

# Disposable Email Domains Data Feedを AWS S3にインポートするには

2023年10月3日

本文書では、AWS Lambdaを活用して、WhoisXML APIが提供するDisposable Email Domain Data FeedをAWS S3バケットにダウンロードする基本的な方法を紹介します。AWS Lambdaは、サーバーをプロビジョニングしたり管理したりすることなくコードを書いて実行できる、サーバーレスのコンピューティングサービスです。AWS S3は、ファイルを保存および取得するためのオブジェクトストレージサービスです。ここでは、AWS LambdaとAWS S3バケットの両方を構成するプロセスを説明します。

## 以下は本文書の対象外です：

- Lambda関数のスケジューリング
- ETLパイプライン
- PythonのRequestsモジュールのインポート

## 前提条件

事前に以下を用意する必要があります：

- AWSアカウント
- AWSサービス、特にAWS LambdaとS3に関する基礎～中級程度の知識



- Lambda関数で使われるPythonの知識
- [WHOIS API Disposable Email Domains](#)データフィールドへのアクセス。APIキーが必要です。詳細につきましては、[sales@whoisxmlapi.com](mailto:sales@whoisxmlapi.com) にお問い合わせください。このデータフィールドの仕様は、[こちら](#)でご確認いただけます。

## ステップ1：AWS S3バケットの作成

最初のステップは、Disposable Email Domains ファイルを書き込むS3バケットの作成です。

- AWS Management Consoleで、S3サービスに移動します。
- 「Create Bucket」をクリックします。
- バケットにユニークな名前をつけ、適切な地域を選択します。



## General configuration

Bucket name

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming.](#)

AWS Region

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

## Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

**ACLs disabled (recommended)**

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

**ACLs enabled**

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

- ここでは、デフォルト設定のまま「Create Bucket」をクリックします。

## ステップ2：IAMロールの作成

AWS Lambdaでは、S3バケットの読み書きに必要な権限を持つIAMロールが必須となります。

以下の手順でIAMロールを作成してください：

- AWSマネジメントコンソールでIAMサービスに移動します。



## Select trusted entity Info

### Trusted entity type

**AWS service**

Allow AWS services like EC2, Lambda, or others to perform actions in this account.

**AWS account**

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

**Web identity**

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

**SAML 2.0 federation**

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

**Custom trust policy**

Create a custom trust policy to enable others to perform actions in this account.

- 「Roles」をクリックし、次に「Create Role」をクリックします。
- このロールのサービスとして「Lambda」を選択し、「Next: Permissions」をクリックします。

## Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Lambda

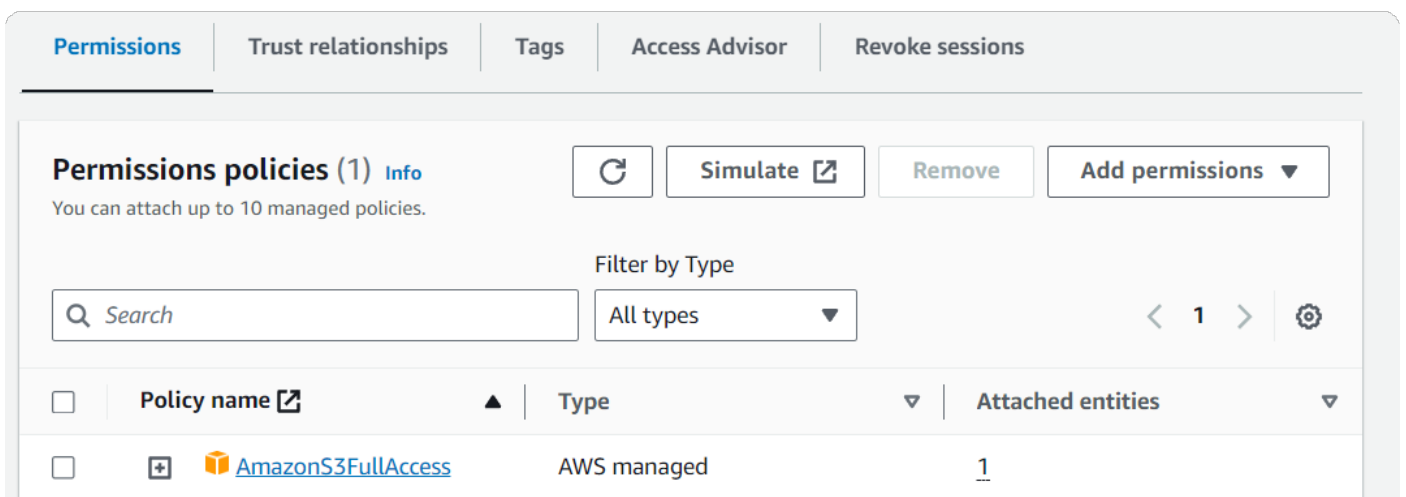
Choose a use case for the specified service.

Use case


Lambda

Allows Lambda functions to call AWS services on your behalf.

- 検索バーに「S3」と入力し、「AWSS3FullAccess」、そして「Next: Tags」を選択します。



The screenshot shows the AWS IAM console interface for managing permissions policies. The 'Permissions' tab is selected. The main heading is 'Permissions policies (1) Info', with a sub-note: 'You can attach up to 10 managed policies.' Action buttons include 'Refresh', 'Simulate', 'Remove', and 'Add permissions'. A search bar contains 'Search' and a 'Filter by Type' dropdown is set to 'All types'. A table lists the attached policies:

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	 AmazonS3FullAccess	AWS managed	1



- タグは任意です。次に「Next: Review」をクリックします。
- あなたのロール、名前、簡単な説明を入力し、「Create Role」をクリックします。

## ステップ3：Lambda関数の作成

Lambda関数の作成は楽しく、簡単です。その方法は以下の通りです：

- AWSマネジメントコンソールでLambdaサービスに移動します。
- 「Create Function」をクリックします。
- 関数にわかりやすい名前を付け、ランタイムとしてPythonを選択します。そして、上記のステップ2で作成したIAMロールを選択します。
- 「Create function」をクリックします。

注：

Execution roleの設定：

**Execution role**

Edit

View role document

Role name

disposable-email-role [↗](#)**Resource summary**

To view the resources and actions that your function has permission to access, choose a service.



Amazon S3

1 action, 1 resource

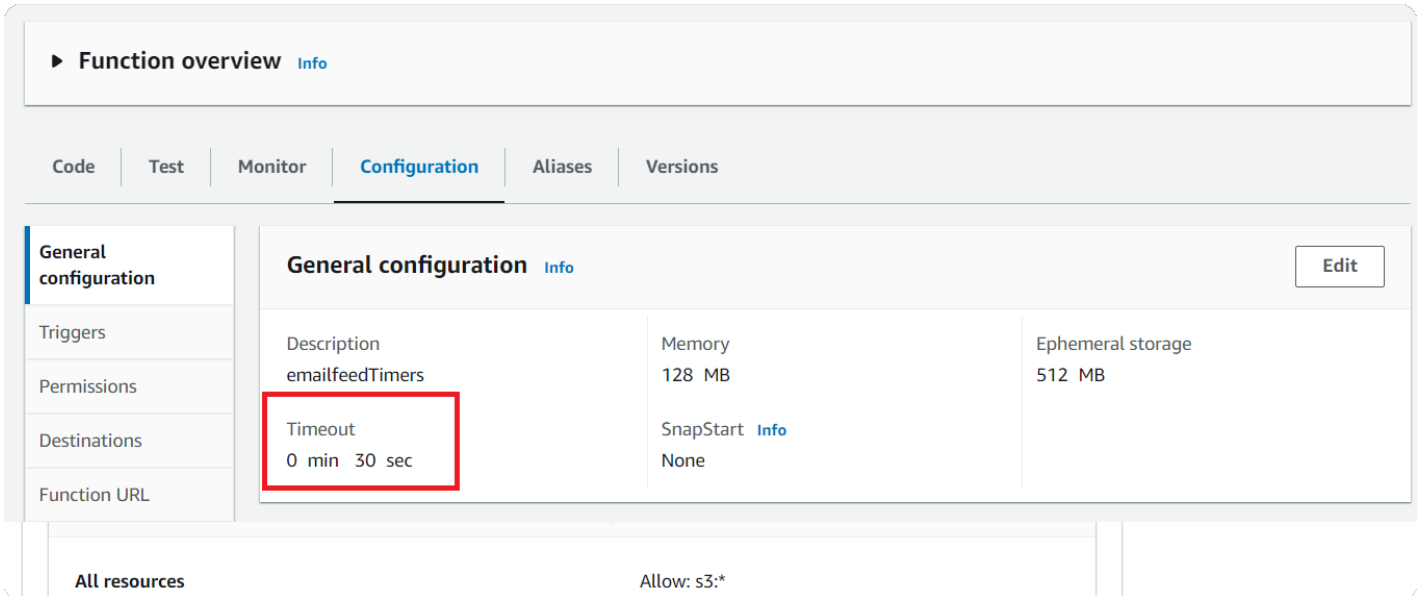


By action

By resource

Resource	Actions
All resources	Allow: s3:*

Lambda関数のタイムアウト値を設定します。今回は30秒に設定しています。



Function overview [Info](#)

Code | Test | Monitor | **Configuration** | Aliases | Versions

General configuration [Info](#) Edit

Description emailfeedTimers	Memory 128 MB	Ephemeral storage 512 MB
<b>Timeout</b> 0 min 30 sec	SnapStart <a href="#">Info</a> None	

All resources Allow: s3:\*

## ステップ4： Disposable Email Domain リストをS3にインポートするLambda関数を記述

この例ではpython requestsモジュールを使用していますが、Boto3の一部ではなくなったため、インポートする必要があるかもしれません。この方法に関するAWSのドキュメントは曖昧ですが、インターネット上で様々な技術記事を見つけることができます。

### コードの例：

以下のPythonコード（GitHubでも入手できます）は、lambda\_handlerのエントリーポイントです：

```
import os
import boto3
import sys
from datetime import datetime, timedelta
sys.path.append('python') #added for requests module
```





```
import requests
from requests.auth import HTTPBasicAuth

def lambda_handler(event, context):
    # Calculate yesterday's date in YYYY-MM-DD format
    yesterday = (datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d")

    # Define the URL of the CSV file you want to download
    csv_url = f"https://emailverification.whoisxmlapi.com/datafeeds/Disposa

    apiKey = "YOUR_API_KEY"

    # Define the username and password for basic authentication
    username = apiKey
    password = apiKey

    # Define the S3 bucket and object/key where you want to store the CSV
    "s3://newbucketname/email/disposable/"
    s3_bucket = "newbucketname"
    s3_key = f"email/disposable/disposable-email-domains-{yesterday}.csv"

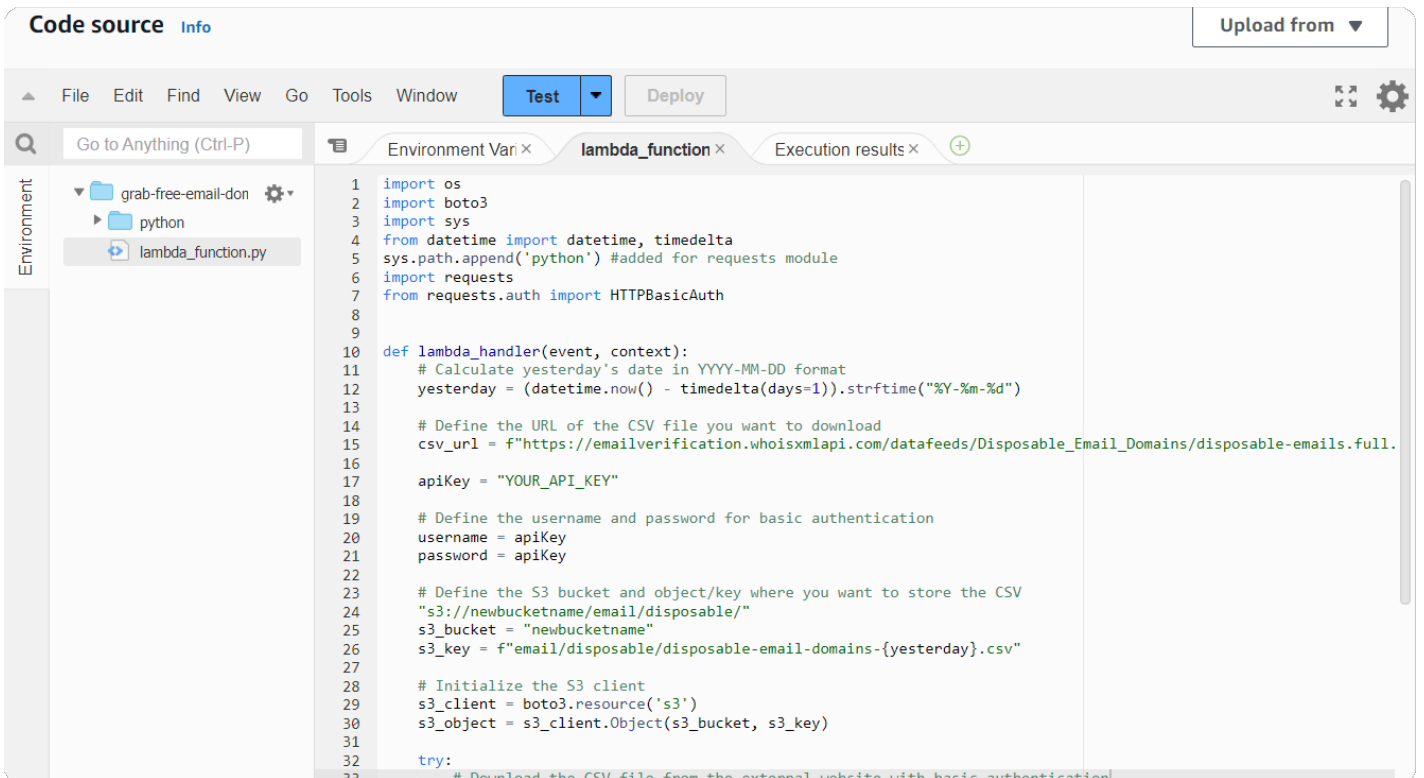
    # Initialize the S3 client
    s3_client = boto3.resource('s3')
    s3_object = s3_client.Object(s3_bucket, s3_key)

    try:
        # Download the CSV file from the external website with basic authen
        response = requests.get(csv_url, auth=HTTPBasicAuth(username, passw

        if response.status_code == 200:
            # Upload the CSV file to S3
            print(f"Uploading file to ", s3_bucket, s3_key)
            s3_object.put(Body=response.content)
            return {
                'statusCode': 200,
                'body': 'CSV file successfully downloaded and uploaded to S
            }
        else:
            bodyStr = f"Failed to download {csv_url}"
```

```
        return {
            'statusCode': response.status_code,
            'body': bodyStr
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': str(e)
        }
```

完成すると、このようなものができるはずです：



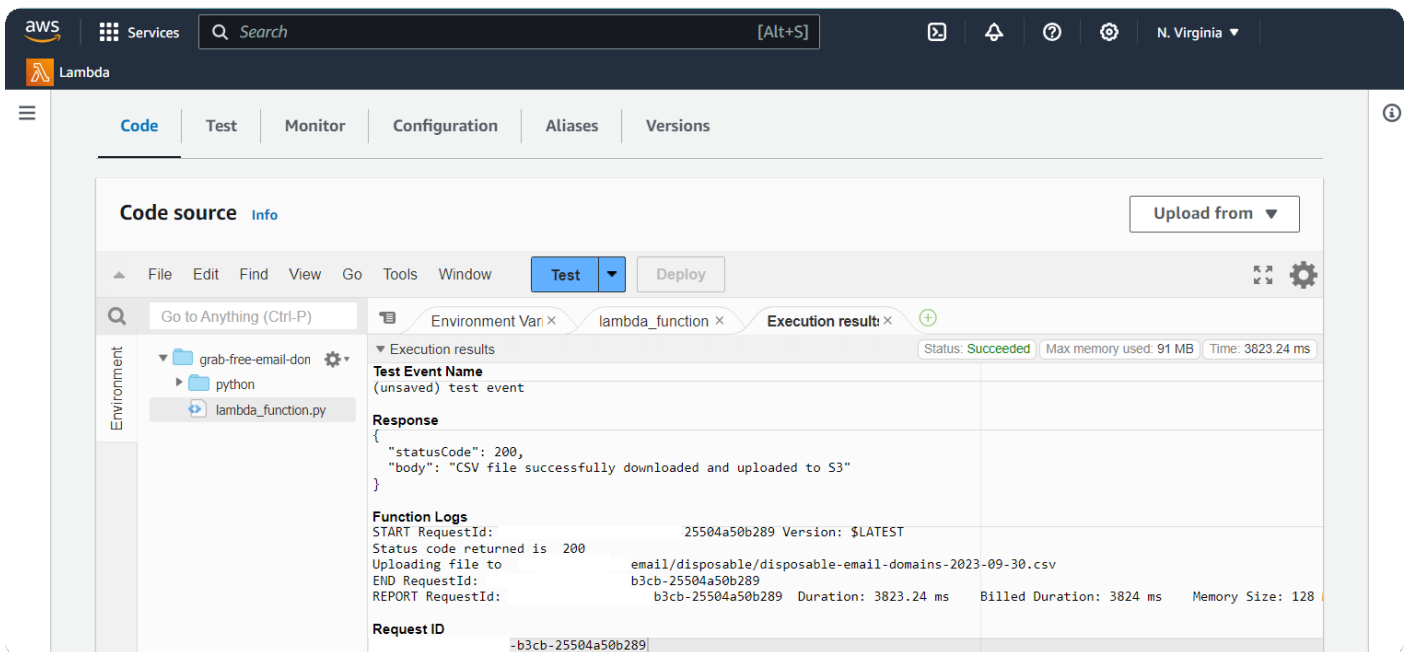
The screenshot shows an IDE window titled "Code source" with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for "Test" and "Deploy". The main editor area displays Python code for a lambda function. The code imports os, boto3, sys, datetime, and requests. It defines a lambda\_handler function that calculates yesterday's date, defines a CSV URL, sets API key, username, and password, and uses boto3 to upload a CSV file to S3. The code is as follows:

```
1 import os
2 import boto3
3 import sys
4 from datetime import datetime, timedelta
5 sys.path.append('python') #added for requests module
6 import requests
7 from requests.auth import HTTPBasicAuth
8
9
10 def lambda_handler(event, context):
11     # Calculate yesterday's date in YYYY-MM-DD format
12     yesterday = (datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d")
13
14     # Define the URL of the CSV file you want to download
15     csv_url = f"https://emailverification.whoisxmlapi.com/datafeeds/Disposable_Email_Domains/disposable-emails.full."
16
17     apiKey = "YOUR_API_KEY"
18
19     # Define the username and password for basic authentication
20     username = apiKey
21     password = apiKey
22
23     # Define the S3 bucket and object/key where you want to store the CSV
24     "s3://newbucketname/email/disposable/"
25     s3_bucket = "newbucketname"
26     s3_key = f"email/disposable/disposable-email-domains-{yesterday}.csv"
27
28     # Initialize the S3 client
29     s3_client = boto3.resource('s3')
30     s3_object = s3_client.Object(s3_bucket, s3_key)
31
32     try:
33         # Download the CSV file from the external website with basic authentication
```

## ステップ5：作成したLambda関数をテストする

最後のステップとして、作成したLambda関数をテストし、a) Disposable Email Domain ファイルを正常に取得できること、b) S3バケットに書き込めることを確認します：

- ページ上部の「Test」をクリックすると、以下のようなものが表示されるはずです。



- 「"requests" module not found」というメッセージを受け取った場合は、python requests ライブラリを正しく設定する必要があります（本文書の範囲外）。

Lambda関数が正しく設定されていれば、関数はファイルを取得し、S3バケットに書き込みます。S3バケットに移動してファイルの存在を確認できます。

disposable/ Copy S3 URI

**Objects** | Properties


**Objects (1)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Refresh Copy S3 URI Copy URL Download Open Delete Actions

Create folder Upload

< 1 > Settings

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 disposable-email-domains-2023-09-30.csv	csv	October 1, 2023, 17:00:05 (UTC-05:00)	2.4 MB	Standard

## まとめ

S3バケットにアクセスできるAWS Lambdaの設定は、クラウドエンジニアにとってはごく一般的なタスクです。このプロセスの後に踏む次のステップは、Athena、Postgres、MySQLデータベースへのインポートなど、このデータで何をするかを決めることです。ETL用のAWS Glueをご存じない方は、そちらもチェックしてみてください。