

NRD2 Data FeedをAWS S3にインポートするには

2023年10月4日

本文書では、Lambda関数を活用して[WhoisXML API](#)が提供する[NRD2 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](#)
- 高度なセキュリティ
- クリーンアップ、ライフサイクルファイル管理

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
- [WhoisXML APIのNRD2 Data Feed](#)へのアクセス。この例では、NRD2 Ultimate:Simpleファイルを使用します。APIキーが必要です。詳細につきましては、sales@whoisxmlapi.com にお問い合わせください。NRD2の仕様は、[こちら](#)でご確認いただけます。

Ultimate

- **Data included:** discovered, registered, updated and dropped domains, generic and country TLDs, WHOIS records.
- **Filename format:** `nrd.%DATE%.ultimate.[daily].[data|stats].[csv|json]`
- **Filename example:** `nrd.2021-12-20.ultimate.daily.data.csv.zip`, `nrd.2021-12-20.ultimate.daily.data.json.zip`
- **Average file sizes:**

File	Gzip size	Unpacked size	Rows
ultimate.daily.data.csv.gz	423.9MiB	3.7GiB	709.3K
ultimate.daily.data.json.gz	526.0MiB	6.2GiB	709.3K

ステップ1：AWS S3バケットの作成

最初のステップは、NRD2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.

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。



nrd2 Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Bucket overview

AWS Region

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

Amazon Resource Name (ARN)

 arn:aws:s3:::nrd2

Creation date

October 2, 2023, 08:26:49 (UTC-05:00)

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

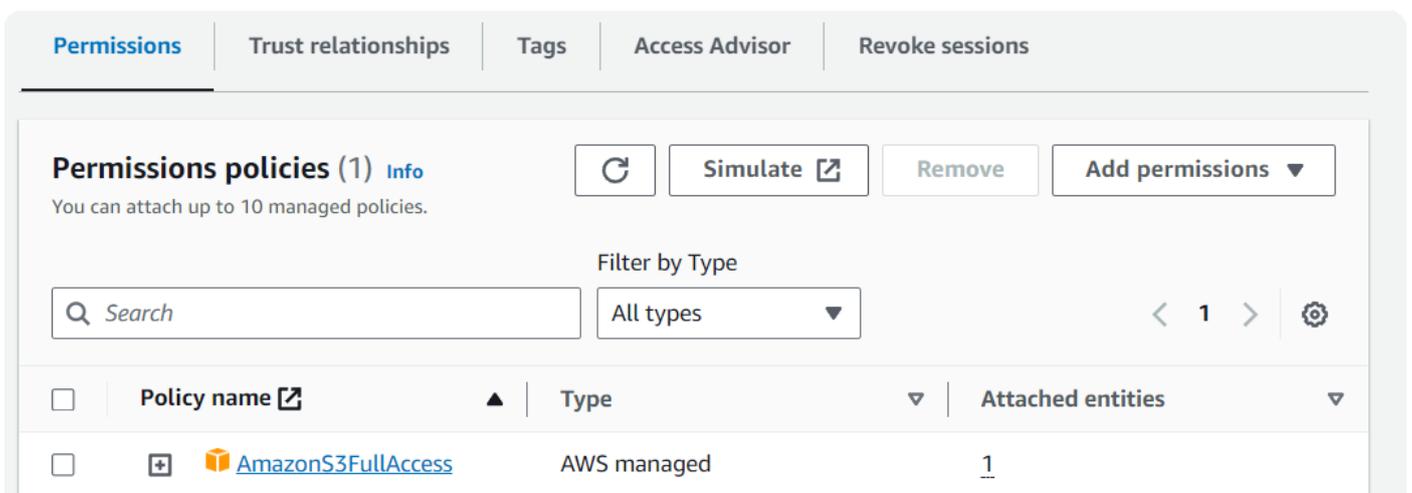
Choose a use case for the specified service.

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 active. 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'. Below the search bar is a table with columns for 'Policy name', 'Type', and 'Attached entities'. One policy is listed: 'AmazonS3FullAccess' (AWS managed) with 1 attached entity.

Policy name	Type	Attached entities
 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：



Role name

[grab-nrd2-ultimate-simple-role-t49whrsl](#)

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3分に設定しています。

grab-nrd2-ultimate-simple

► **Function overview** [Info](#)

Code

Test

Monitor

Configuration

Aliases

Versions

General configuration

Triggers

Permissions

Destinations

Function URL

General configuration

Edit

Description

-

Memory

128 MB

Ephemeral storage

10240 MB

Timeout

3 min 0 sec

SnapStart [Info](#)

None

ステップ4：NRD2 .csvファイルを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('pyrequests') #added for requests module
import requests
from requests.auth import HTTPBasicAuth

# Initialize the S3 client
s3_client = boto3.client('s3')

def download_nrd_file(url, s3_bucket, s3_key, authUserPass):

    chunk_size = 1024*1024

    try:
        # Download the binary file in chunks
        response = requests.get(url, stream=True, auth=HTTPBasicAuth(authUs
        response.raise_for_status())

        # Create a temporary file to store chunks
        temp_file = '/tmp/temp_file'

        with open(temp_file, 'wb') as f:
            for chunk in response.iter_content(chunk_size=chunk_size):
                f.write(chunk)

        # Upload the binary file to S3 from the temporary file
        s3_client.upload_file(temp_file, s3_bucket, s3_key)

        # Clean up the temporary file
        os.remove(temp_file)
```



```
        return True
    except Exception as e:
        print(f'Error: {str(e)}')
        return False

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
    nrd_url = f"https://newly-registered-domains.whoisxmlapi.com/datafeeds/"

    # Define your API Key here
    apiKey = "<YOUR_API_KEY>"

    # Define the S3 bucket and object/key where you want to store the file
    s3_bucket = "nrd2"
    s3_key = f"nrd2-simple-{yesterday}.csv.gz"

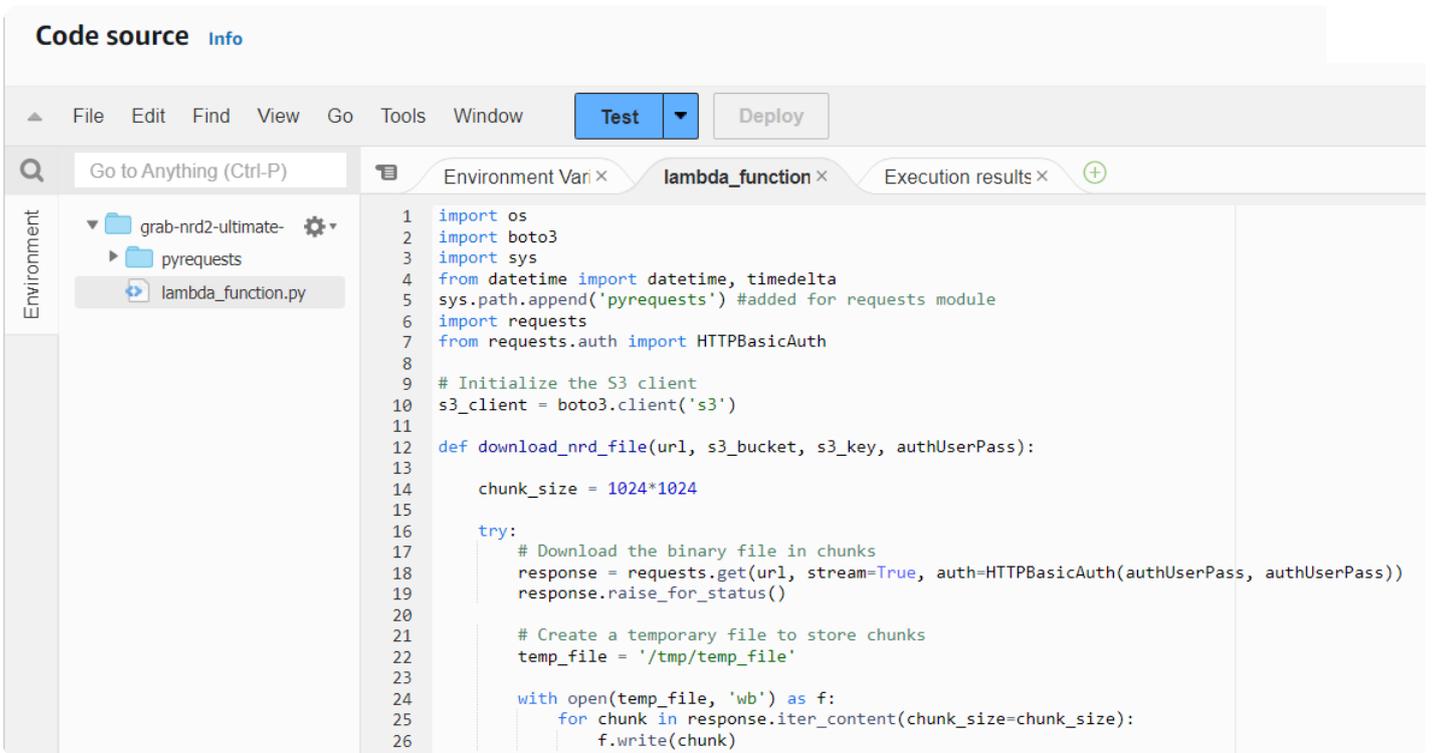
    try:
        # Download the NRD2 file with basic authentication
        success = download_nrd_file(nrd_url, s3_bucket, s3_key, apiKey)

        print("Status code returned is ", str(success))

        if success:
            # Upload the NRD file to S3
            print(f"Uploading file to ", s3_bucket, s3_key)
            return {
                'statusCode': 200,
                'body': 'NRD2 file successfully downloaded and stored in S3'
            }
        else:
            bodyStr = f"Failed to download {nrd_url}"
            return {
                'statusCode': 500,
                '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 the AWS Lambda console's 'Code source' page. The interface includes a menu bar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', and 'Window'. There are 'Test' and 'Deploy' buttons. The left sidebar shows the environment structure: 'Environment' > 'grab-nrd2-ultimate-' > 'pyrequests' > 'lambda_function.py'. The main area displays the Python code for the lambda function, which includes imports for boto3, sys, datetime, and requests, and a function definition for downloading a file from S3 in chunks.

```
1 import os
2 import boto3
3 import sys
4 from datetime import datetime, timedelta
5 sys.path.append('pyrequests') #added for requests module
6 import requests
7 from requests.auth import HTTPBasicAuth
8
9 # Initialize the S3 client
10 s3_client = boto3.client('s3')
11
12 def download_nrd_file(url, s3_bucket, s3_key, authUserPass):
13
14     chunk_size = 1024*1024
15
16     try:
17         # Download the binary file in chunks
18         response = requests.get(url, stream=True, auth=HTTPBasicAuth(authUserPass, authUserPass))
19         response.raise_for_status()
20
21         # Create a temporary file to store chunks
22         temp_file = '/tmp/temp_file'
23
24         with open(temp_file, 'wb') as f:
25             for chunk in response.iter_content(chunk_size=chunk_size):
26                 f.write(chunk)
```

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) NRD2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。

Code source [Info](#) Upload from ▾

File Edit Find View Go Tools Window Test ▾ Deploy

Go to Anything (Ctrl-P) Environment Var × lambda_function × Execution result × (+)

Environment

- grab-nrd2-ultimate-
 - pyrequests
 - lambda_function.py

Execution results Status: **Succeeded** Max memory used: 128 MB Time: 26769.03 ms

Test Event Name
nrdTestEvent

Response

```
{
  "statusCode": 200,
  "body": "NRD2 file successfully downloaded and stored in S3"
}
```

Function Logs

```
START RequestId: b543ebe0-5ee3-4e76-a334-9765e13246a1 Version: $LATEST
Status code returned is True
Uploading file to nrd2 /nrd2-simple-2023-10-01.csv.gz
END RequestId: b543ebe0-5ee3-4e76-a334-9765e13246a1
REPORT RequestId: b543ebe0-5ee3-4e76-a334-9765e13246a1 Duration: 26769.03 ms Billed Duration: 26770 ms Memory Size: 128
```

Request ID
b543ebe0-5ee3-4e76-a334-9765e13246a1

- 「"requests" module not found」というメッセージを受け取った場合は、[python requests library](#)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。

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](#)

Actions ▾

Find objects by prefix < 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 nrd2-simple-2023-10-01.csv.gz	gz	October 2, 2023, 12:00:00 (UTC-05:00)	164.4 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**または**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定めることです。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。