aws

开发人员指南

Amazon Location Service

Amazon Location Service: 开发人员指南

Table of Contents

欢迎使用 Amazon Location Service

欢迎阅读 Amazon Location Service Service 开发者指南。

以下主题可帮助您开始使用文档，具体取决于您正在尝试执行的操作。

获取 Amazon Location

- 了解 Amazon Location 中的概念。
- 深入探讨 如何使用 Amazon Location Service 章中的功能。
- 在 Amazon Location 演示网站上查看演示应用程序。
- 如果您已经有 AWS 账户，则可以使用 Amazon Location Service 控制台亲自探索其功能。

以开发者身份使用 Amazon Location

- 使用构建您的第一个应用程序 快速入门。
- 在 如何使用 Amazon Location Service 章中了解 Amazon Location Service 的各种功能是如何运作的。
- 请参阅 用 Amazon Location 开发 章中可供您使用的开发工具包和工具。
- 查看可在自己的应用程序中使用的代码示例和教程。您也可以访问 Amazon Location 演示网站示例页面，查找可按功能、语言或平台筛选的示例。
- 在 API 参考指南中获取有关 Amazon Location API 的信息。

Amazon Location Service 是什么？

Amazon Location Service 允许您向应用程序添加位置数据和功能，其中包括地图、兴趣点、地理编码、路由、地理围栏和跟踪等功能。Amazon Location 使用来自全球值得信赖的提供程序 Esri、Grab 和 HERE 的高质量数据，提供基于位置的服务 (LBS)。借助经济实惠的数据、跟踪和地理围栏功能以及内置的健康监控指标，您可以构建复杂的支持定位的应用程序。

借助 Amazon Location，您可以保留对组织数据的控制权。Amazon Location 会删除客户元数据和账户信息，从而匿名化发送给数据提供程序的所有查询。此外，敏感的跟踪和地理围栏位置信息（例如设施、资产和人员位置）完全不会离开您的 AWS 账户。这可以帮助您保护敏感信息免受第三方侵害，保护用户隐私，并降低应用程序的安全风险。在 Amazon Location 中，Amazon 和第三方无权出售您的数据或将其用于广告。

Amazon Location 已与亚马逊AWS CloudTrail CloudWatch EventBridge、亚马逊和 AWS Identity and Access Management (IAM) 等服务完全集成。Amazon Location 通过数据集成简化了您的开发工作流程，并通过内置的监控、安全性和合规性功能将应用程序快速跟踪到生产阶段。

有关亮点、产品详细信息和定价，请参阅 Amazon Location Service 服务页面。

Amazon Location 中的主要功能

Amazon Location 提供以下功能：

映射

Amazon Location Service 地图可让您可视化位置信息，并且是许多基于位置的服务功能的基础。Amazon Location Service 提供来自全球位置数据提供程序 Esri、Grab 和 HERE 的不同样式的地图图块，以及开放数据地图。

位数

Amazon Location Service 地点允许您将搜索功能集成到应用程序中，将地址转换为经纬度的地理坐标（地理编码），以及将坐标转换为街道地址（反向地理编码）。Amazon Location Service 从 Esri、Grab 和 HERE 获取高质量的地理空间数据，以支持 Places 函数。

路由

Amazon Location Service Routes 允许您根据 up-to-date 道路和实时交通信息查找路线并估算行程时间。构建功能，使您的应用程序能够请求任意两个地点之间的行驶时间、距离和方向。计算用于路线规划的路由矩阵的时间和距离。

地理围栏

Amazon Location Service 地理围栏可让您的应用程序在设备进入或离开定义的地理边界（称为地理围栏）时进行检测和采取行动。检测到地理围栏漏洞 EventBridge 时，自动向 Amazon 发送进入或退出事件。这使您可以启动下游操作，例如向目标发送通知。

跟踪器

Amazon Location Service 跟踪器允许您检索正在运行支持跟踪的应用程序所在的设备当前位置和历史位置。您还可以将跟踪器与 Amazon Location Service 地理围栏链接起来，根据地理围栏自动

评估来自您设备的位置更新。跟踪器可以在存储或根据地理围栏进行评估之前筛选尚未移动的位置更新，从而帮助您降低成本。

当您使用跟踪器时，被跟踪设备上的敏感位置信息不会从您的 AWS 账户中删除。这有助于保护敏感信息免受第三方侵害，保护用户隐私，降低安全风险。

您可以与 Amazon Location 一起使用的服务

将以下服务与 Amazon Location Service 一起使用。

集成的监控和管理

Amazon Location Service 与亚马逊 CloudWatch和亚马逊集成AWS CloudTrail， EventBridge 可实现高效的监控和数据管理：

- Amazon CloudWatch — 查看有关服务使用情况和运行状况的指标，包括请求、延迟、故障和日志。有关更多信息，请参阅 the section called “使用监控 CloudWatch”。
- AWS CloudTrail——记录和监控您的 API 调用，包括用户、角色或 AWS 服务所采取的操作。有关更多信息，请参阅 the section called “ CloudTrail 与 Amazon 位置一起使用”。
- Amazon EventBridge — 启用事件驱动的应用程序架构，以便您可以使用AWS Lambda函数激活应用程序和工作流程的其他部分。有关更多信息，请参阅 the section called “对事件做出反应 EventBridge”。

开发人员工具

Amazon Location Service 为开发人员提供了各种工具，供他们构建支持定位的应用程序。其中包括标准 AWS SDK、移动和 Web SDK，以及将它们与开源库（例如）组合在一起的示例代码。MapLibre使用 Amazon Location Service 控制台了解资源，并开始使用可视化交互式学习工具。

Amazon Location Service 快速入门

开始使用 Amazon Location Service 的最有效方法是使用 Amazon Location 控制台。您可以使用浏览页面创建和管理您的资源并试用 Amazon Location 功能。

> **Note**
>
> 要使用 Amazon Location Service 控制台或按照本教程的其余部分进行操作使用 Amazon Location Service 的先决条件，需要您先完成操作，包括创建 AWS 账户并允许访问亚马逊位置。

要开始学习 Amazon Location API，请使用以下教程创建一个显示交互式地图并使用搜索功能的简单应用程序。本教程有三个版本：一个版本向您展示了如何使用创建简单网页 JavaScript，第二个版本显示了使用Kotlin的Android应用程序的相同内容，第三个版本显示了使用Swift的iOS应用程序的相同内容。

主题

- 创建 Web 应用程序
- 创建 Android 应用程序
- 创建 iOS 应用程序

创建 Web 应用程序

在本部分中，您将创建一个静态网页，其中包含地图并能够在某个位置进行搜索。首先，您将创建您的 Amazon Location 资源并为您的应用程序创建API密钥。

主题

- 为您的应用程序创建 Amazon Location 资源
- 为应用程序设置身份验证
- HTML为您的应用程序创建
- 向应用程序添加交互式地图
- 将搜索功能添加到应用程序
- 查看最终应用程序
- 接下来做什么

为您的应用程序创建 Amazon Location 资源

如果您还没有，则必须创建应用程序将使用的 Amazon Location 资源。在这里，您可以创建用于在应用程序中显示地图的地图资源，以及用于在地图上搜索位置的地点索引。

向您的应用程序添加位置资源

1. 选择要使用的地图样式。

   a. 在 Amazon Location 控制台的地图页面上，选择创建地图以预览地图样式。

   b. 为新地图资源添加名称和描述。记下您用于地图资源的名称。在本教程的后面部分创建脚本文件时，您将需要它。

   c. 选择一张地图。

   > **Note**
   >
   > 选择地图样式还会选择要使用的地图数据提供程序。如果您的应用程序正在跟踪或路由您在业务中使用的资产，例如送货车辆或员工，则您只能HERE用作地理定位提供商。想要了解更多信息，请参阅 AWS 服务条款的第 82 节。

   d. 同意 Amazon Location 条款和条件，然后选择创建地图。您可以与所选地图进行交互：放大、缩小或向任意方向平移。

   e. 记下为您的新地图资源显示的 Amazon 资源名称 (ARN)。在本教程的后面部分，您将使用它来创建正确的身份验证。

2. 选择要使用的地点索引。

   a. 在 Amazon Location 控制台的地点索引页面上，选择创建地点索引。

   b. 为新的地点索引资源添加名称和描述。记下地点索引资源使用的名称。在本教程的后面部分创建脚本文件时，您将需要它。

   c. 选择数据提供程序。

   > **Note**
   >
   > 在大多数情况下，请选择与您已选择的地图提供程序相匹配的数据提供程序。这有助于确保搜索结果与地图相匹配。

如果您的应用程序正在跟踪或路由您在业务中使用的资产，例如送货车辆或员工，则您只能HERE用作地理定位提供商。想要了解更多信息，请参阅 AWS 服务条款的第 82 节。

d. 选择数据存储选项。在本教程中，不存储结果，因此您可以选择否，仅限一次性使用。

e. 同意 Amazon Location 条款和条件，然后选择创建地点索引。

f. ARN记下为您的新地点索引资源显示的。在本教程的下一部分，您将使用它来创建正确的身份验证。

为应用程序设置身份验证

您在本教程中创建的应用程序是匿名使用的，这意味着您的用户无需登录 AWS 即可使用该应用程序。但是，默认情况下，Amazon Location Service APIs 需要身份验证才能使用。您可以使用 Amazon Cognito 或API密钥为匿名用户提供身份验证和授权。在本教程中，您将创建用于示例应用程序的API密钥。

> **Note**
>
> 有关使用API密钥或 Amazon Cognito 与 Amazon Location Service 配合使用的更多信息，请参阅。授予对 Amazon Location Service 的访问权限

为您的应用程序设置身份验证

1. 前往 Amazon Location 控制台，然后从左侧菜单中选择API按键。
2. 选择创建 API 密钥。

> **Important**
>
> 您创建的API密钥必须 AWS 账户 与您在上一节中创建的 Amazon Location Service 资源位于同一 AWS 区域。

3. 在“创建API密钥”页面上，填写以下信息。

   - 名称-密API钥的名称，例如`MyWebAppKey`。
   - 资源——选择您在上一部分中创建的 Amazon Location Map 和地点索引资源。您可以通过选择添加资源来添加多个资源。这将允许API密钥与这些资源一起使用。

- 操作-指定您要使用此API密钥授权的操作。您必须至少选择 geo: GetMap * 和 geo:，SearchPlaceIndexfForPosition这样本教程才能按预期运行。
- 您可以选择在API密钥中添加描述、到期时间或标签。您也可以添加反向链接（例如 `*.example.com`），将密钥限制为只能在特定域中使用。这意味着本教程只能在该域中运行。

> **Note**
>
> 建议您通过设置到期时间或反向链接（如果不是两者兼而有之）来保护API密钥的使用。

4. 选择创建API密钥以创建API密钥。
5. 选择 Show API ke y，然后复制密钥值以供本教程稍后使用。它将采用 `v1.public.a1b2c3d4...` 形式。

> **Important**
>
> 在本教程的后面部分，在为应用程序编写代码时，您将需要此密钥。

HTML为您的应用程序创建

在本教程中，您将创建一个嵌入地图的静态HTML页面，并允许用户查找地图上某个位置的内容。该应用程序将由三个文件组成：一个HTML用于网页CSS的文件和文件，以及一个用于创建地图并响应用户互动和地图事件的代码的 JavaScript (.js) 文件。

首先，让我们创建将用于应用程序的HTML和CSS框架。这将是一个简单的页面，其中包含一个 用于存放地图容器的<pre>元素和一个用于显示查询JSON响应的元素。

为快速入门应用程序创建 HTML

1. 创建名为 `quickstart.html` 的新文件。
2. 在您选择的文本编辑器或环境中编辑文件。将以下内容HTML添加到文件中。

```
<!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>
```

它HTML有一个指向你将在下一步中创建的CSS文件的指针、应用程序的一些占位符元素和一些解释性文本。

在本教程稍后的部分中，您将使用两个占位符元素。第一个是 <div id="map> 元素，它将保存地图控件。第二个是 <pre id="response"> 元素，它将显示在地图上搜索的结果。

3. 保存文件。

现在CSS为网页添加。这将为应用程序设置文本和占位符元素的样式。

为快速入门应用程序创建 CSS

1. 在与上一个步骤中创建的 quickstart.html 文件相同的文件夹中创建一个名 main.css 为的新文件。
2. 使用您想要使用的任何编辑器编辑该文件。将以下文本添加到文件中。

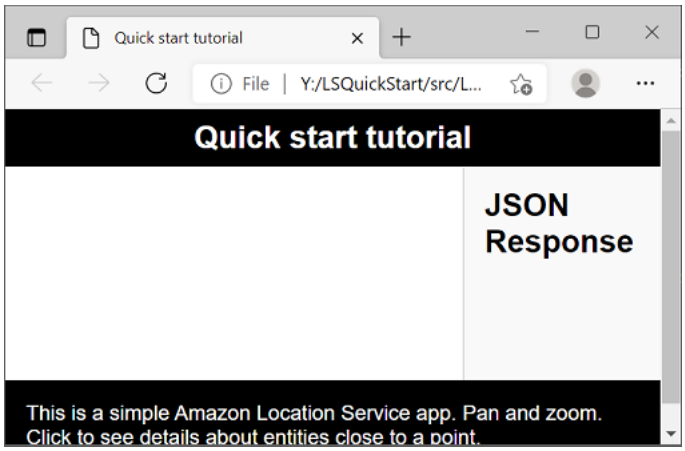
```
* {
  box-sizing: border-box;
  font-family: Arial, Helvetica, sans-serif;
}
```

```
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;
}
```

这将设置地图填满应用中未被其他任何内容占用的空间，设置我们的响应区域占应用宽度的30%，并为标题和解释性文本设置颜色和样式。

3. 保存该文件。
4. 现在，您可以在浏览器中查看 `quickstart.html` 文件以查看应用程序的布局。



接下来，您将向应用程序中添加地图控件。

向应用程序添加交互式地图

现在您已经有了框架和一个 div 占位符，您可以将地图控件添加到您的应用程序中。本教程使用 MapLibre GL JS 作为地图控件，从 Amazon Location Service 获取数据。您还将使用JavaScript 身份验证助手来简化使用API密钥签名到 Amazon 营业地点APIs的电话。

向应用程序添加交互式地图

1. 打开您在上一部分中创建的 `quickstart.html` 文件。
2. 添加对所需库的引用，以及将要创建的脚本文件。**`green`** 中显示了您需要进行的更改。

```
<!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>
```

这将以下依赖项添加到您的应用程序：

- MapLibre GL JS。该库和样式表包括一个显示地图图块的地图控件，并包括平移和缩放等交互性。该控件还允许扩展，例如在地图上绘制自己的要素。
- Amazon Location 客户端。这为获取地图数据和在地图上搜索地点所需的 Amazon Location 功能提供了接口。Amazon Location 客户端基于 f AWS SDK or JavaScript v3。
- Amazon Location 认证帮助程序。这为使用API密钥或 Amazon Cognito 对 Amazon Location Service 进行身份验证提供了有用的功能。

此步骤还添加了对 `main.js` 的引用，接下来您将创建该引用。

3. 保存 `quickstart.html` 文件。
4. 在与您的HTML和文件同一个文件夹`main.js`中创建一个名为的新CSS文件，然后将其打开进行编辑。

5. 在您的文件中添加以下脚本：中的文字 *red* 应替换为您之前创建的API键值、地图资源名称和地点资源名称以及您所在地区的区域标识符（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();
```

此代码设置 Amazon Location 资源，然后配置和初始化 MapLibre GL JS 地图控件，并将其放置在带有 ID 的 元素中。map

理解 initializeMap() 函数很重要。它会创建一个新的 MapLibre 地图控件（map在mlglMap本地调用，但在代码的其余部分中调用），用于在应用程序中渲染地图。

```
  // Initialize the map
  const mlglMap = new maplibregl.Map({
    container: "map", // HTML element ID of map element
```

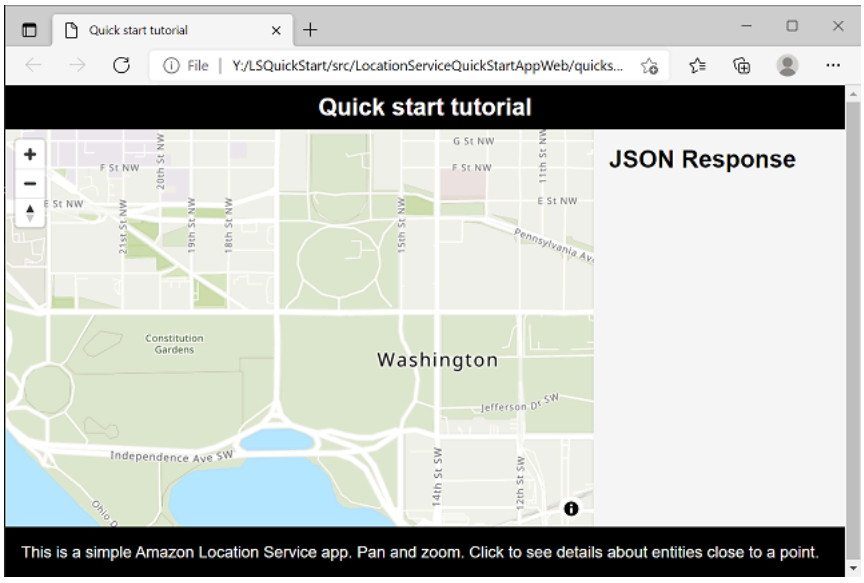
```
    center: [-77.03674, 38.891602], // Initial map centerpoint
    zoom: 16, // Initial map zoom
    style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=${apiKey}`, // Defines the appearance of the map and authenticates
 using an API key
  });
```

创建新的 MapLibre 地图控件时，您传递的参数表示地图控件的初始状态。这里，我们设置以下参数。

- HTML容器，它使用我们的 map div 元素HTML。
- 地图的初始中心到华盛顿特区某一点。
- 缩放级别为 16（放大到邻里或街区级别）。
- 用于地图的样式，它 MapLibre 提供了URL用于获取地图图块和其他信息以渲染地图。请注意，这URL包括您的身份验证API密钥。

6. 保存您的 JavaScript 文件，然后用浏览器将其打开。现在，您的页面上有一张地图，可以在其中使用平移和缩放操作。

> **Note**
>
> 您可以使用此应用程序来查看 MapLibre 地图控件的行为方式。在使用拖动操作时，可以尝试使用 Ctrl 或 Shift 来查看与地图交互的其他方式。所有这些功能都是可定制的。



您的应用程序已接近完成。在下一部分中，您将负责在地图上选择一个位置，并显示所选位置的地址。您还将在页面JSON上显示结果，以查看完整结果。

将搜索功能添加到应用程序

应用程序的最后一步是在地图上添加搜索功能。在这种情况下，您将添加反向地理编码搜索，以便在某个位置找到项目。

> **Note**
>
> Amazon Location Service 还提供按名称或地址进行搜索以在地图上查找地点位置的功能。

向应用程序添加搜索功能

1. 打开您在上一部分中创建的 `main.js` 文件。
2. 修改 `main` 函数，如图所示。**`green`** 中显示了您需要进行的更改。

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

此代码首先设置 Amazon Location 身份验证助手以使用您的API密钥。

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

然后，它会使用该身份验证帮助程序以及您用来创建新的 Amazon Location 客户端的区域。

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

接下来，代码会响应用户在地图控件上选择一个位置。它通过捕捉 MapLibre 提供的事件来做到这一点`click`。

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

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

该 MapLibre `click`事件提供的参数包括用户选择的纬度和经度 (`e.lngLat`)。在 `click` 事件中，代码创建 `searchPlaceIndexForPositionCommand` 以查找给定纬度和经度的实体。

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

这里，`IndexName` 是您之前创建的地点索引资源的名称，`Position` 是要搜索的纬度和经度，`Language` 是搜索结果的首选语言，`MaxResults` 告诉 Amazon Location 最多只返回五个结果。

其余代码检查是否存在错误，然后在名为 `response` 的 `<pre>` 元素中显示搜索结果，并在警告框中显示最上面的结果。

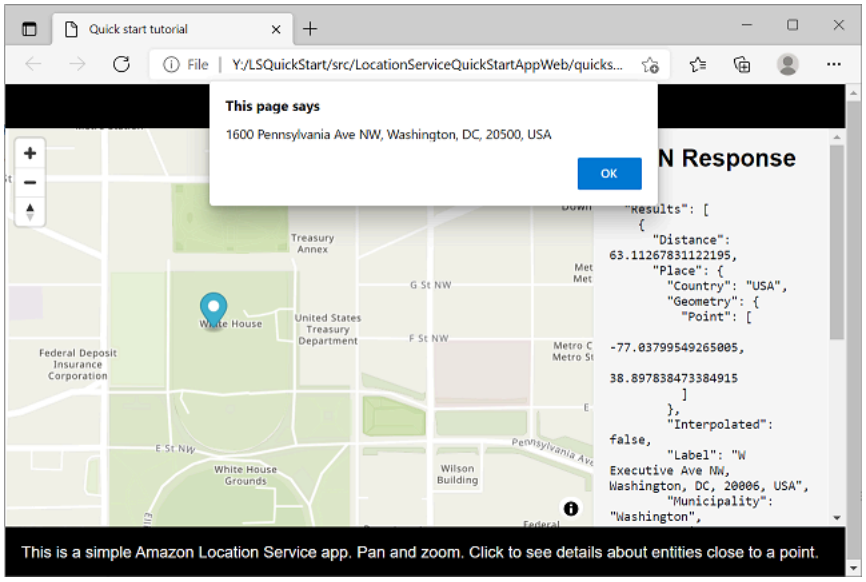
3. （可选）如果您现在在浏览器中保存并打开 `quickstart.html` 文件，则在地图上选择一个位置将显示所选地点的名称或地址。
4. 应用程序的最后一步是使用该 MapLibre 功能在用户选择的位置添加标记。按如下方式修改 `main` 函数。**`green`** 中显示了您需要进行的更改。

```
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);

...
```

此代码声明了一个 marker 变量，用户每次选择地点时都会填充该变量，显示他们选择的位置。一旦通过 .addTo(map); 添加到地图上，标记就会被地图控件自动渲染。该代码还会检查之前的标记并将其删除，因此屏幕上一次只有 1 个标记。

5. 保存 main.js 文件，然后在浏览器中打开该 quickstart.html 文件。您可以像以前一样平移和缩放地图，但是现在，如果您选择一个位置，您将看到有关所选位置的详细信息。



您的快速入门应用程序已完成。本教程向您展示了如何创建静态HTML应用程序，该应用程序具有以下特性：

- 创建用户可以与之交互的地图。
- 处理地图事件 (`click`)。
- 调用 Amazon Loc API ation Service，专门使用在某个位置搜索地图`searchPlaceIndexForPosition`。
- 使用 MapLibre 地图控件添加标记。

查看最终应用程序

此应用程序的最终源代码包含在本部分中。你也可以在上找到最终的项目 GitHub。

[您还可以找到使用 Amazon Cognito 而不是API密钥的应用程序版本。 GitHub]

Overview

选择每个选项卡，查看本快速入门教程中文件的最终源代码。

文件如下：

- quickstart.html — 应用程序的框架，包括地图和搜索结果的HTML元素持有者。
- main.css——应用程序的样式表。

- main.js——应用程序的脚本，用于对用户进行身份验证、创建地图和在 `click` 事件上进行搜索。

quickstart.html

快速入门应用程序的HTML框架。

```
<!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

快速入门应用程序的样式表。

```
* {
  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

快速入门应用程序的代码。中的文字 *red* 应替换为相应的 Amazon Location 对象名称。

```
// 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();
```

接下来做什么

您已经完成了快速入门教程，应该对如何使用 Amazon Location Service 来构建应用程序有所了解。要了解更多关于 Amazon Location 的信息，您可以查看以下资源：

- 深入了解 Amazon Location Service 的概念
- 获取有关如何使用 Amazon Location 功能的更多信息
- 查看使用 Amazon Location 的代码示例，了解如何扩展此示例并构建更复杂的应用程序

创建 Android 应用程序

在本节中，您将创建一个具有地图、在某个位置进行搜索和在前景中进行跟踪的Android应用程序。首先，您将为应用程序创建您的亚马逊位置资源、亚马逊 Cognito 身份和 API 密钥。

主题

- 为您的应用程序创建 Amazon Location 资源
- 为应用程序设置身份验证
- 创建基本的安卓应用程序
- 向应用程序添加交互式地图
- 向您的应用程序添加反向地理编码搜索
- 向您的应用程序添加追踪功能
- 接下来做什么

为您的应用程序创建 Amazon Location 资源

如果您还没有，则必须创建应用程序将使用的 Amazon Location 资源。在这里，您可以创建用于在应用程序中显示地图的地图资源、用于在地图上搜索位置的地点索引以及用于在地图上跟踪对象的跟踪器。

向您的应用程序添加位置资源

1. 选择要使用的地图样式。

   a. 在 Amazon Location 控制台的地图页面上，选择创建地图以预览地图样式。

   b. 为新地图资源添加名称和描述。记下您用于地图资源的名称。在本教程的后面部分创建脚本文件时，您将需要它。

   c. 我们建议您为地图选择 HERE 地图风格。

   > **Note**
   >
   > 选择地图样式还会选择要使用的地图数据提供程序。如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则只能使用 HERE 作为地理位置提供程序。想要了解更多信息，请参阅 AWS 服务条款的第 82 节。

   d. 同意 Amazon Location 条款和条件，然后选择创建地图。您可以与所选地图进行交互：放大、缩小或向任意方向平移。

   e. 请记下新地图资源显示的 Amazon 资源名称 (ARN)。在本教程的后面部分，您将使用它来创建正确的身份验证。

2. 选择要使用的地点索引。

   a. 在 Amazon Location 控制台的地点索引页面上，选择创建地点索引。

   b. 为新的地点索引资源添加名称和描述。记下地点索引资源使用的名称。在本教程的后面部分创建脚本文件时，您将需要它。

   c. 选择数据提供程序。

   > **Note**
   >
   > 在大多数情况下，请选择与您已选择的地图提供程序相匹配的数据提供程序。这有助于确保搜索结果与地图相匹配。

如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则只能使用 HERE 作为地理位置提供程序。想要了解更多信息，请参阅 AWS 服务条款的第 82 节。

d. 选择数据存储选项。在本教程中，不存储结果，因此您可以选择否，仅限一次性使用。

e. 同意 Amazon Location 条款和条件，然后选择创建地点索引。

f. 记下为您的新地点索引资源显示的 ARN。在本教程的下一部分，您将使用它来创建正确的身份验证。

3. 使用 Amazon Location 控制台创建追踪器。

a. 打开 Amazon Location Service 控制台。

b. 在左侧导航窗格中，选择跟踪器。

c. 选择创建跟踪器。

d. 填写所有必填字段。

e. 在 “位置筛选” 下，我们建议您使用默认设置：TimeBased。

f. 选择 “创建跟踪链接” 完成操作。

为应用程序设置身份验证

您在本教程中创建的应用程序是匿名使用的，这意味着您的用户无需登录 AWS 即可使用该应用程序。但是，Amazon Location Service API 需要身份验证才能使用。您可以使用 API 密钥或 Amazon Cognito 为匿名用户提供身份验证和授权。本教程将使用 Amazon Cognito 和 API 密钥对您的应用程序进行身份验证。

> **Note**
>
> 有关将 Amazon Cognito 或 API 密钥用于 Amazon Location Service，请参阅 授予对 Amazon Location Service 的访问权限。

以下教程向您展示如何为在中创建的地图、地点索引和追踪器设置身份验证，以及如何为 Amazon Location 设置权限。

设置身份验证

1. 导航至 Amazon Location 控制台，然后从左侧菜单中选择 API 密钥。

2. 点击 “创建 API 密钥”。请记住，API 密钥必须与之前创建的 Amazon Location Service 资源位于相同的 AWS 账户和区域。
3. 在 “创建 API 密钥” 页面上填写所需的详细信息：
   - 名称：为您的 API 密钥提供一个名称，比如`MyAppKey`。
   - 资源：选择之前创建的 Amazon Location Service 地图和地点索引资源。您可以通过选择 “添加资源” 来添加多个资源。这允许将 API 密钥用于指定的资源。
   - 操作：指定此 API 密钥的授权操作。至少要选择`geo:GetMap`和，`geo:SearchPlaceIndexForPosition`以确保教程按预期运行。
   - （可选）您可以添加描述、到期时间、标签或反`https://www.example.com`向链接，例如，将密钥的使用限制在特定域中，从而使本教程只能在该域中运行。
4. 点击创建 API 密钥以生成 API 密钥。
5. 选择 “显示 API 密钥”，然后复制密钥值以`v1.public.a1b2c3d4`供以后在本教程中使用。

创建用于跟踪的 IAM 策略

1. 使用您具有管理员权限的账户，通过以下网址登录到 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在内容窗格中，选择创建策略。
4. 选择 JSON 选项，然后将此 JSON 策略复制并粘贴到 JSON 文本框中。

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

这是跟踪的政策示例。要将示例用于您自己的策略，请替换RegionAccount、和TrackerName占位符。

> **Note**
>
> 虽然未经身份验证的身份池旨在在不安全的互联网站点上公开，但请注意，它们将被交换为标准的、有时间限制的 AWS 证书。
> 适当确定与未经身份验证的身份池关联的 IAM 角色的范围很重要。有关在亚马逊定位服务的 Amazon Cognito 中使用策略和适当确定政策范围的更多信息，请参阅授予亚马逊定位服务访问权限。

5. 在“查看并创建”页面上，为策略名称字段提供一个名称。查看您的策略授予的权限，然后选择创建策略以保存您所做的工作。

将在托管策略列表中显示新策略，并已准备好附加该策略。

为追踪设置身份验证

1. 在 Amazon Cognito 控制台中为您的地图应用程序设置身份验证。
2. 打开身份池页面。

> **Note**
>
> 您创建的资源池必须与您在上一节中创建的 Amazon Location Service 资源位于相同的 AWS 账户和 AWS 区域。

3. 选择创建身份池。
4. 从配置身份池信任步骤开始。要进行用户访问身份验证，请选择访客访问权限，然后按下一步。
5. 在配置权限页面上，选择使用现有 IAM 角色并输入您在上一步中创建的 IAM 角色的名称。准备就绪后，按下一步继续下一步。

6. 在配置属性页面上，为您的身份池提供一个名称。然后按 “下一步”。
7. 在 “查看并创建” 页面上，查看所有存在的信息，然后按创建身份池。
8. 打开身份池页面，然后选择您刚刚创建的身份池。然后复制或写下 IdentityPoolId 稍后将在浏览器脚本中使用的内容。

创建基本的安卓应用程序

在本教程中，您将创建一个 Android 应用程序，该应用程序可以嵌入地图并允许用户在地图上找到某个位置的内容。

首先，使用 Android Studio 的新项目向导创建一个空的 Kotlin 应用程序。

创建空应用程序 (AndroidStudio)

1. 开始 AndroidStudio。打开菜单，然后选择 “文件”、“新建”、“新建项目”。
2. 从手机和平板电脑选项卡中，选择空活动，然后选择下一步。
3. 为您的应用程序选择名称、程序包名称和保存位置。
4. 在语言的下拉列表中，选择 Kotlin。
5. 选择完成以创建您的空白应用程序。

向应用程序添加交互式地图

现在，您已经创建了一个基本应用程序，可以向应用程序中添加地图控件。本教程使用 API 密钥来管理地图视图。地图控件本身是MapLibre 原生库的一部分，具有 API 密钥和 MapLibre，地图数据来自亚马逊位置。

要向应用程序添加地图，必须执行以下操作：

- 将 MapLibre 依赖项添加到您的项目中。
- 使用 compose 设置地图视图代码。
- 编写代码来显示地图。

使用以下步骤将地图添加到您的应用程序：

1. 将 MapLibre 依赖项添加到您的项目中

a. 在中 AndroidStudio，选择 “查看” 菜单，然后选择 “工具”、“窗口”、“项目”。这将打开项目窗口，通过该窗口，您可以访问项目中的所有文件。

b. 在 “项目” 窗口中，打开 gradle，然后在树视图中打开libs.versions.toml文件。这将会打开 libs.versions.toml 文件以进行编辑。现在在libs.versions.toml文件中添加以下版本和库数据。

```
[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. 编辑完libs.versions.toml文件后， AndroidStudio 必须重新同步项目。在libs.versions.toml编辑窗口的顶部， AndroidStudio 提示您进行同步。选择 “立即同步” 以同步您的项目，然后再继续。

d. 在 “项目” 窗口中，在树视图中打开 Gradle 脚本，然后为您的应用程序模块选择build.gradle文件。这将会打开 build.gradle 文件以进行编辑。

e. 在文件底部的依赖关系部分中，添加以下依赖关系。

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

f. 添加 Gradle 依赖项后，Android Studio 必须重新同步项目。在 build.gradle 编辑窗口的顶部，Android Studio，选择立即同步以同步您的项目，然后再继续。

2. 现在，您将使用 compose 设置地图视图代码。使用以下步骤：

a. 在 “项目” 窗口中，在树视图中打开 App、Java、#######，然后转到 ui 文件夹，在 ui 文件夹中创建一个视图目录。

b. 在视图目录中创建一个MapLoadScreen.kt文件。

c. 将以下代码添加到您的 MapLoadScreen.kt文件。

```
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. 编写代码来显示地图。

a. 将以下代码添加到您的 MainActivity.kt文件。

```
// ...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. 保存 MainActivity.kt 文件。您现在可以构建应用程序了。要运行它，你可能需要设置一台设备来模拟它 AndroidStudio，或者在你的设备上使用该应用程序。使用此应用程序查看地图控件的行为方式。您可以通过在地图上拖动它并捏住它进行缩放来进行平移。

在下一节中，您将在地图上添加标记，并在移动地图时显示标记所在位置的地址。

向您的应用程序添加反向地理编码搜索

现在，您将在应用程序中添加反向地理编码搜索，您可以在其中找到某个位置的项目。为了简化Android应用程序的使用，我们将搜索屏幕中央。要查找新位置，请将地图移动到要搜索的位置。我们将在地图中心放置一个标记，以显示我们正在搜索的地方。

添加反向地理编码搜索将包括两部分。

- 在屏幕中央添加一个标记，向用户显示我们在哪里搜索。
- 为结果添加文本框，然后搜索标记所在位置的内容并将其显示在文本框中。

向应用程序添加标记

1. 将此图像保存到项目中的app/res/drawable文件夹中red_marker.png (您也可以从中访问该图像GitHub。或者，您可以创建自己的图像。对于不想显示的部分，也可以使用具有透明度的.png 文件。
2. 将以下代码添加到您的 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. 构建并运行您的应用程序以预览功能。

现在，您的应用程序屏幕上有一个标记。在本例中，它是一个静态图像，不会移动。它用于显示地图视图的中心，这是我们要搜索的地方。在以下步骤中，我们将在该位置添加搜索。

在应用程序中添加在某个位置的反向地理编码搜索

1. 在 “项目” 窗口中，在树视图中打开 gradle 到`libs.versions.toml`文件。这将会打开 `libs.versions.toml` 文件以进行编辑。现在在`libs.versions.toml`文件中添加以下版本和库数据。

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

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

 [plugins]
 ...
```

2. 编辑完libs.versions.toml文件后， AndroidStudio 必须重新同步项目。在libs.versions.toml编辑窗口的顶部， AndroidStudio 提示您进行同步。选择 “立即同步” 以同步您的项目，然后再继续。
3. 在 “项目” 窗口中，在树视图中打开 Gradle 脚本，然后为您的应用程序模块选择build.gradle文件。这将会打开 build.gradle 文件以进行编辑。
4. 在文件底部的依赖关系部分中，添加以下依赖关系。

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

5. 编辑 Gradle 依赖项后， AndroidStudio 必须重新同步项目。在build.gradle编辑窗口的顶部， AndroidStudio 提示您进行同步。选择同步SyncNow您的项目，然后再继续。
6. 现在在树视图中将数据添加到请求目录中，然后创建ReverseGeocodeRequest.kt数据类。将以下代码添加到类中。

```
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. 现在在树视图中将数据添加到响应目录，然后创建ReverseGeocodeResponse.kt数据类。在其中添加以下代码。

```
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. 现在，从 “项目” 窗口，在树视图中打开 App、Java、*******，然后转到 ui 文件夹，在 ui 文件夹中创建 ViewModel 目录。
9. 在 ViewModel 目录中创建MainViewModel.kt文件。
10. 将以下代码添加到您的 MainViewModel.kt文件。

```
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. 如果文件尚未打开，请按照前面的步骤打开 MapLoadScreen.kt 文件。添加以下代码。这将创建一个撰写文本视图，您将在其中看到该位置的反向地理编码搜索结果。

```
// ...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. 在应用程序中，在 java 下的软件包名称文件夹中 AndroidStudio，打开该`MainActivity.kt`文件。如图所示修改代码。

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

此代码适用于地图视图。虚拟摄像机位置定义了地图视图 MapLibre。移动地图可以看作是移动虚拟摄像机。

- ViewModel 有一个标签变量：此变量在撰写文本视图中设置数据。

- onMapReady:此函数已更新为注册两个新事件。
- 每当用户移动地图时，就会发生该onCameraMove事件。通常，在移动地图时，我们希望隐藏搜索内容，直到用户完成地图的移动。
- 当用户暂停移动地图时，就会发生该onCameraIdle事件。此事件调用我们的反向地理编码函数以在地图中心进行搜索。
- `reverseGeocode(latLng: LatLng, apiKey: String)`: 此函数在事件中调用onCameraIdle，它在地图中心搜索位置并更新标签以显示结果。它使用摄像机目标，它定义了地图的中心（摄像机正在看的地方）。

13. 保存您的文件，然后构建并运行您的应用程序，看看它是否有效。

具有搜索功能的快速入门应用程序已完成。

向您的应用程序添加追踪功能

要向示例应用程序添加追踪功能，请按照以下步骤操作：

1. 向您的项目添加跟踪和身份验证 SDK 依赖项。
2. 在 AndroidManifest .xml 文件中包含权限和服务条目。
3. 使用 compose 设置开始/停止跟踪按钮代码。
4. 添加用于创建 LocationTracker 对象并开始和停止跟踪的代码。
5. 使用安卓模拟器创建测试路线。

1. 向您的项目添加跟踪和身份验证 SDK 依赖项。

   a. 在“项目”窗口中，打开 gradle，然后在树视图中打开`libs.versions.toml`文件。这将会打开 `libs.versions.toml` 文件以进行编辑。现在在`libs.versions.toml`文件中添加以下版本和库数据。

```
[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. 编辑完libs.versions.toml文件后， AndroidStudio 必须重新同步项目。在libs.versions.toml编辑窗口的顶部， AndroidStudio 提示您进行同步。选择 “立即同步” 以同步您的项目，然后再继续。

c. 在 “项目” 窗口中，在树视图中打开 Gradle 脚本，然后为您的应用程序模块选择build.gradle文件。这将会打开 build.gradle 文件以进行编辑。

d. 在文件底部的依赖关系部分中，添加以下依赖关系。

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

e. 编辑 Gradle 依赖项后， AndroidStudio 必须重新同步项目。在 build.gradle 编辑窗口的顶部， AndroidStudio 会提示你进行同步。选择同步SyncNow您的项目，然后再继续。

2. 在 AndroidManifest .xml 文件中包含权限和服务条目。

- 要在中包含正确的权限和服务条目AndroidManifest.xml file，请使用以下代码更新文件：

```
<?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 compose 设置开始/停止跟踪按钮代码。

   a. 在名为 ic_pause 和 ic_play 的可绘制对象下添加两张以 res 形式播放和暂停的图像。您也可以从中访问图像GitHub。

   b. 如果文件尚未打开，请按照前面的步骤打开 `MapLoadScreen.kt` 文件。添加以下代码。这将创建一个 compose Button 视图，我们可以在其中单击它来开始和停止跟踪。

```
// ...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. 添加用于创建 LocationTracker 对象并开始和停止跟踪的代码。

   a. 在MainViewModel.kt文件中添加以下代码。

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

   b. 将以下代码添加到您的 MainActivity.kt文件。

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

上面的代码显示了如何使用创建`LocationTracker`对象`AuthHelper`以及如何开始和停止跟踪 LocationTracker。

- `authenticateUser()`: 此方法创建 AuthHelper 和 LocationTracker 对象。
- `onStartStopTrackingClick`：当用户点击开始/停止跟踪按钮时触发此回调，该按钮将使用跟踪 SDK 开始/停止跟踪。

5. 使用安卓模拟器创建测试路线。

a. 使用 Android Studio 启动 AVD，打开模拟器。
b. 点击模拟器工具栏中的更多（三个点）图标@@ 打开扩展控件。
c. 从侧栏中选择 “位置”，打开 “位置”。
d. 使用 GPX 数据或单击地图并选择源和目的地数据来@@ 创建路线。
e. 单击 PLA Y ROUTE 开始模拟，开始模拟 GPS 路线。
f. 通过运行您的@@ 应用程序并观察它如何处理模拟路径来测试应用程序。

这是 Android 快速入门应用程序的完整演示。

接下来做什么

此应用程序的源代码可在上找到GitHub。

要了解更多关于 Amazon Location 的信息，您可以查看以下资源：

- 深入了解 Amazon Location Service 的概念
- 获取有关如何使用 Amazon Location 功能的更多信息
- 查看使用 Amazon Location 的代码示例，了解如何扩展此示例并构建更复杂的应用程序

创建 iOS 应用程序

在本节中，您将创建一个能够在前台进行位置搜索和跟踪的 iOS 应用程序。首先，您将创建您的亚马逊位置资源，并为您的应用程序创建一个 Amazon Cognito 身份。

主题

为您的应用程序创建 Amazon Location 资源

如果您还没有，则必须创建应用程序将使用的 Amazon Location 资源。您将创建一个用于在应用程序中显示地图的地图资源、一个用于在地图上搜索位置的地点索引以及一个用于在地图上跟踪对象的跟踪器。

向您的应用程序添加位置资源

1. 选择要使用的地图样式。

   a. 在 Amazon Location 控制台的地图页面上，选择创建地图以预览地图样式。

   b. 为新地图资源添加名称和描述。记下您用于地图资源的名称。在本教程的后面部分创建脚本文件时，您将需要它。

   c. 选择一张地图。

   > **Note**
   >
   > 选择地图样式还会选择要使用的地图数据提供程序。如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则只能使用 HERE 作为地理位置提供程序。想要了解更多信息，请参阅 AWS 服务条款的第 82 节。

   d. 同意 Amazon Location 条款和条件，然后选择创建地图。您可以与所选地图进行交互：放大、缩小或向任意方向平移。

   e. 请记下新地图资源显示的 Amazon 资源名称 (ARN)。在本教程的后面部分，您将使用它来创建正确的身份验证。

2. 选择要使用的地点索引。

   a. 在 Amazon Location 控制台的地点索引页面上，选择创建地点索引。

   b. 为新的地点索引资源添加名称和描述。记下地点索引资源使用的名称。在本教程的后面部分创建脚本文件时，您将需要它。

   c. 选择数据提供程序。

   > **Note**
   >
   > 在大多数情况下，请选择与您已选择的地图提供程序相匹配的数据提供程序。这有助于确保搜索结果与地图相匹配。

如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则只能使用 HERE 作为地理位置提供程序。想要了解更多信息，请参阅 AWS 服务条款的第 82 节。

d. 选择数据存储选项。在本教程中，不存储结果，因此您可以选择否，仅限一次性使用。

e. 同意 Amazon Location 条款和条件，然后选择创建地点索引。

f. 记下为您的新地点索引资源显示的 ARN。在本教程的下一部分，您将使用它来创建正确的身份验证。

3. 使用 Amazon Location 控制台创建追踪器。

a. 打开 Amazon Location Service 控制台。

b. 在左侧导航窗格中，选择跟踪器。

c. 选择创建跟踪器。

d. 填写所有必填字段。

e. 在 “位置筛选” 下，我们建议您使用默认设置：TimeBased。

f. 选择 “创建跟踪链接” 以完成操作。

为应用程序设置身份验证

您在本教程中创建的应用程序是匿名使用的，这意味着您的用户无需登录 AWS 即可使用该应用程序。但是，Amazon Location Service API 需要身份验证才能使用。您将使用 Amazon Cognito 为匿名用户提供身份验证和授权。本教程将使用 Amazon Cognito 对您的应用程序进行身份验证。

> **Note**
>
> 有关将 Amazon Cognito 与 Amazon Location Service 配合使用的更多信息，请参阅。授予对 Amazon Location Service 的访问权限

以下教程向您展示如何为在中创建的地图、地点索引和追踪器设置身份验证，以及如何为 Amazon Location 设置权限。

创建用于跟踪的 IAM 策略

1. 使用您具有管理员权限的账户，通过以下网址登录到 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 在导航窗格中，选择策略。
3. 在内容窗格中，选择创建策略。
4. 选择 JSON 选项，然后将此 JSON 策略复制并粘贴到 JSON 文本框中。

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

这是跟踪的政策示例。要将示例用于您自己的策略，请替换`Region`、`AccountIndexName`、`MapName`和TrackerName占位符。

> **Note**
>
> 虽然未经身份验证的身份池旨在在不安全的互联网站点上公开，但请注意，它们将被交换为标准的、有时间限制 AWS 的凭证。
> 适当确定与未经身份验证的身份池关联的 IAM 角色的范围很重要。有关在亚马逊定位服务的 Amazon Cognito 中使用策略和适当确定政策范围的更多信息，请参阅授予亚马逊定位服务访问权限。

5. 在“查看并创建”页面上，为策略名称字段提供一个名称。查看您的策略授予的权限，然后选择创建策略以保存您所做的工作。

将在托管策略列表中显示新策略，并已准备好附加该策略。

为追踪设置身份验证

1. 在 Amazon Cognito 控制台中为您的地图应用程序设置身份验证。
2. 打开身份池页面。

> **Note**
>
> 您创建的资源池必须与您在上一节中创建的 Amazon Location Service 资源位于相同的 AWS 账户和 AWS 区域。

3. 选择创建身份池。
4. 从配置身份池信任步骤开始。要进行用户访问身份验证，请选择访客访问权限，然后按下一步。
5. 在配置权限页面上，选择使用现有 IAM 角色并输入您在上一步中创建的 IAM 角色的名称。准备就绪后，按下一步继续下一步。
6. 在配置属性页面上，为您的身份池提供一个名称。然后按 “下一步”。
7. 在 “查看并创建” 页面上，查看所有存在的信息，然后按创建身份池。
8. 打开身份池页面，然后选择您刚刚创建的身份池。然后复制或写下 IdentityPoolId 稍后将在浏览器脚本中使用的内容。

创建基本 iOS 应用程序

在本教程中，您将创建一个嵌入地图的 iOS 应用程序，并允许用户查找地图上某个位置的内容。

首先，让我们使用 Xcode 的项目向导创建一个 Swift 应用程序。

创建空应用程序 (Xcode)

1. 打开 Xcode，然后从菜单中选择 “文件”、“新建”、“新建项目”。
2. 在 iOS 选项卡中，选择 “应用程序”，然后选择 “下一步”。
3. 提供产品名称、组织标识符，然后在 “接口” 字段中输入`SwiftUI`。选择 “下一步” 以完成选择。
4. 选择要保存项目的位置，然后按创建按钮创建空应用程序。

创建基础应用程序后，您需要为示例应用程序安装所需的软件包。

安装所需的依赖项

1. 在 Xcode 中，右键单击项目并选择 “添加包...” 。这将打开 “包” 窗口，您可以在其中向项目中添加包。
2. 在 “包” 窗口中，添加以下软件包：
   - 要获取 Maplibre 原生软件包，请使用以下网址：https://github.com/maplibre/。maplibre-gl-native-distribution从 URL 中添加以下软件包：`maplibre-gl-native-distribution`、和`Mapbox`。
   - 要获取亚马逊位置身份验证 iOS SDK，请使用以下网址：https://github.com/aws-geospatial/amazon-location-mobile-auth-sdk- ios。从 URL 中添加以下软件包：`amazon-location-mobile-auth-sdk-ios`、和`AmazonLocationiOSAuthSDK`。
   - 要获取亚马逊位置追踪 iOS SDK，请使用以下网址：https://github.com/aws-geospatial/amazon-location-mobile-tracking-sdk- ios。从 URL 中添加以下软件包：`amazon-location-mobile-tracking-sdk-ios`、和`AmazonLocationiOSTrackingSDK`。

设置初始代码

在您的应用程序中启用位置权限

1. 打开你的 Xcode 项目。
2. 找到项目的`Info.plist`文件。
3. 根据您的应用程序要求添加必要的位置权限密钥。以下是钥匙：
   - `NSLocationWhenInUseUsageDescription`：描述为什么您的应用程序在使用时需要位置访问权限。
   - `NSLocationAlwaysAndWhenInUseUsageDescription`：描述为什么您的应用需要持续的位置访问权限。

现在，您需要在应用程序中配置资源值。添加一个名为的新文件`Config.xcconfig`并填写您之前在 Amazon 控制台中创建的值。

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

```
TRACKER_NAME =
```

1. 从左侧的导航器部分中，选择项目。
2. 在 “目标” 部分下，选择您的应用程序，然后单击 “信息” 选项卡。
3. 添加信息属性，其值如下所示：
4. 添加包含以下内容的Config.swift文件，该文件将从 Bundle 信息文件中读取配置值。

```
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. 使用名称创建一个新文件夹，ViewModel并在其中添加一个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
    }
```

```
}
```

向应用程序添加交互式地图

现在，您将向应用程序中添加地图控件。本教程使用 MapLibre 和 AWS API 来管理应用程序中的地图视图。地图控件本身是 MapLibre GL Native iOS 库的一部分。

1. 使用以下代码在 V ie ws 文件夹下添加`MapView.swift`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ViewModel文件夹下添加AWSSignatureV4Delegate文件。此文件用于对所有 MapView http 请求进行签名以渲染地图：

```
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 “视图” 文件夹下添加UserLocationView.swift文件。这会添加一个按钮，该按钮可将地图居中放置在用户所在位置

```
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. 使用以下代码添加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)
            }
        }
    }
}
```

您现在可以构建应用程序了。要运行它，你可能需要设置一台设备才能在 Xcode 中模拟它，或者在你的设备上使用该应用程序。使用此应用程序查看地图控件的行为方式。您可以通过在地图上拖动来进行平移，然后捏住即可缩放。您可以自行更改地图控件的工作方式，以根据应用程序的需要对其进行自定义。

将搜索功能添加到应用程序

现在，您将向应用程序添加反向地理编码搜索，您可以在其中找到某个位置的项目。为了简化 iOS 应用程序的使用，我们将搜索屏幕中央。要查找新位置，请将地图移动到要搜索的位置。我们将在地图的中心放置一个标记，以显示我们正在搜索的位置。

1. 在 `TrackingViewModel.swift` 文件中添加以下与反向地理编码搜索相关的代码

```
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. 使用以下代码更新TrackingView.swift文件，该代码将显示地图视图中心位置的地址

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

向您的应用程序添加追踪功能

应用程序的最后一步是向应用程序添加跟踪功能。在这种情况下，您将在应用程序上添加开始跟踪、停止跟踪、获取和显示跟踪点。

1. 将TrackingBottomView.swift文件添加到您的项目中。它有一个用于开始和停止跟踪用户位置的按钮，并在地图上显示跟踪点。

```
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. 使用以下代码更新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文件中添加以下代码。这些函数负责启动和停止跟踪。如果用户定位权限被拒绝，它还会显示错误警报。

4. 要实现前台跟踪，请复制粘贴以下代码示例：

```
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`将要求用户获得定位许可。应用程序必须使用 “使用时” 或 “仅使用一次” 权限。否则，应用程序将抛出权限被拒绝的错误。

要获取和显示追踪位置，请按照以下步骤操作：

1. 要从用户的设备上获取位置，您需要提供开始和结束日期和时间。单个调用最多返回 100 个跟踪位置，但如果追踪位置超过 100 个，它将返回 `nextToken` 值。你需要用 `nextToken getTrackerDevice `调用后续的 “Location” 调用，以便在给定的开始和结束时间加载更多的跟踪点。

```
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. 现在用以下代码替换MapView.swift文件中的代码：

```
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]?)
}
```

要本地化字符串值，请按以下步骤操作。

1. 创建并添加一个名为的新文件Localizable.xcstrings。
2. 右键单击该Localizable.xcstrings文件并将其作为源代码打开。
3. 将其内容替换为以下内容：

```
{
  "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. 保存您的文件，然后构建并运行您的应用程序以预览功能。
5. 允许位置权限，然后点击跟踪按钮。该应用程序将开始上传用户位置并将其上传到亚马逊位置追踪器。它还将在地图上显示用户位置的变化、追踪点和当前地址。

您的快速入门应用程序已完成。本教程向您展示了如何创建一个 iOS 应用程序，该应用程序：

- 创建用户可以与之交互的地图。
- 处理与用户更改地图视图相关的多个地图事件。
- 调用 Amazon Location Service API，专门用于使用亚马逊定位的 searchByPosition API 在某个位置搜索地图。

接下来做什么

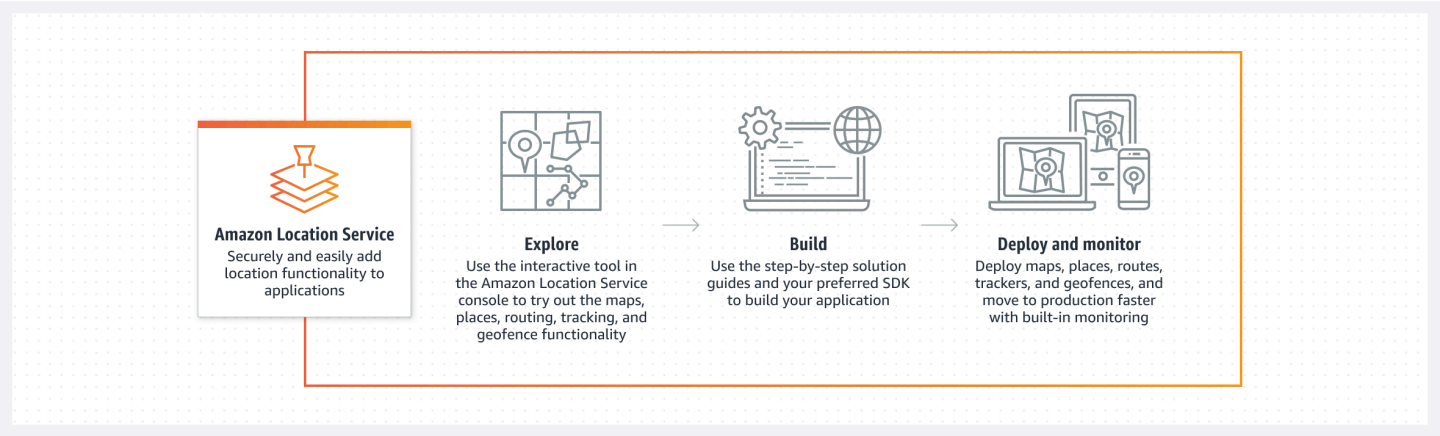
此应用程序的源代码可在上找到GitHub。

要了解更多关于 Amazon Location 的信息，您可以查看以下资源：

- 深入了解 Amazon Location Service 的概念
- 获取有关如何使用 Amazon Location 功能的更多信息
- 查看使用 Amazon Location 的代码示例，了解如何扩展此示例并构建更复杂的应用程序

Amazon Location Service 概念

借助 Amazon Location Service，您可以安全地将位置数据添加到您的应用程序中。使用 Amazon Location 控制台上提供的可视化交互式工具探索其中的一些功能。使用浏览工具，您可以操作默认地图、搜索兴趣点、在感兴趣区域周围绘制地理围栏，以及模拟将设备位置发送到跟踪器。



准备好建造时，请创建您的资源并从各种地图样式和数据提供程序中进行选择。然后，您可以安装与您的开发环境SDK相匹配的，并按照本指南中的说明使用 Amazon APIs Location。此外，您可以使用 Amazon CloudWatch 和，集成监控功能 AWS CloudTrail。

本部分中的主题概述了 Amazon Location 的核心概念，并让您为开始在自己的应用程序中使用位置做好准备。

主题

- Amazon Location 概述
- 映射
- 地点搜索
- 路线
- 地理围栏和跟踪器
- 使用 Amazon Location Service 的常见用例
- 什么是数据提供程序？
- Amazon Location 区域和终端节点
- Amazon Location Service 配额

Amazon Location 概述

Amazon Location Service 允许通过 AWS 资源访问基于位置的功能和数据提供程序。Amazon Location 提供五种类型的 AWS 资源，具体取决于您需要的功能类型。结合使用不同的资源来创建完整的基于位置的应用程序。您可以使用 Amazon Location 控制台、Amazon Location 或SDKs。APIs

每个资源都定义要使用的底层数据提供程序（如果适用），并允许访问与其类型相关的功能。

例如：

- Amazon Location Service Map 允许您从地图提供程序中选择要在移动或网络应用程序上使用的地图。
- Amazon Location Service 地点允许您选择用于搜索兴趣点、完成部分文本、地理编码和反向地理编码的数据提供程序。
- Amazon Location Servic e Routes 允许您选择数据提供商，根据 up-to-date 道路和实时交通信息查找路线并估算出行时间。
- Amazon Location Service 地理围栏允许您将感兴趣的区域定义为虚拟边界。然后，您可以对照这些位置进行评估，并获得有关进入和退出事件的通知。
- Amazon Location Service 跟踪器会从您的设备接收位置更新。您可以将跟踪器链接到地理围栏集合，以便根据您的地理围栏自动评估所有位置更新。

您可以使用IAM政策来管理和授权访问您的 Amazon Location 资源。您还可以将资源组织成资源组，以便随着资源数量的增长管理和自动执行任务。有关管理 AWS 资源的更多信息，请参阅什么是AWS资源组？ 在 Res AWSource Groups 用户指南中。

位置是通过使用遵循世界大地测量系统 (84) 的纬度和经度坐标来定义的，世界大地测量系统 (WGS84) 通常用作全球定位系统 (GPS) 服务的标准坐标参考系。

下面几部分介绍 Amazon Location 的组成部分的工作原理。

映射

Amazon Location Service 地图资源允许您访问地图的基础底图数据。您可以将地图资源与地图渲染库一起使用，将交互式地图添加到您的应用程序中。您可以根据应用程序的需要向地图添加其他功能，例如标记（或图钉）、路线和多边形区域。

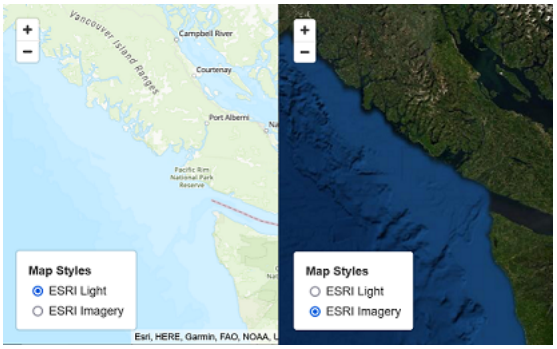
> **Note**
>
> 有关如何在实践中使用地图资源的信息，请参阅 在您的应用程序中使用 Amazon Location 地图。

下面概述了如何创建和使用地图资源：

1. 您可以通过从数据提供商中选择地图样式在 AWS 账户中创建地图资源。
2. 然后，您可以选择并安装与您的开发环境和应用程序相匹配的。SDK有关可用选项的更多信息，请参阅有关访问 Amazon Location 的主题。
3. 要在应用程序中显示地图，请将地图资源与渲染库（例如 Amplify 或 Tangram）组合在一起。MapLibre有关更多信息，请参阅本指南中的使用地图。
4. 然后，您可以使用诸如亚马逊之类的服务 CloudWatch 和 Amazon Locat AWS CloudTrail ion 来集成监控。有关更多信息，请参阅、使用亚马逊监控 Amazon Location Service CloudWatch和使用 AWS CloudTrail 进行日志记录和监控。

地图样式

创建地图资源时，必须为该资源选择地图样式。地图样式定义了渲染地图的外观。例如，下图显示了来自 Amazon Location 中不同地图资源的同一个数据提供程序，具有两种不同的样式。一种风格是基于地图中的矢量数据的典型道路风格。另一个包括显示卫星影像的栅格数据。当你放大或缩小地图时，样式可能会发生变化，但通常样式的主题是一致的。在将样式信息传递到地图渲染库之前，可以覆盖部分或全部样式信息。



政治观点

Amazon Location Service 中的某些地图样式支持其他政治观点。

> **Note**
>
> 政治观点的使用必须遵守适用的法律，包括那些关于绘制您通过 Amazon Location Service 访问的地图、图像和其他数据和第三方内容的国家或地区的法律。

以下地图样式支持印度 (IND) 的政治观点。

- Esri 地图样式：
  - Esri 导航
  - Esri 浅色
  - Esri 街道图
  - Esri 深灰色画布
  - Esri 浅灰色画布
- 开放数据地图样式：
  - 开放数据标准（浅色）
  - 开放数据标准（深色）
  - 开放数据可视化（浅色）
  - 开放数据可视化（深色）

在 Amazon Location Service 控制台中，你可以筛选显示的样式，只显示支持印度政治观点的风格。

自定义层

自定义层是可以为地图风格启用的附加层。目前只有 VectorEsriNavigation 地图样式支持POI自定义图层。

启用 POI 自定义层后，即会在地图中添加更多的地点，例如商店、服务设施、餐厅、景点和其他兴趣点。默认情况下，自定义层为 unset。有关更多信息，请参阅位置API参考MapConfiguration中的。

地图渲染

要在应用程序中渲染地图，通常需要使用地图渲染库。库有几个常用的选项可供使用：

- MapLibre— MapLibre 是一个专门用于渲染交互式地图的开源库，也是通过 Amazon Location Service 渲染地图的首选方法。 MapLibre 包括渲染来自数据源（例如 Amazon Location 地图资源）的栅格和矢量数据的功能。您可以扩展 MapLibre 以在地图上绘制自己的数据。
- Amplify——Amplify 是一个开源框架，用于为网络、iOS、安卓等平台构建应用程序。如果您的应用程序使用 Amplify，则可以将其扩展，包含 Amazon Location 功能。Amplify 包含专门用于创建基于 Amazon Location 应用程序（包括渲染地图）的库。Amplify 用于 MapLibre 渲染地图，但提供了亚马逊定位服务特有的额外功能，以提高其使用效率，还增加了搜索和其他功能。
- 七巧板 —七巧板是一个替代的开源库，可以渲染交互式地图，类似于。 MapLibre

地图渲染库在运行时从 Amazon Location Service 提取数据，根据您选择的地图资源渲染地图数据。地图资源定义了将要使用的数据提供程序和地图样式。

下图显示了如何在 Amazon Location Service 中使用地图资源以及地图渲染库来创建最终地图。

1. 您可以使用 AWS Management Console 或在 Amazon Location Service 中创建地图资源 AWS CLI。这定义了您要使用的数据提供程序和地图样式。
2. 您的应用程序包括地图渲染库。您可以为地图渲染库指定要使用的地图资源的名称。地图渲染库从 Amazon Location 中提取该地图资源的数据和样式信息，并在屏幕上渲染地图。

地图术语

地图资源

允许您访问选中的供应商的地图数据。使用地图资源获取包含地图数据和样式描述符的地图图块，以指定特征在地图上的渲染方式。

底图

为您的地图提供地理环境，该地图存储为矢量切片图层。切片图层包括地理环境，例如街道名称、建筑物和土地用途，以供视觉参考。

Vector

矢量数据是由点、线和面组成的形状数据。它通常用于在地图上存储和显示道路、位置和区域。矢量形状也可以用作地图上标记的图标。

栅格

栅格数据是图像数据，由网格组成，通常由颜色组成。它通常用于在地图上存储和显示连续数据的表示形式，例如地形、卫星影像或热点图。光栅图像也可以用作图像或图标。

地图样式

矢量数据本质上并不包含有关如何绘制数据层以创建最终地图的信息。地图样式定义数据的颜色和其他样式信息，以定义其在渲染时的外观。地图资源包括地图的样式信息。

Amazon Location Service 提供的样式符合 Mapbox GL 样式规范。

矢量切片

一种使用矢量形状存储地图数据的切片格式。这些数据生成的地图可以根据显示分辨率进行调整，并以多种方式有选择地渲染要素，同时保持较小的文件大小以获得最佳性能。

支持的矢量文件格式：Mapbox 矢量图块 (MVT)。

字形文件

包含已编码的 Unicode 字符的二进制文件。由地图渲染器用来显示标签。

Sprite 文件

一种便携式网络图形 (PNG) 图像文件，其中包含小型光栅图像，JSON文件中包含位置描述。由地图渲染器用于在地图上渲染图标或纹理。

地点搜索

Amazon Location Service 的一项关键功能是能够搜索地理位置信息。Amazon Location 通过地点索引资源提供此功能。

> **Note**
>
> 有关如何在实践中使用地点索引资源进行搜索的信息，请参阅 使用 Amazon Location 搜索地点和地理位置数据。

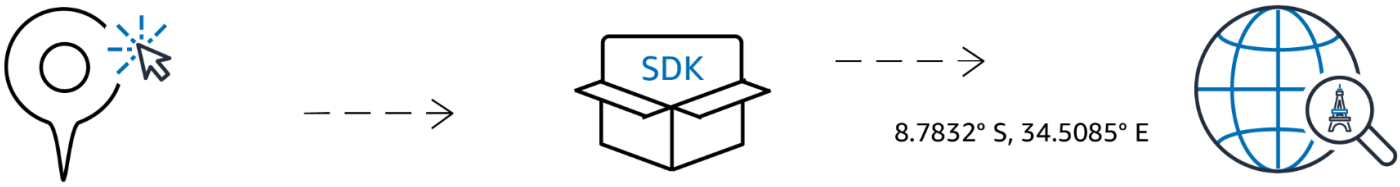
您可以使用地点索引APIs来搜索：

- 兴趣点，例如餐厅和地标。按名称和可选位置进行搜索，并获得按相关性排序的选项列表。
- 街道地址，接收该地址的纬度和经度。这称为地理编码。
- 纬度和经度位置，接收相关的街道地址或其他有关该位置的信息。这称为反向地理编码。
- 部分或拼写错误的自由格式文本查询，通常是在用户键入时进行的。这称为自动完成、自动建议或模糊匹配。

地点索引包括要使用哪个数据提供程序进行搜索。

> **Note**
>
> 地图数据和其他地理位置信息（包括确切位置）可能因数据提供程序而异。最佳做法是，为您的地点索引、地图和其他 Amazon Location 资源使用相同的数据提供程序。例如，如果您的地点索引返回的地点与地图资源提供的相同地点的位置不匹配，则可以在地图上看似错误的位置放置标记。

下面显示如何创建和使用地点索引资源：



1. 首先，您可以通过选择数据提供商在 AWS 账户中创建地点索引资源。
2. 然后，您可以选择并安装与您的开发环境和应用程序相匹配的。SDK有关可用选项的更多信息，请参阅有关访问 Amazon Location 的主题。
3. 开始使用 Amazon 定位地点APIs。有关更多信息，请参阅有关使用地点搜索的主题。
4. 然后，您可以使用诸如Amazon CloudWatch 和之类的服务来集成监控 AWS CloudTrail。有关更多信息，请参阅、the section called “使用监控 CloudWatch”和the section called “ CloudTrail 与 Amazon 位置一起使用”。

地理编码概念

Amazon Location 地点索引提供了一个名为 `SearchPlaceIndexForText` 的操作，允许您指定要搜索的文本。例如，您可以搜索：

- 地点——搜索 **`Paris`** 可能会返回该城市在法国的位置。
- 企业——搜索 **`coffee shop`** 可能会返回咖啡店的列表，包括它们的名称和地点。您还可以指定要搜索的位置或要在其中搜索的边界框，以使结果更具相关性。在这种情况下，提供华盛顿州西雅图市中心的一个位置，将返回该区域的咖啡店。
- 地址——搜索 **`1600 Pennsylvania Ave, Washington D.C.`** 可能会返回美国白宫的位置（位于该地址）。

以这种方式搜索文本通常称为地理编码，包括为地址或地点查找地理位置。

Amazon Location Service 还提供名为 `SearchPlaceIndexForPosition` 的反向地理编码操作。这会获取一个地理位置并返回该位置的地址、公司或其他有关该位置的信息。

搜索结果

当您在 Amazon Location Service 中成功提出搜索请求时，系统会返回一个或多个结果。每个结果都包含一个标签，即结果的名称或描述。例如，搜索 **`coffee shop`** 可能会返回带有标签的结果 `Hometown Cafe`，告诉您找到了一家名为“Hometown Cafe”的咖啡店。搜索结果通常还会包括结构化

地址（包括地址号、单位、街道和邮政编码等属性）。根据数据提供程序的不同，它还将包括其他元数据，例如国家和时区。

要搜索企业名称或类别（例如 **`coffee shop`**），您可能需要在地图上显示所有返回的结果。对于地址搜索，您可能只想自动使用第一个结果。有关相关性的信息，请参阅下一个主题。

多种结果和相关性

通过文字搜索时，Amazon Location Service 通常会找到多个结果。例如，搜索 **`Paris`** 可能会返回法国的城市，但也会返回德克萨斯州的城市。结果按相关性排序，由数据提供程序确定。

> **Note**
>
> 所有提供程序按相关性顺序返回结果。如果您选择 Esri 或 Grab 作为数据提供程序，则结果将包含一个相关性值，您可以使用该值来了解单个请求结果之间的相对相关性。

指定其他信息，例如国家/地区名称或要搜索的地点，可以更改结果顺序，减少结果数量，甚至更改返回的结果集。例如，在德克萨斯州搜索某个地点的 **`Paris`**，`Paris, Texas` 更有可能作为第一个结果返回，而不是 `Paris, France`。

在交互式应用程序中，您可以使用相关性来帮助决定是接受排名靠前的结果，还是要求用户在多个返回的结果之间进行区分。如果第一个结果具有很高的相关性，则可以接受它作为正确答案。如果有多个高相关性结果或没有高相关性结果，则可能需要列出这些结果并让用户选择最佳结果。

地址结果

您可以使用相同的 `SearchPlaceIndexForText` 操作通过 Amazon Location Service 搜索地址。您提供的信息越多，返回的地址与给定地址匹配的可能性就越大。例如，**`123 Main St`** 不太可能找到正确的结果 **`123 Main St, Anytown, California, 90210`**。

地址具有多个属性，例如门牌号、街道、城市、地区和邮政编码等。这些属性用于在地点索引中查找与尽可能多方面匹配的地址。找到的属性越多，匹配项的相关性就越高，返回匹配项的可能性也就越大。

> **Note**
>
> 地址结果的相关性取决于结果与输入的匹配程度。这可能是匹配的属性的数量，也可以是结果与输入的匹配程度。例如，当数据中找到 `Main St` 时，输入 **`123 Main St`** 的相关性会比只有 `Maine St` 作为结果时更高。`Maine St` 仍然会被返回，但可能具有较低的相关性值。

搜索结果包括完整地址 (123 Main St, Anytown, California, 90210) 的标签，还包括返回地址的各个结构化属性。这很有用，因为例如，您可以使用它来填充数据库中的地址字段，或者检查结果并找到找到的位置的城市、地区或邮政编码。

插值

地点索引数据中的地址包括精确的地址匹配项。例如，假设有一条街道 9th street，一个街区有 2 栋房屋，220 和 240，如下图所示。

数据提供程序使用这两个已知地址创建地理位置数据。您可以搜索这两个地址，它们就会被找到。在数据提供程序创建地图数据之后，假设在前两个地址之间添加了一座新房子。这所新房子被赋予了地址 230。如果您搜索 **230 S 9th St**，数据提供程序仍会找到结果。它不会使用已知地址，而是在已知地址之间进行插值，然后根据这些地址估计新地址的位置。在这种情况下，它可以假设 230 介于 220 和 240 之间（并且在街道的同一边），并据此返回一个大致的位置。

> **Note**
>
> 数据提供程序会定期使用新地址更新其地理位置数据。在这种情况下，230 S 9th St 会被添加到数据提供程序数据中，但通常会有一段时间新地址已创建但尚未添加到数据中。

在这种情况下，数据提供程序无法分辨出世界上是否存在新地址，因为它尚未出现在数据中，但可以从其所拥有的信息中提供最佳答案。此结果称为插值，可由数据提供程序在结果中返回。如果

`interpolated` 返回 `false`，则为已知地址。如果返回 `true`，则为近似地址。如果未返回，则数据提供程序不会提供有关结果是否来自插值的信息。

> **Important**
>
> 对于根本不存在的地址，数据提供程序还可能返回插值结果。例如，在本例中，如果您输入 **`232 S 9th St`**，提供程序将找到这个不存在的地址，并返回一个接近 230 但位于 240 一侧的位置。插值地址对于将您带到正确位置很有用，但请记住，它们不是已知地址。

存储地理编码结果

创建地点索引资源时，必须指定数据存储选项（`IntendedUse`在中调用API）。可以将其设置为一次性使用或存储结果。这是在询问您对结果的预期用途。如果要存储结果（即使是出于缓存目的），则必须选择存储选项，而不是一次性使用选项。

> **Note**
>
> 当您选择已存储选项（标记为 “是，结果将存储在控制台中，或者在`storage`中选择”`CreatePlaceIndex`API）时，Amazon Location Service 不会为您存储结果。这表明您计划存储结果。

在考虑如何使用您对 Amazon Location Service 的查询结果时，您应始终了解适用的AWS 服务条款。

地点术语

放置索引资源

允许您选择支持搜索查询的数据源。例如，您可以搜索兴趣点、地址或坐标。当向地点索引资源发送搜索查询时，将使用该资源的配置数据源来完成查询。

地理编码

地理编码是获取文本输入、在地点索引中搜索文本并返回带有位置的结果的过程。

反向地理编码

反向地理编码是从地点索引中获取位置并返回有关该位置的信息的过程，例如该位置的地址、城市或商家。

相关性

相关性是指结果与输入的匹配程度。它不是衡量正确性的标准。

插值

插值法是使用已知地址位置作为指导点来查找未知地址的过程。

ISO3166 个国家/地区代码

Amazon Location Service Places 使用国际标准化组织 (ISO) 3166 个国家/地区代码来指代国家或地区。

要查找特定国家或地区的代码，请使用ISO在线浏览平台。

路线

本部分概览有关使用 Amazon Location Service 进行路由的概述。

> **Note**
>
> 有关如何在实践中使用路线资源的信息，请参阅 使用 Amazon Location Service 计算路线。

路线计算器资源

路径计算器资源允许您根据所选数据提供商提供的 up-to-date 道路网络和实时交通信息查找路线并估算行驶时间。

您可以使用路径APIs来构建功能，允许您的应用程序请求任意两个位置之间路径的行驶时间、距离和几何形状。您还可以使用路径API在单个请求中请求一组出发地和目的地之间的行驶时间和距离，以计算矩阵。

下面显示如何创建和使用路线计算器资源：

1. 首先，您可以通过选择数据提供商在您的 AWS 账户中创建路径计算器资源。
2. 然后，您可以选择并安装与您的开发环境和应用程序相匹配的。SDK
3. 开始使用 Amazon 定位路线APIs。有关如何使用路由的更多信息APIs，请参阅中的主题使用 Amazon Location Service 计算路线。
4. 然后，您可以使用诸如Amazon CloudWatch 和之类的服务来集成监控 AWS CloudTrail。有关更多信息，请参阅、使用亚马逊监控 Amazon Location Service CloudWatch和使用 AWS CloudTrail 进行日志记录和监控。

计算路线

Amazon Location 路线计算器资源提供了一个名为 `CalculateRoute` 的操作，您可以使用该操作在两个地理位置（出发地和目的地）之间创建路线。计算出的路线包括用于在地图上绘制路线的几何图形，以及路线的总时间和距离。

使用路径点

在创建路线请求时，您可以向路线添加其他路径点。这些点位于起点和目的地之间，充当路线上的停靠点。将通过指定的每个路径点计算路线。从请求中的一个点到下一个点的路线称为一个 `Leg`。每段路线包括距离、时间以及该部分路线的几何形状。

> **Note**
>
> 路径点按请求中给出的顺序进行路线。对于最短路径，它们不会重新排序。有关查找最短路径的信息，请参阅 规划路线 部分。

您可以在单个请求中包含最多 25 个路径点来计算路线。

交通和出发时间

Amazon Location Service 在计算路线时会考虑流量。它考虑的流量基于您指定的时间。您可以指定立即出发，也可以提供您想要离开的特定时间，这将通过调整指定时间的交通来影响路线结果。

> **Note**
>
> 例如，您可以使用出发时间和路线响应时间来计算到达时间，以估算例如司机的到达时间。

如果您想让 Amazon Location 不考虑流量，请不要指定出发时间，也不要指定立即出发。这将计算出一条假设路线上交通状况最佳的路线。

出行模式选项

在使用 Amazon Location Service 计算路线时，您可以设置出行模式。默认出行模式为汽车，但您可以交替选择卡车或步行。

如果您指定汽车或卡车模式，则还可以指定其他选项。

对于汽车模式，您可以指定希望避开收费公路或渡轮。这将尽量避开渡轮和收费公路，但如果渡轮和收费公路是到达目的地的唯一途径，仍然会沿着它们行驶。

对于卡车模式，您也可以避开渡轮和收费公路，但此外，您还可以指定卡车的大小和重量，以避开无法容纳卡车的路线。

规划路线

您可以使用 Amazon Location Service 为路线规划和优化软件创建输入。您可以为一组出发位置和一组目的地位置之间的路线创建路线结果，包括行驶时间和行驶距离。这称为创建路线矩阵。

> **Note**
>
> 路线规划和优化软件可以解决许多不同的情况。例如，规划软件可以使用一组时间和点之间的距离来计算在每个点停靠的最短路径，从而为单个驾驶员提供高效的路线。或者，规划软件可用于在多辆卡车之间分配停靠点，从而提高整个车队的效率，或者确保在他们要求的时间范围内拜访每位客户。Amazon Location 以有效的方式提供路由函数，使规划软件能够完成其任务。

例如，给定出发位置 A 和 B 以及目的地位置 X 和 Y，Amazon Location Service 将返回从 A 到 X、A 到 Y、B 到 X 以及 B 到 Y 的路线的行驶时间和行驶距离。

与计算单条路线一样，您可以计算具有不同交通方式、避让点和交通状况的路线。例如，您可以指定车辆是一辆长 35 英尺的卡车，计算的路线将使用这些限制来确定行驶时间和行驶距离。您不能在路线矩阵计算中包含路径点。

返回的结果（和计算的路线）数量等于出发位置数乘以目的地位置的数量。您需要为计算的每条路线付费，而不是为每次服务请求付费，因此，包含 10 个出发地和 10 个目的地的路线矩阵将按照 100 条路线计费。

路线术语

路线计算器资源

一种 AWS 资源，使您能够使用来自所选数据提供商的交通和道路网络数据估算出行时间、距离并在地图上绘制路线。

使用路线计算器资源，您可以计算不同交通方式、绕行路线和交通状况的路线。

路线

路线包含从出发位置、路径点位置和目的地位置沿着路径行驶时使用的详细信息。

路线中的详细信息示例包括：

- 从一个位置到另一个位置的距离。
- 从一个位置移动到下一个位置所花费的时间。
- 表示路径路径的 LineString 几何图形。

有关路线的更多信息，请参阅 Amazon Location Service Routes API 参考中的 CalculateRoute 操作响应语法。

路线矩阵

从一组出发位置到一组目的地位置的路线列表。可用作路线规划或优化软件的输入。

有关计算路线矩阵的更多信息，请参阅 Amazon Location Service Routes API 参考中的 CalculateRouteMatrix 操作语法。

`LineString` 几何图形

Amazon Location 路线由一条或多条路线组成（在整个路线中从一个路径点到另一个路径点的路线）。每段路线的几何形状是一条折线，表示为 `LineString`。`LineString` 是一个有序的位置数组，可用于在地图上绘制路线。

以下是带有三个点的 `LineString` 的示例：

```
[
```

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

路径点

路径点是中间位置，充当出发位置和目的地位置之间的路线上的停靠点。路径上的中途停留顺序遵循您在请求中提供路径点位置的顺序。

路段

单个路段是从一个位置到另一个位置的旅程。如果位置不在道路上，则会将其移至最近的道路。路径中的路段数比位置总数少一个。

没有路径点的路线由一条路段组成，从出发位置到目的地。具有 1 个路径点的路线由 2 个路段组成，分别从出发位置到路径点，然后从路径点到目的地。

步骤

步骤是路段的一个子部分 每个步骤为该路段中的步骤提供摘要信息。

地理围栏和跟踪器

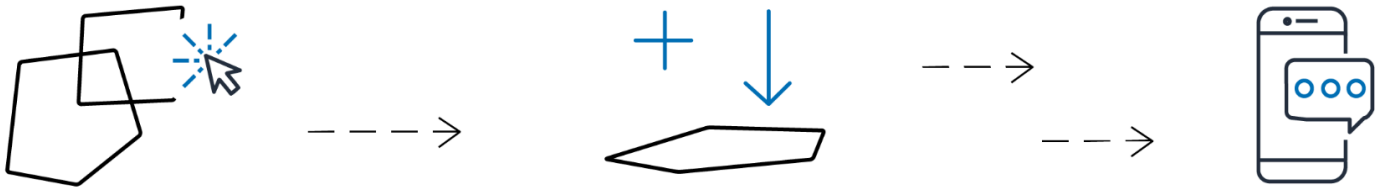
本部分概述了使用 Amazon Location Service 地理围栏和跟踪器的概念。地理围栏是多边形边界，当设备或位置移入和移出区域时，您可以使用它来收到通知。跟踪器资源用于存储和更新设备移动时的位置。

> **Note**
>
> 有关如何在实践中使用地理围栏和跟踪器的信息，请参阅 使用 Amazon Location 对感兴趣的区域进行地理围栏。

地理围栏

地理围栏集合资源允许您存储和管理地理围栏（地图上的虚拟边界）。您可以根据地理围栏集合资源评估位置，并在位置更新越过地理围栏集合中任何地理围栏的边界时收到通知。

下面显示如何创建和使用地理围栏集合资源：



1. 在您的 AWS 账户中创建地理围栏收集资源。
2. 将地理围栏添加到该集合中。您可以通过使用亚马逊定位控制台上的地理围栏上传工具或使用亚马逊定位地理围栏来实现。API有关可用选项的更多信息，请参阅访问 Amazon Location。

   地理围栏可以由多边形或圆形定义。使用多边形查找设备何时进入特定区域。使用圆圈查找设备何时到达距离某一点的特定距离（半径）以内。
3. 您可以开始根据所有地理围栏评估位置。当位置更新跨越一个或多个地理围栏的边界时，您的地理围栏收集资源会在亚马逊上发出以下地理围栏事件类型之一： EventBridge
   - ENTER— 每个地理围栏都会生成一个事件，其中位置更新通过进入其边界来越过其边界。
   - EXIT— 为每个地理围栏生成一个事件，其中位置更新通过退出其边界而越过其边界。

   有关更多信息，请参阅 the section called “对事件做出反应 EventBridge”。您还可以使用诸如 Amazon CloudWatch 和之类的服务来集成监控 AWS CloudTrail。有关更多信息，请参阅、the section called “使用监控 CloudWatch”和the section called “ CloudTrail 与 Amazon 位置一起使用”。

例如，如果您正在追踪一支卡车车队，并且希望在卡车进入任何仓库的特定区域时收到通知。您可以为每个仓库周围的区域创建地理围栏。然后，当卡车向您发送更新的位置时，您可以使用 Amazon Location Service 来评估这些位置，并查看卡车是否进入（或退出）了其中一个地理围栏区域。

> **Note**
>
> 按您评估的地理围栏集合数量计费。您的计费不受每个集合中地理围栏数量的影响。由于每个地理围栏集合可能包含多达 50,000 个地理围栏，因此您可能需要尽可能将地理围栏合并成更少的集合，以降低地理围栏评估的成本。生成的事件将包括集合中各个地理围栏的 ID 以及该集合的 ID。

地理围栏事件

您正在监控的位置由名为 DeviceId 的 ID 引用（这些位置被称为设备位置）。您可以将要评估的设备位置列表直接发送到地理围栏集合资源，也可以使用跟踪器。有关跟踪器的更多信息，请参见下一部分。

只有当设备进入或退出地理围栏时，您才会收到事件（通过 Amazon EventBridge），而不是每次位置变化都会收到事件。这意味着您通常会收到事件，并且响应事件的频率要比每次设备位置更新的频率低得多。

> **Note**
>
> 对于特定 DeviceID 的首次位置评估，假设该设备以前不在任何地理围栏中。因此，如果在集合中的地理围栏内，则第一次更新将生成一个 ENTER 事件，如果不是，则不会生成任何事件。

为了计算设备是进入还是退出地理围栏，Amazon Location Service 必须保持该设备的先前位置状态。此位置状态可存储 30 天。在设备未更新 30 天后，新的位置信息更新将被视为首次位置更新。

跟踪器

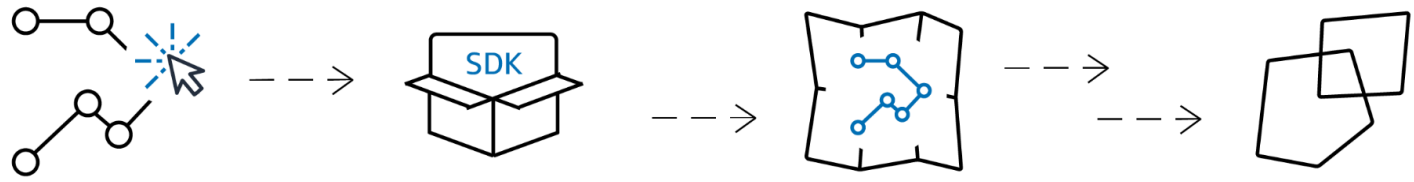
跟踪器存储一组设备的位置更新。跟踪器可用于查询设备的当前位置或位置记录。它存储更新，但通过在存储更新之前筛选位置来减少存储空间和视觉噪音。

存储在您的跟踪器资源中的每个位置更新都可能包括一个位置精度指标，以及最多 3 个与您要存储的位置或设备相关的元数据字段。元数据以键值对的形式存储，可以存储速度、方向、轮胎压力或发动机温度等信息。

> **Note**
>
> Tracker 存储空间会自动使用 AWS 自有密钥进行加密。您可以使用自己管理的KMS密钥添加另一层加密，以确保只有您才能访问您的数据。有关更多信息，请参阅 Amazon Location Service 中的静态数据加密。

跟踪器位置筛选和存储本身很有用，但是与地理围栏配对时，跟踪器尤其有用。您可以将跟踪器链接到您的一个或多个地理围栏集合资源，并根据这些集合中的地理围栏自动评估位置更新。正确使用筛选还可以大大降低地理围栏评估的成本。

下图显示如何创建和使用跟踪器资源：



1. 首先，在 AWS 账户中创建跟踪器资源。
2. 接下来，决定如何向跟踪器资源发送位置更新。用于AWS SDKs将跟踪功能集成到您的移动应用程序中。或者，您可以MQTT按照跟踪中的 step-by-step 说明使用MQTT。
3. 现在，您可以使用跟踪器资源来记录位置记录并在地图上将其可视化。
4. 您还可以将您的跟踪器资源链接到一个或多个地理围栏集合，这样发送到跟踪器资源的每个位置更新都会根据所有链接的地理围栏集合中的所有地理围栏自动进行评估。您可以在 Amazon Location 控制台的追踪器资源详情页面上关联资源，也可以使用亚马逊位置追踪器API来关联资源。
5. 然后，您可以使用诸如Amazon CloudWatch 和之类的服务来集成监控 AWS CloudTrail。有关更多信息，请参阅、the section called “使用监控 CloudWatch”和the section called “ CloudTrail 与 Amazon 位置一起使用”。

使用带有地理围栏的跟踪器

与地理围栏配对时，跟踪器可提供更多功能。您可以通过 Amazon Location 控制台或，将追踪器与地理围栏集合相关联API，以自动评估跟踪器位置。每次跟踪器收到更新的位置时，系统都会根据集合中的每个地理围栏评估该位置，并在 Amazon 中生成相应EXIT的事件ENTER和事件。 EventBridge您还可以对跟踪器应用筛选，根据筛选情况，您可以通过仅评估有意义的位置更新来降低地理围栏评估的成本。

如果您在跟踪器已经收到一些位置更新后将其与地理围栏集合相链接，则链接后的第一个位置更新将被视为地理围栏评估的初始更新。如果它在地理围栏内，您将收到一个 ENTER 事件。如果它不在任何地理围栏内，则无论之前的状态如何，您都不会收到任何 EXIT 事件。

位置筛选

跟踪器可以自动筛选发送给他们的位置。您可能出于多个原因想要筛选掉某些设备位置更新。如果您的系统每分钟左右发送一次报告，则可能需要按时间筛选设备，仅每 30 秒存储和评估一次位置。即使监视频率更高，也可能需要过滤位置更新以清除GPS硬件的噪音。GPS位置位置本质上是嘈杂的。它们的精度并不是 100% 完美，因此即使是静止的设备也看似在轻微移动。在低速下，这种抖动会导致凌乱的视觉效果，如果设备靠近地理围栏的边缘，则可能导致错误的进入和退出事件。

位置筛选的工作原理是跟踪器收到位置更新，从而减少设备路径中的视觉噪音（抖动），减少错误的地理围栏进入和退出事件的数量，并通过减少存储的位置更新和触发的地理围栏评估的数量来帮助管理成本。

跟踪器提供三种位置筛选选项，以帮助管理成本并减少位置更新中的抖动。

- 基于精度——与任何提供精度测量的设备一起使用。大多数GPS和移动设备都提供此信息。每次位置测量的准确性受许多环境因素的影响，包括GPS卫星接收、景观以及无线和蓝牙设备的距离。大多数设备，包括大多数移动设备，都可以提供测量精度的估计值以及测量结果。通过 `AccuracyBased` 筛选，如果设备移动小于测量的精度，Amazon Location 将忽略位置更新。例如，如果一个设备两次连续更新的精度范围分别为 5 米和 10 米，则该设备的移动距离小于 15 米时 Amazon Location 忽略第二次更新。Amazon Location 既不会根据地理围栏评估被忽略的更新，也不会存储这些更新。

  如果未提供精度，则将其视为零，并且测量结果被视为完全准确，并且不会对更新进行筛选。

  > **Note**
  >
  > 您可以使用基于精度的筛选来移除所有筛选。如果您选择基于精度的筛选，但将所有精度数据覆盖为零，或者完全省略准确性，那么 Amazon Location 将不会筛选出任何更新。

  在大多数情况下，基于精度的筛选是筛选位置更新的不错选择，它可以平衡跟踪位置，同时筛选掉不需要的更新，从而降低成本。

- 基于距离——当您的设备不提供精度测量值，但您仍希望利用筛选功能来减少抖动并管理成本时使用。`DistanceBased` 筛选忽略设备移动小于 30 米（98.4 英尺）的位置更新。当您使用 `DistanceBased` 位置筛选时，Amazon Location 既不会根据地理围栏评估这些被忽略的更新，也不会存储更新。

  大多数移动设备的精度，包括 iOS 和 Android 设备的平均精度，都在 15 米以内。在大多数应用程序中，`DistanceBased` 筛选可以减少在地图上显示设备轨迹时位置不准确的影响，以及设备靠近地理围栏边界时连续多次进入和退出事件的反弹效果。它还可以减少对照链接的地理围栏进行评估或检索设备位置的调用，从而帮助降低应用程序的成本。

  如果你想进行筛选，但你的设备不提供精度测量，或者你想筛选掉比基于精度的更新数量更多的更新，那么基于距离的筛选非常有用。

- 基于时间——（默认）当您的设备非常频繁地发送位置更新（每 30 秒钟超过一次），并且您希望在不存储每个更新的情况下实现近乎实时的地理围栏评估时使用。在 `TimeBased` 筛选中，每个位置更

新根据链接的地理围栏集合进行评估，但并非每个位置更新都会存储。如果更新频率超过 30 秒，则每 30 秒仅为每个唯一的设备 ID 存储一次更新。

当您想要存储更少的位置，但希望根据链接的地理围栏集合评估每个位置更新时，基于时间的筛选特别有用。

> **Note**
>
> 在决定筛选方法和位置更新的频率时，请注意追踪应用程序的成本。您需要为每次位置更新付费，并根据每个链接的地理围栏集合评估位置更新一次付费。例如，使用基于时间的筛选时，如果您的跟踪链接到两个地理围栏集合，则每次位置更新都将计为一个位置更新请求和两次地理围栏收集评估。如果您报告设备每 5 秒更新一次位置并使用基于时间的筛选，则每台设备将按每小时 720 次位置更新和 1,440 次地理围栏评估计费。

地理围栏术语

地理围栏集合

包含零或多个地理围栏。它能够在接到请求时发出“进入”和“退出”事件，从而根据其地理围栏评估设备的位置，从而进行地理围栏监控。

地理围栏

在地图上定义虚拟边界的多边形或圆形几何。

多边形几何

Amazon Location 地理围栏是地理区域的虚拟边界，以多边形几何或圆形表示。

圆是围绕一个点而具有一定距离的一组点的集合。如果您想在设备距离某个位置一定距离内时收到通知，请使用圆。

多边形是由一个或多个线性环组成的数组。如果要为设备通知定义特定边界，请使用多边形。线性环是由四个或更多顶点组成的数组，其中第一个和最后一个顶点相同，形成封闭边界。每个顶点都是形态的二维点 *`[longitude, latitude]`*，其中经度和纬度的单位是度。必须按多边形周围的逆时针顺序列出顶点。

> **Note**
>
> Amazon Location Service 不支持带有多个环的多边形。这包括孔洞、岛屿或多面。Amazon Location 也不支持顺时针缠绕或穿过反子午线的多边形。

以下是单个线性外环的示例：

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

跟踪器术语

跟踪器资源

一种从设备接收位置更新的 AWS 资源。跟踪器资源为位置查询提供支持，例如当前和历史设备位置。将跟踪器资源链接到地理围栏集合会根据链接的地理围栏集合中的所有地理围栏自动评估位置更新。

跟踪位置数据

跟踪器资源会随着时间的推移存储有关您的设备的信息。这些信息包括一系列位置更新，其中每次更新都包括地点、时间和可选的元数据。元数据可以包括位置的精度，以及最多三个键值对，以帮助您跟踪每个位置的关键信息，例如正在跟踪的车辆的速度、方向、轮胎压力、剩余燃油或发动机温度。跟踪器会将设备位置记录保存 30 天。

位置筛选

在存储更新或根据地理围栏评估更新之前，位置筛选可以筛选掉不提供有价值信息的位置更新，从而帮助您控制成本并提高跟踪应用程序的质量。

您可以选择 `AccuracyBased`、`DistanceBased` 或 `TimeBased` 筛选。默认情况下，位置筛选设置为 `TimeBased`。

您可以在创建或更新跟踪器资源时配置位置筛选。

RFC3339 时间戳格式

Amazon Location Service Trackers 使用 RFC3339 格式，该格式遵循国际标准化组织 (ISO) 8601 格式的日期和时间。

格式为 “YYYY-MM--: mm: ss.sssz+ 00:00DDThh”：

- YYYY-MM-DD——表示日期格式。
- T——表明接下来将是时间值。
- hh:mm:ss.sss——以 24 小时制表示时间。
- Z— 表示使用的时区是UTC，后面可以有与UTC时区的偏差。
- +00:00—（可选）指明与UTC时区的偏差。例如，+ 01:00 表示 UTC + 1 小时。

示例

2020 年 7 月 2 日下午 12:15:20，时区再调整 1 小时。UTC

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

使用 Amazon Location Service 的常见用例

Amazon Location Service 允许您构建一系列应用程序，从资产跟踪到基于位置的营销。以下是的常见使用案例：

用户参与和地理营销

使用位置数据来构建解决方案，提高用户对目标客户的营销参与度。例如，当通过移动应用程序订购咖啡的客户在附近时，Amazon Location 可以触发一个事件，该事件会提示通知。此外，您还可以构建地理定位功能，以便零售商可以向目标商店附近的顾客发送折扣码或数字传单。

资产跟踪

构建资产跟踪功能，帮助企业了解其产品、人员和基础设施的当前位置和历史位置。借助资产跟踪功能，您可以构建多种解决方案，以优化远程人员配备，确保途中运输安全，并最大限度地提高调度效率。

送货

将定位功能集成到配送应用程序中，以存储、跟踪和协调出发地点、送货车辆及其目的地。例如，内置 Amazon Location 功能的送餐应用程序具有位置跟踪和地理围栏功能，当送货司机在附近时，可以自动通知餐厅。这减少了等待时间，有助于保持所送食物的质量。

本主题概述了您可以使用 Amazon Location 构建的应用程序的架构和步骤。

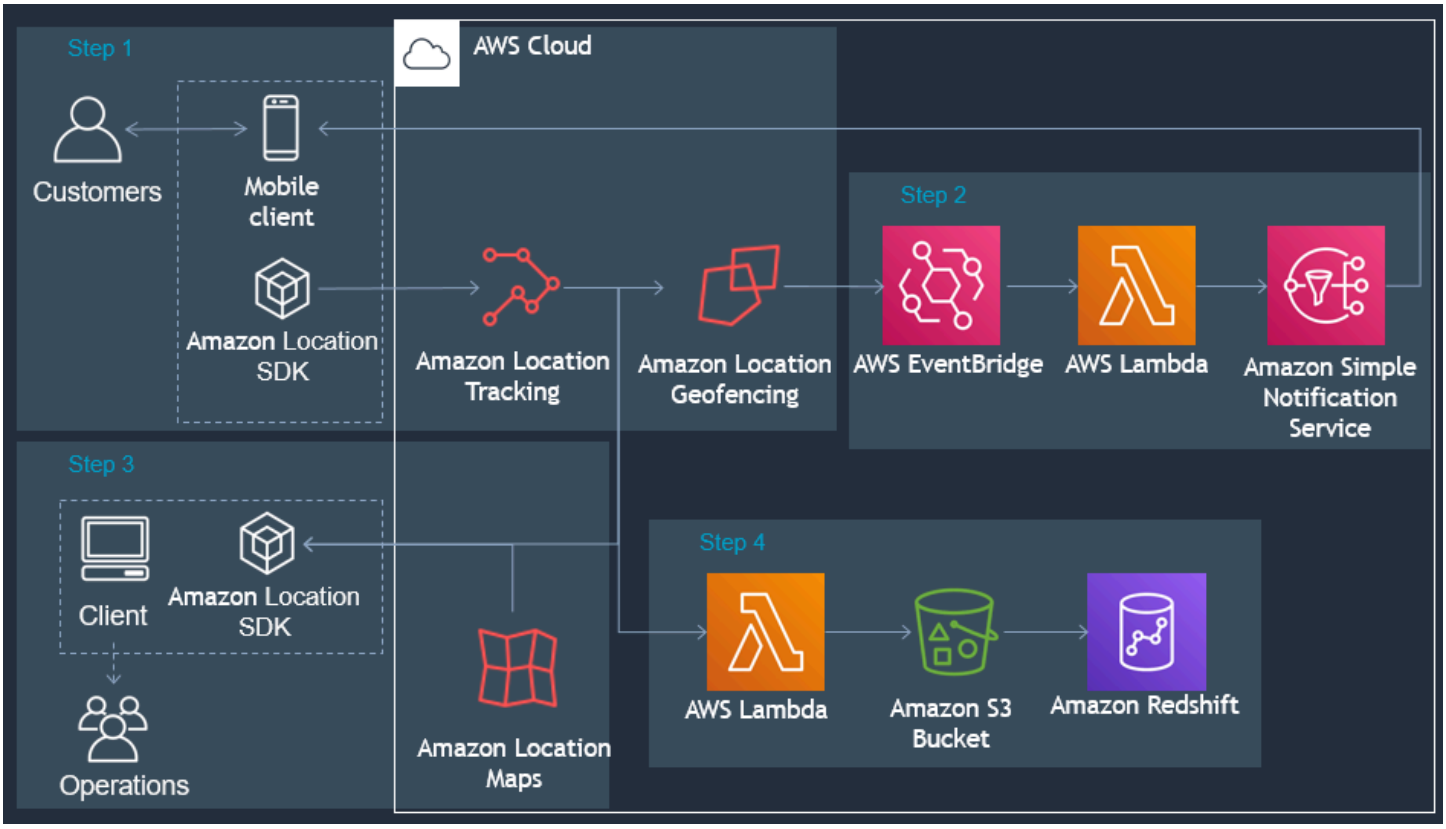
主题

用户参与和地理营销应用程序

以下是使用 Amazon Location 的用户参与和地理营销应用程序架构的示例：

使用此架构，您可以：

- 根据目标的距离发起活动，这样你就可以向附近的客户发送报价，或者吸引那些最近离开你机构的客户（称为地理定位）。
- 在地图上可视化客户设备的位置，以监控一段时间内的趋势。
- 存储客户设备位置，供您随时间推移进行分析。
- 分析位置历史记录，确定趋势和优化的机会。



以下概述了构建用户参与和地理营销应用程序所需的步骤：

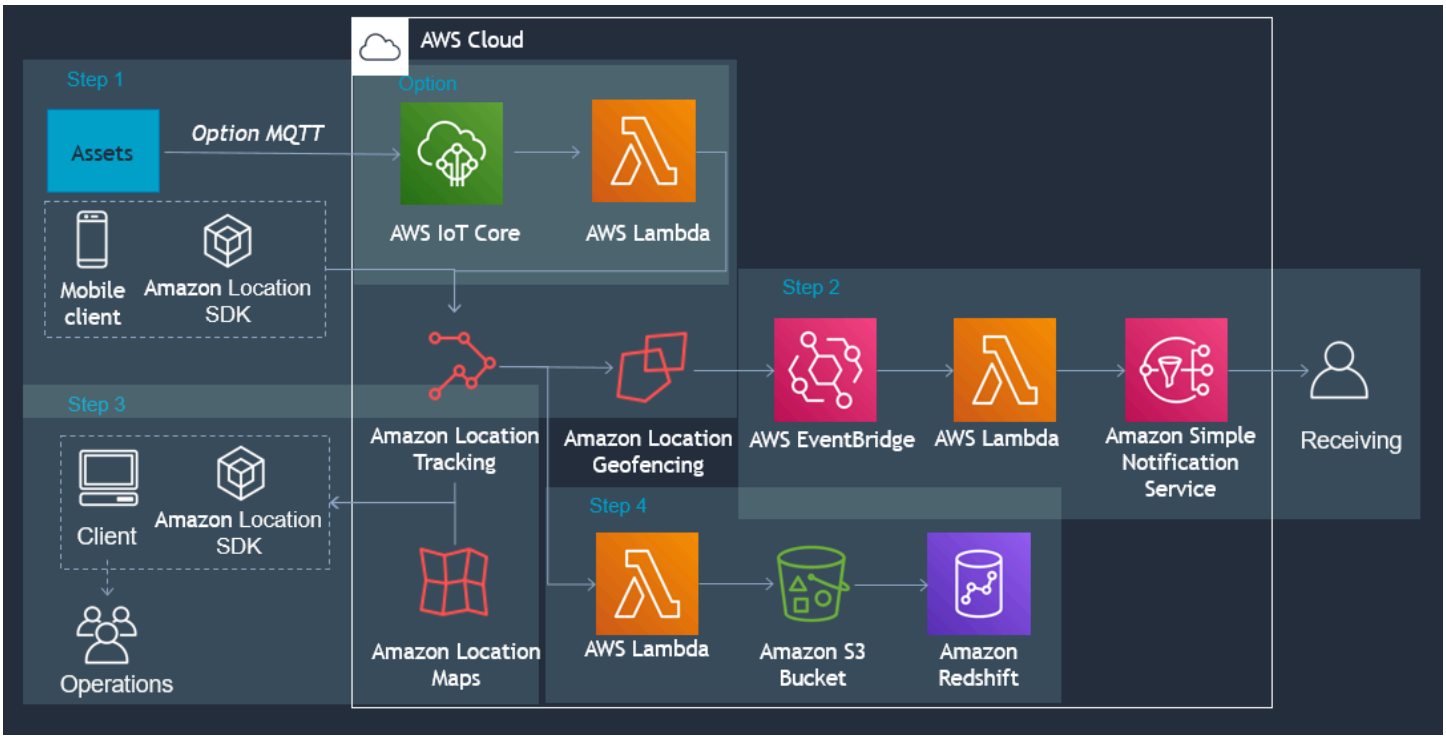
1. 在地理围栏集合中创建您的地理围栏，并将跟踪器链接到该集合。有关更多信息，请参阅 the section called “地理围栏和跟踪”。
2. 将 Amazon 配置为 EventBridge 向进入或离开地理围栏感兴趣区域的客户发送通知。有关更多信息，请参阅 the section called “对事件做出反应 EventBridge”。
3. 在地图上显示客户位置和地理围栏。有关更多信息，请参阅使用地图。
4. 将位置数据保存到长期存储中，以供进一步分析。
5. 一旦您构建了应用程序，就可以使用 Amazon CloudWatch 和 AWS CloudTrail 来管理您的应用程序。有关更多信息，请参阅the section called “使用监控 CloudWatch” 和the section called “CloudTrail 与 Amazon 位置一起使用”。

资产跟踪应用程序

以下是使用 Amazon Location 的资产跟踪应用程序架构的示例：

使用此架构，您可以：

- 在地图上显示资产位置，以展示整体情况。例如，使用历史位置或事件显示热点图，以帮助运营团队或规划团队。
- 根据资产邻近程度启动事件，通知收货部门，为货物到达做准备并缩短处理时间。
- 存储资产位置以在后端应用程序中启动操作或分析一段时间内的数据。
- 分析位置历史记录，确定趋势和优化的机会。



以下内容概述了构建资产跟踪应用程序所需的步骤：

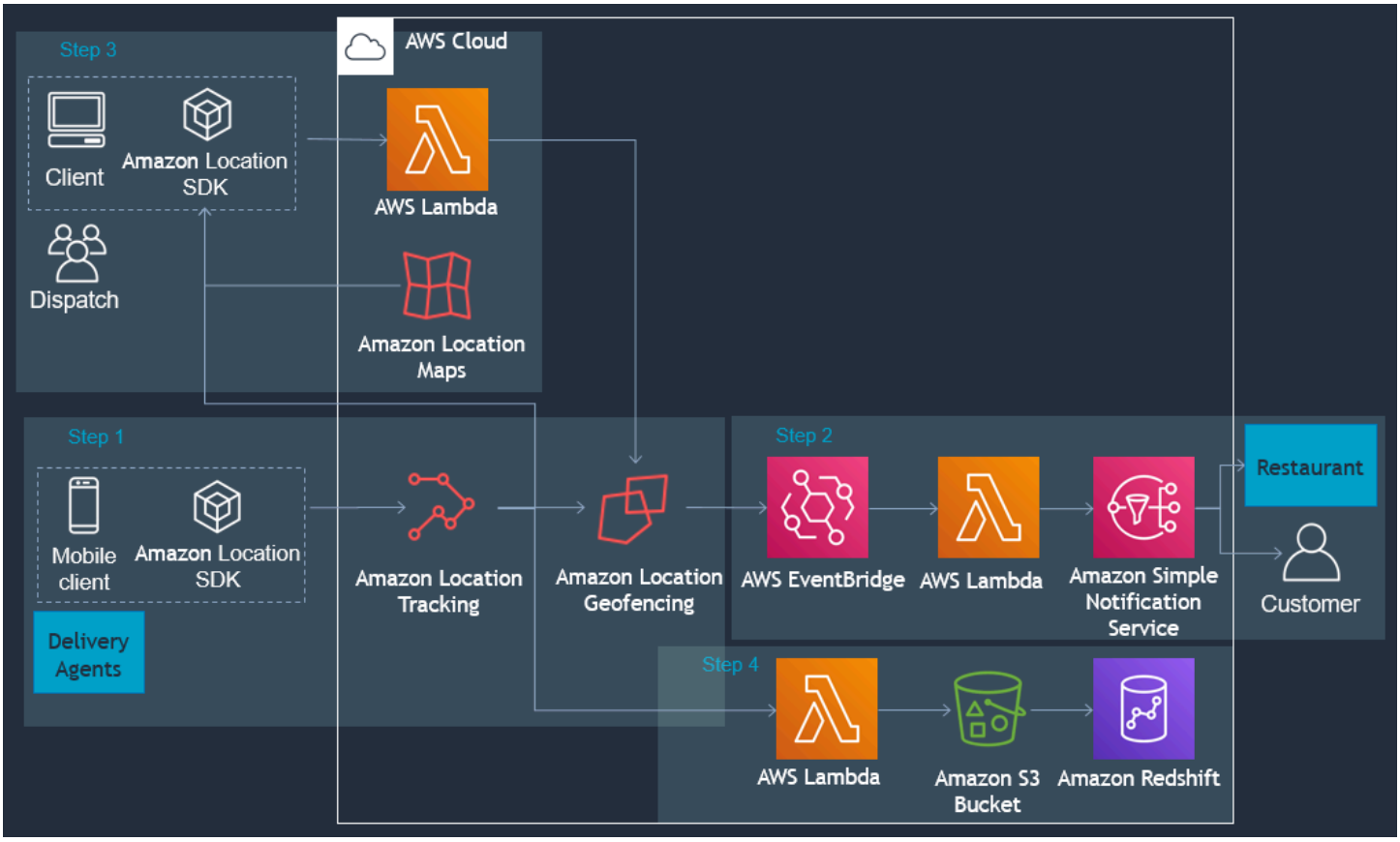
1. 在地理围栏集合中创建您的地理围栏，并将跟踪器链接到该集合。有关更多信息，请参阅 the section called “地理围栏和跟踪”。
2. EventBridge 将 Amazon 配置为发送通知或启动流程。有关更多信息，请参阅 the section called “对事件做出反应 EventBridge”。
3. 在地图上显示您追踪的资产和活跃的地理围栏。有关更多信息，请参阅使用地图。
4. 将位置数据保存在长期存储中，以供进一步分析。
5. 一旦您构建了应用程序，就可以使用 Amazon CloudWatch 和 AWS CloudTrail 来管理您的应用程序。有关更多信息，请参阅the section called “使用监控 CloudWatch” 和the section called “CloudTrail 与 Amazon 位置一起使用”。

配送应用程序

以下是使用 Amazon Location 的配送应用程序架构的示例。

使用此架构，您可以：

- 根据送货代理的距离启动事件，以便及时准备好取件，并且可以在送货到达时通知客户。
- 在地图上近乎实时地显示司机的位置以及上车和下车地点，向调度团队展示整体情况。
- 存储送货代理的位置，以便您可以在后端应用程序中对其进行操作或随着时间的推移对其进行分析。
- 分析位置历史记录，确定趋势和优化的机会。



以下是构建配送应用程序所需的步骤的概述：

1. 创建您的地理围栏集合，并将追踪的设备链接到该集合。有关更多信息，请参阅the section called “地理围栏和跟踪”。
2. 创建一项 AWS Lambda 功能，以便在预订订单时自动添加和移除地理围栏。

3. EventBridge 将 Amazon 配置为发送通知或启动流程。有关更多信息，请参阅 the section called “对事件做出反应 EventBridge”。
4. 在地图上显示追踪的资产和活跃的地理围栏。有关更多信息，请参阅使用地图。
5. 将位置数据保存到长期存储中，以供进一步分析。
6. 一旦您构建了应用程序，就可以使用 Amazon CloudWatch 和 AWS CloudTrail 来管理您的应用程序。有关更多信息，请参阅the section called “使用监控 CloudWatch” 和the section called “CloudTrail 与 Amazon 位置一起使用”。

什么是数据提供程序？

使用 Amazon Location Service 通过您的 AWS 账户访问来自多个数据提供程序的地理定位资源，无需第三方合同或集成。这可以帮助您专注于构建应用程序，而不必管理第三方帐户、证书、许可证和账单。

以下 Amazon Location Service 使用数据提供程序。

- 地图——创建地图资源时，从不同的地图提供程序中选择样式。您可以使用地图资源来构建交互式地图，以实现数据可视化。
- 地点——在创建地点索引资源时选择数据提供程序，以支持地理编码、反向地理编码和搜索查询。
- 路线——在创建路线计算器资源时，选择一个数据提供程序，以支持查询不同地理位置和应用程序中的路径计算。通过您选择的数据提供商，Amazon Location Service 使您能够根据 up-to-date 道路网络数据、实时交通数据、计划封闭和历史交通模式计算路线。

每个数据提供程序使用不同的方式收集和整理其数据。他们也可能在世界不同地区拥有不同的专业知识。本部分提供有关我们的数据提供程序的详细信息。您可以根据自己的喜好选择任何数据提供程序。

在使用 Amazon Location Service 数据提供程序时，请务必阅读条件条款。有关更多信息，请参阅AWS服务条款。另请参阅 the section called “数据隐私” 部分，了解有关 Amazon Location 如何保护您的隐私的更多信息。

数据提供程序的覆盖范围和功能

下表详细显示了每个数据提供程序的覆盖范围和功能。

数据提供程序	地理覆盖范围	功能覆盖范围	AWS 区域
Esri	全局	地图、地点、路线	提供 Amazon Location 的所有区域。
Grab	东南亚	地图、地点、路线	亚太地区（新加坡）；仅 `ap-southeast-1` 。
HERE	全局	地图、地点、路线	提供 Amazon Location 的所有区域。
Open Data (打开数据)	全局	映射	提供 Amazon Location 的所有区域。

有关每个数据提供程序的特定功能的更多信息，请参阅 数据提供程序提供的功能。

每个数据提供程序以不同的方式收集和生成数据。您可以在以下主题中了解有关其覆盖范围的更多信息：

- 覆盖范围：Esri
- 覆盖范围：Grab
- 覆盖范围：HERE
- 覆盖范围：开放数据

如果您遇到数据问题并想向数据提供程序报告错误，请参阅以下主题：

- 向 Esri 的错误报告
- GrabMaps 数据错误报告
- 向报告错误 HERE
- 报告和为开放数据做出贡献时出错

地图样式

每个数据提供程序都提供一组地图样式来呈现它们提供的地图数据。例如，样式可能包含卫星图像，或者可能经过优化以显示道路以供导航。您可以在以下主题中找到每个提供程序的样式列表和示例。

- Esri 地图样式
- Grab 地图样式
- HERE地图样式
- 开放数据地图样式

有关每个数据提供程序的更多信息

以下部分链接提供有关每个数据提供程序的信息。

- Esri
- GrabMaps
- HERE科技
- Open Data (打开数据)

Esri

Amazon Location Service 使用 Esri 的定位服务来帮助 AWS 客户有效地使用地图、进行地理编码和计算路线。Esri的定位服务由制图师、地理学家和人口学家组成的专家团队精心策划的高质量、权威的 ready-to-use 位置数据构建。

有关更多功能信息，请参阅 Amazon Location Service 数据提供程序上的 Esri。

主题

- Esri 地图样式
- 覆盖范围：Esri
- 使用条款和数据属性：Esri
- 向 Esri 的错误报告

Esri 地图样式

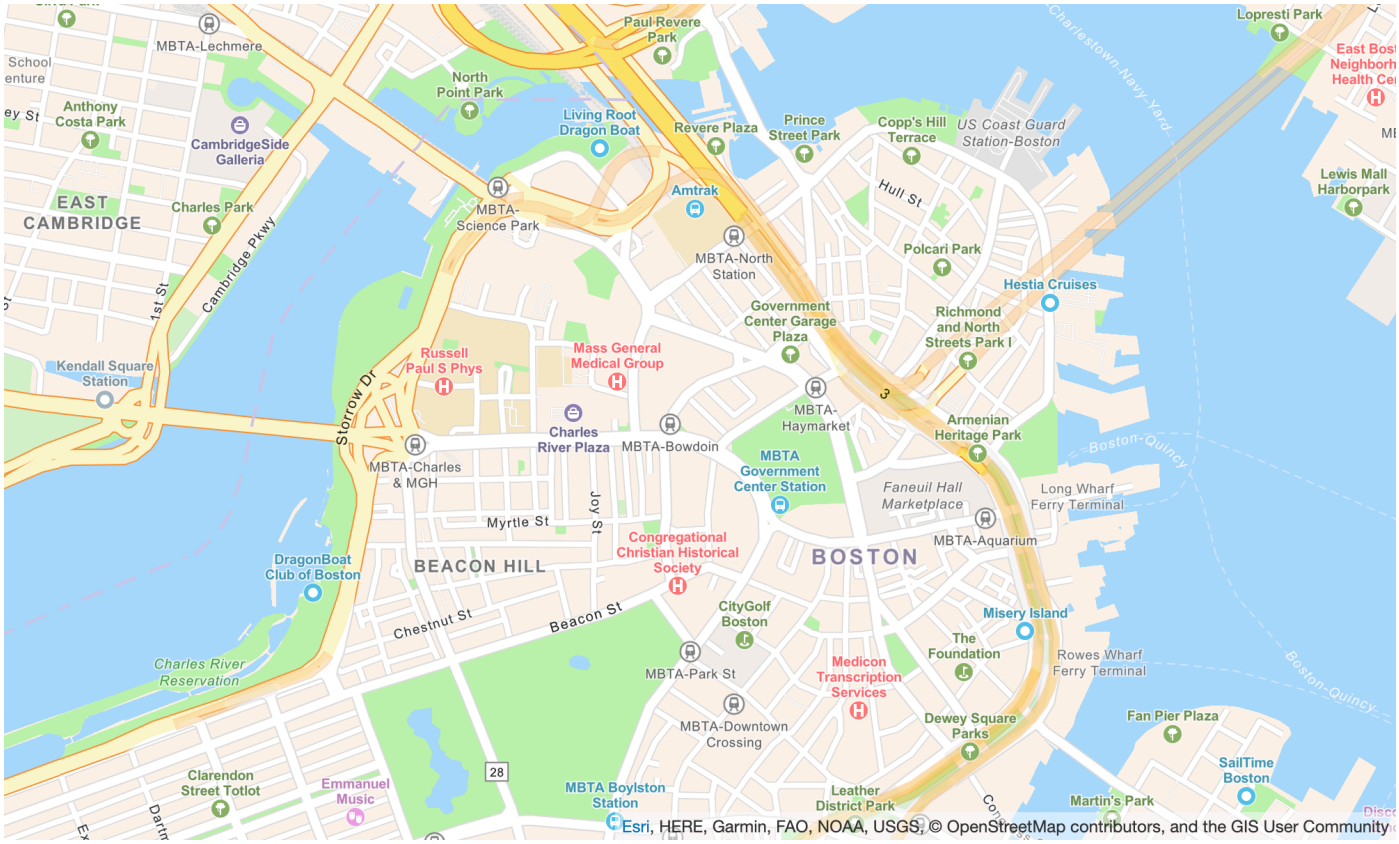
创建地图资源时，Amazon Location Service 支持以下 Esri 地图样式。

> **Note**
>
> 不支持本部分中未列出的 Esri 地图样式。

Esri 矢量样式支持备用 政治观点。

Esri Navigation

Esri 导航



地图样式名称：`VectorEsriNavigation`

这张地图提供详尽的世界符号化底图，并采用为日间移动设备使用而设计的自定义导航地图样式。

这张全面的街道图包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。此地图中的矢量切片图层是使用与世界街道图和其他 Esri 底图相同的数据源构建的。通过设置`POI`图层来启用该图层 CustomLayers，以利用其他地点数据。

有关更多信息，请参阅 Esri 网站上的 Esri 世界导航。

> **Note**
>
> 上图所示的 `VectorEsriNavigation` 地图已启用 `POI` 层。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Arial 斜体
- Arial 常规
- Arial 粗体
- Arial Unicode MS Bold
- Arial Unicode MS Regular

Esri Imagery

Esri 影像

地图样式名称：`RasterEsriImagery`

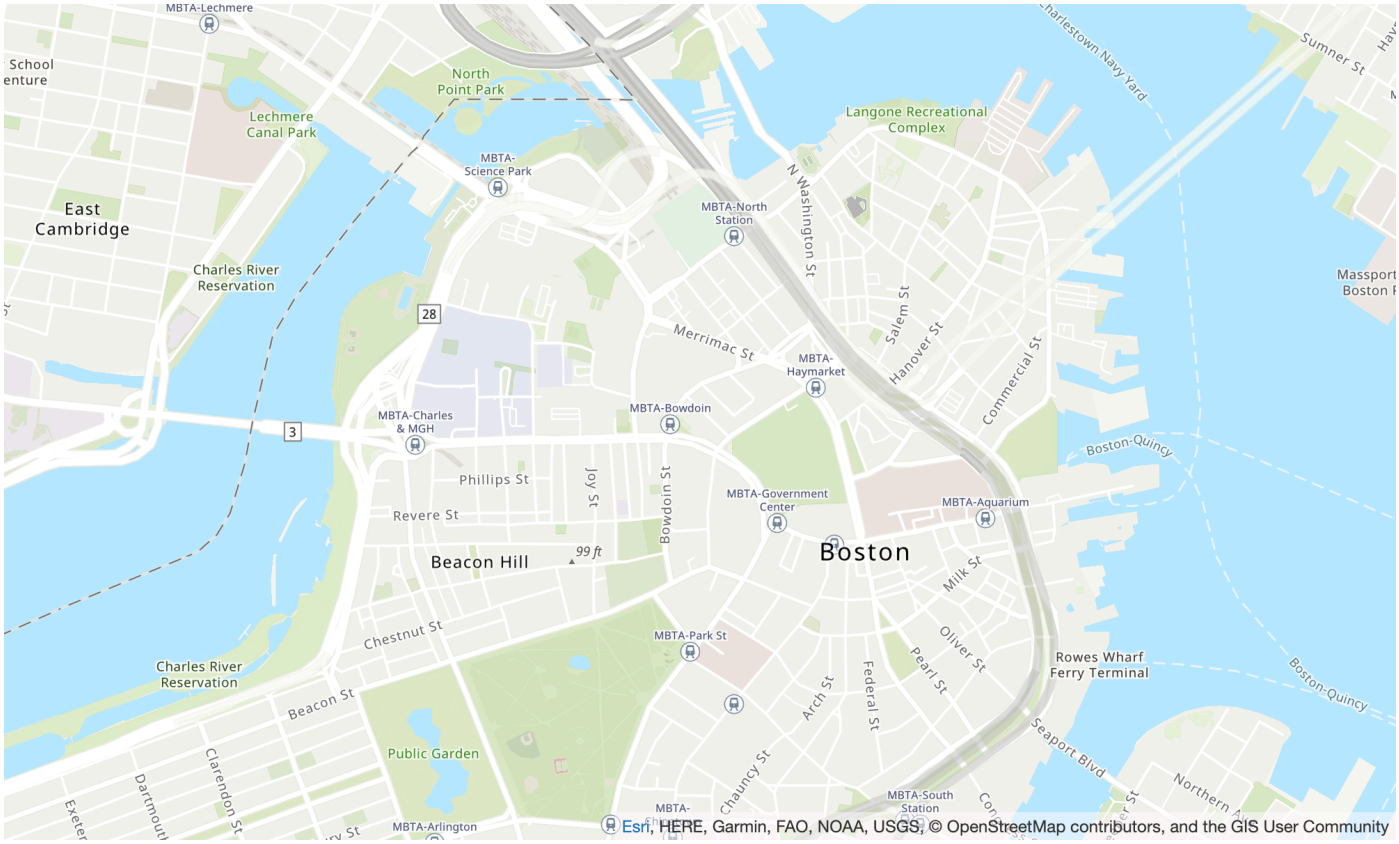
这张地图提供一米或更精细卫星和航拍图像的光栅底图，而全球范围的卫星图像分辨率较低。

该地图包括15米小比例和中比例的影像（约 1:591 M 至约 1:72 k）和 2.5 米的世界SPOT影像（约 1:288 k 到 1:72 k）。该地图包含来自 Maxar 的美国大陆和西欧部分地区的 0.5 米分辨率图像。这张地图包含世界许多地方的其他 Maxar 亚米级分辨率的影像。在世界其他地区，GIS用户社区贡献了不同分辨率的图像。在某些社区中，可以获得比例低至约 1:280 的超高分辨率图像（低至 0.03 米）。

有关更多信息，请参阅 Esri 网站上的 Esri 世界图像。

Esri Light

Esri 浅色



地图样式名称：`VectorEsriTopographic`

这提供了一个用经典 Esri 地图样式符号化的全球详细底图。这包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。

该底图由多个数据提供商提供的各种权威来源汇编而成，包括美国地质调查局 (USGS)、美国环境保护署 (EPA)、美国国家公园管理局 (NPS)、联合国粮食及农业组织 (FAO)、加拿大自然资源部 (NRCAN) 和 Esri。HERE特定区域的数据来自 OpenStreetMap 贡献者。此外，数据由GIS社区提供。

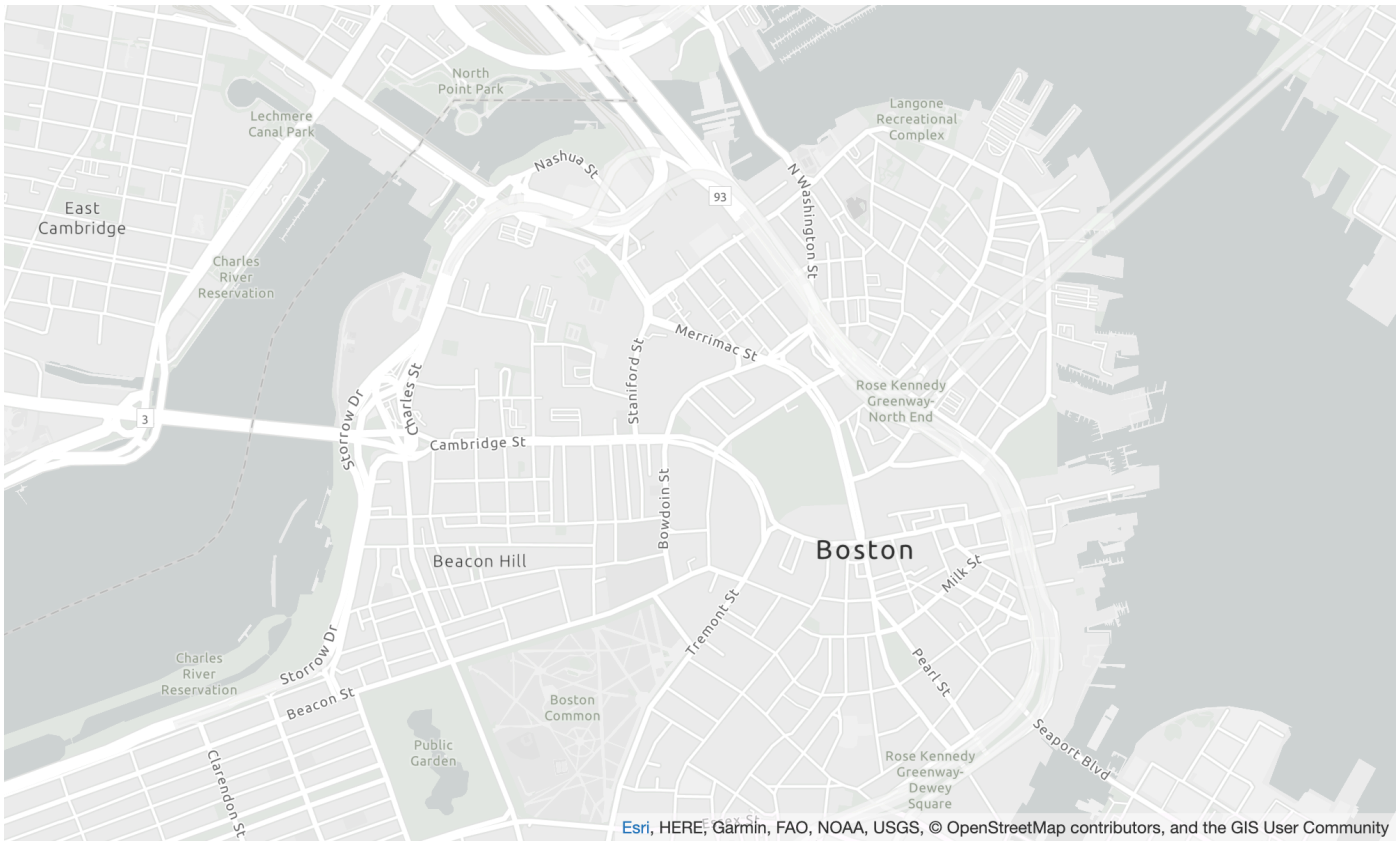
字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Noto Sans 斜体
- Noto Sans 常规
- Noto Sans 粗体
- Noto Serif 常规
- Roboto Condensed Light 斜体

Esri Light Gray Canvas

Esri 浅灰色画布



地图样式名称：`VectorEsriLightGrayCanvas`

这张地图为全球提供了一个用浅灰色、中性背景风格符号化的详细底图，颜色、标签和特征都很少，旨在吸引人们对您主题内容的注意。

此矢量切片图层使用与浅灰色画布和其他 Esri 底图相同的数据源构建。这张地图包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。

有关更多信息，请参阅 Esri 网站上的 Esri 浅灰色画布。

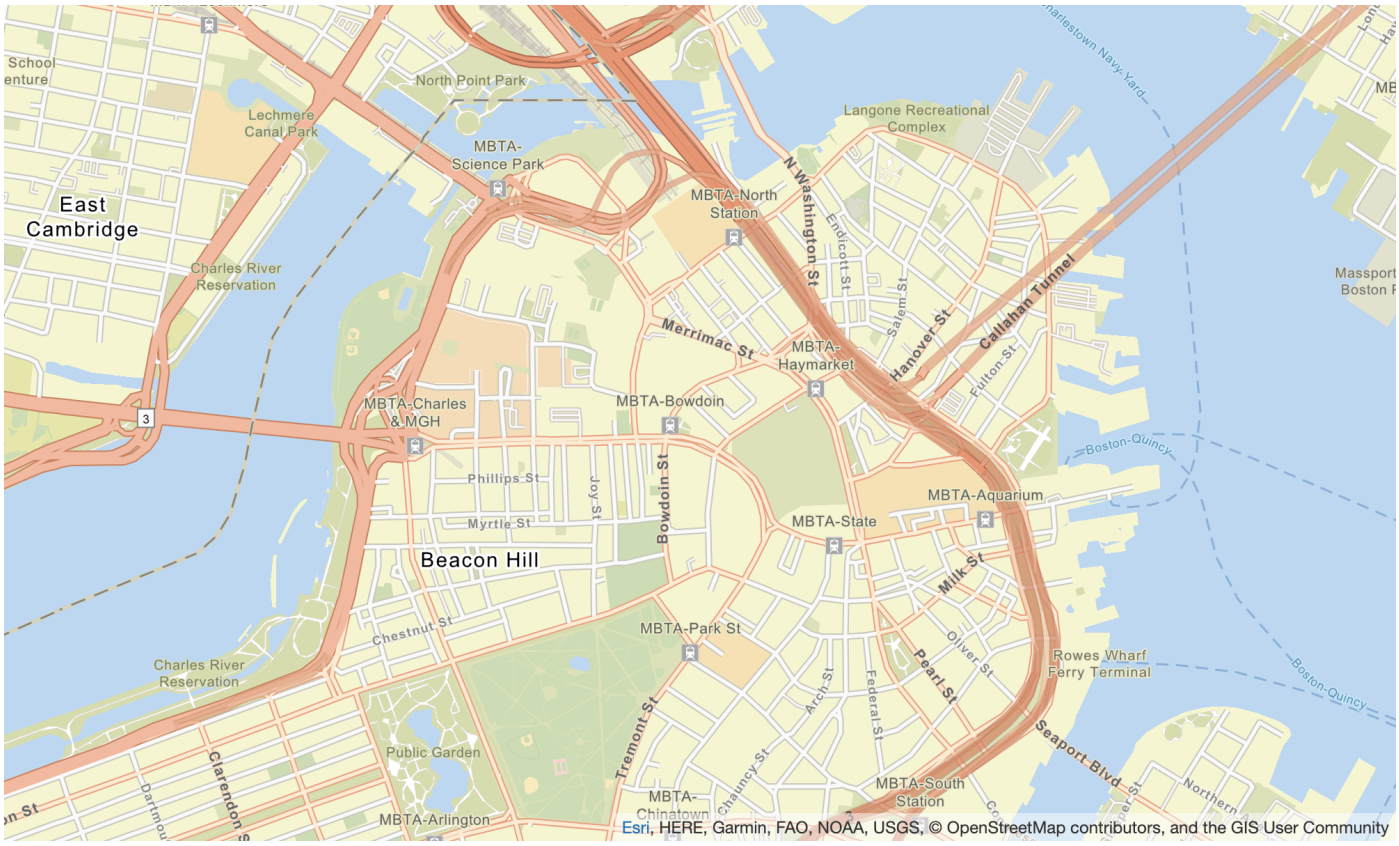
字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Ubuntu 斜体
- Ubuntu 常规
- Ubuntu 轻体
- Ubuntu 粗体

Esri Street Map

Esri 街道图



地图样式名称：`VectorEsriStreets`

这张地图提供详尽的世界符号化底图，并采用为日间移动设备使用而设计的自定义导航地图样式。

这张全面的街道图包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。其中还包括一系列更丰富的地点，例如商店、服务设施、餐厅、景点和其他兴趣点。此地图中的矢量切片图层是使用与世界街道图和其他 Esri 底图相同的数据源构建的。

有关更多信息，请参阅 Esri 网站上的 Esri 世界街道。

字体

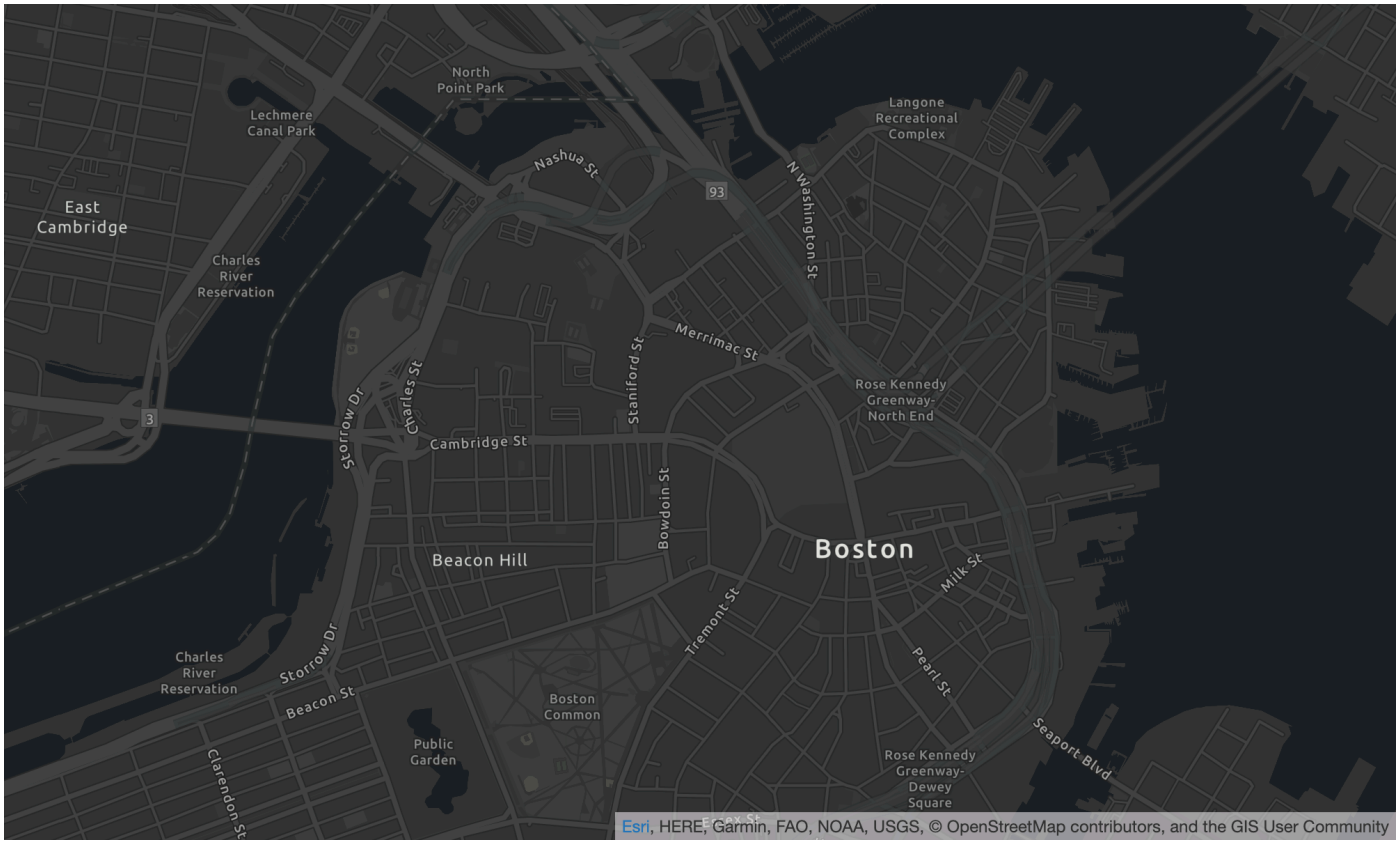
Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Arial 斜体
- Arial 常规
- Arial 粗体

- Arial Unicode MS Bold
- Arial Unicode MS Regular

Esri Dark Gray Canvas

Esri 深灰色画布



地图样式名称：`VectorEsriDarkGrayCanvas`

这张地图为全球提供了一个用深灰色、中性背景风格符号化的详细矢量底图，颜色、标签和特征都很少，旨在吸引人们对您主题内容的注意。

这张地图包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。此地图中的矢量切片图层使用与深灰色画布栅格地图和其他 Esri 底图相同的数据源构建。

有关更多信息，请参阅 Esri 网站上的 Esri 深灰色画布。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Ubuntu 中等斜体

- Ubuntu 中等
- Ubuntu 斜体
- Ubuntu 常规
- Ubuntu 粗体

覆盖范围：Esri

在创建地点索引资源时，可以使用 Esri 作为数据提供程序来支持地理编码、反向地理编码和搜索查询，或者在创建路线计算器资源时支持查询以计算路线。

Esri 在世界不同地区提供不同级别的数据质量。有关您感兴趣的地区覆盖范围的更多信息，请参阅：

- Esri 关于地理编码覆盖范围的详细信息
- Esri 关于街道网络和交通覆盖范围的详细信息

使用条款和数据属性：Esri

在使用 Esri 的数据之前，请务必遵守所有适用的法律要求，包括适用于 Esri 的许可条款和。 AWS

有关 AWS 要求的更多信息，请参阅AWS服务条款。

有关 Esri 属性指南的信息，请参阅 Esri 的数据属性和使用条款。

向 Esri 的错误报告

如果您遇到数据问题并想向 Esri 报告错误和差异，请按照 Esri 的技术支持文章中的操作方法：提供有关底图和地理编码的反馈。

GrabMaps

Grab 是东南亚最大的送货组织，拥有数百万的合作车主和客户。他们的子公司在这些国家/地区创建up-to-date地图数据供自己和其他人使用。GrabMapsAmazon Location Service 使用 GrabMaps “定位服务” 来帮助 AWS 客户有效地使用地图、进行地理编码和计算路线。 GrabMaps'定位服务旨在为东南亚国家提供高质量、权威的 ready-to-use 位置数据。

有关其他功能的信息，请参阅 Amazon Locat GrabMapsion Service 数据提供商。

> **Important**
>
> Grab 仅为东南亚地区提供地图，并且仅在亚太地区（新加坡）区域 (ap-southeast-1) 中可用。有关更多信息，请参阅 覆盖的国家/地区和区域。

主题

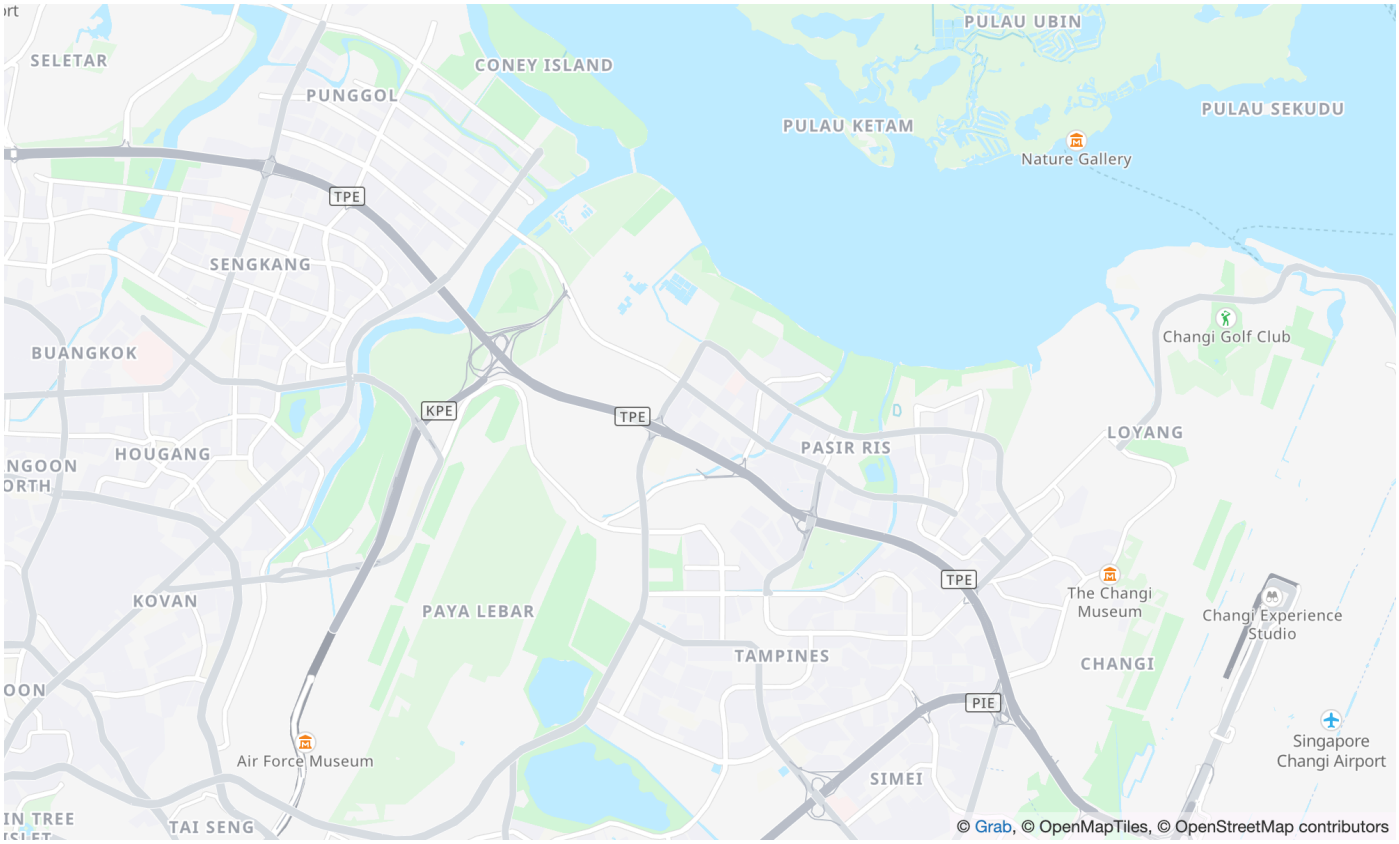
Grab 地图样式

创建地图资源时，Amazon Location Service 支持以下 Grab 地图样式：

> **Note**
>
> 目前不支持本部分中未列出的 Grab 地图样式。

Grab Standard Light Map

Grab 标准（浅色）地图



地图样式名称：`VectorGrabStandardLight`

Grab 标准底图，其中包含详细的土地使用着色、区域名称、道路、地标和覆盖东南亚地区的兴趣点。

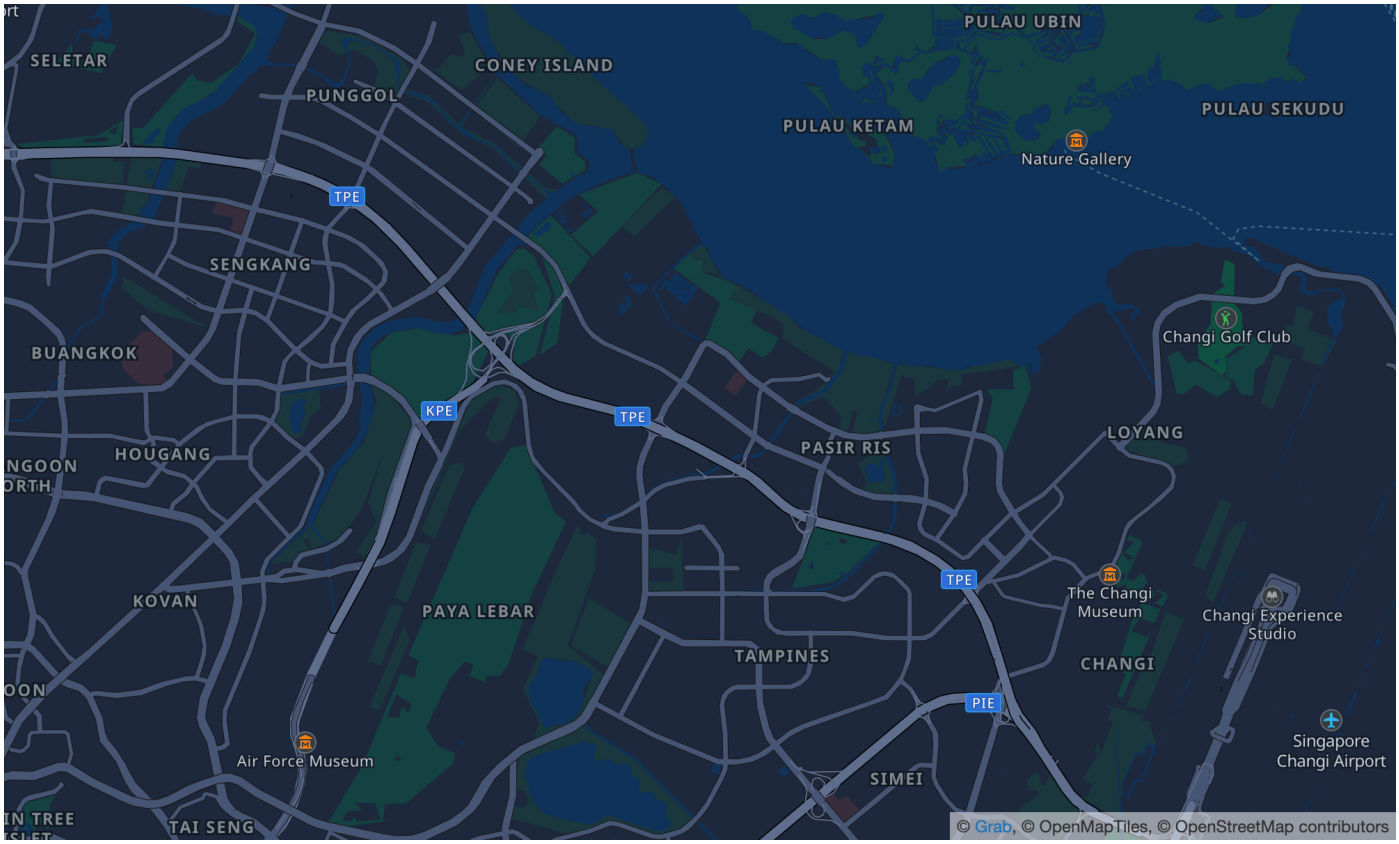
字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Noto Sans 常规
- Noto Sans 中等
- Noto Sans 粗体

Grab Standard Dark Map

Grab 标准（深色）地图



地图样式名称：`VectorGrabStandardDark`

Grab 标准底图的深色变体，其中包含详细的土地使用着色、区域名称、道路、地标和覆盖东南亚地区的兴趣点。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Noto Sans 常规
- Noto Sans 中等
- Noto Sans 粗体

覆盖范围：Grab

在创建地点索引资源时，您可以使用 Grab 作为数据提供程序来支持地理编码、反向地理编码和搜索查询，或者在创建路线计算器资源时支持查询以计算路线。

覆盖的国家/地区和区域

Grab 仅为东南亚地区提供地图，并且仅在亚太地区（新加坡）区域 (`ap-southeast-1`) 中可用。

Grab 提供以下国家/地区的详细数据：

- 马来西亚
- 菲律宾
- 泰国
- 新加坡
- 越南
- 印度尼西亚
- 缅甸
- 柬埔寨

> **Note**
>
> 在这些区域之外，使用 Grab 作为数据提供程序创建的 Amazon Location Service 资源不会提供任何结果。这包括搜索结果或路线。

来自 Grab 的地图在以下边界内：

- 南——纬度 -21.943045533438166
- 西——经度 90.0
- 北——纬度 31.952162238024968
- 东——经度 146.25

对于 1–4 的缩放级别，Grab 包括全球覆盖范围。对于缩放级别 5 及以下，地图图块仅在此限定框内提供。

> **Note**
>
> 在此限定框之外，使用 Grab 作为数据提供程序创建的 Amazon Location Service 地图资源将不会返回地图图块。为避免在应用程序中看到 404 错误，您可以使用边界框限制地图，如 使用设置地图的范围 MapLibre 中所述。

Grab 路由出行模式

在路线方面，Grab 提供先前列出的所有国家/地区的汽车和摩托车路线。

Grab 不支持卡车路线。

对于自行车和步行路线, Grab 支持以下城市:.

- 新加坡
- 雅加达
- 马尼拉
- 巴生谷
- 曼谷
- 胡志明市
- 河内

使用条款和数据属性：Grab

使用Grab的数据时，您必须遵守所有适用的法律要求，包括适用于Grab和的许可条款 AWS。

有关 AWS 要求的更多信息，请参阅AWS服务条款。

有关 GrabMaps “归因指南” 的信息，请参阅 Grab 的数据归因和使用条款第 9.23 节。

GrabMaps 数据错误报告

如果您在使用来自的数据时遇到问题 GrabMaps，并且想要报告错误或差异，请联系 AWS 技术支持。

HERE科技

Amazon Location Service 使用 HERE Technologies 的定位服务来帮助 AWS 客户有效地使用地图、地理编码和计算路线。HERE的位置数据提供了一个开放、安全和私密的以位置为中心的平台。通过选择HERE位置数据，您就是在选择本机部署在 AWS 云端的准确、新鲜和强大的数据。

有关其他功能信息，请参阅 Amaz HEREon Location Service 数据提供商。

主题

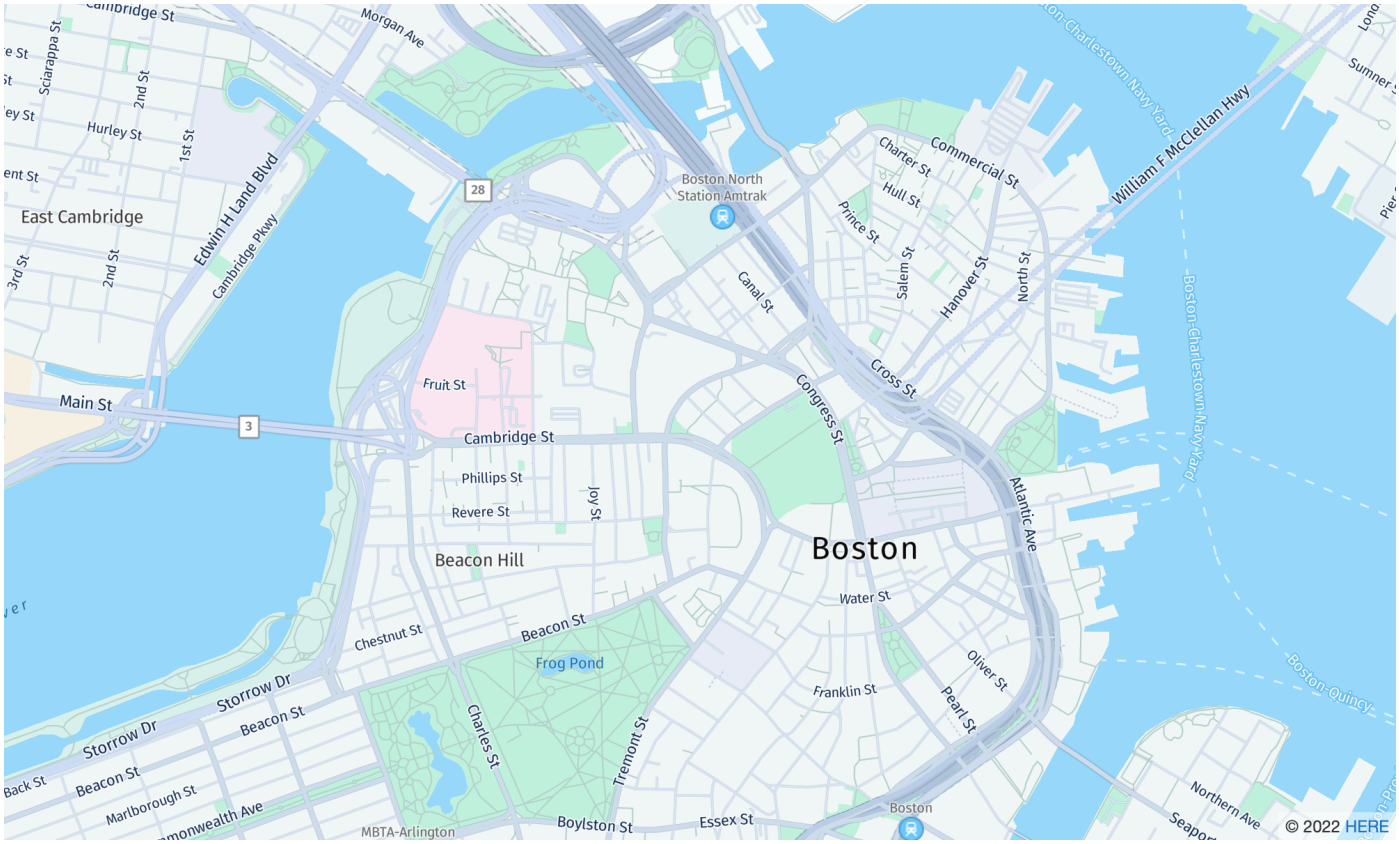
HERE地图样式

创建地图资源时，Amazon Location Service 支持以下HERE地图样式：

> **Note**
>
> HERE目前不支持本节中未列出的地图样式。

HERE Explore

HERE探索



地图样式名称：`VectorHereExplore`

HERE探索

一张详细、中立的世界底图。该街道图包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。包含完全设计的日本地图。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Fira GO 斜体
- Fira GO 常规
- Fira GO 粗体
- Noto Sans CJK JP 灯
- Noto Sans CJK JP 普通版
- Noto Sans CJK JP Bold

HERE Imagery

HERE影像

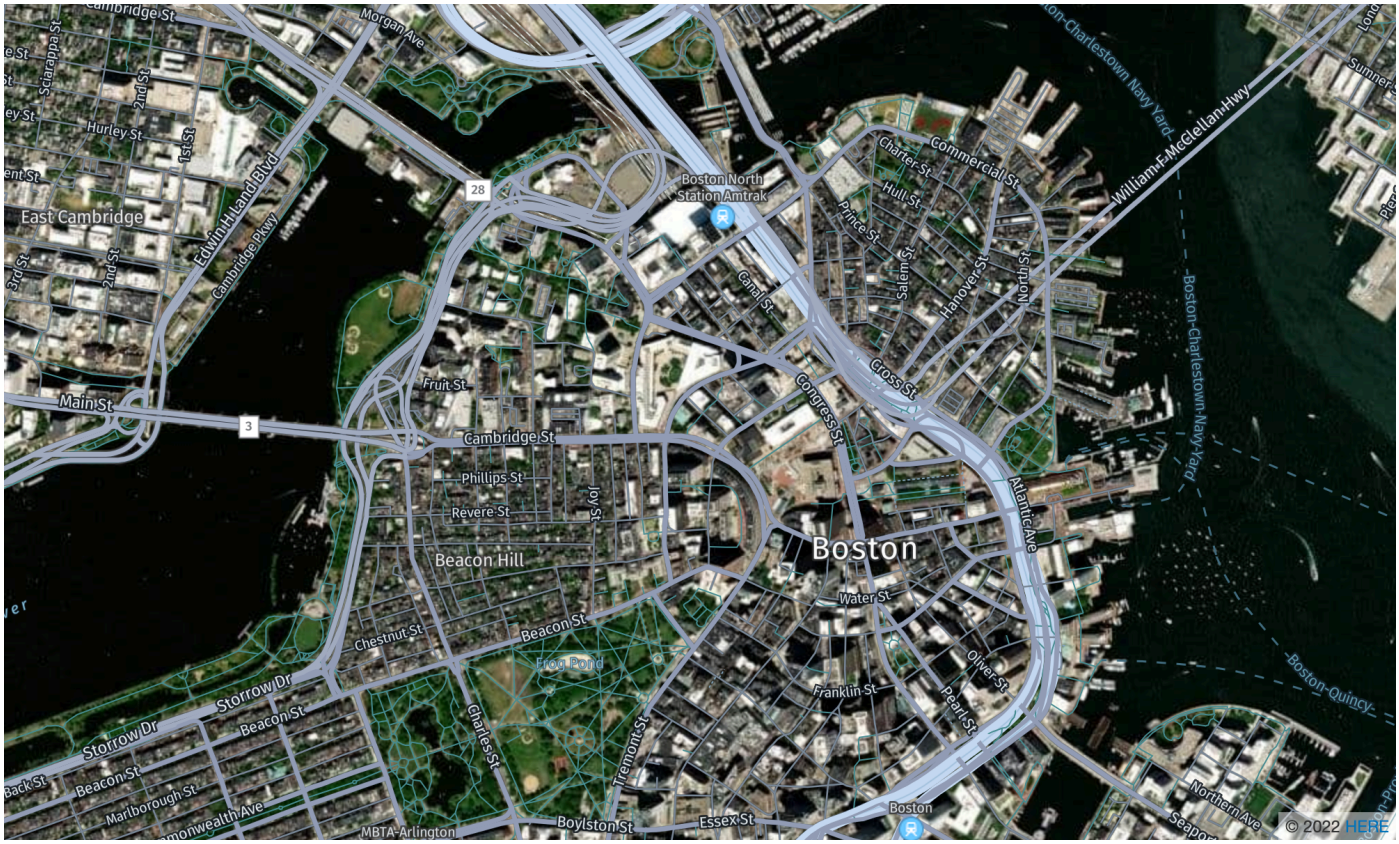
地图样式名称：`RasterHereExploreSatellite`

HERE影像

HERE影像提供覆盖全球的高分辨率卫星图像。

HERE Hybrid

HERE混合动力



地图样式名称：`HybridHereExploreSatellite`

HERE混合动力

HERE混合样式在卫星影像上显示道路网络、街道名称和城市标签。此样式可叠加两个地图图块：背景为卫星图像（光栅图块），顶部为道路网络和标注（矢量切片）。此样式将自动检索渲染地图所需的光栅和矢量切片。

> **Note**
>
> 在渲染您所看到的地图时，混合样式将同时使用矢量和光栅图块。这意味着检索到的图块数量要比单独使用矢量或光栅图块时更多。您的费用将包括检索到的所有图块。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Fira GO 斜体
- Fira GO 常规

- Fira GO 粗体
- Noto Sans CJK JP 灯
- Noto Sans CJK JP 普通版
- Noto Sans CJK JP Bold

HERE Contrast (Berlin)

HERE对比（柏林）

地图样式名称：`VectorHereContrast`

HERE对比（柏林）

混合 3D 和 2D 渲染的详尽世界底图。高对比度街道图包括高速公路、主要道路、次要道路、铁路、水域特征、城市、公园、地标、建筑物轮廓和行政边界。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

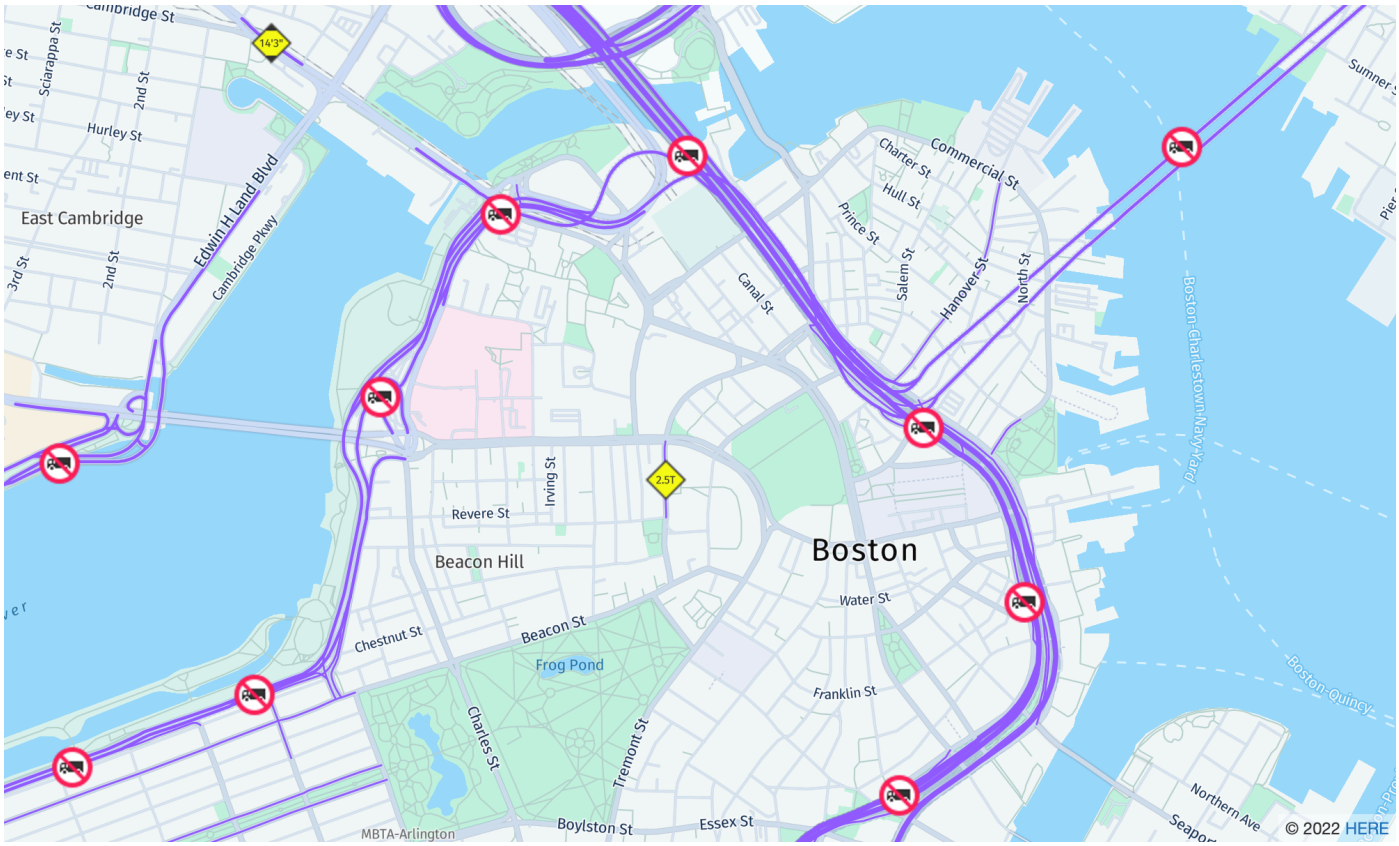
- Fira GO 常规

- Fira GO 粗体

> **Note**
>
> 此风格已从VectorHereBerlin（HERE柏林地图）重命名。 VectorHereBerlin已弃用，但仍可在使用它的应用程序中使用。

HERE Explore Truck

HERE探索卡车



地图样式名称：VectorHereExploreTruck

HERE探索卡车

一张详细、中立的世界底图。街道地图建立在 “HERE探索” 样式之上，并使用符号和图标突出显示轨道限制和属性（包括宽度、高度和HAZMAT），以支持运输和物流领域的用例。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Fira GO 斜体
- Fira GO 常规
- Fira GO 粗体
- Noto Sans CJK JP 灯
- Noto Sans CJK JP 普通版
- Noto Sans CJK JP Bold

有关世界不同地区地图数据质量的更多信息，请参阅HERE地图覆盖范围。

覆盖范围：HERE

在创建地点索引资源时，可用HERE作数据提供器来支持地理编码、反向地理编码和搜索查询，或者在创建路径计算器资源时支持查询以计算路径。

HERE在世界不同地区提供不同水平的数据质量。有关您感兴趣地区的保险的其他信息，请参阅以下内容：

- HERE地理编码覆盖范围
- HERE汽车路线覆盖范围
- HERE卡车路线覆盖范围

使用条款和数据归因：HERE

在使用HERE数据之前，请务必遵守所有适用的法律要求，包括适用于HERE和的许可条款 AWS。由于许可限制，您不得使用HERE来存储日本位置的地理编码结果。

有关 AWS 要求的信息，请参阅AWS服务条款。

有关归因指南HERE的更多信息，请参阅 Te HERE chnologies 的《适用于位置和其他内容的供应商条款》第 2 节。

向报告错误 HERE

要向报告地图错误和差异HERE，请转至https://www.here.com/contact并选择报告地图错误。

Open Data (打开数据)

Amazon Location Service 允许通过开放数据提供程序访问开源地图数据。Open Data 提供基于 OpenStreetMap (OSM)、自然地球和其他开放数据源的日光地图分布构建的全球底图。所提供的地图旨在支持不同的应用程序和用例，包括物流和配送，以及网络和移动环境中的数据可视化。该OSM社区拥有超过一百万的地图制作者，每天更新数十万个要素。Amazon Location Service 会定期纳入这些编辑内容。

有关更多功能信息，请参阅 Amazon Location Service 数据提供程序上的开放数据。

主题

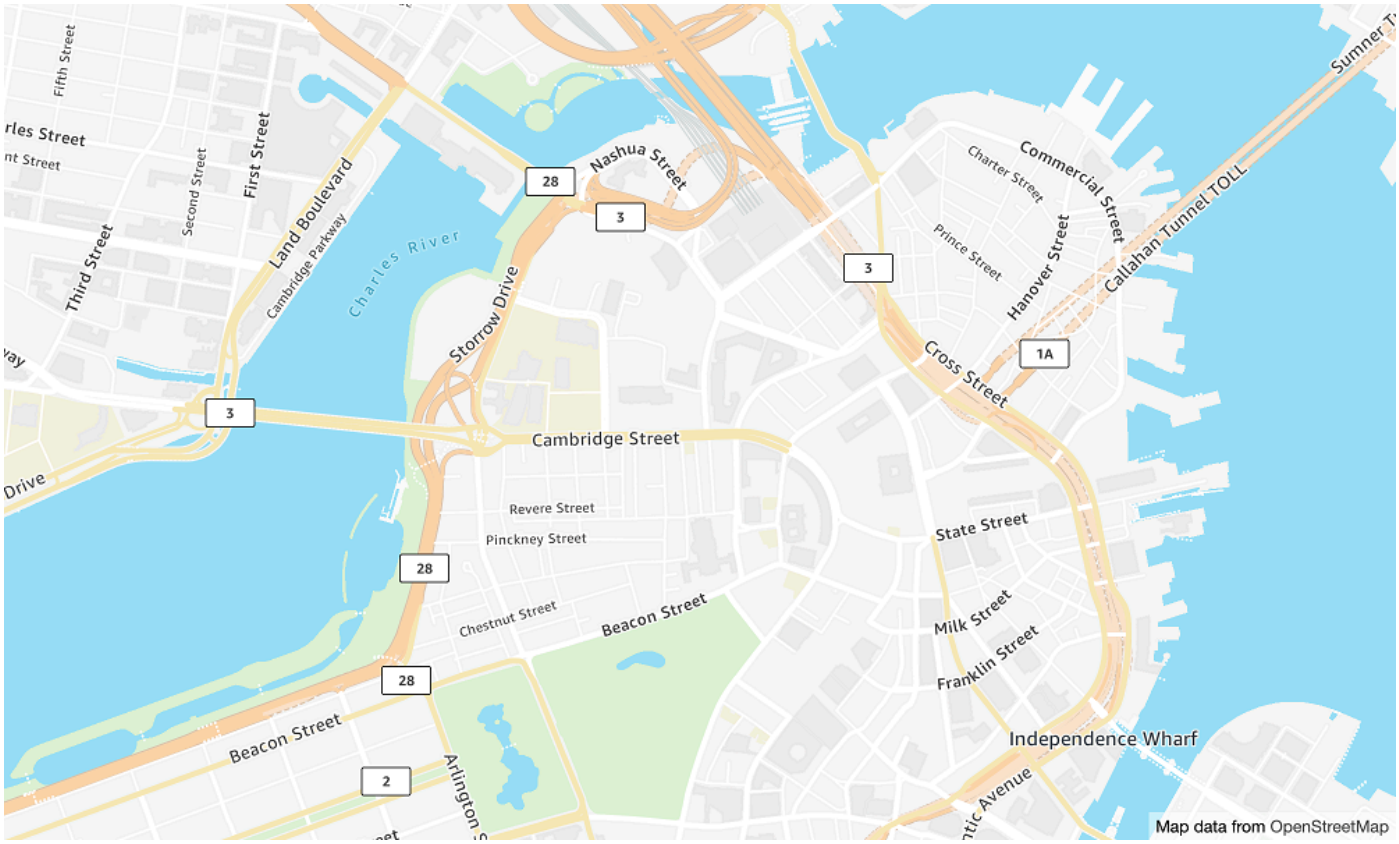
开放数据地图样式

创建地图资源时，Amazon Location Service 支持以下地图样式：

开放数据地图样式支持备用 政治观点。

Open Data Standard Light

开放数据标准（浅色）



地图样式名称：`VectorOpenDataStandardLight`

这以 Light 地图样式为世界各地提供了适合网站和移动应用程序使用的详细底图。这包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。

此底图基于 OpenStreetMap (OSM) 贡献者编译的OSM日光地图分布。该OSM社区包括超过180万贡献者，他们每天更新超过50万个功能。Amazon Location Service 会定期整合这些编辑内容。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

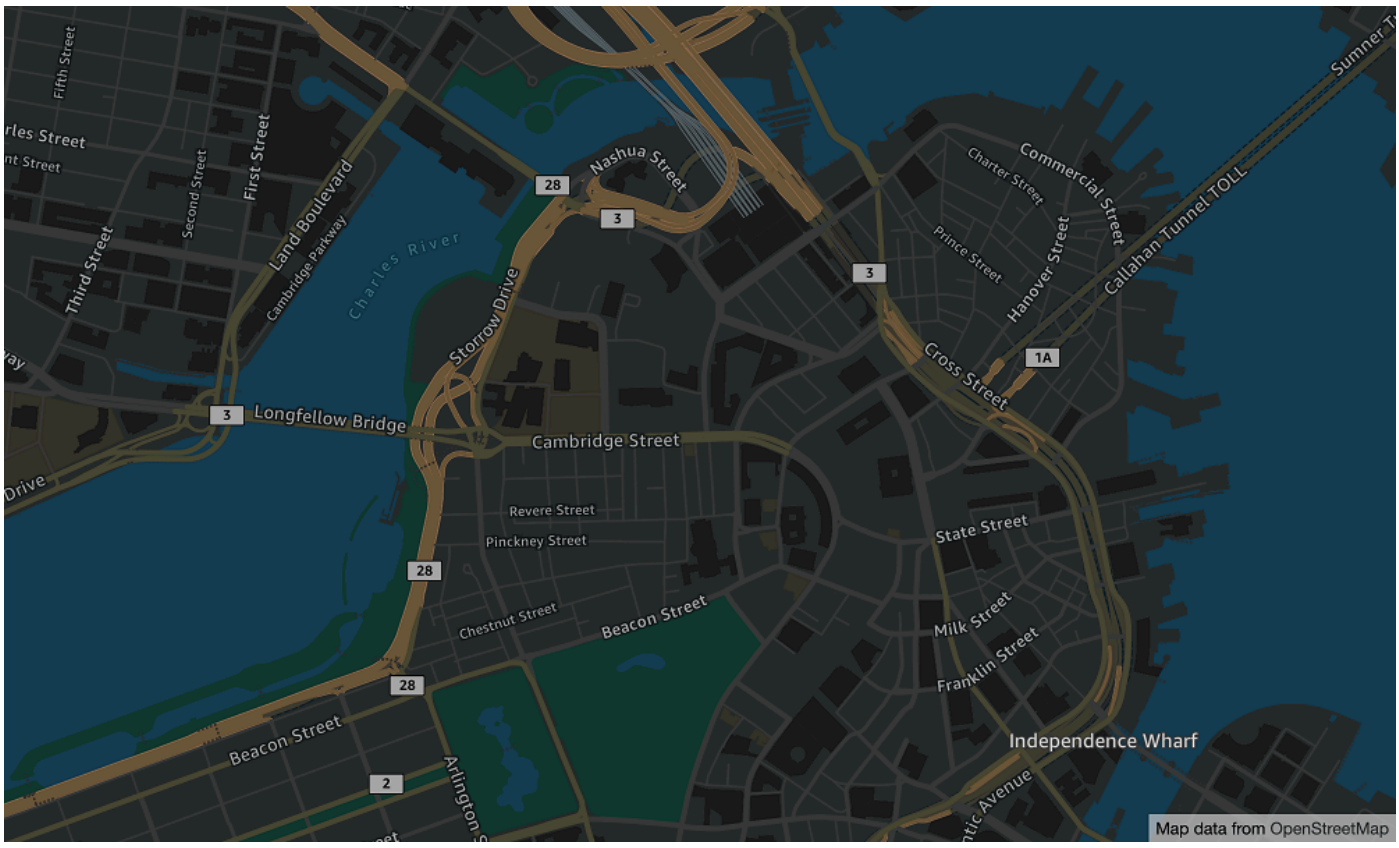
- Amazon Ember 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 常规、Noto Sans 常规
- Amazon Ember 中等、Noto Sans 中等
- Amazon Ember 常规斜体、Noto Sans 斜体
- Amazon Ember 常规、Noto Sans 常规

- Amazon Ember 常规、Noto Sans 常规、Noto Sans Arabic 常规
- Amazon Ember Consending RC 粗体、Noto Sans 粗体、Noto Sans
- Amazon Ember 粗体、Noto Sans 粗体、Noto Sans Arabic 粗体
- Amazon Ember 常规斜体、Noto Sans 斜体、Noto Sans Arabic 常规
- Amazon Ember Condensed RC 常规、Noto Sans 常规、Noto Sans Arabic Condensed 常规
- Amazon Ember 中等、Noto Sans 中等、Noto Sans Arabic 中等

> **Note**
>
> `VectorOpenDataStandardLight` 使用的字体是组合字体，把 `Amazon Ember` 用于大多数字形，但把 `Noto Sans` 用于 `Amazon Ember` 不支持的字形。

Open Data Standard Dark

开放数据标准（深色）



地图样式名称：`VectorOpenDataStandardDark`

这是一种深色主题的地图样式，提供详尽的底图，适合网站和移动应用程序使用。这包括高速公路、主要道路、次要道路、铁路、水景、城市、公园、地标、建筑足迹和行政边界。

此底图基于 OpenStreetMap (OSM) 贡献者编译的OSM日光地图分布。该OSM社区包括超过180万贡献者，他们每天更新超过50万个功能。Amazon Location Service 会定期整合这些编辑内容。

字体

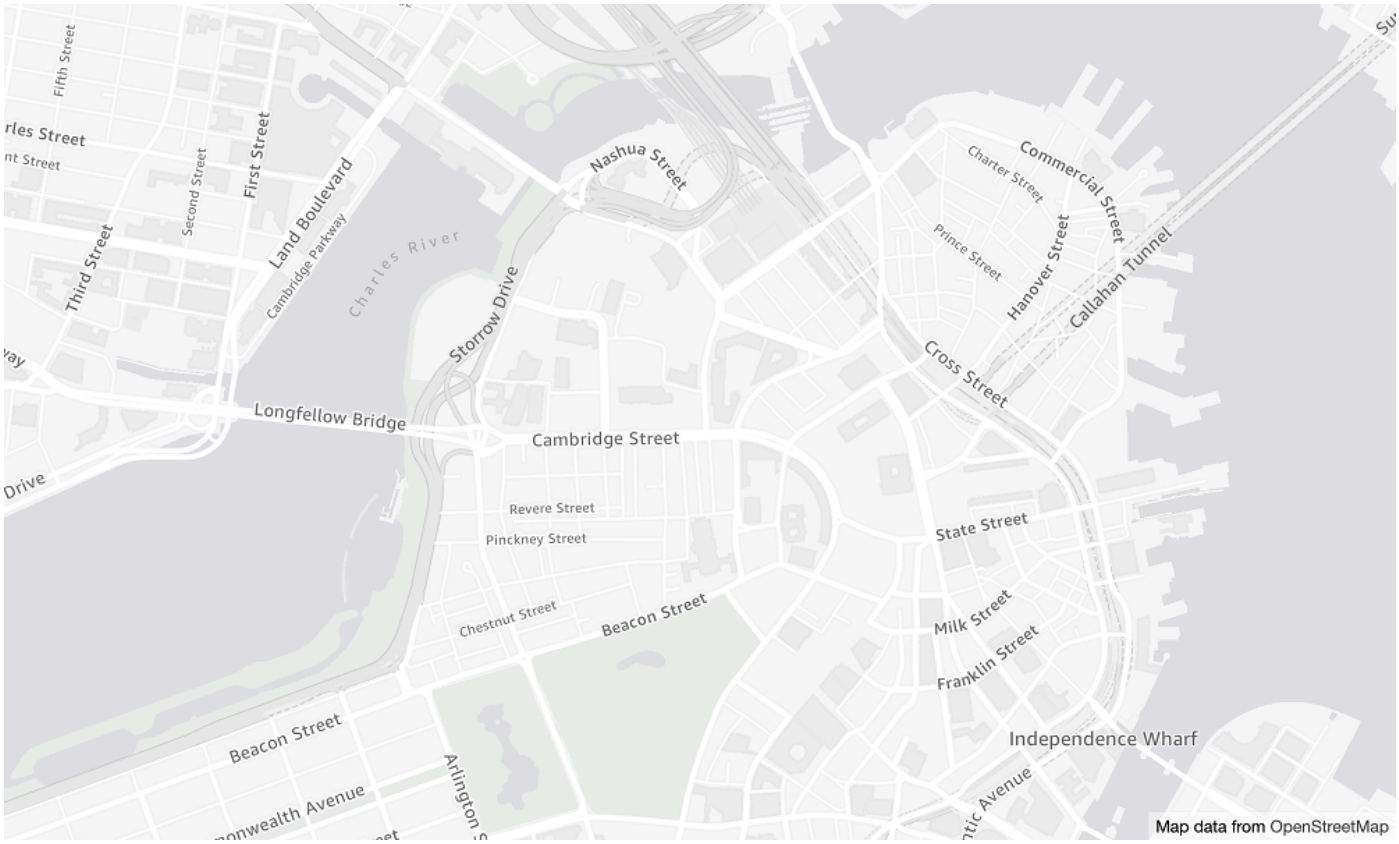
Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Amazon Ember 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 常规、Noto Sans 常规
- Amazon Ember 中等、Noto Sans 中等
- Amazon Ember 常规斜体、Noto Sans 斜体
- Amazon Ember 常规、Noto Sans 常规
- Amazon Ember 常规、Noto Sans 常规、Noto Sans Arabic 常规
- Amazon Ember Consending RC 粗体、Noto Sans 粗体、Noto Sans
- Amazon Ember 粗体、Noto Sans 粗体、Noto Sans Arabic 粗体
- Amazon Ember 常规斜体、Noto Sans 斜体、Noto Sans Arabic 常规
- Amazon Ember Condensed RC 常规、Noto Sans 常规、Noto Sans Arabic Condensed 常规
- Amazon Ember 中等、Noto Sans 中等、Noto Sans Arabic 中等

> **Note**
>
> `VectorOpenDataStandardDark` 使用的字体是组合字体，把 `Amazon Ember` 用于大多数字形，但把 `Noto Sans` 用于 `Amazon Ember` 不支持的字形。

Open Data Visualization Light

开放数据可视化（浅色）



地图样式名称：`VectorOpenDataVisualizationLight`

这是一种以浅色为主题的风格，色彩柔和，功能较少，有助于理解叠加的数据。

此底图基于 OpenStreetMap (OSM) 贡献者编译的OSM日光地图分布。该OSM社区包括超过180万贡献者，他们每天更新超过50万个功能。Amazon Location Service 会定期整合这些编辑内容。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

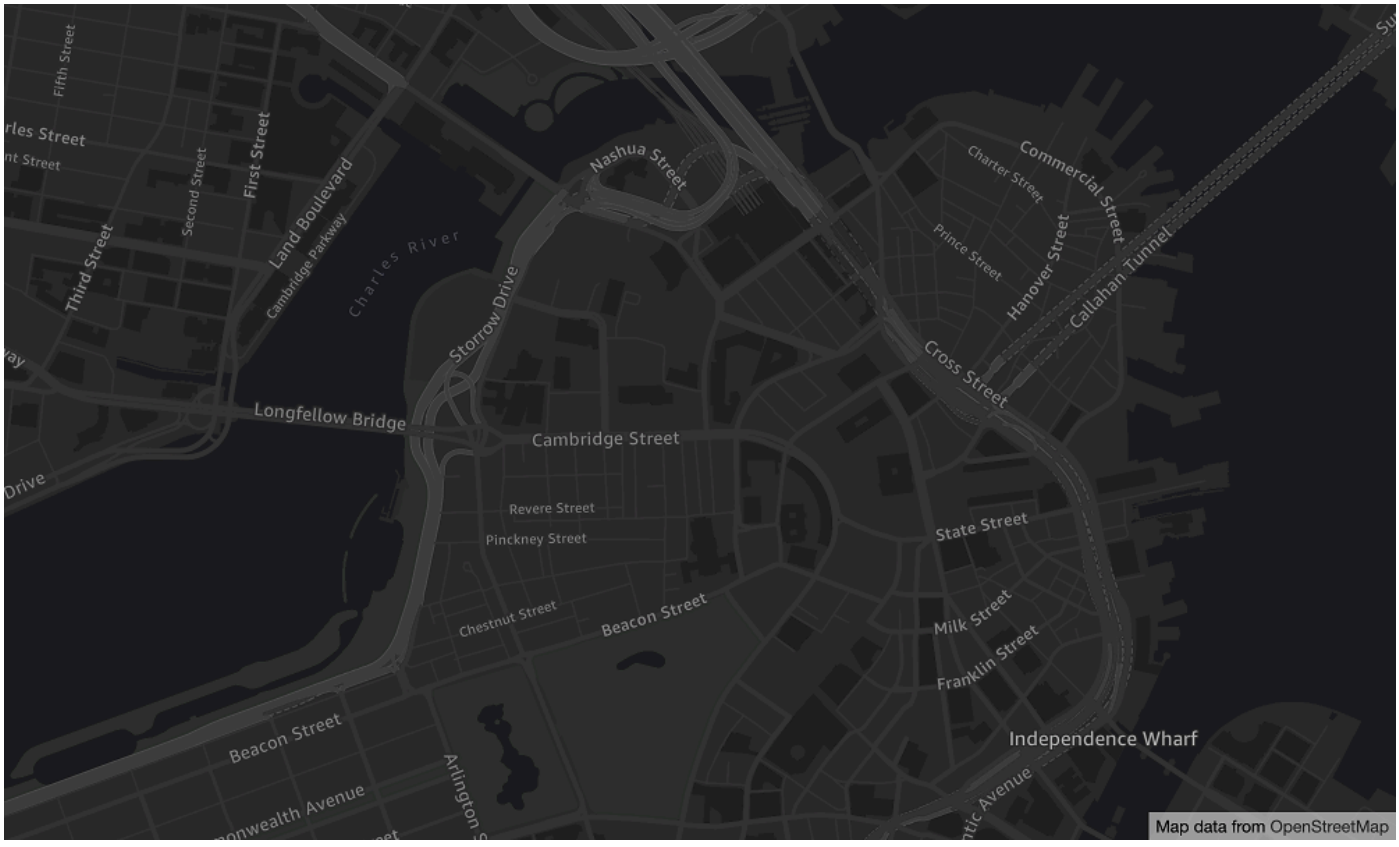
- Amazon Ember 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 常规、Noto Sans 常规
- Amazon Ember 中等、Noto Sans 中等
- Amazon Ember 常规斜体、Noto Sans 斜体
- Amazon Ember 常规、Noto Sans 常规
- Amazon Ember 常规、Noto Sans 常规、Noto Sans Arabic 常规

- Amazon Ember Consending RC 粗体、Noto Sans 粗体、Noto Sans
- Amazon Ember 粗体、Noto Sans 粗体、Noto Sans Arabic 粗体
- Amazon Ember 常规斜体、Noto Sans 斜体、Noto Sans Arabic 常规
- Amazon Ember Condensed RC 常规、Noto Sans 常规、Noto Sans Arabic Condensed 常规
- Amazon Ember 中等、Noto Sans 中等、Noto Sans Arabic 中等

> **Note**
>
> `VectorOpenDataVisualizationLight` 使用的字体是组合字体，把 `Amazon Ember` 用于大多数字形，但把 `Noto Sans` 用于 `Amazon Ember` 不支持的字形。

Open Data Visualization Dark

开放数据可视化（深色）



地图样式名称：`VectorOpenDataVisualizationDark`

这是一种深色主题风格，色彩柔和，功能较少，有助于理解叠加的数据。

此底图基于 OpenStreetMap (OSM) 贡献者编译的OSM日光地图分布。该OSM社区包括超过180万贡献者，他们每天更新超过50万个功能。Amazon Location Service 会定期整合这些编辑内容。

字体

Amazon Location 使用 `GetMapGlyphs` 提供字体。以下是此地图的可用字体堆栈：

- Amazon Ember 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 粗体、Noto Sans 粗体
- Amazon Ember Condensed RC 常规、Noto Sans 常规
- Amazon Ember 中等、Noto Sans 中等
- Amazon Ember 常规斜体、Noto Sans 斜体
- Amazon Ember 常规、Noto Sans 常规
- Amazon Ember 常规、Noto Sans 常规、Noto Sans Arabic 常规
- Amazon Ember Consending RC 粗体、Noto Sans 粗体、Noto Sans
- Amazon Ember 粗体、Noto Sans 粗体、Noto Sans Arabic 粗体
- Amazon Ember 常规斜体、Noto Sans 斜体、Noto Sans Arabic 常规
- Amazon Ember Condensed RC 常规、Noto Sans 常规、Noto Sans Arabic Condensed 常规
- Amazon Ember 中等、Noto Sans 中等、Noto Sans Arabic 中等

> **Note**
>
> `VectorOpenDataVisualizationDark` 使用的字体是组合字体，把 `Amazon Ember` 用于大多数字形，但把 `Noto Sans` 用于 `Amazon Ember` 不支持的字形。

覆盖范围：开放数据

开放数据包括覆盖全球的地图，可使用 Amazon Location Service 地图资源进行渲染。

> **Note**
>
> 开放数据仅适用于 Amazon Location Service 地图资源。您不能将开放数据用作数据提供程序来支持地理编码、反向地理编码和搜索查询，也不能用于支持计算路径的查询。

使用条款和数据属性：开放数据

在使用开放数据之前，请务必遵守所有适用的法律要求，包括适用于开放数据的许可条款和 AWS。

有关 AWS 要求的更多信息，请参阅AWS服务条款。

有关开放数据归因指南的信息，请参阅 OpenStreetMap “版权和许可” 和 “许可 /归因指南”。OpenStreetMap

报告和为开放数据做出贡献时出错

OpenStreetMap (OSM) 和 Natural Earth 是社区驱动的开放数据项目。如果您遇到数据问题，可以报告错误或直接提供修复或建议。

- 要在中报告错误或提供建议OSM，可以在地图上创建注释。这是对地图的评论，可帮助贡献者修复地图。您可以通过OpenStreetMap 网站创建笔记。有关笔记的更多信息，请参阅 OpenStreetMap wiki 中的笔记。
- 有关直接参与的更多信息 OpenStreetMap，包括添加位置和修复错误，请参阅 OpenStreetMap wiki 中的贡献地图数据。
- 要提交对 Natural Earth 数据的更正请求，您可以通过 Natural Earth 网址提交问题。

> **Note**
>
> 更正中的错误 OpenStreetMap 可能很快就会发生，但是，更正可能需要一段时间才能显示在开放数据提供者使用的OSM数据的日光地图分布中。Daylight 地图分布网站提供了有关该过程的更多信息。此外，Amazon Location Service 大约每月更新 Amazon Location Service 中使用的地图数据。

数据提供程序提供的功能

本部分介绍按数据提供程序分类的 Amazon Location Service 中提供的功能。

下表高度概括了这些功能。

数据提供程序	地理覆盖范围	功能覆盖范围	AWS 区域
Esri	全局	地图、地点、路线	提供 Amazon Location 的所有区域。
Grab	东南亚	地图、地点、路线	亚太地区（新加坡）；仅 `ap-southeast-1` 。
HERE	全局	地图、地点、路线	提供 Amazon Location 的所有区域。
Open Data (打开数据)	全局	映射	提供 Amazon Location 的所有区域。

以下选项卡显示每个功能区域内的详细信息。

Map Features

下表按数据提供程序显示了地图要素。有关地图概念的更多信息，请参阅 映射。

数据提供程序	受支持的地图类型	矢量缩放级别	栅格缩放级别
Esri	Vector 栅格（影像） 有关更多信息，请参阅 Esri 地图样式。	0-15	0-23
Grab	Vector （仅限东南亚） 有关更多信息，请参阅 Grab 地图样式。	0-14	none

数据提供程序	受支持的地图类型	矢量缩放级别	栅格缩放级别
HERE	Vector 栅格（影像） 混合 有关更多信息，请参阅 HERE地图样式。	1-17	0-19
Open Data (打开数据)	Vector 有关更多信息，请参阅 开放数据地图样式。	0-15	none

> **Note**
>
> 缩放级别代表每个提供商中定义的最大和最小设置APIs。地图的不同区域可能具有不同的最大值；例如，海洋图块的详细缩放级别可能低于主要城市的区域。
> MapLibre（以及其他地图渲染引擎）允许您设置最小和最大缩放级别，并且还支持数据提供者在某个区域中的缩放级别，因此您不必编写代码来处理这些差异。

Places and Search

下表按数据提供程序显示了地址和搜索功能。有关地点概念的更多信息，请参阅 地点搜索。

数据提供程序	地理编码	反向地理编码	自动完成	GetPlace
Esri	所有功能，除了： PlaceId	所有功能，除了： TimeZone PlaceId	所有功能	所有功能

数据提供程序	地理编码	反向地理编码	自动完成	GetPlace
Grab	所有功能，除了： 单位类型 不支持类别	所有功能	所有功能	所有功能，除了： 单位类型 SubMunicipality
HERE	所有功能，除了： 单位数字 单位类型 相关性 筛选的其他限制	所有功能	所有功能	所有功能，除了： 单位数字 单位类型 SubMunicipality
Open Data (打开数据)	不支持	不支持	不支持	仅支持： SubMunicipality

Route features

下表显示了按数据提供程序划分的路线功能。有关路线概念的更多信息，请参阅 路线。有关路线矩阵限制的更多详细说明，请参阅 对出发地和目的地位置的限制。

数据提供程序	出行模式	计算路线	路线矩阵
Esri	汽车、卡车、步行	出发地和目的地之间的距离必须在 400 千米以内。总行程时间不能超过 400 分钟。 ArrivalTime 不支持。	最多 10 个出发和目的地位置。 韩国不支持。

数据提供程序	出行模式	计算路线	路线矩阵
			所有出发地和目的地之间的距离必须在 400 千米以内。
Grab	汽车、摩托车。 在部分城市步行和骑自行车。	没有距离限制。	多达 350 个出发和目的地位置。
HERE	汽车、卡车、步行	没有距离限制。在围绕出发和目的地位置的圆圈外行驶超过 10 千米的路线将不计算在内。	多达 350 个出发和目的地位置。 有出发点和目的地位置必须位于一个直径为 180 公里的圆内。 支持更长的路线，但有其他限制。
Open Data (打开数据)	不支持	不支持	不支持

数据提供程序的使用条款和数据属性

在使用数据提供程序之前，请务必遵守所有适用的法律要求，包括适用于使用该提供程序的许可条款。

有关 AWS 要求的更多信息，请参阅AWS服务条款。

将数据提供程序与您的 Amazon Location 资源一起用于应用程序或文档时，请务必为您使用的每个数据提供程序提供属性。

有关每个数据提供程序的合规性和属性的更多信息，请参阅以下主题。

- Esri – 使用条款和数据属性：Esri
- Grab – 使用条款和数据属性：Grab
- HERE – 使用条款和数据归因：HERE
- 开放数据 – 使用条款和数据属性：开放数据

Amazon Location 区域和终端节点

Amazon 位置可在以下 AWS 地区使用：

区域

区域名称	区域	端点	协议	
美国东部（俄亥俄州）	us-east-2	geo.us-east-2.amazonaws.com	HTTPS	
美国东部（弗吉尼亚州北部）	us-east-1	geo.us-east-1.amazonaws.com	HTTPS	
美国西部（俄勒冈州）	us-west-2	geo.us-west-2.amazonaws.com	HTTPS	
亚太地区(孟买)	ap-south-1	geo.ap-south-1.amazonaws.com	HTTPS	
亚太地区（新加坡）	ap-southeast-1	geo.ap-southeast-1.amazonaws.com	HTTPS	
亚太地区（悉尼）	ap-southeast-2	geo.ap-southeast-2.amazonaws.com	HTTPS	
亚太地区（东京)	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	

区域名称	区域	端点	协议
欧洲地区（法兰克福）	eu-central-1	geo.eu-central-1.amazonaws.com	HTTPS
欧洲地区（爱尔兰）	eu-west-1	geo.eu-west-1.amazonaws.com	HTTPS
欧洲地区（伦敦）	eu-west-2	geo.eu-west-2.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	geo.eu-north-1.amazonaws.com	HTTPS
南美洲（圣保罗）	sa-east-1	geo.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国西部）	us-gov-west-1	geo.us-gov-west-1.amazonaws.com geo-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS

> **Note**
>
> 有关如何使用此表中终端节点的更多信息，请参阅以下部分。

端点

Amazon Location 区域端点的一般语法如下所示：

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

在此语法中，Amazon Location 使用以下服务代码：

服务	服务代码
Amazon Location 映射	映射
Amazon Location 位数	位数
Amazon Location 地理围栏	地理围栏
Amazon Location 跟踪器	跟踪
Amazon Location 路线	路线

例如，美国东部（弗吉尼亚北部）亚马逊位置地图的区域终端节点将是：https://*maps*.geo。*us-east-1*.amazonaws.com。

API操作端点

Amazon Location Service 控制面板端点的语法如下所示：

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

Amazon Location Service 的控制面板操作是：

服务	终端节点	API操作
Amazon Location 映射	https://cp.maps.geo。*region*.amazonaws.com	CreateMap DeleteMap DescribeMap ListMaps UpdateMap
Amazon Location 位数	https://cp.places.geo。*region*.amazonaws.com	CreatePlaceIndex DeletePlaceIndex DescribePlaceIndex

服务	终端节点	API操作
		ListPlaceIndexes UpdatePlaceIndex
Amazon Location 地理围栏	https://cp.geofencing.geo。*region*.amazonaws.com	CreateGeofenceCollection DeleteGeofenceCollection DescribeGeofenceCollection ListGeofenceCollections UpdateGeofenceCollection
Amazon Location 跟踪器	https://cp.tracking.geo。*region*.amazonaws.com	CreateTracker DeleteTracker DescribeTracker UpdateTracker ListTrackers AssociateTrackerConsumer DisassociateTrackerConsumer ListTrackerConsumers
Amazon Location 路线	https://cp.routes.geo。*region*.amazonaws.com	CreateRouteCalculator DeleteRouteCalculator DescribeRouteCalculator ListRouteCalculators UpdateRouteCalculator

服务	终端节点	API操作
Amazon Location 元数据	https://cp.metadata.geo。*region*.amazonaws.com	CreateKey DeleteKey DescribeKey ListKeys UpdateKey

Amazon Location Service 配额

本主题概述了 Amazon Location Service 的速率限制和配额。

> **Note**
>
> 如果您需要更高的配额，可以使用服务配额控制台请求增加可调配额的配额。申请增加配额时，请选择您需要增加配额的区域，因为大多数配额都是特定于该 AWS 地区的。

Service Quotas 是每个 AWS 账户和每个 AWS 地区可以拥有的最大资源数量。Amazon Location Service 拒绝超过服务配额的其他请求。

速率限制（以... 开头的配额）是每秒的最大请求数，每秒钟任意部分内的突发速率为限制的 80%，为每个API操作定义。Amazon Location Service 会限制超过操作速率限制的请求。

名称	默认值	可调整	描述
API每个账户的关键资源	每个受支持的区域：500 个	否	每个账户可以拥有的最大API密钥资源（活跃或已过期）数量。

名称	默认值	可调整	描述
每个账户的地理围栏集合资源	每个受支持的区域：1,500 个	是	对于每个账户可以创建的地理围栏集合资源的最大数量。
每个地理围栏集合的地理围栏数	每个受支持的区域：5 万个	否	对于每个地理围栏集合可以创建的地理围栏的最大数量。
每个账户的映射资源	每个受支持的区域：40 个	是	对于每个账户可以创建的映射资源的最大数量。
每个账户的地点索引资源	每个受支持的区域：40 个	是	对于每个账户可以创建的地点索引资源的最大数量。
AssociateTrackerConsumer API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 AssociateTrackerConsumer 请求数。额外的请求将被阻止。
BatchDeleteDevicePositionHistory API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 BatchDeleteDevicePositionHistory 请求数。额外的请求将被阻止。
BatchDeleteGeofence API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 BatchDeleteGeofence 请求数。额外的请求将被阻止。
BatchEvaluateGeofences API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 BatchEvaluateGeofences 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
BatchGetDevicePosition API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 BatchGetDevicePosition 请求数。额外的请求将被阻止。
BatchPutGeofence API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 BatchPutGeofence 请求数。额外的请求将被阻止。
BatchUpdateDevicePosition API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 BatchUpdateDevicePosition 请求数。额外的请求将被阻止。
CalculateRoute API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 CalculateRoute 请求数。额外的请求将被阻止。
CalculateRouteMatrix API请求率	每个受支持的区域：每秒 5 个	是	您每秒可以发出的最大 CalculateRouteMatrix 请求数。额外的请求将被阻止。
CreateGeofenceCollection API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 CreateGeofenceCollection 请求数。额外的请求将被阻止。
CreateKey API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 CreateKey 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
CreateMap API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 CreateMap 请求数。额外的请求将被阻止。
CreatePlaceIndex API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 CreatePlaceIndex 请求数。额外的请求将被阻止。
CreateRouteCalculator API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 CreateRouteCalculator 请求数。额外的请求将被阻止。
CreateTracker API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 CreateTracker 请求数。额外的请求将被阻止。
DeleteGeofenceCollection API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DeleteGeofenceCollection 请求数。额外的请求将被阻止。
DeleteKey API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DeleteKey 请求数。额外的请求将被阻止。
DeleteMap API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DeleteMap 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
DeletePlaceIndex API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DeletePlaceIndex 请求数。额外的请求将被阻止。
DeleteRouteCalculator API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DeleteRouteCalculator 请求数。额外的请求将被阻止。
DeleteTracker API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DeleteTracker 请求数。额外的请求将被阻止。
DescribeGeofenceCollection API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DescribeGeofenceCollection 请求数。额外的请求将被阻止。
DescribeKey API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DescribeKey 请求数。额外的请求将被阻止。
DescribeMap API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DescribeMap 请求数。额外的请求将被阻止。
DescribePlaceIndex API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DescribePlaceIndex 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
DescribeRouteCalculator API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DescribeRouteCalculator 请求数。额外的请求将被阻止。
DescribeTracker API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DescribeTracker 请求数。额外的请求将被阻止。
DisassociateTrackerConsumer API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 DisassociateTrackerConsumer 请求数。额外的请求将被阻止。
ForecastGeofenceEvents API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 ForecastGeofenceEvents 请求数。额外的请求将被阻止。
GetDevicePosition API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 GetDevicePosition 请求数。额外的请求将被阻止。
GetDevicePositionHistory API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 GetDevicePositionHistory 请求数。额外的请求将被阻止。
GetGeofence API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 GetGeofence 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
GetMapGlyphs API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 GetMapGlyphs 请求数。额外的请求将被阻止。
GetMapSprites API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 GetMapSprites 请求数。额外的请求将被阻止。
GetMapStyleDescriptor API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 GetMapStyleDescriptor 请求数。额外的请求将被阻止。
GetMapTile API请求率	每个受支持的区域：每秒 500 个	是	您每秒可以发出的最大 GetMapTile 请求数。额外的请求将被阻止。
GetPlace API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 GetPlace 请求数。额外的请求将被阻止。
ListDevicePositions API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 ListDevicePositions 请求数。额外的请求将被阻止。
ListGeofenceCollections API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListGeofenceCollections 请求数。额外的请求将被阻止。
ListGeofences API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 ListGeofences 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
ListKeys API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListKeys 请求数。额外的请求将被阻止。
ListMaps API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListMaps 请求数。额外的请求将被阻止。
ListPlaceIndexes API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListPlaceIndexes 请求数。额外的请求将被阻止。
ListRouteCalculators API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListRouteCalculators 请求数。额外的请求将被阻止。
ListTagsForResource API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListTagsForResource 请求数。额外的请求将被阻止。
ListTrackerConsumers API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListTrackerConsumers 请求数。额外的请求将被阻止。
ListTrackers API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 ListTrackers 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
PutGeofence API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 PutGeofence 请求数。额外的请求将被阻止。
SearchPlaceIndexForPosition API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 SearchPlaceIndexForPosition 请求数。额外的请求将被阻止。
SearchPlaceIndexForSuggestions API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 SearchPlaceIndexForSuggestions 请求数。额外的请求将被阻止。
SearchPlaceIndexForText API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 SearchPlaceIndexForText 请求数。额外的请求将被阻止。
TagResource API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 TagResource 请求数。额外的请求将被阻止。
UntagResource API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 UntagResource 请求数。额外的请求将被阻止。
UpdateGeofenceCollection API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 UpdateGeofenceCollection 请求数。额外的请求将被阻止。

名称	默认值	可调整	描述
UpdateKey API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 UpdateKey 请求数。额外的请求将被阻止。
UpdateMap API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 UpdateMap 请求数。额外的请求将被阻止。
UpdatePlaceIndex API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 UpdatePlaceIndex 请求数。额外的请求将被阻止。
UpdateRouteCalculator API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 UpdateRouteCalculator 请求数。额外的请求将被阻止。
UpdateTracker API请求率	每个受支持的区域：每秒 10 个	是	您每秒可以发出的最大 UpdateTracker 请求数。额外的请求将被阻止。
VerifyDevicePosition API请求率	每个受支持的区域：每秒 50 个	是	您每秒可以发出的最大 VerifyDevicePosition 请求数。额外的请求将被阻止。
每个账户的路线计算器资源	每个受支持的区域：40 个	是	对于每个账户可以创建的路线计算器资源的最大数量。
每个跟踪器的跟踪器使用者人数	每个受支持的区域：5 个	否	可以与跟踪器资源关联的地理围栏集合的最大数量。

名称	默认值	可调整	描述
每个账户的跟踪器资源	每个受支持的区域：500 个	是	对于每个账户可以创建的跟踪器资源的最大数量。

> **Note**
>
> 您可以使用 Cloudwatch 监控您对配额的使用情况。有关更多信息，请参阅 CloudWatch 用于根据配额监控使用情况。

管理您的 Amazon Location Service 配额

Amazon Location Service 与 AWS 服务配额集成，该服务使您能够从中央位置查看和管理配额。有关更多信息，请参阅《服务限额用户指南》中的什么是服务限额？

使用 Service Quotas，可使用轻松查找 Amazon Location Service 配额的值。

AWS Management Console

使用控制台查看 Amazon Location Service 配额

1. 打开 Service Quotas 控制台，网址为https://console.aws.amazon.com/servicequotas/。
2. 在导航窗格中，选择 AWS 服务。
3. 从 AWS 服务列表中，搜索并选择 Amazon Location。

   在服务限额列表中，您可以查看服务限额名称、应用的值（如果该值可用）、 AWS 默认限额以及限额值是否可调整。
4. 要查看有关服务限额的其他信息（如描述），请选择限额名称。
5. （可选）要请求增加配额，请选择要增加的配额，选择 Request quota increase（请求增加配额），输入或选择所需信息，然后选择 Request（请求）。

要使用控制台进一步处理服务配额，请参阅 Service Quotas 用户指南。要请求提高配额，请参阅 Service Quotas 用户指南中的请求增加配额。

AWS CLI

要使用 AWS CLI查看 Amazon Location Service 配额

运行以下命令查看 Amazon Location 原定设置配额。

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

要使用更多地使用服务配额 AWS CLI，请参阅 Ser vice Quotas AWS CLI 命令参考。要请求提高配额，请参阅 AWS CLI 命令参考中的 `request-service-quota-increase` 命令。

以开发者身份开始使用 Amazon Location Service

您可以使用 Amazon Location Service 为跨多种不同外形规格和系统的应用程序提供与地理相关的功能，包括后端 Web 服务、Web 应用程序和移动应用程序。有许多工具可以帮助您构建应用程序，包括开发工具包、库和示例代码。

本部分提供了一些可帮助您开始使用 Amazon Location Location 的信息和链接。特别是，以下主题提供的信息可能对您最有帮助：

- 场景和用例——开发场景列表以及 Amazon Location Service 如何帮助您完成这些场景。
- Amazon Location 开发工具包和工具——开发工具包（SDK）和库，可在使用 Amazon Location 进行编程时为您提供帮助。
- Amazon Location Service API 参考 — 对 AWS 软件开发工具包附带的核心亚马逊定位 API 的引用。
- 代码示例——本部分提供的示例将帮助您入门或向现有应用程序添加功能。
- 快速入门教程——本教程旨在介绍如何创建您的第一个应用程序。本教程有一些版本用于创建 Web 应用程序或基于 Android 的移动应用程序。
- Amazon Location Service 概念——本指南的这一部分描述了 Amazon Location 的基本概念，包括地图、地点搜索、路线、地理围栏和追踪器等章节。
- Amplify——Amplify 是一个完整的解决方案，它封装了使用 AWS Cloud创建网络和移动应用程序所需的许多功能。如果您已经在使用 Amplify，或者选择使用 Amplify，那么它有一个内置使用 Amazon Location Service 的地理库供您使用。要开始使用 Amplify Geo，请参阅此处的文档。

场景和用例

Amazon Location Service 是一项在 AWS Cloud中运行的服务。您可以从自己在云端的 Amazon EC2 实例中调用它，但是许多映射应用程序将在设备上运行，或者在设备和云端的组合上运行。以下仅列出了一些典型场景以及如何开发它们。

- 一款后端应用程序，可帮助您优化车队中司机的路线。

  您可以在中编写一个在 Amazon EC2 上运行的应用程序 AWS Cloud ，该应用程序使用 Amazon Location Service 计算路线矩阵，作为对车队路线优化器的输入。使用 AWS 开发工具包调用 Amazon Location Service。
- 一款 Web 应用程序，可让您的客户查找您的业务地点。

您可以创建一个在 Amazon EC2 实例上运行的网站，包括基于位置的应用程序。使用的AWS JavaScript SDK 开发一个 Web 应用程序，使用地点搜索功能查找位置，并使用在地图上显示结果 MapLibre。使用 Amazon Location 开发工具包可以更轻松地进行定位编程。

- 为现有的 iOS 或 Android 应用程序添加定位功能。

  你可以使用适用于 Swift 的 AWS 软件开发工具包 (iOS) 或 Kotlin (Android) 调用 Amazon Location，为你的应用程序添加地点搜索和地图功能。 MapLibre 用于渲染地图。还有其他适用于其他语言的 AWS 开发工具包。

- 跟踪资产（设备或车辆），并在它们进入或离开您定义的区域时获取更新。

  跟踪设备的应用程序由几个部分组成。

  - 您要跟踪的每台设备都必须创建跟踪器资源才能对其进行跟踪。例如，它必须使用 MQTT 向 Amazon Location Service 发送位置更新。
  - 创建地理围栏以定义您想要获取资产进入和退出事件的区域。
  - 当资产进入或离开地理围栏区域时，您可以使用 Amazon EC2 或 AWS Lambda 来响应您的事件。
  - 您可以在此基础上进行扩展，创建 Web 或设备应用程序，以便在地图上跟踪和显示您的资产位置。

以下部分详细介绍了可用于 Amazon Location Service 各个方面的工具和库。

用于使用 Amazon Location Service 的开发工具包和工具

有几种工具可以帮助您使用 Amazon Location Service。

- AWS SDK — AWS 软件开发套件 (SDK) 提供多种常用编程语言版本，提供了 API、代码示例和文档，可让您更轻松地使用首选语言构建应用程序。这些 AWS 软件开发工具包包括核心的 Amazon Location API 和功能，包括访问地图、地点搜索、路线、地理围栏和追踪器。要详细了解可用于 Amazon Location Service 的不同应用程序和语言的开发工具包，请参阅 按语言划分的开发工具包。
- MapLibre— Amazon Location Service 建议使用MapLibre渲染引擎渲染地图。 MapLibre 是用于在 Web 或移动应用程序中显示地图的引擎。 MapLibre 还具有插件模型，并支持某些语言和平台的用户界面进行搜索和路由。要了解有关使用 MapLibre 及其提供的功能的更多信息，请参阅MapLibre。
- Amazon Location 开发工具包——Amazon Location 开发工具包是一组开源库，可让您更轻松地使用 Amazon Location Service 开发应用程序。这些库提供了支持移动和网络应用程序身份验证、移动应用程序位置跟踪、亚马逊位置数据类型与 GeoJSON 之间的转换以及适用于 S AWS DK v3 的

Amazon Location 客户端托管软件包的功能。要了解有关 Amazon Location 开发工具包的更多信息，请参阅 Amazon Location 开发工具包。

按语言划分的开发工具包

下表按应用程序类型（Web、移动应用程序或后端应用程序）提供了有关语言和框架的 AWS SDK 和 MapLibre 版本的信息。

> **开发工具包版本**
>
> 我们建议您使用项目中使用的最新版本的 SD AWS K 以及任何其他 SDK，并使 SDK 保持最新。S AWS DK 为您提供最新的特性和功能以及安全更新。例如，要查找最新版本的 AWS SDK JavaScript，请参阅软件开发AWS 工具包中的浏览器安装主题以获取 JavaScript文档。

Web frontend

以下 AWS SDK 和 MapLibre 版本可用于 Web 前端应用程序开发。

语言/框架	AWS 开发工具包	渲染框架
完全支持		
JavaScript	https://aws.amazon.com/sdk-for-javascript/	https://maplibre.org/projects/maplibre-gl-js/
ReactJS	https://aws.amazon.com/sdk-for-javascript/	https://github.com/maplibre/maplibre-react-native
TypeScript	https://aws.amazon.com/sdk-for-javascript/	https://maplibre.org/projects/maplibre-gl-js/
部分支持		
Flutter	https://docs.amplify.aws/start/q/integration/flutter/	https://github.com/maplibre/flutter-maplibre-gl MapLibre Flutter 库被认为是实验性的。

语言/框架	AWS 开发工具包	渲染框架
	Amplify 尚未完全支持 Flutter AWS，但通过 Amplify 提供的支持有限。	
Node.js	https://aws.amazon.com/sdk-for-javascript/	不 MapLibre 支持 Node.js。
PHP	https://aws.amazon.com/sdk-for-php/	不 MapLibre 支持 PHP。

Mobile frontend

以下 AWS SDK 和 MapLibre 版本可用于移动前端应用程序开发。

语言/框架	AWS SDK	渲染框架
完全支持		
Java	https://aws.amazon.com/sdk-for-java/	https://maplibre.org/projects/maplibre-native/
Kotlin	https://aws.amazon.com/sdk-for-kotlin/ 适用于安卓的 Amazon Location Service 移动身份验证 SDK：https://github.com/aws-geospatial/ amazon-location-mobile-auth-sdk-android 适用于 Android 的 Amazon Location Service 移动追踪 SDK：https://github.com/aws-geospatial/ amazon-lo	https://maplibre.org/projects/maplibre-native/ 需要自定义绑定，就像基 MapLibre 于 Java 的绑定一样。

语言/框架	AWS SDK	渲染框架
	cation-mobile-tracking-sdk- android	
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/ 适用于 iOS 的 Amazon Location Service 移动身份验证 SDK：https://github.com/aws-geospatial/ amazon-location-mobile-auth-sdk- ios 适用于 iOS 的 Amazon Location Service 移动追踪 SDK：https://github.com/aws-geospatial/ amazon-location-mobile-tracking-sdk- ios	https://maplibre.org/projects/maplibre-native/
部分支持		
Flutter	https://docs.amplify.aws/start/q/integration/flutter/ Amplify 尚未完全支持 Flutter AWS，但通过 Amplify 提供的支持有限。	https://github.com/maplibre/flutter-maplibre-gl MapLibre Flutter 库被认为是实验性的。

Backend application

以下 AWS SDK 可用于后端应用程序开发。 MapLibre 此处未列出，因为后端应用程序通常不需要地图渲染。

Language	AWS 开发工具包
.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/ 适用于 Rust 的 AWS SDK 处于开发者预览版中。

在 Amazon Location 中使用 MapLibre 工具和库

使用 Amazon Location 创建交互式应用程序的重要工具之一是 MapLibre。MapLibre主要是用于在 Web 或移动应用程序中显示地图的渲染引擎。但是，它还包括对插件的支持，并提供处理 Amazon Location 其他方面的功能。以下内容根据您要使用的位置区域描述了您可以使用的工具。

> **Note**
>
> 要使用 Amazon Location Location 的任何方面，请为您要使用的语言安装AWS 开发工具包。

- 映射

  要在应用程序中显示地图，您需要一个地图渲染引擎，该引擎将使用 Amazon Location 提供的数据并绘制到屏幕上。地图渲染引擎还提供平移和缩放地图，或者向地图添加标记、图钉和其他注释的功能。

  Amazon Location Service 建议使用MapLibre渲染引擎渲染地图。 MapLibre GL JS 是用于在中显示地图的引擎 JavaScript，而 MapLibre Native 则提供适用于 iOS 或 Android 的地图。

  MapLibre 还具有用于扩展核心功能的插件生态系统。欲了解更多信息，请访问 https://maplibre.org/maplibre-gl-js-docs /plugins/。
- 地点搜索

  为了简化搜索用户界面的创建，您可以使用网页版MapLibre 地理编码器（Android 应用程序可以使用 Android Plac es 插件）。

  使用 Maplibre 的亚马逊位置地理编码器库来简化在应用程序中使用亚马逊位置信息的过程。`amazon-location-for-maplibre-gl-geocoder` JavaScript
- 路线

  要在地图上显示路线，请使用MapLibre路线。
- 地理围栏和跟踪器

  MapLibre 没有任何用于地理围栏和跟踪的特定渲染或工具，但您可以使用渲染功能和插件在地图上显示地理围栏和被跟踪的设备。

  被跟踪的设备可以使用 MQTT 或手动向 Amazon Location Service 发送更新。可以使用 AWS Lambda 对地理围栏事件进行响应。

许多开源库可用于为 Amazon Location Service 提供其他功能，例如提供空间分析功能的 Turf。

许多库都使用开放标准 GeoJSON 格式的数据。Amazon Location Service 提供了一个库，支持在应用程序中使用 GeoJSON。 JavaScript 有关更多信息，请参阅下一部分：Amazon Location 开发工具包和库。

Amazon 位置 MapLibre 地理编码器插件

Amazon Location MapLibre 地理编码器插件旨在让您在使用库进行地图渲染和地理编码时更轻松地将 Amazon Location 功能整合到 JavaScript 应用程序中。maplibre-gl-geocoder

安装

您可以使用以下命令从 NPM 安装 Amazon Location MapLibre 地理编码器插件以用于模块：

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

您可以使用脚本导入到 HTML 文件中直接在浏览器中使用：

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

与模块一起使用

此代码用于设置具有亚马逊位置地理编码功能的 Maplibre GL JavaScript 地图。它使用通过 Amazon Cognito 身份池进行身份验证来访问亚马逊位置资源。地图使用指定的样式和中心坐标进行渲染，并允许在地图上搜索地点。

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

使用浏览器

此示例使用亚马逊定位客户端发出使用 Amazon Cognito 进行身份验证的请求。

> **Note**
>
> 其中一些示例使用 Amazon 定位客户端。亚马逊定位客户端基于AWS 适用于 JavaScript V3 的软件开发工具包，允许通过 HTML 文件中引用的脚本调用亚马逊位置。

在 HTML 文件中包含以下内容：

```
< 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 JavaScript 文件中包含以下内容：

```
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());
```

下面列出了 Amazon Location MapLibre 地理编码器插件中使用的函数和命令：

- **`buildAmazonLocationMaplibreGeocoder`**

  该类创建了一个实例`AmazonLocationMaplibreGeocder`，该实例是其他所有调用的入口点：

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

- **`getPlacesGeocoder`**

  返回一个可以直接添加到地图的即用型 iControl 对象。

```
const geocoder = getPlacesGeocoder();

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

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

Amazon Location 开发工具包和库

Amazon Location 开发工具包是一组开源库，为开发 Amazon Location 应用程序提供了有用的功能。包括以下功能：

- Amazon Location 客户端 — AWS SDK v3 中的 Amazon Location 对象是捆绑和打包的，便于在网页开发中使用。
- 身份验证 — 在为亚马逊定位服务构建网页、JavaScriptiO S 或 Android 应用程序时，身份验证实用程序简化了身份验证（使用 Amazon Cognito 或 API 密钥）。
- 跟踪 — 移动追踪软件开发工具包适用于 iOS 和安卓系统。此 SDK 使移动应用程序可以更轻松地与 Amazon 位置追踪器进行交互。
- 亚马逊定位 GeoJSON 函数 — GeoJ SON 转换实用程序可以轻松地在行业标准 GeoJSON 格式的数据和亚马逊定位 API 格式之间进行转换。

主题

- 如何开始使用 Amazon Location 开发工具包
- Amazon Location 客户端
- JavaScript 身份验证助手
- GeoJSON 转换帮助程序
- 安卓手机认证 SDK
- iOS 移动身份验证软件开发工具包
- 安卓手机追踪 SDK
- iOS 移动追踪软件开发工具包

如何开始使用 Amazon Location 开发工具包

Amazon Location 开发工具包是一组函数，可以简化在应用程序中使用 Amazon Location Service。您可以将这些函数安装并导入到您的 JavaScript 应用程序中。以下各部分介绍 Amazon Location 客户端、身份验证和 GeoJSON 帮助程序库。

Amazon Location 客户端

在 S AWS DK v3 中，软件开发工具包是按服务分开的。您可以只安装您需要的部件。例如，要安装 Amazon Location 客户端和 Amazon Cognito 的凭证提供程序，请使用以下命令。

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

为了便于在 JavaScript 网络前端应用程序中使用 Amazon Location Servic AWS e，提供了亚马逊位置库和凭证提供商的托管包。要使用捆绑的客户端，请在脚本标签中将其添加到 HTML 中，如下所示：

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

> **Note**
>
> 该软件包保持最新状态并向后兼容，便于使用。使用此脚本标签或 NPM 安装将始终获得最新版本。

JavaScript 身份验证助手

通过您的 JavaScript 应用程序调用 Amazon Location API 时，Amazon Location 身份验证助手可以更轻松地进行身份验证。 JavaScript 在使用 Amazon Cognito 或 API 密钥作为身份验证方法时，此身份验证助手专门为您提供帮助。这是一个开源库，可在以下网址获得：https://github.com/aws-geospatial/ amazon-location-utilities-auth-helper-js。 GitHub

> **Note**
>
> 身份验证帮助程序中的 Amazon Cognito 支持不支持 Amazon Cognito 的联合身份功能。

安装

如果你使用像 webpack 这样的构建系统，或者在 html 中包含带有<script>标签的预建 JavaScript 包，则可以在本地安装中使用这些库。

- 使用以下命令通过 NPM 安装库：

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

- 在 HTML 文件中使用以下命令加载脚本：

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

导入

要在 JavaScript 应用程序中使用特定函数，必须导入该函数。以下代码用于将 withIdentityPoolId 函数导入到您的应用程序中。

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

身份验证函数

Amazon Location 身份验证帮助程序包括以下返回 AuthHelper 对象的函数：

- async withIdentityPoolId(identityPoolId: string): AuthHelper— 此函数返回一个 AuthHelper 对象，该对象已初始化为可与 Amazon Cognito 配合使用
- async withAPIKey(API_KEY: string): AuthHelper— 此函数返回一个 AuthHelper 对象，该对象已初始化为使用 API 密钥。

AuthHelper 目标提供以下函数：

- AuthHelper.getMapAuthenticationOptions()— 该 AuthHelper 对象的此函数返回一个 JavaScript 对象transformRequest，该对象可与 MapLibre JS 中的地图选项一起使用。仅在使用身份池进行初始化时提供。
- AuthHelper.getLocationClientConfig()— 该 AuthHelper 对象的此函数返回一个 JavaScript 对象credentials，其中的可用于初始化 LocationClient。
- AuthHelper.getCredentials()— AuthHelper 对象的此函数返回来自 Amazon Cognito 的内部证书。仅在使用身份池进行初始化时提供。

示例：使用 Amazon Cognito 初始化 MapLibre 地图对象 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 密钥初始化 MapLibre 地图对象（**AuthHelper**本例中不需要）

```
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',
});
```

示例：使用 Amazon Cognito 从 AWS 适用于 JS 的软件开发工具包初始化定位客户端 AuthHelper

此示例使用 AWS 适用于 JavaScript v3 的 SDK。

```
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 密钥从 AWS 适用于 JS 的 SDK 初始化位置客户端 AuthHelper

此示例使用 AWS 适用于 JavaScript v3 的 SDK。

```
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 转换帮助程序

Amazon Location GeoJSON 转换帮助程序提供了将 Amazon Location Service 数据类型与行业标准 GeoJSON 格式相互转换和转换的工具。例如，使用 GeoJSON 在 MapLibre 地图上呈现地理数据。这是一个开源库，可在以下网址获得：https://github.com/aws-geospatial/ amazon-location-utilities-datatypes-js。 GitHub

安装

你可以在本地安装中使用这些库，比如 webpack，也可以在 html 中加入带有<script>标签的预建 JavaScript 捆绑包。

- 使用以下命令通过 NPM 安装库：

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

- 在 HTML 文件中使用以下命令加载脚本：

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

导入

要在 JavaScript 应用程序中使用特定函数，必须导入该函数。以下代码用于将 `placeToFeatureCollection` 函数导入到您的应用程序中。

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

GeoJSON 转换函数

Amazon Location GeoJSON 转换帮助程序包括以下函数：

- `placeToFeatureCollection(place: GetPlaceResponse | searchPlaceIndexForPositionResponse | searchPlaceIndexForTextResponse, keepNull: boolean): Feature`— 此函数将来自地点搜索功能的响应转换为具 FeatureCollection 有 1 个或更多点要素的 GeoJSON。
- `devicePositionToFeatureCollection(devicePositions: GetDevicePositionResponse | BatchGetDevicePositionResponse | GetDevicePositionHistoryResponse | ListDevicePositionsResponse, keepNull: boolean)`— 此函数将来自跟踪器设备位置函数的响应转换为具 FeatureCollection 有 1 个或更多点要素的 GeoJSON。
- `routeToFeatureCollection(legs: CalculateRouteResponse): FeatureCollection`— 此函数将来自计算路径函数的响应转换为 FeatureCollection 具有单个 MultiStringLine 要素的 GeoJSON。路径的每一段都由中的一个 LineString 条目表示 MultiStringLine。
- `geofenceToFeatureCollection(geofences: GetGeofenceResponse | PutGeofenceRequest | BatchPutGeofenceRequest | ListGeofencesResponse): FeatureCollection`— 此函数将地理围栏函数的请求或响应转换为具有 Polygon 功能的 GeoJSON FeatureCollection 。它可以在响应和请求中转换地理围栏，从而允许您在使用或上传地理围栏之前在地图上显示地理围栏。 PutGeofence BatchPutGeofence

此函数会将圆形地理围栏转换为具有近似多边形的特征，但也具有“中心”和“半径”属性，以便在必要时重新创建圆形地理围栏（参见下一个函数）。

- `featureCollectionToGeofences(featureCollection: FeatureCollection): BatchPutGeofenceRequestEntry[]`— 此函数将具 FeatureCollection 有 Polygon 特征的 GeoJSON 转换为 BatchPutGeofenceRequestEntry 对象数组，因此结果可用于创建对的请求。BatchPutGeofence

  如果中的要素具 FeatureCollection 有 “中心” 和 “半径” 属性，则会将其转换为圆形地理围栏请求条目，忽略多边形的几何形状。

示例：将搜索结果转换为点图层 MapLibre

此示例使用 AWS 适用于 JavaScript v3 的 SDK。

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

安卓手机认证 SDK

这些实用程序可帮助您在通过安卓应用程序调用 Amazon Location Service API 时进行身份验证。这在使用 Amazon Cognito 或 API 密钥作为身份验证方法时特别有用。

安卓移动身份验证 SDK 可在 github 上找到：适用于安卓的 Amazon Location Service 移动身份验证 SDK。此外，M AWS aven 存储库中还提供了移动身份验证 AWS SDK 和 SDK。

安装

要使用移动身份验证 SDK，请在 Android Studio 中将以下导入语句添加到您的`build.gradle`文件中。

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

身份验证函数

身份验证助手 SDK 具有以下功能：

- `authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider`：此函数返回`LocationCredentialsProvider`初始化后的 API 密钥。
- `authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider`：此函数返回`LocationCredentialsProvider`初始化后的，以便与 Amazon Cognito 身份池配合使用。

使用量

要在代码中使用 SDK，请导入以下类：

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

在创建身份验证帮助程序和位置客户端提供程序实例时，您有两个选项。你可以使用 Amazon Location API 密钥或 Amazon Cognito 创建实例。

- 要使用 Amazon Location API 密钥创建身份验证帮助程序实例，请按如下方式声明帮助程序类：

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

- 要使用 Amazon Cognito 创建身份验证帮助程序实例，请按如下方式声明该帮助程序类：

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

您可以使用位置凭证提供程序创建 Amazon Location 客户端实例，然后调用亚马逊定位服务。以下示例搜索靠近指定纬度和经度的地方。

```
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 移动身份验证软件开发工具包

这些实用程序可帮助您在通过 iOS 应用程序调用 Amazon Location Service API 时进行身份验证。这在使用 Amazon Cognito 或 API 密钥作为身份验证方法时特别有用。

iOS 移动身份验证 SDK 可在 github 上找到：适用于 iO S 的 Amazon Location Service 移动身份验证 SDK。

安装

在 Xcode 项目中安装软件开发工具包：

1. 转到 “文件”，然后在 XCode 项目中选择 “添加 Package 依赖关系”。
2. 在搜索栏中键入包裹网址：https://github.com/aws-geospatial/ amazon-location-mobile-auth-sdk-ios/，然后按回车键。
3. 选择`amazon-location-mobile-auth-sdk-ios`包裹，然后按 Add Packag e。
4. 选择`AmazonLocationiOSAuthSDK`套餐产品并按 Add Packag e。

身份验证函数

身份验证助手 SDK 具有以下功能：

- `authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider`：此函数返回`LocationCredentialsProvider`初始化后的 API 密钥。
- `authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider`：此函数返回`LocationCredentialsProvider`初始化后的，以便与 Amazon Cognito 身份池配合使用。

使用量

要使用移动身份验证 SDK，请在您的活动中添加以下语句：

```
import AmazonLocationiOSAuthSDK
import AWSLocationXCF
```

在创建身份验证帮助程序和位置客户端提供程序实例时，您有两个选项。你可以使用 Amazon Location API 密钥或 Amazon Cognito 创建实例。

- 要使用 Amazon Location API 密钥创建身份验证帮助程序实例，请按如下方式声明帮助程序类：

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

- 要使用 Amazon Cognito 创建身份验证帮助程序实例，请按如下方式声明该帮助程序类：

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

您可以使用位置凭证提供程序创建 Amazon Location 客户端实例，然后调用亚马逊定位服务。以下示例搜索靠近指定纬度和经度的地方。

```
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)
```

安卓手机追踪 SDK

Amazon Location 移动追踪 SDK 提供的实用程序可帮助轻松进行身份验证、捕获设备位置以及向亚马逊位置追踪器发送位置更新。SDK 支持使用可配置的更新间隔对位置更新进行本地筛选。这可以降低数据成本，并优化您的 Android 应用程序的间歇性连接。

Android 追踪 SDK 可在以下网址获得 GitHub：适用于安卓的亚马逊定位移动追踪 SDK。此外，M AWS aven 存储库中还提供了移动身份验证 AWS SDK 和 SDK。安卓追踪 SDK 专为与通用 AWS SDK 配合使用而设计。

本节涵盖亚马逊定位移动追踪 Android SDK 的以下主题：

主题

- 安装
- 使用量
- 筛选条件
- 安卓移动 SDK 追踪功能
- 示例

安装

要安装 SDK，请在 Android Studio 中 build.gradle 文件的依赖项部分添加以下几行：

```
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")
```

使用量

此过程向您展示如何使用 SDK 进行身份验证和创建`LocationTracker`对象：

> **Note**
>
> 此过程假设您已导入本安装节中提到的库。

1. 在你的代码中导入以下类：

```
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. 接下来创建一个AuthHelper，因为LocationCredentialsProvider参数是创建LocationTracker对象所必需的：

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

3. 现在，使用LocationCredentialsProvider和LocationTrackerConfig来创建一个LocationTracker对象：

```
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,
)
```

筛选条件

Amazon Location 移动追踪 Android SDK 有三个内置的位置过滤器。

- `TimeLocationFilter`：根据定义的时间间隔筛选当前要上传的位置。
- `DistanceLocationFilter`：根据指定的距离阈值过滤位置更新。
- `AccuracyLocationFilter`：通过将自上次更新以来的移动距离与当前位置的精度进行比较来筛选位置更新。

此示例在创建时`LocationTracker`在中添加过滤器：

```
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,
)
```

此示例使用以下命令在运行时启用和禁用过滤器：`LocationTracker`

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

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

安卓移动 SDK 追踪功能

适用于 Android 的 Amazon Location 移动追踪 SDK 包括以下功能：

- 班级:`LocationTracker`

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

这是一个用于创建LocationTracker对象的初始化函数。它需要的实例 LocationCredentialsProvidertrackerName，也可以需要一个实例LocationTrackingConfig。如果未提供配置，则将使用默认值对其进行初始化。

- 班级:LocationTracker

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

  启动访问用户位置并将其发送到 Amazon 位置追踪器的过程。

- 班级:LocationTracker

  ```
  isTrackingInForeground()
  ```

  检查位置追踪当前是否正在进行中。

- 班级:LocationTracker

  ```
  stop()
  ```

  停止跟踪用户位置的过程。

- 班级:LocationTracker

  ```
  startTracking()
  ```

  启动访问用户位置并将其发送到 AWS 跟踪器的过程。

- 班级:LocationTracker

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

  启动访问用户位置的过程，并在应用程序处于后台时将其发送到 AWS 跟踪器。BackgroundTrackingMode 有以下选项：

  - ACTIVE_TRACKING：此选项会主动跟踪用户的位置更新。
  - BATTERY_SAVER_TRACKING：此选项每 15 分钟跟踪一次用户的位置更新。

- 班级:LocationTracker

```
stopBackgroundService()
```

当应用程序处于后台时，停止访问用户位置并将其发送到 AWS 跟踪器的过程。

- 班级:`LocationTracker`

```
getTrackerDeviceLocation()
```

从 Amazon 定位服务中检索设备位置。

- 班级:`LocationTracker`

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

从融合的位置信息提供商客户端检索当前设备位置，并将其上传到 Amazon 位置追踪器。

- 班级:`LocationTracker`

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

根据配置的位置筛选条件进行筛选后，将设备位置上传到 Amazon 定位服务。

- 班级:`LocationTracker`

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

启用特定的位置过滤器。

- 班级:`LocationTracker`

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

禁用特定位置过滤器。

- 班级:`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())

这将LocationTrackerConfig使用用户定义的参数值初始化。如果未提供参数值，则会将其设置为默认值。

- 班级:LocationFilter

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

LocationFilter是用户可以为其自定义过滤器实现实现的协议。你需要实现这个shouldUpload函数来比较以前和现在的位置，并返回是否应该上传当前的位置。

示例

以下代码示例显示了移动跟踪 SDK 的功能。

此示例使用在后台LocationTracker开始和停止跟踪：

```
// 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 移动追踪软件开发工具包

Amazon Location 移动追踪 SDK 提供的实用程序可帮助轻松进行身份验证、捕获设备位置以及向亚马逊位置追踪器发送位置更新。SDK 支持使用可配置的更新间隔对位置更新进行本地筛选。这可以降低数据成本，并优化 iOS 应用程序的间歇性连接。

iOS 追踪软件开发工具包可在以下网址获得 GitHub：适用于 iOS 的亚马逊定位移动追踪软件开发工具包。

本节涵盖亚马逊定位移动追踪 iOS SDK 的以下主题：

主题

- 安装
- 使用量
- 筛选条件
- iOS 移动 SDK 跟踪功能
- 示例

安装

使用以下步骤安装适用于 iOS 的移动追踪 SDK：

1. 在 Xcode 项目中，转到 “文件”，然后选择 “添加 Package 依赖关系”。
2. 在搜索栏中键入以下 URL：https://github.com/aws-geospatial/ amazon-location-mobile-tracking-sdk-ios/，然后按回车键。
3. 选择`amazon-location-mobile-tracking-sdk-ios`包裹，然后单击 Add Packag e。
4. 选择`AmazonLocationiOSTrackingSDK`套餐产品并点击 Add Packag e。

使用量

以下过程向您展示如何使用来自 Cognito 的凭据创建身份验证助手。

1. 安装库后，你需要在`info.plist`文件中添加一个或两个描述：

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

2. 接下来， AuthHelper 在你的课堂中导入：

```
import AmazonLocationiOSAuthSDKimport AmazonLocationiOSTrackingSDK
```

3. 然后，您将使用来自 Amazon Cognito 的凭证创建身份验证助手，从而创建一个`AuthHelper`对象并将其与 AWS 软件开发工具包一起使用。

```
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)
```

筛选条件

Amazon Location 移动追踪 iOS SDK 有三个内置的位置过滤器。

- `TimeLocationFilter`：根据定义的时间间隔筛选当前要上传的位置。
- `DistanceLocationFilter`：根据指定的距离阈值过滤位置更新。
- `AccuracyLocationFilter`：通过将自上次更新以来的移动距离与当前位置的精度进行比较来筛选位置更新。

此示例在创建时`LocationTracker`在中添加过滤器：

```
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,
)
```

此示例使用以下命令在运行时启用和禁用过滤器：LocationTracker

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

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

iOS 移动 SDK 跟踪功能

适用于 iOS 的 Amazon Location 移动追踪 SDK 包括以下功能：

- 班级:LocationTracker

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

  这是一个用于创建LocationTracker对象的初始化函数。它需要的实例 LocationCredentialsProvidertrackerName，也可以需要一个实例LocationTrackingConfig。如果未提供配置，则将使用默认值对其进行初始化。
- 班级:LocationTracker

  setTrackerConfig(config: LocationTrackerConfig)

  这会将 Tracker 的配置设置为在位置跟踪器初始化后的任何时候生效
- 班级:LocationTracker

  getTrackerConfig()

  这将获取位置跟踪配置，以便在您的应用程序中使用或修改。

  退货：LocationTrackerConfig

- 班级:LocationTracker

  getDeviceId()

  获取位置追踪器生成的设备 ID。

  退货：String?
- 班级:LocationTracker

  startTracking()

  启动访问用户位置并将其发送到 AWS 跟踪器的过程。
- 班级:LocationTracker

  resumeTracking()

  恢复访问用户位置并将其发送到 AWS 跟踪器的过程。
- 班级:LocationTracker

  stopTracking()

  停止跟踪用户位置的过程。
- 班级:LocationTracker

  startBackgroundTracking(mode: BackgroundTrackingMode)

  启动访问用户位置的过程，并在应用程序处于后台时将其发送到 AWS 跟踪器。
  BackgroundTrackingMode有以下选项：
  - Active:此选项不会自动暂停位置更新。
  - BatterySaving:此选项会自动暂停位置更新
  - None:此选项总体上会禁用后台位置更新
- 班级:LocationTracker

  resumeBackgroundTracking(mode: BackgroundTrackingMode)

  当应用程序处于后台时，恢复访问用户位置并将其发送到 AWS 跟踪器的过程。
- 班级:LocationTracker

  stopBackgroundTracking()

当应用程序处于后台时，停止访问用户位置并将其发送到 AWS 跟踪器的过程。

- 班级:LocationTracker

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

  检索用户设备在开始和结束日期和时间之间上传的跟踪位置。

  退货：Void

- 班级:LocationTrackerConfig

  init()

  这将 LocationTrackerConfig 使用默认值初始化。

- 班级: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)

  这将LocationTrackerConfig使用用户定义的参数值初始化。如果未提供参数值，则会将其设置为默认值。

- 班级:LocationFilter

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

  LocationFilter是用户可以为其自定义过滤器实现实现的协议。用户需要实现shouldUpload函数来比较以前和当前的位置，并返回是否应上传当前位置。

示例

本节详细介绍了使用适用于 iOS 的 Amazon 定位移动追踪 SDK 的示例。

> **Note**
>
> 确保在info.plist文件中设置了必要的权限。这些权限与本使用量节中列出的权限相同。

以下示例演示了跟踪设备位置和检索追踪位置的功能：

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

开始追踪地点：

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

继续追踪地点：

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

停止追踪地点：

```
locationTracker.stopTracking()
```

开始后台跟踪：

```
do {

 locationTracker.startBackgroundTracking(mode: .Active) // .Active, .BatterySaving, .None
    }
```

```
catch TrackingLocationError.permissionDenied {
   // Handle permissionDenied by showing the alert message or opening the app settings
   }
```

恢复背景跟踪：

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

要停止背景跟踪，请执行以下操作：

```
locationTracker.stopBackgroundTracking()
```

从追踪器中检索设备的追踪位置：

```
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 Location API

Amazon Location Service 提供 API 操作，可通过编程方式访问定位功能。这包括地图、地点、路线、跟踪器、地理围栏和标记资源的 API。有关可用的 API 操作的信息，请参阅 Amazon Location Service API 参考。

您可以在本指南的 代码示例 章节中找到示例。

将 Amazon 定位与 S AWS DK 配合使用

AWS 软件开发套件 (SDK) 可用于许多流行的编程语言。每个 SDK 都提供一个 API、代码示例和文档，便于开发人员使用自己的首选语言构建 AWS 应用程序。

有关按语言划分的可用于 Amazon Location Service 的开发工具包的更多信息，请参阅本指南 按语言划分的开发工具包 中的。

> **开发工具包版本**
>
> 我们建议您使用项目中使用的最新版本的 SD AWS K 以及任何其他 SDK，并使 SDK 保持最新。S AWS DK 为您提供最新的特性和功能以及安全更新。例如，要查找最新版本的 AWS SDK JavaScript，请参阅软件开发AWS 工具包中的浏览器安装主题以获取 JavaScript文档。

Amazon Location API 错误消息更新

从 2023 年 8 月 1 日起，Amazon Location 团队将更改 API 错误消息，如下表所述。错误代码不会更改。如果您的应用程序依赖于确切的错误消息字符串，则必须使用新字符串更新应用程序。如需有关疑问或问题的帮助，请联系 AWS Support。

主题

- 位数
- 映射
- 跟踪器
- 路线
- 元数据
- 地理围栏

位数

位数

错误代码	例外	旧的错误消息。	新的错误消息
500	InternalServerException	Internal Server Exception	Internal server error. Try again later.
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.

错误代码	例外	旧的错误消息。	新的错误消息
400	ValidationException	BiasPosition[0] must be between -180 and 180	'BiasPosition[0]' must be between -180 and 180.
400	ValidationException	BiasPosition[1] must be between -90 and 90	'BiasPosition[1]' must be between -90 and 90.
400	ValidationException	FilterBBox must have exactly 4 entries	'FilterBBox' must have exactly 4 entries.
400	ValidationException	FilterBBox[0] must be between -180 and 180	'FilterBBox[0]' must be between -180 and 180.
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.

错误代码	例外	旧的错误消息。	新的错误消息
400	ValidationException	Position[1] must be between -90 and 90	'Position[1]' must be between -90 and 90.
400	ValidationException	Language is not a valid BCP 47 language tag	'Language' must comply with the BCP 47 Language Tag standard, but was set to <GivenValue>. For more information, see https://wikipedia.org/wiki/IETF_language_tag.
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.

错误代码	例外	旧的错误消息。	新的错误消息
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>.

映射

映射

错误代码	例外	旧的错误消息。	新的错误消息
500	InternalServerException	Internal Server Exception unable to find style template Error fetching style	Internal server error. Try again later.

错误代码	例外	旧的错误消息。	新的错误消息
		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>.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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>.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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>.

跟踪器

跟踪器

错误代码	例外	旧的错误消息。	新的错误消息
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>.

错误代码	例外	旧的错误消息。	新的错误消息
		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.

错误代码	例外	旧的错误消息。	新的错误消息
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'.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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>.

路线

路线

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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>.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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>.

元数据

元数据

错误代码	例外	旧的错误消息。	新的错误消息
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>.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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>.

地理围栏

地理围栏

地理围栏

错误代码	例外	旧的错误消息。	新的错误消息
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>.

错误代码	例外	旧的错误消息。	新的错误消息
		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'

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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.

错误代码	例外	旧的错误消息。	新的错误消息
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 Location Service 的代码示例和教程

本主题显示了一系列代码示例、教程和博客文章，可帮助您了解 Amazon Location Service。每个代码示例都包含对其工作原理的描述。

您可以在AWS 地理空间 GitHub页面、Amazon Location 的AWS 示例 GitHub 页面和AWS 博客网站上找到更多示例。

> **Note**
>
> 了解 AWS 地理空间 GitHub 页面和 AWS 示例 GitHub 页面之间的区别是件好事。

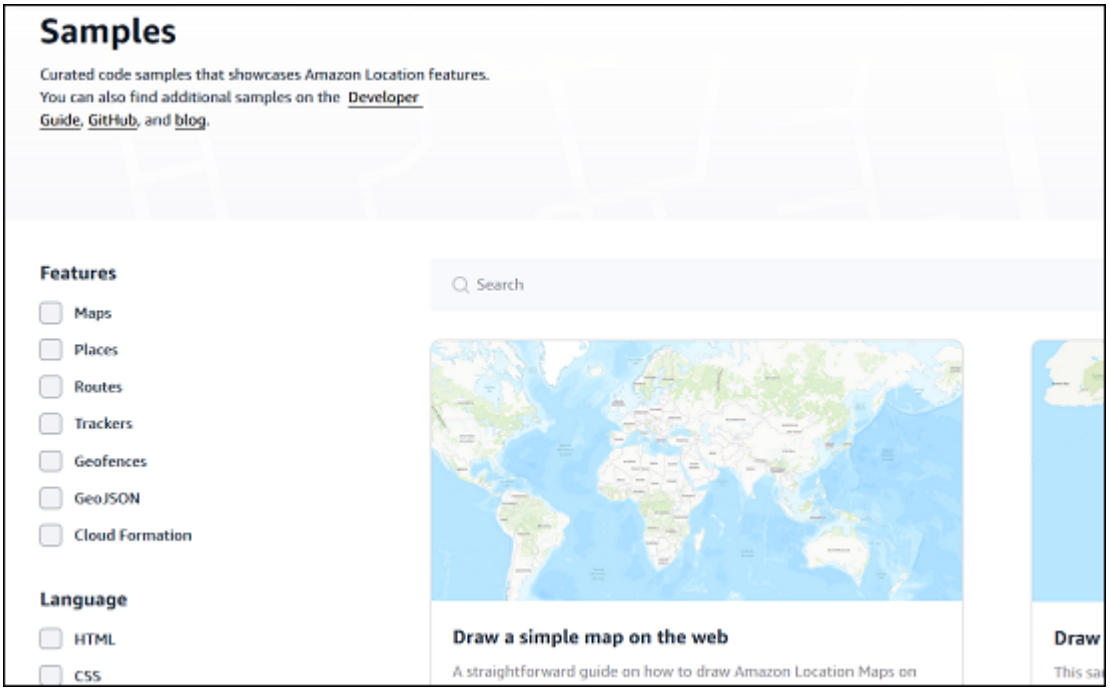
- 地理空间 GitHub-AWS 地理空间 GitHub 页面包含由 Amazon Location Service 团队创建和维护的示例。
- 示例 GitHub — Amazon Location 的示例 GitHub 页面包含为亚马逊位置创建的样本，但可能会被积极维护，也可能不会被积极维护。AWS

在使用其他示例之前，快速入门教程是一个不错的起点，因为它展示了如何完成对大多数示例都有用的先决条件。

主题

- Amazon Location 演示网站
- 教程：快速入门
- 教程：数据库扩充
- 示例：探索应用程序
- 示例：设置地图样式
- 示例：绘制标记
- 示例：绘制聚集点
- 示例：绘制多边形
- 示例：更改地图语言
- 博客：预计送达时间通知
- 示例：直播位置更新
- 示例：地理围栏和跟踪移动应用程序

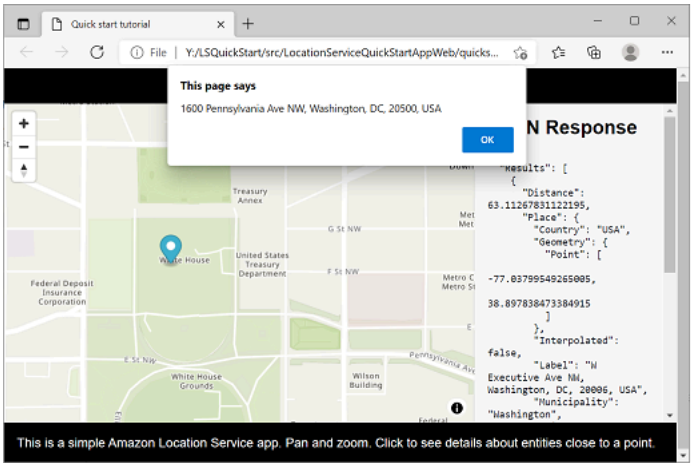
Amazon Location 演示网站

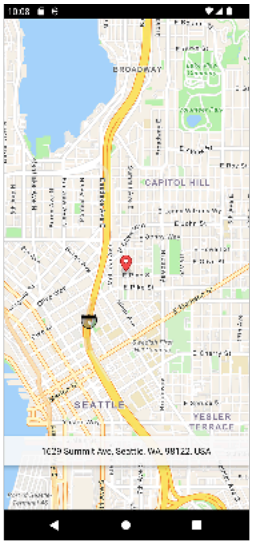


你可以在 Amazon Location 演示网站上观看带有 Amazon Location Service 源代码的演示。该网站包括托管的 Web 演示，以及适用于 Android 的演示应用程序。

您还可以在网站的示例页面中找到各种各样的示例，可按功能、语言和平台进行筛选。

教程：快速入门

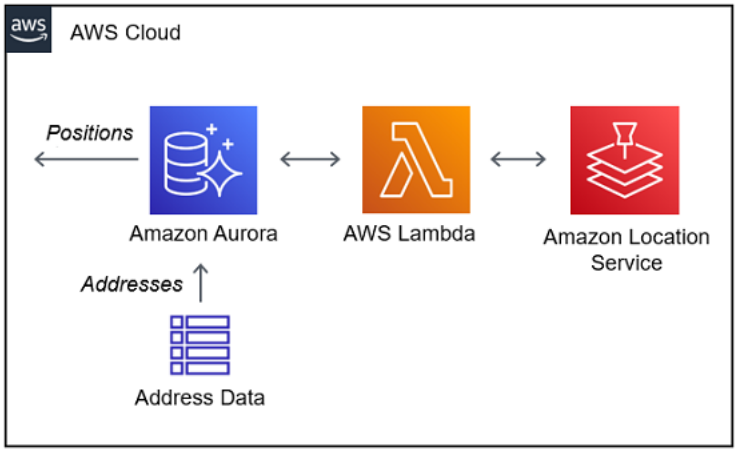




有适用于网页、iOS 和安卓设备的快速入门教程。对于每个平台，本教程将向您展示如何向应用程序添加交互式地图，以及如何从您的应用程序调用 Amazon Location Service API。本教程适用于 JavaScript 静态网页，Kotlin 适用于安卓手机应用程序，Swift 适用于iOS应用程序。

- JavaScript 有关静态网页文档链接：创建 Web 应用程序
- 安卓版 Kotlin 应用程序文档链接：Amazon Location Service 快速入门
- iOS 应用程序的 Swift 文档链接：创建 iOS 应用程序

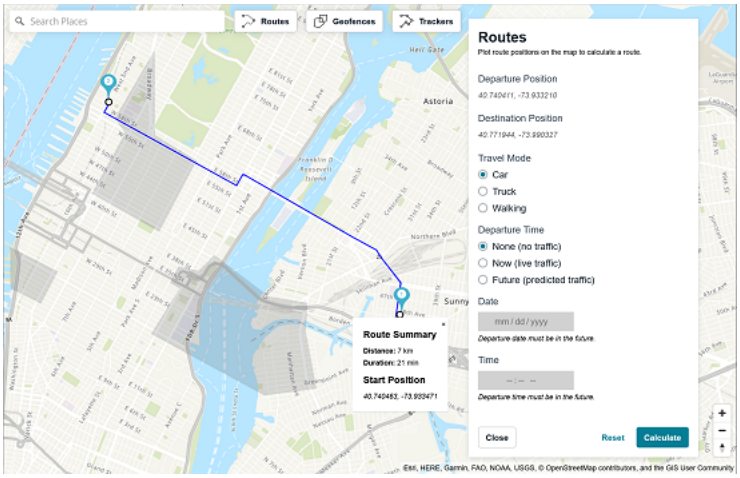
教程：数据库扩充



本教程向您展示如何使用 Amazon Location Service（从中调用） AWS Lambda 来标准化地址，并向 Amazon Aurora 数据库中的记录添加纬度和经度。使用亚马逊 Aurora 和 AWS Lambda.

文档链接：Amazon Location Service 的 Amazon Aurora PostgreSQL 用户定义函数

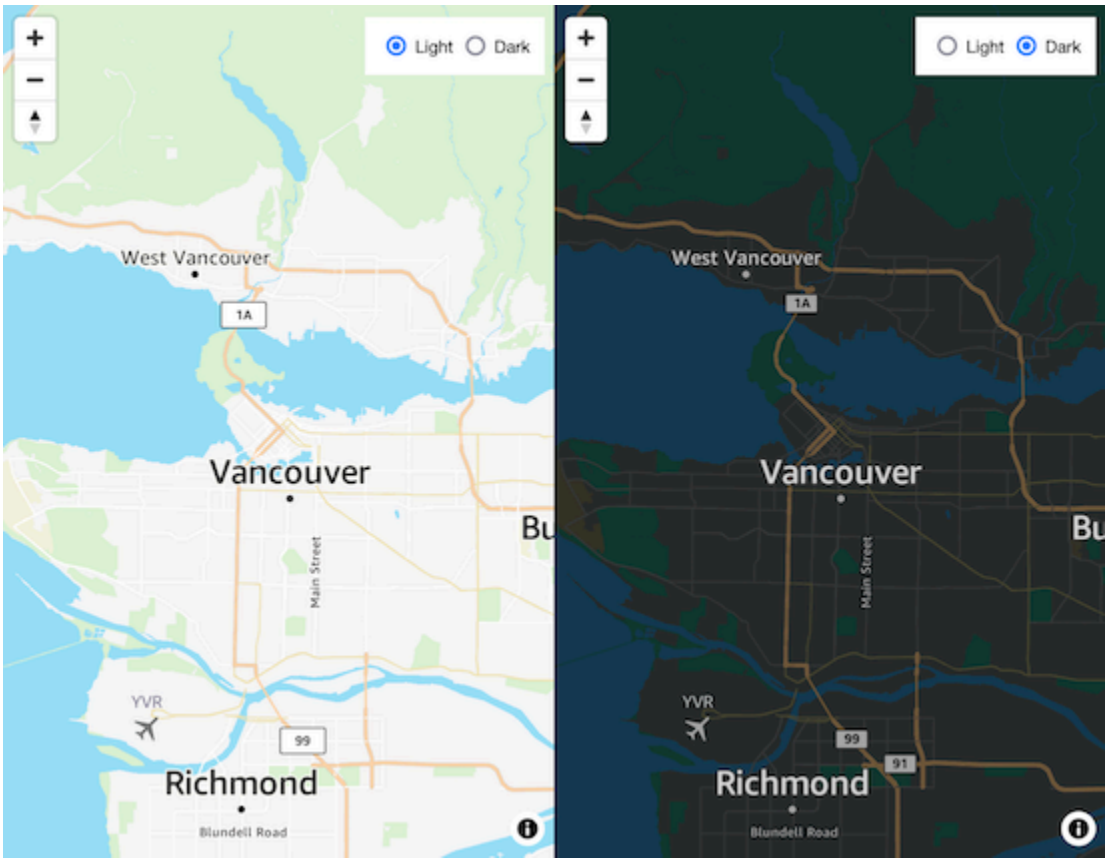
示例：探索应用程序



了解 Amazon Location Service 功能的最佳方法之一是使用 Amazon Location 控制台中的浏览功能。这个完整的 Web 应用程序示例模仿了控制台中的地图、地点、路线、地理围栏和跟踪器功能，向您展示了如何在自己的应用程序中重新创建这些功能。使用 Amplify、React 和。 JavaScript

示例 GitHub 链接：浏览示例应用程序

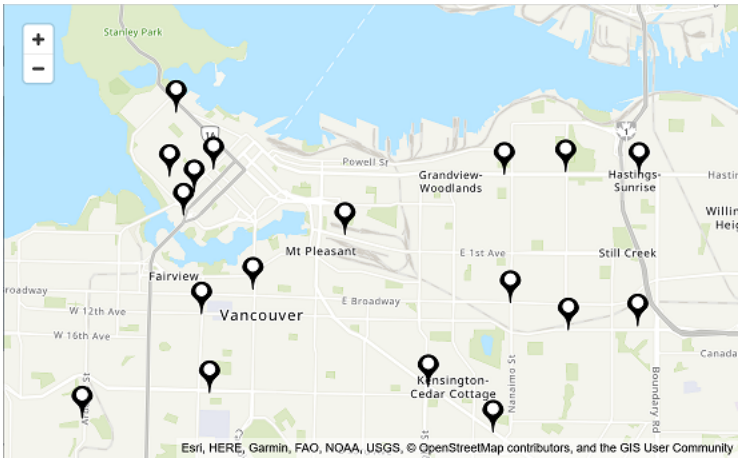
示例：设置地图样式



此代码示例说明如何使用 MapLibre 中的在卫星地图和矢量路线图之间切换 JavaScript。使用 MapLibre Amazon 位置认证助手和 JavaScript。

地理空间 GitHub 链接：具有样式切换功能的交互式地图

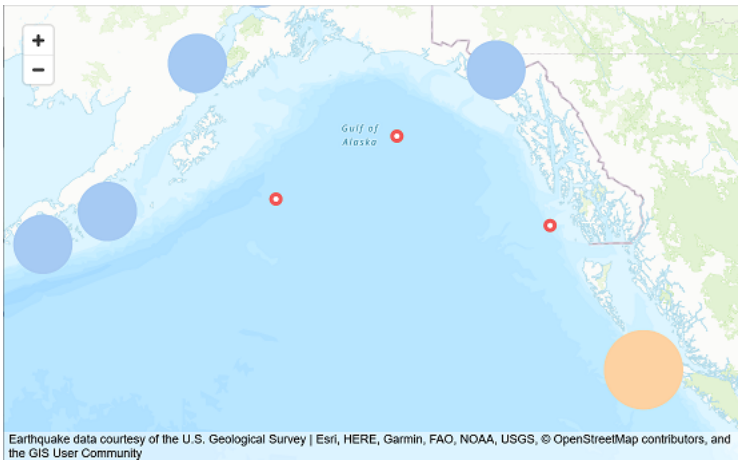
示例：绘制标记



此代码示例显示了加拿大不列颠哥伦比亚省温哥华市的 Amazon Locker 位置。它显示了如何在点位置绘制标记。使用 Node.js MapLibre、React、亚马逊位置认证助手和 JavaScript。

地理空间 GitHub 链接：带有点标记的交互式地图

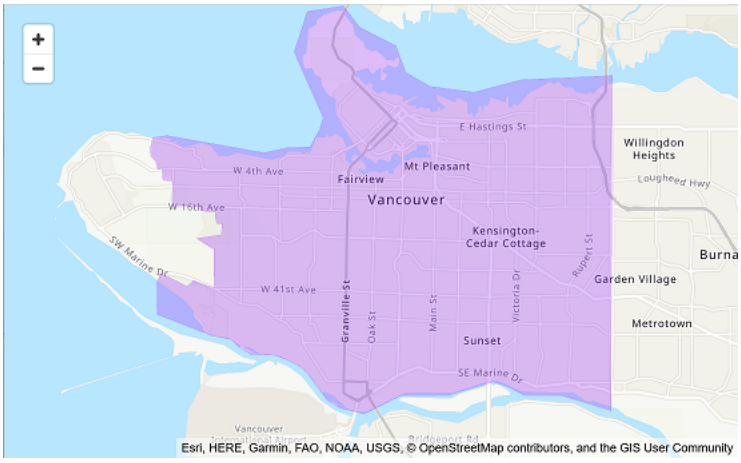
示例：绘制聚集点



此代码示例使用美国地质调查局的地震数据，演示如何绘制在地图上聚集在一起的点。使用 MapLibre、Node.js、React、Amplify 和。 JavaScript

示例 GitHub 链接：包含点聚类的交互式地图

示例：绘制多边形



此代码示例说明如何在地图上绘制多边形。使用 Node.js MapLibre、React、亚马逊位置认证助手和 JavaScript。

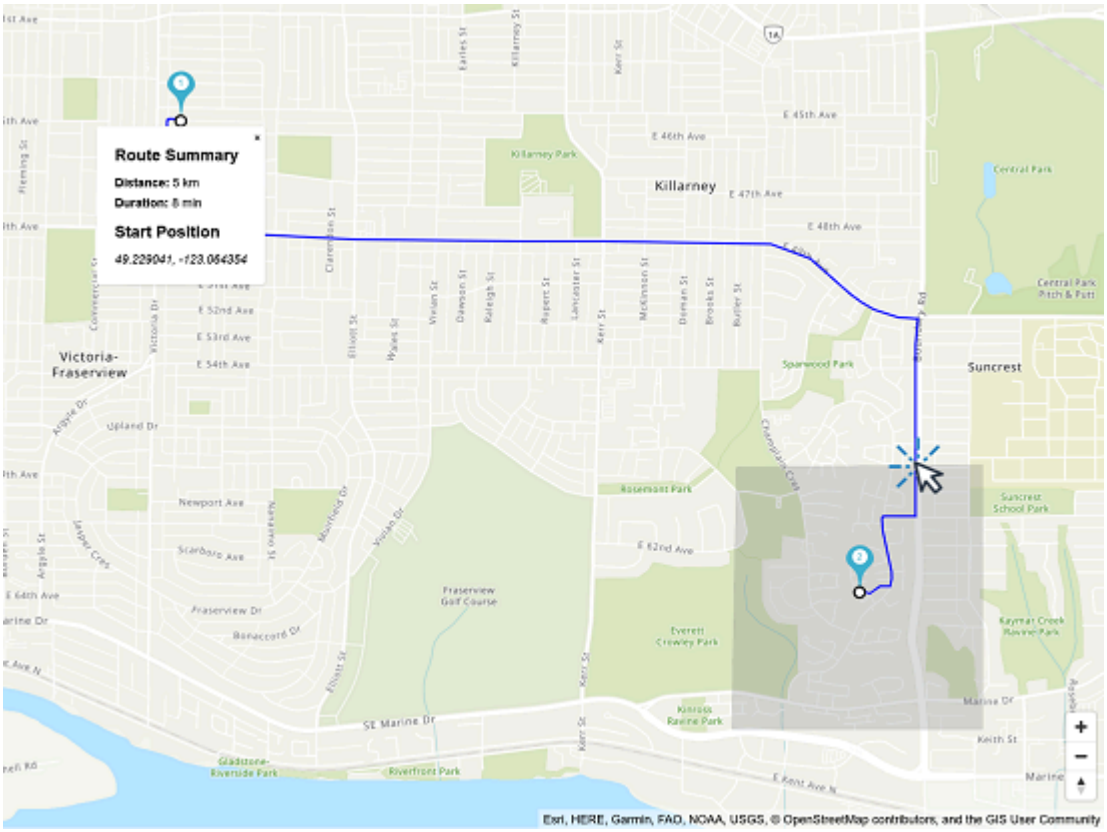
地理空间 GitHub 链接：包含多边形的交互式地图

示例：更改地图语言

此代码示例说明如何更改 Amazon Location 中地图的显示语言。使用 Amplify、React 和。 MapLibre

示例 GitHub 链接：更改地图语言示例

博客：预计送达时间通知



这篇博文展示了通知买家预计送达时间的不同方式。它解释了使用路线来显示预计的行驶时间，然后使用跟踪器和地理围栏来通知司机何时靠近客户。使用 Amplify、React、亚马逊和亚马逊简单通知服务 (Amazon S EventBridge NS) Semple Notification Service。

博客链接：预计到达时间和邻近通知

示例：直播位置更新

Kinesis Stream Tracker 应用程序：此示例演示如何使用 Kinesis 数据流通过亚马逊定位服务发布追踪器更新。该示例是一个用 python 编写的可部署 lambda 应用程序，可以与 Kinesis 数据流集成以使用 Kinesis 事件和批量更新设备位置。

存储库链接：亚马逊 Location Amazon Kinesis Data Streams Stream Tracker 应用程序

有关跟踪和地理围栏的更多信息，请参阅 Geofence and Trackers 文档。开发人员可以通过按照 AWS 的 Serverless Application Repository 文档部署应用程序，也可以直接从 Lambda 控制台部署应用程序。

设备位置直播示例应用程序：此代码示例展示了如何将设备位置数据流式传输到 Kinesis 数据流以及地理围栏通知的工作原理。此应用程序依赖于上面列出的 Kinesis Stream to Tracker 示例应用程序，才能在亚马逊定位服务中更新直播的跟踪器位置。

存储库链接：Amazon 定位设备位置直播示例应用程序

示例：地理围栏和跟踪移动应用程序

此示例应用程序展示了跟踪器和地理围栏如何结合使用 Lambda 和 AWS IoT Amazon Location 功能进行交互。有适用于 iOS 和安卓系统的教程。

教程链接：Geofence 和 Tracker 移动应用程序示例

如何使用 Amazon Location Service

您可以使用 Amazon Location Service 功能来完成与地理和位置相关的任务。然后，您可以组合这些任务来解决更复杂的用例，例如地理营销、配送和资产跟踪。

当您准备好在应用程序中构建定位功能时，请根据您的目标和倾向，使用以下方法来使用 Amazon Location Service 功能：

- 浏览工具——如果您想试用 Amazon Location 资源，以下工具是访问和试用 API 的最快方法：
  - Amazon Location 控制台提供了各种快速访问工具。您可以使用浏览页面创建和管理您的资源并试用 API。控制台还可用于创建资源（通常是一次性任务），为使用后面描述的任何其他方法做准备。
  - AWS命令行界面 (CLI) 允许您使用终端创建资源和访问 Amazon Location API。当您使用您的凭证对其进行配置时，AWS CLI 会处理身份验证。
  - 您可以查看显示如何使用 Amazon Location Service API 执行任务的代码示例和教程。其中包括一个模仿控制台中“浏览”页面大部分功能的示例。
- 平台开发工具包——如果您不在地图上可视化数据，则可以使用任何 AWS标准工具在 AWS 上进行构建。
  - 以下软件开发工具包可用：C++、Go、Java、、.NET JavaScript、Node.js、PHP、Python 和 Ruby。
- 前端开发工具包和库——如果您想使用 Amazon Location 在移动平台上构建应用程序或在任何平台上的地图上可视化数据，则有以下选择：
  - 这些AWS Amplify库在 iOS、安卓和 JavaScript网络应用程序中集成了 Amazon Location。
  - 这些 MapLibre 库允许您使用 Amazon Location 将客户端地图渲染到 iOS、Andro id 和 JavaScriptWeb 应用程序中。
  - Tangram ES 库允许您在 iOS 和 Android 网络应用程序中使用 OpenGL ES 从矢量数据渲染 2D 和 3D 地图。还有适用于 JavaScript网络应用程序的七巧板。
- 直接发送 HTTPS 请求——如果您使用的编程语言没有开发工具包可用，或者您想更好地控制向其向 AWS 发送请求的方式，则可以通过发送经 Signature Version 4 签名流程验证的直接 HTTPS 请求来访问 Amazon Location 。有关 Signature Version 4 签名流程的更多信息，请参阅 AWS 一般参考。

本章介绍使用位置数据的应用程序常见的许多任务。常见用例部分描述了如何将这些用例与其他 AWS 服务结合起来以实现更复杂的用例。

主题

使用 Amazon Location Service 的先决条件

本部分介绍您需要执行哪些操作才能使用 Amazon Location Service。对于想要使用 Amazon Location 的用户，您必须拥有 AWS 账户 并已设置访问 Amazon Location 的权限。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

授予使用 Amazon Location Service

默认状态下，您的非管理员用户没有权限。在他们访问 Amazon Location 之前，您必须通过附加具有特定权限的 IAM policy 来授予权限。在授予对资源的访问权限时，请务必遵循最低权限原则。

> **Note**
>
> 有关向未经身份验证的用户授予访问 Amazon Location Service 功能的权限（例如，在基于 Web 的应用程序中）的信息，请参阅 授予对 Amazon Location Service 的访问权限。

以下示例策略授予用户访问所有 Amazon Location 操作的权限。有关更多示例，请参阅Amazon Location Service 基于身份的策略示例。

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

要提供访问权限，请为您的用户、组或角色添加权限：

- 中的用户和群组 AWS IAM Identity Center：

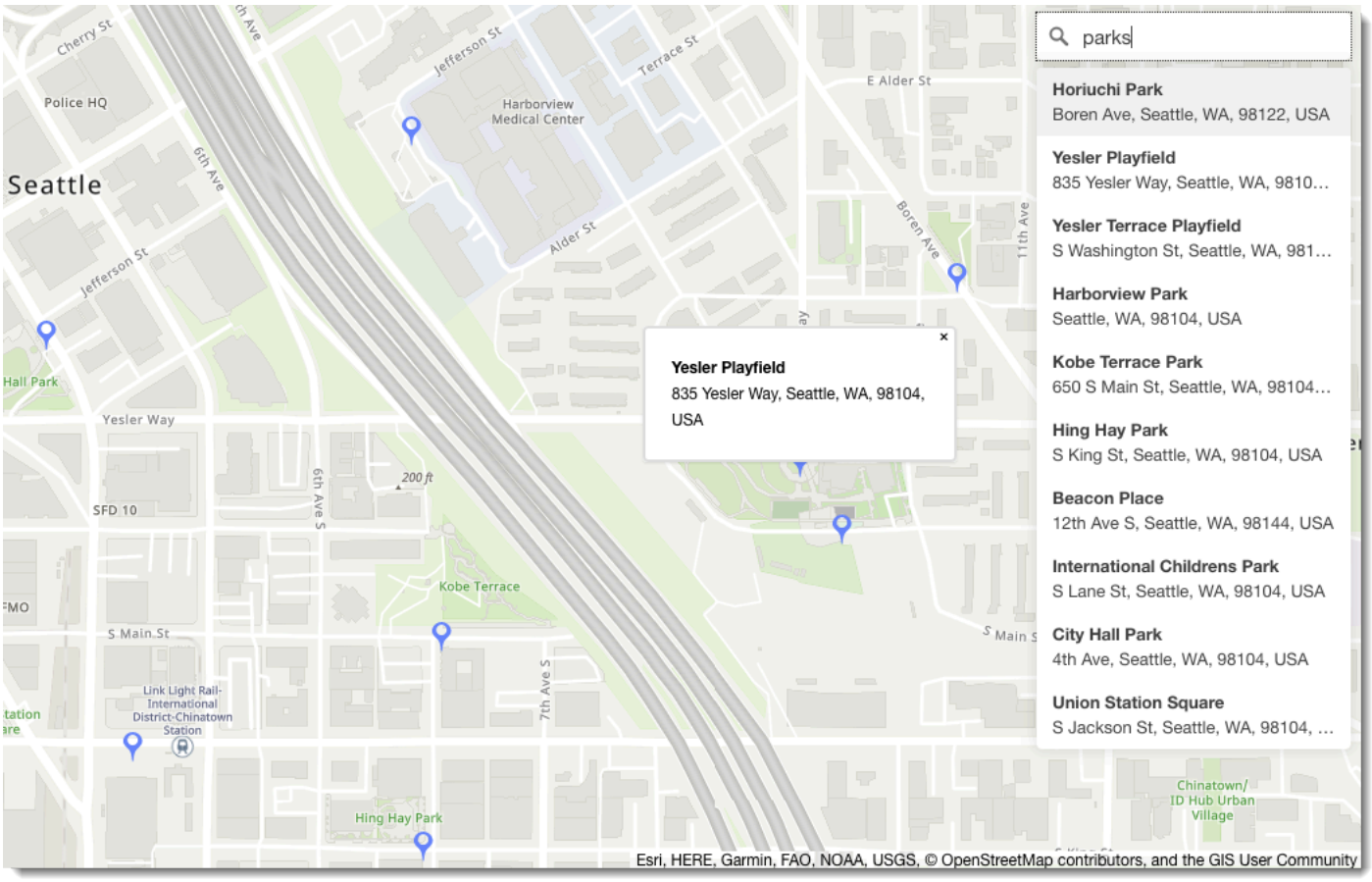
  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

在创建使用 Amazon Location Service 的应用程序时，您可能需要一些用户拥有未经身份验证的访问权限。有关这些使用案例，请参阅使用 Amazon Cognito 启用未经身份验证的访问。

在您的应用程序中使用 Amazon Location 地图

Amazon Location 地图具有成本效益且具有交互性。您可以替换应用程序中的现有地图以节省资金，也可以添加新地图以直观地显示基于位置的数据，例如您的商店位置。



Amazon Location Service 允许您通过创建和配置地图资源来选择用于地图操作的数据提供程序。地图资源用于配置数据提供程序和用于渲染地图的样式。

创建资源后，您可以直接使用 AWS 开发工具包发送请求，也可以使用专为在环境中渲染地图而制作的库来发送请求。

> **Note**
>
> 有关地图概念的概述，请参阅 映射。

主题

先决条件

在应用程序中显示地图之前，请先执行以下先决步骤：

主题

创建地图资源

要在应用程序中使用地图，您必须拥有地图资源，该资源用于指定要在地图中使用的地图样式和数据提供程序。

> **Note**
>
> 如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则不得使用 Esri 作为地理位置提供程序。有关更多详细信息，请参阅 AWS 服务条款的第 82 节。

您可以使用 Amazon Location Service 控制台、AWS CLI 或 Amazon Location API 创建地图资源。

Console

使用 Amazon Location Service 控制台创建地图资源

1. 在 Amazon Location 控制台的地图页面上，选择创建地图以预览地图样式。
2. 为新的地图资源添加名称和描述。
3. 选择地图样式。

> **Note**
>
> 如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则不得使用 Esri 作为地理位置提供程序。有关更多详细信息，请参阅 AWS 服务条款的第 82 节。

4. 从 政治观点 中选择以使用。
5. 同意 Amazon Location 条款和条件，然后选择创建地图。您可以与所选地图进行交互：放大、缩小或向任意方向平移。
6. 要允许用户切换样式（例如，允许他们在卫星图像和矢量样式之间切换），必须为每种样式创建地图资源。

   您可以在控制台的地图主页上删除不想使用的地图样式的资源。

API

使用 Amazon Location API 创建地图资源

使用 Amazon Location API 中的 `CreateMap` 操作。

以下示例是*ExampleMap*使用*VectorEsriStreets*地图样式创建名为的地图资源的 API 请求。

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

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

> **Note**
>
> 如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则不得使用 Esri 作为地理位置提供程序。有关更多详细信息，请参阅 AWS 服务条款的第 82 节。

AWS CLI

使用 AWS CLI 命令创建地图资源

使用 `create-map` 命令。

以下示例创建了一个名为 *ExampleMap*using *VectorEsriStreets*作为地图样式的地图资源。

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

> **Note**
>
> 如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则不得使用 Esri 作为地理位置提供程序。有关更多详细信息，请参阅 AWS 服务条款的第 82 节。

对您的请求进行身份验证

创建地图资源并准备开始在应用程序中构建位置要素后，您需要选择如何对请求进行身份验证。

> **Note**
>
> 大多数地图前端应用程序都需要在未经身份验证的情况下访问 Amazon Location Service 的地图或其他功能。根据您的应用程序，您可能需要使用 AWS Signature v4 对请求进行身份验证，也可以使用 Amazon Cognito 或 Amazon Location API 密钥进行未经身份验证的使用。要了解有关所有这些选项的更多信息，请参阅 授予对 Amazon Location Service 的访问权限。

在应用程序中显示地图

本部分提供有关在使用 Amazon Location API 时如何使用地图渲染工具在移动或 Web 应用程序中显示地图的教程。如如何使用 Amazon Location Service主题中所述，在使用亚马逊位置渲染地图时，您可以选择使用多种库，包括 Amplify 和 Tangra MapLibre m。

执行以下某种操作以在应用程序中显示地图：

- 在 Web 和移动前端应用程序中显示地图的最直接方法是使用 MapLibre。你可以按照MapLibre 教程甚至快速入门教程来学习如何使用 MapLibre。
- 如果您是现有 AWS Amplify 开发者，则可能需要使用 Amplify Geo开发工具包。要了解更多信息，请按照 Amplify 教程进行操作。
- 如果您是 Tangram 的现有用户，并且想在迁移到 Amazon Location Service 时继续使用它来渲染地图，请按照 Tangram 教程进行操作。

主题

将 MapLibre 图书馆与 Amazon Location Service 配合使用

以下教程将引导您使用带有 Amazon L MapLibre ocation 的图书馆。

主题

在 Amazon MapLibre Location Service 中使用 GL JS

使用 MapLibre GL JS 将客户端地图嵌入到 Web 应用程序中。

MapLibre GL JS 是一个开源 JavaScript 库，与 Amazon Location Service Maps API 提供的样式和图块兼容。您可以将 MapLibre GL JS 集成到基本 HTML 或 JavaScript 应用程序中，以嵌入可自定义的响应式客户端地图。

本教程介绍如何在基本的 HTM MapLibre L 和 JavaScript 应用程序中将 GL JS 与 Amazon Location 集成。本教程中介绍的相同库和技术也适用于框架，例如 React 和 Angular。

本教程的示例应用程序已作为 Amazon Location Service 示例存储库的一部分提供GitHub。

构建应用程序：支架

本教程创建了一个 Web 应用程序， JavaScript 用于在 HTML 页面上构建地图。

首先创建一个包含地图容器的 HTML 页面 (index.html)：

- 输入带有 id 为 map 的 div 元素，将地图的尺寸应用于地图视图。尺寸继承自视区。

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

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

构建应用程序：添加依赖项

将以下依赖项添加到您的应用程序：

- MapLibre GL JS (v3.x) 及其关联的 CSS。
- Amazon Location JavaScript 身份验证助手。

```
<!-- 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>
```

这将创建一个包含地图容器的空白页面。

构建应用程序：配置

要使用以下方法配置应用程序 JavaScript：

1. 输入资源的名称和标识符。

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

2. 使用您在使用地图——步骤 2，设置身份验证中创建的未经身份验证的身份池来实例化凭证提供程序。我们将把它放在一个名为 `initializeMap` 的函数中，该函数还将包含其他地图初始化代码，将在下一步中添加

```
// 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
}
```

构建应用程序：地图初始化

要在页面加载后显示地图，必须初始化地图。您可以调整初始地图位置、添加其他控件和叠加数据。

```
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**
>
> 您必须在应用程序或文档中为使用的每个数据提供程序提供文字标记或文本属性。归因字符串包含在样式描述符响应中的`sources.esri.attribution`、`sources.here.attribution`和`sources.grabmaptiles.attribution`键下。 MapLibre GL JS 将自动提供归因。在数据提供程序处使用 Amazon Location 资源时，请务必阅读服务条款和条件。

运行应用程序

您可以通过在本地 Web 服务器中使用此示例应用程序或在浏览器中将其打开来运行它。

要使用本地 Web 服务器，您可以使用 npx，因为它是作为 Node.js 的一部分安装的。您可以在与 `index.html` 相同的目录中使用 `npx serve`。这在 `localhost:5000` 上为应用程序提供服务。

> **Note**
>
> 如果您为未经身份验证的 Amazon Cognito 角色创建的策略包含 `referer` 条件，则可能会阻止您使用 `localhost:` URL 进行测试。在这种情况下。您可以使用提供策略中的 URL 的 Web 服务器进行测试。

完成教程后，最终的应用程序类似于以下示例。

```
<!-- 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>
```

运行此应用程序会使用您选择的地图样式显示全屏地图。此示例可在上的 Amazon Location Service 示例存储库中找到GitHub。

将适用于安卓 MapLibre 的原生 SDK 与 Amazon Location Service 配合使用

使用MapLibre原生 SDK 将交互式地图嵌入到您的 Android 应用程序中。

Android MapLibre 原生 SDK 是一个基于 Mapbox Nativ e 的库，与亚马逊定位服务地图 API 提供的样式和图块兼容。您可以集成适用于 Android 的 MapLibre Native SDK，在您的 Android 应用程序中嵌入带有可缩放、可自定义的矢量地图的交互式地图视图。

本教程介绍如何将适用于 Android MapLibre 的原生 SDK 与亚马逊定位集成。本教程的示例应用程序已作为 Amazon Location Service 示例存储库的一部分提供GitHub。

构建应用程序：初始化

要初始化应用程序，请执行以下操作：

1. 使用空活动模板创建一个新的 Android Studio 项目。
2. 确保为项目语言选择了 Kotlin。
3. 选择 API 14 的最低开发工具包：Android 4.0 (Ice Cream Sandwich) 或更高版本。
4. 打开项目结构，然后转到文件 > 项目结构... 选择依赖项部分。
5. <All Modules> 选中后，选择 + 按钮以添加新的库依赖项。
6. 添加 AWS Android 开发工具包版本 2.20.0 或更高版本。例如：`com.amazonaws:aws-android-sdk-core:2.20.0`
7. 添加适用于 Android MapLibre 的原生 SDK 版本 9.4.0 或更高版本。例如：`org.maplibre.gl:android-sdk:9.4.0`
8. 在 build.gradle 文件的项目级别，添加以下 maven 存储库以访问适用于 Android 的 MapLibre软件包：

```
allprojects {
    repositories {
        // Retain your existing repositories
        google()
        jcenter()

        //  Declare the repositories for MapLibre
```

```
        mavenCentral()
    }
}
```

构建应用程序：配置

要使用您的资源和 AWS 区域配置应用程序，请执行以下操作：

```
<?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>
```

构建应用程序：活动布局

编辑 app/src/main/res/layout/activity_main.xml：

- 添加 MapView，用于渲染地图。这还将设置地图的初始中心点。
- 添加 TextView，显示属性。

```
<?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**
>
> 您必须在应用程序或文档中为使用的每个数据提供程序提供文字标记或文本属性。属性字符串包含在样式描述符响应中的 `sources.esri.attribution`、`sources.here.attribution` 和 `source.grabmaptiles.attribution` 键下。在数据提供程序处使用 Amazon Location 资源时，请务必阅读服务条款和条件。

构建应用程序：请求转换

创建一个名为 `SigV4Interceptor`，拦截 AWS 请求的类，并使用 Signature Version 4 对请求进行签名。创建主活动时，将在用于获取地图资源的 HTTP 客户端中进行注册。

```
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()
        }
    }
}
```

构建应用程序：主活动

主活动负责初始化将向用户显示的视图。这涉及：

- 实例化 Amazon Cognito `CredentialsProvider`。
- 注册 Signature Version 4 拦截程序。
- 通过将地图指向地图样式描述符并显示适当的属性来配置地图。

`MainActivity` 还负责将生命周期事件转发到地图视图，使其能够在两次调用之间保持活动视区。

```
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()
    }
}
```

运行此应用程序会以您选择的样式显示全屏地图。此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

在 Amazon Location Service 中使用适用于 iOS 的 MapLibre 本机 SDK

使用适用于 iOS 的MapLibre本机 SDK 将客户端地图嵌入到 iOS 应用程序中。

iOS MapLibre 原生 SDK 是一个基于 Mapbox GL Native 的库，与亚马逊定位服务地图 API 提供的样式和图块兼容。您可以集成适用于 iOS 的 MapLibre Native SDK，将带有可缩放、可自定义的矢量地图的交互式地图视图嵌入到您的 iOS 应用程序中。

本教程介绍如何将 iOS MapLibre 原生 SDK 与 Amazon Location 集成。本教程的示例应用程序已作为 Amazon Location Service 示例存储库的一部分提供GitHub。

构建应用程序：初始化

要初始化应用程序，请执行以下操作：

1. 在应用程序模板中创建新的 Xcode 项目。
2. 选择 SwiftUI 作为其界面。
3. 选择 SwiftUI 应用程序作为其生命周期。
4. 选择 Swift 作为其语言。

使用 Swift 软件包添加 MapLibre依赖关系

要向 Xcode 项目添加软件包依赖项，请执行以下操作：

1. 导航到文件 > Swift 软件包 > 添加包依赖项。
2. 输入存储库 URL：**`https://github.com/maplibre/maplibre-gl-native-distribution`**

> **Note**
>
> 有关 Swift Packages 的更多信息，请参阅 Apple .com 上的向应用程序添加软件包依赖项

3. 在您的终端中，安装 CocoaPods：

```
sudo gem install cocoapods
```

4. 导航到应用程序的项目目录并使用 CocoaPods 包管理器初始化 Podfile：

```
pod init
```

5. 打开要添加 `AWSCore` 为依赖项的 Podfile：

```
platform :ios, '12.0'

target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
end
```

6. 下载并安装依赖项：

```
pod install --repo-update
```

7. 打开 CocoaPods 创建以下内容的 Xcode 工作区：

```
xed .
```

构建应用程序：配置

将以下键和值添加到 Info.plist 以配置应用程序：

键	值
AWSRegion	us-east-1
IdentityPoolId	us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd
MapName	ExampleMap

构建应用程序： ContentView 布局

要渲染地图，请编辑 `ContentView.swift`：

- 添加渲染地图的 `MapView`。
- 添加显示属性的 `TextField`。

这还会设置地图的初始中心点。

```
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**
>
> 您必须在应用程序或文档中为使用的每个数据提供程序提供文字标记或文本属性。属性字符串包含在样式描述符响应中的 `sources.esri.attribution`、`sources.here.attribution` 和 `source.grabmaptiles.attribution` 键下。在数据提供程序处使用 Amazon Location 资源时，请务必阅读服务条款和条件。

构建应用程序：请求转换

创建一个名为 `AWSSignatureV4Delegate.swift` 的新 Swift 文件，其中包含以下类定义，以拦截 AWS 请求并使用 Signature Version 4 对其进行签名。该类的一个实例将被分配为离线存储委托，该委托人还负责在地图视图中重写 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
    }
}
```

构建应用程序：地图视图

地图视图负责初始化实例 AWSSignatureV4Delegate 和配置底层 MGLMapView，底层视图获取资源并渲染地图。它还处理将属性字符串从样式描述符的来源传播回 ContentView。

创建一个名为 MapView.swift 的新 Swift 文件，其中包含以下 struct 定义：

```
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
    }
}
```

运行此应用程序会以您选择的样式显示全屏地图。此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

将 Amplify 库与 Amazon Location Service 配合使用

以下教程将指导您完通过 Amazon Location 使用 AWS Amplify。Amplify 使用 MapLibre GL JS 在 JavaScript基于你的应用程序中渲染地图。

Amplify 是一组开源客户端库，为不同类别的服务提供接口，包括由 Amazon Location Service 提供支持的 Amplify Geo。 了解有关 AWS Amplify Geo JavaScript 库的更多信息。

> **Note**
>
> 本教程假设您已经按照使用地图——向应用程序添加地图中的步骤进行操作。

构建应用程序：支架

本教程创建了一个 Web 应用程序， JavaScript 用于在 HTML 页面上构建地图。

首先创建一个包含地图容器的 HTML 页面 (`index.html`)：

- 输入带有 `id` 为 `map` 的 `div` 元素，将地图的尺寸应用于地图视图。尺寸继承自视区。

```
<html>
  <head>
    <style>
      body { margin: 0; }
      #map { height: 100vh; } /* 100% of viewport height */
    </style>
  </head>

  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

构建应用程序：添加依赖项

将以下依赖项添加到您的应用程序：

- AWS Amplify 地图和地理库。
- AWS Amplify 核心库。
- AWS Amplify 身份验证库。
- AWS Amplify 样式表。

```
<!-- 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>
```

这将创建一个包含地图容器的空白页面。

构建应用程序：配置

要使用以下方法配置应用程序 JavaScript：

1. 输入您在使用地图——步骤 2，设置身份验证中创建的未经身份验证的身份池的标识符。

   ```
   // 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. 配置 AWS Amplify 为使用您创建的资源，包括身份池和地图资源（此处显示的默认名称为 `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,
    },
  }
});
```

构建应用程序：地图初始化

要在页面加载后显示地图，必须初始化地图。您可以调整初始地图位置、添加其他控件和叠加数据。

```
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**
>
> 您必须在应用程序或文档中为使用的每个数据提供程序提供文字标记或文本属性。属性字符串包含在样式描述符响应中的 `sources.esri.attribution`、`sources.here.attribution` 和 `sources.grabmaptiles.attribution` 键下。Amplify 将自动提供属性。在数据提供程序处使用 Amazon Location 资源时，请务必阅读服务条款和条件。

运行应用程序

您可以通过在本地 Web 服务器中使用此示例应用程序或在浏览器中将其打开来运行它。

要使用本地 Web 服务器，您可以使用作为 Node.js 一部分安装的 npx，也可以使用您选择的任何其他 Web 服务器。要使用 npx，请在与 `npx serve` 相同的目录中键入 `index.html`。这在 `localhost:5000` 上为应用程序提供服务。

> **Note**
>
> 如果您为未经身份验证的 Amazon Cognito 角色创建的策略包含 `referer` 条件，则可能会阻止您使用 `localhost:` URL 进行测试。在这种情况下。您可以使用提供策略中的 URL 的 Web 服务器进行测试。

完成教程后，最终的应用程序类似于以下示例。

```
<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>
```

运行此应用程序会使用您选择的地图样式显示全屏地图。Amazon Location Service 控制台中任何地图资源页面的嵌入地图选项卡上也描述了此示例。

完成本教程后，请转至 AWS Amplify 文档中的显示地图主题以了解更多信息，包括如何在地图上显示标记。

通过 Amazon Location Service 使用 Tangram

本部分提供以下关于如何将 Tangram 与 Amazon Location 集成的教程。

> **Important**
>
> 以下教程中的 Tangram 样式仅与使用 `VectorHereContrast` 样式配置的 Amazon Location 地图资源兼容。

以下是*TangramExampleMap*使用该*VectorHereContrast*样式创建名为的新地图资源的 AWS CLI 命令示例：

```
aws --region us-east-1 \
  location \
  create-map \
  --map-name "TangramExampleMap" \
  --configuration "Style=VectorHereContrast"
```

> **Note**
>
> 计费取决于您的使用情况。您可能会因为使用其他 AWS 服务而产生费用。想要了解更多信息，请参阅 Amazon Location Service 定价。

主题

- 通过 Amazon Location Service 使用 Tangram
- 在 Amazon Location Service 上使用 Android 版 Tangram ES
- 通过 Amazon Location Service，把 ES 用于 iOS

通过 Amazon Location Service 使用 Tangram

Tangram 是一款灵活的地图引擎，专为实时渲染矢量图块中的 2D 和 3D 地图而设计。它可以与 Mapzen 设计的样式和 Amazon Location Service Maps API 提供的 HERE 图块一起使用。本指南描述了如何在基本的 HTML/ JavaScript 应用程序中将 Tangram 与 Amazon Location 集成，尽管在使用 React 和 Angular 等框架时也适用相同的库和技术。

七巧板建立在Leaflet之上，Le aflet 是一个用于移动设备友好型交互式地图的开源 JavaScript 库。这意味着许多与 Leaflet兼容的插件和控件也可以与 Tangram 一起使用。

使用 HERE 中的地图时，专为 Tilezen 架构而设计的 Tangram 风格在很大程度上与 Amazon Location 兼容。其中包括：

- Bubble Wrap——功能齐全的寻路风格，带有用于显示兴趣点的有用图标
- Cinnabar——经典外观，是普通测绘应用的首选
- Refill——一种专为数据可视化叠加而设计的极简主义地图风格，灵感来自 Stamen Design 开创性的 Toner 风格
- Tron——浏览 TRON 视觉语言中的比例变换
- Walkabout——以户外为重点的风格，非常适合徒步旅行或外出旅行

本指南介绍如何使用名为 Bubble Wrap 的七巧板风格在基本 HTML/ JavaScript 应用程序中将七巧板与 Amazon Location 集成。此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

虽然其他 Tangram 风格最好搭配编码地形信息的栅格图块，但 Amazon Location 尚不支持此功能。

> **Important**
>
> 以下教程中的 Tangram 样式仅与使用 `VectorHereContrast` 样式配置的 Amazon Location 地图资源兼容。

构建应用程序：支架

该应用程序是一个 HTML 页面， JavaScript 用于在您的 Web 应用程序上构建地图。创建 HTML 页面 (`index.html`) 并创建地图的容器：

- 输入带有 `id` 为地图的 `div` 元素，将地图的尺寸应用于地图视图。
- 尺寸继承自视区。

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

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

构建应用程序：添加依赖项

添加以下依赖项：

- Leaflet 及其链接的 CSS。
- Tangram。
- 适用于 AWS 开发工具包 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>
```

这将创建一个包含必要先决条件的空页面。下一步将指导您为应用程序编写 JavaScript 代码。

构建应用程序：配置

要使用您的资源和证书配置应用程序，请执行以下操作：

1. 输入资源的名称和标识符。

   ```
   // 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. 使用您在使用地图——步骤 2，设置身份验证中创建的未经身份验证的身份池来实例化凭证提供程序。由于这使用的是正常 AWS 开发工具包工作流程之外的凭证，因此会话将在 1 小时后过期。

   ```
   // 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. 虽然 Tangram 允许您覆盖用于获取方块的 URL，但它不包括拦截请求以便对其进行签名的功能。

   要解决此问题，请使用合成主机名 `amazon.location` 重写 `sources.mapzen.url` 指向 Amazon Location，该主机名将由 Service Worker 处理。以下是使用 Bubble Wrap 配置场景的示例：

   ```
   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,
    },
  },
};
```

构建应用程序：请求转换

要注册和初始化 Service Worker，请创建一个要在地图初始化之前调用的 `registerServiceWorker` 函数。这将注册一个名`sw.js`为服务工作者控制的单独文件中提供的 JavaScript 代码`index.html`。

凭证从 Amazon Cognito 加载，并与该地区一起传递给 Service Worker，以提供使用 Signature Version 4 签署图切片请求的信息。

```
/**
 * Register a service worker that will rewrite and sign requests using Signature
 Version 4.
 */
async function registerServiceWorker() {
  if ("serviceWorker" in navigator) {
    try {
      const reg = await navigator.serviceWorker.register("./sw.js");
```

```
      // refresh credentials from Amazon Cognito
      await credentials.refreshPromise();

      await reg.active.ready;

      if (navigator.serviceWorker.controller == null) {
        // trigger a navigate event to active the controller for this page
        window.location.reload();
      }

      // pass credentials to the service worker
      reg.active.postMessage({
        credentials: {
          accessKeyId: credentials.accessKeyId,
          secretAccessKey: credentials.secretAccessKey,
          sessionToken: credentials.sessionToken,
        },
        region: AWS.config.region,
      });
    } catch (error) {
      console.error("Service worker registration failed:", error);
    }
  } else {
    console.warn("Service worker support is required for this example");
  }
}
```

sw.js 中的 Service Worker 实现会侦听 message 事件以获取凭据和区域配置更改。它还通过监听 fetch 事件来充当代理服务器。以 amazon.location 合成主机名为目标的 fetch 事件将被重写为针对相应的 Amazon Location API，并使用 Amplify Core 的 Signer 进行签名。

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

要自动续订凭证并在证书到期之前将其发送给 Service Worker，请在 `index.html` 中使用以下函数：

```
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());
}
```

构建应用程序：地图初始化

要在页面加载后显示地图，必须初始化地图。您可以选择调整初始地图位置、添加其他控件和叠加数据。

> **Note**
>
> 您必须在应用程序或文档中为使用的每个数据提供程序提供文字标记或文本属性。属性字符串包含在样式描述符响应中的

sources.esri.attribution、sources.here.attribution 和 source.grabmaptiles.attribution 键下。
由于 Tangram 不请求这些资源，并且仅与 HERE 的地图兼容，因此请使用“© 2020 HERE”。
在数据提供程序处使用 Amazon Location 资源时，请务必阅读服务条款和条件。

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

运行应用程序

要运行此示例，您可以：

- 使用支持 HTTPS 的主机，
- 使用本地 Web 服务器遵守 Service Worker 的安全限制。

要使用本地 Web 服务器，您可以使用 npx，因为它是作为 Node.js 的一部分安装的。可以在 index.html 和 sw.js 的同一个目录中使用 npx serve。这为 localhost:5000 上的应用程序提供服务。

以下是 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 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));
});
```

此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

在 Amazon Location Service 上使用 Android 版 Tangram ES

Tangram ES 是一个 C++ 库，用于使用 OpenGL ES 根据矢量数据渲染 2D 和 3D 地图。它是 Tangram 的原生对应部分。

使用 HERE 中的地图时，专为 Tilezen 架构而设计的 Tangram 风格在很大程度上与 Amazon Location 兼容。其中包括：

- Bubble Wrap——一种功能齐全的寻路风格，带有用于显示兴趣点的有用图标。
- Cinnabar——经典外观，是普通测绘应用的首选。
- Refill——一种极简主义的地图风格，专为数据可视化叠加而设计，灵感来自 Stamen Design 开创性的 Toner 风格。
- Tron——浏览 TRON 视觉语言中的比例变换
- Walkabout——以户外为重点的风格，非常适合徒步旅行或外出旅行

本指南介绍如何使用名为 Cinnabar 的 Tangram 风格将 Android 的 Tangram ES 与 Amazon Location 集成。此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

虽然其他 Tangram 风格最好搭配编码地形信息的栅格图块，但 Amazon Location 尚不支持此功能。

> **Important**
>
> 以下教程中的 Tangram 样式仅与使用 `VectorHereContrast` 样式配置的 Amazon Location 地图资源兼容。

构建应用程序：初始化

要初始化应用程序，请执行以下操作：

1. 使用空活动模板创建一个新的 Android Studio 项目。
2. 确保为项目语言选择了 Kotlin。
3. 选择最低 API 16：Android 4.1（Jelly Bean）或更高版本的开发工具包。
4. 打开项目结构，选择文件、项目结构...，然后选择依赖项部分。
5. <All Modules> 选中后，选择 + 按钮以添加新的库依赖项。
6. 添加 AWS Android 开发工具包 2.19.1 或更高版本。例如：`com.amazonaws:aws-android-sdk-core:2.19.1`
7. 添加 Tangram 版本 0.13.0 或更高版本。例如：`com.mapzen.tangram:tangram:0.13.0`。

> **Note**
>
> 搜索 Tangram：`com.mapzen.tangram:tangram:0.13.0` 将生成一条消息，提示“未找到”，但选择“确定”将允许将其添加。

构建应用程序：配置

要使用您的资源和 AWS 区域配置应用程序，请执行以下操作：

1. 创建`app/src/main/res/values/configuration.xml`。
2. 输入您的资源的名称和标识符，以及它们的创建 AWS 区域：

```
<?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>
```

构建应用程序：活动布局

编辑 app/src/main/res/layout/activity_main.xml：

- 添加 MapView，用于渲染地图。这还将设置地图的初始中心点。
- 添加 TextView，显示属性。

这还将设置地图的初始中心点。

> **Note**
>
> 您必须在应用程序或文档中为使用的每个数据提供程序提供文字标记或文本属性。属性字符串包含在样式描述符响应中的 sources.esri.attribution、sources.here.attribution 和 source.grabmaptiles.attribution 键下。
> 由于 Tangram 不请求这些资源，并且仅与 HERE 的地图兼容，因此请使用“© 2020 HERE”。
> 在数据提供程序处使用 Amazon Location 资源时，请务必阅读服务条款和条件。

```
<?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>
```

构建应用程序：请求转换

创建一个名为 SigV4Interceptor，拦截 AWS 请求的类，并使用 Signature Version 4 对请求进行签名。创建主活动时，将在用于获取地图资源的 HTTP 客户端中进行注册。

```
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()
        }
    }
}
```

构建应用程序：主活动

主活动负责初始化将向用户显示的视图。这涉及：

- 实例化 Amazon Cognito `CredentialsProvider`。
- 注册 Signature Version 4 拦截程序。
- 通过将地图指向地图样式、覆盖图块 URL 并显示适当的属性来配置地图。

`MainActivity` 还负责将生命周期事件转发到地图视图。

```
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()
    }
}
```

运行此应用程序会以您选择的样式显示全屏地图。此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

通过 Amazon Location Service，把 ES 用于 iOS

Tangram ES 是一个 C++ 库，用于使用 OpenGL ES 根据矢量数据渲染 2D 和 3D 地图。它是 Tangram 的原生对应部分。

使用 HERE 中的地图时，专为 Tilezen 架构而设计的 Tangram 风格在很大程度上与 Amazon Location 兼容。其中包括：

- Bubble Wrap——功能齐全的寻路风格，带有用于显示兴趣点的有用图标
- Cinnabar——经典外观，是普通测绘应用的首选
- Refill——一种专为数据可视化叠加而设计的极简主义地图风格，灵感来自 Stamen Design 开创性的 Toner 风格
- Tron——浏览 TRON 视觉语言中的比例变换
- Walkabout——以户外为重点的风格，非常适合徒步旅行或外出旅行

本指南介绍如何使用名为 Cinnabar 的 Tangram 风格将 iOS 版 Tangram ES 与 Amazon Location 集成。此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

虽然其他 Tangram 风格最好搭配编码地形信息的栅格图块，但 Amazon Location 尚不支持此功能。

> **Important**
>
> 以下教程中的 Tangram 样式仅与使用 `VectorHereContrast` 样式配置的 Amazon Location 地图资源兼容。

构建应用程序：初始化

初始化应用程序：

1. 在应用程序模板中创建新的 Xcode 项目。
2. 选择 SwiftUI 作为其界面。
3. 选择 SwiftUI 应用程序作为其生命周期。
4. 选择 Swift 作为其语言。

构建应用程序：添加依赖项

要添加依赖关系，您可以使用依赖关系管理器，例如 CocoaPods：

1. 在您的终端中，安装 CocoaPods：

```
sudo gem install cocoapods
```

2. 导航到应用程序的项目目录并使用 CocoaPods包管理器初始化 Podfile：

```
pod init
```

3. 打开 Podfile，以添加 `AWSCore` 和 `Tangram-es` 作为依赖项：

```
platform :ios, '12.0'

target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
  pod 'Tangram-es'
end
```

4. 下载并安装依赖项：

```
pod install --repo-update
```

5. 打开 CocoaPods 创建以下内容的 Xcode 工作区：

```
xed .
```

构建应用程序：配置

将以下键和值添加到 Info.plist 以配置应用程序并禁用遥测：

键	值
AWSRegion	us-east-1
IdentityPoolId	us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd
MapName	ExampleMap
SceneURL	https://www.nextzen.org/carto/cinnabar-style/9/cinnabar-style.zip

构建应用程序： ContentView 布局

要渲染地图，请编辑 `ContentView.swift`：

- 添加渲染地图的 `MapView`。
- 添加显示属性的 `TextField`。

这还会设置地图的初始中心点。

> **Note**
>
> 您必须在应用程序或文档中为使用的每个数据提供程序提供文字标记或文本属性。属性字符串包含在样式描述符响应中的 `sources.esri.attribution`、`sources.here.attribution` 和 `source.grabmaptiles.attribution` 键下。在数据提供程序处使用 Amazon Location 资源时，请务必阅读服务条款和条件。

```
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()
    }
}
```

构建应用程序：请求转换

创建一个名为 AWSSignatureV4URLHandler.swift 的新 Swift 文件，其中包含以下类定义，用于拦截 AWS 请求并使用 Signature Version 4 对其进行签名。这将在 Tangram MapView 中注册为 URL 处理程序。

```
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)
    }
}
```

构建应用程序：地图视图

地图视图负责初始化实例 AWSSignatureV4Delegate 和配置底层 MGLMapView，底层视图获取资源并渲染地图。它还处理将属性字符串从样式描述符的来源传播回 ContentView。

创建一个名为 MapView.swift 的新 Swift 文件，其中包含以下 struct 定义：

```
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) {
    }
}
```

运行此应用程序会以您选择的样式显示全屏地图。此示例作为 Amazon Location Service 示例存储库的一部分提供GitHub。

在地图上绘制数据要素

有了渲染地图、使用 Amp MapLibre lify 或 Tangram 渲染地图的应用程序之后，下一步自然就是在地图顶部绘制要素。例如，您可能希望将客户位置渲染为地图上的标记。

通常，您可以使用 “地点” 搜索功能从数据中查找位置，然后使用 Amplify MapLibre、或 Tangram 的功能来渲染位置。

要查看在地图上渲染不同类型对象的示例，请参阅以下 MapLibre 示例：

- 示例：绘制标记
- 示例：绘制聚集点
- 示例：绘制多边形

有关更多示例和教程，请参阅 使用 Amazon Location Service 的代码示例和教程。

使用设置地图的范围 MapLibre

有时您不希望用户能够在整个世界范围内进行平移或缩放。如果您使用的是 MapLibre的地图控件，则可以使用选项限制地图控件的范围或边界，并使用和maxBounds选项限制缩放。minZoom maxZoom

以下代码示例演示如何初始化地图控件以将平移限制到特定边界（在本例中为 Grab 数据源的范围）。

> **Note**
>
> 这些示例已包含在本教程中 JavaScript，并且在创建 Web 应用程序教程的上下文中起作用。

```
// 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,
    }
);
```

同样，您可以为地图设置最小和最大缩放级别。两者的值都可以介于 0 和 24 之间，但最小缩放的默认值为 0，最大缩放的默认值为 22（数据提供程序可能不会提供所有缩放级别的数据）。 大多数地图库都会自动处理这个问题）。以下示例初始化 MapLibre Map 控件上的minZoom和maxZoom选项。

```
// Set the minimum and maximum zoom levels
var mlglMap = new maplibregl.Map(
    {
    container: 'map',
    style: mapName,
    maxZoom: 12,
    minZoom: 5,
    transformRequest,
    }
);
```

> **Tip**
>
> MapLibre Map 控件还允许在运行时而不是初始化期间使用`get...`和`set...`函数设置这些选项。例如，使用 `getMaxBounds` 和 `setMaxBounds` 在运行时更改地图边界。

管理您的地图资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 来管理您的地图资源。

列出地图资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 查看地图资源列表。

Console

使用 Amazon Location 控制台查看现有地图资源列表

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地图。
3. 在我的地图下查看您的地图资源列表。

API

使用 Amazon Location 地图 API 中的 `ListMaps` 操作。

以下示例是获取 AWS 账户中地图资源列表的 API 请求。

```
POST /maps/v0/list-maps
```

以下为 `ListMaps` 响应示例：

```
{
   "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。

以下示例是获取 AWS 账户中地图资源列表的 AWS CLI。

```
aws location list-maps
```

获取地图资源详情

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 获取有关您的 Amazon Web Services 账户中任何地图资源的详细信息。

Console

使用 Amazon Location 控制台查看地图资源的详细信息

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地图。
3. 在我的地图下，选择目标地图资源的名称链接。

API

使用 Amazon Location 地图 API 中的 `DescribeMap` 操作。

以下示例是获取地图资源详细信息的 API 请求*ExampleMap*。

```
GET /maps/v0/maps/ExampleMap
```

以下为 `DescribeMap` 响应示例：

```
{
    "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。

以下示例是AWS CLI获取地图资源详细信息的示例*ExampleMap*。

```
aws location describe-map \
    --map-name "ExampleMap"
```

删除地图资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 从您的 AWS 账户中删除地图资源。

> **Warning**
>
> 此操作将永久删除资源。

Console

使用 Amazon Location 控制台删除现有地图资源

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地图。
3. 在我的地图列表下，从列表中选择目标地图。

4. 选择删除地图

API

使用 Amazon Location 地图 API 中的 DeleteMap 操作。

以下示例是删除地图资源的 API 请求*ExampleMap*。

```
DELETE /maps/v0/maps/ExampleMap
```

以下是 DeleteMap 成功响应的示例：

```
HTTP/1.1 200
```

CLI

使用 delete-map 命令。

以下示例是删除地图资源的AWS CLI命令*ExampleMap*。

```
aws location delete-map \
    --map-name "ExampleMap"
```

使用 Amazon Location 搜索地点和地理位置数据

Amazon Location 包括搜索您所选提供程序的地理位置或地点数据的功能。有几种搜索可供选择。

- 地理编码——地理编码是根据文本输入搜索地址、区域、公司名称或其他兴趣点的过程。它返回找到的结果的详细信息和位置（以纬度和经度表示）。
- 反向地理编码——反向地理编码允许您查找给定位置附近的地点。
- 自动完成——自动完成是指用户在查询中键入内容时自动提出建议的过程。例如，如果他们键入 **Par**，一个建议可能是 Paris, France。

Amazon Location 允许您通过创建和配置地点索引资源来选择用于地点搜索操作的数据提供程序。

创建资源后，您可以使用首选语言的 AWS 软件开发工具包、Amplify 或 REST API 端点发送请求。您可以使用响应中的数据在地图上标记位置、丰富位置数据以及将位置转换为人类可读的文本。

> **Note**
>
> 有关搜索地点概念的概述，请参阅 地点搜索。

主题

先决条件

在开始进行地理编码、反向地理编码或搜索地点之前，请按照先决条件进行操作：

主题

创建地点索引资源

首先在您的 AWS 账户中创建地点索引资源。

创建地点索引资源时，您可以从支持地理编码、反向地理编码和搜索查询的数据提供程序中进行选择：

1. Esri——有关 Esri 在您感兴趣区域中的覆盖范围的更多信息，请参阅 Esri 文档中的 Esri 地理编码覆盖范围的更多信息。
2. HERE Technologies——有关 HERE 在您感兴趣的区域覆盖范围的更多信息，请参阅 HERE 文档中的 HERE 地理编码覆盖范围。
3. Grab——Grab 仅为东南亚地区提供数据。有关 Grab 的覆盖范围的更多信息，请参阅本指南中的 覆盖的国家/地区和区域。

您可以使用 Amazon Location Service 控制台 AWS CLI、或亚马逊定位 API 来执行此操作。

Console

使用 Amazon Location Service 控制台创建地点索引资源

1. 打开 Amazon Location Service 控制台：https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择地点索引。
3. 选择创建地点索引。
4. 填写以下选框：
   - 名称——输入地点索引资源的名称。例如，*ExamplePlaceIndex*。最多 100 个字符。有效条目包括：字母数字字符、连字符、句号和下划线。
   - 描述——输入可选描述。
5. 在数据提供程序下，选择要与您的位置索引资源配合使用的可用数据提供程序。

   > **Note**
   >
   > 如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则不得使用 Esri 作为地理位置提供程序。有关更多详细信息，请参阅 AWS 服务条款的第 82 节。

6. 在数据存储选项下，指定是否要存储来自位置索引资源的搜索结果。
7. （可选）在 Tags (标签) 下，输入标签 Key (键) 和 Value (值)。这会为您的新地点索引资源添加标签。有关更多信息，请参阅标记资源。
8. 选择创建地点索引。

API

使用 Amazon Location API 创建地点索引资源

使用 Amazon Location 地点 API 中的 `CreatePlaceIndex` 操作。

以下示例是*ExamplePlaceIndex*使用数据提供程序 *Esri ########### API* 请求。

```
POST /places/v0/indexes
Content-type: application/json
```

```
{
   "DataSource": "Esri",
   "DataSourceConfiguration": {
      "IntendedUse": "SingleUse"
   },
   "Description": "string",
   "IndexName": "ExamplePlaceIndex",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

使用 AWS CLI 命令创建地点索引资源

使用 create-place-index 命令。

以下示例*ExamplePlaceIndex*使用 *Esri* 作为数据提供器创建名为的地点索引资源。

```
aws location \
  create-place-index \
  --data-source "Esri" \
  --description "Example place index" \
  --index-name "ExamplePlaceIndex" \
  --tags Tag1=Value1
```

> **Note**
>
> 计费取决于您的使用情况。您可能会因为使用其他 AWS 服务而产生费用。想要了解更多信息，请参阅 Amazon Location Service 定价。

对您的请求进行身份验证

创建地点索引资源并准备开始在应用程序中构建位置功能后，请选择如何对请求进行身份验证：

- 要探索访问服务的方式，请参阅 Accessing Amazon Location Service。
- 如果您的网站有匿名用户，则可能需要使用 API 密钥或 Amazon Cognito。

示例

以下示例演示如何使用 API 密钥进行授权、使用 AWS JavaScript SDK v3 和亚马逊地点JavaScript身份验证助手。

```
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

地理编码是一个将文本（例如地址、区域、公司名称或兴趣点）转换为一组地理坐标的过程。您可以使用地点索引资源提交地理编码请求，并合并从地理编码中检索到的数据，以便在地图上显示适用于 Web 或移动应用程序的数据。

本部分将指导您如何发送简单的地理编码请求，以及如何发送具有可选规格的地理编码请求。

地理编码

您可以使用将地址转换为一组坐标的 SearchPlaceIndexForText 操作提交简单的地理编码请求。简单请求包含以下必需参数：

- Text——要转换为一组坐标的地址、名称、城市或地区。例如：字符串 Any Town。

要指定每页的最大结果数，请使用以下可选参数：

- MaxResults——限制查询响应中返回的最大结果数。

您可以使用 AWS CLI 或 Amazon 定位 API。

API

以下示例SearchPlaceIndexForText请求在地点索引资源中搜索名为 *ExamplePlaceIndex*Any T *own* 的地址、名称、城市或区域。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
    "Text": "Any Town",
    "MaxResults": 10
}
```

AWS CLI

以下示例是一个在地点索引资源中搜索名为 An *ExamplePlaceIndex*y T *own* 的地址、名称、城市或地区的search-place-index-for-text命令。

```
aws location \
    search-place-index-for-text \
        --index-name ExamplePlaceIndex \
        --text "Any Town" \
        --max-results 10
```

在位置附近进行地理编码

进行地理编码时，您可以使用以下可选参数在给定位置附近进行地理编码：

- BiasPosition——您要在附近搜索的位置。这会通过搜索最接近给定位置的结果来缩小搜索范围。定义为 [longitude, latitude]

以下示例是一个 SearchPlaceIndexForText 请求，在地点索引资源中搜索位置 [*-123.4567*,*45.6789*] 附近名为 *Any Town* 的地址、名称、城市或地区。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "BiasPosition": [-123.4567,45.6789]
}
```

在边界框内进行地理编码

您可以使用以下可选参数在边界框内进行地理编码，将结果缩小到给定边界内的坐标：

- `FilterBBox`——您指定的边界框，用于将结果过滤到方框边界内的坐标。定义为 `[LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]`

> **Note**
>
> 请求不能同时包含 `FilterBBox` 和 `BiasPosition` 参数。在请求中同时指定这两个参数会返回 `ValidationException` 错误。

以下示例是一个 SearchPlaceIndexForText 请求，在边界框中搜索名为 *Any Town* 的地址、名称、城市或地区。边界框如下所示：

- 西南角的经度为 *-124.1450*。
- 西南角的纬度为 *41.7045*。
- 东北角的经度为 *-124.1387*。
- 东北角的纬度为 *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
    ]
}
```

在一个国家内进行地理编码

您可以使用以下可选参数在您指定的一个或多个国家/地区内进行地理编码：

- `FilterCountries`——您要在其中进行地理编码的国家或地区。使用 ISO 3166 三个字母的国家/地区代码，您可以在一个请求中定义多达 100 个国家/地区。例如，AUS 用于澳大利亚。

以下示例是一个 `SearchPlaceIndexForText` 请求，搜索德国和法国名为 *Any Town* 的地址、名称、城市或地区。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "FilterCountries": ["DEU","FRA"]
}
```

按类别筛选

您可以使用以下可选参数筛选地理编码请求中返回的类别：

- `FilterCategories`——您要在查询中返回的结果类别。在单个请求中，您最多可以指定 5 个类别。您可以在类别部分找到 Amazon Location Service 类别列表。例如，您可以指定 `Hotel` 在查询中仅指定返回酒店。

以下示例是一个 `SearchPlaceIndexForText` 请求，在美国搜索一家名为 *Hometown Coffee* 的咖啡店。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Hometown Coffee",
   "FilterCategories": ["Coffee Shop"],
   "FilterCountries": ["USA"]
```

```
}
```

有关筛选类别的详细信息，请参阅 放置类别和筛选结果

使用首选语言进行地理编码

您可以使用可选 Language 参数为搜索结果设置语言首选项。例如，默认情况下，搜索 **100 Main St, Anytown, USA** 可能会返回 100 Main St, Any Town, USA。但是，如果您选择 fr 作为 Language，则结果可能会改为返回 100 Rue Principale, Any Town, États-Unis。

- Language——用于呈现查询结果的语言代码。该值必须是有效的 BCP 47 语言代码。例如，en 用于英语。

> **Note**
>
> 如果 Language 未指定，或者结果不支持指定的语言，则将使用该结果的合作伙伴的默认语言。

以下示例是一个 SearchPlaceIndexforText 请求，搜索一个名为 **Any Town** 的地点，首选语言指定为 de。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json
{
   "Text": "Any Town",
   "Language": "de"
}
```

响应示例

Example

以下是您从 Amazon Location 地点 API 调用 SearchPlaceIndexForText 操作时的响应示例。结果包括相关地点和请求摘要。根据选择 Esri 或 HERE 作为合作伙伴，将显示两个回复。

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
        }
    ]
}
```


反向地理编码

反向地理编码是将一组坐标转换为有意义的文本（例如地址、区域、公司名称或兴趣点）的过程。您可以使用地点索引资源提交反向地理编码请求，并整合从反向地理编码中检索的数据，以便在地图上显示适用于 Web 或移动应用程序的数据。

本部分将指导您如何发送简单的反向地理编码请求。

反向地理编码

您可以提交一个简单的请求，对一组坐标进行反向地理编码，然后使用 SearchPlaceIndexForPosition 操作将其转换为有意义的地址、兴趣点或没有地址的大致位置。简单请求包含以下必需参数：

- Position——要转换为地址、兴趣点或大致位置的一组坐标。使用格式定义 [longitude,latitude]。

要指定每页的最大结果数，请添加以下可选参数：

- MaxResults——限制查询响应中返回的最大结果数。

如果要为查询结果指定首选语言，请使用以下可选参数：

- Language——用于呈现结果的语言代码。该值必须是有效的 BCP 47 语言代码。例如，en 用于英语。

> **Note**
>
> 如果 Language 未指定，或者结果不支持指定的语言，则将使用该结果的合作伙伴的默认语言。

您可以使用 AWS CLI 或 Amazon 定位 API。

API

####SearchPlaceIndexForPosition##################ExamplePlaceIndex###### ### [122.3394#4 7.6159] ######

```
POST /places/v0/indexes/ExamplePlaceIndex/search/position
Content-type: application/json

{
   "Position": [-122.3394,47.6159],
   "MaxResults": 5,
   "Language": "de"
}
```

AWS CLI

######################ExamplePlaceIndex###############search-place-index-for-position## [122.3394#4 7.6159]#

```
aws location \
    search-place-index-for-position \
        --index-name ExamplePlaceIndex \
        --position -122.3394 47.6159 \
        --max-results 5 \
        --language de
```

响应示例

Example

以下是从 Amazon Location 地点 API 调用 SearchPlaceIndexForPosition 操作时的响应示例。结果将返回相关地点和请求摘要。根据选择 Esri 或 Here 作为合作伙伴，将显示两个回复。

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
        }
    ]
}
```

自动完成

当最终用户输入搜索查询时，自动完成功能会向他们提供响应式反馈。提供地址和兴趣点的建议，基于部分或拼写错误的自由格式文本。您可以使用地点索引资源来请求自动完成建议，并在您的应用程序中显示生成的建议。

Amazon Location 不支持存储自动完成建议。如果将用于自动完成调用的地点索引配置为与存储的地理编码一起使用，则会返回错误。要使用存储的地理编码并查询建议，请创建和配置多个地点索引。

本部分描述如何发送自动完成请求。它从最基本的请求形式开始，然后显示可选参数，您可以使用这些参数来提高自动填充搜索结果的相关性。

使用自动完成功能

您可以使用该 SearchPlaceIndexForSuggestions 操作提交一个简单的自动完成建议请求。最简单的请求形式只有一个必需的参数，即查询 `Text`：

- `Text`——用于生成地点建议的自由格式部分文本。例如：字符串 `eiffel tow`。

要限制返回的结果数量，请添加可选 `MaxResults` 参数：

- `MaxResults`——限制查询响应中返回的结果数量。

您可以使用 Amazon Location API 或 AWS CLI。

API

以下示例SearchPlaceIndexForSuggestions请求根据部分地名 *kamp* 在地点索引资源中搜索最多 *5* #建议。*ExamplePlaceIndex*

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
```

```
Content-type: application/json

{
    "Text": "kamp",
    "MaxResults": 5
}
```

AWS CLI

以下示例是一个search-place-index-for-suggestions命令，用于根据部分地名 *kamp* 在地点索引资源中搜索最多 *5* #建议。*ExamplePlaceIndex*

```
aws location \
              search-place-index-for-suggestions \
              --index-name ExamplePlaceIndex \
              --text kamp \
              --max-results 5
```

调用 SearchPlaceIndexForSuggestions 结果会生成一个地点列表，每个地点都有名称和 ID。您可以使用这些结果来提供用户在键入时可能正在搜索的内容的建议，例如在文本框下方提供选项的下拉列表。例如，以下是基于用户键入 *kamp* 的建议结果。

```
{
    "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
        }
    ]
}
```


下一部分介绍如何使用这些结果中的 PlaceID。

使用自动完成结果

调用 SearchPlaceIndexForSuggestions 结果会生成一个地点列表，每个地点都有名称和 ID。您可以使用这些结果来提供用户在键入时可能正在搜索的内容的建议，例如在文本框下方提供选项的下拉列表。当用户选择其中一个结果时，您可以使用他们选择的 ID 调用该GetPlace操作以返回该地点的详细信息，包括位置、地址或其他详细信息。

> **Note**
>
> 仅当原始搜索请求和调用 GetPlace 的以下所有内容都相同时，PlaceId 才有效。
>
> - 客户 AWS 账户
> - AWS 区域
> - 地点索引资源中指定的数据提供程序

通常，您可以 GetPlace 与 Amazon Location API 一起使用。以下示例是一个 GetPlace 请求，查找上一节中的建议之一。此示例基于部分地名 *kamp*。

```
POST /places/v0/indexes/ExamplePlaceIndex/
places/AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

在位置附近自动完成

当您使用 SearchPlaceIndexForSuggestions 去搜索自动完成地点建议时，您可以通过添加以下可选参数来获得更多与当地相关的建议：

- BiasPosition——您要在附近搜索的位置。定义为 [longitude, latitude]。

######SearchPlaceIndexForSuggestions#########ExamplePlaceIndex######
[32.5827#0.3 169] ####### kamp #########

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "BiasPosition": [32.5827,0.3169]
}
```

如果选择了不同的 BiasPosition，例如 [*-96.7977*, *32.7776*]，则针对同一 Text 返回的建议可能会有所不同。

在边界框内自动完成

通过添加以下可选参数，您可以缩小自动完成搜索范围，仅接收位于给定边界内的地点的建议：

- FilterBBox——您指定的边界框，用于将结果过滤到方框边界内的坐标。定义为 [LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]

> **Note**
>
> 请求不能同时包含 FilterBBox 和 BiasPosition 参数。在请求中同时指定这两个参数会返回 ValidationException 错误。

以下示例使用SearchPlaceIndexForSuggestions请求在地点索引资源*ExamplePlaceIndex*中搜索与部分查询 *kamp* 匹配的地点建议，这些建议包含在边界框中，其中：

- 边界框西南角的经度为 *32.5020*。
- 边界框西南角的纬度为 *0.2678*。
- 边界框的东北角的经度为 *32.6129*。
- 边界框的东北角的纬度为 *0.3502*。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterBBox": [
        32.5020, 0.2678,
        32.6129, 0.3502
    ]
}
```

如果选择不同的 FilterBBox，则针对同一 Text 返回的建议会有所不同，例如 [*-97.9651*、*32.0640*、 *-95.1196*、*34.0436*]。

在某个国家/地区内自动完成

通过添加以下可选参数，您可以缩小自动完成搜索范围，仅接收位于给定国家/地区或一组国家/地区内的地点的建议：

- FilterCountries——您要在其中搜索地点建议的国家/地区。使用 ISO 3166 三个字母的国家/地区代码，您最多可以在一次请求中指定 100 个国家/地区。例如，AUS 用于澳大利亚。

以下示例使用SearchPlaceIndexForSuggestions请求在地点索引资源*ExamplePlaceIndex*中搜索与部分查询 *kamp* 相匹配且包含在乌干达、肯尼亚或坦桑尼亚的地点建议：

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterCountries": ["UGA", "KEN", "TZA"]
```

```
}
```

如果选择不同的 FilterCountries 列表，例如 [“*USA*”]，则针对同一 Text 返回的建议会有所不同。

响应示例

以下是使用文本 *kamp* 对操作建议的自动完成 SearchPlaceIndexForSuggestions 操作的响应示例。


```
{
    "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
        }
    ]
}
```

使用地点 ID

搜索地点会返回结果列表。对于该结果，大多数结果都包含一个 PlaceId。您可以在 GetPlace 操作中使用 PlaceId 来返回有关该地点的信息（包括名称、地址、位置或其他详细信息）。

> **Note**
>
> 使用SearchPlaceIndexForSuggestions将返回使用任何数据源创建的任何地点索引的PlaceId结果。PlaceId仅当使用的数据源为 HERE 时，使用SearchPlaceIndexForText或才SearchPlaceIndexForPosition会返回。

每个 PlaceId 都唯一地定义了它所指的地点，但是随着时间的推移，根据上下文，一个地点可以有多个 PlaceId。以下规则描述了 PlaceId 的独特性和寿命。

- 您发出的调用中PlaceId返回的值特定于您 AWS 账户的、该 AWS 地区以及您的PlaceIndex资源中的数据提供商。 GetPlace只有当这三个属性与创建的原始调用匹配时，才会找到结果PlaceId。
- 当某个地点的数据发生变化时，该地点的 PlaceId 将发生变化。例如，当它所指的企业搬迁地点或更改名称时。
- 当后端服务进行更新时，重复搜索调用返回的 PlaceId 可能会发生变化。PlaceId将继续找到较旧的，但是新的搜索调用可能会返回不同的 ID。

PlaceId 是一个字符串。PlaceId 的长度没有具体限制。以下是有效的 PlaceId 的示例。

```
AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

对于数据已更改的地点（例如，营业地点已停业），使用 PlaceId 调用 GetPlace 将导致 404、ResourceNotFound 错误。GetPlace使用无效的PlaceId，或者一个脱离上下文（例如从另一个 AWS 账户上下文中调用）调用400将返回ValidationException错误。

虽然您可以在后续请求中使用 placeID，但 placeID 并不是永久标识符，并且在连续的 API 调用之间，ID 可能会发生变化。请查看每个数据提供者的以下 placeID 行为：

- Esri：地点 ID 将至少每季度更改一次。这些变化的典型时间段为三月、六月、九月和十二月。地点 ID 也可能在典型的季度变化之间发生变化，但频率要低得多。
- 这里：我们建议您缓存数据的时间不超过一周，以保持数据最新状态。你可以假设只有不到 1% 的ID移位会随着发布而释放，大约每周1-2次。
- Grab：在以下情况下，地点 ID 可能会过期或失效。
  - 数据操作：Grab Map Ops 可能会根据实际情况从 Grab POI 数据库中删除 POI，例如在现实世界中被关闭、被检测为重复的 POI 或包含不正确的信息。Grab 将每周将数据同步到 Waypoint 环境。
  - 插值 POI：插值 POI 是在处理请求时实时生成的临时 POI，它将在响应的字段中标记为派生。`place.result_type`插值后的兴趣点信息将保留至少 30 天，这意味着您可以在 30 天内从“地点详情 API”中按地点 ID 获取 POI 详情。30 天后，插值后的 POI（包括地点 ID 和详情）可能会过期，且无法通过“地点详情”API 进行访问。

放置类别和筛选结果

地点已分类。例如，如果您搜索一家企业，则该企业可能是 `Restaurant`。即使是搜索地址的结果也可以根据地址是否匹配到地址、街道或交叉路口进行分类。

从广义上讲，Amazon Location Service 将地点分为地点类型。兴趣点进一步归类为兴趣点类型。

> **Note**
>
> 并非所有结果都有类别。

您可以使用这些类别来筛选地理编码搜索。

筛选结果

使用时 `SearchPlaceIndexForText`，您可以筛选按要使用的类别返回的结果。例如：

- 如果您想搜索一个名为“Hometown Coffee”的地方，并且只返回归类为咖啡店的结果，则可以通过调用 `SearchPlaceIndexForText` 并在 `FilterCategories` 参数中包含兴趣点类别，`Coffee Shop` 来实现。

- 搜索“123 Main St, Anytown, WA, 98123, USA”时，您可以将结果筛选为仅限地址，这样您就不会得到邮政编码等匹配项。通过在 `FilterCategories` 参数中的 `AddressType`，包含地点类型来筛选到仅限地址。

> **Note**
>
> 并非所有数据提供程序都支持筛选，也不是所有数据提供程序都以相同的方式支持筛选。有关更多信息，请参阅按数据提供程序划分的筛选限制。

下一部分列出了您可以筛选的类别。

类别

以下列表显示了 Amazon Location Service 用于分类和筛选的类别。这些类别用于所有语言，与语言参数设置为不同的语言无关。

> **Note**
>
> Amazon Location Service 将数据提供程序的类别映射到这组类别。如果数据提供程序将某个地点归入不在 Amazon Location Service 类别列表中的类别，则该提供程序类别将作为补充类别包含在结果中。

地点类型——这些类型用于指示用于查找结果的匹配类型。

- `AddressType`——当结果与地址匹配时返回。
- `StreetType`——当结果与街道匹配时返回。
- `IntersectionType`——当结果与两条街道的交叉点匹配时返回。
- `PointOfInterestType`——当结果与兴趣点（例如企业或市政地点）匹配时返回。
- `CountryType`——当结果与国家或主要地区匹配时返回。
- `RegionType`——当结果与一个国家的某个区域（例如州或省）匹配时返回。
- `SubRegionType`——当结果与一个国家的子区域（例如县或大都市区）匹配时返回。
- `MunicipalityType`——当结果与城市或城镇匹配时返回。
- `NeighborhoodType`——当结果与城市中的某个街区或区域匹配时返回。
- `PostalCodeType`——当结果与邮政编码匹配时返回。

兴趣点类别——这些类别用于指明兴趣点结果的业务类型或地点。

- 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

按数据提供程序划分的筛选限制

并非所有提供程序都具有相同的过滤功能。下表介绍了区别。

提供商	支持筛选条件的 API	支持筛选的类别	返回值
Esri	SearchPlaceIndexForText，SearchPlaceIndexForSuggestions	按地点类型和兴趣点类别进行筛选。	分类由 SearchPlaceIndexForText、SearchPlaceIndexForPosition 和 GetPlace 返回
此处	SearchPlaceIndexForText，SearchPlaceIndexForSuggestions	仅按地点类型筛选。	分类由 SearchPlaceIndexForText、SearchPlaceIndexForSuggestions、SearchPlaceIndexForPosition 和 GetPlace 返回
Grab	不支持	不支持	不支持
Open Data (打开数据)	n/a（不支持搜索地点）	不适用	不适用

Amazon Location Service 的 Amazon Aurora PostgreSQL 用户定义函数

您可以使用 Amazon Location Service 处理存储在数据库表中的坐标和地址，以清理和丰富您的地理空间数据。

例如：

- 您可以使用地理编码将地址转换为坐标，以便对存储在数据库表中的地址进行规范化并填补数据中的空白。
- 您可以对地址进行地理编码以获取其位置，并将坐标与数据库空间函数（例如显示指定区域中的行的函数）一起使用。
- 您可以使用丰富的数据来生成自动报告，例如生成说明给定区域内所有设备的自动报告，或者生成自动机器学习报告，说明发送位置更新时故障率较高的区域。

本教程介绍如何使用 Amazon Location Service 格式化和丰富存储在 Amazon Aurora PostgreSQL 数据库表中的地址。

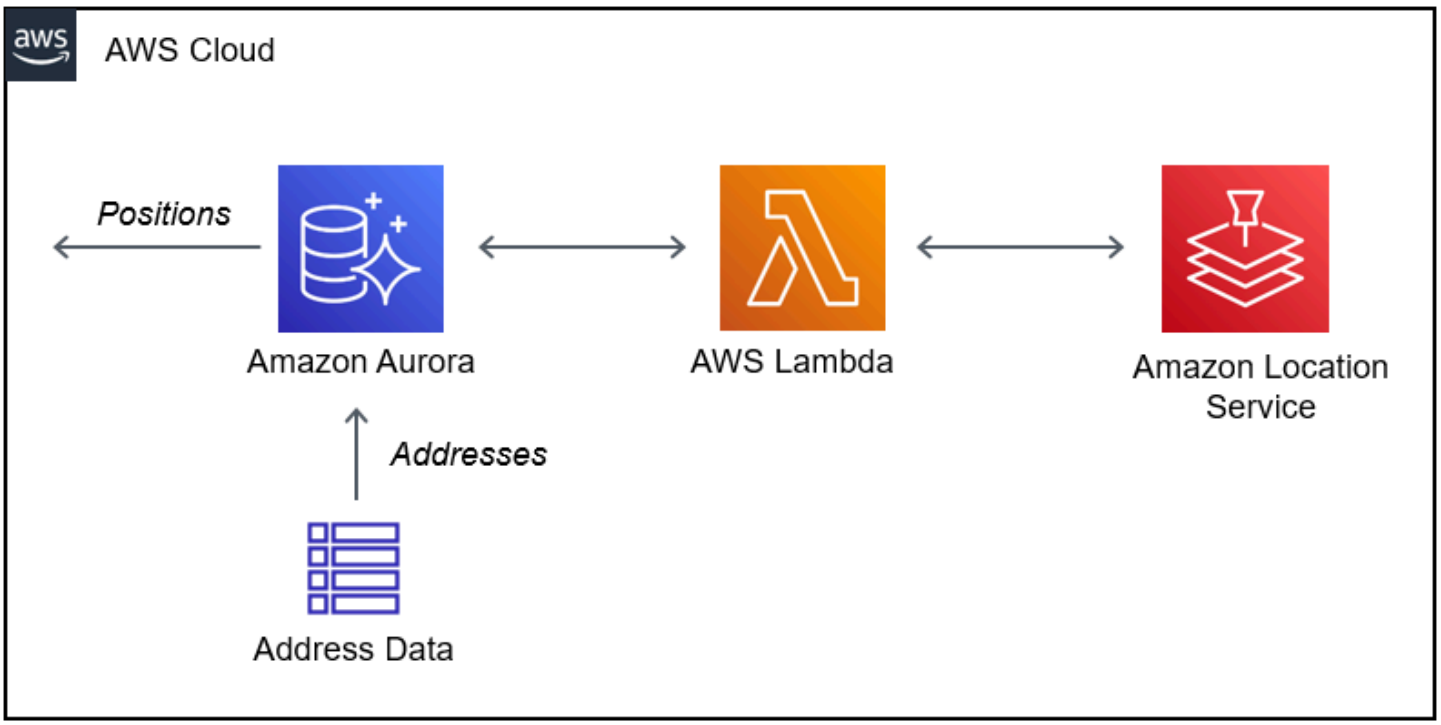
- Amazon Aurora PostgreSQL——一种完全托管的关系数据库引擎，与 MySQL 和 PostgreSQL 兼容，其输出的吞吐量是 MySQL 的五倍，是 PostgreSQL 的三倍，而无需更改您大部分现有应用程序。有关更多信息，请参阅 Amazon Aurora 用户指南中的什么是 Amazon Aurora？。

> **Important**
>
> 本教程中生成的应用程序使用存储地理编码结果的地点索引。有关存储地理编码结果的适用费用的信息，请参阅 Amazon Location Service 定价。

示例代码可在上的 Amazon Location Service 示例存储库中找到 GitHub，其中包含一个AWS CloudFormation模板。

主题

概述



该架构涉及以下集成：

- 此解决方案使用 Amazon Location 地点索引资源来支持使用操作 `SearchPlaceIndexForText` 进行地理编码查询。
- 当 IAM policy 允许 AWS Lambda 调用 Amazon Location 地理编码操作 `SearchPlaceIndexForText` 时，AWS Lambda 使用 Python Lambda 对地址进行地理编码。
- 授予 Amazon Aurora PostgreSQL 使用 SQL 用户定义函数调用地理编码 Lambda 函数的权限。

先决条件

在开始之前，您需要满足以下先决条件：

- 一个 Amazon Aurora PostgreSQL 集群。有关创建 Amazon Aurora 数据库集群的更多信息，请参阅 Amazon Aurora 用户指南。

> **Note**
>
> 如果您的 Amazon Aurora 集群未公开，则您还必须配置 Amazon Aurora，在您的 AWS 账号的虚拟私有云 (VPC) 中连接 AWS Lambda。有关更多信息，请参阅 授予 Amazon Aurora PostgreSQL 对访问 AWS Lambda 的权限。

- 用于连接到 Amazon Aurora PostgreSQL 集群的 SQL 开发人员工具。

快速入门

除了完成本教程中的步骤外，您还可以启动快速堆栈来部署支持 Amazon Location 操作 `SearchPlaceIndexForText` 的 AWS Lambda 函数。这会自动将您的 AWS 账户配置为允许 Amazon Aurora 调用 AWS Lambda。

配置 AWS 账户后，您需要：

- 将 Lambda 功能添加到 Amazon Aurora。请参阅将 IAM 角色添加到 授予 Amazon Aurora PostgreSQL 对访问 AWS Lambda 的权限 中的 Amazon Aurora 数据库集群。
- 将用户定义的函数加载到数据库中。请参阅调用 AWS Lambda 函数。

创建地点索引资源

首先创建一个支持地理编码查询的地点索引资源。

1. 打开 Amazon Location Service 控制台：https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择地点索引。
3. 填写以下选框：
   - 名称——输入地点索引资源的名称。例如，*AuroraPlaceIndex*。最多 100 个字符。有效条目包括：字母数字字符、连字符、句号和下划线。
   - 描述——输入可选描述。例如，*Amazon Aurora #####*。
4. 在数据提供程序下，选择要与您的位置索引资源配合使用的可用数据提供程序。如果您没有偏好，我们建议您从 *Esri* 开始。

5. 在数据存储选项下，指定是，将存储结果。这表示您打算将地理编码结果保存在数据库中。
6. （可选）在 Tags (标签) 下，输入标签 Key (键) 和 Value (值)。这会为您的新地点索引资源添加标签。有关更多信息，请参阅标记资源。
7. 选择创建地点索引。

创建用于地理编码的 AWS Lambda 函数

要在 Amazon Aurora PostgreSQL 和 Amazon Location Service 之间建立连接，您需要一个 AWS Lambda 函数来处理来自数据库引擎的请求。此函数转换 Lambda 用户定义的函数事件并调用 Amazon Location 操作 `SearchPlaceIndexForText`。

您可以使用 AWS Lambda 控制台、AWS Command Line Interface 或 AWS Lambda API 创建函数。

使用控制台创建 Lambda 用户定义的函数

1. 通过 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
2. 从左侧导航窗格中，选择函数。
3. 选择创建函数，并确保选择从头开始创作。
4. 填写以下选框：
   - 函数名称——输入您的函数的唯一名称。有效条目包括字母数字字符、连字符和下划线，不带空格。例如，*AuroraGeocoder*。
   - 运行时系统——选择 *Python 3.8*。
5. 选择创建函数。
6. 选择 代码 选项卡以打开编辑器。
7. 在 `lambda_function.py` 中使用以下内容覆盖占位符代码：

```
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. 如果您将地点索引命名为以外的其他名称 *AuroraPlaceIndex*，请创建一个名为的环境变量PLACE_INDEX_NAME以将资源名称分配给：

   - 选择配置选项卡中，选择环境变量。
   - 选择编辑，然后选择添加环境变量。
   - 对于键，输入 PLACE_INDEX_NAME。
   - 对于值：输入您的地点索引资源的名称。

9. 选择部署以保存您更新的函数。
10. 从测试下拉菜单中，选择配置测试事件。
11. 选择 Create new test event（新建测试事件）。
12. 输入以下测试事件：

```
{
  "text": "Baker Beach",
  "biasPosition": [-122.483, 37.790],
```

```
  "filterCountries": ["USA"]
}
```

13. 选择测试来测试 Lambda 函数。
14. 选择配置选项卡。
15. 在常规配置下：选择权限。
16. 在执行角色下：选择超链接角色名称以授予 Amazon Location Service 对您的 Lambda 函数的权限。
17. 在权限选项卡中，选择添加权限下拉菜单，然后选择创建内联策略。
18. 选择 JSON 选项卡。
19. 添加以下 IAM policy：

    - 以下策略授予向地点索引资源发送SearchPlaceIndexForText数据的权限*AuroraPlaceIndex*。

    ```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "geo:SearchPlaceIndexForText",
          "Resource": "arn:aws:geo:<Region>:<AccountId>:place-index/AuroraPlaceIndex"
        }
      ]
    }
    ```

20. 选择查看策略。
21. 输入策略名称。例如，*AuroraPlaceIndexReadOnly*。
22. 选择创建策略。

授予 Amazon Aurora PostgreSQL 对访问 AWS Lambda 的权限

必须先授予访问权限，然后 Amazon Aurora PostgreSQL 才能调用 AWS Lambda 函数。

如果您的 Amazon Aurora PostgreSQL 集群不可公开访问，则需要先为 AWS Lambda 创建一个 VPC 端点，这样 Amazon Aurora 才能调用您的 Lambda 函数。

为 AWS Lambda 创建 VPC 端点

> **Note**
>
> 只有在 Amazon Aurora PostgreSQL 集群不可公开访问时，才需要执行此步骤。

1. 打开 Amazon Virtual Private Cloud Console。
2. 在左侧导航窗格中，选择终端节点。
3. 选择创建端点。
4. 在服务名称筛选器中，输入“lambda”，然后选择 `com.amazonaws.<region>.lambda`。
5. 选择包含您的 Aurora 集群的 VPC。
6. 对于每个可用区，选择一个子网。
7. 在安全组筛选器中，输入“默认”或您的 Aurora 集群所属的安全组的名称，然后选择安全组。
8. 选择创建端点。

创建 IAM policy 以授予调用您的 AWS Lambda 函数的权限

1. 打开 IAM 控制台。
2. 在左侧导航窗格中，展开访问管理，然后选择策略。
3. 选择创建策略。
4. 在 JSON 选项卡上，输入以下策略：

   - 以下是提供 Amazon Aurora PostgreSQL 权限以调用 `AuroraGeocoder` AWS Lambda 函数的 IAM policy 示例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:<Region>:<AccountId>:function:AuroraGeocoder"
            ]
        }
    ]
}
```

5. 选择下一步：标签，以添加可选标签。
6. 选择下一步：审核。
7. 查看您的策略并输入该策略的以下详细信息：
   - 名称——使用字母数字和 '+=、.@-_'字符。最多 128 个字符。例如，*AuroraGeocoderInvoke*。
   - 描述——输入可选描述。使用字母数字和 '+=、.@-_' 字符。最多 1000 个字符。
8. 选择创建策略。记下此策略的 ARN，您使用该 ARN 将策略附加到 IAM 角色。

创建一个 IAM 角色来授予对 Amazon Relational Database Service (Amazon RDS) 的权限

通过创建 IAM 角色，Amazon Aurora PostgreSQL 可以代表您代入该角色来访问 Lambda 函数。有关更多信息，请参阅《IAM 用户指南》中的创建向 IAM 用户委派权限的角色。

以下示例是一个创建名为的角色的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"
        }
    ]
}'
```

将您的 IAM policy 附加到 IAM 角色

在拥有 IAM 角色时，附加您创建的 IAM policy。

以下示例是将策略附加*AuroraGeocoderInvoke*到角色的AWS CLI命令*AuroraGeocoderInvokeRole*。

```
aws iam attach-role-policy  --policy-arn AuroraGeocoderInvoke  --role-
name AuroraGeocoderInvokeRole
```

将该 IAM 角色添加到 Amazon Aurora 数据库集群中

以下示例是向名为的Amazon Aurora PostgreSQL数据库集群添加 IAM 角色的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
```

调用 AWS Lambda 函数

授予 Amazon Aurora PostgreSQL 调用地理编码 Lambda 函数的权限后，您可以创建 Amazon Aurora PostgreSQL 用户定义的函数来调用地理编码 AWS Lambda 函数。有关更多信息，请参阅 Amazon Aurora 用户指南中的从 Amazon Aurora PostgreSQL 数据库集群调用 AWS Lambda 函数。

安装所需的 PostgreSQL 扩展

要安装所需的 PostgreSQL 扩展 `aws_lambda` 和扩展 `aws _commons`，请参阅 Amazon Aurora 用户指南中的 Lambda 函数使用概述。

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
```

安装所需的 PostGIS 扩展

PostGIS 是一个 PostgreSQL 扩展，用于存储和管理空间信息。有关更多信息，请参阅 Amazon Relational Database Service 用户指南中的使用 PostGIS 扩展。

```
CREATE EXTENSION IF NOT EXISTS postgis;
```

创建用于调用 Lambda 函数的 SQL 用户定义函数

在 SQL 编辑器中，创建一个新的用户定义函数`f_SearchPlaceIndexForText`来调用该函数*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;
$function$;
```

调用 SQL 函数从 Aurora 进行地理编码

运行 SQL 语句会调用 Lambda 函数 *AuroraGeocoder*，该函数从数据库的数据库表中获取地址记录，并使用位置索Amazon Aurora PostgreSQL引资源对其进行地理编码。

> **Note**
>
> Amazon Aurora PostgreSQL 每次调用 SQL 用户定义函数时都会调用 Lambda 函数。
> 如果您要对 50 行进行地理编码，请 Amazon Aurora PostgreSQL 调用 Lambda 函数 50 次。
> 每行调用一次。

以下 f_SearchPlaceIndexForText SQL 函数通过 L *AuroraGeocoder*ambda 函数向亚马逊位置的 SearchPlaceIndexForText API 发出请求。该函数返回一 geom 列 PostGIS 几何图形，该列 ST_AsText(geom) 会转换为文本。

```
SELECT *, ST_AsText(geom)
FROM f_SearchPlaceIndexForText('Vancouver, BC');
```

默认情况下，返回值将包含一行。要请求更多行，直到 MaxResults 限制，请运行以下 SQL 语句，同时提供 BiasPosition 并限制结果在加拿大。

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', ST_MakePoint(-123.113, 49.260), null,
 '{"CAN"}', 5);
```

要使用边界框筛选结果，请将 Box2D 作为 filter_bbox 传递：

- FilterBBox——通过返回边界框内的位置来筛选结果。此参数为可选参数。

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', null, 'BOX(-139.06 48.30, -114.03
 60.00)'::box2d, '{"CAN"}', 5);
```

有关 PostGIS 类型和函数的更多信息，请参阅 PostGIS 参考。

丰富包含地址数据的数据库

您可以构造一个格式化的地址，同时使用 Amazon Location 操作 `SearchPlaceIndexForText` 进行规范化和地理编码，给定一个数据库表，其中包含以下数据，分为以下几列：

- `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;
```

以下示例对此数据表行进行了说明：

```
 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)
```

更新数据库表并填充列

以下示例更新表并用 SearchPlaceIndexForText 查询结果填充列：

```
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;
```

后续步骤

示例代码可在上的 Amazon Location Service 示例存储库中找到 GitHub，其中包含一个AWS CloudFormation模板。

管理您的地点索引资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 来管理您的地点索引资源。

列出您的地点索引资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 查看您的地点索引资源列表：

Console

使用 Amazon Location 控制台查看地点索引资源列表

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地点索引数。
3. 在我的地点索引下查看您的地点索引资源列表。

API

使用 Amazon Location 地点 API 中的 `ListPlaceIndexes` 操作。

以下示例是获取 AWS 账户中地点索引资源列表的 API 请求。

```
POST /places/v0/list-indexes
```

以下为 `ListPlaceIndexes` 响应示例：

```
{
   "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。

以下示例是获取 AWS 账户中地点索引资源列表的 AWS CLI。

```
aws location list-place-indexes
```

获取地点索引资源详情

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 获取有关您 AWS 账户中任何地点索引资源的详细信息：

Console

使用 Amazon Location 控制台查看地点索引资源的详细信息

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地点索引数。
3. 在我的地点索引下，选择目标地点索引资源的名称链接。

API

使用 Amazon Location 地点 API 中的 `DescribePlaceIndex` 操作。

以下示例是获取地点索引资源详细信息的 API 请求*ExamplePlaceIndex*。

```
GET /places/v0/indexes/ExamplePlaceIndex
```

以下为 `DescribePlaceIndex` 响应示例：

```
{
    "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。

以下示例是AWS CLI获取地点索引资源详细信息的示例*ExamplePlaceIndex*。

```
aws location describe-place-index \
    --index-name "ExamplePlaceIndex"
```

删除地点索引资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 从您的 Amazon Web Services 账户中删除地点索引资源：

Console

使用 Amazon Location 控制台删除地点索引资源

> ⚠ Warning
>
> 此操作将永久删除资源。

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地点索引数。
3. 在我的地点索引下，选择目标地点索引资源。
4. 选择删除地点索引。

API

使用 Amazon Location 地点 API 中的 DeletePlaceIndex 操作。

以下示例是删除地点索引资源的 API 请求*ExamplePlaceIndex*。

```
DELETE /places/v0/indexes/ExamplePlaceIndex
```

以下是 DeletePlaceIndex 成功响应的示例：

```
HTTP/1.1 200
```

CLI

使用 delete-place-index 命令。

以下示例是删除地点索引资源的AWS CLI命令*ExamplePlaceIndex*。

```
aws location delete-place-index \
    --index-name "ExamplePlaceIndex"
```

使用 Amazon Location Service 计算路线

Amazon Location 允许您通过创建和配置路由计算器资源来选择用于计算路线的数据提供程序。

您可以使用 AWS 开发工具包或 REST API 端点使用路由计算器资源来计算给定特定参数的路由。使用此路由计算器资源来计算起点、目的地和最多 23 个路径点之间的路线，以适应不同的交通方式、避让点和交通状况。

您还可以使用路由计算器资源，通过计算路由矩阵来为您的路线规划算法或产品创建输入。计算一组出发位置和一组目的地位置之间的行驶时间和行驶距离。路线规划软件可以使用该时间和距离数据来优化一条或一组路线；例如，如果您正在规划多条配送路线，并且想要为每个停靠点找到最佳路线和时间。您可以计算出不同交通方式、避让点和交通状况的路由矩阵。

> **Note**
>
> 有关路由概念的概述，请参阅 路线。

主题

先决条件

在开始计算路线之前，请按照先决条件执行以下步骤：

主题

创建路由计算器资源

计算路由之前，请在您的 AWS 账户中创建路由计算器资源。

创建路由计算器资源时，可以从可用的数据提供程序中进行选择：

1. Esri——有关 Esri 在您感兴趣区域中的覆盖范围的更多信息，请参阅有关街道网络和交通覆盖范围的 Esri 详细信息。
2. HERE Technologies——有关 HERE 在您感兴趣区域中的覆盖范围的更多信息，请参阅 HERE 汽车的路由覆盖范围和 HERE 卡车的路由覆盖范围。
3. Grab——有关 Grab 覆盖范围的更多信息，请参阅 覆盖的国家/地区和区域。

> **Note**
>
> 如果您的应用程序正在跟踪或路由您在企业中使用的资产，例如运载车辆或员工，则不得使用 Esri 作为地理位置提供程序。有关更多详细信息，请参阅 AWS 服务条款的第 82 节。

您可以使用 Amazon Location Service 控制台、AWS CLI 或 Amazon Location API 完成此操作。

Console

使用 Amazon Location 控制台创建路由计算器资源

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择路由计算器。
3. 选择创建路由计算器。
4. 填写以下选框：
   - 名称——输入路由计算器资源的名称。例如，*ExampleCalculator*。最多 100 个字符。有效条目包括：字母数字字符、连字符、句号和下划线。
   - 描述——输入可选描述。
5. 对于数据提供程序，请选择要用作路由计算器的数据提供程序。
6. （可选）在 Tags (标签) 下，输入标签 Key (键) 和 Value (值)。这会为您的新路由计算器资源添加标签。有关更多信息，请参阅标记资源。
7. 选择创建路由计算器。

API

使用 Amazon Location API 创建路由计算器资源

使用 Amazon Location 地点 API 中的 `CreateRouteCalculator` 操作。

以下示例是*ExampleCalculator*使用数据提供程序 *Esri ############ API* 请求。

```
POST /routes/v0/calculators
Content-type: application/json

{
   "CalculatorName": "ExampleCalculator",
   "DataSource": "Esri",
   "Description": "string",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

使用 AWS CLI 命令创建路由计算器资源

使用 `create-route-calculator` 命令。

以下示例*ExampleCalculator*使用 *Esri* 作为数据提供器创建名为的路径计算器资源。

```
aws location \
  create-route-calculator \
  --calculator-name "ExampleCalculator" \
  --data-source "Esri" \
  --tags Tag1=Value1
```

> **Note**
>
> 计费取决于您的使用情况。您可能会因为使用其他 AWS 服务而产生费用。想要了解更多信息，请参阅 Amazon Location Service 定价。

对您的请求进行身份验证

创建路由计算器资源并准备好开始在应用程序中构建位置功能后，请选择如何对请求进行身份验证：

- 要探索访问服务的方式，请参阅 Accessing Amazon Location Service。
- 如果您的网站有匿名用户，则可能需要使用 API 密钥或 Amazon Cognito。

示例

以下示例演示如何使用 API 密钥进行授权、使用 AWS JavaScript SDK v3 和亚马逊地点JavaScript身份验证助手。

```
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);
```

计算路线

您可以使用 Amazon Location Service 计算出发地和目的地之间的路线，沿途最多有 23 个路径点，以适应不同的交通方式、避让和交通状况。

> **Note**
>
> 您必须先创建路由计算器资源，并为向 Amazon Location 发出的请求设置身份验证。有关更多信息，请参阅 先决条件。

开始计算路线

使用 `CalculateRoute` 操作提交一个简单的请求。一个简单的请求包含以下必填字段：

- `DeparturePosition`——计算路线的起始位置。定义为 `[longitude, latitude]`
- `DestinationPosition`——计算路线的终点位置。定义为 `[longitude, latitude]`。

> **Note**
>
> 如果您指定的出发或目的地位置不在道路上，Amazon Location 会将该位置移动到最近的道路。

您可以选择在请求中指定路径点、出发时间和出行模式。

您可以使用 AWS CLI 或 Amazon Location API。

API

以下示例是使用路径计算器资源的CalculateRoute请求*ExampleCalculator*。该请求指定计算从出发位置 [*-122.7565*, *49.0021*] 到目的地位置 [*-122.3394*, *47.6159*] 的路线。

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159]
}
```

AWS CLI

以下示例是使用路径计算器资源的calculate-route命令*ExampleCalculator*。该请求指定计算从出发位置 [*-122.7565*, *49.0021*] 到目的地位置 [*-122.3394*, *47.6159*] 的路线。

```
aws location \
    calculate-route \
        --calculator-name ExampleCalculator \
        --departure-position -122.7565 49.0021 \
        --destination-position -122.3394 47.6159
```

默认情况下，响应以每公里 Distance 为单位返回。您可以使用以下可选参数将计量单位更改为英里：

- DistanceUnit——指定用于距离结果的单位制。

Example

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "DistanceUnit": "Miles"
```

```
}
```

设置路径点

计算路线时，您可以使用路径点位置在出发位置和目的地位置之间指定最多 23 个中间中途停留点。

- `WaypointPositions`——指定要在出发位置和目的地位置之间的路线上包含的中间位置的有序列表。

> **Note**
>
> 如果您指定的路径点位置不在道路上，Amazon Location 会将该位置移动到最近的道路。

Example

以下 `CalculateRoute` 请求计算出具有 2 个路径点的路线：

- 出发位置是 [-122.7565, 49.0021]，目的地位置是 [-122.3394, 47.6159]。
- 对于请求参数 `WaypointPositions`：
  - 第一个途经点的位置是 [*-122.1884, 48.0936*]。
  - 第二个途经点的位置是 [*-122.3493, 47.6205*]。
- *################################ true#*
  - `IncludeLegGeometry`——包括响应中一对位置之间每条路线的几何形状。

```
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
}
```

响应示例

以下是一个请求示例，当从 Amazon Location Routes API 调用 `CalculateRoute` 操作时，`IncludeLegGeometry` 设置为 *`true`* 时会得到相应的响应，其中包括响应中一对位置之间每条路线的线串几何形状。

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
    ]
   }
}
```

使用路由矩阵进行路线规划

您可以使用 Amazon Location Service 为路线规划和优化软件创建输入。您可以为一组出发位置和一组目的地位置之间的路线创建路线结果，包括行驶时间和行驶距离。

例如，给定出发位置 A 和 B 以及目的地位置 X 和 Y，Amazon Location Service 将返回从 A 到 X、A 到 Y、B 到 X 以及 B 到 Y 的路线的行驶时间和行驶距离。

您可以计算具有不同交通方式、避让点和交通状况的路线。例如，您可以指定车辆是一辆长 35 英尺的卡车，计算的路线将使用这些限制来确定行驶时间和行驶距离。

返回的结果（和计算的路线）数量等于出发位置数乘以目的地位置的数量。您需要为计算的每条路线付费，而不是为每次服务请求付费，因此，包含 10 个出发地和 10 个目的地的路由矩阵将按照 100 条路线计费。

计算路由矩阵

您可以计算一组出发位置和一组目的地位置之间的路由矩阵。路线结果将包括行驶时间和行驶距离。

先决条件

- 您必须先创建路由计算器资源，并为向 Amazon Location 发出的请求设置身份验证。有关更多信息，请参阅 先决条件。

使用 `CalculateRouteMatrix` 操作提交请求。最低请求包含以下必填字段：

- `DeparturePositions`——要计算路线的起始位置集。定义为一个数组 `[longitude, latitude]`

- `DestinationPositions`——要计算路线的终点位置集。定义为一个数组 `[longitude, latitude]`。

> **Note**
>
> 如果您指定的出发或目的地位置不在道路上，Amazon Location 会将该位置移动到最近的道路。

您可以选择在请求中指定出发时间和出行模式。

您可以使用 AWS CLI 或 Amazon Location API。

API

以下示例是使用路径计算器资源的`CalculateRouteMatrix`请求*`ExampleCalculator`*。该请求指定计算从出发位置 [*-122.7565*, *49.0021*] 和 [*-122.2014*, *47.6101*] 到目的地位置 [*-122.3394*, *47.6159*] 和 [*-122.4813*, *48.7511*] 的路由矩阵。

```
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

以下示例是使用路径计算器资源的`calculate-route-matrix`命令*`ExampleCalculator`*。该请求指定计算从出发位置 [*-122.7565*, *49.0021*] 和 [*-122.2014*, *47.6101*] 到目的地位置 [*-122.3394*, *47.6159*] 和 [*-122.4813*, *48.7511*] 的路由矩阵。

```
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]]"
```

默认情况下，响应以每公里 Distance 为单位返回。您可以使用以下可选参数将计量单位更改为英里：

- DistanceUnit——指定用于距离结果的单位制。

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"
}
```

对出发地和目的地位置的限制

计算路由矩阵时，对出发和目的地位置有限制。这些限制因 RouteCalculator 资源使用的提供程序而异。

限制	Esri	Grab	HERE
位置数量	最多 10 个出发位置和 10 个目的地位置。	多达 350 个出发位置和 350 个目的地位置。	多达 350 个出发位置和 350 个目的地位置。 对于较长的路线，则适用其他限制。请参阅 部分。

限制	Esri	Grab	HERE
位置之间的距离	任何一对出发和目的地位置之间的距离必须在 400 千米以内（步行路线为 40 千米）。		所有出发和目的地位置必须位于直径为 180 千米的圆圈内。 对于较长的路线，则适用其他限制。请参阅 部分。
路线长度	如果路线的总行驶时间超过 400 分钟，则该路线将无法完成。		在出发点和目的地点周围的圆圈外偏离超过 10 千米的路线将不予计算。 对于较长的路线，则适用其他限制。请参阅 部分。
区域	韩国不支持计算路由矩阵。	在东南亚有售。有关支持的国家/地区列表和更多信息，请参阅覆盖的国家/地区和区域。	没有其他限制。

较长的路线规划

计算路线结果矩阵对于高效的路线规划很有用，但计算可能需要一些时间。所有 Amazon Location Service 数据提供程序都对可以计算的路线数量或路线距离施加了限制。例如，HERE 允许在 350 个出发和目的地位置之间创建路线，但这些位置必须在 180 千米的圆圈内。如果您想计划更长的路线，该怎么办？

您可以使用带有 HERE 的 `RouteCalculator` 作为数据提供程序，为较少数量的路径计算长度不受限制的路由矩阵。这不会改变您调用 `CalculateRouteMatrix` API 的方式，Amazon Location 只是在您满足要求时允许更长的路线。

计算更长路线的要求是：

- `RouteCalculator` 必须使用 HERE 数据提供程序。

- 出发位置的数量不得大于 15。
- 要计算的路线总数不得大于 100。
- 当路线大于 1,000 km 时，不允许进行长途路由，避免通行费的卡车路线。 这种组合的计算速度较慢，并且可能导致呼叫超时。您可以通过CalculateRoute操作单独计算这些路径。

如果您的调用不符合这些要求（例如，您在一次调用中请求计算 150 条路线），则 `CalculateRouteMatrix` 将恢复为仅允许使用较短的路线规则。然后，只要位置在 180 千米的圆圈内，您就可以计算路线。

计算较长的路线时，请记住以下几点：

- 较长的路线可能需要更长的时间来计算，甚至比 Amazon Location API 的最长时间还要长。如果您在特定路由上频繁超时，则可以在每次调用 `CalculateRouteMatrix` 中尝试少量路线。
- 如果您在 `CalculateRouteMatrix` 请求中添加更多目的地或出发位置，则操作可能会切换到限制性更强的模式，并且当要创建的路线较少时，可以毫无问题地计算出一条路线，则可能会出现错误。在这种情况下，请减少目的地或出发位置的数量，并发出多个请求以获得所需的全套路线计算结果。

响应示例

以下是从 Amazon Location Routes API 调用 `CalculateRouteMatrix` 操作时带有相应响应的示例请求。

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"
    }
}
```


位置不在道路上

使用 CalculateRoute 或 CalculateRouteMatrix 时，如果您指定的出发地、目的地或路径点位置不在道路上，Amazon Location 会将该位置移动到附近的道路。

以下 CalculateRoute 请求指定了不在道路上的出发位置和目的地位置：

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
```

```
{
   "DeparturePosition": [-123.128014, 49.298472],
   "DestinationPosition": [-123.134701, 49. 294315]
}
```

生成的响应会返回一个与附近道路对齐的位置：

```
{
   "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"
   }
}
```


出发时间

默认情况下，当您调用 `CalculateRoute` 或 `CalculateRouteMatrix` 时，如果您未在请求中提供出发时间，则计算出的路线将反映最佳的交通状况。

您可以使用以下选项之一来设置特定的出发时间，以使用所选数据提供程序提供的实时和预测交通状况：

- DepartNow——设置为 *true* 时，它使用实时交通状况来计算最快的行驶路线。
- DepartureTime——如果提供，它将在请求的时间内使用预测和已知的交通状况。按以下格式定义:YYYY-MM-DDThh:mm:ss.sssZ。

Example

以下 CalculateRoute 请求将出发时间设置为世界标准时间 2024 年 7 月 2 日 12:15:20 UTC。

```
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,
}
```

出行模式

使用 CalculateRoute 或 CalculateRouteMatrix 时可以设置出行模式。出行方式会影响行驶速度和道路兼容性。虽然默认的出行模式是开车，但您可以使用以下可选参数指定沿着路线行驶时所使用的出行模式：

- TravelMode——在计算路线时指定传输模式，例如：*Bicycle*、*Car*、*Motorcycle*、*Truck* 或 *Walking*。

限制

- 如果您指定 Walking 为出行模式并且您的数据提供程序为 Esri，则起点和目的地必须在 40 公里以内。
- Bicycle 或 Motorcycle 仅在使用 Grab 作为数据提供程序时可用。

- Grab 仅提供某些城市的 `Bicycle` 和 `Walking` 路线。有关更多信息，请参阅 覆盖的国家/地区和区域。
- `Truck` 使用 Grab 作为数据提供程序时不可用。

其他首选项

如果指定为 *`Car`* 的 `TravelMode`，则可以使用以下可选参数指定其他路线首选项：

- `CarModeOptions`——指定乘车旅行时的路线偏好，例如 *`AvoidFerries`* 或 *`AvoidTolls`*。

如果指定为 *`Truck`* 的 `TravelMode`，则可以使用以下可选参数指定其他路线首选项：

- `TruckModeOptions`——指定乘坐卡车行驶时的路线首选项，例如 *`AvoidFerries`* 或 *`AvoidTolls`*，此外还可指定可以容纳 *`TruckDimensions`* 和 *`TruckWeight`* 路线。

Example

以下 `CalculateRoute` 请求指定 *`Truck`* 为出行方式。其他路线限制包括：避开使用渡轮的路线，避开无法容纳卡车尺寸和重量的道路。

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

管理您的路由计算器资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 来管理您的路由计算器资源。

列出您的路由计算器资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 查看您的路由计算器列表：

Console

使用 Amazon Location 控制台查看路由计算器列表

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 选择左侧导航窗格中的路由计算器。
3. 在我的路由计算器下查看路由计算器的详细信息。

API

使用 Amazon Location 路线 API 中的 `ListRouteCalculators` 操作。

以下示例是一个用于获取 AWS 账户中的路由计算器列表的 API 请求。

```
POST /routes/v0/list-calculators
```

以下为 `ListRouteCalculators` 响应示例：

```
{
       "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。

以下示例是获取 AWS 账户中路由计算器列表的 AWS CLI 示例。

```
aws location list-route-calculators
```

获取路由计算器详情

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 获取有关您 AWS 账户中任何路由计算器资源的详细信息：

Console

使用 Amazon Location 控制台查看路由计算器的详细信息

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 选择左侧导航窗格中的路由计算器。
3. 在我的路由计算器下，选择目标路由计算器的名称链接。

API

使用 Amazon Location 路线 API 中的 `DescribeRouteCalculator` 操作。

以下示例是获取路径计算器详细信息的 API 请求*`ExampleCalculator`*。

```
GET /routes/v0/calculators/ExampleCalculator
```

以下为 `DescribeRouteCalculator` 响应示例：

```
{
       "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。

以下示例是AWS CLI获取路径计算器详细信息的示例*ExampleCalculator*。

```
aws location describe-route-calculator \
        --calculator-name "ExampleCalculator"
```

删除路由计算器

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 从 AWS 账户中删除路由计算器：

Console

使用 Amazon Location 控制台删除路由计算器

> **Warning**
> 此操作将永久删除资源。

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 选择左侧导航窗格中的路由计算器。
3. 在我的路由计算器下，选择目标路由计算器。
4. 选择删除路由计算器。

API

使用 Amazon Location 路线 API 中的 `DeleteRouteCalculator` 操作。

以下示例是删除地理围栏集合的 API 请求。*ExampleCalculator*

```
DELETE /routes/v0/calculators/ExampleCalculator
```

以下为 DeleteRouteCalculator 响应示例：

```
HTTP/1.1 200
```

CLI

使用 delete-route-calculator 命令。

以下示例是删除 geofence 集合的AWS CLI命令。*ExampleCalculator*

```
aws location delete-route-calculator \
        --calculator-name "ExampleCalculator"
```

使用 Amazon Location 对感兴趣的区域进行地理围栏

地理围栏应用程序评估被跟踪设备相对于先前注册的感兴趣区域的位置。这使您可以根据位置更新采取行动。例如，当通过移动应用程序订购咖啡的客户在商店附近时，您可以启动一个活动来提示通知。

> **Note**
>
> 有关地理围栏和跟踪器概念的概述，请参阅 地理围栏和跟踪器。

本指南的这一部分提供了使用 Amazon Location Service 创建地理围栏应用程序的 step-by-step 说明。

步骤概述

1. 在感兴趣的区域周围添加地理围栏，并将其存储在地理围栏集合资源中。
2. 开始跟踪您的目标设备，并将设备位置记录存储在跟踪器资源中。
3. 将您的跟踪器资源链接到您的地理围栏集合资源，以便根据您的所有地理围栏自动评估设备位置更新。
4. 如果您不想使用 Amazon Location 跟踪器来保存设备的位置记录，则可以直接根据地理围栏收集资源评估设备位置。

实施地理围栏解决方案后，您的地理围栏集合资源会发出以下事件：

- ENTER——被跟踪的设备进入地理围栏集合中的地理围栏。
- EXIT——被跟踪的设备退出地理围栏集合中的地理围栏。

您可以使用 Amazon 通过 EventBridge 将事件路由到其他地方来对事件做出反应。

除了通过 Amazon Location Service APIs 从每台设备发送更新之外，您还可以使用MQTT发送设备更新。

以下主题详细描述了这些步骤和备选方案。

主题

添加地理围栏

地理围栏包含形成封闭边界的点和顶点，封闭边界定义了感兴趣区域。地理围栏集合存储并管理一个或多个地理围栏。

Amazon Location 地理围栏集合存储使用名为 Geo (7946) 的标准地理空间数据格式定义的地理围栏。JSON RFC您可以免费使用诸如 geo json.io 之类的工具来以图形方式绘制地理围栏并保存输出的 Geo 文件。JSON

> **Note**
>
> Amazon Location 不支持带有孔洞的多边形、多面、顺时针多边形和穿过对向子午线的地理围栏。

创建地理围栏集合

使用 Amazon Location 控制台、 AWS CLI或 Amazon Location 创建用于存储和管理地理围栏的 geofence 集合。APIs

Console

使用 Amazon Location 控制台创建地理围栏集合

1. 打开 Amazon Location Service 控制台，网址为https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择地理围栏集合。
3. 选择创建地理围栏集合。
4. 填写以下选框：
   - 名称——输入唯一名称。例如，*ExampleGeofenceCollection*。 最多 100 个字符。有效条目包括：字母数字字符、连字符、句号和下划线。
   - 描述——输入可选描述以区分您的资源。
5. 在以目标 CloudWatch 为目标的EventBridge 规则下，您可以创建一条可选 EventBridge 规则来开始对地理围栏事件做出反应。这样，Amazon Location 就可以将事件发布到亚马逊 CloudWatch 日志。
6. （可选）在 Tags (标签) 下，输入标签 Key (键) 和 Value (值)。这会为您的新地理围栏集合添加一个标签。有关更多信息，请参阅 为您的 Amazon Location Service 资源添加标签。
7. （可选）在“客户托管密钥加密”下，您可以选择添加客户托管密钥。这会添加一个对称的客户托管密钥，您可以通过默认 AWS 拥有的加密来创建、拥有和管理该密钥。想要了解更多信息，请参阅加密静态数据。
8. 选择创建地理围栏集合。

API

使用 Amazon 位置创建地理围栏收藏 APIs

使用 Amazon 定位地理围APIs栏中的`CreateGeofenceCollection`操作。

以下示例使用API请求创建名为 geofence 集合 *ExampleGeofenceCollection*。 Geofence 集合与客户管理的 AWS KMS 密钥相关联，用于加密客户数据。

```
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 AWS CLI 命令创建地理围栏集合

使用 create-geofence-collection 命令。

以下示例使用创建名 AWS CLI 为的地理围栏集合 *ExampleGeofenceCollection*。 Geofence 集合与客户管理的 AWS KMS 密钥相关联，用于加密客户数据。

```
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**
>
> 计费取决于您的使用情况。您可能会因为使用其他 AWS 服务而产生费用。想要了解更多信息，请参阅 Amazon Location Service 定价。

绘制地理围栏

现在，您已经创建了地理围栏集合，可以定义您的地理围栏了。地理围栏既可以定义为多边形，也可以定义为圆形。要绘制多边形地理围栏，你可以使用地理JSON编辑工具，例如 geojson.io。

要将地理围栏创建为圆形，必须定义圆的中心点和半径。例如，如果您要创建一个地理围栏，以便在设备距离特定位置 50 米以内时收到通知，则应使用该位置的纬度和经度并将半径指定为 50 米。

使用 Amazon Location ServiceAPIs，您还可以以键值对的形式向地理围栏添加元数据。它们可用于存储有关地理围栏的信息（例如地理围栏的类型）或其他特定于您的应用程序的信息。您可以在以下 通过亚马逊对亚马逊定位服务事件做出反应 EventBridge 情况下使用此元数据。

添加多边形地理围栏

本部分介绍创建多边形地理围栏的情况

使用地理工具绘制地理围栏 JSON

现在你已经创建了地理围栏集合，你可以使用地理JSON编辑工具（例如 geojson.io）来定义地理围栏。

创建 Geo JSON 文件

1. 打开地理JSON编辑工具。例如，geojson.io。
2. 选择绘制多边形图标并绘制感兴趣的区域。
3. 选择 “保存”，然后JSON从下拉菜单中选择 “地理位置”。

将地理围栏放入JSON地理围栏集合

您可以使用生成的地理JSON文件通过 Amazon Location Service 控制台 AWS CLI、或亚马逊位置上传您的地理围栏：APIs

Console

使用 Amazon Location Service 控制台向地理围栏集合添加地理围栏

1. 打开 Amazon Location Service 控制台，网址为https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择地理围栏集合。
3. 从地理围栏集合列表中，选择目标地理围栏集合的名称链接。
4. 在地理围栏下，选择创建地理围栏。
5. 在添加地理围栏窗口中，将您的 Geo 拖放JSON到窗口中。
6. 选择添加地理围栏。

API

使用 Amazon 位置添加地理围栏 APIs

使用 Amazon 定位地理围APIs栏中的PutGeofence操作。

以下示例使用API请求在给定 ID 的情况下添加地理围栏 *GEOFENCE-EXAMPLE1* 到一个名为的地理围栏集合 *ExampleGeofenceCollection*。 它还指定了带有键Type和值的单个 geofence 元数据属性。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 BatchPutGeofence 操作添加多个地理围栏。

```
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

使用命令向地理围栏集合添加地理围栏 AWS CLI

使用 put-geofence 命令。

以下示例使用 AWS CLI 向名为的地理围栏集合添加地理围栏 *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"
    }
```

添加圆形地理围栏

本部分介绍如何创建圆形地理围栏。您必须知道要作为圆心的点的纬度和经度，以及圆的半径（以米为单位）。您可以使用Amazon位置APIs或创建圆形地理围栏. AWS CLI

API

使用 Amazon 位置添加圆形地理围栏 APIs

使用 Amazon 定位地理围APIs栏中的PutGeofence操作。

以下示例使用API请求在给定 ID 的情况下添加地理围栏 *GEOFENCE-EXAMPLE2* 到一个名为的地理围栏集合 *ExampleGeofenceCollection*:

```
PUT /geofencing/v0/collections/ExampleGeofenceCollection/geofence/GEOFENCE-EXAMPLE2
    Content-type: application/json

    {
       "Geometry": {
          "Circle": {
            "Center": [-5.716667, -15.933333],
            "Radius": 50
          }
       }
    }
```

AWS CLI

使用命令向地理围栏集合添加圆形地理围栏 AWS CLI

使用 put-geofence 命令。

以下示例使用 AWS CLI 向名为的地理围栏集合添加地理围栏 *ExampleGeofenceCollection*.

```
$ aws location \
        put-geofence \
            --collection-name ExampleGeofenceCollection \
            --geofence-id ExampleGeofenceCircle \
            --geometry 'Circle={Center=[-5.716667, -15.933333], Radius=50}'
```

> **Note**
>
> 您也可以将复杂几何体放JSON入其自己的文件中，如以下示例所示。
>
> ```
> $ aws location \
>         put-geofence \
>             --collection-name ExampleGeofenceCollection \
>             --geofence-id ExampleGeofenceCircle \
>             --geometry file:circle.json
> ```
>
> 在示例中，circle.json 文件包含圆形几何JSON图形。
>
> ```
> {
>     "Circle": {
>         "Center": [-74.006975, 40.717127],
> ```

```
        "Radius": 287.7897969218057
    }
}
```

开启跟踪

本部分将指导您构建用于捕获设备位置的跟踪应用程序。

创建跟踪器

创建跟踪器资源以存储和处理来自您设备的位置更新。您可以使用 Amazon Location Service 控制台 AWS CLI、或亚马逊位置APIs。

存储在您的跟踪器资源中的每个位置更新都可能包括一个位置精度指标，以及最多三个与您要存储的位置或设备相关的元数据字段。元数据以键值对的形式存储，可以存储速度、方向、轮胎压力或发动机温度等信息。

跟踪器在收到的位置更新时对其进行过滤。这可以减少设备路径中的视觉噪音（称为抖动），并减少错误的地理围栏进入和退出事件的数量。这还可以减少启动的地理围栏评估的数量，从而帮助管理成本。

跟踪器提供三种位置筛选选项，以帮助管理成本并减少位置更新中的抖动。

- 基于精度——与任何提供精度测量的设备一起使用。大多数移动设备都提供此信息。每次位置测量的准确性受许多环境因素的影响，包括GPS卫星接收、景观以及 Wi-Fi 和蓝牙设备的距离。大多数设备，包括大多数移动设备，都可以提供测量精度的估计值以及测量结果。通过 `AccuracyBased` 筛选，如果设备移动小于测量的精度，Amazon Location 将忽略位置更新。例如，如果一个设备两次连续更新的精度范围分别为 5 米和 10 米，则该设备的移动距离小于 15 米时 Amazon Location 忽略第二次更新。Amazon Location 既不会根据地理围栏评估被忽略的更新，也不会存储这些更新。

  如果未提供精度，则将其视为零，并且测量结果被视为完全准确。

  > **Note**
  >
  > 您也可以使用基于精度的筛选来移除所有筛选。如果您选择基于精度的筛选，但将所有精度数据覆盖为零，或者完全省略准确性，那么 Amazon Location 将不会筛选出任何更新。

- 基于距离——当您的设备不提供精度测量值，但您仍希望利用筛选功能来减少抖动并管理成本时使用。`DistanceBased` 筛选忽略设备移动小于 30 米（98.4 英尺）的位置更新。当您使用

`DistanceBased` 位置筛选时，Amazon Location 既不会根据地理围栏评估这些被忽略的更新，也不会存储更新。

大多数移动设备的精度，包括 iOS 和 Android 设备的平均精度，都在 15 米以内。在大多数应用程序中，`DistanceBased` 筛选可以减少在地图上显示设备轨迹时位置不准确的影响，以及设备靠近地理围栏边界时连续多次进入和退出事件的反弹效果。它还可以减少对照链接的地理围栏进行评估或检索设备位置的调用，从而帮助降低应用程序的成本。

- 基于时间——（默认）当您的设备非常频繁地发送位置更新（每 30 秒钟超过一次），并且您希望在不存储每个更新的情况下实现近乎实时的地理围栏评估时使用。在 `TimeBased` 筛选中，每个位置更新根据链接的地理围栏集合进行评估，但并非每个位置更新都会存储。如果更新频率超过 30 秒，则每 30 秒仅为每个唯一的设备 ID 存储一次更新。

> **Note**
>
> 在决定筛选方法和位置更新的频率时，请注意追踪应用程序的成本。您需要为每次位置更新付费，并根据每个链接的地理围栏集合评估位置更新一次付费。例如，使用基于时间的筛选时，如果您的跟踪链接到两个地理围栏集合，则每次位置更新都将计为一个位置更新请求和两次地理围栏收集评估。如果您报告设备每 5 秒更新一次位置并使用基于时间的筛选，则每台设备将按每小时 720 次位置更新和 1,440 次地理围栏评估计费。
> 您的计费不受每个集合中地理围栏数量的影响。由于每个地理围栏集合可能包含多达 50,000 个地理围栏，因此您可能需要尽可能将地理围栏合并成更少的集合，以降低地理围栏评估的成本。

默认情况下，每次被跟踪的设备进入或退出关联的地理围栏时，您都会收到 EventBridge 事件。有关更多信息，请参阅 将跟踪器关联到地理围栏集合。

您可以为跟踪器资源的所有筛选位置更新启用事件。有关更多信息，请参阅 为跟踪器启用更新事件。

> **Note**
>
> 如果您希望使用自己的 AWS KMS 客户托管密钥加密数据，则默认情况下将禁用 “边界多边形查询” 功能。这是因为使用此边界多边形查询功能，您的设备位置表示不会使用您的 AWS KMS 托管密钥进行加密。但是，设备的确切位置仍使用您的托管密钥进行加密。
> 在创建或更新跟踪器时，您可以通过将 `KmsKeyEnableGeospatialQueries` 参数设置为 true 来选择使用“边界多边形查询”功能。

Console

使用 Amazon Location 控制台创建跟踪器

1. 打开 Amazon Location Service 控制台，网址为https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择跟踪器。
3. 选择创建跟踪器。
4. 填写以下字段：
   - 名称——输入唯一名称。例如，*ExampleTracker*。 最多 100 个字符。有效条目包括：字母数字字符、连字符、句号和下划线。
   - 描述——输入可选描述。
5. 在位置筛选下，选择最适合您打算如何使用跟踪器资源的选项。如果未设置位置筛选，则默认设置为 `TimeBased`。有关更多信息，请参阅跟踪器本指南和`PositionFiltering`《Amazon Location Service 追踪器API参考》。
6. （可选）在 Tags (标签) 下，输入标签 Key (键) 和 Value (值)。这会为您的新地理围栏集合添加一个标签。有关更多信息，请参阅标记资源。
7. （可选）在“客户托管密钥加密”下，您可以选择添加客户托管密钥。这会添加一个对称的客户托管密钥，您可以通过默认 AWS 拥有的加密来创建、拥有和管理该密钥。想要了解更多信息，请参阅加密静态数据。
8. （可选）在下方 KmsKeyEnableGeospatialQueries，您可以选择启用地理空间查询。这允许您使用边界多边形查询功能，同时使用客户AWSKMS管理的密钥对数据进行加密。

   > **Note**
   >
   > 当您使用“边界多边形查询”功能时，您的设备位置表示不会使用您的 AWS KMS 托管密钥进行加密。但是，设备的确切位置仍使用您的托管密钥进行加密。

9. （可选）在EventBridge 配置下，您可以选择为筛选的职位更新启用 EventBridge 事件。每当该跟踪器中设备的位置更新符合位置筛选评估时，都会发送一个事件。
10. 选择创建跟踪器。

API

使用 Amazon 位置创建追踪器 APIs

使用 Amazon 位置追踪器APIs中的`CreateTracker`操作。

以下示例使用API请求创建名为的跟踪器 *ExampleTracker*。 跟踪器资源与客户管理的 AWS KMS 密钥相关联，用于加密客户数据，并且不在中启用位置更新 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"
   }
}
```

在 **KmsKeyEnableGeospatialQueries** 启用状态下创建跟踪器

以下示例将参数 KmsKeyEnableGeospatialQueries 设置为 true。这允许您使用边界多边形查询功能，同时使用客户 AWS KMS 管理的密钥对数据进行加密。

有关使用“边界面查询”功能的信息，请参阅 ???

> **Note**
>
> 当您使用边界多边形查询功能时，您的设备位置表示不会使用您的 AWS KMS 托管密钥进行加密。但是，设备的确切位置仍使用您的托管密钥进行加密。

```
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 命令创建跟踪器

使用 create-tracker 命令。

以下示例使用创建名 AWS CLI 为的跟踪器 *ExampleTracker*。 跟踪器资源与客户管理的 AWS KMS 密钥相关联，用于加密客户数据，并且不在中启用位置更新 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"
```

在 **KmsKeyEnableGeospatialQueries** 启用状态下创建跟踪器

以下示例将参数 KmsKeyEnableGeospatialQueries 设置为 true。这允许您使用边界多边形查询功能，同时使用客户 AWS KMS 管理的密钥对数据进行加密。

有关使用“边界面查询”功能的信息，请参阅 ???

> **Note**
>
> 当您使用边界多边形查询功能时，您的设备位置表示不会使用您的 AWS KMS 托管密钥进行加密。但是，设备的确切位置仍使用您的托管密钥进行加密。

```
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**
>
> 计费取决于您的使用情况。您可能会因为使用其他 AWS 服务而产生费用。想要了解更多信息，请参阅 Amazon Location Service 定价。

创建跟踪链接后，您可以通过选择编辑跟踪链接来编辑描述、位置筛选和EventBridge 配置。

对您的请求进行身份验证

创建跟踪器资源并准备好开始根据地理围栏评估设备位置后，请选择如何对请求进行身份验证：

- 要探索访问服务的方式，请参阅 Accessing Amazon Location Service。
- 如果您想通过未经身份验证的请求发布设备位置，则可能需要使用 Amazon Cognito。

  示例

  以下示例显示了使用 Amazon Cognito 身份池进行授权、使用 AWS JavaScript SDKv3 和亚马逊位置。JavaScript 身份验证助手

```
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);
```

使用设备位置更新跟踪器

要追踪您的设备，您可以将设备位置更新发布到跟踪器。您稍后可以从您的跟踪器资源中检索这些设备位置或设备位置历史记录。

每次位置更新都必须包括设备 ID、时间戳和位置。您可以选择添加其他元数据，包括精度和最多 3 个键值对供您自己使用。

如果您的跟踪器关联到一个或多个地理围栏集合，则系统将根据这些地理围栏对更新进行评估（遵循您为跟踪器指定的过滤规则）。如果设备突破了地理围栏区域（从该区域内部移动到外部，反之亦然），您将在其中收到事件。 EventBridge这些 ENTER 或 EXIT 事件包括位置更新详细信息，包括设备 ID、时间戳和任何相关的元数据。

> **Note**
>
> 有关位置筛选的更多信息，请参阅 创建跟踪器。
> 有关地理围栏事件的更多信息，请参阅 通过亚马逊对亚马逊定位服务事件做出反应 EventBridge。

使用以下任一方法发送设备更新：

- 向 AWS IoT Core 资源@@ 发送MQTT更新并将其链接到您的跟踪器资源。
- 使用亚马逊位置追踪器API AWS CLI、或亚马逊位置信息发送位置APIs更新。您可以使用APIs从 iOS 或 Android 应用程序中调用。AWS SDKs

API

使用 Amazon 位置发送职位更新 APIs

使用 Amazon 位置追踪器APIs中的BatchUpdateDevicePosition操作。

以下示例使用API请求发布的设备位置更新 *ExampleDevice* 到追踪器 *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 AWS CLI 命令发送位置更新

使用 batch-update-device-position 命令。

以下示例使用发布 AWS CLI 的设备位置更新 *ExampleDevice-1* 以及 *ExampleDevice-2* 到追踪器 *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"}}]'
```

通过跟踪器获取设备的位置记录

您的 Amazon Location 跟踪器资源会将所有被追踪设备的位置记录保存 30 天。您可以从跟踪器资源中检索设备位置记录，包括所有关联的元数据。以下示例使用或 AWS CLI Amazon 地点APIs。

API

使用 Amazon 位置从追踪器获取设备位置记录 APIs

使用 Amazon 位置追踪器APIs中的GetDevicePositionHistory操作。

以下示例使用APIURI请求获取的设备位置历史记录 *ExampleDevice* 来自名为的追踪器 *ExampleTracker* 从19:05:07（含）开始，到19:20:07（不包括）结束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 命令从追踪器获取设备位置记录

使用 get-device-position-history 命令。

以下示例使用获取 AWS CLI 的设备位置历史记录 *ExampleDevice* 来自名为的追踪器 *ExampleTracker* 从19:05:07（含）开始，到19:20:07（不包括）结束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"
```

列出您的设备位置

您可以使用或 Amazon 位置来查看追踪器的设备位置APIs列表 ListDevicePositions API。 AWS CLI当您调用时 ListDevicePositions API，系统会返回与给定跟踪器关联的所有设备的最新位置列表。默认情

况下，这API会在给定跟踪链接的每页结果中返回 100 个最新的设备位置。要仅返回特定区域内的设备，请使用 `FilterGeometry` 参数创建“边界多边形查询”。这样，当你调用时 ListDevicePositions，只会返回多边形内的设备。

> **Note**
>
> 如果您希望使用自己的 AWS KMS 客户托管密钥加密数据，则默认情况下将禁用 “边界多边形查询” 功能。这是因为使用此功能，您的设备位置表示不会使用您的 AWS KMS 托管密钥进行加密。但是，设备的确切位置仍使用您的托管密钥进行加密。
> 您可以选择加入“边界多边形查询”功能。这是通过在创建或更新跟踪器时将 `KmsKeyEnableGeospatialQueries` 参数设置为 true 来完成的。

API

使用 Amazon 位置追踪器APIs中的`ListDevicePositions`操作。

以下示例是使用可选参数API请求获取多边形区域中的设备位置列表。`FilterGeometry`该示例返回 `Polygon` 数组定义的区域中存在的 3 个设备位置。

```
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"
}
```

以下为 ListDevicePositions 响应示例：

```
{
    "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。

以下示例是 AWS CLI 获取多边形区域中的设备列表的示例。

```
aws location list-device-positions TODO: add arguments add props for filter geo
```

将跟踪器关联到地理围栏集合

现在你已经有了地理围栏集合和跟踪器，你可以将它们链接在一起，这样就可以根据你的所有地理围栏自动评估位置更新。如果您不想评估所有位置更新，或者如果您不想将某些位置存储在跟踪器资源中，则可以按需根据地理围栏评估设备位置。

根据地理围栏评估设备位置时，会生成事件。您可以为这些事件设置操作。有关您可以为地理围栏事件设置的操作的更多信息，请参阅通过亚马逊对亚马逊定位服务事件做出反应。 EventBridge

Amazon Location 事件包括生成该事件的设备位置更新的属性以及进入或退出的地理围栏的一些属性。有关地理围栏事件中包含的数据的更多信息，请参阅 Amazon Location Service 的亚马逊 EventBridge 活动示例。

以下示例使用控制台、或 Amazon 位置将跟踪器资源链接到地理围栏集合。 AWS CLI APIs

Console

使用 Amazon Location Service 控制台将跟踪器资源关联到地理围栏集合

1. 打开 Amazon Location Service 控制台，网址为https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择跟踪器。

3. 在设备跟踪器下，选择目标跟踪器的名称链接。
4. 在链接的地理围栏集合下，选择链接地理围栏集合。
5. 在链接的地理围栏集合窗口中，从下拉菜单中选择一个地理围栏集合。
6. 选择关联。

在您关联跟踪器资源后，它将处于活动状态。

API

使用 Amazon 位置将追踪器资源关联到地理围栏集合 APIs

使用 Amazon 位置追踪器APIs中的AsssociateTrackerConsumer操作。

以下示例使用关联的API请求 *ExampleTracker* 其地理围栏集合使用其 A mazon 资源名称 () ARN。

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json

{
   "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection"
}
```

AWS CLI

使用命令将跟踪器资源链接到地理围栏集合 AWS CLI

使用 associate-tracker-consumer 命令。

以下示例使用创建名 AWS CLI 为的地理围栏集合 *ExampleGeofenceCollection*.

```
aws location \
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection" \
        --tracker-name "ExampleTracker"
```

根据地理围栏评估设备位置

有两种方法可以根据地理围栏评估位置以生成地理围栏事件：

- 您可以关联跟踪器和地理围栏集合。想要了解更多信息，请参阅 将跟踪器关联到地理围栏集合 部分。
- 您可以使用 geofence 收集资源直接请求来评估一个或多个位置。BatchEvaluateGeofencesAPI

此外，您还可以预测设备在地理围栏内进入、退出或保持闲置状态的传入地理围栏事件。使用ForecastGeofenceEventsAPI来预测事件。

如果您还想跟踪设备位置记录或在地图上显示位置，请将跟踪器链接到地理围栏集合。或者，您可能不想评估所有位置更新，或者您不打算将位置数据存储在跟踪器资源中。如果是其中任何一种情况，则可以直接向地理围栏集合发出请求，并根据其地理围栏评估一个或多个设备的位置。

根据地理围栏评估设备位置会生成事件。您可以对这些事件做出反应并将它们路由到其他 AWS 服务。有关接收地理围栏事件时可以采取的操作的更多信息，请参阅通过亚马逊对亚马逊定位服务事件做出反应。 EventBridge

Amazon Location 事件包括生成该事件的设备位置更新的属性，包括时间、位置、准确性和键值元数据，以及进入或退出的地理围栏的一些属性。有关地理围栏事件中包含的数据的更多信息，请参阅 Amazon Location Service 的亚马逊 EventBridge 活动示例。

以下示例使用或 AWS CLI Amazon 地点APIs。

API

使用 Amazon 位置根据地理围栏的位置评估设备位置 APIs

使用 Amazon 定位地理围APIs栏中的BatchEvaluateGeofences操作。

以下示例使用API请求来评估设备的位置 *ExampleDevice* 到关联的地理围栏集合 *ExampleGeofenceCollection*。 将这些值替换为您自己的地理围栏和设备。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

使用命令根据地理围栏的位置评估设备位置 AWS CLI

使用 batch-evaluate-geofences 命令。

以下示例使用 AWS CLI 来评估的位置 *ExampleDevice* 反对关联的地理围栏集合 *ExampleGeofenceCollection*。 将这些值替换为您自己的地理围栏和设备。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"}}]'
```

根据地理围栏评估设备位置会生成事件。传统上，你可以使用对事件做出反应 Amazon EventBridge，但是这个过程只允许你在事件发生之后对事件做出反应。如果您需要预测设备何时进入或退出地理围栏，例如设备是否越过边界，因此将受到不同的法规的约束，那么您可以使用来ForecastGeofenceEventsAPI预测未来的地理围栏事件。

ForecastGeofenceEventsAPI使用设备 time-to-breach、距离、速度和位置等标准来预测事件。API将返回 a `ForecastedBreachTime`，表示地理围栏事件发生的估计时间。

以下示例使用了 Amazon 地点APIs。

API

使用 Amazon 位置预测地理围栏事件 APIs

使用 Amazon 定位地理围APIs栏中的 ForecastGeofenceEvents操作。

以下示例使用API请求来预测地理围栏事件 *ExampleDevice* 相对于 *ExampleGeofence*。 将这些值替换为您自己的地理围栏和设备。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
}
```

验证设备位置

要检查设备位置的完整性，请使用VerifyDevicePositionAPI。这将通过评估诸如设备的手机信号、Wi-Fi 接入点、IPv4 地址以及是否正在使用代理等属性来API返回有关设备位置完整性的信息。

先决条件

在能够使用列APIs出的设备验证之前，请确保您具备以下先决条件：

- 您已经为要查看的一个或多个设备创建了跟踪器。有关更多信息，请参阅 开启跟踪。

以下示例显示了对 Amazon 营业地点的请求VerifyDevicePositionAPI。

API

使用 Amazon 位置验证设备位置 APIs

使用 Amazon 位置追踪中的VerifyDevicePosition操作APIs。

以下示例显示了评估设备位置完整性的API请求。用您自己的设备替换这些值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**
>
> Integrity SDK 提供了与设备验证相关的增强功能，可根据要求使用。要获得访问权限，请联系 S SDK al es Suppor t。

通过亚马逊对亚马逊定位服务事件做出反应 EventBridge

Amazon EventBridge 是一种无服务器事件总线，它使用来自诸如 Amazon Location 之类的AWS服务的数据有效地将应用程序连接在一起。 EventBridge 从 Amazon Location 接收事件并将这些数据路由到目标，例如AWS Lambda。您可以设置路由规则来确定发送数据的目的地，以便构建能够实时响应的应用程序架构。

默认情况下，只有地理围栏EXIT事件（ENTER以及设备进入或离开地理围栏区域时的事件）才会发送到。 EventBridge 您也可以为跟踪器资源启用所有已筛选的位置更新事件。有关更多信息，请参阅 为跟踪器启用更新事件。

有关更多信息，请参阅 Amazon EventBridge 用户指南中的事件和事件模式。

主题

- 为跟踪器启用更新事件
- 为 Amazon Location 创建活动规则
- Amazon Location Service 的亚马逊 EventBridge 活动示例

为跟踪器启用更新事件

默认情况下，Amazon Location 仅ENTER向发送EXIT地理围栏事件。 EventBridge您可以启用所有筛选过的位置UPDATE事件，以便将跟踪器发送到该跟踪器 EventBridge。您可以在创建或更新跟踪器时执行此操作。

例如，要使用更新现有跟踪器AWS CLI，您可以使用以下命令（使用您的跟踪器资源名称代替 *MyTracker*）。

```
aws location update-tracker --tracker-name MyTracker --event-bridge-enabled
```

要关闭跟踪器的位置事件，您必须使用 API 或 Amazon Location Service 控制台。

为 Amazon Location 创建活动规则

您可以为每个事件总线创建多达 300 条规则， EventBridge 以配置为响应 Amazon Location 事件而采取的操作。

例如，您可以为地理围栏事件创建规则，在该规则中，当在地理围栏边界内检测到电话时，将发送推送通知。

为 Amazon Location 事件创建规则

使用以下值创建基于 Amazon 位置事件的 EventBridge 规则：

- 对于规则类型，选择具有事件模式的规则。
- 在事件模式框中，添加以下模式。

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"]
}
```

要创建跟踪器位置更新规则，您可以改用以下模式：

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Device Position Event"]
}
```

您可以选择通过添加 `detail` 标签来仅指定 `ENTER` 或 `EXIT` 事件（如果您的规则是针对跟踪器位置更新，则只有一个 `EventType`，因此无需对其进行筛选）：

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"]
  }
}
```

您也可以选择根据位置的属性或地理围栏进行筛选：

```
{
```

```
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"],
    "GeofenceProperties": {
      "Type": "LoadingDock"
    },
    "PositionProperties": {
      "VehicleType": "Truck"
    }
  }
}
```

- 对于选择目标，选择目标从 Amazon Location Service 收到事件时要执行的目标操作。

  例如，使用 Amazon Simple Notification Service (SNS) 主题在事件发生时发送电子邮件或短信。您首先需要使用 Amazon SNS 控制台创建 Amazon SNS 主题。想要了解更多信息，请参阅使用 Amazon SNS 发送通知。

> **Warning**
>
> 最佳做法是确认事件规则已成功应用，否则您的自动操作可能无法按预期启动。要验证您的事件规则，请为事件规则启动条件。例如，模拟进入地理围栏区域的设备。

您也可以从 Amazon Location 捕获所有事件，只需排除 `detail-type` 部分即可。例如：

```
{
  "source": [
    "aws.geo"
  ]
}
```

> **Note**
>
> 同一事件可以多次交付。您可以使用事件 ID 对收到的事件进行重复数据删除。

Amazon Location Service 的亚马逊 EventBridge 活动示例

以下是通过调用 BatchUpdateDevicePosition 启动的进入地理围栏的事件示例。

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

以下是通过调用 BatchUpdateDevicePosition 启动的退出地理围栏的事件示例。

```
{
```

```
  "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"
    }
  }
}
```

以下是通过调用 BatchUpdateDevicePosition 启动的位置更新事件的示例。

```
{
  "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 Loc MQTT ation Service 进行追踪

MQTT是一种轻量级且被广泛采用的消息传递协议，专为受限的设备而设计。 AWS IoT Core 支持使用MQTT协议和MQTT通过 S WebSocket ecure (WSS) 协议的设备连接。

AWS IoT Core 将设备连接到设备 AWS 并使您能够在它们之间发送和接收消息。 AWS IoT Core 规则引擎存储有关设备消息主题的查询，并允许您定义向其他 AWS 服务（例如 Amazon Location Service）发送消息的操作。知道自己位置为坐标的设备可以通过规则引擎将其位置转发到 Amazon Location 。

> **Note**
>
> 设备可能知道自己的位置，例如通过内置GPS。 AWS IoT 还支持第三方设备位置跟踪。想要了解更多信息，请参阅 AWS IoT Core 开发人员指南中的 AWS IoT Core Device Location。

以下演练介绍了使用 AWS IoT Core 规则进行跟踪。如果您需要在将设备信息发送到 Amazon Loc AWS Lambda ation 之前对其进行处理，也可以将设备信息发送到您自己的部门。有关使用 Lambda 处理您的设备位置的更多详细信息，请参阅 AWS Lambda 与一起使用 MQTT。

主题

先决条件

在您开始跟踪之前，您必须完成以下前提条件：

- 创建跟踪器资源，将设备位置数据发送到该资源。
- 创建IAM角色以授予对跟踪链接的 AWS IoT Core 访问权限。

  执行这些步骤时，请使用以下策略授予对您的跟踪器的访问权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
}
```

创建 AWS IoT Core 规则

接下来，创建一条 AWS IoT Core 规则，将设备的位置遥测数据转发给 Amazon Location Service。有关创建规则的更多信息，请参阅 AWS IoT Core 开发人员指南中的以下主题：

- 为有关@@ 创建新 AWS IoT 规则的信息创建规则。
- 位置操作，用于特定于创建发布到 Amazon Location 的规则的信息

在控制台中测试你的 AWS IoT Core 规则

如果当前没有设备发布包含位置的遥测数据，则可以使用 AWS IoT Core 控制台测试规则。控制台有一个测试客户端，您可以在其中发布一条示例消息来验证解决方案的结果。

1. 登录 AWS IoT Core 控制台，网址为https://console.aws.amazon.com/iot/。
2. 在左侧导航栏中，展开 “测试”，然后选择 “MQTT测试客户端”。
3. 在 “发布到主题” 下，将主题名称设置为 *iot/topic*（或您在 AWS IoT Core 规则中设置的主题的名称，如果不同），并为消息负载提供以下内容。

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. 选择发布到主题来发送测试消息。
5. 要验证 Amazon Location Service 是否已收到该消息，请使用以下 AWS CLI 命令。如果您在设置过程中对其进行了修改，请将跟踪器名称替换为您使用的名称。

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

AWS Lambda 与一起使用 MQTT

虽然在将设备位置数据发送到 Amazon Location 进行跟踪时不再需要使用 AWS Lambda ，但在某些情况下，您可能仍想使用 Lambda。例如，如果您想在将设备位置数据发送到 Amazon Location 之前自己处理设备位置数据。以下主题介绍如何在消息发送到您的跟踪器之前使用 Lambda 处理消息。有关此模式的更多信息，请参阅参考架构。

主题

- 先决条件
- 创建 Lambda 函数
- 创建 AWS IoT Core 规则

- 在控制台中测试你的 AWS IoT Core 规则

先决条件

在开始追踪之前，必须先创建跟踪器资源。要创建追踪器资源，您可以使用亚马逊定位控制台 AWS CLI、或亚马逊位置APIs。

以下示例使用 Amazon Location Service 控制台创建跟踪器资源：

1. 打开 Amazon Location Service 控制台，网址为https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择跟踪器。
3. 选择创建跟踪器。
4. 填写以下选框：
   - 名称——输入一个最多包含 100 个字符的唯一名称。有效条目包括：字母数字字符、连字符和下划线。例如，*MyTracker*.
   - 描述——输入可选描述。例如，*Tracker for storing AWS IoT Core device positions*.
   - 位置筛选——选择要用于位置更新的筛选。例如，基于精度的筛选。
5. 选择创建跟踪器。

创建 Lambda 函数

要在 AWS IoT Core 和 Amazon Location Service 之间建立连接，您需要一个 AWS Lambda 函数来处理由转发的消息 AWS IoT Core。此函数将提取所有位置数据，将其格式化为 Amazon Location Service，然后通过亚马逊位置追踪器API提交。您可以通过 AWS Lambda 控制台创建此函数，也可以使用 AWS Command Line Interface (AWS CLI) 或 AWS Lambda APIs。

要使用控制台创建 Lambda 函数，将位置更新发布到 Amazon Location，请执行以下操作：

1. 打开 AWS Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 从左侧导航窗格中，选择函数。
3. 选择创建函数，并确保选择从头开始创作。
4. 填写以下选框：
   - 函数名称——输入您的函数的唯一名称。有效条目包括字母数字字符、连字符和下划线，不带空格。例如，*MyLambda*.

- 运行时间-选择 *Python 3.8*.

5. 选择创建函数。
6. 选择 代码 选项卡以打开编辑器。
7. 用以下代码替换 lambda_function.py 中的占位符代码，将分配给 TRACKER_NAME 的值替换为您作为先决条件创建的跟踪器的名称。

```
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. 选择部署以保存您更新的函数。
9. 选择配置选项卡。
10. 在权限部分，选择超链接的角色名称以授予 Amazon Location Service 对您的 Lambda 函数的权限。
11. 在角色的摘要页面中，选择添加权限，然后从下拉列表中选择创建内联策略。
12. 选择该JSON选项卡，然后使用以下文档覆盖该策略。这允许您的 Lambda 函数更新由所有区域的所有跟踪器资源管理的设备位置。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
```

```
    }
  ]
}
```

13. 选择查看策略。
14. 输入策略名称。例如，*AmazonLocationTrackerWriteOnly*.
15. 选择 创建策略。

您可以根据需要修改此函数代码，以适应您自己的设备消息架构。

创建 AWS IoT Core 规则

接下来，创建一条 AWS IoT Core 规则，将设备的位置遥测数据转发给该 AWS Lambda 函数，以便转换并发布到 Amazon Location Service。提供的示例规则假设设备有效负载的任何必要转换均由您的 Lambda 函数处理。您可以通过 AWS IoT Core 控制台、 AWS Command Line Interface (AWS CLI) 或创建此规则 AWS IoT Core APIs。

> **Note**
>
> 虽然 AWS IoT 控制台处理 AWS IoT Core 允许调用 Lambda 函数所需的权限，但如果您从 AWS CLI 或创建规则SDK，则必须配置策略以授予权限。 AWS IoT

AWS IoT Core 使用控制台创建

1. 登录 AWS IoT Core 控制台，网址为https://console.aws.amazon.com/iot/。
2. 在左侧导航窗格中，展开行动，并选择规则。
3. 选择创建规则以启动新规则向导。
4. 为您的规则输入名称和描述。
5. 对于规则查询语句，请更新 FROM 属性，以引用至少一台设备正在发布包含位置的遥测的主题。如果您正在测试解决方案，则无需进行任何修改。

```
SELECT * FROM 'iot/topic'
```

6. 在设置一个或多个操作下，选择添加操作。
7. 选择向 Lambda 函数发送消息。
8. 选择 Configure action。

9. 从列表中选择您的 Lambda 函数。
10. 选择添加操作。
11. 选择创建规则。

在控制台中测试你的 AWS IoT Core 规则

如果当前没有设备发布包含位置信息的遥测数据，则可以使用 AWS IoT Core 控制台测试您的规则和此解决方案。控制台有一个测试客户端，您可以在其中发布一条示例消息来验证解决方案的结果。

1. 登录 AWS IoT Core 控制台，网址为https://console.aws.amazon.com/iot/。
2. 在左侧导航栏中，展开 “测试”，然后选择 “MQTT测试客户端”。
3. 在 “发布到主题” 下，将主题名称设置为 *iot/topic*（或您在 AWS IoT Core 规则中设置的主题的名称，如果不同），并为消息负载提供以下内容。替换时间戳 *1604940328* 具有最近 30 天内的有效时间戳（任何超过 30 天的时间戳都将被忽略）。

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. 选择发布到主题来发送测试消息。
5. 要验证 Amazon Location Service 是否已收到该消息，请使用以下 AWS CLI 命令。如果您在设置过程中对其进行了修改，请将跟踪器名称和设备 ID 替换为您使用的跟踪器名称和设备 ID。

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

管理您的地理围栏集合资源

使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 管理您的地理围栏集合。

列出您的地理围栏集合资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 查看您的地理围栏集合列表：

Console

使用 Amazon Location 控制台查看地理围栏集合列表

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地理围栏集合
3. 在我的地理围栏集合下查看您的地理围栏集合列表。

API

使用 Amazon Location 地理围栏 API 中的 `ListGeofenceCollections` 操作。

以下示例是获取 AWS 账户中地理围栏集合列表的 API 请求。

```
POST /geofencing/v0/list-collections
```

以下为 `ListGeofenceCollections` 响应示例：

```
{
    "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。

以下示例是一个用于获取 AWS 账户中地理围栏集合的列表的 AWS CLI。

```
aws location list-geofence-collections
```

获取地理围栏集合的详细信息

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 获取有关您的 AWS 账户中任何地理围栏集合资源的详细信息：

Console

使用 Amazon Location 控制台查看地理围栏集合的详细信息

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地理围栏集合
3. 在我的地理围栏集合下，选择目标地理围栏集合的名称链接。

API

使用 Amazon Location 地理围栏 API 中的 DescribeGeofenceCollection 操作。

以下示例是获取地理围栏收集详细信息的 API 请求。*ExampleCollection*

```
GET /geofencing/v0/collections/ExampleCollection
```

以下为 DescribeGeofenceCollection 响应示例：

```
{
    "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。

以下示例是AWS CLI获取地理围栏集合详细信息的示例。*ExampleCollection*

```
aws location describe-geofence-collection \
    --collection-name "ExampleCollection"
```

删除地理围栏集合

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 从您的 AWS 账户中删除地理围栏集合。

Console

使用 Amazon Location 控制台删除地理围栏集合

> **Warning**
>
> 此操作将永久删除资源。

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地理围栏集合
3. 在我的地理围栏集合下，选择目标地理围栏集合。
4. 选择删除地理围栏集合。

API

使用 Amazon Location API 中的 DeleteGeofenceCollection 操作。

以下示例是删除地理围栏集合的 API 请求。*ExampleCollection*

```
DELETE /geofencing/v0/collections/ExampleCollection
```

以下为 DeleteGeofenceCollection 响应示例：

```
HTTP/1.1 200
```

CLI

使用 delete-geofence-collection 命令。

以下示例是删除地理围栏集合的AWS CLI命令。*ExampleCollection*

```
aws location delete-geofence-collection \
    --collection-name "ExampleCollection"
```

列出存储的地理围栏

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 列出存储在指定地理围栏集合中的地理围栏。

Console

使用 Amazon Location 控制台查看地理围栏列表

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地理围栏集合
3. 在我的地理围栏集合下，选择目标地理围栏集合的名称链接。
4. 在地理围栏下查看地理围栏集合中的地理围栏。

API

使用 Amazon Location 地理围栏 API 中的 `ListGeofences` 操作。

以下示例是获取存储在地理围栏集合中的地理围栏列表的 API 请求。*ExampleCollection*

```
POST /geofencing/v0/collections/ExampleCollection/list-geofences
```

以下为 `ListGeofences` 响应示例：

```
{
   "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。

以下示例是获取存储在 geofence 集合中的地理围栏列表的示例。AWS CLI *ExampleCollection*

```
aws location list-geofences \
    --collection-name "ExampleCollection"
```

获取地理围栏详细信息

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 从地理围栏集合中获取特定地理围栏的详细信息，例如创建时间、更新时间、几何图形和状态。

Console

使用 Amazon Location 控制台查看地理围栏的状态

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地理围栏集合
3. 在我的地理围栏集合下，选择目标地理围栏集合的名称链接。
4. 在地理围栏 ，你可以查看地理围栏的状态。

API

使用 Amazon Location 地理围栏 API 中的 `GetGeofence` 操作。

以下示例是从地理围栏集合中获取地理围栏详细信息的 API 请求。*ExampleCollection*

```
GET /geofencing/v0/collections/ExampleCollection/geofences/ExampleGeofence1
```

以下为 GetGeofence 响应示例：

```
{
   "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。

以下示例是AWS CLI获取地理围栏集合详细信息的示例。*ExampleCollection*

```
aws location get-geofence \
    --collection-name "ExampleCollection" \
    --geofence-id "ExampleGeofence1"
```

删除地理围栏

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 从地理围栏集合中删除地理围栏。

Console

使用 Amazon Location 控制台删除地理围栏

> **Warning**
>
> 此操作将永久删除资源。

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择地理围栏集合
3. 在我的地理围栏集合下，选择目标地理围栏集合的名称链接。
4. 在地理围栏下，选择目标地理围栏。
5. 选择删除地理围栏。

API

使用 Amazon Location 地理围栏 API 中的 BatchDeleteGeofence 操作。

以下示例是从地理围栏集合中删除地理围栏的 API 请求。*ExampleCollection*

```
POST /geofencing/v0/collections/ExampleCollection/delete-geofences
Content-type: application/json

{
   "GeofenceIds": [ "ExampleGeofence11" ]
}
```

以下是 BatchDeleteGeofence 成功响应的示例。

```
HTTP/1.1 200
```

CLI

使用 batch-delete-geofence 命令。

以下示例是从地理围栏集合中删除地理围栏的AWS CLI命令。*ExampleCollection*

```
aws location batch-delete-geofence \
    --collection-name "ExampleCollection" \
    --geofence-ids "ExampleGeofence11"
```

管理您的跟踪器资源

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 来管理您的跟踪器。

列出您的跟踪器

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 查看您的跟踪器列表：

Console

使用 Amazon Location 控制台查看现有跟踪器列表

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择跟踪器。
3. 在我的跟踪器下查看您的跟踪器资源列表。

API

使用 Amazon Location 跟踪器 API 中的 `ListTrackers` 操作。

以下示例是一个用于获取您 AWS 账户中的跟踪器列表的 API 请求。

```
POST /tracking/v0/list-trackers
```

以下为 `ListTrackers` 响应示例：

```
{
   "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。

以下示例是获取您 AWS 账户中跟踪器列表的 AWS CLI 示例。

```
aws location list-trackers
```

断开跟踪器与地理围栏集合的连接

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 断开跟踪器与地理围栏集合的连接：

Console

使用 Amazon Location 控制台取消跟踪器与关联的地理围栏集合的关联

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择跟踪器。
3. 在我的跟踪器下，选择目标跟踪器的名称链接。
4. 在链接的地理围栏集合下，选择状态为已链接的地理围栏集合。
5. 选择断开关联。

API

使用 Amazon Location 跟踪器 API 中的 DisassociateTrackerConsumer 操作。

以下示例是一个用于取消跟踪器与链接的地理围栏集合的关联的 API 请求。

```
DELETE /tracking/v0/trackers/ExampleTracker/consumers/arn:aws:geo:us-
west-2:123456789012:geofence-collection/ExampleCollection
```

以下为 DisassociateTrackerConsumer 响应示例：

```
HTTP/1.1 200
```

CLI

使用 disassociate-tracker-consumer 命令。

以下示例是取消跟踪器与链接的地理围栏集合的关联的 AWS CLI 命令。

```
aws location disassociate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-collection/
ExampleCollection" \
    --tracker-name "ExampleTracker"
```

获取跟踪器详细信息

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 来获取有关您 AWS 账户中任何跟踪器的详细信息。

Console

使用 Amazon Location 控制台查看跟踪器详情

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择跟踪器。
3. 在我的跟踪器下，选择目标跟踪器的名称链接。
4. 在信息下查看跟踪器详情。

API

使用 Amazon Location 跟踪器 API 中的 `DescribeTracker` 操作。

以下示例是获取其跟踪链接详情的 API 请求*ExampleTracker*。

```
GET /tracking/v0/trackers/ExampleTracker
```

以下为 `DescribeTracker` 响应示例：

```
{
   "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。

以下示例是获取跟踪器详细信息的AWS CLI命令*ExampleTracker*。

```
aws location describe-tracker \
    --tracker-name "ExampleTracker"
```

删除跟踪器

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 从 AWS 账户中删除跟踪器：

Console

使用 Amazon Location 控制台删除现有地图资源

> **Warning**
>
> 此操作将永久删除资源。如果跟踪器资源正在使用中，您可能会遇到错误。确保目标资源不是应用程序的依赖项。

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 从左侧导航窗格中选择跟踪器。
3. 在我的跟踪器下，选择目标跟踪器。
4. 选择删除跟踪器。

API

使用 Amazon Location 跟踪器 API 中的 `DeleteTracker` 操作。

以下示例是删除跟踪器的 API 请求*ExampleTracker*。

```
DELETE /tracking/v0/trackers/ExampleTracker
```

以下为 `DeleteTracker` 响应示例：

```
HTTP/1.1 200
```

CLI

使用 delete-tracker 命令。

以下示例是删除跟踪器的AWS CLI命令*ExampleTracker*。

```
aws location delete-tracker \
    --tracker-name "ExampleTracker"
```

地理围栏和跟踪移动应用程序示例

本主题介绍的教程旨在演示在移动应用程序中使用 Amazon Location 地理围栏和追踪器的主要功能。这些应用程序演示了跟踪器和地理围栏如何使用 Lambda 和 AWS IoT Amazon Location 功能的组合进行交互。有两个教程可供选择。

- Android 版跟踪和地理围栏应用程序示例，你可以从 GitHub以下位置克隆项目文件：/tree/main/。https://github.com/aws-geospatial/ amazon-location-samples-android tracking-with-geofence-notifications
- iOS 版跟踪和地理围栏应用程序示例，你可以从 GitHub以下位置克隆项目文件：/tree/main/。https://github.com/aws-geospatial/ amazon-location-samples-ios tracking-with-geofence-notifications

适用于 Android 的示例跟踪和地理围栏应用程序

本主题介绍了 Android 教程，该教程旨在演示在移动应用程序中使用 Amazon Location 地理围栏和追踪器的主要功能。这些应用程序演示了跟踪器和地理围栏如何使用 Lambda 和 AWS IoT Amazon Location 功能的组合进行交互。

主题

- 为您的应用程序创建 Amazon Location 资源
- 创建 Geofence 收藏集
- 将跟踪器关联到地理围栏集合
- 将 AWS Lambda 与 MQTT 配合使用
- 设置示例应用程序代码
- 使用示例应用程序

为您的应用程序创建 Amazon Location 资源

首先，您需要创建所需的 Amazon Location 资源。这些资源对于应用程序的功能和执行所提供的代码片段至关重要。

> **Note**
>
> 如果您尚未创建账户，请按照 AWS AWS 账户管理用户指南中的说明进行操作。

首先，您需要创建 Amazon Cognito 身份池 ID，请按以下步骤操作：

1. 打开 Amazon Cognito 控制台，从左侧菜单中选择身份池，然后选择创建身份池。
2. 确保选中 “访客访问权限”，然后按 “下一步” 继续。
3. 接下来创建新的 IAM 角色或使用现有的 IAM 角色。
4. 输入身份池名称，并确保身份池可以访问您将在下一个步骤中创建的地图和追踪器的 Amazon Location（geo）资源。

接下来，您需要在 AWS Amazon Location 控制台中创建地图并设置地图样式，请按以下步骤操作：

1. 导航至 Amazon Location 控制台的 “地图” 部分，然后选择 “创建地图”。
2. 为新地图指定名称和描述。记录您指定的名称，本教程稍后将使用该名称。
3. 选择地图样式时，请考虑地图数据提供器。有关更多详细信息，请参阅AWS 服务条款的第 82 节。
4. 接受 Amazon 位置条款和条件，然后选择创建地图，完成地图创建过程。

接下来，您需要在 Amazon Location 控制台中创建追踪器，请按以下步骤操作：

1. 在 Amazon Location 控制台中打开 “地图” 部分。
2. 选择创建跟踪器。
3. 填写必填字段。记下跟踪器的名称，因为本教程中将引用该名称。
4. 在 “位置筛选” 字段下，选择最适合您打算如何使用跟踪链接资源的选项。如果未设置位置筛选，则默认设置为`TimeBased`。有关更多信息，请参阅追踪器和 Amazon 定PositionFiltering位 API 参考。
5. 选择 “创建跟踪链接” 以完成追踪器的创建。

创建 Geofence 收藏集

现在，您将创建一个地理围栏集合。您可以使用控制台、API 或 CLI。以下过程将引导您完成每个选项。

- 使用 Amazon Location 控制台创建地理围栏集合：

  1. 打开 Amazon Location 控制台的 Geofence 收藏部分。
  2. 选择创建地理围栏集合。
  3. 为集合提供名称和描述。
  4. 根据以 Amazon CloudWatch 为目标的 EventBridge 规则，您可以创建一条可选 EventBridge 规则来开始对地理围栏事件做出反应。这使得 Amazon Location 可以向发布事件 Amazon CloudWatch Logs。
  5. 按下创建 geofence 收藏集以完成收藏夹的创建。

- 使用 Amazon Location API 创建地理围栏集合：

  使用 Amazon 定位地理围栏 API 中的CreateGeofenceCollection操作。以下示例使用 API 请求创建名为的地理围栏集合。*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 CL AWS I 命令创建地理围栏集合：

  使用 `create-geofence-collection` 命令。以下示例使用 AWS CLI 创建名为的地理围栏集合。*GEOCOLLECTION_NAME*有关使用 AWS CLI 的更多信息，请参阅AWS 命令行界面文档。

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --tags Tag1=Value1
```

将跟踪器关联到地理围栏集合

要将跟踪器关联到地理围栏集合，您可以使用控制台、API 或 CLI。以下过程将引导您完成每个选项。

使用 Amazon Location Service 控制台将追踪器资源链接到地理围栏集合：

1. 打开 Amazon Location 控制台。
2. 在左侧导航窗格中，选择跟踪器。
3. 在 “设备跟踪器” 下，选择目标跟踪链接的名称链接。
4. 在链接的地理围栏集合下，选择链接地理围栏集合。
5. 在链接的地理围栏集合窗口中，从下拉菜单中选择一个地理围栏集合。
6. 选择关联。
7. 在您关联跟踪器资源后，它将处于活动状态。

使用 Amazon Location API 将追踪器资源关联到地理围栏集合：

使用 Amazon Location 跟踪器 API 中的 `AsssociateTrackerConsumer` 操作。以下示例使用一个 API 请求，该请求使用其亚马逊资源名称 (ARN) 将`ExampleTracker`与地理围栏集合关联起来。

```
        POST /tracking/v0/trackers/ExampleTracker/consumers
        Content-type: application/json
        {
           "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME"
            }
```

使用 CL AWS I 命令将跟踪器资源链接到地理围栏集合：

使用 `associate-tracker-consumer` 命令。以下示例使用 AWS CLI 创建名为的地理围栏集合。*GOECOLLECTION_NAME*

```
aws location \
associate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME" \
    --tracker-name "ExampleTracker"
```

将 AWS Lambda 与 MQTT 配合使用

为了在 AWS IoT 和 Amazon Location 之间建立连接，您需要一个 Lambda 函数来处理由事件转发的 EventBridge CloudWatch 消息。此函数将提取所有位置数据，将其格式化为亚马逊位置，然后通过亚马逊位置追踪器 API 提交。

以下过程向您展示如何通过 Lambda 控制台创建此函数：

1. 打开控制台。
2. 从左侧导航窗格中，选择函数。
3. 然后选择 “创建函数”，并确保选中 “从头开始创作” 选项。
4. 提供函数名称，然后在 “运行时” 选项中选择 Node.js 16.x。
5. 选择创建函数。
6. 打开 “代码” 选项卡以访问编辑器。
7. 使用以下内容覆盖index.js文件中的占位符代码：

```
            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. 按下 Deploy 保存更新的函数。
9. 接下来打开 “配置” 选项卡。
10. 在 “触发器” 部分中，按添加触发器按钮。
11. 在 “来源” 字段中选择 EventBridge （CloudWatch 事件）。
12. 选择 “现有规则” 选项。
13. 例如，输入规则名称`AmazonLocationMonitor-GEOFENCECOLLECTION_NAME`。
14. 按 “添加” 按钮。
15. 这还将在 “权限” 选项卡中附加基于资源的策略声明

现在，您将使用 AWS IoT以下步骤设置 MQTT 测试客户端：

1. 打开 https://console.aws.amazon.com/iot/。
2. 在左侧导航窗格中，选择 MQTT 测试客户端。
3. 您将看到标题为 MQTT 测试客户端的部分，您可以在其中配置 MQTT 连接。
4. 配置必要设置后，单击 Connect 按钮，使用提供的参数建立与 MQTT 代理的连接。
5. 记录端点，本教程稍后将使用该端点。

   连接到测试客户端后，您可以使用 MQTT 测试客户端界面中提供的相应输入字段订阅 MQTT 主题或向主题发布消息。接下来，您将创建一个 AWS IoT 策略。
6. 在左侧菜单的 “管理” 下，展开 “安全” 选项，然后单击 “策略”。
7. 单击 “创建策略” 按钮。
8. 输入策略名称。
9. 在政策文档上，选择 JSON 选项卡。
10. 复制粘贴下方显示的政策，但请务必使用您的*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. 选择 “创建” 按钮即可完成。

完成上述步骤后，您现在将按如下方式更新访客角色的权限：

1. 导航至 Amazon Cognito 并打开您的身份池。然后，进入用户访问权限并选择访客角色。
2. 点击权限策略以启用编辑。

```
{
    '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. 通过上述策略更改，现在已为应用程序正确配置了所有必要的 AWS 资源。

设置示例应用程序代码

1. 将此存储库：https://github.com/aws-geospatial/ amazon-location-samples-android /tree/main/ 克隆tracking-with-geofence-notifications到你的本地计算机。
2. 在安卓工作室中打开AmazonSampleSDKApp项目。
3. 在您的 Android 设备或模拟器上构建并运行该应用程序。

使用示例应用程序

要使用该示例，请按照以下步骤操作：

- 创建一个 **`custom.properties`**：

  要配置您的custom.properties文件，请按照以下步骤操作：

  1. 打开您首选的文本编辑器或 IDE。
  2. 创建新的文件。
  3. 使用文件名 `custom.properties` 保存该文件。

4. custom.properties使用以下代码示例更新，并使用您的资源名称替换MQTT_END_POINTPOLICY_NAMEGEOFENCE_COLLECTION_NAME、、和TOPIC_TRACKER：

```
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. 清理并重建项目。之后，您可以运行该项目。

- 登录：

  要登录应用程序，请按照以下步骤操作：

  1. 按 “登录” 按钮。
  2. 提供身份池 ID、跟踪器名称和地图名称。
  3. 再次按 “登录” 完成操作。

- 管理过滤器：

  打开配置屏幕，然后执行以下操作：

  1. 使用 Switch UI 打开或关闭过滤器。
  2. 需要时更新时间和距离过滤器。

- 追踪操作：

  打开跟踪屏幕并执行以下操作：

  - 您可以通过按下相应的按钮在前景、后台或省电模式下开始和停止跟踪。

iOS 版跟踪和地理围栏应用示例

本主题介绍了 iOS 教程，该教程旨在演示在移动应用程序中使用 Amazon Location 地理围栏和追踪器的主要功能。这些应用程序演示了跟踪器和地理围栏如何使用 Lambda 和 AWS IoT Amazon Location 功能的组合进行交互。

主题

- 为您的应用程序创建 Amazon Location 资源
- 创建 Geofence 收藏集

- 将跟踪器关联到地理围栏集合
- 将 AWS Lambda 与 MQTT 配合使用
- 设置示例应用程序代码
- 使用示例应用程序

为您的应用程序创建 Amazon Location 资源

首先，您需要创建所需的 Amazon Location 资源。这些资源对于应用程序的功能和执行所提供的代码片段至关重要。

> **Note**
>
> 如果您尚未创建账户，请按照 AWS AWS 账户管理用户指南中的说明进行操作。

首先，您需要创建 Amazon Cognito 身份池 ID，请按以下步骤操作：

1. 打开 Amazon Cognito 控制台，从左侧菜单中选择身份池，然后选择创建身份池。
2. 确保选中 “访客访问权限”，然后按 “下一步” 继续。
3. 接下来创建新的 IAM 角色或使用现有的 IAM 角色。
4. 输入身份池名称，并确保身份池可以访问您将在下一个步骤中创建的地图和追踪器的 Amazon Location（geo）资源。

接下来，您需要在 AWS Amazon Location 控制台中创建地图并设置地图样式，请按以下步骤操作：

1. 导航至 Amazon Location 控制台的 “地图” 部分，然后选择 “创建地图”。
2. 为新地图指定名称和描述。记录您指定的名称，本教程稍后将使用该名称。
3. 选择地图样式时，请考虑地图数据提供器。有关更多详细信息，请参阅AWS 服务条款的第 82 节。
4. 接受 Amazon 位置条款和条件，然后选择创建地图，完成地图创建过程。

接下来，您需要在 Amazon Location 控制台中创建追踪器，请按以下步骤操作：

1. 在 Amazon Location 控制台中打开 “地图” 部分。
2. 选择创建跟踪器。

3. 填写必填字段。记下跟踪器的名称，因为本教程中将引用该名称。
4. 在 “位置筛选” 字段下，选择最适合您打算如何使用跟踪链接资源的选项。如果未设置位置筛选，则默认设置为`TimeBased`。有关更多信息，请参阅开始追踪和《亚马逊定PositionFiltering位 API 参考》。
5. 选择 “创建跟踪链接” 以完成追踪器的创建。

创建 Geofence 收藏集

现在，您将创建一个地理围栏集合。您可以使用控制台、API 或 CLI。以下过程将引导您完成每个选项。

- 使用 Amazon Location 控制台创建地理围栏集合：

  1. 打开 Amazon Location 控制台的 Geofence 收藏部分。
  2. 选择创建地理围栏集合。
  3. 为集合提供名称和描述。
  4. 根据以 Amazon CloudWatch 为目标的 EventBridge 规则，您可以创建一条可选 EventBridge 规则来开始对地理围栏事件做出反应。这使得 Amazon Location 可以向发布事件 Amazon CloudWatch Logs。
  5. 按下创建 geofence 收藏集以完成收藏夹的创建。

- 使用 Amazon Location API 创建地理围栏集合：

  使用 Amazon 定位地理围栏 API 中的CreateGeofenceCollection操作。以下示例使用 API 请求创建名为的地理围栏集合。*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 CL AWS I 命令创建地理围栏集合：

使用 `create-geofence-collection` 命令。以下示例使用 AWS CLI 创建名为的地理围栏集合。*GEOCOLLECTION_NAME*有关使用 AWS CLI 的更多信息，请参阅AWS 命令行界面文档。

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --tags Tag1=Value1
```

将跟踪器关联到地理围栏集合

要将跟踪器关联到地理围栏集合，您可以使用控制台、API 或 CLI。以下过程将引导您完成每个选项。

使用 Amazon Location Service 控制台将追踪器资源链接到地理围栏集合：

1. 打开 Amazon Location 控制台。
2. 在左侧导航窗格中，选择跟踪器。
3. 在 “设备跟踪器” 下，选择目标跟踪链接的名称链接。
4. 在链接的地理围栏集合下，选择链接地理围栏集合。
5. 在链接的地理围栏集合窗口中，从下拉菜单中选择一个地理围栏集合。
6. 选择关联。
7. 在您关联跟踪器资源后，它将处于活动状态。

使用 Amazon Location API 将追踪器资源关联到地理围栏集合：

使用 Amazon Location 跟踪器 API 中的 `AssociateTrackerConsumer` 操作。以下示例使用一个 API 请求，该请求使用地理围栏集合的 Amazon 资源名称 (ARN) ExampleTracker 与该集合相关联。

```
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 命令将跟踪器资源关联到地理围栏集合：

使用 associate-tracker-consumer 命令。以下示例使用 AWS CLI 创建名为的地理围栏集合。*GEOCOLLECTION_NAME*

```
aws location \
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GEOCOLLECTION_NAME" \
        --tracker-name "ExampleTracker"
```

将 AWS Lambda 与 MQTT 配合使用

为了在 AWS IoT 和 Amazon Location 之间建立连接，您需要一个 Lambda 函数来处理由事件转发的 EventBridge CloudWatch 消息。此函数将提取所有位置数据，将其格式化为亚马逊位置，然后通过亚马逊位置追踪器 API 提交。

以下过程向您展示如何通过 Lambda 控制台创建此函数：

1. 打开控制台。
2. 从左侧导航窗格中，选择函数。
3. 然后选择 “创建函数”，并确保选中 “从头开始创作” 选项。
4. 提供函数名称，然后在 “运行时” 选项中选择 Node.js 16.x。
5. 选择创建函数。
6. 打开 “代码” 选项卡以访问编辑器。
7. 使用以下内容覆盖index.js文件中的占位符代码：

```
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. 按下 Deploy 保存更新的函数。
9. 接下来打开 “配置” 选项卡。
10. 在 “触发器” 部分中，按添加触发器按钮。
11. 在 “来源” 字段中选择 EventBridge （CloudWatch 事件）。
12. 选择 “现有规则” 选项。

13. 例如，输入规则名称AmazonLocationMonitor-GEOFENCECOLLECTION_NAME。
14. 按 “添加” 按钮。
15. 这还将在 “权限” 选项卡中附加基于资源的策略声明

现在，您将按以下步骤设置 AWS IoT MQTT 测试客户端：

1. 打开 https://console.aws.amazon.com/iot/。
2. 在左侧导航窗格中，选择 MQTT 测试客户端。
3. 您将看到标题为 MQTT 测试客户端的部分，您可以在其中配置 MQTT 连接。
4. 配置必要设置后，单击 Connect 按钮，使用提供的参数建立与 MQTT 代理的连接。
5. 记录端点，本教程稍后将使用该端点。

   连接到测试客户端后，您可以使用 MQTT 测试客户端界面中提供的相应输入字段订阅 MQTT 主题或向主题发布消息。接下来，您将创建一个 AWS IoT 策略。
6. 在左侧菜单的 “管理” 下，展开 “安全” 选项，然后单击 “策略”。
7. 单击 “创建策略” 按钮。
8. 输入策略名称。
9. 在政策文档上，选择 JSON 选项卡。
10. 复制粘贴下方显示的政策，但请务必使用您的*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. 选择 “创建” 按钮即可完成。

设置示例应用程序代码

要设置示例代码，必须安装以下工具：

- Git
- xCode 15.3 或更高版本
- iOS 模拟器 16 或更高版本

使用以下步骤设置示例应用程序代码：

1. 从这个网址克隆 git 存储库：https://github.com/aws-geospatial/ amazon-location-samples-ios / tree/main/ tracking-with-geofence-notifications。
2. 打开 AWSLocationSampleApp.xcodeproj 项目文件。
3. 等待包解析过程完成。
4. 在项目导航菜单上ConfigTemplate.xcconfig，重命名为Config.xcconfig并填写以下值：

```
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`
```

使用示例应用程序

设置好示例代码后，您现在可以在 iOS 模拟器或物理设备上运行该应用程序。

1. 构建并运行应用程序。
2. 该应用程序将要求您提供位置和通知权限。你需要允许他们。

3. 按下 Cognito 配置按钮。
4. 保存配置。
5. 现在，您可以看到时间、距离和精度的筛选选项。根据需要使用它们。
6. 前往应用程序中的追踪选项卡，您将看到地图和开始追踪按钮。
7. 如果您已在模拟器上安装了该应用程序，则可能需要模拟位置更改。这可以在 “位置” 菜单选项下的 “功能” 中完成。例如，依次选择功能、位置、Freeway Dri ve。
8. 按下开始追踪按钮。你应该在地图上看到追踪点。
9. 该应用程序还在后台跟踪位置。因此，当你在后台移动应用程序时，它会要求你允许在后台模式下继续跟踪。
10. 您可以通过点击 “停止跟踪” 按钮来停止跟踪。

为您的 Amazon Location Service 资源添加标签

使用 Amazon Location 中的资源标签来创建标签，按用途、拥有者、环境或标准对您的资源进行分类。标记您的资源可以帮助您管理、识别、组织、搜索和筛选资源。

例如，使用 AWS Resource Groups，您可以基于一个或多个标签或部分标签创建 AWS 资源组。您还可以根据组在 AWS CloudFormation 堆栈中的出现情况创建组。使用 Resource Groups 和标签编辑器，您可以在一个位置整合和查看由多个服务、资源和区域组成的应用程序的数据。有关常见标签策略的更多信息，请参阅 AWS 一般参考。

每个标签都是包含您定义的一个键和值的标记：

- 标签密钥——对标签值进行分类的通用标签。例如，`CostCenter`。
- 标签值——标签密钥类别的可选描述。例如，`MobileAssetTrackingResourcesProd`。

本主题通过查看标签限制来帮助您开始标记。它还向您展示了如何使用成本分配报告创建标签和使用标签来跟踪每个有效标签的 AWS 成本。

主题

- 使用标签控制对 Amazon Location Service 资源的访问
- 了解更多信息

添加标签限制

下面是适用于标签的基本限制：

- 每个资源的标签数上限为 50
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。

> **Note**
>
> 如果您添加的新标签的标签键与现有标签的相同，则新标签将覆盖现有标签。

- 最大键长度 – 128 个 Unicode 字符 (采用 UTF-8 格式)
- 最大值长度 – 256 个 Unicode 字符（采用 UTF-8 格式）
- 允许在不同的服务中使用的字符包括：可以使用 UTF-8 表示的字母、数字和空格以及以下字符：+ - = . _ : / @。
- 标签键和值区分大小写。
- `aws:` 前缀专门预留供 AWS 使用。如果某个标签具有带有此标签键，则您无法编辑该标签的键或值。具有 `aws:` 前缀的标签不计入每个资源的标签数限制。

授予标记资源的权限

您可以使用 IAM policy 来控制对您的 Amazon Location 资源的访问权限，并授予在创建资源时标记资源的权限。除了授予创建资源的权限外，该策略还可以包括允许标记操作的 `Action` 权限：

- `geo:TagResource`——允许用户将一个或多个标签分配给指定的 Amazon Location 资源。
- `geo:UntagResource`——允许用户从指定的 Amazon Location 资源中删除一个或多个标签。
- `geo:ListTagsForResource`——允许用户列出分配给 Amazon Location 资源的所有标签。

以下是允许用户创建地理围栏集合并标记资源的策略示例：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowTaggingForGeofenceCollectionOnCreation",
            "Effect": "Allow",
            "Action": [
                "geo:CreateGeofenceCollection",
                "geo:TagResource"
            ],
            "Resource": "arn:aws:geo:region:accountID:geofence-collection/*"
    ]
}
```

向 Amazon Location Service 资源添加标签

您可以使用 Amazon Location 控制台、AWS CLI 或 Amazon Location API 在创建资源时添加标签：

- 创建地图资源
- 创建地点索引资源
- 创建路由计算器资源
- 创建地理围栏集合
- 创建跟踪器资源

为现有资源添加、编辑或删除标签

1. 打开 Amazon Location 控制台：https://console.aws.amazon.com/location/。
2. 在左侧导航窗格中，选择要标记的资源。例如，地图。
3. 从列表中选择一个资源。
4. 选择管理标签以添加、编辑或删除标签。

按标签跟踪资源成本

您可以使用标签进行成本分配，以详细跟踪您的 AWS 成本。激活成本分配标签后，AWS 将使用成本分配标签在成本分配报告上组织您的资源计费。这可以帮助您对使用成本进行分类和跟踪。

您可以激活两种类型的成本分配标签：

- AWS-generated——这些标签由 AWS 生成。AWS 标签使用 `aws:` 前缀，例如 `aws:createdBy`。

- 用户定义——这些是您创建的自定义标签。用户定义的标签使用 `user:` 前缀，例如 `user:CostCenter`。

必须单独激活每种标签类型。激活标签后，您可以启用AWS Cost Explorer或查看每月成本分配报告。

AWS-generated tags

激活 AWS 生成的标签

1. 打开 Billing and Cost Management 控制台：https://console.aws.amazon.com/billing/。
2. 在左侧导航窗格中，选择成本分配标签。
3. 在 AWS-生成的成本分配标签选项卡下，选择要激活的标签密钥。
4. 选择激活。

User-defined tags

激活用户定义的标签：

1. 打开 Billing and Cost Management 控制台：https://console.aws.amazon.com/billing/。
2. 在左侧导航窗格中，选择成本分配标签。
3. 在用户定义的成本分配标签选项卡下，选择要激活的标签密钥。
4. 选择激活。

激活标签后，AWS 会针对您的资源使用情况和成本生成每月成本分配报告。本成本分配报告包括您每个账单周期的所有 AWS 成本，包括标记和未标记的资源。有关更多信息，请参阅《AWS Billing and Cost Management 用户指南》中的使用成本分配标签。

使用标签控制对 Amazon Location Service 资源的访问

AWS Identity and Access Management(IAM) 策略支持基于标签的条件，使您能够根据特定标签密钥和值管理对资源的访问。例如，IAM 角色策略可以包含条件来限制对特定环境的访问，如基于标签的开发、测试或生产环境。

想要了解更多信息，请参阅基于标签控制资源访问的主题。

了解更多信息

有关更多信息如下：

- 有关标签使用的最佳实践，请参阅 AWS 一般参考中的 AWS 资源标记。
- 使用标签控制对 AWS 资源的访问，请参阅 AWS Identity and Access Management 用户指南中的使用标签控制对 AWS 资源的访问。

授予对 Amazon Location Service 的访问权限

要使用 Amazon Location Service，必须向用户授予访问构成Amazon Location 的资源和 API 的权限。您可以使用授予对资源的访问权限的三种策略：

- 使用 IAM——要向使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 进行身份验证的用户授予访问权限，创建允许访问所需资源的 IAM policy。有关 IAM 和 Amazon Location 的更多信息，请参阅 适用于 Amazon Location Service 的身份和访问管理。
- 使用 API 密钥——要向未经身份验证的用户授予访问权限，您可以创建 API 密钥来授予对您的 Amazon Location Service 资源的只读访问权限。在您不想对每个用户进行身份验证的情况下，这很有用。例如，Web 应用程序。有关 API 密钥的更多信息，请参阅 允许未经身份验证的访客使用 API 密钥访问您的应用程序。
- 使用 Amazon Cognito——API 密钥的替代方法是使用 Amazon Cognito 授予匿名访问权限。Amazon Cognito 允许您使用策略创建更丰富的授权，以定义未经身份验证的用户可以执行的操作。有关使用 Amazon Cognito 的更多信息，请参阅 使用 Amazon Cognito 允许未经身份验证的访客访问您的应用程序。

> **Note**
>
> 您也可以使用 Amazon Cognito 使用自己的身份验证流程，或者使用Amazon Cognito 联合身份组合多种身份验证方法。想要了解更多信息，请参阅 Amazon Cognito 开发人员指南 中的开始使用联合身份。

主题

允许未经身份验证的访客使用 API 密钥访问您的应用程序

当您在应用程序中调用 Amazon Location Service API 时，您通常以经过身份验证且有权进行 API 调用的用户身份进行此调用。但是，在某些情况下，您不希望对应用程序的每个用户进行身份验证。例如，

您可能希望任何使用该网站的人都可以使用显示您的营业地点的 Web 应用程序，无论他们是否登录。在这种情况下，一种替代方法是使用 API 密钥进行 API 调用。

API 密钥是一个密钥值，它与您 AWS 账户中的特定 Amazon Location Service 资源以及您可以对这些资源执行的特定操作相关联。您可以在应用程序中使用 API 密钥对这些资源的 Amazon Location API 进行未经身份验证的调用。例如，如果您将 API 密钥与地图资源 myMap 以及 `GetMap*` 操作相关联，则使用该 API 密钥的应用程序将能够查看使用该资源创建的地图，并且您的账户将按您账户中的任何其他使用量收费。相同的 API 密钥不会授予更改或更新地图资源的权限，只允许使用该资源。

> **Note**
>
> API 密钥仅适用于地图、地点和路线资源，您无法修改或创建这些资源。如果您的应用程序需要访问其他资源或为未经身份验证的用户执行操作，则可以使用 Amazon Cognito 与 API 密钥一起提供访问权限或取代 API 密钥。有关更多信息，请参阅使用 Amazon Cognito 允许未经身份验证的访客访问您的应用程序。

API 密钥包含一个纯文本值，用于访问您的 AWS 账户中的一个或多个资源。如果有人复制您的 API 密钥，他们就可以访问相同的资源。为避免这种情况，您可以在创建密钥时指定可以使用 API 密钥的域。这些域名称为反向链接。如果需要，您还可以通过设置 API 密钥的到期时间来创建短期 API 密钥。

主题

API 密钥与Amazon Cognito 的比较

在类似的场景中，API 密钥和 Amazon Cognito 的使用方式类似，那么您为什么会选择使用其中一个而不是另一个呢？ 以下列表重点介绍两者之间的一些区别：

- API 密钥仅适用于地图、地点和路线资源，并且仅适用于某些操作。Amazon Cognito 可用于对大多数 Amazon Location Service API 的访问进行身份验证。

- 使用 API 密钥进行地图请求的性能通常比 Amazon Cognito 的类似场景要快。更简单的身份验证意味着在短时间内再次获得相同的地图图块时，可以减少往返服务的次数和缓存的请求。
- 借助 Amazon Cognito，您可以使用自己的身份验证流程或组合多种身份验证方法，使用 Amazon Cognito 联合身份。想要了解更多信息，请参阅 Amazon Cognito 开发人员指南 中的开始使用联合身份。

创建 API 密钥

您可以创建 API 密钥，并将其与您的 AWS 账户中的一个或多个资源关联。

您可以使用亚马逊定位服务控制台 AWS CLI、或亚马逊定位 API 创建 API 密钥。

Console

使用 Amazon Location Service 控制台创建 API 密钥

1. 在 Amazon Location 控制台中，从左侧菜单中选择 API 密钥。
2. 在 API 密钥页面上，选择创建 API 密钥。
3. 在创建 API 密钥页面中，填写以下信息：
   - 名称——您的 API 密钥的名称，例如 `MyWebAppKey`。
   - 描述——API 密钥的可选描述。
   - 资源——从下拉列表中选择要使用此 API 密钥访问的 Amazon Location 资源。您可以通过选择添加资源来添加多个资源。
   - 操作——指定您要使用此 API 密钥授权的操作。必须至少选择一个操作才能匹配所选的每种资源类型。例如，如果您选择了地点资源，则必须在地点操作下选择至少一个选项。
   - 到期时间——（可选）添加 API 密钥的到期日期和时间。有关更多信息，请参阅管理 API 密钥的生命周期。
   - 引用站点——（可选）添加一个或多个可以使用 API 密钥的域名。例如，如果 API 密钥是为了允许应用程序在网站 `example.com` 上运行，那么您可以将 `*.example.com/` 设置为允许的引用站点。
   - 标签——（可选）向 API 密钥添加标签。
4. 选择创建 API 密钥以创建 API 密钥。
5. 在 API 密钥的详情页面上，您可以看到有关您创建的 API 密钥的信息。选择显示 API 密钥以查看您在调用 Amazon Location API 时使用的密钥值。键值的格式为

v1.public.*a1b2c3d4...*。有关使用 API 密钥渲染地图的更多信息，请参阅 使用 API 密钥渲染地图。

API

使用 Amazon Location API 创建 API 密钥

使用 Amazon Location API 中的 CreateKey 操作。

以下示例是一个 API 请求，用于创建名为*ExampleKey*且没有到期日期的 API 密钥并访问单个地图资源。

```
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
   }
}
```

响应中包含访问应用程序中的资源时要使用的 API 密钥值。键值的格式为 v1.public.*a1b2c3d4...*。要了解有关使用 API 密钥渲染地图的更多信息，请参阅 使用 API 密钥渲染地图。

您也可以使用 DescribeKey API 在以后查找密钥的密钥值。

AWS CLI

使用 AWS CLI 命令创建 API 密钥

使用 create-key 命令。

以下示例创建了一个名为的 API 密钥 *ExampleKey*，该密钥没有到期日期，并且可以访问单个地图资源。

```
aws location \
  create-key \
  --key-name ExampleKey \
  --restrictions '{"AllowActions":["geo:GetMap*"],"AllowResources":
["arn:aws:geo:region:map/mapname"]}' \
  --no-expiry
```

响应中包含访问应用程序中的资源时要使用的 API 密钥值。键值的格式为 `v1.public.a1b2c3d4...`。要了解有关使用 API 密钥渲染地图的更多信息，请参阅 使用 API 密钥渲染地图。对 `create-key` 的响应看起来与以下内容类似。

```
{
    "Key": "v1.public.a1b2c3d4...",
    "KeyArn": "arn:aws:geo:region:accountId:api-key/ExampleKey",
    "KeyName": "ExampleKey",
    "CreateTime": "2023-02-06T22:33:15.693Z"
}
```

您也可以在以后使用 `describe-key` 来查找键值。以下示例说明如何调用名为`describe-key`的 API 密钥*ExampleKey*。

```
aws location describe-key \
    --key-name ExampleKey
```

使用 API 密钥调用 Amazon Location API

创建 API 密钥后，您可以使用该密钥值在应用程序中调用 Amazon Location API。

支持 API 密钥的 API 还有一个采用 API 密钥值的附加参数。例如，如果您调用 `GetPlace` API，则可以填写密钥参数，如下所示

```
GET /places/v0/indexes/IndexName/places/PlaceId?key=KeyValue
```

如果您填写此值，则无需像往常一样 AWS 使用 Sig v4 对 API 调用进行身份验证。

对于 JavaScript 开发者，您可以使用亚马逊位置JavaScript 身份验证助手来帮助使用 API 密钥对 API 操作进行身份验证。

对于移动开发者，您可以使用以下 Amazon Location 移动身份验证软件开发工具包：

- 适用于 iOS 的 Amazon Location Service 移动身份验证 SDK
- 适用于安卓的 Amazon Location Service 移动身份验证 SDK

对于 AWS CLI 用户，在使用`--key`参数时，还应使用`--no-sign-request`参数，以避免使用 Sig v4 进行签名。

> **Note**
>
> 如果您在调用 Amazon Location AWS Service 时同时包含`key`和和 Sig v4 签名，则仅使用 API 密钥。

使用 API 密钥渲染地图

您可以使用 API 密钥值在应用程序中使用渲染地图 MapLibre。这与在您直接调用的其他 Amazon Location API 中使用 API 密钥略有不同，因为 MapLibre 这些调用是为你调用的。

以下示例代码演示如何使用 API 密钥通过 MapLibre GL JS 地图控件在简单网页中呈现地图。要使此代码正常运行，请替换 *v1.public# your-api-key-value*、*us-east-1* 以及值与*ExampleMap*您匹配的字符串。 AWS 账户

```
<!-- 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 密钥的生命周期

您可以创建无限期有效的 API 密钥。但是，如果您想创建临时 API 密钥、定期轮换 API 密钥或撤销现有 API 密钥，则可以使用 API 密钥过期时间。

在创建新 API 密钥或更新现有 API 密钥时，您可以设置该 API 密钥的到期时间。

- API 密钥在到达其到期时间时将会自动停用。非活动密钥不能再用于发出地图请求。
- 您可以在 API 密钥停用 90 天后将其删除。
- 如果您有尚未删除的非活动密钥，则可以通过将到期时间更新为将来的时间来恢复该密钥。
- 要创建永久密钥，您可以删除过期时间。
- 如果您尝试停用过去 7 天内使用过的 API 密钥，系统会提示您确认是否要进行更改。如果您使用的是 Amazon Location Service API 或 AWS CLI，则除非您将`ForceUpdate`参数设置为 true，否则您将收到错误消息。

使用 Amazon Cognito 允许未经身份验证的访客访问您的应用程序

您可以使用 Amazon Cognito 身份验证作为直接使用 AWS Identity and Access Management (IAM) 进行前端软件开发工具包和直接 HTTPS 请求的替代方案。

出于以下原因，您可能需要使用这种形式的身份验证：

- 未经身份验证的用户——如果您的网站包含匿名用户，则可以使用 Amazon Cognito 身份池。想要了解更多信息，请在 the section called “使用 Amazon Cognito” 上参阅本部分。

- 您自己的身份验证——如果您想使用自己的身份验证流程或组合使用多种身份验证方法，则可以使用 Amazon Cognito 联合身份。想要了解更多信息，请参阅 Amazon Cognito 开发人员指南 中的开始使用联合身份。

Amazon Cognito 为 Web 和移动应用程序提供身份验证、授权和用户管理。您可以使用带有 Amazon Location 的 Amazon Cognito 未经身份验证的身份池作为应用程序检索限定范围内的临时凭证的一种方式。 AWS

想要了解更多信息，请参阅 Amazon Cognito 开发人员指南中的用户池入门。

> **Note**
>
> 对于移动开发者，Amazon Location 提供了适用于 iOS 和 Android 的移动身份验证软件开发工具包，有关更多信息，请参阅以下 github 存储库：
>
> - 适用于 iOS 的 Amazon Location Service 移动身份验证 SDK
> - 适用于安卓的 Amazon Location Service 移动身份验证 SDK

创建一个 Amazon Cognito 身份池

您可以创建 Amazon Cognito 身份池，允许未经身份验证的访客通过 Amazon Cognito 控制台、或 Amazon Cognito API AWS CLI访问您的应用程序。

> **Important**
>
> 您创建的资源池必须与您正在使用的 Amazon Location Service 资源位于相同 AWS 账户 且所在的 AWS 区域相同。

您可以将与未经身份验证的身份角色关联的 IAM policy 用于以下操作：

- `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 Location 操作将无效，未经身份验证的身份将无法调用这些操作。

Example

使用 Amazon Cognito 控制台创建身份池

1. 转到 Amazon Cognito 控制台。
2. 选择 Manage Identity Pools（管理身份池）。
3. 选择创建新身份池，然后输入身份池的名称。
4. 从未经验证的身份可折叠部分中，选择启用未经验证的身份的访问权限。
5. 选择创建池。
6. 选择您希望用于身份池的 IAM 角色。
7. 展开查看详情。
8. 在未经身份验证的身份下，输入角色名称。
9. 展开查看策略文档部分，然后选择编辑以添加您的策略。
10. 添加您的策略来授予对您的资源的访问权限。

    以下是地图、地点、跟踪器和路线的政策示例。要将示例用于您自己的策略，请替换*##*和 *account ID* 占位符：

    Maps policy example

    以下策略授予对名为的地图资源的只读访问权限*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"
    }
  ]
}
```

通过添加匹配的 IAM 条件，aws:referer 允许您将浏览器对资源的访问权限限制为 URL 或 URL 前缀列表。以下示例仅允许从网站 example.com 访问名为 RasterEsriImagery 的地图资源：

> **Warning**
>
> 虽然 aws:referer 可以限制访问，但它不是一种安全机制。包含公共已知的引用站点标头值是非常危险的。未经授权方可能会使用修改的浏览器或自定义浏览器提供他们选择的任何 aws:referer 值。因此，aws:referer不应使用来阻止未经授权的各方直接提出 AWS 请求。提供它只是为了允许客户保护其数字内容（如存储在 Amazon S3 中的内容），以免在未经授权的第三方站点上引用。有关更多信息，请参阅 AWS：引用站点。

```
{
    "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/*"
                ]
            }
        }
    }
]
}
```

如果你使用 Tangram 来显示地图，它不会使用 Maps API 返回的样式描述符、图像字符或精灵。相反，它是通过指向包含样式规则和必要资产的 .zip 文件来配置的。以下策略授予对*ExampleMap*为该GetMapTile操作命名的地图资源的只读访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MapsReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:GetMapTile"
      ],
      "Resource": "arn:aws:geo:region:accountID:map/ExampleMap"
    }
  ]
}
```

Places policy example

以下策略授予对名为的地点索引资源的只读访问权限 *ExamplePlaceIndex*，该资源用于按文本或位置搜索地点。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PlacesReadOnly",
      "Effect": "Allow",
      "Action": [
          "geo:SearchPlaceIndex*",
          "geo:GetPlace"
      ],
```

```
      "Resource": "arn:aws:geo:region:accountID:place-index/ExamplePlaceIndex"
    }
  ]
}
```

通过添加匹配的 IAM 条件，aws:referer 允许您将浏览器对资源的访问权限限制为 URL 或 URL 前缀列表。以下示例拒绝所有引用网站访问名*ExamplePlaceIndex*为的地点索引资源，但除外example.com。

> **Warning**
>
> 虽然 aws:referer 可以限制访问，但它不是一种安全机制。包含公共已知的引用站点标头值是非常危险的。未经授权方可能会使用修改的浏览器或自定义浏览器提供他们选择的任何 aws:referer 值。因此，aws:referer不应使用来阻止未经授权的各方直接提出 AWS 请求。提供它只是为了允许客户保护其数字内容（如存储在 Amazon S3 中的内容），以免在未经授权的第三方站点上引用。有关更多信息，请参阅 AWS：引用站点。

```
{
    "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

以下政策允许访问名为更新设备位置*ExampleTracker*的跟踪器资源。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePosition",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": "arn:aws:geo:region:accountID:tracker/ExampleTracker"
    }
  ]
}
```

通过添加匹配的 IAM 条件，`aws:referer` 允许您将浏览器对资源的访问权限限制为 URL 或 URL 前缀列表。以下示例拒绝所有引用网站访问名为*ExampleTracker*的跟踪器资源，但除外`example.com`。

> **Warning**
>
> 虽然 `aws:referer` 可以限制访问，但它不是一种安全机制。包含公共已知的引用站点标头值是非常危险的。未经授权方可能会使用修改的浏览器或自定义浏览器提供他们选择的任何 `aws:referer` 值。因此，`aws:referer`不应使用来阻止未经授权的各方直接提出 AWS 请求。提供它只是为了允许客户保护其数字内容（如存储在 Amazon S3 中的内容），以免在未经授权的第三方站点上引用。有关更多信息，请参阅 AWS：引用站点。

```
{
    "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

以下策略授予访问名*ExampleCalculator*为计算路径的路径计算器资源的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:region:accountID:route-
calculator/ExampleCalculator"
    }
  ]
}
```

通过添加匹配的 IAM 条件，aws:referer 允许您将浏览器对资源的访问权限限制为 URL 或 URL 前缀列表。以下示例拒绝所有引用网站访问名为*ExampleCalculator*的路径计算器，但除外example.com。

> **Warning**
>
> 虽然 aws:referer 可以限制访问，但它不是一种安全机制。包含公共已知的引用站点标头值是非常危险的。未经授权方可能会使用修改的浏览器或自定义浏览器提

供他们选择的任何 aws:referer 值。因此，aws:referer不应使用来阻止未经授权的各方直接提出 AWS 请求。提供它只是为了允许客户保护其数字内容（如存储在 Amazon S3 中的内容），以免在未经授权的第三方站点上引用。有关更多信息，请参阅 AWS：引用站点。

```
{
    "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**
>
> 虽然未经身份验证的身份池旨在在不安全的互联网站点上公开，但请注意，它们将被交换为标准的、有时间限制 AWS 的凭证。
> 适当确定与未经身份验证的身份池关联的 IAM 角色的范围很重要。

11. 选择允许以创建您的身份池。

生成的身份池遵循以下 <*region*>:<*GUID*>. 语法

例如：

```
us-east-1:1sample4-5678-90ef-aaaa-1234abcd56ef
```

有关 Amazon Location 的更多策略示例，请参阅 the section called “基于身份的策略示例”。

在中使用 Amazon Cognito 身份池 JavaScript

以下示例将您创建的未经身份验证的身份池交换为证书，然后使用这些证书获取地图资源的样式描述符。*ExampleMap*

```
const AWS = require("aws-sdk");

const credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: "<identity pool ID>" // for example, us-east-1:1sample4-5678-90ef-
aaaa-1234abcd56ef
});

const client = new AWS.Location({
   credentials,
   region: AWS.config.region || "<region>"
});

console.log(await client.getMapStyleDescriptor("ExampleMap").promise());
```

> **Note**
>
> 从未经身份验证的身份检索到的凭证有效期为一小时。

以下是一个在凭证到期之前自动续订凭证的函数示例。

```
async function refreshCredentials() {
  await credentials.refreshPromise();
  // schedule the next credential refresh when they're about to expire
  setTimeout(refreshCredentials, credentials.expireTime - new Date());
}
```

为了简化这项工作，您可以使用 Amazon Location JavaScript 身份验证助手。这可以代替获取凭证和刷新凭证。此示例使用适用于 JavaScript v3 的 AWS 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));
```

后续步骤

- 要修改您的角色，请前往 IAM 控制台。
- 要管理您的身份池，请前往 Amazon Cognito 控制台。

监控 Amazon Location Service

使用 Amazon Location Service 时，您可以使用以下方法监控一段时间内的使用情况和资源使用情况：

- 亚马逊 CloudWatch。监控您的 Amazon Location Service 资源，并近乎实时地提供指标和统计数据。
- AWS CloudTrail。提供对所有调用 Amazon Location Service API 的事件跟踪。

此部分提供有关使用这些服务的信息。

主题

- 使用亚马逊监控 Amazon Location Service CloudWatch

- 使用 AWS CloudTrail 进行日志记录和监控

使用亚马逊监控 Amazon Location Service CloudWatch

Amazon 近乎实时地 CloudWatch 监控您的AWS资源和您运行AWS的应用程序。您可以使用监控 Amazon Location 资源 CloudWatch，该资源可以近乎实时地收集原始数据并将指标处理成有意义的统计数据。您可以查看最长 15 个月的历史信息，也可以搜索指标以在亚马逊 CloudWatch 控制台中查看，以更深入地了解您的亚马逊位置资源。您还可以通过定义阈值来设置警报，并在达到相应阈值时发送通知或执行操作。

有关更多信息，请参阅 Amazon CloudWatch 用户指南

主题

- 导出到亚马逊的 Amazon Location Service 指标 CloudWatch
- 查看 Amazon Location Service
- 为 Amazon Location Service 指标创建 CloudWatch 警报
- CloudWatch 用于根据配额监控使用情况
- CloudWatch Amazon Location Service 的指标示例

导出到亚马逊的 Amazon Location Service 指标 CloudWatch

指标是导出到 CloudWatch的时间顺序数据点。维度是用于标识指标的名称/值对。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用 CloudWatch 指标和CloudWatch维度。

以下是 Amazon Location Service 在`AWS/Location`命名空间 CloudWatch 中导出的指标。

指标	描述
`CallCount`	对给定 API 端点的调用次数。 有效维度：Amazon Location Service API 名称 有效统计数据：Sum 单位：计数
`ErrorCount`	对给定 API 端点的调用所产生的错误响应数量。

指标	描述
	有效维度：Amazon Location Service API 名称 有效统计数据：Sum 单位：计数
SuccessCount	成功调用给定 API 端点的次数。 有效维度：Amazon Location Service API 名称 有效统计数据：Sum 单位：计数
CallLatency	向给定 API 端点发出调用时，该操作处理和返回响应所花费的时间。 有效维度：Amazon Location Service API 名称 有效统计数据：平均值 单位：毫秒

查看 Amazon Location Service

您可以在 Amazon CloudWatch 控制台上或使用 Amazon CloudWatch API 查看 Amazon Location Service 的指标。

使用 CloudWatch 控制台查看指标

Example

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择指标。
3. 在所有指标选项卡上，选择 Amazon Location 命名空间。
4. 选择要查看的指标类型。
5. 选择指标，然后添加到图表。

有关更多信息，请参阅 Amazon CloudWatch 用户指南中的查看可用指标。

为 Amazon Location Service 指标创建 CloudWatch 警报

您可以使用 CloudWatch 对您的 Amazon Location Service 指标设置警报。例如，您可以在中创建警报， CloudWatch 以便在错误计数出现峰值时发送电子邮件。

以下主题从较高层面上概括介绍了如何使用 CloudWatch 设置警报。有关详细说明，请参阅 Amazon CloudWatch 用户指南中的使用警报。

使用 CloudWatch 控制台设置警报

Example

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择警报。
3. 选择创建警报。
4. 选择选择指标。
5. 在所有指标选项卡上，选择 Amazon Location 命名空间。
6. 选择指标类别。
7. 找到您要为其创建警报的指标所在行，然后选中该行旁边的复选框。
8. 选择选择指标。
9. 在指标下，填写值。
10. 指定警报条件。
11. 请选择 Next（下一步）。
12. 如果您想在满足警报条件时发送通知，请执行以下操作：
    - 在警报状态触发器下，选择警报状态以提示发送通知。
    - 在选择 SNS 主题下，选择创建新主题以创建新的 Amazon Simple Notification Service (Amazon SNS) 主题。输入主题名称和要向其发送通知的电子邮件。
    - 在发送通知下，输入要向其发送通知的其他电子邮件地址。
    - 选择 Add notification（添加通知）。此列表将保存下来并会在将来的警报字段中显示出来。
13. 完成后选择下一步。
14. 输入警报的名称和描述，然后选择下一步。
15. 确认警报详情，然后选择下一步。

> **Note**
>
> 创建新的 Amazon SNS 主题时，必须先验证电子邮件地址，然后才能发送通知。如果电子邮件未通过验证，则当警报因状态变化而启动时，将不会收到通知。

有关如何使用 CloudWatch 控制台设置警报的更多信息，请参阅 Amazon CloudWatch 用户指南中的创建发送电子邮件的警报。

CloudWatch 用于根据配额监控使用情况

您可以创建 Amazon CloudWatch 警报，以便在给定配额的使用率超过可配置的阈值时通知您。这使您能够识别何时接近配额限制，并调整利用率以避免成本超支，或者在需要时请求增加配额。有关 CloudWatch 如何使用监控配额的信息，请参阅 Amazon CloudWatch 用户指南中的可视化服务配额和设置警报。

CloudWatch Amazon Location Service 的指标示例

您可以使用 GetMetricDataAPI 来检索 Amazon 位置的指标。

- 例如，您可以在数字下降时监控 `CallCount` 并设置警报。

  监控 `SendDeviceLocation` 的 `CallCount` 指标可以帮助您深入了解被跟踪的资产。如果 `CallCount` 下降，则意味着被追踪的资产（例如卡车车队）已停止发送其当前位置。为此设置警报可以帮助通知您发生了问题。
- 再举一个例子，您可以监控 `ErrorCount` 并设置警报，以了解何时出现数量激增。

  跟踪器必须与地理围栏集合相链接，才能根据地理围栏评估设备位置。如果您的设备群需要持续更新位置，看到 `BatchEvaluateGeofence` 或 `BatchPutDevicePosition` 的 `CallCount` 为零表示更新不再流动。

以下是创建地图资源的示例输出 GetMetricData，其中包含`ErrorCount`用于创建地图资源的指标。`CallCount`

```
{
  "StartTime": 1518867432,
  "EndTime": 1518868032,
  "MetricDataQueries": [
    {
```

```
      "Id": "m1",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Location",
          "MetricName": "CallCount",
          "Dimensions": [
            {
              "Name": "SendDeviceLocation",
              "Value": "100"
            }
          ]
        },
        "Period": 300,
        "Stat": "SampleCount",
        "Unit": "Count"
      }
    },
    {
      "Id": "m2",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Location",
          "MetricName": "ErrorCount",
          "Dimensions": [
            {
              "Name": "AssociateTrackerConsumer",
              "Value": "0"
            }
          ]
        },
        "Period": 1,
        "Stat": "SampleCount",
        "Unit": "Count"
      }
    }
  ]
}
```

使用 AWS CloudTrail 进行日志记录和监控

AWS CloudTrail是一项提供用户、角色或服务所执行操作记录的AWS服务。 CloudTrail 将所有 API 调用记录为事件。您可以将 Amazon Location Service 与配合使用 CloudTrail 来监控您的 API 调用，其中包括来自亚马逊定位服务控制台的调用和对亚马逊定位服务 API 操作的 AWS SDK 调用。

创建跟踪时，您可以允许将 CloudTrail 事件持续传输到 S3 存储桶，包括针对 Amazon Location Service 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。通过收集的信息 CloudTrail，您可以确定向 Amazon Location Service 发出的请求、发出请求的 IP 地址、谁提出了请求、何时提出请求以及其他详细信息。

有关的更多信息 CloudTrail，请参阅《AWS CloudTrail用户指南》。

主题

- 中的 Amazon Location Service 信息 CloudTrail
- 了解 Amazon Location Service 日志文件条目

中的 Amazon Location Service 信息 CloudTrail

CloudTrail 在您创建AWS账户时已在您的账户上启用。当 Amazon Location Service 中发生活动时，该活动会与其他AWS服务 CloudTrail 事件一起记录在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录 AWS 账户中的事件（包括 Amazon Location Service 的事件），请创建跟踪记录。跟踪允许 CloudTrail 将日志文件传送到 S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有 AWS 区域。此跟踪在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 S3 存储桶。此外，您可以配置其他AWS服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。

有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

亚马逊定位服务的所有操作均由亚马逊定位服务 API 参考记录 CloudTrail 并记录在案。例如，调用`UpdateTracker`和`DescribeTracker`操作会在 CloudTrail 日志文件中生成条目。`CreateTracker`

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于确定发出的请求是否：

- 使用根或 AWS Identity and Access Management (IAM) 用户凭证。
- 使用角色或联合身份用户的临时安全凭证。

- 通过另一个 AWS 服务。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Amazon Location Service 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 S3 存储桶或 Amazon L CloudWatch ogs。有关更多信息，请参阅《AWS CloudTrail用户指南》中的使用 CloudTrail 日志文件。

CloudTrail 日志文件包含一个或多个日志条目。一个事件表示一个来自任何源的请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。

> **Note**
>
> CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。要确定操作顺序，请使用 `eventTime`。

以下示例显示了一个 CloudTrail 日志条目，该条目演示了创建跟踪器资源的`CreateTracker`操作。

```
{
    "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",
}
```

下图显示了 DescribeTracker 操作的日志条目，该条目返回了跟踪器资源的详细信息。

```
{
    "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",
}
```


使用 AWS CloudFormation 创建Amazon Location Service 资源

Amazon Location Service 与 AWS CloudFormation 集成，后者是一项服务，可帮助您对 AWS 资源进行建模和设置，这样您只需花较少的时间来创建和管理资源与基础设施。您可以创建一个描述所需的全部 AWS 资源的模板（例如 Amazon Location 资源），将为您 AWS CloudFormation 预置和配置这些资源。

在您使用 AWS CloudFormation 时，可重复使用您的模板来不断地重复设置您的 Amazon Location 资源。仅描述您的资源一次，然后在多个 Amazon Web Services 账户和区域中反复配置相同的资源。

Amazon Location 和 AWS CloudFormation 模板

要为 Amazon Location 和相关服务设置和配置资源，您必须了解 AWS CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板可描述您要在 AWS CloudFormation 堆栈中调配的资

源。如果您不熟悉 JSON 或 YAML，可以在 AWS CloudFormation Designer 的帮助下开始使用 AWS CloudFormation 模板。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的什么是 AWS CloudFormation Designer？。

Amazon Location 支持在 AWS CloudFormation 中创建以下资源类型：

- AWS::Location::Map
- AWS::Location::PlaceIndex
- AWS::Location::RouteCalculator
- AWS::Location::Tracker
- AWS::Location::TrackerConsumer
- AWS::Location::GeofenceCollection

有关更多信息（包括 Amazon Location 资源的 JSON 和 YAML 模板示例），请参阅 AWS CloudFormation 用户指南中的 Amazon Location Service 资源类型参考。

了解有关 AWS CloudFormation 的更多信息

要了解有关 AWS CloudFormation 的更多信息，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation API Reference
- AWS CloudFormation 命令行界面用户指南

Amazon Location Service 中的安全

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方 AWS 的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Location Service 的合规计划，请参阅按合规计划AWS 提供的范围内的AWS 服务划分的范围内服务。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon Location 时应用责任共担模式。以下主题说明如何配置 Amazon Location 以实现您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Amazon Location 资源。

主题

Amazon Location Service 中的数据保护

分担责任模式 AWS 分担责任模型适用于 Amazon Location Service 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础架构上的内

容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私FAQ。 有关欧洲数据保护的信息，请参阅责任AWS 共担模型和AWS安全GDPR博客上的博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭据并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用SSL/TLS与 AWS 资源通信。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 使用API进行设置和用户活动记录 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或访问时需要 FIPS 140-3 经过验证的加密模块API，请使用端点。FIPS有关可用FIPS端点的更多信息，请参阅联邦信息处理标准 (FIPS) 140-3。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、、API或 AWS 服务 使用 Amazon Location 或其他网站时 AWS SDKs。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您URL向外部服务器提供，我们强烈建议您不要在中包含凭据信息，URL以验证您对该服务器的请求。

数据隐私

使用 Amazon Location Service，您可以保留对组织数据的控制权。Amazon Location 会删除客户元数据和账户信息，从而匿名化发送给数据提供程序的所有查询。

Amazon Location 不使用数据提供程序进行跟踪和地理围栏。这意味着您的敏感数据仍保留在您的 AWS 帐户中。这有助于保护敏感的位置信息（例如设施、资产和人员位置）免受第三方的侵害，保护用户隐私，并降低应用程序的安全风险。

有关更多信息，请参阅AWS 数据隐私FAQ。

Amazon Location 的数据保留

以下特征与 Amazon Location 如何收集和存储服务数据有关：

- Amazon Location Servic e 追APIs踪器 — 当您使用跟踪器跟踪实体的位置时，可以存储其坐标。设备位置会存储 30 天，然后才会被服务删除。
- Amazon Loc ation Service Geofences — 当您使用地理围栏APIs定义感兴趣区域时，该服务会存储您提供的几何图形。必须明确删除它们。

> **Note**
>
> 删除您的 AWS 账户会删除其中的所有资源。有关更多信息，请参阅AWS 数据隐私FAQ。

Amazon Location Service 中的静态数据加密

Amazon Location Service 默认提供加密，以使用 AWS 自有的加密密钥保护敏感的静态客户数据。

- AWS 自有密钥 — Amazon Location 默认使用这些密钥来自动加密个人身份数据。您无法查看、管理或使用 AWS 自有密钥，也无法审核其使用情况。但是，无需采取任何措施或更改任何计划即可保护用于加密数据的密钥。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS 自有密钥。

默认情况下，静态数据加密有助于减少保护敏感数据时涉及的操作开销和复杂性。同时，它还支持构建符合严格加密合规性和监管要求的安全应用程序。

虽然您无法禁用此加密层或选择其他加密类型，但您可以在创建跟踪器和地理围栏收集资源时选择客户管理的密钥，从而在现有的 AWS 加密密钥上添加第二层加密：

- 客户托管密钥 — Amazon Location 支持使用您创建、拥有和管理的对称客户托管密钥，在现有 AWS 自有加密的基础上添加第二层加密。由于您可以完全控制这层加密，因此可以执行以下任务：
  - 制定和维护关键策略
  - 制定和维护IAM政策和补助金
  - 启用和禁用密钥策略
  - 轮换加密材料
  - 添加标签
  - 创建密钥别名
  - 计划删除密钥

  有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的客户托管密钥。

下表汇总了 Amazon Location 如何加密个人身份数据。

数据类型	AWS 自有密钥加密	客户托管密钥加密（可选）
`Position` 包含设备位置详细信息的造点几何体。	已启用	已启用
`PositionProperties` 一组与位置更新关联的键值对。	已启用	已启用
`GeofenceGeometry` 表示地理围栏区域的多边形地理围栏几何。	已启用	已启用
`DeviceId` 将设备位置更新上传到跟踪器资源时指定的设备标识符。	已启用	不支持
`GeofenceId` 在给定地理围栏集合中存储地理围栏几何或一批地理围栏时指定的标识符。	已启用	不支持

> **Note**
>
> Amazon Location 使用 AWS 自有密钥自动启用静态加密，从而免费保护个人身份数据。但是，使用客户管理的密钥需要 AWS KMS 付费。有关定价的更多信息，请参阅 AWS Key Management Service 定价。

有关的更多信息 AWS KMS，请参阅什么是 AWS Key Management Service？

Amazon Location Service 如何使用补助金 AWS KMS

Amazon Location 需要授权，才能使用客户托管密钥。

当您创建使用客户托管密钥加密的追踪器资源或地理围栏集合时，Amazon Location 会通过向发送CreateGrant请求来代表您创建授权。 AWS KMS中的授权 AWS KMS 用于让 Amazon Location 访问客户账户中的KMS密钥。

Amazon Location 需要授权，才能将客户托管的密钥用于以下内部操作：

- 向发送DescribeKey请求， AWS KMS 以验证在创建跟踪器或地理围栏集合时输入的对称客户托管KMS密钥 ID 是否有效。
- 向发送GenerateDataKeyWithoutPlaintext请求 AWS KMS 以生成由您的客户托管密钥加密的数据密钥。
- 将 Decrypt 请求发送 AWS KMS 到以解密加密的数据密钥，以便它们可用于加密您的数据。

您可以随时撤销授予访问权限，或删除服务对客户托管密钥的访问权限。如果您这样做，Amazon Location 将无法访问由客户托管的密钥加密的任何数据，这会影响依赖于该数据的操作。例如，如果您尝试从 Amazon Location 无法访问的加密跟踪器中获取设备位置，则该操作将返回 `AccessDeniedException` 错误。

创建客户托管密钥

您可以使用 AWS Management Console、或，创建对称的客户托管密钥。 AWS KMS APIs

创建对称的客户托管密钥

按照《AWS Key Management Service 开发人员指南》中创建对称的客户托管密钥的步骤进行操作。

密钥策略

密钥策略控制对客户托管密钥的访问。每个客户托管式密钥必须只有一个密钥政策，其中包含确定谁可以使用密钥以及如何使用密钥的声明。创建客户托管式密钥时，可以指定密钥政策。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的管理对客户托管密钥的访问。

要将您的客户托管密钥与您的 Amazon Location 资源一起使用，密钥政策中必须允许以下API操作：

- `kms:CreateGrant` – 添加客户托管密钥授权。授予对指定KMS密钥的控制权限，从而允许访问授予 Amazon Location 所需的操作。有关使用授权的更多信息，请参阅 AWS Key Management Service 开发人员指南。

这将允许 Amazon Location 执行以下操作：

- 调用 GenerateDataKeyWithoutPlainText 生成加密的数据密钥并将其存储，因为数据密钥不会立即用于加密。
- 调用 Decrypt 使用存储的加密数据密钥访问加密数据。
- 设置停用委托人，以允许服务 RetireGrant。

- kms:DescribeKey——提供客户托管式密钥详细信息以允许 Amazon Location 验证密钥。

以下是您可以为 Amazon Location 添加的政策声明示例：

```
 "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" : "*"
    }
  ]
```

有关在策略中指定权限的更多信息，请参阅 AWS Key Management Service 开发人员指南。

有关密钥访问故障排除更多的信息，请参阅 AWS Key Management Service 开发人员指南。

指定 Amazon Location 的客户托管密钥

您可以将客户托管密钥指定为以下资源的第二层加密：

- 跟踪器资源
- 地理围栏集合

创建资源时，您可以通过输入 KMSID 来指定数据密钥，Amazon Location 使用该ID来加密资源存储的可识别个人数据。

- KMSID — AWS KMS 客户托管密钥的密钥标识符。输入密钥 ID、密钥ARN、别名或别名ARN。

Amazon Location Service 加密上下文

加密上下文是一组可选的键值对，包含有关数据的其他上下文信息。

AWS KMS 使用加密上下文作为其他经过身份验证的数据来支持经过身份验证的加密。当您在加密数据的请求中包含加密上下文时，会将加密上下文 AWS KMS 绑定到加密数据。要解密数据，您需要在请求中包含相同的加密上下文。

Amazon Location Service 加密上下文

Amazon Location 在所有 AWS KMS 加密操作中使用相同的加密上下文，其中密钥是`aws:geo:arn`，值是资源 Amazon 资源名称 (ARN)。

Example

```
"encryptionContext": {
    "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/SAMPLE-
GeofenceCollection"
}
```

使用加密上下文进行监控

当您使用对称的客户托管密钥来加密您的跟踪器或地理围栏集合时，您还可以使用审计记录和日志中的加密上下文来识别客户托管密钥的使用情况。加密上下文还会显示在AWS CloudTrail 或 Amazon Logs 生成的 CloudWatch 日志中。

使用加密上下文控制对客户托管式密钥的访问

您可以使用密钥策略和IAM策略中的加密上下文`conditions`来控制对称客户托管密钥的访问权限。您还可以在授权中使用加密上下文限制。

Amazon Location 在授权中使用加密上下文限制，以控制对您账户或区域中的客户管理密钥的访问。授权约束要求授权允许的操作使用指定的加密上下文。

Example

以下是密钥策略声明示例，用于授予对特定加密上下文的客户托管密钥的访问权限。此策略声明中的条件要求授权具有指定加密上下文的加密上下文约束。

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

监控您的 Amazon Location Service 的加密密钥

当您将 AWS KMS 客户托管密钥与 Amazon Location Service 资源一起使用时，您可以使用AWS CloudTrail或 Amazon CloudWatch Logs 来跟踪亚马逊位置发送到的请求 AWS KMS。

以下示例是CreateGrant、GenerateDataKeyWithoutPlainTextDecrypt、和监控 Amazon L DescribeKey ocation 为访问由您的客户托管密钥加密的数据而调用的KMS操作 AWS CloudTrail 的事件：

CreateGrant

当您使用 AWS KMS 客户托管密钥加密您的追踪器或地理围栏收集资源时，Amazon Location 会代表您发送访问您 AWS 账户中KMS密钥的CreateGrant请求。Amazon Location 创建的授权特定于与 AWS KMS 客户托管密钥关联的资源。此外，当您删除资源时，Amazon Location 会使用 RetireGrant 操作来删除授权。

以下示例事件记录了 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

当您为追踪器或地理围栏收集资源启用 AWS KMS 客户托管密钥时，Amazon Location 会创建一个唯一的表密钥。它向发送GenerateDataKeyWithoutPlainText请求 AWS KMS ，指定资源的AWS KMS客户托管密钥。

以下示例事件记录了 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

当您访问加密的跟踪器或地理围栏集合时，Amazon Location 会地理围栏 `Decrypt` 操作以使用存储的加密数据密钥来访问加密数据。

以下示例事件记录了 `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 使用 DescribeKey 操作来验证账户和地区中是否存在与您的跟踪器或地理围栏集合关联的 AWS KMS 客户托管密钥。

以下示例事件记录了 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"
}
```

了解更多

以下资源提供有关静态数据加密的更多信息。

- 有关 AWS Key Management Service 基本概念的更多信息，请参阅 《AWS Key Management Service 开发人员指南》。
- 有关安全最佳实践的更多信息 AWS Key Management Service，请参阅《AWS Key Management Service 开发人员指南》。

Amazon Location Service 的传输中数据加密

Amazon Location 使用传输层安全 (TLS) 1.2 加密协议自动加密所有网络间数据，从而保护传输中的数据在往返服务时。发送到 Amazon Location Servic APIs e 的直接HTTPS请求使用AWS签名版本 4 算法进行签名，以建立安全连接。

适用于 Amazon Location Service 的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可以帮助管理员安全地控制对 AWS 资源的访问权限。IAM管理员控制谁可以通过身份验证（登录）和授权（拥有权限）来使用 Amazon Location 资源。IAM无需支付额外费用即可使用。 AWS 服务

主题

受众

您使用 AWS Identity and Access Management (IAM) 的方式会有所不同，具体取决于您在 Amazon Location 中所做的工作。

服务用户——如果您使用 Amazon Location Service 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon Location 功能来完成任务，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon Location 中的特征，请参阅Amazon Location Service 身份和访问权限故障排查。

服务管理员——如果您在公司负责管理 Amazon Location 资源，您可能对 Amazon Location 具有完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon Location 功能和资源。然后，您必须向IAM管理员提交更改服务用户权限的请求。查看此页面上的信息以了解的基本概念IAM。要详细了解贵公司如何使用 Amazon IAM Location，请参阅Amazon Location Service 是如何使用的 IAM。

IAM管理员 — 如果您是IAM管理员，则可能需要详细了解如何编写策略来管理 Amazon Location 的访问权限。要查看您可以在中使用的基于Amazon Location身份的政策示例IAM，请参阅。Amazon Location Service 基于身份的策略示例

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 AWS 账户根用户、IAM用户身份或通过担任IAM角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM身份中心）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员之前使用IAM角色设置了联合身份。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅IAM用户指南中的签署 AWS API请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅用户指南中的多重身份验证和AWS IAM Identity Center 用户指南 AWS中的使用多因素身份验证 (MFA)。IAM

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以 root 用户身份登录的任务的完整列表，请参阅《用户指南》中的 “需要根用户凭据的IAM任务”。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 Ident IAM ity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户

和应用程序中使用。有关IAM身份中心的信息，请参阅什么是IAM身份中心？ 在《AWS IAM Identity Center 用户指南》中。

IAM 用户和组

IAM用户是您内部 AWS 账户 对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时证书，而不是创建拥有密码和访问密钥等长期凭证的IAM用户。但是，如果您有需要IAM用户长期凭证的特定用例，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM用户指南》中的定期轮换需要长期凭证的用例的访问密钥。

IAM群组是指定IAM用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并授予该群组管理IAM资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM用户指南》中的何时创建IAM用户（而不是角色）。

IAM角色

IAM角色是您内部具有特定权限 AWS 账户 的身份。它与IAM用户类似，但与特定人员无关。您可以 AWS Management Console 通过切换IAM角色在中临时扮演角色。您可以通过调用 AWS CLI 或 AWS API操作或使用自定义操作来代入角色URL。有关使用角色的方法的更多信息，请参阅《IAM用户指南》中的使用IAM角色。

IAM具有临时证书的角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅IAM用户指南中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，Ident IAM ity Center 会将权限集关联到中的IAM角色。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时IAM用户权限-IAM 用户或角色可以代入一个IAM角色，为特定任务临时获得不同的权限。
- 跨账户访问-您可以使用IAM角色允许其他账户中的某人（受信任的委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序EC2或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS)-当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。
  - 服务角色-服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅《IAM用户指南》 AWS 服务中的创建角色以向委派权限。
  - 服务相关角色-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon 上运行的应用程序 EC2 — 您可以使用IAM角色管理在EC2实例上运行并发出 AWS CLI 或 AWS API请求的应用程序的临时证书。这比在EC2实例中存储访问密钥更可取。要为EC2实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在EC2实例上运行的程序获得临时证书。有关更多信息，请参阅IAM用户指南中的使用IAM角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用IAM角色还是使用IAM用户，请参阅《用户指南》中的何时创建IAM角色（而不是IAM用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以JSON文档 AWS 形式存储在中。有关JSON策略文档结构和内容的更多信息，请参阅《IAM用户指南》中的JSON策略概述。

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入这些角色。

IAM无论您使用何种方法执行操作，策略都会定义该操作的权限。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或获取角色信息 AWS API。

基于身份的策略

基于身份的策略是可以附加到身份（例如IAM用户、用户组或角色）的JSON权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM用户指南中的创建IAM策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅《IAM用户指南》中的在托管策略和内联策略之间进行选择。

基于资源的策略

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括IAM角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略IAM中使用 AWS 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACLs与基于资源的策略类似，尽管它们不使用JSON策略文档格式。

Amazon S3 AWS WAF、和亚马逊VPC就是支持的服务示例ACLs。要了解更多信息ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体（IAM用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中

的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM用户指南》中的IAM实体的权限边界。

- 服务控制策略 (SCPs)-SCPs 是为中的组织或组织单位 (OU) 指定最大权限的JSON策略 AWS Organizations。 AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。对成员账户中的实体（包括每个实体）的权限进行了SCP限制 AWS 账户根用户。有关 Organization SCPs s 和的更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅IAM用户指南中的策略评估逻辑。

Amazon Location Service 是如何使用的 IAM

在使用管理IAM对亚马逊位置的访问权限之前，请先了解亚马逊位置有哪些IAM功能可供使用。

IAM您可以在 Amazon Location Service 中使用的功能

IAM功能	Amazon Location 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
ACLs	不支持
ABAC（策略中的标签）	是

IAM功能	Amazon Location 支持
临时凭证	是
主体权限	否
服务角色	否
服务相关角色	否

要全面了解 Amazon Location 和其他 AWS 服务如何与大多数IAM功能配合使用，请参阅IAM用户指南IAM中与之配合使用的AWS 服务。

Amazon Location 基于身份的策略

支持基于身份的策略：是

基于身份的策略是可以附加到身份（例如IAM用户、用户组或角色）的JSON权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM用户指南中的创建IAM策略。

使用IAM基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可以在JSON策略中使用的所有元素，请参阅IAM用户指南中的IAMJSON策略元素参考。

Amazon Location 基于身份的策略示例

要查看 Amazon Location 基于身份的策略的示例，请参阅 Amazon Location Service 基于身份的策略示例。

Amazon Location 内基于资源的策略

支持基于资源的策略：否

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括IAM角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或另一个账户中的IAM实体指定为基于资源的策略中的委托人。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的IAM管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM用户指南》IAM中的跨账户资源访问权限。

Amazon Location 的策略操作

支持策略操作：是

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

JSON策略`Action`元素描述了可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API操作同名。也有一些例外，例如没有匹配API操作的仅限权限的操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

有关 Amazon Location 操作的列表，请参阅《服务授权参考》中的 Amazon Location Service 定义的操作。

Amazon Location 中的策略操作在操作前面使用以下前缀：

```
geo
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "geo:action1",
      "geo:action2"
         ]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 `Get` 开头的所有操作，包括以下操作：

```
"Action": "geo:Get*"
```

要查看 Amazon Location 基于身份的策略的示例，请参阅 Amazon Location Service 基于身份的策略示例。

Amazon Location 的策略资源

支持策略资源：是

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

`Resource`JSON策略元素指定要应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。最佳做法是使用资源的 Amazon 资源名称 (ARN) 来指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看亚马逊定位资源类型及其列表ARNs，请参阅服务授权参考中的 Amazon Location Service 定义的资源。要了解您可以为每种资源指定哪些操作，请参阅 Amazon Location Service 定义的操作。ARN

要查看 Amazon Location 基于身份的策略的示例，请参阅 Amazon Location Service 基于身份的策略示例。

Amazon Location 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在资源上标有IAM用户的用户名时，您才能向IAM用户授予访问该资源的权限。有关更多信息，请参阅《IAM用户指南》中的IAM策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅《IAM用户指南》中的AWS 全局条件上下文密钥。

要查看 Amazon Location 条件键的列表，请参阅《服务授权参考》中的 Amazon Location Service 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Location Service 定义的操作。

Amazon Location 支持条件键，允许您在政策声明中允许或拒绝访问特定地理围栏或设备。以下条件键可用于：

- `geo:GeofenceIds` 用于地理围栏操作。类型是 `ArrayOfString`。
- `geo:DeviceIds` 用于跟踪器操作。类型是 `ArrayOfString`。

可以在您的IAM策略`geo:GeofenceIds`中使用以下操作：

- `BatchDeleteGeofences`
- `BatchPutGeofences`
- `GetGeofence`
- `PutGeofence`

可以在您的IAM策略`geo:DeviceIds`中使用以下操作：

- `BatchDeleteDevicePositionHistory`
- `BatchGetDevicePosition`
- `BatchUpdateDevicePosition`
- `GetDevicePosition`
- `GetDevicePositionHistory`

> **Note**
>
> 您不能将这些条件键与 `BatchEvaluateGeofences`、`ListGeofences` 或 `ListDevicePosition` 操作配合使用。

要查看 Amazon Location 基于身份的策略的示例，请参阅 Amazon Location Service 基于身份的策略示例。

ACLs在亚马逊所在地

支持ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACLs与基于资源的策略类似，尽管它们不使用JSON策略文档格式。

ABAC使用 Amazon 位置

支持ABAC（策略中的标签）：是

基于属性的访问控制 (ABAC) 是一种基于属性定义权限的授权策略。在中 AWS，这些属性称为标签。您可以将标签附加到IAM实体（用户或角色）和许多 AWS 资源。为实体和资源添加标签是的第一步。ABAC然后，您可以设计ABAC策略，允许在委托人的标签与他们尝试访问的资源上的标签匹配时进行操作。

ABAC在快速增长的环境中很有用，也有助于解决策略管理变得繁琐的情况。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关的更多信息ABAC，请参阅什么是ABAC？ 在《IAM用户指南》中。要查看包含设置步骤的教程ABAC，请参阅IAM用户指南中的使用基于属性的访问控制 (ABAC)。

有关标记 Amazon Location 资源的更多信息，请参阅 为您的 Amazon Location Service 资源添加标签。

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅基于标签控制对资源的访问。

将临时凭证用于 Amazon Location

支持临时凭证：是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关其他信息，包括哪些 AWS 服务 适用于临时证书 AWS 服务 ，请参阅《IAM用户指南》IAM中的 “适用于临时证书”。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM用户指南》中的切换到角色（控制台）。

您可以使用 AWS CLI 或手动创建临时证书 AWS API。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅中的临时安全证书IAM。

Amazon Location 的跨服务主体权限

支持转发访问会话 (FAS)：否

当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。

Amazon Location 的服务角色

支持服务角色：否

服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅《IAM用户指南》 AWS 服务中的创建角色以向委派权限。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Amazon Location 的功能。仅当 Amazon Location 提供相关指导时才编辑服务角色。

Amazon Location 的服务相关角色

支持服务相关角色：否

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅与之配合IAM使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

Amazon Location Service 如何处理未经身份验证的用户

使用 Amazon Location Service 的许多场景（包括在网络或移动应用程序中显示地图）都需要允许尚未登录的用户进行访问IAM。对于这些未经身份验证的场景，您有两种选择。

- 使用API密钥-要向未经身份验证的用户授予访问权限，您可以创建API密钥来授予对您的 Amazon Location Service 资源的只读访问权限。在您不想对每个用户进行身份验证的情况下，这很有用。例如，Web 应用程序。有关API密钥的更多信息，请参阅允许未经身份验证的访客使用 API 密钥访问您的应用程序。
- 使用 Amazon Cognito — API 密钥的替代方法是使用 Amazon Cognito 授予匿名访问权限。Amazon Cognito 允许您使用IAM策略创建更丰富的授权，以定义未经身份验证的用户可以执行的操作。有关使用 Amazon Cognito 的更多信息，请参阅 使用 Amazon Cognito 允许未经身份验证的访客访问您的应用程序。

有关向未经身份验证的用户提供访问权限的概述，请参阅 授予对 Amazon Location Service 的访问权限。

Amazon Location Service 基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Amazon Location 资源的权限。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或执行任务 AWS API。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入这些角色。

要了解如何使用这些示例策略文档创建IAM基于身份的JSON策略，请参阅IAM用户指南中的创建IAM策略。

有关 Amazon Location 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 Amazon Location Service 的操作、资源和条件密钥。ARNs

主题

- 策略最佳实践
- 使用 Amazon Location 控制
- 允许用户查看他们自己的权限

- 在策略中使用 Amazon Location Service Service 资源
- 更新设备位置的权限
- 跟踪器资源的只读策略
- 创建地理围栏的策略
- 地理围栏的只读策略
- 渲染地图资源的权限
- 允许搜索操作的权限
- 路线计算器的只读策略
- 根据条件键控制资源访问权限
- 基于标签控制对资源的访问

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon Location 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM用户指南》中的AWS 托AWS 管策略或工作职能托管策略。
- 应用最低权限权限-使用IAM策略设置权限时，仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用应用权限IAM的更多信息，请参阅IAM用户指南IAM中的策略和权限。
- 使用IAM策略中的条件进一步限制访问权限-您可以在策略中添加条件以限制对操作和资源的访问权限。例如，您可以编写一个策略条件来指定所有请求都必须使用发送SSL。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM用户指南》中的IAMJSON策略元素：条件。
- 使用 A IAM ccess Analyzer 验证您的IAM策略以确保权限的安全性和功能性 — A IAM ccess Analyzer 会验证新的和现有的策略，以便策略符合IAM策略语言 (JSON) 和IAM最佳实践。IAMAccess Analyzer 提供了 100 多项策略检查和可行的建议，可帮助您制定安全和实用的策略。有关更多信息，请参阅《IAM用户指南》中的 IAMAccess Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果您的场景需要IAM用户或 root 用户 AWS 账户，请打开MFA以提高安全性。要要求MFA何时调用API操作，请在策略中添加MFA条件。有关更多信息，请参阅《IAM用户指南》中的配置MFA受保护的API访问权限。

有关中最佳做法的更多信息IAM，请参阅《IAM用户指南》IAM中的安全最佳实践。

使用 Amazon Location 控制

要访问 Amazon Location Service 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 Amazon Location 资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

您无需为仅拨打 AWS CLI 或的用户设置最低控制台权限 AWS API。相反，只允许访问与他们尝试执行的API操作相匹配的操作。

为确保用户和角色可以使用 Amazon Location 控制台，请将以下策略添加到实体上。有关更多信息，请参阅《用户指南》中的向IAM用户添加权限。

以下政策允许访问 Amazon Location Service 控制台，以便能够在您的 AWS 账户中创建、删除、列出和查看有关 Amazon Location 资源的详细信息。

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

或者，您可以授予只读权限以简化只读访问。使用只读权限时，如果用户尝试写入操作（例如创建或删除资源），则会显示一条错误消息。有关示例，请参阅 the section called “跟踪器的只读读策略”。

允许用户查看他们自己的权限

此示例说明如何创建允许IAM用户查看附加到其用户身份的内联和托管策略的策略。此策略包括在控制台上或使用或以编程方式完成此操作的 AWS CLI 权限。 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": "*"
        }
    ]
}
```

在策略中使用 Amazon Location Service Service 资源

Amazon Location Service 对资源使用以下前缀：

Amazon Location 资源前缀

资源	资源前缀
地图资源	`map`
放置资源	`place-index`

资源	资源前缀
路线资源	route-calculator
跟踪器资源	tracker
地理围栏集合资源	geofence-collection

使用以下ARN语法：

```
arn:Partition:geo:Region:Account:ResourcePrefix/ResourceName
```

有关格式的更多信息ARNs，请参阅 Amazon 资源名称 (ARNs) 和 AWS 服务命名空间。

示例

- 使用以下ARN命令允许访问指定的地图资源。

  ```
  "Resource": "arn:aws:geo:us-west-2:account-id:map/map-resource-name"
  ```

- 要指定对属于特定账户的所有 map 资源的访问，使用通配符（*）：

  ```
  "Resource": "arn:aws:geo:us-west-2:account-id:map/*"
  ```

- 无法对特定资源执行某些 Amazon Location 操作，例如用于创建资源的操作。在这些情况下，您必须使用通配符 （*)。

  ```
  "Resource": "*"
  ```

要查看亚马逊定位资源类型及其列表ARNs，请参阅服务授权参考中的 Amazon Location Service 定义的资源。要了解您可以为每种资源指定哪些操作，请参阅 Amazon Location Service 定义的操作。ARN

更新设备位置的权限

要更新多个跟踪器的设备位置，您需要向用户授予访问您的一个或多个跟踪器资源的权限。您还希望允许用户更新一批设备位置。

在本示例中，除了授予访问权限外 *Tracker1* 以及 *Tracker2* 资源，以下策略授予对资源使用geo:BatchUpdateDevicePosition操作的权限 *Tracker1* 以及 *Tracker2* 资源的费用。

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

如果您想限制用户只能更新特定设备的设备位置，则可以为该设备 ID 添加条件键。

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

跟踪器资源的只读策略

要为 AWS 账户中的所有跟踪器资源创建只读政策，您需要授予对所有跟踪器资源的访问权限。您还需要授予用户访问操作的权限，这些操作允许他们获取多台设备的设备位置，从一台设备获取设备位置并获取位置历史记录。

在这个示例中，以下策略授予对以下操作的权限：

- `geo:BatchGetDevicePosition` 检索多个设备的位置。
- `geo:GetDevicePosition` 检索单个设备的位置。
- `geo:GetDevicePositionHistory` 检索设备的位置历史记录。

```
{
  "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/*"
    }
  ]
}
```

创建地理围栏的策略

要创建允许用户创建地理围栏的策略，您需要向允许用户在地理围栏集合上创建一个或多个地理围栏的特定操作授予访问权限。

以下政策授予对以下操作的权限 *Collection*:

- `geo:BatchPutGeofence` 创建多个地理围栏。
- `geo:PutGeofence` 创建单个地理围栏。

```
{
```

```
  "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"
    }
  ]
}
```

地理围栏的只读策略

要为存储在 AWS 账户地理围栏集合中的地理围栏创建只读策略，您需要授予访问从存储地理围栏的地理围栏集合中读取的操作的访问权限。

以下政策授予对以下操作的权限 *Collection*:

- geo:ListGeofences 列出指定地理围栏集合中的地理围栏。
- geo:GetGeofence 从地理围栏集合中检索地理围栏集合。

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

渲染地图资源的权限

要授予足够的权限来渲染地图，你需要授予对地图图块、精灵、字形和样式描述符的访问权限：

- geo:GetMapTile 检索用于在地图上有选择地渲染要素的地图图块。
- geo:GetMapSprites检索 PNG sprite 表和描述其中的偏移量的相应JSON文档。
- geo:GetMapGlyphs 检索用于显示文本的字形。
- geo:GetMapStyleDescriptor 检索地图的样式描述符，其中包含渲染规则。

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

允许搜索操作的权限

要创建允许搜索操作的政策，您首先需要授予对 AWS 账户中地点索引资源的访问权限。您还需要授予访问权限，这些操作允许用户通过地理编码使用文本进行搜索，并通过反向地理编码使用位置进行搜索。

在本示例中，除了授予访问权限外 *PlaceIndex*，以下策略还授予对以下操作的权限：

- geo:SearchPlaceIndexForPosition 允许您搜索给定位置附近的地点或兴趣点。
- geo:SearchPlaceIndexForText 允许您使用自由格式文本搜索地址、名称、城市或地区。

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

路线计算器的只读策略

您可以创建只读策略，允许用户访问路线计算器资源来计算路线。

在本示例中，除了授予访问权限外 *ExampleCalculator*，以下策略授予对以下操作的权限：

- `geo:CalculateRoute` 计算给定出发位置、目的地位置和路径点位置列表的路线。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:us-west-2:accountID:route-calculator/ExampleCalculator"
    }
  ]
}
```

根据条件键控制资源访问权限

在创建IAM策略以授予使用地理围栏或设备位置的访问权限时，您可以使用条件运算符更精确地控制用户可以访问哪些地理围栏或设备。为此，您可以将地理围栏 ID 或设备 ID 包含在策略的 `Condition` 元素中。

以下示例策略演示了如何创建允许用户更新特定设备的设备位置的策略。

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

基于标签控制对资源的访问

当您创建IAM策略以授予使用您的 Amazon Loc ation 资源的权限时，您可以使用基于属性的访问控制来更好地控制用户可以修改、使用或删除哪些资源。为此，您可以将标签信息包含在策略 Condition 元素中，以便根据资源标签控制访问权限。

以下示例策略演示了如何创建允许用户创建地理围栏的策略。这允许以下操作在名为的地理围栏集合上创建一个或多个地理围栏 *Collection*:

- geo:BatchPutGeofence 创建多个地理围栏。
- geo:PutGeofence 创建单个地理围栏。

但是，此策略仅在Condition以下情况下才使用元素授予权限 *Collection* tagOwner，，具有该用户的用户名的值。

- 例如，如果名为的用户richard-roe尝试查看 Amazon 地点 *Collection*，*Collection* 必须加标签Owner=richard-roe或owner=richard-roe。否则，该用户将被拒绝访问。

> **Note**
>
> 条件标签键 Owner 匹配 Owner 和 owner，因为条件键名称不区分大小写。有关更多信息，请参阅《IAM用户指南》中的 “IAMJSON策略元素：条件”。

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

有关如何根据标签定义AWS资源访问权限的教程，请参阅AWS Identity and Access Management 用户指南。

Amazon Location Service 身份和访问权限故障排查

使用以下信息来帮助您诊断和修复在使用 Amazon Location 时可能遇到的常见问题，以及IAM。

主题

- 我无权在 Amazon Location 中执行操作
- 我无权执行 iam：PassRole
- 我想允许我以外的人访问我 AWS 账户 的 Amazon Location 资源

我无权在 Amazon Location 中执行操作

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当mateojacksonIAM用户尝试使用控制台查看虚构*my-example-widget*资源的详细信息但没有虚构权限时，就会出现以下示例错误。geo:*GetWidget*

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 geo:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 geo:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到一个错误，指明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 Amazon Location。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的IAM用户marymajor尝试使用控制台在 Amazon Location 中执行操作时，会出现以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人访问我 AWS 账户 的 Amazon Location 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon Location 是否支持这些特征，请参阅 Amazon Location Service 是如何使用的 IAM。
- 要了解如何提供对您拥有的资源的访问权限，请参阅《IAM用户指南》中的 AWS 账户 向其他IAM用户提供访问权限。 AWS 账户
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅IAM用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过联合身份验证提供访问权限，请参阅《用户指南》中的向经过外部身份验证的用户提供访问权限（联合身份验证）。IAM
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

Amazon Location Service 中的事件响应

AWS非常重视安全性。作为 AWS 云责任共担模式的一部分， AWS 管理满足大多数安全敏感组织要求的数据中心和网络架构。作为 AWS 客户，您有责任维护云端的安全。这意味着你可以从你有权访问的 AWS 工具和功能中控制你选择实施的安全性。

通过建立符合云端运行应用程序目标的安全基准，您可以检测出可以响应的偏差。由于安全事件响应可能是一个复杂的主题，因此我们鼓励您查看以下资源，以便更好地了解事件响应 (IR) 和您的选择对企业目标的影响：AWS安全事件响应指南、AWS安全最佳实践白皮书和AWS云采用框架 (AWSCAF)。

Amazon Location Service 中的日志记录和监控

记录和监控是事件响应的重要组成部分。它允许您建立安全基线来检测可以调查和响应的偏差。通过对 Amazon Location Service 实施日志记录和监控，您可以保持项目和资源的可靠性、可用性、性能。

AWS 提供了多种工具，可以帮助您记录和收集事件响应数据：

AWS CloudTrail

Amazon Location Service 与 AWS CloudTrail集成，后者是一项记录用户、角色或 AWS 服务所执行操作的服务。这包括来自亚马逊定位服务控制台的操作以及对亚马逊定位API操作的编程调用。这些操作记录称为事件。有关更多信息，请参阅使用记录和监控 Amazon Location Service AWS CloudTrail。

亚马逊 CloudWatch

您可以使用亚马逊收集和分析 CloudWatch 与您的亚马逊定位服务账户相关的指标。您可以启用 CloudWatch 警报，以便在指标满足特定条件并达到指定阈值时通知您。当您创建警报时，

CloudWatch会向您定义的 Amazon 简单通知服务发送通知。有关更多信息，请参阅 “使用亚马逊监控亚马逊定位服务” CloudWatch。

AWS Health 仪表板

使用 AWS Health 控制面板，您可以验证 Amazon Location Service 服务的状态。您还可以监控和查看有关可能影响 AWS 环境的任何事件或问题的历史数据。有关更多信息，请参阅 用户指南。AWS Health

Amazon Location Service 的合规性验证

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了部署以安全性和合规性为重点 AWS 的基准环境的步骤。
- 在 Amazon Web Services 上进行HIPAA安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建HIPAA符合条件的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都有HIPAA资格。有关更多信息，请参阅《HIPAA符合条件的服务参考》。

- AWS 合AWS 规资源 — 此工作簿和指南集可能适用于您的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务 并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。

- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务 检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 可以帮助您满足各种合规性要求 PCIDSS，例如满足某些合规性框架规定的入侵检测要求。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

Amazon Location Service 中的韧性

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。 AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

除了 AWS 全球基础设施外，Amazon Location 还提供多项功能来帮助支持您的数据弹性和备份需求。

Amazon Location Service 中的基础设施安全性

作为一项托管服务，Amazon Location Service 受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您使用 AWS 已发布的API呼叫通过网络访问 Amazon 位置。客户端必须支持以下内容：

- 传输层安全 (TLS)。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 具有完美前向保密性的密码套件 ()，例如（Ephemeral Diffie-HellmanPFS）或（Elliptic C DHE urve Ephemeral Diffie-Hellman）。ECDHE大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与IAM委托人关联的私有访问密钥对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

Amazon Location 中的配置和漏洞分析

配置和 IT 控制由您（我们的客户）共同 AWS 负责。有关更多信息，请参阅责任 AWS 共担模型。

防止跨服务混淆代理

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况， AWS 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务主体有权限访问账户中的资源。

Amazon Location Service 不会代表您充当其他 AWS 服务的呼叫服务，因此在这种情况下，您无需添加这些保护。要了解更多关于混淆代理问题，请参阅 AWS Identity and Access Management 用户指南中的混淆代理问题。

Amazon Location Service 的安全最佳实践

Amazon Location Service 提供了在您开发和实施自己的安全策略时需要考虑的大量安全特征。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合环境或不满足环境要求，请将其视为有用的考虑因素而不是惯例。

Amazon Location Service 的检测性安全最佳实践

Amazon Location Service 的以下最佳实践可以帮助检测安全事故：

实施AWS监控工具

监控对于事件响应至关重要，可以维护 Amazon Location Service 资源和解决方案的可靠性和安全性。您可以从多种可用工具和服务中实施监控工具AWS，以监控您的资源和其他AWS服务。

例如，亚马逊 CloudWatch 允许您监控亚马逊定位服务（Amazon Location Service）的指标，并允许您设置警报，以便在指标满足您设定的特定条件并达到您定义的阈值时通知您。创建警报时，您可以设置 CloudWatch 为使用 Amazon 简单通知服务发送提醒通知。有关更多信息，请参阅 the section called “日志记录和监控”。

启用AWS日志工具

日志记录记录用户、角色或AWS服务在 Amazon Location Service 中采取的操作。您可以实现日志工具，例如 AWS CloudTrail 收集有关操作的数据以检测异常API活动。

创建跟踪时，可以配置 CloudTrail 为记录事件。事件是对资源或资源内部执行的资源操作的记录，如向 Amazon Location 发出的请求、发出请求的 IP 地址、请求的发出请求的 IP 地址、时间、时间，以及其他数据。有关更多信息，请参阅 AWS CloudTrail 用户指南中的记录跟踪的数据事件。

Amazon Location Service 的预防性安全最佳实践

Amazon Location Service 的以下最佳实践可以帮助预防安全事故：

使用安全连接

请务必使用加密连接，例如以 `https://` 开头的连接，以确保敏感信息在传输过程中的安全。

实施最低权限访问资源

当您为 Amazon Location 资源创建自定义策略时，只授予执行任务所需的权限。建议最开始只授予最低权限，然后根据需要授予其他权限，则样会更加安全。实施最低权限访问对于减小风险以及可能由错误或恶意攻击造成的影响至关重要。有关更多信息，请参阅 the section called “Identity and Access Management”。

使用全球唯一的设备作为设备 IDs IDs

对设备使用以下约定IDs。

- 设备IDs必须是唯一的。
- 设备IDs不应是秘密的，因为它们可以用作其他系统的外键。
- 设备IDs不应包含个人身份信息 (PII)，例如电话设备IDs或电子邮件地址。
- 设备IDs不应是可预测的。建议使用不透明的标识符UUIDs，例如。

不要包含PII在设备位置属性中

发送设备更新（例如，使用 DevicePositionUpdate ）时，请勿在中包含个人身份信息 (PII)，例如电话号码或电子邮件地址。`PositionProperties`

Amazon Location Service 的最佳实践

本主题提供了可帮助您使用 Amazon Location Service 的最佳实践。虽然这些最佳实践可以帮助您充分利用 Amazon Location Service，但它们并不代表完整的解决方案。您应该只遵循适用于您的环境的建议。

主题

- 资源管理
- 账单和成本管理
- 配额和使用量

安全性

为了帮助管理甚至避免安全风险，请考虑以下最佳实践：

- 使用联合身份验证和IAM角色来管理、控制或限制对您的 Amazon Location 资源的访问。有关更多信息，请参阅《IAM用户指南》中的IAM最佳实践。
- 遵循最低权限原则，仅授予对您的 Amazon Location Service 资源所需的最低访问权限。有关更多信息，请参阅 the section called “使用策略管理访问”。
- 对于网络应用程序中使用的 Amazon Location Service 资源，请使用`aws:referer`IAM条件限制访问权限，限制允许名单中包含的网站以外的网站使用。
- 使用监控和日志工具来跟踪器资源的访问和使用情况。有关更多信息，请参阅 AWS CloudTrail 用户指南中的the section called “日志记录和监控”和记录跟踪的数据事件。
- 使用安全连接，例如以 `https://` 开头的连接，以增加安全性并在服务器与浏览器之间传输数据时保护用户免受攻击。

有关侦测性和预防性安全最佳实践的更多信息，请参阅的 the section called “安全最佳实操” 主题。

资源管理

为了帮助有效管理您在 Amazon Location Service 中的位置资源，请考虑以下最佳实践：

- 使用对您的预期用户群至关重要的区域终端节点来改善他们的体验。有关区域端点的更多信息，请参阅 Amazon Location 区域和终端节点。
- 对于使用数据提供程序的资源，例如地图资源和位置索引资源，请务必遵守特定数据提供程序的使用条款协议。有关更多信息，请参阅数据提供程序。
- 地图、地点索引或路线的每种配置都有一个资源，从而最大限度地减少资源的创建。在一个区域内，每个数据提供程序或地图样式通常只需要一个资源。大多数应用程序使用现有资源，并且不会在运行时创建资源。
- 在单个应用程序中使用不同的资源（例如地图资源和路线计算器）时，请在每个资源中使用相同的数据提供程序来确保数据匹配。例如，使用路线计算器创建的路线几何与使用地图资源绘制的地图上的街道对齐。

账单和成本管理

为了帮助您管理费用和账单，请考虑以下最佳实践：

- 使用诸如 Amazon CloudWatch 之类的监控工具来跟踪您的资源使用情况。您可以设置警报，在使用量即将超过您的指定限制时通知您。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的创建账单警报以监控您的预估AWS费用。

配额和使用量

您 AWS 账户 包含的配额会为您的使用量设置默认限制。您可以设置警报，以便在使用量接近上限时提醒您，也可以在需要时请求提高配额。有关如何处理配额的信息，请参阅以下主题。

- Amazon Location Service 配额
- CloudWatch 用于根据配额监控使用情况
- 在 Amazon CloudWatch 用户指南中@@ 可视化您的服务配额并设置警报。

您可以创建警报，以便在接近超出限制时提前向您发出警告。我们建议您在使用 Amazon 地点的每个配额中为每个 AWS 区域 配额设置警报。例如，您可以监控 `SearchPlaceIndexForText` 操作的使用情况，并在超过当前配额的 80% 时创建警报。

当你收到有关配额的警报警告时，你必须决定该怎么做。您可能正在使用其他资源，因为您的客户群已经扩大。在这种情况下，您可能需要申请增加配额，例如将该地区的API通话配额增加 50%。或者，您的服务中可能存在错误，导致您对 Amazon Location 进行其他不必要的调用。在这种情况下，你需要解决服务中的问题。

文档历史记录

下表介绍 Amazon Location Service 的文档。如需有关更新的通知，您可以订阅 RSS 源。

变更	说明	日期
Amazon Location Service 发布了一个新的软件开发工具包 JavaScript	为了便于在网络前端开发亚马逊定位应用程序，Amazon Location 添加了一个新的开源软件开发工具包，该软件开发工具包支持适用于 JavaScript v3 的 AWS 开发工具包，从而简化了身份验证和使用 GeoJSON。有关更多信息，请参阅 Amazon Location 开发工具包。	2023 年 7 月 6 日
Amazon Location Service 正式发布 API 密钥	Amazon Location 增加了对地点和路由的支持，并宣布 API 密钥功能正式发布。想要了解更多信息，请参阅使用 API 密钥。	2023 年 7 月 6 日
Amazon Location Service 添加了用于位置更新的亚马逊 EventBridge 事件	Amazon Location 增加了对向发送追踪器位置更新事件的支持 EventBridge。如需了解更多信息，包括如何为跟踪器启用事件，请参阅使用 EventBridge 对事件做出反应。	2023 年 7 月 6 日
Amazon Location 将元数据添加到地理围栏	使用 Amazon Location API，您现在可以将元数据属性添加到您的地理围栏中。它们存储在您的地理围栏中，并包含在 Amazon 中与地理围栏相关的事件中。 EventBridge想要了解	2023 年 6 月 15 日

	更多信息，请参阅绘制地理围栏和创建事件规则。	
Amazon Location 为地点添加类别	Amazon Location 在地点搜索结果中添加类别，并按类别筛选结果。想要了解更多信息，请参阅类别和筛选。	2023 年 6 月 15 日
Amazon Location 引入政治观点	Amazon Location 为某些地图样式添加了政治观点。想要了解更多信息，请参阅政治观点。	2023 年 5 月 23 日
Amazon Location 推出新的演示和示例网站	Amazon Location 宣布推出一个新网站，让您可以访问 Amazon Location 演示和示例。想要了解更多信息，请参阅 Amazon Location 演示网站。	2023 年 5 月 3 日
Amazon Location 推出了更长的路线 CalculateRouteMatrix	Amazon Location 现在允许使用 HERE 数据提供程序创建的路由矩阵路由不限长度。想要了解更多信息，请参阅更长的路由规划。	2023 年 4 月 24 日
Amazon Location 文档增加了数据提供程序的功能差异	Amazon Location 文档已更新，其中包含有关地图、地点搜索和路线选择中每个数据提供程序之间差异的信息。想要了解更多信息，请参阅数据提供程序的功能。	2023 年 3 月 30 日
Amazon Location 开放数据地图正式发布	基于日光地图，Amazon Location Service 数据提供程序和样式 OpenStreetMap的正式上市。有关更多信息，请参阅开放数据。	2023 年 3 月 7 日

Amazon Location 在预览版中添加了新的授权方法	Amazon Location Service 在预览模式下添加了 API 密钥作为匿名用户的新授权方法。想要了解更多信息，请参阅允许未经身份验证的访客使用 API 密钥访问您的应用程序。	2023 年 2 月 23 日
使用最新的 IAM 最佳实践更新了 Amazon Location 文档	Amazon Location Service 文档已更新，以符合最新的 AWS Identity and Access Management 最佳实践。想要了解更多信息，请参阅 Amazon Location Service 的安全。	2023 年 1 月 26 日
Amazon Location Service 在东南亚增加了 GrabMaps 数据提供商的地位	Amazon Location GrabMaps 作为数据提供商在东南亚推出。有关更多信息，请参阅GrabMaps。	2023 年 1 月 10 日
Amazon Location Service 打开预览版数据地图	基于日光地图，在公共预览版中添加了新的亚马逊位置数据提供程序和样式。 OpenStreetMap想要了解更多信息，请参阅开放数据（预览版）。	2022 年 12 月 15 日
全新 HERE 卫星图像样式	使用 HERE 作为数据提供程序的地图添加了两种新的地图样式，即 HERE 卫星图像和 HERE 混合地图样式。想要了解更多信息，请单击 HERE 地图样式。	2022 年 10 月 25 日
地址中的单位	Amazon Location Service 现在支持地址中的单位，例如“美国任何城镇主街 123 号，3B 公寓”。	2022 年 9 月 20 日

通过 ID 获取地点	Amazon Location Service 现在支持使用 `GetPlace` 操作查找 `SearchPlaceIndexForSuggestions` 操作建议的确切位置。请参阅使用自动完成功能。	2022 年 9 月 20 日
IAM policy 的其他条件密钥	Amazon Location Service 现在支持其他条件键，允许您在 IAM policy 中为特定地理围栏或设备设置访问权限。请参阅条件键。	2022 年 8 月 23 日
圆形地理围栏	Amazon Location Service 现在支持定义为具有中心点和半径的圆圈的地理围栏，以便在设备距离某个位置一定距离内时获取事件。请参阅添加圆形地理围栏。	2022 年 8 月 11 日
合并 API 参考	Amazon Location Service 现在只有一个 API 参考指南，而不是每个子服务的单独指南。有关 API 的更多信息，请参阅 Amazon Location API。	2022 年 7 月 7 日
服务限额集成	Amazon Location 现已与服务限额集成，可让您通过 AWS Management Console 或使用 AWS CLI 查看和管理您的配额。	2022 年 7 月 6 日
更新了概念文档章节	Amazon Location 概念章节已更新，为 Amazon Location 的用户提供了更多信息。	2022 年 4 月 22 日

全新安卓快速入门教程	新增了使用 Kotlin 的 Android 开发快速入门教程，以帮助开发者快速上手和运行。	2022 年 4 月 15 日
全新 HERE 地图风格	使用 HERE 作为数据提供程序的地图添加了两种新的地图样式。想要了解更多信息，请单击 HERE 地图样式。	2022 年 3 月 15 日
使用添加的代码示例和教程重构文档	本开发者指南经过重构，可以更轻松地查找主题，包括新的快速入门章节和代码示例章节。	2022 年 2 月 25 日
基于精度的跟踪器位置过滤	现在，您可以在创建跟踪器资源时使用基于精度的过滤器。	2021 年 12 月 7 日
地点索引的自动完成	现在，您可以在搜索地点索引时使用自动完成功能。	2021 年 12 月 6 日
全新 Amplify 地图使用教程	提供了一个新教程，介绍如何在 Web 应用程序中使用 AWS Amplify 来显示地图。本教程可在使用 Amazon Location Service 的 Amplify 库中找到。	2021 年 11 月 24 日
放置查询扩展	Amazon Location Service 现在支持在进行地理编码或反向地理编码时为结果设置首选语言，并将时区和其他信息添加到结果中。有关地理编码和反向地理编码的更多信息，请参阅地理编码、反向地理编码和搜索。	2021 年 11 月 16 日

跟踪器位置筛选	Amazon Location Service 为跟踪器添加了一项新的位置筛选功能，可以帮助您控制成本。在存储更新或根据地理围栏评估更新之前，此功能会过滤掉设备上的某些位置更新。有关位置筛选的更多信息，请参阅跟踪器。	2021 年 10 月 5 日
更新操作	以下操作已添加到 Amazon Location Service API 参考中：UpdateMapUpdatePlaceIndexUpdateRouteCalculator、 UpdateGeofenceCollection、、和UpdateTracker。	2021 年 7 月 19 日
教程更新：Amazon Aurora PostgreSQL 用户定义的函数	添加了有关如何在 Amazon Location 中使用 Amazon Aurora PostgreSQL 用户定义的函数来验证、清理和丰富地理空间数据的新教程。	2021 年 7 月 19 日
AWS CloudFormation 资源	Amazon Location 现在支持在 AWS CloudFormation 资源中创建以下资源类型：`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

为资源添加标签	现在，您可以向您的 Amazon Location 资源添加标签，以帮助管理、识别、整理、搜索和筛选您的资源。	2021 年 6 月 1 日
正式发布	Amazon Location Service 开发者文档的正式发布版本：区域和端点以及服务限额已更新。	2021 年 6 月 1 日
Esri 影像	Amazon Location 现在支持使用 Esri 地图样式：Esri 影像。想要了解更多信息，请参阅 Esri 网站上的 Esri 世界影像。	2021 年 6 月 1 日
计算路由	现在，您可以使用 Amazon Location 路线计算器根据 up-to-date 道路网络和所选数据提供商提供的实时交通信息来计算路线并估算行程时间。	2021 年 6 月 1 日
AWS KMS客户托管的静态数据密钥加密	Amazon Location 现在支持使用由您创建、拥有和管理的对称客户托管密钥，在现有 AWS 自有的加密基础上添加第二层加密。	2021 年 6 月 1 日
公开预览版	公开预览版文档的首次发布。	2020 年 12 月 16 日
教程更新：显示地图	使用安卓和 iOS 版显示地图 MapLibre 的教程已更新为使用 MapLibre 原生 SDK。	2020 年 3 月 17 日

AWS 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》中的 AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。