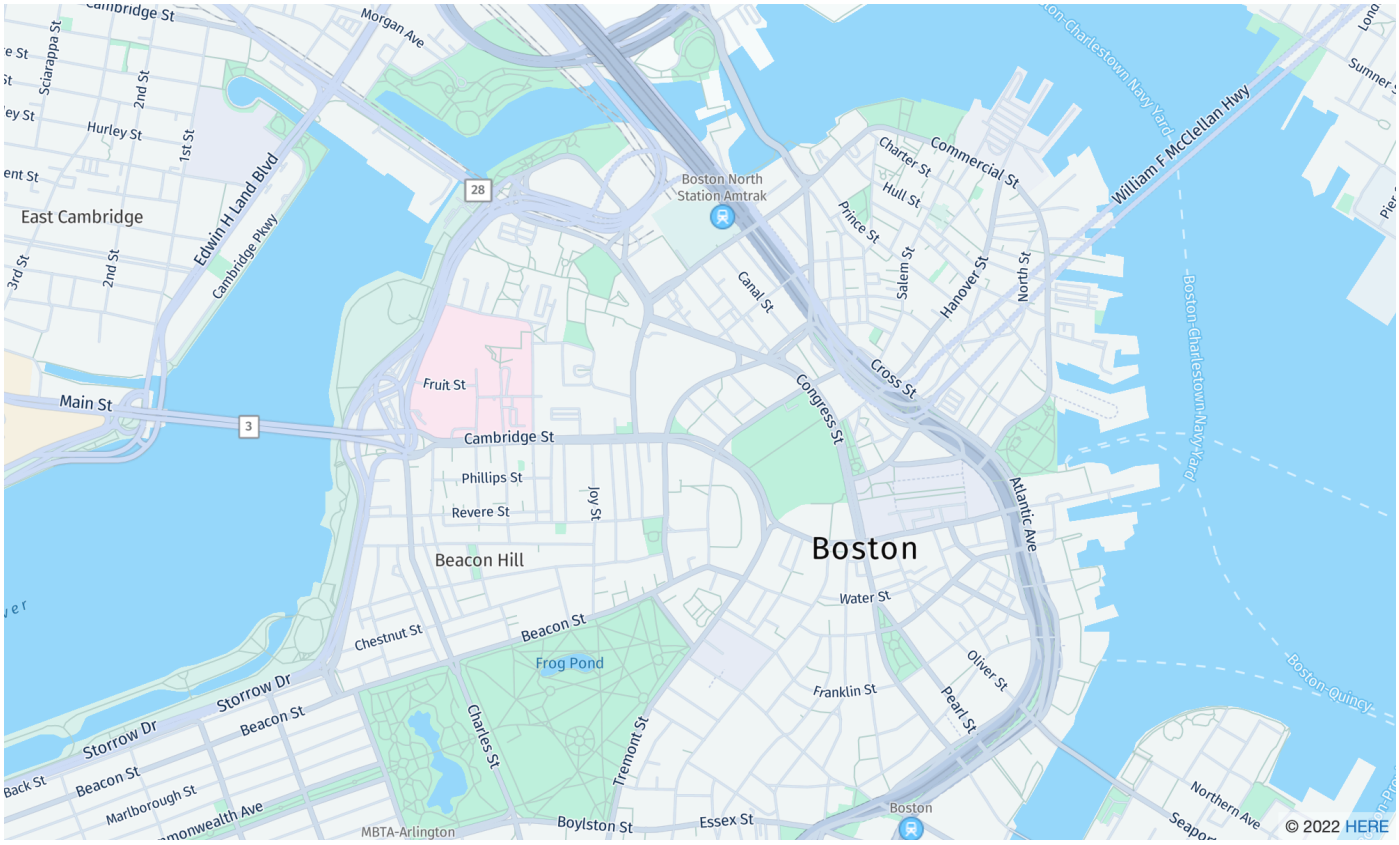
HERE地圖型式

Amazon 定 Location Service 在建立HERE地圖資源時支援以下地圖樣式：

> **Note**
>
> HERE目前不支援未列示在本節中的地圖型式。

HERE Explore

HERE探索



地圖樣式名稱：`VectorHereExplore`

HERE探索

詳細、中立的世界基地地圖。街道地圖包括高速公路，主要道路，次要道路，鐵路，水景，城市，公園，地標，建築物足跡和行政邊界。包括一張完全設計的日本地圖。

字型

Amazon 位置使用[GetMapGlyphs](). 以下是此地圖的可用字體堆疊：

- 費拉去斜體
- 菲拉去常規
- 費拉去大膽
- 能登三世太平紳士燈 CJK
- 能登三世 JP CJK 常規
- 諾托三世 JP CJK 大膽

HERE Imagery

HERE意象

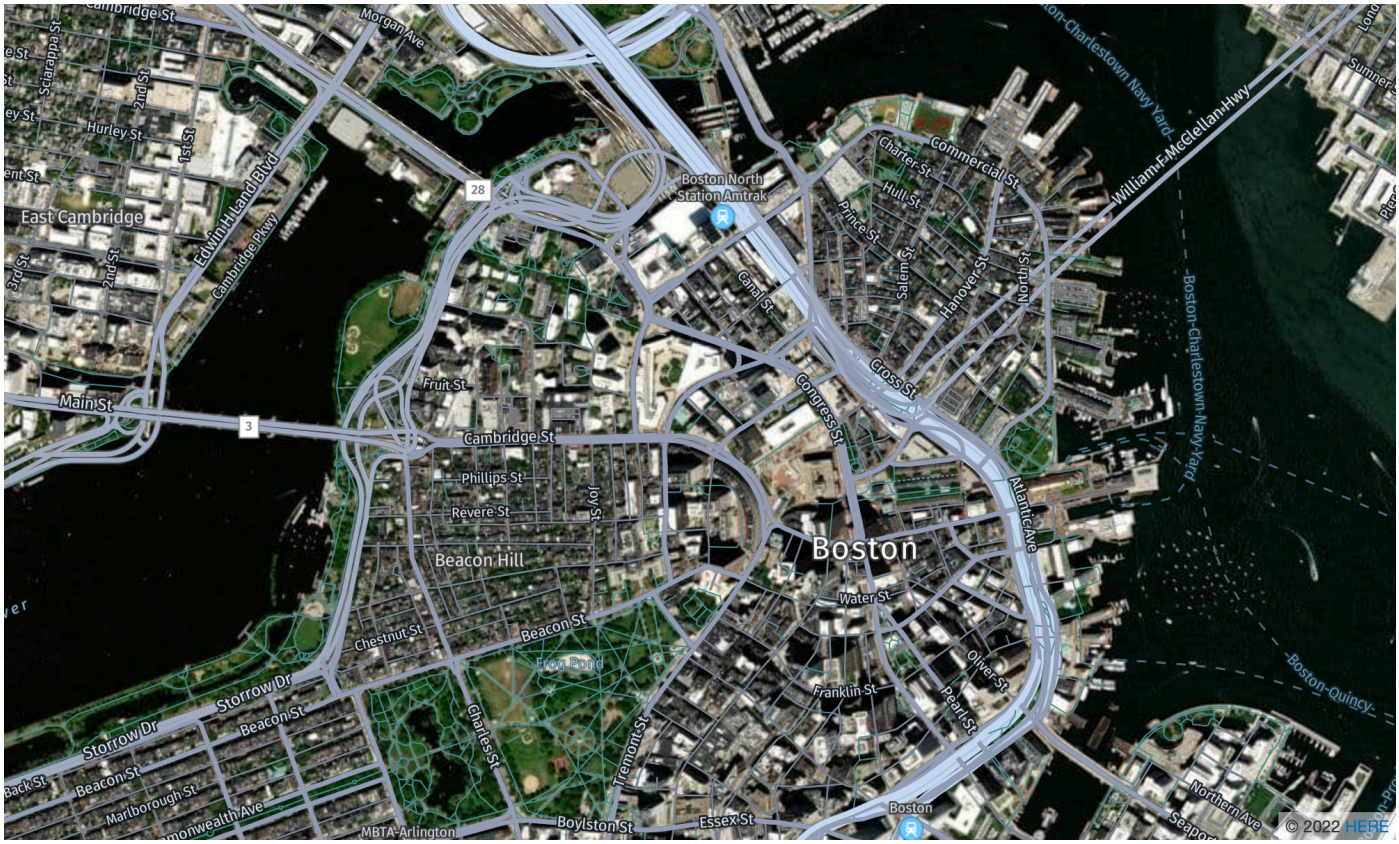
地圖樣式名稱：RasterHereExploreSatellite

HERE意象

HERE影像提供覆蓋全球的高解析度衛星圖像。

HERE Hybrid

HERE混合



地圖樣式名稱：HybridHereExploreSatellite

HERE混合

HEREHybrid style 透過衛星影像顯示道路網路、街道名稱和城市標籤。此樣式會覆蓋兩個地圖框：背景中的衛星影像 (點陣圖塊)，上方的道路網路和標籤 (向量圖塊)。此型式將自動擷取彩現地圖所需的點陣式和向量圖框。

> **Note**
>
> 轉譯所看到的地圖時，混合樣式會同時使用向量和點陣圖磚。這表示擷取的圖磚會比單獨使用向量或點陣圖磚時的更多。費用將包括所有擷取的圖磚。

字型

Amazon 位置使用[GetMapGlyphs](). 以下是此地圖的可用字體堆疊：

- 費拉去斜體
- 菲拉去常規
- 費拉去大膽
- 能登三世太平紳士燈 CJK
- 能登三世 JP CJK 常規
- 諾托三世 JP CJK 大膽

HERE Contrast (Berlin)

HERE對比度 (柏林)

地圖樣式名稱：`VectorHereContrast`

HERE對比度 (柏林)

融合 3D 和 2D 渲染的世界的詳細基礎地圖。高對比度的街道地圖包括高速公路，主要道路，次要道路，鐵路，水景，城市，公園，地標，建築物足跡和行政邊界。

字型

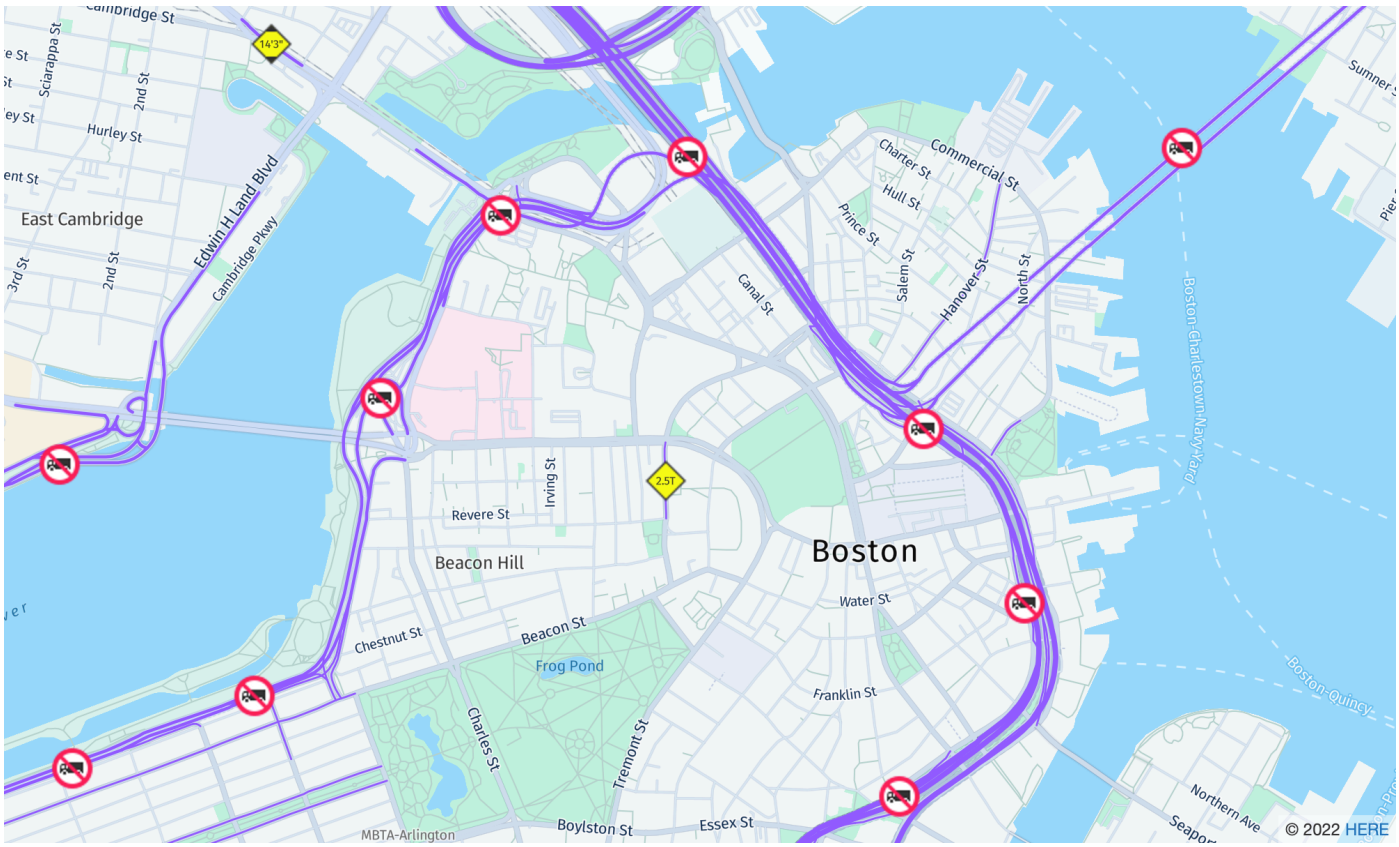
Amazon 位置使用[GetMapGlyphs](). 以下是此地圖的可用字體堆疊：

- 菲拉去常規
- 費拉去大膽

> **Note**
>
> 此樣式已從 `VectorHereBerlin` (HERE柏林地圖) 重新命名。 `VectorHereBerlin`已取代，但會繼續在使用它的應用程式中運作。

HERE Explore Truck

HERE探索卡車



地圖樣式名稱：`VectorHereExploreTruck`

HERE探索卡車

詳細、中立的世界基地地圖。街道地圖建立在 Ex HERE plore 風格之上，並以符號和圖示突出顯示軌道限制和屬性（包括寬度、高度和HAZMAT），以支援運輸和物流中的使用案例。

字型

Amazon 位置使用`GetMapGlyphs`. 以下是此地圖的可用字體堆疊：

- 費拉去斜體
- 菲拉去常規
- 費拉去大膽
- 能登三世太平紳士燈 CJK
- 能登三世 JP CJK 常規
- 諾托三世 JP CJK 大膽

有關全球不同地區的地圖資料品質的其他資訊，請參閱HERE地圖覆蓋範圍。

覆蓋範圍:HERE

當您建立位置索引資源時，您可以HERE做為資料提供者來支援地理編碼、反向地理編碼和搜尋的查詢，或者在建立路線計算器資源時支援查詢以計算路由。

HERE在世界上不同地區提供不同級別的數據質量。如需您感興趣的地區涵蓋範圍的其他資訊，請參閱下列內容：

- HERE地理編碼覆蓋
- HERE汽車路線覆蓋
- HERE卡車路由覆蓋

使用條款和數據歸因：HERE

在您使用HERE資料之前，請務必遵守所有適用的法律要求，包括適用於HERE和的授權條款 AWS。由於授權限制，您可能無法用HERE來儲存日本地區的地理編碼結果。

如 AWS 需需求的相關資訊，請參閱AWS服務條款。

如需有關HERE歸因指引的其他資訊，請參閱HERE技術適用於地點和其他內容的供應商條款第 2 節。

錯誤報告給 HERE

若要報告對映錯誤和差異HERE，請移至https://www.here.com/contact並選擇「報告對映錯誤」。

開啟資料

Amazon 定 Location Service 可透過開放資料供應商存取開放原始碼地圖資料。開放資料提供從 OpenStreetMap (OSM)、自然地球和其他開放資料來源的日光地圖分佈建立的全域基準地圖。提供的地圖旨在支持不同的應用和用例，包括物流和交付以及 Web 和移動環境中的數據可視化。OSM社群擁有超過一百萬個地圖製作者，每天更新數十萬個功能。Amazon 定 Location Service 會定期納入這些編輯內容。

如需其他功能資訊，請參閱 Amazon 定 Location Service 資料提供者上的開放資料。

主題

- 開啟資料對映型式
- 涵蓋範圍：開放數據
- 使用條款和數據歸因：開放數據
- 錯誤報告和貢獻開放資料

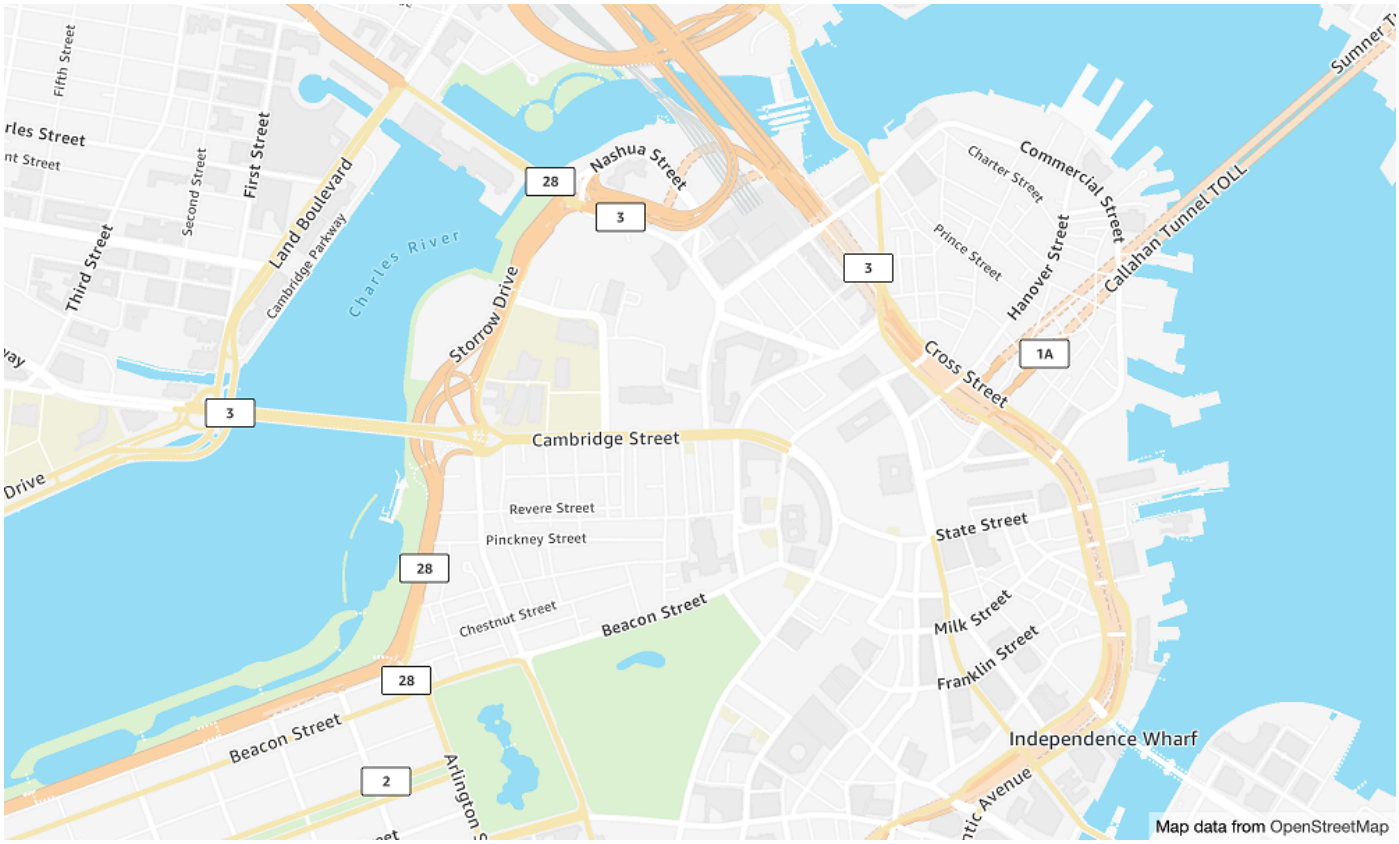
開啟資料對映型式

Amazon 定 Location Service 在建立地圖資源時支援以下地圖樣式：

開啟資料對映樣式支援替代政治觀點。

Open Data Standard Light

開放式資料標準燈



地圖樣式名稱：`VectorOpenDataStandardLight`

這為世界提供了一個詳細的底圖在光地圖風格，適用於網站和移動應用程序使用。這包括高速公路，主要道路，次要道路，鐵路，水景，城市，公園，地標，建築物足跡和行政邊界。

此底圖是基於從 OpenStreetMap (OSM) 貢獻者編譯的OSM日光地圖分佈。OSM社群包括超過 180 萬個貢獻者，他們每天更新超過 500,000 個功能。Amazon 定 Location Service 會定期納入這些編輯內容。

字型

Amazon 位置使用`GetMapGlyphs`. 以下是此地圖的可用字體堆疊：

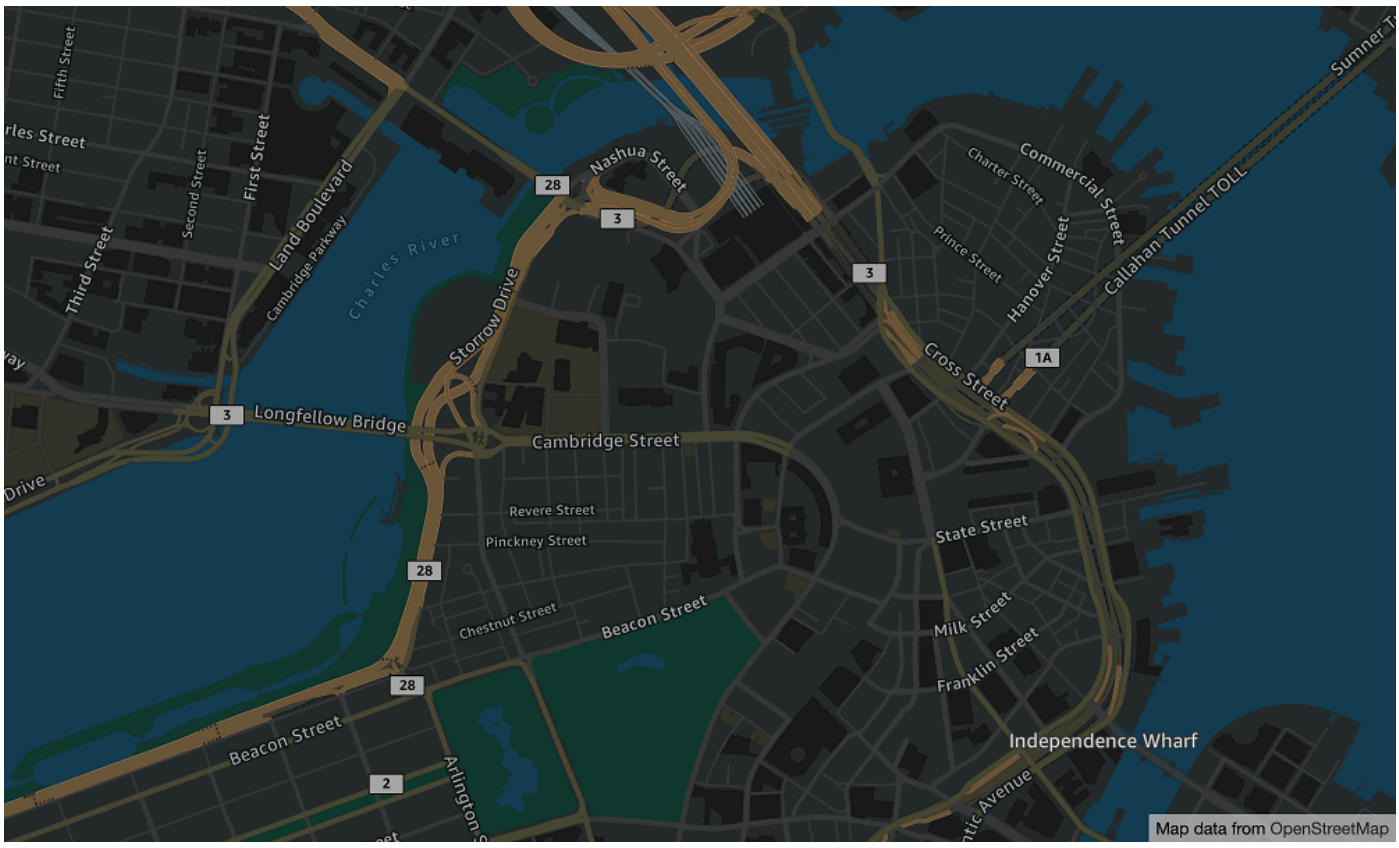
- Amazon 餘燼大膽，諾托沒有大膽
- Amazon 灰燼濃縮 RC 大膽，諾托無大膽
- Amazon 灰燼濃縮 RC 定期，諾托無常規
- Amazon 灰燼中，諾托無中
- Amazon 灰燼常規斜體，諾托無斜體
- Amazon 灰燼定期，諾托無常規

- Amazon 灰燼定期，諾托無常規，諾托無阿拉伯語常規
- Amazon 灰燼濃縮 RC 大膽，諾托無大膽，諾托無阿拉伯語濃縮大膽
- Amazon 灰燼大膽，諾托無大膽，諾托無阿拉伯語粗體
- Amazon 灰燼常規斜體，諾托無斜體，諾托三世阿拉伯語常規
- Amazon 灰燼濃縮 RC 定期，諾托無常規，諾托三世阿拉伯語濃縮定期
- Amazon 灰燼中，諾托無中，諾托無阿拉伯語媒體

> **Note**
>
> 所使用的字體`VectorOpenDataStandardLight`是組合字體，用`Amazon Ember`於大多數字符，但用`Noto Sans`於不支援的字符。`Amazon Ember`

Open Data Standard Dark

開放數據標準黑暗



地圖樣式名稱：`VectorOpenDataStandardDark`

這是一種深色主題的地圖風格，為世界提供了詳細的底圖，適合網站和移動應用程序使用。這包括高速公路，主要道路，次要道路，鐵路，水景，城市，公園，地標，建築物足跡和行政邊界。

此底圖是基於從 OpenStreetMap (OSM) 貢獻者編譯的OSM日光地圖分佈。OSM社群包括超過 180 萬個貢獻者，他們每天更新超過 500,000 個功能。Amazon 定 Location Service 會定期納入這些編輯內容。

字型

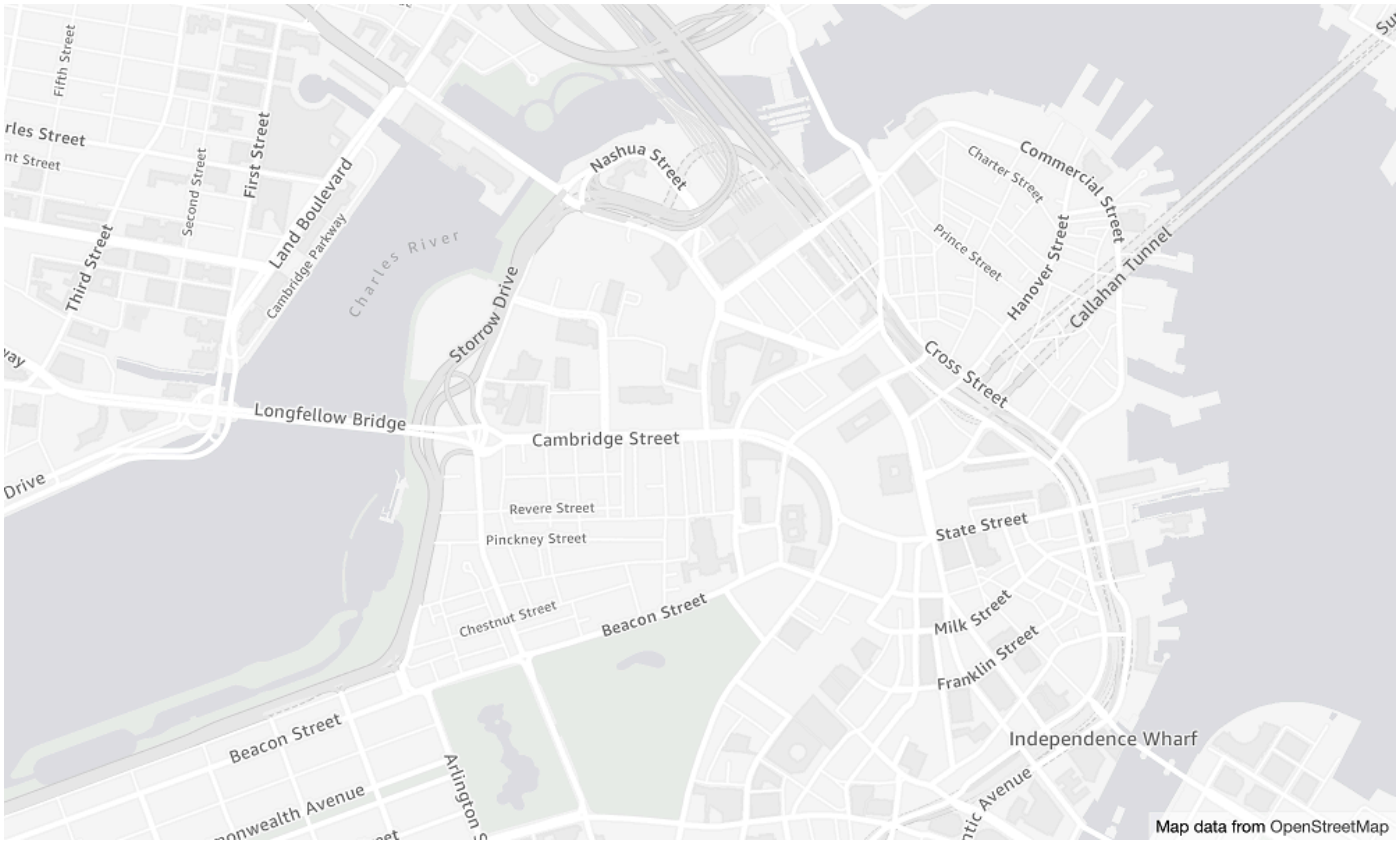
Amazon 位置使用GetMapGlyphs. 以下是此地圖的可用字體堆疊：

- Amazon 餘燼大膽，諾托沒有大膽
- Amazon 灰燼濃縮 RC 大膽，諾托無大膽
- Amazon 灰燼濃縮 RC 定期，諾托無常規
- Amazon 灰燼中，諾托無中
- Amazon 灰燼常規斜體，諾托無斜體
- Amazon 灰燼定期，諾托無常規
- Amazon 灰燼定期，諾托無常規，諾托無阿拉伯語常規
- Amazon 灰燼濃縮 RC 大膽，諾托無大膽，諾托無阿拉伯語濃縮大膽
- Amazon 灰燼大膽，諾托無大膽，諾托無阿拉伯語粗體
- Amazon 灰燼常規斜體，諾托無斜體，諾托三世阿拉伯語常規
- Amazon 灰燼濃縮 RC 定期，諾托無常規，諾托三世阿拉伯語濃縮定期
- Amazon 灰燼中，諾托無中，諾托無阿拉伯語媒體

> **Note**
>
> 所使用的字體VectorOpenDataStandardDark是組合字體，用Amazon Ember於大多數字符，但用Noto Sans於不支援的字符。Amazon Ember

Open Data Visualization Light

開放式資料視覺化燈



地圖樣式名稱：`VectorOpenDataVisualizationLight`

這是一種以淺色為主題的風格，具有柔和的色彩和較少的功能，有助於理解覆蓋的數據。

此底圖是基於從 OpenStreetMap (OSM) 貢獻者編譯的OSM日光地圖分佈。OSM社群包括超過 180 萬個貢獻者，他們每天更新超過 500,000 個功能。Amazon 定 Location Service 會定期納入這些編輯內容。

字型

Amazon 位置使用`GetMapGlyphs`. 以下是此地圖的可用字體堆疊：

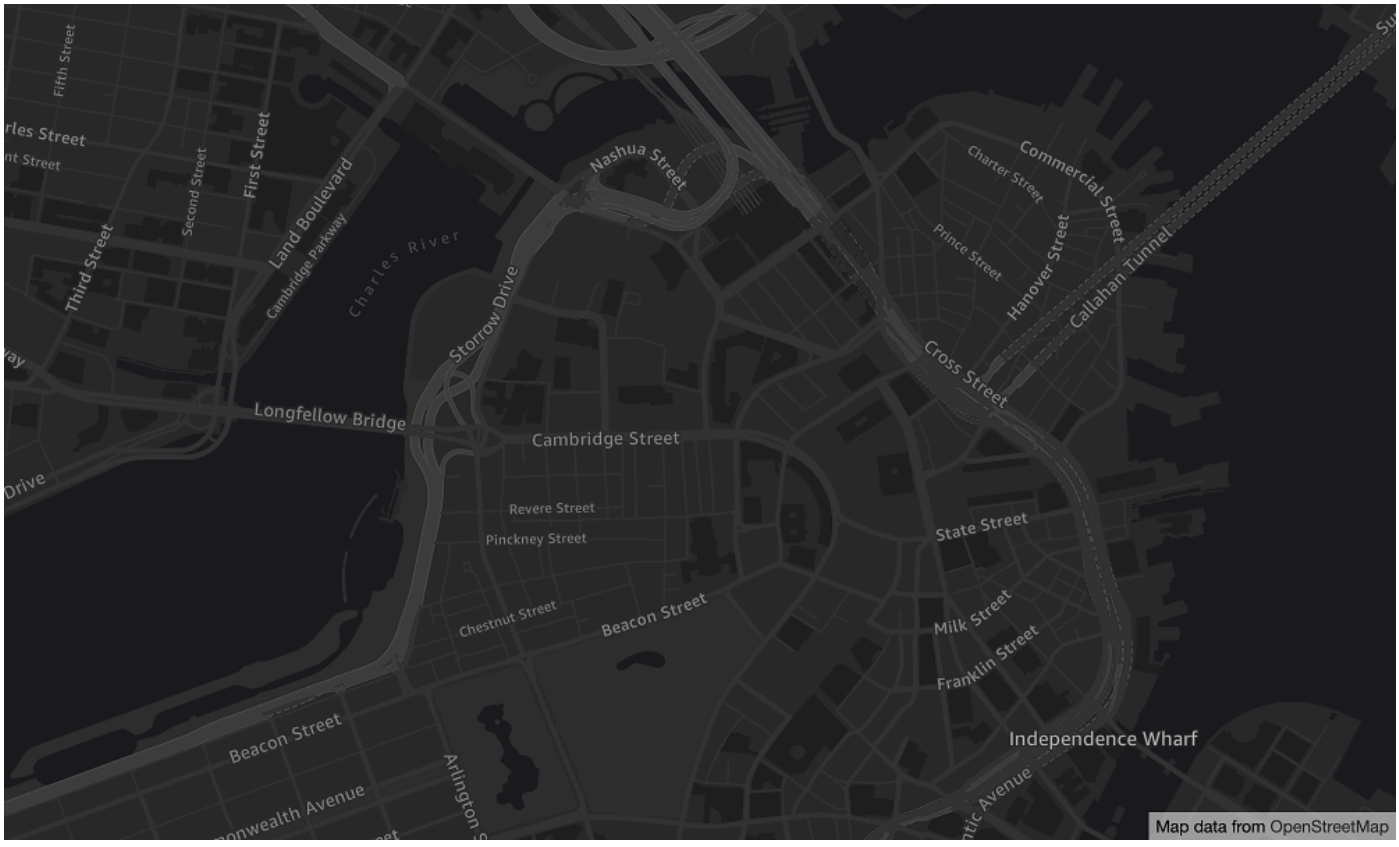
- Amazon 餘燼大膽，諾托沒有大膽
- Amazon 灰燼濃縮 RC 大膽，諾托無大膽
- Amazon 灰燼濃縮 RC 定期，諾托無常規
- Amazon 灰燼中，諾托無中
- Amazon 灰燼常規斜體，諾托無斜體
- Amazon 灰燼定期，諾托無常規
- Amazon 灰燼定期，諾托無常規，諾托無阿拉伯語常規

- Amazon 灰燼濃縮 RC 大膽，諾托無大膽，諾托無阿拉伯語濃縮大膽
- Amazon 灰燼大膽，諾托無大膽，諾托無阿拉伯語粗體
- Amazon 灰燼常規斜體，諾托無斜體，諾托三世阿拉伯語常規
- Amazon 灰燼濃縮 RC 定期，諾托無常規，諾托三世阿拉伯語濃縮定期
- Amazon 灰燼中，諾托無中，諾托無阿拉伯語媒體

> **Note**
>
> 所使用的字體`VectorOpenDataVisualizationLight`是組合字體，用`Amazon Ember`於大多數字符，但用`Noto Sans`於不支援的字符。`Amazon Ember`

Open Data Visualization Dark

開放數據可視化黑暗



地圖樣式名稱：`VectorOpenDataVisualizationDark`

這是一種深色主題的樣式，具有柔和的色彩和較少的功能，有助於理解覆蓋的數據。

此底圖是基於從 OpenStreetMap (OSM) 貢獻者編譯的OSM日光地圖分佈。OSM社群包括超過 180 萬個貢獻者，他們每天更新超過 500,000 個功能。Amazon 定 Location Service 會定期納入這些編輯內容。

字型

Amazon 位置使用GetMapGlyphs. 以下是此地圖的可用字體堆疊：

- Amazon 餘燼大膽，諾托沒有大膽
- Amazon 灰燼濃縮 RC 大膽，諾托無大膽
- Amazon 灰燼濃縮 RC 定期，諾托無常規
- Amazon 灰燼中，諾托無中
- Amazon 灰燼常規斜體，諾托無斜體
- Amazon 灰燼定期，諾托無常規
- Amazon 灰燼定期，諾托無常規，諾托無阿拉伯語常規
- Amazon 灰燼濃縮 RC 大膽，諾托無大膽，諾托無阿拉伯語濃縮大膽
- Amazon 灰燼大膽，諾托無大膽，諾托無阿拉伯語粗體
- Amazon 灰燼常規斜體，諾托無斜體，諾托三世阿拉伯語常規
- Amazon 灰燼濃縮 RC 定期，諾托無常規，諾托三世阿拉伯語濃縮定期
- Amazon 灰燼中，諾托無中，諾托無阿拉伯語媒體

> **Note**
>
> 所使用的字體VectorOpenDataVisualizationDark是組合字體，用Amazon Ember於大多數字符，但用Noto Sans於不支援的字符。Amazon Ember

涵蓋範圍：開放數據

開放資料包括具有全球覆蓋範圍的地圖，可使用 Amazon 定 Location Service 地圖資源進行轉譯。

> **Note**
>
> 開放資料僅適用於 Amazon 定 Location Service 地圖資源。您無法使用 Open Data 做為資料提供者來支援地理編碼、反向地理編碼和搜尋的查詢，或支援查詢以計算路線。

使用條款和數據歸因：開放數據

在使用開放資料之前，請務必遵守所有適用的法律要求，包括適用於開放資料和 AWS.

如需有關 AWS 需求的詳細資訊，請參閱AWS服務條款。

如需有關「開放資料」歸因指南 OpenStreetMap的資訊，請參閱「版權與授權條款」和 OpenStreetMap「授權/署名指南」。

錯誤報告和貢獻開放資料

OpenStreetMap （OSM）和自然地球是社區驅動的開放數據項目。如果您遇到資料問題，您可以回報錯誤或直接提供修正或建議。

- 若要在中報告錯誤或提供建議OSM，您可以在地圖上建立註記。這是對地圖的評論，可協助貢獻者對地圖進行修復。您可以通過OpenStreetMap 網站創建筆記。如需有關附註的詳細資訊，請參閱 OpenStreetMap Wiki 中的註記。
- 如需有關直接貢獻的詳細資訊 OpenStreetMap，包括新增位置和修正錯誤，請參閱 OpenStreetMap Wiki 中的貢獻地圖資料。
- 要提交自然地球中數據的更正請求，您可以通過自然地網站提交問題。

> **Note**
>
> 中的修正錯誤 OpenStreetMap 可能會很快發生，但是，修正顯示在「開放資料」提供者所使用之OSM資料的日光地圖分佈中可能需要一些時間。「日光地圖分佈」網站提供有關此程序的更多資訊。此外，Amazon 定 Location Service 大約每月更新一次 Amazon 定 Location Service 中使用的地圖資料。

資料提供者的功能

本節說明 Amazon 定 Location Service 中可用的功能，並依資料提供者分類。

下表提供這些功能的高階概觀。

資料提供者	地理覆蓋	功能覆蓋	AWS 區域
埃斯里	全球服務	地圖，地方，路線	提供 Amazon 位置服務的所有區域。

資料提供者	地理覆蓋	功能覆蓋	AWS 區域
抓斗	東南亞	地圖，地方，路線	亞太區域 (新加坡) `ap-southeast-1` 、僅限。
HERE	全球服務	地圖，地方，路線	提供 Amazon 位置服務的所有區域。
開啟資料	全球服務	地圖	提供 Amazon 位置服務的所有區域。

下列索引標籤顯示每個功能區域內的詳細資訊。

Map Features

下表展示了依資料提供者的地圖圖徵。如需有關地圖概念的更多資訊，請參閱地圖。

資料提供者	支援的貼圖類型	向量縮放層級	點陣式縮放層級
埃斯里	向量 光柵 (影像) 如需詳細資訊，請參閱Esri 地圖樣式。	0-15	0-23
抓斗	向量 (僅限東南亞) 如需詳細資訊，請參閱抓取地圖樣式。	0-14	無
HERE	向量 光柵 (影像) 混合	1-17	0-19

資料提供者	支援的貼圖類型	向量縮放層級	點陣式縮放層級
	如需詳細資訊，請參閱HERE地圖型式。		
開啟資料	向量 如需詳細資訊，請參閱開啟資料對映型式。	0-15	無

> **Note**
>
> 縮放層級代表每個提供者的定義的最大和最小設定APIs。地圖的不同區域可能具有不同的最大值；例如，海洋磚的詳細縮放等級可能比主要城市的區域少。
> MapLibre （和其他地圖渲染引擎）允許您設置最小和最大縮放級別，並且還將遵循一個區域中的數據提供程序縮放級別，因此您不必編寫代碼來處理這些差異。

Places and Search

下表顯示資料提供者的位置和搜尋功能。如需地點概念的詳細資訊，請參閱地點搜尋。

資料提供者	地理编码	反向地理編碼	自動完成	GetPlace
埃斯里	所有功能，除了： PlaceId	所有功能，除了： TimeZone PlaceId	所有功能	所有功能
抓斗	所有功能，除了： 單位類型 不支援類別	所有功能	所有功能	所有功能，除了： 單位類型 SubMunicipality

資料提供者	地理编码	反向地理編碼	自動完成	GetPlace
HERE	所有功能，除了： 單位編號 單位類型 相關性 篩選的其他限制	所有功能	所有功能	所有功能，除了： 單位編號 單位類型 SubMunicipality
開啟資料	不支援	不支援	不支援	僅支持： SubMunicipality

Route features

下表顯示了由數據提供者的路由功能。如需有關路由概念的更多資訊，請參閱路由。如需有關路由矩陣限制的更多詳細說明，請參閱出發和目的地位置的限制。

資料提供者	旅行模式	計算路線	路由矩陣
埃斯里	汽車, 卡車, 步行	出發地和目的地必須彼此相距 400 公里以內。總旅行時間不能超過 400 分鐘。 ArrivalTime 不支援。	最多 10 個出發和目的地位置。 韓國不支援。 所有出發和目的地配對必須彼此相距 400 公里以內。
抓斗	汽車摩托車 步行和騎自行車在選定的城市。	沒有距離限制。	最多 350 個出發和目的地位置。

資料提供者	旅行模式	計算路線	路由矩陣
HERE	汽車, 卡車, 步行	沒有距離限制。出發地和目的地位置周圍超過 10 公里的路線將不計算。	最多 350 個出發和目的地位置。 所有出發和目的地位置必須以 180 公里的圓圈掉落。 支援較長的路由，但有額外的限制。
開啟資料	不支援	不支援	不支援

資料提供者的使用條款和資料歸因

在您使用資料提供者之前，請務必遵守所有適用的法律要求，包括適用於提供者使用的授權條款。

如需有關 AWS 需求的詳細資訊，請參閱AWS服務條款。

為您的應用程式或文件使用資料提供者搭配 Amazon Location 資源時，請務必為您使用的每個資料提供者提供屬性。

如需每個資料提供者合規性和歸因的詳細資訊，請參閱下列主題。

- 埃斯里 — 使用條款和數據歸因：Esri
- 抓斗 — 使用條款和數據歸因：Grab
- HERE – 使用條款和數據歸因：HERE
- 開放資料 — 使用條款和數據歸因：開放數據

Amazon 位置區域和端點

Amazon 位置在以下 AWS 區域提供：

區域

區域名稱	區域	端點	通訊協定	
美國東部 (俄亥俄)	us-east-2	geo.us-east-2.amazonaws.com	HTTPS	
美國東部 (維吉尼亞北部)	us-east-1	geo.us-east-1.amazonaws.com	HTTPS	
美國西部 (奧勒岡)	us-west-2	geo.us-west-2.amazonaws.com	HTTPS	
亞太區域 (孟買)	ap-south-1	geo.ap-south-1.amazonaws.com	HTTPS	
亞太區域 (新加坡)	ap-southeast-1	geo.ap-southeast-1.amazonaws.com	HTTPS	
亞太區域 (雪梨)	ap-southeast-2	geo.ap-southeast-2.amazonaws.com	HTTPS	
亞太區域 (東京)	ap-northeast-1	geo.ap-northeast-1.amazonaws.com	HTTPS	
加拿大 (中部)	ca-central-1	geo.ca-central-1.amazonaws.com	HTTPS	
歐洲 (法蘭克福)	eu-central-1	geo.eu-central-1.amazonaws.com	HTTPS	
歐洲 (愛爾蘭)	eu-west-1	geo.eu-west-1.amazonaws.com	HTTPS	
歐洲 (倫敦)	eu-west-2	geo.eu-west-2.amazonaws.com	HTTPS	

區域名稱	區域	端點	通訊協定
歐洲 (斯德哥爾摩)	eu-north-1	geo.eu-north-1.amazonaws.com	HTTPS
南美洲 (聖保羅)	sa-east-1	geo.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (美國西部)	us-gov-west-1	geo.us-gov-west-1.amazonaws.com geo-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS

> **Note**
>
> 如需如何使用此表格中端點的詳細資訊，請參閱下一節。

端點

Amazon 位置區域端點的一般語法如下：

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

在此語法中，Amazon 位置使用下列服務代碼：

服務	服務代碼
Amazon 位置地圖	地圖
Amazon 位置地方	地方
Amazon 地理圍欄位置	地理圍欄
Amazon 位置跟踪器	追蹤
Amazon 位置路線	路線

例如，適用於美國東部 (維吉尼亞北部) 的 Amazon 位置對應的區域端點為：https://*maps*. 地理。*us-east-1*. 亞馬遜.

API作業端點

Amazon 定 Location Service 控制平面端點的語法如下：

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

Amazon 定 Location Service 的控制平面動作如下：

服務	端點	API操作
Amazon 位置地圖	https://cp.maps.geo.*region*.amazonaws.com	CreateMap DeleteMap DescribeMap ListMaps UpdateMap
Amazon 位置地方	https://cp.places.geo.*region*.amazonaws.com	CreatePlaceIndex DeletePlaceIndex DescribePlaceIndex ListPlaceIndexes UpdatePlaceIndex
Amazon 地理圍欄位置	https://cp.geofencing.geo.*region*.amazonaws.com	CreateGeofenceCollection DeleteGeofenceCollection DescribeGeofenceCollection ListGeofenceCollections UpdateGeofenceCollection

服務	端點	API操作
Amazon 位置跟踪器	https://cp.tracking.geo.*region*.amazonaws.com	CreateTracker DeleteTracker DescribeTracker UpdateTracker ListTrackers AssociateTrackerConsumer DisassociateTrackerConsumer ListTrackerConsumers
Amazon 位置路線	https://cp.routes.geo.*region*.amazonaws.com	CreateRouteCalculator DeleteRouteCalculator DescribeRouteCalculator ListRouteCalculators UpdateRouteCalculator
Amazon 位置元數	https://cp.metadata.geo.*region*.amazonaws.com	CreateKey DeleteKey DescribeKey ListKeys UpdateKey

Amazon 定 Location Service 配額

本主題提供 Amazon 定 Location Service 的費率限制和配額摘要。

> **Note**
>
> 如果您需要更高的配額，可以使用 Service Quotas 主控台來要求增加可調配額的配額。請求增加配額時，請選取您需要增加配額的地區，因為大多數配額都是特定於該 AWS 地區。

Service Quotas 是您可以擁有每個 AWS 帳戶和 AWS 區域的資源數量上限。Amazon 定 Location Service 會拒絕超出服務配額的其他請求。

速率限制 (以速率開頭的配額...) 是針對每個API作業定義的每秒最大要求數目，突發速率為第二個任何部分的限制的 80%。Amazon 定 Location Service 會限制超出作業速率限制的請求。

名稱	預設	可調整	描述
API每個帳戶的關鍵資源	每個受支援的區域：500	否	每個帳號可以擁有的API關鍵資源數目上限 (作用中或已過期)。
每個帳號的地理圍欄收集資源	每個支持的地區：1,500	是	您可以為每個帳號建立的地理圍欄收集資源數目上限。
地理圍欄收藏的地理圍欄	每個支持的地區：50,000	否	每個地理圍欄集合可以建立的地理圍欄數目上限。
每個帳號對應資源	每個受支援的區域：40	是	您可以為每個帳號建立的地圖資源數目上限。
每個帳號放置索引資源	每個受支援的區域：40	是	您可以為每個帳號建立的位置索引資源數目上限。
AssociateTrackerConsumer API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 AssociateTrackerConsumer 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
BatchDeleteDevicePositionHistory API 要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 BatchDeleteDevicePositionHistory 要求數目上限。其他請求會受到限流。
BatchDeleteGeofence API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 BatchDeleteGeofence 要求數目上限。其他請求會受到限流。
BatchEvaluateGeofences API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 BatchEvaluateGeofences 要求數目上限。其他請求會受到限流。
BatchGetDevicePosition API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 BatchGetDevicePosition 要求數目上限。其他請求會受到限流。
BatchPutGeofence API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 BatchPutGeofence 要求數目上限。其他請求會受到限流。
BatchUpdateDevicePosition API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 BatchUpdateDevicePosition 要求數目上限。其他請求會受到限流。
CalculateRoute API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 CalculateRoute 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
CalculateRouteMatrix API要求率	每個支援的區域：每秒 5 個	是	您每秒可發出的 CalculateRouteMatrix 要求數目上限。其他請求會受到限流。
CreateGeofenceCollection API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 CreateGeofenceCollection 要求數目上限。其他請求會受到限流。
CreateKey API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 CreateKey 要求數目上限。其他請求會受到限流。
CreateMap API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 CreateMap 要求數目上限。其他請求會受到限流。
CreatePlaceIndex API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 CreatePlaceIndex 要求數目上限。其他請求會受到限流。
CreateRouteCalculator API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 CreateRouteCalculator 要求數目上限。其他請求會受到限流。
CreateTracker API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 CreateTracker 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
DeleteGeofenceCollection API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DeleteGeofenceCollection 要求數目上限。其他請求會受到限流。
DeleteKey API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DeleteKey 要求數目上限。其他請求會受到限流。
DeleteMap API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DeleteMap 要求數目上限。其他請求會受到限流。
DeletePlaceIndex API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DeletePlaceIndex 要求數目上限。其他請求會受到限流。
DeleteRouteCalculator API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DeleteRouteCalculator 要求數目上限。其他請求會受到限流。
DeleteTracker API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DeleteTracker 要求數目上限。其他請求會受到限流。
DescribeGeofenceCollection API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DescribeGeofenceCollection 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
DescribeKey API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DescribeKey 要求數目上限。其他請求會受到限流。
DescribeMap API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DescribeMap 要求數目上限。其他請求會受到限流。
DescribePlaceIndex API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DescribePlaceIndex 要求數目上限。其他請求會受到限流。
DescribeRouteCalculator API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DescribeRouteCalculator 要求數目上限。其他請求會受到限流。
DescribeTracker API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DescribeTracker 要求數目上限。其他請求會受到限流。
DisassociateTrackerConsumer API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 DisassociateTrackerConsumer 要求數目上限。其他請求會受到限流。
ForecastGeofenceEvents API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 ForecastGeofenceEvents 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
GetDevicePosition API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 GetDevicePosition 要求數目上限。其他請求會受到限流。
GetDevicePositionHistory API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 GetDevicePositionHistory 要求數目上限。其他請求會受到限流。
GetGeofence API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 GetGeofence 要求數目上限。其他請求會受到限流。
GetMapGlyphs API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 GetMapGlyphs 要求數目上限。其他請求會受到限流。
GetMapSprites API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 GetMapSprites 要求數目上限。其他請求會受到限流。
GetMapStyleDescriptor API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 GetMapStyleDescriptor 要求數目上限。其他請求會受到限流。
GetMapTile API要求率	每個支援的區域：每秒 500	是	您每秒可發出的 GetMapTile 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
GetPlace API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 GetPlace 要求數目上限。其他請求會受到限流。
ListDevicePositions API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 ListDevicePositions 要求數目上限。其他請求會受到限流。
ListGeofenceCollections API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListGeofenceCollections 要求數目上限。其他請求會受到限流。
ListGeofences API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 ListGeofences 要求數目上限。其他請求會受到限流。
ListKeys API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListKeys 要求數目上限。其他請求會受到限流。
ListMaps API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListMaps 要求數目上限。其他請求會受到限流。
ListPlaceIndexes API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListPlaceIndexes 要求數目上限。其他請求會受到限流。
ListRouteCalculators API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListRouteCalculators 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
ListTagsForResource API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListTagsForResource 要求數目上限。其他請求會受到限流。
ListTrackerConsumers API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListTrackerConsumers 要求數目上限。其他請求會受到限流。
ListTrackers API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 ListTrackers 要求數目上限。其他請求會受到限流。
PutGeofence API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 PutGeofence 要求數目上限。其他請求會受到限流。
SearchPlaceIndexForPosition API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 SearchPlaceIndexForPosition 要求數目上限。其他請求會受到限流。
SearchPlaceIndexForSuggestions API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 SearchPlaceIndexForSuggestions 要求數目上限。其他請求會受到限流。
SearchPlaceIndexForText API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 SearchPlaceIndexForText 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
TagResource API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 TagResource 要求數目上限。其他請求會受到限流。
UntagResource API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 UntagResource 要求數目上限。其他請求會受到限流。
UpdateGeofenceCollection API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 UpdateGeofenceCollection 要求數目上限。其他請求會受到限流。
UpdateKey API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 UpdateKey 要求數目上限。其他請求會受到限流。
UpdateMap API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 UpdateMap 要求數目上限。其他請求會受到限流。
UpdatePlaceIndex API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 UpdatePlaceIndex 要求數目上限。其他請求會受到限流。
UpdateRouteCalculator API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 UpdateRouteCalculator 要求數目上限。其他請求會受到限流。

名稱	預設	可調整	描述
UpdateTracker API要求率	每個支援的區域：每秒 10	是	您每秒可發出的 UpdateTracker 要求數目上限。其他請求會受到限流。
VerifyDevicePosition API要求率	每個支援的區域：每秒 50 個	是	您每秒可發出的 VerifyDevicePosition 要求數目上限。其他請求會受到限流。
每個帳戶的路線計算資源	每個受支援的區域：40	是	您可以為每個帳戶建立的路線計算機資源數目上限。
追踪器消費者	每個受支援的區域：5	否	Tracker 資源可與之關聯的地理圍欄收集數目上限。
每個帳戶的追蹤資源	每個受支援的區域：500	是	您可以為每個帳號建立的 Tracker 資源數目上限。

> **Note**
>
> 您可以使用 Cloudwatch 根據配額監控您的使用情況。如需詳細資訊，請參閱用 CloudWatch 於監控配額的使用情況。

管理您的 Amazon 定位服務配額

Amazon 定 Location Service 與 Service Quotas 整合，這項 AWS 服務可讓您從中央位置檢視和管理配額。如需詳細資訊，請參閱Service Quotas 使用者指南中的什麼是 Service Quotas？。

Service Quotas 可讓您輕鬆查詢 Amazon 位置服務配額的價值。

AWS Management Console

使用主控台檢視 Amazon 位置服務配額

1. 在開啟「Service Quotas」主控台https://console.aws.amazon.com/servicequotas/。
2. 在導覽窗格中，選擇 AWS services (AWS 服務)。
3. 從AWS 服務清單中搜尋並選取 Amazon 位置。

   在 Service quotas (服務配額) 清單中，您可以看到服務配額名稱、套用的值 (如果有的話)、AWS 預設配額，以及配額值是否可調整。
4. 若要檢視服務配額的其他資訊 (例如說明)，請選擇配額名稱。
5. (選用) 若要請求增加配額，請選取您要增加的配額、選取 Request quota increase (請求增加配額)、輸入或選取必要資訊，然後選取 Request (請求)。

若要使用主控台來進一步處理服務配額，請參閱《Service Quotas 使用者指南》。若要請求提高配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

AWS CLI

若要檢視 Amazon 定位服務配額，請使用 AWS CLI

執行下列命令以檢視預設的 Amazon 位置配額。

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

若要使用更多使用 Service Quotas AWS CLI，請參閱服務配額 AWS CLI 命令參考。若要請求提升配額，請參閱 AWS CLI 命令參考中的 `request-service-quota-increase` 命令。

以開發人員身分開始使用 Amazon 定 Location Service

您可以使用 Amazon 定 Location Service，為多種不同的外形規格和系統 (包括後端 Web 服務、Web 應用程式和行動應用程式) 的應用程式提供地理位置相關功能。提供許多工具可協助您建置應用程式，包括 SDK、程式庫和範例程式碼。

本節提供可協助您開始使用 Amazon 位置的資訊和連結。特別是，以下主題提供了對您最有幫助的信息：

- 案例和使用案例— 開發案例清單，以及 Amazon 定 Location Service 如何協助您完成這些案例。
- Amazon 位置開發套件和工具 — 可在使用 Amazon 位置進行程式設計時協助您的軟體開發套件 (SDK) 和程式庫。
- Amazon 定 Location Service API 參考 — 隨 AWS 開發套件隨附的核心 Amazon 位置 API 的參考資料。
- 程式碼範例 — 本節提供的範例可協助您開始使用，或將功能新增至現有應用程式。
- 快速入門教學課程 — 本教學課程向您展示如何建立第一個應用程式。有用於創建一個 Web 應用程序或基於 Android 的移動應用程序教程的版本。
- Amazon Location Service 概念— 本指南的本部分介紹了 Amazon 位置的基本概念，包括地圖，地點搜索，路線以及地理圍欄和跟踪器的部分。
- 擴大 — Amplify 是一個完整的解決方案，封裝了許多使用創建 Web 和移動應用程序所需的功能。AWS 雲端如果您已經在使用 Amplify，或選擇使用 Amplify，它有一個使用內置 Amazon 定位服務的地理庫，您可以使用。若要開始使用 Amplify 地理位置，請參閱這裡的文件。

案例和使用案例

Amazon 定 Location Service 務是在 AWS 雲端. 您可以從雲端中自己的 Amazon EC2 執行個體呼叫它，但許多映射應用程式將在裝置上執行，或是裝置和雲端的組合上執行。下面列出了一些典型的場景，以及如何開發它們。

- 後端應用程式，可協助您最佳化叢集中驅動程式的路由。

  您可以撰寫在 Amazon EC2 上執行的應用程式， AWS 雲端 該應用程式使用 Amazon Location Service 來計算路由矩陣，做為叢集路由優化器的輸入。使用開AWS 發套件撥打電話給 Amazon 位置。
- 一個 Web 應用程序，可讓您的客戶找到您的業務位置。

您可以建立在 Amazon EC2 執行個體上執行的網站，包括基於位置的應用程式。使用 AWS SDK JavaScript 來開發 Web 應用程式，以使用地點搜尋來查詢位置，並使用 MapLibre。使用 Amazon 位置開發套件讓使用位置進行程式設計變得更輕鬆

- 將定位功能新增至現有的 iOS 或 Android 應用程式。

  您可以使用適用於斯威夫特的 AWS SDK（iOS）或 Kotlin（Android）撥打電話給 Amazon 位置，以將地點搜索和地圖功能添加到您的應用程序。用 MapLibre 於彩現地圖。還有其他語言可用的其他 AWS SDK。

- 追蹤資產 (裝置或車輛)，並在進入或離開您定義的區域時取得更新。

  跟踪設備的應用程序由幾個部分組成。

  - 您要跟踪的每個設備都必須創建一個跟踪器資源以對其進行跟踪。它必須將位置更新發送到 Amazon 定 Location Service，例如使用 MQ TT。
  - 建立地理圍欄，以定義您想要取得資產進入和退出事件的區域。
  - 您可以使用 Amazon EC2 或AWS Lambda在資產進入或退出地理圍欄區域時回應事件。
  - 您可以進行擴展以創建 Web 或設備應用程序，以在地圖上跟踪和顯示您的資產位置。

以下部分提供可與 Amazon 定位服務各方面搭配使用的工具和程式庫的詳細 Location Service。

使用 Amazon 定 Location Service 的開發套件和工具

有幾種工具可以幫助您使用 Amazon 定 Location Service。

- AWS SDK — AWS 軟體開發套件 (SDK) 提供許多常用的程式設計語言，提供 API、程式碼範例和文件，讓您更輕鬆地以偏好的語言建置應用程式。 AWS 開發套件包括核心 Amazon 位置 API 和功能，包括存取地圖、地點搜尋、路線、地理圍欄和追蹤器。若要進一步了解可用於不同應用程式和語言的 Amazon 定 Location Service 的 SDK，請參閱依語言分類的 SDK。
- MapLibre— Amazon 定 Location Service 建議使用渲染引擎MapLibre渲染地圖。 MapLibre 是用於在 Web 或移動應用程序中顯示地圖的引擎。 MapLibre 還具有插件模型，並支持用戶界面，用於在某些語言和平台上進行搜索和路由。若要進一步瞭解使用 MapLibre 及其提供的功能，請參閱MapLibre。
- Amazon 位置開發套件 — Amazon 位置開發套件是一組開放原始碼程式庫，可讓您更輕鬆地使用 Amazon 定 Location Service 開發應用程式。這些程式庫提供的功能可支援行動和 Web 應用程式的身分驗證、行動應用程式的位置追蹤、Amazon Location 資料類型和 GeoJSON 之間的轉換，以

及 AWS SDK v3 的 Amazon 位置用戶端託管套件。若要進一步了解 Amazon 定位開發套件，請參閱Amazon 位置 SDK。

依語言分類的 SDK

下表依應用程式類型提供 AWS SDK 和語言和架構 MapLibre 版本的相關資訊：Web、行動裝置或後端應用程式。

> **SDK 版本**
>
> 我們建議您使用最新版本的 AWS SDK，以及您在專案中使用的任何其他 SDK，並使 SDK 保持在最新狀態。 AWS SDK 提供您最新的特色和功能，以及安全性更新。例如，若要尋找 AWS SDK 的最新版本 JavaScript，請參閱 AWS SDK 中的瀏覽器安裝主題以取得 JavaScript 說明文件。

Web frontend

下列 AWS SDK 和 MapLibre 版本可用於 Web 前端應用程式開發。

語言/框架	AWS 開發套件	渲染框架
完全支援		
JavaScript	https://aws.amazon.com/sdk-for-javascript/	https://maplibre.org/projects/maplibre-gl-js/
反應 JS	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/
部分支援		
撲	https://docs.amplify.aws/start/q/integration/flutter/	https://github.com/maplibre/flutter-maplibre-gl

語言/框架	AWS 開發套件	渲染框架
	Flutter 尚未完全支持 AWS，但通過 Amplify 提供有限的支持。	MapLibre 撲圖書館被認為是實驗性的。
Node.js	https://aws.amazon.com/sdk-for-javascript/	沒有對 Node.js 的 MapLibre 支持。
PHP	https://aws.amazon.com/sdk-for-php/	沒有對 PHP 的 MapLibre 支持。

Mobile frontend

下列 AWS SDK 和 MapLibre 版本適用於行動前端應用程式開發。

語言/框架	AWS SDK	渲染框架
完全支援		
Java	https://aws.amazon.com/sdk-for-java/	https://maplibre.org/projects/maplibre-native/
Kotlin	https://aws.amazon.com/sdk-for-kotlin/ Amazon 定 amazon-location-mobile-authL ocation Service 安卓移動身份驗證 SDK：https://github.com/aws-geospatial/ Amazon 定 amazon-location-mobile-trackingLocation Service 移動跟踪 SDK for Android 卓版:https://github.com/aws-geospatial/	https://maplibre.org/projects/maplibre-native/ 需要自定義綁定，基 MapLibre 於 Java 的也是如此。

語言/框架	AWS SDK	渲染框架
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/ 適用於 iOS 的 Amazon 定 amazon-location-mobile-authL ocation Service 移動身份驗證開發套件：https://github.com/aws-geospatial/ 適用於 iOS 的 Amazon 定 amazon-location-mobile-trackingL ocation Service 移動跟踪 SDK：https://github.com/aws-geospatial/	https://maplibre.org/projects/maplibre-native/
部分支援		
撲	https://docs.amplify.aws/start/q/integration/flutter/ Flutter 尚未完全支持 AWS，但通過 Amplify 提供有限的支持。	https://github.com/maplibre/flutter-maplibre-gl MapLibre 撲圖書館被認為是實驗性的。

Backend application

下列 AWS SDK 可用於後端應用程式開發。 MapLibre 此處未列出，因為後端應用程式通常不需要地圖呈現。

語言	AWS 開發套件
.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/
Rust	https://aws.amazon.com/sdk-for-rust/ Rust 適用的 AWS SDK 正在開發人員預覽中。

透過 Amazon 位置使用 MapLibre 工具和程式庫

使用 Amazon 位置創建交互式應用程序的重要工具之一是 MapLibre。MapLibre主要是用於在 Web 或移動應用程序中顯示地圖的渲染引擎。不過，它也包括對外掛程式的支援，並提供與 Amazon 位置其他層面搭配使用的功能。以下說明您可以使用的工具，根據您要使用的位置區域。

> **Note**
>
> 若要使用 Amazon 位置的任何方面，請為您要使用的語言安裝AWS 開發套件。

- 地圖

  要在應用程序中顯示地圖，您需要一個地圖渲染引擎，該引擎將使用 Amazon Location 提供的數據並繪製到屏幕上。地圖渲染引擎還提供了平移和縮放地圖，或將標記或圖釘和其他註釋添加到地圖的功能。

  Amazon 定 Location Service 建議使用轉MapLibre譯引擎呈現地圖。 MapLibre GL JS 是用於顯示地圖的引擎 JavaScript，而 MapLibre 本機提供了 iOS 或安卓系統的地圖。

  MapLibre 還具有插件生態系統來擴展核心功能。欲了解更多信息，請訪問 https://maplibre.org/maplibre-gl-js-docs /插件/。

- 地點搜尋

  為了使創建搜索用戶界面更簡單，您可以使用網絡MapLibre 地理編碼器（Android 應用程序可以使用 Android 地點插件）。

  使用適用於 Maplibre 地理編碼器的 Amazon 位置庫來簡化在應用程式中使用 Amazon 位置的`amazon-location-for-maplibre-gl-geocoder`程序。 JavaScript

- 路線

  要在地圖上顯示路線，請使用MapLibre方向。

- 地理圍欄和跟踪器

  MapLibre 沒有任何用於地理圍欄和跟踪的特定渲染或工具，但是您可以使用渲染功能和插件在地圖上顯示地理圍欄和跟踪的設備。

  正在追蹤的裝置可以使用 MQTT 或手動傳送更新至 Amazon 定 Location Service。地理圍欄事件可以使用. AWS Lambda

許多開放原始碼程式庫都可以為 Amazon 定 Location Service 提供其他功能，例如提供空間分析功能的草坪。

許多圖書館使用開放標準 GeoJSON 格式的數據。Amazon 定 Location Service 提供了一個庫，以支持在 JavaScript 應用程序中使用 GeoJSON。如需詳細資訊，請參閱下一節「Amazon 位置開發套件和庫」。

Amazon 位置 MapLibre 地理編碼器插件

Amazon 位置 MapLibre 地理編碼器外掛程式旨在讓您在使用程式庫處理地圖渲染和地理編碼時，更輕鬆地將 Amazon 位置功能整合到 JavaScript 應用程式中。maplibre-gl-geocoder

安裝

您可以使用以下命 MapLibre 令從 NPM 安裝 Amazon 位置地理編碼器插件以與模塊一起使用：

```
npm install @aws/amazon-location-for-maplibre-gl-geocoder
```

您可以使用腳本導入到 HTML 文件中直接在瀏覽器中使用：

```
<script src="https://www.unpkg.com/@aws/amazon-location-for-maplibre-gl-
geocoder@1">/script<
```

與模塊的用法

此代碼設置了具有 Amazon 位置 JavaScript 地理編碼功能的 Maplibre GL 地圖。它使用透過 Amazon Cognito 身分集區的身份驗證來存取 Amazon 位置資源。該地圖被渲染與指定的樣式和中心坐標，並允許搜索地圖上的地方。

```
// 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());
```

使用瀏覽器

此範例使用 Amazon 位置用戶端提出使用 Amazon Cognito 進行身份驗證的請求。

> **Note**
>
> 其中一些範例使用 Amazon 位置用戶端。Amazon 位置用戶端是以 JavaScript V3 適用的 AWS 開發套件為基礎，可透過 HTML 檔案中參照的指令碼撥打電話給 Amazon 位置。

在 HTML 檔案中包含下列項目：

```
< 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>
```

在檔案中包含下列項 JavaScript 目：

```
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());
```

下面列出的是 Amazon 位置 MapLibre 地理編碼器插件中使用的功能和命令：

- **`buildAmazonLocationMaplibreGeocoder`**

  這個類別會建立一個執行個體，AmazonLocationMaplibreGeocder這是其他所有其他呼叫的進入點：

  ```
  const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client,
   placesIndex, {enableAll: true});
  ```

- **`getPlacesGeocoder`**

  返回一個準備使用的 iControl 對象，可以直接添加到地圖。

```
const geocoder = getPlacesGeocoder();

// Initialize map
let map = await initializeMap();

// Add the geocoder to the map.
map.addControl(geocoder);
```

Amazon 位置開發套件和庫

Amazon 位置開發套件是一組開放原始碼程式庫，可提供開發 Amazon 位置應用程式的實用功能。包括以下功能：

- Amazon 位置客戶端 — AWS SDK v3 中的 Amazon 位置對象捆綁和打包，以便於 Web 開發中使用。
- 身份驗證 — 在為 Amazon 定 Location Service 務構建網頁、JavaScriptiOS 或 Android 應用程序時，身份驗證實用程序簡化了身份驗證（使用 Amazon Cognito 或 API 密鑰）。
- 追蹤 — 行動追蹤 SDK 適用於 iOS 和安卓系統。此開發套件可讓行動應用程式更輕鬆地與 Amazon 位置追蹤器互動。
- Amazon 位置 GeoJSON 函數 — GeoJSON 轉換公用程式可讓您輕鬆地在業界標準 GeoJSON 格式的資料和 Amazon 位置 API 格式之間進行轉換。

主題

- 如何開始使用 Amazon 定位開發套件
- Amazon 位置客戶
- JavaScript 驗證助手
- GeoJSON 助手
- 安卓行動驗證 SDK
- iOS 行動驗證開發套件
- 安卓手機跟蹤 SDK
- iOS 行動追蹤開發套件

如何開始使用 Amazon 定位開發套件

Amazon 定位開發套件是一組功能，可讓您在應用程式中更簡單地使用 Amazon 定位服務。您可以安裝這些功能並將其導入到您的 JavaScript 應用程序中。以下各節說明 Amazon 位置用戶端，以及身份驗證和 GeoJSON 協助程式庫。

Amazon 位置客戶

隨著 AWS SDK v3，SDK 是由服務分開。您可以只安裝所需的零件。例如，若要安裝 Amazon 位置用戶端和 Amazon Cognito 的登入資料提供者，請使用下列命令。

```
npm install @aws-sdk/client-location
npm install @aws-sdk/credential-providers
```

為了方便在 JavaScript Web 前端應用程式中使用 Amazon 定 Location Service，請 AWS 提供 Amazon 位置資料庫和登入資料提供者的託管組合包。若要使用隨附的用戶端，請在指令碼標記中將其新增至 HTML，如下所示：

```
<script src="https://unpkg.com/@aws/amazon-location-client@1.x/dist/
amazonLocationClient.js"></script>
```

> **Note**
>
> 該包裝保持最新狀態，並向後兼容以便於使用。使用此腳本標記或 NPM 安裝將始終獲得最新版本。

JavaScript 驗證助手

Amazon 位置 JavaScript 身份驗證協助程式可讓您更輕鬆地從應用程式進行 Amazon 位置 API 呼叫時進行驗 JavaScript 證。使用 Amazon Cognito 或 API 金鑰做為身份驗證方法時，此身份驗證協助程式會特別為您提供協助。這是一個開放源代碼庫，可以在 GitHub這裡找到：https://github.com/aws-geospatial/ amazon-location-utilities-auth-幫助者 JS。

> **Note**
>
> 身分驗證協助程式中的 Amazon Cognito 支援不支援 Amazon Cognito 的聯合身分識別功能。

安裝

如果您使用 webpack 之類的構建系統，或者在 html 中包含帶有<script>標籤的預 JavaScript 構建包，則可以將庫與本地安裝一起使用。

- 使用下面的命令來安裝庫，使用 NPM：

```
npm install @aws/amazon-location-utilities-auth-helper
```

- 在 HTML 檔案中使用下列命令來載入指令碼：

```
<script src="https://unpkg.com/@aws/amazon-location-utilities-auth-helper@1.x/dist/
amazonLocationAuthHelper.js"></script>
```

Import (匯入)

若要在 JavaScript 應用程式中使用特定函數，您必須匯入該函數。下面的代碼用於withIdentityPoolId將函數導入到您的應用程序。

```
import { withIdentityPoolId } from '@aws/amazon-location-utilities-auth-helper';
```

驗證功能

Amazon 位置身份驗證協助程式包括下列傳回AuthHelper物件的函數：

- async withIdentityPoolId(identityPoolId: string): AuthHelper— 此函數返回一個 AuthHelper 對象，初始化為與 Amazon Cognito 一起工作
- async withAPIKey(API_KEY: string): AuthHelper-此函數返回一個 AuthHelper 對象，初始化為與 API 密鑰一起使用。

該AuthHelper對象提供了以下功能：

- AuthHelper.getMapAuthenticationOptions()— 物件的此函數會傳回 AuthHelper 物 JavaScript 件，其中包含transformRequest可與 MapLibre JS 中的對應選項搭配使用的物件。僅在使用識別集區初始化時才提供。
- AuthHelper.getLocationClientConfig()— AuthHelper 物件的此函數會傳回具有可credentials用來初始化的 JavaScript 物件 LocationClient。

- `AuthHelper.getCredentials()`— AuthHelper 物件的此函數會從 Amazon Cognito 傳回內部登入資料。僅在使用識別集區初始化時才提供。

示例：初始化 MapLibre 地圖對象與 Amazon Cognito，使用 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
});
```

示例：使用 API 密鑰初始化 MapLibre 映射對象（**AuthHelper**在這種情況下不需要）

```
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',
});
```

範例：使用 Amazon Cognito，從 JS AWS 開發套件初始化位置用戶端 AuthHelper

這個例子使用 AWS SDK 作為 JavaScript v3。

```
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
});
```

示例：使用 API 密鑰，從 AWS SDK 初始化位置客戶端 AuthHelper

這個例子使用 AWS SDK 作為 JavaScript v3。

```
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 助手

Amazon 位置 GeoJSON 轉換助手提供工具，可將 Amazon 定 Location Service 數據類型與業界標準 Geo JSON 格式進行轉換。例如，GeoJSON 用於在地圖上渲染地理數據。 MapLibre 這是一個開源庫 GitHub，可以在這裡找到：https://github.com/aws-geospatial/ amazon-location-utilities-datatypes-js。

安裝

您可以將這些庫與本地安裝（例如 webpack）一起使用，也可以在 html 中 JavaScript 包含帶有<script>標籤的預構建包。

- 使用下面的命令來安裝庫，使用 NPM。

```
npm install @aws/amazon-location-utilities-datatypes
```

- 在 HTML 檔案中使用下列命令來載入指令碼：

```
<script src="https://unpkg.com/@aws/amazon-location-utilities-datatypes@1.x/dist/
amazonLocationDataConverter.js"></script>
```

Import (匯入)

若要在 JavaScript 應用程式中使用特定函數，您必須匯入該函數。下面的代碼用於`placeToFeatureCollection`將函數導入到您的應用程序。

```
import { placeToFeatureCollection } from '@aws/amazon-location-utilities-datatypes';
```

GeoJSON 函數

Amazon 位置 GeoJSON 轉換助手包括以下功能：

- `placeToFeatureCollection(place: GetPlaceResponse | searchPlaceIndexForPositionResponse | searchPlaceIndexForTextResponse, keepNull: boolean): Feature`— 此功能將地點搜索功能的響應轉換為 FeatureCollection 具有 1 個或多個 Point 功能的 GeoJSON。
- `devicePositionToFeatureCollection(devicePositions: GetDevicePositionResponse | BatchGetDevicePositionResponse | GetDevicePositionHistoryResponse | ListDevicePositionsResponse, keepNull: boolean)`-此功能將跟踪器設備位置功能的響應轉換為 FeatureCollection 具有 1 個或多個點功能的 GeoJSON。
- `routeToFeatureCollection(legs: CalculateRouteResponse): FeatureCollection`— 此函數將響應從計算路由函數轉換為 FeatureCollection 具有單一 MultiStringLine 功能的 GeoJSON。路線的每一段都由中的一個 LineString 項目表示 MultiStringLine。
- `geofenceToFeatureCollection(geofences: GetGeofenceResponse | PutGeofenceRequest | BatchPutGeofenceRequest | ListGeofencesResponse): FeatureCollection`-此功能將地理圍欄函數請求或響應轉換為 FeatureCollection 具有多邊形功能

的 GeoJSON。它可以在響應和請求中轉換地理圍欄，使您可以在使用或上傳地理圍欄之前在地圖上顯示地理圍欄。 PutGeofence BatchPutGeofence

此功能將圓形地理圍欄轉換為具有近似多邊形的圖徵，但是如果需要，也將具有「中心」和「radius」屬性以重新創建圓形地理圍欄（請參閱下一個功能）。

- `featureCollectionToGeofences(featureCollection: FeatureCollection): BatchPutGeofenceRequestEntry[]`-此函數將 FeatureCollection 具有多邊形特徵的 GeoJSON 轉換為 BatchPutGeofenceRequestEntry 對象數組，因此結果可用於創建請求。 BatchPutGeofence

  如果中的特徵 FeatureCollection 具有「中心」和「半徑」屬性，它將被轉換為一個圓的地理圍欄請求輸入，忽略多邊形的幾何圖形。

範例：將搜尋結果轉換為點圖層 MapLibre

這個例子使用 AWS SDK 作為 JavaScript v3。

```
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",
    },
});
```

安卓行動驗證 SDK

這些公用程式可協助您在從 Android 應用程式進行 Amazon 定 Location Service API 呼叫時進行驗證。這在使用 Amazon Cognito 或 API 金鑰做為身份驗證方法時特別有幫助。

Android 移動身份驗證 SDK 可在 github 上找到：Amazon 定 Location Service 移動身份驗證 SDK for Android。此外，移動身份驗證 SDK 和 AWS SDK 都可以在 AWS Maven 存儲庫中使用。

安裝

若要使用行動驗證 SDK，請在 Android Studio 中將下列匯入陳述式新增至您的`build.gradle`檔案。

```
implementation("software.amazon.location:auth:0.0.1")
implementation("com.amazonaws:aws-android-sdk-location:2.72.0")
```

驗證功能

驗證協助程式 SDK 具有以下功能：

- `authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider`：此函數返回`LocationCredentialsProvider`初始化以使用 API 密鑰。
- `authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider`：此函數會傳回`LocationCredentialsProvider`初始化，以便與 Amazon Cognito 身分識別集區搭配使用。

用量

若要在程式碼中使用 SDK，請匯入下列類別：

```
import com.amazonaws.services.geo.AmazonLocationClient
import software.amazon.location.auth.AuthHelper
import software.amazon.location.auth.LocationCredentialsProvider
```

建立驗證協助程式和位置用戶端提供者執行個體時，有兩個選項。您可以使用亞馬遜位置 API 金鑰或 Amazon Cognito 建立執行個體。

- 若要使用 Amazon 位置 API 金鑰建立身份驗證協助程式執行個體，請按如下方式宣告協助程式類別：

```
var authHelper = AuthHelper(applicationContext)
var locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key")
```

- 若要使用 Amazon Cognito 建立身份驗證協助程式執行個體，請如下宣告協助程式類別：

```
var authHelper = AuthHelper(applicationContext)
var locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id")
```

您可以使用位置登入資料供應商建立 Amazon 位置用戶端執行個體，並撥打電話給 Amazon 位置服務。下列範例會搜尋指定緯度和經度附近的地點。

```
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 行動驗證開發套件

這些公用程式可協助您在從 iOS 應用程式進行 Amazon 定 Location Service API 呼叫時進行驗證。這在使用 Amazon Cognito 或 API 金鑰做為身份驗證方法時特別有幫助。

iOS 移動身份驗證 SDK 可在 github 上找到：適用於 iOS 的 Amazon 定 Location Service 移動身份驗證 SDK。

安裝

在 Xcode 項目中安裝 SDK：

1. 轉到文件，然後在 XCode 項目中選擇添加 Package 依賴項。
2. 在搜索欄中輸入軟件包網址：https://github.com/aws-geospatial/ amazon-location-mobile-auth-sdk-iOS/ 進入搜索欄，然後按回車鍵。
3. 選擇 P `amazon-location-mobile-auth-sdk-ios` ackage 件，然後按下「新增套件」。
4. 選擇 P `AmazonLocationiOSAuthSDK` ackage 產品，然後按添加包裹。

驗證功能

驗證協助程式 SDK 具有以下功能：

- `authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider`：此函數返回`LocationCredentialsProvider`初始化以使用 API 密鑰。
- `authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider`：此函數會傳回`LocationCredentialsProvider`初始化，以便與 Amazon Cognito 身分識別集區搭配使用。

用量

若要使用行動驗證 SDK，請將下列陳述式新增至您的活動：

```
import AmazonLocationiOSAuthSDK
import AWSLocationXCF
```

建立驗證協助程式和位置用戶端提供者執行個體時，有兩個選項。您可以使用亞馬遜位置 API 金鑰或 Amazon Cognito 建立執行個體。

- 若要使用 Amazon 位置 API 金鑰建立身份驗證協助程式執行個體，請按如下方式宣告協助程式類別：

  ```
  let authHelper = AuthHelper()
  let locationCredentialsProvider = authHelper.authenticateWithAPIKey(apiKey: "My-Amazon-Location-API-Key", region: "account-region")
  ```

- 若要使用 Amazon Cognito 建立身份驗證協助程式執行個體，請如下宣告協助程式類別：

  ```
  let authHelper = AuthHelper()
  let locationCredentialsProvider =
   authHelper.authenticateWithCognitoUserPool(identityPoolId: "My-Amazon-Location-API-Key", region: "account-region")
  ```

您可以使用位置登入資料供應商建立 Amazon 位置用戶端執行個體，並撥打電話給 Amazon 位置服務。下列範例會搜尋指定緯度和經度附近的地點。

```
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)
```

安卓手機跟蹤 SDK

Amazon Location 行動追蹤開發套件提供的公用程式可協助您輕鬆驗證、擷取裝置位置，以及將位置更新傳送至 Amazon 位置追蹤器。SDK 支援使用可設定的更新間隔對位置更新進行本機篩選。這樣可以降低數據成本並優化 Android 應用程序的間歇性連接。

安卓跟蹤 SDK 可用於 GitHub:Amazon 定位移動跟蹤 SDK for Android. 此外，移動身份驗證 SDK 和 AWS SDK 都可以在 AWS Maven 存儲庫中使用。安卓跟踪 SDK 旨在與一般 AWS SDK 一起使用。

本節涵蓋 Amazon 定位移動跟踪 Android SDK 的以下主題：

主題

- 安裝
- 用量
- 篩選條件
- 安卓手機 SDK 追蹤功能
- 範例

安裝

要安裝 SDK，請將以下幾行添加到 Android 工作室中構建 .gradle 文件的依賴關係部分：

```
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")
```

用量

此程序說明如何使用 SDK 驗證和建立`LocationTracker`物件：

> **Note**
>
> 此程序假設您已匯入本安裝節中提到的資源庫。

1. 在程式碼中匯入下列類別：

```
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. 接下來創建一個AuthHelper，因為LocationCredentialsProvider參數是創建對LocationTracker象所需的：

```
// Create an authentication helper using credentials from Cognito
val authHelper = AuthHelper(applicationContext)
val locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id")
```

3. 現在，使用LocationCredentialsProvider和LocationTrackerConfig來創建一個LocationTracker對象：

```
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,
)
```

篩選條件

Amazon 位置移動跟踪 Android SDK 具有三個內置的位置過濾器。

- `TimeLocationFilter`：根據定義的時間間隔過濾要上傳的當前位置。
- `DistanceLocationFilter`：根據指定的距離閾值篩選位置更新。
- `AccuracyLocationFilter`：透過比較自上次更新後移動的距離與目前位置的準確度來篩選位置更新。

此範例會在建立時`LocationTracker`在中新增篩選器：

```
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,
)
```

此範例會在執行階段啟用和停用篩選器`LocationTracker`：

```
// To enable the filter
locationTracker?.enableFilter(TimeLocationFilter())

// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())
```

安卓手機 SDK 追蹤功能

適用於 Android 的 Amazon 位置移動跟踪 SDK 包括以下功能：

- 類別：`LocationTracker`

```
constructor(context: Context,locationCredentialsProvider:
LocationCredentialsProvider,trackerName: String), 或
constructor(context: Context,locationCredentialsProvider:
LocationCredentialsProvider,clientConfig: LocationTrackerConfig)
```

這是一個初始化函數來創建一個LocationTracker對象。它需要的執行個體LocationCredentialsProvider，以trackerName及選擇性的執行個體LocationTrackingConfig。如果未提供配置，則將使用默認值初始化。

- 類別：LocationTracker

```
start(locationTrackingCallback: LocationTrackingCallback)
```

開始存取使用者位置並將其傳送至 Amazon 位置追蹤器的程序。

- 類別：LocationTracker

```
isTrackingInForeground()
```

檢查位置跟踪當前是否正在進行中。

- 類別：LocationTracker

```
stop()
```

停止追蹤使用者位置的程序。

- 類別：LocationTracker

```
startTracking()
```

開始訪問用戶的位置並將其發送到 AWS 跟踪器的過程。

- 類別：LocationTracker

```
startBackground(mode: BackgroundTrackingMode, serviceCallback:
ServiceCallback)
```

啟動訪問用戶的位置，並在應用程序在後台發送到 AWS 跟踪器的過程。 BackgroundTrackingMode 具有以下選項：

  - ACTIVE_TRACKING：此選項會主動追蹤使用者的位置更新。
  - BATTERY_SAVER_TRACKING：此選項每 15 分鐘跟踪用戶的位置更新。

- 類別：LocationTracker

```
stopBackgroundService()
```

停止在應用程序在後台訪問用戶的位置並將其發送到 AWS 跟踪器的過程。

- 類別：LocationTracker

```
getTrackerDeviceLocation()
```

從 Amazon 定位服務擷取裝置位置。

- 類別：LocationTracker

```
getDeviceLocation(locationTrackingCallback: LocationTrackingCallback?)
```

從融合的位置供應商用戶端擷取目前的裝置位置，並將其上傳到 Amazon 位置追蹤器。

- 類別：LocationTracker

```
uploadLocationUpdates(locationTrackingCallback:
LocationTrackingCallback?)
```

根據設定的位置篩選器進行篩選後，將裝置位置上傳至 Amazon 位置服務。

- 類別：LocationTracker

```
enableFilter(filter: LocationFilter)
```

啟用特定的位置篩選。

- 類別：LocationTracker

```
checkFilterIsExistsAndUpdateValue(filter: LocationFilter)
```

禁用特定的位置過濾器。

- 類別：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())`

這會使用使用者定義`LocationTrackerConfig`的參數值來初始化。如果未提供參數值，則會將其設定為預設值。

- 類別：`LocationFilter`

  `shouldUpload(currentLocation: LocationEntry, previousLocation: LocationEntry?): Boolean`

  這`LocationFilter`是使用者可以為其自訂篩選器實作實作的通訊協定。您需要實現該`shouldUpload`功能來比較以前和當前位置，如果當前位置應該上傳返回。

範例

下列程式碼範例顯示行動追蹤 SDK 功能。

此範例使用在`LocationTracker`背景中啟動和停止追蹤：

```
// 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 行動追蹤開發套件

Amazon Location 行動追蹤開發套件提供的公用程式可協助您輕鬆驗證、擷取裝置位置，以及將位置更新傳送至 Amazon 位置追蹤器。SDK 支援使用可設定的更新間隔對位置更新進行本機篩選。這樣可以降低數據成本並優化 iOS 應用程序的間歇性連接。

iOS 跟踪 SDK 可在以下網站上使用 GitHub：適用於 iOS 的 Amazon 位置移動跟踪 SDK。

本節涵蓋 Amazon 定位移動跟踪 iOS SDK 的以下主題：

主題

- 安裝
- 用量
- 篩選條件
- iOS 行動 SDK 追蹤功能
- 範例

安裝

使用下列程序來安裝 iOS 版行動追蹤 SDK：

1. 在您的 Xcode 項目中，轉到文件並選擇添加 Package 依賴項。
2. 輸入以下網址：https://github.com/aws-geospatial/ amazon-location-mobile-tracking-sdk-iOS/ 進入搜索欄，然後按回車鍵。
3. 選擇amazon-location-mobile-tracking-sdk-ios軟件包，然後單擊添加包。
4. 選擇 P AmazonLocationiOSTrackingSDK ackage 產品，然後單擊添加包裹。

用量

下列程序說明如何使用 Cognito 的認證建立驗證協助程式。

1. 安裝程式庫之後，您需要在檔案中新增一或兩個info.plist描述：

```
Privacy - Location When In Use Usage Description
Privacy - Location Always and When In Use Usage Description
```

2. 接下來， AuthHelper 在你的班級中導入：

```
import AmazonLocationiOSAuthSDKimport AmazonLocationiOSTrackingSDK
```

3. 然後，您將建立AuthHelper物件並將其與 AWS SDK 搭配使用，方法是使用 Amazon Cognito 的登入資料建立身份驗證協助程式。

```
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)
```

篩選條件

Amazon 位置移動跟踪 iOS SDK 具有三個內置的位置過濾器。

- `TimeLocationFilter`：根據定義的時間間隔過濾要上傳的當前位置。
- `DistanceLocationFilter`：根據指定的距離閾值篩選位置更新。
- `AccuracyLocationFilter`：透過比較自上次更新後移動的距離與目前位置的準確度來篩選位置更新。

此範例會在建立時`LocationTracker`在中新增篩選器：

```
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,
)
```

此範例會在執行階段啟用和停用篩選器LocationTracker：

```
// To enable the filter
locationTracker?.enableFilter(TimeLocationFilter())

// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())
```

iOS 行動 SDK 追蹤功能

適用於 iOS 的 Amazon 位置移動跟踪 SDK 包含以下功能：

- 類別：LocationTracker

  init(provider: LocationCredentialsProvider, trackerName: String, config: LocationTrackerConfig? = nil)

  這是一個初始化函數來創建一個LocationTracker對象。它需要的執行個體LocationCredentialsProvider，以trackerName及選擇性的執行個體LocationTrackingConfig。如果未提供配置，則將使用默認值初始化。

- 類別：LocationTracker

  setTrackerConfig(config: LocationTrackerConfig)

  這將 Tracker 的配置在位置跟踪器初始化後的任何時候生效

- 類別：LocationTracker

  getTrackerConfig()

  這將獲取要在應用程序中使用或修改的位置跟踪配置。

  返回：LocationTrackerConfig

- 類別：LocationTracker

`getDeviceId()`

獲取位置跟踪器生成的設備 ID。

返回：`String?`

- 類別：`LocationTracker`

`startTracking()`

開始訪問用戶的位置並將其發送到 AWS 跟踪器的過程。

- 類別：`LocationTracker`

`resumeTracking()`

恢復訪問用戶位置並將其發送到 AWS 跟踪器的過程。

- 類別：`LocationTracker`

`stopTracking()`

停止追蹤使用者位置的程序。

- 類別：`LocationTracker`

`startBackgroundTracking(mode: BackgroundTrackingMode)`

啟動訪問用戶的位置，並在應用程序在後台發送到 AWS 跟踪器的過程。
`BackgroundTrackingMode`具有以下選項：

- `Active`:此選項不會自動暫停位置更新。
- `BatterySaving`:此選項會自動暫停位置更新
- `None`:此選項整體停用背景位置更新

- 類別：`LocationTracker`

`resumeBackgroundTracking(mode: BackgroundTrackingMode)`

恢復訪問用戶的位置，並在應用程序在後台發送到 AWS 跟踪器的過程。

- 類別：`LocationTracker`

`stopBackgroundTracking()`

停止在應用程序在後台訪問用戶的位置並將其發送到 AWS 跟踪器的過程。

- 類別：LocationTracker

  getTrackerDeviceLocation(nextToken: String?, startTime: Date? = nil, endTime: Date? = nil, completion: @escaping (Result<GetLocationResponse, Error>)

  檢索在開始和結束日期和時間之間的用戶設備上傳的跟踪位置。

  返回：Void
- 類別：LocationTrackerConfig

  init()

  這將 LocationTrackerConfig 使用默認值初始化。
- 類別：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)

  這會使用使用者定義LocationTrackerConfig的參數值來初始化。如果未提供參數值，則會將其設定為預設值。
- 類別：LocationFilter

  shouldUpload(currentLocation: LocationEntity, previousLocation: LocationEntity?, trackerConfig: LocationTrackerConfig)

  這LocationFilter是使用者可以為其自訂篩選器實作實作的通訊協定。用戶將需要實現shouldUpload功能來比較以前和當前的位置，如果當前位置應該上傳返回。

範例

本節詳細介紹了 iOS 版 Amazon 定位移動跟踪 SDK 的使用示例。

> **Note**
>
> 請確定已在info.plist檔案中設定必要的權限。這些權限與用量區段中列出的權限相同。

下列範例示範追蹤裝置位置和擷取追蹤位置的功能：

```
Privacy - Location When In Use Usage Description
Privacy - Location Always and When In Use Usage Description
```

開始跟踪位置：

```
do {
    try locationTracker.startTracking()
    }
catch TrackingLocationError.permissionDenied {
        // Handle permissionDenied by showing the alert message or opening the app
 settings
        }
```

繼續跟踪位置：

```
do {
    try locationTracker.resumeTracking()
    }
catch TrackingLocationError.permissionDenied {
    // Handle permissionDenied by showing the alert message or opening the app settings
    }
```

停止跟踪位置：

```
locationTracker.stopTracking()
```

開始背景追蹤：

```
do {

 locationTracker.startBackgroundTracking(mode: .Active) // .Active, .BatterySaving, .None
    }
catch TrackingLocationError.permissionDenied {
   // Handle permissionDenied by showing the alert message or opening the app settings
   }
```

繼續背景追蹤：

```
do {
```

```
    locationTracker.resumeBackgroundTracking(mode: .Active)
    }
catch TrackingLocationError.permissionDenied {
    // Handle permissionDenied by showing the alert message or opening the app settings
    }
```

若要停止背景追蹤：

```
locationTracker.stopBackgroundTracking()
```

從跟踪器中檢索設備的跟踪位置：

```
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 位置 API

Amazon 定 Location Service 提供 API 操作，以程式設計方式存取位置功能。這包括用於地圖，地點，路線，跟踪器，地理圍欄和標記資源的 API。如需可用 API 動作的相關資訊，請參閱 Amazon 定 Location Service API 參考資料。

您可以在本指南的一程式碼範例章中找到範例。

使用 Amazon 位置與 AWS SDK

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和文件，讓開發人員能夠更輕鬆地以慣用的語言建置 AWS 應用程式。

如需有關可用於依語言與 Amazon 定 Location Service 搭配使用的 SDK 的詳細資訊，請參閱本指南依語言分類的 SDK中的。

> **SDK 版本**
>
> 我們建議您使用最新版本的 AWS SDK，以及您在專案中使用的任何其他 SDK，並使 SDK 保持在最新狀態。 AWS SDK 提供您最新的特色和功能，以及安全性更新。例如，要查找 AWS SDK 的最新版本 JavaScript，請參閱 AWS SDK 中的瀏覽器安裝主題以獲取 JavaScript文檔。

Amazon 位置 API 錯誤消息更新

自 2023 年 8 月 1 日起，Amazon 位置團隊正在變更 API 錯誤訊息，如下表所述。錯誤代碼將不會更改。如果您的應用程式依賴於確切的錯誤訊息字串，則必須使用新字串更新應用程式。有關問題或問題的幫助，請聯繫 AWS Support。

主題

- 地方
- 地圖
- 追蹤器
- 路由
- 中繼資料
- 地理圍欄

地方

地方

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
500	InternalServerException	Internal Server Exception	Internal server error. Try again later.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
404	ResourceNotFoundException	resource <PlaceIndexName> not found, reason: <Reason> Resource '<PlaceIndexName>' not found placeIdx<PlaceIndexName> not found, reason: <Reason> no place index with name '%s' found	Place index not found: <PlaceIndexName>.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
400	ValidationException	'IntendedUse' and 'PricingPlan' must both be provided to update either property	'IntendedUse' and 'PricingPlan' must both be provided to update either attribute .
402	ServiceQuotaExceeded Exception	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>.

地圖

地圖

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
500	InternalServerException	Internal Server Exception unable to find style template Error fetching style	Internal server error. Try again later.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
		was not able to serialize the map style file	
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>.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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>).
400	ValidationException	file type extension part of 'Y' must be one of <Supporte dFileFormats> (or may be omitted) for style '<Style>'	File type extension part of 'Y' must be one of <Supporte dFileFormats> (or may be omitted) for style <Style>.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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>.

追蹤器

追蹤器

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
500	InternalServerException	Internal Server Exception internal server error unable to retrieve point from the storage unable to verify tracker Error processing List request	Internal server error. Try again later.
404	ResourceNotFoundException	tracker not found: <TrackerName>	Tracker not found: <TrackerName>.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
		Tracker with name <TrackerName> was not found	
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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'.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
400	ValidationException	invalid key state	KMS key must be a symmetric Customer Master Key (CMK). Invalid state found. For more informati on about how key state affects the use of a KMS key, see https://docs.aws.a mazon.com/kms/late st/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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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/.
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>.

路由

路由

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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]'.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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>.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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>.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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>.

中繼資料

中繼資料

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
500	InternalServerException	Internal Server Error 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>.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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>.

地理圍欄

地理圍欄

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
500	InternalServerException	internal server error Internal server error Unsupported geofence geometry encountered geometry marshal error geometry load error unable to get geofence collection unable to delete geofences unable to retrieve geofence Error processing List request	Internal server error. Try again later.
404	ResourceNotFoundException	collection not found: <GeofenceCollectionName> <GeofenceCollectionName> geofence collection not found Resource not found error	Geofence Collection not found: <GeofenceCollectionName>.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
		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'.
400	ValidationException	nothing to update	At least one of the following fields must be set: 'Description'

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
400	ValidationException	duplicate geofence ID in batch	'GeofenceId' <DuplicatedGeofenceID> is duplicated in batch.
400	ValidationException	missing GeofenceId	'GeofenceId' must not be empty.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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'.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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>.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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.
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.

錯誤代碼	異常情形	舊的錯誤訊息	新錯誤訊息
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>.

使用 Amazon 定 Location Service 的程式碼範例和教學課程

本主題顯示程式碼範例、教學課程和部落格文章的清單，以協助您了解 Amazon 定 Location Service。每個程式碼範例都包含其運作方式的描述。

您可以在地AWS 理空間 GitHub頁面、Amazon 位置的AWS 範例 GitHub 頁面和AWS 部落格網站上找到其他範例。

> **Note**
>
> 最好理解 AWS 地理空間 GitHub 頁面和 AWS 示例 GitHub 頁面之間的區別。

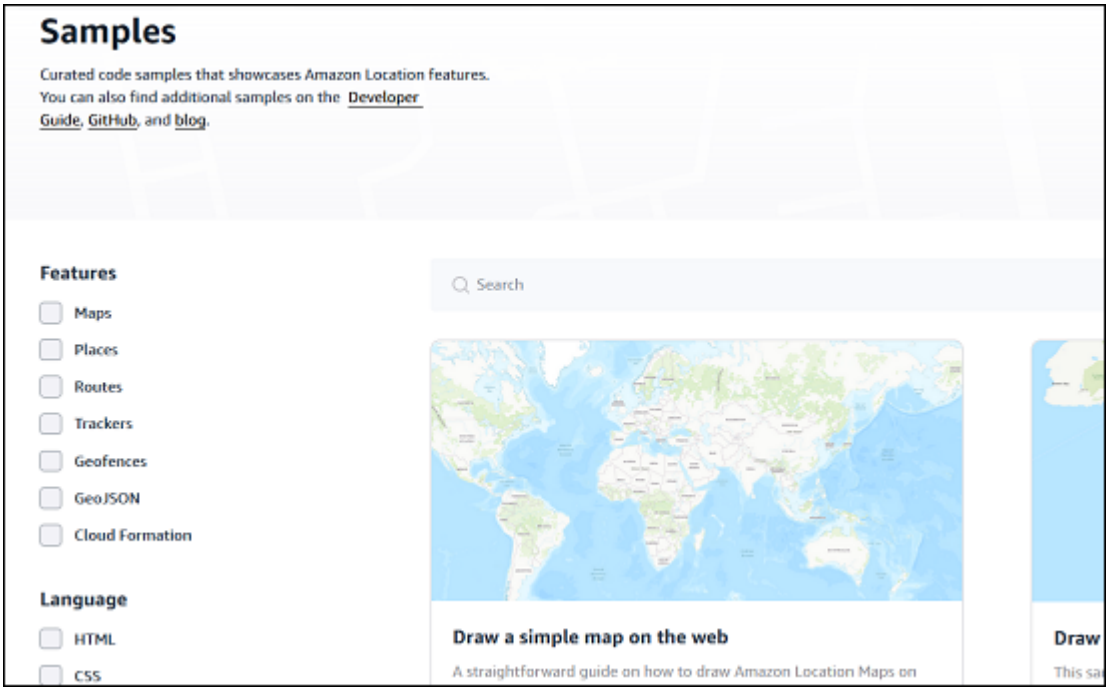
- 地理空間 GitHub — AWS 地理空間 GitHub 頁面包含由 Amazon 定 Location Service 團隊建立和維護的範例。
- 範例 GitHub — Amazon 位置的AWS 範例 GitHub 頁面包含針對 Amazon 位置建立的範例，但可能會或可能不會主動維護。

快速入門教學課程是在使用其他範例之前開始的好地方，因為它展示瞭如何完成對大多數範例有用的必要條件。

主題

- Amazon 位置演示站點
- 教學課程：快速入門
- 教學課程：資料庫豐富
- 範例：探索應用程式
- 範例：設定地圖型式
- 範例：繪製標記
- 範例：繪製叢集點
- 範例：繪製多邊形
- 範例：變更地圖語言
- 部落格：預估交貨時間通知
- 範例：串流位置更新
- 範例：地理圍欄和追蹤行動應用程式

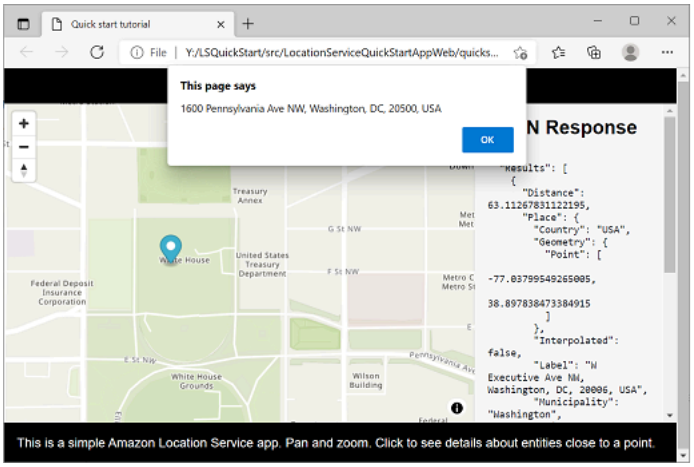
Amazon 位置演示站點

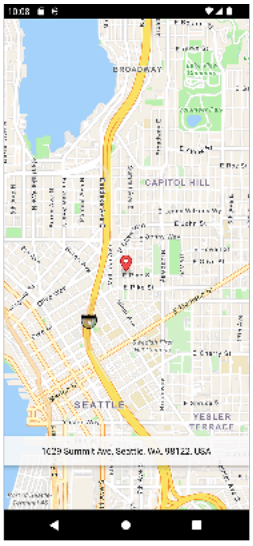


您可以在 Amazon 位置演示站點上看到帶有 Amazon Location Service 源代碼的演示。該網站包括託管的網絡演示，以及適用於 Android 的演示應用程序。

您也可以在網站的「範例」頁面中找到各式各樣的範例，可依功能、語言和平台篩選。

教學課程：快速入門

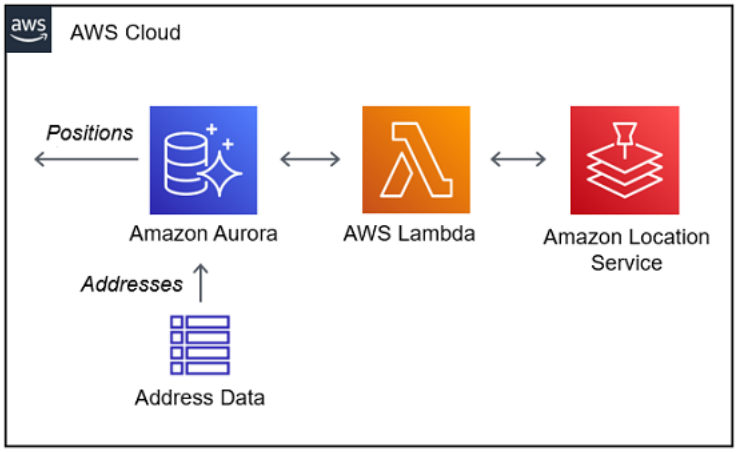




有一些適用於網頁、iOS 和安卓裝置的快速入門教學課程。對於每個平台，本教學會示範如何將互動式地圖新增至應用程式，以及如何從應用程式呼叫 Amazon 定 Location Service API。該教程可用 JavaScript 於靜態網頁, Kotlin 的 Android 手機應用程序, 或斯威夫特的 iOS 應用程序.

- JavaScript 對於靜態網頁文檔鏈接：建立網路應用程式
- 安卓應用程序文檔鏈接的科特林：快速入門 Amazon Location Service
- 斯威夫特的 iOS 應用程序文檔鏈接：建立 iOS 應用程式

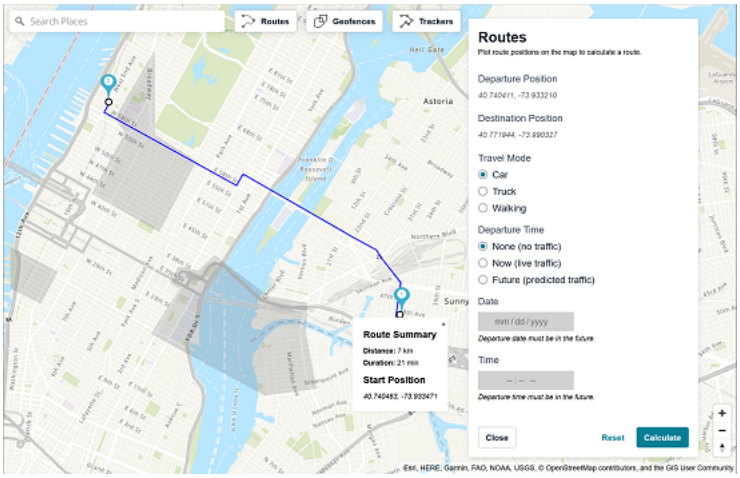
教學課程：資料庫豐富



本教學將說明如何使用 Amazon 定 Location Service，呼叫 AWS Lambda 來源以標準化地址，以及如何將緯度和經度新增至 Amazon Aurora 資料庫中的記錄。使用 Amazon Aurora 和 AWS Lambda.

文檔鏈接：Amazon Aurora PostgreSQLAmazon 定 Location Service 的使用者定義函數

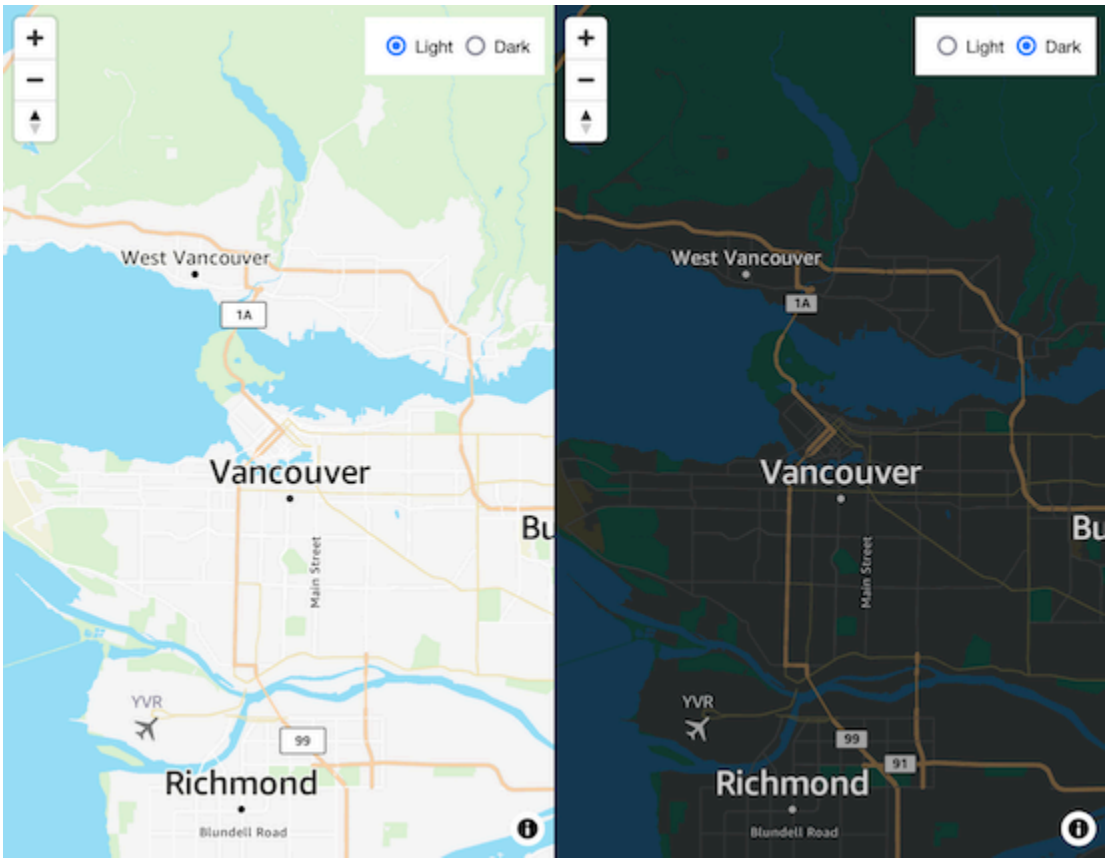
範例：探索應用程式



瞭解 Amazon 定位服務功能的最佳方式之一，就是使用 Amazon 位置主控台內的探索功能。這個完整的 Web 應用程序示例模仿地圖，地點，路線，地理圍欄和跟踪器功能從控制台向您展示如何在自己的應用程序中重新創建這些功能。使用 Amplify，反應和 JavaScript。

範例 GitHub 連結：探索範例應用程式

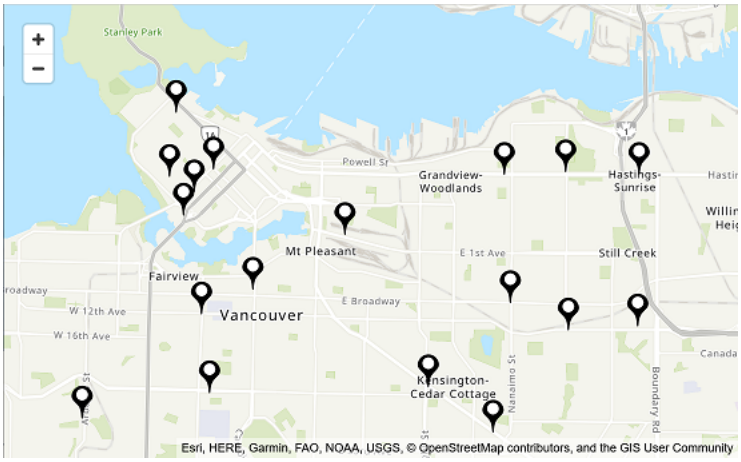
範例：設定地圖型式



此程式碼範例顯示如何使用 MapLibre in 在衛星地圖和向量路線圖之間切換 JavaScript。使用方式 MapLibre、Amazon 位置身份驗證協助程式和 JavaScript.

地理空間 GitHub 鏈接：交互式地圖與樣式切換

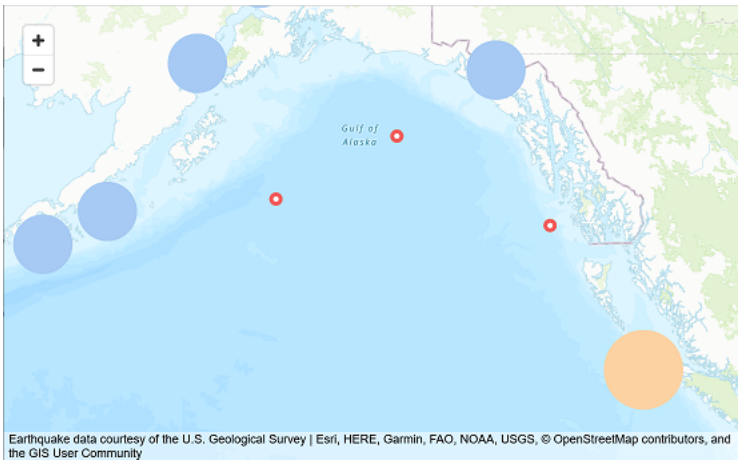
範例：繪製標記



此程式碼範例顯示位於加拿大卑詩省溫哥華的 Amazon 儲物櫃位置。它展示了如何在點位置繪製標識。使用 MapLibre，Node.js，反應，Amazon 位置身份驗證助手和 JavaScript.

空間 GitHub 連結：在點處具有標識的互動式地圖

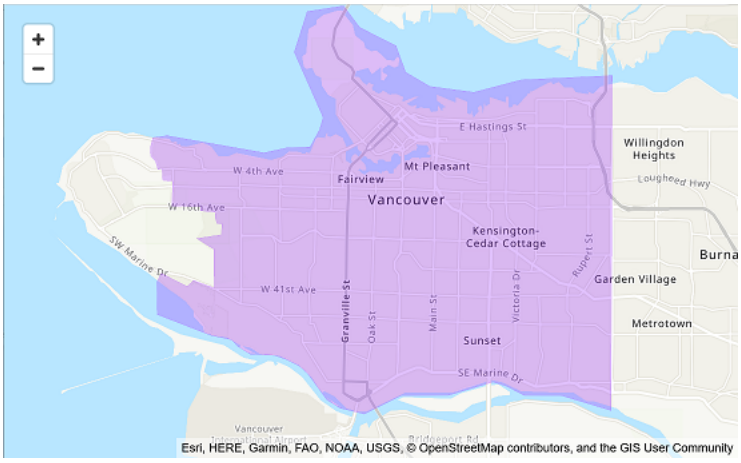
範例：繪製叢集點



使用 USGS 地震資料，此程式碼範例顯示如何在地圖上繪製聚集在一起的點。用途 MapLibre, Node.js, 反應, Amplify, 和 JavaScript.

範例 GitHub 連結：具有點叢集的互動式地圖

範例：繪製多邊形



此代碼示例演示瞭如何在地圖上繪製多邊形。使用 MapLibre，Node.js，反應，Amazon 位置身份驗證幫助程序和 JavaScript.

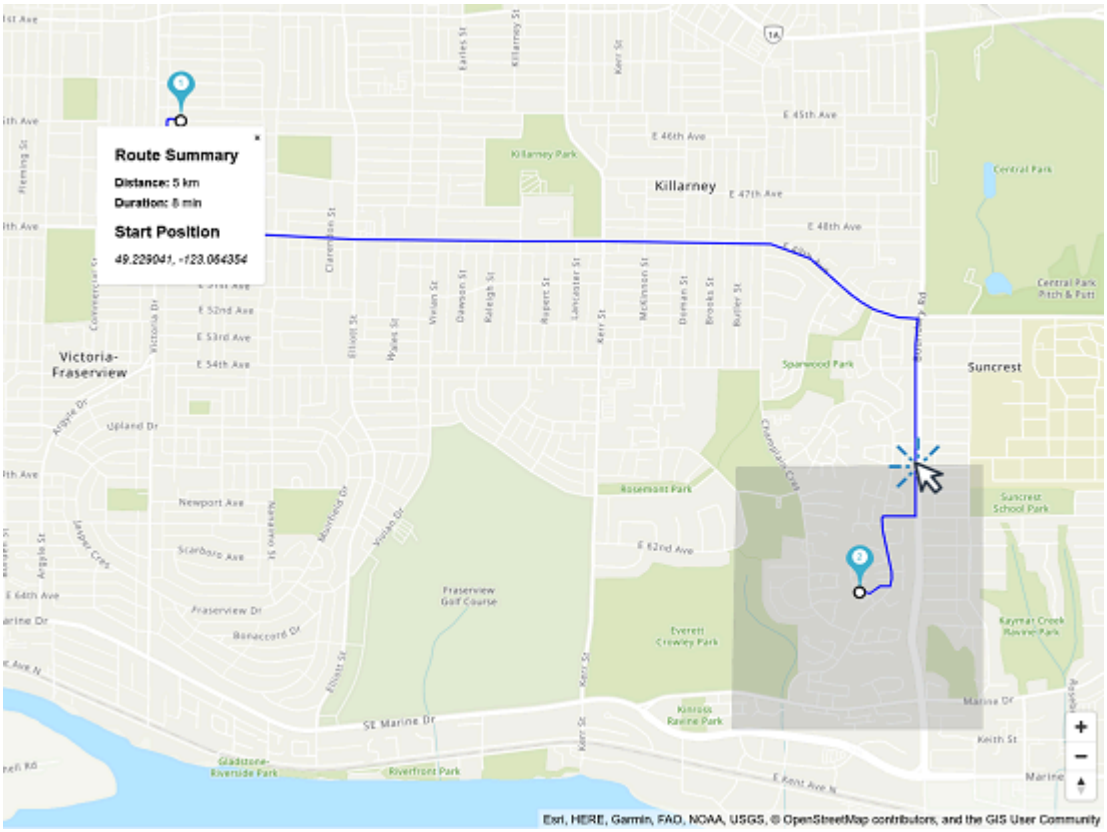
地理空間 GitHub 鏈接：交互式地圖與多邊

範例：變更地圖語言

此程式碼範例顯示如何變更 Amazon 位置中地圖的顯示語言。使用 Amplify，反應和 MapLibre。

範例 GitHub 連結：變更地圖語言範例

部落格：預估交貨時間通知



此部落格文章顯示不同的方式來通知客戶預估的交貨時間。它解釋了使用路線來顯示估計的行駛時間，然後使用跟踪器和地理圍欄來通知當驅動程序接近客戶。使用 Amplify，反應 EventBridge，Amazon 和 Amazon Simple Notification Service (Amazon SNS)。

博客鏈接：預計到達時間和接近通知

範例：串流位置更新

Kinesis 串流至追蹤器應用程式：本範例示範如何使用 Kinesis 資料串流，透過 Amazon 定 Location Service 發佈追蹤器更新。此範例是以 python 撰寫的可部署 lambda 應用程式，可與 Kinesis 資料串流整合，以使用 Kinesis 事件和批次更新裝置位置。

存儲庫鏈接：Amazon 位置 Amazon Kinesis Data Streams 流到跟踪器應用

有關跟踪和地理圍欄的更多信息，請參閱地理圍欄和跟踪器文檔。開發人員可以遵循 AWS 的無伺服器應用程式儲存庫文件，或從 Lambda 主控台直接部署應用程式。

裝置位置串流範例應用程式：此程式碼範例顯示如何將裝置位置資料串流至 Kinesis Data Stream，以及地理圍欄通知的運作方式。此應用程式取決於上面列出的 Kinesis 串流到追蹤器範例應用程式，以便在 Amazon 定位服務中更新串流追蹤器位置。

儲存庫連結：Amazon 定位裝置位置串流範例應用程式

範例：地理圍欄和追蹤行動應用程式

此範例應用程式顯示追蹤器和地理圍欄如何使用 Lambda AWS IoT 和 Amazon 位置功能的組合進行互動。有適用於 iOS 和安卓系統的教程。

教程鏈接：示例地理圍欄和跟踪器移動應用程序

如何使用 Amazon 定 Location Service

您可以使用 Amazon 定 Location Service 功能來完成地理和位置相關的任務。然後，您可以結合這些任務以解決更複雜的使用案例，例如地理營銷，交付和資產跟踪。

當您準備好在應用程式中建置位置功能時，請根據您的目標和傾向，使用下列方法使用 Amazon 定 Location Service 功能：

- 探索工具 — 如果您想嘗試使用 Amazon 位置資源，以下工具是存取和試用 API 的最快方法：
  - Amazon 位置主控台提供各種快速存取工具。您可以使用「探索」頁面建立和管理資源，並嘗試使用 API。主控台也適用於建立資源 (通常是一次性工作)，以準備使用後面描述的任何其他方法。
  - 命AWS令列界面 (CLI) 可讓您使用終端機建立資源和存取 Amazon 位置 API。當您使用認證設定驗證時，會AWS CLI處理驗證。
  - 您可以查看程式碼範例和教學課程，說明如何使用 Amazon 定位服務 API 執行任務。這包括一個範例，模仿主控台中「探索」(Explore) 頁面的大部分功能。
- 平台 SDK — 如果您沒有在地圖上以視覺化方式呈現資料，則可以使用任何AWS標準工具進行建置。AWS
  - 下列軟體開發套件可供使用：C++、圍棋、Java JavaScript、.NET、Node.js、PHP、Python 和紅寶石。
- 前端 SDK 和程式庫 — 如果您想要使用 Amazon Location 在行動平台上建置應用程式，或在任何平台上將地圖上的資料視覺化，您可以使用下列選項：
  - 這些AWS Amplify圖書館在 iOS，安卓系統和 JavaScriptWeb 應用程序中集成了 Amazon 位置。
  - 這些程式 MapLibre 庫可讓您使用 Amazon 位置將用戶端地圖轉譯到 iOS、Android 和 JavaScriptWeb 應用程式中。
  - 七巧板 ES 程式庫使您能夠在 iOS 和安卓網頁應用程式中使用 OpenGL ES 來呈現向量資料的 2D 和 3D 地圖。還有適用於 JavaScriptWeb 應用程序的七巧板。
- 傳送直接 HTTPS 請求 — 如果您使用的是沒有可用 SDK 的程式設計語言，或者想要對傳送請求的方式進行更多控制AWS，則可以透過傳送經簽名第 4 版簽署程序驗證的直接 HTTPS 請求來存取 Amazon 位置。如需簽名第 4 版簽署程序的詳細資訊，請參閱 AWS 一般參考.

本章說明使用位置資料的應用程式常見的許多工作。常見使用案例一節說明如何將這些服務與其他AWS服務結合起來，以實現更複雜的使用案例。

主題

- 使用 Amazon 定 Location Service 的先決條件
- 在您的應用程式中使用 Amazon 位置地圖
- 使用 Amazon 位置搜尋地點和地理位置資料
- 使用 Amazon 定 Location Service 計算路線
- 使用 Amazon 位置對感興趣的區域進行地理圍欄
- 標記您的 Amazon Location Service 資源
- 授予對 Amazon 定 Location Service 的訪問
- 監控 Amazon Location Service
- 創建 Amazon Location Service 資源 AWS CloudFormation

使用 Amazon 定 Location Service 的先決條件

本節說明使用 Amazon 定 Location Service 需執行的動作。您必須擁有 AWS 帳戶 並且已為想要使用它的使用者設定對 Amazon 位置的存取權限。

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

授予對 Amazon 定 Location Service 的訪問

依預設，您的非管理員使用者沒有權限。在他們可以存取 Amazon 位置之前，您必須透過附加具有特定許可的 IAM 政策來授予許可。授予資源存取權時，請務必遵循最低權限原則。

> **Note**
>
> 如需授予未經驗證使用者存取 Amazon Location Service 功能的相關資訊 (例如，在網頁型應用程式中)，請參閱授予對 Amazon 定 Location Service 的訪問。

以下範例政策授予使用者存取所有 Amazon 位置操作的權限。如需更多範例，請參閱Amazon 定 Location Service 的身分識別政策範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "geo:*"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。
- 透過身分提供者在 IAM 中管理的使用者：

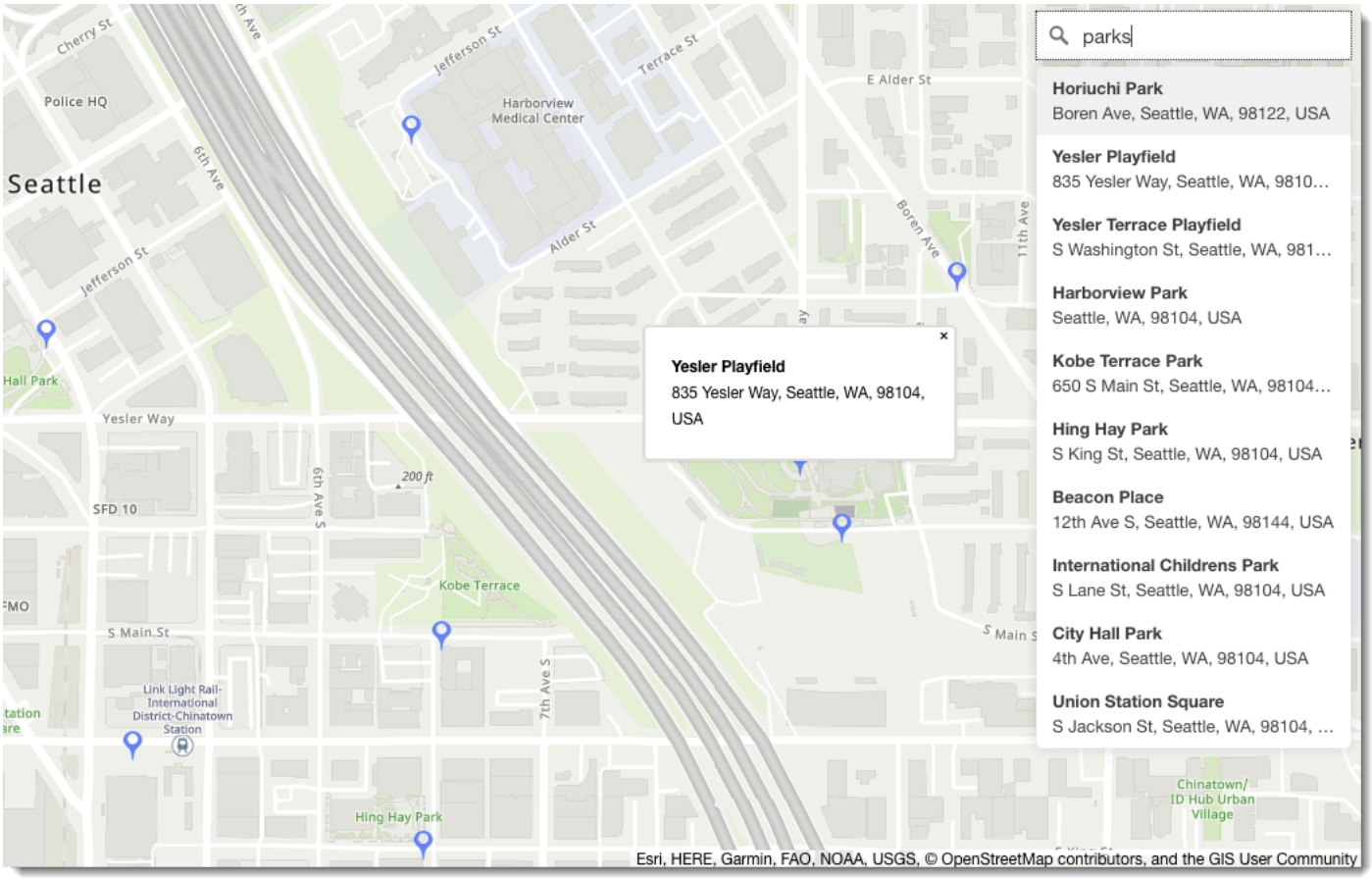
  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。

- (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的新增許可到使用者 (主控台)中的指示。

建立使用 Amazon 定 Location Service 的應用程式時，您可能需要某些使用者擁有未經驗證的存取權。如需這些使用案例，請參閱使用 Amazon Cognito 啟用未驗證存取。

在您的應用程式中使用 Amazon 位置地圖

Amazon 位置地圖具有成本效益且互動。您可以取代應用程式中現有的地圖以節省金錢，或新增地圖以視覺化方式顯示基於位置的資料，例如您的商店位置。



Amazon Location Service 可讓您透過建立和設定地圖資源來選擇地圖操作的資料提供者。map 資源配置數據提供程序和用於渲染地圖的樣式。

建立資源之後，您可以直接使用 AWS SDK 傳送要求，或使用專門用於在環境中呈現地圖的程式庫來傳送要求。

> **Note**
>
> 如需地圖概念的概述，請參閱地圖。

主題

- 必要條件
- 在您的應用程式中顯示地圖
- 在地圖上繪製資料圖徵
- 使用以下方式設定地圖的範圍 MapLibre
- 管理您的地圖資源

必要條件

在應用程式中顯示地圖之前，請遵循先決條件步驟：

主題

- 建立地圖資源
- 驗證您的請求

建立地圖資源

要在應用程序中使用地圖，您必須具有地圖資源，該資源指定要在地圖中使用的地圖樣式和數據提供程序。

> **Note**
>
> 如果您的應用程式正在對業務中使用的資產 (例如送貨車輛或員工) 進行追蹤或路線規劃，則不得使用 Esri 作為地理位置提供者。如需詳細資訊，請參閱 AWS 服務條款第 82 條。

您可以使用 Amazon 定位服務主控台AWS CLI、或 Amazon 位置 API 建立地圖資源。

Console

使用 Amazon 定 Location Service 主控台建立地圖資源

1. 在 Amazon 位置主控台的「地圖」頁面上，選擇「建立地圖」以預覽地圖樣式。
2. 為新地圖資源新增名稱和描述。
3. 選擇地圖樣式。

> **Note**
>
> 如果您的應用程式正在對業務中使用的資產 (例如送貨車輛或員工) 進行追蹤或路線規劃，則不得使用 Esri 作為地理位置提供者。如需詳細資訊，請參閱 AWS 服務條款第 82 條。

4. 從政治觀點要使用的中選擇。
5. 同意 Amazon 位置條款與條件，然後選擇「建立地圖」。您可以與選擇的地圖互動：放大、縮小或向任何方向平移。
6. 若要允許使用者切換樣式 (例如，要允許他們在衛星影像和向量樣式之間切換)，您必須為每個樣式建立地圖資源。

   您可以刪除不想在主控台的 [地圖] 首頁上使用的具有地圖樣式的資源。

API

若要使用 Amazon 位置 API 建立地圖資源

使用來自 Amazon 位置 API 的CreateMap操作。

下列範例是建立*ExampleMap*使用地*VectorEsriStreets*圖樣式呼叫的對應資源的 API 要求。

```
POST /maps/v0/maps HTTP/1.1
Content-type: application/json

{
   "Configuration": {
      "Style": "VectorEsriStreets"
   },
   "MapName": "ExampleMap"
   }
```

```
}
```

> **Note**
>
> 如果您的應用程式正在對業務中使用的資產 (例如送貨車輛或員工) 進行追蹤或路線規劃，則不得使用 Esri 作為地理位置提供者。如需詳細資訊，請參閱 AWS 服務條款第 82 條。

AWS CLI

使用指令建立地圖資源的AWS CLI步驟

使用 `create-map` 命令。

下列範例會建立稱為*ExampleMap*地圖樣式的對映資源。*VectorEsriStreets*

```
aws location \
  create-map \
  --configuration Style="VectorEsriStreets" \
  --map-name "ExampleMap"
```

> **Note**
>
> 如果您的應用程式正在對業務中使用的資產 (例如送貨車輛或員工) 進行追蹤或路線規劃，則不得使用 Esri 作為地理位置提供者。如需詳細資訊，請參閱 AWS 服務條款第 82 條。

驗證您的請求

建立地圖資源並準備好開始在應用程式中建置位置功能後，您需要選擇驗證請求的方式。

> **Note**
>
> 大多數地圖前端應用程式都需要未經驗證存取地圖或 Amazon 定 Location Service 的其他功能。視應用程式而定，您可能想要使用AWS簽名 v4 來驗證請求，或者您可以使用 Amazon Cognito 或 Amazon 位置 API 金鑰進行未經驗證的使用。若要深入瞭解所有這些選項，請參閱授予對 Amazon 定 Location Service 的訪問。

在您的應用程式中顯示地圖

本節提供有關如何在使用 Amazon 位置 API 時使用地圖轉譯工具在行動或 Web 應用程式中顯示地圖的教學課程。如本如何使用 Amazon 定 Location Service主題所述，您可以選擇在使用 Amazon 位置轉譯地圖時使用的程式庫，包括 Amplify MapLibre、和七巧板。

執行下列任一項作業，在應用程式中顯示地圖：

- 在 Web 和移動前端應用程序中顯示地圖的最直接的方法是使用 MapLibre。您可以按照MapLibre 教程甚至快速入門教程學習如何使用 MapLibre。
- 如果您是現有的AWS Amplify開發人員，您可能需要使用 Amplify 地理 SDK。若要進一步了解，請依照 Amplify 教學課程進行。
- 如果您是七巧板的現有使用者，並且想要在移至 Amazon 定 Location Service 時繼續使用它來彩現地圖，請遵循七巧板教學課程。

主題

- 將 MapLibre 程式庫與 Amazon 定 Location Service 搭配使用
- 使用 Amplify 庫與 Amazon 定 Location Service
- 使用七巧板搭配 Amazon Location Service

將 MapLibre 程式庫與 Amazon 定 Location Service 搭配使用

以下教學將逐步引導您完成將程式 MapLibre 庫與 Amazon 位置搭配使用。

主題

- 使用 MapLibre GL JS 與 Amazon Location Service
- 使用適用於 Android 的 MapLibre 原生 SDK 與 Amazon 定 Location Service
- 使用適用於 iOS 的 MapLibre 原生 SDK 與 Amazon 定 Location Service

使用 MapLibre GL JS 與 Amazon Location Service

使用 MapLibre GL JS 將客戶端地圖嵌入到 Web 應用程序中。

MapLibre GL JS 是一個開放原始碼程式 JavaScript 庫，與 Amazon 定 Location Service 地圖 API 提供的樣式和圖塊相容。您可以將 MapLibre GL JS 整合到基本 HTML 或應 JavaScript 用程式中，以嵌入可自訂且回應迅速的用戶端地圖。

本教程介紹如何在基本 HTML 和 JavaScript 應用程序中將 MapLibre GL JS 與 Amazon 位置集成。本教程中介紹的相同庫和技術也適用於框架，例如 React 和 Angular。

本教學課程的範例應用程式可作為上 Amazon 定 Location Service 範例儲存庫的一部分使用GitHub。

構建應用程序：腳手架

本教程創建一個 Web 應用程序， JavaScript 用於在 HTML 頁面上構建地圖。

首先創建一個包含地圖容器的 HTML 頁面（`index.html`）：

- 輸入含有`id`的`div`元素，`map`將地圖的維度套用至地圖檢視。尺寸是從視埠繼承而來的。

```
<html>
  <head>
    <style>
      body {
        margin: 0;
      }

      #map {
        height: 100vh; /* 100% of viewport height */
      }
    </style>
  </head>
  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

構建應用程序：添加依賴關係

將下列依存項目新增至您的應用程式：

- MapLibre GL JS（v3.x 版），及其相關聯的 CSS。
- Amazon 的位置JavaScript 驗證助手。

```
<!-- 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>
```

這將創建一個帶有地圖容器的空白頁面。

構建應用程序：配置

若要使用下 JavaScript列方式來設定應用

1. 輸入資源的名稱和識別碼。

   ```
   // Cognito Identity Pool ID
   const identityPoolId = "us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd";
   // Amazon Location Service Map name
   const mapName = "ExampleMap";
   ```

2. 使用您在使用 map-步驟 2，設定驗證中建立的未驗證身分集區實例化認證提供者。我們將把它放在一個名為的函數中initializeMap，它也將包含其他映射初始化代碼，在下一步中添加

   ```
   // 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
   }
   ```

構建應用程序：映射初始化

要使地圖在頁面加載後顯示，您必須初始化映射。您可以調整初始地圖位置，添加其他控件和覆蓋數據。

```
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**
>
> 您必須在應用程式或文件上為您使用的每個資料提供者提供文字標記或文字歸因。歸因字串包含
> 在`sources.esri.attribution`、`sources.here.attribution`和`sources.grabmaptiles.at`下的樣式描述元回應中。 MapLibre GL JS 將自動提供歸因。搭配資料供應商使用 Amazon 位置資源時，請務必閱讀服務條款與條件。

執行應用程式

您可以在本機 Web 伺服器中使用此範例應用程式，或在瀏覽器中開啟此範例應用程式。

要使用本地 Web 服務器，您可以使用 npx，因為它是作為 Node.js 的一部分安裝的。您可以在與相同的目錄`npx serve`中使用`index.html`。這會在上提供應用程式`localhost:5000`。

> **Note**
>
> 如果您為未經驗證的 Amazon Cognito 角色建立的政策包含`referer`條件，則可能會封鎖您使用 `localhost:` URL 進行測試。在這種情況下，您可以使用提供策略中 URL 的 Web 服務器進行測試。

完成教學課程後，最終的應用程式看起來如下範例所示。

```
<!-- 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>
```

執行此應用程式會使用您選擇的地圖樣式顯示全螢幕地圖。此範例可在上的 Amazon 定 Location Service 範例儲存庫中取得GitHub。

使用適用於 Android 的 MapLibre 原生 SDK 與 Amazon 定 Location Service

使用MapLibre原生 SDK 將交互式地圖嵌入到您的 Android 應用程序中。

Android 版 MapLibre 本的原生 SDK 是一個基於 Mapbox 原生的庫，並且與 Amazon 定 Location Service 地圖 API 提供的樣式和圖塊兼容。您可以集成適用於 Android 的 MapLibre 原生 SDK，以在 Android 應用程序中嵌入交互式地圖視圖以及可擴展的可自定義矢量地圖。

本教程介紹了如何將 Android 的 MapLibre 本機 SDK 與 Amazon 位置集成。本教學課程的範例應用程式可作為上 Amazon 定 Location Service 範例儲存庫的一部分使用GitHub。

構建應用程序：初始化

要初始化您的應用程序：

1. 從空活動模板創建一個新的 Android 工作室項目。
2. 確保為項目語言選擇了 Kotlin。
3. 選擇 API 14：安卓 4.0（冰淇淋三明治）或更高版本的最低開發套件。
4. 打開項目結構，然後轉到文件 > 項目結構... 以選擇「依賴關係」部分。
5. <All Modules>選取後，選擇 + 按鈕以新增新的程式庫相依性。
6. 新增 AWS 安卓開發套件 2.20.0 版或更新版本。例如：`com.amazonaws:aws-android-sdk-core:2.20.0`
7. 添加安卓版本 9.4.0 或更高版本的MapLibre 原生 SDK。例如：`org.maplibre.gl:android-sdk:9.4.0`

8. 在構建 .gradle 文件的項目級別，添加以下 Maven 存儲庫以訪問適用於 Android 的 MapLibre軟件包：

```
allprojects {
    repositories {
        // Retain your existing repositories
        google()
        jcenter()

        //  Declare the repositories for MapLibre
        mavenCentral()
    }
}
```

構建應用程序：配置

若要使用您的資源和AWS區域來設定應用程式：

```
<?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>
```

構建應用程序：活動佈局

編輯app/src/main/res/layout/activity_main.xml：

- 添加一個MapView，用於渲染地圖。這也會設定地圖的初始中心點。
- 添加一個TextView，顯示歸因。

```
<?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**
>
> 您必須在應用程式或文件上為您使用的每個資料提供者提供文字標記或文字歸因。歸因字串包含
> 在`sources.esri.attribution`、`sources.here.attribution`和`source.grabmaptiles.att`
> 下的樣式描述元回應中。搭配資料供應商使用 Amazon 位置資源時，請務必閱讀服務條款與條件。

構建應用程序：請求轉換

建立名為的類別`SigV4Interceptor`以攔截 AWS 請求，並使用簽名版本 4 簽署。創建主要活動時，這將與用於獲取地圖資源的 HTTP 客戶端進行註冊。

```
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()
        }
    }
}
```

構建應用程序：主要活動

主要活動負責初始化將顯示給使用者的檢視。這涉及到：

- 實例化亞 Amazon Cognito `CredentialsProvider`。
- 註冊簽名版本 4 攔截器。
- 通過將地圖指向地圖樣式描述符並顯示適當的歸因來配置地圖。

`MainActivity`也負責將生命週期事件轉寄至地圖檢視，以便在呼叫之間保留作用中的視埠。

```
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()
    }
}
```

執行此應用程式會以您選擇的樣式顯示全螢幕地圖。此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

使用適用於 iOS 的 MapLibre 原生 SDK 與 Amazon 定 Location Service

使用 iOS 版本的MapLibre原生 SDK 將用戶端對應內嵌到 iOS 應用程式中。

iOS 版 MapLibre 本的原生 SDK 是以 Mapbox GL 原生為基礎的程式庫，與 Amazon 定 Location Service 對應 API 提供的樣式和圖塊相容。您可以整合 iOS 版本的 MapLibre 原生 SDK，將互動式地圖檢視與可調整、可自訂的向量地圖嵌入到您的 iOS 應用程式中。

本教程介紹如何將 iOS 的 MapLibre 原生 SDK 與 Amazon 位置集成。本教學課程的範例應用程式可作為上 Amazon 定 Location Service 範例儲存庫的一部分使用GitHub。

構建應用程序：初始化

要初始化您的應用程序：

1. 從應用程序模板創建一個新的 Xcode 項目。
2. 選擇屏幕作為其界面。
3. 為其生命週期選取 SwiftUI 應用程式。
4. 選擇斯威夫特作為它的語言。

使用 Swift 包添加 MapLibre依賴關係

要將包依賴項添加到您的 Xcode 項目中：

1. 導航到文件 > 斯威夫特包 > 添加 Pack age 依賴關係。
2. 輸入儲存庫 URL：**`https://github.com/maplibre/maplibre-gl-native-distribution`**

   > **Note**
   >
   > 有關 Swift 軟件包的更多信息，請參閱在 Apple.com 上向您的應用程序添加軟件包依賴項

3. 在您的終端中，安裝 CocoaPods：

   ```
   sudo gem install cocoapods
   ```

4. 導航到應用程序的項目目錄，並使用軟件 CocoaPods 包管理器初始化 Podfile：

   ```
   pod init
   ```

5. 打開要添加AWSCore為依賴項的 Podfile：

   ```
   platform :ios, '12.0'
   ```

```
target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
end
```

6. 下載並安裝依賴關係：

```
pod install --repo-update
```

7. 打開創建的 Xcode 工 CocoaPods 作區：

```
xed .
```

構建應用程序：配置

將下列機碼和值新增至 Info.plist 以配置應用程式：

金鑰	值
AWSRegion	us-east-1
IdentityPoolId	美國東部-1：54 福爾巴 88-9390-498D
MapName	ExampleMap

構建應用程序： ContentView 佈局

若要彩現地圖，請編輯`ContentView.swift`：

- 添加`MapView`渲染地圖的一個。
- 添加一`TextField`個顯示歸因。

這也會設定地圖的初始中心點。

```
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**
>
> 您必須在應用程式或文件上為您使用的每個資料提供者提供文字標記或文字歸因。歸因字串包含在`sources.esri.attribution`、`sources.here.attribution`和`source.grabmaptiles.att`下的樣式描述元回應中。搭配資料供應商使用 Amazon 位置資源時，請務必閱讀服務條款與條件。

構建應用程序：請求轉換

建立名為的新 Swift 檔案，其中`AWSSignatureV4Delegate.swift`包含下列類別定義，以攔截 AWS 請求，並使用簽名版本 4 簽署。此類別的執行個體會被指派為離線儲存委派，也負責在地圖檢視中重寫 URL。

```
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
    }
}
```

構建應用程序：地圖視圖

[地圖檢視] 負責初始化的執行個體，AWSSignatureV4Delegate並設定基礎執行個體MGLMapView，以擷取資源並呈現對應。它也會處理將歸因字串從樣式描述元的來源傳播回. ContentView

創建一個名為MapView.swift包含以下struct定義的新 Swift 文件：

```
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
    }
}
```

執行此應用程式會以您選擇的樣式顯示全螢幕地圖。此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

使用 Amplify 庫與 Amazon 定 Location Service

以下教學將逐步引導您完成使AWS Amplify用 Amazon 位置的過程。Amplify 使用 MapLibre GL JS 在 JavaScript基於應用程序中渲染地圖。

Amplify 是一組開放原始碼用戶端程式庫，可為不同類別的服務提供介面，包括由 Amazon 定 Location Service 提供支援的 Amplify Geo。 了解有關 AWS Amplify Geo JavaScript 圖書館的更多信息。

> **Note**
>
> 本教學課程假設您已經遵循使用地圖-將地圖加入至您的應用程式中的步驟。

構建應用程序：腳手架

本教程創建一個 Web 應用程序， JavaScript 用於在 HTML 頁面上構建地圖。

首先創建一個包含地圖容器的 HTML 頁面（`index.html`）：

- 輸入含有`id`的`div`元素，`map`將地圖的維度套用至地圖檢視。尺寸是從視埠繼承而來的。

```
<html>
  <head>
    <style>
      body { margin: 0; }
      #map { height: 100vh; } /* 100% of viewport height */
    </style>
  </head>

  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

構建應用程序：添加依賴關係

將下列依存項目新增至您的應用程式：

- AWS Amplify地圖和地理資源庫。
- AWS Amplify核心程式庫。
- AWS Amplify身份驗證庫。
- AWS Amplify樣式表。

```
<!-- 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>
```

這將創建一個帶有地圖容器的空白頁面。

構建應用程序：配置

若要使用下 JavaScript列方式來設定應用

1. 輸入您在使用對應-步驟 2，設定驗證中所建立之未驗證身分集區的識別碼。

   ```
   // 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. 設定AWS Amplify為使用您建立的資源，包括身分識別集區和 Map 資源 (此處顯示為預設名稱`explore.map`)。

   ```
   // 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,
    },
  }
});
```

構建應用程序：映射初始化

要使地圖在頁面加載後顯示，您必須初始化映射。您可以調整初始地圖位置，添加其他控件和覆蓋數據。

```
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**
>
> 您必須在應用程式或文件上為您使用的每個資料提供者提供文字標記或文字歸因。歸因字串包含在`sources.esri.attribution`、`sources.here.attribution`和`sources.grabmaptiles.at`下的樣式描述元回應中。Amplify 會自動提供歸因。搭配資料供應商使用 Amazon 位置資源時，請務必閱讀服務條款與條件。

執行應用程式

您可以在本機 Web 伺服器中使用此範例應用程式，或在瀏覽器中開啟此範例應用程式。

要使用本地 Web 服務器，您可以使用 npx，作為 Node.js 的一部分安裝，或者您選擇的任何其他 Web 服務器。要使用 npx，請在相同的目錄`npx serve`中鍵入。`index.html`這會在上提供應用程式`localhost:5000`。

> **Note**
>
> 如果您為未經驗證的 Amazon Cognito 角色建立的政策包含`referer`條件，則可能會封鎖您使用 `localhost:` URL 進行測試。在這種情況下，您可以使用提供策略中 URL 的 Web 服務器進行測試。

完成教學課程後，最終的應用程式看起來如下範例所示。

```
<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>
```

執行此應用程式會使用您選擇的地圖樣式顯示全螢幕地圖。此範例也會在 Amazon 定 Location Service 主控台中任何地圖資源頁面的「內嵌地圖」索引標籤上進行說明。

完成此自學課程後，請移至AWS Amplify文件中的「顯示地圖」主題以瞭解更多資訊，包括如何在地圖上顯示標記。

使用七巧板搭配 Amazon Location Service

本節提供以下有關如何將七巧板與 Amazon 位置整合的教學課程。

> **Important**
>
> 以下教學課程中的七巧板樣式僅與使用該樣`VectorHereContrast`式設定的 Amazon 位置地圖資源相容。

以下是 AWS CLI 命令的示例，用於創建使*`TangramExampleMap`*用該*`VectorHereContrast`*樣式調用的新映射資源：

```
aws --region us-east-1 \
  location \
  create-map \
  --map-name "TangramExampleMap" \
  --configuration "Style=VectorHereContrast"
```

> **Note**
>
> 帳單取決於您的使用情況。您可能會因使用其他AWS服務而產生費用。如需詳細資訊，請參閱 Amazon 定 Location Service 定價。

主題

- 使用七巧板搭配 Amazon Location Service
- 使用七巧板 ES 安卓系統與 Amazon 定 Location Service
- 使用七巧板 ES 適用於 iOS 與 Amazon 定 Location Service

使用七巧板搭配 Amazon Location Service

七巧板是一個靈活的地圖引擎，設計用於從矢量瓷磚實時渲染 2D 和 3D 地圖。它可以與地圖禪設計的樣式和 Amazon 定 Location Service 地圖 API 提供的 HERE 瓷磚一起使用。本指南說明如何在基本的 HTML/ 應用JavaScript 程式中整合七巧板與 Amazon 位置，雖然在使用 React 和 Angular 等架構時，同樣的程式庫和技術也適用。

七巧板建立在小冊子上，這是一個開放源碼的 JavaScript 圖書館，用於移動友好的互動地圖 這意味著許多與葉子兼容的插件和控件也可以與七巧板一起使用。

使用來自 HERE 的地圖時，為與 Tlezen 架構搭配使用而建立的七巧板樣式與 Amazon 位置大致相容。其中包含：

- 泡沫包裝-一個全功能的尋路風格與興趣點有用的圖標
- 朱砂 — 一般地圖應用的經典外觀和首選
- 筆芯 — 一種簡約的地圖風格，專為數據可視化覆蓋而設計，靈感來自 Stamen Design 的開創性碳粉樣式
- Tron — TRON 視覺語言中規模轉換的探索
- Walkabout — 一種以戶外為中心的風格，非常適合徒步旅行或外出

本指南說明如何使用稱為氣泡紙的七巧板樣式在基本 HTML/ JavaScript 應用程式中整合七巧板與 Amazon 位置。此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

雖然其他七巧板樣式最好配有對地形資訊進行編碼的點陣圖磚，但 Amazon Location 尚不支援此功能。

> **Important**
>
> 以下教學中的七巧板樣式僅與使用該樣`VectorHereContrast`式設定的 Amazon 位置地圖資源相容。

構建應用程序：腳手架

該應用程序是一個 HTML 頁面，用 JavaScript 於在 Web 應用程序上構建地圖。創建一個 HTML 頁面（`index.html`）並創建地圖的容器：

- 輸入具有 map `id` 的`div`元素，以將地圖的維度套用至地圖檢視。
- 尺寸是從視埠繼承而來的。

```
<html>
  <head>
    <style>
      body {
        margin: 0;
      }

      #map {
        height: 100vh; /* 100% of viewport height */
      }
    </style>
  </head>
  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

構建應用程序：添加依賴關係

新增下列相依性：

- 傳單及其相關的 CSS。
- 七巧板。
- AWS 用於 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>
```

這將創建一個具有必要先決條件的空白頁面。下一個步驟會引導您完成編寫應用程式的程式 JavaScript 碼。

構建應用程序：配置

若要使用您的資源和認證來設定應用程式：

1. 輸入資源的名稱和識別碼。

   ```
   // 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. 使用您在使用 map-步驟 2，設定驗證中建立的未驗證身分集區實例化認證提供者。由於這會使用一般 AWS SDK 工作流程以外的登入資料，因此工作階段會在一小時後過期。

   ```
   // 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. 雖然七巧板允許您覆蓋用於獲取圖塊的 URL，但它不包括攔截請求以便簽署請求的功能。

   若要解決此問題，`sources.mapzen.url`請使用綜合主機名稱覆寫指向 Amazon 位置`amazon.location`，該名稱將由服務工作線程處理。以下是使用氣泡包裝的場景配置的示例：

```
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,
    },
  },
};
```

構建應用程序：請求轉換

要註冊和初始化服務工作線程，請在映射初始化之前創建一個要調用的`registerServiceWorker`函數。這將提供的 JavaScript 代碼註冊在一個單獨的文件中，稱`sw.js`為服務工作線程控制`index.html`。

登入資料會從 Amazon Cognito 載入，並與區域一起傳遞至服務工作線程，以提供使用簽名版本 4 簽署磚請求的資訊。

```
/**
 * 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");
  }
```

```
}
```

中的服務工作線程實現sw.js偵聽message事件以獲取憑據和區域配置更改。它還通過監聽fetch事件充當代理服務器。 fetch將重寫以合amazon.location成主機名稱為目標的事件，以指定適當的 Amazon 位置 API，並使用 Amplify 核心簽署。Signer

```
// 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));
});
```

要在憑據到期之前自動更新憑據並將其發送給服務工作線程，請在中使用以下功能index.html：

```
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());
}
```

構建應用程序：映射初始化

要使地圖在頁面加載後顯示，您必須初始化映射。您可以選擇調整初始地圖位置，添加其他控件和覆蓋數據。

> **Note**
>
> 您必須在應用程式或文件上為您使用的每個資料提供者提供文字標記或文字歸因。歸因字串包含
> 在sources.esri.attribution、sources.here.attribution和source.grabmaptiles.att
> 下的樣式描述元回應中。
> 由於七巧板不會要求這些資源，而且只能與 HERE 的地圖相容，因此請使用「© 2020 這裡」。搭配資料供應商使用 Amazon 位置資源時，請務必閱讀服務條款與條件。

```
/**
 * 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();
```

執行應用程式

若要執行此範例，您可以：

- 使用支持 HTTPS 的主機，
- 使用本地 Web 服務器以遵守服務工作線程安全限制。

要使用本地 Web 服務器，您可以使用 npx，因為它是作為 Node.js 的一部分安裝的。您可以在與和相同的目錄`npx serve`中使`index.html`用`sw.js`。這服務於本地主機上的應用程序：5000。

以下是該`index.html`文件：

```
<!-- 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>
```

以下是該sw.js文件：

```
// 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));
});
```

此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

使用七巧板 ES 安卓系統與 Amazon 定 Location Service

七巧板 ES 是一個 C ++ 庫，用於使用 OpenGL ES 從矢量數據渲染 2D 和 3D 地圖。這是七巧板的原生對應物。

使用來自 HERE 的地圖時，為與 Tlezen 架構搭配使用而建立的七巧板樣式與 Amazon 位置大致相容。其中包含：

- 泡沫包裝-一個全功能的尋路風格，帶有興趣點的有用圖標。
- 朱砂 — 一般地圖應用的經典外觀和首選。

- 筆芯 — 一種簡約的地圖風格，專為數據可視化覆蓋而設計，靈感來自 Stamen Design 的開創性碳粉樣式。
- Tron — TRON 視覺語言中規模轉換的探索。
- Walkabout — 一種以戶外為中心的風格，非常適合遠足或外出。

本指南介紹如何使用稱為朱砂的七巧板風格將 Android 版七巧板 ES 與 Amazon 位置集成。此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

雖然其他七巧板樣式最好配有對地形資訊進行編碼的點陣圖磚，但 Amazon Location 尚未支援此功能。

> **Important**
>
> 以下教學中的七巧板樣式僅與使用該樣`VectorHereContrast`式設定的 Amazon 位置地圖資源相容。

構建應用程序：初始化

要初始化您的應用程序：

1. 從空活動模板創建一個新的 Android 工作室項目。
2. 確保為項目語言選擇了 Kotlin。
3. 選擇 API 16：安卓 4.1（果凍豆）或更高版本的最低開發套件。
4. 開啟專案結構以選取檔案、專案結構... ，然後選擇「相依性」區段。
5. 選取後<All Modules>，選擇「+」按鈕以新增「程式庫相依性」。
6. 新增 AWS 安卓開發套件 2.19.1 版或更新版本。例如：`com.amazonaws:aws-android-sdk-core:2.19.1`
7. 加入七巧板版本 0.13.0 或更高版本。例如：`com.mapzen.tangram:tangram:0.13.0`。

> **Note**
>
> 搜尋七巧板:`com.mapzen.tangram:tangram:0.13.0`會產生「找不到」的訊息，但選擇「確定」將允許新增該訊息。

構建應用程序：配置

若要使用您的資源和AWS區域來設定應用程式：

1. 建立 app/src/main/res/values/configuration.xml。
2. 輸入資源的名稱和標識符，以及它們在其中創建的AWS地區：

```
<?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>
```

構建應用程序：活動佈局

編輯app/src/main/res/layout/activity_main.xml：

- 添加一個MapView，用於渲染地圖。這也會設定地圖的初始中心點。
- 添加一個TextView，顯示歸因。

這也會設定地圖的初始中心點。

> **Note**
>
> 您必須在應用程式或文件上為您使用的每個資料提供者提供文字標記或文字歸因。歸因字串包含
> 在sources.esri.attribution、sources.here.attribution和source.grabmaptiles.att
> 下的樣式描述元回應中。
> 由於七巧板不會要求這些資源，而且只能與 HERE 的地圖相容，因此請使用「© 2020 這裡」。搭配資料供應商使用 Amazon 位置資源時，請務必閱讀服務條款與條件。

```
<?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>
```

構建應用程序：請求轉換

創建一個名SigV4Interceptor為攔截AWS請求並使用簽名版本 4 簽名它們的類。創建主要活動時，這將與用於獲取地圖資源的 HTTP 客戶端進行註冊。

```
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()
        }
    }
}
```

構建應用程序：主要活動

主要活動負責初始化將顯示給使用者的檢視。這涉及到：

- 實例化亞 Amazon Cognito `CredentialsProvider`。
- 註冊簽名版本 4 攔截器。
- 透過將地圖指向地圖樣式、覆寫並排 URL，以及顯示適當的歸因來設定地圖。

`MainActivity`也負責將生命週期事件轉寄至地圖檢視。

```
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()
    }
}
```

執行此應用程式會以您選擇的樣式顯示全螢幕地圖。此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

使用七巧板 ES 適用於 iOS 與 Amazon 定 Location Service

七巧板 ES 是一個 C ++ 庫，用於使用 OpenGL ES 從矢量數據渲染 2D 和 3D 地圖。這是七巧板的原生對應物。

使用來自 HERE 的地圖時，為與 Tlezen 架構搭配使用而建立的七巧板樣式與 Amazon 位置大致相容。其中包含：

- 泡沫包裝-一個全功能的尋路風格與興趣點有用的圖標
- 朱砂 — 一般地圖應用的經典外觀和首選
- 筆芯 — 一種簡約的地圖風格，專為數據可視化覆蓋而設計，靈感來自 Stamen Design 的開創性碳粉樣式

- Tron — TRON 視覺語言中規模轉換的探索
- Walkabout — 一種以戶外為中心的風格，非常適合徒步旅行或外出

本指南說明如何使用稱為朱砂的七巧板風格整合 iOS 版的七巧板 ES 與 Amazon 位置。此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

雖然其他七巧板樣式最好配有對地形資訊進行編碼的點陣圖磚，但 Amazon Location 尚未支援此功能。

> **Important**
>
> 以下教學中的七巧板樣式僅與使用該樣`VectorHereContrast`式設定的 Amazon 位置地圖資源相容。

構建應用程序：初始化

若要初始化應用程式：

1. 從應用程序模板創建一個新的 Xcode 項目。
2. 選擇屏幕作為其界面。
3. 為其生命週期選取 SwiftUI 應用程式。
4. 選擇斯威夫特作為它的語言。

構建應用程序：添加依賴關係

要添加依賴關係，您可以使用依賴關係管理器，例如 CocoaPods：

1. 在您的終端中，安裝 CocoaPods：

   ```
   sudo gem install cocoapods
   ```

2. 導航到應用程序的項目目錄，並使用軟件 CocoaPods包管理器初始化 Podfile：

   ```
   pod init
   ```

3. 打開要添加的 Podfile `AWSCore` 並`Tangram-es`作為依賴關係：

   ```
   platform :ios, '12.0'
   ```

```
target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
  pod 'Tangram-es'
end
```

4. 下載並安裝依賴關係：

```
pod install --repo-update
```

5. 打開創建的 Xcode 工 CocoaPods 作區：

```
xed .
```

構建應用程序：配置

將下列機碼和值新增至 Info.plist 以設定應用程式並停用遙測：

金鑰	值
AWSRegion	us-east-1
IdentityPoolId	美國東部-1：54 福爾巴 88-9390-498D
MapName	ExampleMap
場景網址	https://www.nextzen.org/carto/cinnabar-style/9/cinnabar-style.zip

構建應用程序： ContentView 佈局

若要彩現地圖，請編輯`ContentView.swift`：

- 添加`MapView`渲染地圖的一個。
- 添加一`TextField`個顯示歸因。

這也會設定地圖的初始中心點。

> **Note**
>
> 您必須在應用程式或文件上為您使用的每個資料提供者提供文字標記或文字歸因。歸因字串包含在`sources.esri.attribution`、`sources.here.attribution`和`source.grabmaptiles.attribution`下的樣式描述元回應中。搭配資料供應商使用 Amazon 位置資源時，請務必閱讀服務條款與條件。

```
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()
    }
}
```

構建應用程序：請求轉換

創建一個名為AWSSignatureV4URLHandler.swift包含以下類定義的新 Swift 文件，以攔截AWS請求並使用簽名版本 4 對其進行簽名。這將在七巧板MapView中註冊為 URL 處理程序。

```
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)
    }
}
```

構建應用程序：地圖視圖

地圖檢視負責初始化的執行個體，AWSSignatureV4Delegate並設定基礎MGLMapView，以擷取資源並呈現對應。它也會處理將歸因字串從樣式描述元的來源傳播回. ContentView

創建一個名為MapView.swift包含以下struct定義的新 Swift 文件：

```
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) {
    }
}
```

執行此應用程式會以您選擇的樣式顯示全螢幕地圖。此範例可作為上 Amazon 定 Location Service 範例儲存庫的一部分提供GitHub。

在地圖上繪製資料圖徵

使用「Amplify」、或「七巧板」彩現地圖的應用程式來彩現地圖後，下一個自然的步驟就是在地圖頂部繪製圖徵。 MapLibre例如，您可能想要將客戶位置呈現為地圖上的標記。

一般而言，您可以使用「位置」搜尋功能從資料中尋找位置，然後使用「Amplify」 MapLibre、或「七巧板」的功能來彩現位置。

若要查看在地圖上彩現不同類型物件的範例，請參閱以下 MapLibre 範例：

- 範例：繪製標記
- 範例：繪製叢集點
- 範例：繪製多邊形

如需更多範例和自學課程，請參閱使用 Amazon 定 Location Service 的程式碼範例和教學課程。

使用以下方式設定地圖的範圍 MapLibre

有時候，您不希望您的用戶能夠在整個世界上平移或縮放。如果您使用的 MapLibre是地圖控制項，您可以使用選項限制地圖控制maxBounds項的範圍或邊界，並使用minZoom和maxZoom選項限制縮放。

下列程式碼範例顯示如何初始化地圖控制項，以限制平移至特定邊界 (在本例中為 Grab 資料來源的範圍)。

> **Note**
>
> 這些範例位於自學課程的內容中 JavaScript，並在建立網路應用程式自學課程的上下文中工作。

```
// 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,
    }
);
```

同樣，您可以為地圖設置最小和最大縮放級別。兩者的值都可以介於 0 和 24 之間，儘管最小縮放的預設值為 0，最大值為 22（資料提供者可能不會在所有縮放層級提供資料。 大多數地圖庫會自動處理此問題）。下列範例會初始化 MapLibre 地圖控制maxZoom項上的minZoom和選項。

```
// Set the minimum and maximum zoom levels
var mlglMap = new maplibregl.Map(
    {
    container: 'map',
    style: mapName,
    maxZoom: 12,
    minZoom: 5,
    transformRequest,
    }
);
```

> **Tip**
>
> MapLibre Map 控件還允許在運行時設置這些選項，而不是在初始化過程中，與get...和set...功能。例如，使用getMaxBounds和setMaxBounds來在執行階段變更對應邊界。

管理您的地圖資源

您可以使用 Amazon 位置主控台AWS CLI、或 Amazon 位置 API 來管理地圖資源。

列出地圖資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 來檢AWS CLI視地圖資源清單。

Console

使用 Amazon 位置主控台檢視現有地圖資源的清單

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇 [地圖]。
3. 在「我的地圖」下檢視您的地圖資源清單。

API

使用來自 Amazon 位置地圖 API 的ListMaps操作。

下列範例是取得AWS帳戶中對映資源清單的 API 要求。

```
POST /maps/v0/list-maps
```

以下為範例回應ListMaps：

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-30T01:38:36Z,
         "DataSource": "Esri",
         "Description": "string",
         "MapName": "ExampleMap",
         "UpdateTime": 2020-10-30T01:38:36Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

使用 list-map 命令。

下列範例是取AWS CLI得AWS帳戶中對映資源清單的範例。

```
aws location list-maps
```

獲取地圖資源的詳細

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，取得 AWS 帳戶中任何地圖資源的詳細資訊。AWS CLI

Console

使用 Amazon 位置主控台檢視地圖資源的詳細資訊

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇 [地圖]。
3. 在我的地圖下，選擇目標地圖資源的名稱鏈接。

API

使用來自 Amazon 位置地圖 API 的DescribeMap操作。

下列範例是取得其對映資源詳細資訊的 API 要求*ExampleMap*。

```
GET /maps/v0/maps/ExampleMap
```

以下為範例回應DescribeMap：

```
{
   "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

使用 describe-map 命令。

下面的例子是一個AWS CLI獲取的映射資源詳細信息*ExampleMap*。

```
aws location describe-map \
    --map-name "ExampleMap"
```

刪除地圖資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 從AWS帳戶刪除地圖資源。AWS CLI

> **Warning**
>
> 此作業會永久刪除資源。

Console

使用 Amazon 位置主控台刪除現有的地圖資源

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇 [地圖]。
3. 在「我的地圖清單」下，從清單中選取目標地圖。
4. 選擇「刪除地圖」。

API

使用來自 Amazon 位置地圖 API 的DeleteMap操作。

以下示例是刪除地圖資源的 API 請求*ExampleMap*。

```
DELETE /maps/v0/maps/ExampleMap
```

以下是成功回應的範例DeleteMap：

```
HTTP/1.1 200
```

CLI

使用 delete-map 命令。

下面的例子是刪除地圖資源的AWS CLI命令*ExampleMap*。

```
aws location delete-map \
    --map-name "ExampleMap"
```

使用 Amazon 位置搜尋地點和地理位置資料

Amazon 位置包括搜尋所選供應商的地理位置或位置資料的功能。有幾種類型的搜索可用。

- 地理編碼 — 地理編碼是根據文本輸入搜索地址，地區，商業名稱或其他興趣點的過程。它返回詳細信息和找到的結果的位置（以緯度和經度為單位）。
- 反向地理編碼 — 反向地理編碼允許您查找給定位置附近的位置。
- 自動完成 — 自動完成是使用者在查詢中輸入自動建議的過程。例如，如果他們鍵入**Par**一個建議可能是`Paris, France`。

Amazon Location 可讓您透過建立和設定位置索引資源，選擇用於地點搜尋操作的資料提供者。

建立資源後，您可以使用 AWS SDK 針對您慣用的語言、Amplify 或 REST API 端點傳送請求。您可以使用來自回應的資料在地圖上標記位置、豐富位置資料，以及將位置轉換為人類可讀的文字。

> **Note**
>
> 如需搜尋地點概念的概述，請參閱地點搜尋。

主題

- 必要條件
- 地理编码
- 反向地理編碼
- 自動完成
- 使用地點 ID
- 放置類別和過濾結果
- Amazon Aurora PostgreSQLAmazon 定 Location Service 的使用者定義函數
- 管理您的地方索引資源

必要條件

在開始進行地理編碼、反向地理編碼或搜尋地點之前，請遵循先決步驟：

主題

- 建立位置索引資源
- 驗證您的請求

建立位置索引資源

首先在您的 AWS 帳號中建立位置索引資源。

建立地點索引資源時，您可以從可用於支援地理編碼、反向地理編碼和搜尋查詢的資料提供者中進行選擇：

1. Esri — 有關 Esri 在您感興趣的地區覆蓋範圍的更多信息，請參閱 Esri 文檔中的 Esri 地理編碼覆蓋範圍。
2. HERE 技術 — 如需您感興趣地區 HERE 涵蓋範圍的詳細資訊，請參閱 HERE 文件中的 HERE 地理編碼涵蓋範圍。
3. 抓取 — Grab 僅為東南亞提供資料。有關 Grab 覆蓋範圍的更多信息，請參閱本指南所涵蓋的國家 / 地區中的。

您可以使用 Amazon 定 Location Service 主控台 AWS CLI、或 Amazon 位置 API 來執行此操作。

Console

使用 Amazon 定位服務主控台建立位置索引資源

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 定 Location Service 控制台。
2. 在左側導覽窗格中，選擇 [置入索引]。
3. 選擇 [建立地點索引]。
4. 填寫下列方塊：
   - 名稱 — 輸入位置索引資源的名稱。例如，*ExamplePlaceIndex*. 最多 100 個字元。有效的項目包括英數字元、連字號、句號和底線。
   - 說明 — 輸入選擇性說明。
5. 在「資料提供者」下，選擇可用的資料提供者，以與您的 place 索引資源搭配使用。

> **Note**
>
> 如果您的應用程式正在對業務中使用的資產 (例如送貨車輛或員工) 進行追蹤或路線規劃，則不得使用 Esri 作為地理位置提供者。如需詳細資訊，請參閱 AWS 服務條款第 82 條。

6. 在「資料儲存選項」下，指定是否要儲存位置索引資源的搜尋結果。
7. (選用) 在 Tags (標籤) 底下，輸入標籤 Key (金鑰) 與 Value (值)。這將添加一個標籤您的新位置索引資源。如需詳細資訊，請參閱標記您的資源。
8. 選擇 [建立地點索引]。

API

若要使用 Amazon 位置 API 建立位置索引資源

使用來自 Amazon 位置位置 API 的CreatePlaceIndex操作。

下列範例是建立*ExamplePlaceIndex*使用資料提供者 *Esri* 呼叫的位置索引資源的 API 要求。

```
POST /places/v0/indexes
Content-type: application/json

{
   "DataSource": "Esri",
   "DataSourceConfiguration": {
      "IntendedUse": "SingleUse"
   },
   "Description": "string",
   "IndexName": "ExamplePlaceIndex",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

使用指令建立位置索引資源的 AWS CLI 步驟

使用 create-place-index 命令。

下列範例會建立*ExamplePlaceIndex*使用 *Esri* 做為資料提供者呼叫的位置索引資源。

```
aws location \
  create-place-index \
  --data-source "Esri" \
  --description "Example place index" \
  --index-name "ExamplePlaceIndex" \
  --tags Tag1=Value1
```

> **Note**
>
> 帳單取決於您的使用情況。您可能會因使用其他 AWS 服務而產生費用。如需詳細資訊，請參閱 Amazon 定 Location Service 定價。

驗證您的請求

建立地點索引資源並準備好開始在應用程式中建置位置功能之後，請選擇驗證請求的方式：

- 若要探索存取服務的方式，請參閱存取 Amazon 定 Location Service。
- 如果您有一個擁有匿名用戶的網站，則可能需要使用 API 密鑰或 Amazon Cognito。

  範例

  下列範例顯示如何使用 API 金鑰進行授權、使用 JavaScriptAWS 開發套件 v3 和 Amazon 位置JavaScript 驗證助手。

```
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);
```

地理编码

地理編碼是將文字 (例如地址、地區、商家名稱或興趣點) 轉換為一組地理座標的程序。您可以使用地點索引資源來提交地理編碼請求，並合併從地理編碼擷取的資料，以便在 Web 或行動應用程式的地圖上顯示資料。

本節將引導您如何傳送簡單的地理編碼要求，以及如何使用選用規格傳送地理編碼請求。

地理编码

您可以使用將地址轉換為一組坐標的SearchPlaceIndexForText操作提交一個簡單的地理編碼請求。一個簡單的請求包含以下必要的參數：

- Text— 要轉換為一組座標的地址、名稱、城市或地區。例如，字串Any Town。

若要指定每頁的結果數目上限，請使用下列選用參數：

- MaxResults— 限制查詢回應中傳回的結果數目上限。

您可以使用 AWS CLI 或 Amazon 位置 API。

API

下列範例是搜尋地點索引資源的SearchPlaceIndexForText請求 *ExamplePlaceIndex*，以取得名為「*####*」的地址、名稱、城市或地區。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
```

```
Content-type: application/json

{
    "Text": "Any Town",
    "MaxResults": 10
}
```

AWS CLI

下列範例是用來搜尋地點索引資源的指search-place-index-for-text令*ExamplePlaceIndex*，以尋找名為 Any *T* own 的地址、名稱、城市或區域。

```
aws location \
    search-place-index-for-text \
        --index-name ExamplePlaceIndex \
        --text "Any Town" \
        --max-results 10
```

位置附近的地理編碼

地理編碼時，您可以使用以下可選參數在給定位置附近進行地理編碼：

- BiasPosition— 您想要在附近搜索的位置。這會搜尋最接近指定位置的結果，藉此縮小搜尋範圍。定義為 [longitude, latitude]

######## [-123.4567, 45.6789] ################################
#SearchPlaceIndexForText###

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "BiasPosition": [-123.4567,45.6789]
}
```

邊界框內的地理編碼

您可以使用下列可選參數，在邊界框內進行地理編碼，將結果縮小為指定邊界內的座標：

- `FilterBBox`— 您指定用來將結果篩選為方塊邊界內座標的邊界方塊。定義為 `[LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]`

> **Note**
>
> 請求不能同時包含`FilterBBox`和`BiasPosition`參數。在要求中指定這兩個參數會傳回`ValidationException`錯誤。

下列範例是SearchPlaceIndexForText要求在邊界方塊內搜尋稱為「*####*」的地址、名稱、城市或區域。邊界框如下所示：

- 西南角的經度是 *-12* 4.1450。
- 西南角落的緯度為 *41.7045*。
- 東北角的經度為 *-12* 4.1387。
- 東北角的緯度為 *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
    ]
}
```

一個國家/地區內的地理編碼

您可以使用下列選用參數，在指定的一或多個國家/地區內進行地理編碼：

- `FilterCountries`— 您要在其中進行地理編碼的國家或地區。您可以使用 ISO 3166 三個字母的國家/地區代碼，在一個請求中定義多達 100 個國家/地區。例如，用AUS於澳大利亞。

下列範例是SearchPlaceIndexForText要求搜尋位址、名稱、城市或地區 (位於德國和法國的 A *ny Town*)。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "FilterCountries": ["DEU","FRA"]
}
```

按類別過濾

您可以使用下列選用參數來篩選地理編碼要求中傳回的類別：

- `FilterCategories`— 您要在查詢中傳回的結果類別。您最多可以在一個請求中指定 5 個類別。您可以在類別部分找到 Amazon 定 Location Service 類別列表。例如，您可以指定`Hotel`在查詢中僅指定返回的酒店。

以下範例是在美國搜尋名為「`########`店的`SearchPlaceIndexForText`請求。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Hometown Coffee",
   "FilterCategories": ["Coffee Shop"],
   "FilterCountries": ["USA"]
}
```

如需篩選類別的詳細資訊，請參閱 放置類別和過濾結果

偏好語言的地理編碼

您可以使用選用參數來設定搜尋結果的語言偏好設`Language`定。例如，`100 Main St, Any Town, USA`依預設，**`100 Main St, Anytown, USA`**可能會傳回搜尋。但是，如果您選擇`fr`為`Language`，則結果可能會`100 Rue Principale, Any Town, États-Unis`返回。

- `Language`— 用於轉譯查詢結果的語言代碼。此值必須是有效的 BCP 47 語言代碼。例如，`en`對於英文。

> **Note**
>
> 如果Language未指定，或結果不支援指定的語言，則會使用該結果的合作夥伴預設語言。

下列範例是SearchPlaceIndexforText要求以慣**Any Town**用語言指定為來搜尋呼叫的地點de。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json
{
   "Text": "Any Town",
   "Language": "de"
}
```

回應範例

Example

以下是當您從 Amazon 位置位置 API 呼叫SearchPlaceIndexForText作業時的範例回應。結果包括相關的地方和要求摘要。根據選擇 Esri 或 HERE 作為合作夥伴，會顯示兩個回應。

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
        }
    ]
}
```

反向地理編碼

反向地理編碼是一種將一組坐標轉換為有意義的文本的過程，例如地址，地區，商業名稱或興趣點。您可以使用 place index 資源來提交反向地理編碼請求，並合併從反向地理編碼擷取的資料，以便在 Web 或行動應用程式的地圖上顯示資料。

本節將引導您如何傳送簡單的反向地理編碼要求。

反向地理編碼

您可以提交一個簡單的請求，以反向地理編碼一組坐標，並使用該SearchPlaceIndexForPosition操作將其轉換為有意義的地址，興趣點或沒有地址的一般位置。一個簡單的請求包含以下必要的參數：

- `Position`— 您要轉換為地址、興趣點或一般位置的一組座標。使用格式定義`[longitude,latitude]`。

若要指定每頁的結果數目上限，請新增下列選用參數：

- `MaxResults`— 限制查詢回應中傳回的結果數目上限。

如果您要為查詢結果指定偏好的語言，請使用下列選用參數：

- `Language`— 用於呈現結果的語言代碼。此值必須是有效的 BCP 47 語言代碼。例如，en對於英文。

> **Note**
>
> 如果`Language`未指定，或結果不支援指定的語言，則會使用該結果的合作夥伴預設語言。

您可以使用 AWS CLI 或 Amazon 位置 API。

API

##############SearchPlaceIndexForPosition## ExamplePlaceIndex########## ############ [122.3394, 47.6159]#

```
POST /places/v0/indexes/ExamplePlaceIndex/search/position
```

```
Content-type: application/json

{
   "Position": [-122.3394,47.6159],
   "MaxResults": 5,
   "Language": "de"
}
```

AWS CLI

#######search-place-index-for-position############ ExamplePlaceIndex#### ################## [122.3394, 47.6159]#

```
aws location \
    search-place-index-for-position \
        --index-name ExamplePlaceIndex \
        --position -122.3394 47.6159 \
        --max-results 5 \
        --language de
```

回應範例

Example

以下是從 Amazon 位置位置 API 呼叫SearchPlaceIndexForPosition作業時的範例回應。結果會傳回相關位置和要求摘要。根據選擇 Esri 或 Here 作為合作夥伴，會顯示兩個回應。

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
        }
    ]
}
```

自動完成

自動完成提供響應反饋給最終用戶，因為他們正在鍵入他們的搜索查詢。它會根據部分或拼錯的自由格式文字，提供地址和興趣點的建議。您可以使用 place 索引資源來請求自動完成建議，並在應用程序中顯示結果建議。

Amazon 位置不支援儲存自動完成建議。如果配置用於自動完成調用的地點索引與存儲的地理編碼一起使用，則返回錯誤。要使用存儲的地理編碼並查詢建議，請創建和配置多個位置索引。

本節說明如何傳送自動完成要求。它從請求的最基本形式開始，然後顯示可選參數，您可以使用這些參數來增加自動完成搜索結果的相關性。

使用自動完成

您可以使用SearchPlaceIndexForSuggestions操作來提交自動完成建議的簡單請求。最簡單的請求形式有一個必需的參數，查詢Text：

- Text— 用於產生地點建議的自由格式部分文字。例如，字串eiffel tow。

若要限制傳回的結果數目，請新增選用MaxResults參數：

- MaxResults— 限制查詢回應中傳回的結果數。

您可以使用 Amazon 定位 API 或 AWS CLI.

API

下列範例是搜尋地點索引資源的SearchPlaceIndexForSuggestions請求 *ExamplePlaceIndex*，根據部分地點名稱 *kamp* 提供最多 *5* 個建議。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "MaxResults": 5
}
```

AWS CLI

下列範例是用來搜尋地點索引資源的search-place-index-for-suggestions命令 *ExamplePlaceIndex*，最多可根據部分地名 *kamp* 提供 *5* 個建議。

```
aws location \
              search-place-index-for-suggestions \
              --index-name ExamplePlaceIndex \
              --text kamp \
              --max-results 5
```

呼叫會SearchPlaceIndexForSuggestions產生具有名稱和每個 ID 的位置清單。您可以使用這些結果來呈現使用者可能正在搜尋的內容的建議，例如在文字方塊下方提供選項的下拉式清單。例如，以下是根據使用者輸入 *kamp* 的建議結果。


```
{
    "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
        }
    ]
}
```

下一節將說明如何使用這些結果PlaceID中的。

使用自動完成結果

呼叫會SearchPlaceIndexForSuggestions產生具有名稱和每個 ID 的位置清單。您可以使用這些結果來呈現使用者可能正在搜尋的內容的建議，例如在文字方塊下方提供選項的下拉式清單。當用戶選擇其中一個結果時，您可以使用其選擇的 ID 調用GetPlace操作以返回該地點的詳細信息，包括位置，地址或其他詳細信息。

> **Note**
>
> 只PlaceId有在原始搜尋請求中的下列所有內容都相同且呼叫時，A 才有效GetPlace。
>
> - 顧客 AWS 帳戶
> - AWS 區域
> - 在 place 索引資源中指定的數據提供者

通常，您可以GetPlace搭配 Amazon 位置 API 使用。下列範例是尋找上一節建議之一的GetPlace要求。這個例子是基於部分地名 *kamp*。

```
POST /places/v0/indexes/ExamplePlaceIndex/
places/AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

位置附近的自動完成

當您使用搜索自動完成地點建議時SearchPlaceIndexForSuggestions，可以通過添加以下可選參數來獲得更多與本地相關的建議：

- BiasPosition— 您想要在附近搜索的位置。定義為[longitude, latitude]。

#######SearchPlaceIndexForSuggestions############ExamplePlaceIndex######## ### kamp ####### [32.5827, 0.3169]#

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json
```

```
{
    "Text": "kamp",
    "BiasPosition": [32.5827,0.3169]
}
```

################Text###BiasPosition####### [-96.7977#32.7776]#

在邊界框內自動完成

您可以透過新增下列選用參數來縮小自動完成搜尋範圍，以僅接收位於指定邊界內之位置的建議：

- `FilterBBox`— 您指定用來將結果篩選為方塊邊界內座標的邊界方塊。定義為 `[LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]`

> **Note**
>
> 請求不能同時包含`FilterBBox`和`BiasPosition`參數。在要求中指定這兩個參數會傳回`ValidationException`錯誤。

下列範例會使用SearchPlaceIndexForSuggestions要求來搜尋地點索引資源，*ExamplePlaceIndex*尋找符合部分查詢 *kamp* 的地點建議，而這些建議包含在邊界方框中：

- 邊界框的西南角的經度為 *32.5020*。
- 邊界框的西南角點緯度為 *0.26* 78。
- 邊界框的東北角的經度為 *32.6* 129。
- 邊界框的東北角的緯度為 *0.350* 2。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterBBox": [
        32.5020, 0.2678,
        32.6129, 0.3502
    ]
}
```

##################Text###FilterBBox#### [-97.9651#32.0640#-95.1 196#34.0436]#

在一個國家/地區內自動完成

您可以透過新增下列選用參數來縮小自動完成搜尋範圍，以僅接收位於指定國家或國家/地區集內的地點的建議：

- FilterCountries— 你想搜索的地方建議內的國家. 您可以使用 ISO 3166 三個字母的國家/地區代碼，在一個請求中指定多達 100 個國家/地區。例如，用AUS於澳大利亞。

下列範例會使用SearchPlaceIndexForSuggestions要求來搜尋地點索引資源，*ExamplePlaceIndex*尋找符合部分查詢 *kamp* 且包含在烏干達、肯亞或坦桑尼亞的地點建議：

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterCountries": ["UGA", "KEN", "TZA"]
}
```

如果選擇了不同Text的FilterCountries列表，則返回相同的建議會有所不同，例如 [」*US* A「]。

回應範例

####### kamp #SearchPlaceIndexForSuggestions##############

```
{
    "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
        }
    ]
}
```


使用地點 ID

搜尋地點會傳回結果清單。大多數結果包PlaceId括該結果。您可以在GetPlace作業PlaceId中使用 a 來傳回有關該位置的資訊 (包括名稱、地址、位置或其他詳細資訊)。

> **Note**
>
> 使用SearchPlaceIndexForSuggestions將傳回使用任何資料來源建立的任何位置索引的PlaceId結果。PlaceId僅當使SearchPlaceIndexForText用SearchPlaceIndexForPosition的資料來源為 HERE 時，使用或才會傳回。

每個PlaceId唯一定義它所指的地方，但是PlaceId隨著時間的推移，並且基於上下文，單個地方可以有多個地方。以下規則描述了 a 的唯一性和壽命PlaceId。

- 您所進行的呼叫中PlaceId傳回的資料特定於您 AWS 帳戶、對該 AWS 地區以及PlaceIndex資源中的資料提供者。 GetPlace只有當這三個屬性符合建立的原始呼叫時，才會尋找結果PlaceId。
- 當有PlaceId關該地點的數據發生變化時，該地點的數據將發生變化。例如，當其參照的商家移動位置或變更名稱時。
- 重複搜尋呼叫PlaceId傳回的可能會在後端服務進行更新時變更。PlaceId將繼續找到較舊的，但是要搜索的新呼叫可能會返回不同的 ID。

PlaceId是一個字符串。a 的長度沒有特定限制PlaceId。以下是有效的範例PlaceId。

```
AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

GetPlace使PlaceId用資料已變更的地點呼叫 (例如，已停業的營業地點)，將會導致ResourceNotFound錯誤。404GetPlace使用無效的調用，或者一個不在上下文中（例如來自另 AWS 帳戶一個）將返回一400個ValidationException錯誤。PlaceId

雖然您可以在後續請求中使用 PlaceId，但 PlaceId 並不打算成為永久標識符，並且 ID 可以在連續的 API 調用之間更改。請參閱每個資料提供者的下列 PlaceId 行為：

- Esri：地點 ID 將至少每季度更改一次。這些變更的典型時段為三月、六月、九月和十二月。地點 ID 也可能會在典型的季度變更之間發生變化，但這樣做的頻率會降低得多。
- HER E：我們建議您快取資料的時間不超過一週，以保持資料資料的最新狀態。您可以假設少於 1% 的 ID 偏移將在發行版本中釋放，每週約 1-2 次。
- 抓取：在下列情況下，地點 ID 可能會過期或變成無效。
  - 數據操作：根據地面真相，例如在現實世界中關閉，被檢測為重複的 POI 或具有不正確的信息，可以通過 Grab Map Ops 從 Grab POI 數據庫中刪除 POI。Grab 將每週將數據同步到航點環境。
  - 內插 POI：內插 POI 是提供請求時實時生成的臨時 POI，並且在響應中將其標記為派生。place.result_type內插的 POI 資料將保留至少 30 天，這意味著您可以在 30 天內從地點詳細信息 API 獲取地點 ID 的 POI 詳細信息。30 天後，內插的 POI（地點 ID 和詳細信息）可能會過期且無法從「地點詳細信息」API 訪問。

放置類別和過濾結果

地方被分類。例如，如果您搜尋商家，則該商家可能是一Restaurant家。即使是搜尋地址的結果，也可以依據地址是否與地址、街道或交叉點相符來分類。

一般而言，Amazon 定 Location Service 會將地點分類為「地點」類型。興趣點會進一步分類為興趣點類型。

> **Note**
>
> 並非所有結果都會有類別。

您可以使用類別來篩選地理編碼搜尋。

篩選結果

使用時`SearchPlaceIndexForText`，您可以篩選要使用的類別傳回的結果。例如：

- 如果您想要搜尋名為「家鄉咖啡」的地方，並且只傳回分類為咖啡店的結果，您可以透過呼叫`SearchPlaceIndexForText`並`Coffee Shop`在`FilterCategories`參數中加入「興趣點」類別來執行此操作。
- 搜尋「123 Main St, Anytown, WA, 98123, USA」時，您可以將結果篩選為只有地址，這樣您就不會看到相符項目，例如郵遞區號。透過在`FilterCategories`參數`AddressType`中包含 P lace 類型來篩選至位址。

> **Note**
>
> 並非所有資料提供者都支援篩選，或以相同的方式支援。如需詳細資訊，請參閱 資料提供者篩選限制。

下一節會列出您可以篩選的類別。

類別

下列清單顯示 Amazon 定 Location Service 用於分類和篩選的類別。這些類別用於所有語言，獨立於語言參數設置為不同的語言。

> **Note**
>
> Amazon 定 Location Service 會將資料提供者類別對應到這組類別。如果資料提供者將位置放在不屬於 Amazon 定 Location Service 類別清單的類別中，則提供者類別將作為補充類別包含在結果中。

放置類型 — 這些類型用於指示用於尋找結果的相符類型。

- `AddressType`— 當結果與位址相符時傳回。
- `StreetType`— 當結果與街道匹配時返回。
- `IntersectionType`— 當結果與兩條街道的交叉點相符時傳回。
- `PointOfInterestType`— 當結果與興趣點 (例如商業或公民地點) 相符時傳回。
- `CountryType`— 當結果與國家或主要地區相符時傳回。
- `RegionType`— 當結果與國家/地區 (例如州或省) 內的區域相符時傳回。
- `SubRegionType`— 當結果與國家/地區（例如縣或都市區）內的附屬區域相符時傳回。
- `MunicipalityType`— 當結果與城市或城鎮匹配時返回。
- `NeighborhoodType`— 當結果與城市內的鄰里或區域匹配時返回。
- `PostalCodeType`— 當結果與郵遞區號相符時傳回。

興趣點類別 — 這些類別用於指示興趣點結果的業務類型或地點。

- `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

資料提供者篩選限制

並非所有提供者都具有相同的篩選功能。下表說明差異。

供應商	具有篩選器支援的 API	支援篩選的類別	傳回值
埃斯里	SearchPlaceIndexForText ，SearchPlaceIndexForSuggestions	依地點類型和興趣點類別進行篩選。	類別由SearchPlaceIndexForText SearchPlaceIndexForPosition 、和傳回 GetPlace

供應商	具有篩選器支援的 API	支援篩選的類別	傳回值
這裡	`SearchPlaceIndexForText` ，`SearchPlaceIndexForSuggestions`	僅依「放置」類型篩選。	類別由`SearchPlaceIndexForText` 和`SearchPlaceIndexForSuggestions` `SearchPlaceIndexForPosition` 、和傳回 `GetPlace`
抓斗	不支援	不支援	不支援
開啟資料	n/a (不支援搜尋地點)	N/A	無

Amazon Aurora PostgreSQLAmazon 定 Location Service 的使用者定義函數

您可以使用 Amazon 定 Location Service 來處理儲存在資料庫表格中的座標和地址，以清理和豐富您的地理空間資料。

例如：

- 您可以使用地理編碼將位址轉換為座標，以標準化和填補儲存在資料庫表格中位址的資料間隙。
- 您可以對位址進行地理編碼以取得其位置，並將座標與資料庫空間函數搭配使用，例如顯示指定區域中資料列的函數。
- 您可以使用豐富的資料來產生自動報告，例如產生說明特定區域中所有裝置的自動報告，或是機器學習的自動報告，以說明傳送位置更新時失敗率較高的區域。

本教學說明如何使用 Amazon 定 Location Service 來格式化和豐富Amazon Aurora PostgreSQL資料庫表格中存放的地址。

- Amazon Aurora PostgreSQL— 完全受控的關聯式資料庫引擎，與 MySQL 和 PostgreSQL 相容，可輸出高達 MySQL 輸送量的五倍，輸出高達 PostgreSQL 輸送量的三倍，而無需變更大部分現有應用程式。如需詳細資訊，請參閱什麼是 Amazon Aurora？ 在 Amazon Aurora 用戶指南。

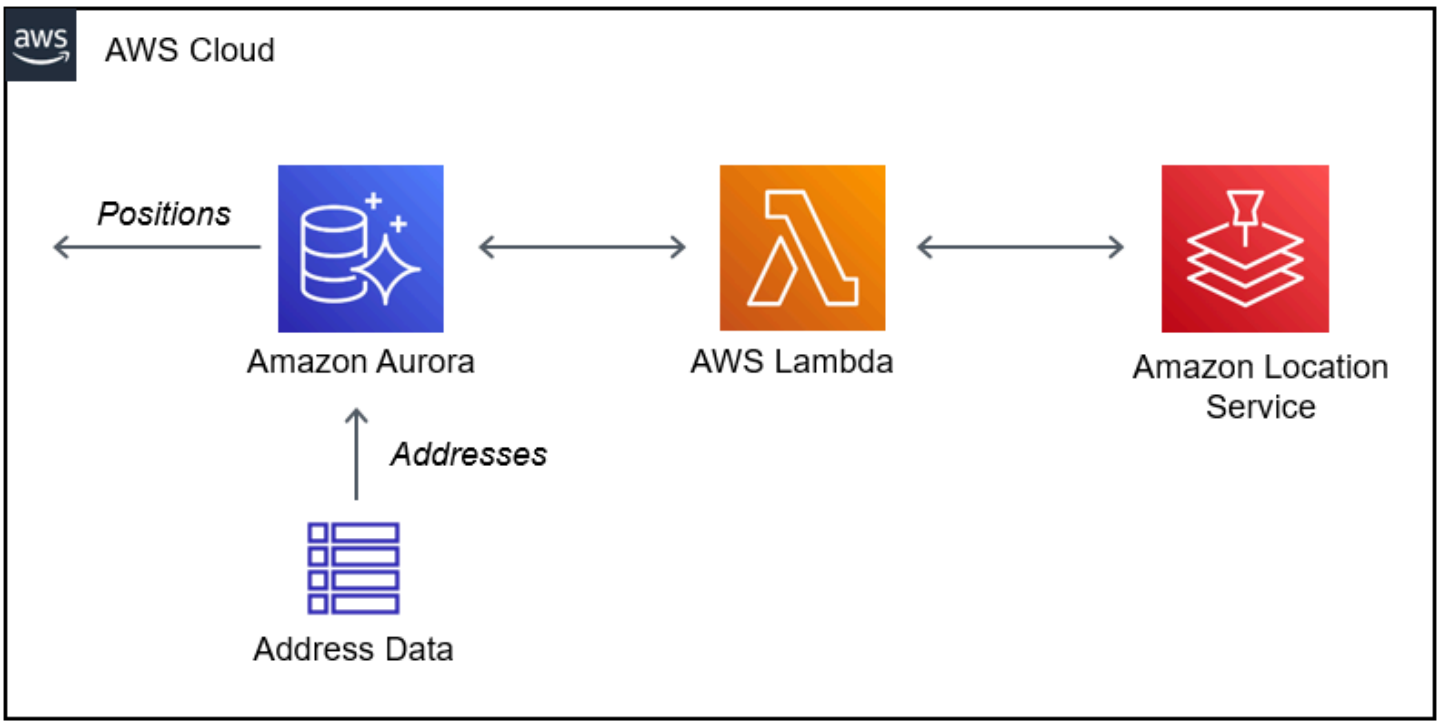
> **Important**
>
> 本教學課程中產生的應用程式會使用儲存地理編碼結果的 place 索引。如需儲存地理編碼結果適用費用的詳細資訊，請參閱 Amazon 定 Location Service 定價。

範例程式碼可在上的 Amazon 定 Location Service 範例儲存庫中取得 GitHub，其中包含AWS CloudFormation範本。

主題

- 概要
- 必要條件
- 快速入門
- 建立位置索引資源
- 建立用於地理AWS Lambda編碼的函數
- 授予Amazon Aurora PostgreSQL存取權 AWS Lambda
- 叫用 AWS Lambda 函數
- 豐富包含地址數據的數據庫
- 後續步驟

概要



該架構涉及以下集成：

- 此解決方案使用 Amazon 位置索引資源來支援使用該操作`SearchPlaceIndexForText`的地理編碼查詢。
- AWS Lambda使用 Python Lambda，當 IAM 政策授予允許AWS Lambda呼叫 Amazon 位置地理編碼操作的權限時，對地址進行地理編碼。`SearchPlaceIndexForText`
- 授與使用 SQL 使Amazon Aurora PostgreSQL用者定義函數叫用地理編碼 Lambda 函數的權限。

必要條件

開始之前，您需要下列先決條件：

- Amazon Aurora PostgreSQL叢集。如需有關建立 Amazon Aurora 資料庫叢集的詳細資訊，請參閱 Amazon Aurora 使用者指南。

> **Note**
>
> 如果您的 Amazon Aurora 叢集無法公開使用，您還必須將 Amazon Aurora 設定為AWS Lambda在AWS帳戶中的虛擬私有雲端 (VPC) 中連線到。如需詳細資訊，請參閱 授予 Amazon Aurora PostgreSQL存取權 AWS Lambda。

- 用於連接到Amazon Aurora PostgreSQL叢集的 SQL 開發人員工具。

快速入門

除了執行本教學中的步驟之外，您還可以啟動快速堆疊以部署支援 Amazon Location 操作的AWS Lambda函數`SearchPlaceIndexForText`。這會自動設定您的AWS帳戶以允許 Amazon Aurora 撥打電話。AWS Lambda

設定AWS帳戶後，您將需要：

- 將 Lambda 功能添加到 Amazon Aurora。請參閱中的將 IAM 角色新增至 Amazon Aurora 資料庫叢集授予Amazon Aurora PostgreSQL存取權 AWS Lambda。
- 將用戶定義的函數加載到數據庫中。請參閱 叫用 AWS Lambda 函數。

Launch Stack

建立位置索引資源

首先創建地點索引資源以支持地理編碼查詢。

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 定 Location Service 控制台。
2. 在左側導覽窗格中，選擇 [置入索引]。
3. 填寫下列方塊：
   - 名稱 — 輸入位置索引資源的名稱。例如，*AuroraPlaceIndex*. 最多可輸入 100 個字元 有效的項目包括英數字元、連字號、句號和底線。
   - 說明 — 輸入選擇性說明。例如，*Amazon Aurora #####*。
4. 在「資料提供者」下，選擇可用的資料提供者，以與您的 place 索引資源搭配使用。如果您沒有偏好，我們建議您從 *Esri* 開始。

5. 在資料儲存選項下，指定是，將儲存結果。這表示您要將地理編碼結果儲存在資料庫中。
6. (選用) 在 Tags (標籤) 底下，輸入標籤 Key (金鑰) 與 Value (值)。這將添加一個標籤您的新位置索引資源。如需詳細資訊，請參閱標記您的資源。
7. 選擇 [建立地點索引]。

建立用於地理AWS Lambda編碼的函數

若要建立Amazon Aurora PostgreSQL和 Amazon Location Service 之間的連線，您需要一個AWS Lambda函數來處理來自資料庫引擎的請求。此函數會轉譯 Lambda 使用者定義函數事件，並呼叫 Amazon 位置作業`SearchPlaceIndexForText`。

您可以使用AWS Lambda主控台AWS Command Line Interface、或 AWS Lambda API 建立函數。

若要使用主控台建立 Lambda 使用者定義函數

1. 前往 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 在左側導覽中，選擇 [功能]。
3. 選擇「建立函數」，並確定已選取「從頭開始作者」。
4. 填寫下列方塊：
   - 函數名稱 — 輸入函數的唯一名稱。有效的項目包括英數字元、連字號和底線 (不含空格)。例如，*AuroraGeocoder*.
   - 執行階段 — 選擇 *Python 3.8*。
5. 選擇 建立函式。
6. 選擇 [程式碼] 索引標籤以開啟編輯器。
7. 使用下列項目覆寫中`lambda_function.py`的預留位置程式碼：

```
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. 如果您將 place index 命名為其他名稱 *AuroraPlaceIndex*，請創建一個名為的環境變量，PLACE_INDEX_NAME將資源名稱分配給：
   - 從組態索引標籤中，選擇環境變數。
   - 選擇編輯，然後選擇新增環境變數。
   - 對於鍵：輸入PLACE_INDEX_NAME。
   - 對於值：輸入位置索引資源的名稱。
9. 選擇部署以儲存更新的功能。
10. 從「測試」下拉式功能表中選擇「設定測試事件」。
11. 選擇 建立新測試事件。
12. 輸入下列測試事件：

```
{
  "text": "Baker Beach",
  "biasPosition": [-122.483, 37.790],
```

```
  "filterCountries": ["USA"]
}
```

13. 選擇「測試」以測試 Lambda 函數。
14. 選擇 Configuration (組態) 索引標籤。
15. 在 [一般設定] 下：選擇 [權限]
16. 在執行角色下：選擇超連結的角色名稱，以將 Amazon Location Service 許可授與您的 Lambda 函數。
17. 在「權限」標籤下：選取「新增權限」下拉式清單，然後選擇「建立內嵌原則」。
18. 選擇 JSON 標籤。
19. 新增下列 IAM 政策：

    - 下列原則授予傳送`SearchPlaceIndexForText`至 place 索引資源的權限*`AuroraPlaceIndex`*。

    ```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "geo:SearchPlaceIndexForText",
          "Resource": "arn:aws:geo:<Region>:<AccountId>:place-index/AuroraPlaceIndex"
        }
      ]
    }
    ```

20. 選擇檢閱政策。
21. 輸入政策名稱。例如，*`AuroraPlaceIndexReadOnly`*.
22. 選擇建立政策。

授予Amazon Aurora PostgreSQL存取權 AWS Lambda

在Amazon Aurora PostgreSQL可以調用AWS Lambda函數之前，您必須授予訪問權限。

如果您的Amazon Aurora PostgreSQL叢集無法公開存取，您必須先建立 VPC 端點，以便 AWS Lambda Amazon Aurora 呼叫您的 Lambda 函數。

為以下項目建立 VPC 端點 AWS Lambda

> **Note**
>
> 只有當您的Amazon Aurora PostgreSQL叢集無法公開存取時，才需要執行此步驟。

1. 開啟 Amazon Virtual Private Cloud Console。
2. 在左側導覽中，選擇「端點」。
3. 選擇 建立端點。
4. 在「服務名稱」篩選器中，輸入「lambda」，然後選擇`com.amazonaws.<region>.lambda`。
5. 選擇包含您的 Aurora 叢集的 VPC。
6. 為每個可用區域選擇一個子網路。
7. 在安全群組篩選器中，輸入「default」或 Aurora 叢集所屬安全群組的名稱，然後選擇安全性群組。
8. 選擇 建立端點。

建立 IAM 政策以授予叫用AWS Lambda函數的權限

1. 開啟 IAM 主控台。
2. 在左側導覽中，展開 [存取管理] 以選擇 [原則]。
3. 選擇建立政策。
4. 在 JSON 索引標籤上，輸入下列原則：

   - 以下是授與叫用`AuroraGeocoder`AWS Lambda函數之Amazon Aurora PostgreSQL權限的 IAM 政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:<Region>:<AccountId>:function:AuroraGeocoder"
            ]
        }
    ]
```

```
}
```

5. 選擇下一步:標籤以新增選擇性標籤。
6. 選擇 下一步：檢閱。
7. 檢閱您的保單，並輸入保單的下列詳細資料：
   - 名稱 — 使用字母數字和 '+=, .@-_' 字元。最多 128 個字元。例如，*AuroraGeocoderInvoke*.
   - 說明 — 輸入選擇性說明。使用英數字元和 '+=, .@-_' 字元。最多可輸入 1000 個字元
8. 選擇建立政策。請注意此政策的 ARN，您可以使用該 ARN 將政策附加到 IAM 角色。

建立 IAM 角色以授予權限給 Amazon Relational Database Service 服務 (Amazon RDS)

透過建立 IAM 角色，Amazon Aurora PostgreSQL可以代表您擔任該角色來存取您的 Lambda 函數。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將許可委派給 IAM 使用者。

下列範例是建立名為之角色的AWS CLI命令 *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"
        }
    ]
}'
```

將您的 IAM 政策附加到 IAM 角色

當您具有 IAM 角色時，請附加您已建立的 IAM 政策。

下列範例是將原則附加*AuroraGeocoderInvoke*至角色的AWS CLI命令*AuroraGeocoderInvokeRole*。

```
aws iam attach-role-policy  --policy-arn AuroraGeocoderInvoke  --role-
name AuroraGeocoderInvokeRole
```

將 IAM 角色新增至 Amazon Aurora 資料庫叢集

下列範例是將 IAM 角色新增至名為的Amazon Aurora PostgreSQL資料庫叢集的AWS CLI命令*MyAuroraCluster*。

```
aws rds add-role-to-db-cluster \
--db-cluster-identifier MyAuroraCluster \
--feature-name Lambda \
--role-arn AuroraGeocoderInvokeRole   \
--region your-region
```

叫用 AWS Lambda 函數

授與叫用地理編碼 Lambda 函數的權限後，您可以建立使用Amazon Aurora PostgreSQL者定義函數來叫用地理編碼AWS Lambda函數。Amazon Aurora PostgreSQL如需詳細資訊，請參閱 Amazon Aurora 使用者指南中的從資Amazon Aurora PostgreSQL料庫叢集叫用AWS Lambda函數。

安裝所需的擴充功能

若要安裝必要的 PostgreSQL 擴充功能`aws_lambda`和`aws _commons`擴充功能，請參閱 Amazon Aurora 使用者指南中的使用 Lambda 函數概觀。

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
```

安裝所需的擴充 PostGIS 能

PostGIS 是 PostgreSQL 的擴充功能，可用於儲存和管理空間資訊。如需詳細資訊，請參閱 Amazon Relational Database Service 使用者指南中的使用 PostGIS 擴充功能。

```
CREATE EXTENSION IF NOT EXISTS postgis;
```

建立叫用 Lambda 函數的 SQL 使用者定義函數

在 SQL 編輯器中，建立新的使用者定義函數`f_SearchPlaceIndexForText`來叫用函數*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;
$function$;
```

調用 SQL 函數從 Aurora 進行地理編碼

執行 SQL 陳述式會叫用 Lambda 函數 *AuroraGeocoder*，該函數會從資料庫中的資料庫表格取得地址記錄，並使用地點索引Amazon Aurora PostgreSQL資源對這些記錄進行地理編碼。

> **Note**
>
> Amazon Aurora PostgreSQL針對 SQL 使用者定義函數的每次呼叫叫用 Lambda 函數。如果您要對 50 個資料列進行地理編碼，請Amazon Aurora PostgreSQL呼叫 Lambda 函數 50 次。每一列都有一次叫用。

下列 `f_SearchPlaceIndexForText` SQL 函數透過 *AuroraGeocoder*Lambda 函數向 Amazon 位置的 SearchPlaceIndexForText API 發出請求。該函數返回一個作為 PostGIS 幾何圖形的`geom`列，該列`ST_AsText(geom)`轉換為文本。

```
SELECT *, ST_AsText(geom)
FROM f_SearchPlaceIndexForText('Vancouver, BC');
```

默認情況下，返回將包含一行。若要要求其他資料列，請在加拿大的`MaxResults`限制範圍內執行下列 SQL 陳述式，同時提供`BiasPosition`和限制結果。

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', ST_MakePoint(-123.113, 49.260), null,
 '{"CAN"}', 5);
```

若要使用邊界方塊篩選結果，請將 a 傳遞Box2D為`filter_bbox`：

- FilterBBox— 傳回邊界方框內的位置來篩選結果。這是選擇性的參數。

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', null, 'BOX(-139.06 48.30, -114.03
 60.00)'::box2d, '{"CAN"}', 5);
```

如需有關 PostGIS 類型和函數的詳細資訊，請參閱 PostGIS 參考資料。

豐富包含地址數據的數據庫

您可以建構格式化的地址，並使用 Amazon Location 操作同時標準化和地理編碼`SearchPlaceIndexForText`給定的資料庫表格，並將下列資料分成以下各欄：

- `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;
```

下面的例子說明了一個生成的數據表行：

```
 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)
```

更新資料庫表格並填入欄

下列範例會更新資料表，並將SearchPlaceIndexForText查詢結果填入資料行：

```
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;
```

後續步驟

範例程式碼可在上的 Amazon 定 Location Service 範例儲存庫中取得 GitHub，其中包含AWS CloudFormation範本。

管理您的地方索引資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 來AWS CLI管理地點索引資源。

列出您的地點索引資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 來檢視地點索引資源清單：AWS CLI

Console

使用 Amazon 位置主控台檢視地點索引資源清單

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇 [放置索引]。
3. 在「我的位置」索引下檢視您的位置索引資源清單。

API

使用來自 Amazon 位置位置 API 的ListPlaceIndexes操作。

下列範例是取得AWS帳戶中位置索引資源清單的 API 要求。

```
POST /places/v0/list-indexes
```

以下為範例回應ListPlaceIndexes：

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-30T01:38:36Z,
         "DataSource": "Esri",
         "Description": "string",
         "IndexName": "ExamplePlaceIndex",
         "UpdateTime": 2020-10-30T01:40:36Z
      }
```

```
    ],
    "NextToken": "1234-5678-9012"
}
```

CLI

使用 `list-place-indexes` 命令。

下列範例是取AWS CLI得AWS帳號中放置索引資源的清單。

```
aws location list-place-indexes
```

取得位置索引資源詳細資

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，取得AWS帳戶中任何位置索引資源的詳細資訊：AWS CLI

Console

使用 Amazon 位置主控台檢視地點索引資源的詳細資料

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇 [放置索引]。
3. 在 [我的位置索引] 下，選取目標位置索引資源的名稱連結。

API

使用來自 Amazon 位置地點 API 的`DescribePlaceIndex`操作。

下列範例是取得位置索引資源詳細資料的 API 要求*`ExamplePlaceIndex`*。

```
GET /places/v0/indexes/ExamplePlaceIndex
```

以下為範例回應`DescribePlaceIndex`：

```
{
    "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

使用 describe-place-index 命令。

下面的實例是一個獲AWS CLI取的地方索引資源的詳細信息*ExamplePlaceIndex*。

```
aws location describe-place-index \
    --index-name "ExamplePlaceIndex"
```

刪除位置索引資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，從 AWS 帳戶刪除地點索引資源：AWS CLI

Console

使用 Amazon 位置主控台刪除地點索引資源

> **Warning**
> 此作業會永久刪除資源。

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇 [放置索引]。
3. 在我的位置索引下，選擇目標位置索引資源。
4. 選擇刪除位置索引。

API

使用來自 Amazon 位置位置 API 的DeletePlaceIndex操作。

下列範例是刪除位置索引資源的 API 要求*ExamplePlaceIndex*。

```
DELETE /places/v0/indexes/ExamplePlaceIndex
```

以下是成功回應的範例DeletePlaceIndex：

```
HTTP/1.1 200
```

CLI

使用 delete-place-index 命令。

下面的例子是刪除地點索引資源的AWS CLI命令*ExamplePlaceIndex*。

```
aws location delete-place-index \
    --index-name "ExamplePlaceIndex"
```

使用 Amazon 定 Location Service 計算路線

Amazon Location 可讓您透過建立和設定路由計算器資源來選取用於計算路由的資料提供者。

您可以使用路由計算器資源，使用 AWS SDK 或 REST API 端點來計算給定特定參數的路由。使用此路線計算器資源來計算出發地，目的地和多達 23 個航點之間的路線，用於不同的運輸方式，避免和交通狀況。

您也可以使用路線計算器資源，透過計算路線矩陣，為您的路線規劃演算法或產品建立輸入。計算一組出發位置和一組目的地位置之間的旅行時間和旅行距離。路線規劃軟件可以使用該時間和距離數據優化一條路線或一組路線；例如，如果您正在規劃多條配送路線，並希望為每個站點找到最佳路線和時間。您可以計算不同運輸模式，避免和交通狀況的路線矩陣。

> **Note**
>
> 如需繞線概念的概觀，請參閱路由。

主題

- 必要條件
- 計算路線
- 使用路線矩陣規劃路線
- 不位於道路上的位置
- 出發時間
- 旅行模式
- 管理您的路線計算器資源

必要條件

在開始計算路線之前，請遵循先決條件步驟：

主題

- 建立路線計算器資源
- 驗證您的請求

建立路線計算器資源

在您可以計算路線之前，請先在AWS帳戶中建立路線計算機資源。

建立路線計算器資源時，您可以從可用的資料提供者中進行選擇：

1. Esri — 有關 Esri 在您感興趣的地區覆蓋的更多信息，請參閱 Esri 有關街道網絡和流量覆蓋範圍的詳細信息。
2. HERE 技術 — 有關您感興趣地區 HERE 覆蓋範圍的更多信息，請參閱 HERE 汽車路線覆蓋範圍和 HERE 卡車路線覆蓋範圍。
3. 抓取 — 如需 Grab 涵蓋範圍的詳細資訊，請參閱所涵蓋的國家 / 地區。

> **Note**
>
> 如果您的應用程式正在對業務中使用的資產 (例如送貨車輛或員工) 進行追蹤或路線規劃，則不得使用 Esri 作為地理位置提供者。如需詳細資訊，請參閱 AWS 服務條款第 82 條。

您可以使用 Amazon 定 Location Service 主控台AWS CLI、或 Amazon 位置 API 來執行此操作。

Console

使用 Amazon 位置主控台建立路線計算器資源

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 在左側導覽窗格中，選擇「路線計算器」。
3. 選擇建立路線計算器。
4. 填寫下列方塊：
   - 名稱 — 輸入路線計算器資源的名稱。例如，*ExampleCalculator*. 最多可輸入 100 個字元 有效的項目包括英數字元、連字號、句號和底線。
   - 說明 — 輸入選擇性說明。
5. 對於資料提供者，請選擇要用作路線計算器的資料提供者。
6. (選用) 在 Tags (標籤) 底下，輸入標籤 Key (金鑰) 與 Value (值)。這將為您的新路線計算器資源添加一個標籤。如需詳細資訊，請參閱標記您的資源。
7. 選擇建立路線計算器。

API

使用 Amazon 位置 API 創建路線計算器資源

使用來自 Amazon 位置位置 API 的CreateRouteCalculator操作。

下列範例是建立*ExampleCalculator*使用資料提供者 *Esri* 呼叫的路由計算器資源的 API 要求。

```
POST /routes/v0/calculators
Content-type: application/json

{
   "CalculatorName": "ExampleCalculator",
   "DataSource": "Esri",
   "Description": "string",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

使用指令建立路線計算器資源的AWS CLI步驟

使用 `create-route-calculator` 命令。

下列範例會建立*ExampleCalculator*使用 *Esri* 做為資料提供者呼叫的路由計算器資源。

```
aws location \
  create-route-calculator \
  --calculator-name "ExampleCalculator" \
  --data-source "Esri" \
  --tags Tag1=Value1
```

> **Note**
>
> 帳單取決於您的使用情況。您可能會因使用其他AWS服務而產生費用。如需詳細資訊，請參閱 Amazon 定 Location Service 定價。

驗證您的請求

一旦您建立路由計算機資源，並準備好開始在應用程式中建置位置功能，請選擇驗證要求的方式：

- 若要探索存取服務的方式，請參閱存取 Amazon 定 Location Service。
- 如果您有一個擁有匿名用戶的網站，則可能需要使用 API 密鑰或 Amazon Cognito。

範例

下列範例顯示如何使用 API 金鑰進行授權、使用 JavaScriptAWS 開發套件 v3 和 Amazon 位置JavaScript 驗證助手。

```
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);
```

計算路線

您可以使用 Amazon 定 Location Service 來計算起點和目的地之間的路線，路線上最多 23 個航點，適用於不同的運輸、避免和交通狀況。

> **Note**
>
> 您必須先建立路由計算機資源，並為您向 Amazon 位置的請求設定身份驗證。如需詳細資訊，請參閱 必要條件。

開始計算路線

使用`CalculateRoute`操作提交簡單的請求。一個簡單的請求包含以下必填字段：

- `DeparturePosition`— 要從中計算路線的起始位置。定義為 `[longitude, latitude]`
- `DestinationPosition`— 要計算路線的結束位置。定義為`[longitude, latitude]`。

> **Note**
>
> 如果您指定不在道路上的出發地或目的地位置，Amazon Location 會將該位置移至最近的道路。

您可以選擇在請求中指定航點，出發時間和旅行模式。

您可以使用AWS CLI或 Amazon 位置 API。

API

下面的例子是使用路線計算器資源的CalculateRoute請求*ExampleCalculator*。*######### ### [-122.7565, 49.0021] ######### [-122.3394, 47.6159]#*

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159]
}
```

AWS CLI

以下範例是使用路線計算機資源的calculate-route指令*ExampleCalculator*。*########## ## [-122.7565, 49.0021] ######### [-122.3394, 47.6159]#*

```
aws location \
    calculate-route \
        --calculator-name ExampleCalculator \
        --departure-position -122.7565 49.0021 \
        --destination-position -122.3394 47.6159
```

依預設，回應會以公里為單Distance位傳回。您可以使用下列選用參數將測量單位變更為英哩：

- DistanceUnit— 指定用於距離結果的單位系統。

Example

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "DistanceUnit": "Miles"
}
```

設定航點

計算路線時，您可以使用航點位置在出發位置和目的地位置之間指定多達 23 個中間中途停留點。

- `WaypointPositions`— 指定沿著起飛位置和目標位置之間的路線包含的中間位置的有序清單。

> **Note**
>
> 如果您指定的航點位置不在道路上，Amazon Location 會將該位置移至最近的道路。

Example

以下`CalculateRoute`請求計算帶有 2 個航點的路線：

- 出發位置為 [-122.7565, 49.0021]，目的地位置為 [-122.3394, 47.6159]。
- 對於請求參數`WaypointPositions`：
  - 位置的第一個止損為 [*-122.1884*, 48.0936]。
  - 位置的第二個止損為 [*-122.3493*, 47.6205]。
- *############################## true#*
  - `IncludeLegGeometry`— 包括回應中一對位置之間每個路徑的幾何圖形。

```
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
}
```

回應範例

以下是在`IncludeLegGeometry`設為 *true* 的 Amazon Location Route API 呼叫`CalculateRoute`作業時，具有相應回應的範例請求，其中包含回應中一對位置之間每個路徑的線串幾何。

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
    ]
   }
}
```

使用路線矩陣規劃路線

您可以使用 Amazon 定 Location Service 為您的路線規劃和優化軟體建立輸入。您可以為一組出發位置和一組目的地位置之間的路線創建路線結果，包括旅行時間和行駛距離。

例如，指定出發位置 A 和 B，以及目的地位置 X 和 Y，Amazon 定 Location Service 將返回從 A 到 X、A 到 Y、B 到 X 的路線的旅行時間和旅行距離，以及從 B 返回 Y 的路線的旅行時間和旅行距離。

您可以計算具有不同交通方式，避免和交通狀況的路線。例如，您可以指定車輛是 35 英呎長的卡車，計算的路線將使用這些限制來確定行駛時間和行駛距離。

傳回的結果數 (以及計算的路線) 是出發位置數乘以目的地位置數目。您需按計算出的每條路線收費，而不是每個服務的請求，因此包含 10 個出發地和 10 個目的地的路線矩陣將以 100 條路線計費。

計算路由矩陣

您可以計算一組出發位置和一組目標位置之間的路線矩陣。路線結果將包括旅行時間和旅行距離。

必要條件

- 您必須先建立路由計算機資源，並為您向 Amazon 位置的請求設定身份驗證。如需詳細資訊，請參閱 必要條件。

使用`CalculateRouteMatrix`作業提交請求。最小要求包含下列必要欄位：

- `DeparturePositions`— 要計算路線的起始位置集。定義為陣列 `[longitude, latitude]`
- `DestinationPositions`— 要計算路線的結束位置集。定義為的陣列`[longitude, latitude]`。

> **Note**
>
> 如果您指定不在道路上的出發地或目的地位置，Amazon Location 會將該位置移至最近的道路。

您可以選擇在您的要求中指定出發時間和旅行模式。

您可以使用AWS CLI或 Amazon 位置 API。

API

下面的例子是使用路線計算器資源的CalculateRouteMatrix請求*ExampleCalculator*。*### ######### [-122.7565, 49.0021] # [-122.2014, 47.6101] ############*

```
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

以下範例是使用路線計算機資源的calculate-route-matrix指令*ExampleCalculator*。*### ######### [-122.7565, 49.0021] # [-122.2014, 47.6101] ############*

```
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]]"
```

依預設，回應會以公里為單Distance位傳回。您可以使用下列選用參數將測量單位變更為英哩：

- DistanceUnit— 指定用於距離結果的單位系統。

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"
}
```

出發和目的地位置的限制

計算路線矩陣時，出發和目的地位置有限制。這些限制會根據RouteCalculator資源使用的提供者而有所不同。

限制	埃斯里	抓斗	這裡
職位數	最多 10 個出發位置和 10 個目的地位置。	多達 350 個出發位置和 350 個目的地位置。	多達 350 個出發位置和 350 個目的地位置。 對於較長的路線，則適用其他限制。請參閱章節。
位置之間的距離	任何一對出發和目的地位置必須彼此相距 400 公里（步行路線為 40 公里）。		所有出發地和目的地位置必須在直徑 180 公里的圓圈內。 對於較長的路線，則適用其他限制。請參閱章節。
路線長度	如路線的總行程時間超過 400 分鐘，路線將不會完成。		出發地和目的地點周圍偏離超過 10 公里的路線將不計算。

限制	埃斯里	抓斗	這裡
			對於較長的路線，則適用其他限制。請參閱章節。
區域	韓國不支援計算路由矩陣。	適用於東南亞。如需支援的國家/地區清單以及更多資訊，請參閱。所涵蓋的國家 / 地區	沒有額外的限制。

更長的路線規劃

計算路由結果矩陣對於有效的路線規劃很有用，但計算可能需要一些時間。所有 Amazon 定 Location Service 資料供應商都會限制路線數量或可計算的路線距離。例如，HERE 允許在 350 個出發位置和目的地位置之間創建路線，但這些位置必須在 180 公里的圓圈內。如果您想計劃更長的路線怎麼辦？

您可以計算較少路由數目不受限制的路由的路由矩陣，使用, a `RouteCalculator` 與 HERE 作為資料提供者。這並不會改變您呼叫 `CalculateRouteMatrix` API 的方式，當您符合要求時，Amazon 位置只會允許更長的路由。

較長的路線計算需求如下：

- `RouteCalculator`必須使用 HERE 資料提供者。
- 出發位置的數量不得大於 15。
- 要計算的路線總數不得大於 100。
- 當路線超過 1,000 公里時，不允許使用長途路線進行避免收費的卡車路線。 此組合的計算速度較慢，而且可能會導致通話逾時。您可以使用CalculateRoute作業個別計算這些路線。

如果您的呼叫不符合這些要求 (例如，您在單一呼叫中要求 150 個路由計算)，則`CalculateRouteMatrix`會還原為僅允許較短的路由規則。然後，只要位置在 180 公里圓內，您就可以計算路線。

計算較長的路線時，請記住以下幾點：

- 較長的路由可能需要更長的時間來計算，甚至比 Amazon 位置 API 的最長時間還要長。如果您經常使用特定路由逾時，您可以在每次呼叫中嘗試較少數量的路由。`CalculateRouteMatrix`

- 如果您在CalculateRouteMatrix請求中新增更多目的地或出發位置，則作業可以切換到更受限制的模式，而且當要建立的路線較少時，可以毫無問題地計算路由的錯誤訊息。在這種情況下，請減少目的地或出發位置的數量，並提出多個請求以獲得所需的全套路線計算。

回應範例

以下是從 Amazon 位置路由 API 呼叫CalculateRouteMatrix作業時，具有對應回應的範例請求。

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"
    }
}
```

不位於道路上的位置

使用CalculateRoute或時CalculateRouteMatrix，如果您指定不在道路上的出發地、目的地或航點位置，Amazon Location 會將該位置移至附近的道路。

以下CalculateRoute請求指定了不在道路上的出發位置和目的地位置：

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-123.128014, 49.298472],
   "DestinationPosition": [-123.134701, 49. 294315]
}
```

產生的回應會傳回鎖點至附近道路的位置：

```
{
   "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"
   }
}
```

出發時間

默認情況下，當您致電CalculateRoute或者CalculateRouteMatrix，如果您未在請求中提供出發時間，則計算出的路線將反映最佳交通狀況。

您可以使用下列其中一個選項，設定特定的出發時間，以使用來自所選資料提供者的即時和預測交通狀況：

- DepartNow— 設置為 *true* 時，它使用實時交通狀況來計算最快的行駛路徑。
- DepartureTime— 如果提供，它會在要求的時間內使用預測性和已知的交通狀況。以下列格式定義：YYYY-MM-DDThh:mm:ss.sssZ

Example

下列CalculateRoute要求會將出發時間設為世界標準時間 2024 年 7 月 2 日 12:15:20。

```
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,
}
```

旅行模式

您可以在使用或時設定行程模CalculateRoute式CalculateRouteMatrix。行駛方式會影響行駛速度和道路兼容性。雖然默認的旅行模式是開車，但您可以使用以下可選參數指定沿著路線旅行時所使用的旅行模式：

- TravelMode— 指定計算路線時的傳輸模式，例如：*BicycleCar*、*Motorcycle*、*Truck*、或*Walking*。

限制

- 如果您指定旅Walking行模式，而您的資料提供者是 Esri，則起點和目的地必須在 40 公里內。
- Bicycle或者僅Motorcycle在使用 Grab 作為數據提供者時可用。
- Grab 僅在某些城市提供Bicycle和Walking路線。如需詳細資訊，請參閱 所涵蓋的國家 / 地區。
- Truck使用 Grab 作為資料提供者時無法使用。

其他偏好

如果您指定TravelMode的是 *Car*，您可以使用下列選用參數指定其他路由偏好設定：

- CarModeOptions— 指定在汽車旅行時的路線偏好，例如*AvoidFerries*或*AvoidTolls*。

如果您指定TravelMode的是 *Truck*，您可以使用下列選用參數指定其他路由偏好設定：

- TruckModeOptions— 指定在卡車中行駛時的路線偏好，例如*AvoidFerries*或 *AvoidTolls*，除了指定可容納*TruckDimensions*和的路線之外*TruckWeight*。

Example

以下CalculateRoute請求指定*Truck*為旅行模式。其他路線限制包括：避免使用渡輪的路線，並避開無法容納卡車尺寸和重量的道路。

```
{
   "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"
      }
   }
}
```

管理您的路線計算器資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 來AWS CLI管理路線計算器資源。

列出您的路線計算器資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 來檢AWS CLI視路線計算器清單：

Console

使用 Amazon 位置主控台檢視路線計算器清單

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇「路線計算器」。
3. 在「我的路線計算器」下查看路線計算器詳細資料。

API

使用來自 Amazon 位置路由 API 的`ListRouteCalculators`操作。

下列範例是取得AWS帳戶中路由計算器清單的 API 要求。

```
POST /routes/v0/list-calculators
```

以下為範例回應[ListRouteCalculators](#)：

```
{
       "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

使用 `list-route-calculators` 命令。

下面的例子是一個AWS CLI獲取AWS帳戶中的路由計算器列表。

```
aws location list-route-calculators
```

獲取路線計算器詳情

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，取得AWS帳戶中任何路線計算器資源的詳細資訊：AWS CLI

Console

使用 Amazon 位置主控台檢視路線計算器的詳細資料

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇「路線計算器」。
3. 在我的路線計算器下，選擇目標路線計算器的名稱鏈接。

API

使用來自 Amazon 位置路由 API 的DescribeRouteCalculator操作。

下列範例是取得路線計算器詳細資料的 API 要求*ExampleCalculator*。

```
GET /routes/v0/calculators/ExampleCalculator
```

以下為範例回應DescribeRouteCalculator：

```
{
       "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

使用 `describe-route-calculator` 命令。

下面的例子是一個獲AWS CLI取路線計算器的詳細信息*ExampleCalculator*。

```
aws location describe-route-calculator \
        --calculator-name "ExampleCalculator"
```

刪除路線計算器

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，從AWS帳戶刪除路線計算器：AWS CLI

Console

使用 Amazon 位置控制台刪除路線計算器

> **Warning**
>
> 此作業會永久刪除資源。

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇「路線計算器」。
3. 在我的路線計算器下，選擇目標路線計算器。
4. 選擇刪除路線計算機。

API

使用來自 Amazon 位置路由 API 的DeleteRouteCalculator操作。

下列範例是刪除地理圍欄集合的 API 要求。*ExampleCalculator*

```
DELETE /routes/v0/calculators/ExampleCalculator
```

以下為範例回應DeleteRouteCalculator：

```
HTTP/1.1 200
```

CLI

使用 `delete-route-calculator` 命令。

下列範例是刪除地理圍欄集合的AWS CLI命令。*ExampleCalculator*

```
aws location delete-route-calculator \
        --calculator-name "ExampleCalculator"
```

使用 Amazon 位置對感興趣的區域進行地理圍欄

地理圍欄應用程序評估跟踪設備相對於先前註冊的感興趣區域的位置。這可讓您根據位置更新採取動作。例如，您可以啟動事件，當客戶在其行動應用程式上訂購咖啡時，就會發出通知。

> **Note**
>
> 如需地理圍欄和追蹤器概念的概述，請參閱。地理圍欄和跟踪器

本指南的本節提供使用 Amazon 定位服務建立地理圍欄應用程式的指 step-by-step 示。

步驟概觀

1. 在感興趣的區域周圍添加地理圍欄，並將其存儲在地理圍欄收集資源中。
2. 開始跟踪您的目標設備並將設備位置歷史記錄存儲在跟踪器資源中。
3. 將您的跟踪器資源鏈接到地理圍欄收集資源，以便根據您的所有地理圍欄自動評估設備位置更新。
4. 如果您不想使用 Amazon 位置追蹤器保留裝置的位置記錄，可以直接對照地理圍欄收集資源評估裝置位置。

實施地理圍欄解決方案後，您的地理圍欄收集資源會發出以下事件：

- `ENTER`— 跟踪的設備進入地理圍欄集合中的地理圍欄。
- `EXIT`— 跟踪設備退出地理圍欄集合中的地理圍欄。

您可以使用 Amazon EventBridge 將事件路由到其他地方來對事件做出反應。

作為透過 Amazon 定位服務APIs從每個裝置傳送更新的替代方法，您可以使用MQTT來傳送裝置更新。

下列主題詳細說明這些步驟和替代方法。

主題

- 添加地理圍欄
- 開始追蹤
- 將追蹤器連結至地理圍欄集合
- 根據地理圍欄評估設備位置
- 驗證裝置位置
- 使用 Amazon 對 Amazon 定 Location Service 事件做出反應 EventBridge
- 使用 AWS IoT 和使用 Amazon Location Service MQTT 進行追蹤
- 管理您的地理圍欄收集資源

- 管理您的追蹤器資源
- 地理圍欄和跟踪移動應用程序示例

添加地理圍欄

地理圍欄包含形成封閉邊界的點和頂點，它定義了感興趣的區域。地理圍欄集合存儲和管理一個或多個地理圍欄。

Amazon 位置地理圍欄集合會儲存使用稱為 Geo (7946) 的標準地理空間資料格式定義的地理JSON圍欄。RFC您可以免費使用諸如 geojson.io 之類的工具以圖形方式繪製地理圍欄並保存輸出的 Geo 文件。JSON

> **Note**
>
> Amazon Location 不支援跨越反經絡的孔、多重多邊形、順時針多邊形和地理圍欄的多邊形。

建立地理圍欄集合

使用 Amazon 位置主控台、或 Amazon 位置，建立地理圍欄集合以存放和管理地理圍欄。 AWS CLI APIs

Console

若要使用 Amazon 位置主控台建立地理圍欄集合

1. 打開 Amazon 定 Location Service 主控台，網址為https://console.aws.amazon.com/location/。
2. 在左側導覽窗格中，選擇「地理圍欄集合」。
3. 選擇 [建立地理圍欄集合]。
4. 填寫下列方塊：
   - 名稱 — 輸入唯一的名稱。例如 `ExampleGeofenceCollection`。 最多可輸入 100 個字元有效的項目包括英數字元、連字號、句號和底線。
   - 說明 — 輸入可選描述以區分您的資源。
5. 在以 CloudWatch 作為目標的EventBridge 規則下，您可以創建一個可選 EventBridge 規則以開始對地理圍欄事件做出反應。這使 Amazon 位置能夠將事件發佈到 Amazon CloudWatch 日誌。

6. (選用) 在 Tags (標籤) 底下，輸入標籤 Key (金鑰) 與 Value (值)。這將為您的新地理圍欄集合添加一個標籤。如需詳細資訊，請參閱標記您的 Amazon Location Service 資源。
7. (選擇性) 在客戶受管金鑰加密底下，您可以選擇新增客戶管理的金鑰。這會新增您透過預設 AWS 擁有的加密建立、擁有和管理的對稱客戶管理金鑰。如需詳細資訊，請參閱加密靜態資料。
8. 選擇 [建立地理圍欄集合]。

API

使用 Amazon 位置創建地理圍欄集合 APIs

使用來自 Amazon 位置地理圍欄APIs的CreateGeofenceCollection操作。

下面的例子使用一個API請求來創建一個名為的地理圍欄集合 *ExampleGeofenceCollection*。地理圍欄集合與客戶管理的 AWS KMS 金鑰相關聯，以加密客戶資料。

```
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

使用指令建立地理圍欄集合的步驟 AWS CLI

使用 create-geofence-collection 命令。

下列範例使用建立 AWS CLI 稱為的地理圍欄集合 *ExampleGeofenceCollection*。 地理圍欄集合與客戶管理的 AWS KMS 金鑰相關聯，以加密客戶資料。

```
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**
>
> 帳單取決於您的使用情況。您可能會因使用其他 AWS 服務而產生費用。如需詳細資訊，請參閱 Amazon 定 Location Service 定價。

繪製地理圍欄

現在您已經建立了地理圍欄集合，您可以定義地理圍欄。地理圍欄被定義為多邊形或圓形。要繪製多邊形地理圍欄，您可以使用地理JSON編輯工具，例如 geojson.io。

若要將地理圍欄建立為圓，您必須定義圓的中心點和半徑。例如，如果您想要建立地理圍欄，以便在設備距離特定位置 50 公尺以內時收到通知，則可以使用該位置的緯度和經度，並將半徑指定為 50 公尺。

使用 Amazon 定 Location ServiceAPIs，您還可以以鍵值對的形式將中繼資料新增至地理圍欄。這些對於存儲地理圍欄的信息（例如其類型或其他應用程序特定的信息）非常有用。您可以在何時使用此中繼資料使用 Amazon 對 Amazon 定 Location Service 事件做出反應 EventBridge。

添加多邊形地理圍欄

本節介紹創建多邊形地理圍欄

使用地理工具繪製地理圍欄 JSON

現在您已經建立了地理圍欄集合，您可以使用 Geo JSON 編輯工具 (例如 geojson.io) 來定義地理圍欄。

若要建立地理JSON檔案

1. 開啟地理JSON編輯工具。例如，巨星 .io。
2. 選擇繪製多邊形圖標並繪製您感興趣的區域。
3. 選擇「儲存」，然後JSON從下拉式選單中選擇「地理」。

將地理圍欄放入JSON地理圍欄集合

您可以使用產生的 Geo JSON 檔案，使用 Amazon 定位服務主控台、或 Amazon 位置來上傳 AWS CLI地理圍欄：APIs

Console

使用 Amazon 定位服務主控台將地理圍欄新增至 Location Service 圍欄集合

1. 打開 Amazon 定 Location Service 主控台，網址為https://console.aws.amazon.com/location/。
2. 在左側導覽窗格中，選擇「地理圍欄集合」。
3. 從「地理圍欄」集合清單中，選取目標地理圍欄集合的名稱連結。
4. 在「地理圍欄」下，選擇「建立地理圍欄」。
5. 在「新增地理圍欄」視窗中，將 Ge o 拖放JSON到視窗中。
6. 選擇 [新增地理圍欄]。

API

使用 Amazon 位置添加地理圍欄 APIs

使用來自 Amazon 位置地理圍欄APIs的PutGeofence操作。

以下示例使用API請求添加給定 ID 的地理圍欄 *GEOFENCE-EXAMPLE1* 到一個名為的地理圍欄集合 *ExampleGeofenceCollection*。 它還使用鍵Type和值指定單個地理圍欄元數據屬性。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]
            ]
        ]
    }
}
```

或者，您可以使用該[BatchPutGeofence](#)操作添加多個地理圍欄。

```
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

使用指令將地理圍欄加入至地理圍欄集合的步驟 AWS CLI

使用 [put-geofence](#) 命令。

下列範例會使用將地理圍欄新增 AWS CLI 至名為的地理圍欄集合 *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"
    }
```

添加循環地理圍欄

本節介紹創建循環地理圍欄。您必須知道要做為圓心點的緯度和經度，以及圓的半徑（以米為單位）。您可以使用 Amazon 位置APIs或. AWS CLI

API

使用 Amazon 位置添加圓形地理圍欄 APIs

使用來自 Amazon 位置地理圍欄APIs的PutGeofence操作。

以下示例使用API請求添加給定 ID 的地理圍欄 *GEOFENCE-EXAMPLE2* 到一個名為的地理圍欄集合 *ExampleGeofenceCollection*:

```
PUT /geofencing/v0/collections/ExampleGeofenceCollection/geofence/GEOFENCE-EXAMPLE2
    Content-type: application/json

    {
       "Geometry": {
          "Circle": {
            "Center": [-5.716667, -15.933333],
            "Radius": 50
          }
       }
    }
```

AWS CLI

使用指令將圓形地理圍欄加入至地理圍欄集合的步驟 AWS CLI

使用 put-geofence 命令。

下列範例會使用將地理圍欄新增 AWS CLI 至名為的地理圍欄集合 *ExampleGeofenceCollection*.

```
$ aws location \
        put-geofence \
            --collection-name ExampleGeofenceCollection \
            --geofence-id ExampleGeofenceCircle \
            --geometry 'Circle={Center=[-5.716667, -15.933333], Radius=50}'
```

> **Note**
>
> 您也可以將複雜JSON的幾何圖形放入其自己的檔案中，如下列範例所示。
>
> ```
> $ aws location \
>         put-geofence \
>             --collection-name ExampleGeofenceCollection \
>             --geofence-id ExampleGeofenceCircle \
>             --geometry file:circle.json
> ```
>
> 在此範例中，圓形 .json 檔案包含圓幾何圖形JSON的檔案。
>
> ```
> {
>     "Circle": {
>         "Center": [-74.006975, 40.717127],
>         "Radius": 287.7897969218057
>     }
> }
> ```

開始追蹤

本節將引導您完成構建捕獲設備位置的跟踪應用程序。

建立追蹤器

創建跟踪器資源以存儲和處理來自您設備的位置更新。您可以使用 Amazon 定 Location Service 主控台 AWS CLI、或 Amazon 位置APIs。

儲存在追蹤器資源中的每個位置更新，都可以包含定位準確度的度量，以及最多三個與您要儲存的位置或裝置有關的中繼資料欄位。元數據存儲為鍵值對，並可以存儲諸如速度，方向，輪胎壓力或引擎溫度等信息。

追蹤器會在收到位置更新時篩選位置更新。這樣可以減少設備路徑中的視覺噪聲（稱為抖動），並減少錯誤的地理圍欄進入和退出事件的數量。這也有助於通過減少啟動地理圍欄評估的數量來管理成本。

追蹤器提供三種位置篩選選項，可協助管理成本並減少位置更新中的抖動。

- 基於準確性 — 與任何提供精度測量的設備一起使用。大多數移動設備都提供此信息。每個位置量測的準確度都會受到許多環境因素的影響，包括GPS衛星接收、景觀以及 Wi-Fi 和藍牙裝置的距離。大多數設備（包括大多數移動設備）都可以提供測量精度的估計以及測量結果。透`AccuracyBased`過篩選功能，如果裝置的移動速度低於測量的準確度，Amazon Location 會忽略位置更新。例如，如果裝置的連續兩次更新精確度範圍為 5 m 和 10 m，則 Amazon Location 會在裝置移動小於 15 公尺時忽略第二次更新。Amazon 位置既不會針對地理圍欄評估被忽略的更新，也不會存儲它們。

  如果未提供精度，則將其視為零，並且測量被認為是完全準確的。

  > **Note**
  >
  > 您也可以使用基於精確度的篩選來移除所有篩選。如果您選取精確度篩選，但將所有準確度資料覆寫為零，或完全忽略準確度，則 Amazon Location 不會篩選出任何更新。

- 以距離為基礎 — 當您的裝置未提供精確度量測，但您仍希望利用濾波功能來減少抖動並管理成本時使用。 `DistanceBased`篩選會忽略裝置移動小於 30 公尺 (98.4 英呎) 的位置更新。當您使用`DistanceBased`位置篩選時，Amazon Location 既不會針對地理圍欄評估這些被忽略的更新，也不會儲存更新。

  大多數移動設備的準確度（包括 iOS 和 Android 設備的平均準確度）在 15 米內。在大多數應用程式中，`DistanceBased`篩選可以減少在地圖上顯示裝置軌跡時，位置不準確的影響，以及當裝置靠近地理圍欄邊界時，多個連續進入和退出事件的反彈效果。它還可以通過減少對鏈接地理圍欄進行評估或檢索設備位置的調用來幫助降低應用程序的成本。

- 以時間為基礎 — (預設) 當您的裝置非常頻繁地傳送位置更新 (每 30 秒超過一次)，且您希望在不儲存每次更新的情況下達到近乎即時的地理圍欄評估時使用。在`TimeBased`過濾中，每個位置更新都會根據鏈接的地理圍欄集合進行評估，但不會存儲每個位置更新。若您的更新頻率超過 30 秒，則每個唯一裝置 ID 每 30 秒只會存放一次更新。

> **Note**
>
> 在決定篩選方法和職位更新頻率時，請注意追蹤應用程式的成本。您需支付每次位置更新的費用，以及針對每個連結的地理圍欄集合評估位置更新一次的費用。例如，使用基於時間的過濾時，如果您的跟踪器鏈接到兩個地理圍欄集合，則每個位置更新將計為一個位置更新請求和兩個地理圍欄集合評估。如果您每 5 秒報告裝置的位置更新，並使用基於時間的篩選功能，則每部裝置每小時需支付 720 個位置更新和 1,440 次地理圍欄評估的費用。
> 您的帳單不受每個集合中的地理圍欄數量的影響。由於每個地理圍欄集合可能包含多達 50,000 個地理圍欄，因此您可能希望將地理圍欄合併為更少的集合，以減少地理圍欄評估的成本。

默認情況下，每次跟踪的設備進入或退出鏈接的地理圍欄時，您都會收到 EventBridge 事件。如需詳細資訊，請參閱將追蹤器連結至地理圍欄集合。

您可以為追蹤器資源的所有篩選職位更新啟用事件。如需詳細資訊，請參閱啟用追蹤器的更新事件。

> **Note**
>
> 如果您希望使用自己的 AWS KMS 客戶管理密鑰加密數據，則默認情況下將禁用「邊界多邊形查詢」功能。這是因為通過使用此邊界多邊形查詢功能，您的設備位置的表示不會使用 AWS KMS 託管密鑰進行加密。不過，確切的裝置位置仍會使用您的受管理金鑰加密。
> 您可以在建立或更新「追蹤器」時將`KmsKeyEnableGeospatialQueries`參數設定為 true，以選擇加入「邊界多邊形查詢」功能。

Console

使用 Amazon 位置主控台建立追蹤器

1. 打開 Amazon 定 Location Service 主控台，網址為https://console.aws.amazon.com/location/。
2. 在左側導覽窗格中，選擇「追蹤器」。
3. 選擇建立追蹤器。
4. 填寫下列欄位：
   - 名稱 — 輸入唯一的名稱。例如 *`ExampleTracker`*。 最多可輸入 100 個字元 有效的項目包括英數字元、連字號、句號和底線。
   - 說明 — 輸入選擇性說明。

5. 在「位置篩選」下，選擇最適合您要如何使用追蹤器資源的選項。如果您未設定「位置篩選」，則預設設定為TimeBased。如需詳細資訊，請參閱本指南追蹤器中的以及 Amazon 定 Location Service 追蹤器API參考PositionFiltering中的。
6. (選用) 在 Tags (標籤) 底下，輸入標籤 Key (金鑰) 與 Value (值)。這將為您的新地理圍欄集合添加一個標籤。如需詳細資訊，請參閱標記您的資源。
7. (選擇性) 在客戶受管金鑰加密底下，您可以選擇新增客戶管理的金鑰。這會新增您透過預設 AWS 擁有的加密建立、擁有和管理的對稱客戶管理金鑰。如需詳細資訊，請參閱加密靜態資料。
8. (可選) 在下 KmsKeyEnableGeospatialQueries，您可以選擇啟用「空間查詢」。這可讓您使用「邊界多邊形查詢」功能，同時使用客戶AWSKMS管理的金鑰加密資料。

   > **Note**
   >
   > 當您使用邊界多邊形查詢功能時，不會使用 AWS KMS 託管密鑰對設備位置的表示進行加密。不過，確切的裝置位置仍會使用您的受管理金鑰加密。

9. (選擇性) 在EventBridge 組態下，您可以選擇為篩選職位更新啟用 EventBridge 事件。每當此追蹤器中裝置的位置更新符合位置篩選評估時，就會傳送事件。
10. 選擇建立追蹤器。

API

使用 Amazon 位置創建跟踪器 APIs

使用 Amazon 位置跟踪器APIs的CreateTracker操作。

下面的例子使用一個API請求來創建一個名為跟踪 *ExampleTracker*。 追蹤器資源與客戶管理的 AWS KMS 金鑰相關聯，用於加密客戶資料，且不會在中啟用位置更新 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"
   }
}
```

建立KmsKeyEnableGeospatialQueries已啟用的追蹤器

下列範例的參數KmsKeyEnableGeospatialQueries設定為 true。這可讓您使用「邊界多邊形查詢」功能，同時使用客戶 AWS KMS 管理的金鑰加密資料。

如需使用「邊界多邊形查詢」功能的資訊，請參閱〈???

> **Note**
>
> 當您使用邊界多邊形查詢功能時，不會使用 AWS KMS 託管密鑰對設備位置的表示進行加密。不過，確切的裝置位置仍會使用您的受管理金鑰加密。

```
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

使用 AWS CLI 指令建立追蹤器

使用 create-tracker 命令。

下列範例會使用建 AWS CLI 立名為的追蹤器 *ExampleTracker*。 追蹤器資源與客戶管理的 AWS KMS 金鑰相關聯，用於加密客戶資料，且不會在中啟用位置更新 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"
```

建立KmsKeyEnableGeospatialQueries已啟用的追蹤器

下列範例的參數KmsKeyEnableGeospatialQueries設定為 true。這可讓您使用「邊界多邊形查詢」功能，同時使用客戶 AWS KMS 管理的金鑰加密資料。

如需使用「邊界多邊形查詢」功能的資訊，請參閱〈???

> **Note**
>
> 當您使用邊界多邊形查詢功能時，不會使用 AWS KMS 託管密鑰對設備位置的表示進行加密。不過，確切的裝置位置仍會使用您的受管理金鑰加密。

```
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**
>
> 帳單取決於您的使用情況。您可能會因使用其他 AWS 服務而產生費用。如需詳細資訊，請參閱 Amazon 定 Location Service 定價。

您可以在建立追蹤器之後，透過選擇編輯追蹤器來編輯描述、位置篩選和EventBridge 組態。

驗證您的請求

創建跟踪器資源並準備好開始針對地理圍欄評估設備位置後，請選擇驗證請求的方式：

- 若要探索存取服務的方式，請參閱存取 Amazon 定 Location Service。
- 如果您想要發佈具有未驗證請求的裝置位置，可能需要使用 Amazon Cognito。

  範例

  下列範例顯示使用 Amazon Cognito 身分集區進行授權、使用 AWS JavaScript SDKv3 和 Amazon 位置JavaScript 驗證助手。

```
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);
```

使用裝置位置更新您的智能設備

若要追蹤您的裝置，您可以將裝置位置更新發佈至智能設備。您稍後可以從跟踪器資源中檢索這些設備位置或設備位置歷史記錄。

每個位置更新都必須包含裝置 ID、時間戳記和位置。您可以選擇包含其他元數據，包括準確性和最多 3 個鍵值對供您自己使用。

如果您的跟踪器鏈接到一個或多個地理圍欄集合，則將根據這些地理圍欄評估更新（遵循您為跟踪器指定的過濾規則）。如果設備違反了地理圍欄區域（通過從區域內移動到外部，反之亦然），您將在中收到事件。 EventBridge這些ENTER或EXIT事件包括位置更新詳細資料，包括裝置 ID、時間戳記和任何相關聯的中繼資料。

> **Note**
>
> 如需位置篩選的詳細資訊，請參閱建立追蹤器。
> 如需地理圍欄事件的詳細資訊，請參閱。使用 Amazon 對 Amazon 定 Location Service 事件做出反應 EventBridge

請使用下列其中一種方法來傳送裝置更新：

- 將MQTT更新傳送至 AWS IoT Core 資源，並將其連結至您的追蹤器資源。
- 使用 Amazon 位置追蹤器傳送位置更新API，方法是使用 AWS CLI、或 Amazon 位置APIs。您可以使用AWS SDKsAPIs從您的 iOS 或安卓應用程式呼叫。

API

使用 Amazon 位置傳送職位更新 APIs

使用 Amazon 位置跟踪器APIs的`BatchUpdateDevicePosition`操作。

下列範例會使用API要求來張貼裝置位置更新 *ExampleDevice* 到追蹤器 *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

使用指令傳送位置更新的 AWS CLI 步驟

使用 batch-update-device-position 命令。

下列範例使 AWS CLI 用張貼的裝置位置更新 *ExampleDevice-1* 以及 *ExampleDevice-2* 到追蹤器 *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"}}]'
```

從追蹤器取得裝置的位置記錄

Amazon 位置追蹤器資源會保留所有追蹤裝置的位置歷史記錄，為期 30 天。您可以從跟踪器資源中檢索設備位置歷史記錄，包括所有關聯的元數據。下列範例使用 AWS CLI、或 Amazon 位置APIs。

API

使用 Amazon 位置從跟踪器獲取設備位置歷史記錄 APIs

使用 Amazon 位置跟踪器APIs的GetDevicePositionHistory操作。

以下示例使用APIURI請求來獲取設備位置歷史記錄 *ExampleDevice* 從一個名為的跟踪器 *ExampleTracker* 從19:05:07（含）開始，結束於19:20:07（獨家）上2020-10-02。

```
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

使用 AWS CLI 指令從追蹤器取得裝置位置歷程記錄

使用 get-device-position-history 命令。

以下示例使用了獲 AWS CLI 取的設備位置歷史記錄 *ExampleDevice* 從一個名為的跟踪器 *ExampleTracker* 從19:05:07（含）開始，結束於19:20:07（獨家）上2020-10-02。

```
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"
```

列出您的設備位置

您可以使用 AWS CLI，或使用 Amazon 位置檢視追蹤器的清單裝置位置APIs ListDevicePositions API。當您呼叫時 ListDevicePositions API，會傳回與指定追蹤器相關聯之所有裝置的最新位置清單。依預設，這會針對特定追蹤器API傳回每頁結果的 100 個最新裝置位置。若要僅傳回特定區域內的裝置，請使用FilterGeometry參數建立邊界多邊形查詢。這樣，當您打電話時 ListDevicePositions，只有多邊形內的設備將被返回。

> **Note**
>
> 如果您希望使用自己的 AWS KMS 客戶管理密鑰加密數據，則默認情況下將禁用「邊界多邊形查詢」功能。這是因為使用此功能，將不會使用 AWS KMS 託管密鑰對設備位置的表示進行加密。但是，確切的裝置位置仍會使用您的受管理金鑰加密。

您可以選擇加入「邊界多邊形查詢」功能。這是通過創建或更新跟踪器時將KmsKeyEnableGeospatialQueries參數設置為 true 來完成的。

API

使用 Amazon 位置跟踪器APIs的ListDevicePositions操作。

下列範例是使用選用參數FilterGeometry取得多邊形區域中裝置位置清單的API要求。此範例會傳回Polygon陣列所定義區域中存在的 3 個裝置位置。

```
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"
}
```

以下為範例回應[ListDevicePositions](#)：

```
{
    "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

使用 `list-trackers` 命令。

下列範例是取 AWS CLI 得多邊形區域中裝置清單的範例。

```
aws location list-device-positions TODO: add arguments add props for filter geo
```

將追蹤器連結至地理圍欄集合

現在您有了地理圍欄集合和跟踪器，您可以將它們鏈接在一起，以便根據所有地理圍欄自動評估位置更新。如果您不想評估所有位置更新，或者，如果您不將某些位置存儲在跟踪器資源中，則可以根據需求評估設備位置與地理圍欄。

根據地理圍欄評估設備位置時，會生成事件。您可以設定這些事件的動作。如需可針對地理圍欄事件設定動作的詳細資訊，請參閱使用 Amazon 對 Amazon 定 Location Service 事件做出回應。
EventBridge

Amazon 位置事件包括產生裝置位置更新的屬性，以及輸入或退出的地理圍欄的某些屬性。如需地理圍欄事件中包含之資料的詳細資訊，請參閱。Amazon 定 Location Service 的 Amazon EventBridge 事件示例

下列範例會使用主控台、或 Amazon 位置，將追蹤器資源連結至地理圍欄集合。 AWS CLI APIs

Console

使用 Amazon 定位服務主控台將追蹤器資源連結至 Location Service 圍欄集合

1. 打開 Amazon 定 Location Service 主控台，網址為https://console.aws.amazon.com/location/。
2. 在左側導覽窗格中，選擇「追蹤器」。
3. 在 [裝置追蹤器] 下，選取目標追蹤器的名稱連結。
4. 在「連結的地理圍欄集合」下，選擇「連結地理圍欄集合」。
5. 在「連結的地理圍欄集合」視窗中，從下拉式選單中選取地理圍欄集合。
6. 選擇 Link (連結)。

在您連結追蹤器資源之後，系統會為其指派「作用中」狀態。

API

使用 Amazon 位置將跟踪器資源鏈接到地理圍欄集合 APIs

使用 Amazon 位置跟踪器APIs的AsssociateTrackerConsumer操作。

下列範例會使用建立關聯的API要求 *ExampleTracker* 使用其 Amazon 資源名稱（ ）ARN的地理圍欄集合。

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json

{
   "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection"
}
```

AWS CLI

使用指令將追蹤器資源連結至地理圍欄集合的步驟 AWS CLI

使用 associate-tracker-consumer 命令。

下列範例使用建立 AWS CLI 稱為的地理圍欄集合 *ExampleGeofenceCollection*.

```
aws location \
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection" \
        --tracker-name "ExampleTracker"
```

根據地理圍欄評估設備位置

有兩種方法可以根據地理圍欄評估位置以生成地理圍欄事件：

- 您可以連結追蹤器和地理圍欄集合。如需詳細資訊，請參閱下列章節：將追蹤器連結至地理圍欄集合。
- 您可以直接向地理圍欄集合資源發出請求，以評估一個或多個位置，使用. BatchEvaluateGeofencesAPI

此外，您還可以預測設備進入，退出或在地理圍欄中保持閒置狀態的傳入地理圍欄事件。使用ForecastGeofenceEventsAPI預測事件。

如果您還想跟踪設備的位置歷史記錄或在地圖上顯示位置，請將跟踪器與地理圍欄集合鏈接。或者，您可能不想評估所有位置更新，或者您不打算將位置數據存儲在跟踪器資源中。如果是這種情況之一，則可以直接向地理圍欄集合發出請求，並根據其地理圍欄評估一個或多個設備位置。

針對地理圍欄評估設備位置會產生事件。您可以對這些事件做出反應，並將它們路由到其他 AWS 服務。如需有關接收地理圍欄事件時可採取之動作的詳細資訊，請參閱使用 Amazon 對 Amazon 定 Location Service 事件做出回應。 EventBridge

Amazon Location 事件包括產生該事件的裝置位置更新的屬性，包括時間、位置、準確性和鍵值中繼資料，以及輸入或退出的地理圍欄的某些屬性。如需地理圍欄事件中包含之資料的詳細資訊，請參閱。Amazon 定 Location Service 的 Amazon EventBridge 事件示例

下列範例使用 AWS CLI、或 Amazon 位置APIs。

API

使用 Amazon 位置根據地理圍欄的位置評估設備位置 APIs

使用來自 Amazon 位置地理圍欄APIs的BatchEvaluateGeofences操作。

下面的例子使用一個API請求來評估設備的位置 *ExampleDevice* 到關聯的地理圍欄集合 *ExampleGeofenceCollection*。 用您自己的地理圍欄和設備替換這些值。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

使用命令根據地理圍欄的位置評估設備位置 AWS CLI

使用 batch-evaluate-geofences 命令。

下列範例使 AWS CLI 用 a 來評估 *ExampleDevice* 針對關聯的地理圍欄集合 *ExampleGeofenceCollection*。 用您自己的地理圍欄和設備替換這些值。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"}}]'
```

針對地理圍欄評估設備位置會產生事件。傳統上，您可以通過使用來對事件做出反應 Amazon EventBridge，但是此過程只允許您在那之後對事件做出反應。如果您需要預測裝置何時進入或離開地理圍欄，例如，如果裝置越過邊界，因此會受到不同的規定限制，則可以使用ForecastGeofenceEventsAPI來預測 future 的地理圍欄事件。

ForecastGeofenceEventsAPI使用諸如設備 time-to-breach，接近度，速度和位置之類的標準來預測事件。API將返回一個 `ForecastedBreachTime`，這表示地理圍欄事件將發生的估計時間。

下列範例使用 Amazon 位置APIs。

API

使用 Amazon 位置預測地理圍欄事件 APIs

使用來自 Amazon 位置地理圍欄APIs的 ForecastGeofenceEvents操作。

下列範例使用API要求來預測地理圍欄事件 *ExampleDevice* 相對於 *ExampleGeofence*。 用您自己的地理圍欄和設備替換這些值。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
}
```

驗證裝置位置

若要檢查裝置位置的完整性，請使用 VerifyDevicePositionAPI. 通過評估設備的小區信號，Wi-Fi 接入點，IPv4 地址以及代理是否正在使用中，這將API返回有關設備位置完整性的信息。

必要條件

在能夠使用列出APIs的設備驗證之前，請確保您具有以下必要條件：

- 您已經為要檢查的一個或多個設備創建了跟踪器。如需詳細資訊，請參閱開始追蹤。

下列範例顯示 Amazon 位置的請求VerifyDevicePositionAPI。

API

使用 Amazon 位置驗證裝置位置 APIs

使用 Amazon 位置跟踪中的VerifyDevicePosition操作APIs。

下列範例顯示評估裝置位置完整性的API要求。用您自己的設備替換這些值IDs。

```
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**
>
> In tegrity SDK 提供了與設備驗證相關的增強功能，並且可以根據請求使用。若要存取SDK，請聯絡銷售 Support。

使用 Amazon 對 Amazon 定 Location Service 事件做出反應 EventBridge

Amazon EventBridge 是一種無伺服器事件匯流排，可使用 Amazon 位置等AWS服務的資料，有效地將應用程式連接在一起。 EventBridge 接收來自 Amazon 位置的事件，並將資料路由到類似的目標 AWS Lambda。您可以設定路由規則，決定要將資料傳送到何處，以建置即時反應的應用程式架構。

只有地理圍欄事件（ENTER和EXIT事件，因為設備進入或離開地理圍欄區域）被默認發送到。EventBridge 您也可以為追蹤器資源啟用所有篩選的職位更新事件。如需詳細資訊，請參閱 啟用追蹤器的更新事件。

如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的事件和事件模式。

主題

- 啟用追蹤器的更新事件
- 為 Amazon 地點創建事件規則
- Amazon 定 Location Service 的 Amazon EventBridge 事件示例

啟用追蹤器的更新事件

根據預設，Amazon 位置僅會傳送地理圍欄事件ENTER和EXIT地理圍欄事件至。 EventBridge您可以為追蹤器啟用要傳送至的所有篩選職位UPDATE事件 EventBridge。您可以在建立或更新追蹤器時執行此操作。

例如，若要使用更新現有的追蹤器AWS CLI，您可以使用下列命令 (使用您的追蹤器資源的名稱來取代 *MyTracker*)。

```
aws location update-tracker --tracker-name MyTracker --event-bridge-enabled
```

若要關閉追蹤器的位置事件，您必須使用 API 或 Amazon 定 Location Service 主控台。

為 Amazon 地點創建事件規則

您可以在每個事件匯流排中建立最多 300 個規則， EventBridge 以設定針對 Amazon 位置事件採取的動作。

例如，您可以為地理圍欄事件建立規則，當在地理圍欄邊界內偵測到電話時，將傳送推播通知。

若要建立 Amazon 位置事件的規則

使用下列值，根據 Amazon 位置事件建立 EventBridge 規則：

- 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
- 在 [事件模式] 方塊中，新增下列模式：

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"]
}
```

若要建立追蹤器位置更新的規則，您可以改為使用下列模式：

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Device Position Event"]
}
```

您可以選擇通過添加`detail`標籤來指定僅`ENTER`或`EXIT`事件（如果您的規則用於跟踪器位置更新`EventType`，則只有一個，因此無需對其進行過濾）：

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"]
  }
}
```

您也可以選擇篩選位置或地理圍欄的屬性：

```
{
  "source": ["aws.geo"],
```

```
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"],
    "GeofenceProperties": {
      "Type": "LoadingDock"
    },
    "PositionProperties": {
      "VehicleType": "Truck"
    }
  }
}
```

- 對於選取目標，請選擇從 Amazon 定 Location Service 收到事件時要採取的目標動作。

  例如，使用 Amazon Simple Notification Service (SNS) 主題在事件發生時傳送電子郵件或文字訊息。您首先需要使用 Amazon SNS 主控台建立 Amazon SNS 主題。如需詳細資訊，請參閱使用 Amazon SNS 取得使用者通知。

> **Warning**
>
> 最佳做法是確認事件規則已成功套用，或您的自動化動作可能無法如預期般啟動。若要驗證您的事件規則，請啟動事件規則的條件。例如，模擬進入地理圍欄區域的設備。

您還可以通過排除該`detail-type`部分來捕獲 Amazon 位置的所有事件。例如：

```
{
  "source": [
    "aws.geo"
  ]
}
```

> **Note**
>
> 同一個事件可以傳遞一次以上。您可以使用事件 ID 來刪除接收到的事件的重複資料。

Amazon 定 Location Service 的 Amazon EventBridge 事件示例

以下是輸入通過調用發起的地理圍欄的事件的示例。`BatchUpdateDevicePosition`

```
{
  "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"
    }
  }
}
```

以下是用於退出通過調用發起的地理圍欄的事件的示例。BatchUpdateDevicePosition

```
{
  "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"
    }
  }
}
```

以下是透過呼叫起始的職位更新事件範例BatchUpdateDevicePosition。

```
{
  "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"
    }
  }
}
```

使用 AWS IoT 和使用 Amazon Location Service MQTT 進行追蹤

MQTT是專為受限設備設計的輕量級且廣泛採用的消息傳遞協議。 AWS IoT Core 支援使用MQTT通訊協定和 MQTT WebSocket Secure (WSS) 通訊協定的裝置連線。

AWS IoT Core將設備連接到 AWS 並使您能夠在它們之間發送和接收消息。 AWS IoT Core 規則引擎會儲存有關裝置訊息主題的查詢，並可讓您定義將訊息傳送至其他 AWS 服務的動作，例如 Amazon 定 Location Service。將其位置視為座標的裝置，可透過規則引擎將其位置轉送到 Amazon 位置。

> **Note**
>
> 設備可能知道自己的位置，例如通過內置GPS。 AWS IoT 還支持第三方設備位置跟踪。如需詳細資訊，請參閱AWS IoT 核心開發人員指南中的AWS IoT 核心裝置位置。

以下逐步解說說明使用 AWS IoT Core 規則進行追蹤。如果您需要在傳送至 Amazon 位置之前處理裝置資訊，也可以將裝置資訊傳送至您自己的 AWS Lambda 功能。如需使用 Lambda 處理裝置位置的詳細資訊，請參閱AWS Lambda 搭配使用 MQTT。

主題

- 先決條件
- 建立 AWS IoT Core 規則
- 在主控台中測試 AWS IoT Core 規則
- AWS Lambda 搭配使用 MQTT

先決條件

您必須先完成下列先決條件，才能開始追蹤：

- 創建一個跟踪器資源，您將向其發送設備位置數據。
- 創建一個IAM角色以授予跟踪器的 AWS IoT Core 訪問權限。

  按照這些步驟進行操作時，請使用以下策略授予跟踪器的訪問權限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
}
```

建立 AWS IoT Core 規則

接下來，建立 AWS IoT Core 規則，將裝置的位置遙測轉寄至 Amazon 定 Location Service。如需有關建立規則的詳細資訊，請參閱AWS IoT Core 開發人員指南中的下列主題：

- 建立 AWS IoT 規則以取得有關建立新規則的資訊。
- 適用於建立發佈到 Amazon 位置之規則的特定資訊的位置動作

在主控台中測試 AWS IoT Core 規則

如果目前沒有裝置發佈包含位置的遙測，您可以使用 AWS IoT Core 主控台測試規則。控制台具有測試客戶端，您可以在其中發布示例消息以驗證解決方案的結果。

1. 請在以下位置登入 AWS IoT Core 主控台https://console.aws.amazon.com/iot/。
2. 在左側導覽列中，展開 [測試]，然後選擇 [MQTT測試用戶端]。
3. 在「發佈至主題」下，將主題名稱設定為 *iot/topic* (或您在 AWS IoT Core 規則中設定的主題名稱 (如果不同)，並為「訊息」承載提供下列資訊。

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. 選擇「發佈至主題」以傳送測試訊息。
5. 若要驗證 Amazon 定 Location Service 是否收到訊息，請使用下列 AWS CLI 命令。如果您在設定期間對其進行修改，請將追蹤器名稱取代為您使用的名稱。

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

AWS Lambda 搭配使用 MQTT

雖然將裝置位置資料傳送至 Amazon 位置進行追蹤時，不再需要使 AWS Lambda 用，但在某些情況下，您可能仍希望使用 Lambda。例如，如果您希望自己處理裝置位置資料，請在將裝置位置資料傳送至 Amazon 位置之前。下列主題說明如何使用 Lambda 在將訊息傳送至追蹤器之前處理訊息。如需有關此模式的詳細資訊，請參閱參考架構。

主題

- 先決條件
- 建立 Lambda 函數
- 建立 AWS IoT Core 規則
- 在主控台中測試 AWS IoT Core 規則

先決條件

您必須先建立追蹤器資源，才能開始追蹤。若要建立追蹤器資源，您可以使用 Amazon 位置主控台 AWS CLI、或 Amazon 位置APIs。

下列範例使用 Amazon 定 Location Service 主控台建立追蹤器資源：

1. 開啟 Amazon 定 Location Service 主控台，網址為https://console.aws.amazon.com/location/。
2. 在左側導覽窗格中，選擇「追蹤器」。
3. 選擇建立追蹤器。
4. 填寫下列方塊：
   - 名稱 — 輸入最多 100 個字元的唯一名稱。有效的項目包括英數字元、連字號和底線。例如 *MyTracker*.
   - 說明 — 輸入選擇性說明。例如 *Tracker for storing AWS IoT Core device positions*.
   - 位置篩選 — 選取您要用於位置更新的篩選。例如，基於準確度的過濾。
5. 選擇建立追蹤器。

建立 Lambda 函數

若要建立 AWS IoT Core 和 Amazon 定 Location Service 之間的連線，您需要一個 AWS Lambda 函數來處理轉寄的訊息 AWS IoT Core。此功能將擷取任何位置資料，將其格式化為 Amazon 定 Location Service，然後透過 Amazon 位置追蹤器API提交。您可以透過 AWS Lambda 主控台建立此函數，也可以使用 AWS Command Line Interface (AWS CLI) 或 AWS Lambda APIs.

若要使用主控台建立 Lambda 函數，以便將職位更新發佈到 Amazon 位置：

1. 在開啟 AWS Lambda 主控台https://console.aws.amazon.com/lambda/。
2. 在左側導覽中，選擇 [功能]。
3. 選擇「建立函數」，並確定已選取「從頭開始作者」。
4. 填寫下列方塊：
   - 函數名稱 — 輸入函數的唯一名稱。有效的項目包括英數字元、連字號和底線 (不含空格)。例如 *MyLambda*.
   - 執行階段 — 選擇 *Python 3.8*.
5. 選擇建立函數。

6. 選擇 [程式碼] 索引標籤以開啟編輯器。
7. 使用下列項目覆寫中lambda_function.py的預留位置代碼，並以您建立為先決條件的追蹤器名稱取代指定給的值。TRACKER_NAME

```
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. 選擇部署以儲存更新的功能。
9. 選擇 Configuration (組態) 索引標籤。
10. 在「權限」區段中，選擇超連結的角色名稱，以將 Amazon Location Service 許可授與 Lambda 函數。
11. 在角色的 [摘要] 頁面中，選擇 [新增權限]，然後從下拉式清單中選取 [建立內嵌原則]。
12. 選擇索JSON引標籤，然後使用下列文件覆寫原則。這可讓您的 Lambda 函數更新由所有區域的所有追蹤器資源所管理的裝置位置。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
}
```

13. 選擇檢閱政策。
14. 輸入政策名稱。例如 *AmazonLocationTrackerWriteOnly*.
15. 選擇建立政策。

您可以視需要修改此函數程式碼，以適應您自己的裝置訊息結構描述。

建立 AWS IoT Core 規則

接下來，建立 AWS IoT Core 規則，將裝置的位置遙測轉寄至 AWS Lambda 功能，以便轉換並發佈至 Amazon 定 Location Service。提供的範例規則假設任何必要的裝置承載轉換都是由 Lambda 函數處理。您可以透過 AWS IoT Core 主控台、 AWS Command Line Interface (AWS CLI) 或建立此規則 AWS IoT Core APIs。

> **Note**
>
> 當 AWS IoT 主控台處理允許 AWS IoT Core 呼叫 Lambda 函數所需的權限時，如果您要從 AWS CLI 或建立規則SDK，則必須設定政策以授與權限 AWS IoT。

若要使 AWS IoT Core 用控制台建立

1. 請在以下位置登入 AWS IoT Core 主控台https://console.aws.amazon.com/iot/。
2. 在左側導覽中，展開「動作」，然後選擇「規則」。
3. 選擇 [建立規則] 以啟動新規則精靈。
4. 輸入規則的名稱和說明。
5. 對於規則查詢陳述式，請更新FROM屬性以參考至少有一個裝置正在發佈包含位置之遙測的主題。如果您正在測試解決方案，則不需要修改。

   ```
   SELECT * FROM 'iot/topic'
   ```
6. 在 [設定一或多個動作] 底下，選擇 [新增動作]。
7. 選取 [傳送訊息至 Lambda 函數]。
8. 選擇 Configure action (設定動作)。
9. 從清單中尋找並選取您的 Lambda 函數。
10. 選擇新增動作。
11. 選擇建立規則。

在主控台中測試 AWS IoT Core 規則

如果目前沒有裝置發佈包含位置的遙測，您可以使用 AWS IoT Core 主控台測試規則和此解決方案。控制台具有測試客戶端，您可以在其中發布示例消息以驗證解決方案的結果。

1. 請在以下位置登入 AWS IoT Core 主控台https://console.aws.amazon.com/iot/。
2. 在左側導覽列中，展開 [測試]，然後選擇 [MQTT測試用戶端]。
3. 在「發佈至主題」下，將主題名稱設定為 *iot/topic* (或您在 AWS IoT Core 規則中設定的主題名稱 (如果不同)，並為「訊息」承載提供下列資訊。替換時間戳 *1604940328* 具有過去 30 天內的有效時間戳記（超過 30 天的任何時間戳記都會被忽略）。

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. 選擇「發佈至主題」以傳送測試訊息。
5. 若要驗證 Amazon 定 Location Service 是否收到訊息，請使用下列 AWS CLI 命令。如果您在設定期間對其進行了修改，請將追蹤器名稱和裝置 ID 取代為您使用的名稱和裝置 ID。

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

管理您的地理圍欄收集資源

使用 Amazon 位置主控台、或 Amazon 位置 API 管理您的AWS CLI地理圍欄集合。

列出您的地理圍欄收集資源

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 來檢視您的AWS CLI地理圍欄收集清單：

Console

使用 Amazon 位置主控台檢視地理圍欄集合清單

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇地理圍欄集合。
3. 在「我的地理圍欄集合」下檢視地理圍欄集合的清單。

API

使用來自 Amazon 位置地理圍欄 API 的ListGeofenceCollections操作。

下列範例是取得帳戶中地理圍欄集合清單的 API 要求。AWS

```
POST /geofencing/v0/list-collections
```

以下為範例回應ListGeofenceCollections：

```
{
    "Entries": [
    {
        "CollectionName": "ExampleCollection",
        "CreateTime": 2020-09-30T22:59:34.142Z,
        "Description": "string",
        "UpdateTime": 2020-09-30T23:59:34.142Z
    },
    "NextToken": "1234-5678-9012"
}
```

CLI

使用 list-geofence-collections 命令。

下列範例是取AWS CLI得帳戶中的地理圍欄集合清單。AWS

```
aws location list-geofence-collections
```

獲取地理圍欄收集詳細信息

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，取得AWS帳戶中任何地理圍欄收集資源的AWS CLI詳細資訊：

Console

使用 Amazon 位置主控台檢視地理圍欄收集的詳細資訊

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇地理圍欄集合。
3. 在「我的地理圍欄集合」下，選取目標地理圍欄集合的名稱連結。

API

使用來自 Amazon 位置地理圍欄 API 的DescribeGeofenceCollection操作。

下列範例是取得地理圍欄集合詳細資訊的 API 要求。*ExampleCollection*

```
GET /geofencing/v0/collections/ExampleCollection
```

以下為範例回應DescribeGeofenceCollection：

```
{
    "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

使用 describe-geofence-collection 命令。

下面的例子是一個獲AWS CLI取地理圍欄集合的詳細信息。*ExampleCollection*

```
aws location describe-geofence-collection \
    --collection-name "ExampleCollection"
```

刪除地理圍欄集合

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，從AWS帳戶刪除AWS CLI地理圍欄集合。

Console

若要使用 Amazon 位置主控台刪除地理圍欄集合

> ⚠ Warning
>
> 此作業會永久刪除資源。

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇地理圍欄集合。
3. 在「我的地理圍欄集合」下，選取目標地理圍欄集合。
4. 選擇 [刪除地理圍欄集合]。

API

使用來自 Amazon 位置 API 的DeleteGeofenceCollection操作。

下列範例是刪除地理圍欄集合的 API 要求。*ExampleCollection*

```
DELETE /geofencing/v0/collections/ExampleCollection
```

以下為範例回應DeleteGeofenceCollection：

```
HTTP/1.1 200
```

CLI

使用 delete-geofence-collection 命令。

下列範例是刪除地理圍欄集合的AWS CLI命令。*ExampleCollection*

```
aws location delete-geofence-collection \
    --collection-name "ExampleCollection"
```

列出存儲的地理圍欄

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 列出存放在指定地理圍欄集合中的AWS CLI地理圍欄。

Console

使用 Amazon 位置主控台檢視地理圍欄清單

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇地理圍欄集合。
3. 在「我的地理圍欄集合」下，選取目標地理圍欄集合的名稱連結。
4. 在地理圍欄下查看地理圍欄集合中的地理圍欄

API

使用來自 Amazon 位置地理圍欄 API 的ListGeofences操作。

下面的例子是一個 API 請求，用於獲取存儲在地理圍欄集合中的地理圍欄列表。*ExampleCollection*

```
POST /geofencing/v0/collections/ExampleCollection/list-geofences
```

以下為範例回應ListGeofences：

```
{
   "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

使用 `list-geofences` 命令。

下面的例子是一個獲AWS CLI取存儲在地理圍欄集合中的地理圍欄列表。*ExampleCollection*

```
aws location list-geofences \
    --collection-name "ExampleCollection"
```

取得地理圍欄詳細資訊

您可以使用 Amazon 位置主控台或 Amazon 位置 API，從地理圍欄收集取得特定地理圍欄的詳細資訊，例如建立時間、更新時間AWS CLI、幾何圖形和狀態。

Console

使用 Amazon 位置主控台檢視地理圍欄的狀態

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇地理圍欄集合。
3. 在「我的地理圍欄集合」下，選取目標地理圍欄集合的名稱連結。
4. 在地理圍欄下，您將能夠查看地理圍欄的狀態。

API

使用來自 Amazon 位置地理圍欄 API 的`GetGeofence`操作。

以下示例是從地理圍欄集合獲取地理圍欄詳細信息的 API 請求。*ExampleCollection*

```
GET /geofencing/v0/collections/ExampleCollection/geofences/ExampleGeofence1
```

以下為範例回應`GetGeofence`：

```
{
    "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

使用 get-geofence 命令。

下面的例子是一個獲AWS CLI取地理圍欄集合的詳細信息。*ExampleCollection*

```
aws location get-geofence \
    --collection-name "ExampleCollection" \
    --geofence-id "ExampleGeofence1"
```

刪除地理圍欄

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，從地理圍欄集合刪除AWS CLI地理圍欄。

Console

使用 Amazon 位置控制台刪除地理圍欄

> **Warning**
> 此作業會永久刪除資源。

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇地理圍欄集合。
3. 在「我的地理圍欄集合」下，選取目標地理圍欄集合的名稱連結。
4. 在「地理圍欄」下，選取目標地理圍欄。

5. 選擇 [刪除地理圍欄]。

API

使用來自 Amazon 位置地理圍欄 API 的BatchDeleteGeofence操作。

下列範例是從地理圍欄集合中刪除地理圍欄的 API 要求。*ExampleCollection*

```
POST /geofencing/v0/collections/ExampleCollection/delete-geofences
Content-type: application/json

{
   "GeofenceIds": [ "ExampleGeofence11" ]
}
```

以下是的成功回應範例BatchDeleteGeofence。

```
HTTP/1.1 200
```

CLI

使用 batch-delete-geofence 命令。

下面的例子是從地理圍欄集合刪除地理圍欄的AWS CLI命令。*ExampleCollection*

```
aws location batch-delete-geofence \
    --collection-name "ExampleCollection" \
    --geofence-ids "ExampleGeofence11"
```

管理您的追蹤器資源

您可以使用 Amazon 位置主控台AWS CLI、或 Amazon 位置 API 來管理追蹤器。

列出您的追蹤器

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 來AWS CLI檢視追蹤器清單：

Console

使用 Amazon 位置主控台檢視現有追蹤器清單

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽列中選擇「追蹤器」。
3. 在「我的追蹤器」下方檢視追蹤器資源清單。

API

使用 Amazon 位置跟踪器 API 中的ListTrackers操作。

以下示例是一個 API 請求，用於獲取您AWS帳戶中的跟踪器列表。

```
POST /tracking/v0/list-trackers
```

以下為範例回應ListTrackers：

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-02T19:09:07.327Z,
         "Description": "string",
         "TrackerName": "ExampleTracker",
         "UpdateTime": 2020-10-02T19:10:07.327Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

使用 list-trackers 命令。

以下示例是獲AWS CLI取您AWS帳戶中的跟踪器列表。

```
aws location list-trackers
```

斷開跟踪器與地理圍欄集合的連接

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 中斷追蹤器與地理圍欄集合的連線：AWS CLI

Console

使用 Amazon 位置主控台將追蹤器與關聯的地理圍欄集合取消關聯

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇「追蹤器」。
3. 在我的追蹤器下，選擇目標追蹤器的名稱連結。
4. 在「連結的地理圍欄集合」下，選取具有「已連結」狀態的地理圍欄集合。
5. 選擇「取消連結」。

API

使用 Amazon 位置跟踪器 API 中的DisassociateTrackerConsumer操作。

下列範例是將追蹤器與關聯地理圍欄集合取消關聯的 API 要求。

```
DELETE /tracking/v0/trackers/ExampleTracker/consumers/arn:aws:geo:us-
west-2:123456789012:geofence-collection/ExampleCollection
```

以下為範例回應DisassociateTrackerConsumer：

```
HTTP/1.1 200
```

CLI

使用 disassociate-tracker-consumer 命令。

下列範例是將追蹤器與關聯的地理圍欄集合取消關聯的AWS CLI命令。

```
aws location disassociate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-collection/
ExampleCollection" \
    --tracker-name "ExampleTracker"
```

取得追蹤器詳細

您可以使用 Amazon 位置主控台、或 Amazon 位置 API，取得AWS帳戶中任何追蹤器的詳細資訊。AWS CLI

Console

使用 Amazon 位置主控台檢視追蹤器詳細資料

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽列中選擇「追蹤器」。
3. 在我的追蹤器下，選擇目標追蹤器的名稱連結。
4. 在資訊下檢視追蹤器詳細資訊。

API

使用 Amazon 位置追蹤器 API 中的DescribeTracker操作。

下列範例是取得追蹤器詳細資料的 API 要求*ExampleTracker*。

```
GET /tracking/v0/trackers/ExampleTracker
```

以下為範例回應DescribeTracker：

```
{
   "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

使用 describe-tracker 命令。

下列範例是取得追蹤器詳細資訊的AWS CLI命令*ExampleTracker*。

```
aws location describe-tracker \
    --tracker-name "ExampleTracker"
```

刪除追蹤器

您可以使用 Amazon 位置主控台、或 Amazon 位置 API 從AWS帳戶刪除追蹤器：AWS CLI

Console

使用 Amazon 位置主控台刪除現有的地圖資源

> **Warning**
>
> 此作業會永久刪除資源。如果追蹤器資源正在使用中，您可能會遇到錯誤。請確定目標資源不是應用程式的相依性。

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 從左側導覽窗格中選擇「追蹤器」。
3. 在我的追蹤器下，選擇目標追蹤器。
4. 選擇刪除追蹤器。

API

使用 Amazon 位置追蹤器 API 中的DeleteTracker操作。

下列範例是刪除追蹤器的 API 要求*ExampleTracker*。

```
DELETE /tracking/v0/trackers/ExampleTracker
```

以下為範例回應DeleteTracker：

```
HTTP/1.1 200
```

CLI

使用 delete-tracker 命令。

下列範例是刪除追蹤器的AWS CLI命令*ExampleTracker*。

```
aws location delete-tracker \
    --tracker-name "ExampleTracker"
```

地理圍欄和跟踪移動應用程序示例

本主題涵蓋旨在展示在行動應用程式中使用 Amazon 位置地理圍欄和追蹤器的主要功能的教學課程。這些應用程式展示了追蹤器和地理圍欄如何使用 Lambda AWS IoT 和 Amazon 位置功能的組合進行互動。有兩個教程可用。

- 適用於 Android 的示例跟踪和地理圍欄應用程序，您可以從 GitHub以下位置克隆項目文件：/樹/主/。https://github.com/aws-geospatial/ amazon-location-samples-android tracking-with-geofence-notifications
- 適用於 iOS 的示例跟踪和地理圍欄應用程序，您可以從 GitHub以下位置克隆項目文件：/樹/主/。https://github.com/aws-geospatial/ amazon-location-samples-ios tracking-with-geofence-notifications

適用於 Android 的樣本跟踪和地理圍欄應用程序

本主題涵蓋了旨在演示在移動應用程序中使用 Amazon 位置地理圍欄和跟踪器的主要功能的 Android 教程。這些應用程序演示了跟踪器和地理圍欄如何使用 Lambda AWS IoT 和 Amazon 位置功能的組合進行交互。

主題

- 為您的應用程式建立 Amazon 位置資源
- 建立地理圍欄集合
- 將追蹤器連結至地理圍欄集合
- 搭配 M AWS QTT 使用 Lambda
- 設定範例應用程式程式碼
- 使用範例應用程式

為您的應用程式建立 Amazon 位置資源

首先，您需要創建所需的 Amazon 位置資源。這些資源對於應用程序的功能和執行提供的代碼片段至關重要。

> **Note**
>
> 如果您尚未建立 AWS 帳戶，請依照AWS 帳戶管理使用手冊中的指示操作。

若要開始，您需要建立 Amazon Cognito 身分識別集區 ID，請使用下列程序：

1. 開啟 Amazon Cognito 主控台，然後從左側功能表中選取身分集區，然後選取建立身分集區。
2. 確保選中訪客訪問權限，然後按下一步繼續。
3. 接下來，建立新的 IAM 角色或使用現有的 IAM 角色。
4. 輸入身分集區名稱，並確保身分集區可以存取您要在下一個程序中建立的地圖和追蹤器的 Amazon 位置(geo)資源。

接下來，您需要在 AWS Amazon 位置控制台中創建和設置地圖樣式，請使用以下步驟：

1. 導覽至 Amazon 位置主控台的「地圖」區段，然後選取「建立地圖」。
2. 為新地圖提供「名稱」和「描述」。記錄您指定的名稱，如同稍後在自學課程中使用的名稱一樣。
3. 選擇地圖型式時，請考慮地圖資料提供者。如需詳細資訊，請參閱AWS 服務條款第 82 節。
4. 接受 Amazon 位置條款與條件，然後選取建立地圖，以完成地圖建立程序。

接下來，您需要在 Amazon 位置控制台中創建跟踪器，請使用以下步驟：

1. 在 Amazon 位置主控台中開啟「地圖」區段。
2. 選擇建立追蹤器。
3. 填寫必填欄位。記下跟踪器的名稱，因為它將在整個術語中被拒絕。
4. 在「位置篩選」欄位下，選擇最符合您要使用追蹤器資源的方式的選項。如果您未設定「位置篩選」，則預設設定為TimeBased。如需詳細資訊，請參閱追蹤器和 Amazon 位置 API 參考PositionFiltering中的。
5. 選擇「建立追蹤器」以完成追蹤器的建立。

建立地理圍欄集合

現在，您將創建一個地理圍欄集合。您可以使用控制台、API 或 CLI。下列程序會逐步引導您完成每個選項。

- 使用 Amazon 位置主控台建立地理圍欄集合：
  1. 打開 Amazon 位置控制台的地理圍欄集合部分。
  2. 選擇 [建立地理圍欄集合]。
  3. 提供集合的名稱和描述。

4. 在以 Amazon CloudWatch 做為目標的 EventBridge 規則下，您可以建立選擇性 EventBridge 規則，以開始對地理圍欄事件做出反應。這使 Amazon 位置能夠將事件發佈到 Amazon CloudWatch Logs.
5. 按「建立地理圍欄」集合以完成建立集合。

- 使用 Amazon 定位 API 建立地理圍欄集合：

  使用來自 Amazon 位置地理圍欄 API 的CreateGeofenceCollection操作。下列範例會使用 API 要求來建立名為的地理圍欄集合。*GEOCOLLECTION_NAME*

```
POST /geofencing/v0/collections
Content-type: application/json

{
   "CollectionName": "GEOCOLLECTION_NAME",
   "Description": "Geofence collection 1 for shopping center",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

- 使用 AWS CLI 命令建立地理圍欄集合：

  使用 `create-geofence-collection` 命令。下列範例會使用 AWS CLI 建立名為的地理圍欄集合。*GEOCOLLECTION_NAME*如需有關使用 AWS CLI 的詳細資訊，請參閱命AWS 令列介面文件。

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --tags Tag1=Value1
```

將追蹤器連結至地理圍欄集合

若要將追蹤器連結至地理圍欄集合，您可以使用主控台、API 或 CLI。下列程序會逐步引導您完成每個選項。

使用 Amazon 定位服務主控台將追蹤器資源連結至 Location Service 圍欄集合：

1. 打開 Amazon 位置控制台。
2. 在左側導覽窗格中，選擇「追蹤器」。

3. 在「裝置追蹤器」下，選取目標追蹤器的名稱連結。
4. 在「連結的地理圍欄集合」下，選擇「連結地理圍欄集合」。
5. 在「連結的地理圍欄集合」視窗中，從下拉式選單中選取地理圍欄集合。
6. 選擇 Link (連結)。
7. 在您連結追蹤器資源之後，系統會為其指派「作用中」狀態。

使用 Amazon 位置 API 將追蹤器資源連結至地理圍欄集合：

使用 Amazon 位置追蹤器 API 中的`AsssociateTrackerConsumer`操作。下列範例使用 API 請求，該要求使用其 Amazon 資源名稱 (ARN) 將某個地理圍欄集合關聯`ExampleTracker`起來。

```
        POST /tracking/v0/trackers/ExampleTracker/consumers
        Content-type: application/json
        {
           "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME"
            }
```

使用 AWS CLI 命令將跟踪器資源鏈接到地理圍欄集合：

使用 `associate-tracker-consumer` 命令。下列範例會使用 AWS CLI 建立名為的地理圍欄集合。*GOECOLLECTION_NAME*

```
aws location \
associate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME" \
    --tracker-name "ExampleTracker"
```

搭配 M AWS QTT 使用 Lambda

為了建立 AWS IoT 和 Amazon 位置之間的連線，您需要 Lambda 函數來處理 EventBridge CloudWatch 事件轉寄的訊息。此功能將擷取任何位置資料，將其格式化為 Amazon 位置，然後透過 Amazon 位置追蹤器 API 提交。

下列程序說明如何透過 Lambda 主控台建立此函數：

1. 開啟 主控台。
2. 在左側導覽中，選擇 [功能]。
3. 然後選擇「建立函數」，並確定已選取「從頭開始作者」選項。
4. 提供「函數」名稱，並針對「執行階段」選項選擇 Node.js 16.x。
5. 選擇建立函數。
6. 開啟 [程式碼] 索引標籤以存取編輯器。
7. 使用下列指令覆寫index.js檔案中的預留位置代碼：

```
            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. 按部署保存更新的功能。
9. 接下來打開配置選項卡。
10. 在「觸發器」區段中，按下「新增觸發器」按鈕。
11. 在「來源」欄位中選取 EventBridge (CloudWatch 事件)。
12. 選取「現有規則」選項。
13. 輸入規則名稱，例如AmazonLocationMonitor-GEOFENCECOLLECTION_NAME。
14. 按下「新增」按鈕。
15. 這也將在權限選項卡中附加基於資源的策略語句

現在，您將使用設置 MQTT 測試客戶端 AWS IoT，請使用以下步驟：

1. 打開網站。https://console.aws.amazon.com/iot/
2. 在左側導覽窗格中，選取 MQTT 測試用戶端。
3. 您將看到一個標題為 MQTT 測試客戶端的部分，您可以在其中配置 MQTT 連接。
4. 配置必要的設置後，單擊「Connec t」按鈕以使用提供的參數建立與 MQTT 代理的連接。
5. 記錄端點，因為它稍後在管理中使用。

   連接到測試用戶端後，您可以訂閱 MQTT 主題或使用 MQTT 測試用戶端介面中提供的相應輸入欄位，將訊息發佈到主題。接下來，您將建立 AWS IoT 策略。

6. 在左側菜單中，在管理下展開安全選項，然後單擊策略。
7. 點擊創建策略按鈕。
8. 輸入政策名稱。
9. 在政策文件上，選取 JSON 索引標籤。
10. 複製粘貼下面顯示的策略，但請確保使用您的*REGION*和更新所有元素*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. 選取「建立」按鈕以完成作業。

完成上一個程序之後，您現在將更新 Guest 角色的權限，如下所示：

1. 瀏覽至 Amazon Cognito 並開啟您的身分集區。然後，繼續使用者存取並選取 Guest 角色。
2. 按一下權限原則以啟用編輯功能。

```
{
    '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. 隨著上述原則的變更，所有必要的 AWS 資源現在都會針對應用程式進行適當的設定。

設定範例應用程式程式碼

1. 克隆此存儲庫：https://github.com/aws-geospatial/ amazon-location-samples-android /樹/主/到您的本地計算tracking-with-geofence-notifications機。
2. 在安卓工作室中打開AmazonSampleSDKApp項目。
3. 在您的 Android 設備或模擬器上構建並運行該應用程序。

使用範例應用程式

若要使用範例，請遵循下列程序：

- 創建一個 **custom.properties**：

  若要設定custom.properties檔案，請依照下列步驟執行：

  1. 開啟您偏好的文字編輯器或 IDE。
  2. 建立新檔案。
  3. 儲存檔案，並將其命名為 custom.properties。
  4. custom.properties使用下列程式碼範例更新，並以您的MQTT_END_POINT資源名稱取代GEOFENCE_COLLECTION_NAME、、和TOPIC_TRACKER：POLICY_NAME

     ```
     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. 清理並重建項目。在此之後，您可以運行該項目。
- 登入:

要登錄該應用程序，請按照以下步驟操作：

1. 按「登入」按鈕。
2. 提供識別集區 ID、追蹤器名稱和對應名稱。
3. 再次按「登入」即可完成操作。

- 管理篩選器：

  開啟設定畫面，然後執行下列動作：

  1. 使用切換 UI 切換過濾器的開啟或關閉。
  2. 視需要更新時間和距離篩選條件。

- 追蹤作業：

  開啟追蹤畫面並執行下列動作：

  - 您可以按下相應的按鈕，以前景、背景或省電模式開始和停止追蹤。

適用於 iOS 的樣本跟踪和地理圍欄應用程序

本主題涵蓋 iOS 教學課程，旨在示範在行動應用程式中使用 Amazon 位置地理圍欄和追蹤器的主要功能。這些應用程序演示了跟踪器和地理圍欄如何使用 Lambda AWS IoT 和 Amazon 位置功能的組合進行交互。

主題

- 為您的應用程式建立 Amazon 位置資源
- 建立地理圍欄集合
- 將追蹤器連結至地理圍欄集合
- 搭配 M AWS QTT 使用 Lambda
- 設定範例應用程式程式碼
- 使用範例應用程式

為您的應用程式建立 Amazon 位置資源

首先，您需要創建所需的 Amazon 位置資源。這些資源對於應用程序的功能和執行提供的代碼片段至關重要。

> **Note**
>
> 如果您尚未建立 AWS 帳戶，請依照AWS 帳戶管理使用手冊中的指示操作。

若要開始，您需要建立 Amazon Cognito 身分識別集區 ID，請使用下列程序：

1. 開啟 Amazon Cognito 主控台並從左側功能表中選取身分集區，然後選取建立身分集區。
2. 確保選中訪客訪問權限，然後按下一步繼續。
3. 接下來，建立新的 IAM 角色或使用現有的 IAM 角色。
4. 輸入身分集區名稱，並確保身分集區可以存取您要在下一個程序中建立的地圖和追蹤器的 Amazon 位置(geo)資源。

接下來，您需要在 AWS Amazon 位置控制台中創建和設置地圖樣式，請使用以下步驟：

1. 導覽至 Amazon 位置主控台的「地圖」區段，然後選取「建立地圖」。
2. 為新地圖提供「名稱」和「描述」。記錄您指定的名稱，如同稍後在自學課程中使用的名稱一樣。
3. 選擇地圖型式時，請考慮地圖資料提供者。詳情請參閱AWS 服務條款第 82 節。
4. 接受 Amazon 位置條款與條件，然後選取建立地圖以完成地圖建立程序。

接下來，您需要在 Amazon 位置控制台中創建跟踪器，請使用以下步驟：

1. 在 Amazon 位置主控台中開啟「地圖」區段。
2. 選擇建立追蹤器。
3. 填寫必填欄位。記下跟踪器的名稱，因為它將在整個術語中被拒絕。
4. 在「位置篩選」欄位下，選擇最適合您使用追蹤器資源的選項。如果您未設定「位置篩選」，則預設設定為`TimeBased`。如需詳細資訊，請參閱開始追蹤和 Amazon PositionFiltering位置 API 參考中的。
5. 選擇「建立追蹤器」以完成追蹤器的建立。

建立地理圍欄集合

現在，您將創建一個地理圍欄集合。您可以使用控制台、API 或 CLI。下列程序會逐步引導您完成每個選項。

- 使用 Amazon 位置主控台建立地理圍欄集合：

  1. 打開 Amazon 位置控制台的地理圍欄集合部分。
  2. 選擇 [建立地理圍欄集合]。
  3. 提供集合的名稱和描述。
  4. 在 Amazon CloudWatch 作為目標的 EventBridge 規則下，您可以建立選擇性 EventBridge 規則以開始對地理圍欄事件做出反應。這使 Amazon 位置能夠將事件發佈到 Amazon CloudWatch Logs.
  5. 按「建立地理圍欄」集合以完成建立集合。

- 使用 Amazon 定位 API 建立地理圍欄集合：

  使用來自 Amazon 位置地理圍欄 API 的CreateGeofenceCollection操作。下列範例會使用 API 要求來建立名為的地理圍欄集合。*GEOCOLLECTION_NAME*

```
POST /geofencing/v0/collections
Content-type: application/json

{
   "CollectionName": "GEOCOLLECTION_NAME",
   "Description": "Geofence collection 1 for shopping center",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

- 使用 AWS CLI 命令建立地理圍欄集合：

  使用 `create-geofence-collection` 命令。下列範例使用 AWS CLI 建立名為的地理圍欄集合。*GEOCOLLECTION_NAME*如需有關使用 AWS CLI 的詳細資訊，請參閱命AWS 令列介面文件。

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --tags Tag1=Value1
```

將追蹤器連結至地理圍欄集合

若要將追蹤器連結至地理圍欄集合，您可以使用主控台、API 或 CLI。下列程序會逐步引導您完成每個選項。

使用 Amazon 定位服務主控台將追蹤器資源連結至 Location Service 圍欄集合：

1. 打開 Amazon 位置控制台。
2. 在左側導覽窗格中，選擇「追蹤器」。
3. 在「裝置追蹤器」下，選取目標追蹤器的名稱連結。
4. 在「連結的地理圍欄集合」下，選擇「連結地理圍欄集合」。
5. 在「連結的地理圍欄集合」視窗中，從下拉式選單中選取地理圍欄集合。
6. 選擇 Link (連結)。
7. 在您連結追蹤器資源之後，系統會為其指派「作用中」狀態。

使用 Amazon 位置 API 將追蹤器資源連結至地理圍欄集合：

使用 Amazon 位置追蹤器 API 中的`AsssociateTrackerConsumer`操作。下列範例使用 API 請求，該要求使用其 Amazon 資源名稱 (ARN) ExampleTracker 與地理圍欄集合相關聯。

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json
    {
       "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GEOCOLLECTION_NAME"
    }
```

使用 AWS CLI 命令將追蹤器資源連結至地理圍欄集合：

使用 `associate-tracker-consumer` 命令。下列範例使用 AWS CLI 建立名為的地理圍欄集合。*GEOCOLLECTION_NAME*

```
aws location \
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GEOCOLLECTION_NAME" \
        --tracker-name "ExampleTracker"
```

搭配 M AWS QTT 使用 Lambda

為了建立 AWS IoT 和 Amazon 位置之間的連線，您需要 Lambda 函數來處理 EventBridge CloudWatch 事件轉寄的訊息。此功能將擷取任何位置資料，將其格式化為 Amazon 位置，然後透過 Amazon 位置追蹤器 API 提交。

下列程序說明如何透過 Lambda 主控台建立此函數：

1. 開啟 主控台。
2. 在左側導覽中，選擇 [功能]。
3. 然後選擇「建立函數」，並確定已選取「從頭開始作者」選項。
4. 提供「函數」名稱，並針對「執行階段」選項選擇 Node.js 16.x。
5. 選擇建立函數。
6. 開啟 [程式碼] 索引標籤以存取編輯器。
7. 使用下列指令覆寫`index.js`檔案中的預留位置代碼：

```
            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. 按部署保存更新的功能。
9. 接下來打開配置選項卡。
10. 在「觸發器」區段中，按下「新增觸發器」按鈕。
11. 在「來源」欄位中選取 EventBridge (CloudWatch 事件)。
12. 選取「現有規則」選項。
13. 輸入規則名稱，例如`AmazonLocationMonitor-GEOFENCECOLLECTION_NAME`。
14. 按下「新增」按鈕。
15. 這也將在權限選項卡中附加基於資源的策略語句

現在，您將設置 AWS IoT MQTT 測試客戶端，請使用以下步驟：

1. 打開網站。https://console.aws.amazon.com/iot/

2. 在左側導覽窗格中，選取 MQTT 測試用戶端。
3. 您將看到一個標題為 MQTT 測試客戶端的部分，您可以在其中配置 MQTT 連接。
4. 配置必要的設置後，單擊「Connec t」按鈕以使用提供的參數建立與 MQTT 代理的連接。
5. 記錄端點，因為它稍後在管理中使用。

   連接到測試用戶端後，您可以使用 MQTT 測試用戶端介面中提供的相應輸入欄位訂閱 MQTT 主題或將訊息發佈至主題。接下來，您將建立 AWS IoT 策略。
6. 在左側菜單中，在管理下展開安全選項，然後單擊策略。
7. 點擊創建策略按鈕。
8. 輸入政策名稱。
9. 在政策文件上，選取 JSON 索引標籤。
10. 複製粘貼下面顯示的策略，但請確保使用您的*REGION*和更新所有元素*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. 選取「建立」按鈕以完成作業。

設定範例應用程式程式碼

若要設定範例程式碼，您必須安裝下列工具：

- Git
- XCode 15.3 或更高版本
- iOS 模擬器 16 或更高版本

使用以下程序來設定範例應用程式程式碼：

1. 從以下網址克隆 git 存儲庫：https://github.com/aws-geospatial/ amazon-location-samples-ios /樹/主/ tracking-with-geofence-notifications。
2. 開啟 AWSLocationSampleApp.xcodeproj 專案檔案。
3. 等待包解析過程完成。
4. 在項目導航菜單上，ConfigTemplate.xcconfig重命名為Config.xcconfig並填寫以下值：

```
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`
```

使用範例應用程式

設置示例代碼後，您現在可以在 iOS 模擬器或物理設備上運行該應用程序。

1. 建置並執行應用程式。
2. 該應用程序將要求您提供位置和通知權限。你需要允許他們。
3. 按下「Co gnito 組態」按鈕。
4. 儲存組態。
5. 現在，您可以看到時間，距離和精度的過濾器選項。根據您的需要使用它們。
6. 轉到應用程序中的「跟踪」選項卡，您將看到地圖和開始跟踪按鈕。
7. 如果您已在模擬器上安裝了該應用程序，則可能需要模擬位置更改。這可以在「位置」菜單選項下的功能中完成。例如，選取「功能」、「位置」、「高速公路磁碟」。
8. 按下開始追蹤按鈕。您應該在地圖上看到跟踪點。

9. 該應用程序還在後台跟踪位置。因此，當您在後台移動應用程序時，它將要求您允許以後台模式繼續跟踪。
10. 您可以通過點擊停止跟踪按鈕停止跟踪。

標記您的 Amazon Location Service 資源

使用 Amazon Location 中的資源標記建立標籤，依據用途、擁有者、環境或條件對資源進行分類。標記資源可協助您管理、識別、組織、搜尋和篩選資源。

例如，使用AWS Resource Groups，您可以根據一個或多個標籤或部分標籤來建立AWS資源群組。您也可以根據其在 AWS CloudFormation 堆疊中出現的次數建立群組。使用資源群組和標籤編輯器，您可以合併將多項服務、資源和區域集結在一處的應用程式資料，然後進行檢視。如需常見標記策略的詳細資訊，請參閱 AWS 一般參考。

每個標籤都是由您定義的鍵和值組成的標籤：

- 標籤鍵 — 對標籤值進行分類的一般標籤。例如 `CostCenter`。
- 標籤值 — 標籤關鍵品類的可選描述。例如 `MobileAssetTrackingResourcesProd`。

本主題可協助您透過檢閱標記限制開始使用標籤。它也會示範如何使用AWS成本分配報告來建立標籤，以及如何使用標籤來追蹤每個作用中標籤的成本。

主題

- 標記限制
- 授予標記資源的權限
- 將標籤新增至 Amazon 定 Location Service 資源
- 透過標籤追蹤資源成本
- 使用標籤控制對 Amazon 定 Location Service 資源的存取
- 進一步了解

標記限制

以下基本限制適用於標籤：

- 每個資源的標籤上限 — 50

- 對於每一個資源，每個標籤金鑰必須是唯一的，且每個標籤金鑰只能有一個值。

> **Note**
>
> 若您使用與現有標籤相同的標籤金鑰新增新的標籤，新的標籤會覆寫現有的標籤。

- 索引鍵長度上限 - 128 個 UTF-8 Unicode 字元
- 值的長度上限 - 256 個 UTF-8 Unicode 字元
- 服務間允許的字元包括：可用 UTF-8 表示的英文字母、數字和空格，還有以下字元：+ - = . _ : / @。
- 標籤金鑰與值皆區分大小寫。
- 此 aws: 字首已保留供 AWS 使用。如果標籤具有此字首的標籤金鑰，則您無法編輯或刪除標籤的金鑰或值。帶有aws:前綴的標籤不會計入每個資源限制的標籤。

授予標記資源的權限

您可以使用 IAM 政策來控制對 Amazon 位置資源的存取，並授予在建立時標記資源的權限。除了授與建立資源的權限之外，此原則還可以包含允許標記作業的Action權限：

- geo:TagResource— 允許使用者將一個或多個標籤指派給指定的 Amazon 位置資源。
- geo:UntagResource— 允許使用者從指定的 Amazon 位置資源中移除一或多個標籤。
- geo:ListTagsForResource— 允許使用者列出指派給 Amazon 位置資源的所有標籤。

以下是允許使用者建立地理圍欄集合和標記資源的政策範例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowTaggingForGeofenceCollectionOnCreation",
            "Effect": "Allow",
            "Action": [
                "geo:CreateGeofenceCollection",
                "geo:TagResource"
            ],
            "Resource": "arn:aws:geo:region:accountID:geofence-collection/*"
```

```
    ]
}
```

將標籤新增至 Amazon 定 Location Service 資源

您可以在使用 Amazon 位置主控台、或 Amazon 位置 API 建立資源時新增標籤：AWS CLI

- 建立地圖資源
- 建立位置索引資源
- 建立路線計算器資源
- 建立地理圍欄集合
- 建立追蹤器資源

若要標記現有資源，請編輯或刪除標籤

1. 在 https://console.aws.amazon.com/location/ 打開 Amazon 位置控制台。
2. 在左側導覽窗格中，選擇要標記的資源。例如，地圖。
3. 從清單中選擇資源。
4. 選擇「管理標籤」以新增、編輯或刪除標籤。

透過標籤追蹤資源成本

您可以使用標籤進行成本分配，以詳細追蹤您的AWS成本。啟動成本分攤標籤之後，AWS會使用成本配置標記，在成本分攤報表上整理資源帳單。這有助於您對使用成本進行分類和追蹤。

您可以啟動兩種類型的成本配置標籤：

- AWS-生成-這些標籤由AWS生成。 AWS標籤使用`aws:`前置詞，例如，`aws:createdBy`。
- 使用者定義 — 這些是您建立的自訂標籤。使用者定義的標籤會使用`user:`前置詞，例如，`user:CostCenter`。

您必須個別啟動每個標籤類型。啟用標籤後，您可以啟用AWS Cost Explorer或檢視每月費用分配報告。

AWS-generated tags

啟動 AWS 產生的標籤

1. 開啟 Billing and Cost Management 主控台，網址為 https://console.aws.amazon.com/billing/。
2. 在左側導覽窗格中，選擇「成本配置標記」。
3. 在「AWS產生的成本分配標籤」標籤下，選取您要啟動的標籤金鑰。
4. 選擇 Activate (啟用)。

User-defined tags

啟動使用者定義標籤

1. 開啟 Billing and Cost Management 主控台，網址為 https://console.aws.amazon.com/billing/。
2. 在左側導覽窗格中，選擇「成本配置標記」。
3. 在「使用者定義的成本配置標籤」標籤下，選取您要啟動的標籤金鑰。
4. 選擇 Activate (啟用)。

啟用標籤後，AWS會針對資源使用量和成本產生每月成本分配報告。此成本分配報表包含您每個帳單週期的所有AWS成本，包括標記和未標記的資源。如需詳細資訊，請參閱 AWS Billing and Cost Management 使用者指南中的使用成本分配標籤。

使用標籤控制對 Amazon 定 Location Service 資源的存取

AWS Identity and Access Management(IAM) 政策支援以標籤為基礎的條件，可讓您根據特定標籤金鑰和值來管理資源的授權。例如，IAM 角色政策可以包含條件，以根據標籤限制對特定環境的存取，例如開發、測試或生產環境。

如需詳細資訊，請參閱根據標籤控制資源存取的主題。

進一步了解

如需更多相關資訊：

- 標記最佳實務，請參閱 AWS 一般參考中的標記 AWS 資源。
- 使用標籤來控制對AWS資源的存取，請參閱《使用指南》中的〈使用標籤控制AWS資源的AWS Identity and Access Management存取〉。

授予對 Amazon 定 Location Service 的訪問

若要使用 Amazon 定 Location Service，使用者必須獲得組成 Amazon 位置的資源和 API 的存取權。您可以使用三種策略來授與資源的存取權。

- 使用 IAM — 若要將存取權授予透過AWS IAM Identity Center或 AWS Identity and Access Management (IAM) 驗證的使用者，請建立允許存取所需資源的 IAM 政策。如需 IAM 和 Amazon 位置的詳細資訊，請參閱Amazon 定 Location Service 務的 Identity and Access Management。
- 使用 API 金鑰 — 若要將存取權授予未經驗證的使用者，您可以建立 API 金鑰，以提供 Amazon 定 Location Service 資源的唯讀存取權。這在您不想驗證每個用戶的情況下非常有用。例如，一個 Web 應用程序。如需 API 金鑰的詳細資訊，請參閱允許未經驗證的訪客使用 API 金鑰存取您的應用程式。
- 使用亞馬遜認知 — API 金鑰的替代方法是使用 Amazon Cognito 授予匿名存取權。Amazon Cognito 可讓您透過政策建立更豐富的授權，以定義未經驗證的使用者可以執行的動作。如需使用 Amazon Cognito 的詳細資訊，請參閱允許未經驗證的訪客使用 Amazon Cognito 存取您的應用程式。

> **Note**
>
> 您也可以使用 Amazon Cognito 使用自己的身分驗證程序，或使用 Amazon Cognito 聯合身分來合併多種身份驗證方法。如需詳細資訊，請參閱 Amazon Cognito 開發人員指南中的聯合身分入門。

主題

- 允許未經驗證的訪客使用 API 金鑰存取您的應用程式
- 允許未經驗證的訪客使用 Amazon Cognito 存取您的應用程式

允許未經驗證的訪客使用 API 金鑰存取您的應用程式

當您在應用程式中呼叫 Amazon 定 Location Service API 時，通常會以經過驗證的使用者身分進行呼叫，並獲得授權進行 API 呼叫。但是，在某些情況下，您不想對應用程序的每個用戶進行身份驗證。例如，您可能希望使用該網站的任何人都可以使用顯示您的業務位置的 Web 應用程序，無論他們是否已登錄。在這種情況下，一種替代方法是使用 API 密鑰進行 API 調用。

API 金鑰是與您的特定 Amazon 定 Location Service 資源相關聯的索引鍵值 AWS 帳戶，以及您可以在這些資源上執行的特定動作。您可以在應用程式中使用 API 金鑰，對這些資源的 Amazon 位置 API 進

行未經驗證的呼叫。例如，如果您將 API 金鑰與地圖資源 MyMap 和`GetMap*`動作產生關聯，則使用該 API 金鑰的應用程式將能夠檢視使用該資源建立的地圖，而您的帳戶將按照您帳戶的任何其他用量收費。相同的 API 密鑰不會授予更改或更新地圖資源的權限-只允許使用資源。

> **Note**
>
> API 金鑰僅可用於對應、位置和路由資源，而且您無法修改或建立這些資源。如果您的應用程式需要存取未驗證使用者的其他資源或動作，您可以使用 Amazon Cognito 提供存取權以及 (或取代) API 金鑰。如需詳細資訊，請參閱 允許未經驗證的訪客使用 Amazon Cognito 存取您的應用程式。

API 金鑰包含純文字值，可讓您存取. AWS 帳戶如果有人複製您的 API 金鑰，他們就可以存取這些相同的資源。若要避免這種情況，您可以指定建立金鑰時可以使用 API 金鑰的網域。這些網域稱為「參照者」。如果需要，您還可以通過設置 API 密鑰的到期時間來創建短期 API 密鑰。

主題

- API 密鑰與 Amazon Cognito
- 建立 API 金鑰
- 使用 API 密鑰調用 Amazon 位置 API
- 使用 API 密鑰渲染地圖
- 管理 API 金鑰生命週期

API 密鑰與 Amazon Cognito

在類似的情況下，API 金鑰和 Amazon Cognito 的使用方式類似，那麼為什麼您要使用其中一個金鑰呢？ 下面的列表突出了兩者之間的一些差異。

- API 金鑰僅適用於地圖、位置和路由資源，而且僅適用於特定動作。Amazon Cognito 可用來驗證對大多數 Amazon 定 Location Service API 的存取。
- 使用 API 金鑰的對應請求效能通常會比使用 Amazon Cognito 的類似案例快。更簡單的身份驗證意味著在短時間內再次獲取相同的地圖磚時，對服務和緩存請求的往返次數更少。
- 透過 Amazon Cognito，您可以使用自己的身分驗證程序，或使用 Amazon Cognito 聯合身分來組合多種身分驗證方法。如需詳細資訊，請參閱 Amazon Cognito 開發人員指南中的聯合身分入門。

建立 API 金鑰

您可以建立 API 金鑰，並將其與 AWS 帳戶.

您可以使用 Amazon 定 Location Service 主控台 AWS CLI、或 Amazon 位置 API 建立 API 金鑰。

Console

使用 Amazon 定 Location Service 主控台建立 API 金鑰

1. 在 Amazon 位置主控台中，從左側功能表選擇 API 金鑰。
2. 在 [API 金鑰] 頁面上，選擇 [建立 API 金鑰]。
3. 在「建立 API 金鑰」頁面上，填寫下列資訊：
   - 名稱 — API 金鑰的名稱，例如`MyWebAppKey`。
   - 說明 — API 金鑰的選用說明。
   - 資源 — 從下拉式清單中選擇要使用此 API 金鑰存取權的 Amazon 位置資源。您可以選擇 [新增資源] 來新增多個資源。
   - 動作 — 指定您要使用此 API 金鑰授權的動作。您必須至少選取一個動作，才能符合您選取的每個資源類型。例如，如果您選取了位置資源，則必須在「置入動作」下選取至少一個選項。
   - 到期時間 — 選擇性地新增 API 金鑰的到期日期和時間。如需詳細資訊，請參閱 管理 API 金鑰生命週期。
   - 參照者 — 選擇性地新增一個或多個網域，您可以在其中使用 API 金鑰。例如，如果 API 密鑰是允許在網站上運行的應用程序`example.com`，那麼您可以將其`*.example.com/`作為允許的引用者。
   - 標籤 — 選擇性地將標籤新增至 API 金鑰。
4. 選擇「建立 API 金鑰」以建立 API 金鑰。
5. 在 API 金鑰的詳細資料頁面上，您可以查看已建立之 API 金鑰的相關資訊。選擇顯示 API 金鑰以查看您在呼叫 Amazon 位置 API 時使用的金鑰值。鍵值將具有格式`v1.public.a1b2c3d4...`。如需使用 API 金鑰呈現地圖的詳細資訊，請參閱使用 API 密鑰渲染地圖。

API

若要使用 Amazon 位置 API 建立 API 金鑰

使用來自 Amazon 位置 API 的CreateKey操作。

下列範例是一個 API 要求，用來建立名為沒*ExampleKey*有到期日的 API 金鑰，以及存取單一地圖資源。

```
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
   }
}
```

回應包括存取應用程式中資源時要使用的 API 金鑰值。鍵值將具有格式v1.public.*a1b2c3d4...*。若要進一步瞭解如何使用 API 金鑰來呈現地圖，請參閱使用 API 密鑰渲染地圖。

您也可以在稍後使用 DescribeKey API 尋找金鑰的金鑰值。

AWS CLI

若要使用 AWS CLI 指令建立 API 金鑰

使用 create-key 命令。

下列範例會建立呼叫的 API 金鑰，不*ExampleKey*含到期日，並可存取單一對應資源。

```
aws location \
  create-key \
  --key-name ExampleKey \
  --restrictions '{"AllowActions":["geo:GetMap*"],"AllowResources":
["arn:aws:geo:region:map/mapname"]}' \
  --no-expiry
```

回應包括存取應用程式中資源時要使用的 API 金鑰值。鍵值將具有格式`v1.public.a1b2c3d4...`。若要進一步瞭解如何使用 API 金鑰來呈現地圖，請參閱使用 API 密鑰渲染地圖。對的響應`create-key`如下所示。

```
{
    "Key": "v1.public.a1b2c3d4...",
    "KeyArn": "arn:aws:geo:region:accountId:api-key/ExampleKey",
    "KeyName": "ExampleKey",
    "CreateTime": "2023-02-06T22:33:15.693Z"
}
```

您也可以在稍後使用`describe-key`來尋找索引鍵值。下列範例顯示如何呼叫`describe-key`名為的 API 金鑰*ExampleKey*。

```
aws location describe-key \
    --key-name ExampleKey
```

使用 API 密鑰調用 Amazon 位置 API

建立 API 金鑰後，您可以使用金鑰值來呼叫應用程式中的 Amazon 位置 API。

支持 API 密鑰的 API 具有一個額外的參數，該參數採用 API 密鑰值。例如，如果您呼叫 GetPlace API，您可以填入 key 參數，如下所示

```
GET /places/v0/indexes/IndexName/places/PlaceId?key=KeyValue
```

如果您填入此值，則不需要像平常一樣使用 AWS Sigv4 驗證 API 呼叫。

對於 JavaScript 開發人員，您可以使用 Amazon 位置JavaScript 驗證助手協助使用 API 金鑰驗證 API 操作。

對於行動開發人員，您可以使用下列 Amazon 位置行動身份驗證開發套件：

- Amazon 定 Location Service 移動身份驗證 SDK for iOS
- Amazon 定 Location Service 安卓系統行動身份驗證

對於用 AWS CLI 戶，當您使用`--key`參數時，您還應該使用該`--no-sign-request`參數，以避免使用 Sig v4 進行簽名。

> **Note**
>
> 如果您在對 Amazon 定 key Location Service 的呼叫中同時包含和和 AWS Sig v4 簽名，則只會使用 API 金鑰。

使用 API 密鑰渲染地圖

您可以使用 API 金鑰值來呈現應用程式中的地圖 MapLibre。這與在您直接呼叫的其他 Amazon 位置 API 中使用 API 金鑰有點不同，因為會為您進行 MapLibre 這些呼叫。

下列範例程式碼示範如何使用 MapLibre GL JS 地圖控制項，使用 API 金鑰呈現簡單網頁中的地圖。為了使此代碼正常工作，請替換 *v1.public# your-api-key-value*、*us-east-1*，以及值*ExampleMap*符合您的. AWS 帳戶

```
<!-- 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>
```

管理 API 金鑰生命週期

您可以建立無限期運作的 API 金鑰。但是，如果您想要建立暫時 API 金鑰、定期輪換 API 金鑰或撤銷現有的 API 金鑰，您可以使用 API 金鑰到期時間。

建立新的 API 金鑰或更新現有的 API 金鑰時，您可以設定該 API 金鑰的到期時間。

- 當 API 金鑰達到其到期時間時，金鑰會自動停用。非作用中的金鑰無法再用來發出對應要求。
- 您可以在停用後 90 天刪除 API 金鑰。
- 如果您有尚未刪除的非作用中金鑰，您可以將到期時間更新為 future 時間來還原。
- 若要建立永久金鑰，您可以移除到期時間。
- 如果您嘗試停用過去 7 天內使用過的 API 金鑰，系統會提示您確認要進行變更。如果您使用的是 Amazon 定 Location Service API，或者 AWS CLI，您將收到錯誤訊息，除非您將`ForceUpdate`參數設定為 true。

允許未經驗證的訪客使用 Amazon Cognito 存取您的應用程式

您可以使用 Amazon Cognito 身份驗證作為直接將 AWS Identity and Access Management (IAM) 與前端開發套件和直接 HTTPS 請求搭配使用的替代方法。

您可能基於以下原因想要使用這種身份驗證形式：

- 未經驗證的使用者 — 如果您擁有匿名使用者的網站，您可以使用 Amazon Cognito 身分集區。如需詳細資訊，請參閱（詳見）一節the section called “使用 Amazon Cognito”。
- 您自己的身份驗證 — 如果您想要使用自己的身份驗證程序，或結合多種身份驗證方法，可以使用 Amazon Cognito 聯合身分。如需詳細資訊，請參閱 Amazon Cognito 開發人員指南中的聯合身分入門。

Amazon Cognito 為網頁和行動應用程式提供身份驗證、授權和使用者管理功能。您可以將 Amazon Cognito 未經驗證的身分集區與 Amazon 位置搭配使用，作為應用程式擷取臨時、速 AWS 度較低的登入資料的一種方式。

如需詳細資訊，請參閱 Amazon Cognito 開發人員指南中的使用者集區入門。

> **Note**
>
> 對於行動開發人員，Amazon 位置提供適用於 iOS 和 Android 的行動驗證 SDK，請參閱下列 github 儲存庫以取得詳細資訊：
>
> - Amazon 定 Location Service 移動身份驗證 SDK for iOS
> - Amazon 定 Location Service 安卓系統行動身份驗證

建立 Amazon Cognito 身分集區

您可以建立 Amazon Cognito 身分集區，以允許未經驗證的訪客透過 Amazon Cognito 主控台、或 Amazon Cognito API 存取 AWS CLI您的應用程式。

> **Important**
>
> 您建立的集 AWS 區必須 AWS 帳戶 與您正在使用的 Amazon 定位服務資源位於相同的區域。

您可以透過下列動作使用與未驗證身分角色相關聯的 IAM 政策：

- `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

包括其他 Amazon 位置動作將不會產生任何作用，且未經驗證的身分將無法呼叫它們。

Example

使用 Amazon Cognito 主控台建立身分集區

1. 前往 Amazon Cognito 主控台。
2. 選擇 Manage Identity Pools (管理身分集區)。
3. 選擇 [建立新身分集區]，然後輸入身分集區的名稱。
4. 從 [未驗證的身分可收合] 區段中，選擇 [啟用存取未驗證身分]。
5. 選擇 Create Pool (建立集區)。
6. 選擇要搭配身分集區使用的 IAM 角色。
7. 展開檢視詳細資料。
8. 在「未驗證的身分」下，輸入角色名稱。
9. 展開「檢視原則文件」區段，然後選擇 「編輯」以新增原則。
10. 新增您的政策以授與資源的存取權。

    以下是「地圖」、「地點」、「追蹤器」和「路線」的政策範例。若要將範例用於您自己的政策，請取代*##*和 *accountID* 預留位置符號：

    Maps policy example

    下列策略授與名為的對映資源的唯讀存取權*ExampleMap*。

```
{
  "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"
    }
  ]
}
```

新增符合的 IAM 條件`aws:referer`可讓您將瀏覽器存取資源限制為 URL 或 URL 首碼清單。下列範例允許存取僅`RasterEsriImagery`從網站命名的地圖資源`example.com`：

> **Warning**
>
> 雖然`aws:referer`可以限制訪問，但它不是一個安全的機制。包含公開已知推薦者標頭值相當危險。未授權方可以使用修改的或自訂瀏覽器來提供他們選擇的任何 `aws:referer` 值。因此，不`aws:referer`應用於防止未經授權的人士提出直接 AWS 要求。它僅用於允許客戶保護其數位內容 (例如存放在 Amazon S3 中的內容)，使其不被未經授權的第三方網站參考。有關更多信息，請參閱：裁判員。AWS

```
{
    "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/*"
                    ]
                }
            }
        }
    ]
}
```

如果您使用七巧板顯示地圖，則不會使用地圖 API 傳回的樣式描述元、字符或精靈。而是透過指向包含樣式規則和必要資產的 .zip 檔案進行配置。下列策略授與*ExampleMap*為GetMapTile作業命名的對映資源的唯讀存取權。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MapsReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:GetMapTile"
      ],
      "Resource": "arn:aws:geo:region:accountID:map/ExampleMap"
    }
  ]
}
```

Places policy example

下列策略授與名*ExamplePlaceIndex*為的地點索引資源的唯讀存取權，該資源可依文字或位置搜尋位置。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PlacesReadOnly",
      "Effect": "Allow",
      "Action": [
          "geo:SearchPlaceIndex*",
          "geo:GetPlace"
      ],
      "Resource": "arn:aws:geo:region:accountID:place-index/ExamplePlaceIndex"
    }
  ]
}
```

新增符合的 IAM 條件`aws:referer`可讓您將瀏覽器存取資源限制為 URL 或 URL 首碼清單。下列範例會拒絕存取*ExamplePlaceIndex*從所有反向連結網站命名的地點索引資源 (除`example.com`外)。

> **Warning**
>
> 雖然`aws:referer`可以限制訪問，但它不是一個安全的機制。包含公開已知推薦者標頭值相當危險。未授權方可以使用修改的或自訂瀏覽器來提供他們選擇的任何 `aws:referer` 值。因此，不`aws:referer`應用於防止未經授權的人士提出直接 AWS 要求。它僅用於允許客戶保護其數位內容 (例如存放在 Amazon S3 中的內容)，使其不被未經授權的第三方網站參考。有關更多信息，請參閱：裁判員。AWS

```
{
    "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

以下策略授予對名為*ExampleTracker*更新設備位置的跟踪器資源的訪問權限。

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

```
    {
      "Sid": "UpdateDevicePosition",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": "arn:aws:geo:region:accountID:tracker/ExampleTracker"
    }
  ]
}
```

新增符合的 IAM 條件`aws:referer`可讓您將瀏覽器存取資源限制為 URL 或 URL 首碼清單。下列範例會拒絕存取*ExampleTracker*從所有反向連結網站命名的追蹤器資源 (除`example.com`外)。

> **Warning**
>
> 雖然`aws:referer`可以限制訪問，但它不是一個安全的機制。包含公開已知推薦者標頭值相當危險。未授權方可以使用修改的或自訂瀏覽器來提供他們選擇的任何 `aws:referer` 值。因此，不`aws:referer`應用於防止未經授權的人士提出直接 AWS 要求。它僅用於允許客戶保護其數位內容 (例如存放在 Amazon S3 中的內容)，使其不被未經授權的第三方網站參考。有關更多信息，請參閱：裁判員。AWS

```
{
    "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

以下策略授予存取名為*ExampleCalculator*計算路由的路由計算器資源。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:region:accountID:route-
calculator/ExampleCalculator"
    }
  ]
}
```

新增符合的 IAM 條件`aws:referer`可讓您將瀏覽器存取資源限制為 URL 或 URL 首碼清單。下列範例會拒絕存取*ExampleCalculator*從所有參照網站命名的路線計算器 (除`example.com`外)。

> **Warning**
>
> 雖然`aws:referer`可以限制訪問，但它不是一個安全的機制。包含公開已知推薦者標頭值相當危險。未授權方可以使用修改的或自訂瀏覽器來提供他們選擇的任何 `aws:referer` 值。因此，不`aws:referer`應用於防止未經授權的人士提出直接 AWS 要求。它僅用於允許客戶保護其數位內容 (例如存放在 Amazon S3 中的內容)，使其不被未經授權的第三方網站參考。有關更多信息，請參閱：裁判員。AWS

```
{
    "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**
>
> 雖然未驗證身分集區的目的是在不安全的網際網路網站上曝光，但請注意，這些身分集區會被交換成標準、有時間限制 AWS 的認證。
> 適當範圍與未驗證身分集區相關聯的 IAM 角色非常重要。

11. 選擇 [允許] 建立您的身分集區。

產生的身分集區遵循語法 <*region*>:<*GUID*>.

例如：

```
us-east-1:1sample4-5678-90ef-aaaa-1234abcd56ef
```

如需 Amazon 位置特定的更多政策範例，請參閱the section called “身分型政策範例”。

在中使用 Amazon Cognito 身份集區 JavaScript

下列範例會交換您為憑證建立的未驗證身分集區，然後用來擷取地圖資源*ExampleMap*的樣式描述元。

```
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**
>
> 從未驗證身分擷取的認證有效期為一小時。

以下是在認證過期之前自動更新的函數範例。

```
async function refreshCredentials() {
  await credentials.refreshPromise();
  // schedule the next credential refresh when they're about to expire
  setTimeout(refreshCredentials, credentials.expireTime - new Date());
}
```

為了簡化這項工作，您可以使用 Amazon 位置JavaScript 驗證助手。這代替了獲取憑據並刷新它們。此範例使用第三 AWS 版的 JavaScript SDK。

```
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));
```

後續步驟

- 若要修改您的角色，請前往 IAM 主控台。
- 若要管理您的身分集區，請前往 Amazon Cognito 主控台。

監控 Amazon Location Service

使用 Amazon 定 Location Service 時，您可以使用以下方式監控一段時間內的使用情況和資源：

- Amazon CloudWatch。監控您的 Amazon 定 Location Service 資源，並以近乎即時的方式提供指標與統計資料。
- AWS CloudTrail。 提供對 Amazon 定 Location Service API 所有呼叫的事件追蹤。

本節提供有關使用這些服務的資訊。

主題

- 監控 Amazon Location Service 與 Amazon CloudWatch
- 記錄和監控 AWS CloudTrail

監控 Amazon Location Service 與 Amazon CloudWatch

Amazon 以近乎即時的方式 CloudWatch 監控您的AWS資源和執行AWS的應用程式。您可以使用監控 Amazon Location 資源 CloudWatch，該資源會以近乎即時的方式收集原始資料並將指標處理為有意義的統計資料 您可以檢視長達 15 個月的歷史資訊，或搜尋指標以在 Amazon CloudWatch 主控台中檢視，以取得有關 Amazon 位置資源的更多觀點。您也可以透過定義閾值來設定警示，並在達到這些閾值時傳送通知或採取動作。

如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南

主題

- Amazon 定 Location Service 指標匯出到 Amazon CloudWatch
- 檢視 Amazon Location Service 指標
- 為 Amazon 定 CloudWatch Location Service 指標建立警示
- 用 CloudWatch 於監控配額的使用情況
- CloudWatch Amazon 定 Location Service 的指標示例

Amazon 定 Location Service 指標匯出到 Amazon CloudWatch

量度是匯出至 CloudWatch的時間排序資料點。維度是識別量度的名稱/值組。如需詳細資訊，請參閱 Amazon 使用者指南中的使 CloudWatch 用指 CloudWatch 標和CloudWatch維度。

以下是 Amazon 定 Location Service 在`AWS/Location`命名空間 CloudWatch 中匯出到的指標。

指標	描述
`CallCount`	對指定 API 端點進行的呼叫數目。 有效維度：Amazon 定 Location Service API 名稱 有效統計資訊：總和 單位：計數
`ErrorCount`	對指定 API 端點進行呼叫的錯誤回應數。 有效維度：Amazon 定 Location Service API 名稱 有效統計資訊：總和 單位：計數
`SuccessCount`	對指定 API 端點進行的成功呼叫次數。 有效維度：Amazon 定 Location Service API 名稱 有效統計資訊：總和 單位：計數

指標	描述
CallLatency	對指定 API 端點進行呼叫時，作業處理和傳回回應所需的時間量。 有效維度：Amazon 定 Location Service API 名稱 有效的統計數字：平均 單位：毫秒

檢視 Amazon Location Service 指標

您可以在Amazon CloudWatch主控台或使用 Amazon CloudWatch API 來檢視 Amazon 定 Location Service 的指標。

若要使用 CloudWatch 主控台檢視指標

Example

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 在所有指標索引標籤上，選擇 Amazon 位置命名空間。
4. 選取要檢視的測量結果類型。
5. 選取量度並將其新增至圖表。

如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的檢視可用指標。

為 Amazon 定 CloudWatch Location Service 指標建立警示

您可以使用在 Amazon CloudWatch 定 Location Service 指標上設定警示。例如，您可以在中建立警示，以 CloudWatch 便在發生錯誤計數尖峰時傳送電子郵件。

以下主題給您如何使用 CloudWatch 設定警示的高階概觀。如需詳細指示，請參閱 Amazon 使用 CloudWatch 者指南中的使用警示。

使用 CloudWatch 主控台設定鬧鐘

Example

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在功能窗格中，選擇 [警報]。
3. 選擇 Create Alarm (建立警示)。
4. 選擇 Select metric (選取指標)。
5. 在所有指標索引標籤上，選取 Amazon 位置命名空間。
6. 選取測量結果類別。
7. 找出含有您要建立警示之量度的資料列，然後選取此列旁的核取方塊。
8. 選擇選取指標。
9. 在「度量」下，填入值。
10. 指定警示條件。
11. 選擇下一步。
12. 如果您要在符合鬧鐘條件時傳送通知：
    - 在警示狀態觸發下，選取警示狀態以提示要傳送的通知。
    - 在 「選取 SNS 主題」下，選擇「建立新主題」以建立新的 Amazon Simple Notification Service (Amazon SNS) 主題。輸入主題名稱和要傳送通知的電子郵件。
    - 在「傳送通知以輸入要傳送通知的其他電子郵件地址」下方。
    - 選擇 Add notification (新增通知)。此清單會儲存並顯示在欄位中供未來警示使用。
13. 完成時，請選擇下一步。
14. 輸入鬧鐘的名稱和說明，然後選擇「下一步」。
15. 確認鬧鐘詳細資料，然後選擇「下一步」。

> **Note**
>
> 建立新的 Amazon SNS 主題時，您必須先驗證電子郵件地址，才能傳送通知。如果未驗證電子郵件，當狀態變更啟動警示時，將不會收到通知。

如需如何使用 CloudWatch 主控台設定警示的詳細資訊，請參閱 Amazon 使用 CloudWatch者指南中的建立可傳送電子郵件的警示。

用 CloudWatch 於監控配額的使用情況

您可以建立 Amazon CloudWatch 警示，以在指定配額的使用率超過可設定的閾值時通知您。這可讓您識別何時接近配額限制，並調整使用率以避免成本超支，或視需要要求增加配額。如需如何使用監控

配額 CloudWatch 的相關資訊，請參閱 Amazon 使用 CloudWatch 者指南中的視覺化服務配額和設定警示。

CloudWatch Amazon 定 Location Service 的指標示例

您可以使用 GetMetricDataAPI 擷取 Amazon 位置的指標。

- 例如，您可以監視CallCount並設定發生數量下降時的警示。

  監控的CallCount指標SendDeviceLocation可協助您瞭解追蹤的資產。如果下CallCount降，則意味著跟踪的資產（例如卡車車隊）已停止發送其當前位置。為此設置警報可以幫助通知您發生了問題。

- 對於另一個範例，您可以監視ErrorCount並設定發生數字尖峰的時間警示。

  跟踪器必須與地理圍欄集合相關聯，以便根據地理圍欄進行評估設備位置。如果您的裝置叢集需要持續的位置更新，看到 CallCount for BatchEvaluateGeofence或BatchPutDevicePosition降到零表示更新不再流動。

以下是GetMetricData用於建立地圖資源CallCount和ErrorCount建立地圖資源的度量的範例輸出。

```
{
  "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"
      }
    }
  ]
}
```

記錄和監控 AWS CloudTrail

AWS CloudTrail是提供使用者、角色或服務所採取之動作記錄的AWS服務。 CloudTrail 將所有 API 呼叫記錄為事件。您可以搭配使用 Amazon 定 Location Service CloudTrail 來監控您的 API 呼叫，其中包括來自 Amazon 定 Location Service 主控台的呼叫和 AWS SDK 呼叫 Amazon 定 Location Service API 作業。

建立追蹤時，您可以啟用持續向 S3 儲存貯體傳遞 CloudTrail 事件，包括 Amazon Location Service 的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷向 Amazon 定 Location Service 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。

若要取得有關的更多資訊 CloudTrail，請參閱AWS CloudTrail使用者指南。

主題

- Amazon Location Service 信息 CloudTrail
- 了解 Amazon Location Service 日誌檔項目

Amazon Location Service 信息 CloudTrail

CloudTrail 在您創建AWS帳戶時，您的帳戶已啟用。在 Amazon 定 Location Service 中發生活動時，該活動會與 CloudTrail 事件歷史記錄中的其他AWS服務事件一起記錄在事件中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需AWS帳戶中持續的事件記錄 (包括 Amazon 定 Location Service 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔傳遞至 S3 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 S3 儲存貯體。此外，您還可以設定其他AWS服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。

如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔

所有 Amazon 定 Location Service 動作都會記錄下來， CloudTrail 並記錄在 Amazon 定 Location Service API 參考資料中。例如，呼叫`UpdateTracker`和`DescribeTracker`動作會`CreateTracker`在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷提出請求的身分是：

- 使用根或 AWS Identity and Access Management (IAM) 使用者登入資料。
- 使用某個角色的暫時安全憑證登入資料或聯合身分使用者。
- 透過其他 AWS 服務。

如需詳細資訊，請參閱 CloudTrail 使用者身分元素。

了解 Amazon Location Service 日誌檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 S3 儲存貯體或 Amazon CloudWatch 日誌。若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail使用 CloudTrail 記錄檔〉。

CloudTrail 記錄檔包含一或多個記錄項目。一個事件代表任何來源提出的單一請求，並包含所請求之操作的相關資訊、操作的日期和時間、請求參數等等。

> **Note**
>
> CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。若要確定作業的順序，請使用`eventTime`。

下列範例顯示示範建立追蹤器資源之`CreateTracker`作業的 CloudTrail 記錄項目。

```
{
    "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",
}
```

以下顯示DescribeTracker作業的記錄項目，此項目會傳回追蹤器資源的詳細資訊。

```
{
    "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",
}
```


創建 Amazon Location Service 資源 AWS CloudFormation

Amazon 定 Location Service 與整合AWS CloudFormation，這項服務可協助您建立資源模型和設定AWS資源，以減少建立和管理資源和基礎設施的時間。您可以建立一個範本來描述所需的所有AWS資源 (例如 Amazon Location 資源)，並為您AWS CloudFormation佈建和設定這些資源。

使用時AWS CloudFormation，您可以重複使用範本，以一致且重複地設定 Amazon 位置資源。只需描述一次您的資源，即可在多個 AWS 帳戶與區域內重複佈建相同資源。

Amazon 位置和AWS CloudFormation模板

若要佈建和設定 Amazon 位置和相關服務的資源，您必須瞭解AWS CloudFormation範本。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation 設計器協助您開始使用 AWS CloudFormation 範本。如需更多詳細資訊，請參閱 AWS CloudFormation 使用者指南 中的 什麼是 AWS CloudFormation 設計器？。

Amazon 位置支援在中建立下列資源類型AWS CloudFormation：

- AWS::Location::Map
- AWS::Location::PlaceIndex
- AWS::Location::RouteCalculator
- AWS::Location::Tracker
- AWS::Location::TrackerConsumer
- AWS::Location::GeofenceCollection

如需詳細資訊，包括 Amazon 位置資源的 JSON 和 YAML 範本範本範例，請參閱AWS CloudFormation使用者指南中的 Amazon 定 Location Service 資源類型參考。

進一步了解 AWS CloudFormation

如需進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation使用者指南
- AWS CloudFormation API 參考
- AWS CloudFormation 命令列介面使用者指南

Amazon Location Service 中的安全

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同承擔的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端安全性 — AWS 負責保護中執行 AWS 服務的基礎架構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。若要了解適用於 Amazon Location Service 的合規計劃，請參閱AWS 合規計劃的合規計劃AWS 服務範的服務。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 Amazon 位置時套用共同的責任模型。以下主題說明如何設定 Amazon 位置以符合安全和合規目標。您也會學到如何使用其他可 AWS 協助您監控和保護 Amazon 位置資源的服務。

主題

- Amazon Location Service 中的數據保護
- Amazon 定 Location Service 務的 Identity and Access Management
- Amazon Location Service 中的事件回應
- Amazon 定 Location Service 的合規驗證
- Amazon Location Service 的彈性
- Amazon Location Service 中的基礎設施安全
- Amazon 位置中的組態和漏洞分析
- 預防跨服務混淆代理人
- Amazon 定位服務的安全最佳實務
- Amazon 定 Location Service 的最佳實踐

Amazon Location Service 中的數據保護

AWS 共同責任模型適用於 Amazon 定 Location Service 中的資料保護。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS

服務 的安全組態和管理任務。如需有關資料隱私權的詳細資訊，請參閱資料隱私權FAQ。 如需歐洲資料保護的相關資訊，請參閱AWS 安全性GDPR部落格上的AWS 共同責任模型和部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 認證並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 對每個帳戶使用多重要素驗證 (MFA)。
- 使用SSL/TLS與 AWS 資源溝通。我們需要 TLS 1.2 並推薦 TLS 1.3。
- 使用設定API和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果 AWS 透過命令列介面或存取時需要 FIPS 140-3 驗證的密碼編譯模組API，請使用端點。FIPS 如需有關可用FIPS端點的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、、API或 AWS 服務 使用 Amazon 位置或其他位置時 AWS SDKs。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供URL給外部伺服器，我們強烈建議您不要在中包含認證資訊，URL以驗證您對該伺服器的要求。

資料隱私

使用 Amazon 定 Location Service，您可以保留組織資料的控制權。Amazon Location 會移除客戶中繼資料和帳戶資訊，以匿名方式傳送給資料提供者的所有查詢。

Amazon 位置不會使用資料提供者進行追蹤和地理圍欄。這意味著您的敏感數據仍保留在您的 AWS 帳戶中。這有助於保護敏感的位置資訊，例如設施、資產和人員位置，不受第三方的侵害，保護使用者隱私，並降低應用程式的安全風險。

如需其他資訊，請參閱資AWS 料隱私權FAQ。

Amazon 位置的資料保留

下列特性與 Amazon 位置如何收集和存放服務的資料有關：

- Amazon Location Service 追蹤器 — 當您使用追蹤器APIs追蹤實體的位置時，可以存放其座標。裝置位置會儲存 30 天，然後才會被服務刪除。

- Amazon 定 Location Service 地理圍欄 — 當您使用地理圍欄定義感興趣的區域時，服務會儲存您提供的幾APIs何圖形。必須明確刪除它們。

> **Note**
>
> 刪除 AWS 帳號會刪除其中的所有資源。如需其他資訊，請參閱資AWS 料隱私權FAQ。

Amazon 定 Location Service 的靜態資料加密

Amazon 定 Location Service 預設提供加密功能，以使用 AWS 擁有的加密金鑰保護靜態的敏感客戶資料。

- AWS 擁有的金鑰 — Amazon Location 預設會使用這些金鑰來自動加密個人識別資料。您無法檢視、管理或使用 AWS 擁有的金鑰，也無法稽核其使用情況。不過，您不需要採取任何動作或變更任何程式，即可保護加密您資料的金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS 擁有的金鑰。

依預設加密靜態資料，有助於降低保護敏感資料所涉及的營運開銷和複雜性。同時，其可讓您建置符合嚴格加密合規性和法規要求的安全應用程式。

雖然您無法停用此層加密或選取替代加密類型，但您可以在建立追蹤器和地理圍欄收集資源時選擇客戶管理的金鑰，在現有 AWS 擁有的加密金鑰上新增第二層加密：

- 客戶受管金鑰 — Amazon Location 支援使用您建立、擁有和管理的對稱客戶受管金鑰，以針對現有 AWS 擁有的加密新增第二層加密。您可以完全控制此層加密，因此能執行以下任務：
  - 建立和維護金鑰政策
  - 建立和維護IAM政策和補助金
  - 啟用和停用金鑰政策
  - 輪換金鑰密碼編譯資料
  - 新增標籤
  - 建立金鑰別名
  - 安排金鑰供刪除

  如需詳細資訊，請參閱AWS Key Management Service 開發人員指南中的客戶管理金鑰。

下表摘要說明 Amazon 位置如何加密個人識別資料。

資料類型	AWS 擁有的金鑰加密	客戶自管金鑰加密 (選用)
`Position` 包含裝置位置詳細資料的點幾何圖形。	已啟用	已啟用
`PositionProperties` 一組與位置更新相關聯的鍵值對。	已啟用	已啟用
`GeofenceGeometry` 表示地理圍欄區域的多邊形地理圍欄幾何圖形。	已啟用	已啟用
`DeviceId` 將裝置位置更新上傳至追蹤器資源時指定的裝置識別碼。	已啟用	不支援
`GeofenceId` 存儲地理圍欄幾何圖形或給定地理圍欄集合中的一批地理圍欄時指定的標識符。	已啟用	不支援

> **Note**
>
> Amazon Location 可使用 AWS 擁有的金鑰自動啟用靜態加密，以免費保護個人識別資料。但是，使用客戶管理的金鑰需要 AWS KMS 支付費用。如需有關定價的詳細資訊，請參閱定AWS Key Management Service 價。

如需有關的詳細資訊 AWS KMS，請參閱「什麼是 AWS Key Management Service？」

Amazon Location Service 如何使用贈款 AWS KMS

Amazon 位置需要授權才能使用您的客戶受管金鑰。

當您建立使用客戶管理金鑰加密的追蹤器資源或地理圍欄集合時，Amazon Location 會將CreateGrant請求傳送至以代表您建立授權。 AWS KMS中的授權 AWS KMS 用於授予 Amazon 位置存取客戶帳戶中KMS金鑰的權限。

Amazon 位置需要授權，才能在下列內部操作中使用客戶受管金鑰：

- 發送DescribeKey請求 AWS KMS 以驗證在創建跟踪器或地理圍欄集合時輸入的對稱客戶託管KMS密鑰 ID 是否有效。
- 傳送GenerateDataKeyWithoutPlaintext要求 AWS KMS 以產生由客戶管理金鑰加密的資料金鑰。
- 發送解密請求 AWS KMS 以解密加密的數據密鑰，以便將其用於加密您的數據。

您可以隨時撤銷授予的存取權，或移除服務對客戶受管金鑰的存取權。如果這樣做，Amazon Location 將無法存取客戶受管金鑰加密的任何資料，這會影響依賴該資料的操作。例如，如果您嘗試從 Amazon Location 無法存取的加密追蹤器取得裝置位置，則作業會傳回錯`AccessDeniedException`誤。

建立客戶受管金鑰

您可以使用 AWS Management Console、或建立對稱的客戶管理金鑰。 AWS KMS APIs

建立對稱客戶受管金鑰

請依照《AWS Key Management Service 開發人員指南》中建立對稱客戶受管金鑰的步驟進行。

金鑰政策

金鑰政策會控制客戶受管金鑰的存取權限。每個客戶受管金鑰都必須只有一個金鑰政策，其中包含決定誰可以使用金鑰及其使用方式的陳述式。在建立客戶受管金鑰時，可以指定金鑰政策。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。

若要將客戶受管金鑰與 Amazon 位置資源搭配使用，金鑰政策中必須允許下列API操作：

- `kms:CreateGrant`：新增客戶受管金鑰的授權。授予對指定KMS金鑰的控制權限，以允許存取授與 Amazon 位置所需的操作。如需有關使用授權的詳細資訊，請參閱開AWS Key Management Service 發人員指南。

  這允許 Amazon 位置執行以下操作：

- 呼叫 GenerateDataKeyWithoutPlainText 以產生加密的資料金鑰並加以儲存，因為資料金鑰不會立即用來加密。
- 呼叫 Decrypt 以使用儲存的加密資料金鑰來存取加密的資料。
- 設定退休本金以允許服務。RetireGrant

- kms:DescribeKey— 提供客戶受管的金鑰詳細資訊，以允許 Amazon 位置驗證金鑰。

以下是您可以為 Amazon 位置新增的政策聲明範例：

```
 "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" : "*"
    }
  ]
```

如需有關在政策中指定許可的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

如需有關故障診斷金鑰存取的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

指定 Amazon 位置的客戶受管金鑰

您可以將客戶自管金鑰指定為下列資源的第二層加密：

- 追蹤器資源
- 地理圍欄集合

建立資源時，您可以透過輸入 ID 來指定資料金鑰，Amazon Location 使用該 KMSID 來加密資源所儲存的可識別個人資料。

- KMSID — AWS KMS 客戶管理金鑰的金鑰識別碼。輸入金鑰 ID、金鑰ARN、別名或別名ARN。

Amazon Location Service 加密內容

加密內容是一組選用的金鑰值對，包含資料的其他相關內容資訊。

AWS KMS 使用加密內容作為其他驗證資料，以支援已驗證的加密。當您在加密資料的要求中包含加密內容時，會將加密內容 AWS KMS 繫結至加密的資料。若要解密資料，您必須在請求中包含相同的加密內容。

Amazon Location Service 加密內容

Amazon 位置在所有密 AWS KMS 碼編譯操作中使用相同的加密內容，其中金鑰所在，值為資源 Amazon 資源名稱 (ARN)。`aws:geo:arn`

Example

```
"encryptionContext": {
    "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/SAMPLE-
GeofenceCollection"
}
```

使用加密內容進行監控

當您使用對稱的客戶管理金鑰來加密追蹤器或地理圍欄集合時，您也可以在稽核記錄和記錄中使用加密內容，以識別客戶管理金鑰的使用方式。加密內容也會出現在AWS CloudTrail 或 Amazon 日誌產生的CloudWatch 日誌中。

使用加密內容控制對客戶受管金鑰的存取

您可以在金鑰政策和IAM政策中使用加密內容conditions來控制對稱客戶管理金鑰的存取。您也可以在授予中使用加密內容條件。

Amazon Location 在授權中使用加密內容約束來控制對您帳戶或區域中客戶受管金鑰的存取。授予條件會要求授予允許的操作使用指定的加密內容。

Example

以下是授予特定加密內容之客戶受管金鑰存取權的金鑰政策陳述式範例。此政策陳述式中的條件會要求具有指定加密內容的加密內容條件。

```
{
    "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"
         }
    }
}
```

監控 Amazon 定 Location Service 的加密金鑰

當您將 AWS KMS 客戶受管金鑰與 Amazon 定 Location Service 資源搭配使用時，您可以使用AWS CloudTrail或 Amazon CloudWatch 日誌來追蹤 Amazon 位置傳送到的請求 AWS KMS。

下列範例是 Amazon Location 呼叫以存取由客戶受管金鑰加密的資料所呼叫的KMS操作Decrypt、、和DescribeKey監控 AWS CloudTrail 事件：CreateGrantGenerateDataKeyWithoutPlainText

CreateGrant

當您使用 AWS KMS 客戶受管金鑰加密追蹤器或地理圍欄收集資源時，Amazon Location 會代表您傳送存取帳戶中的KMS金鑰的CreateGrant AWS 請求。Amazon 位置建立的授權僅限於與 AWS KMS 客戶受管金鑰相關聯的資源。此外，當您刪除資源時，Amazon 位置會使用該RetireGrant操作移除授權。

下面的範例事件會記錄 CreateGrant 操作：

```
{
    "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

當您為追蹤器或地理圍欄收集資源啟用 AWS KMS 客戶受管金鑰時，Amazon Location 會建立唯一的表格金鑰。它會將要GenerateDataKeyWithoutPlainText求傳送至 AWS KMS 指定資源的 AWS KMS客戶管理金鑰。

下面的範例事件會記錄 GenerateDataKeyWithoutPlainText 操作：

```
{
    "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

當您存取加密的追蹤器或地理圍欄集合時，Amazon Location 會呼叫該Decrypt作業，使用儲存的加密資料金鑰存取加密的資料。

下面的範例事件會記錄 Decrypt 操作：

```
{
    "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 Location 會使用此DescribeKey操作來驗證帳 AWS KMS 戶和區域中是否存在與您的追蹤器或地理圍欄收集相關聯的客戶管理金鑰。

下面的範例事件會記錄 DescribeKey 操作：

```
{
    "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"
}
```


進一步了解

下列資源會提供有關靜態資料加密的詳細資訊。

- 如需 AWS Key Management Service 基本概念的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。
- 如需有關的安全性最佳做法的詳細資訊 AWS Key Management Service，請參閱開AWS Key Management Service 發人員指南。

Amazon 定 Location Service 的傳輸中資料加密

Amazon 位置會使用傳輸層安全性 (TLS) 1.2 加密協定自動加密所有網路間資料，藉此保護傳輸中的資料 (往返服務)。傳送至 Amazon Location Service 的直接HTTPS請求APIs是使用簽AWS章版本 4 演算法來簽署，以建立安全連線。

Amazon 定 Location Service 務的 Identity and Access Management

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。AWS 服務 IAM管理員控制哪些人可以進行身份驗證 (登入) 和授權 (具有權限) 以使用 Amazon 位置資源。IAM是一種您 AWS 服務 可以使用，無需額外費用。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon Location Service 如何與 IAM
- Amazon 定 Location Service 如何與未經身份驗證的用戶
- Amazon 定 Location Service 的身分識別政策範例
- 疑難排解 Amazon Location Service 身分和存取

物件

您如何使用 AWS Identity and Access Management (IAM) 會有所不同，具體取決於您在 Amazon 位置所做的工作。

服務使用者 — 如果您使用 Amazon Location 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您使用更多 Amazon 位置功能完成工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon 位置中的功能，請參閱疑難排解 Amazon Location Service 身分和存取。

服務管理員 — 如果您負責公司的 Amazon 位置資源，您可能擁有完整的 Amazon 位置存取權。您的任務是確定服務使用者應存取哪些 Amazon 位置功能和資源。然後，您必須向IAM管理員提交請求，才能變更服務使用者的權限。檢閱此頁面上的資訊，以瞭解的基本概念IAM。若要進一步了解貴公司如何IAM搭配 Amazon 定位使用，請參閱Amazon Location Service 如何與 IAM。

IAM管理員 — 如果您是管理IAM員，您可能想要了解如何撰寫政策以管理 Amazon 位置存取權的詳細資訊。若要檢視可在中使用的 Amazon 位置身分型政策範例IAM，請參閱。Amazon 定 Location Service 的身分識別政策範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以IAM使用者身分或假設IAM角色來驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM身分識別中心) 使用者、貴公司的單一登入驗證，以及您的 Google 或 Facebook 認證都是聯合身分識別的範例。當您以同盟身分登入時，您的管理員先前會使用IAM角色設定聯合身分識別。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以密碼編譯方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署要求的詳細資訊，請參閱使用IAM者指南中的簽署 AWS API要求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。若要深入瞭解，請參閱使用AWS IAM Identity Center 者指南中的多重要素驗證和使用多重要素驗證 (MFA) AWS的使用IAM者指南。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需需要您以 root 使用者身分登入的完整工作清單，請參閱《使用指南》中的〈需要 root 使用者認證的IAM工作〉。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步至您自己身分識別來源中的一組使用者和群組，以便在所有應用

程式 AWS 帳戶 和應用程式中使用。如需IAM身分識別中心的相關資訊，請參閱IAM識別中心是什麼？在《AWS IAM Identity Center 使用者指南》中。

IAM 使用者和群組

IAM使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定權限。在可能的情況下，我們建議您仰賴臨時登入資料，而不要建立具有長期認證 (例如密碼和存取金鑰) 的IAM使用者。不過，如果您的特定使用案例需要使用IAM者的長期認證，建議您輪換存取金鑰。如需詳細資訊，請參閱《使用指南》中的「IAM定期輪換存取金鑰」以瞭解需要長期認證的使用案例。

IAM群組是指定IAM使用者集合的身分識別。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為的群組，IAMAdmins並授與該群組管理IAM資源的權限。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。要了解更多信息，請參閱《IAM用戶指南》中的創建用戶（而不是角色）的IAM時間。

IAM角色

IAM角色是您 AWS 帳戶 中具有特定權限的身份。它類似於用IAM戶，但不與特定人員相關聯。您可以 AWS Management Console 透過切換角色來暫時擔任中的角色。IAM您可以透過呼叫 AWS CLI 或 AWS API作業或使用自訂來擔任角色URL。如需有關使用角色方法的詳細資訊，請參閱《使用指南》中的IAM〈使用IAM角色〉。

IAM具有臨時認證的角色在下列情況下很有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需聯合角色的相關資訊，請參閱《使用指南》中的〈建立第三方身分識別提供IAM者的角色〉。如果您使用IAM身分識別中心，則需要設定權限集。為了控制身分驗證後可以存取的內IAM容，IAMIdentity Center 會將權限集與中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時IAM使用者權限 — IAM 使用者或角色可以假定某個IAM角色，暫時取得特定工作的不同權限。
- 跨帳戶存取 — 您可以使用IAM角色允許不同帳戶中的某個人 (受信任的主體) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要瞭解跨帳戶存取角色與以資源為基礎的政策之間的差異，請參閱《IAM使用指南》IAM中的〈跨帳號資源存取〉。

- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中撥打電話時，該服務通常會在 Amazon 中執行應用程式EC2或將物件存放在 Amazon S3 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用使用IAM者或角色執行中的動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。
  - 服務角色 — 服務角色是指服務代表您執行動作所代表的IAM角色。IAM管理員可以從中建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱《IAM使用指南》 AWS 服務中的建立角色以將權限委派給
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。
- 在 Amazon 上執行的應用程式 EC2 — 您可以使用IAM角色來管理在執行個體上EC2執行的應用程式以及發出 AWS CLI 或 AWS API請求的臨時登入資料。這比在EC2實例中存儲訪問密鑰更好。若要將 AWS 角色指派給EC2執行個體並讓其所有應用程式都能使用，請建立附加至執行個體的執行個體設定檔。執行個體設定檔包含角色，可讓執行個體上EC2執行的程式取得臨時登入資料。如需詳細資訊，請參閱使用者指南中的使用IAM角色將許可授與在 Amazon EC2 執行個體上執行的應IAM用程式。

要了解是否使用IAM角色還是用IAM戶，請參閱《用戶指南》中的「IAM創建IAM角色的時機（而不是用戶）」。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以JSON文件的形式儲存在中。如需有關JSON原則文件結構和內容的詳細資訊，請參閱《IAM使用指南》中的策略概觀。JSON

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

IAM原則會定義動作的權限，不論您用來執行作業的方法為何。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或取得角色資訊 AWS API。

身分型政策

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管策略或內嵌策略之間進行選擇，請參閱《IAM使用手冊》中的「在受管策略和內嵌策略之間進行選擇」。

資源型政策

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的策略IAM中使用 AWS 受管政策。

存取控制清單 (ACLs)

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

Amazon S3 和 Amazon VPC 是支持服務的示例ACLs。 AWS WAF若要進一步了解ACLs，請參閱 Amazon 簡單儲存服務開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 權限界限 — 權限界限是一項進階功能，您可以在其中設定以身分識別為基礎的原則可授與給IAM實體 (IAM使用者或角色) 的最大權限。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界

限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需有關權限界限的詳細資訊，請參閱《IAM使用指南》中的IAM實體的權限界限。

- 服務控制策略 (SCPs) — SCPs 是指定中組織或組織單位 (OU) 最大權限的JSON策略 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。如果您啟用組織中的所有功能，則可以將服務控制策略 (SCPs) 套用至您的任何或所有帳戶。SCP限制成員帳戶中實體的權限，包括每個帳戶 AWS 帳戶根使用者。如需有關 Organizations 的詳細資訊SCPs，請參閱AWS Organizations 使用指南中的服務控制原則。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱《IAM使用指南》中的工作階段原則。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要瞭解如何在涉及多個原則類型時 AWS 決定是否允許要求，請參閱IAM使用指南中的原則評估邏輯。

Amazon Location Service 如何與 IAM

在您用IAM來管理 Amazon 位置的存取權限之前，請先了解哪些IAM功能可與 Amazon 位置搭配使用。

IAM您可以搭配 Amazon 定 Location Service 使用的功能

IAM特徵	Amazon 位置支持
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACLs	否
ABAC(策略中的標籤)	是

IAM特徵	Amazon 位置支持
暫時性憑證	是
主體許可	否
服務角色	否
服務連結角色	否

若要深入瞭解 Amazon Location 和其他 AWS 服務如何使用大多數IAM功能，請參閱IAM使用者指南IAM中的可使用AWS 服務。

Amazon 位置的基於身份的政策

支援身分型政策：是

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

使用以IAM身分識別為基礎的策略，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。若要瞭解可在JSON策略中使用的所有元素，請參閱《使用IAM者指南》中的IAMJSON策略元素參考資料。

Amazon 位置的基於身份的政策示例

若要檢視 Amazon 位置身分識別政策的範例，請參閱。Amazon 定 Location Service 的身分識別政策範例

Amazon 位置內的資源型政策

支援資源型政策：否

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以在以資源為基礎的策略中指定一個或多個帳戶中的一個或多個帳戶中的IAM實體作為主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主參與者和資源位於不同時 AWS 帳戶，受信任帳戶中的IAM管理員也必須授與主參與者實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM使用指南》IAM中的〈跨帳號資源存取〉。

Amazon 位置的政策動作

支援政策動作：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON策略`Action`元素描述了您可以用來允許或拒絕策略中存取的動作。策略動作通常與關聯的 AWS API操作具有相同的名稱。有一些例外情況，例如沒有匹配API操作的僅限權限的操作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon 位置動作清單，請參閱服務授權參考中由 Amazon 定 Location Service 務定義的動作。

Amazon 位置中的政策動作會在動作前使用下列前置詞：

```
geo
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "geo:action1",
      "geo:action2"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 `Get` 文字的所有動作，請包含以下動作：

```
"Action": "geo:Get*"
```

若要檢視 Amazon 位置身分識別政策的範例，請參閱。Amazon 定 Location Service 的身分識別政策範例

Amazon 地點的政策資源

支援政策資源：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

ResourceJSON原則元素會指定要套用動作的一個或多個物件。陳述式必須包含 Resource 或 NotResource 元素。最佳做法是使用其 Amazon 資源名稱 (ARN) 指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 Amazon 位置資源類型及其清單ARNs，請參閱服務授權參考中由 Amazon 定 Location Service 務定義的資源。若要了解可以針對每個資源指定哪些動作，請參閱 Amazon 定 Location Service 定義ARN的動作。

若要檢視 Amazon 位置身分識別政策的範例，請參閱。Amazon 定 Location Service 的身分識別政策範例

Amazon 位置的政策條件密鑰

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，只有在IAM使用者名稱標記資源時，您才可以授與IAM使用者存取資源的權限。如需詳細資訊，請參閱《IAM使用指南》中的IAM政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《使用指南》中的AWS 全域條件內IAM容索引鍵。

若要查看 Amazon 位置條件金鑰清單，請參閱服務授權參考中的 Amazon 定 Location Service 務的條件金鑰。若要了解可以使用條件金鑰的動作和資源，請參閱 Amazon 定 Location Service 定義的動作。

Amazon Location 支援條件金鑰，可讓您允許或拒絕存取政策陳述式中的特定地理圍欄或裝置。以下是可用的條件鍵：

- `geo:GeofenceIds`與地理圍欄操作一起使用。類型為`ArrayOfString`。
- `geo:DeviceIds`與「追蹤器」動作搭配使用。類型為`ArrayOfString`。

您的IAM政策中可以使用`geo:GeofenceIds`下列動作：

- `BatchDeleteGeofences`
- `BatchPutGeofences`
- `GetGeofence`
- `PutGeofence`

您的IAM政策中可以使用`geo:DeviceIds`下列動作：

- `BatchDeleteDevicePositionHistory`
- `BatchGetDevicePosition`
- `BatchUpdateDevicePosition`
- `GetDevicePosition`
- `GetDevicePositionHistory`

> **Note**
>
> 您無法將這些條件鍵與BatchEvaluateGeofences、ListGeofences或ListDevicePosition動作搭配使用。

若要檢視 Amazon 位置身分識別政策的範例，請參閱。Amazon 定 Location Service 的身分識別政策範例

ACLs在 Amazon 位置

支持ACLs：無

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

ABAC與 Amazon 位置

支援 ABAC (策略中的標籤): 是

以屬性為基礎的存取控制 (ABAC) 是一種授權策略，可根據屬性定義權限。在中 AWS，這些屬性稱為標籤。您可以將標籤附加至IAM實體 (使用者或角色) 和許多 AWS 資源。標記實體和資源是的第一步ABAC。然後，您可以設計ABAC策略，以便在主參與者的標籤與他們嘗試存取的資源上的標籤相符時允許作業。

ABAC在快速成長的環境中很有幫助，並且有助於原則管理變得繁瑣的情況。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需有關的詳細資訊ABAC，請參閱什麼是ABAC？ 在《IAM使用者指南》中。若要檢視包含設定步驟的自學課程ABAC，請參閱《使用指南》中的〈使用以屬性為基礎的存取控制 (ABAC) IAM〉。

如需標記 Amazon 位置資源的詳細資訊，請參閱標記您的 Amazon Location Service 資源。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱根據標籤控制資源存取。

透過 Amazon 位置使用臨時登入資料

支援臨時憑證：是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配使用 AWS 服務 ，請參閱《IAM使用指南》IAM中的使用方式。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需有關切換角色的詳細資訊，請參閱《IAM使用者指南》中的〈切換到角色 (主控台)〉。

您可以使用 AWS CLI 或手動建立臨時認證 AWS API。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而非使用長期存取金鑰。如需詳細資訊，請參閱IAM。

Amazon 位置的跨服務主體許可

支持轉發訪問會話（FAS）：否

當您使用使用IAM者或角色在中執行動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。

Amazon 位置的服務角色

支援服務角色：否

服務角色是服務假定代表您執行動作的IAM角色。IAM管理員可以從中建立、修改和刪除服務角色 IAM。如需詳細資訊，請參閱《IAM使用指南》 AWS 服務中的建立角色以將權限委派給

> **Warning**
>
> 變更服務角色的許可可可可可能會破壞 Amazon 位置功能。只有在 Amazon 位置提供指導時才編輯服務角色。

適用於 Amazon 位置的服務連結角色

支援服務連結角色：否

服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。

如需有關建立或管理服務連結角色的詳細資訊，請參閱使用IAM的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amazon 定 Location Service 如何與未經身份驗證的用戶

使用 Amazon 定 Location Service 的許多案例 (包括在網路或行動應用程式中顯示地圖) 都需要允許存取尚未登入的使用者IAM。對於這些未經驗證的案例，您有兩個選擇。

- 使用API金鑰 — 若要將存取權授予未經驗證的使用者，您可以建立授與 Amazon 定 Location Service 資源的唯讀存取權限的API金鑰。這在您不想驗證每個用戶的情況下非常有用。例如，一個 Web 應用程序。如需API金鑰的詳細資訊，請參閱允許未經驗證的訪客使用 API 金鑰存取您的應用程式。
- 使用 Amazon Cognito — API 金鑰的替代方法是使用 Amazon Cognito 授予匿名存取權。Amazon Cognito 可讓您透過IAM政策建立更豐富的授權，以定義未經驗證的使用者可以執行的動作。如需使用 Amazon Cognito 的詳細資訊，請參閱允許未經驗證的訪客使用 Amazon Cognito 存取您的應用程式。

如需提供對未驗證使用者存取權的概觀，請參閱授予對 Amazon 定 Location Service 的訪問。

Amazon 定 Location Service 的身分識別政策範例

依預設，使用者和角色沒有建立或修改 Amazon 位置資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或執行工作 AWS API。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

若要瞭解如何使用這些範例原則文件來建立以IAM身分識別為基礎的JSON策略，請參閱使用指南中的IAM建立IAM策略。

有關 Amazon Location 定義的動作和資源類型的詳細資訊，包括每種資源類型的格式，請參閱服務授權參考中的 Amazon Location Service 的動作、資源和條件金鑰。ARNs

主題

- 政策最佳實務
- 使用 Amazon 位置控制台
- 允許使用者檢視他們自己的許可

- 在政策中使用 Amazon Location Service 資源
- 更新裝置位置的權限
- 追蹤器資源的唯讀政策
- 建立地理圍欄的政策
- 地理圍欄的只讀策略
- 渲染地圖資源的權限
- 允許搜尋作業的權限
- 路由計算器的唯讀策略
- 根據條件鍵控制資源存取
- 根據標籤控制資源存取

政策最佳實務

以身分識別為基礎的政策決定某人是否可以在您的帳戶中建立、存取或刪除 Amazon 位置資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需詳細資訊，請參閱AWS 《IAM使用指南》中針對工作職能的AWS 受管理策略或受管理的策略。
- 套用最低權限權限 — 當您使用原則設定權限時，IAM只授與執行工作所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需有關使用套用權限IAM的詳細資訊，請參閱《使用指南》IAM中的IAM《策略與權限》。
- 使用IAM策略中的條件進一步限制存取 — 您可以在策略中新增條件，以限制對動作和資源的存取。例如，您可以撰寫政策條件，以指定必須使用傳送所有要求SSL。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱《IAM使用指南》中的IAMJSON策略元素：條件。
- 使用 IAM Access Analyzer 驗證您的原IAM則，以確保安全和功能性的權限 — IAM Access Analyzer會驗證新的和現有的原則，以便原則遵循IAM原則語言 (JSON) 和IAM最佳做法。IAMAccess Analyzer 提供超過 100 項原則檢查和可行的建議，協助您撰寫安全且功能正常的原則。如需詳細資訊，請參閱IAM使IAM用指南中的存取分析器原則驗證。
- 需要多因素驗證 (MFA) — 如果您的案例需要使IAM用者或 root 使用者 AWS 帳戶，請開啟以取得額外MFA的安全性。若要在呼叫API作業MFA時需要，請在原則中新增MFA條件。如需詳細資訊，請參閱《IAM使用指南》中的 < 設定MFA受保護的API存取 > 。

如需有關中最佳作法的詳細資訊IAM，請參閱《IAM使用指南》IAM中的「安全性最佳作法」。

使用 Amazon 位置控制台

若要存取 Amazon 定 Location Service 主控台，您必須擁有最少一組許可。這些許可必須允許您列出和檢視有關 AWS 帳戶. 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為只對 AWS CLI 或撥打電話的使用者允許最低主控台權限 AWS API。相反地，只允許存取符合他們嘗試執行之API作業的動作。

為確保使用者和角色可以使用 Amazon 位置主控台，請將以下政策附加到實體。如需詳細資訊，請參閱《使用指南》中的〈將權限新增至IAM使用者〉。

下列政策提供 Amazon 定 Location Service 主控台的存取權，以便能夠建立、刪除、列出和檢視 AWS 帳戶中 Amazon 位置資源的詳細資料。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GeoPowerUser",
      "Effect": "Allow",
      "Action": [
        "geo:*"
      ],
      "Resource": "*"
    }
  ]
}
```

或者，您可以授予唯讀權限，以促進唯讀存取。使用唯讀權限時，如果使用者嘗試寫入動作 (例如建立或刪除資源)，就會顯示錯誤訊息。例如，請參閱 the section called “追蹤器的唯讀政策”

允許使用者檢視他們自己的許可

此範例顯示如何建立原則，讓使IAM用者檢視附加至其使用者身分識別的內嵌和受管理原則。此原則包含在主控台上或以程式設計方式使用或完成此動作的 AWS CLI 權限 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": "*"
        }
    ]
}
```

在政策中使用 Amazon Location Service 資源

Amazon 定 Location Service 使用下列資源前置詞：

Amazon 位置資源前綴

資源	資源前綴
地圖資源	`map`
放置資源	`place-index`
路線資源	`route-calculator`

資源	資源前綴
追蹤資源	`tracker`
地理圍欄收集資源	`geofence-collection`

使用下列ARN語法：

```
arn:Partition:geo:Region:Account:ResourcePrefix/ResourceName
```

如需的格式的詳細資訊ARNs，請參閱 Amazon 資源名稱 (ARNs) 和 AWS 服務命名空間。

範例

- 使用以下內容ARN允許訪問指定的映射資源。

  ```
  "Resource": "arn:aws:geo:us-west-2:account-id:map/map-resource-name"
  ```

- 若要指定對屬於特定帳號的所有map資源的存取權，請使用萬用字元 (*)：

  ```
  "Resource": "arn:aws:geo:us-west-2:account-id:map/*"
  ```

- 某些 Amazon 位置動作 (例如用於建立資源的動作) 無法在特定資源上執行。在這些情況下，您必須使用萬用字元 (*)。

  ```
  "Resource": "*"
  ```

若要查看 Amazon 位置資源類型及其清單ARNs，請參閱服務授權參考中由 Amazon 定 Location Service 務定義的資源。若要了解可以針對每個資源指定哪些動作，請參閱 Amazon 定 Location Service 定義ARN的動作。

更新裝置位置的權限

要更新多個跟踪器的設備位置，您需要授予用戶訪問一個或多個跟踪器資源的訪問權限。您還希望允許用戶更新一批設備位置。

在此範例中，除了授與存取權 *Tracker1* 以及 *Tracker2* 資源，下列政策授予使用動geo:BatchUpdateDevicePosition作的權限 *Tracker1* 以及 *Tracker2* 的費用。

```
{
  "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"
      ]
    }
  ]
}
```

如果您想要限制使用者只能更新特定裝置的裝置位置，您可以為該裝置 ID 新增條件金鑰。

```
{
  "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"
          ]
        }
      }
    }
  ]
}
```

追蹤器資源的唯讀政策

要為 AWS 帳戶中的所有跟踪器資源創建只讀策略，您需要授予對所有跟踪器資源的訪問權限。您還需要授予用戶訪問操作的權限，這些操作允許他們獲取多個設備的設備位置，從單個設備獲取設備位置並獲取位置歷史記錄。

在此範例中，下列原則會授與下列動作的權限：

- geo:BatchGetDevicePosition以擷取多個裝置的位置。
- geo:GetDevicePosition以擷取單一裝置的位置。
- geo:GetDevicePositionHistory檢索設備的位置歷史記錄。

```
{
  "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/*"
    }
  ]
}
```

建立地理圍欄的政策

若要建立原則以允許使用者建立地理圍欄，您需要授與特定動作的存取權，這些動作允許使用者在地理圍欄集合上建立一或多個地理圍欄。

下列政策授予下列動作的權限 *Collection*:

- geo:BatchPutGeofence創建多個地理圍欄。
- geo:PutGeofence創建一個單一的地理圍欄。

```
{
```

```
  "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"
    }
  ]
}
```

地理圍欄的只讀策略

若要為儲存在您 AWS 帳戶的地理圍欄集合中的地理圍欄建立唯讀原則，您需要授與存放地理圍欄之地理圍欄集合讀取的動作的存取權。

下列政策授予下列動作的權限 *Collection*:

- geo:ListGeofences列出指定地理圍欄集合中的地理圍欄。
- geo:GetGeofence從地理圍欄集合中檢索地理圍欄。

```
{
  "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"
    }
  ]
}
```

渲染地圖資源的權限

若要授予足夠的權限來呈現地圖，您需要授予對地圖磚、精靈、字符和樣式描述元的存取權限：

- `geo:GetMapTile`擷取用於在地圖上選擇性地彩現圖徵的地圖框。
- `geo:GetMapSprites`擷取 PNG Sprite 工作表和描述其中偏移的對應JSON文件。
- `geo:GetMapGlyphs`檢索用於顯示文本的字符。
- `geo:GetMapStyleDescriptor`檢索包含渲染規則的映射樣式描述符。

```
{
  "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"
    }
  ]
}
```

允許搜尋作業的權限

若要建立允許搜尋作業的政策，您必須先授與 AWS 帳號中放置索引資源的存取權。您還需要授予訪問權限，這些操作允許用戶通過地理編碼使用文本進行搜索，並通過反向地理編碼使用位置進行搜索。

在此範例中，除了授與存取權 *PlaceIndex*，下列原則也會授與下列動作的權限：

- `geo:SearchPlaceIndexForPosition`可讓您搜尋指定位置附近的地點或景點。
- `geo:SearchPlaceIndexForText`可讓您使用任意格式的文字來搜尋地址、名稱、城市或地區。

```
{
  "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"
    }
  ]
}
```

路由計算器的唯讀策略

您可以建立唯讀策略，以允許使用者存取路由計算器資源以計算路由。

在此範例中，除了授與存取權 *ExampleCalculator*，下列原則會授與下列作業的權限：

- geo:CalculateRoute計算給定出發位置，目標位置和航點位置列表的路線。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:us-west-2:accountID:route-calculator/ExampleCalculator"
    }
  ]
}
```

根據條件鍵控制資源存取

當您建立IAM原則以授予使用地理圍欄或裝置位置的存取權時，您可以使用條件運算子來更精確地控制使用者可以存取的地理圍欄或裝置。您可以通過在策略的Condition元素中包含地理圍欄 ID 或設備 ID 來執行此操作。

下列範例原則顯示如何建立允許使用者更新特定裝置的裝置位置的原則。

```
{
  "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"
          ]
        }
      }
    }
  ]
}
```

根據標籤控制資源存取

建立IAM政策以授與使用 Amazon Location 資源的存取權時，您可以使用以屬性為基礎的存取控制來更好地控制使用者可以修改、使用或刪除的資源。您可以透過在原則的Condition元素中包含標籤資訊，以根據資源標籤控制存取權限來執行此操作。

下列範例原則顯示如何建立允許使用者建立地理圍欄的原則。這將授予以下操作的權限，以便在名為的地理圍欄集合上創建一個或多個地理圍欄 *Collection*:

- geo:BatchPutGeofence創建多個地理圍欄。
- geo:PutGeofence創建一個單一的地理圍欄。

不過，此原則只會在下列情況下才會使用Condition元素來授與權限 *Collection* tag，Owner，具有該使用者使用者名稱的值。

- 例如，如果名為的使用者richard-roe嘗試檢視 Amazon 位置 *Collection*，該 *Collection* 必須加上標籤Owner=richard-roe或owner=richard-roe。否則，用戶將被拒絕訪問。

> **Note**
>
> 條件標籤鍵 `Owner` 符合 `Owner` 和 `owner`，因為條件索引鍵名稱不區分大小寫。如需詳細資訊，請參閱《IAM使用指南》中的《IAMJSON策略元素：條件》。

```
{
  "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}"}
      }
    }
  ]
}
```

如需有關如何根據標籤定義存取AWS資源權限的教學課程，請參閱《AWS Identity and Access Management 使用指南》。

疑難排解 Amazon Location Service 身分和存取

使用下列資訊協助您診斷和修正使用 Amazon 位置和時可能遇到的常見問題IAM。

主題

- 我沒有授權在 Amazon 位置執行操作
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪 AWS 帳戶 問我的 Amazon 位置資源

我沒有授權在 Amazon 位置執行操作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

當使用mateojacksonIAM者嘗試使用主控台來檢視虛構*my-example-widget*資源的詳細資料，但沒有虛構的geo:*GetWidget*權限時，就會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 geo:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 geo:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon 位置。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的使用IAM者marymajor嘗試使用主控台在 Amazon Location 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪 AWS 帳戶 問我的 Amazon 位置資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。對於支援以資源為基礎的政策或存取控制清單 (ACLs) 的服務，您可以使用這些政策授與人員存取您的資源。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon 位置是否支援這些功能，請參閱Amazon Location Service 如何與 IAM。

- 若要瞭解如何提供您所擁有資 AWS 帳戶 源的存取權，請參閱《IAM使用指南》中的〈提供存取權給您 AWS 帳戶 所擁有的其他IAM使用者〉。
- 若要瞭解如何將您的資源存取權提供給第三方 AWS 帳戶，請參閱《IAM使用指南》中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要瞭解如何透過聯合身分識別提供存取權，請參閱使用指南中的提供對外部驗證使用IAM者的存取權 (身分聯合)。
- 若要瞭解針對跨帳號存取使用角色與以資源為基礎的政策之間的差異，請參閱《使用IAM者指南》IAM中的〈跨帳號資源存取〉。

Amazon Location Service 中的事件回應

安全性是最高的優先事項 AWS。作為 AWS 雲端共同責任模式的一部分，您可以 AWS 管理符合最敏感安全性組織需求的資料中心和網路架構。身為 AWS 客戶，您有責任維護雲端中的安全性。這表示您可以透過可存取的 AWS 工具和功能來控制您選擇實作的安全性。

透過為雲端中執行的應用程式建立符合目標的安全性基準，您可以偵測可以回應的偏差。由於安全性事件回應可能是一個複雜的主題，因此我們建議您檢閱下列資源，以便更清楚瞭解事件回應 (IR) 和您的選擇對公司目標的影響：AWS安全性事件回應指南、AWS安全性最佳實務白皮書，以及AWS雲端採用架構 (AWSCAF)。

Amazon Location Service 中的記錄和監控

記錄和監控是事件回應的重要組成部分。它可讓您建立安全性基準，以偵測您可以調查和回應的偏差。透過為 Amazon 定 Location Service 實作記錄和監控，您可以維持專案和資源的可靠性、可用性和效能。

AWS 提供數種工具，可協助您記錄和收集資料以進行事件回應：

AWS CloudTrail

Amazon 定 Location Service 與整合 AWS CloudTrail，這是一項服務，可提供使用者、角色或 AWS 服務所採取的動作記錄。這包括來自 Amazon 定 Location Service 主控台的動作，以及以程式設計方式呼叫 Amazon 位置API操作。這些動作記錄稱為事件。如需詳細資訊，請參閱使用記錄和監控 Amazon 定 Location Service AWS CloudTrail。

Amazon CloudWatch

您可以使用 Amazon CloudWatch 收集和分析與 Amazon 定 Location Service 帳戶相關的指標。您可以啟用 CloudWatch 警示，在量度符合特定條件且達到指定臨界值時通知您。建立警示

時， CloudWatch會將通知傳送至您定義的 Amazon 簡易通知服務。如需詳細資訊，請參閱使用 Amazon 監控 Amazon 定 Location Service CloudWatch。

AWS Health 儀表板

使用AWS Health 儀表板，您可以驗證 Amazon 定 Location Service 的狀態。您也可以監視和檢視可能會影響 AWS 環境之任何事件或問題的歷史資料。如需詳細資訊，請參閱《AWS Health 使用者指南》https://docs.aws.amazon.com/health/latest/ug/what-is-aws-health.html。

Amazon 定 Location Service 的合規驗證

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上進行HIPAA安全與合規架構 — 本白皮書說明公司如何使用建立符合資格的應 AWS 用程HIPAA式。

> **Note**
>
> 並非所有 AWS 服務 人都HIPAA符合資格。如需詳細資訊，請參閱合HIPAA格服務參考資料。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準 AWS 服務 與技術研究所 (NIST)、支付卡產業安全標準委員會 () 和國際標準化組織 ()) 中保護安全控制指引的最佳實務作法，並將其對應至安全性控制。PCI ISO
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。

- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您因應各種合規性需求 PCIDSS，例如符合特定合規性架構所要求的入侵偵測需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon Location Service 的彈性

AWS 全球基礎架構是圍繞 AWS 區域 和可用區域建立的。 AWS 區域 提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 和可用區域的詳細資訊，請參閱AWS 全域基礎結構。

除了 AWS 全球基礎設施之外，Amazon Location 還提供多種功能，以協助支援您的資料彈性和備份需求。

Amazon Location Service 中的基礎設施安全

作為受管服務，Amazon 定 Location Service 受到 AWS 全球網路安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的API呼叫透過網路存取 Amazon 位置。使用者端必須支援下列專案：

- 傳輸層安全性 (TLS)。我們需要 TLS 1.2 並推薦 TLS 1.3。
- 具有完美前向保密（ ）的密碼套件，例如（短暫的迪菲-赫爾曼PFS）或DHE（橢圓曲線短暫迪菲-赫爾曼）。ECDHE現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與IAM主體相關聯的秘密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon 位置中的組態和漏洞分析

配置和 IT 控制是與您（我們的客戶）AWS 之間的共同責任。如需詳細資訊，請參閱 AWS 共用的責任模型。

預防跨服務混淆代理人

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在中 AWS，跨服務模擬可能會導致混淆的副問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況， AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

Amazon 定 Location Service 不會代表您使用其他服務的通話 AWS 服務，因此在這種情況下，您不需要新增這些保護。要了解更多有關混淆的副手，請參閱AWS Identity and Access Management 用戶指南中的混淆副問題。

Amazon 定位服務的安全最佳實務

Amazon 定 Location Service 提供許多安全功能，可在您開發和實作自己的安全政策時考慮。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

Amazon 定 Location Service 的 Detective 安全最佳實踐

Amazon 定位服務的下列最佳實務可協助偵測安全事件：

實作AWS監控工具

監控對於事件回應至關重要，並維持 Amazon 定 Location Service 資源和您的解決方案的可靠性和安全性。您可以透過多種可用的工具和服務來實作監視工具，AWS以監視您的資源和其他AWS服務。

例如，Amazon 可 CloudWatch 讓您監控 Amazon Location Service 的指標，並讓您設定警示，以便在指標符合您設定的特定條件且達到您定義的閾值時通知您。建立警示時，您可以設定 CloudWatch 為使用 Amazon 簡單通知服務傳送通知以提醒。如需詳細資訊，請參閱the section called “記錄和監控”。

啟用AWS記錄工具

記錄可提供使用者、角色或AWS服務在 Amazon 定 Location Service 服務中所採取的動作記錄。您可以實施日誌記錄工具，例 AWS CloudTrail 如收集有關檢測異常API活動的操作數據。

建立追蹤時，您可以設定 CloudTrail 為記錄事件。事件是在資源上或在資源內執行的資源操作記錄，例如向 Amazon Location 發出的請求、發出請求的 IP 地址、發出請求的人員、提出請求的時間以及其他資料。如需詳細資訊，請參閱《 AWS CloudTrail 使用指南》中的 < 記錄追蹤的資料事件 > 。

Amazon 定位服務的預防性安全最佳實 Location Service

Amazon 定位服務的下列最佳實務可協助預防安全事件：

使用安全連線

始終使用加密的連接，例如開頭的連接，以確`https://`保敏感信息在傳輸過程中的安全。

實作資源的最低權限存取

向 Amazon 位置資源建立自訂政策時，只授與執行任務所需的許可。建議您從最低限度的一組權限開始，並視需要授予其他權限。實施最低權限存取對於降低錯誤或惡意攻擊可能導致的風險和影響至關重要。如需詳細資訊，請參閱the section called “身分和存取權管理”。

使用全球唯一的設備 IDs IDs

對設備使用以下約定IDs。

- 裝置IDs必須是唯一的。
- 設備不IDs應該是秘密的，因為它們可以用作其他系統的外鍵。
- 設備不IDs應包含個人身份信息（PII），例如電話設備IDs或電子郵件地址。
- 設備不IDs應該是可預測的。建議使用不透明UUIDs的識別碼。

不要包含PII在裝置位置內容

傳送裝置更新 (例如，使用 DevicePositionUpdate) 時，請勿在中包含個人識別資訊 (PII)，例如電話號碼或電子郵件地址。`PositionProperties`

Amazon 定 Location Service 的最佳實踐

本主題提供協助您使用 Amazon 定位服務的最佳實務。雖然這些最佳實務可協助您充分利用 Amazon 定 Location Service，但並不代表完整的解決方案。您應該只遵循適用於您環境的建議。

主題

- 安全
- 資源管理
- 帳單與成本管理
- 配額和用量

安全

若要協助管理甚至避免安全風險，請考慮下列最佳做法：

- 使用聯合身分和IAM角色來管理、控制或限制對 Amazon 位置資源的存取。如需詳細資訊，請參閱《IAM使用指南》中的IAM最佳作法。
- 遵循最低權限原則，僅授與 Amazon 定 Location Service 資源所需的最低存取權限。如需詳細資訊，請參閱the section called “使用政策管理存取權”。
- 對於 Web 應用程式中使用的 Amazon Location Service 資源，請使用`aws:refererIAM`條件限制存取，限制允許清單中包含的網站以外的網站的使用。
- 使用監視和記錄工具來追蹤資源存取和使用情況。如需詳細資訊，請參閱the section called “記錄和監控”《 AWS CloudTrail 使用指南》中的《記錄軌跡的資料事件》。
- 在伺服器和瀏覽器之間傳輸資料時，請使用安全連線 (例如開頭的連線) `https://` 來增加安全性並保護使用者免受攻擊。

有關偵探和預防性安全最佳實踐的更多信息，請參閱上的主題the section called “安全最佳實務”。

資源管理

若要在 Amazon 定位服務中有效管理您的位置資源，請考慮下列最佳實務：

- 使用您預期使用者群中心的區域端點來改善使用體驗。如需有關區域端點的資訊，請參閱Amazon 位置區域和端點。
- 對於使用資料提供者的資源 (例如地圖資源和放置索引資源)，請務必遵循特定資料提供者的使用條款。如需詳細資訊，請參閱資料提供者。
- 通過為地圖，地點索引或路由的每個配置都有一個資源，最大限度地減少資源的創建。在一個區域內，每個資料提供者或地圖樣式通常只需要一個資源。大多數應用程式會使用現有資源，而不會在執行階段建立資源。

- 在單一應用程式中使用不同的資源 (例如地圖資源和路由計算機) 時，請在每個資源中使用相同的資料提供者，以確保資料相符。例如，您使用佈線計算器建立的佈線幾何圖形與使用地圖資源繪製的地圖上的街道對齊。

帳單與成本管理

若要協助管理您的成本和帳單，請考慮下列最佳做法：

- 使用監控工具 (例如 Amazon CloudWatch) 追蹤您的資源使用情況。您可以設定警示，在使用量即將超出指定限制時通知您。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的建立帳單警示以監控估計AWS費用。

配額和用量

您 AWS 帳戶 包含設定預設使用量限制的配額。您可以設定鬧鐘，在使用量接近上限時提醒您，也可以在需要時要求提高配額。如需如何使用配額的相關資訊，請參閱下列主題。

- Amazon 定 Location Service 配額
- 用 CloudWatch 於監控配額的使用情況
- 在 Amazon CloudWatch 使用者指南中視覺化您的服務配額和設定警示。

您可以創建警報，以便在接近超出限制時提前警告。我們建議您針對每個使用 Amazon 位置的每個 AWS 區域 配額設定警示。例如，您可以監控`SearchPlaceIndexForText`作業的使用情況，並在超過目前配額的 80% 時建立警示。

當您收到有關配額的警示警告時，您必須決定該怎麼做。您可能正在使用其他資源，因為您的客戶群已經增長。在這種情況下，您可能會想要求提高配額，例如增加該區域的API通話配額 50%。或者，也許您的服務中存在錯誤，導致您向 Amazon 位置撥打其他不必要的電話。在這種情況下，您需要解決服務中的問題。

文件歷史記錄

下表說明 Amazon 定 Location Service 的說明文件。如需有關更新的通知，您可以訂閱 RSS 摘要。

變更	描述	日期
Amazon 定 Location Service 發布了一個新的 SDK JavaScript	為了讓開發 Amazon 位置應用程式更容易在網路前端，Amazon 位置新增了一個新的開放原始碼開發套件，該開發套件支援適用於 JavaScript v3 的 AWS 開發套件，簡化身份驗證和使用 GeoJSON。如需主要資訊，請參閱 Amazon 位置開發套件。	2023 年 7 月 6 日
Amazon 定 Location Service 發布 API 密鑰以實現正式推出	Amazon 位置新增對位置和路由的支援，並宣布 API 金鑰功能的正式推出。如需詳細資訊，請參閱使用 API 金鑰。	2023 年 7 月 6 日
Amazon 定 Location Service 為位置更新添了 Amazon EventBridge 活動	Amazon 位置添加了對將跟踪器位置更新事件發送到 EventBridge. 如需詳細資訊，包括如何啟用追蹤器的事件，請參閱使 EventBridge用.	2023 年 7 月 6 日
Amazon 位置將元數據添加到地理圍欄	您現在可以使用 Amazon 位置 API 將中繼資料屬性新增至地理圍欄。這些都與您的地理圍欄一起存儲，並包含在與 Amazon 地理圍欄相關的事件中。 EventBridge如需詳細資訊，請參閱繪製地理圍欄和建立事件規則。	2023 年 6 月 15 日

Amazon 位置增加了類別的地方	Amazon 地點在搜索結果中添加類別，並按類別過濾結果。如需詳細資訊，請參閱類別和篩選。	2023 年 6 月 15 日
Amazon 位置介紹政治觀點	Amazon 位置為某些地圖樣式添加了政治觀點。如需詳細資訊，請參閱政治觀點。	2023 年 5 月 23 日
Amazon 位置引入了新的演示和示例站點	Amazon 位置宣布了一個新的網站，可讓您訪問 Amazon 位置演示和示例。有關更多信息，請參閱 Amazon 位置演示站點。	2023 年 5 月 3 日
Amazon 位置引入了更長的路線 CalculateRouteMatrix	Amazon 位置現在允許使用 HERE 資料提供者建立的路由矩陣路由的無限長度路由。如需詳細資訊，請參閱較長的路線規劃。	2023 年 4 月 24 日
Amazon 位置文件新增資料供應商的功能差異	Amazon Location 文件已更新，其中包含地圖、地點搜尋和路由中每個資料提供者之間差異的相關資訊。如需詳細資訊，請參閱資料提供者的圖徵。	2023 年 3 月 30 日
Amazon 位置開放資料地圖一般可用性	根據日光地圖，Amazon 定 Location Service 資料供應商和樣式的一般可用性。OpenStreetMap如需詳細資訊，請參閱開放資料。	2023 年 3 月 7 日

Amazon 位置在預覽中添加新的授權方法	Amazon 定 Location Service 會在預覽模式下新增 API 金鑰做為匿名使用者的新授權方法。如需詳細資訊，請參閱允許未經驗證的訪客使用 API 金鑰存取您的應用程式。	2023 年 2 月 23 日
使用最新 IAM 最佳實務更新的 Amazon 位置文件	Amazon 定 Location Service 文件已更新，以符合最新的最AWS Identity and Access Management佳實務。如需詳細資訊，請參閱 Amazon 定 Location Service 中的安全性。	2023 年 1 月 26 日
Amazon 定 Location Service 新增 GrabMaps 為東南亞的資料提供者	Amazon 位置 GrabMaps 作為東南亞的數據提供商引入。如需詳細資訊，請參閱GrabMaps。	2023 年 1 月 10 日
Amazon 定 Location Service 開放數據地圖預覽	根據日光地圖，在公開預覽中新增 Amazon 位置資料提供者和樣式。 OpenStreetMap如需詳細資訊，請參閱開放資料 (預覽)。	2022 年 12 月 15 日
全新 HERE 衛星圖像樣式	使用 HERE 作為數據提供商（HERE 衛星圖像和 HERE 混合地圖樣式）的地圖添加了兩種新的地圖樣式。如需詳細資訊，請參閱 HERE 地圖樣式。	2022 年 10 月 25 日
地址中的單位	Amazon 定 Location Service 現在支持地址中的單位，例如「123 主街，美國任何城市 3B 公寓」。	2022 年 9 月 20 日

通過 ID 獲取地點	Amazon 定 Location Service 現在包含使用操作尋找SearchPlaceIndexForSuggestions 操作建議的確切位置的GetPlace支援。請參閱使用自動完成。	2022 年 9 月 20 日
IAM 政策的其他條件金鑰	Amazon 定 Location Service 現在支援額外的條件金鑰，可讓您在 IAM 政策中為特定地理圍欄或裝置設定存取權。請參閱條件關鍵字。	2022 年 8 月 23 日
圓形地理圍欄	Amazon 定 Location Service 現在支援定義為具有中心點和半徑的圓形的地理圍欄，以便在裝置位於某個位置的特定距離內時取得事件。請參閱新增循環地理圍欄。	2022 年 8 月 11 日
結合 API 參考資料	Amazon 定 Location Service 現在有單一 API 參考指南，而不是針對每個子服務的單獨指南。如需 API 的詳細資訊，請參閱 Amazon 位置 API。	2022 年 7 月 7 日
Service Quotas 整合	Amazon 位置現在已與 Service Quotas 整合，可讓您透過 AWS Management Console或使用AWS CLI.	2022 年 7 月 6 日
更新概念文檔章節	Amazon 位置概念章節已更新為 Amazon 位置的用戶提供更多信息。	2022 年 4 月 22 日

新安卓快速入門教程	已添加了一個針對使用 Kotlin 進行 Android 開發的新快速入門教程，以使開發人員快速啟動並運行。	2022 年 4 月 15 日
新的 HERE 地圖風格	使用 HERE 做為資料提供者的地圖新增了兩種新的地圖型式。如需詳細資訊，請參閱 HERE 地圖樣式。	2022 年 3 月 15 日
重組文檔添加的代碼示例和教程	此開發人員指南經過重新架構，可以更輕鬆地找到主題，包括新的快速入門和程式碼範例章節。	2022 年 2 月 25 日
適用於追蹤器的準確性位置篩選	現在，您可以在創建跟踪器資源時使用基於準確性的過濾器。	2021 年 12 月 7 日
自動完成放置索引	現在，您可以在搜索地點索引時使用自動完成功能。	2021 年 12 月 6 日
使用地圖的全新 Amplify 教學	提供了一個新的教程，顯示如AWS Amplify何使用在 Web 應用程序中顯示地圖。本教學可在使用具有 Amazon 定 Location Service 的 Amplify 程式庫中取得。	2021 年 11 月 24 日
放置查詢副檔名	Amazon 定 Location Service 現在支援在地理編碼或反向地理編碼時設定結果的偏好語言，並將時區和其他資訊新增至結果。有關地理編碼和反向地理編碼的詳細資訊，請參閱地理編碼、反向地理編碼和搜尋。	2021 年 11 月 16 日

追蹤器位置篩選	Amazon 定 Location Service 為追蹤器新增職位篩選功能，協助您控制成本。此功能會先篩選出裝置上的某些位置更新，然後再根據地理圍欄儲存或評估更新。如需有關位置篩選的詳細資訊，請參閱追蹤器。	2021 年 10 月 5 日
更新作業	下列操作已新增至 Amazon 定 Location Service API 參考：UpdateMapUpdatePlaceIndexUpdateRouteCalculator、、 UpdateGeofenceCollection、和UpdateTracker。	2021 年 7 月 19 日
教程更新：Amazon Aurora PostgreSQL用戶定義函數	已新增一個新的教學課程，說明如何將使用Amazon Aurora PostgreSQL者定義函數與 Amazon Location 搭配使用，以驗證、清理和豐富地理空間資料。	2021 年 7 月 19 日
AWS CloudFormation 資源	Amazon 位置現在支援在 AWS 資源中建立下列 CloudFormation資源類型：AWS::Location::Map AWS::Location::PlaceIndex AWS::Location::RouteCalculator 、AWS::Location::Tracker 、AWS::Location::TrackerConsumer 、、和AWS::Location::GeofenceCollection 。	2021 年 6 月 7 日

標記資源	您現在可以在 Amazon Location 資源中新增標籤，以協助管理、識別、組織、搜尋和篩選資源。	2021 年 6 月 1 日
一般可用性	Amazon 定 Location Service 開發人員文件的正式發行版本：區域和端點以及服務配額已更新。	2021 年 6 月 1 日
埃斯里圖像	Amazon 位置現在支持使用 Esri 地圖樣式：Esri 圖像。有關更多信息，請參閱 Esri 網站上的 Esri 世界圖像。	2021 年 6 月 1 日
計算路線	您現在可以使用 Amazon Location 路線計算器，根據道路網路和所選資料供應商的即時交通資訊，計算 up-to-date 路線和預估行駛時間。	2021 年 6 月 1 日
AWS KMS客戶管理的靜態資料金鑰加密	Amazon Location 現在支援使用您建立、擁有和管理的對稱客戶受管金鑰，以透過現有 AWS擁有的加密新增第二層加密。	2021 年 6 月 1 日
公開預覽版	公開預覽文件的初始版本。	2020 年 12 月 16 日
教學課程更新：顯示地圖	使用 Android 和 iOS 顯示地圖 MapLibre 的教學課程已更新為使用 MapLibre 原生 SDK。	2020 年 3 月 17 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。