Advanced Data Engineering in Cloud

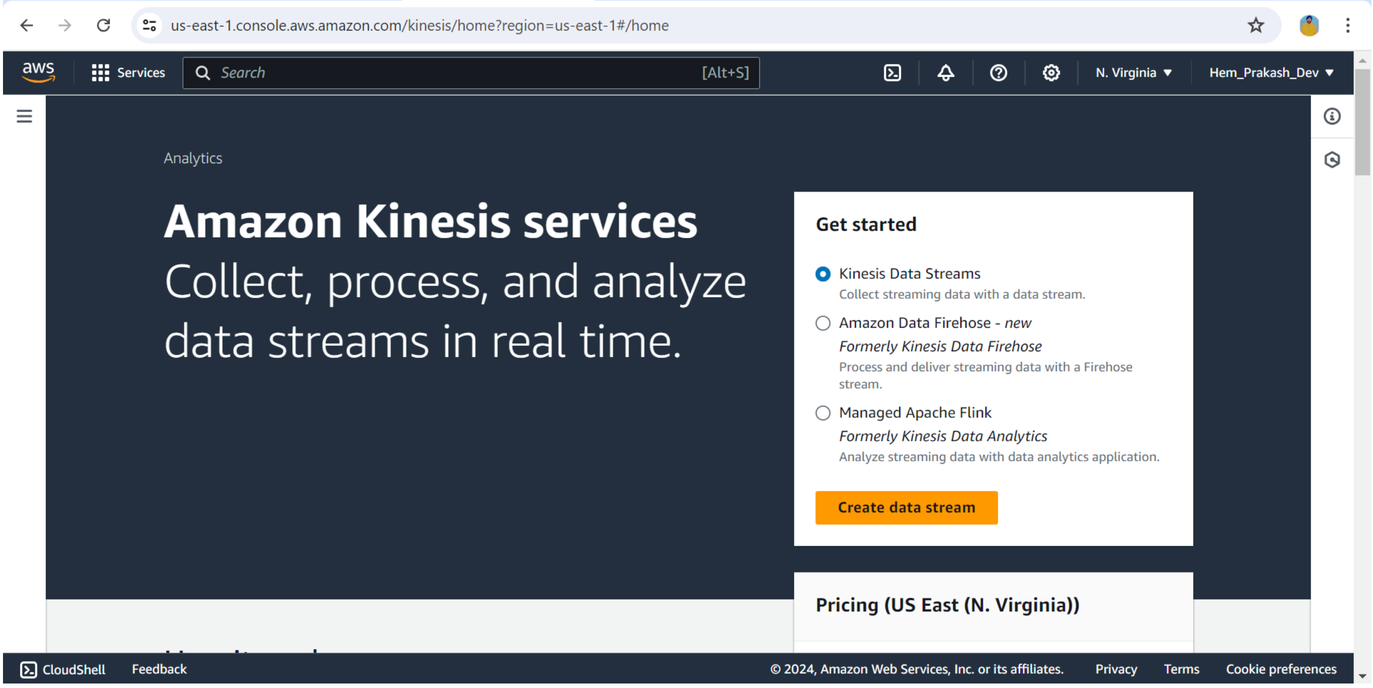
ASSIGNMENT-2

Data Ingestion and Processing

Hem Prakash Dev **Roll-G23AI1054**

1. **Implement the data ingestion mechanism using AWS Kinesis Data Streams or AWS Direct Connect to stream data from a source to Amazon S3.**

   a. **Create a Kinesis Data Stream**:



us-east-1.console.aws.amazon.com/kinesis/home?region=us-east-1#/home

Analytics

Amazon Kinesis services
Collect, process, and analyze data streams in real time.

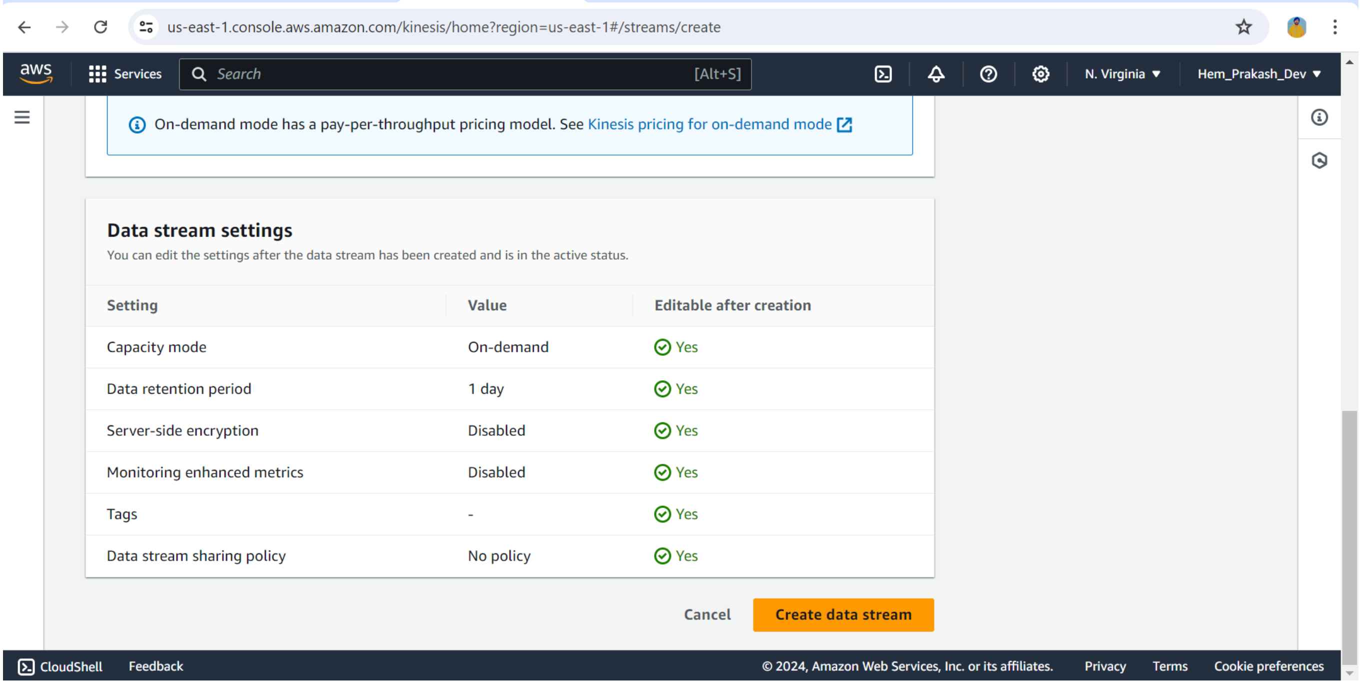
Get started

● Kinesis Data Streams
Collect streaming data with a data stream.

○ Amazon Data Firehose - *new*
Formerly Kinesis Data Firehose
Process and deliver streaming data with a Firehose stream.

○ Managed Apache Flink
Formerly Kinesis Data Analytics
Analyze streaming data with data analytics application.

b.

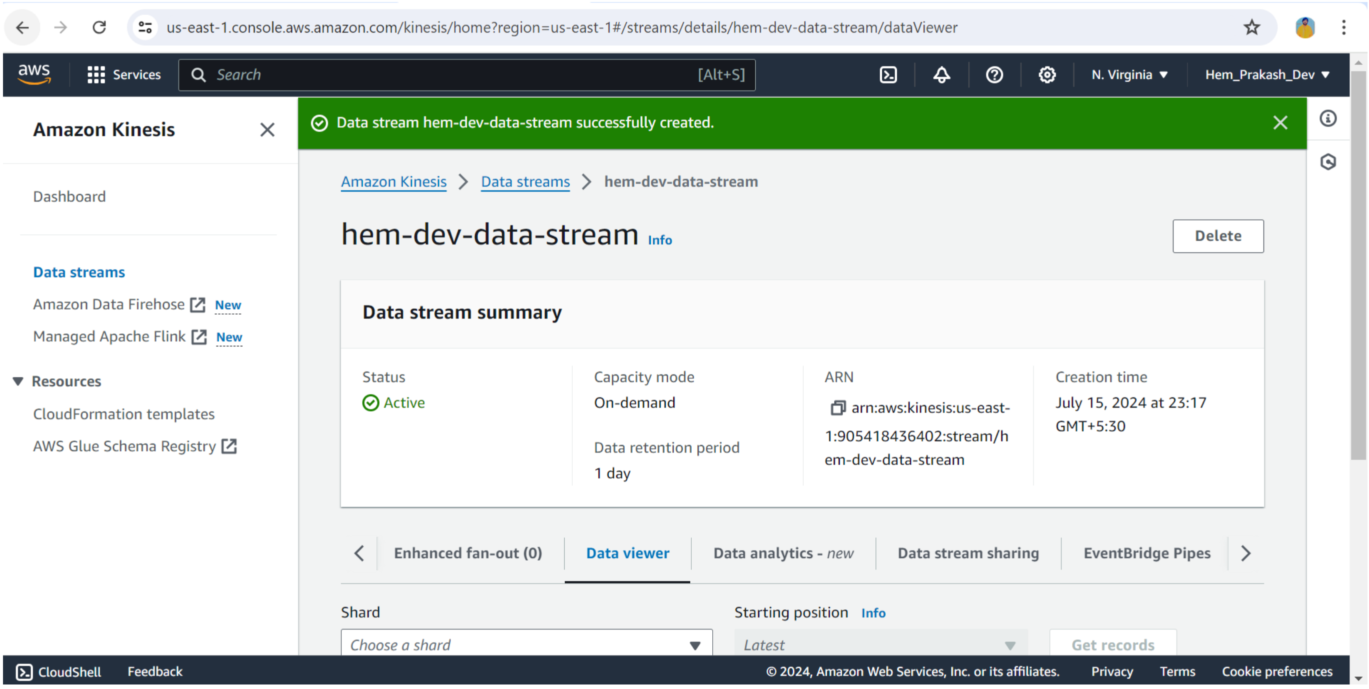


us-east-1.console.aws.amazon.com/kinesis/home?region=us-east-1#/streams/create

On-demand mode has a pay-per-throughput pricing model. See Kinesis pricing for on-demand mode

Data stream settings
You can edit the settings after the data stream has been created and is in the active status.

Setting	Value	Editable after creation
Capacity mode	On-demand	✓ Yes
Data retention period	1 day	✓ Yes
Server-side encryption	Disabled	✓ Yes
Monitoring enhanced metrics	Disabled	✓ Yes
Tags	-	✓ Yes
Data stream sharing policy	No policy	✓ Yes



us-east-1.console.aws.amazon.com/kinesis/home?region=us-east-1#/streams/details/hem-dev-data-stream/dataViewer

Data stream hem-dev-data-stream successfully created.

Amazon Kinesis > Data streams > hem-dev-data-stream

hem-dev-data-stream Info

Data stream summary

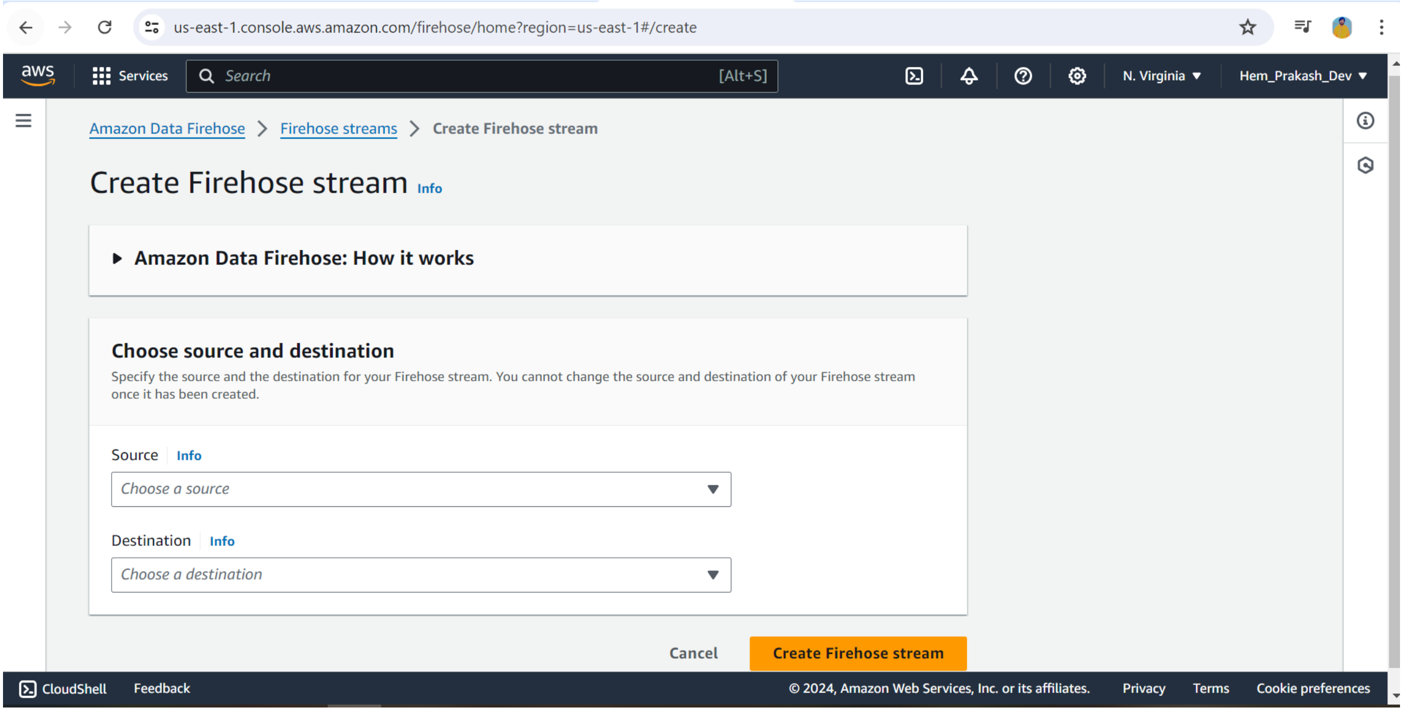
Status	Capacity mode	ARN	Creation time
Active	On-demand	arn:aws:kinesis:us-east-1:905418436402:stream/hem-dev-data-stream	July 15, 2024 at 23:17 GMT+5:30
	Data retention period 1 day		

Enhanced fan-out (0) | Data viewer | Data analytics - *new* | Data stream sharing | EventBridge Pipes

Shard: *Choose a shard*

Starting position Info: *Latest*

b. stream Data to S3 using Kinesis Firehose:



us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/create

Amazon Data Firehose > Firehose streams > Create Firehose stream

Create Firehose stream Info

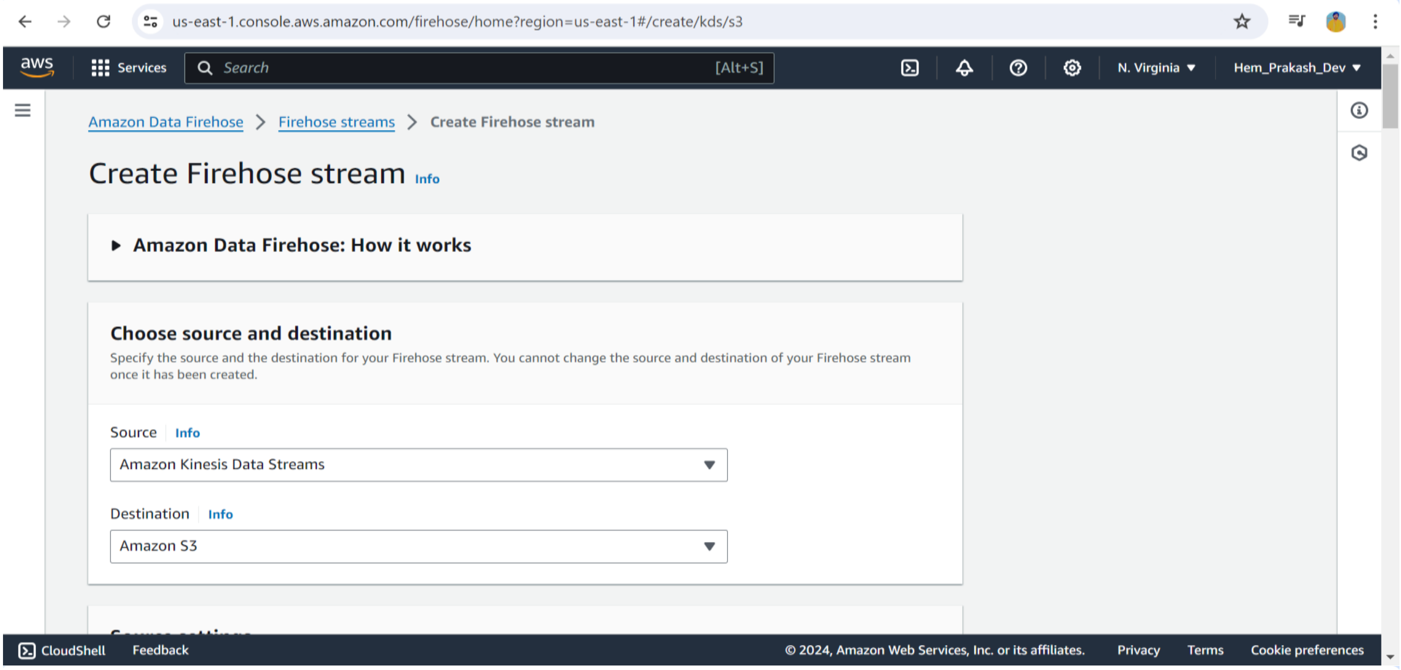
Amazon Data Firehose: How it works

Choose source and destination

Specify the source and the destination for your Firehose stream. You cannot change the source and destination of your Firehose stream once it has been created.

Source Info: *Choose a source*

Destination Info: *Choose a destination*


us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/create/kds/s3
aws
Services
Search
[Alt+S]
N. Virginia ▼
Hem_Prakash_Dev ▼
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Amazon Data Firehose > Firehose streams > Create Firehose stream

Create Firehose stream Info

▸ **Amazon Data Firehose: How it works**

Choose source and destination

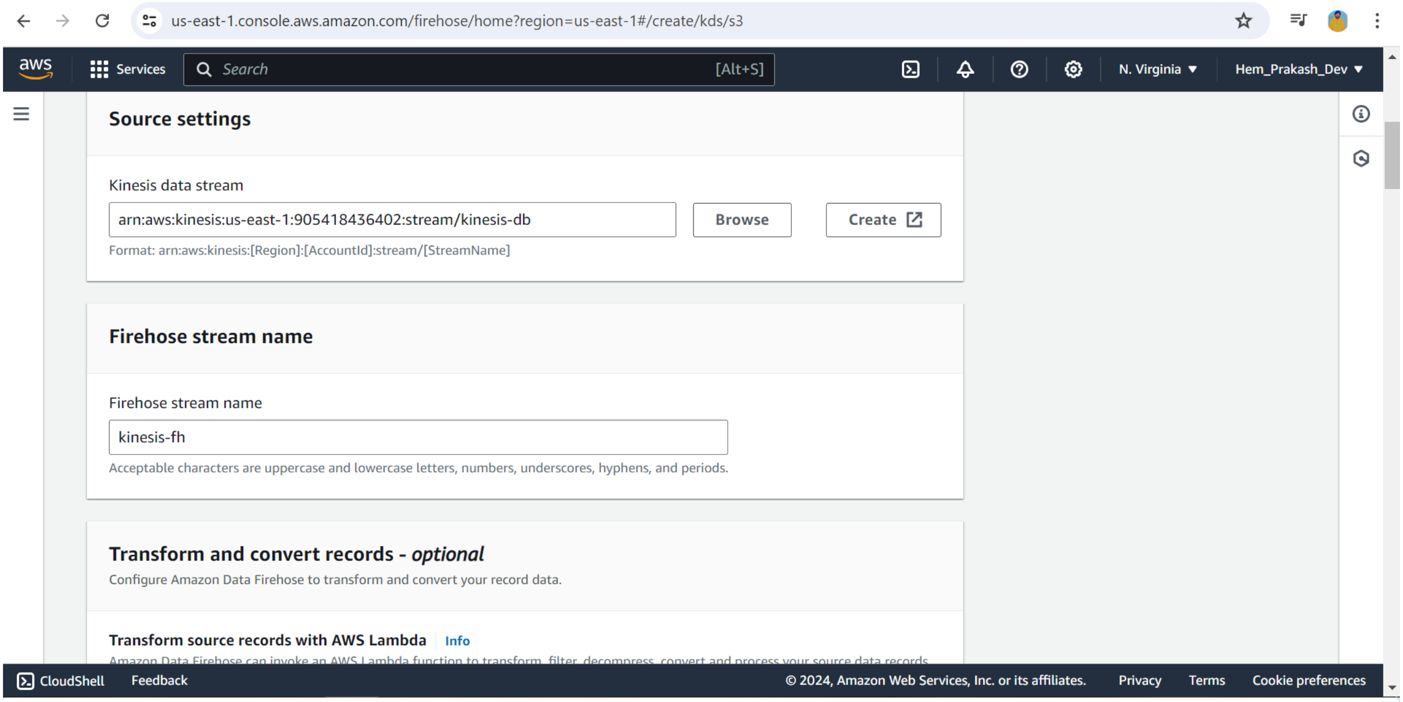
Specify the source and the destination for your Firehose stream. You cannot change the source and destination of your Firehose stream once it has been created.

Source Info

Amazon Kinesis Data Streams ▼

Destination Info

Amazon S3 ▼


us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/create/kds/s3
aws
Services
Search
[Alt+S]
N. Virginia ▼
Hem_Prakash_Dev ▼
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Source settings

Kinesis data stream

arn:aws:kinesis:us-east-1:905418436402:stream/kinesis-db

Browse Create

Format: arn:aws:kinesis:[Region]:[AccountId]:stream/[StreamName]

Firehose stream name

Firehose stream name

kinesis-fh

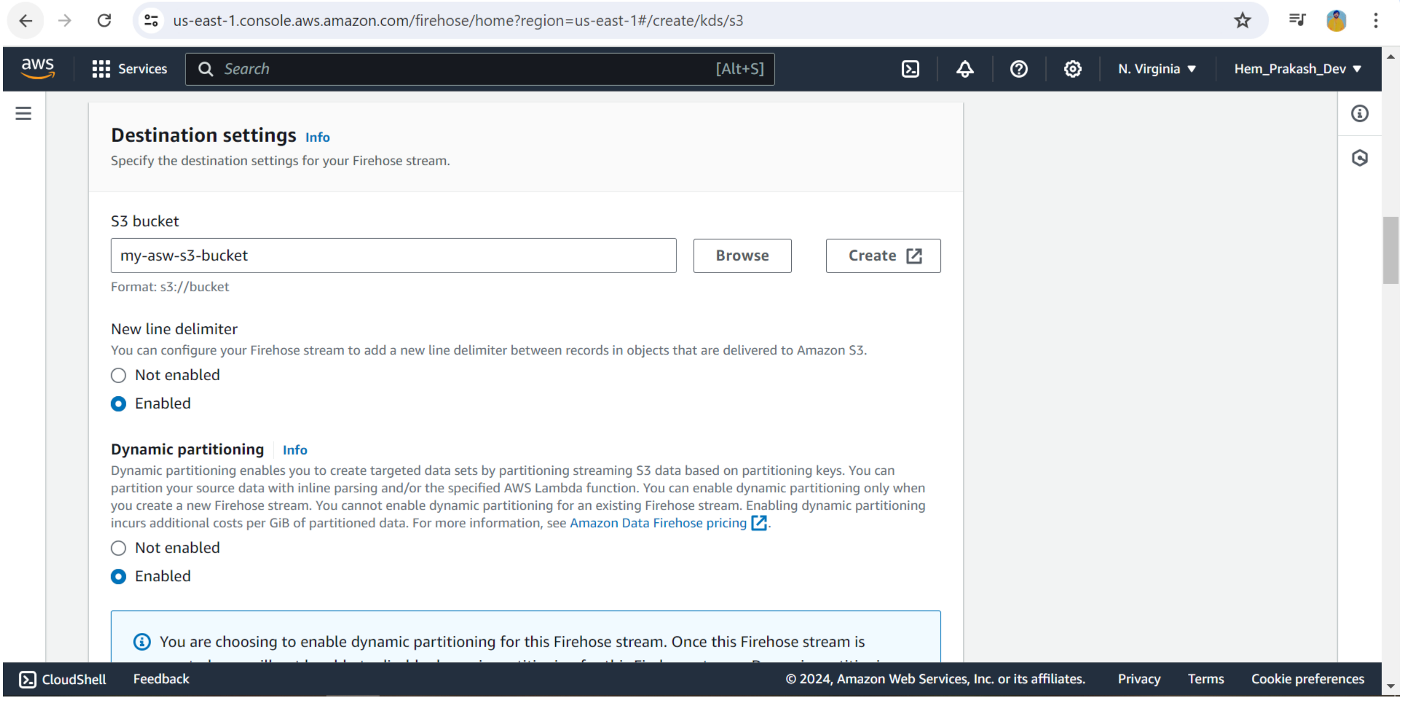
Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.

Transform and convert records - *optional*

Configure Amazon Data Firehose to transform and convert your record data.

Transform source records with AWS Lambda Info

Amazon Data Firehose can invoke an AWS Lambda function to transform, filter, decompress, convert and process your source data records.


us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/create/kds/s3
aws
Services
Search
[Alt+S]
N. Virginia ▼
Hem_Prakash_Dev ▼
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Destination settings Info

Specify the destination settings for your Firehose stream.

S3 bucket

my-asw-s3-bucket

Browse Create

Format: s3://bucket

New line delimiter

You can configure your Firehose stream to add a new line delimiter between records in objects that are delivered to Amazon S3.

○ Not enabled

● Enabled

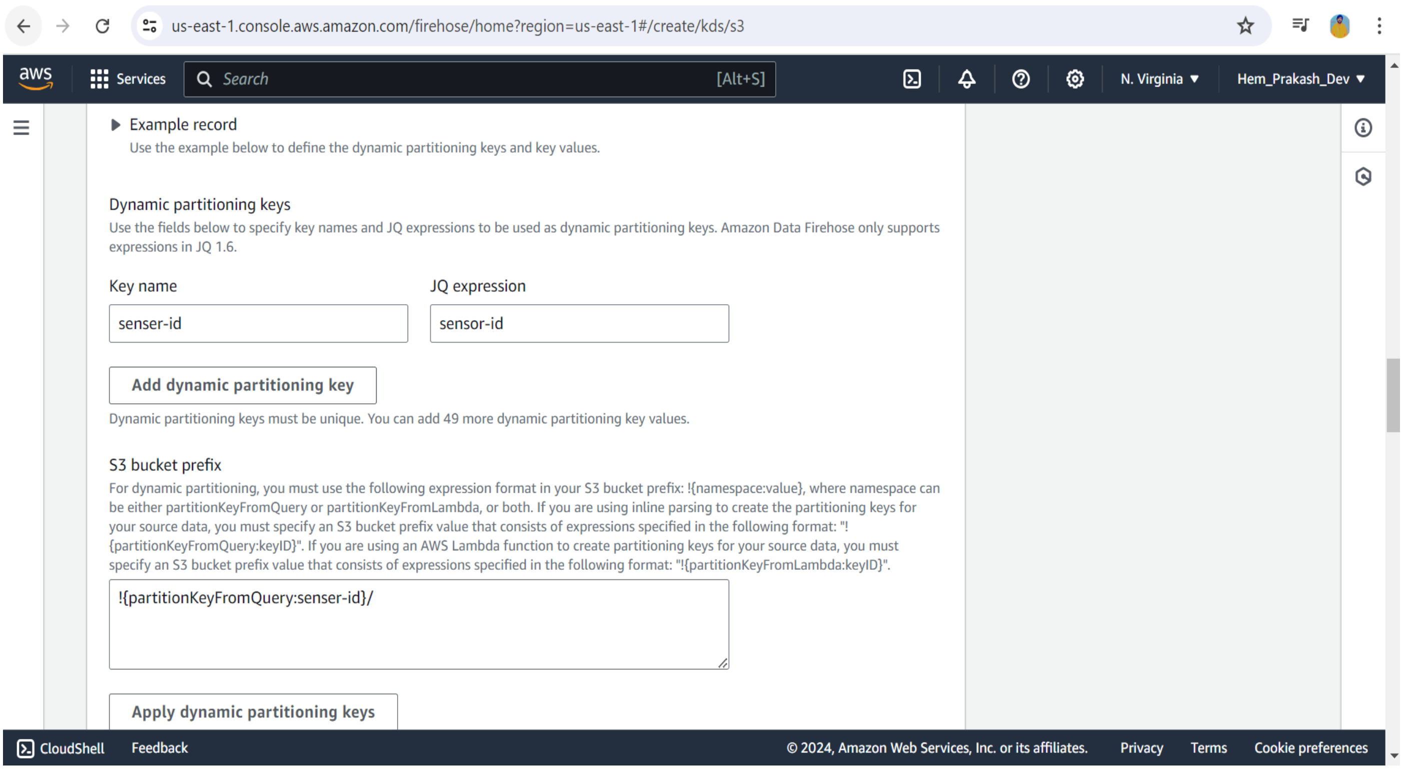
Dynamic partitioning Info

Dynamic partitioning enables you to create targeted data sets by partitioning streaming S3 data based on partitioning keys. You can partition your source data with inline parsing and/or the specified AWS Lambda function. You can enable dynamic partitioning only when you create a new Firehose stream. You cannot enable dynamic partitioning for an existing Firehose stream. Enabling dynamic partitioning incurs additional costs per GiB of partitioned data. For more information, see Amazon Data Firehose pricing.

○ Not enabled

● Enabled

ⓘ You are choosing to enable dynamic partitioning for this Firehose stream. Once this Firehose stream is


us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/create/kds/s3
aws
Services
Search
[Alt+S]
N. Virginia
Hem_Prakash_Dev
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

▶ Example record

Use the example below to define the dynamic partitioning keys and key values.

Dynamic partitioning keys

Use the fields below to specify key names and JQ expressions to be used as dynamic partitioning keys. Amazon Data Firehose only supports expressions in JQ 1.6.

Key name	JQ expression
senser-id	sensor-id

Add dynamic partitioning key

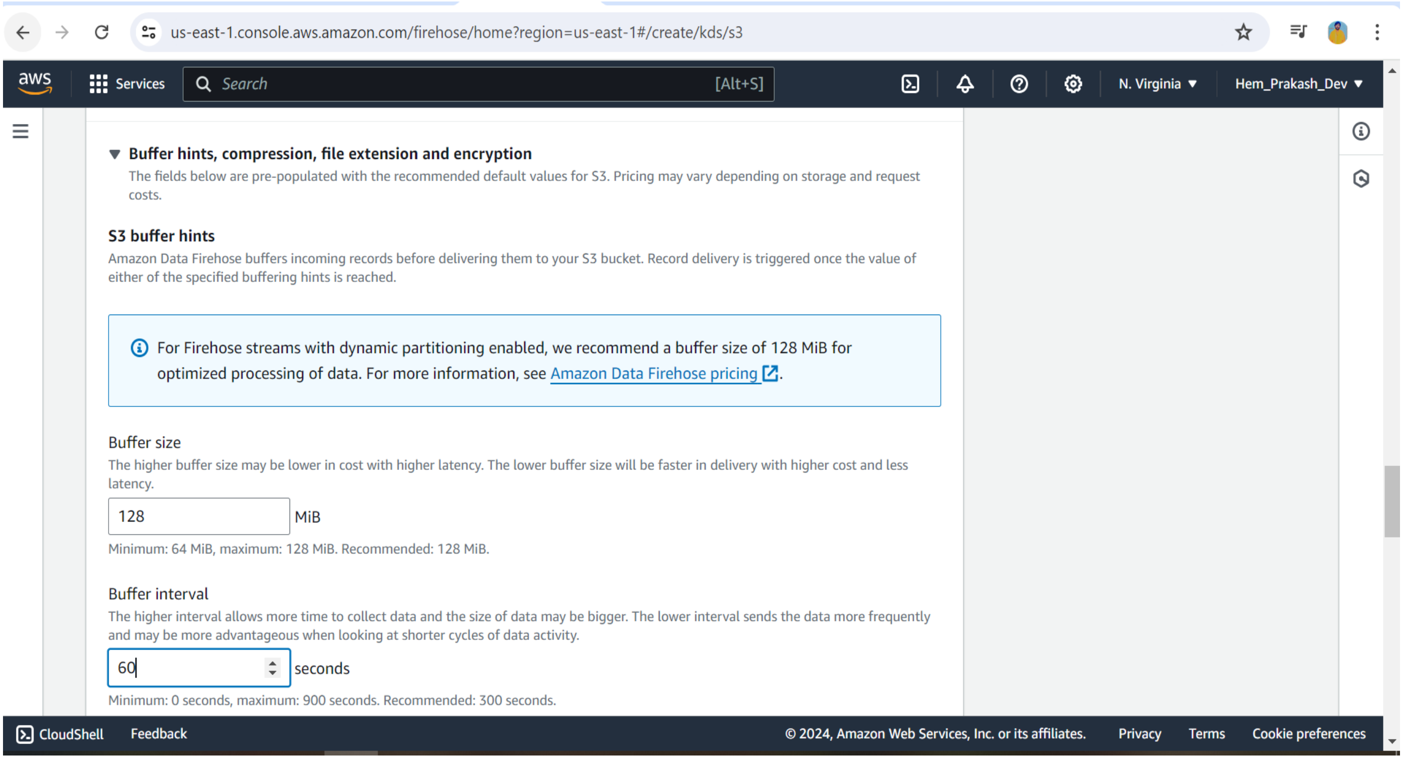
Dynamic partitioning keys must be unique. You can add 49 more dynamic partitioning key values.

S3 bucket prefix

For dynamic partitioning, you must use the following expression format in your S3 bucket prefix: !{namespace:value}, where namespace can be either partitionKeyFromQuery or partitionKeyFromLambda, or both. If you are using inline parsing to create the partitioning keys for your source data, you must specify an S3 bucket prefix value that consists of expressions specified in the following format: "!{partitionKeyFromQuery:keyID}". If you are using an AWS Lambda function to create partitioning keys for your source data, you must specify an S3 bucket prefix value that consists of expressions specified in the following format: "!{partitionKeyFromLambda:keyID}".

```
!{partitionKeyFromQuery:senser-id}/
```

Apply dynamic partitioning keys


us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/create/kds/s3
aws
Services
Search
[Alt+S]
N. Virginia
Hem_Prakash_Dev
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

▼ **Buffer hints, compression, file extension and encryption**

The fields below are pre-populated with the recommended default values for S3. Pricing may vary depending on storage and request costs.

S3 buffer hints

Amazon Data Firehose buffers incoming records before delivering them to your S3 bucket. Record delivery is triggered once the value of either of the specified buffering hints is reached.

ⓘ For Firehose streams with dynamic partitioning enabled, we recommend a buffer size of 128 MiB for optimized processing of data. For more information, see Amazon Data Firehose pricing.

Buffer size

The higher buffer size may be lower in cost with higher latency. The lower buffer size will be faster in delivery with higher cost and less latency.

128 MiB

Minimum: 64 MiB, maximum: 128 MiB. Recommended: 128 MiB.

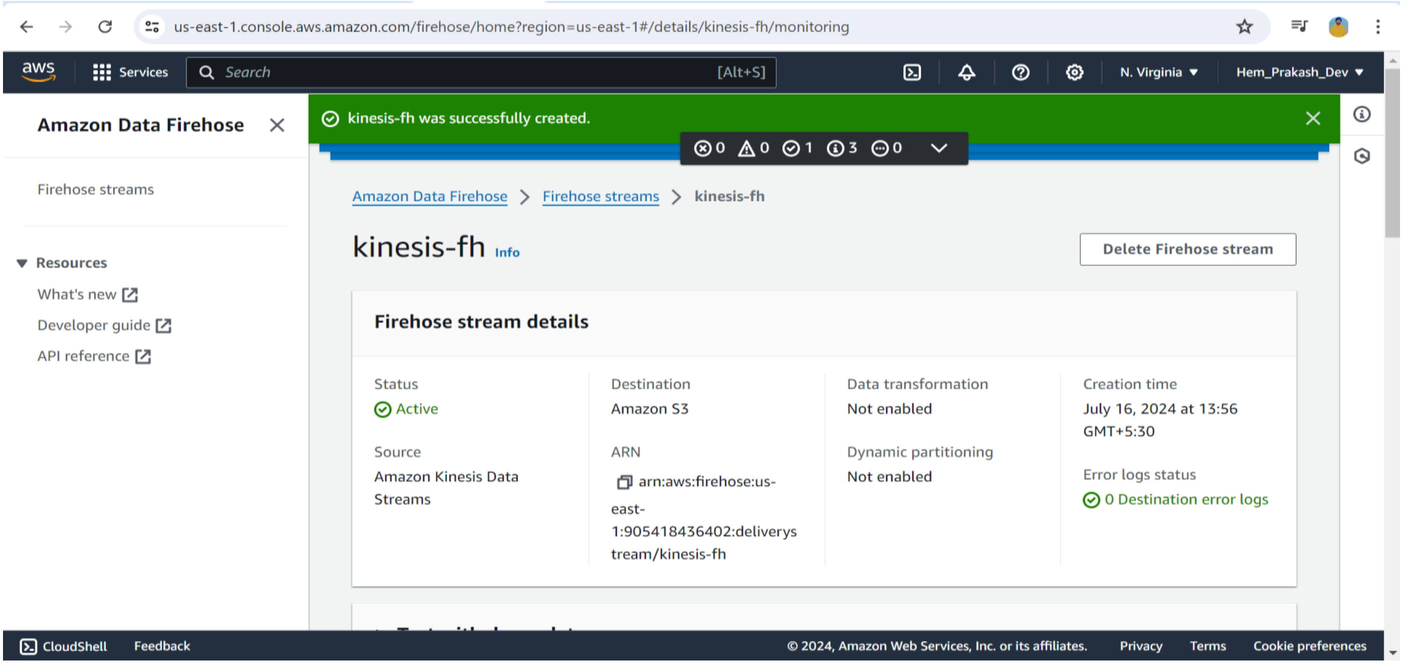
Buffer interval

The higher interval allows more time to collect data and the size of data may be bigger. The lower interval sends the data more frequently and may be more advantageous when looking at shorter cycles of data activity.

60 seconds

Minimum: 0 seconds, maximum: 900 seconds. Recommended: 300 seconds.

Kinesis firehose successfully created



us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/details/kinesis-fh/monitoring

kinesis-fh was successfully created.

Amazon Data Firehose > Firehose streams > kinesis-fh

kinesis-fh Info

Delete Firehose stream

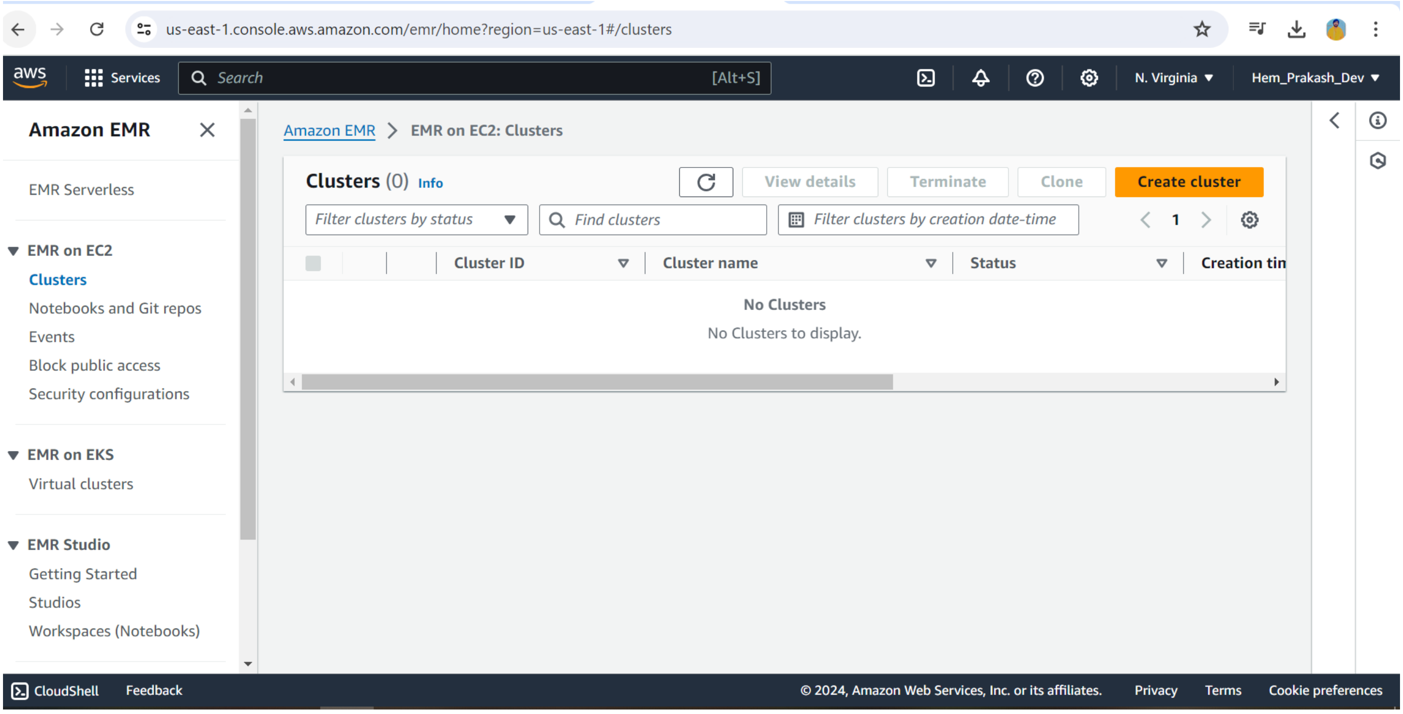
Firehose stream details

Status	Destination	Data transformation	Creation time
Active	Amazon S3	Not enabled	July 16, 2024 at 13:56 GMT+5:30
Source	ARN	Dynamic partitioning	Error logs status
Amazon Kinesis Data Streams	arn:aws:firehose:us-east-1:905418436402:deliverystream/kinesis-fh	Not enabled	0 Destination error logs

2. **Develop and test the data processing pipeline using AWS Glue or Amazon EMR (Elastic MapReduce) with Apache Spark or Hadoop**.

Data processing pipeline using Amazon EMR (Elastic MapReduce) with Apache Spark

EMR-Console



us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/clusters

Amazon EMR > EMR on EC2: Clusters

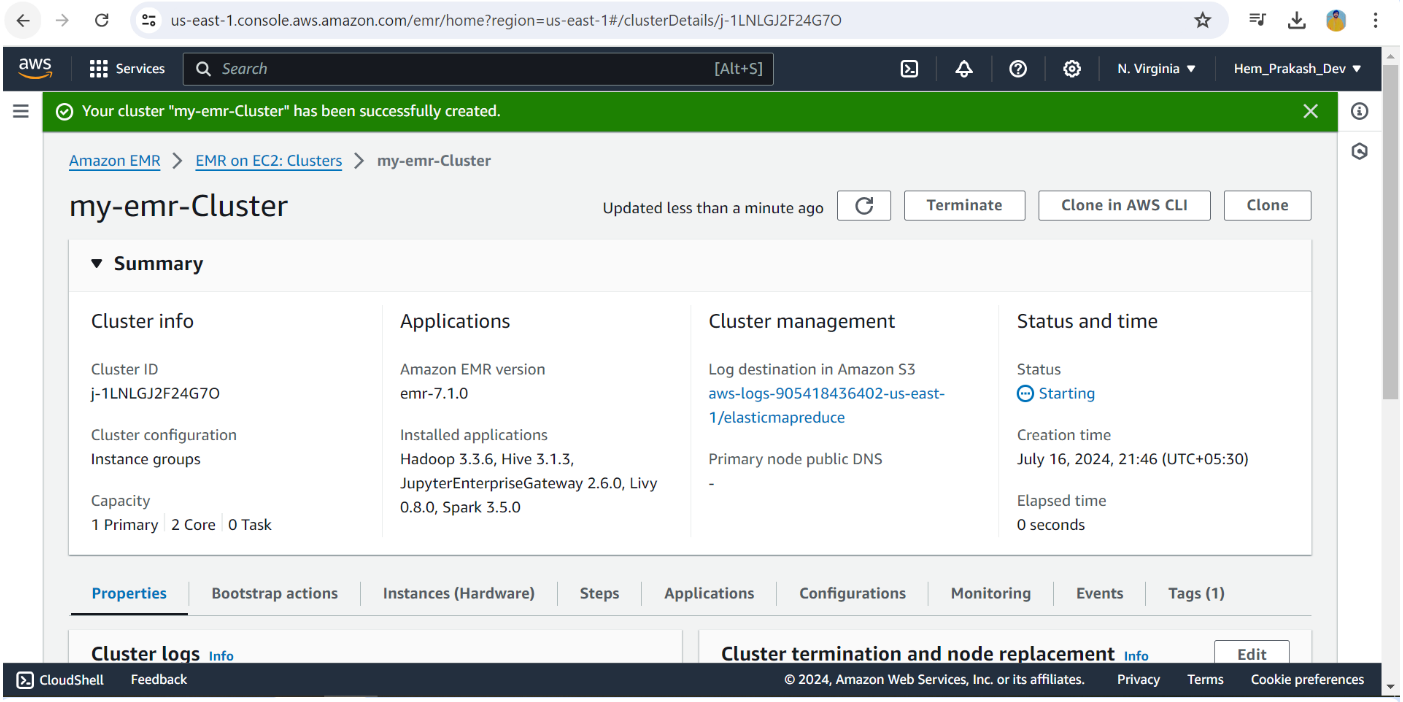
Clusters (0) Info

Cluster ID	Cluster name	Status	Creation tin

No Clusters

No Clusters to display.

Cluster creation



us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/clusterDetails/j-1LNLGJ2F24G7O

Your cluster "my-emr-Cluster" has been successfully created.

Amazon EMR > EMR on EC2: Clusters > my-emr-Cluster

my-emr-Cluster

Updated less than a minute ago

▼ Summary

Cluster info

Cluster ID
j-1LNLGJ2F24G7O

Cluster configuration
Instance groups

Capacity
1 Primary | 2 Core | 0 Task

Applications

Amazon EMR version
emr-7.1.0

Installed applications
Hadoop 3.3.6, Hive 3.1.3, JupyterEnterpriseGateway 2.6.0, Livy 0.8.0, Spark 3.5.0

Cluster management

Log destination in Amazon S3
aws-logs-905418436402-us-east-1/elasticmapreduce

Primary node public DNS
-

Status and time

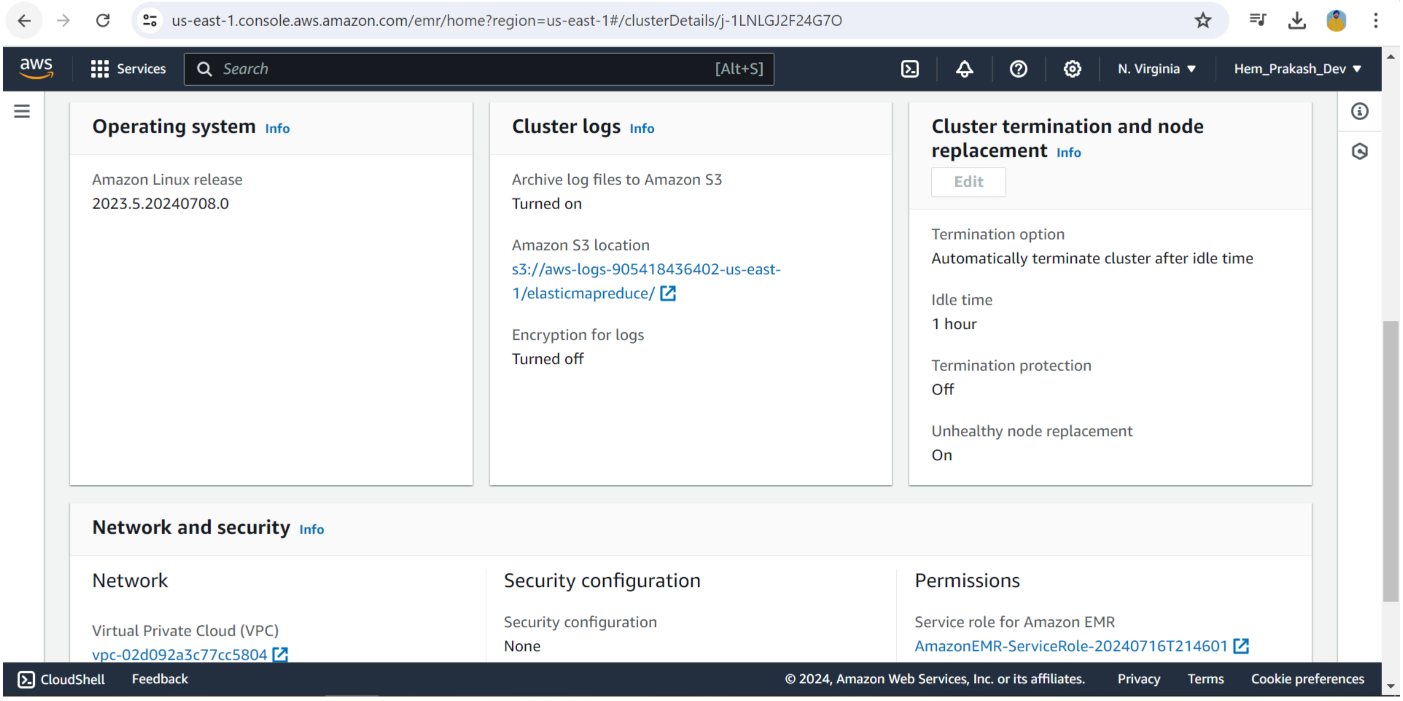
Status
Starting

Creation time
July 16, 2024, 21:46 (UTC+05:30)

Elapsed time
0 seconds

Cluster logs Info

Cluster termination and node replacement Info



us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/clusterDetails/j-1LNLGJ2F24G7O

Operating system Info

Amazon Linux release
2023.5.20240708.0

Cluster logs Info

Archive log files to Amazon S3
Turned on

Amazon S3 location
s3://aws-logs-905418436402-us-east-1/elasticmapreduce/

Encryption for logs
Turned off

Cluster termination and node replacement Info

Termination option
Automatically terminate cluster after idle time

Idle time
1 hour

Termination protection
Off

Unhealthy node replacement
On

Network and security Info

Network

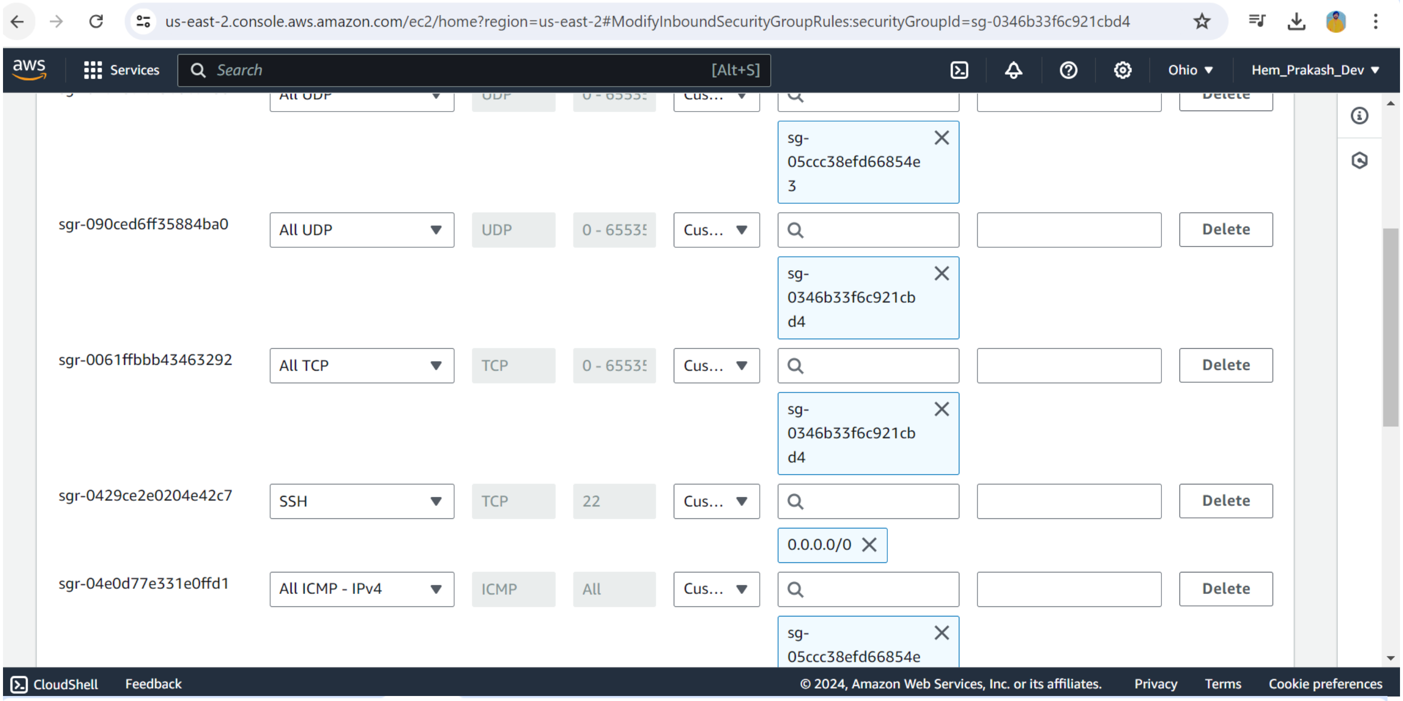
Virtual Private Cloud (VPC)
vpc-02d092a3c77cc5804

Security configuration

Security configuration
None

Permissions

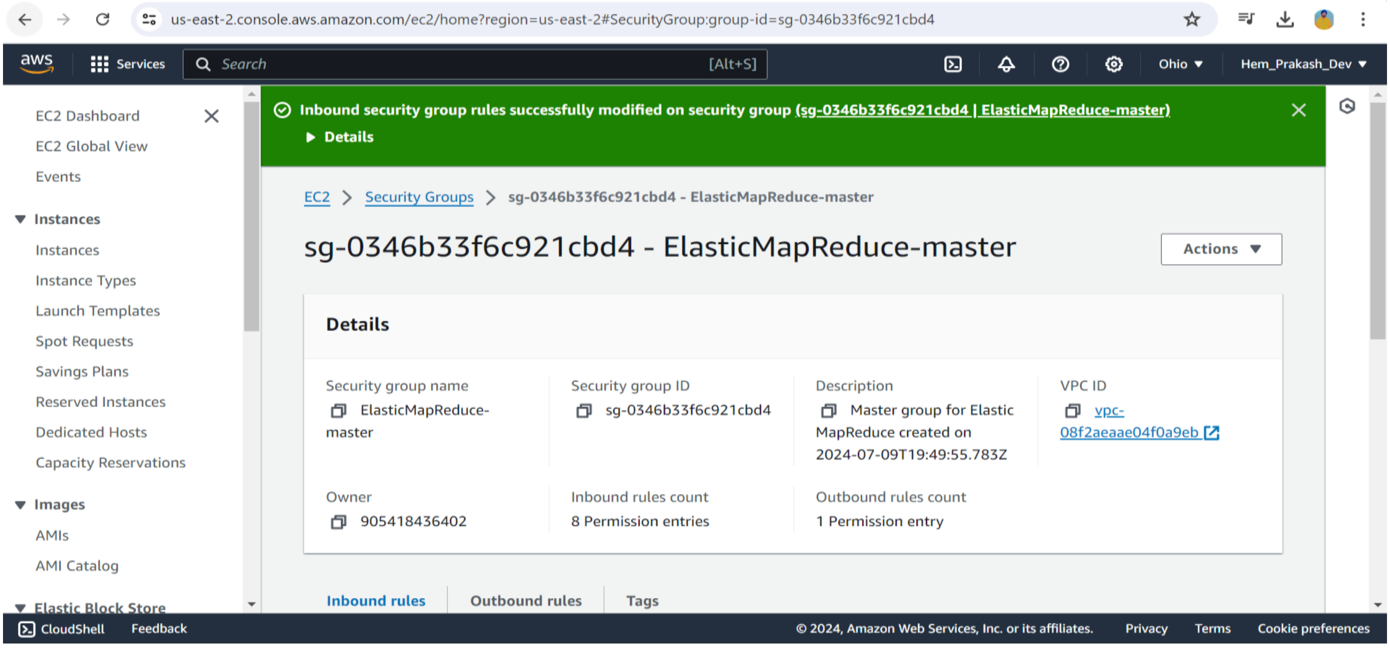
Service role for Amazon EMR
AmazonEMR-ServiceRole-20240716T214601



us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#ModifyInboundSecurityGroupRules:securityGroupId=sg-0346b33f6c921cbd4

	All UDP ▼	UDP	0 - 65535	Cus... ▼	sg-05ccc38efd66854e3 ✕	Delete
sgr-090ced6ff35884ba0	All UDP ▼	UDP	0 - 65535	Cus... ▼	sg-0346b33f6c921cbd4 ✕	Delete
sgr-0061ffbbb43463292	All TCP ▼	TCP	0 - 65535	Cus... ▼	sg-0346b33f6c921cbd4 ✕	Delete
sgr-0429ce2e0204e42c7	SSH ▼	TCP	22	Cus... ▼	0.0.0.0/0 ✕	Delete
sgr-04e0d77e331e0ffd1	All ICMP - IPv4 ▼	ICMP	All	Cus... ▼	sg-05ccc38efd66854e	Delete

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#SecurityGroup:group-id=sg-0346b33f6c921cbd4

Inbound security group rules successfully modified on security group (sg-0346b33f6c921cbd4 | ElasticMapReduce-master)

▶ Details

EC2 > Security Groups > sg-0346b33f6c921cbd4 - ElasticMapReduce-master

sg-0346b33f6c921cbd4 - ElasticMapReduce-master

Details

Security group name: ElasticMapReduce-master

Security group ID: sg-0346b33f6c921cbd4

Description: Master group for Elastic MapReduce created on 2024-07-09T19:49:55.783Z

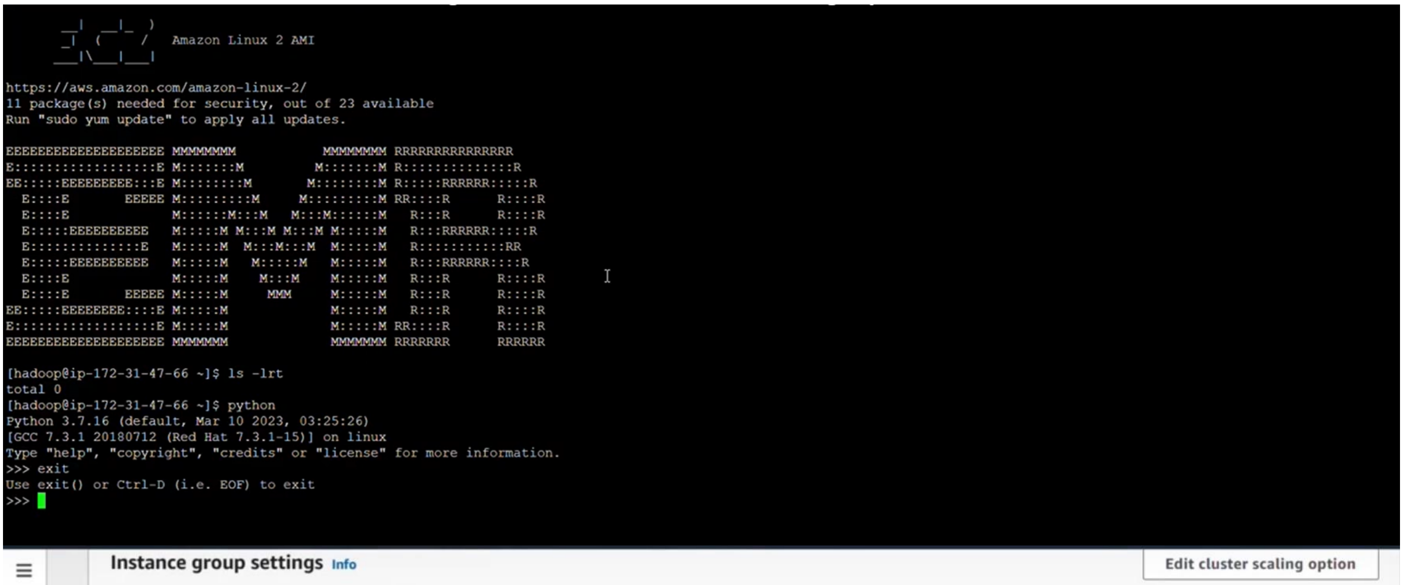
VPC ID: vpc-08f2aeaae04f0a9eb

Owner: 905418436402

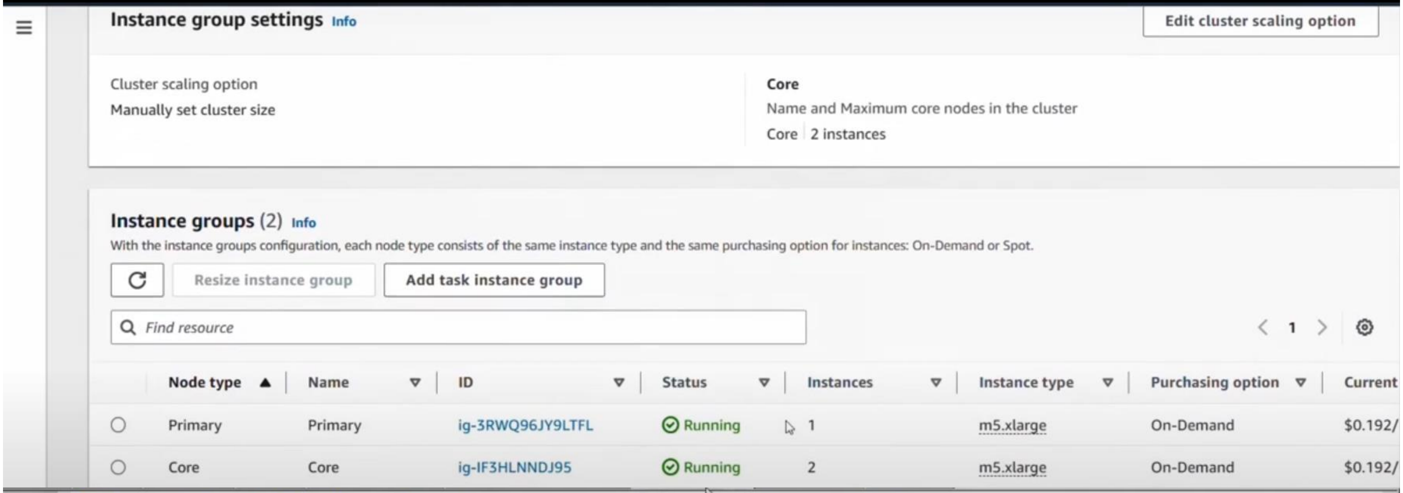
Inbound rules count: 8 Permission entries

Outbound rules count: 1 Permission entry

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
11 package(s) needed for security, out of 23 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-47-66 ~]$ ls -lrt
total 0
[hadoop@ip-172-31-47-66 ~]$ python
Python 3.7.16 (default, Mar 10 2023, 03:25:26)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-15)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
>>> 
```



Instance group settings Info

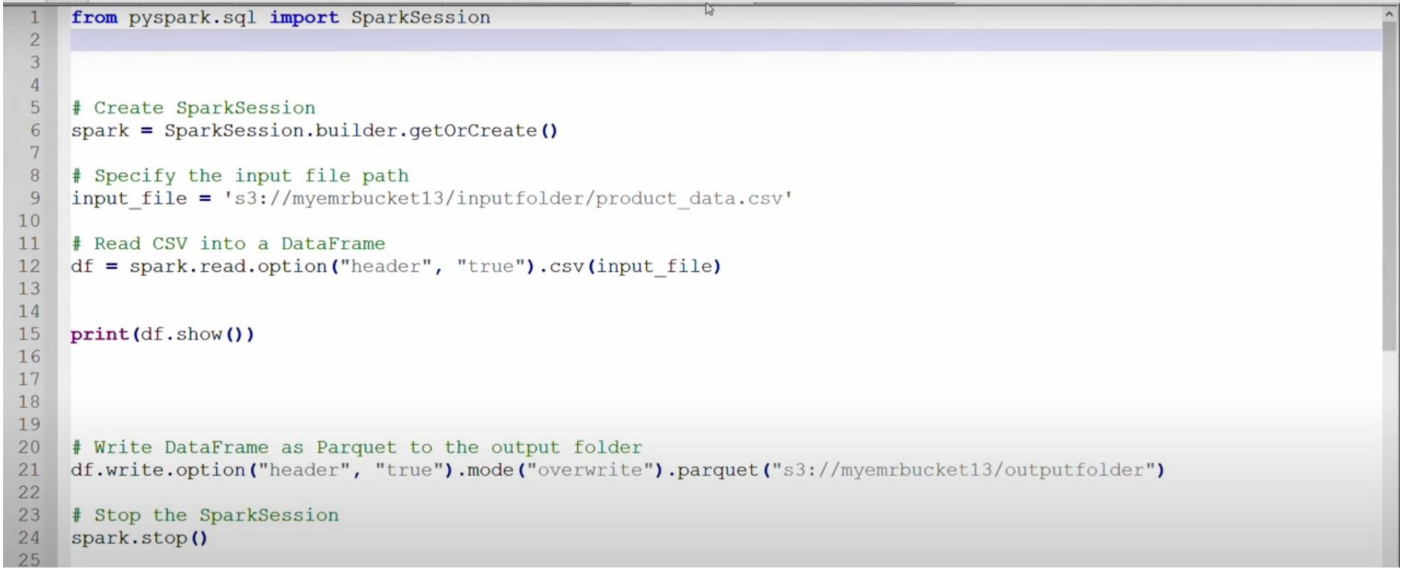
Cluster scaling option
Manually set cluster size

Core
Name and Maximum core nodes in the cluster
Core 2 instances

Instance groups (2) Info

With the instance groups configuration, each node type consists of the same instance type and the same purchasing option for instances: On-Demand or Spot.

	Node type	Name	ID	Status	Instances	Instance type	Purchasing option	Current
○	Primary	Primary	ig-3RWQ96JY9LTFL	Running	1	m5.xlarge	On-Demand	$0.192/
○	Core	Core	ig-IF3HLNNDJ95	Running	2	m5.xlarge	On-Demand	$0.192/

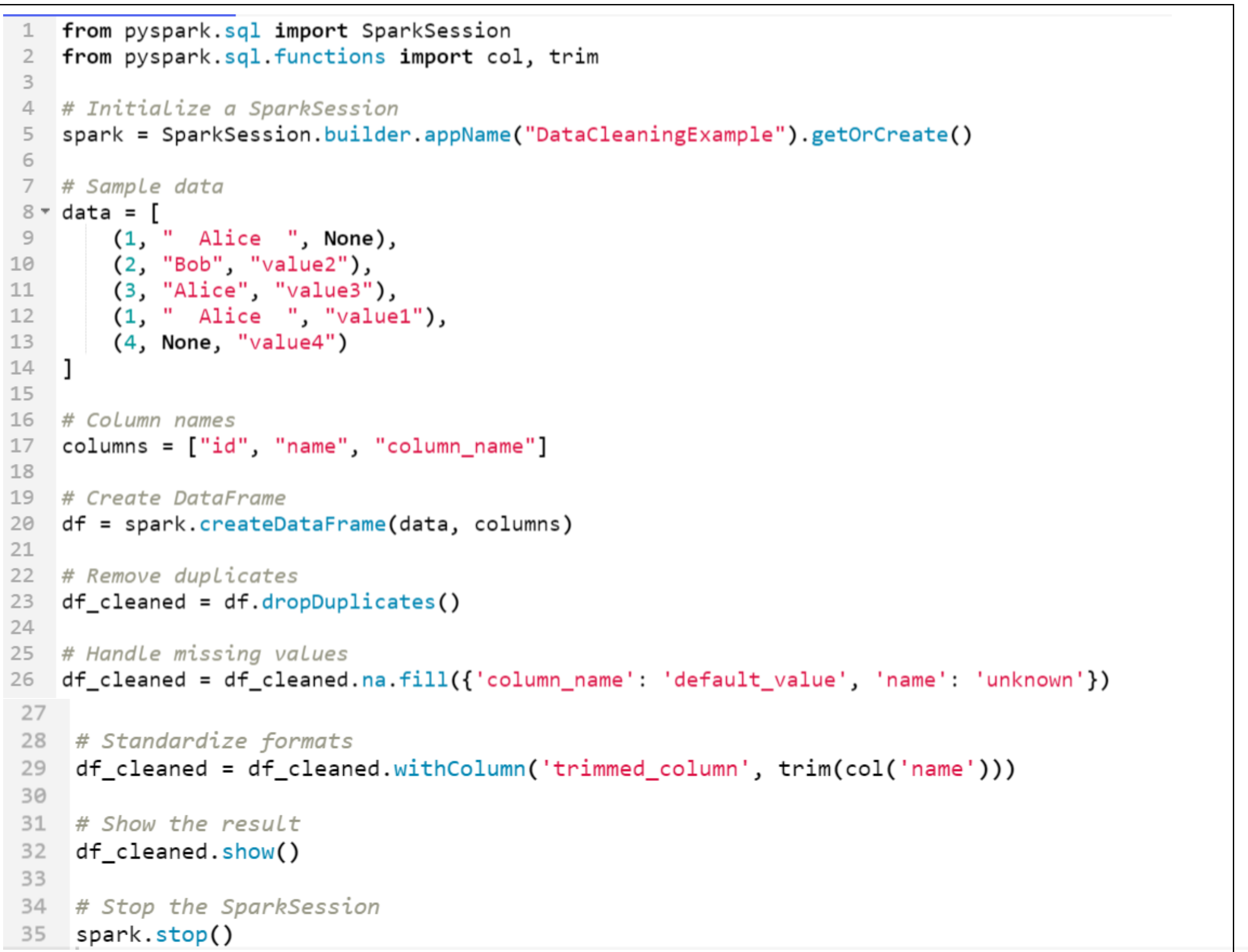


```python
from pyspark.sql import SparkSession



# Create SparkSession
spark = SparkSession.builder.getOrCreate()

# Specify the input file path
input_file = 's3://myemrbucket13/inputfolder/product_data.csv'

# Read CSV into a DataFrame
df = spark.read.option("header", "true").csv(input_file)


print(df.show())




# Write DataFrame as Parquet to the output folder
df.write.option("header", "true").mode("overwrite").parquet("s3://myemrbucket13/outputfolder")

# Stop the SparkSession
spark.stop()

```

3. Apply data transformation and cleansing techniques to prepare the data for aggregation and analysis.

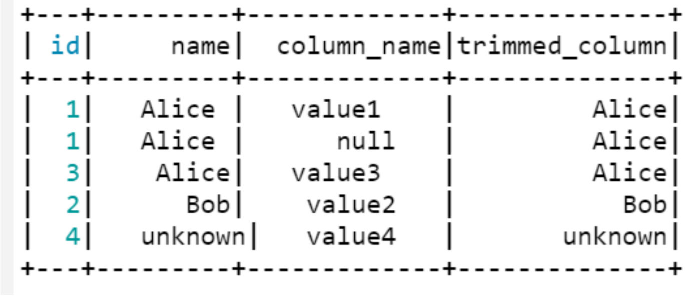
Applying data transformation and cleansing techniques to prepare the data for aggregation and analysis.

- **Transformation**: Filter, select, join, and aggregate data as needed.
- **Cleansing**: Handle missing values, remove duplicates, and standardize formats.



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, trim

# Initialize a SparkSession
spark = SparkSession.builder.appName("DataCleaningExample").getOrCreate()

# Sample data
data = [
    (1, "  Alice  ", None),
    (2, "Bob", "value2"),
    (3, "Alice", "value3"),
    (1, "  Alice  ", "value1"),
    (4, None, "value4")
]

# Column names
columns = ["id", "name", "column_name"]

# Create DataFrame
df = spark.createDataFrame(data, columns)

# Remove duplicates
df_cleaned = df.dropDuplicates()

# Handle missing values
df_cleaned = df_cleaned.na.fill({'column_name': 'default_value', 'name': 'unknown'})

# Standardize formats
df_cleaned = df_cleaned.withColumn('trimmed_column', trim(col('name')))

# Show the result
df_cleaned.show()

# Stop the SparkSession
spark.stop()
```

Output of the sample taken

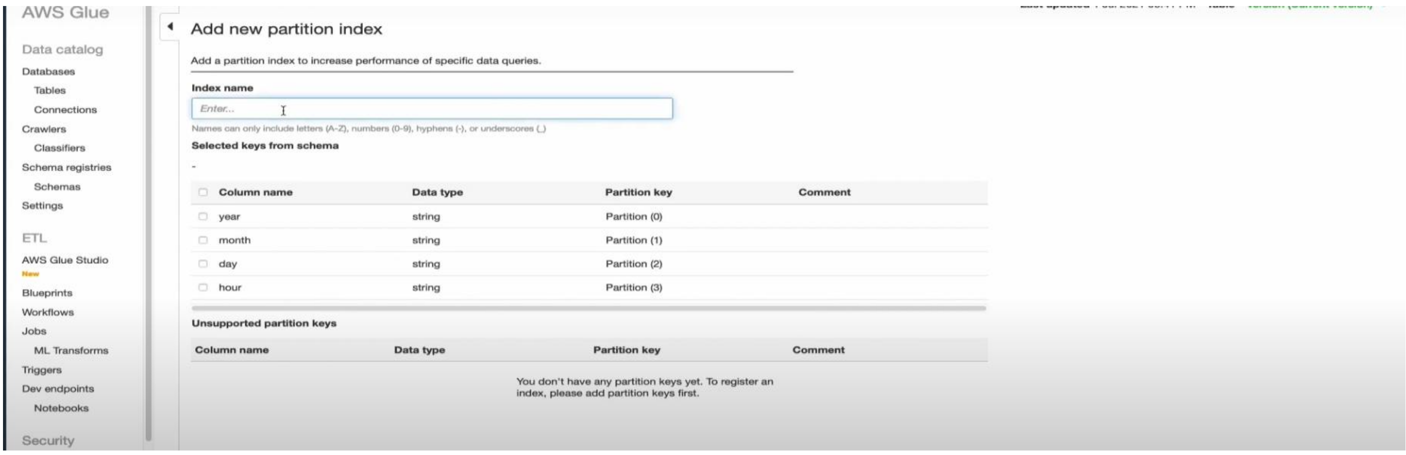


```
+---+---------+------------+--------------+
| id|     name|  column_name|trimmed_column|
+---+---------+------------+--------------+
|  1|   Alice |   value1    |         Alice|
|  1|   Alice |      null   |         Alice|
|  3|    Alice|   value3    |         Alice|
|  2|      Bob|    value2   |           Bob|
|  4|  unknown|    value4   |       unknown|
+---+---------+------------+--------------+
```

4. Implement data partitioning and indexing strategies to optimize query performance.

Implementing data partitioning and indexing strategies to optimize query performance.

Partitioning: Partition data based on commonly queried fields.



Add new partition index

Add a partition index to increase performance of specific data queries.

Index name

Enter...

Names can only include letters (A-Z), numbers (0-9), hyphens (-), or underscores (_)

Selected keys from schema

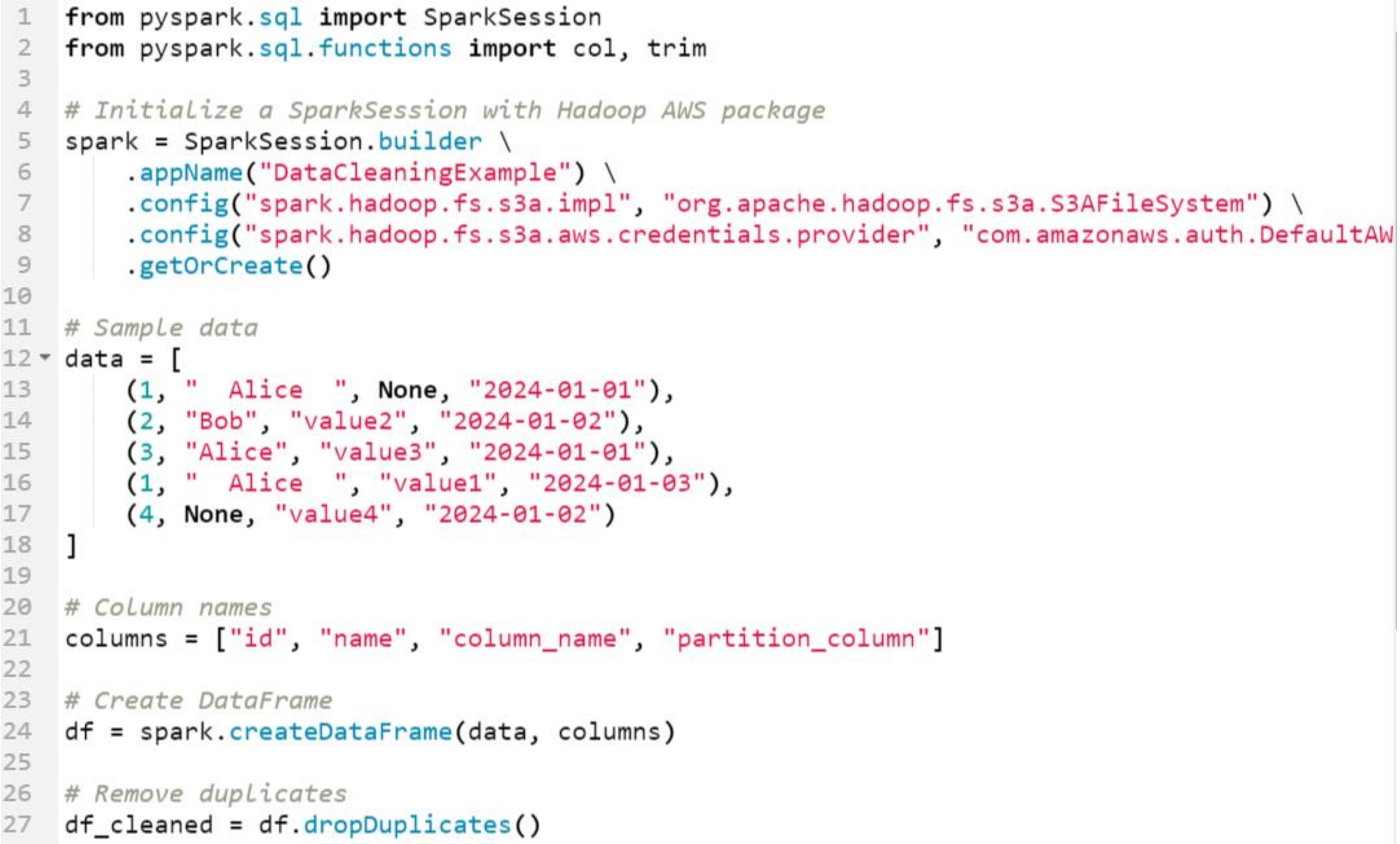
-

□ Column name	Data type	Partition key	Comment
□ year	string	Partition (0)	
□ month	string	Partition (1)	
□ day	string	Partition (2)	
□ hour	string	Partition (3)	

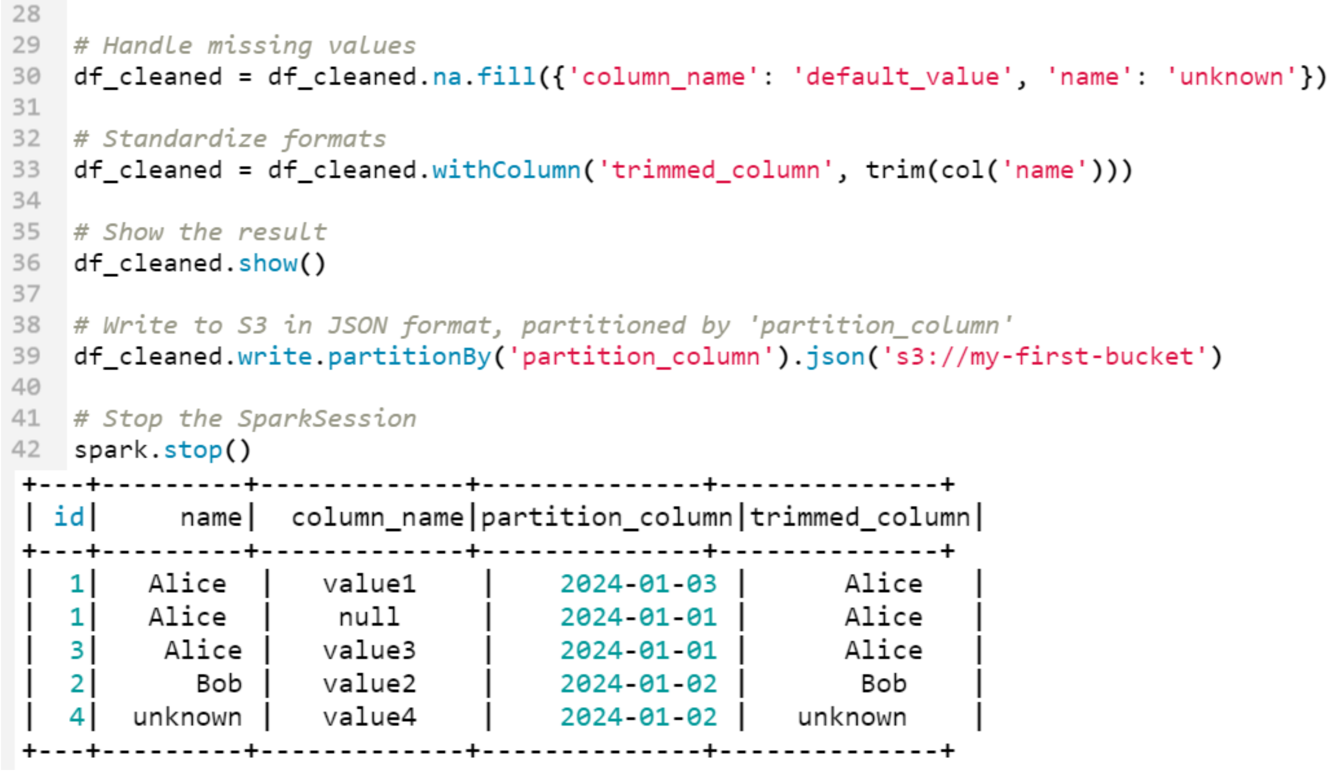
Unsupported partition keys

Column name	Data type	Partition key	Comment

You don't have any partition keys yet. To register an index, please add partition keys first.



```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, trim

# Initialize a SparkSession with Hadoop AWS package
spark = SparkSession.builder \
    .appName("DataCleaningExample") \
    .config("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem") \
    .config("spark.hadoop.fs.s3a.aws.credentials.provider", "com.amazonaws.auth.DefaultAW
    .getOrCreate()

# Sample data
data = [
    (1, "  Alice  ", None, "2024-01-01"),
    (2, "Bob", "value2", "2024-01-02"),
    (3, "Alice", "value3", "2024-01-01"),
    (1, "  Alice  ", "value1", "2024-01-03"),
    (4, None, "value4", "2024-01-02")
]

# Column names
columns = ["id", "name", "column_name", "partition_column"]

# Create DataFrame
df = spark.createDataFrame(data, columns)

# Remove duplicates
df_cleaned = df.dropDuplicates()
```



```

# Handle missing values
df_cleaned = df_cleaned.na.fill({'column_name': 'default_value', 'name': 'unknown'})

# Standardize formats
df_cleaned = df_cleaned.withColumn('trimmed_column', trim(col('name')))

# Show the result
df_cleaned.show()

# Write to S3 in JSON format, partitioned by 'partition_column'
df_cleaned.write.partitionBy('partition_column').json('s3://my-first-bucket')

# Stop the SparkSession
spark.stop()
```

```
+---+---------+-------------+--------------+--------------+
| id|     name|  column_name|partition_column|trimmed_column|
+---+---------+-------------+--------------+--------------+
|  1|   Alice  |   value1     |   2024-01-03 |     Alice    |
|  1|   Alice  |    null      |   2024-01-01 |     Alice    |
|  3|    Alice |   value3     |   2024-01-01 |     Alice    |
|  2|      Bob |   value2     |   2024-01-02 |      Bob     |
|  4|  unknown |   value4     |   2024-01-02 |   unknown    |
+---+---------+-------------+--------------+--------------+
```

5. Update the GitHub repository with the code and configuration files for data ingestion and processing.

Updating the GitHub repository with the code and configuration files for data ingestion and processing.

1. **Initializing a Git Repository**:

    git init

2. **Adding and Commit Code**:

    git add .

    git commit -m "Initial commit with data ingestion and processing scripts"

3. **Pushing to GitHub:**

```
git remote add origin https://github.com/Hem-Prakash-Dev-Bharadwaj/data-engineering
git push -u origin main
```