Importing NRD2 Data Feed to AWS S3

Posted on October 4, 2023

The intention of this document is to show you the basics of how to download the WhoisXML API's NRD2 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.

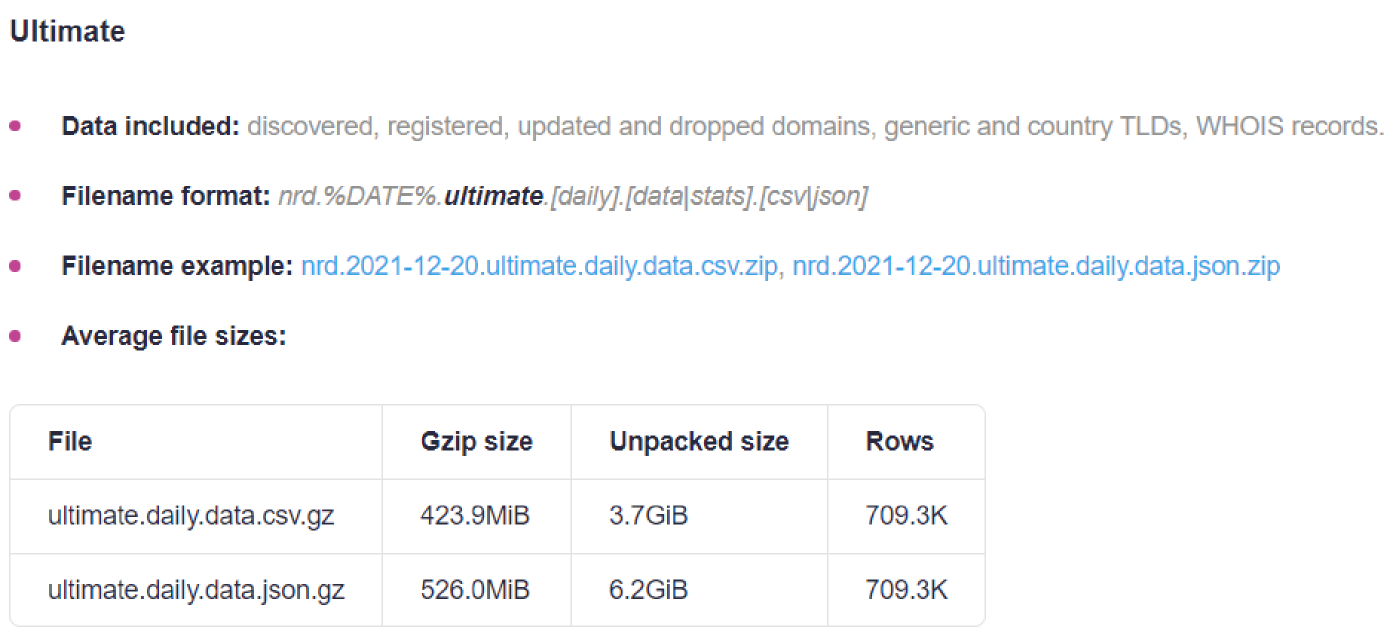
Out of scope:

- Scheduling a Lambda function
- ETL pipelining
- Importing the python requests module
- Advanced Security
- Clean-up, life cycle file management

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 WhoisXML API's NRD2 data feed. In this example, we will be using the NRD2 Ultimate:Simple files. You will need an API key with access to the data feed. Please contact us for more information. For more information on the NRD2 specifications, please visit here.



Ultimate

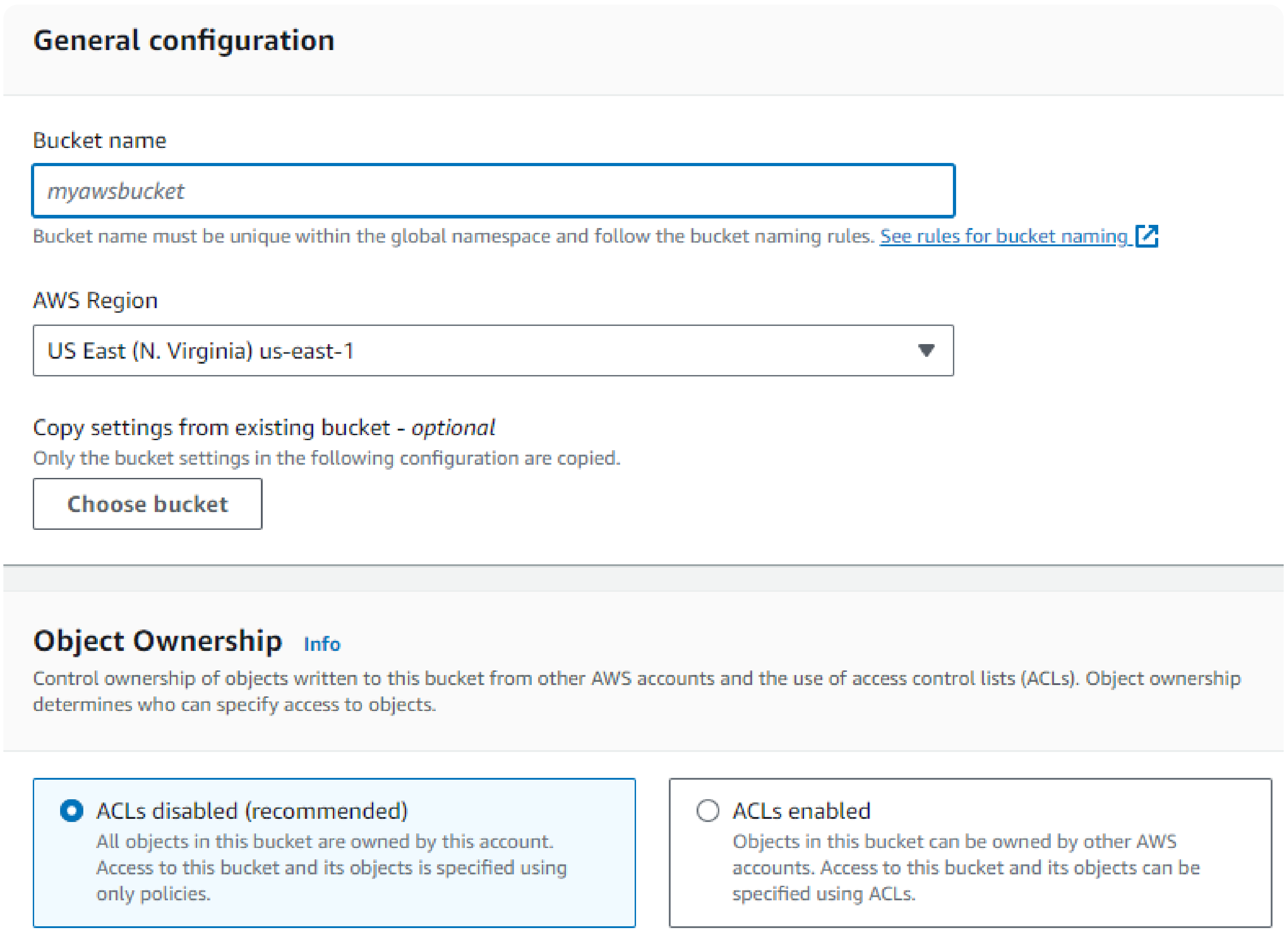
- **Data included:** discovered, registered, updated and dropped domains, generic and country TLDs, WHOIS records.
- **Filename format:** *nrd.%DATE%.**ultimate**.[daily].[data|stats].[csv|json]*
- **Filename example:** nrd.2021-12-20.ultimate.daily.data.csv.zip, nrd.2021-12-20.ultimate.daily.data.json.zip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

Step 1: Create an AWS S3 Bucket

The first step is to create an S3 bucket to write the NRD2 file 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

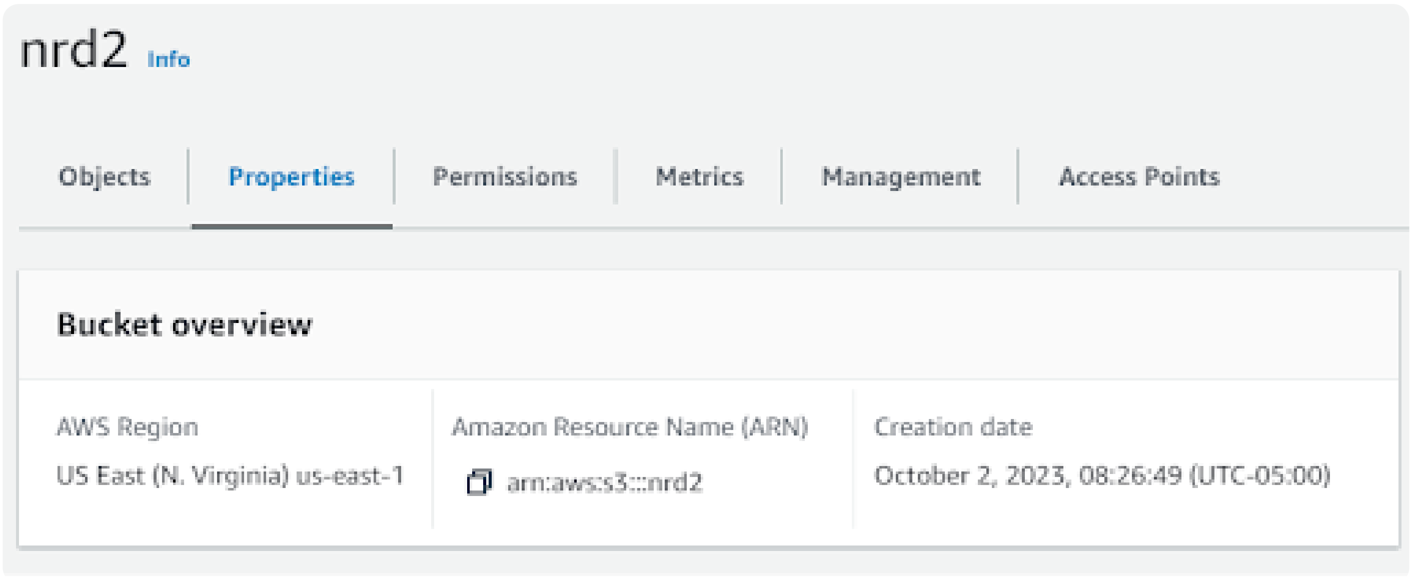
Object Ownership Info

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”.


nrd2

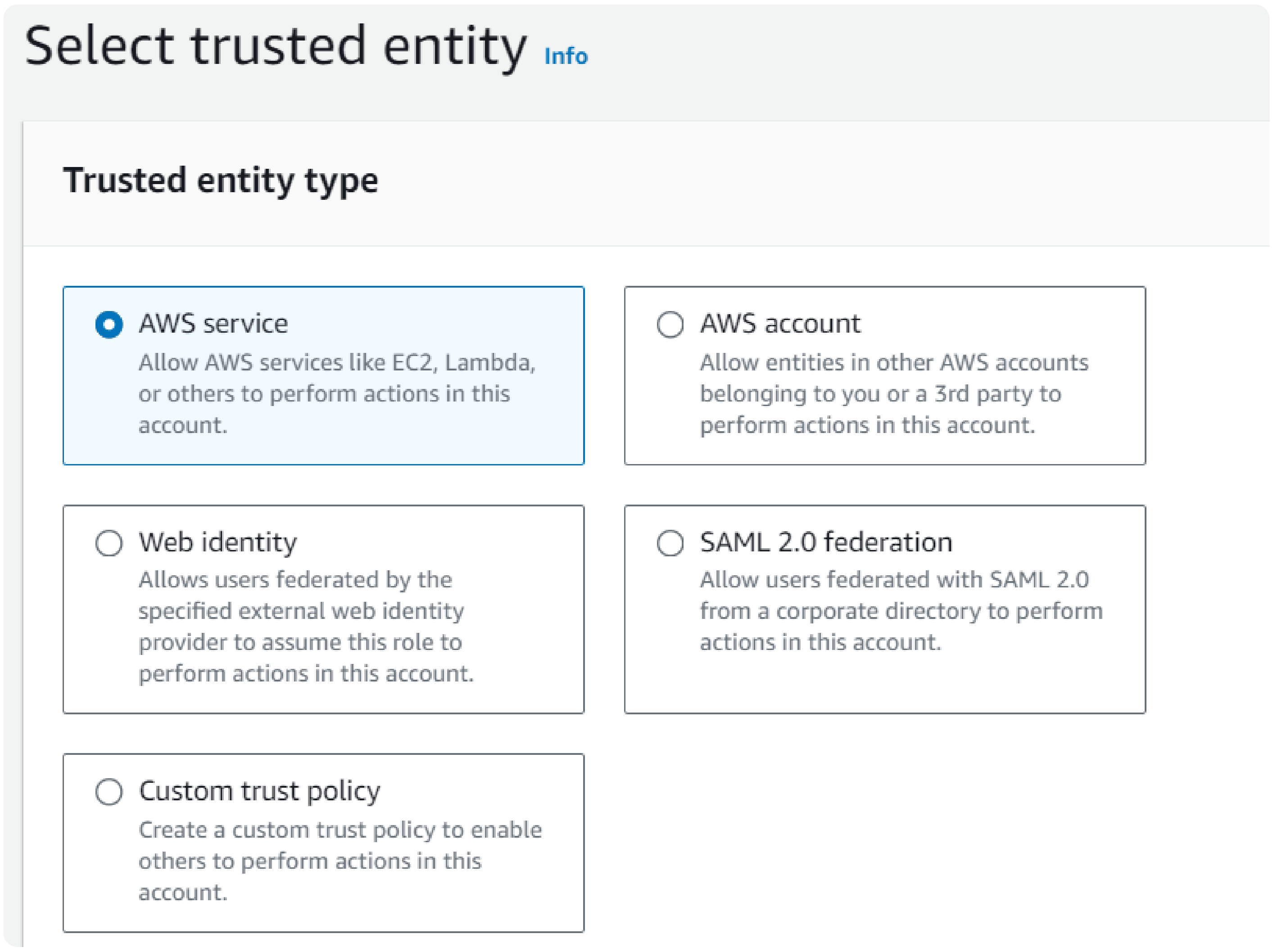
Bucket overview

AWS Region	Amazon Resource Name (ARN)	Creation date
US East (N. Virginia) us-east-1	arn:aws:s3:::nrd2	October 2, 2023, 08:26:49 (UTC-05:00)

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.

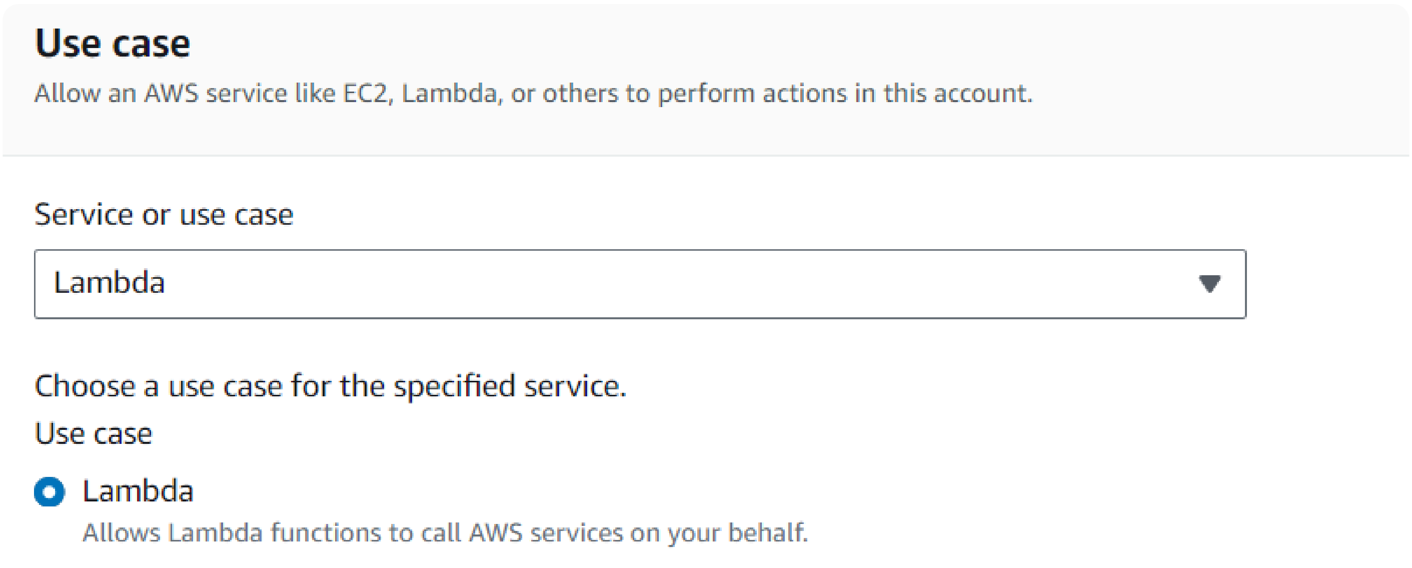
AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

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.

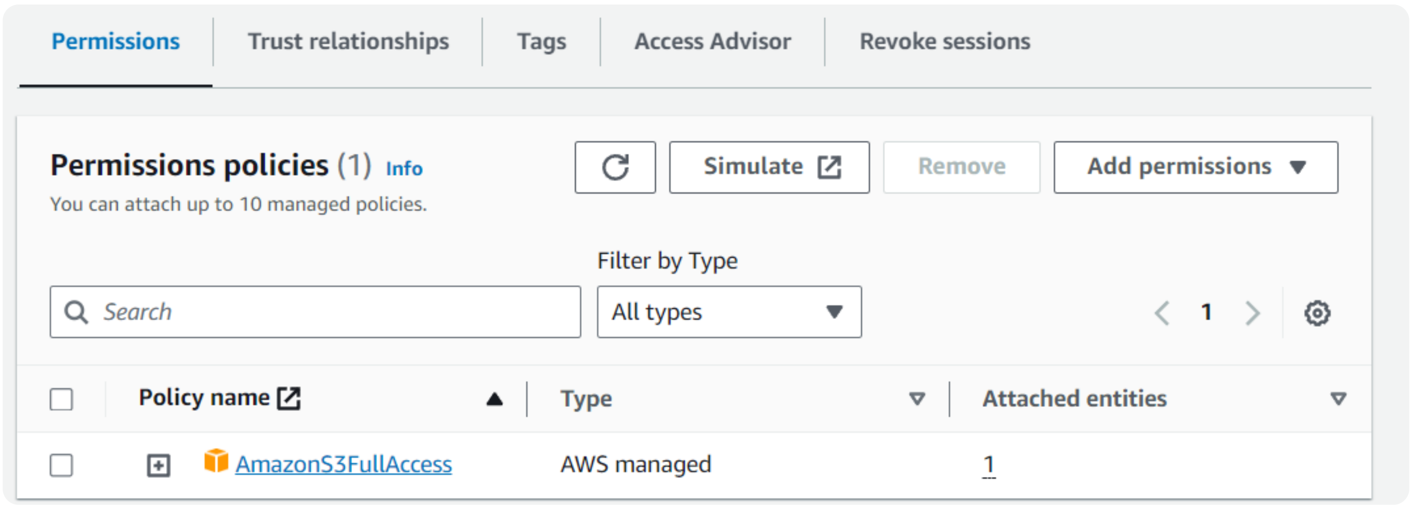
Service or use case: Lambda

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".



Permissions policies (1) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	1

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

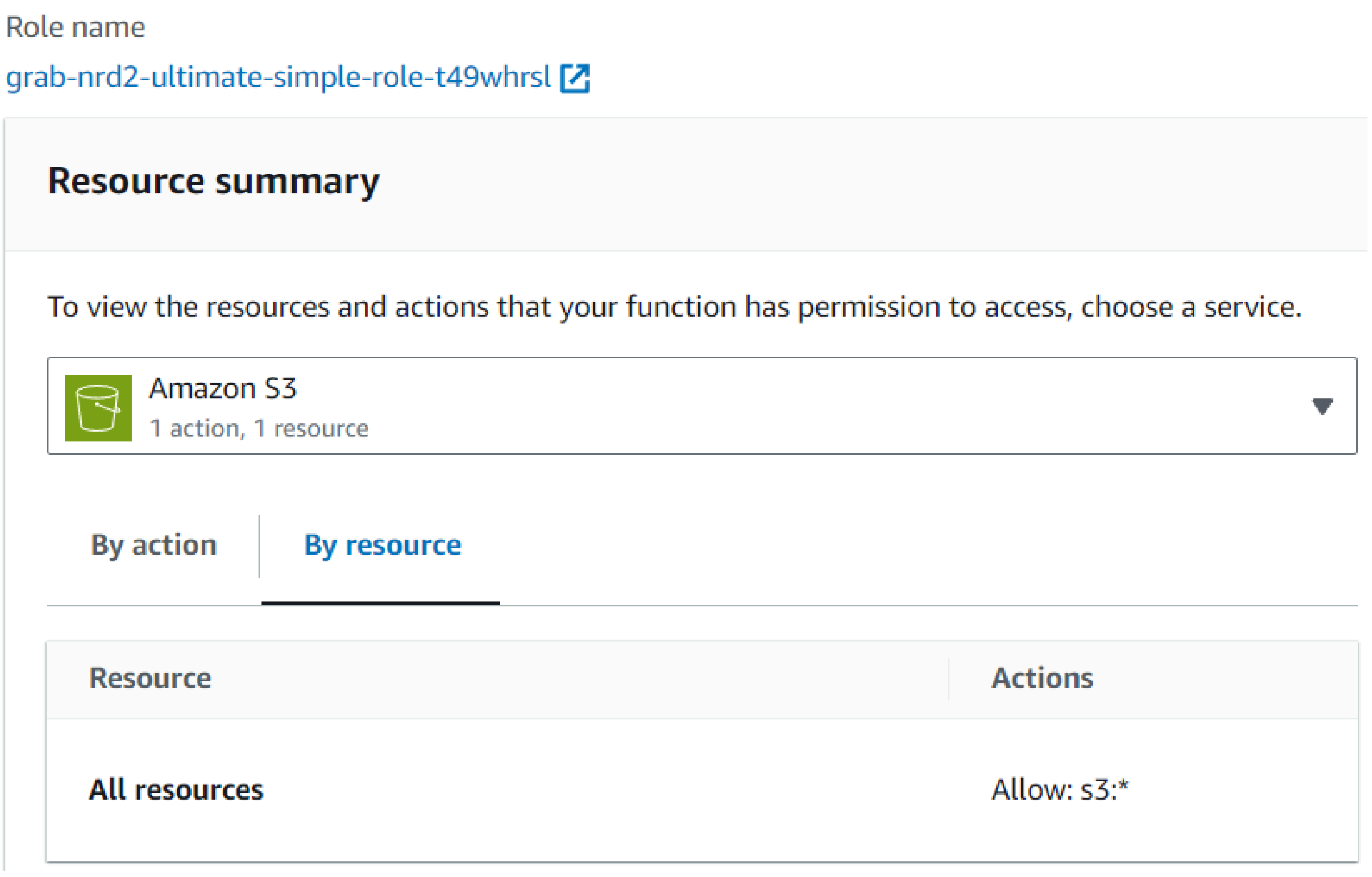
Step 3: Creating a Lambda Function

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:


Role name

grab-nrd2-ultimate-simple-role-t49whrsl

Resource summary

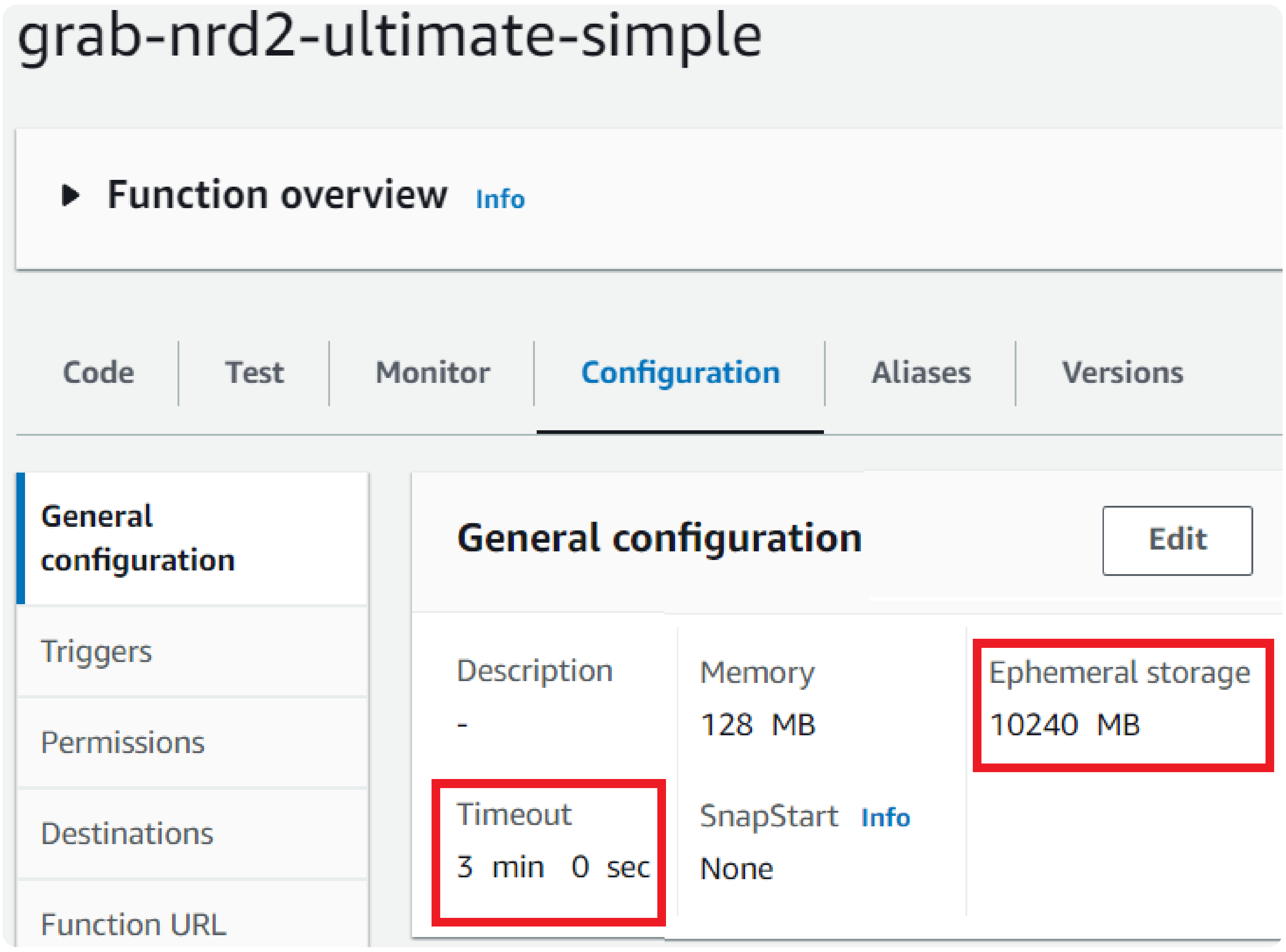
To view the resources and actions that your function has permission to access, choose a service.

Amazon S3
1 action, 1 resource

By action | By resource

Resource	Actions
All resources	Allow: s3:*

Setting the time-out value for the Lambda function. In this case, I've set it to 3 minutes.


grab-nrd2-ultimate-simple

Function overview Info

General configuration

Description	Memory	Ephemeral storage
-	128 MB	10240 MB
Timeout	**SnapStart Info**	
3 min 0 sec	None	

Step 4: Write the Lambda function to import the NRD2 .csv file 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.

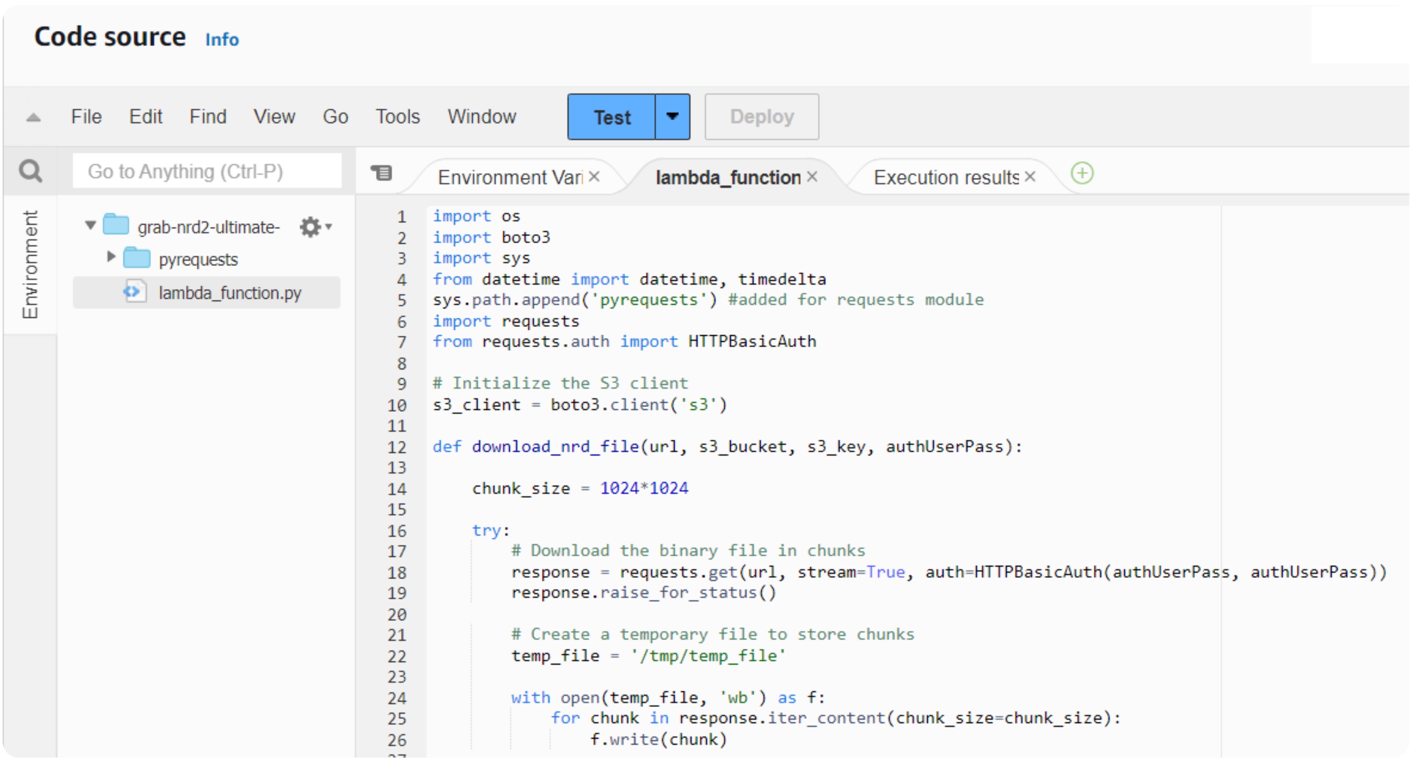
Example code:

The below python code provides the entry point for the lambda_handler function:

```
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(authUserPass, authUserPass))
        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/Newly_Registered_Domai

        # 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)
            }
```

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

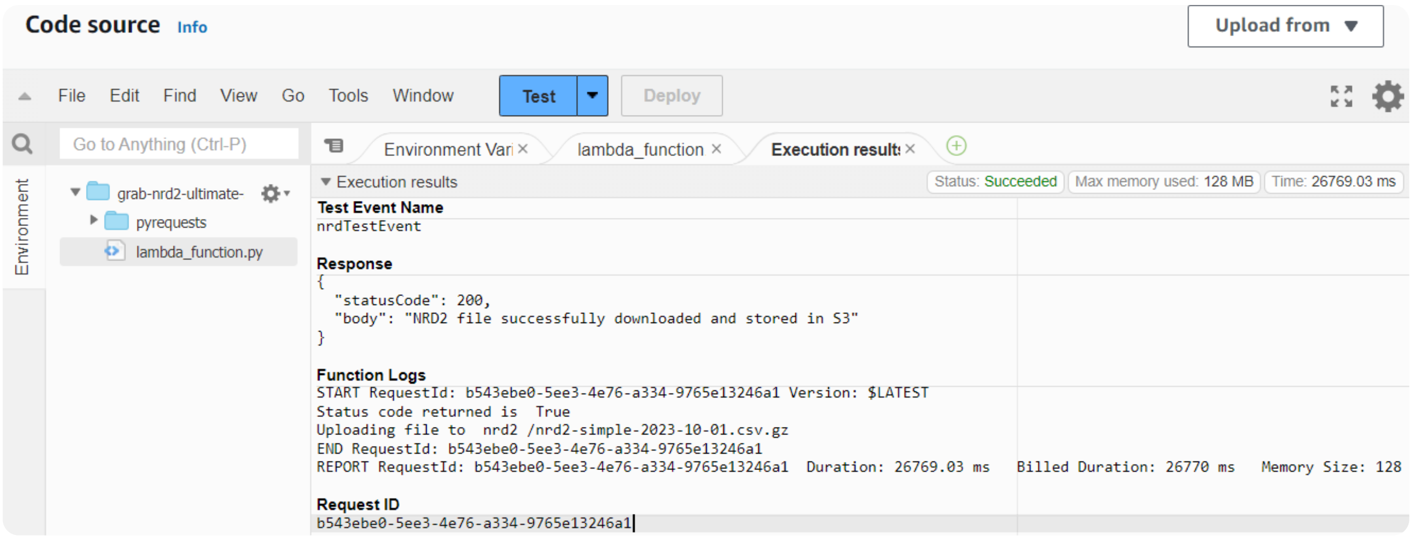


```python
import os
import boto3
import sys
from datetime import datetime, timedelta
sys.path.append('pyrequests') #added for requests module
import requests
from requests.auth import HTTPBasicAuth

# 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(authUserPass, authUserPass))
        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)
```

Step 5: Testing your new Lambda function

The last step is to test the Lambda function to ensure it can a) successfully retrieve the NRD2 file, and b) write it to the S3 bucket:

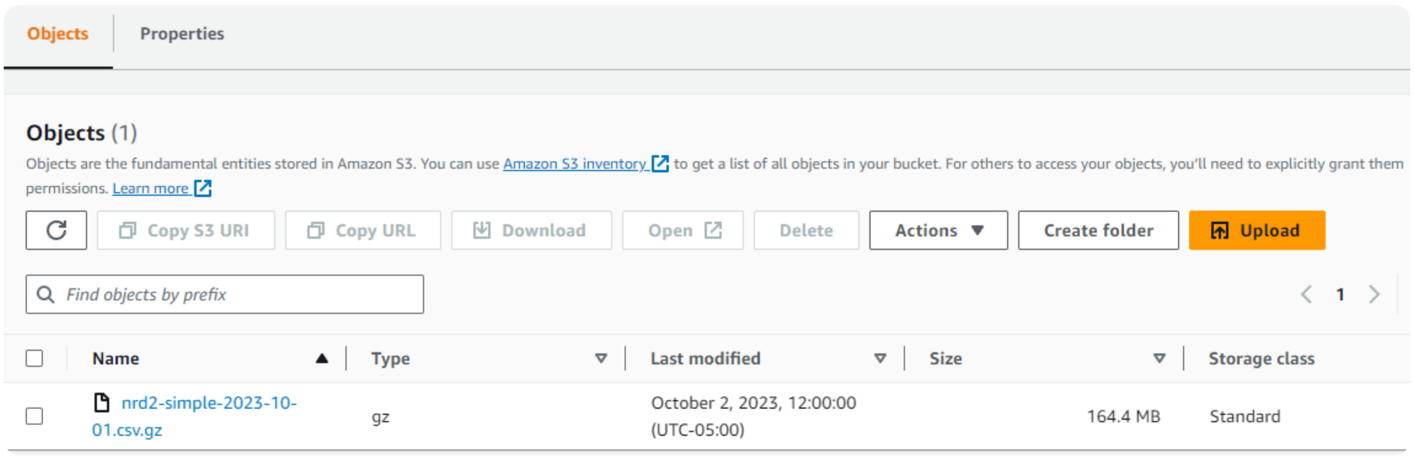
- Click on “Test” at the top of the page, and you should see something similar.



```
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
```

- 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.



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

Name	Type	Last modified	Size	Storage class
nrd2-simple-2023-10-01.csv.gz	gz	October 2, 2023, 12:00:00 (UTC-05:00)	164.4 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.