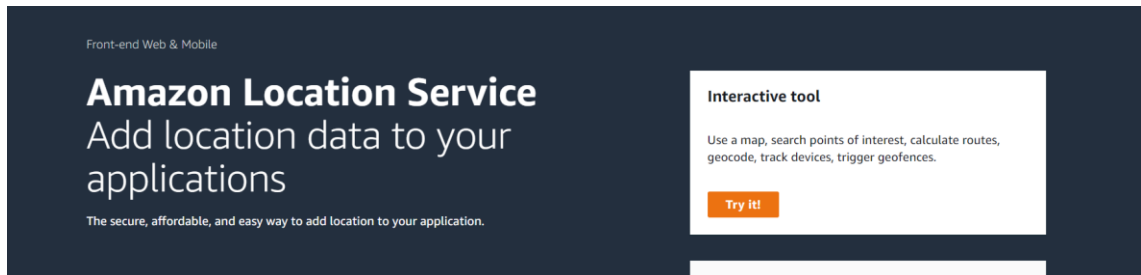
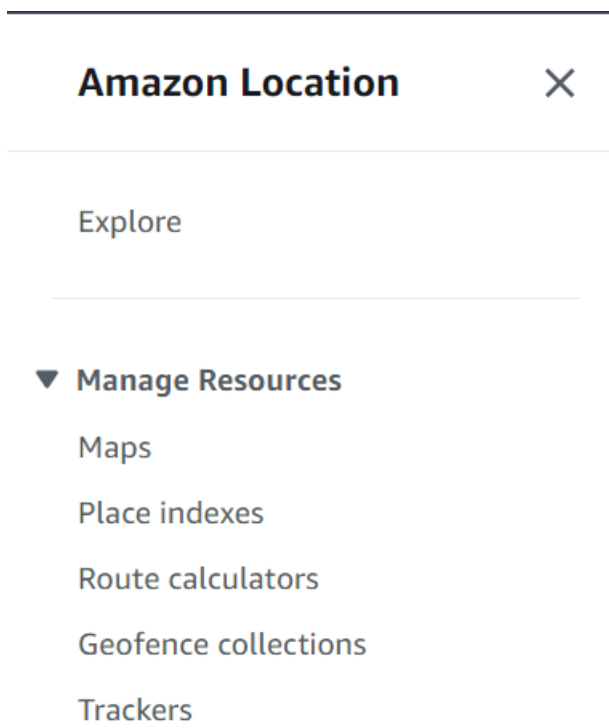


【地図情報の作成】

1. マネージメントコンソールで Location Service にアクセスします



2. 左上、三本線（三）をクリックし、[Maps]をおします



3. [Create map]ボタンをおします
4. 適当な名前をつけます。この名前は後で使うのでメモっておいてください
5. 利用する Map を選びます。種別はなんでも OK です

Maps

Select one of these maps

☒ **Esri Light**
 A detailed basemap for the world, symbolized with a classic Esri map style.

☐ **Esri Street Map**
 This comprehensive street map includes highways, major roads, minor roads, railways, water features, cities, parks, landmarks, building footprints, and administrative boundaries.

6. [Pricing plan use case]で[Yes]をえらびます

Pricing plan use case

The questions in this section guide you to the correct pricing plan for your use case.

1. Will you use this map with simulated/sample location data only?
For example, when using it locally or in a test environment.

☒ Yes
 ☐ No

7. 以下にチェックをつけて[Create map]をおします

☒ To use Amazon Location Maps, I have read and agree to the [Terms and Conditions](#) for Amazon Location Service. AWS may transmit my API queries to my chosen third party data provider for processing, which may be outside of the AWS Region that I am currently using.

Cancel **Create map**

8. ARN を後で使うのでコピーしておきます

Amazon Location > Maps > 20210714

20210714 [情報](#)

Delete map

Information		
Name	Description	ARN
20210714		arn:aws:geo:ap-northeast-1:294963776963:map/20210714
Data provider	Map style	
Esri	<u>Esri Street Map</u>	

【認証基盤の作成】

以上で地図ができました。地図が埋め込まれた HTML にアクセスするユーザーに地図へアクセスするための認証基盤を作ります。(一時的に IAM ロールを引き渡す仕組み)

みです)

9. Cognito のマネージメントコンソールにアクセスします
10. [ID プールの管理]をおします
11. [新しい ID プールの作成]をおします
12. 適当な名前を以下につけます

新しい ID プールの作成

ID プールはエンドユーザー ID を保存するために使用されます。新しい ID プールを宣言するには、一意の名前を入力します。

ID プール名*

例: My App Name

13. [認証されていない ID に対してアクセスを有効にする]にチェックを付けます。これによるログイン不要で IAM ロール権限が HTML にアクセスした人に一時的に引き渡されるようになります。

▼ 認証されていない ID ⓘ

Amazon Cognito は、ID プロバイダーを使って認証を行わないユーザー用に、一意の識別子や AWS 認証情報を提供して、認証されていない ID をサポートできます。アプリケーションで、ログインすることなくユーザーにアプリケーションの使用を許可している場合、認証されていない ID に対してアクセスを有効にできます。[認証されていない ID の詳細を参照してください。](#)



認証されていない ID に対してアクセスを有効にする

このオプションを有効にすると、インターネットアクセスが可能なユーザー全員に AWS 認証情報を付与することができます。認証されていないアイデンティティとは通常、お客様のアプリケーションにログインしていないユーザーのことです。通常、認証されていないアイデンティティに割り当てられるアクセス許可のほうは、認証されたアイデンティティに割り当てられるアクセス許可よりも制限が厳しくなります。

14. [プールの作成]をおします
15. [許可]をおします
16. 以下のプールの ID をメモっておきます

Amazon Cognito での作業開始

プラットフォーム **Android ▼**

▼ AWS SDK のダウンロード

[AWS SDK for Android をダウンロード](#) [開発者ガイド](#)

▼ AWS 認証情報の取得

```
// Amazon Cognito 認証情報プロバイダーを初期化します
CognitoCachingCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(
    getApplicationContext(),
    "ap-northeast-1:9df1b361-7e72-4239-9c72-645f0f8313a0", // ID プールの ID
    Regions.AP_NORTHEAST_1 // リージョン
);
```

17. IAM ロールのマネージメントコンソールにアクセスします
18. 先程作成した cognitoID プールの名前で検索すると 2 つ IAM ロールができています。認証済ユーザー用と非認証一般ユーザー様です

ロール (148) 情報

IAM ロールは、特定のアクセス許可を持つ作成可能な ID です。ロールは、1 人のユーザーに一意に関連付けられるのではなく、必要とするユーザーが誰でも継承できることを目的としています。IAM ロールは、短期間有効な認証情報を発行します。これは、信頼するエンティティにアクセス許可を付与するための、よりセキュアな方法です。

🔄 削除 **ロールを作成**

🔍 0714 ✕ 2 一致

<input type="checkbox"/>	ロール名	信頼エンティティ	最後のアクティビティ
<input type="checkbox"/>	Cognito_20210714Auth_Role	ID プロバイダー: cognito-identity.amazonaws.com	なし
<input type="checkbox"/>	Cognito_20210714Unauth_Role	ID プロバイダー: cognito-identity.amazonaws.com	なし

19. このハンズオンでは誰でもアクセスできる HTML 環境を作りますので、[Unauth]の方を選びクリックします
20. [ポリシーをアタッチします]をおします
21. [ポリシーの作成]おします。(ブラウザ別タブが起動します)
22. JSON タブを選び、コピー用コマンド集の 1 番をコピーします
23. “Resource”の部分を先程作成した map の ARN に置き換えます

ポリシーの作成

1 2 3

ポリシーにより、ユーザー、グループ、またはロールに割り当てることができる AWS アクセス権限が定義されます。ビジュアルエディタで JSON を使用してポリシーを作成または編集できます。詳細はこちら

ビジュアルエディタ JSON 管理ポリシーのインポート

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "MapsReadOnly",
6       "Effect": "Allow",
7       "Action": [
8         "geo:GetMapStyleDescriptor",
9         "geo:GetMapGlyphs",
10        "geo:GetMapSprites",
11        "geo:GetMapTile"
12      ],
13      "Resource": "arn:aws:geo:ap-northeast-1:294963776963:map/20210714",
14      "Condition": {
15        "ForAnyValue:StringLike": {
16          "aws:referer": [
17            "http://localhost*",
18            "http://localhost:3000*"
19          ]
20        }
21      }
22    }
23  ]
24 }
```

🔒 セキュリティ: 0 🚫 エラー: 0 ⚠️ 警告: 0 💡 提案: 0

24. [次のステップ:タグ]をおし、次の画面で[次のステップ:確認]をおします
25. 適当名前をつけ、[ポリシーの作成]をおします
26. 先程の IAM ロール設定のタブに戻り、先程作成したポリシーを検索から探して、チェックを付けます。出てこない場合、右上の丸い矢印マークをおすと出てきます

Cognito_20210714Unauth_Role にアクセス権限を追加する
アクセス権限をアタッチする

ポリシーの作成

ポリシーのフィルタ 🔍 0714 1 件の結果を表示中

<input type="checkbox"/>	ポリシー名	タイプ	次として使用
<input checked="" type="checkbox"/>	20210714policy	ユーザーによる管理	なし

27. [ポリシーのアタッチ]をおします。

以上で、非認証一般ユーザーでも地図にアクセスできるようになる権限が作成されま

した。

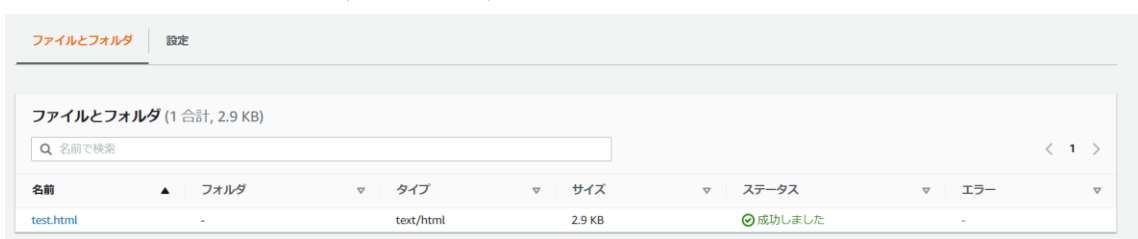
【地図埋め込み HTML の作成とホスティング】

先程作成した地図を HTML に埋め込み、一般ユーザーがアクセスできるように設定し S3 にホスティング。

28. S3 のマネージメントコンソールに移動します
29. [バケットを作成]をおします
30. 適当な名前を入力し、[バケットを作成]をおします
31. GitHub から test.html をダウンロードしてエディタで開きます
32. identityPoolId に先程作成した Cognito プール ID の ID をコピペします。同様に const mapName に先程作成した地図の名前をいれます。(ARN ではなく地図の名前で)
33. ファイルを保存し、先程作成した S3 バケットのアップロードします。マネージメントコンソールでバケット名をクリックすると以下の画面が表示されます。[アップロード]ボタンをおしてください



34. [ファイルの追加]をおし、ファイルが選択出来たら[アップロード]ボタンをおします
35. アップされたファイルをクリックします

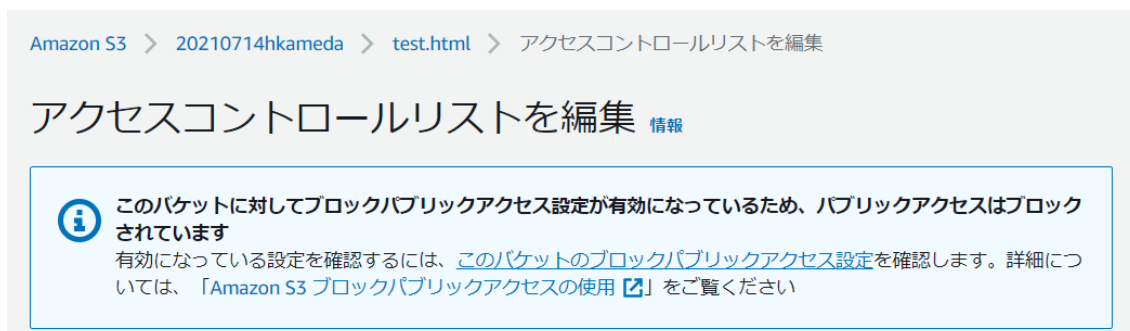


36. [アクセス許可]のタブをおします



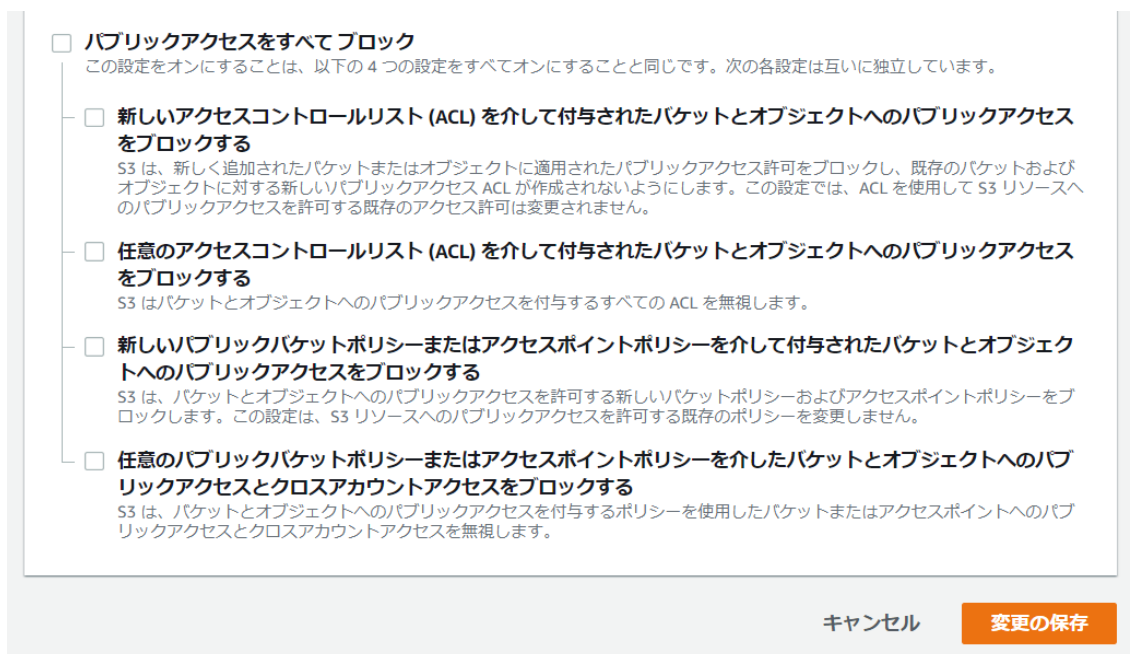
37. [編集]をおします

38. 以下の画面の[このバケットのブロックパブリックアクセス設定]をクリックします



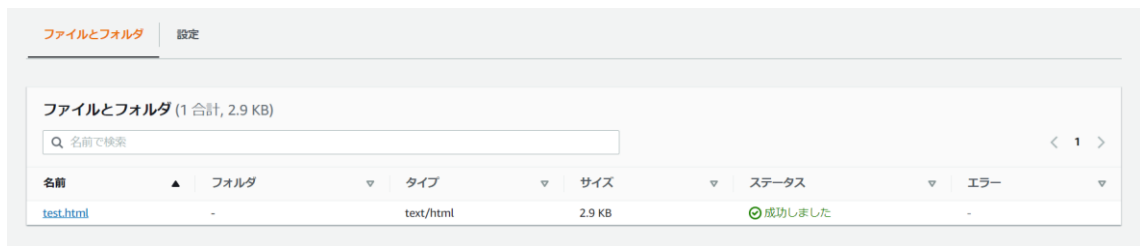
39. [ブロックパブリックアクセス (バケット設定)]の[編集]ボタンをおします

40. 以下の様にチェックを外して[変更の保存]をおします



41. 次の画面で[確認]と入力し[確認]ボタンをおします

42. 再度バケットの中にある test.html をクリックします



43. [アクセス許可]をタブを選びます

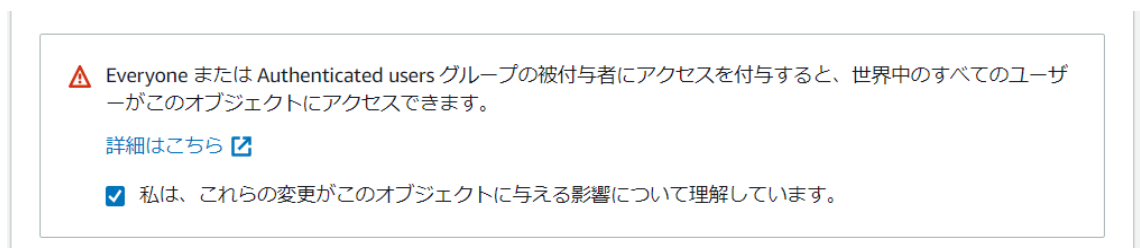


44. [編集]ボタンをおします

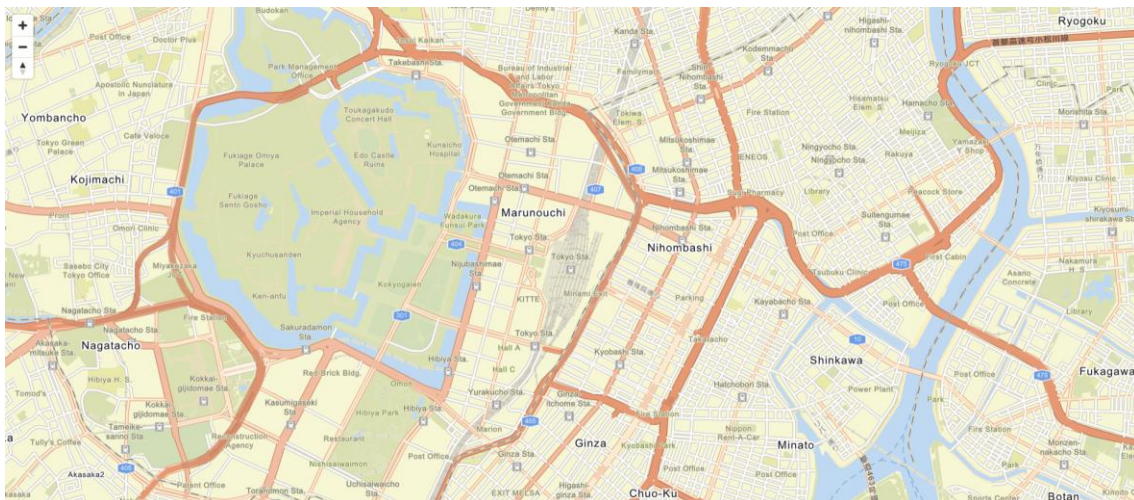
45. 全員 (パブリックアクセス)の設定を以下のようにします。



46. 注意事項にチェックをつけ、[変更の保存]をおします



47. 以下のように地図が表示されます



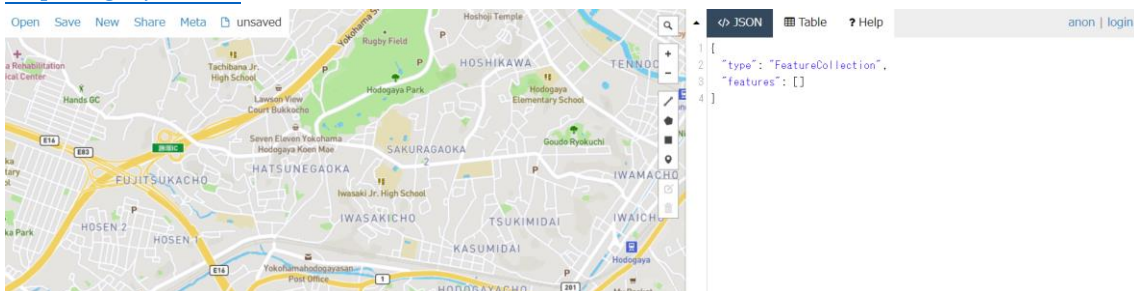
真っ白な画面が出る場合 test.html の修正をミスしていると思われるので、再度確認してみてください。

【Geo Fence 機能の実装】

Location Service には GPS をベースとして任意の区画にデバイスが[Exit]した、もしくは[Enter]したというイベントを取得することができます。

AWS IoT Core 経由でデバイスから GPS 情報が MQTT で挿入され、それを Lambda で Location Service に送信し、イベントを CloudWatch Logs に出力する環境を作ります。

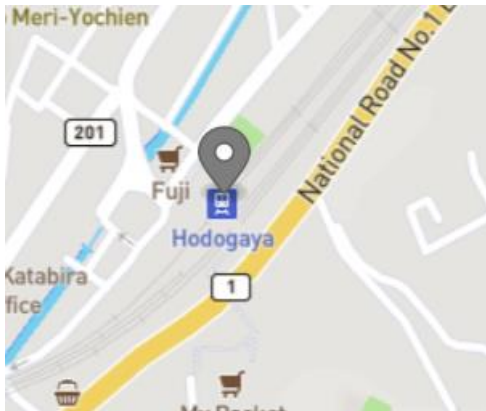
48. <https://geojson.io/> にアクセスします



49. 自分の最寄駅をさがします

50. 駅が見つかったら以下のボタンをおしてピンを落とします





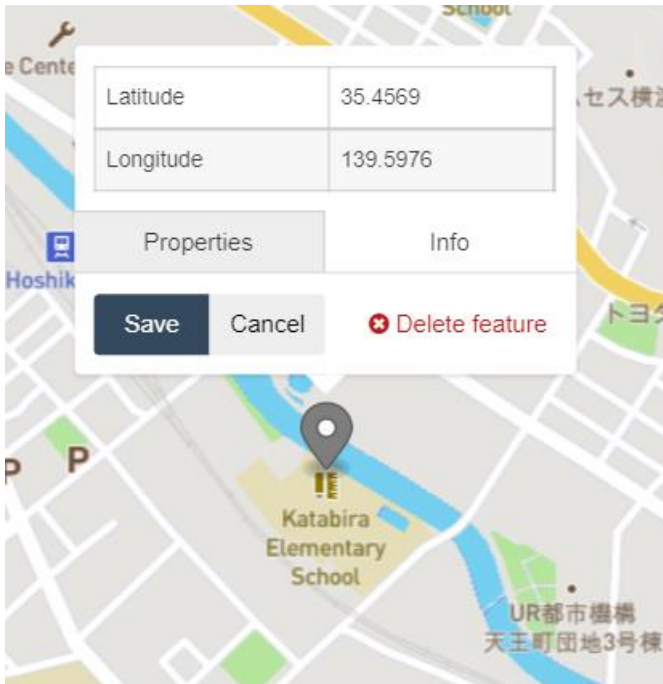
51. 画面右に GPS 情報が出てきますので、coordinates で表示される座標をメモります

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "type": "Point",
        "coordinates": [
          139.59964513778687,
          35.44680904209397
        ]
      }
    }
  ]
}
```

52. 地図上のピンをおして[Delete feature]をクリックし、ピンをけします
53. 今度は5角形のマークをおして、駅周辺に領域を作成します

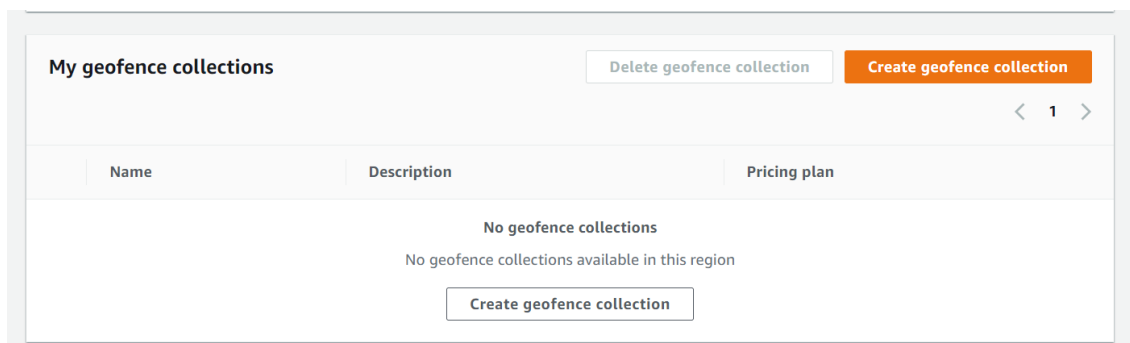


54. 画面右側に表示される JSON をコピーし、[geo.json]として保存してください
55. 次に、領域の外にある場所のどこかでピンを再度落とし、GPS 情報をメモして下さい



56. Location Service のマネージメントコンソールに戻り左のペインから[Geofence

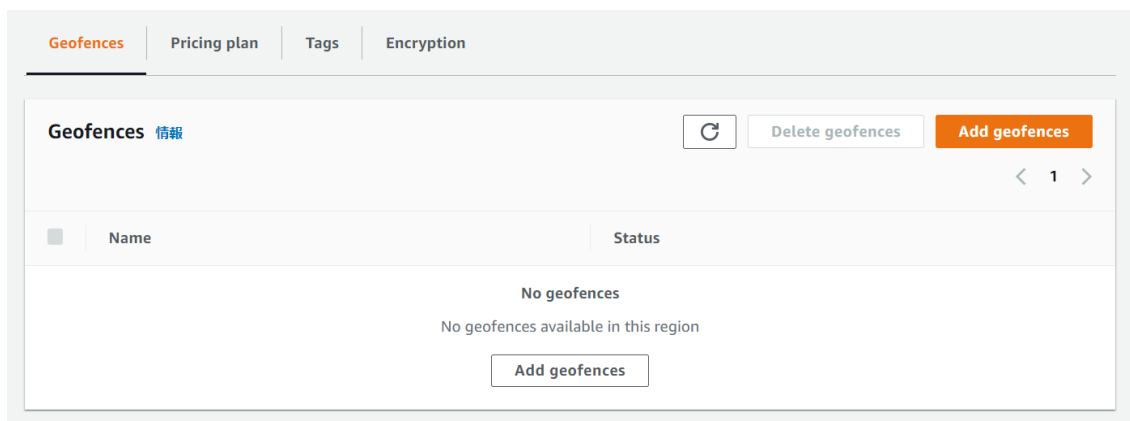
collections]をクリックします



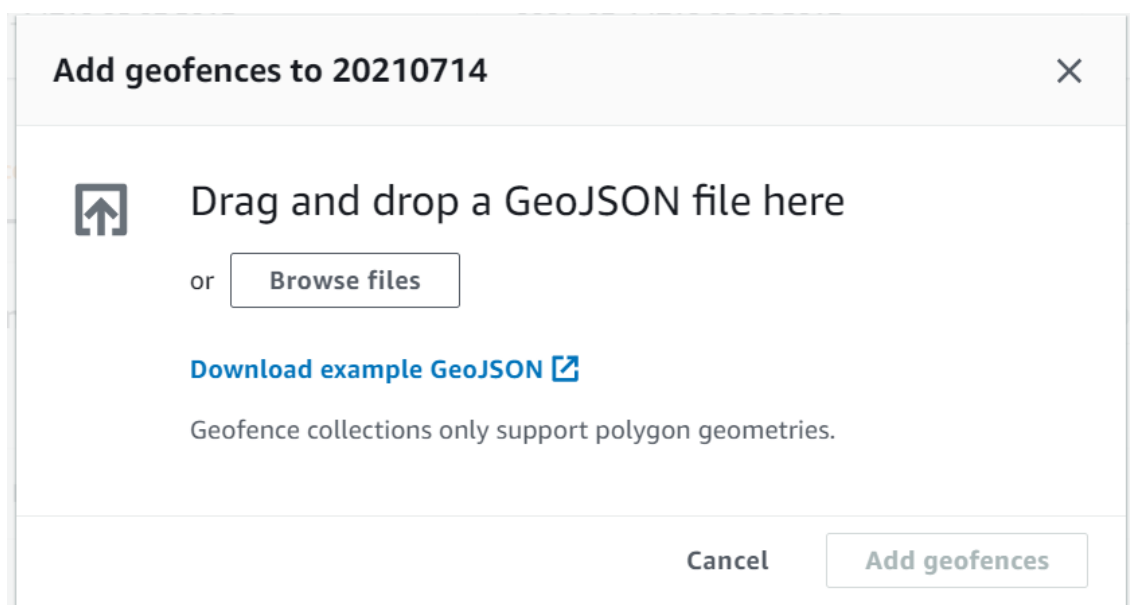
57. [Create geofence collection]をおします

58. 適当な名前をつけ、[Pricing plan use case]で Yes を選び、[Create geofence collection]をおします

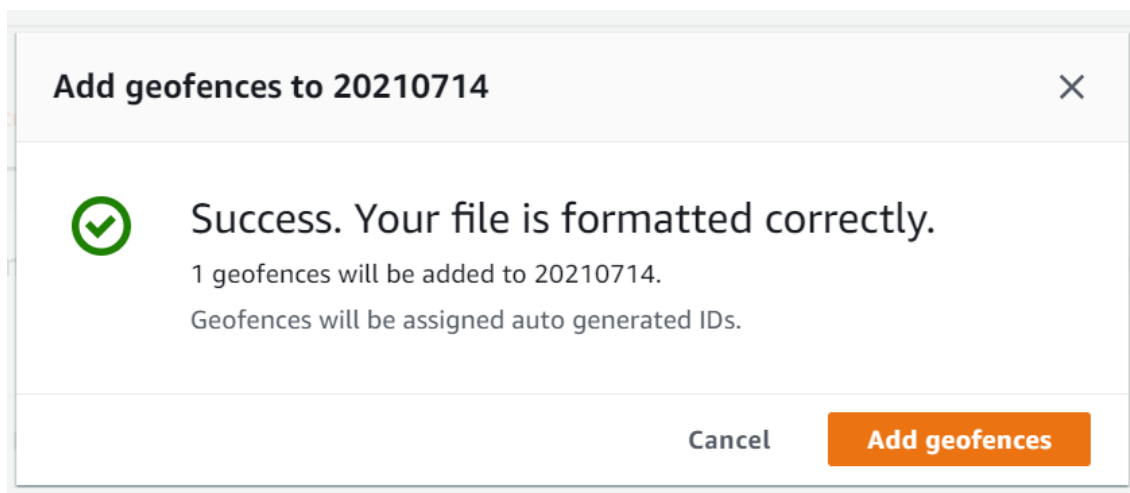
59. [Add geofence]をおします



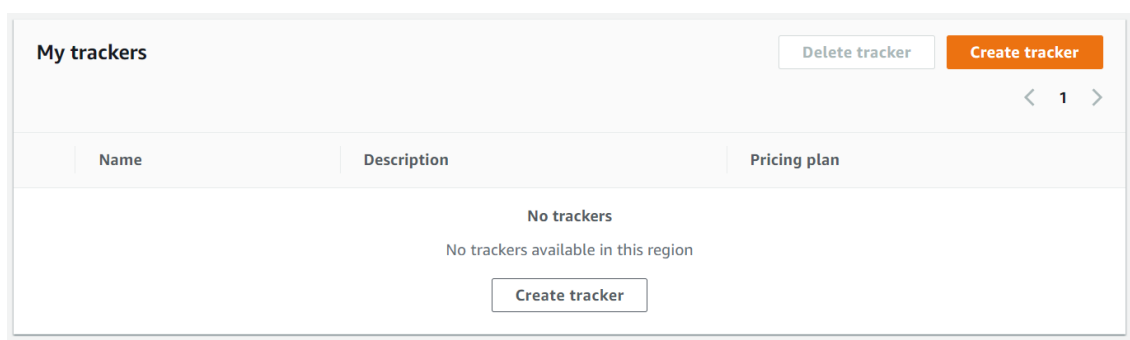
60. 先程作成した geo.json をアップロードします



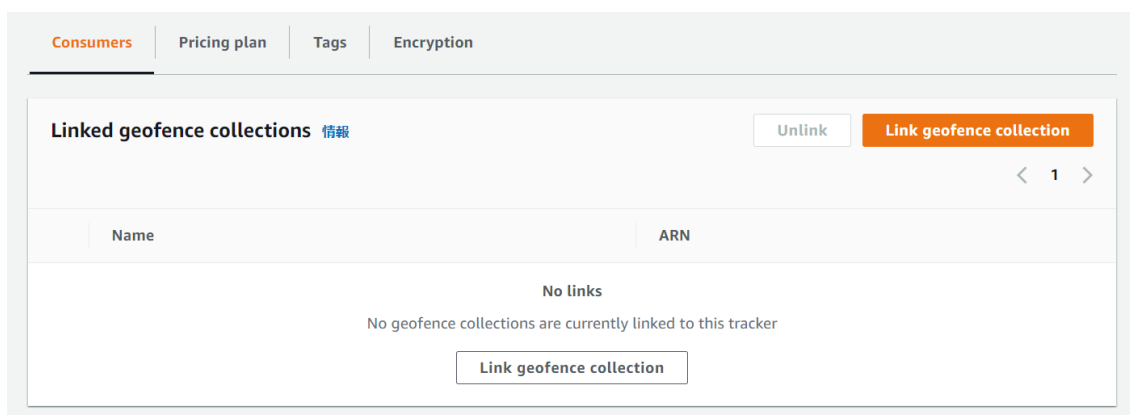
61. ファイルが正しければ以下が表示されますので[Add geofences]をおします



62. これで最寄駅を周辺とした領域を Location Service が認識しました。次に GPS デバイスと連携させるために、左ペインから[Trackers]をクリックします



63. [Create trackers]をおします
64. 適当な名前をつけ、[Pricing plan use case]で Yes を選び、[Create tracker]をおします
65. [Link geofence collection]をおします



66. 先程作成した geofence を選んで[Link]をおします

Link geofence collection to tracker20210714 [X]

Geofence collection
Select an existing collection

20210714 ▼ [Refresh]

[Cancel] [Link]

67. これで特定の Tracker が特定の Geo fence と連携するようになりました。Tracker には複数の Geo fence を連携させることも可能です。これからの手順では Tracker に対して AWS IoT Core から MQTT ベースで GPS 情報が送られるように設定をおこないます。
68. まず Lambda 関数にアタッチする IAM ロールを作成するために、IAM のマネジメントコンソールにアクセスします
69. 左のペインから[ロール]をクリックします
70. [ロールを作成]をおします
71. 以下で Lambda を選び、[次のステップ:アクセス権限]をおします

ユースケースの選択

一般的なユースケース

EC2

Allows EC2 instances to call AWS services on your behalf.

Lambda

Allows Lambda functions to call AWS services on your behalf.

72. [Administrator Access]を選び、[次のステップ:タグ]をおします

ロールの作成

1 2 3 4

▼ Attach アクセス権限ポリシー

新しいロールにアタッチするポリシーを1つ以上選択します。

ポリシーの作成

[Refresh]

ポリシーのフィルタ

Q adminis

16 件の結果を表示中

	ポリシー名	次として使用
<input checked="" type="checkbox"/>	AdministratorAccess	Permissions policy (9)

注意：本来は Location Service のみのアクセス権でいいのですが、2021 年 7 月時点でビルトインの Location Service 用ポリシーが要されておらず、そちらの作成は手順が煩雑となるため、このハンズオンでは割愛します。商用環境における Admin 権限はお勧めしませんので、詳しくは以下を参考にしてください。

https://docs.aws.amazon.com/ja_jp/location/latest/developerguide/security-iam.html

73. 次の画面ではそのまま[次のステップ:確認]をおします
74. 適当な名前をロールにつけ[ロールの作成]をおします
75. 次に Lambda 関数を作成します。この関数は AWS IoT Core から上がってきた GPS 情報を Location Service に渡す役割をにないます。Lambda のマネージメントコンソールにアクセスします
76. [関数の作成]をおします
77. 適当な名前を関数につけ、ランタイムには Python3.8 を選びます

基本的な情報

関数名
関数の目的を名前として入力します。

map20210714

半角英数字、ハイフン、アンダースコアのみを使用でき、スペースは使用できません。

ランタイム 情報
関数の記述に使用する言語を選択します。コンソールコードエディタは Node.js、Python、および Ruby のみをサポートすることに注意してください。

Python 3.8

78. [デフォルトの実行ロールの変更]をクリックし、[既存のロールを使用する]を選びます

▼ デフォルトの実行ロールの変更

実行ロール
関数のアクセス権限を定義するロールを選択します。カスタムロールを作成するには、IAM コンソールに移動します。

☐ 基本的な Lambda アクセス権限で新しいロールを作成

☐ 既存のロールを使用する

☒ AWS ポリシーテンプレートから新しいロールを作成

79. 先程作成した IAM ロールを選びます、[関数の作成]をおします

▼ デフォルトの実行ロールの変更

実行ロール
関数のアクセス権限を定義するロールを選択します。カスタムロールを作成するには、IAM コンソールに移動します。

☐ 基本的な Lambda アクセス権限で新しいロールを作成

☒ 既存のロールを使用する

☐ AWS ポリシーテンプレートから新しいロールを作成

既存のロール
この Lambda 関数で使用するために作成した既存のロールを選択します。このロールには、Amazon CloudWatch Logs にログをアップロードするアクセス許可が必要です。

lambdamap20210714

IAM コンソールで lambdamap20210714 ロールを表示します。

80. GitHub から lambda.txt をダウンロードし、その中身をコードに貼り付けます
81. TRACKER_NAME の部分を先程作成した Tracker の名前に置き換えます

```

1 from datetime import datetime
2 import json
3 import os
4 import boto3
5
6 # Update this to match the name of your Tracker resource
7 TRACKER_NAME = "tracker20210714"
8
9 """
10 This Lambda function receives a payload from AWS IoT Core and publishes device updates to Amazon Location Service via the BatchUpdateDevice
11 API.
12 Parameter 'event' is the payload delivered from AWS IoT Core.
13
14 In this sample, we assume that the payload has a single top-level key 'payload' and a nested key
15 'location' with keys 'lat' and 'long'. We also assume that the name of the device is nested in
16 the payload as 'deviceid'. Finally, the timestamp of the payload is present as 'timestamp'. For
17 example:
18
19 >>> event
20 { 'payload': { 'deviceid': 'thing123', 'timestamp': 1604940328,
21               'location': { 'lat': 49.2819, 'long': -123.1187 } } }
22
23 If your data does not match this schema, you can either use the AWS IoT Core rules engine to
24 format the data before delivering it to this Lambda function, or you can modify the code below to
25 match it.
26 """

```

82. [Deploy]をおします
83. 次に IoT Core のマネージメントコンソールにアクセスします
84. 左ペインから[ACT]→[ルール]をクリックします



85. [作成]をおします
86. 適当な名前をつけ、[ルールクエリステートメント]を以下で置換します

SELECT * FROM 'iot/location'

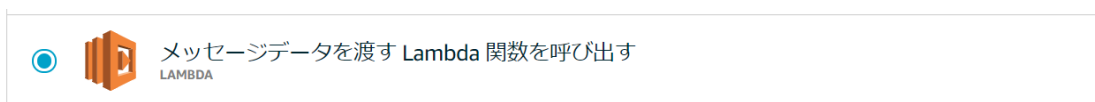
87. [1 つ以上のアクションを設定する]の[アクションの追加]をおします

1 つ以上のアクションを設定する

インバウンドメッセージが上記のルールに一致すると、1 つ以上のアクションが選択されます。メッセージ受信時に発生する追加アクティビティ（データベースへの格納、クラウド関数の呼び出し、通知の送信など）を定義するアクション。（*必須）

アクションの追加

88. [Lambda]を選び、[アクションの設定]をおします



89. 先程作成した関数を選び[アクションの追加]をおします
 90. [ルールの作成]をおします
- ここまでの手順で、IoT Core から MQTT ベースで上がってきた GPS 情報が Lambda 経由で Location Service の Tracker に渡され、Tracker と Link されている Geo fence 情報と

突合されその結果が CloudWatch Logs に出力されるようになりました。
先程作成した Lambda の関数画面に行くと正しく IoT Core がイベントソースとして設定されていることがわかります



91. テストを行う前に CloudWatch Logs を開きます。CloudWatch マネージメントコンソールの左ペインから[ロググループ]をクリックします

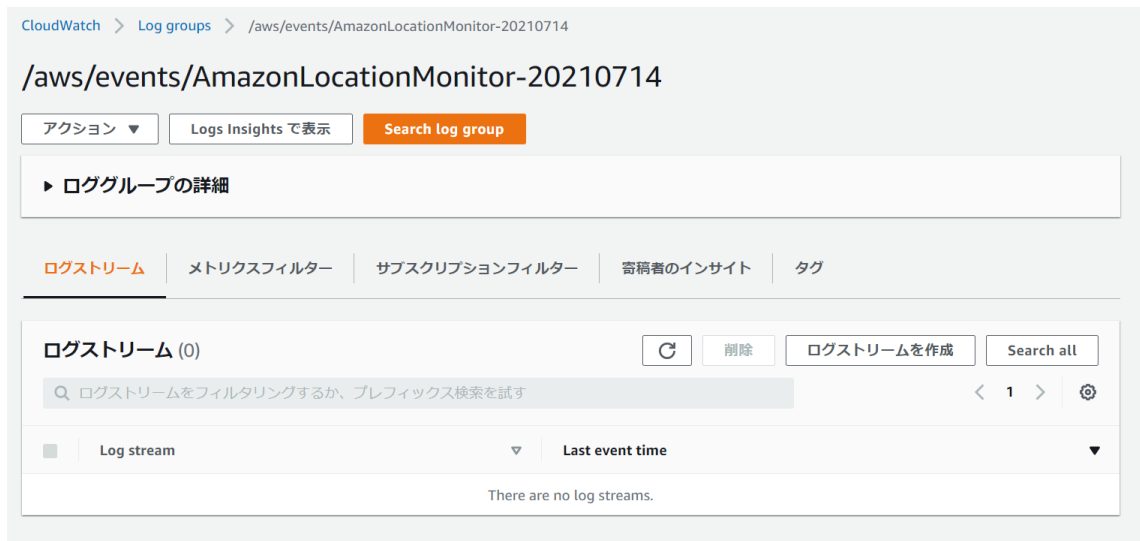
▼ ログ

ロググループ

Log Insights

92. [location]で検索すると今日作成したロググループが出てきますので、クリックしておきます





今の時点ではログは何もふくんでいません。この画面は開きっぱなしにしておいてください。

93. いよいよテストです。IoT Core 左ペインから[MQTT テストクライアント]をクリックし、[トピックに公開する]タブを選んでください



94. トピック名は[**iot/location**]と入力します
95. コピペ用コマンド 2 番の内容を別のエディタにコピーし、先程取得した最寄り駅の GPS 情報に置き換えます。置き換えたものを[メッセージペイロード]に貼り付け、[発行]をおします
96. Location Service はデバイスが Geo fence の中にいることを認識しました。次に timestamp の値を 10 増やし、geo fence の外にある GPS 情報に置き換え、再度メッセージを発行してください
97. CloudWatch Logs を確認すると 2 つのログがでています
Enter と EXIT です

```
▼ 2021-07-14T13:10:13.000Z {"version":"0","id":"3e6e03db-1b08-abe3-a573-9254f0cf610e","detail-type":"Location Geofence Event","sour...
{
  "version": "0",
  "id": "3e6e03db-1b08-abe3-a573-9254f0cf610e",
  "detail-type": "Location Geofence Event",
  "source": "aws.geo",
  "account": "294963776963",
  "time": "2021-07-14T13:10:13Z",
  "region": "ap-northeast-1",
  "resources": [
    "arn:aws:geo:ap-northeast-1:294963776963:geofence-collection/20210714",
    "arn:aws:geo:ap-northeast-1:294963776963:tracker/tracker20210714"
  ],
  "detail": {
    "EventType": "ENTER",
    "GeofenceId": "monitoring-685566c9-02e3-4cb7-9081-22315b39968a",
    "DeviceId": "GPS-Device-001",
    "SampleTime": "2020-12-17T01:00:00Z",
    "Position": [
      139.59964513778687,
      35.44680904209397
    ]
  }
}
```

コピー

```
▼ 2021-07-14T13:11:53.000Z {"version":"0","id":"b9326473-ee9a-86d7-863b-81cbca71d532","detail-type":"Location Geofence Event","sour...
{
  "version": "0",
  "id": "b9326473-ee9a-86d7-863b-81cbca71d532",
  "detail-type": "Location Geofence Event",
  "source": "aws.geo",
  "account": "294963776963",
  "time": "2021-07-14T13:11:53Z",
  "region": "ap-northeast-1",
  "resources": [
    "arn:aws:geo:ap-northeast-1:294963776963:geofence-collection/20210714",
    "arn:aws:geo:ap-northeast-1:294963776963:tracker/tracker20210714"
  ],
  "detail": {
    "EventType": "EXIT",
    "GeofenceId": "monitoring-685566c9-02e3-4cb7-9081-22315b39968a",
    "DeviceId": "GPS-Device-001",
    "SampleTime": "2020-12-17T01:00:10Z",
    "Position": [
      139.5976,
      35.4569
    ]
  }
}
```

コピー

おつかれさまでした！ハンズオンは以上です

以下を削除してください

Location Service

Trackers

Geofence collection

Maps

Lambda 関数

S3 バケット

IAM ポリシーとロール

AWS IoT Core

ルール