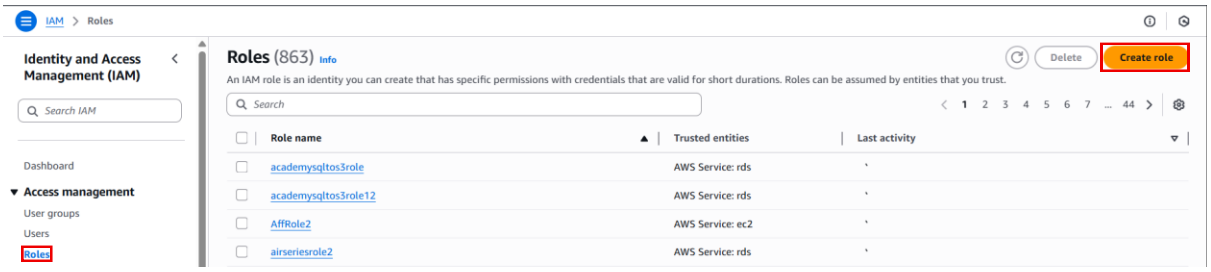
Calling a Lambda function from Amazon Kinesis

To Begin with the lab

Summary of the lab

This lab demonstrates how to connect an AWS Lambda function to a Kinesis Data Stream for real-time data processing. A Lambda role with Kinesis read and S3 write permissions is created, and a trigger is added to invoke Lambda when new records arrive. The function decodes and stores data in an S3 bucket, enabling automated, serverless stream-to-storage processing.

- **Prerequisites**
    - An **AWS Kinesis Data Stream** (e.g., myfirstdatastream)
    - An **S3 bucket** (e.g., kinesis-target-bucket01)
- Go to the **AWS Management Console** → Search **IAM** → open **Roles**.
- Click **Create role**.


IAM > Roles

Identity and Access Management (IAM)

Search IAM

Dashboard
Access management
User groups
Users
Roles

Roles (863) Info

Delete | Create role

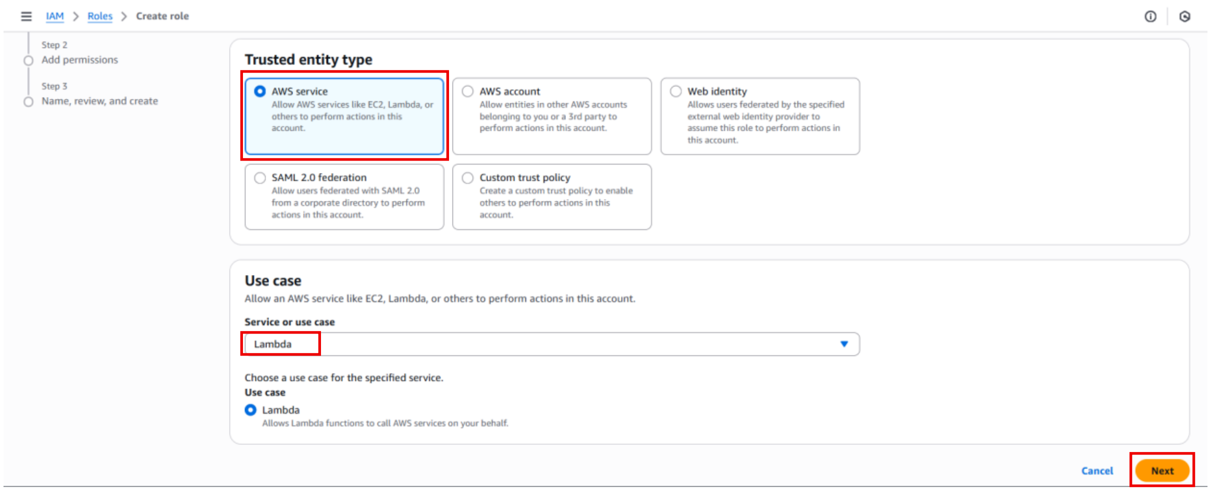
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Search

< 1 2 3 4 5 6 7 ... 44 >

Role name	Trusted entities	Last activity
academysqltos3role	AWS Service: rds	-
academysqltos3role12	AWS Service: rds	-
AffRole2	AWS Service: ec2	-
airseriesrole2	AWS Service: rds	-

- Select **Trusted entity type** → choose **AWS service**.
- Choose **Lambda** as the use case → click **Next**.


IAM > Roles > Create role

Step 2
Add permissions

Step 3
Name, review, and create

Trusted entity type

[✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.

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

Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case: Lambda

Choose a use case for the specified service.

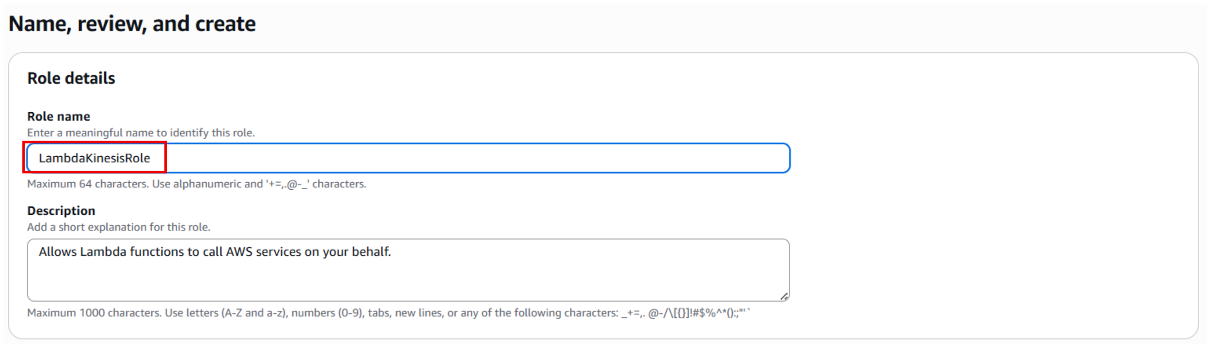
Use case

[✓ Lambda] Allows Lambda functions to call AWS services on your behalf.

Cancel | Next

- Under **Permissions**, attach the following policies:
- AmazonKinesisReadOnlyAccess
- AmazonS3FullAccess
- Click **Next**.

- Enter a role name, e.g. LambdaKinesisRole



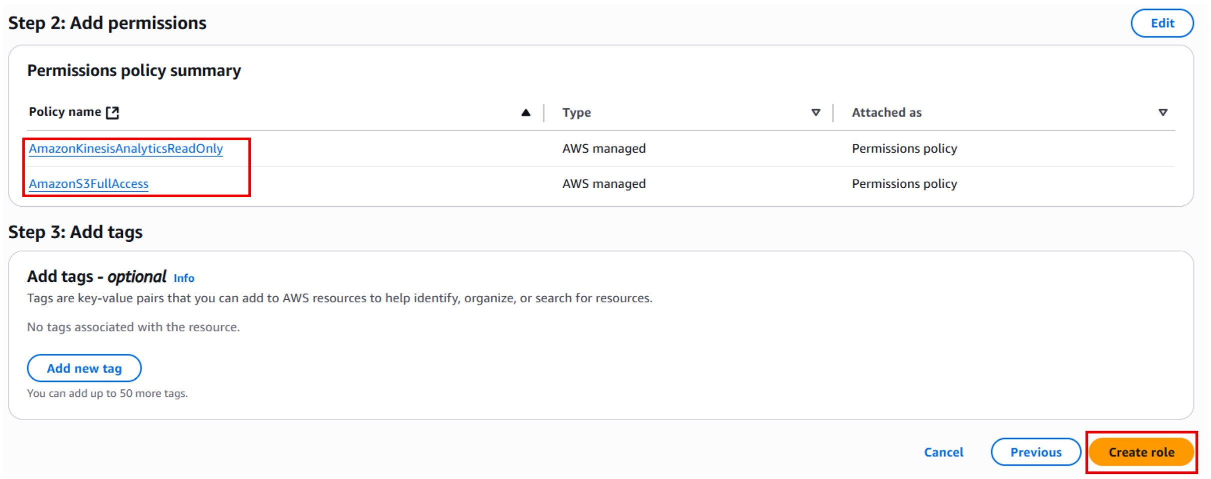
Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
LambdaKinesisRole
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 letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{()]!#$%^*();"'`

- Review and click **Create role**.



Step 2: Add permissions

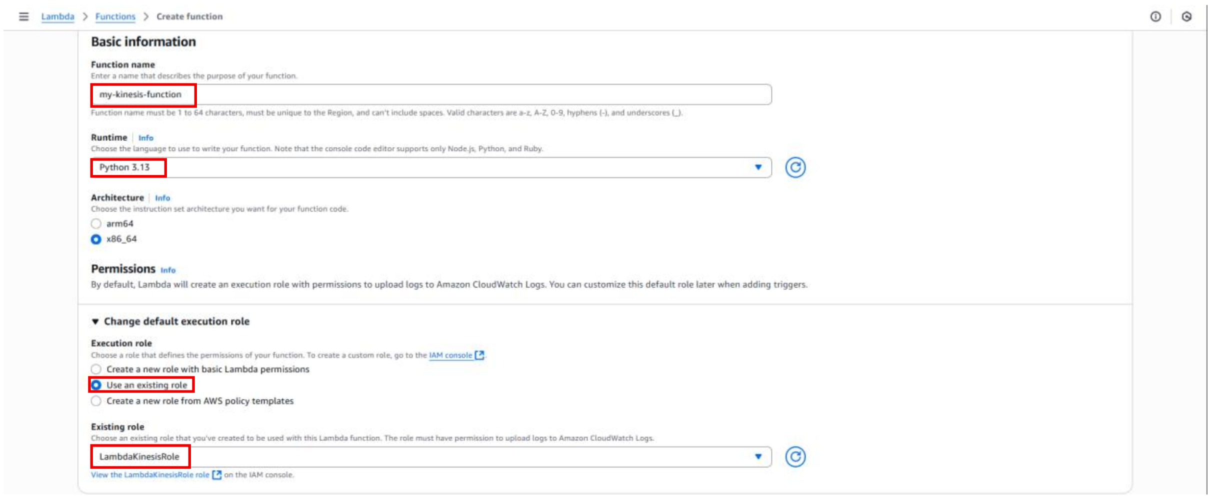
Permissions policy summary

Policy name	Type	Attached as
AmazonKinesisAnalyticsReadOnly	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - *optional*
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.
You can add up to 50 more tags.

- Navigate to **AWS Lambda → Create function**.
- Choose **Author from scratch**.
- Enter a function name, e.g.: my-kinesis-function
- Runtime: **Python 3.x**
- Under **Change default execution role**, choose:
  → **Use an existing role** → select LambdaKinesisRole.



Lambda > Functions > Create function

Basic information

Function name
Enter a name that describes the purpose of your function.
my-kinesis-function
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.13

Architecture
Choose the instruction set architecture you want for your function code.
arm64 [✓ x86_64]

Permissions
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

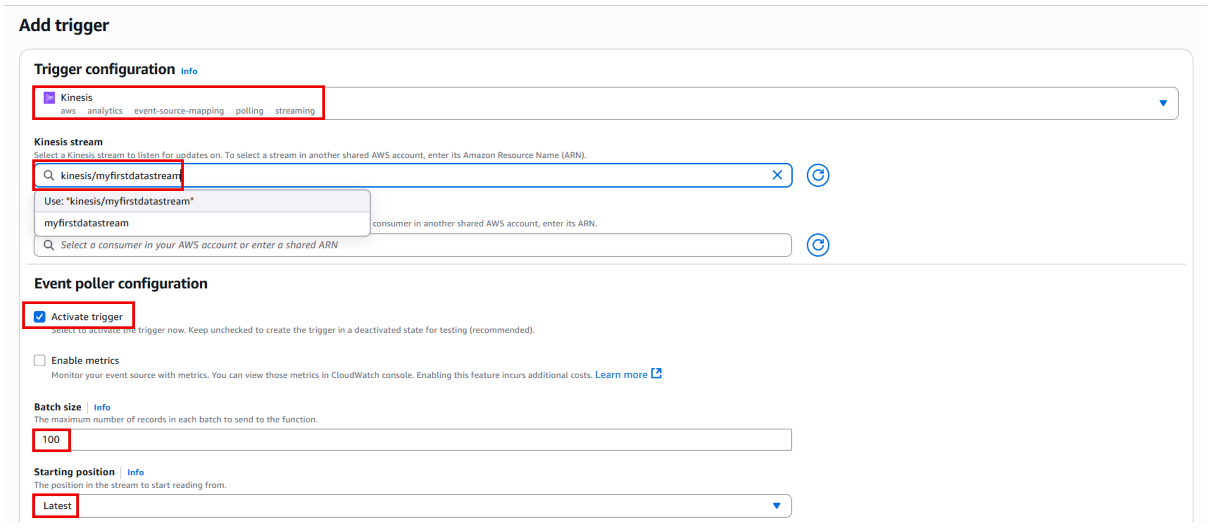
▼ Change default execution role

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.
LambdaKinesisRole
View the LambdaKinesisRole role on the IAM console.

- Click **Create function**.

- In the Lambda function page → click **Add trigger**.
- Choose **Kinesis** as the trigger source.
- Select your stream name (e.g., myfirstdatastream).
- Set **Batch size**: keep default (e.g., 100).
- Set **Starting position**:
    - Choose LATEST (process only new records).



Add trigger

Trigger configuration Info

Kinesis
aws analytics event-source-mapping polling streaming

Kinesis stream
Select a Kinesis stream to listen for updates on. To select a stream in another shared AWS account, enter its Amazon Resource Name (ARN).
kinesis/myfirstdatastream
Use: "kinesis/myfirstdatastream"
myfirstdatastream
consumer in another shared AWS account, enter its ARN.
Select a consumer in your AWS account or enter a shared ARN

Event poller configuration

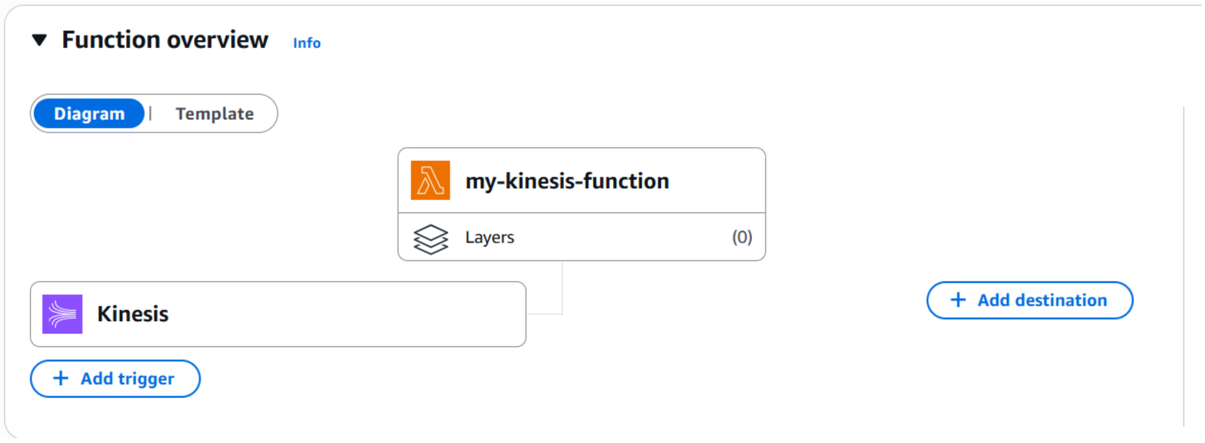
☑ Activate trigger
Select to activate the trigger now. Keep unchecked to create the trigger in a deactivated state for testing (recommended).

☐ Enable metrics
Monitor your event source with metrics. You can view those metrics in CloudWatch console. Enabling this feature incurs additional costs. Learn more

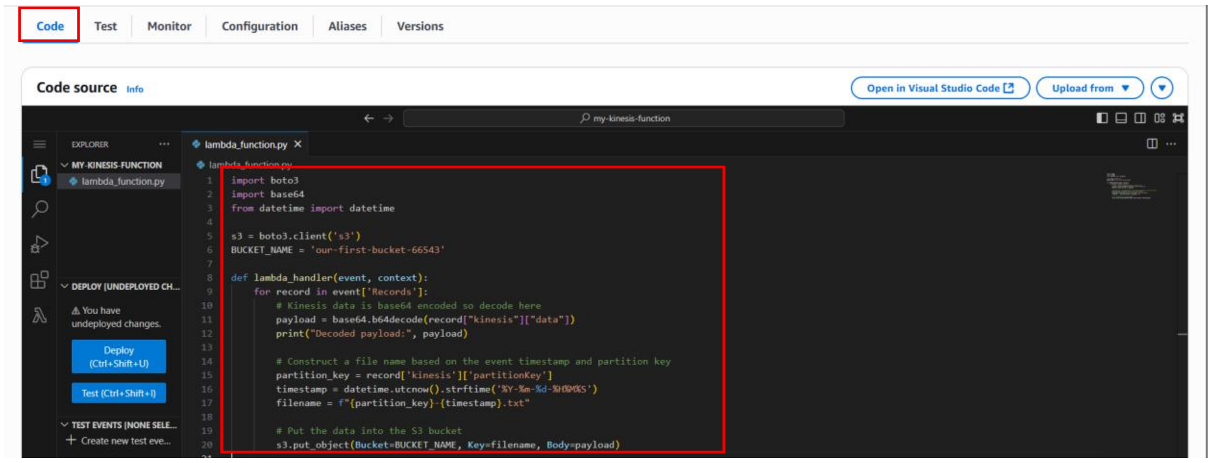
Batch size | Info
The maximum number of records in each batch to send to the function.
100

Starting position | Info
The position in the stream to start reading from.
Latest

- Acknowledge the warning checkbox → click **Add**.
- You'll now see **Kinesis** listed under **Triggers**.

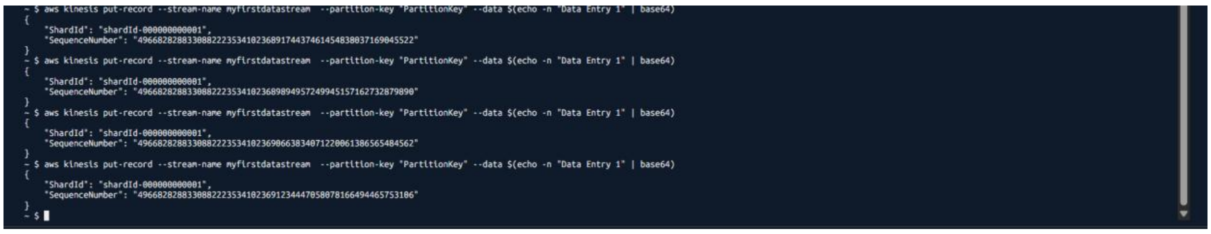


- Now go to the code tab and write the code there and Deploy it.



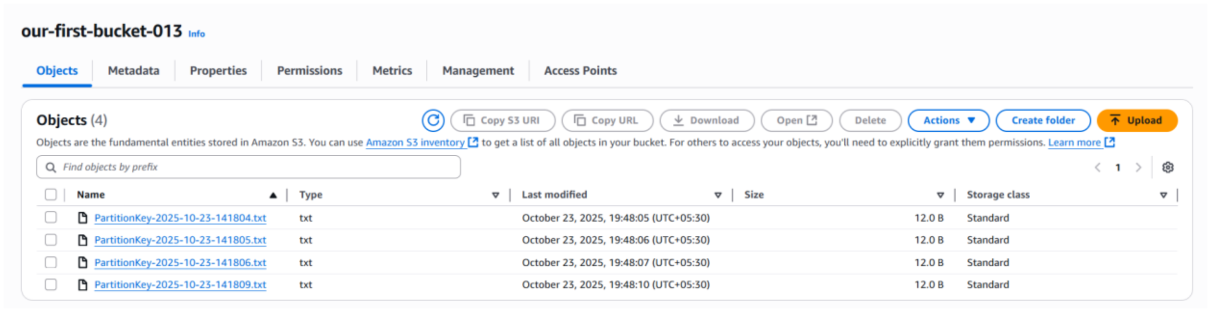
```python
import boto3
import base64
from datetime import datetime

s3 = boto3.client('s3')
BUCKET_NAME = 'our-first-bucket-66543'

def lambda_handler(event, context):
    for record in event['Records']:
        # Kinesis data is base64 encoded so decode here
        payload = base64.b64decode(record["kinesis"]["data"])
        print("Decoded payload:", payload)

        # Construct a file name based on the event timestamp and partition key
        partition_key = record['kinesis']['partitionKey']
        timestamp = datetime.utcnow().strftime('%Y-%m-%d-%H%M%S')
        filename = f"{partition_key}-{timestamp}.txt"

        # Put the data into the S3 bucket
        s3.put_object(Bucket=BUCKET_NAME, Key=filename, Body=payload)
```

- Now that Lambda is connected to your Kinesis stream, let's push new records.
- Put new records into Kinesis
- Use the AWS CLI.



```
~ $ aws kinesis put-record --stream-name myfirstdatastream  --partition-key "PartitionKey" --data $(echo -n "Data Entry 1" | base64)
{
    "ShardId": "shardId-000000000001",
    "SequenceNumber": "49668282883308822235341023689174437461454838037169045522"
}
~ $ aws kinesis put-record --stream-name myfirstdatastream  --partition-key "PartitionKey" --data $(echo -n "Data Entry 1" | base64)
{
    "ShardId": "shardId-000000000001",
    "SequenceNumber": "49668282883308822235341023689894957249945157162732879890"
}
~ $ aws kinesis put-record --stream-name myfirstdatastream  --partition-key "PartitionKey" --data $(echo -n "Data Entry 1" | base64)
{
    "ShardId": "shardId-000000000001",
    "SequenceNumber": "49668282883308822235341023690663834071220061386565484562"
}
~ $ aws kinesis put-record --stream-name myfirstdatastream  --partition-key "PartitionKey" --data $(echo -n "Data Entry 1" | base64)
{
    "ShardId": "shardId-000000000001",
    "SequenceNumber": "49668282883308822235341023691234447058078166494465753106"
}
~ $ 
```

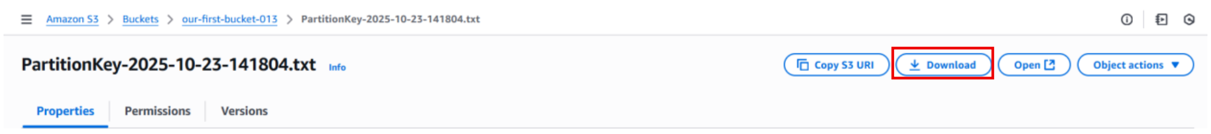
- Now that we can see the records in S3 Buckets



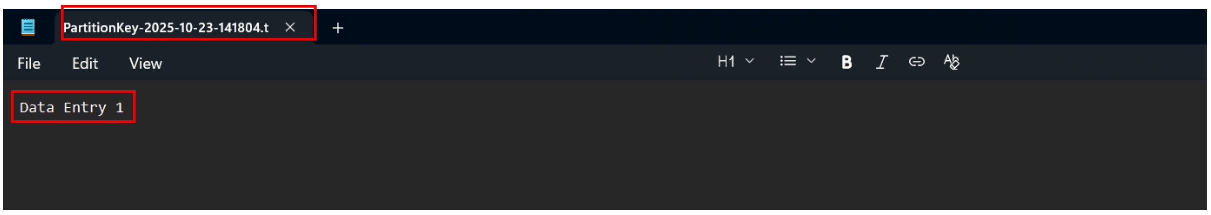
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
PartitionKey-2025-10-23-141804.txt	txt	October 23, 2025, 19:48:05 (UTC+05:30)	12.0 B	Standard
PartitionKey-2025-10-23-141805.txt	txt	October 23, 2025, 19:48:06 (UTC+05:30)	12.0 B	Standard
PartitionKey-2025-10-23-141806.txt	txt	October 23, 2025, 19:48:07 (UTC+05:30)	12.0 B	Standard
PartitionKey-2025-10-23-141809.txt	txt	October 23, 2025, 19:48:10 (UTC+05:30)	12.0 B	Standard

- Click on one of the records and download it.


Amazon S3 > Buckets > our-first-bucket-013 > PartitionKey-2025-10-23-141804.txt
PartitionKey-2025-10-23-141804.txt Info
Copy S3 URI
Download
Open
Object actions
Properties
Permissions
Versions

- You can see that it has downloaded the same record that was pushed



```
Data Entry 1
```