

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〉）

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⑤〉）

2021.11.12 2021.11.03

[【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④〉）](#)[【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⑥〉）](#)[【簡易版】 AmazonConnectによる自動電話通知（まとめ）](#)

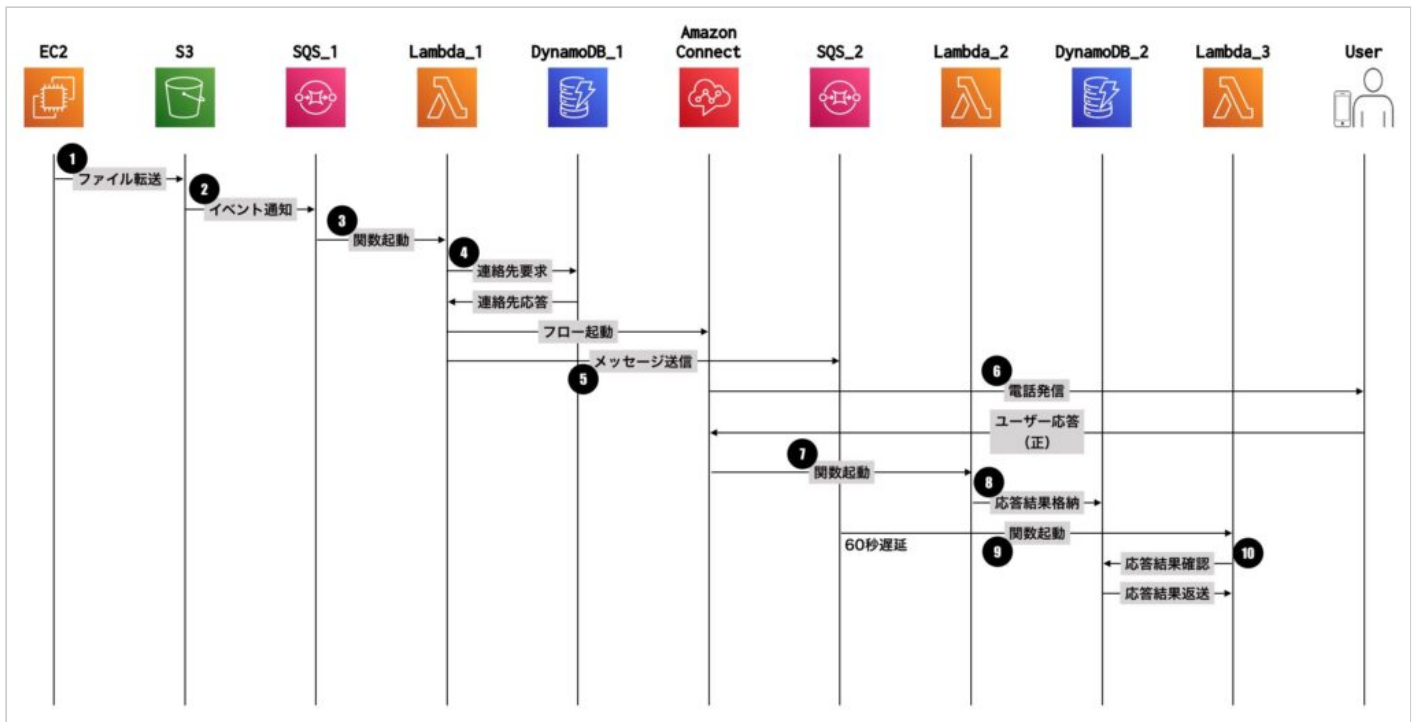
監視サーバーで障害を検知した際に、自動で電話通知できるようにしていきます。ネットワークエンジニアも利用することの多い監視サーバー(Zabbix)で障害検知し、AWS上のAmazonConnectを利用し自動電話を発信します。

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。

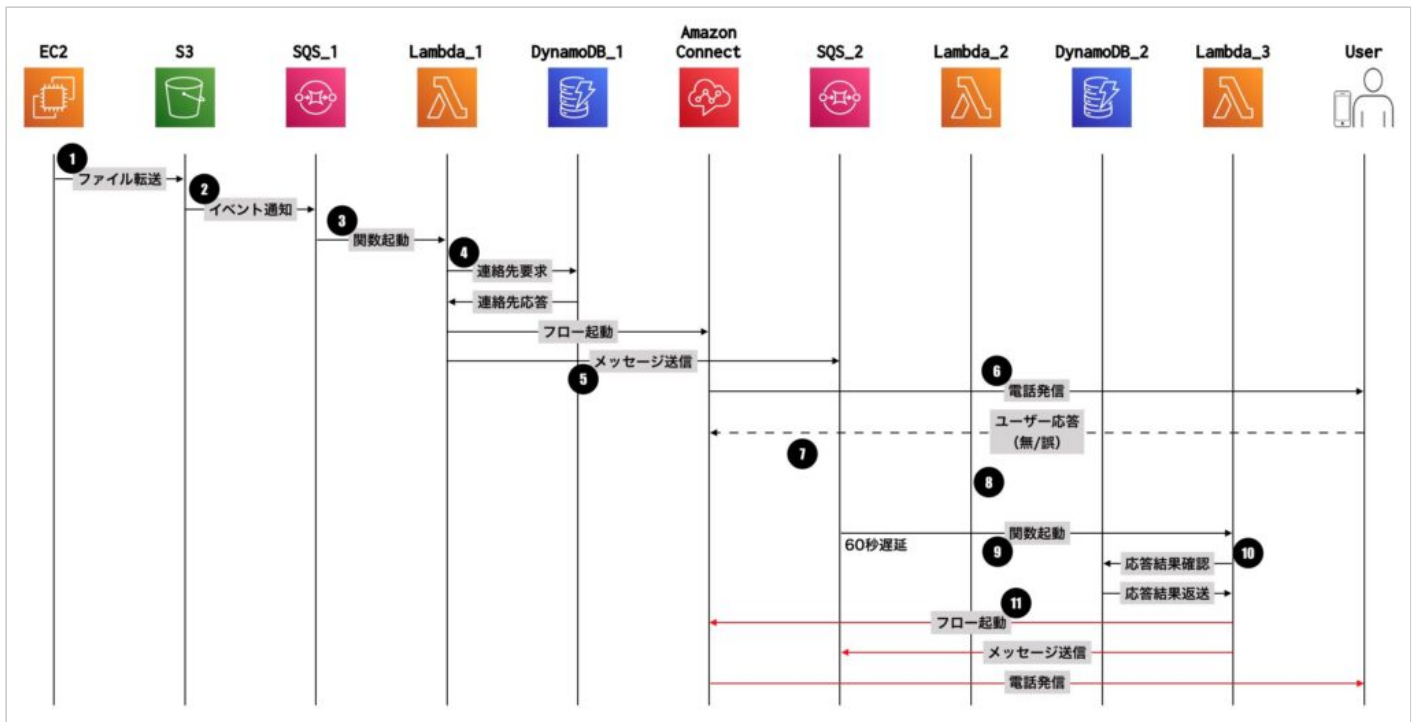
自動電話通知フロー

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から繰り返し)

DynamoDB_2を作成(通知結果を保存するDB)

DynamoDBの「テーブルの作成」をクリックします。

DynamoDB > テーブル

テーブル (1) Info

テーブル名によるテーブルの検索

任意のテーブルタグ

名前	状態	パーティションキー	ソートキー	インデックス	読み取りキャパシティーモード
amazonconnect-contact-list	Active	No (Number)	Name (String)	0	Auto Scaling をプロビジョニング

テーブルの作成

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-response-status”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。
 amazonconnect-response-status
 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。
 No
 数値

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。
 Name
 文字列

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

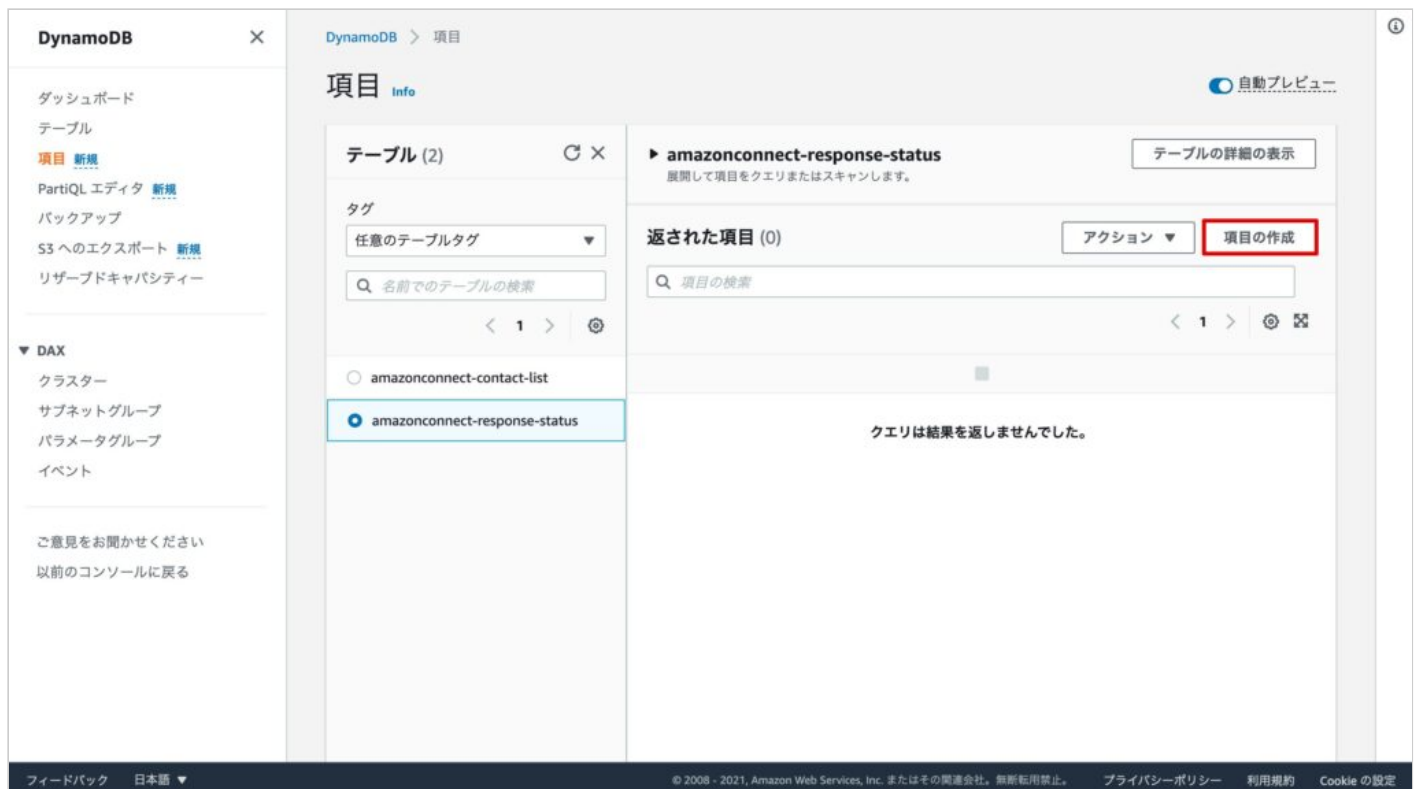
テーブルが作成されたことを確認し、テーブル名をクリックします。

The screenshot shows the AWS DynamoDB console interface. On the left is a navigation menu with options like 'ダッシュボード', 'テーブル', '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'. The main area displays a list of tables under the heading 'テーブル (2) Info'. The table 'amazonconnect-response-status' is highlighted with a red box. The table details include: Name (amazonconnect-response-status), Status (Active), Partition Key (No (Number)), Sort Key (Name (String)), Indexes (0), and Auto Scaling (Auto Scaling をプロビジョニング済み).

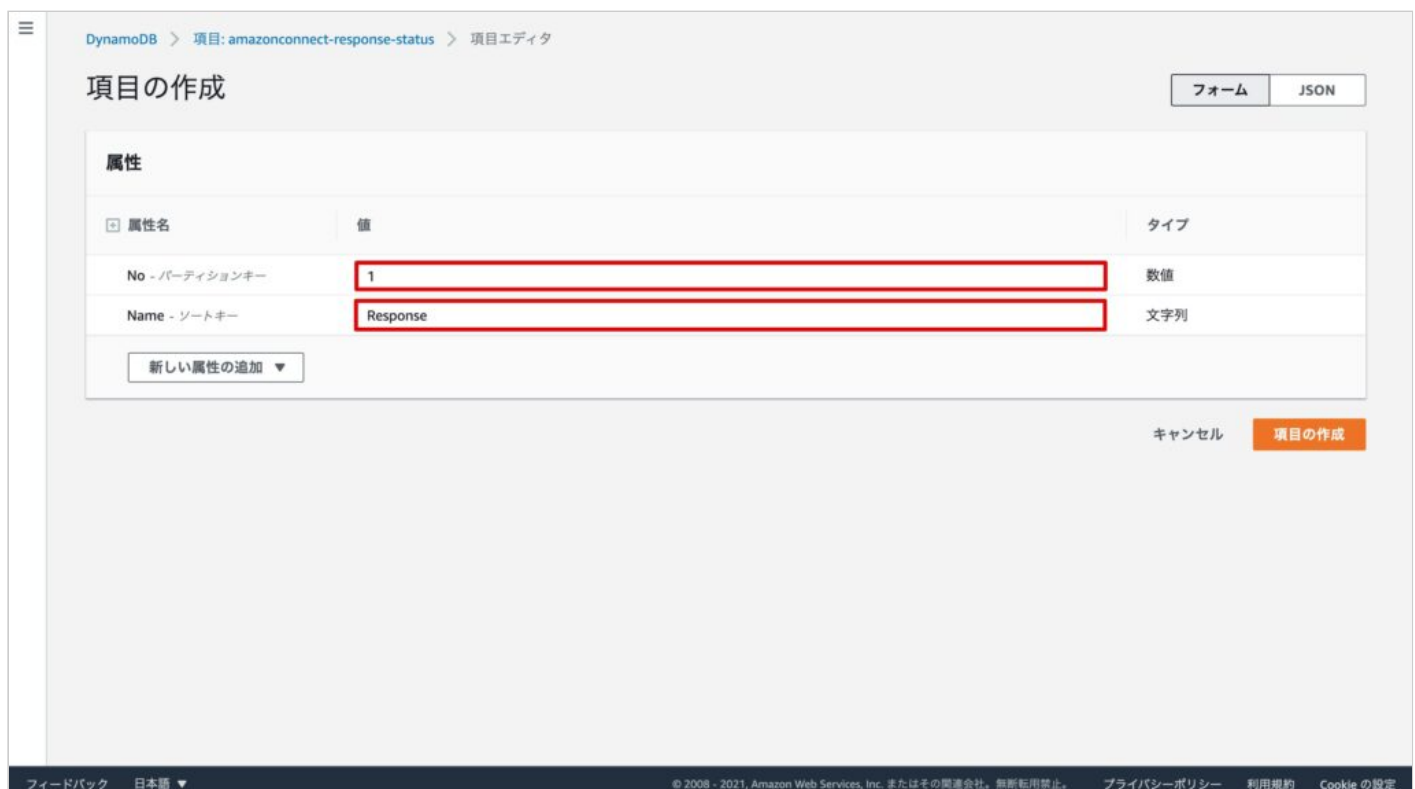
「項目を表示」をクリックします。

The screenshot shows the details of the 'amazonconnect-response-status' table in the AWS DynamoDB console. The breadcrumb navigation shows 'DynamoDB > テーブル > amazonconnect-response-status'. The table name 'amazonconnect-response-status' is highlighted in the left sidebar. The main area shows the '一般的な情報' (General Information) tab. The table's status is 'Active' and 'アクティブなアラームなし' (No active alarms). The '項目を表示' (Show Items) button is highlighted in red. The '項目の概要' (Item Summary) section shows 'ライブ項目数' (Live Item Count) and '項目を表示' (Show Items) buttons.

「項目の作成」をクリックします。



Noの値は「1」を、Nameの値は「Response」を入力します。



「新しい属性の追加」から「文字列」を追加します。

DynamoDB > 項目: amazonconnect-response-status > 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	Response	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「CallStatus」と入力し、値に「NG」を入力します。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-response-status > 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	Response	文字列
CallStatus	NG	文字列

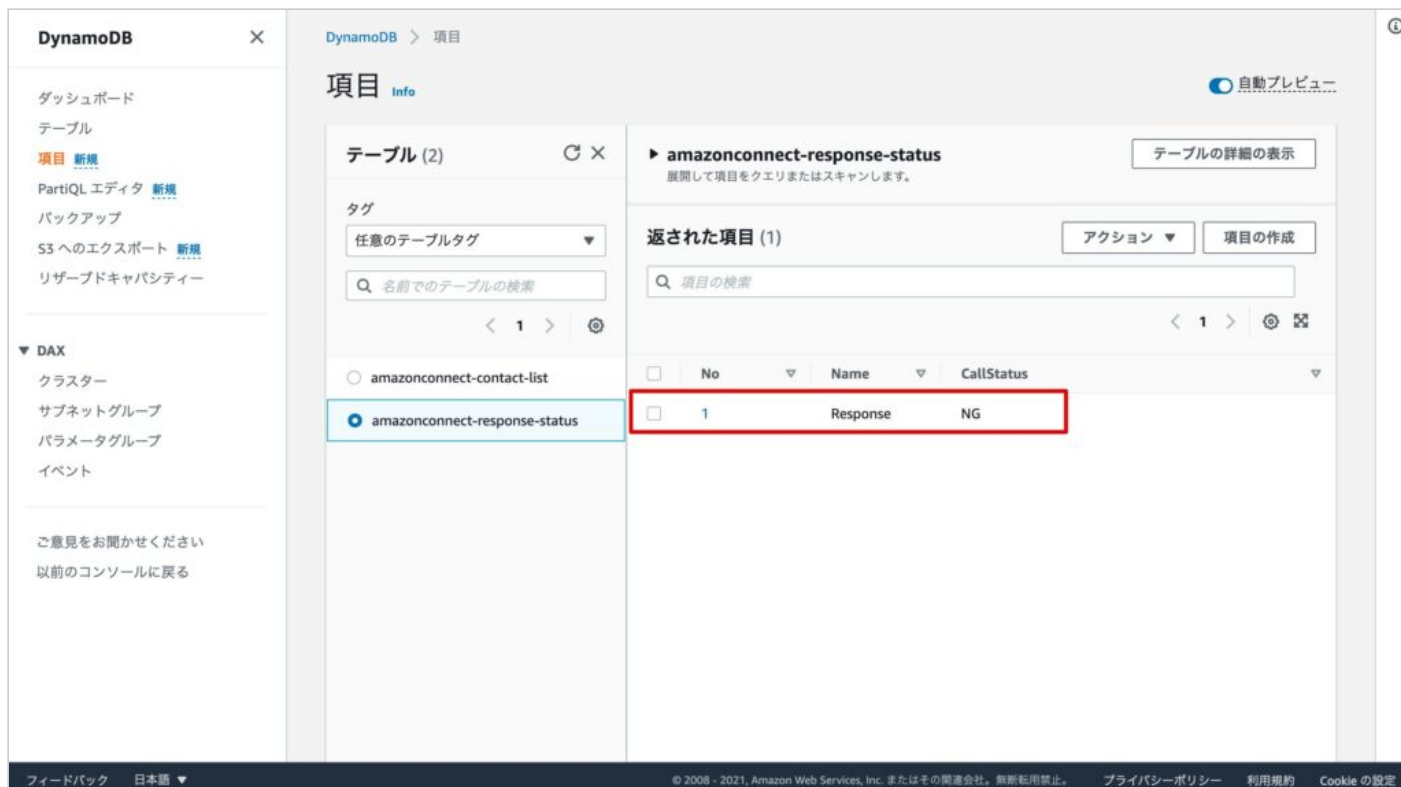
新しい属性の追加 ▼

削除

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

項目が追加されたことを確認します。



Lambda_2を作成(AmazonConnectの通知結果を保存する関数)

関数の作成

Lambdaの「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-status-save”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。

The screenshot shows the 'Create Function' page in the AWS Lambda console. The breadcrumb navigation is 'Lambda > 関数 > 関数の作成'. The page title is '関数の作成' with an 'Info' link. Below the title, it says '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。' (Select one of the following options to create a function).

Four options are presented in boxes:

- 一から作成** (Selected with a blue dot): シンプルな Hello World の例で開始します。 (Start with a simple Hello World example.)
- 設計図の使用** (Unselected): 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。 (Build a Lambda application from sample code and configuration presets for common use cases.)
- コンテナイメージ** (Unselected): 関数にデプロイするコンテナイメージを選択します。 (Select a container image to deploy to the function.)
- Serverless Application Repository の参照** (Unselected): 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。 (Deploy a sample Lambda application from the AWS Serverless Application Repository.)

Below the options is the '基本的な情報' (Basic Information) section:

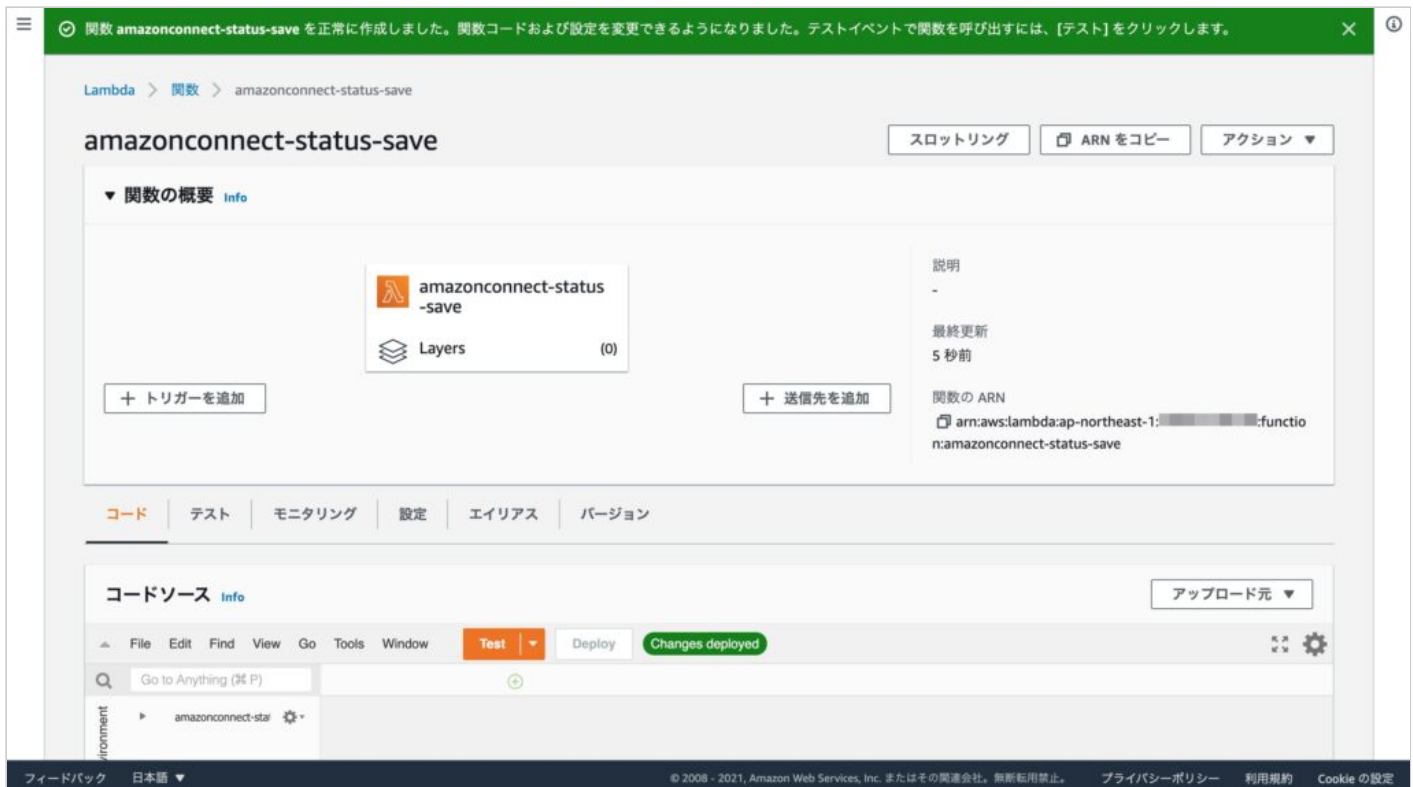
- 関数名** (Function Name): 関数の目的を名前として入力します。 (Enter the purpose of the function as the name.) The input field contains 'amazonconnect-status-save'. A note below says: '半角英数字、ハイフン、アンダースコアのみを使用でき、スペースは使用できません。' (Only alphanumeric characters, hyphens, and underscores are allowed; spaces are not allowed.)
- ランタイム** (Runtime): 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。 (Select the language to use for the function's code. Note that the console code editor only supports Node.js, Python, and Ruby.) The dropdown menu shows 'Python 3.9'.
- アーキテクチャ** (Architecture): 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。 (Select the instruction set architecture required for the function code.) The radio buttons show 'x86_64' (selected) and 'arm64'.
- アクセス権限** (Permissions): 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。 (By default, Lambda creates an execution role with permissions to upload logs to Amazon CloudWatch Logs. This default role can be customized when you add triggers later.)

At the bottom of the '基本的な情報' section, there are two expandable sections: 'デフォルトの実行ロールの変更' (Change default execution role) and '詳細設定' (Advanced settings).

At the bottom right of the page, there are two buttons: 'キャンセル' (Cancel) and '関数の作成' (Create function).

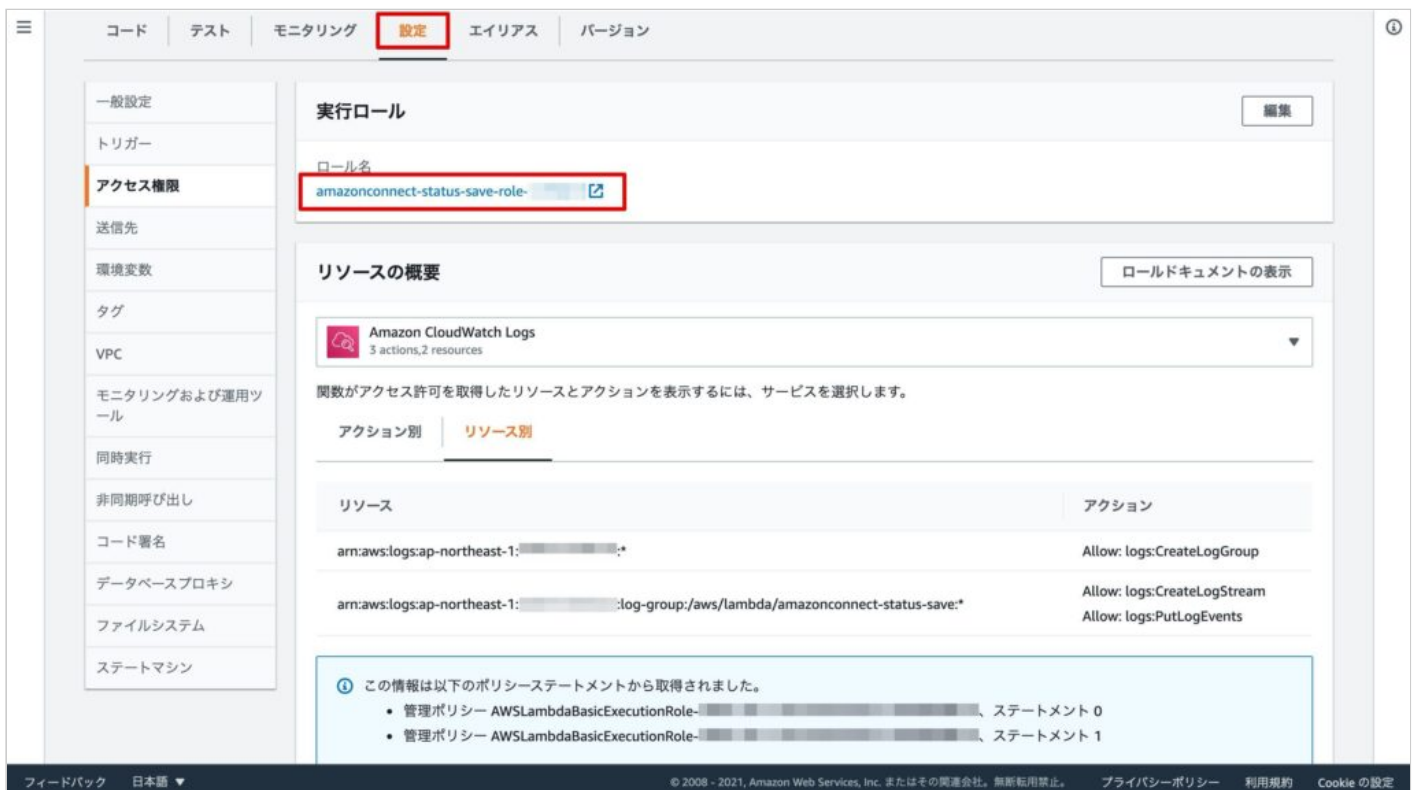
The footer contains: 'フィードバック' (Feedback), '日本語' (Japanese), '© 2008 - 2021, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。' (Copyright 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. No unauthorized reproduction.), 'プライバシーポリシー' (Privacy Policy), '利用規約' (Terms of Use), and 'Cookie の設定' (Cookie settings).

関数が作成されたことを確認します。



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。

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)

Q IAM の検索

AWS アカウント ID: 206013296236

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

ロール > amazonconnect-status-save-role- [redacted]

概要

ロールの削除

ロール ARN: arn:aws:iam::[redacted]:role/service-role/amazonconnect-status-save-role-[redacted]

ロールの説明: 編集

インスタンスプロファイル ARN: [redacted]

パス: /service-role/

作成時刻: [redacted]

最後のアクティビティ: 追跡期間中はアクセスされません

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 (1 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

▼ Permissions boundary (not set)

▼ CloudTrail イベントに基づいてポリシーを生成

この ロール のアクセスアクティビティに基づいて新しいポリシーを生成し、カスタマイズおよび作成したり、このロールにアタッチしたりできます。AWS は CloudTrail イベントを使用して、使用されるサービスとアクションを識別し、ポリシーを生成します。 [詳細はこちら](#)

Share your [feedback](#) and help us improve the policy generation experience.

“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-status-save-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 ▼ Q dynamo 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	Permissions policy (1)
<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)

Q IAM の検索

AWS アカウント ID: 206013296236

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

ロール > amazonconnect-status-save-role

概要

ロールの削除

ロール ARN: arn:aws:iam:::role/service-role/amazonconnect-status-save-role

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:

最後のアクティビティ: 追跡期間中はアクセスされません

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 (2 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	
AmazonDynamoDBFullAccess	AWS 管理ポリシー	✕
AWSLambdaBasicExecutionRole	管理ポリシー	✕

▼ Permissions boundary (not set)

▼ CloudTrail イベントに基づいてポリシーを生成

このロールのアクセスアクティビティに基づいて新しいポリシーを生成し、カスタマイズおよび作成したり、このロールにアタッチしたりできます。AWS は CloudTrail イベントを使用して、使用されるサービスとアクションを識別し、ポリシーを生成します。 [詳細はこちら](#)

コードの記述

Lambdaのコードを記述します。

```
import boto3
import json
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-response-status"へアクセスするためのオブジェクトを取得
responsestatus = dynamodb.Table("amazonconnect-response-status")

# CallStatusを"OK"に変更する関数
def status_ok():
    response = responsestatus.update_item(
        Key={
            'No': 1,
            'Name': "Response"
        },
        UpdateExpression="set CallStatus=:c",
        ExpressionAttributeValues={
```

```

        'c': "OK"
    },
    ReturnValues="UPDATED_NEW"
)
return response

```

```
def lambda_handler(event, context):
```

```

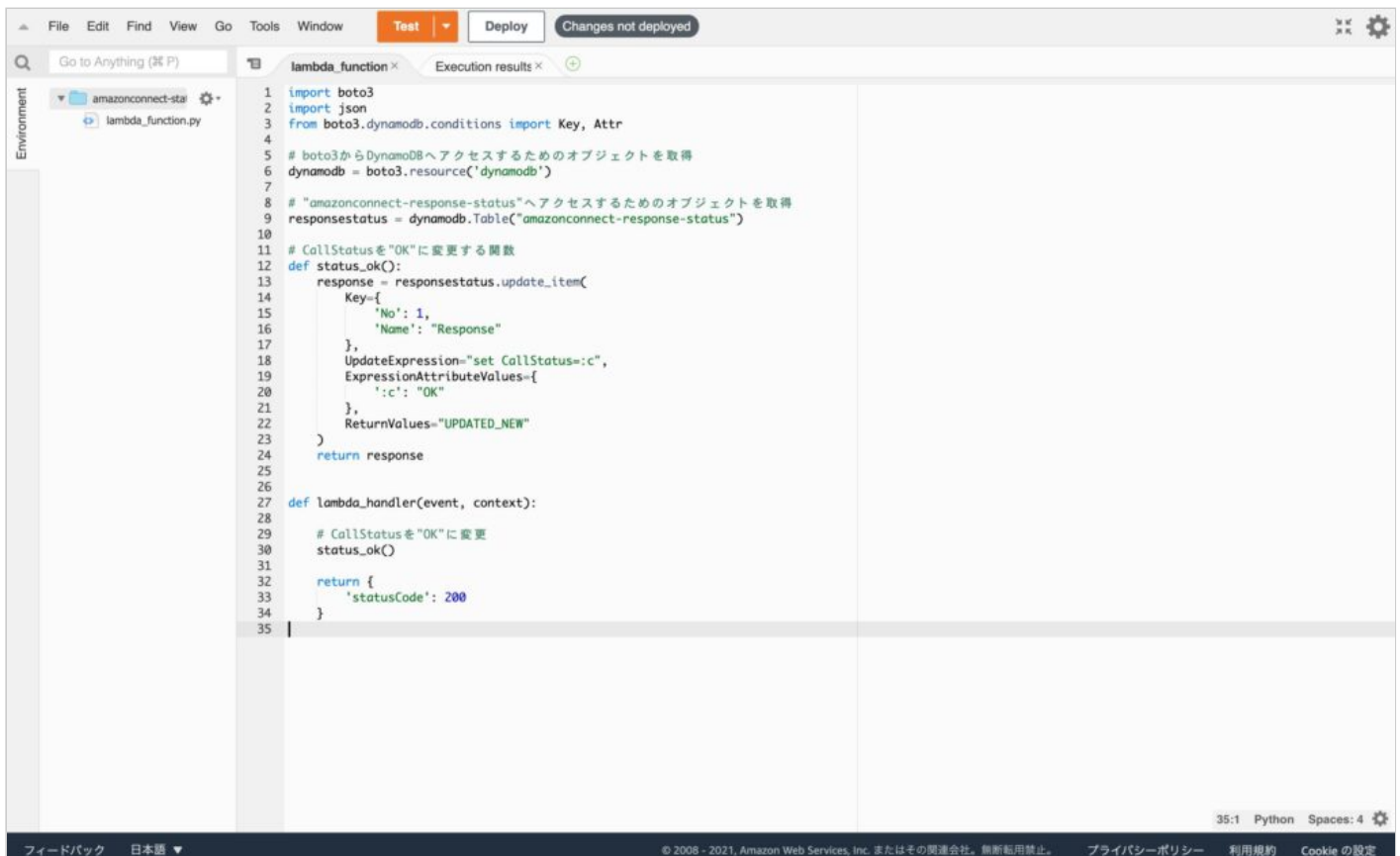
    # CallStatusを"OK"に変更
    status_ok()

```

```

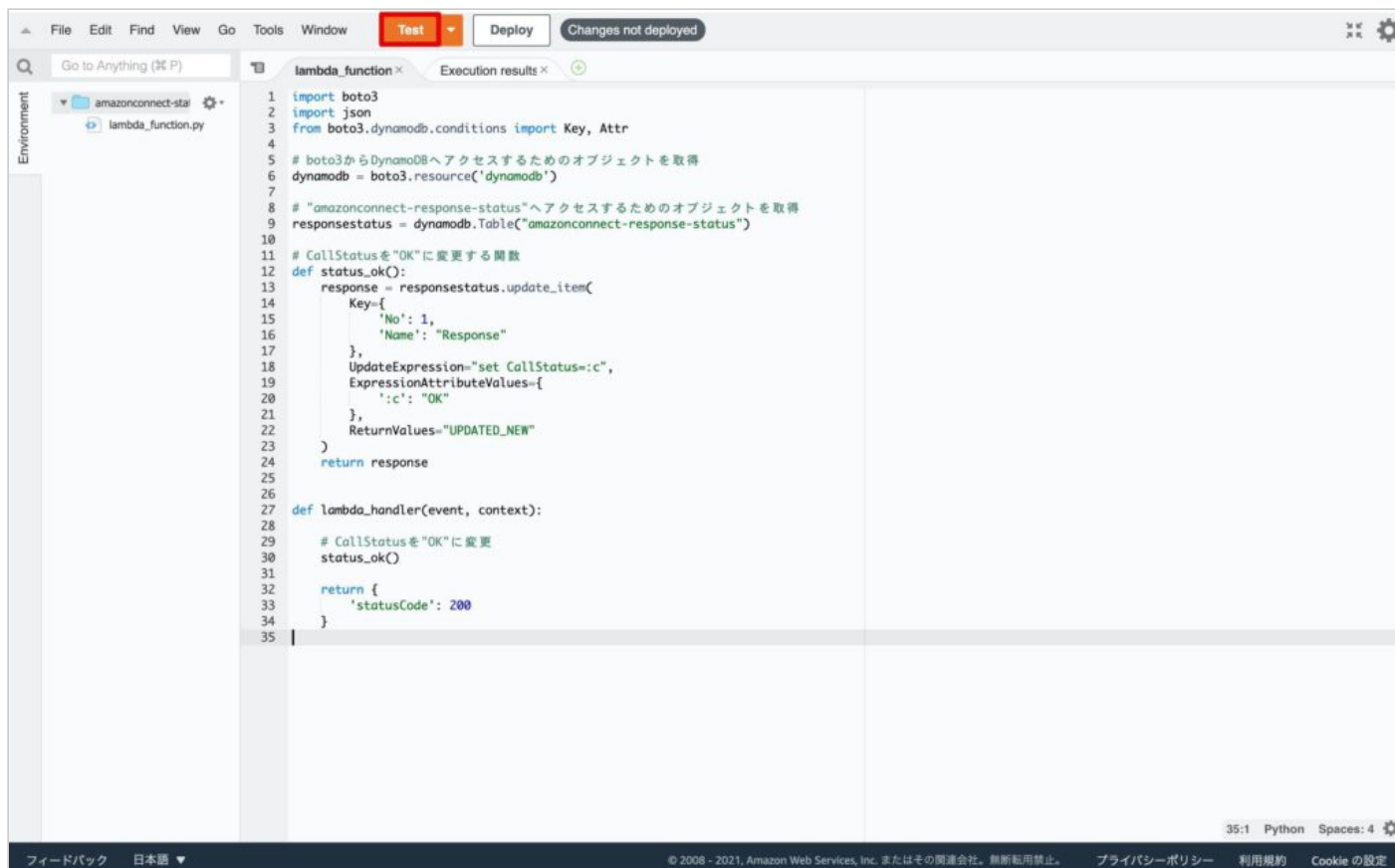
    return {
        'statusCode': 200
    }

```



Lambda_2のテスト

「Test」をクリックします。



```
1 import boto3
2 import json
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-response-status"へアクセスするためのオブジェクトを取得
9 responsestatus = dynamodb.Table("amazonconnect-response-status")
10
11 # CallStatusを"OK"に変更する関数
12 def status_ok():
13     response = responsestatus.update_item(
14         Key={
15             'No': 1,
16             'Name': "Response"
17         },
18         UpdateExpression="set CallStatus=:c",
19         ExpressionAttributeValues={
20             ':c': "OK"
21         },
22         ReturnValues="UPDATED_NEW"
23     )
24     return response
25
26
27 def lambda_handler(event, context):
28
29     # CallStatusを"OK"に変更
30     status_ok()
31
32     return {
33         'statusCode': 200
34     }
35
```

テストイベントの設定画面で下記の通り入力し、「作成」をクリックします。

- ・新しいテストイベントの作成を選択
- ・イベント名に任意の名前を入力
- ・引数は不要のため“{}”を入力

テストイベントの設定

関数を持つことができるテストイベントは最大 10 個です。イベントは保持されているため、別のコンピュータまたはウェブブラウザに切り替えて、同じイベントで関数をテストできます。

☒ 新しいテストイベントの作成

☐ 保存されたテストイベントの編集

イベントテンプレート

hello-world

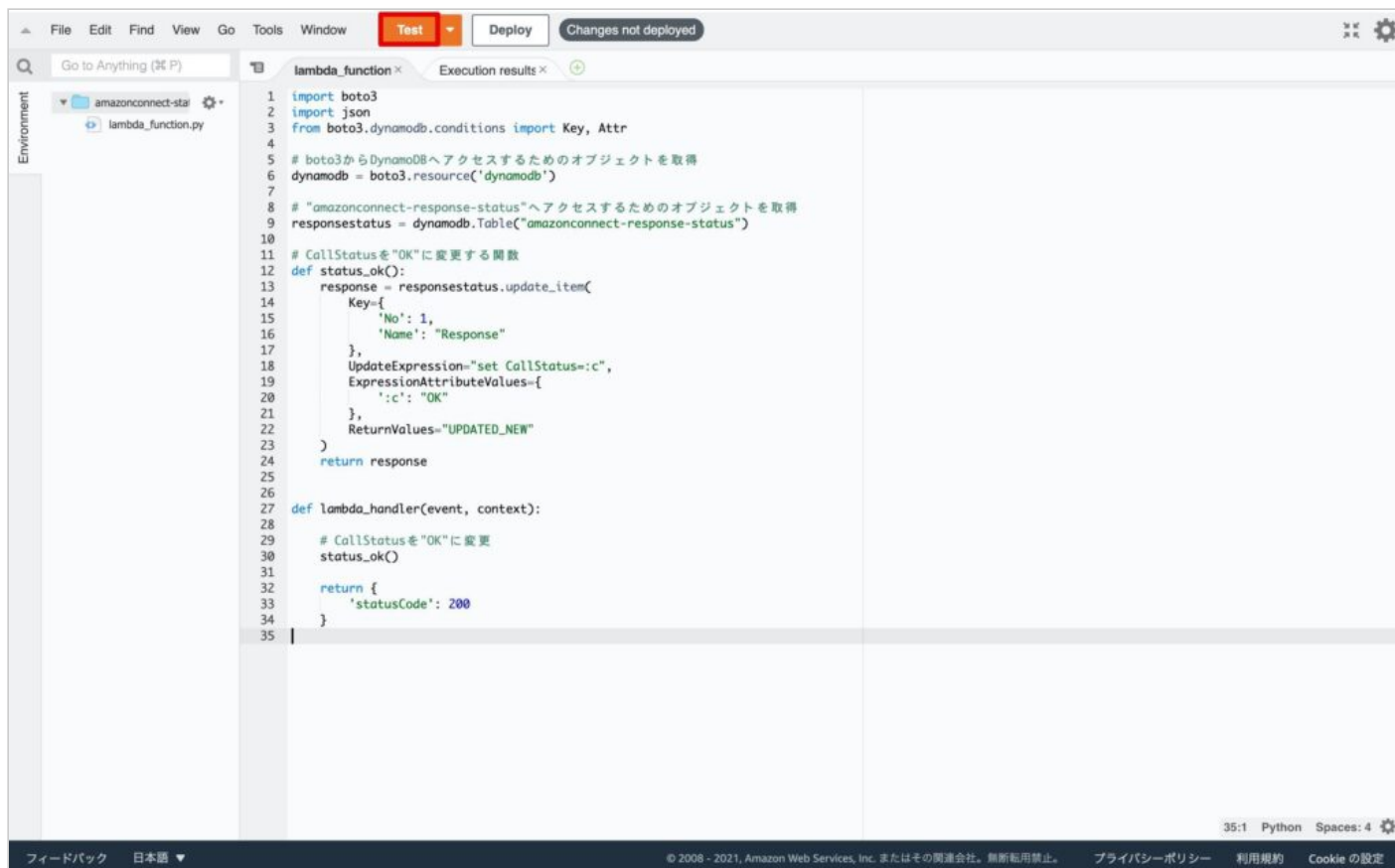
イベント名

Test001

1	{ }
---	-----

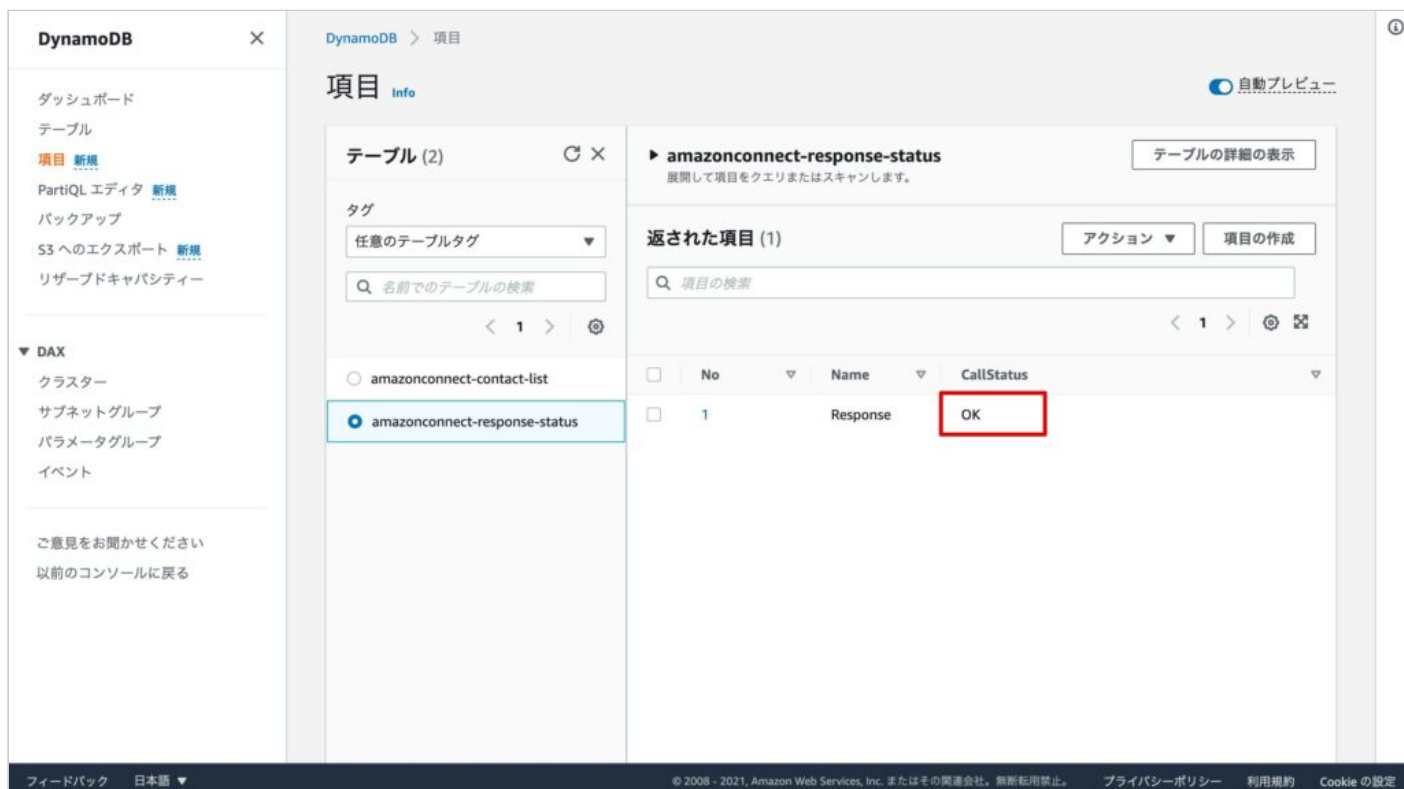
キャンセル JSON 形式 作成

もう一度、「Test」をクリックします。



```
1 import boto3
2 import json
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-response-status"へアクセスするためのオブジェクトを取得
9 responsestatus = dynamodb.Table("amazonconnect-response-status")
10
11 # CallStatusを"OK"に変更する関数
12 def status_ok():
13     response = responsestatus.update_item(
14         Key={
15             'No': 1,
16             'Name': "Response"
17         },
18         UpdateExpression="set CallStatus=:c",
19         ExpressionAttributeValues={
20             ':c': "OK"
21         },
22         ReturnValues="UPDATED_NEW"
23     )
24     return response
25
26
27 def lambda_handler(event, context):
28
29     # CallStatusを"OK"に変更
30     status_ok()
31
32     return {
33         'statusCode': 200
34     }
35
```

DynamoDB_2のCallStatusが「OK」となれば成功です。



No	Name	CallStatus
1	Response	OK

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⑤〉）の説明は完了です！