

AmazonConnectによる自動電話通知（7.複数連絡先への電話通知〈構築④〉）

AWS



×

ZABBIX

×



AmazonConnectによる自動電話通知 （7.複数連絡先への電話通知〈構築④〉）

2021.11.12 2021.11.01

[【前回】 AmazonConnectによる自動電話通知（7.複数連絡先への電話通知〈構築③〉）](#)[【次回】 AmazonConnectによる自動電話通知（7.複数連絡先への電話通知〈構築⑤〉）](#)[【簡易版】 AmazonConnectによる自動電話通知（まとめ）](#)

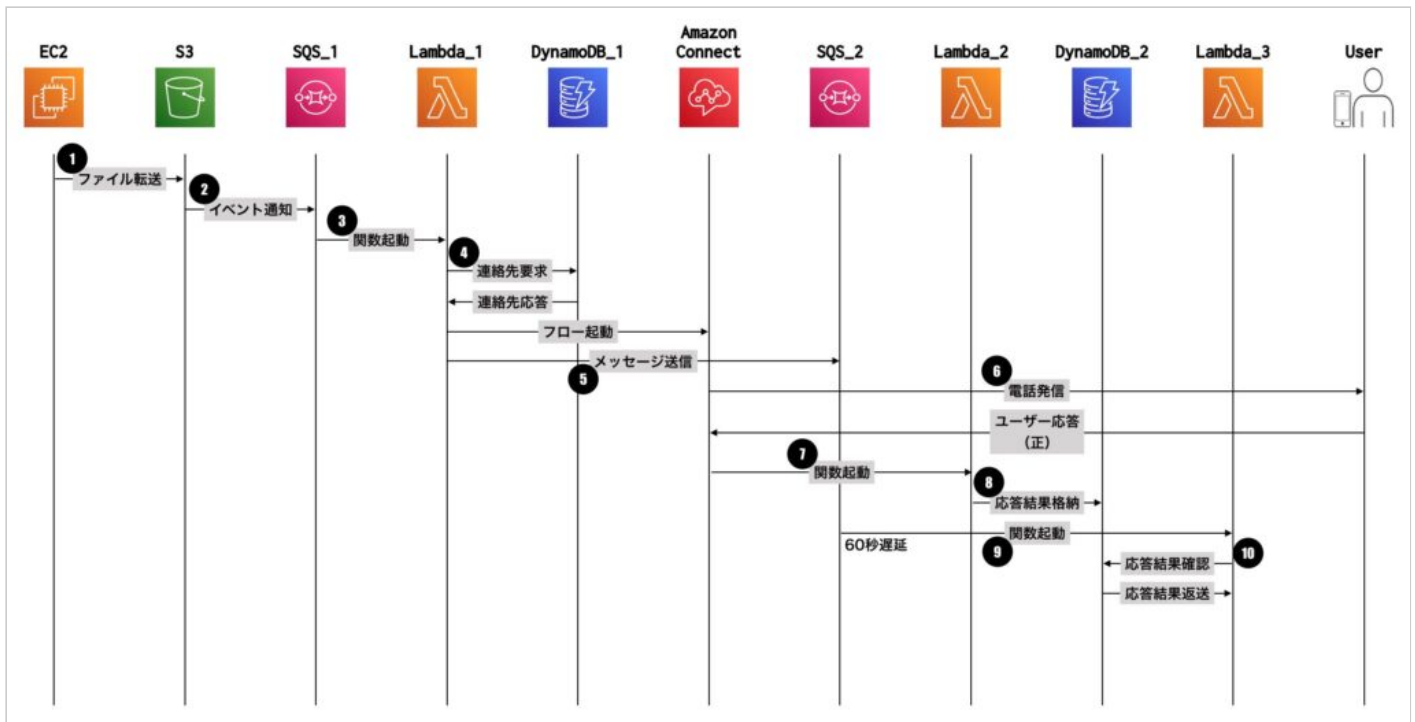
監視サーバーで障害を検知した際に、自動で電話通知できるようにしていきます。ネットワークエンジニアも利用することの多い監視サーバー(Zabbix)で障害検知し、AWS上のAmazonConnectを利用し自動電話を発信します。

今回は下記の条件を満たせるようにAWSの各サービスを利用して自動電話通知の仕組みを導入します。

- 複数の通知先を登録した連絡先リストを持たせる。
- 連絡先リストに優先度(通知順)を設定する。
- 優先度が高い人に最初に電話する。
- 応答が無かった場合、次の優先度の人に順番に電話する。
- 連絡先リストの最後まで電話しても応答が無かった場合、最初に戻って継続する。

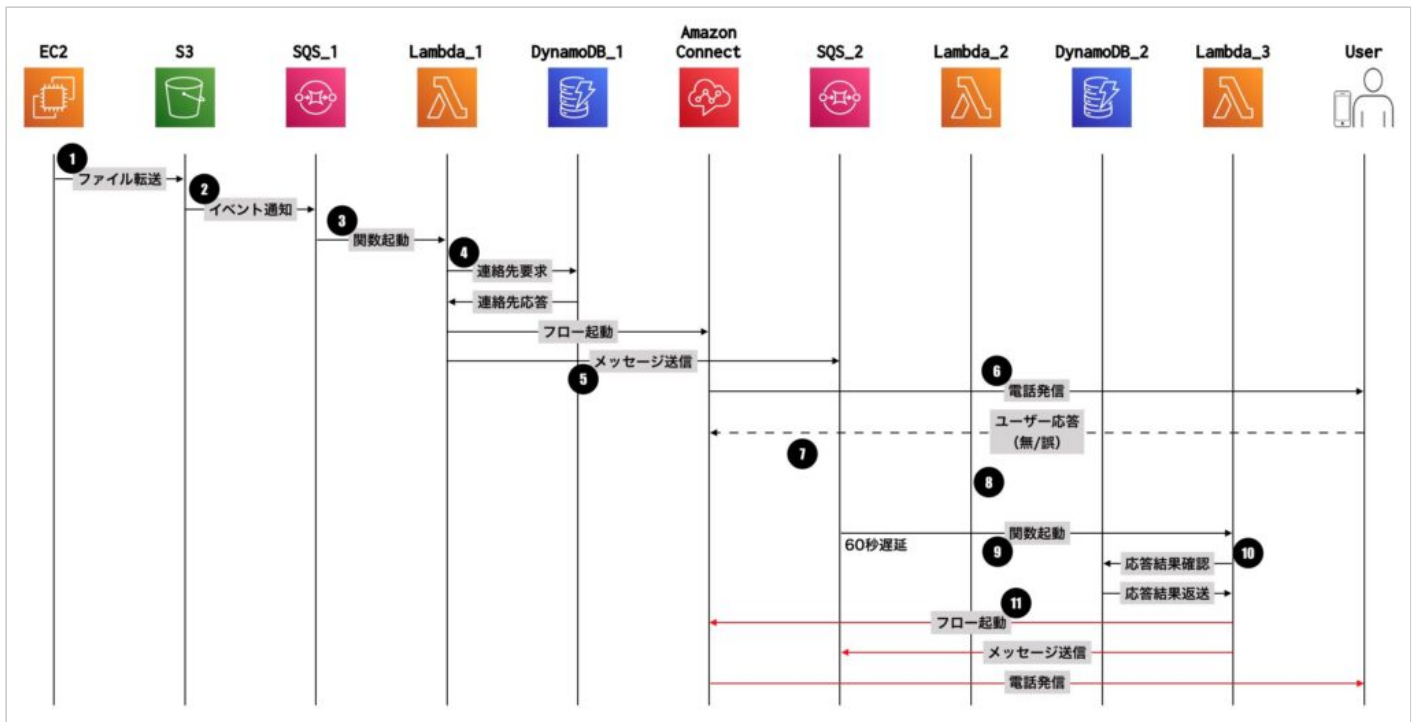
自動電話通知フロー

電話に応答した場合のフロー



1. EC2上の監視サーバーで障害を検知し、S3へトリガーファイルを格納
2. S3のイベント通知機能で、SQS_1にメッセージを送信
3. SQS_1をトリガーとして、Lambda_1を起動
4. Lambda_1がDynamoDB_1から連絡先を取得し、AmazonConnectを起動
5. Lambda_1がAmazonConnectを起動すると同時に、SQS_2へメッセージを送信
6. AmazonConnectがユーザーへ自動電話通知を実施
7. ユーザーが正常応答し、AmazonConnectがLambda_2を起動
8. Lambda_2が応答結果をDynamoDB_2に保存(応答OK)
9. 60秒後にSQS_2をトリガーとしてLambda_3を起動
10. Lambda_3がDynamoDB_2の応答結果を確認(正常応答しているため、何もせずに処理完了)

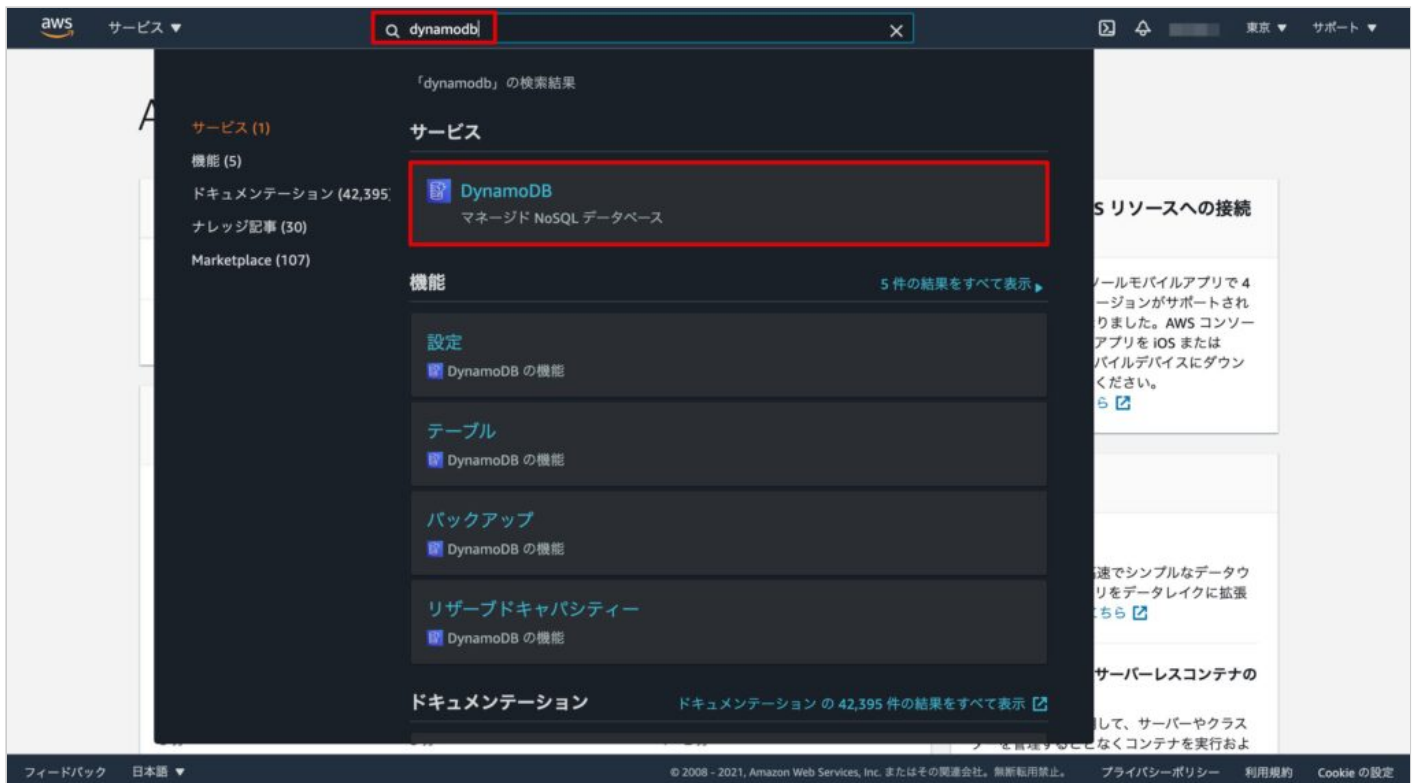
電話に応答しなかった場合のフロー



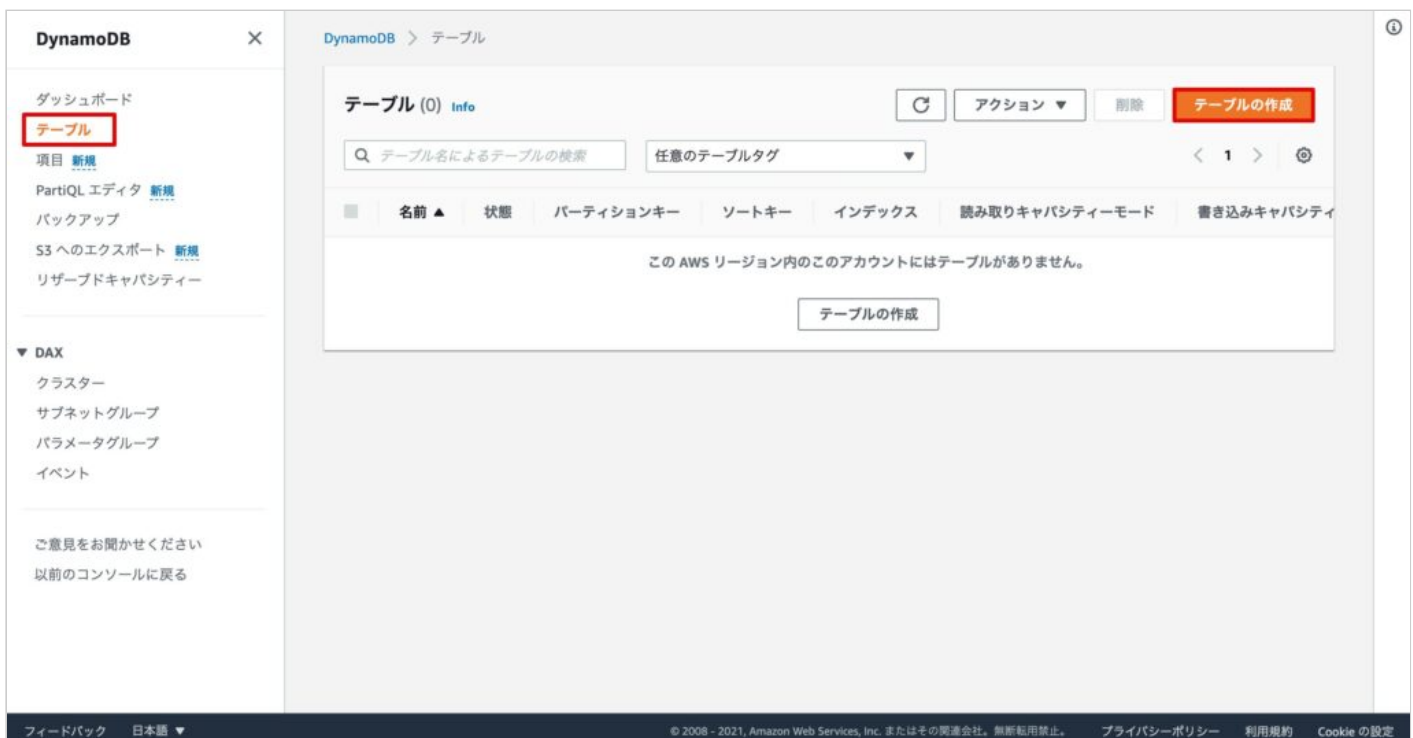
1. EC2上の監視サーバーで障害を検知し、S3へトリガーファイルを格納
2. S3のイベント通知機能で、SQS_1にメッセージを送信
3. SQS_1をトリガーとして、Lambda_1を起動
4. Lambda_1がDynamoDB_1から連絡先を取得し、AmazonConnectを起動
5. Lambda_1がAmazonConnectを起動すると同時に、SQS_2へメッセージを送信
6. AmazonConnectがユーザーへ自動電話通知を実施
7. ユーザーが正常応答せず、AmazonConnectがLambda_2を起動
8. Lambda_2が応答結果をDynamoDB_2に保存(応答NG)
9. 60秒後にSQS_2をトリガーとしてLambda_3を起動
10. Lambda_3がDynamoDB_2の応答結果を確認
11. 正常応答していないため、再度AmazonConnectを起動(以降、5から繰り返す)

DynamoDB_1を作成(連絡先情報を格納するDB)

AWSマネジメントコンソール上で、「dynamodb」を検索します。



「テーブルの作成」をクリックします。



下記の通り入力し、「テーブルの作成」をクリックします。

テーブル名：：任意の名前を入力 ※ここでは、“amazonconnect-contact-list”としています。
 パーティションキー：「No」と入力し、「数値」を選択
 ソートキー：「Name」と入力し、「文字列」を選択



DynamoDB > テーブル > テーブルの作成

テーブルの作成

テーブルの詳細 [Info](#)

DynamoDB は、テーブルの作成時にテーブル名とプライマリキーのみを必要とするスキーマレスデータベースです。

テーブル名

テーブルを識別するために使用されます。

3～255 文字で、文字、数字、アンダースコア (`_`)、ハイフン (`-`)、ピリオド (`.`) のみを使用できます。

パーティションキー

パーティションキーは、テーブルのプライマリキーの一部です。これは、テーブルから項目を取得し、スケーラビリティと可用性のためにホスト間でデータを割り当てるために使用されるハッシュ値です。

1～255 文字 (大文字と小文字が区別されます)。

ソートキー - オプション

ソートキーは、テーブルのプライマリキーの 2 番目の部分として使用できます。ソートキーにより、同じパーティションキーを共有するすべての項目をソートまたは検索できます。

1～255 文字 (大文字と小文字が区別されます)。

設定

☒ デフォルト設定

最も短時間でテーブルを作成します。これらの設定は、今すぐ変更するか、テーブルの作成後に変更できます。

☐ 設定のカスタマイズ

これらの高度な機能を使用して、DynamoDB をニーズに合わせて設定します。

デフォルト設定

読み込み/書き込みキャパシティー [Info](#)

プロビジョンドキャパシティーモードの使用中です。読み込みおよび書き込みキャパシティーは、Auto Scaling を有効にした状態で、それぞれ 5 ユニットに設定されます。

セカンダリインデックス [Info](#)

セカンダリインデックスは作成されていません。クエリは、テーブルのパーティションキーとソートキーのみを使用して実行されます。

保管時の暗号化のキー管理 [Info](#)

AWS 所有のカスタマーマスターキーを使用中です。このキーは追加料金なしで DynamoDB によって管理されます。

タグ

タグは、AWS リソースに割り当てることができるキーとオプション値のペアです。タグを使用して、リソースへのアクセスを制御したり、AWS の使用状況を追跡したりできます。

リソースに関連付けられたタグがありません。

[新しいタグの追加](#)

さらに 50 個のタグを追加できます。

[キャンセル](#)[テーブルの作成](#)[フィードバック](#) [日本語 ▼](#)© 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 [プライバシーポリシー](#) [利用規約](#) [Cookie の設定](#)

テーブルが作成されたことを確認し、テーブル名をクリックします。

The screenshot shows the AWS DynamoDB console. On the left is a navigation sidebar with links to 'ダッシュボード', 'テーブル' (highlighted), '項目', 'PartiQL エディタ', 'バックアップ', 'S3 へのエクスポート', and 'リザーブドキャパシティー'. Below these are 'DAX' options: 'クラスター', 'サブネットグループ', 'パラメータグループ', and 'イベント'. A message at the bottom of the sidebar says 'ご意見をお聞かせください' and '以前のコンソールに戻る'. The main area is titled 'DynamoDB > テーブル'. It shows a table named 'amazonconnect-contact-list' with the following details: Status is 'Active', Partition Key is 'No (Number)', Sort Key is 'Name (String)', Indexes are '0', and Read Capacity Mode is 'Auto Scaling をプロビジョニング'. At the top right of the table list, there are buttons for 'アクション', '削除', and 'テーブルの作成'. A search bar and a tag selector are also present.

「項目を表示」をクリックします。

This screenshot shows the details page for the 'amazonconnect-contact-list' table in the AWS DynamoDB console. The breadcrumb trail is 'DynamoDB > テーブル > amazonconnect-contact-list'. The table name is prominently displayed at the top. Below it are tabs for '概要' (selected), 'インデックス', 'モニタリング', 'グローバルテーブル', and 'バックアップ'. The '概要' (General Information) tab is active, showing a grid of key-value pairs: Partition Key (No (Number)), Sort Key (Name (String)), Capacity Mode (プロビジョンド), Table Status (Active, Activeアラームなし), Indexes (0 グローバル, 0 ローカル), DynamoDB Streams (Disabled), Point-in-Time Recovery (Disabled), Time to Live (TTL) (Disabled), Replication (0 リージョン), Encryption (Amazon が所有), and Creation Date. At the bottom, the Amazon Resource Name (ARN) is shown: 'arn:aws:dynamodb:ap-northeast-1:table/amazonconnect-contact-list'. The left sidebar is identical to the previous screenshot. The footer contains the same copyright and policy links.

「項目の作成」をクリックします。

The screenshot shows the AWS DynamoDB console interface. On the left is a navigation menu with options like 'ダッシュボード', 'テーブル', '項目', 'PartiQL エディタ', 'バックアップ', 'S3 へのエクスポート', 'リザーブドキャパシティー', and 'DAX'. The main area is titled '項目' (Items) and shows details for the 'amazonconnect-contact-list' table. It includes a 'タグ' (Tags) section, a 'スキャン' (Scan) button, and a 'クエリ' (Query) button. Below these is a 'テーブルまたはインデックス' (Table or Index) dropdown set to 'amazonconnect-contact-list'. A 'フィルター' (Filter) section is also present. At the bottom, the '返された項目 (0)' (Returned items (0)) section shows a search bar and a list of items. The '項目の作成' (Create Item) button is highlighted with a red box.

Noの値は「1」を、Nameの値は「User1(任意の名前)」を入力します。

The screenshot shows the '項目の作成' (Create Item) form in the AWS DynamoDB console. The form has a 'フォーム' (Form) tab and a 'JSON' tab. Under the '属性' (Attributes) section, there is a table with columns '属性名' (Attribute Name), '値' (Value), and 'タイプ' (Type). The 'No - パーティションキー' (No - Partition Key) row has a value of '1' and a type of '数値' (Number). The 'Name - ソートキー' (Name - Sort Key) row has a value of 'User1' and a type of '文字列' (String). A '新しい属性の追加' (Add new attribute) button is at the bottom left. At the bottom right, there are 'キャンセル' (Cancel) and '項目の作成' (Create Item) buttons. The '項目の作成' button is highlighted with a red box.

「新しい属性の追加」から「文字列」を追加します。

DynamoDB > 項目: amazonconnect-contact-list > 項目エディタ

項目の作成

フォーム JSON

属性名	値	タイプ
No - パーティションキー	1	数値
Name - ソートキー	User1	文字列

新しい属性の追加 ▲

- 文字列
- 数値
- ブール式
- バイナリ
- Null
- 文字列セット
- 数値セット
- バイナリセット
- リスト
- マップ

キャンセル 項目の作成

フィードバック 日本語 ▼ © 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約 Cookie の設定

属性名に「Phone」と入力し、値に電話番号を入力します。※電話番号は国番号を付けて入力(日本の090の番号の場合、+8190*****)

DynamoDB > 項目: amazonconnect-contact-list > 項目エディタ

項目の作成

フォーム JSON

属性名	値	タイプ
No - パーティションキー	1	数値
Name - ソートキー	User1	文字列
Phone	+81	文字列

新しい属性の追加 ▼

キャンセル 項目の作成

フィードバック 日本語 ▼ © 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約 Cookie の設定

「新しい属性の追加」から「数値」を追加します。

DynamoDB > 項目: amazonconnect-contact-list > 項目エディタ

項目エディタ

数値

属性

ブール式

バイナリ

Null

文字列セット

数値セット

バイナリセット

リスト

マップ

新しい属性の追加 ▲

値	タイプ
1	数値
User1	文字列
+81	文字列

キャンセル 項目の作成

属性名に「Priority」と入力し、値に「1」を入力します。※複数連絡先を登録するときの通知順です。入力が完了したら、「項目の作成」をクリックします。

DynamoDB > 項目: amazonconnect-contact-list > 項目エディタ

項目の作成

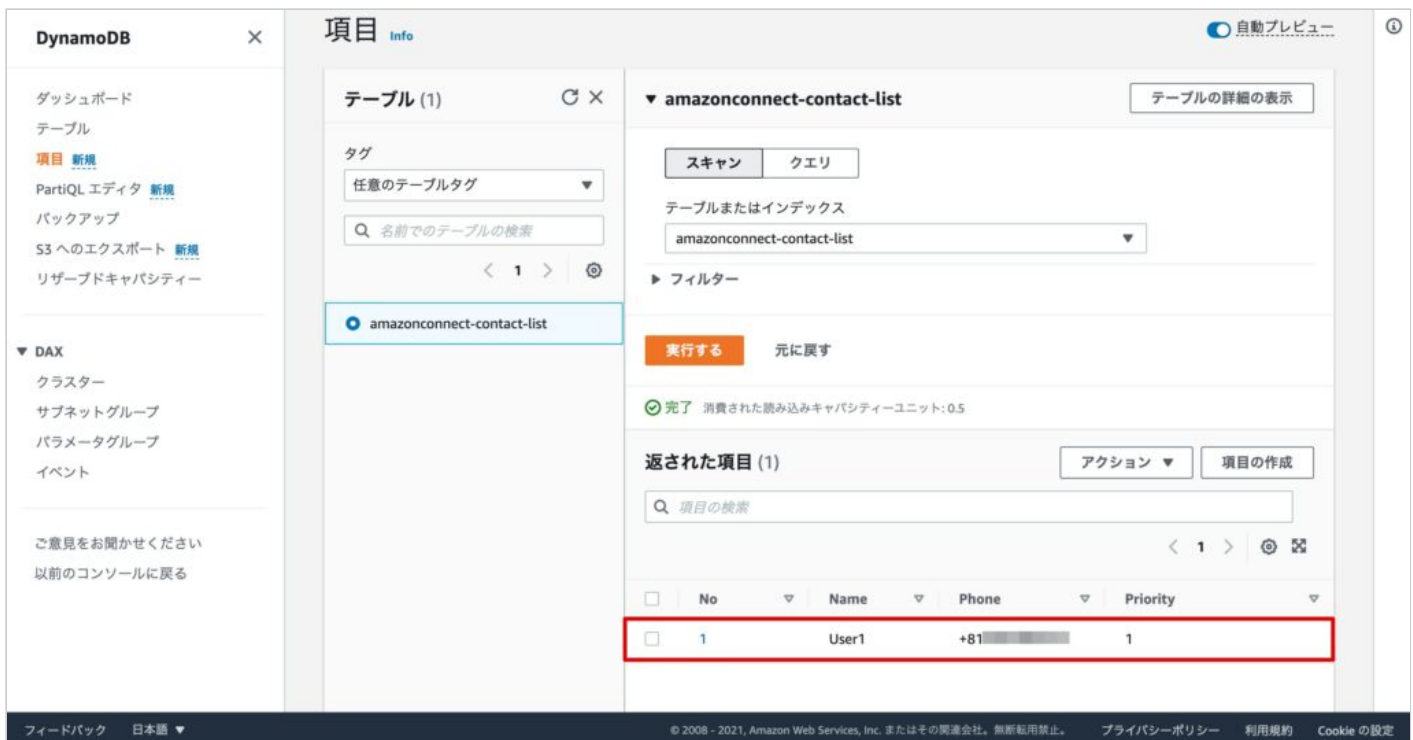
属性

属性名	値	タイプ
No - パーティションキー	1	数値
Name - ソートキー	User1	文字列
Phone	+81	文字列
Priority	1	数値

新しい属性の追加 ▼

キャンセル 項目の作成

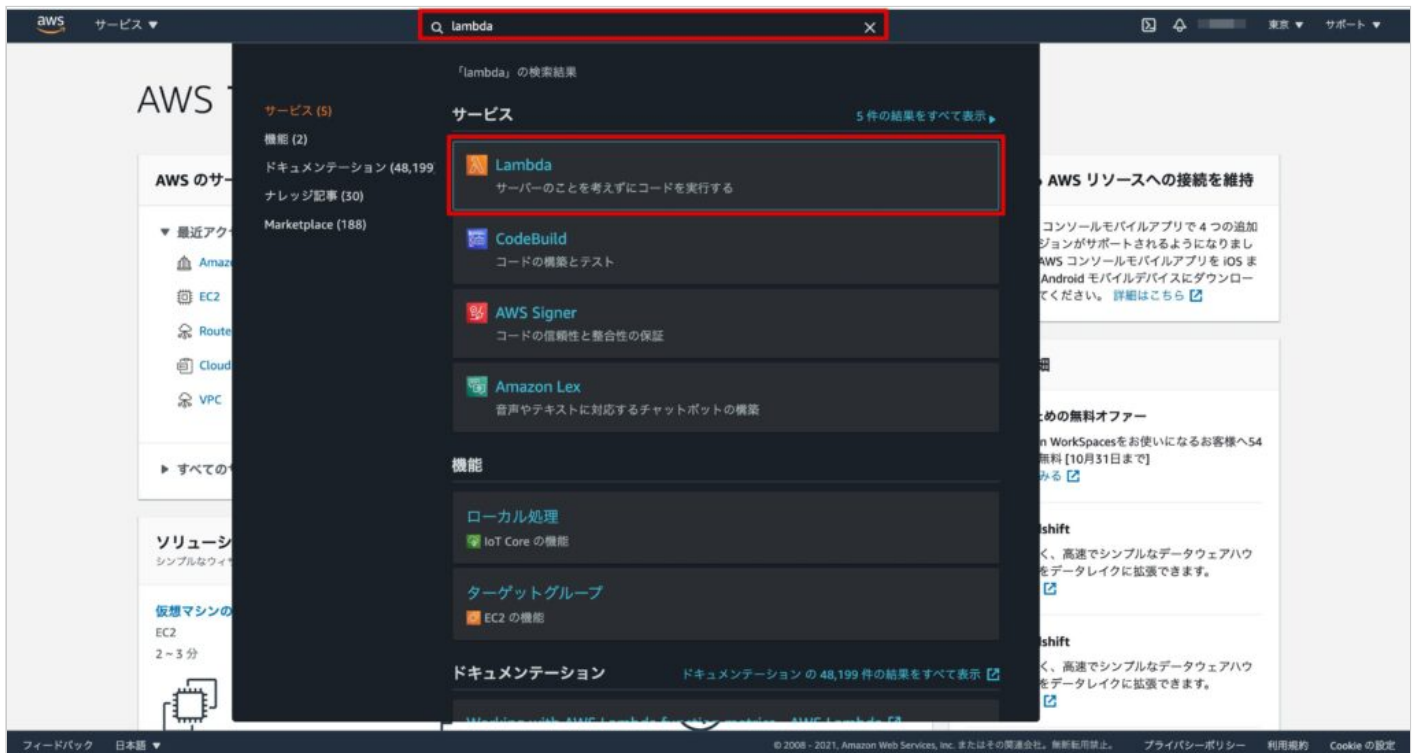
項目が追加されたことを確認します。



Lambda_1を作成(AmazonConnectを初回起動するプログラム)

関数の作成

AWSマネジメントコンソール上で、「lambda」を検索します。



「関数の作成」をクリックします。



下記の通り入力し、「関数の作成」をクリックします。

オプション：一から作成を選択

関数名：任意の名前を入力 ※ここでは、“amazonconnect-initiate”としています。

ランタイム：Pythonを選択 ※ここでは、最新版の“Python 3.9”を選択しています。

Lambda > 関数 > 関数の作成

関数の作成 Info

以下のいずれかのオプションを選択して、関数を作成します。

一から作成 Info
シンプルな Hello World の例で開始します。

設計図の使用
一般的ユースケース用のサンプルコードと設定プリセットから Lambda アプリケーションを構築します。

コンテナイメージ
関数にデプロイするコンテナイメージを選択します。

Serverless Application Repository の参照
AWS Serverless Application Repository からサンプル Lambda アプリケーションをデプロイします。

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。

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 Info
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。

アーキテクチャ Info
関数コードに必要な命令セットアーキテクチャを選択します。
☒ x86_64
☐ arm64

アクセス権限 Info
デフォルトでは、Lambda は Amazon CloudWatch Logs にログをアップロードするアクセス許可を持つ実行ロールを作成します。このデフォルトのロールは、後でトリガーを追加するときにカスタマイズできます。
[▶ デフォルトの実行ロールの変更](#)

[▶ 詳細設定](#)

キャンセル **関数の作成**

フィードバック 日本語 ▼ © 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約 Cookie の設定


関数が作成されたことを確認します。

Lambda > 関数 > amazonconnect-initiate

amazonconnect-initiate

スロットリング ARN をコピー アクション ▼


▼ 関数の概要 Info

 **amazonconnect-initiate**
Layers (0)

[+ トリガーを追加](#) [+ 送信先を追加](#)

説明
-

最終更新
4 秒前

関数の ARN
 arn:aws:lambda:ap-northeast-1:123456789012:function:amazonconnect-initiate

コード テスト モニタリング 設定 **エイリアス** バージョン

コードソース Info

アップロード元 ▼

File Edit Find View Go Tools Window **Test** Deploy **Changes deployed**

Go to Anything (⌘P)

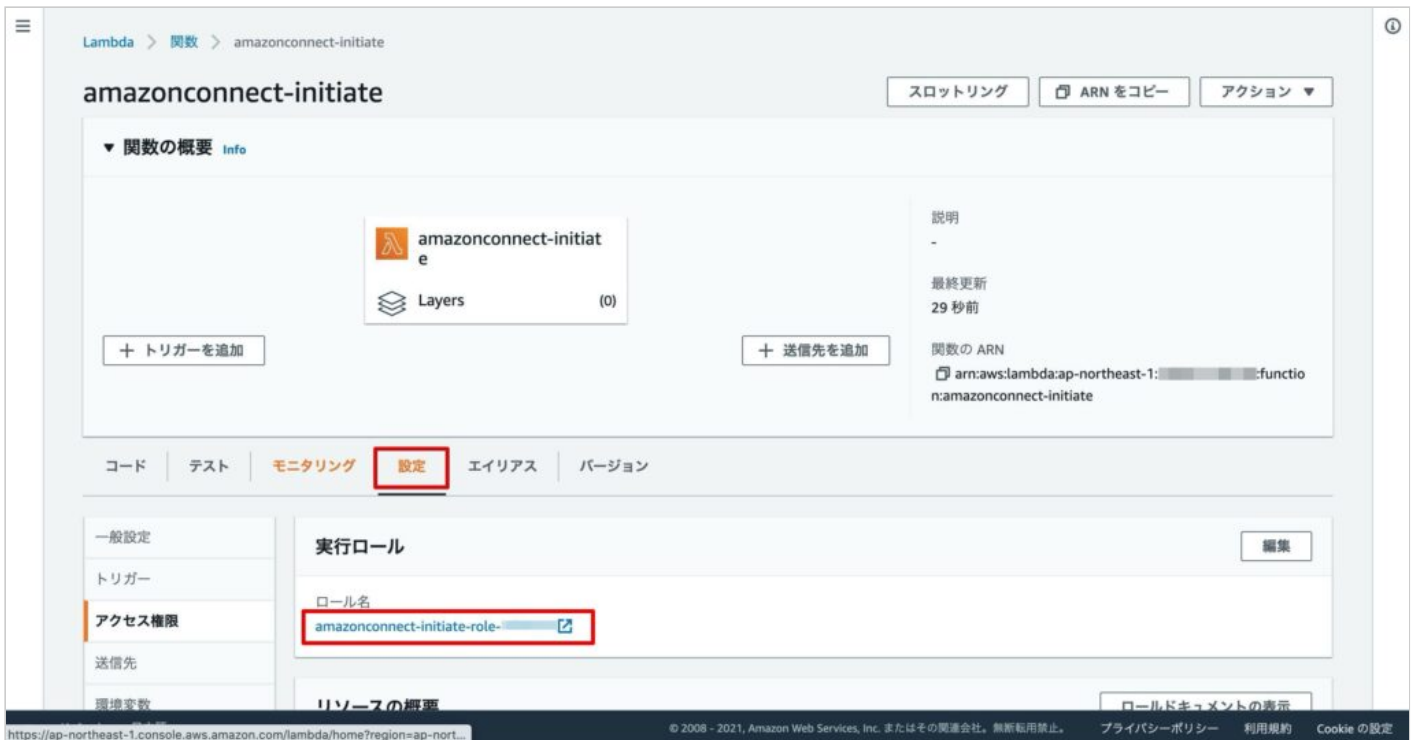
amazonconnect-init
lambda_function.py

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO: Implement the handler logic
```

フィードバック 日本語 ▼ © 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約 Cookie の設定

アクセス権の追加（ロールの設定）

作成された関数の設定タブに移動し、実行ロールをクリックします。



「ポリシーをアタッチします」をクリックします。



“connect”で検索し、「AmazonConnect_FullAccess」にチェックを入れます。

amazonconnect-initiate-role- [redacted] にアクセス権を追加する
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 10 件の結果を表示中

ポリシー名	タイプ	次として使用
<input checked="" type="checkbox"/> AmazonConnect_FullAccess	AWS による管理	Permissions policy (2)
<input type="checkbox"/> AmazonConnectReadOnlyAccess	AWS による管理	なし
<input type="checkbox"/> AmazonConnectVoiceIDFullAccess	AWS による管理	なし
<input type="checkbox"/> AmazonMSKConnectReadOnlyAccess	AWS による管理	なし
<input type="checkbox"/> AWSConnector	AWS による管理	なし
<input type="checkbox"/> AWSDirectConnectFullAccess	AWS による管理	なし
<input type="checkbox"/> AWSDirectConnectReadOnlyAccess	AWS による管理	なし
<input type="checkbox"/> EC2InstanceConnect	AWS による管理	なし
<input type="checkbox"/> ServerMigrationConnector	AWS による管理	なし
<input type="checkbox"/> VMImportExportRoleForAWSConnector	AWS による管理	なし

キャンセル ポリシーのアタッチ

フィードバック 日本語 © 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約 Cookie の設定

“sqs”で検索し、「AmazonSQSFullAccess」にチェックを入れます。

amazonconnect-initiate-role- [redacted] にアクセス権を追加する
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 3 件の結果を表示中

ポリシー名	タイプ	次として使用
<input checked="" type="checkbox"/> AmazonSQSFullAccess	AWS による管理	なし
<input type="checkbox"/> AmazonSQSReadOnlyAccess	AWS による管理	なし
<input type="checkbox"/> AWSLambdaSQSQueueExecutionRole	AWS による管理	なし

キャンセル ポリシーのアタッチ

フィードバック 日本語 © 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約 Cookie の設定

“dynamo”で検索し、「AmazonDynamoDBFullAccess」にチェックを入れ、「ポリシーのアタッチ」をクリックします。

amazonconnect-initiate-role- [redacted] 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 4件の結果を表示中

ポリシー名	タイプ	次として使用
<input checked="" type="checkbox"/> AmazonDynamoDBFullAccess	AWS による管理	なし
<input type="checkbox"/> AmazonDynamoDBReadOnlyAccess	AWS による管理	なし
<input type="checkbox"/> AWSLambdaDynamoDBExecutionRole	AWS による管理	なし
<input type="checkbox"/> AWSLambdaInvocation-DynamoDB	AWS による管理	なし

キャンセル ポリシーのアタッチ

フィードバック 日本語 © 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約 Cookie の設定

ポリシーがアタッチされたことを確認します。

Identity and Access Management (IAM)

ダッシュボード

- アクセス管理
 - ユーザーグループ
 - ユーザー
 - ロール**
 - ポリシー
 - ID プロバイダー
 - アカウント設定
- アクセスレポート
 - アクセスアナライザー
 - アーカイブルール
 - アナライザー
 - 設定
- 認証情報レポート
- 組織アクティビティ
- サービスコントロールポリシー (SCP)

ロール > amazonconnect-initiate-role-[redacted]

概要 ロールの削除

ロール ARN: arn:aws:iam::[redacted]:role/service-role/amazonconnect-initiate-role-[redacted] 🔗

ロールの説明 編集

インスタンスプロファイル ARN: 🔗

パス: /service-role/

作成時刻: [redacted]

最後のアクティビティ: [redacted]

最大セッション時間: 1 時間 編集

アクセス権限 信頼関係 タグ アクセスアドバイザー セッションの無効化

▼ Permissions policies (4 適用済みポリシー)

ポリシーをアタッチします インラインポリシーの追加

ポリシー名	ポリシータイプ	
AWSLambdaBasicExecutionRole-[redacted]	管理ポリシー	✕
<input checked="" type="checkbox"/> AmazonSQSFullAccess	AWS 管理ポリシー	✕
<input checked="" type="checkbox"/> AmazonDynamoDBFullAccess	AWS 管理ポリシー	✕
<input checked="" type="checkbox"/> AmazonConnect_FullAccess	AWS 管理ポリシー	✕

トリガーの追加

「トリガーを追加」をクリックします。



「トリガーを選択」は、「SQS」を選択します。



作成したSQS_1を選択し、「追加」をクリックします。

The screenshot shows the 'Add Trigger' page in the AWS Lambda console. The trigger type is set to 'SQS' (aws queue). The SQS queue ARN is entered as 'arn:aws:sqs:ap-northeast-1: [redacted] :amazonconnect-queue-trigger'. The batch size is set to 10, and the batch window is set to 0. The 'Trigger enabled' checkbox is checked. The 'Add' button is highlighted in orange.

トリガーを追加

トリガーの設定

SQS
aws queue

SQS キュー
SQS キューの ARN を選択または入力します。

arn:aws:sqs:ap-northeast-1: [redacted] :amazonconnect-queue-trigger

バッチサイズ
単一のバッチで取得するメッセージの最大数です。

10

バッチウィンドウ
関数を呼び出すまでにレコードを収集する最大時間 (秒単位)。

0

SQS トリガーから読み取るには、実行ロールに適切なアクセス権限が設定されている必要があります。

☒ トリガーの有効化
今すぐトリガーを有効化するか、テスト用に無効化した状態でトリガーを作成します (推奨)。

キャンセル 追加

トリガーが追加されていることを確認します。（設定タブ→トリガー）

The screenshot shows the 'Settings' page for the 'amazonconnect-initiate' function in the AWS Lambda console. The 'Triggers' tab is selected, and a red box highlights the 'SQS: amazonconnect-queue-trigger (有効)' trigger. The trigger details show the ARN 'arn:aws:sqs:ap-northeast-1:206013296236:amazonconnect-queue-trigger', batch window 'なし', and batch size '10'.

amazonconnect-initiate

スロットリング ARN をコピー アクション

関数の概要 Info

amazonconnect-initiate

Layers (0)

SQS

+ 送信先を追加

+ トリガーを追加

説明
-
最終更新
5 日前
関数の ARN
arn:aws:lambda:ap-northeast-1: [redacted] :function:amazonconnect-initiate

コード テスト モニタリング **設定** エイリアス バージョン

一般設定

トリガー

アクセス権限

送信先

環境変数

タグ

VPC

モニタリングおよび運用ツ

トリガー (1)

トリガーを検索

トリガー

SQS: amazonconnect-queue-trigger (有効)
arn:aws:sqs:ap-northeast-1:206013296236:amazonconnect-queue-trigger

▼ 詳細

バッチウィンドウ: なし
バッチサイズ: 10

コードの記述

Lambdaのコードを記述します。

```
-----  
import json  
import boto3  
from boto3.dynamodb.conditions import Key, Attr  
  
# boto3からDynamoDBへアクセスするためのオブジェクトを取得  
dynamodb = boto3.resource('dynamodb')  
  
# "amazonconnect-contact-list"へアクセスするためのオブジェクトを取得  
contactlist = dynamodb.Table("amazonconnect-contact-list")  
  
# "amazonconnect-contact-list"の内容を返す関数  
def operation_scan():  
  
    scanData = contactlist.scan()  
    items=scanData['Items']  
  
    return scanData["Items"]  
  
# 指定されたプライオリティの電話番号を返す関数  
def phone_get(json_contactinfo, now_priority):  
  
    for line in json_contactinfo:  
        if line['Priority']==now_priority:  
            phone_number = line['Phone']  
  
    return phone_number  
  
def lambda_handler(event, context):  
  
    # 電話番号リストを取得  
    contactinfo = operation_scan()  
  
    # 初回のプライオリティを"1"に設定  
    priority = 1  
  
    # 指定したプライオリティの電話番号を取得  
    phone_number = phone_get(contactinfo, priority)  
  
    connect = boto3.client('connect')
```

```
connect.start_outbound_voice_contact(
    DestinationPhoneNumber=phone_number,
    ContactFlowId='*****',
    InstanceId='*****',
    SourcePhoneNumber='+81*****',
)
```

ContactFlowId: 問い合わせフローID

InstanceId: インスタンスID

SourcePhoneNumber: 発信元の電話番号

※国番号をつけて記述(日本の050の番号の場合、+8150*****)

```
1 import json
2 import boto3
3 from boto3.dynamodb.conditions import Key, Attr
4
5 # boto3からDynamoDBへアクセスするためのオブジェクトを取得
6 dynamodb = boto3.resource('dynamodb')
7
8 # "amazonconnect-contact-list"へアクセスするためのオブジェクトを取得
9 contactlist = dynamodb.Table("amazonconnect-contact-list")
10
11 # "amazonconnect-contact-list"の内容を返す関数
12 def operation_scan():
13     scanData = contactlist.scan()
14     items=scanData['Items']
15     return scanData["Items"]
16
17
18
19 # 指定されたプライオリティの電話番号を返す関数
20 def phone_get(json_contactinfo, now_priority):
21     for line in json_contactinfo:
22         if line['Priority']==now_priority:
23             phone_number = line['Phone']
24     return phone_number
25
26
27
28
29
30 def lambda_handler(event, context):
31     # 電話番号リストを取得
32     contactinfo = operation_scan()
33
34     # 初回のプライオリティを"1"に設定
35     priority = 1
36
37     # 指定したプライオリティの電話番号を取得
38     phone_number = phone_get(contactinfo, priority)
39
40     connect = boto3.client('connect')
41
42     connect.start_outbound_voice_contact(
43         DestinationPhoneNumber=phone_number,
44         ContactFlowId='*****',
45         InstanceId='*****',
46         SourcePhoneNumber='+81*****',
47     )
48
49
50
51
```

Lambda_1のテスト

EC2から下記のコマンドを実施し、S3へファイルをアップロードします。※黄色アンダーライン箇所は、作成したS3バケット名を指定してください。

```
touch /tmp/test.txt
aws s3 cp /tmp/test.txt s3://amazonconnect-alert-notification-bucket
```

```
[ec2-user@ip-10-0-0-100 ~]$ touch /tmp/test.txt
[ec2-user@ip-10-0-0-100 ~]$
[ec2-user@ip-10-0-0-100 ~]$ aws s3 cp /tmp/test.txt s3://amazonconnect-alert-
notification-bucket
upload: ../../tmp/test.txt to s3://amazonconnect-alert-notification-
bucket/test.txt
[ec2-user@ip-10-0-0-100 ~]$
```

自動電話通知フローの1～4までが実行され、プライオリティ"1"に指定した電話番号に着信があり、問い合わせフローで設定した音声再生されることを確認します。

1. EC2上の監視サーバーで障害を検知し、S3へトリガーファイルを格納
2. S3のイベント通知機能で、SQS_1にメッセージを送信
3. SQS_1をトリガーとして、Lambda_1を起動
4. Lambda_1がDynamoDB_1から連絡先を取得し、AmazonConnectを起動

以上で、AmazonConnectによる自動電話通知（7.複数連絡先への電話通知〈構築④〉）の説明は完了です！