

Utilizing Amazon S3 Triggers and Lambda Functions for Automated Data Processing and Storage

This document provides a comprehensive guide on leveraging the Amazon S3 trigger mechanism to seamlessly invoke a Lambda function, enabling the efficient processing of newly ingested data from the input bucket. The processed data is subsequently stored in the designated output bucket, ensuring a streamlined and automated data workflow. This integration allows for the transformation and enrichment of data as it traverses through the AWS ecosystem, facilitating seamless data management and enhancing overall system efficiency.

Steps Involved

To implement this solution, complete the following high-level steps:

1. Creating S3 buckets
 2. Creating read-and-write policy
 3. Creating lambda-specific roles and attaching the policy
 4. Creating lambda function and setting trigger

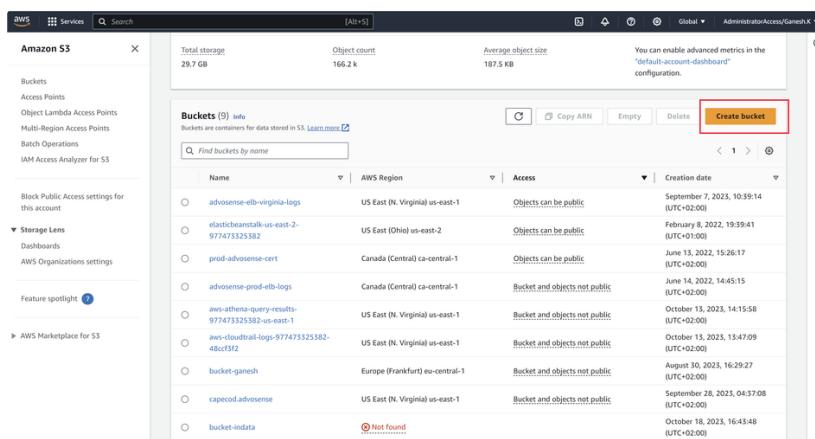
•• Prerequisite

Not expertise level is required at least first-hand experience so that the following steps are ease to understand.

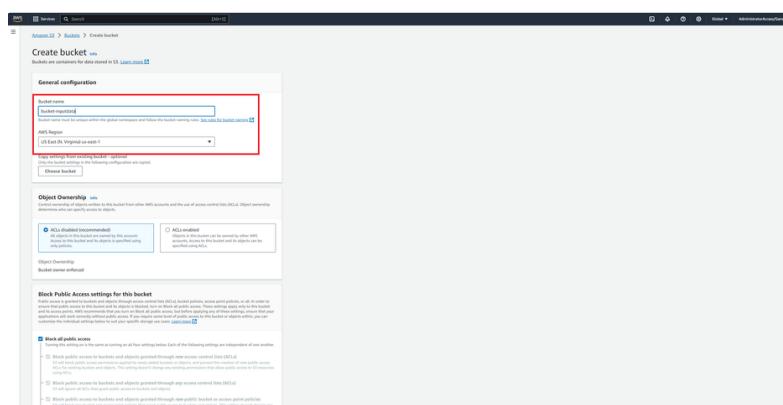
- ## 1. Python & pandas library

Step 1: Creating S3 buckets

1. Open the [Amazon S3 console](#) and select the **Buckets** page.



2. Choose **Create bucket** to load the input data.
 3. Under **General configuration**, do the following:
 - a. For the **Bucket name**, enter a globally unique name that meets the [Amazon S3 Bucket naming rules](#). Bucket names can contain only lowercase letters, numbers, dots (.), and hyphens (-).
 - b. For **AWS Region**, choose a Region. Later in the tutorial, you must create your Lambda function in the same Region.



The screenshot shows the 'Create Bucket' configuration page for AWS S3. It includes sections for 'Bucket Versioning' (Enabled), 'Tags - optional (0)', 'Default encryption' (info), 'Encryption type: SSE', 'Bucket Key' (info), and 'Advanced settings'. A note at the bottom says: 'After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.' The 'Create bucket' button is highlighted with a red box.

4. Leave all other options set to their default values and choose **Create bucket**.
5. Create **another bucket** to store the processed data as per the above procedure
6. Should have **two buckets** to store one for input data and one for processed data

The screenshot shows the 'Account snapshot' page for AWS S3. It displays metrics like Total storage (24.7 TB), Object count (166,548), and Average object size (107.5 KB). The 'Metrics' section shows data from Storage Lens. Below, the 'Buckets' section lists two buckets: 'sensored-data-ingestion' and 'sensored-data-processing'. Both buckets are highlighted with a red box.

Step 2: Creating a read-and-write policy

1. Open the **Management console** and select the **IAM** page
2. On the **IAM** page select **policies**

The screenshot shows the 'IAM Dashboard' page for AWS IAM. It includes sections for 'IAM resources' (User groups: 3, Users: 5, Roles: 23, Policies: 19, Identity providers: 1) and 'What's new' (with a note about IAM Roles Anywhere). The 'Policies' section is highlighted with a red box. The left sidebar shows navigation options like 'Identity and Access Management (IAM)', 'Access management', 'Access reports', 'Related consoles', and 'Tools'.

3. Choose to create a policy

The screenshot shows the AWS IAM Policies page. On the left, there's a sidebar with navigation links like 'Identity and Access Management (IAM)', 'Dashboard', 'Access management', 'Policies', 'Access reports', and 'Related consoles'. The main area displays a list of existing policies, each with a name, type (AWS managed), used as (None), and a brief description. At the top right of the list, there's a red box around the 'Create policy' button.

4. Select JSON

The screenshot shows the 'Specify permissions' step in the 'Create policy' wizard. It has two tabs: 'Visual' (disabled) and 'JSON' (selected). The 'JSON' tab contains a code editor with a red box around it, showing a single statement. To the right of the editor are sections for 'Edit statement', 'Add actions', 'Choose a service' (with a dropdown menu), 'Available' services (API Gateway, API Gateway V2, ASC, etc.), and buttons for 'Add a resource' and 'Add a condition (optional)'. At the bottom, there are status indicators and a 'Next' button.

5. Paste the following code in the policy editor field

```

1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Sid": "putObject",
6              "Effect": "Allow",
7              "Action": "s3:PutObject",
8              "Resource": "arn:aws:s3:::mybucket/*"
9          }
10     ]
11 }

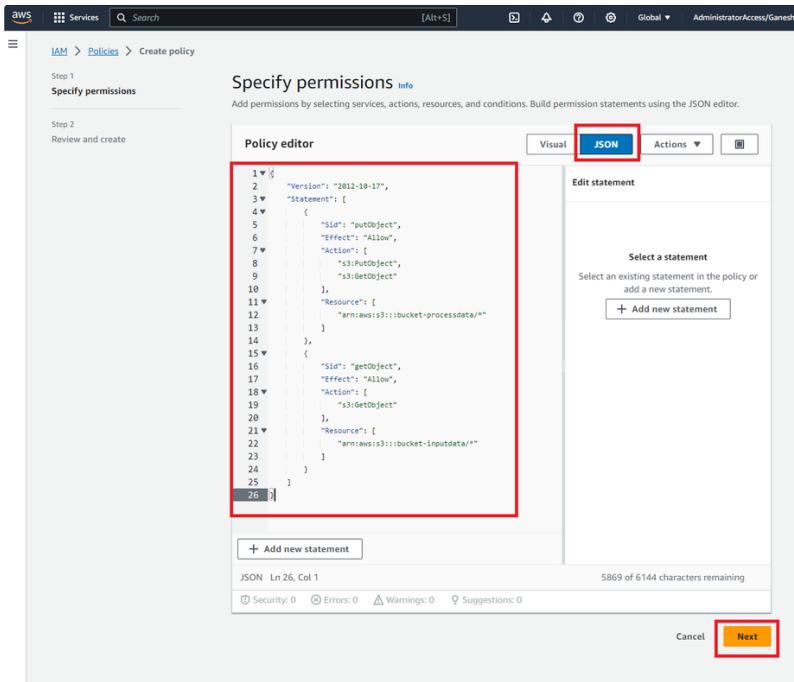
```

```

6         "Effect": "Allow",
7         "Action": [
8             "s3:PutObject",
9             "s3:GetObject"
10            ],
11        "Resource": [
12            "arn:aws:s3:::bucket-processeddata/*"
13           ],
14    },
15    {
16        "Sid": "getObject",
17        "Effect": "Allow",
18        "Action": [
19            "s3:GetObject"
20           ],
21        "Resource": [
22            "arn:aws:s3:::bucket-inputdata/*"
23           ]
24    }
25  ]
26 }
27

```

6. Select Next



7. In the next window, give the **policy name and description**, then select **Create policy**. In our case the name of the policy is **S3writebucket**

IAM > Policies > Create policy

Step 1
Specify permissions

Step 2
Review and create

Review and create Info
Review the permissions, specify details, and tags.

Policy details

Policy name	Enter a meaningful name to identify this policy.
S3writebucket	
Maximum 128 characters. Use alphanumeric and '+_,@_-' characters.	
Description - optional	Add a short explanation for this policy.
This policy is used to give access to store the processed data from lambda to the concern bucket	
Maximum 1,000 characters. Use alphanumeric and '+_,@_-' characters.	

Permissions defined in this policy Info
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Allow (1 of 384 services) Show remaining 383 services

Service	Access level	Resource	Request count
S3	Limited: Read, Write	Multiple	None

Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag
You can add up to 50 more tags.

Cancel Previous **Create policy**

8. Once created, verify by searching the name in the field as shown below

IAM Services Search [Alt+S] Global AdministratorAccess/Ganesh.K ▾

Identity and Access Management (IAM) Policy S3writebucket created. View policy

IAM Policies (1150) Info A policy is an object in AWS that defines permissions.

Filter by Type: All types 14 matches 1 < > ⌂

Policy name	Type	Used as	Description
AmazonDMSRedsh...	AWS managed	None	Provides access to manage S3 settings
AmazonS3FullAccess	AWS managed	None	Provides full access to all buckets via
AmazonS3ObjectL...	AWS managed	None	Provides AWS Lambda functions per
AmazonS3Outpost...	AWS managed	None	Provides full access to Amazon S3 or
AmazonS3Outpost...	AWS managed	None	Provides read only access to Amazon
AmazonS3ReadOn...	AWS managed	None	Provides read only access to all buck
AWSBackupService...	AWS managed	None	Policy containing permissions necess
AWSSBackupService...	AWS managed	None	Policy containing permissions necess
AWSS3OnOutpost...	AWS managed	None	Allow Amazon S3 on Outposts servic
IVSRecordToS3	AWS managed	None	Service Linked Role to perform S3 Pa
QuickSightAccessF...	AWS managed	None	Policy used by QuickSight team to ac
S3StringLambdaSer...	AWS managed	None	Enables access to AWS Services and l
S3writebucket	Customer managed	None	This policy is used to give access to
SageMakerS3BucketP...	Customer managed	None	-

Step 3: Creating lambda-specific roles and attaching the policy

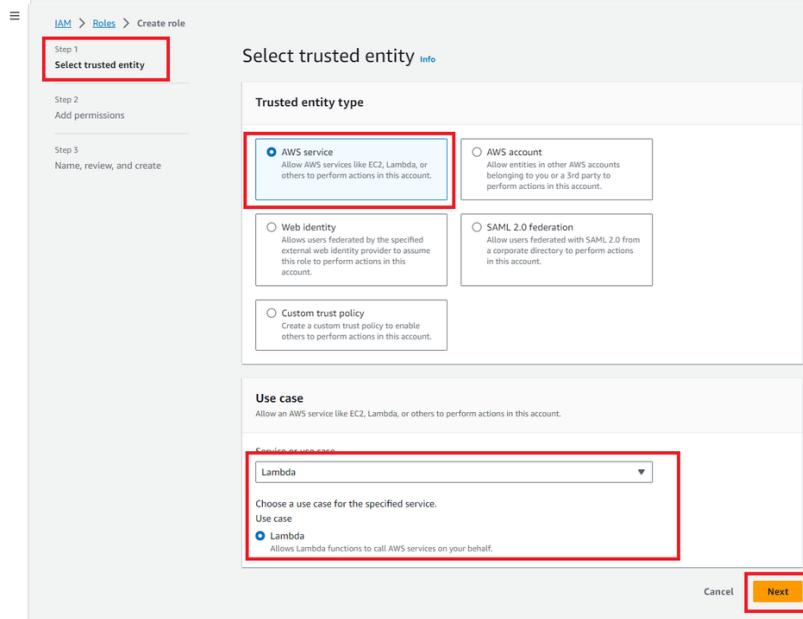
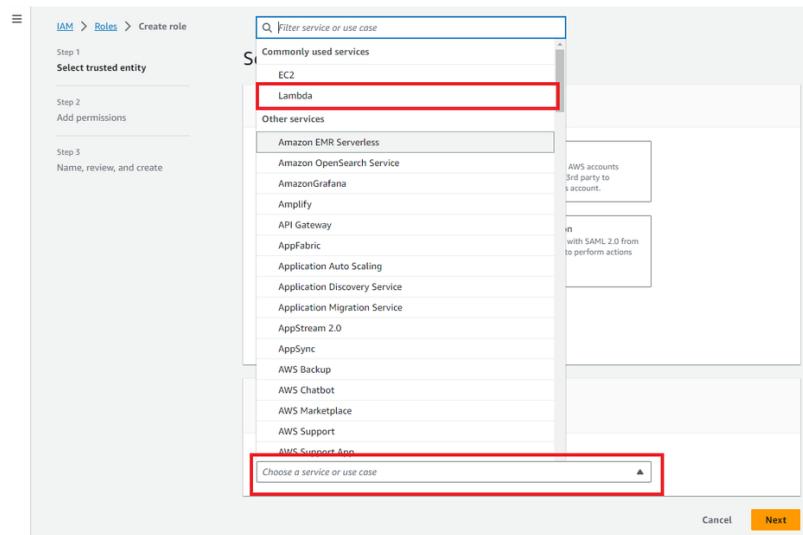
1. On the same IAM page, select **Roles**

2. Select **Create Role**

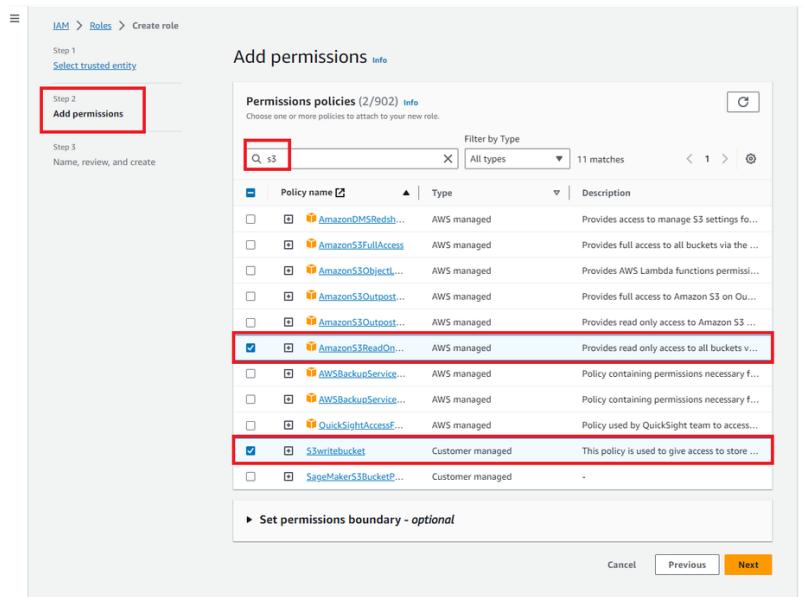
The screenshot shows the AWS Identity and Access Management (IAM) service. On the left, there's a navigation sidebar with sections like 'Identity and Access Management (IAM)', 'Dashboard', 'Access management', 'Access reports', and 'Related consoles'. Under 'Access management', the 'Roles' section is selected and highlighted with a red box. In the main content area, the 'Roles' page is displayed with a title 'Roles (23) Info'. It lists 23 roles, each with a checkbox, a role name, and a description of the trusted entity. A prominent yellow 'Create role' button is located at the top right of the list. Below the list, there are buttons for 'Roles Anywhere' and 'Manage'.

3. In the create role page, select **AWS service** and choose service or use case as lambda, then select next

This screenshot shows the 'Create role' wizard at 'Step 1: Select trusted entity'. The 'Trusted entity type' section contains four options: 'AWS service' (selected and highlighted with a red box), 'AWS account', 'Web identity', and 'SAML 2.0 federation'. Below this, the 'Use case' section asks 'Allow an AWS service like EC2, Lambda, or others to perform actions in this account.' A dropdown menu labeled 'Choose a service or use case' is shown, with a red box highlighting it. At the bottom right of the screen, there are 'Cancel' and 'Next' buttons, with 'Next' also highlighted with a red box.



4. In the next add permission page, select three policies namely **S3ReadonlyAccess**, **S3writebucket** and **AWSLambdaExecute**, then click **Next**



Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Permissions policies (3/902) Info

Choose one or more policies to attach to your new role.

Policy name	Type	Description
AmazonS3Object... (AWS managed)	AWS managed	Provides AWS Lambda functions permis...
AmazonSageMaker... (AWS managed)	AWS managed	Service role policy used by the AWS La...
AmazonSageMaker... (AWS managed)	AWS managed	Service role policy used by the AWS La...
AWSCodeDeploy8... (AWS managed)	AWS managed	Provides CodeDeploy service access to ...
AWSCodeDeploy8... (AWS managed)	AWS managed	Provides CodeDeploy service limited a...
AWSDeerLensLam... (AWS managed)	AWS managed	This policy specifies permissions requir...
AWSLambda_FullA... (AWS managed)	AWS managed	Grants full access to AWS Lambda serv...
AWSLambda_ReadOnly... (AWS managed)	AWS managed	Grants read-only access to AWS Lamb...
AWSLambdaBasicE... (AWS managed)	AWS managed	Provides write permissions to CloudW...
AWSLambdaBasicExec... (Customer managed)	Customer managed	-
AWSLambdaBasicExec... (Customer managed)	Customer managed	-
AWSLambdaBasicExec... (Customer managed)	Customer managed	-
AWSLambdaDyna... (AWS managed)	AWS managed	Provides list and read access to Dynam...
AWSLambdaENIMa... (AWS managed)	AWS managed	Provides minimum permissions for a L...
AWSLambdaExecute (AWS managed)	AWS managed	Provides Put, Get access to S3 and full ...
AWSLambdaInvoke... (AWS managed)	AWS managed	Provides read access to DynamoDB Str...
AWSLambdaKinesi... (AWS managed)	AWS managed	Provides list and read access to Kinesis...
AWSLambdaMSKE... (AWS managed)	AWS managed	Provides permissions required to acces...
AWSLambdaRole (AWS managed)	AWS managed	Default policy for AWS Lambda service...
AWSLambdaSOSO... (AWS managed)	AWS managed	Provides receive message, delete mess...

▶ Set permissions boundary - optional

Cancel Previous **Next**

5. On the next page, give the role name, and description, then select **Create role**

IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
preprocessingwithlambda
Maximum 64 characters. Use alphanumeric and "+", "-", ".", "@" characters.

Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and "+", "-", ".", "@" characters.

Step 1: Select trusted entities

Trust policy

```

1+ [
2+   "Version": "2012-10-17",
3+   "Statement": [
4+     {
5+       "Effect": "Allow",
6+       "Action": [
7+         "sts:AssumeRole"
8+       ],
9+       "Principal": [
10+         "service-role:lambda.amazonaws.com"
11+       ]
12+     }
13+   ]
14+ ]
15+
16]

```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonS3ReadOnlyAccess	AWS managed	Permissions policy
AWSLambdaExecute	AWS managed	Permissions policy
S3writebucket	Customer managed	Permissions policy

Step 2: Add permissions

Policy name	Type	Attached as
AmazonS3ReadOnlyAccess	AWS managed	Permissions policy
AWSLambdaExecute	AWS managed	Permissions policy
S3writebucket	Customer managed	Permissions policy

Step 3: Add tags

Add tags - *optional* [Info](#)
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)
You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create role](#)

6. Once created, verify by searching the name in the field as shown below

Identity and Access Management (IAM)

[Search IAM](#)

Access management

- Users
- Roles** (selected)
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analyzers
- Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

Related consoles

- IAM Identity Center
- AWS Organizations

Role preprocessingwithlambda created.

[View role](#)

IAM > Roles

Roles (24) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

[Delete](#) [Create role](#)

1 match [X](#) [<](#) [1](#) [>](#) [@](#)

<input type="checkbox"/> Role name	▲ Trusted entities
preprocessingwithlambda AWS Service: lambda	

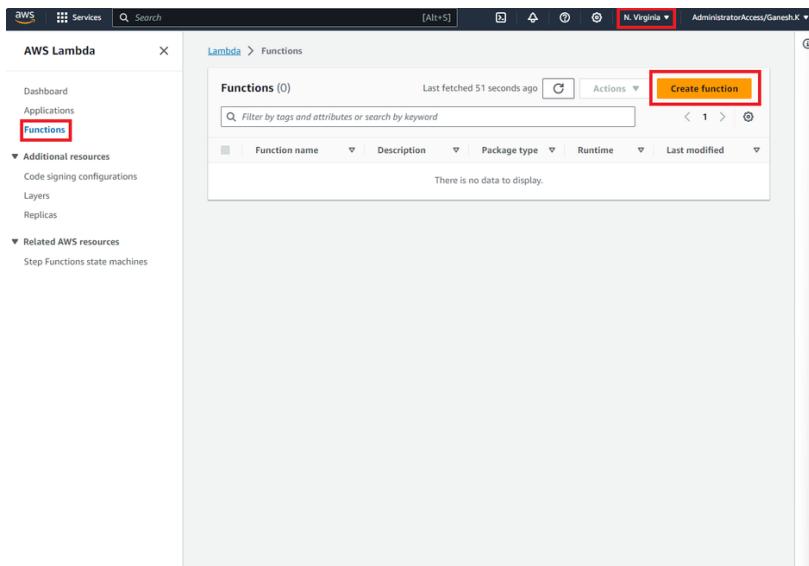
Roles Anywhere [Info](#)
Authenticate your non AWS workloads and securely provide access to AWS services.

[Manage](#)

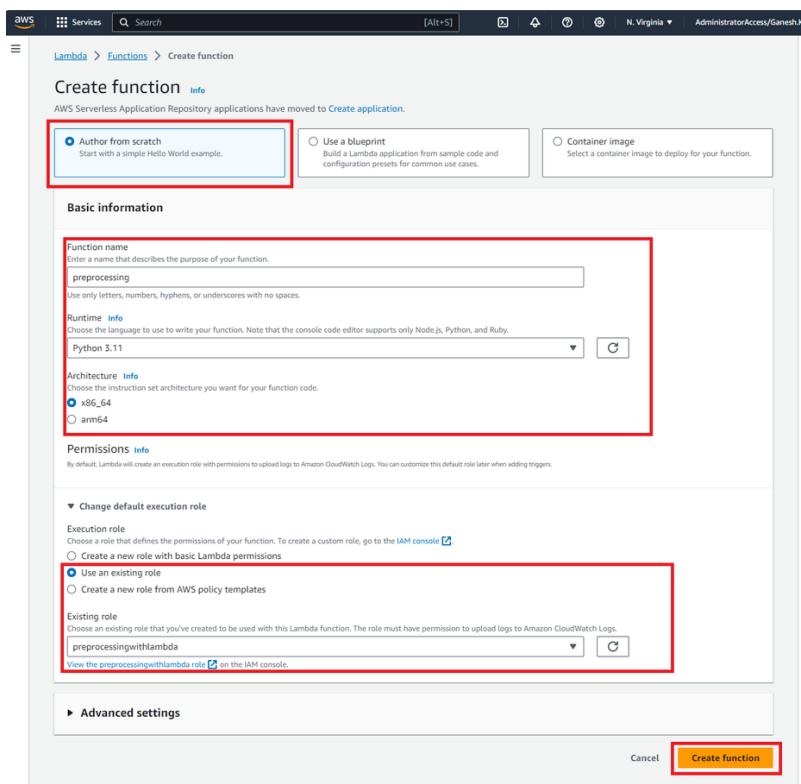
	Access AWS from your non AWS workloads Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.		X.509 Standard Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.		Temporary credentials Use temporary credentials with ease and benefit from the enhanced security they provide.
--	---	--	--	--	--

Step 4: Creating lambda function and setting trigger

1. Open the **Management Console** and select the **lambda** page
2. Select **Create Function**



3. In the creation function, choose an **author from scratch**, give a **function name**, select **runtime as python**, **architecture x86_64**, choose a role as **use an existing role** as created, then select **create function**



4. Paste the below code in the **code source** on the next page

```

1 import json
2 import boto3
3 import pandas as pd
4
5 def json2dataframe(data):
6     tags = []
7
8     for message in data:
9         if 'messages' in message:
10             for nested_message in message['messages']:
11                 if 'payload' in nested_message:

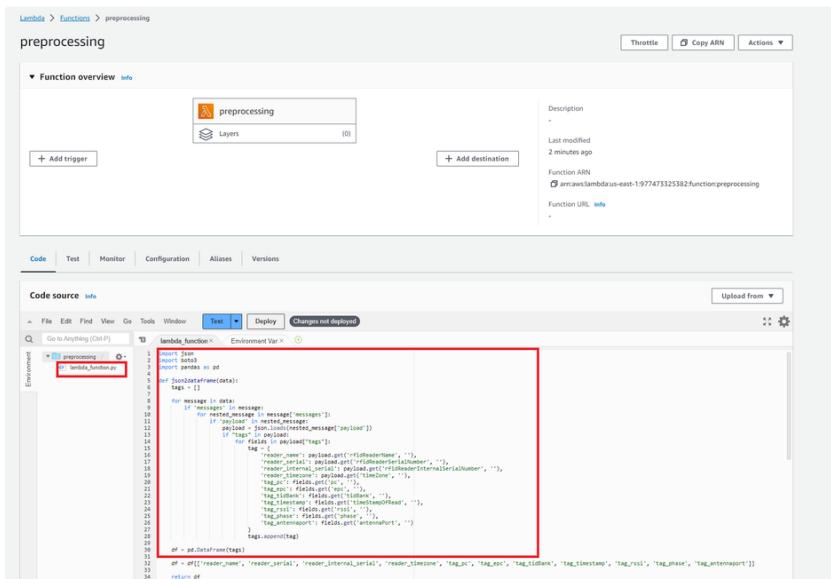
```

```

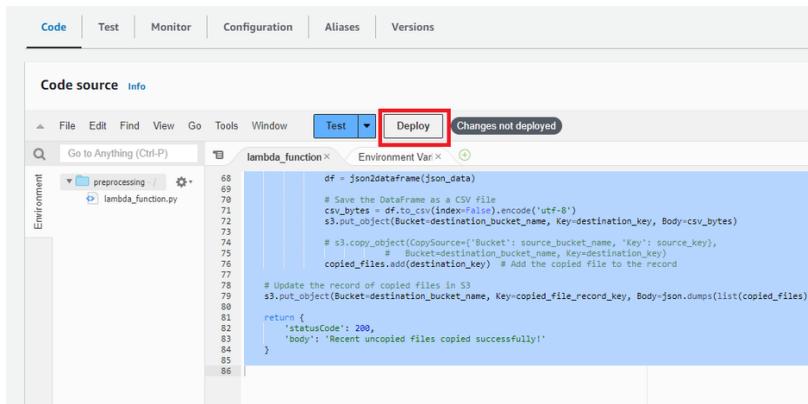
12     payload = json.loads(nested_message['payload'])
13     if "tags" in payload:
14         for fields in payload["tags"]:
15             tag = {
16                 'reader_name': payload.get('rfidReaderName', ''),
17                 'reader_serial': payload.get('rfidReaderSerialNumber', ''),
18                 'reader_internal_serial': payload.get('rfidReaderInternalSerialNumber', ''),
19                 'reader_timezone': payload.get('timeZone', ''),
20                 'tag_pc': fields.get('pc', ''),
21                 'tag_epc': fields.get('epc', ''),
22                 'tag_tidBank': fields.get('tidBank', ''),
23                 'tag_timestamp': fields.get('timeStampOfRead', ''),
24                 'tag_rssi': fields.get('rssi', ''),
25                 'tag_phase': fields.get('phase', ''),
26                 'tag_antennaport': fields.get('antennaPort', '')
27             }
28             tags.append(tag)
29
30     df = pd.DataFrame(tags)
31
32     df = df[['reader_name', 'reader_serial', 'reader_internal_serial', 'reader_timezone', 'tag_pc', 'tag_epc', 'tag_tidBank', 'tag_timestamp', 'tag_rssi', 'tag_phase', 'tag_antennaport']]
33
34     return df
35
36 def lambda_handler(event, context):
37     # Define the source and destination bucket names
38     source_bucket_name = 'bucket-inputdata'
39     destination_bucket_name = 'bucket-processeddata'
40     copied_file_record_key = 'copied_files.json' # Key to store the record of copied files
41
42     # Initialize the S3 client
43     s3 = boto3.client('s3')
44
45     # List objects in the source bucket
46     response = s3.list_objects_v2(Bucket=source_bucket_name)
47
48     # Load the record of copied files from S3 if it exists
49     copied_files = set()
50     try:
51         copied_files_obj = s3.get_object(Bucket=destination_bucket_name, Key=copied_file_record_key)
52         copied_files = set(json.loads(copied_files_obj['Body'].read().decode('utf-8')))
53     except Exception as e:
54         # If the record doesn't exist, it's okay; start with an empty set
55         pass
56
57     # Iterate through the files in the source bucket, copy only if not in the record
58     if 'Contents' in response:
59         for obj in response['Contents']:
60             source_key = obj['Key']
61             destination_key = source_key.rsplit('.', 1)[0] + '.csv'
62             if destination_key not in copied_files:
63
64                 # Read the JSON data
65                 s3_response = s3.get_object(Bucket=source_bucket_name, Key=source_key)
66                 json_data = json.loads(s3_response['Body'].read().decode('utf-8'))
67
68                 df = json2dataframe(json_data)
69
70                 # Save the DataFrame as a CSV file
71                 csv_bytes = df.to_csv(index=False).encode('utf-8')
72                 s3.put_object(Bucket=destination_bucket_name, Key=destination_key, Body=csv_bytes)
73
74                 # s3.copy_object(CopySource={'Bucket': source_bucket_name, 'Key': source_key},
75                 #                 # Bucket=destination_bucket_name, Key=destination_key)
76                 copied_files.add(destination_key) # Add the copied file to the record
77
78     # Update the record of copied files in S3
79     s3.put_object(Bucket=destination_bucket_name, Key=copied_file_record_key, Body=json.dumps(list(copied_files)))
80
81     return {
82         'statusCode': 200,
83         'body': 'Recent uncopied files copied successfully!'

```

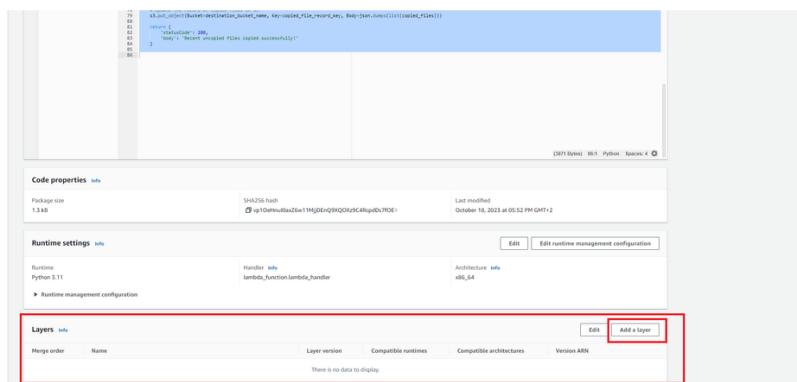
84 }
85
86



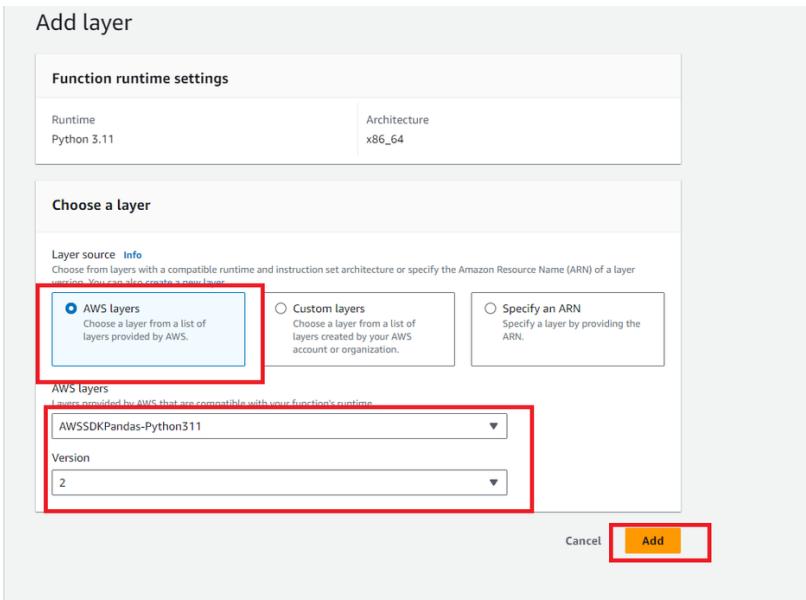
5. Once the code is pasted, click Deploy



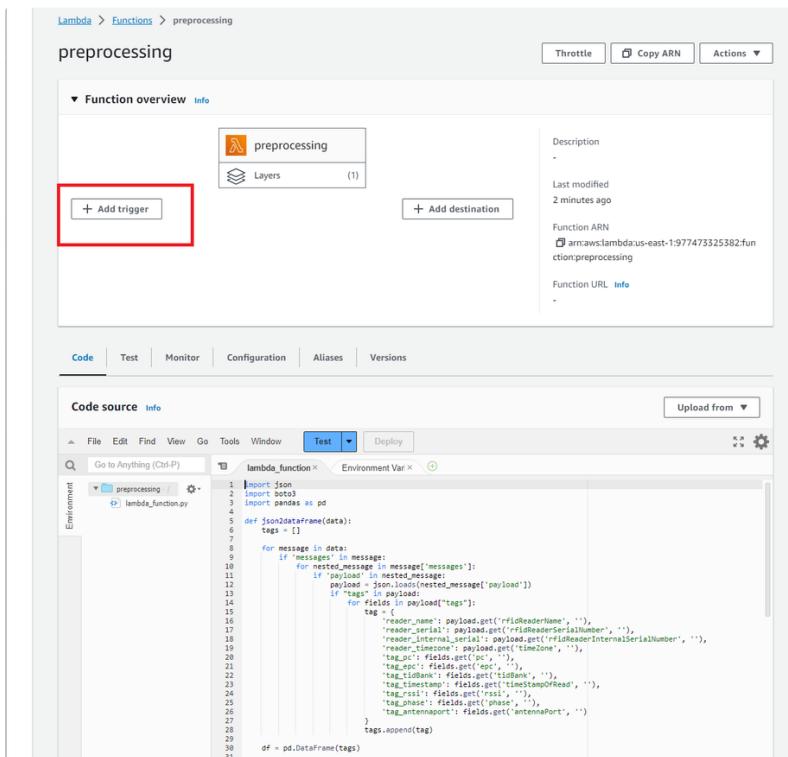
6. Now scroll down to the layers, click add layer



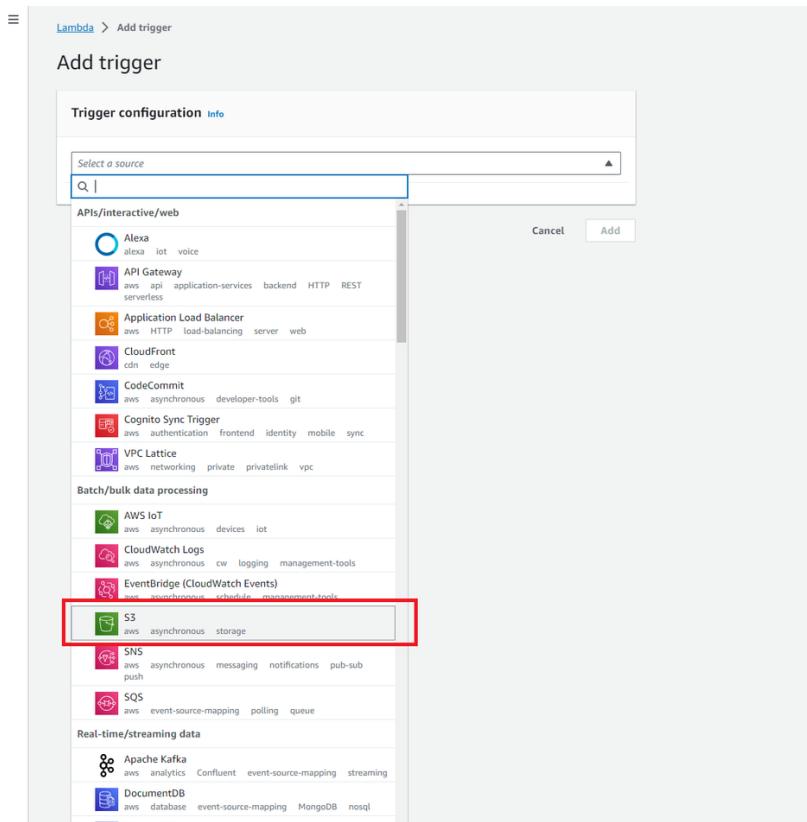
7. In the layer page, select pandas and version, then select add



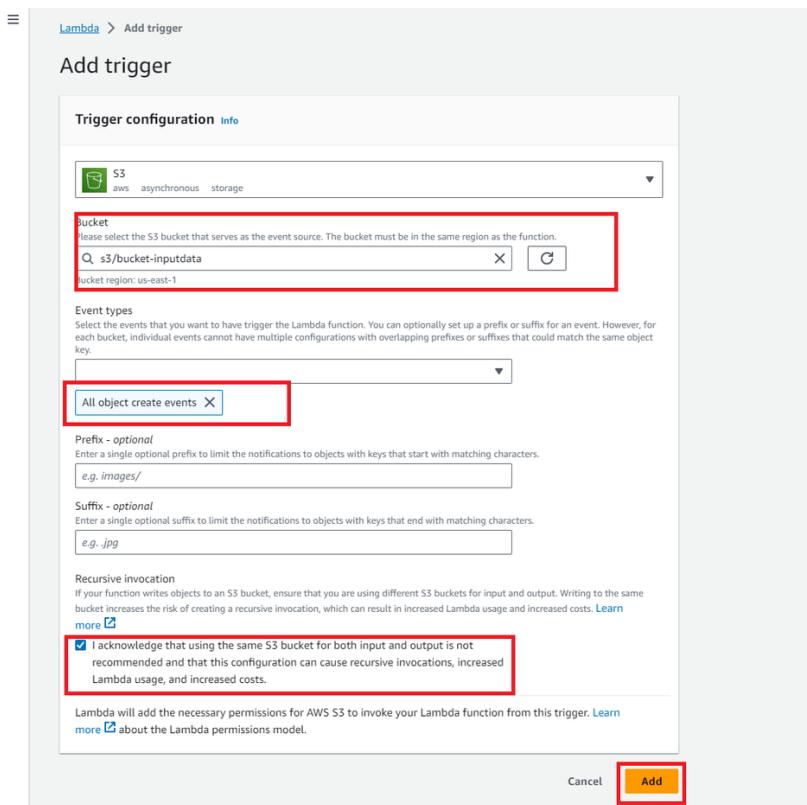
8. Now in the **function overview** click **Add trigger**



9. Select the source **S3** on the next page then click **next**



10. On the next page, select the **input bucket**, check the **acknowledge** then select **Add**



11. Goto configuration tab on the lambda function page and choose edit in General configuration as shown below

Lambda > Functions > preprocessing

preprocessing

Throttle Copy ARN Actions ▾

Function overview Info

Layers (1)

S3 + Add destination

+ Add trigger

Description -

Last modified 8 minutes ago

Function ARN arn:aws:lambda:us-east-1:977473325382:function:preprocessing

Function URL Info -

Code Test Monitor Configuration Aliases Versions

General configuration Info Edit

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

Monitoring and operations tools

Concurrency

Description -

Memory 128 MB

Ephemeral storage 512 MB

Timeout 1 min 3 sec

SnapStart Info None

Lambda > Functions > preprocessing > Edit basic settings

Edit basic settings

Basic settings Info

Description - optional

Memory Info Your function is allocated CPU proportional to the memory configured.

128 MB Set memory to between 128 MB and 10240 MB

Ephemeral storage Info You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing

512 MB Set ephemeral storage (/tmp) to between 512 MB and 10240 MB

SnapStart Info Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations

None

Supported runtimes: Java 11, Java 17.

Timeout 1 min 3 sec

Execution role Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console

Use an existing role

Create a new role from AWS policy templates

Existing role Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

preprocessingwithlambda

View the preprocessingwithlambda role on the IAM console.

Cancel Save

12. After changing the time click save

If we don't change in case the execution will take more time than specified the function will stop due to timeout

Checking

1. Now upload a JSON file in the bucket-inputdata

Amazon S3 > Buckets > bucket-inputdata

bucket-inputdata [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

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](#)

C	Copy S3 URI	Copy URL	Download	Open	Delete	Actions ▾	Create folder	Upload
<input type="text" value="Find objects by prefix"/>								
< 1 > @								
<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class			
<input type="checkbox"/>	Cape Cod Test Data 1.json	json	October 18, 2023, 18:08:52 (UTC+02:00)	505.2 KB	Standard			

2. You should be able to see the same file in the CSV format and a copied_files.json that stores the files that have been processed to avoid duplication.

Amazon S3 > Buckets > bucket-processeddata

bucket-processeddata [Info](#)

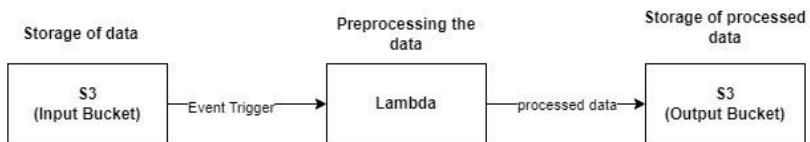
[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (2)
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](#)

C	Copy S3 URI	Copy URL	Download	Open	Delete	Actions ▾	Create folder	Upload
<input type="text" value="Find objects by prefix"/>								
< 1 > @								
<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class			
<input type="checkbox"/>	Cape Cod Test Data 1.csv	csv	October 18, 2023, 18:09:00 (UTC+02:00)	142.4 KB	Standard			
<input type="checkbox"/>	copied_files.json	json	October 18, 2023, 18:09:00 (UTC+02:00)	28.0 B	Standard			

3. In case it didn't give the required output see the error in Cloudwatch (the code given in the document works fine)

Advosense Pipeline in AWS



Improvisation of lambda code

This session discusses the improvement of the lambda function code. The previous code only converted the data from JSON format to CSV format. However, the updated code adds an extra step by converting the timestamps into the proper format of date and time in a separate column. To incorporate the improved functionality, replace the code in STEP 4, the 4th procedure, with the following code. If you wish to retain the old functionality of just data conversion, set the variable "`preprocess_data`" to "`False`" on line 104 of the code below. The code lines 64 and 65 remove the column which contains all values as "`NAN`".

```

1 import json
2 import boto3
3 import pandas as pd
4 import datetime
5 import pytz
6
7 # Function to convert microseconds to datetime objects
8 def convert_microseconds_to_datetime(timestamp_microseconds):
9     """
10     Converts a timestamp in microseconds to a datetime object in a specified timezone.
11
12     Args:
13         timestamp_microseconds (int): A timestamp in microseconds.
14
15     Returns:

```

```

16     str: A datetime string in the format 'YYYY-MM-DD HH:MM:SS' with the desired timezone applied.
17         Microseconds are removed.
18
19
20 """
21
22 timestamp_seconds = int(timestamp_microseconds) / 1_000_000
23 timestamp_datetime = datetime.datetime.fromtimestamp(timestamp_seconds)
24 desired_timezone = pytz.timezone('US/Eastern') # Replace with your desired time zone, currently it's -5
25 timestamp_datetime = timestamp_datetime.astimezone(desired_timezone)
26 timestamp_datetime = timestamp_datetime.replace(microsecond=0) # Remove microseconds
27 timestamp_datetime = timestamp_datetime.strftime('%Y-%m-%d %H:%M:%S') # Format without milliseconds
28 return timestamp_datetime
29
30 # Function to split date and time
31 def formatting_datetime(df, remove_microseconds = True, split_datetime = True):
32 """
33 Formats a DataFrame with timestamp data, optionally removing microseconds and
34 splitting datetime into separate Date and Time columns.
35
36 Args:
37     df (DataFrame): A DataFrame containing a 'tag_timestamp' column with timestamp data.
38     remove_microseconds (bool, optional): A boolean flag to remove microseconds
39             from the timestamp (default is True).
40     split_datetime (bool, optional): A boolean flag to split the timestamp into separate
41             'Date' and 'Time' columns (default is True).
42
43 Returns:
44     DataFrame: A modified DataFrame with 'Date' and 'Time' columns based on the specified flags.
45
46
47 """
48
49 if remove_microseconds:
50     # Apply the conversion function to the DataFrame column
51     df['tag_timestamp'] = df['tag_timestamp'].apply(convert_microseconds_to_datetime)
52 if split_datetime:
53     # Split 'Data and Time' column into 'Date' and 'Time' columns
54     df[['Date', 'Time']] = df['tag_timestamp'].str.split(pat=' ', n=1, expand=True)
55
56     # Drop the original 'Data and Time' column
57     df.drop(columns=['tag_timestamp'], inplace=True)
58
59     # Reorder columns with 'Date' and 'Time' at the start
60     new_column_order = ['Date', 'Time'] + [col for col in df.columns if col not in ['Date', 'Time']]
61     df = df[new_column_order]
62     df['Time'] = df['Time'].str.split('.').str[0].str[:8]
63
64 dropped_columns = df.columns[df.isnull().all()]
65 df = df.dropna(axis=1, how='all')
66 print("\nDropped Columns:")
67 print(dropped_columns)
68 return df
69
70
71 #Function to convert data from
72 #JSON to CSV
73 def json2dataframe(data):
74     tags = []
75
76     for message in data:
77         if 'messages' in message:
78             for nested_message in message['messages']:
79                 if 'payload' in nested_message:
80                     payload = json.loads(nested_message['payload'])
81                     if "tags" in payload:
82                         for fields in payload["tags"]:
83                             tag = {
84                                 'reader_name': payload.get('rfidReaderName', ''),
85                                 'reader_serial': payload.get('rfidReaderSerialNumber', ''),
86                                 'reader_internal_serial': payload.get('rfidReaderInternalSerialNumber', ''),
87                                 'reader_timezone': payload.get('timeZone', ''),

```

```

88         'tag_pc': fields.get('pc', ''),
89         'tag_epc': fields.get('epc', ''),
90         'tag_tidBank': fields.get('tidBank', ''),
91         'tag_timestamp': fields.get('timeStampOfRead', ''),
92         'tag_rssi': fields.get('rssI', ''),
93         'tag_phase': fields.get('phase', ''),
94         'tag_antennaport': fields.get('antennaPort', '')
95     }
96     tags.append(tag)
97
98 df = pd.DataFrame(tags)
99
100 df = df[['reader_name', 'reader_serial', 'reader_internal_serial', 'reader_timezone', 'tag_pc', 'tag_epc', 'tag_tidBank', 'tag_timestamp'
101
102 return df
103
104 def lambda_handler(event, context):
105     # Define the source and destination bucket names
106     source_bucket_name = 'bucket-inputdata'
107     destination_bucket_name = 'bucket-processedata'
108     copied_file_record_key = 'copied_files.json' # Key to store the record of copied files
109     preprocess_data = True
110
111     # Initialize the S3 client
112     s3 = boto3.client('s3')
113
114     # List objects in the source bucket
115     response = s3.list_objects_v2(Bucket=source_bucket_name)
116
117     # Load the record of copied files from S3 if it exists
118     copied_files = set()
119
120     try:
121         copied_files_obj = s3.get_object(Bucket=destination_bucket_name, Key=copied_file_record_key)
122         copied_files = set(json.loads(copied_files_obj['Body'].read().decode('utf-8')))
123     except Exception as e:
124         # If the record doesn't exist, it's okay; start with an empty set
125         pass
126
127     # Iterate through the files in the source bucket, copy only if not in the record
128     if 'Contents' in response:
129         for obj in response['Contents']:
130             source_key = obj['Key']
131             destination_key = source_key.rsplit('.', 1)[0] + '.csv'
132             if destination_key not in copied_files:
133
134                 # Read the JSON data
135                 s3_response = s3.get_object(Bucket=source_bucket_name, Key=source_key)
136                 json_data = json.loads(s3_response['Body'].read().decode('utf-8'))
137
138                 if preprocess_data:
139                     df = json2dataframe(json_data)
140                     df = formatting_datetime(df)
141                 else:
142                     df = json2dataframe(json_data)
143
144                 # Save the DataFrame as a CSV file
145                 csv_bytes = df.to_csv(index=False).encode('utf-8')
146                 s3.put_object(Bucket=destination_bucket_name, Key=destination_key, Body=csv_bytes)
147
148                 # s3.copy_object(CopySource={'Bucket': source_bucket_name, 'Key': source_key},
149                             #           Bucket=destination_bucket_name, Key=destination_key)
150                 copied_files.add(destination_key) # Add the copied file to the record
151
152     # Update the record of copied files in S3
153     s3.put_object(Bucket=destination_bucket_name, Key=copied_file_record_key, Body=json.dumps(list(copied_files)))
154
155     return {
156         'statusCode': 200,
157         'body': 'Recent uncopied files copied successfully!'
158     }
159

```

