

# Importing the Disposable Email Domains Data Feed to AWS S3

Posted on October 3, 2023

The intention of this document is to show you the basis of how to download the disposable email domain data feed provided by WhoisXML API to an AWS S3 bucket by leveraging a serverless Lambda function. AWS Lambda functions act as a serverless compute service that allows you to write and execute code without provisioning or managing servers. AWS S3 is an object storage service for storing and retrieving files. This document will guide you through the process of configuring both AWS Lambda and an AWS S3 bucket.

## Out of scope:

- Scheduling a Lambda function
- ETL pipelining
- Importing the python requests module

## Prerequisites

Please ensure you have the following setup:

- AWS Account
- Basic to Intermediate knowledge of AWS services, specifically AWS Lambda and S3

- Some familiarity with Python which will be used in the Lambda function
- Access to the [WHOIS API Disposable Email Domains](#) data feed. You will need an API key with access to the data feed. Please [contact us](#) for more information. For more information on the specifications of the data feed, please visit [here](#).

## Step 1: Create an AWS S3 Bucket

The first step is to create an S3 bucket to write the Disposable Email Domains files to.

- In the AWS Management Console, navigate to the S3 service.
- Click on “Create Bucket”.
- Give the bucket a unique name and select the appropriate region.



## General configuration

Bucket name

myawsbucket

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

AWS Region

US East (N. Virginia) us-east-1

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.

- At this time, leave the default settings and go ahead and click “Create Bucket”.

## Step 2: Create an IAM Role

AWS Lambda will require an IAM role with the permissions necessary to read/write to the S3 bucket. Please follow these steps to create an IAM role:

- Navigate to the IAM Service in the AWS management console.



## 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.

- Click on “Roles” and then followed by “Create Role”.
- Select “Lambda” as the service for this role, and then click “Next: Permissions”.

## Use case

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

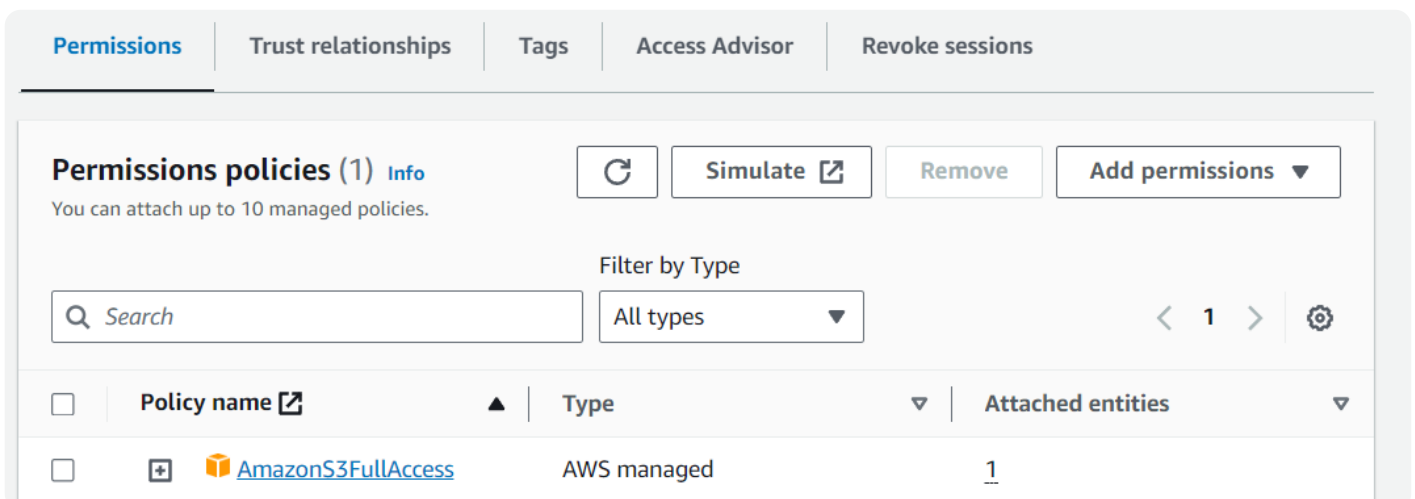
Service or use case

Choose a use case for the specified service.

Use case

- Lambda  
Allows Lambda functions to call AWS services on your behalf.

- In the input search bar, type “S3” and then select “AWSS3FullAccess” followed by “Next: Tags”.



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.' There are buttons for 'Simulate', 'Remove', and 'Add permissions'. A search bar and a 'Filter by Type' dropdown are present. Below is a table with one row for the 'AmazonS3FullAccess' policy.

<input type="checkbox"/>	Policy name <a href="#">↗</a>	Type	Attached entities
<input type="checkbox"/>	<a href="#">+ AmazonS3FullAccess</a>	AWS managed	<a href="#">1</a>

- Tags are optional, then click “Next: Review”.
- Give your Role and name and provide a brief description, followed by “Create Role”.

## Step 3: Creating a Lambda Function

Now the magic begins. Creating Lambda functions is fun, and easy. To create a Lambda function:

- Navigate to the Lambda service in the AWS management console.
- Click on “Create Function”.
- Provide your function with a descriptive name and select Python as the runtime. Then choose the IAM role you created in step 2 above.
- Click on “Create function”.


Notes:

Setting the execution role:

## 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

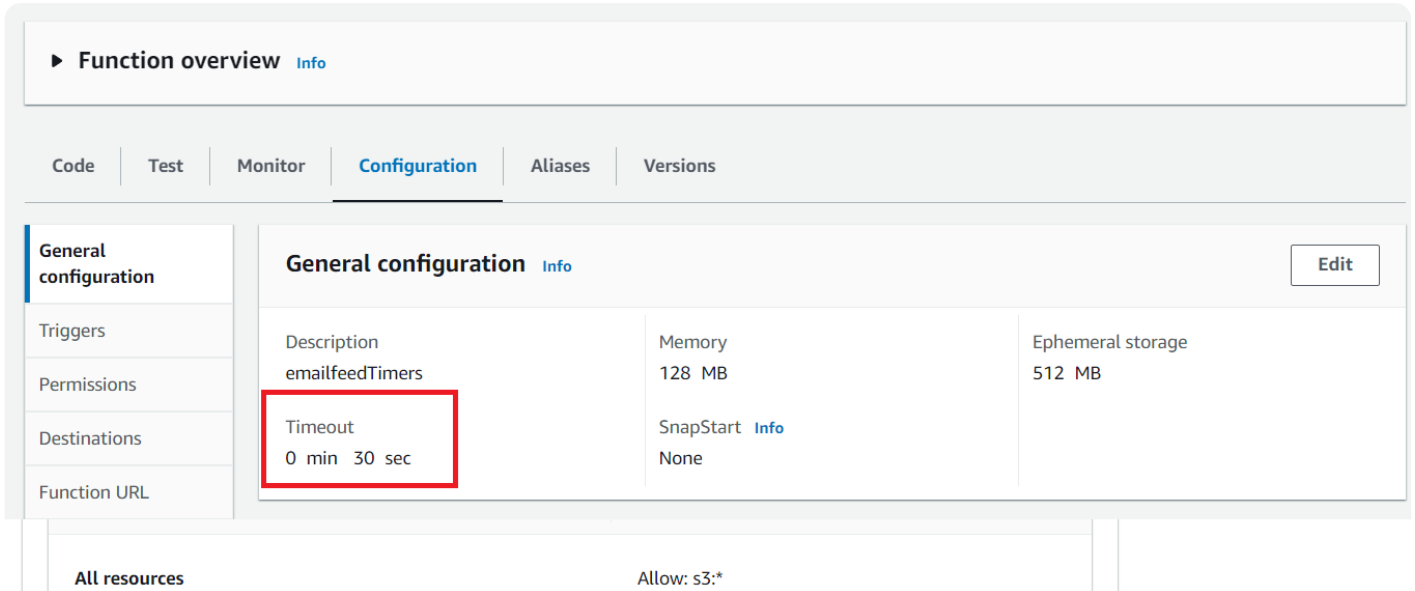


By action

**By resource**

Resource	Actions
All resources	Allow: s3:*

Setting the time-out value for the Lambda function. In this case, I've set it to 30 seconds.



The screenshot shows the AWS Lambda console's Configuration tab for a function named 'emailfeedTimers'. The 'Timeout' field is highlighted with a red box, showing a value of '0 min 30 sec'. Other configuration details include a memory size of 128 MB, 512 MB of ephemeral storage, and SnapStart set to 'None'.

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

## Step 4: Write the Lambda function to import the Disposable Email Domain list to S3

The example Lambda function uses the python requests module, and you may need to import it as it is no longer part of Boto3. AWS provides vague documentation on how to do this, but various tech articles can be found on the Internet.

### Example code:

The below python code (also available on [GitHub](#)) provides the entry point for the lambda\_handler function:

```
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/Disposable_Email_Domains/disposable_email_domains_{yesterday}.csv"

    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_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)

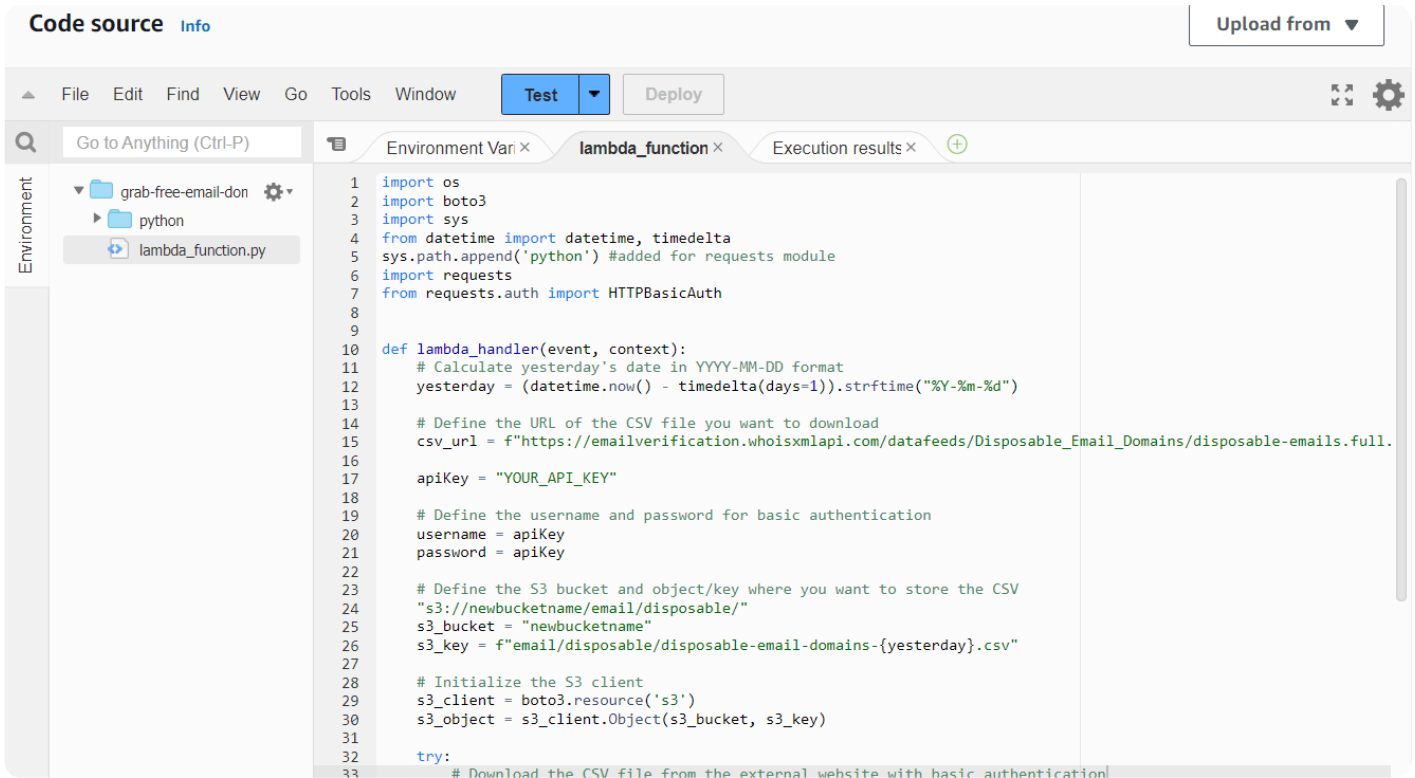
    try:
        # Download the CSV file from the external website with basic authentication
        response = requests.get(csv_url, auth=HTTPBasicAuth(username, password))

        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 S3'
}
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)
    }
```

When you're done, you should have something that resembles this:



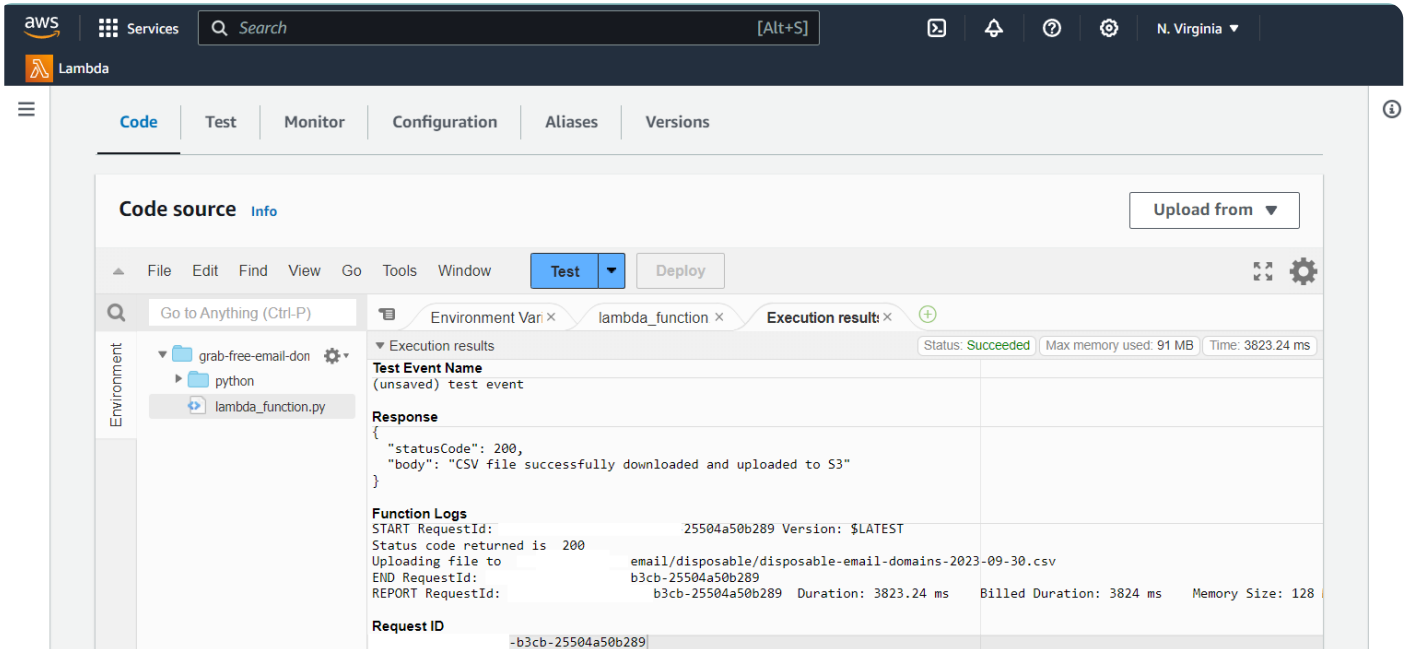
The screenshot shows a code editor interface with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. The main editor area displays a Python script for a Lambda function handler. The script imports necessary modules, calculates the date from yesterday, defines the URL of the CSV file to be downloaded, sets API key and authentication details, and configures the S3 bucket and object key. It then initializes the S3 client and sets up the object key for the CSV file.

```
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
```

## Step 5: Testing your new Lambda function

The last step is to test the Lambda function to ensure it can a) successfully retrieve the disposable email domain file, and b) write it to the S3 bucket:

- Click on “Test” at the top of the page, and you should see something similar.



- If you receive the message “requests” module not found, then you need to set up the python requests library correctly, which is outside the scope of this document.

If your Lambda function is set up correctly, the function will retrieve the file, and write it to the S3 bucket. You can navigate to the S3 bucket to verify it's there.

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

## Conclusion

Configuring AWS Lambda with access to an S3 bucket is a common task for cloud engineers. After walking you through the process, the next step is to determine what you want to do with this data, such as import it into Athena, Postgres or MySQL database. If you're not familiar with AWS Glue for ETL, be sure to check that out as well.