

詳細手順 2（画像変換関数設定）

本資料では、2種類あるLambda関数のうちの後半、

2. convertRawToJPG.py

保存されたRAW画像をS3から取得して、RAW→JPEGフォーマット変換を実施、
得られたJPEG画像をS3に保存、参照用URLをSNS
(Amazon Simple Notification Service)を用いてユーザにメール連絡する機能
について説明します。

利用するサービスは

S3 (ReadWrite)、Lambda (変換と通知)、SNS (通知) です

まずSNSでトピックスを設定しPUSHしたいメンバを追加します。

AWSマネージメントコンソールでSNSと入力して検索を実行するとSNS画面に遷移します。トピック名として、uploadNotifyを入力し、「次のステップ」メニューを押下します。



トピックの作成画面に切り替わります。トピック名を確認の上、画面下の「トピックの作成」ボタンを押下します



Amazon SNS > トピック > トピックの作成

トピックの作成

詳細

名前
uploadNotify
最大文字数は 256 文字です。英数字、ハイフン(-)、およびアンダースコア(_)を含めることができます。

表示名 - オプション
SMS のサブスクリプションでこのトピックを使用するには、表示名を入力します。SMS メッセージには最初の 10 文字のみが表示されます。[情報](#)
My Topic
ハイフン(-)とアンダースコア(_)を含む最大 100 文字。

▶ **暗号化 - オプション**
Amazon SNS はデフォルトで、転送中に暗号化を行います。サーバー側の暗号化を有効にすると、保存中に行った暗号化をトピックに追加します。

▶ **アクセスポリシー - オプション**
このポリシーでは、トピックにアクセスできるユーザーを定義します。デフォルトでは、トピックの所有者のみがトピックに発行またはサブスクライブできます。[情報](#)

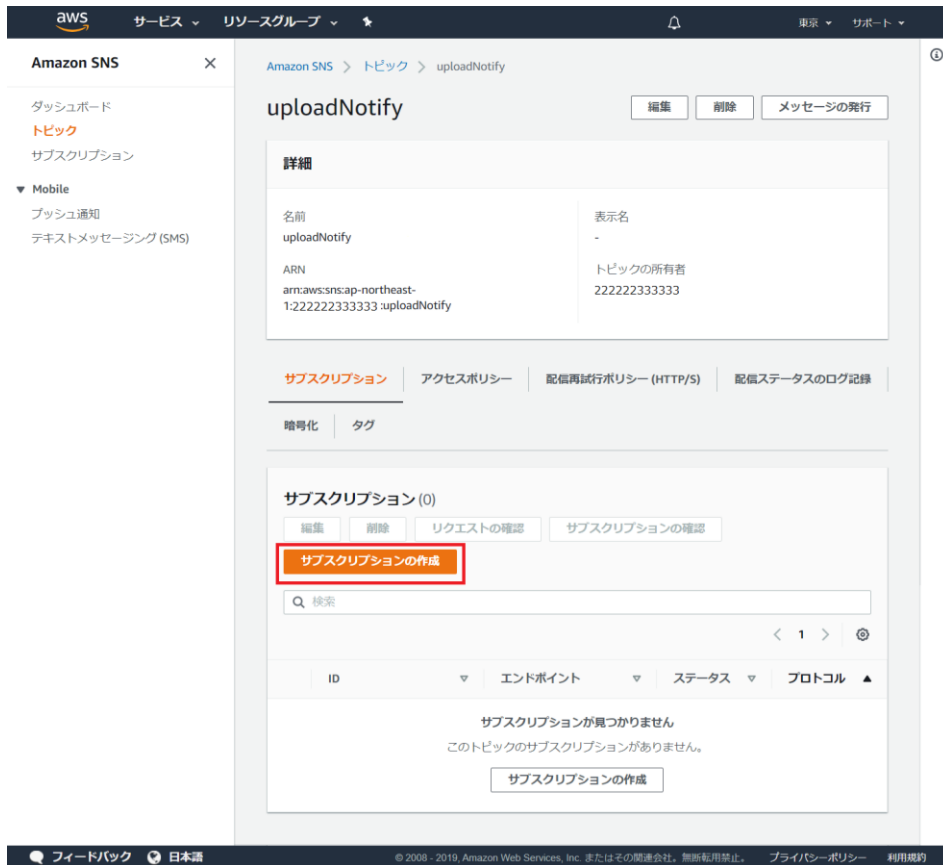
▶ **配信再試行ポリシー (HTTP/S) - オプション**
このポリシーは、Amazon SNS が失敗した HTTP/S エンドポイントへの配信を再試行する方法を定義します。デフォルト設定を変更するには、このセクションを開きます。[情報](#)

▶ **配信ステータスのログ記録 - オプション**
これらの設定により、CloudWatch Logs へのメッセージ配信ステータスのログ記録が設定されます。[情報](#)

▶ **タグ - オプション**
タグは、Amazon SNS トピックに割り当てることができるメタデータラベルです。各タグはキーとオプションの値で構成されていて、タグを使用してトピックの検索やフィルタリングをしたり、コストを追跡したりできます。[詳細はこちら](#)

キャンセル **トピックの作成**

作成したトピック (uploadNotify) の画面が表示されます。画面内の、「サブスクリプションの作成」ボタンを押下します。



Amazon SNS > トピック > uploadNotify

uploadNotify

[編集](#) [削除](#) [メッセージの発行](#)

詳細

名前
uploadNotify
表示名
-
ARN
arn:aws:sns:ap-northeast-1:222222333333:uploadNotify
トピックの所有者
222222333333

サブスクリプション | [アクセスポリシー](#) | [配信再試行ポリシー \(HTTP/S\)](#) | [配信ステータスのログ記録](#)

暗号化 | **タグ**

サブスクリプション (0)

[編集](#) [削除](#) [リクエストの確認](#) [サブスクリプションの確認](#)

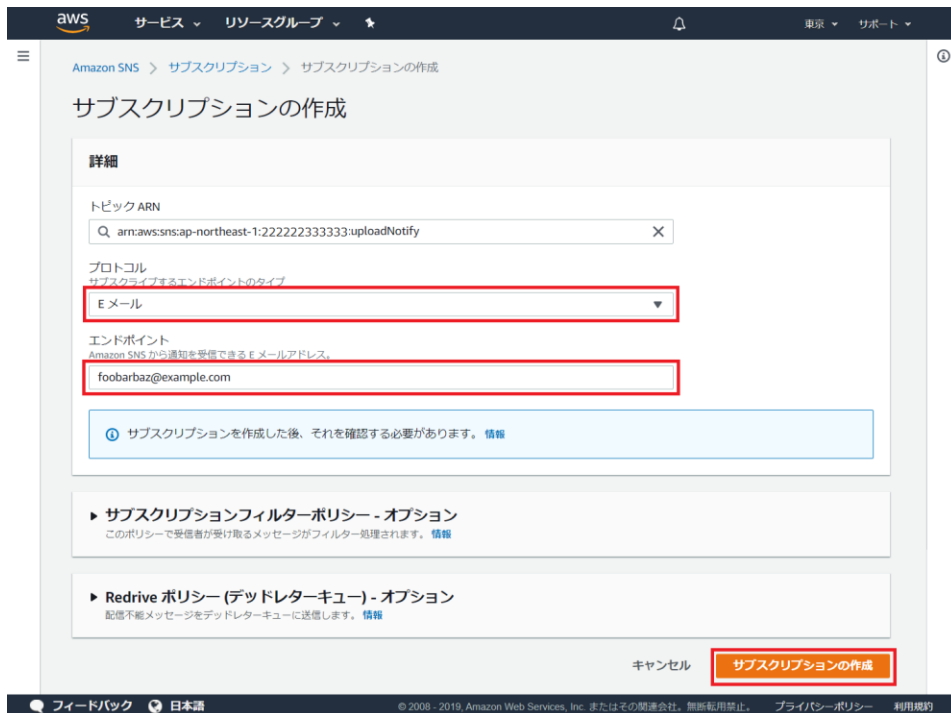
サブスクリプションの作成

検索

ID	エンドポイント	ステータス	プロトコル
サブスクリプションが見つかりません このトピックのサブスクリプションがありません。			

[サブスクリプションの作成](#)

サブスクリプション画面が表示されますので、プロトコル : Eメール、エンドポイントとして、配信したいメンバのメールアドレスを入力します。「サブスクリプションの作成」ボタンを押下して登録します。



Amazon SNS > サブスクリプション > サブスクリプションの作成

サブスクリプションの作成

詳細

トピック ARN
arn:aws:sns:ap-northeast-1:222222333333:uploadNotify

プロトコル
サブスクライブするエンドポイントのタイプ
Eメール

エンドポイント
Amazon SNS から通知を受信できる E メールアドレス。
foobarbaz@example.com

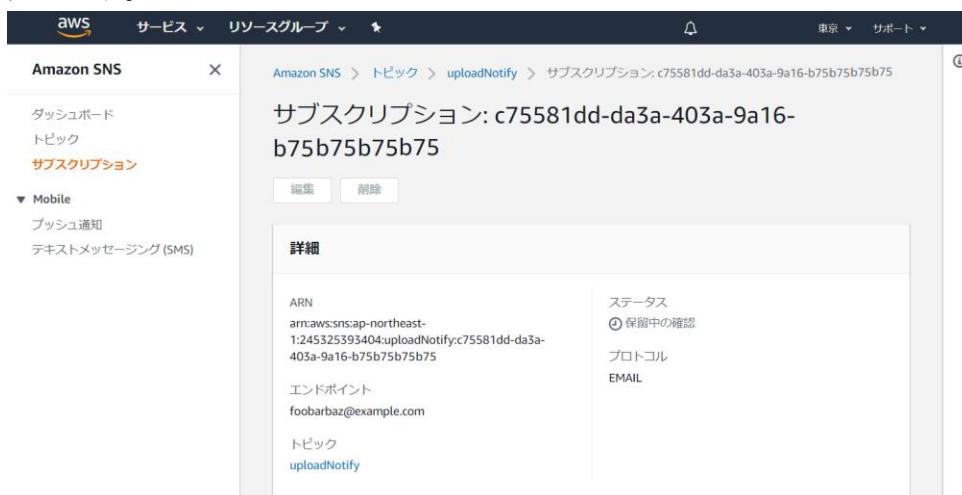
① サブスクリプションを作成した後、それを確認する必要があります。 [情報](#)

▶ サブスクリプションフィルターポリシー - オプション
このポリシーで受信者が受信するメッセージがフィルター処理されます。 [情報](#)

▶ Redrive ポリシー (デッドレターキュー) - オプション
配信不能メッセージをデッドレターキューに送信します。 [情報](#)

キャンセル **サブスクリプションの作成**

サブスクリプション管理画面に遷移します。トピック名：uploadNotifyに参加するかどうかの確認がメールで配信されますので、受信した人は、メール内の承認URLをクリックして承認します。



Amazon SNS > トピック > uploadNotify > サブスクリプション: c75581dd-da3a-403a-9a16-b75b75b75b75

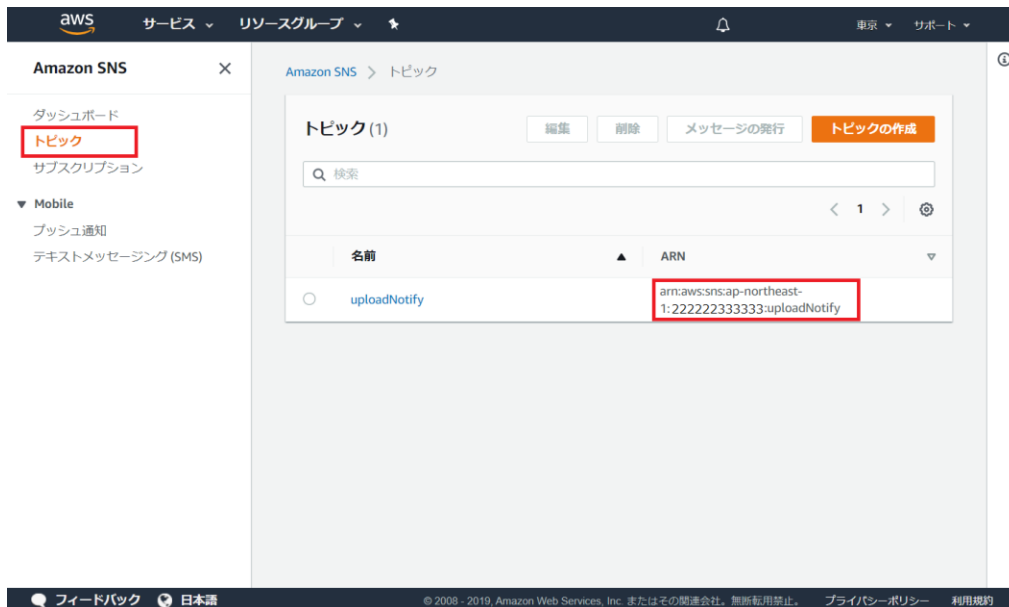
サブスクリプション: c75581dd-da3a-403a-9a16-b75b75b75b75

編集 削除

詳細

ARN arn:aws:sns:ap-northeast-1:245325393404:uploadNotify:c75581dd-da3a-403a-9a16-b75b75b75b75	ステータス ① 保留中の確認
エンドポイント foobarbaz@example.com	プロトコル EMAIL
トピック uploadNotify	

SNS管理画面左側のトピックをクリックすると、作成したトピックが表示されます。画像変換用LambdaからSNSを呼び出す際にARNで指定しますので、表示されているARNをメモします。



SNSの設定は終わりです。次に画像変換用Lambdaを作成します。

AWSマネジメントコンソールで、Lambdaと入力して検索
「関数の作成」ボタンを押下、関数の作成画面が表示されます。



アップロード用Lambda関数と同様に、一から作成、関数名：convertAndNotify、ランタイム：Python3.7を指定、「関数の作成」ボタンを押下します。
lambda関数 convertAndNotifyの画面が表示されますので、ソースのCopy&Pasteした後、「保存」ボタンを押下します。ソースでは、以下となっていますので、

SNS ARN

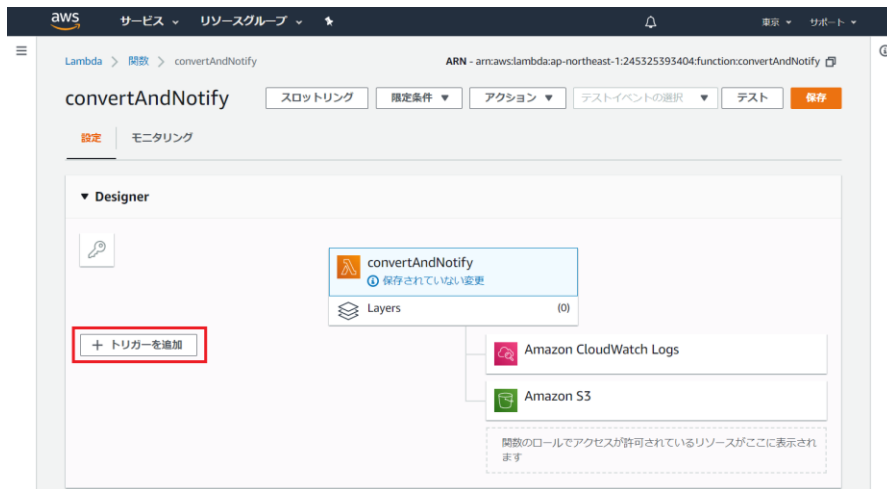
SNS_ARN='arn:aws:sns:ap-northeast-1:xxxxxxx:uploadNotify'

先ほど作成したSNSのARN(Amazon Resource N)の値を変数SNS_ARNに設定します。

The screenshot shows the AWS Lambda console for the function 'convertAndNotify'. The 'Designer' tab is selected, showing a single function block. The '関数コード' (Function code) tab is also visible, showing the Python code for the lambda_handler function. The code includes imports for json, boto3, copy, and datetime, and defines the SNS_ARN variable with the value 'arn:aws:sns:ap-northeast-1:333334444444:uploadNotify'. The code also defines a position variable and a list of positions.

```
1 import json
2 import boto3
3 import copy
4 import datetime
5
6 # PIL from Layer
7 from PIL import Image, ImageDraw
8
9 # SNS ARN
10 SNS_ARN = 'arn:aws:sns:ap-northeast-1:333334444444:uploadNotify'
11
12
13 WIDTH=640
14 HEIGHT=480
15
16 # define for position
17 # [[SQ]] [[PH,PHLR]] [[SQ]]
18 # [[LR,PHLR]] [[IT]] [[LR,PHLR]]
19 # [[SQ]] [[PH,PHLR]] [[SQ]]
20 #
21 #
22 #
23 #
24
25 IT=0
26 PH=1
27 LR=2
28 PHLR=3
29 SQ=4
30
31 COLOR_R=0
32 COLOR_G=1
33 COLOR_B=2
34
35 # notation of position is (x,y)
36 POS_X=0
```

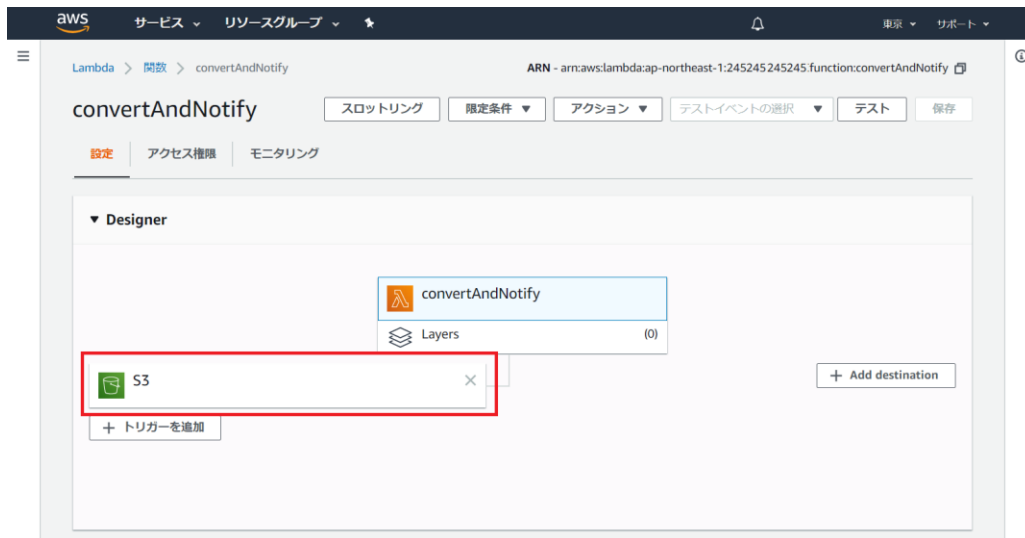
convertAndNotify関数は、S3にRAW画像が配置された時に呼び出される仕様になっています。この仕様を実現するためconvertAndNotify関数が実行されるトリガーを設定します。Lambda関数 (convertAndNotify) のDesigner表示欄の「+トリガーを追加」ボタンを押下します



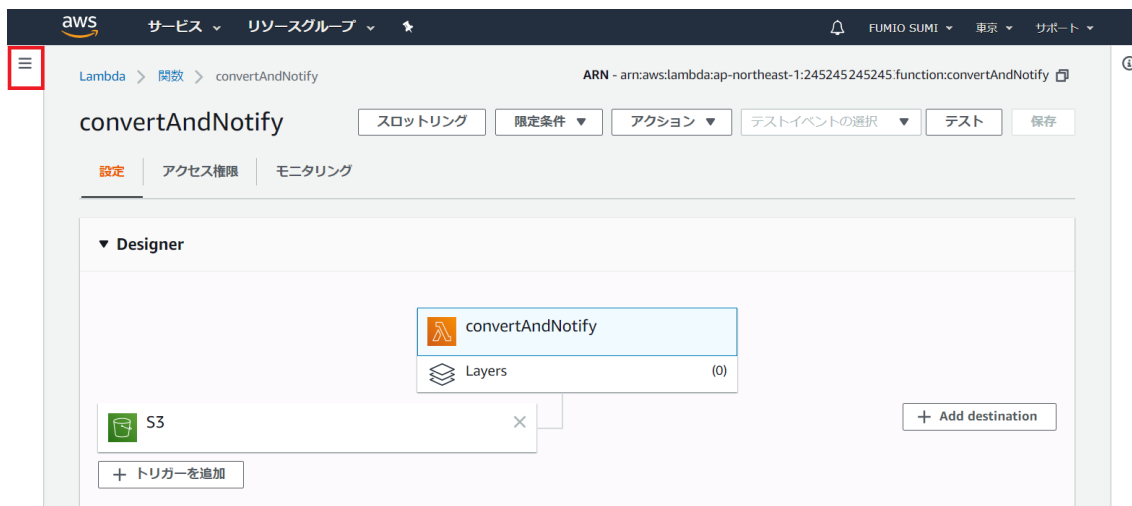
「トリガーを追加」画面に遷移します。トリガーの設定プルダウンメニューより「S3」を選択します。バケット名は、uploadPhotoの手順で作成したバケット名(wificambucket)を指定、イベントタイプは「PUT」、サフィックスはRAW画像なので、.rawを指定します。入力が終わると、「作成」ボタンを押下します。



上記作業により、以下のようにトリガー：S3と追加されます。



変換プログラムでは、PILライブラリが必要ですが、PILライブラリはLambda実行環境では提供されていません。そこで、ローカル環境でPILライブラリを作成した後、AWSにアップロードします（PILライブラリ作成手順は別途をご参照ください）。PILライブラリは複数の関数から利用可能にするため、Lambda実行環境のLayersに登録することになります。Lambda管理画面左上の三本線（ハンバーガーメニューと呼ばれます）をクリックすると隠れているメニューが表示されます。この中からLayersをクリックします。



「レイヤーの作成」ボタンを押下します。

「レイヤー設定」画面が表示されます。以下の内容で設定します。

名前：PIL_for_Python3_7（例）

説明：PILモジュール（例）

互換性のあるランタイム： Python3.7

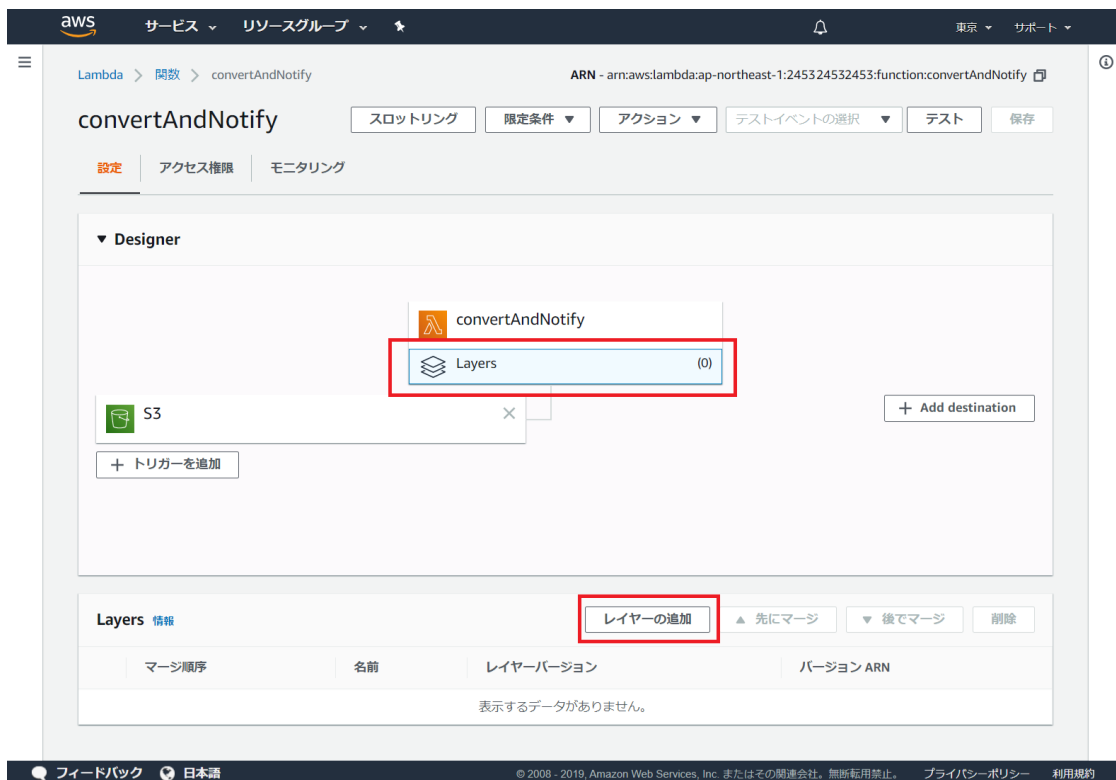
(○) .zipファイルをアップロード を選んだ状態で、「アップロード」ボタンを押下してローカルで作成したPILライブラリ（ファイル名：pillow_pkg_190923.zip）をアップロードします。最後に、右下の「作成」ボタンを押下します。

The screenshot shows the 'AWS Lambda > Layers > レイヤーの作成' page. The 'レイヤー設定' (Layer Settings) section includes:
- Name: PIL_for_Python_3_7
- Description - オプション: PILモジュール
- Selection: zip ファイルをアップロード (selected), Amazon S3 からのファイルのアップロード
- Upload button: アップロード pillow_pkg_190923.zip (2.3 MB)
- Note: 10 MB より大きいファイルの場合は、Amazon S3 を使用したアップロードを検討してください。
- Runtime: 互換性のあるランタイム・オプション (Python 3.7 selected)
- License: ライセンス - オプション
At the bottom right, the '作成' (Create) button is highlighted with a red box.

登録が完了すると以下のようにLayer画面に遷移します。

The screenshot shows the 'AWS Lambda > Layers > PIL_for_Python_3_7' page. The 'PIL_for_Python_3_7' header includes buttons for '削除' (Delete), 'ダウンロード' (Download), and 'バージョンの作成' (Create Version). A green notification bar states: 'レイヤー PIL_for_Python_3_7 のバージョン 1 が正常に作成されました。' (Version 1 of layer PIL_for_Python_3_7 was created successfully).
The 'バージョンの詳細' (Version Details) section shows:
- バージョン: 1
- 説明: PILモジュール
- 作成済み: 11 秒前
- ライセンス: (blank)
The '互換性のあるランタイム' (Compatible Runtime) section shows: python3.7
The 'すべてのバージョン' (All Versions) table shows:
| バージョン | バージョン ARN | 説明 |
| 1 | arn:aws:lambda:ap-northeast-1:245325393404:layer:PIL_for_Python_3_7:1 | PILモジュール |

PILモジュールがLayerとして登録できたので、convertAndNotify関数の管理画面に戻ります。convertAndNotify管理画面でLayersのメニューをクリックするとLayers情報欄が表示されます。「レイヤーの追加」ボタンを押下します。



以下のようにレイヤー追加画面が表示されますので、先ほどLayerに登録したPILモジュールを指定します。

(○) ランタイムと互換性のあるレイヤーのリストから選択

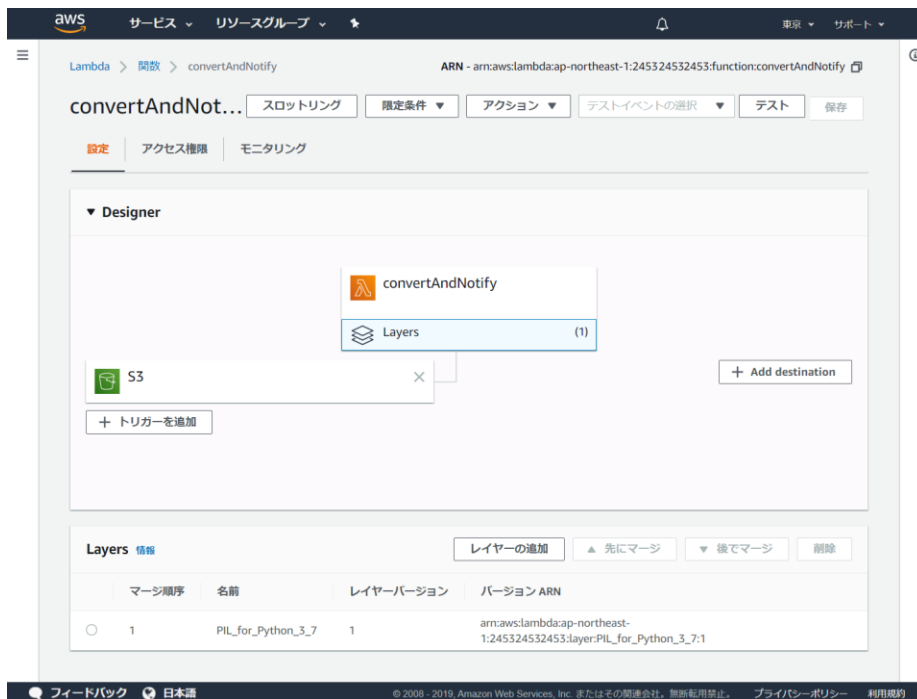
名前 : PIL_for_Python3_7 (登録時の名称を指定)

バージョン : 1

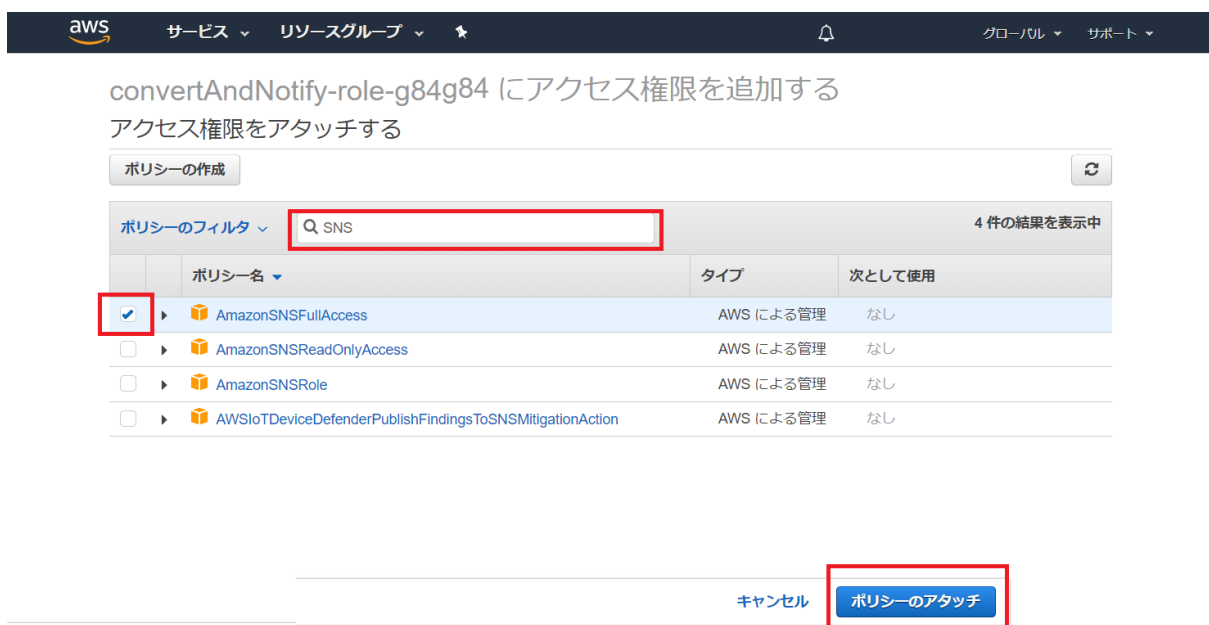
上記を指定して、右下の「追加」ボタンを押下します。



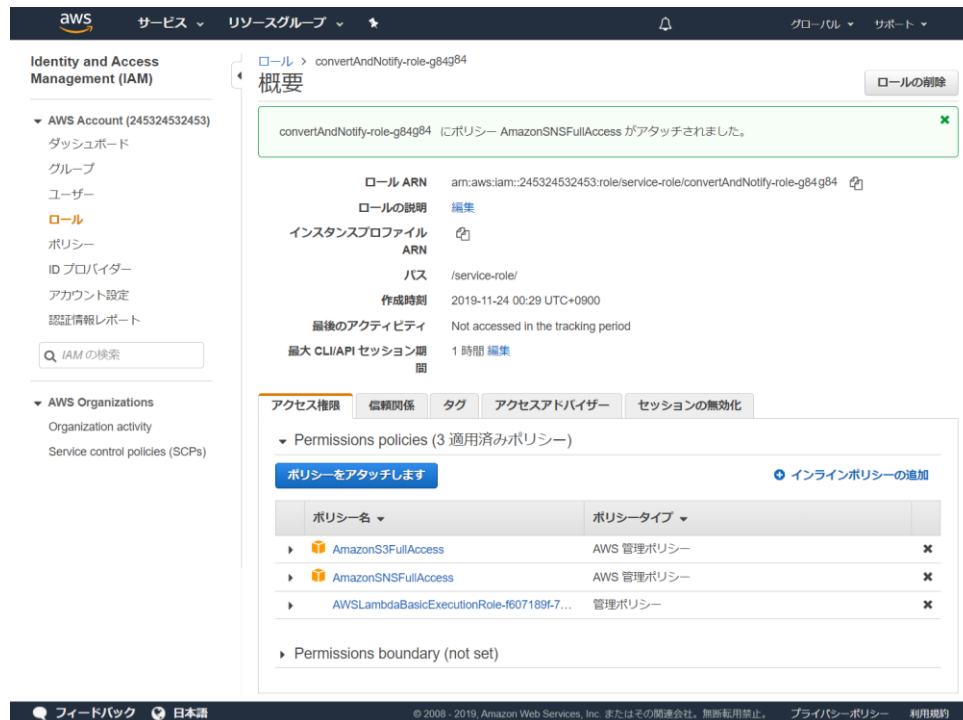
登録が完了すると、convertAndNotify関数の管理画面に戻ります。正しく登録されていると以下の画面になります。



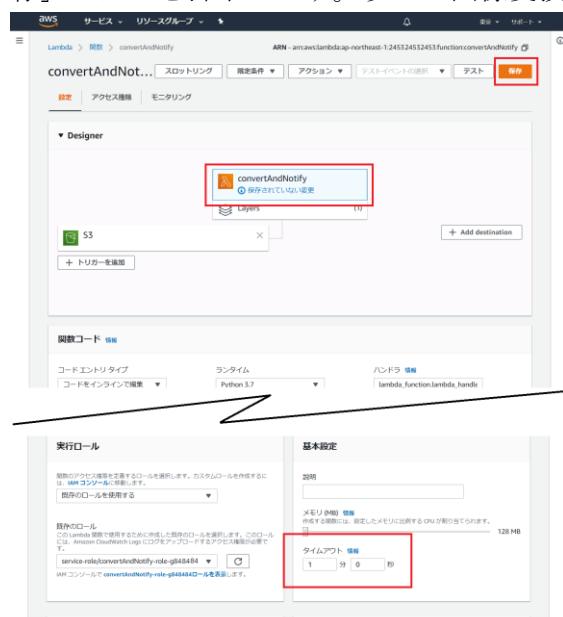
現在の権限では、S3バケットのアクセスが許可されていませんので、ロールにS3バケットへのアクセス権限を付与します。また、SNSを呼び出すための権限も必要となります。S3バケットへのアクセス権限の付与は、uploadPhotoで行った手順と同じです。前章を参照してください。続いて、SNSを呼び出すための権限を付与します。「アクセス権限をアタッチする」画面よりSNSと入力して検索します。AmazonSNSFullAccessのポリシーが表示されますので、これをクリックして、「ポリシーのアタッチ」を押下します。



正しく設定されると、以下のように3種類のポリシーが付与されている画面となります。



以上の操作により、S3にRAW画像が保存されると、convertAndNotify関数が呼び出され、画像変換、ならびに、SNSによるメール配信の設定が完了しました。ただ、convertAndNotify関数は、Numpy等の数値計算モジュールを使っておらず1画像の変換に30秒程度かかります。このままでは実行時間制限に引っかかるので、タイムアウト時間を延ばします。convertAndNotifyの管理画面で、convertAndNotifyのメニューをクリックして基本設定の欄を表示させます。タイムアウト設定用フォームが表示されますので、1分に設定し、画面右上の「保存」ボタンを押下します。以上で画像変換関数の設定は終わりです。



再度以下のように、画像アップロード用APIを呼び出し、メールが配信されることを確認します。

<https://7rcx6s3am4.execute-api.ap-northeast-1.amazonaws.com/test/uploadphoto>

```
curl -X POST -H 'Content-Type: application/octet-stream' -H 'API-Key: setYourAPIKey' --data-binary "@191117_200040.raw" https://7rcx6s3am4.execute-api.ap-northeast-1.amazonaws.com/test/uploadphoto
```

```
$ curl -X POST -H 'Content-Type: application/octet-stream' -H 'API-Key: setYourAPIKey' --data-binary "@191117_200040.raw" https://7rcx6s3am4.execute-api.ap-northeast-1.amazonaws.com/test/uploadphoto
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
100	300k	100	16	100	300k	4	76800
				0:00:04	0:00:04	--:--:--	64459

"upload is done"

もし正常に動作しない場合は、設定誤りにより動作エラーが発生していると考えられます。Lambdaの実行時、CloudWatchにエラー出力されるように設定されていますのでエラーを確認します。AWSのマネジメントコンソールでCloudWatchと入力して検索します。

CloudWatchの管理画面に切り替わりますので、画面左側でロググループメニューを選択します。ロググループ一覧が出ますので、/aws/lambda/convertAndNotify を選択して、convertAndNotifyでエラーが出ていないか確認します。下記画面は設定誤りによりエラーが発生している例です。下三角マークをクリックすると詳細が表示されますので、ソース行のどこでエラーが発生しているのか分かりますので原因を絞り込みます。



The screenshot shows the AWS CloudWatch console. The left sidebar has the 'Logs' menu item highlighted. The main area shows the log group '/aws/lambda/convertAndNotify'. The log stream '2019/12/01[\$LATEST]3850f76c3850f76d3850f76c3850f76c' is selected. The log events show a sequence of messages: 'START RequestId: 4ba1d7cc-72d8-468b-86b2-eba1634af761 Version: \$LATEST', 'start to convert', and an error message: 'InvalidParameterException: An error occurred (InvalidParameter) when calling the Publish operation: Invalid parameter.' The error message is highlighted in red. The log also shows the end of the function execution: 'END RequestId: 4ba1d7cc-72d8-468b-86b2-eba1634af761' and 'REPORT RequestId: 4ba1d7cc-72d8-468b-86b2-eba1634af761 Duration: 46904.56 ms Billed Duration: 47000 ms Memory Used: 96 MB'.

正常に設定が行われていると、SNSでサブクリプションしたメールアドレスにJPG画像参照用URLを含むメールが配信されます。メール内のURLをクリックするとスマホアプリの場合、ブラウザに切り替わりJPG画像が表示されます。

