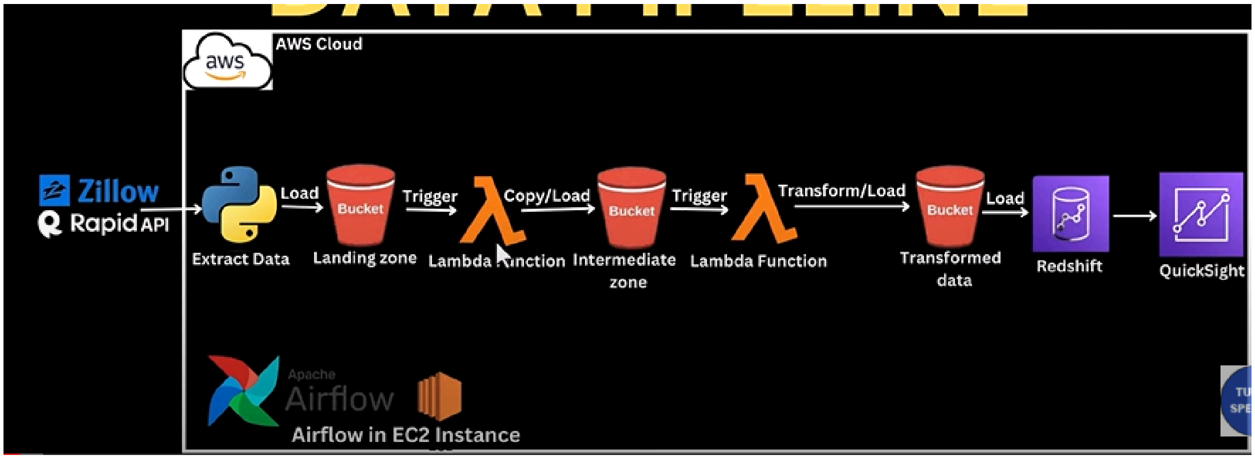
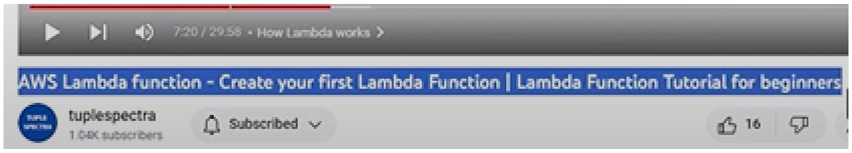
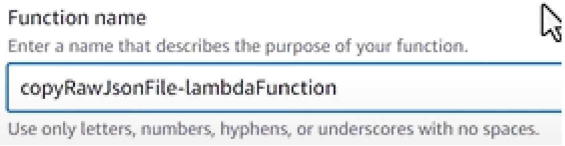
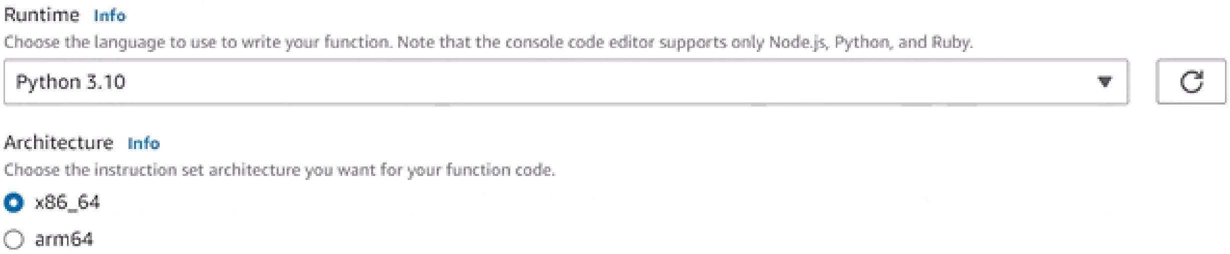
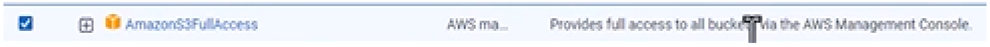
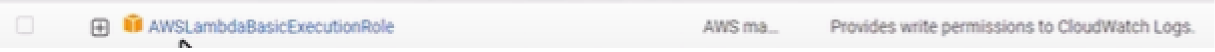
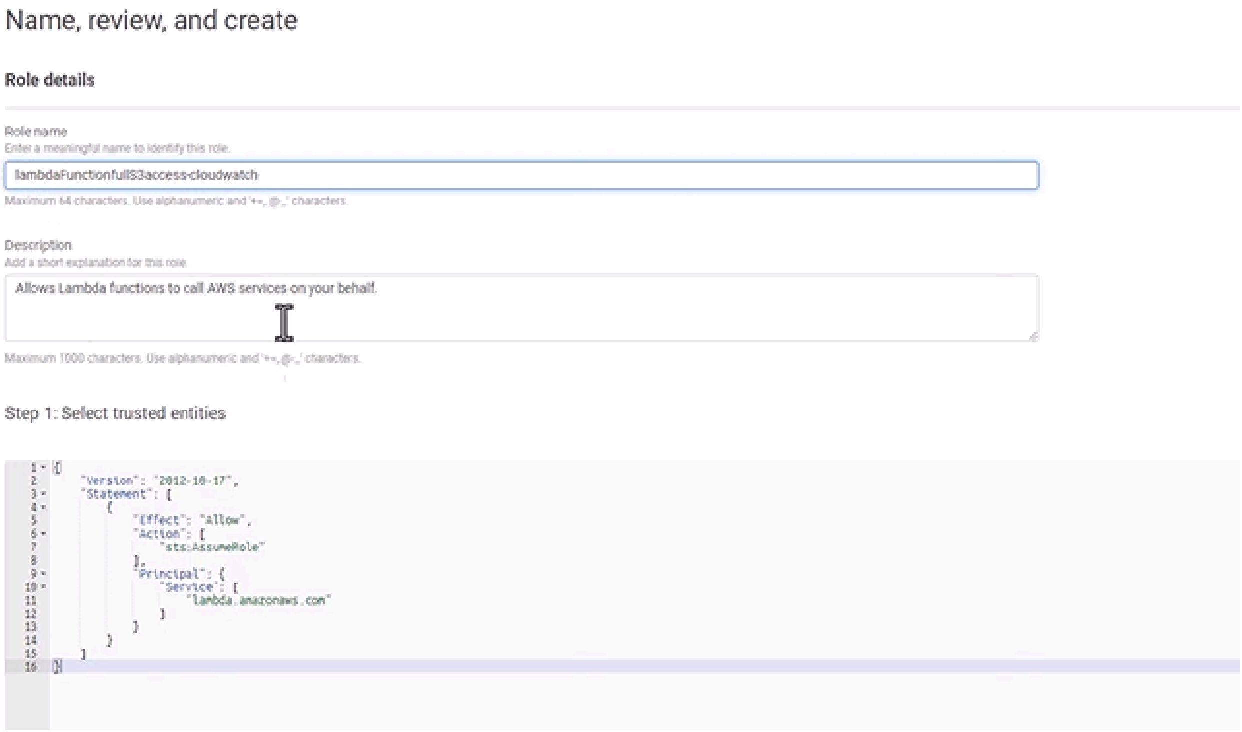
Landing Zone - Intermediate zone

1. 

2. Here we have extracted data from api and copies into initial s3 bucket
3. Our task is trigger a lambda function and load this data to intermediate zone
4. Now head to AWS and search for AWS lambda
5. Lambda tutorial : 
   7:20 / 29:58 • How Lambda works >
   AWS Lambda function - Create your first Lambda Function | Lambda Function Tutorial for beginners
   tuplespectra
   1.04K subscribers
   Subscribed
   16
6. Now we will create a lambda function 
   Function name
   Enter a name that describes the purpose of your function.
   copyRawJsonFile-lambdaFunction
   Use only letters, numbers, hyphens, or underscores with no spaces.
   
   Runtime Info
   Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
   Python 3.10
   Architecture Info
   Choose the instruction set architecture you want for your function code.
   ● x86_64
   ○ arm64
7. Next we will create a role for our lambda function …we will our s3 access to our lambda role 
   ☑ ⊞ AmazonS3FullAccess AWS ma... Provides full access to all buck… via the AWS Management Console.
8. We also need to give this policy to the role
   
   □ ⊞ AWSLambdaBasicExecutionRole AWS ma... Provides write permissions to CloudWatch Logs.
9. So whatever is happening with our lambda function..we can watch them in cloudwatch

10. Next we give a name to our role



Name, review, and create

Role details

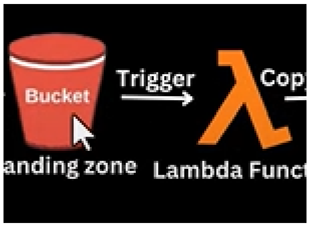
Role name
Enter a meaningful name to identify this role.
lambdaFunctionfullS3access-cloudwatch
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

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.

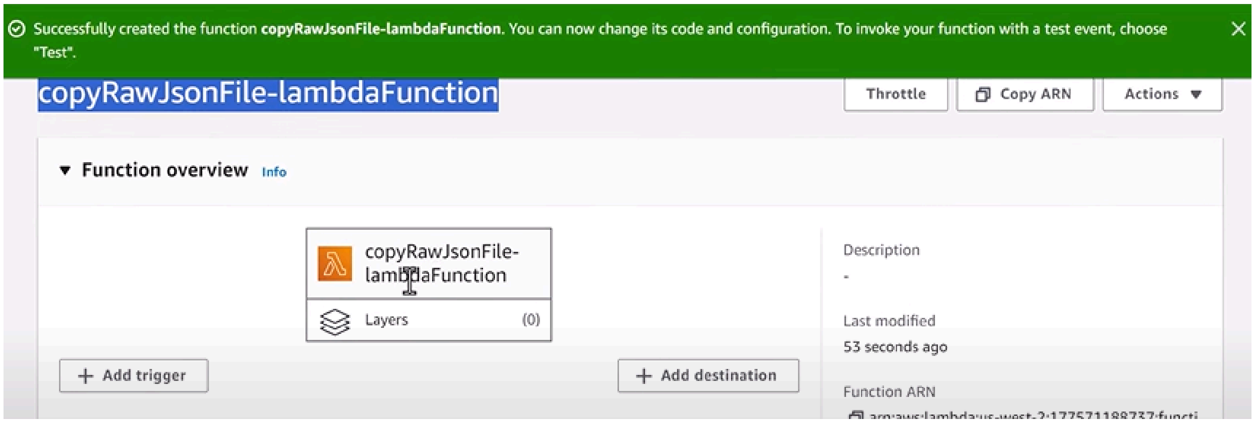
Step 1: Select trusted entities

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "lambda.amazonaws.com"
                ]
            }
        }
    ]
}
```

11. Next we will attach our role to the lambda function..and we create the lambda function
12. So now our landing_zone bucket needs to trigger the lambda..when ever there is data in it



13. Here we can add trigger to our lambda function



Successfully created the function **copyRawJsonFile-lambdaFunction**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

copyRawJsonFile-lambdaFunction

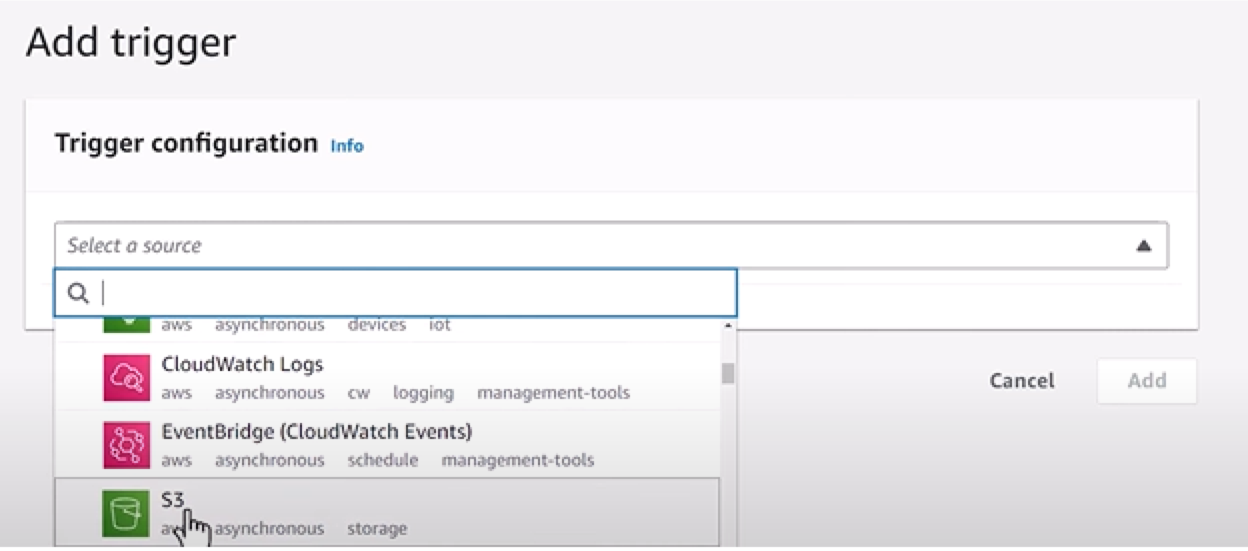
▼ Function overview Info

Description
-

Last modified
53 seconds ago

Function ARN
arn:aws:lambda:us-west-2:177571188737:functi

14. Then we select source as s3



Add trigger

Trigger configuration Info

Select a source

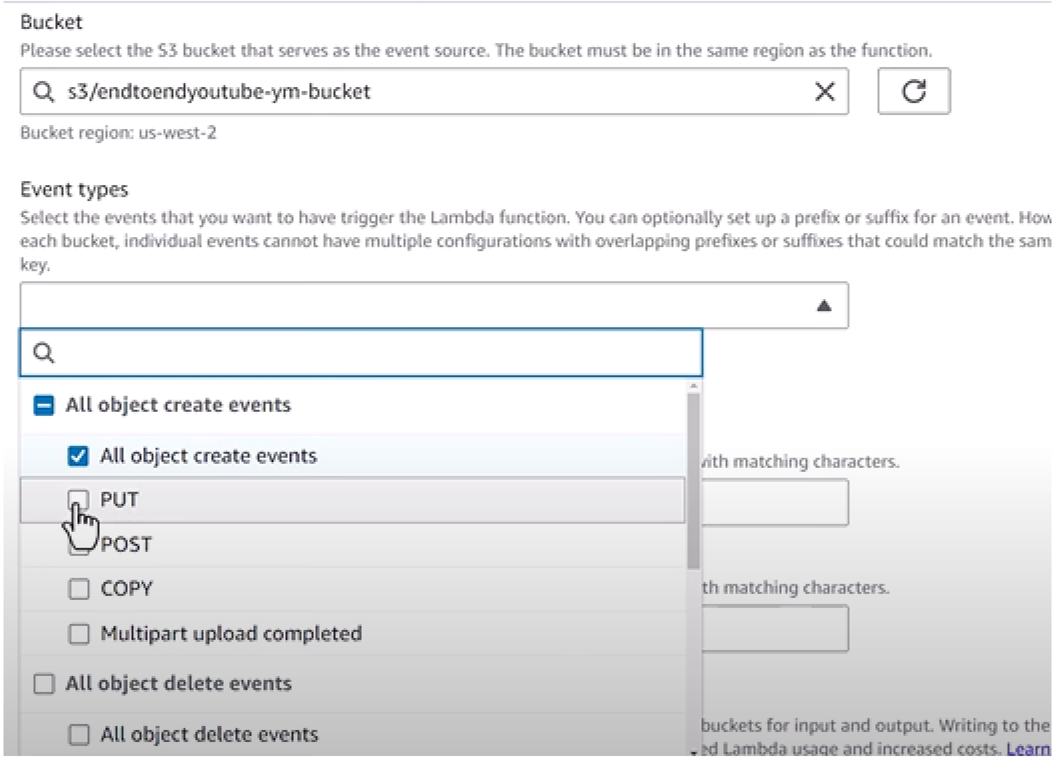
aws asynchronous devices iot

CloudWatch Logs
aws asynchronous cw logging management-tools

EventBridge (CloudWatch Events)
aws asynchronous schedule management-tools

S3
aws asynchronous storage

15. Next we have to choose the bucket and the event_type in the bucket



Bucket

Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

s3/endtoendyoutube-ym-bucket

Bucket region: us-west-2

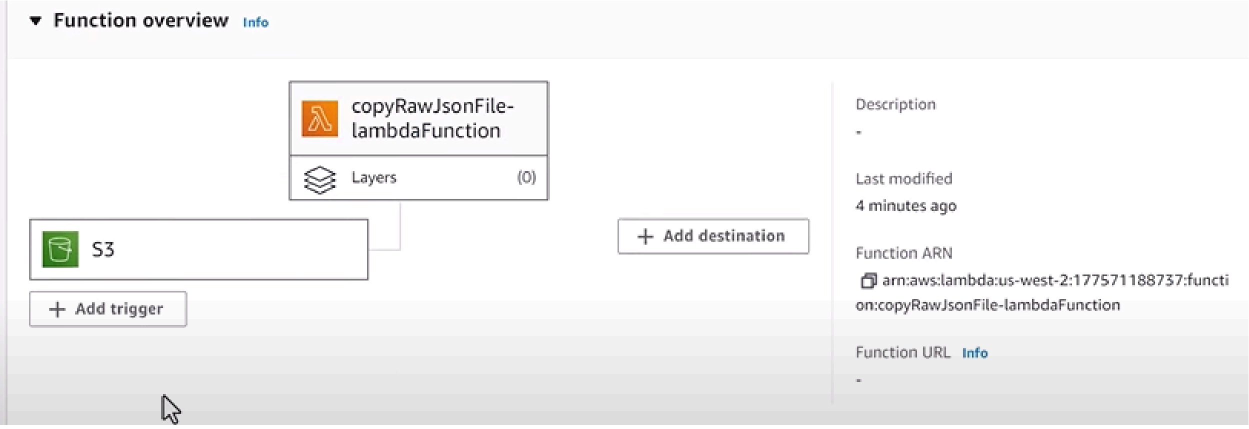
Event types

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. How each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the sam key.

- ☐ All object create events
  - ☑ All object create events
  - ☐ PUT
  - ☐ POST
  - ☐ COPY
  - ☐ Multipart upload completed
- ☐ All object delete events
  - ☐ All object delete events

with matching characters.
th matching characters.
buckets for input and output. Writing to the
ed Lambda usage and increased costs. Learn

16. Here we can see .we’ve added trigger to our lambda function



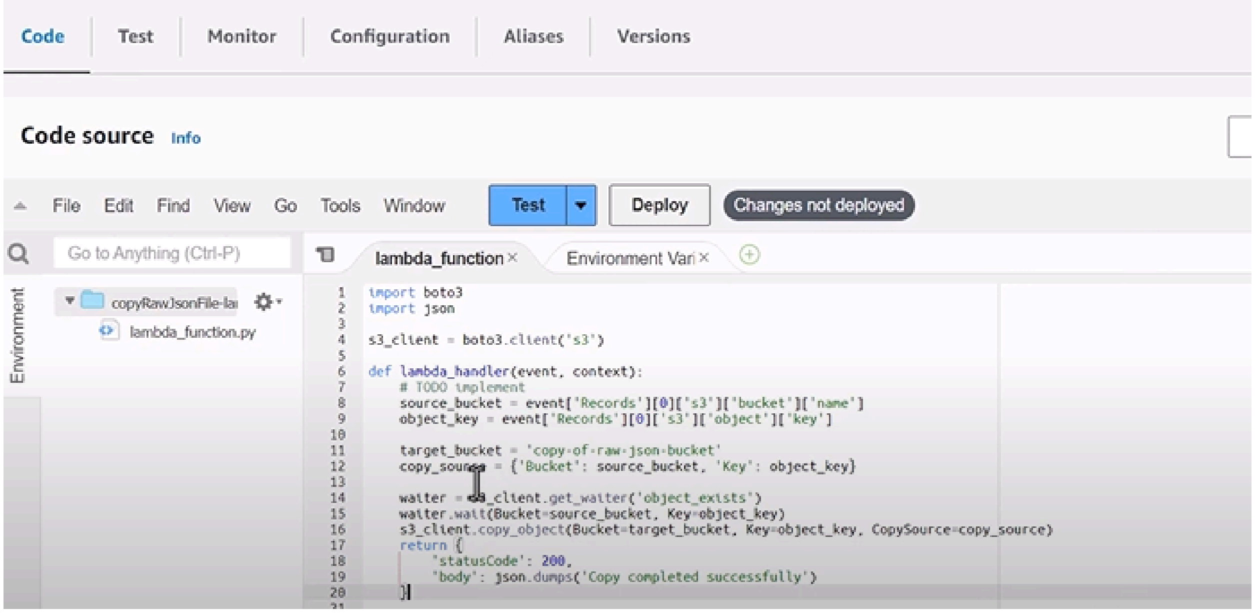
Description
-

Last modified
4 minutes ago

Function ARN
arn:aws:lambda:us-west-2:177571188737:function:copyRawJsonFile-lambdaFunction

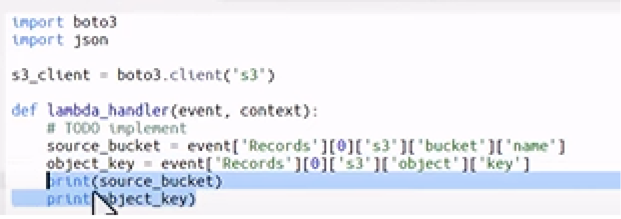
Function URL
-

17. Next we have coded our lambda function



```python
import boto3
import json

s3_client = boto3.client('s3')

def lambda_handler(event, context):
    # TODO implement
    source_bucket = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']

    target_bucket = 'copy-of-raw-json-bucket'
    copy_source = {'Bucket': source_bucket, 'Key': object_key}

    waiter = s3_client.get_waiter('object_exists')
    waiter.wait(Bucket=source_bucket, Key=object_key)
    s3_client.copy_object(Bucket=target_bucket, Key=object_key, CopySource=copy_source)
    return {
        'statusCode': 200,
        'body': json.dumps('Copy completed successfully')
    }
```

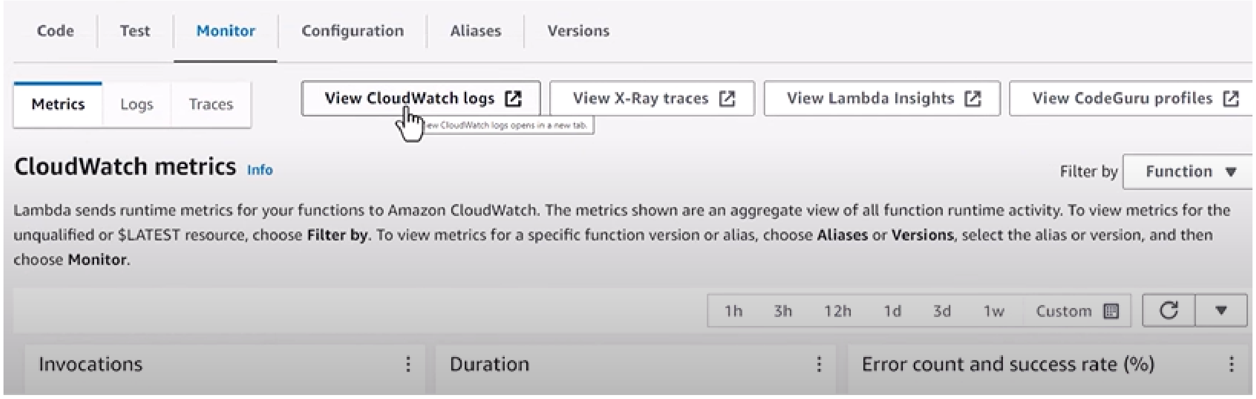
18. Later to test the code..we have uploaded a sample file to pur s3 bucket..now as soon something lands in our s3 bucket..our lambda gets triggered



```python
import boto3
import json

s3_client = boto3.client('s3')

def lambda_handler(event, context):
    # TODO implement
    source_bucket = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']
    print(source_bucket)
    print(object_key)
```

19. So now we have to get our source_bucket name and object

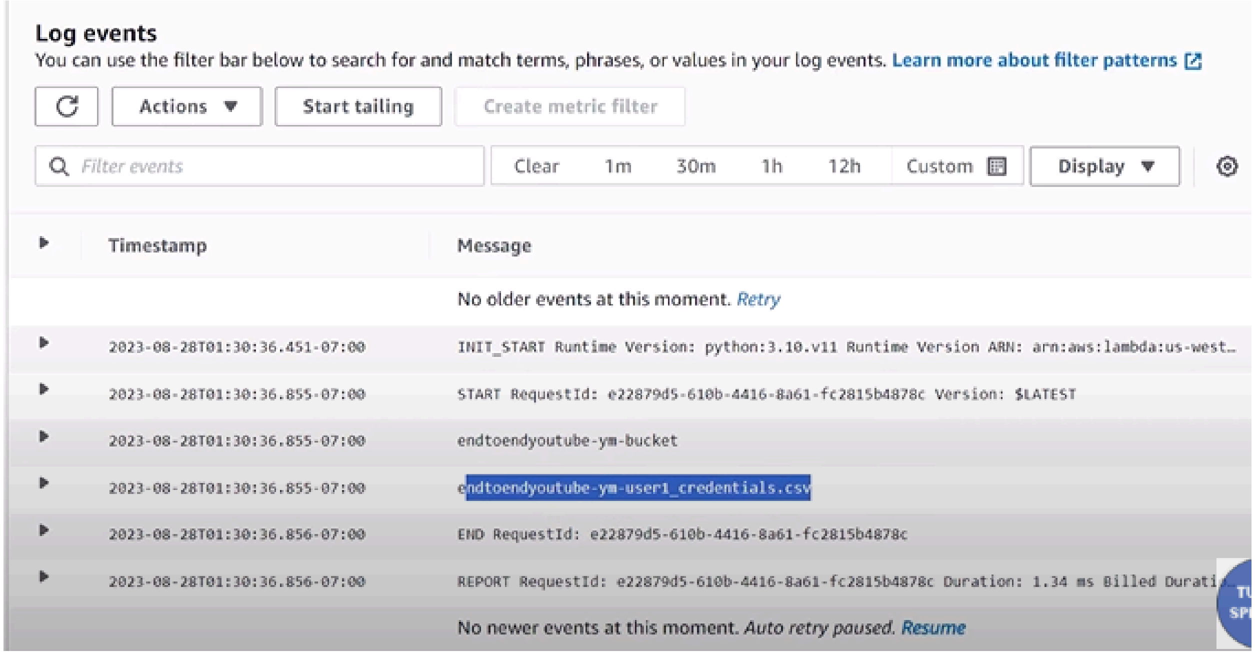
20. To view them we go to monitor=>cloudwatch



CloudWatch metrics Info

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select the alias or version, and then choose **Monitor**.

21. Here we can see our bucket name and file name(that we have uploaded)



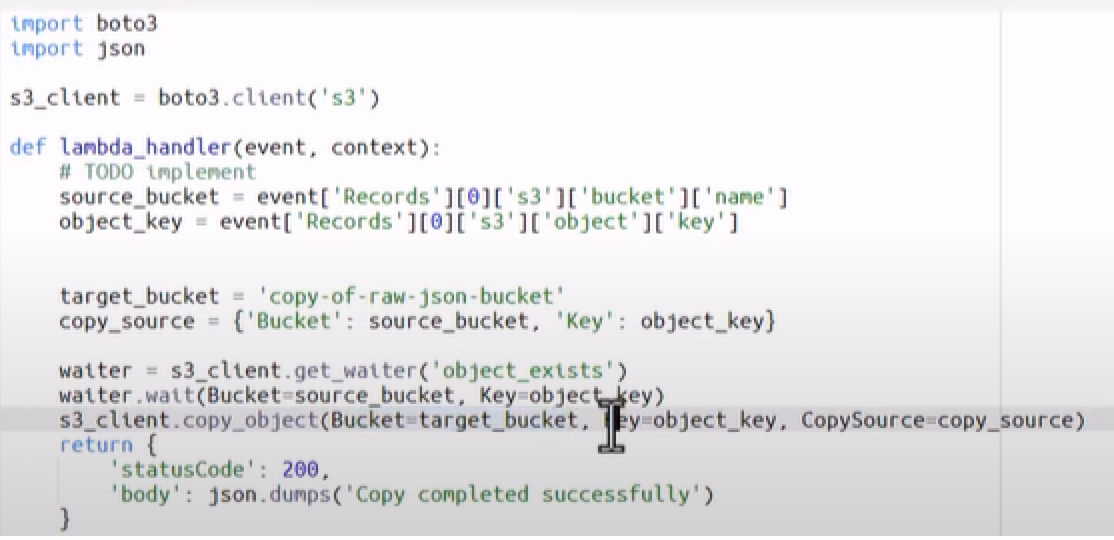
Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                        Message
                                 No older events at this moment. Retry
2023-08-28T01:30:36.451-07:00    INIT_START Runtime Version: python:3.10.v11 Runtime Version ARN: arn:aws:lambda:us-west…
2023-08-28T01:30:36.855-07:00    START RequestId: e22879d5-610b-4416-8a61-fc2815b4878c Version: $LATEST
2023-08-28T01:30:36.855-07:00    endtoendyoutube-ym-bucket
2023-08-28T01:30:36.855-07:00    endtoendyoutube-ym-user1_credentials.csv
2023-08-28T01:30:36.856-07:00    END RequestId: e22879d5-610b-4416-8a61-fc2815b4878c
2023-08-28T01:30:36.856-07:00    REPORT RequestId: e22879d5-610b-4416-8a61-fc2815b4878c Duration: 1.34 ms Billed Durati
                                 No newer events at this moment. Auto retry paused. Resume
```

22. Basically what our code is doing is…getting the source_bucket and the files uploaded to source bucket

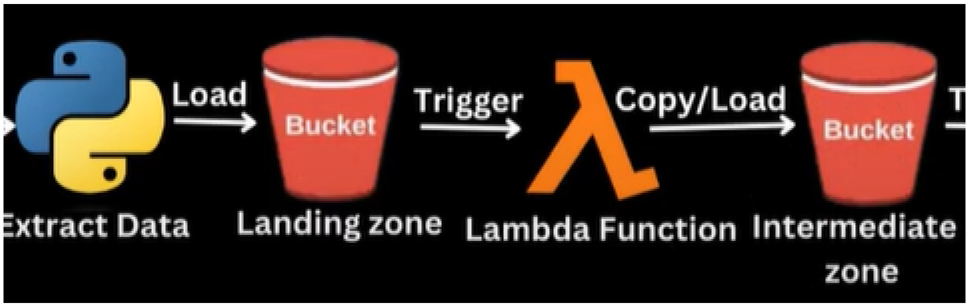
23. And copying this data to the target bucket(intermediate bucket)



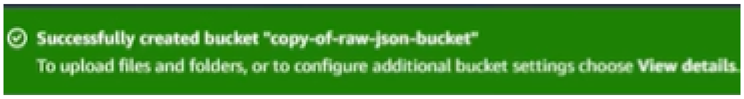
```
import boto3
import json

s3_client = boto3.client('s3')

def lambda_handler(event, context):
    # TODO implement
    source_bucket = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']


    target_bucket = 'copy-of-raw-json-bucket'
    copy_source = {'Bucket': source_bucket, 'Key': object_key}

    waiter = s3_client.get_waiter('object_exists')
    waiter.wait(Bucket=source_bucket, Key=object_key)
    s3_client.copy_object(Bucket=target_bucket, Key=object_key, CopySource=copy_source)
    return {
        'statusCode': 200,
        'body': json.dumps('Copy completed successfully')
    }
```

Explained : https://g.co/gemini/share/dbbb4f0dc7bc

24. But before copying we use “waiter” to confirm the entire file has been uploaded to source code and then we make copy
25. Now let us create the intermediate bucket

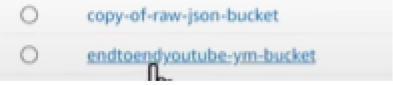


26. Here we have create a bucket



Successfully created bucket "copy-of-raw-json-bucket"
To upload files and folders, or to configure additional bucket settings choose **View details**.

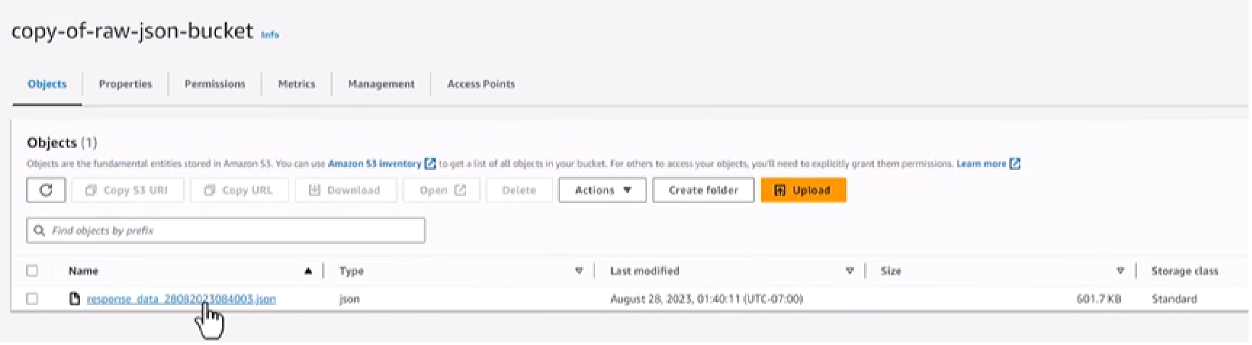
27. Now we need to send data from end_to_end bucket to raw bucket



○ copy-of-raw-json-bucket
○ endtoendyoutube-ym-bucket

28. Lets us trigger our DAG and check what happend

29. Using our DAG we again loaded data to our initial bucket(end2end) and the lambda got triggered and copied the data to raw_bucket



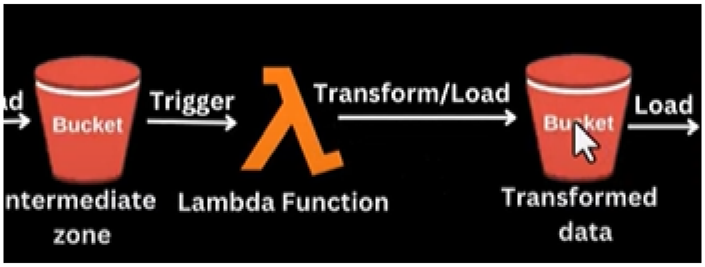
copy-of-raw-json-bucket

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

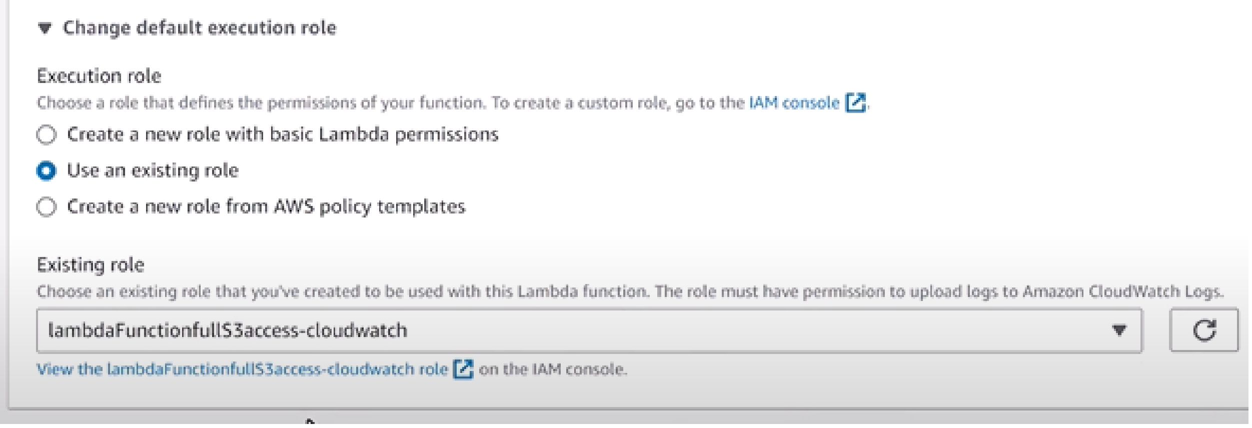
□	Name	Type	Last modified	Size	Storage class
□	response_data_28082023084003.json	json	August 28, 2023, 01:40:11 (UTC-07:00)	601.7 KB	Standard

30. Now we need to trigger another lambda function that gets triggered when the data is in intermediate zone and transforms and load to another bucket



Intermediate - Transformed

1. Now we need to write another lambda function..and while creating that function..we will attach the role which we created earlier



▼ Change default execution role

Execution role

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

○ Create a new role with basic Lambda permissions

● Use an existing role

○ Create a new role from AWS policy templates

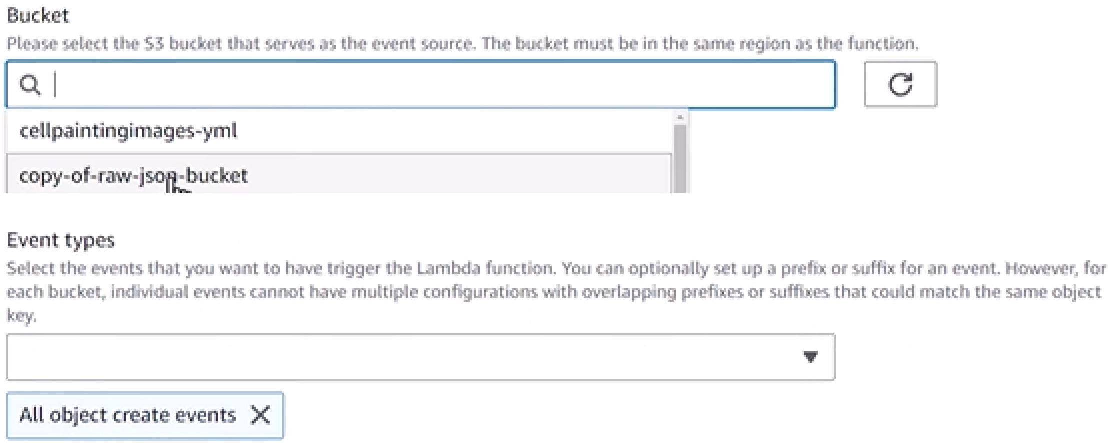
Existing role

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.

lambdaFunctionfullS3access-cloudwatch ▼

View the lambdaFunctionfullS3access-cloudwatch role on the IAM console.

2. Next we will add trigger ..next we'll choose trigger source as s3



Bucket

Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

cellpaintingimages-yml

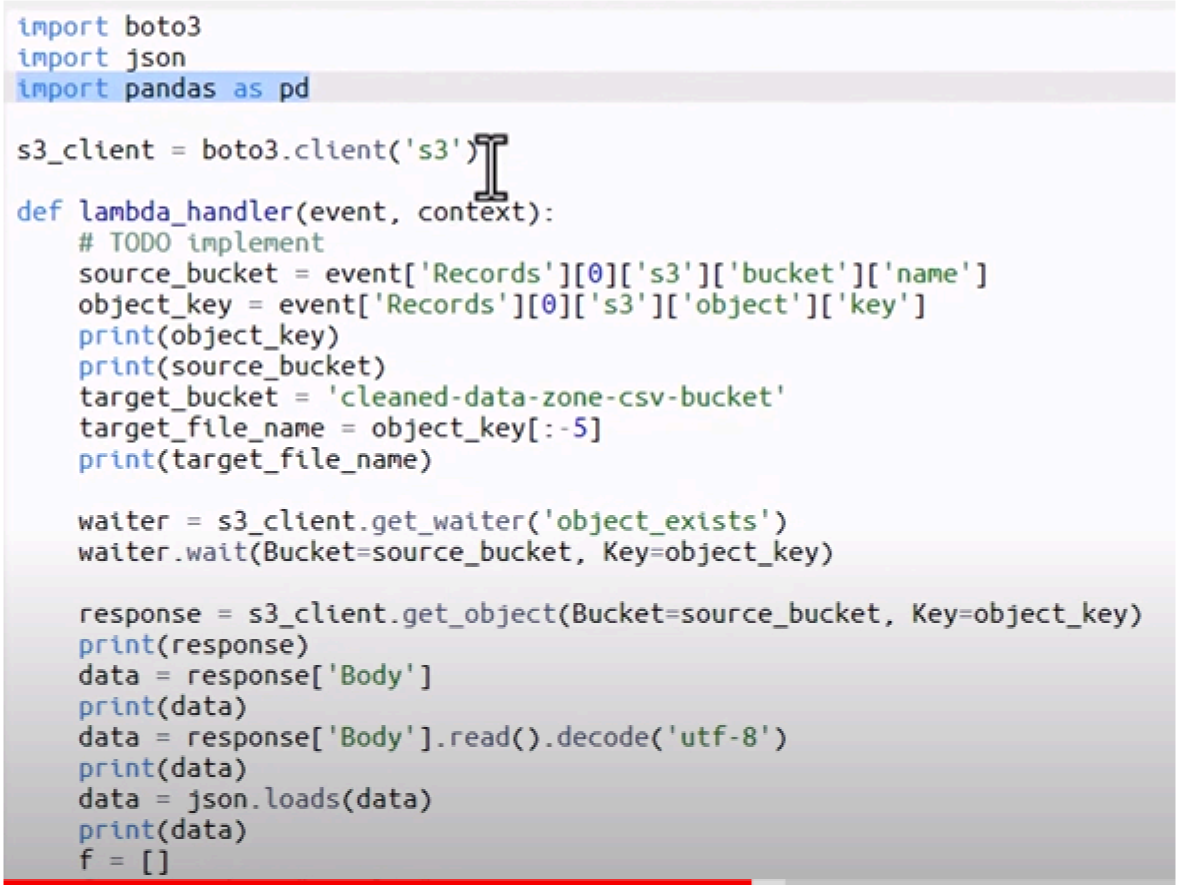
copy-of-raw-json-bucket

Event types

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events ×

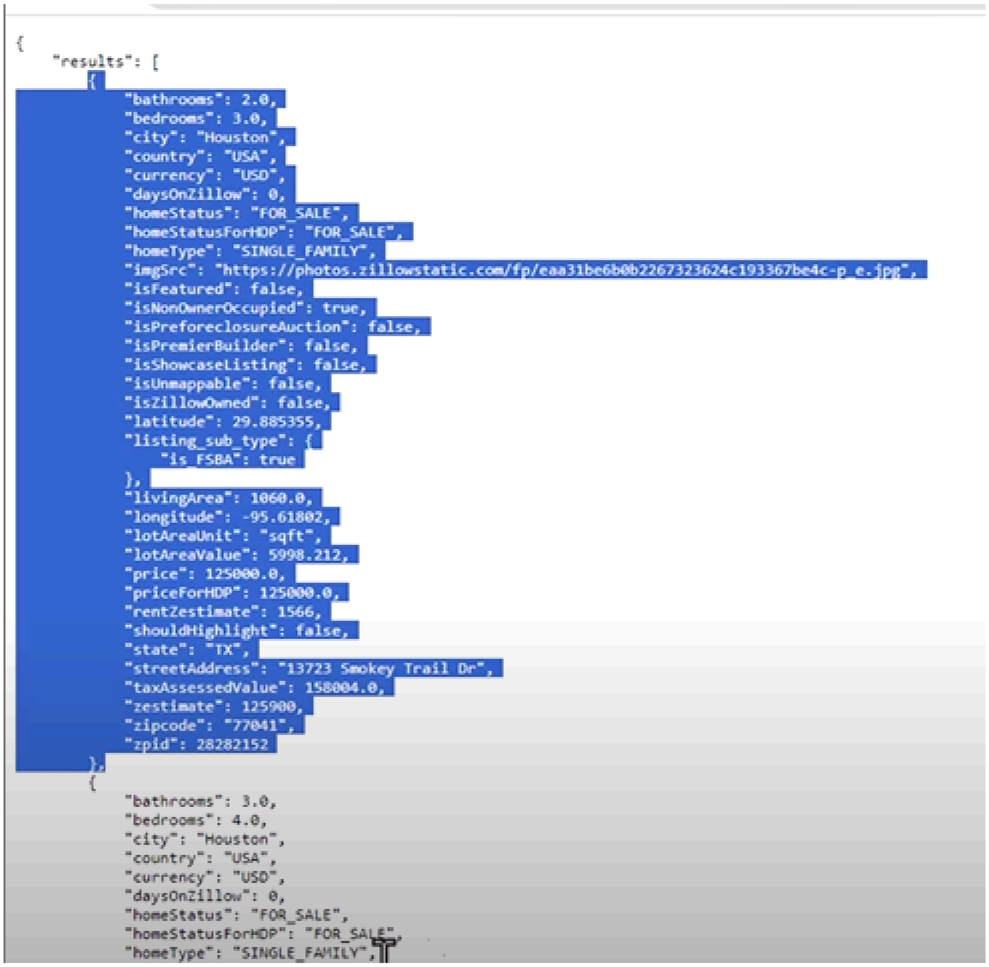
3. Now Lets focus on the lambda code



```
import boto3
import json
import pandas as pd

s3_client = boto3.client('s3')

def lambda_handler(event, context):
    # TODO implement
    source_bucket = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']
    print(object_key)
    print(source_bucket)
    target_bucket = 'cleaned-data-zone-csv-bucket'
    target_file_name = object_key[:-5]
    print(target_file_name)

    waiter = s3_client.get_waiter('object_exists')
    waiter.wait(Bucket=source_bucket, Key=object_key)

    response = s3_client.get_object(Bucket=source_bucket, Key=object_key)
    print(response)
    data = response['Body']
    print(data)
    data = response['Body'].read().decode('utf-8')
    print(data)
    data = json.loads(data)
    print(data)
    f = []
```

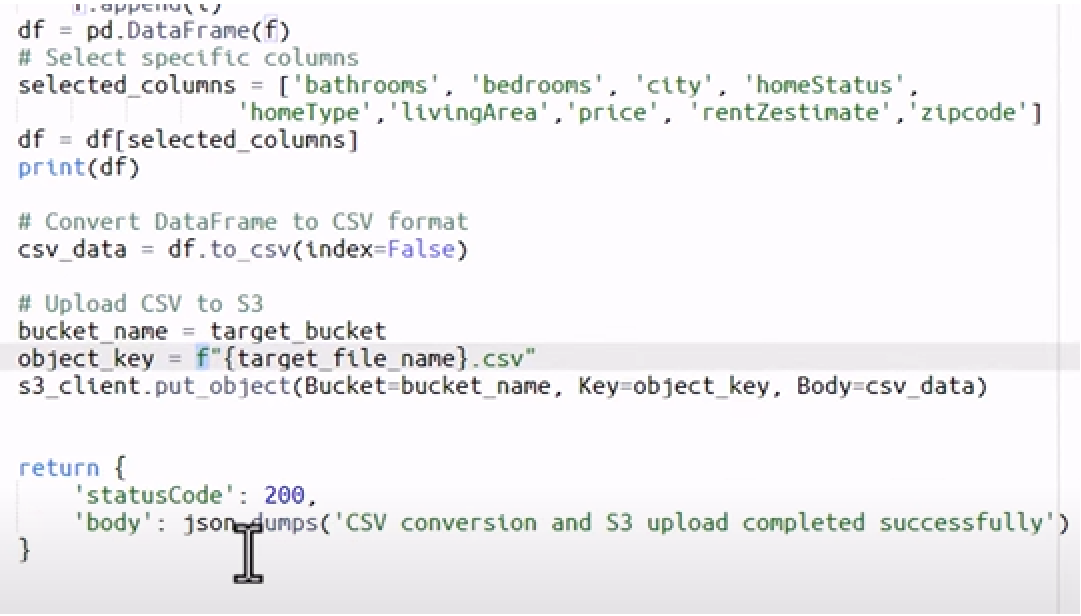
4. Next we'll create a bucket for cleaned_data(target_bucket)
5. Then we'll retrieve data from response['Body'] ..and then we need to decode it to 'utf-8'
6. To see the data…we used json.loads(data)

7. Data will look like this



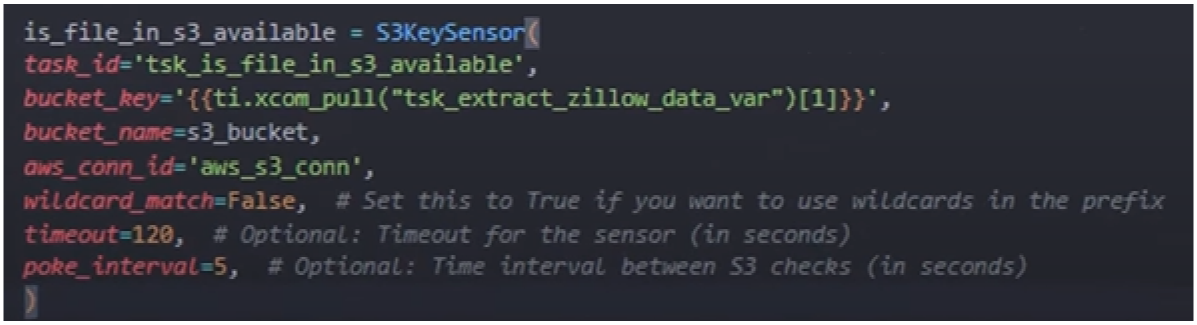
```
{
    "results": [
        {
            "bathrooms": 2.0,
            "bedrooms": 3.0,
            "city": "Houston",
            "country": "USA",
            "currency": "USD",
            "daysOnZillow": 0,
            "homeStatus": "FOR_SALE",
            "homeStatusForHDP": "FOR_SALE",
            "homeType": "SINGLE_FAMILY",
            "imgSrc": "https://photos.zillowstatic.com/fp/eaa31be6b0b2267323624c193367be4c-p_e.jpg",
            "isFeatured": false,
            "isNonOwnerOccupied": true,
            "isPreforeclosureAuction": false,
            "isPremierBuilder": false,
            "isShowcaseListing": false,
            "isUnmappable": false,
            "isZillowOwned": false,
            "latitude": 29.885355,
            "listing_sub_type": {
                "is_FSBA": true
            },
            "livingArea": 1060.0,
            "longitude": -95.61802,
            "lotAreaUnit": "sqft",
            "lotAreaValue": 5998.212,
            "price": 125000.0,
            "priceForHDP": 125000.0,
            "rentZestimate": 1566,
            "shouldHighlight": false,
            "state": "TX",
            "streetAddress": "13723 Smokey Trail Dr",
            "taxAssessedValue": 158004.0,
            "zestimate": 125900,
            "zipcode": "77041",
            "zpid": 28282152
        },
        {
            "bathrooms": 3.0,
            "bedrooms": 4.0,
            "city": "Houston",
            "country": "USA",
            "currency": "USD",
            "daysOnZillow": 0,
            "homeStatus": "FOR_SALE",
            "homeStatusForHDP": "FOR_SALE",
            "homeType": "SINGLE_FAMILY",
```

8. Basically we took some columns and converted it to df…then again we converted this data to CSV



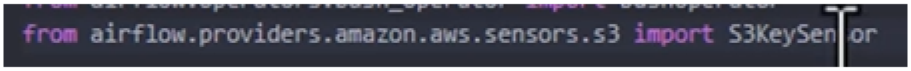
```
    f.append(t)
df = pd.DataFrame(f)
# Select specific columns
selected_columns = ['bathrooms', 'bedrooms', 'city', 'homeStatus',
                    'homeType','livingArea','price', 'rentZestimate','zipcode']
df = df[selected_columns]
print(df)

# Convert DataFrame to CSV format
csv_data = df.to_csv(index=False)

# Upload CSV to S3
bucket_name = target_bucket
object_key = f"{target_file_name}.csv"
s3_client.put_object(Bucket=bucket_name, Key=object_key, Body=csv_data)


return {
    'statusCode': 200,
    'body': json.dumps('CSV conversion and S3 upload completed successfully')
}
```

9. Now we’ll create an another task in our airflow

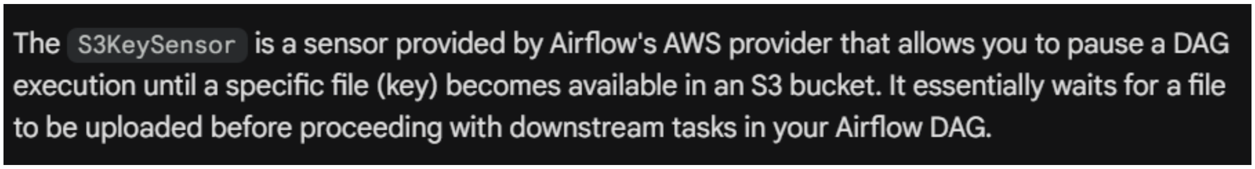


```
is_file_in_s3_available = S3KeySensor(
task_id='tsk_is_file_in_s3_available',
bucket_key='{{ti.xcom_pull("tsk_extract_zillow_data_var")[1]}}',
bucket_name=s3_bucket,
aws_conn_id='aws_s3_conn',
wildcard_match=False,  # Set this to True if you want to use wildcards in the prefix
timeout=120,  # Optional: Timeout for the sensor (in seconds)
poke_interval=5,  # Optional: Time interval between S3 checks (in seconds)
)
```

10. First we need to import this key sensor



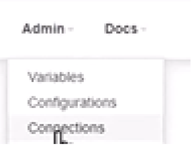
```
from airflow.providers.amazon.aws.sensors.s3 import S3KeySensor
```



The `S3KeySensor` is a sensor provided by Airflow's AWS provider that allows you to pause a DAG execution until a specific file (key) becomes available in an S3 bucket. It essentially waits for a file to be uploaded before proceeding with downstream tasks in your Airflow DAG.

11. More on s3keysensor : https://g.co/gemini/share/04c9d4eb3193

12. And also we need to give connection to our aws..so we go to



Admin
Docs
Variables
Configurations
Connections