

Handle AWS Part

- 1.Create S3 Bucket
- 2.Create IAM Policy for kafka and spark Streaming
- 3.Create IAM User and assign polices
- 4.Create Access token and secret Key
- 5.Prepare Full Needed Connection Info
- 6.Create IAM Policy for Snowflake
- 7.Create IAM Role for Snowflake
- 8.Additional Steps To make connection with snowFlake

1.Create S3 Bucket with the following settings to use as staging layer

Make Prefix Kafka/ for kafka and spark data

The screenshot displays the AWS S3 console interface. The top section is the 'Create bucket' wizard, which is divided into two main parts: 'General configuration' and 'Object Ownership'. In the 'General configuration' section, the 'AWS Region' is set to 'Europe (Stockholm) eu-north-1'. The 'Bucket type' is set to 'General purpose'. The 'Bucket name' is 'amzn-s3-demo-bucket'. The 'Copy settings from existing bucket' section is optional, and the 'Choose bucket' button is highlighted. The 'Object Ownership' section shows 'ACLs disabled (recommended)' as the selected option, with 'Bucket owner enforced' as the 'Object Ownership' setting. The bottom section shows the details of the 'kafka-staging-abdelrahman-2025' bucket. The 'Objects' tab is selected, showing a list of objects. The list has columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. There is one object named 'kafka/' of type 'Folder'. The 'Actions' dropdown menu is open, showing options like 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload'.

Create bucket [Info](#)
Buckets are containers for data stored in S3.

General configuration

AWS Region
Europe (Stockholm) eu-north-1

Bucket type [Info](#)

☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)
amzn-s3-demo-bucket
Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)
Format: s3://bucket/prefix

Object Ownership [Info](#)
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Object Ownership

☒ **ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ **ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Amazon S3 > Buckets > kafka-staging-abdelrahman-2025

Amazon S3

Buckets
General purpose buckets
Directory buckets
Table buckets
Vector buckets

Access management and security
Access Points
Access Points for FSx
Access Grants

kafka-staging-abdelrahman-2025 [Info](#)

Objects | Metadata | Properties | Permissions | Metrics | Management | Access Points

Objects (1)
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	kafka/	Folder	-	-	-

2. Create IAM Policy for kafka and spark Streaming Using Json

Policy name S3andSpark

```
{  
  
    "Version": "2012-10-17",
```

```

    "Statement": [
      {
        "Sid": "AllowListBucketRoot",
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::kafka-staging-abdelrahman-2025"
      },
      {
        "Sid": "AllowListSpecificPrefix",
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::kafka-staging-abdelrahman-2025",
        "Condition": {
          "StringLike": {
            "s3:prefix": [
              "kafka/flights/*",
              "kafka/flights/_spark_metadata/*"
            ]
          }
        }
      },
      {
        "Sid": "AllowObjectOperations",
        "Effect": "Allow",
        "Action": [
          "s3:GetObject",
          "s3:PutObject",
          "s3:DeleteObject"
        ],
        "Resource": "arn:aws:s3:::kafka-staging-abdelrahman-2025/*"
      }
    ]
  }
}

```

Also, I created another one with full access for testing
FullS3Access

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullS3Access",

```

```

    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": [
        "arn:aws:s3:::kafka-staging-abdelrahman-2025",
        "arn:aws:s3:::kafka-staging-abdelrahman-2025/*"
    ]
}
]
}

```

3.Create IAM User and assign policies

Create user “kafka_local_writer” with the following setting and choose the policy created

The screenshot shows the 'Set permissions' step in the AWS IAM console. On the left, a progress bar indicates four steps: 'Specify user details', 'Set permissions' (current), 'Review and create', and 'Create user'. The main content area is titled 'Set permissions' and includes a sub-header 'Permissions options' with three radio buttons: 'Add user to group', 'Copy permissions', and 'Attach policies directly' (which is selected). Below this is a section for 'Permissions policies (1429)' with a search bar containing 's3and' and a dropdown for 'Filter by Type' set to 'All types'. A table lists policies, with 'S3andSpark' highlighted. At the bottom, there is a 'Set permissions boundary - optional' section and 'Cancel', 'Previous', and 'Next' buttons.

4. Create an Access token and a secret Key

From the user page, go to Security credentials and Create access key

The screenshot shows the AWS IAM console for the user **kafka_local_writer**. The left sidebar contains navigation links for Identity and Access Management (IAM), Access management, Access reports, and IAM Identity Center. The main content area shows the user's details under the **Security credentials** tab. Key information includes the user's ARN, creation date, console access status (disabled), and a list of access keys. One access key is shown with ID **AKIAQ6Z3C...** and status **Active**. The **Create access key** button is circled in red.

Access key best practices & alternatives [info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

- ☐ **Command Line Interface (CLI)**
You plan to use this access key to enable the AWS CLI to access your AWS account.
- ☐ **Local code**
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- ☐ **Application running on an AWS compute service**
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- ☐ **Third-party service**
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- ☒ **Application running outside AWS**
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.
- ☐ **Other**
Your use case is not listed here.

⚠️ Alternative recommended
Use IAM Roles Anywhere to generate temporary security credentials for non AWS workloads accessing AWS services. [Learn more about providing access for non AWS workloads.](#)

5. Prepare Full Needed Connection Info

```
{
  "connector.class": "io.confluent.connect.s3.S3SinkConnector",
```

```

"tasks.max": "1",
"topics": "my-topic",
"s3.bucket.name": "kafka-staging-abdelrahman-2025",
"s3.region": "eu-north-1",
"aws.access.key.id": "AKIAQ6Z3*****",
"aws.secret.access.key": "WMzzoVfcbBW*****",
"s3.part.size": 5242880,
"flush.size": 1000,
"storage.class": "io.confluent.connect.s3.storage.S3Storage",
"format.class": "io.confluent.connect.s3.format.csv.CsvFormat",
"s3.prefix": "kafka/"
}

```

and you can test the connection by run this code in your local machine

```

from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("WriteToS3") \
    .config("spark.hadoop.fs.s3a.impl",
"org.apache.hadoop.fs.s3a.S3AFileSystem") \
    .config("spark.hadoop.fs.s3a.access.key", "AKIAQ6Z3*****") \
    .config("spark.hadoop.fs.s3a.secret.key",
"WMzzoVfcbBW*****") \
    .config("spark.hadoop.fs.s3a.endpoint", "s3.eu-north-1.amazonaws.com") \
    .config("spark.hadoop.fs.s3a.path.style.access", "true") \
    .config("spark.hadoop.fs.s3a.aws.credentials.provider",
"org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider") \
    .config("spark.hadoop.fs.s3a.fast.upload", "true") \
    .getOrCreate()

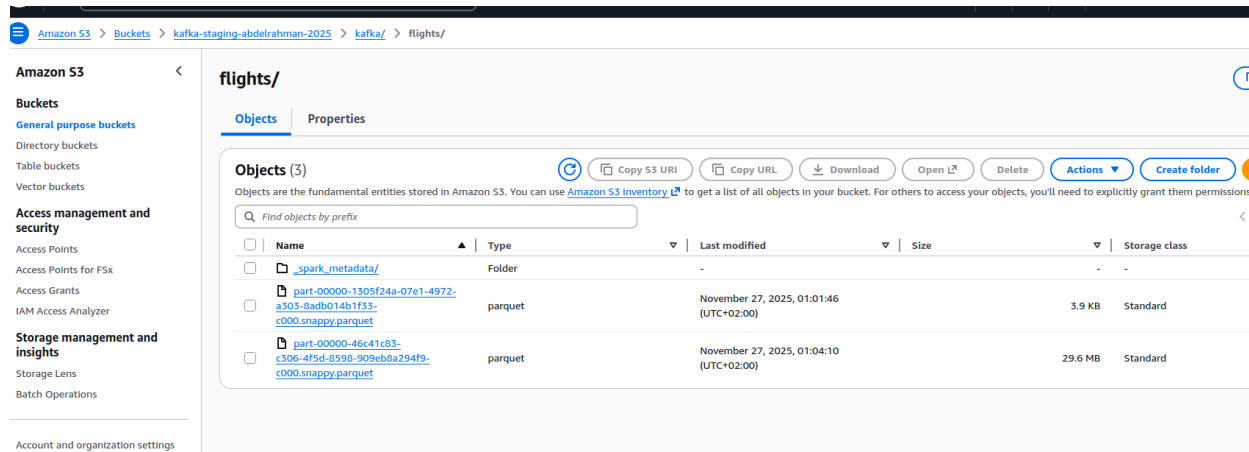
spark.sparkContext.setLogLevel("ERROR")

data = [("Alice", 25), ("Bob", 30)]
df = spark.createDataFrame(data, ["name", "age"])

df.write.mode("overwrite").parquet("s3a://kafka-staging-abdelrahman-2025/kafka/test-staging")

```

Tell this your data Should be uploaded normally into the staging layer in S3



During these steps, we encountered several different problems. For example, I modified the policy more than ten times, and there were problems with the Access Token, Connection config, and others. One of the most significant problems was that the signature had expired.

6. Create IAM Policy for Snowflake

Snowflakegetobj

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::kafka-staging-abdelrahman-2025",
        "arn:aws:s3:::kafka-staging-abdelrahman-2025/kafka/*"
      ]
    }
  ]
}
```

7. Create IAM Role for Snowflake

First, create a Role AWS Account without an external ID

IAM > Roles > Create role

Step 1

Select trusted entity

Step 2

Add permissions

Step 3

Name, review, and create

Select trusted entity

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.

An AWS account

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

☒ This account (066158986333)

☐ Another AWS account

Options

☐ Require external ID (Best practice when a third party will assume this role)

☐ Require MFA

Requires that the assuming entity use multi-factor authentication.

Add the policy

IAM > Roles > Create role

Step 1

Select trusted entity

Step 2

Add permissions

Step 3

Name, review, and create

Add permissions

Permissions policies (1/1101)

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

Q snow

Filter by Type

All types

1 match


☒ Policy name

☒ Snowflakegetobj

Customer managed

-

Set permissions boundary - optional

aws  Search [Alt+S]

IAM > Roles > Create role

Step 3
Name, review, and create

Role name
Enter a meaningful name to identify this role.
SnowflakeS3ReadRole
Maximum 64 characters. Use alphanumeric and '+', '@', '-' characters.

Description
Add a short explanation for this role.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: '_', '+', '@', '/', '(', ')', '#', '%', '^', '!', ':', '&'.

Step 1: Select trusted entities


Trust policy

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "sts:AssumeRole",
7       "Principal": {
8         "AWS": "066158986333"
9       },
10      "Condition": {}
11    }
12  ]
13 }
```

Step 2: Add permissions

Permissions policy summary


Policy name 	Type	Attached as
Snowflakegetobj	Customer managed	Permissions policy

8. Additional Steps to make a connection with Snowflake

Now the Role is ready. We need the Role ARN

SnowflakeS3ReadRole [Info](#)

Summary

Creation date November 21, 2025, 14:39 (UTC+02:00)	ARN  arn:aws:iam::066158986333:role/SnowflakeS3ReadRole
--	--

Next, go to Snowflake and run the following

```
DESC INTEGRATION KAFKA_S3_INT;
```

```

CREATE OR REPLACE STORAGE INTEGRATION KAFKA_S3_INT
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = S3
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN = arn:aws:iam::066158986333:role/SnowflakeS3ReadRole
  STORAGE_ALLOWED_LOCATIONS =
  ('s3://kafka-staging-abdelrahman-2025/kafka/');
```

```
DESC INTEGRATION KAFKA_S3_INT;
```


Then a Table will be shown we need the following information from it
External ID and Principal code
Then, go to the Role and edit Trusted entities :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<aws-snowflake arn code >:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<ExternalId>"
        }
      }
    }
  ]
}
```

Now you can see the data in Snowflake

The screenshot displays the Snowflake Database Explorer interface. On the left, the 'Databases' sidebar shows a tree view with 'SNOWFLAKE' expanded, listing various databases like ACCOUNT_USAGE, ALERT, BCR_ROLLOUT, CORE, CORTEX, DATA_PRIVACY, DATA_SHARING_USAGE, DEFAULT_IMAGE_STORE, IMAGES, INFORMATION_SCHEMA, LOCAL, MARKETPLACE_NOTIFICATION, and ML. The main panel shows the 'GP / RAW / KAFKA_STREAM' stage. It indicates it's an 'External Stage' owned by 'ACCOUNTADMIN' and was created '2 days ago'. The 'Stage Files' tab is active, showing a table of files in the 'KAFKA_STREAM' stage (4 files total). The table has columns for NAME, SIZE, and LAST MODIFIED. The files listed are: '_spark_metadata' (500.0B, 1 hour ago), 'part-00000-46c41c83-c306-4f5d-8598-909eb8a294f9-c000.snappy.parq...' (29.6MB, 1 hour ago), and 'part-00000-1305f24a-07e1-4972-a303-8adb014b1f33-c000.snappy.parq...' (3.9KB, 1 hour ago). Each row has a three-dot menu icon for actions.

NAME	SIZE	LAST MODIFIED
_spark_metadata	500.0B	1 hour ago
part-00000-46c41c83-c306-4f5d-8598-909eb8a294f9-c000.snappy.parq...	29.6MB	1 hour ago
part-00000-1305f24a-07e1-4972-a303-8adb014b1f33-c000.snappy.parq...	3.9KB	1 hour ago