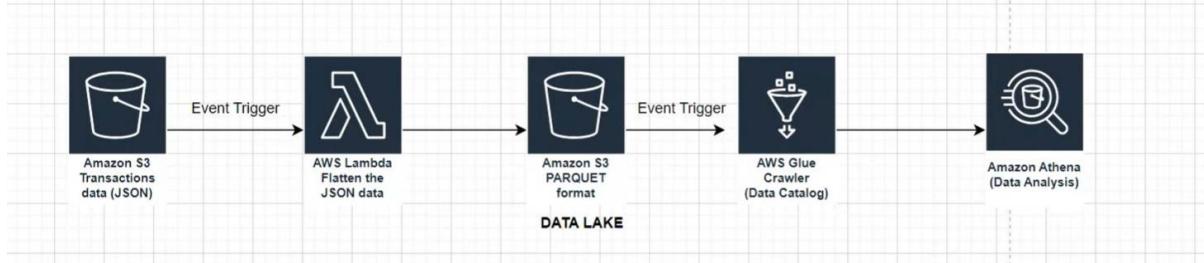


# AWS Serverless ETL Pipeline

Event Driven Data Pipeline:

We will have to load a file in S3 and as soon as we load, everything will happen automatically as per the pipeline, we don't need to schedule



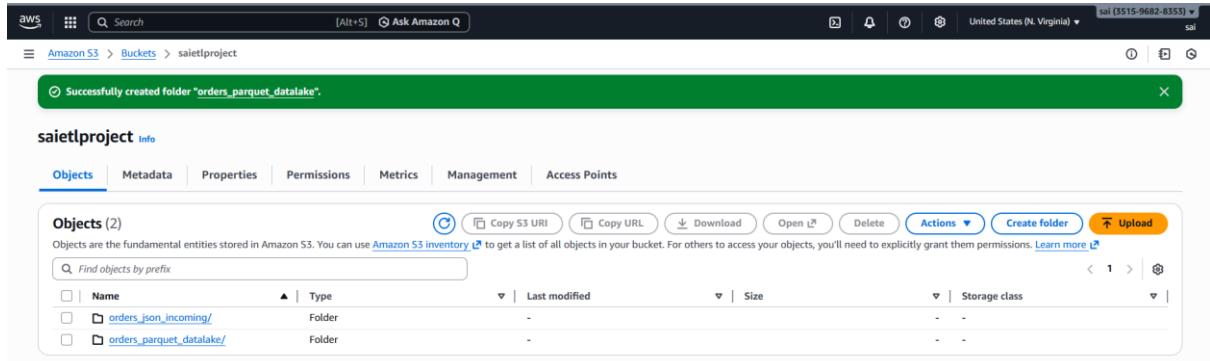
AWS lambda is a serverless compute which can handle any kind of processing without provisioning any EC2 server

Creating a bucket as shown

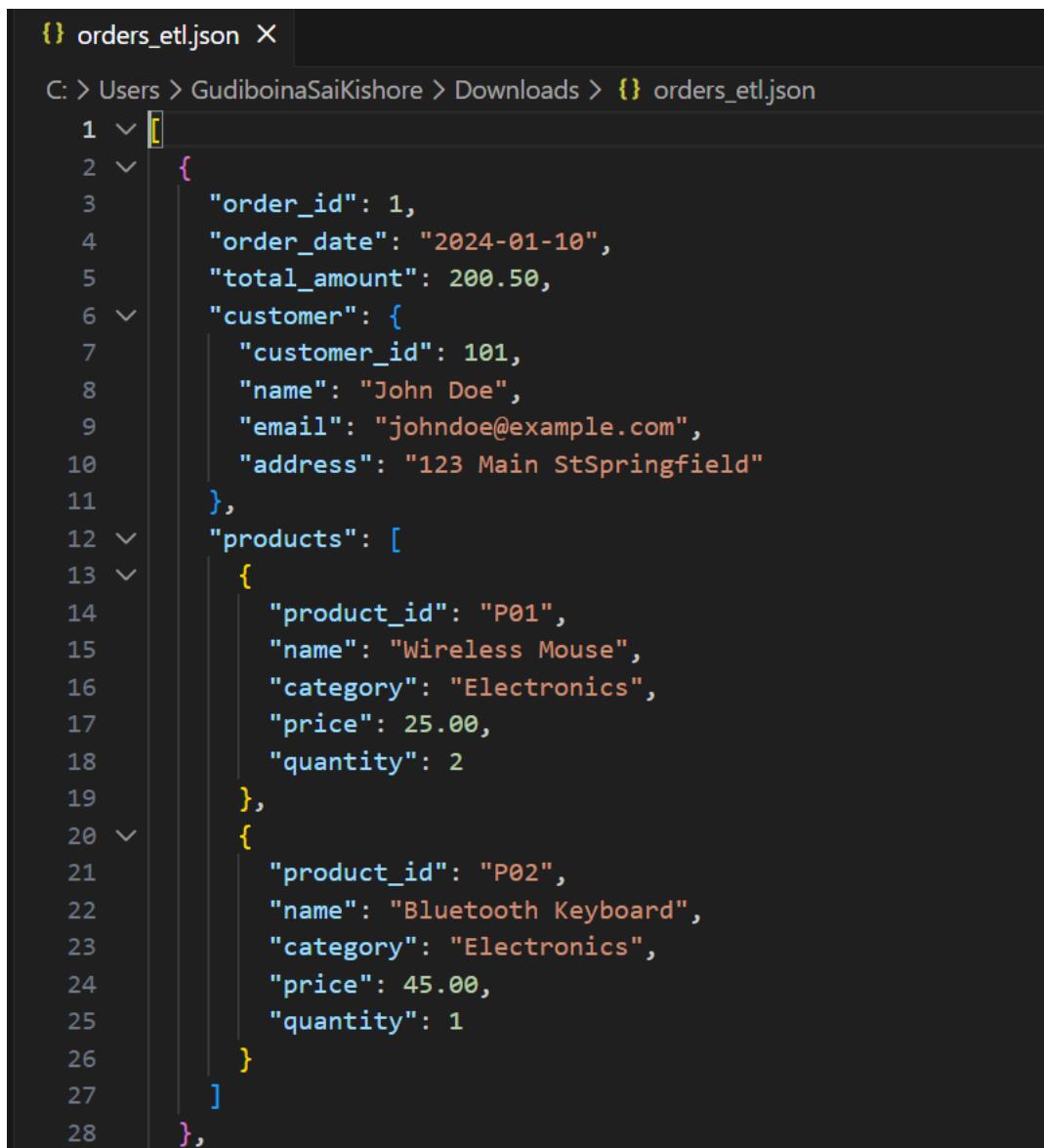
The screenshot shows the 'Create bucket' wizard in the AWS S3 console. Under 'General configuration', the 'AWS Region' is set to 'US East (N. Virginia) us-east-1'. The 'Bucket type' is set to 'General purpose'. The 'Bucket name' field contains 'saielproject'. Other sections include 'Copy settings from existing bucket - optional' and 'Bucket settings'.

The screenshot shows the 'Buckets' page in the AWS S3 console. A green banner at the top indicates that the bucket 'saielproject' was successfully created. The page lists 'General purpose buckets (5)' and 'Directory buckets'. One of the buckets, 'saielproject', is highlighted. The 'Account snapshot' and 'External access summary' sections are also visible on the right.

We had create two folders as shown



Lets see the orders data file

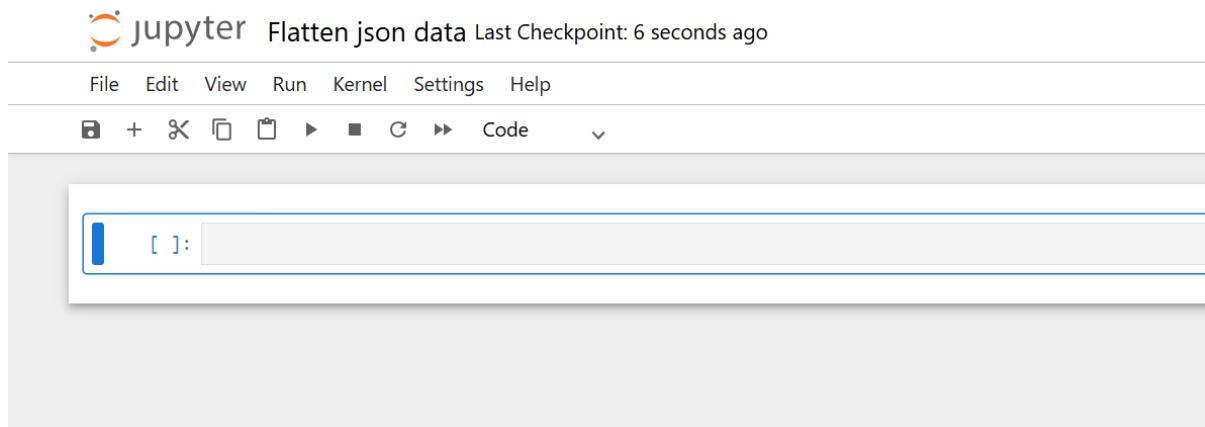


```
{} orders_etl.json X
C: > Users > GudiboinaSaiKishore > Downloads > {} orders_etl.json
1  [ 
2   { 
3     "order_id": 1,
4     "order_date": "2024-01-10",
5     "total_amount": 200.50,
6     "customer": { 
7       "customer_id": 101,
8       "name": "John Doe",
9       "email": "johndoe@example.com",
10      "address": "123 Main St Springfield"
11    },
12    "products": [
13      {
14        "product_id": "P01",
15        "name": "Wireless Mouse",
16        "category": "Electronics",
17        "price": 25.00,
18        "quantity": 2
19      },
20      {
21        "product_id": "P02",
22        "name": "Bluetooth Keyboard",
23        "category": "Electronics",
24        "price": 45.00,
25        "quantity": 1
26      }
27    ]
28  },
```

File is uploaded in GIT

Lets see how to flatten the data

Open ur Jupyter notebook



Load the json data to read

```
[1]: import json
f = open('orders_etl.json', 'r')
data = json.load(f)

[2]: data
```

```
[2]: [{}{'order_id': 1,
    'order_date': '2024-01-10',
    'total_amount': 200.5,
    'customer': {'customer_id': 101,
        'name': 'John Doe',
        'email': 'johndoe@example.com',
        'address': '123 Main St Springfield'},
    'products': [{product_id': 'P01',
        'name': 'Wireless Mouse',
        'category': 'Electronics',
        'price': 25.0,
        'quantity': 2},
        {'product_id': 'P02',
        'name': 'Bluetooth Keyboard',
        'category': 'Electronics',
        'price': 45.0,
```

We loaded the data into variable 'data'

```
[6]: #lets flatten data and convert to a pandas dataframe
import pandas as pd
def flatten(data):
    orders_data = []
    for order in data:
        for product in order['products']:
            row_orders = {
                "order_id": order["order_id"],
                "order_date": order["order_date"],
                "total_amount": order["total_amount"],
                "customer_id": order["customer"]["customer_id"],
                "customer_name": order["customer"]["name"],
                "email": order["customer"]["email"],
                "address": order["customer"]["address"],
                "product_id": product["product_id"],
                "product_name": product["name"],
                "category": product["category"],
                "price": product["price"],
                "quantity": product["quantity"]
            }
            orders_data.append(row_orders)
    df_orders = pd.DataFrame(orders_data)
    return df_orders
```

	order_id	order_date	total_amount	customer_id	customer_name	email	address	product_id	product_name	category	price	quantity
0	1	2024-01-10	200.5	101	John Doe	johndoe@example.com	123 Main StSpringfield	P01	Wireless Mouse	Electronics	25.0	2
1	1	2024-01-10	200.5	101	John Doe	johndoe@example.com	123 Main StSpringfield	P02	Bluetooth Keyboard	Electronics	45.0	1
2	2	2024-01-12	150.0	102	Jane Smith	janesmith@example.com	456 Oak StSpringfield	P03	Laptop Stand	Electronics	75.0	1
3	3	2024-01-12	120.0	103	Alice Johnson	alicejohnson@example.com	789 Birch StSpringfield	P04	Gaming Headset	Electronics	60.0	2
4	4	2024-01-13	300.5	101	John Doe	johndoe@example.com	123 Main StSpringfield	P01	Wireless Mouse	Electronics	25.0	3
5	4	2024-01-13	300.5	101	John Doe	johndoe@example.com	123 Main StSpringfield	P02	Bluetooth Keyboard	Electronics	45.0	2
6	5	2024-01-14	180.0	104	Bob Brown	bobbrown@example.com	101 Maple StSpringfield	P03	Laptop Stand	Electronics	75.0	1
7	5	2024-01-	180.0	104	Bob Brown	bobbrown@example.com	101 Maple	P04	Gaming	Electronics	60.0	1

Now lets go to lambda to process this data

The screenshot shows the AWS Lambda homepage. In the center, there's a "How it works" section with tabs for .NET, Java, Node.js, Python, Ruby, and Custom runtime. The Node.js tab is selected, displaying the following code:

```

1 * exports.handler = async (event) => {
2     console.log(event);
3     return 'Hello from Lambda!';
4 };
5

```

Below the code, there's a "Run" button and a link to "Next: Lambda responds to events". To the right, there's a "Get started" box with the text: "Author a Lambda function from scratch, or choose from one of many preconfigured examples." and a "Create a function" button.

## Lets create a function

The screenshot shows the "Create function" wizard. It has three tabs: "Author from scratch" (selected), "Use a blueprint", and "Container image".

**Basic information**

- Function name:** ETL\_pipeline
- Routine:** Python 3.14
- Durable execution - new:** Enabled
- Architecture:** x86\_64
- Permissions:** Change default execution role
- Additional configurations:** Use additional configurations to set up networking, security, and governance for your function.

At the bottom right are "Cancel" and "Create function" buttons. A large arrow points from the "Function name" input field to the "Create function" button.

The screenshot shows the "ETL\_pipeline" function details page. A green success message at the top says: "Successfully created the function ETL\_pipeline. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

**Function overview**

- Diagram:** Shows a single node labeled "ETL\_pipeline".
- Layers:** (0)
- Actions:** Throttle, Copy ARN, Download, Export to Infrastructure Composer, + Add trigger, + Add destination.
- Description:** -
- Last modified:** 26 seconds ago
- Function ARN:** arn:aws:lambda:us-east-1:351596828353:function:ETL\_pipeline
- Function URL:** -

At the bottom, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. On the right, there's a "Tutorials" sidebar with a "Create a simple web app" section and a "Start tutorial" button.

As this function is created, we want to read the function from s3

Successfully created the function **ETL\_pipeline**. You can now change its code and configuration. To invoke your function with a test event, choose "

When we create a lambda function, it by default created a role as shown above i.e whatever permissions this role has, those only can be done by lambda function

Lets click on that role and check

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:createLogGroup",
            "Resource": "arn:aws:logs:us-east-1:351596828353:log-group:/aws/lambda/ETL_pipeline:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:351596828353:log-group:/aws/lambda/ETL_pipeline:*"
            ]
        }
    ]
}

```

As we want to read data from s3, let us give s3 access as well

Identity and Access Management (IAM)

Access management

Access reports

Permissions policies (1) Info

AWSLambdaBasicExecutionRole-1a04e7ca-7173-4c7f-babe-30d8c822f568

```

1- {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "arn:aws:logs:us-east-1:351596828353:log-group:/aws/lambda/AWSLambdaBasicExecutionRole-1a04e7ca-7173-4c7f-babe-30d8c822f568"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:us-east-1:351596828353:log-stream:/aws/lambda/AWSLambdaBasicExecutionRole-1a04e7ca-7173-4c7f-babe-30d8c822f568"
    }
  ]
}

```

Add permissions ▾

## Giving full access to S3

Attach policy to ETL\_pipeline-role-i49qcwhm

Current permissions policies (1)

Other permissions policies (1/1110)

Policy name	Type	Description
<a href="#">AmazonDMSRedshiftS3Role</a>	AWS managed	Provides access to manage S3 settings...
<a href="#">AmazonS3FullAccess</a> <input checked="" type="checkbox"/>	AWS managed	Provides full access to all buckets via t...
<a href="#">AmazonSQObjectLambdaExecutionRolePolicy</a>	AWS managed	Provides AWS Lambda functions permi...
<a href="#">AmazonS3OutpostsFullAccess</a>	AWS managed	Provides full access to Amazon S3 on ...
<a href="#">AmazonS3OutpostsReadOnlyAccess</a>	AWS managed	Provides read only access to Amazon S...
<a href="#">AmazonS3ReadOnlyAccess</a>	AWS managed	Provides read only access to all bucket...
<a href="#">AmazonS3TablesFullAccess</a>	AWS managed	Provides full access to all S3 table bu...
<a href="#">AmazonS3TablesLakeFormationServiceRole</a>	AWS managed	This managed policy grants AWS Lake ...

## Click on add permission

AmazonS3FullAccess

AmazonS3OutpostsFullAccess

AmazonS3OutpostsReadOnlyAccess

AmazonS3ReadOnlyAccess

AmazonS3TablesFullAccess

AmazonS3TablesLakeFormationServiceRole

AmazonS3TablesReadOnlyAccess

AWSBackupServiceRolePolicyForS3Backup

AWSBackupServiceRolePolicyForS3Restore

AWSGlueServiceRole-project01-EZCRC-s3Policy

AWSQuickSetupSSMDeploymentS3BucketRolePolicy

QuickSightAccessForS3StorageManagementAnalyticsReadOnly

S3UnlockBucketPolicy

Add permissions

The screenshot shows the AWS IAM console with the path: IAM > Roles > ETL\_pipeline-role-i49qcwhm. A green success message at the top says "Policy was successfully attached to role." Below it, a table lists policies attached to the role, including "AmazonS3FullAccess" (AWS managed) and "AWSLambdaBasicExecutionRole-1a04e7ca-7173-4c..." (Customer managed). The JSON code for "AmazonS3FullAccess" is displayed:

```

1-13] [
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "s3-object-lambda:*


At the bottom, there's a note about the "Permissions boundary (not set)".


```

Lets go back to lambda

The screenshot shows the AWS Lambda console with the path: Lambda > Functions > ETL\_pipeline. The "Function overview" section displays the function name "ETL\_pipeline", which has no layers. It includes tabs for "Diagram" (selected), "Template", "Throttle", "Copy ARN", "Actions", "Export to Infrastructure Composer", "Download", "Description", "Last modified" (15 minutes ago), "Function ARN" (arn:aws:lambda:us-east-1:351596828353:function:ETL\_pipeline), and "Function URL". The "Code" tab is selected, showing the code source for "ETL\_pipeline". On the right, there's a "Tutorials" sidebar with a "Create a simple web app" section.

We need to add trigger becoz as soon as file available in s3, this lambda function has to be executed and we write a code in this function to flatten under code section

Click on add trigger

There are lot of ways to trigger lambda as follows but we choose s3 as shown

## Add trigger

### Trigger configuration Info

Select a source

Search trigger sources

APIs/interactive/web

Alexa

alexa iot voice

API Gateway

aws api application-services backend HTTP REST

serverless

Application Load Balancer

aws HTTP load-balancing server web

CloudFront

cdn edge

CodeCommit

aws asynchronous developer-tools git

Cognito Sync Trigger

aws authentication frontend identity mobile sync

VPC Lattice

aws networking private privatelink vpc

Batch/bulk data processing

Cancel

Add

## Add trigger

### Trigger configuration Info

Select a source

Search trigger sources

X

Batch/bulk data processing

S3

aws asynchronous storage

## Add trigger

### Trigger configuration Info

S3

aws asynchronous storage

#### Bucket

Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

Search trigger sources

X

C

Bucket region: us-east-1

#### Event types

**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.

Search Bucket: All object create events

- All object create events
- PUT
- POST
- COPY
- Multipart upload completed
- All object delete events
- All object delete events
- Permanently deleted
- Delete marker created
- Restore from Glacier initiated
- Restore from Glacier initiated
- Restore from Glacier completed
- Restore from Glacier completed

with matching characters. Any special characters [must be URL encoded](#).

with matching characters. Any special characters [must be URL encoded](#).

buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage is not recommended and that this configuration can cause

Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

[Cancel](#) [Add](#)

Whenever u give prefix implies whenever a file is available in that prefix, then lambda gets triggered

**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 [X](#)

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.

orders\_json\_incoming/

**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.

In suffix u can give json in our case as it is optional as the overall meaning is w.r.t to prefix or suffix lambda fn has to be triggered

Acknowledge and add

**Bucket**  
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.  
Q. s3:saietproject [X](#) [C](#)  
Bucket region: us-east-1

**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 [X](#)

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.  
orders\_json\_incoming/

**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.  
json

**Recursive invocation**  
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)  
 I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

[Cancel](#) [Add](#)

U see the interface as shown

The screenshot shows the AWS Lambda console. In the top navigation bar, there's a search bar with 'lambda' and a link to 'Ask Amazon Q'. Below the navigation, it says 'Lambda > Functions > ETL\_pipeline'. The main area displays the 'ETL\_pipeline' function. On the left, there's a 'Function overview' section with a diagram showing an S3 trigger connected to the function. To the right of the diagram, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. Below the diagram, there's a 'Description' section with 'Last modified 25 minutes ago' and a 'Function ARN' field containing 'arn:aws:lambda:us-east-1:351596828353:function:ETL\_pipeline'. Further down, there's a 'Function URL' button. On the far right, there's a sidebar titled 'Create a simple web app' with a brief description and a 'Start tutorial' button.

Lets test by uploading our orders file in that s3 path

This screenshot shows the 'Configuration' tab of the Lambda function. The 'Monitor' tab is highlighted with a black arrow. Below the tabs, there are five buttons: 'View CloudWatch logs', 'View Application Signals', 'View X-Ray traces', 'View Lambda Insights', and 'View CodeGuru profiles'.

Go to monitor, click view cloudwatch logs

This screenshot shows the CloudWatch Logs interface. The left sidebar has a tree structure with 'CloudWatch' at the top, followed by 'Favorites and recents', 'Dashboards', 'AI Operations', 'Alarms', 'Logs' (which is expanded), and 'Metrics'. Under 'Logs', there are sections for 'Log groups', 'Log Anomalies', 'Live Tail', 'Logs Insights', and 'Contributor Insights'. The main area is titled 'Log events' and contains a table with columns 'Timestamp' and 'Message'. The table shows several log entries, starting with a partial 'INIT\_START' message followed by a full 'START' message, indicating the function has been triggered.

When you write a python code in lambda, the below function(lambda\_handler) will automatically be called, you don't need to write separate function

The screenshot shows the AWS Lambda console's 'Code' tab. On the left, the 'EXPLORER' sidebar shows a folder named 'ETL\_PIPELINE' containing a file 'lambda\_function.py'. The main area displays the contents of 'lambda\_function.py':

```
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

Below the code editor, there are two buttons: 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'. A search bar at the top right contains the text 'ETL\_pipeline'.

As we are testing lets write a print statement as shown

The screenshot shows the AWS Lambda console's 'Code' tab. The 'EXPLORER' sidebar shows a folder named 'ETL\_PIPELINE' containing a file 'lambda\_function.py'. The main area displays the contents of 'lambda\_function.py':

```
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     print(event)
6     return {
7         'statusCode': 200,
8         'body': json.dumps('Hello from Lambda!')
9     }
10
```

Below the code editor, there are two buttons: 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'. A search bar at the top right contains the text 'ETL\_pipeline'. The status bar indicates 'Undeployed'.

Click on on deploy

Now upload the file in the s3 bucket

The screenshot shows the AWS S3 console. The path is 'Amazon S3 > Buckets > sailetpoint > orders\_json\_incoming/'. The 'Objects' tab is selected, showing a single object named 'orders\_etl.json'. The object details are:

Name	Type	Last modified	Size	Storage class
orders_etl.json	json	December 31, 2025, 16:02:10 (UTC+05:30)	6.8 KB	Standard

Lets go to cloud watch and check

The screenshot shows the AWS CloudWatch Log Management interface. The left sidebar has sections for Dashboards, Alarms, AI Operations, GenAI Observability, Application Signals (APM), Infrastructure Monitoring, Logs (Log Management, Metrics, Network Monitoring, Setup), and Metrics. The main area shows the log stream details for the path /aws/lambda/ETL\_pipeline. It includes fields for Log class (Info), Standard, ARN (arn:aws:logs:us-east-1:351596828353:log-group:/aws/lambda/ETL\_pipeline), Creation time (2 minutes ago), Retention (Never expire), and Stored bytes (~). On the right, there are sections for Metric filters (0), Subscription filters (0), Contributor Insights rules (-), KMS key ID (-), Data protection (Off), Sensitive data count (-), Custom field indexes (Configure), Transformer (Configure), and Anomaly detection (Configure). Below this is a tab bar with Log streams, Tags, Data protection, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, Field indexes, and Transformer. The Log streams tab is selected, showing one log stream entry: 2025-12-31/[\$LATEST]041002af13fb4d9483816b0890d680ec. The entry timestamp is 2025-12-31 10:32:11 (UTC).

I print the event info and it is giving me as shown

The screenshot shows the AWS CloudWatch Log Management interface, similar to the previous one but with a different view. The left sidebar is identical. The main area shows the log events for the path /aws/lambda/ETL\_pipeline. It includes a filter bar, timestamp, message, and a list of log events. The log events are:

- 2025-12-31T10:32:11.405Z: INIT\_START Runtime Version: python:3.14.v32 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:1ee4e6d61a50fb29d03b87572cc627d0a92de84530be7b21838aaedd9675804
- 2025-12-31T10:32:11.528Z: START RequestId: c32e499a-8fd6-4ff9-be23-e987d96d4d5 Version: \$LATEST
- 2025-12-31T10:32:11.529Z: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2025-12-31T10:32:09.988Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:ADAVDXGN23A3FUKKFONW"}, "requestParameters": {"sourceIPAddress": "205.254.168.58"}, "responseElements": {"x-amz-request-id": "SKAFDTKE56PEVDMV", "x-amz-id-2": "TS90ztx9xaP1e81muUcrgYh/wkZhoyysCbZ15R3tgYgS7NVh0q4VRgk15WjBQ9UnlUqcwVO3MKelP2iuOs2xWluwzgdt"}, "s3": {"s3SchemaVersion": "1.0", "configurationId": "b6bcfa1d-6365-4656-af0b-258af8:ownerIdentity": {"principalId": "A300NUJ09J9R8IU"}, "arn": "arn:aws:s3:::saeti1project"}, "object": {"key": "orders\_json\_incoming/orders\_etl.json", "size": 6059, "eTag": "f8abff6dd94fc7add8918ee91a16900", "sequencer": "000954FBA9E8E1950"}]}]
- 2025-12-31T10:32:11.531Z: END RequestId: c32e499a-8fd6-4ff9-be23-e987d96d4d5
- 2025-12-31T10:32:11.532Z: REPORT RequestId: c32e499a-8fd6-4ff9-be23-e987d96d4d5 Duration: 2.45 ms Billed Duration: 122 ms Memory Size: 128 MB Max Memory Used: 40 MB Init Duration: 119.34 ms

No newer events at this moment. Auto retry paused. [Resume](#)

The event info is about bucket name,file etc

Now let us not hardcode our file name to flatten in the above lambda python code

Becoz tomorrow due to incremental load the file name might be different

Lets copy that event information as per the above image

Now go to notebook as shown

```
[7]: event = {'Records': [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2025-12-31T10:32:09.988Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:AIDAVDXGN23A3FUUKFONW"}, "requestParameters": {"sourceIPAddress": "205.254.168.58"}, "responseElements": {"x-amz-request-id": "5KAFDTKE56PEVDMV", "x-amz-id-2": "TS90zt9xzaPlE81mu1cgYh/aKZNavyShCbZi5R3tygE57NYMv0q4YRgK15Wj8QU9nLuQzcLwO3MKeNPZiuOsg2xNuaYzgdt"}, "s3": {"s3SchemaVersion": "1.0", "configurationId": "b6bcfaid-6365-4656-af0b-258af8567b78", "bucket": {"name": "saietlproject"}, "ownerIdentity": {"principalId": "A300NUJG9JR81U"}, "arn": "arn:aws:s3:::saietlproject"}, "object": {"key": "orders_json_incoming/orders_etl.json", "size": 6959, "eTag": "f8abff6dd94bfc7add8916ae91a16b60", "sequencer": "006954FBA9EB8E1950"}]}}

[8]: event

[8]: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2025-12-31T10:32:09.988Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:AIDAVDXGN23A3FUUKFONW"}, "requestParameters": {"sourceIPAddress": "205.254.168.58"}, "responseElements": {"x-amz-request-id": "5KAFDTKE56PEVDMV", "x-amz-id-2": "TS90zt9xzaPlE81mu1cgYh/aKZNavyShCbZi5R3tygE57NYMv0q4YRgK15Wj8QU9nLuQzcLwO3MKeNPZiuOsg2xNuaYzgdt"}, "s3": {"s3SchemaVersion": "1.0", "configurationId": "b6bcfaid-6365-4656-af0b-258af8567b78", "bucket": {"name": "saietlproject"}, "ownerIdentity": {"principalId": "A300NUJG9JR81U"}, "arn": "arn:aws:s3:::saietlproject"}, "object": {"key": "orders_json_incoming/orders_etl.json", "size": 6959, "eTag": "f8abff6dd94bfc7add8916ae91a16b60", "sequencer": "006954FBA9EB8E1950"}]}}

[9]: event['Records']

[9]: [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2025-12-31T10:32:09.988Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:AIDAVDXGN23A3FUUKFONW"}, "requestParameters": {"sourceIPAddress": "205.254.168.58"}, "responseElements": {"x-amz-request-id": "5KAFDTKE56PEVDMV", "x-amz-id-2": "TS90zt9xzaPlE81mu1cgYh/aKZNavyShCbZi5R3tygE57NYMv0q4YRgK15Wj8QU9nLuQzcLwO3MKeNPZiuOsg2xNuaYzgdt"}, "s3": {"s3SchemaVersion": "1.0", "configurationId": "b6bcfaid-6365-4656-af0b-258af8567b78", "bucket": {"name": "saietlproject"}, "ownerIdentity": {"principalId": "A300NUJG9JR81U"}, "arn": "arn:aws:s3:::saietlproject"}, "object": {"key": "orders_json_incoming/orders_etl.json", "size": 6959, "eTag": "f8abff6dd94bfc7add8916ae91a16b60", "sequencer": "006954FBA9EB8E1950"}}]

[10]: event['Records']

[10]: [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2025-12-31T10:32:09.988Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:AIDAVDXGN23A3FUUKFONW"}, "requestParameters": {"sourceIPAddress": "205.254.168.58"}, "responseElements": {"x-amz-request-id": "5KAFDTKE56PEVDMV", "x-amz-id-2": "TS90zt9xzaPlE81mu1cgYh/aKZNavyShCbZi5R3tygE57NYMv0q4YRgK15Wj8QU9nLuQzcLwO3MKeNPZiuOsg2xNuaYzgdt"}, "s3": {"s3SchemaVersion": "1.0", "configurationId": "b6bcfaid-6365-4656-af0b-258af8567b78", "bucket": {"name": "saietlproject"}, "ownerIdentity": {"principalId": "A300NUJG9JR81U"}, "arn": "arn:aws:s3:::saietlproject"}, "object": {"key": "orders_json_incoming/orders_etl.json", "size": 6959, "eTag": "f8abff6dd94bfc7add8916ae91a16b60", "sequencer": "006954FBA9EB8E1950"}}]

[11]: event['Records'][0]

[11]: {"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2025-12-31T10:32:09.988Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:AIDAVDXGN23A3FUUKFONW"}, "requestParameters": {"sourceIPAddress": "205.254.168.58"}, "responseElements": {"x-amz-request-id": "5KAFDTKE56PEVDMV", "x-amz-id-2": "TS90zt9xzaPlE81mu1cgYh/aKZNavyShCbZi5R3tygE57NYMv0q4YRgK15Wj8QU9nLuQzcLwO3MKeNPZiuOsg2xNuaYzgdt"}, "s3": {"s3SchemaVersion": "1.0", "configurationId": "b6bcfaid-6365-4656-af0b-258af8567b78", "bucket": {"name": "saietlproject"}, "ownerIdentity": {"principalId": "A300NUJG9JR81U"}, "arn": "arn:aws:s3:::saietlproject"}, "object": {"key": "orders_json_incoming/orders_etl.json", "size": 6959, "eTag": "f8abff6dd94bfc7add8916ae91a16b60", "sequencer": "006954FBA9EB8E1950"}}

[12]: event['Records'][0]['s3']['bucket']['name']

[12]: 'saietlproject'

[13]: event['Records'][0]['s3']['bucket']['name']

[13]: 'saietlproject'
```

```
[20]: event['Records'][0]['s3']['object']['key']
[20]: 'orders_json_incoming/orders_etl.json'
```

Lets store in variables

```
[19]: bucket_name = event['Records'][0]['s3']['bucket']['name']
[21]: file_name= event['Records'][0]['s3']['object']['key']
```

```
[ ]:
```

Now remove print and paste as follows

Code | Test | Monitor | Configuration | Aliases | Versions

Code source [Info](#) [Open in Visual Studio Code](#)

EXPLORER ETL\_PIPELINE lambda\_function.py

```
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     bucket_name = event['Records'][0]['s3']['bucket']['name']
6     file_name= event['Records'][0]['s3']['object']['key']
7
8     return {
9         'statusCode': 200,
10        'body': json.dumps('Hello from Lambda!')
11    }
```

DEPLOY Undeployed Deploy (Ctrl+Shift+U) Test (Ctrl+Shift+I)

Now lets process/read the data

For that we require boto3 library which helps to read data from s3

Here we don't have to pass secret keys as whatever role that we are using to trigger the lamda has already access to s3 which we had set,

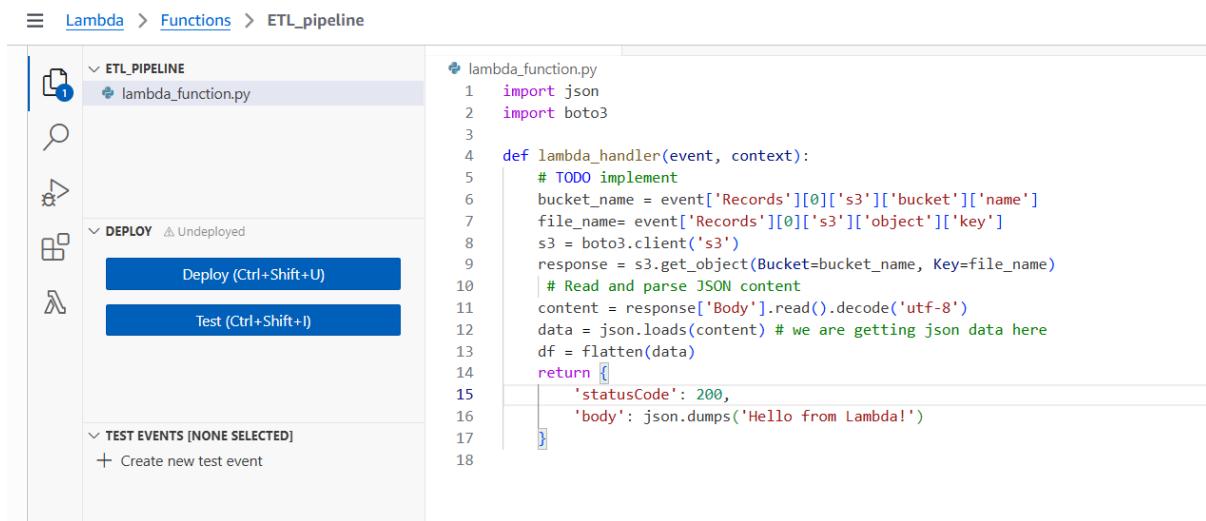
If u r running from ur local like Jupiter notebook trying to read data from s3 then these secret keys have to be mentioned as below



```
lambda_function.py X
lambda_function.py
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     # TODO implement
6     bucket_name = event['Records'][0]['s3']['bucket']['name']
7     file_name= event['Records'][0]['s3']['object']['key']
8     s3 = boto3.client('s3') access_key_id='XXXXXXXXXXXXXXXXXXXX',secret_access_key='XXXXXXXXXXXX'
9     return {
10         'statusCode': 200,
11         'body': json.dumps('Hello from Lambda!')
12     }
13
```

As we want to read data from s3 no need of secret keys

Why we decode is becoz that 'response' variable have the data in the form of binary form i.e bytes. So converting to string format while reading the body we use decode('utf-8')



```
Lambda > Functions > ETL_pipeline
ETL_PIPELINE
lambda_function.py
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     # TODO implement
6     bucket_name = event['Records'][0]['s3']['bucket']['name']
7     file_name= event['Records'][0]['s3']['object']['key']
8     s3 = boto3.client('s3')
9     response = s3.get_object(Bucket=bucket_name, Key=file_name)
10    # Read and parse JSON content
11    content = response['Body'].read().decode('utf-8')
12    data = json.loads(content) # we are getting json data here
13    df = flatten(data)
14    return [
15        'statusCode': 200,
16        'body': json.dumps('Hello from Lambda!')
17    ]
18
```

Now lets create flatten function, just paste the code we had written in notebook

```

lambda_function.py
1 import json
2 import boto3
3
4 def flatten(data):
5     orders_data=[]
6     for order in data:
7         for product in order['products']:
8             row_orders = {
9                 "order_id" : order["order_id"],
10                "order_date" : order["order_date"],
11                "total_amount" : order["total_amount"],
12                "customer_id" : order["customer"]["customer_id"],
13                "customer_name" : order["customer"]["name"],
14                "email" : order["customer"]["email"],
15                "address" : order["customer"]["address"],
16                "product_id" : product["product_id"],
17                "product_name" : product["name"],
18                "category" : product["category"],
19                "price" : product["price"],
20                "quantity" : product["quantity"]
21            }
22            orders_data.append(row_orders)
23    df_orders = pd.DataFrame(orders_data)
24    return df_orders

```

Pls import pandas as well but pandas module is not available in lambda

When u scroll down in the same page u see an option add layer

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN

**Choose a layer**

**Layer source** [Info](#)  
Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also [create a new layer](#).

**AWS layers**  
Choose a layer from a list of layers provided by AWS.

**Custom layers**  
Choose a layer from a list of layers created by your AWS account.

**Specify an ARN**  
Specify a layer by providing the ARN.

**AWS layers**  
Layers provided by AWS that are compatible with your function's runtime.

AWSSDKPandas-Python313

**Version**  
3

**Add**

To save it we need to deploy every time

The screenshot shows the AWS Lambda function editor for the 'ETL\_pipeline' function. The code is as follows:

```
    /          for order in data:
8           for product in order['products']:
24          df_orders = pd.DataFrame(orders_data)
25          return df_orders
26
27 def lambda_handler(event, context):
28     # TODO implement
29     bucket_name = event['Records'][0]['s3']['bucket']['name']
30     file_name= event['Records'][0]['s3']['object']['key']
31     s3 = boto3.client('s3')
32     response = s3.get_object(Bucket=bucket_name, Key=file_name)
33     | # Read and parse JSON content
34     content = response['Body'].read().decode('utf-8')
35     data = json.loads(content) # we are getting json data here
36     df = flatten(data)
37     print(df)
38     return {
39         'statusCode': 200,
40         'body': json.dumps('Hello from Lambda!')
41     }
```

A message at the bottom right says "Successfully updated the function ETL\_pipeline."

We just printed to df to check

To run it, we need to load the file in s3 one more time

Lets go to cloud watch and check

One more entry as shown,click on that

The screenshot shows the AWS CloudWatch Log Management interface. The left sidebar shows navigation options like CloudWatch, Alarms, AI Operations, Application Signals, Infrastructure Monitoring, and Logs. Under Logs, Log Management is selected. The main area shows a log group named 'group:/aws/lambda/ETL\_pipeline:\*'. It displays details such as Creation time (49 minutes ago), Retention (Never expire), and Stored bytes. On the right, there are sections for Contributor Insights rules, KMS key ID, and Deletion protection (Off). Below this, a 'Log streams' section lists two entries:

Log stream	Last event time
2025/12/31/[\$LATEST]aa027ab5a6e24fb84fd2416dd0fbfbf8	2025-12-31 11:19:28 (UTC)
2025/12/31/[\$LATEST]041002af13fb4d9483816b0890d680ec	2025-12-31 10:32:11 (UTC)

**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](#)

Filter events - press enter to search

Clear 1m 30m 1h 12h Custom UTC timezone

Display ▾

Timestamp	Message
2025-12-31T11:19:28.490Z	START RequestId: f20130a4-0ce3-4487-a21e-4ccfd192a969 Version: \$LATEST
2025-12-31T11:19:28.495Z	END RequestId: f20130a4-0ce3-4487-a21e-4ccfd192a969
2025-12-31T11:19:28.495Z	REPORT RequestId: f20130a4-0ce3-4487-a21e-4ccfd192a969 Duration: 277.88 ms Billed Duration: 278 ms Memory Size: 128 MB Max Memory Used: 70 MB
2025-12-31T11:20:25.895Z	[WARNING] 2025-12-31T11:20:25.895Z LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the function code. However, in this case, it's a runtime error.
2025-12-31T11:20:25.895Z	[ERROR] Runtime.ImportModuleError: Unable to import module 'lambda_function': No module named 'pandas'. Traceback (most recent call last):
2025-12-31T11:20:25.944Z	INIT_REPORT Init Duration: 265.08 ms Phase: invoke Status: error Error Type: Runtime.ImportModuleError
2025-12-31T11:20:25.944Z	START RequestId: f20130a4-0ce3-4487-a21e-4ccfd192a969 Version: \$LATEST
2025-12-31T11:20:25.948Z	END RequestId: f20130a4-0ce3-4487-a21e-4ccfd192a969
2025-12-31T11:20:25.948Z	REPORT RequestId: f20130a4-0ce3-4487-a21e-4ccfd192a969 Duration: 274.28 ms Billed Duration: 275 ms Memory Size: 128 MB Max Memory Used: 70 MB

No newer events at this moment. Auto retry paused. [Resume](#)

Back to top ↗

I don't see any change...so to fix this issue we need to change timeout

Ur lambda will timeout under 3 seconds, hence change by going to configuration

Lambda > Functions > ETL\_pipeline

Successfully updated the function ETL\_pipeline.

S3 + Add trigger

n:ETL\_pipeline

Function URL | Info

-

Code Test Monitor Configuration Aliases Versions

General configuration Triggers Permissions Destinations

**Execution role**

Role name [ETL\\_pipeline-role-i49qcwhm](#)

**Resource summary**

To view the resources and actions that your function has permission to access, changes are needed.

Increased to 30 sec

Lambda > Functions > ETL\_pipeline > Edit basic settings

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 operating considerations ↗. For Python and .NET runtimes, [view pricing ↗](#).

None

Supported runtimes: .NET 10 (C#/F#/PowerShell), .NET 8 (C#/F#/PowerShell), Java 11, Java 17, Java 21, Java 25, Python 3.12, Python 3.13, Python 3.14.

**Timeout**

0 min 30 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.

service-role/ETL\_pipeline-role-i49qcwhm

[View the ETL\\_pipeline-role-i49qcwhm role ↗](#) on the IAM console.

Lets upload file once again and check

CloudWatch > Log management > /aws/lambda/ETL\_pipeline

CloudWatch	Creation time 56 minutes ago	KMS key ID	Configure
Favorites and recents	Retention Never expire	Deletion protection	Transformer Configure
Dashboards	Stored bytes	Off	Anomaly detection Configure
Alarms			
AI Operations			
GenAI Observability			
Application Signals			
Infrastructure Monitoring			
Logs			
Log Management			
Log Anomalies			
Live Tail			
Logs Insights			
Contributor Insights			

**Log streams** (3)

By default, we only load the most recent log streams.

Log stream	Last event time
2025/12/31/[LATEST]ja8026bb0fdc44050a8e7265d5366ffa1	2025-12-31 11:27:59 (UTC)
2025/12/31/[LATEST]ja027ab5a6e24fb84fd2416dd0fbfb8	2025-12-31 11:19:28 (UTC)
2025/12/31/[LATEST]j041002af13fb4d9483816b0890d680ec	2025-12-31 10:32:11 (UTC)

U need to see ur dataframe

But testing can be done in other way as shown:

Code | **Test** | Monitor | Configuration | Aliases | Versions

**Test event** Info

To invoke your function without saving an event, configure the JSON event, then choose Test.

**Test event action**

- Create new event
- Edit saved event

**Invocation type**

- Synchronous  
Executes the Lambda function and blocks until receiving the function's response, with a maximum timeout of 15 minutes. Returns function output or error details directly to the calling application.
- Asynchronous  
Enqueues the Lambda function for execution and returns immediately with a request ID. Function processes independently, with results optionally sent to a configured destination like SQS, SNS, or EventBridge.

**Event name**

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

**Event sharing settings**

- Private

**Invocation type**

- Synchronous  
Executes the Lambda function and blocks until receiving the function's response, with a maximum timeout of 15 minutes. Returns function output or error details directly to the calling application.
- Asynchronous  
Enqueues the Lambda function for execution and returns immediately with a request ID. Function processes independently, with results optionally sent to a configured destination like SQS, SNS, or EventBridge.

**Event name**

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

**Event sharing settings**

- Private  
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)
- Shareable  
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

**Template - optional**

Last fetched 01/01/2026, 11:56:14

**Event JSON**

```

1 * []
2   "key1": "value1",
3   "key2": "value2",
4   "key3": "value3"
5 
```

[Format JSON](#)

Click on save on top

When u click on test, u can see as follows

☰ Lambda > Functions > ETL\_pipeline

The test event "Test" was successfully saved. ×

Executing function: failed ([logs](#))

[Diagnose with Amazon Q](#)

▼ Details

```
{
  "errorMessage": "Unable to import module 'lambda_function': No module named 'pandas'",
  "errorType": "Runtime.ImportModuleError",
  "requestId": "",
  "stackTrace": []
}
```

**Summary**

**Code SHA-256**  
LD8KHsEODOlOvd8UBH26+CGbDPOL5jPCWzx9JieInIM=

**Execution time**  
22 seconds ago

**Function version**  
\$LATEST

**Request ID**  
4e7f348b-ac9-4276-9402-ff82a6402ea2

**Duration**  
290.83 ms

**Billed duration**  
291 ms

**Resources configured**  
128 MB

**Max memory used**  
70 MB

<b>Duration</b>	290.83 ms	<b>Billed duration</b>	291 ms
<b>Resources configured</b>	128 MB	<b>Max memory used</b>	70 MB

**Log output**

The area below shows the last 4 KB of the execution log. [Click here](#) to view the corresponding CloudWatch log group.

```
[WARNING] 2026-01-01T06:28:52.436Z LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the function code. However, in rare cases, a Lambda runtime update can cause unexpected function behavior. For functions using managed runtimes, runtime updates can be triggered by a function change, or can be applied automatically. To determine if the runtime has been updated, check the runtime version in the INIT_START log entry. If this error correlates with a change in the runtime version, you may be able to mitigate this error by temporarily rolling back to the previous runtime version. For more information, see https://docs.aws.amazon.com/lambda/latest/dg/runtimes-update.html
[ERROR] Runtime.ImportModuleError: Unable to import module 'lambda_function': No module named 'pandas'
Traceback (most recent call last):
INIT_REPORT Init Duration: 456.23 ms Phase: init Status: error Error Type: Runtime.ImportModuleError
[WARNING] 2026-01-01T06:28:52.733Z LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the
```

**Test event info**

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save.

**Test event action**

## Now lets integrate the python packages with aws console

aws [Search] [Alt+S] United States (N. Virginia) sai (3515-9682-8353)

Lambda > Functions > ETL\_pipeline

**ETL\_pipeline**

**Function overview** [Info]

**Description**

Last modified 17 hours ago

Function ARN arn:aws:lambda:us-east-1:351596828353:function:ETL\_pipeline

Function URL [Info]

**Actions** Throttle, Copy ARN, Download, Export to Infrastructure Composer, Add destination, + Add trigger

**Tutorials**

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Here u can execute all the required details or commands here

aws [Search] [Alt+S] United States (N. Virginia) sai

Lambda > Functions > ETL\_pipeline

**CloudShell**

us-east-1 +

```
~ $ mkdir lambda_deployment
~ $
```

**Actions**

mkdir lambda\_deployment

cd lambda\_deployment/

vi lambda\_function.py

After this type "I" to be in insert mode as shown

The screenshot shows the AWS Lambda CloudShell interface. At the top, there's a navigation bar with three horizontal bars, the text "Lambda > Functions > ETL\_pipeline", and a search bar with the placeholder "Search functions...". Below the navigation is a header bar with tabs for "CloudShell" and "CloudWatch Metrics", and a dropdown menu icon. The main area is a dark-themed code editor titled "us-east-1". It has a status bar at the bottom with the number "1" and a small square icon, followed by a "+" sign and the text "-- INSERT --". The code editor itself is currently empty, showing only the title bar.

Copy the code and paste as shown:

```
import requests

def lambda_handler(event, context):

    response = requests.get("https://www.google.com/")
    print(response.text)
    return response.text
```

CloudShell

us-east-1 +

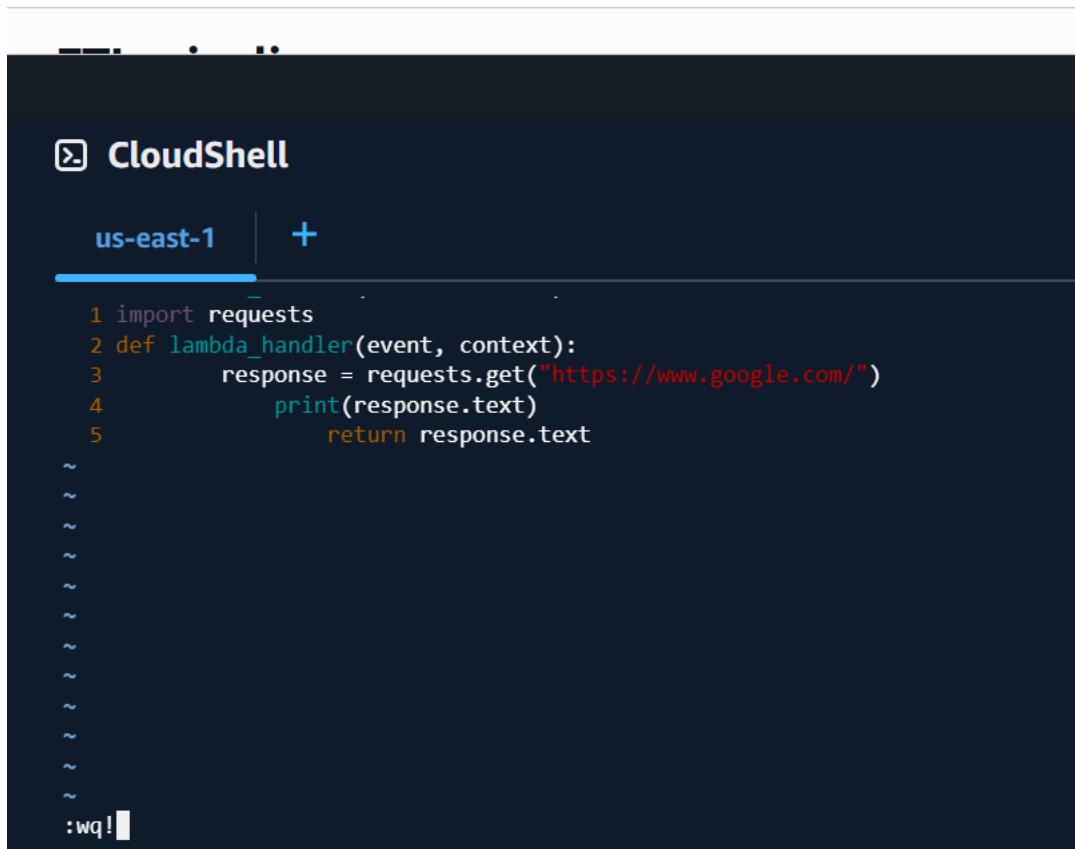
```
1 import requests
2 def lambda_handler(event, context):
3     response = requests.get("https://www.google.com/")
4     print(response.text)
5     return response.text
```

~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
-- INSERT --

[CloudShell](#) [Feedback](#) [Console Mobile App](#)

Now click esc

Then type “:wq!” wq stands for write and quick



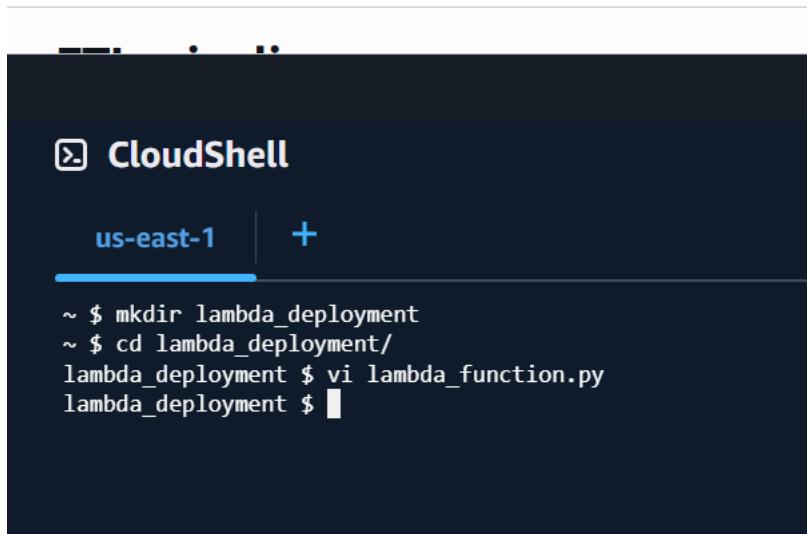
CloudShell

us-east-1 +

```
1 import requests
2 def lambda_handler(event, context):
3     response = requests.get("https://www.google.com/")
4     print(response.text)
5     return response.text
```

:wq!

Click enter



CloudShell

us-east-1 +

```
~ $ mkdir lambda_deployment
~ $ cd lambda_deployment/
lambda_deployment $ vi lambda_function.py
lambda_deployment $
```

Now u have lambda\_function.py in ur cloud shell

Now install the dependent packages

Type “pip install requests -t .”

 CloudShell

us-east-1 +

```
~ $ mkdir lambda_deployment
~ $ cd lambda_deployment/
lambda_deployment $ vi lambda_function.py
lambda_deployment $ pip install requests -t .
```

I also run “pip install pandas”

```
lambda_deployment $ pip install pandas
Defaulting to user installation because normal site-packages is not writeable
Collecting pandas
  Downloading pandas-2.3.3-cp39-cp39-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (12.8 MB)
    ██████████ | 12.8 MB 20.6 MB/s
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.9/site-packages (from pandas) (2.9.0.post0)
Collecting numpy>=1.22.4
  Downloading numpy-2.0.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (19.5 MB)
    ██████████ | 19.5 MB 67.2 MB/s
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.9/site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Installing collected packages: numpy, pandas
Successfully installed numpy-2.0.2 pandas-2.3.3
lambda_deployment $
```

Whatever we download we need to zip it, so use this code

“zip -r ..//lambda\_package.zip .”

Lambda\_package is the name of the zip file we are creating

Go to one step backward as shown

cd ..

ls -lrt

```
lambda_deployment $ cd ..
~ $ ls -lrt
total 920
drwxr-xr-x. 13 cloudshell-user cloudshell-user 4096 Jan  1 06:46 lambda_deployment
-rw-r--r--.  1 cloudshell-user cloudshell-user 934141 Jan  1 06:50 lambda_package.zip
~ $
```

Now let us import this zip in lamda function using s3 bucket

Lets create a bucket for this

The screenshot shows the AWS S3 console interface. At the top, the navigation bar indicates 'Amazon S3 > Buckets > lambda-pkgs'. Below the navigation, there's a header with tabs: 'Objects' (which is selected), 'Metadata', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. A sub-header 'lambda-pkgs' with a 'Info' link follows. The main area is titled 'Objects (0)' and contains a message: '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](#)'. There are buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload'. A search bar 'Find objects by prefix' and a table header 'Name | Type | Last modified | Size | Storage class' are also present. The table body displays 'No objects' with the message 'You don't have any objects in this bucket.' and a 'Upload' button.

Now type as shown “aws s3 cp lambda\_package.zip  
s3://{{bucket\_name}}”

```
~ $ ls -lrt
total 920
drwxr-xr-x. 13 cloudshell-user cloudshell-user 4096 Jan  1 06:46 lambda_deployment
-rw-r--r--. 1 cloudshell-user cloudshell-user 934141 Jan  1 06:50 lambda_package.zip
~ $ aws s3 cp lambda_package.zip s3://lambda-pkgs
upload: ./lambda_package.zip to s3://lambda-pkgs/lambda_package.zip
~ $
```

Upload has been completed as shown above

The screenshot shows the AWS S3 console interface. The left sidebar shows 'Amazon S3' and a 'Buckets' section with 'General purpose buckets' containing 'Directory buckets', 'Table buckets', and 'Vector buckets'. Other sections like 'Access management and security' and 'Storage management and' are also visible. The main area shows 'Objects (1)'. The table lists one object: 'lambda\_package.zip' (Type: zip, Last modified: January 1, 2026, 12:29:13 (UTC+05:30), Size: 912.2 KB, Storage class: Standard). Buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', and 'Create folder' are available. A search bar 'Find objects by prefix' and a table header 'Name | Type | Last modified | Size | Storage class' are also present.

Copy url

The screenshot shows the AWS S3 console interface. The left sidebar shows 'Amazon S3' and a 'Buckets' section with 'General purpose buckets' containing 'Directory buckets', 'Table buckets', and 'Vector buckets'. Other sections like 'Access management and security' and 'Storage management and' are also visible. The main area shows 'Objects (1/1)'. The table lists one object: 'lambda\_package.zip' (Type: zip, Last modified: January 1, 2026, 12:29:13 (UTC+05:30), Size: 912.2 KB, Storage class: Standard). A green message box says 'Object URL Copied'. Buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', and 'Create folder' are available. A search bar 'Find objects by prefix' and a table header 'Name | Type | Last modified | Size | Storage class' are also present.

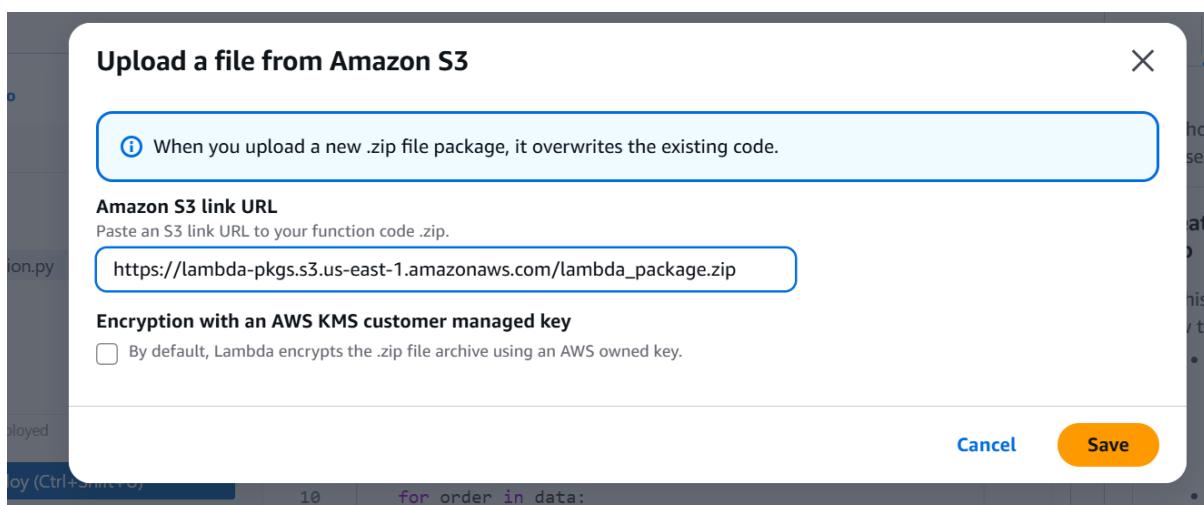
The screenshot shows the AWS Lambda function editor interface. At the top, there are buttons for "Open in Visual Studio Code" and "Upload from". Below that, a search bar contains "ETL\_pipeline". On the left, the "EXPLORER" sidebar shows a project named "ETL\_PIPELINE" containing a file "lambda\_function.py". Under "DEPLOY", there are buttons for "Deploy (Ctrl+Shift+U)" and "Test (Ctrl+Shift+I)". The main area displays the Python code for "lambda\_function.py".

```

1
2
3
4 import json
5 import boto3
6 import pandas as pd
7
8 def flatten(data):
9     orders_data=[]
10    for order in data:
11        for product in order['products']:
12

```

## Choose amazon s3 location



## Save

Now u can test

We are able to read the data now

Now, lets convert this dataframe into parquet

```

40
41 # in-memory binary buffer to hold data like a file, but without writing to disk.
42 parquet_buffer = io.BytesIO()
43 df.to_parquet(parquet_buffer, index=False, engine='pyarrow')
44
45 now = datetime.now()
46 timestamp = now.strftime("%Y%m%d_%H%M%S")
47
48 #key_staging=f'orders_csv/orders_ETL_{timestamp}.csv'
49 key_staging=f'orders_parquet/orders_ETL_{timestamp}.parquet'
50
51 #s3.put_object(Bucket=bucket_name, Key=key_staging, Body=csv_buffer.getvalue())
52 s3.put_object(Bucket=bucket_name, Key=key_staging, Body=parquet_buffer.getvalue())

```

42 - we are converting to inmemory bytes i.e holds the data in the form of bytes. Created a buffer

43 – write by dataframe to this buffer (note: import io library), now parquet\_buffer holds my data in byte format

49 – I can't have a fixed name as if multiple files are coming I cant say all file names as order\_etl.so it overwrites all the time if u have incremental load. So adding a timestamp to my file so that doesn't overwrite

52 – key\_staging is where u want to get the files into

In our case it is as shown

```
lambda_function.py
29 def lambda_handler(event, context):
46     now = datetime.now()
47     timestamp = now.strftime("%Y%m%d_%H%M%S")
48
49     key_staging=f'orders_parquet_datalake/orders_ETL_{timestamp}.parquet'
50
51
52     s3.put_object(Bucket=bucket_name, Key=key_staging, Body=parquet_buffer.getvalue())
53
54
55     return {
56         'statusCode': 200,
57         'body': json.dumps('Hello from Lambda!')
58     }
59
```

Getvalue() gives me the value in byte format

Lets deploy and test like upload ur file in s3 and check

U can see ur parquet file now

The screenshot shows the AWS S3 console interface. At the top, there's a breadcrumb navigation bar with 'orders\_parquet\_datalake/' followed by a 'Copy S3 URI' button. Below this is a toolbar with 'Objects' selected, along with 'Properties', 'Actions', 'Create folder', and 'Upload' buttons. A message below the toolbar states: '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](#)'.

Under the toolbar, there's a search bar labeled 'Find objects by prefix' and a set of filter buttons for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. The main table lists one object:

Name	Type	Last modified	Size	Storage class
orders_ETL_20250708_06041.parquet	parquet	July 8, 2025, 11:34:11 (UTC+05:30)	8.0 KB	Standard

Now lets create a data catalog for it

Create a database

**Database details**

**Name**  
etl\_pipeline

Database name is required, in lowercase characters, and no longer than 255 characters.

**Description - optional**  
Enter text

Descriptions can be up to 2048 characters long.

**Database settings**

**Location - optional**  
Set the URI location for use by clients of the Data Catalog.

## Create a crawler

**Set crawler properties**

**Crawler details** Info

**Name**  
etl\_pipeline\_crawler

Name can be up to 255 characters long. Some character set including control characters are prohibited.

**Description - optional**  
Enter a description

Descriptions can be up to 2048 characters long.

**Tags - optional**

**Data source configuration**

Is your data already mapped to Glue tables?

Not yet  
Select one or more data sources to be crawled.

Yes  
Select existing tables from your Glue Data Catalog.

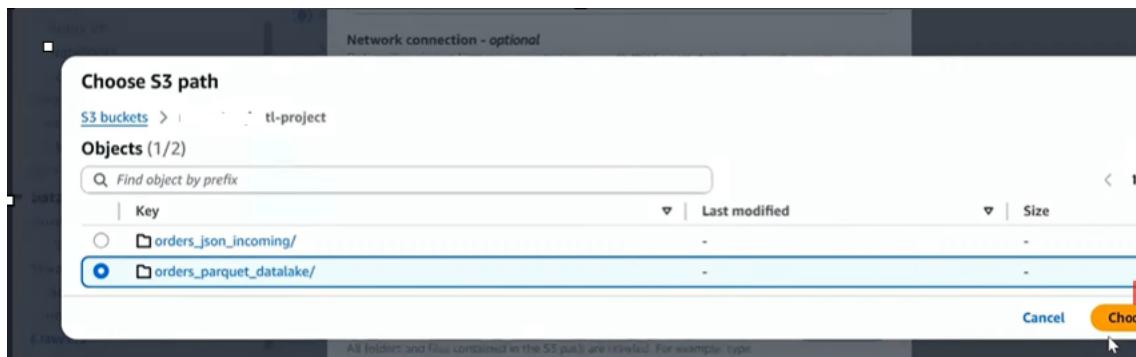
**Data sources (0)** Info

The list of data sources to be scanned by the crawler.

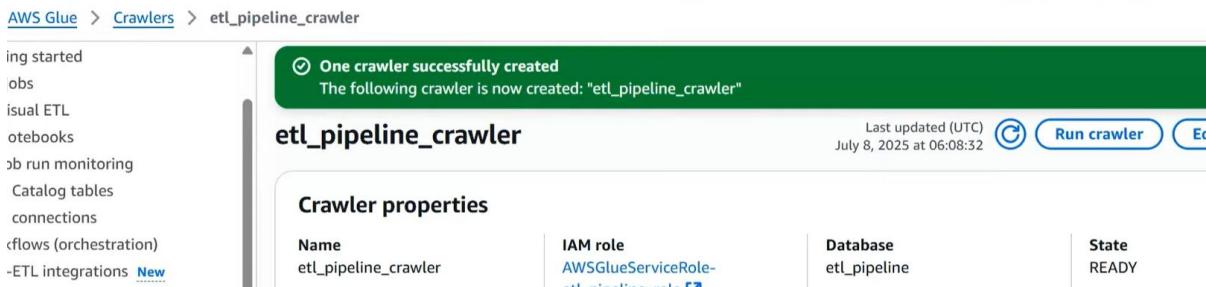
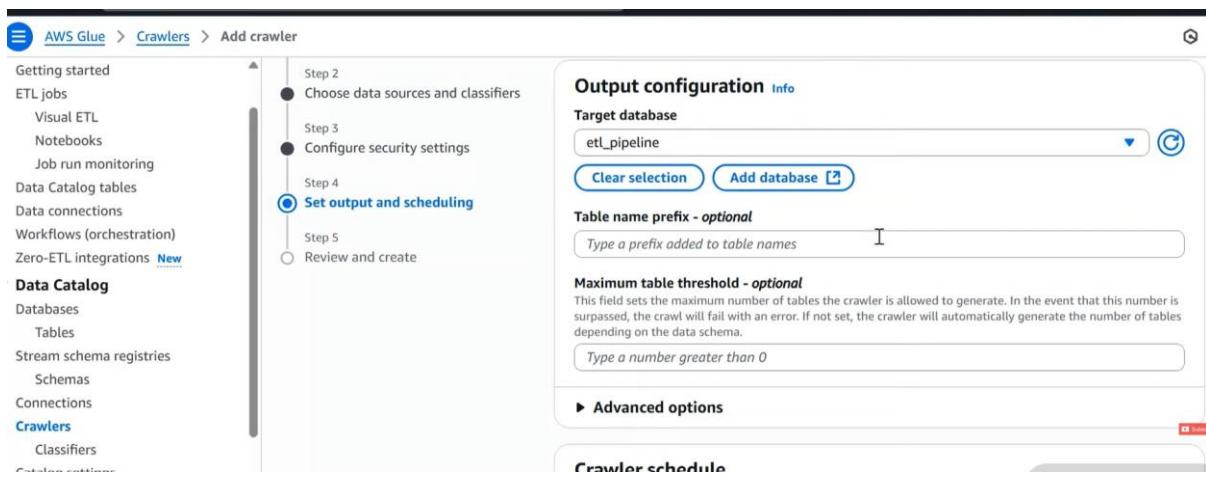
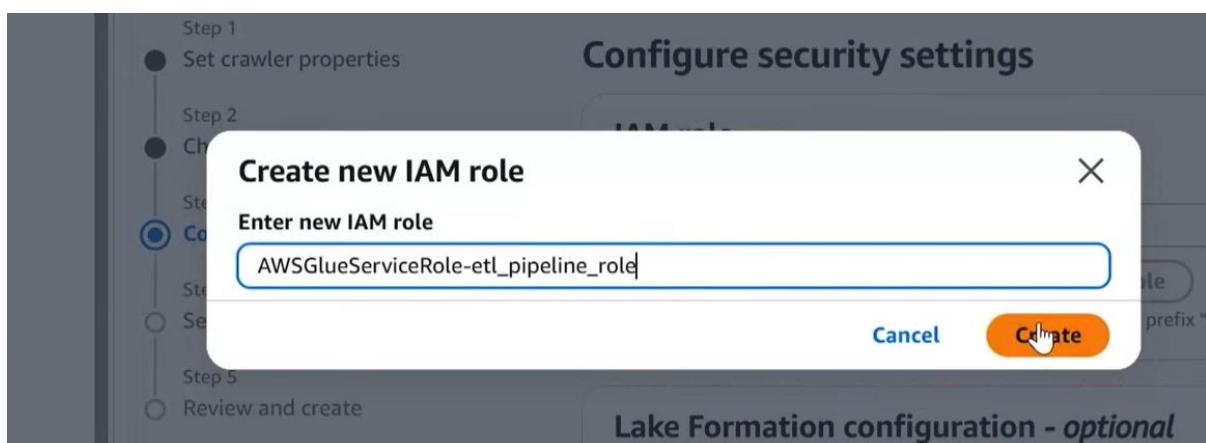
Type	Data source	Parameters
You don't have any data sources.		

Add a data source

Add datasource and select ur data source which is s3 location



Create a new role else u can use existing one but need to know for which things u have access



Lets run the crawler

The screenshot shows the AWS Glue Tables interface. On the left, there's a sidebar with navigation links like 'Getting started', 'ETL jobs', 'Visual ETL', 'Notebooks', 'Job run monitoring', 'Data Catalog tables', 'Data connections', 'Workflows (orchestration)', 'Zero-ETL Integrations', and sections for 'Data Catalog' (Databases, Tables, Stream schema registries, Schemas, Connections, Crawlers). The main area is titled 'Tables' and contains a message about new optimization features for Apache Iceberg tables. Below that, it says 'Tables (3)' and lists three tables: 'orders\_csv' (namastesql, s3://namastesql, CSV), 'orders\_parquet' (namastesql, s3://namastesql, Parquet), and 'orders\_parquet\_datalake' (etl\_pipeline, s3://namastesql, Parquet). Each table has 'Table data' and 'View' buttons.

## Table is created

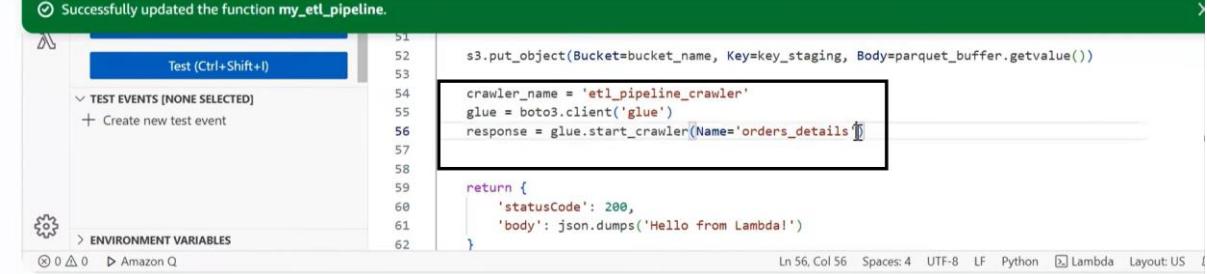
Note that it is not storing the data, data is in s3 only

Go to athen and query it

The screenshot shows the Amazon Athena Query editor. At the top, it has a search bar and navigation links for 'aws' and 'Amazon Athena'. The main area has tabs for 'Query editor' and 'Logs'. On the left, there's a sidebar with 'Data source' set to 'AwsDataCatalog', 'Catalog' set to 'None', 'Database' set to 'etl\_pipeline', and a 'Tables and views' section with a 'Create' button. Below that is a 'Tables (1)' section listing 'orders\_parquet\_datalake' and a 'Views (0)' section. The main panel shows a SQL query: 'SELECT \* FROM "AwsDataCatalog"."etl\_pipeline"."orders\_parquet\_datalake" limit 10;'. The status bar at the bottom indicates the query was completed successfully: 'Completed', 'Time in queue: 93 ms', 'Run time: 417 ms', and 'Data scanned: 1.88 KB'. The results table below shows 10 rows of data from the 'orders\_parquet\_datalake' table.

#	order_id	order_date	total_amount	customer_id	customer_name	email
1	1	2024-01-10	200.5	101	John Doe	johndoe@example.com
2	1	2024-01-10	200.5	101	John Doe	johndoe@example.com
3	2	2024-01-12	150.0	102	Jane Smith	janesmith@example.com
4	5	2024-01-14	180.0	104	Bob Brown	bobbrown@example.com
5	6	2024-01-14	250.0	105	Sania	sania@example.com
6	6	2024-01-14	250.0	105	Sania	sania@example.com
7	3	2024-01-12	120.0	103	Alice Johnson	alicejohnson@example.com

How to make this crawler automated as once as per incremental load in incoming files, I automated them to convert to parquet files now I need to make this crawler run when ever I see my parquet files

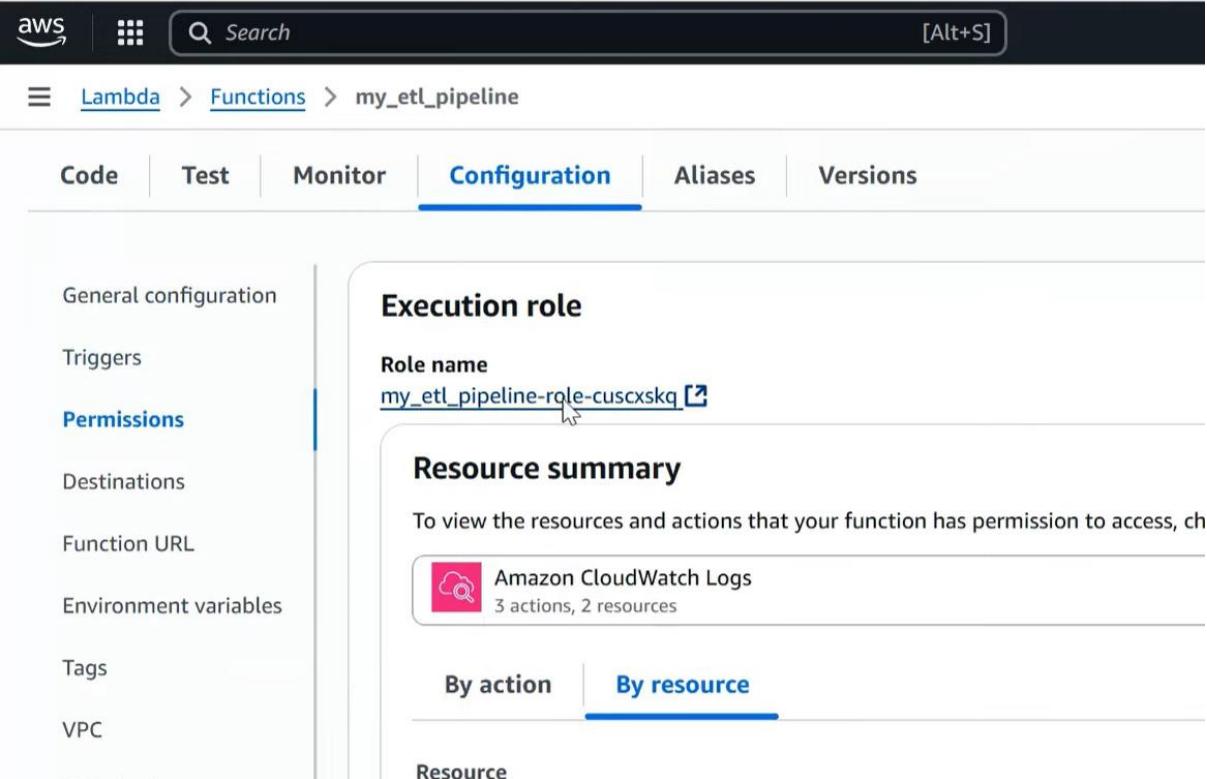


The screenshot shows the AWS Lambda function editor. At the top, a green banner says "Successfully updated the function my\_etl\_pipeline." On the left, there's a sidebar with "Test (Ctrl+Shift+I)" selected under "TEST EVENTS [NONE SELECTED]" and "Create new test event". Below that is an "ENVIRONMENT VARIABLES" section. The main area contains Python code:

```
51 s3.put_object(Bucket=bucket_name, Key=key_staging, Body=parquet_buffer.getvalue())
52
53
54 crawler_name = 'etl_pipeline_crawler'
55 glue = boto3.client('glue')
56 response = glue.start_crawler(Name='orders_details')
57
58
59 return {
60     'statusCode': 200,
61     'body': json.dumps('Hello from Lambda!')
62 }
```

At the bottom, status indicators show 0 errors, 0 warnings, and 0 info. It also shows "Amazon Q" and "Lambda" tabs.

This (lambda) IAM role doesn't have permission to run the crawler



The screenshot shows the AWS Lambda function configuration page for "my\_etl\_pipeline". The top navigation bar includes the AWS logo, search bar, and "Lambda > Functions > my\_etl\_pipeline". The "Configuration" tab is selected. On the left, a sidebar lists "General configuration", "Triggers", "Permissions" (which is selected), "Destinations", "Function URL", "Environment variables", "Tags", and "VPC". The main content area has two sections: "Execution role" (with a role name "my\_etl\_pipeline-role-cuscxskq") and "Resource summary" (listing "Amazon CloudWatch Logs" with 3 actions and 2 resources). Below the resource summary are buttons for "By action" and "By resource", with "By resource" being active.

Click on the role name and add a new permission

IAM > Roles > my\_etl\_pipeline-role-cuscxskq > Add permissions

Attach policy to my\_etl\_pipeline-role-cuscxskq

▶ Current permissions policies (2)

Other permissions policies (1/1076)

Policy name	Type	Description
<input type="checkbox"/> <a href="#">AmazonSageMakerServiceCatalogProductsGlue...</a>	AWS managed	Service role policy used by the AWS Gl...
<input type="checkbox"/> <a href="#">AWSGlueServiceNotebookRole</a>	AWS managed	Policy for AWS Glue service role which ...
<input checked="" type="checkbox"/> <a href="#">AWSGlueServiceRole</a>	AWS managed	Policy for AWS Glue service role which ...
<input type="checkbox"/> <a href="#">AWSGlueServiceRole-credit_card-EZCRC-s3Policy</a>	Customer managed	This policy will be used for Glue Crawl...

Now lets test it

Uploaded another file

orders\_json\_incoming/

Objects Properties

Objects (2) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

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
<a href="#">orders_ETL_incremental.json</a>	json	July 8, 2025, 11:46:43 (UTC+05:30)	628.0 B	Standard
<a href="#">orders_ETL.json</a>	json	July 8, 2025, 11:34:10 (UTC+05:30)	7.1 KB	Standard

Lets check

orders\_parquet\_datalake/

Objects Properties

Objects (2) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

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
<a href="#">orders_ETL_20250708_06041.parquet</a>	parquet	July 8, 2025, 11:34:11 (UTC+05:30)	8.0 KB	Standard
<a href="#">orders_ETL_20250708_06165.parquet</a>	parquet	July 8, 2025, 11:46:54 (UTC+05:30)	7.3 KB	Standard

Run the query in athena and check the count