

Connected Mobility Data Lake Solution Deployment Guide

Document History

No.	Author	Email	Description	Version	Date
1	Jun XU	xujunaws@amazon.com	Baseline. Validated in AWS BJS region.	V1.0	2021-05- 14
2	Jun XU	xujunaws@amazon.com	Added business intelligence section.	V1.1	2021-06- 24
3	Jun XU	xujunaws@amazon.com	Added FOTA.	V1.2	2022-09- 30

Table of Contents

1. Overview	3
2. Document Structure	3
3. System Architecture	3
4. Deployment Steps	5
4.1. Batch Mode	5
4.1.1. AWS Lake Formation.....	5
4.1.2. AWS Glue	6
4.1.3. Amazon Athena.....	7
4.1.4. Data Catalog and Query.....	7
4.1.5. Amazon EMR.....	18
4.1.6. Run Spark Job in Amazon EMR Master Node.....	21
4.1.7. Run Spark Job in Amazon EMR Jupyter Notebook	26
4.2. Streaming Mode.....	27
4.2.1. Amazon Kinesis Data Stream.....	27
4.2.2. Amazon Kinesis Data Analytics	28
4.2.3. Data Streaming and Analytics.....	29
4.2.4. Real Time Analytics.....	32
5. Business Intelligence.....	34
6. Firmware-Over-The-Air.....	34
7. Reference	35

1. Overview

Connected Mobility Solutions - Data Lake, hereafter abbreviated as CMS – Data Lake, on Amazon Web Services help our customers rapidly deploy a robust, scalable and secure CMS data lake platform that meets a wide variety of current and future use cases, without the heavy investment and undifferentiated heavy lifting of deploying and maintaining data centers, custom stacks and proprietary vertically-integrated solutions. Use cases will be implemented across data consumption, driver behavior, anomaly detection, diagnostics, location-based services, OTA, fleet configuration management and provisioning and lifecycle management.

2. Document Structure

- connected_mobility_data_lake_deployment_guide.docx
- connected_mobility_data_lake_first_call_deck.pptx
- your_s3_bucket //Your S3 bucket name.
 - files
 - cfn-templates //Amazon CloudFormation template files.
 - emr
 - jupyter
 - cmdatalake.ipynb //The Jupyter Notebook used for ETL.
 - kinesis
 - python
 - streaming.py //The Python script used for streaming.
 - spark
 - spark-etl.py
 - input
 - ny-taxi
 - trip-data
 - yellow_tripdata_2015-12.csv //New York Taxi dataset.
 - output
 - ny-taxi
 - trip-data
 - spark //Output for EMR Spark jobs.
 - jupyter //Output for EMR Jupyter Notebook projects.
 - logs //Solution logs

3. System Architecture

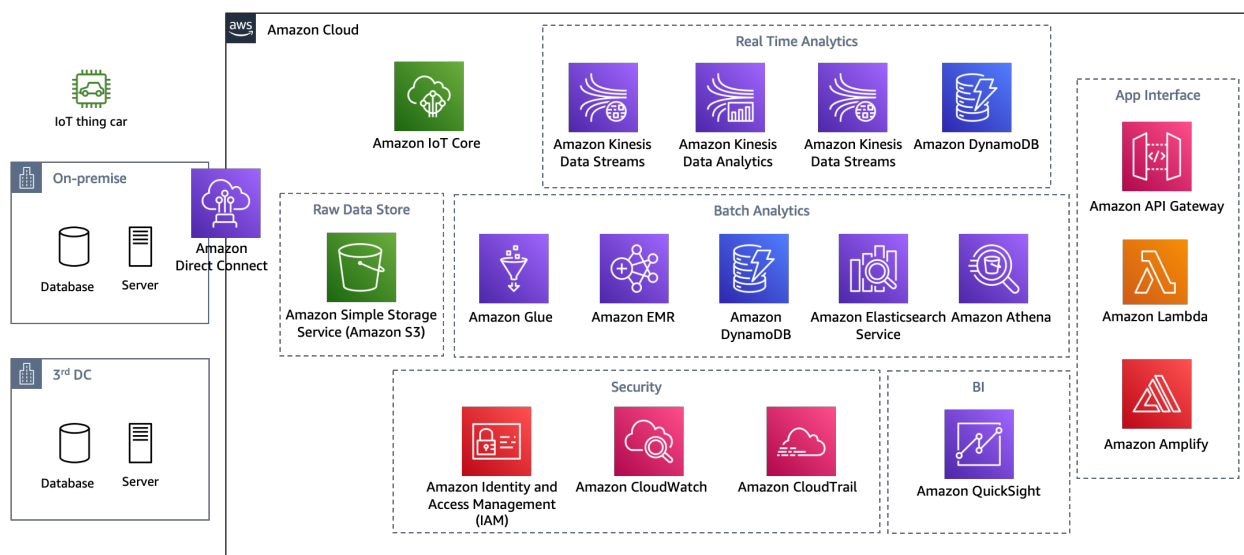
CMS – Data Lake is made up of four parts, which are respectively car fleet with IoT thing car components, on-premise data centers, 3rd party data centers and AWS cloud. Car fleet with IoT thing car components communicate with AWS cloud via AWS IoT Coreⁱ based on MQTT protocol. On-premise and 3rd party data centers communicates and transferred data to and from AWS cloud via AWS Direct Connectⁱⁱ.

On AWS cloud part, it can be divided into 5 groups, which are respectively batch processing and analytics, real time analytics, data storage, security and authorizations, application interface.

For CMS – Data Lake, the typical using scenarios are batch processing and analytics and real time analytics. The batch processing and analytics group is composed of AWS Glueⁱⁱⁱ, Amazon EMR^{iv}, Amazon DynamoDB^v, Amazon Elasticsearch Service^{vi} and Amazon Athena^{vii}. AWS Glue works as the data crawler crawling and cataloging data from Amazon Simple Storage Service (Amazon S3)^{viii}, and then Amazon EMR works for fine grained data processing, and then Amazon DynamoDB can be used as structured data persistent storage. Data query, search and visualization can be implanted via Amazon Elasticsearch Service and Amazon Athena. The real time analytics group is made up of Amazon Kinesis Data Streams^{ix}, Amazon Kinesis Data Analytics^x and Amazon DynamoDB. Amazon Kinesis Data Steams service ingests real time data from AWS IoT Core and outputs to Amazon Kinesis Data Analytics for database schema extraction, templated data analytics. The data can be relayed to another Amazon Kinesis Data Streams service and then archived to Amazon DynamoDB.

For simplification, this solution uses an Amazon EC2^{xi} instance with New York Taxi dataset^{xii} for edge part data synthetization instead of getting data from physical cars equipped with IoT thing car components.

For application interface, Amazon API Gateway^{xiii} with AWS Lambda Function^{xiv} can be used for exposing the services. AWS Amplify^{xv} can be used for web and mobile applications implementation.



4. Deployment Steps

This solution has been validated in AWS BJS region. It can also be applied to AWS ZHY region.

4.1. Batch Mode

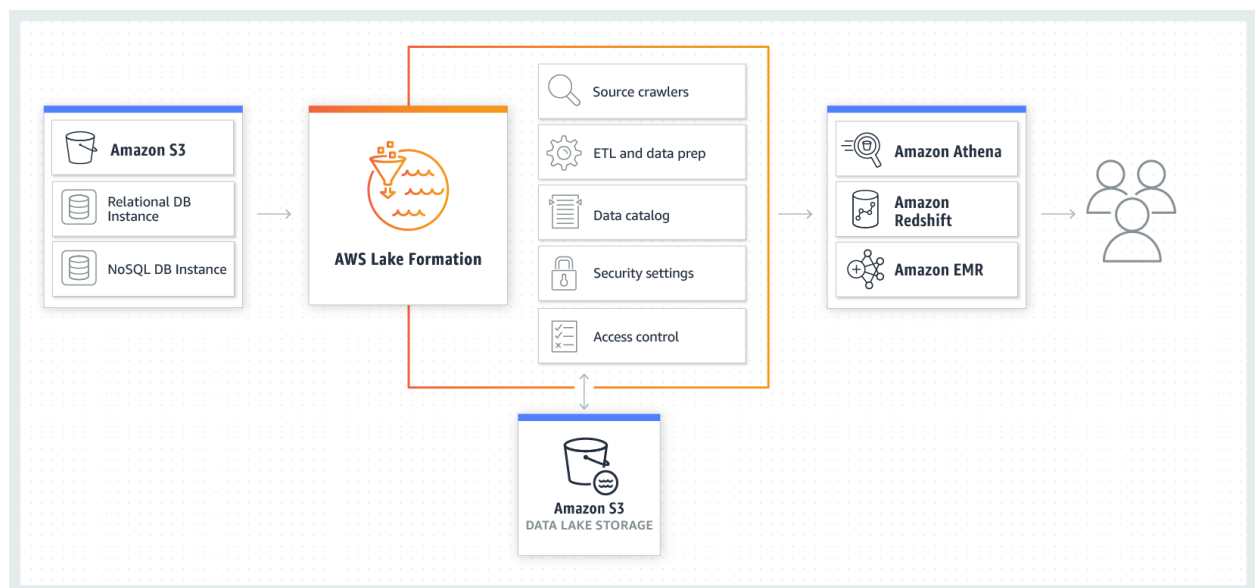
Before diving deep into the batch processing mode, it is necessary to introduce AWS Lake Formation^{xvi}, AWS Glue and Amazon Athena.

4.1.1. AWS Lake Formation

AWS Lake Formation makes it easier for you to build, secure, and manage data lakes. Lake Formation helps you do the following, either directly or through other AWS services:

- Register the Amazon Simple Storage Service (Amazon S3) buckets and paths where your data lake will reside.
- Orchestrate data flows that ingest, cleanse, transform, and organize the raw data.
- Create and manage a Data Catalog containing metadata about data sources and data in the data lake.
- Define granular data access policies to the metadata and data through a grant/revoke permissions model.

The following diagram illustrates how data is loaded and secured in Lake Formation.



As the diagram shows, Lake Formation manages AWS Glue crawlers, AWS Glue ETL jobs, the Data Catalog, security settings, and access control. After the data is securely stored in the

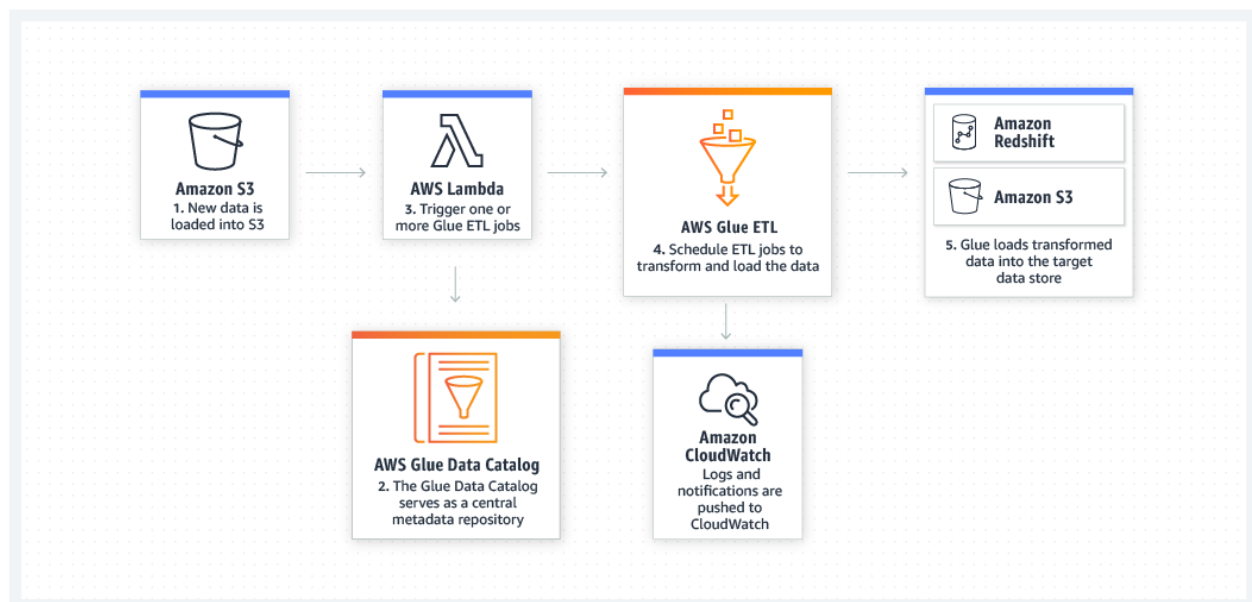
data lake, users can access the data through their choice of analytics services, including Amazon Athena, Amazon Redshift^{xvii}, and Amazon EMR.

4.1.2. AWS Glue

AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development. AWS Glue provides all of the capabilities needed for data integration so that you can start analyzing your data and putting it to use in minutes instead of months.

Data integration is the process of preparing and combining data for analytics, machine learning, and application development. It involves multiple tasks, such as discovering and extracting data from various sources; enriching, cleaning, normalizing, and combining data; and loading and organizing data in databases, data warehouses, and data lakes. These tasks are often handled by different types of users that each use different products.

AWS Glue provides both visual and code-based interfaces to make data integration easier. Users can easily find and access data using the AWS Glue Data Catalog. Data engineers and ETL (extract, transform, and load) developers can visually create, run, and monitor ETL workflows with a few clicks in AWS Glue Studio. Data analysts and data scientists can use AWS Glue DataBrew to visually enrich, clean, and normalize data without writing code. With AWS Glue Elastic Views, application developers can use familiar Structured Query Language (SQL) to combine and replicate data across different data stores.



4.1.3. Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run.

Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there's no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

Athena is out-of-the-box integrated with AWS Glue Data Catalog, allowing you to create a unified metadata repository across various services, crawl data sources to discover schemas and populate your Catalog with new and modified table and partition definitions, and maintain schema versioning.

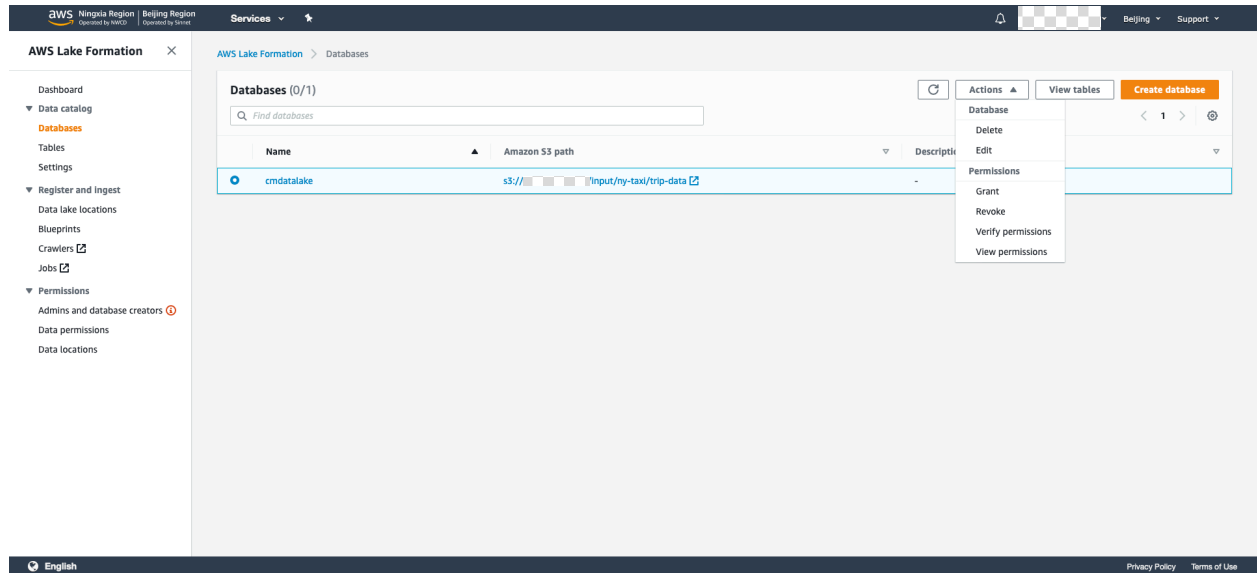
4.1.4. Data Catalog and Query

Log into the AWS Lake Formation console with the following link
<https://console.amazonaws.cn/lakeformation/home?region=cn-north-1#create-database>.
Create a database according to the diagrams listed as following.

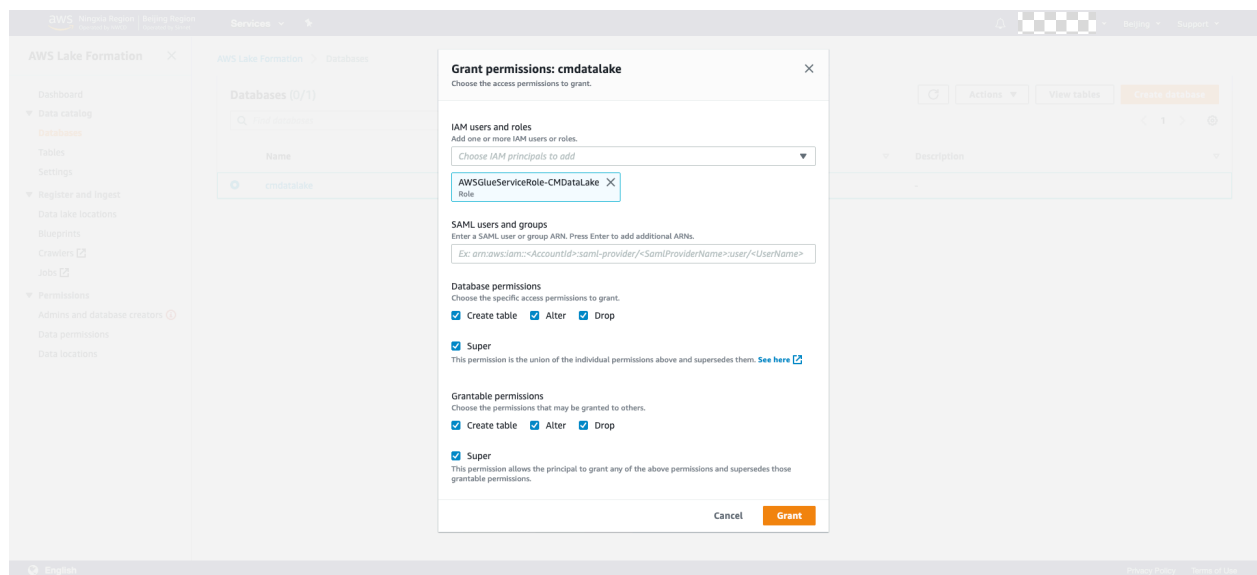
The screenshot shows the AWS Lake Formation console interface for creating a new database. The breadcrumb navigation at the top indicates the path: AWS Lake Formation > Databases > Create database. The main heading is 'Create database'. Below this, there is a section titled 'Database details' with the subtitle 'Create a database in the AWS Glue Data Catalog.' The form contains the following fields and options:

- Name:** A text input field containing 'cmdatalake'.
- Location - optional:** A section with the instruction 'Choose an Amazon S3 path for this database, which eliminates the need to grant data location permissions on catalog table paths that are this location's children.' It includes a text input field with 's3://input/my-taxi/trip-data' and a 'Browse' button.
- Description - optional:** A text area with the placeholder 'Enter a description' and a note 'Descriptions can be up to 2048 characters long.'
- Default permissions for newly created tables:** A section with the instruction 'This setting maintains existing AWS Glue Data Catalog behavior. You can still set individual permissions, which will take effect when you revoke the Super permission from IAMAllowedPrincipals. See [Changing Default Settings for Your Data Lake](#).' It includes a checkbox labeled 'Use only IAM access control for new tables in this database' which is currently unchecked.

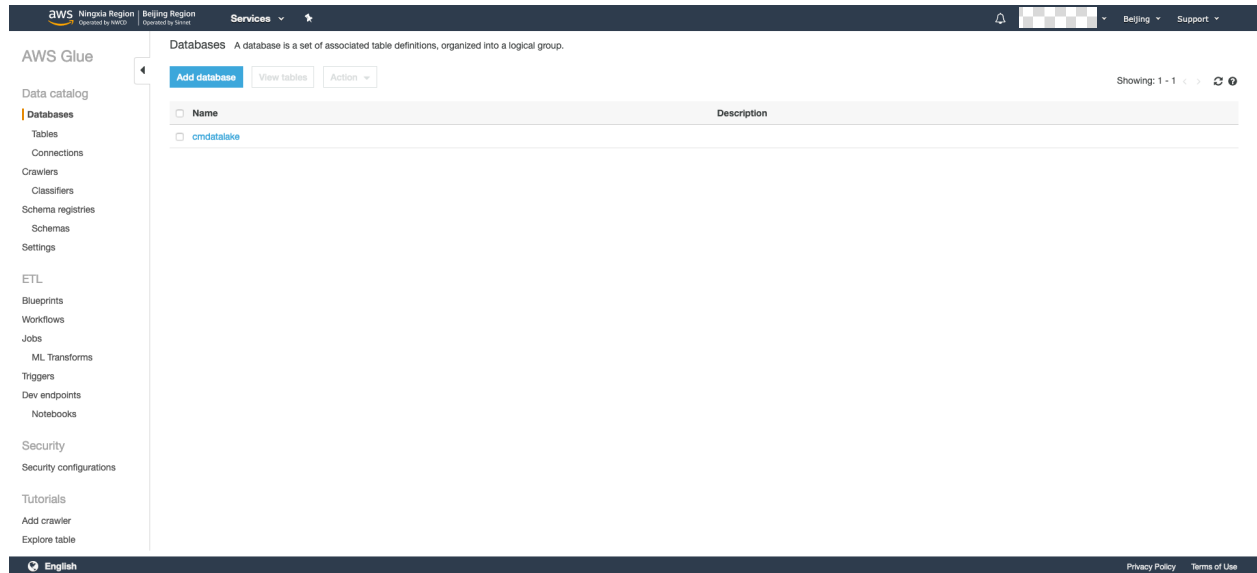
At the bottom of the form, there are two buttons: 'Cancel' and 'Create database'.



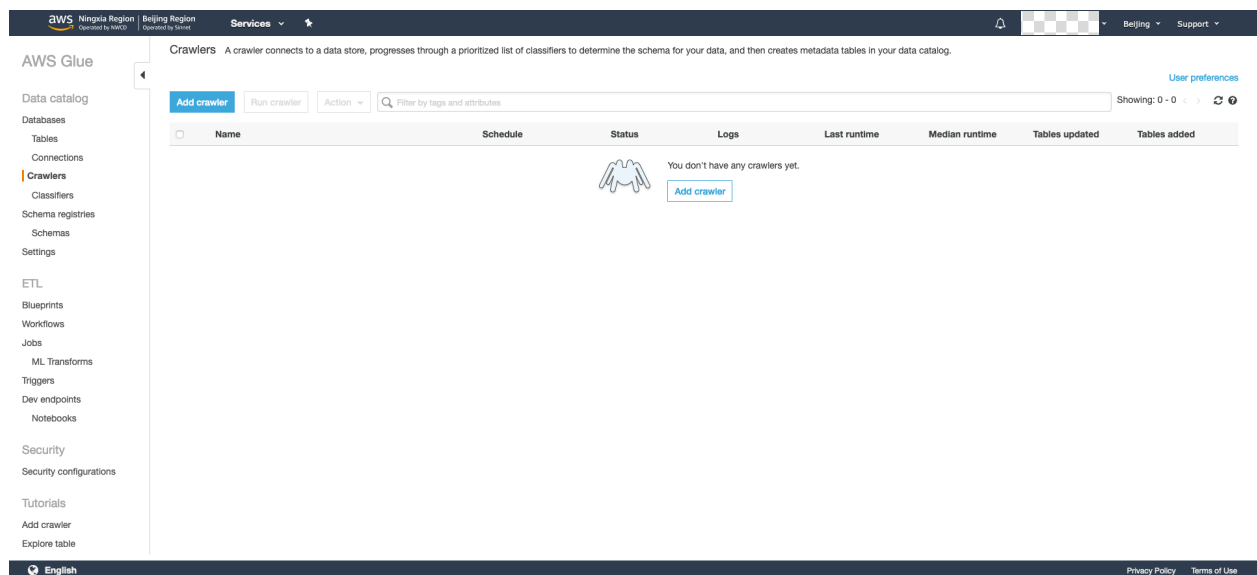
Grant access permissions to your AWS IAM role `AWSGlueServiceRole-CMDDataLake`, which is created by yourself and will be introduced later.



The database created in Lake Formation can also be viewed in the console of AWS Glue.



Then a crawler can be created in the console of AWS Glue. The following diagrams illustrate how to create a crawler and crawl the data from the CSV files in S3, build relevant data catalogs.



aws

Ningxia Region
Operated by NINGBO

Beijing Region
Operated by TSMC

Services

Beijing

Support

Add crawler

Crawler Info

Crawler source type

Data store

IAM Role

Schedule

Output

Review all steps

Add information about your crawler

Crawler name

cmdatalake

Tags, description, security configuration, and classifiers (optional)

Next

English

Privacy Policy

Terms of Use

aws

Ningxia Region
Operated by NINGBO

Beijing Region
Operated by TSMC

Services

Beijing

Support

Add crawler

Crawler Info

cmdatalake

Crawler source type

Data store

IAM Role

Schedule

Output

Review all steps

Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

Crawler source type

☒ Data stores

☐ Existing catalog tables

Repeat crawls of S3 data stores

☒ Crawl all folders

☐ Crawl new folders only

Crawl all folders again with every subsequent crawl.

Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.

Back

Next

English

Privacy Policy

Terms of Use

Add crawler

- ✓ Crawler info
- ✓ cmdatalake
- ✓ Crawler source type
- Data stores
- Data store**
- IAM Role
- Schedule
- Output
- Review all steps

Add a data store

Choose a data store

S3

Connection

Select a connection

Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any future S3 targets will also use the same connection (or none, if left blank).

[Add connection](#)

Crawl data in

☒ Specified path in my account

☐ Specified path in another account

Include path

s3://[bucket]/input/my-taxi/trip-data

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Exclude patterns (optional)

[Back](#) [Next](#)

Add crawler

- ✓ Crawler info
- ✓ cmdatalake
- ✓ Crawler source type
- Data stores
- Data store**
- IAM Role
- Schedule
- Output
- Review all steps

Add another data store

☐ Yes

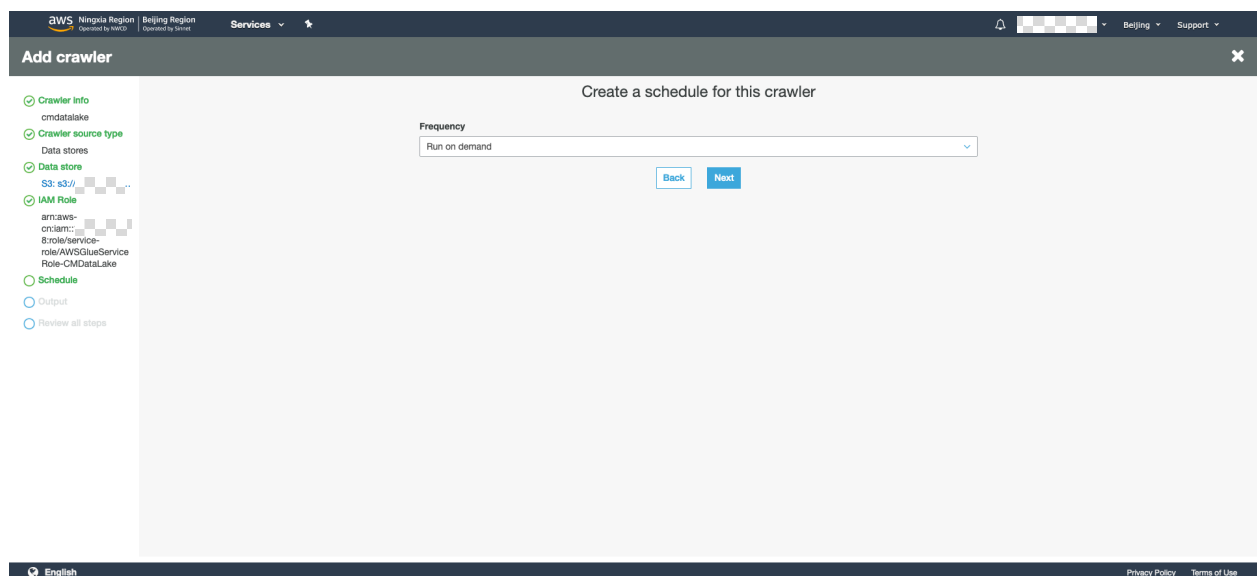
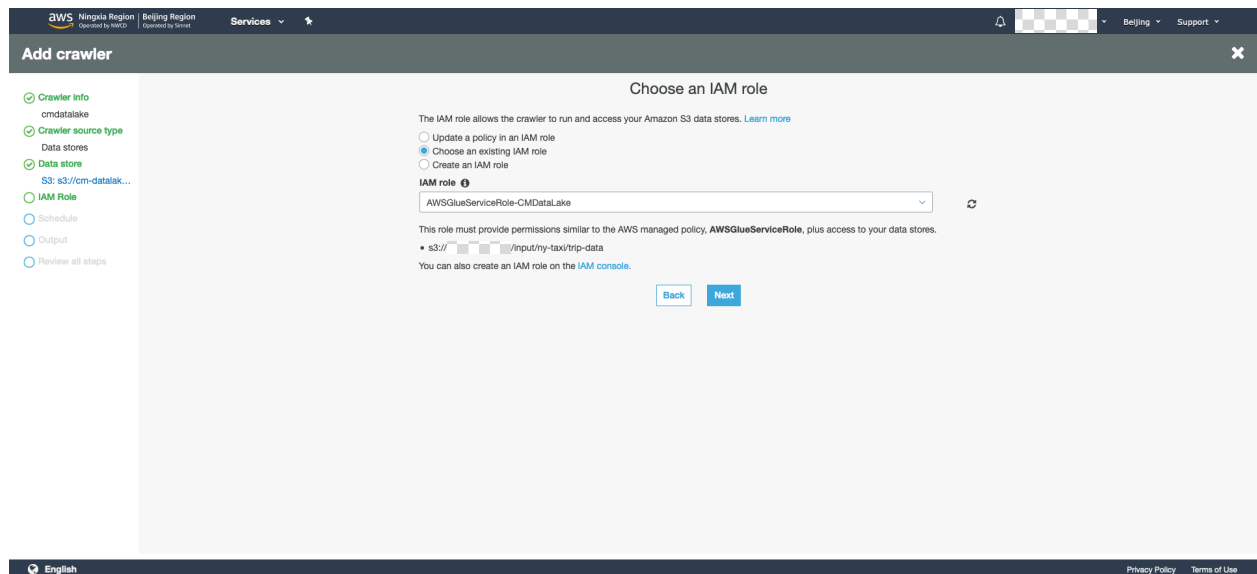
☒ No

[Back](#) [Next](#)

Chosen data stores

S3: s3://[bucket]/input/my-taxi/trip-data

For first time creation, choose the option “Create an IAM role”, otherwise choose the option “Choose an existing IAM role” to specify the role that you have ever created. Here the solution uses the IAM role “AWSGlueServiceRole-CMDDataLake” which has already been created ealier.



Specify the database that you have created in Lake Formation.

The screenshot shows the 'Add crawler' wizard in the AWS Glue console, specifically the 'Configure the crawler's output' step. The left sidebar lists the steps: Crawler info, Crawler source type, Data stores, IAM Role, Schedule, Output, and Review all steps. The main area is titled 'Configure the crawler's output' and contains the following configuration options:

- Database:** A dropdown menu with 'cmdatalake' selected.
- Add database:** A button to add a new database.
- Prefix added to tables (optional):** A text input field with the placeholder 'Type a prefix added to table names'.
- Grouping behavior for S3 data (optional):** A section with a right-pointing arrow.
- Configuration options (optional):** A section with a right-pointing arrow.

At the bottom of the main area are 'Back' and 'Next' buttons. The footer of the console shows 'English', 'Privacy Policy', and 'Terms of Use'.

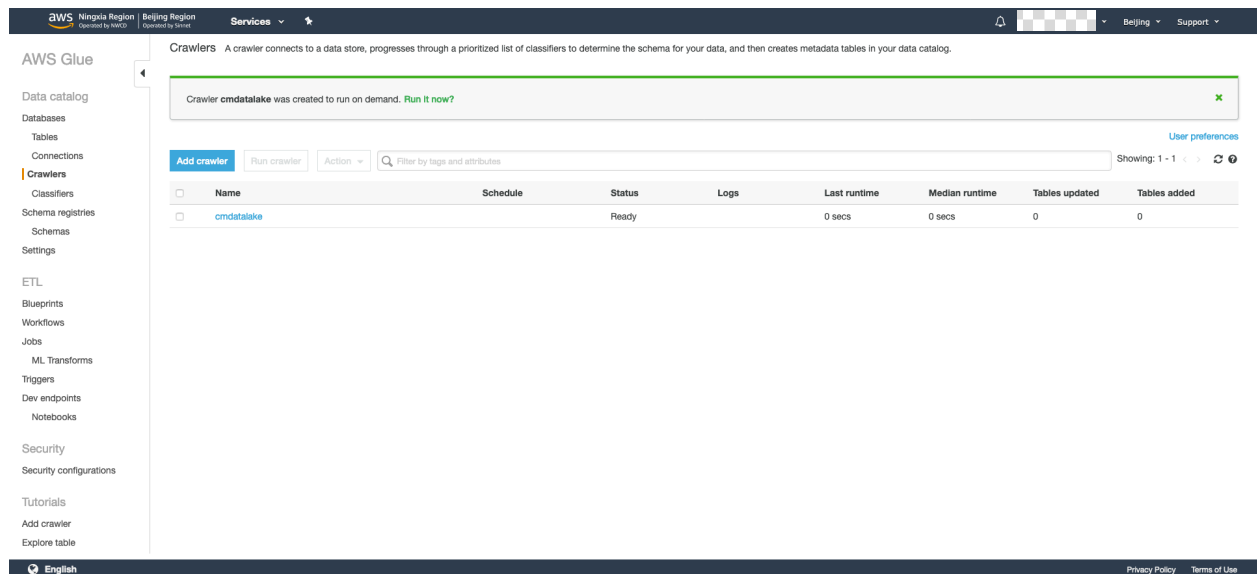
Review the crawler info to make sure that you have set all configurations correctly.

The screenshot shows the 'Add crawler' wizard in the AWS Glue console, specifically the 'Review' step. The left sidebar lists the steps: Crawler info, Crawler source type, Data stores, IAM Role, Schedule, Output, and Review all steps. The main area displays a summary of the crawler configuration in a card-based layout:

- Crawler info:** Name: cmdatalake, Tags: -
- Data stores:** Data store: S3, Include path: s3:///input/ny-taxi/trip-data, Connection: -, Exclude patterns: -
- IAM role:** IAM role: arn:aws-cram:::role/service-role/AWSGlueServiceRole-CMDatalake
- Schedule:** Schedule: Run on demand
- Output:** Database: cmdatalake, Prefix added to tables (optional): Create a single schema for each S3 path: false, Configuration options: -

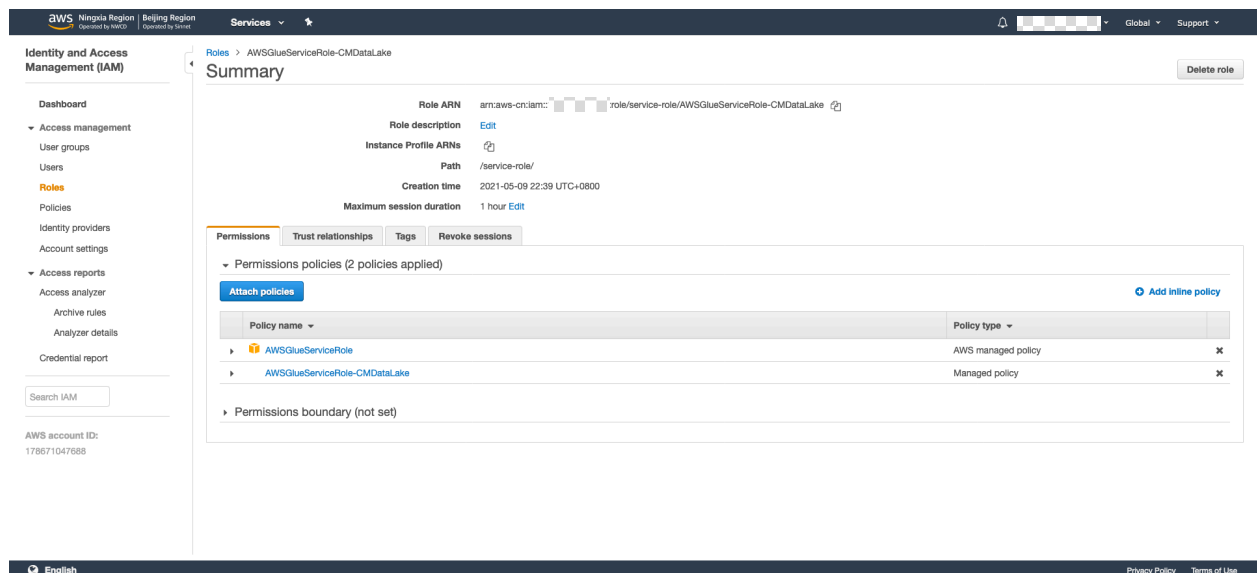
At the bottom of the main area is a 'Finish' button. The footer of the console shows 'English', 'Privacy Policy', and 'Terms of Use'.

Click the “Finish” button to create a crawler.

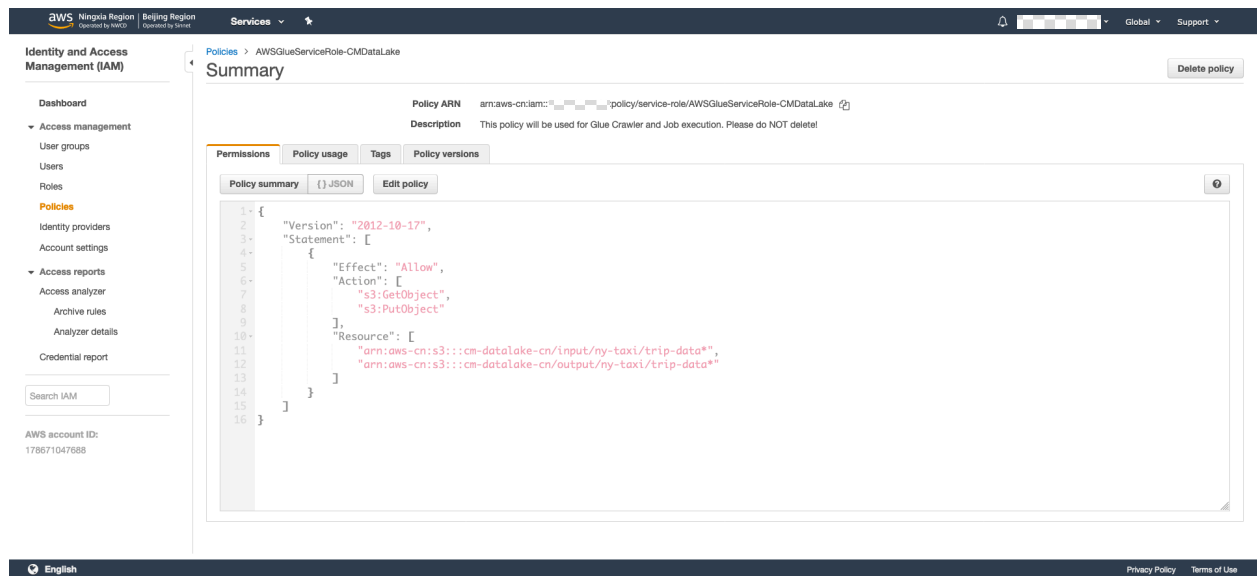


Click the hyper link “Run it now” to launch the crawler.

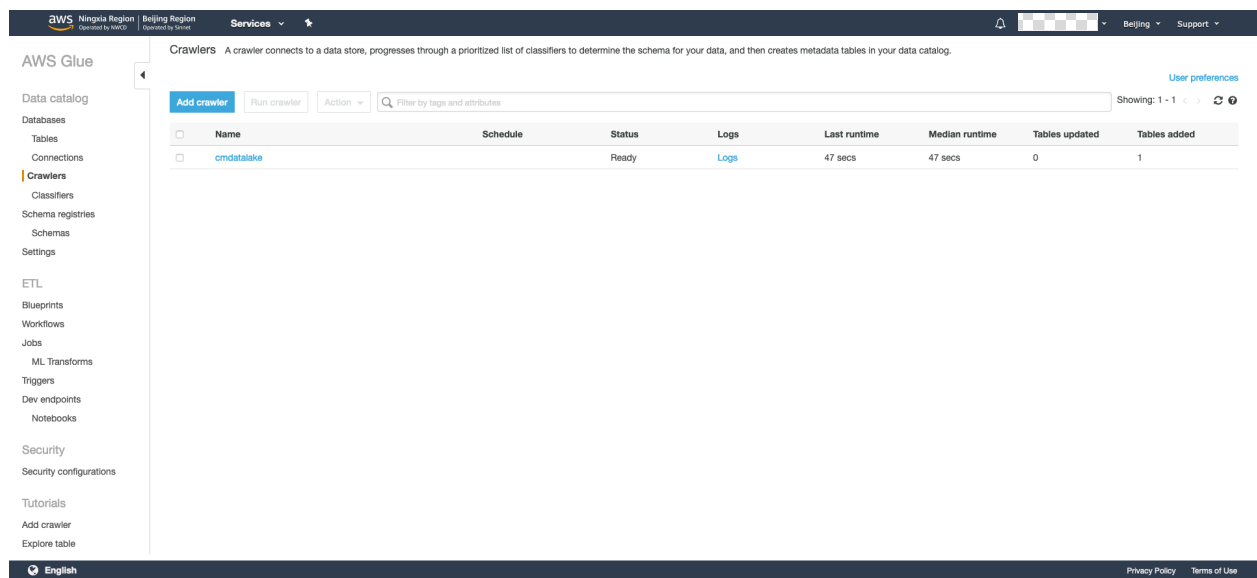
You can go to the IAM console to check the information of the IAM role “AWSGlueServiceRole-CMDDataLake”.



The detailed information for the IAM role “AWSGlueServiceRole-CMDDataLake” is as following. Please be noted that the bucket “arn:aws-cn:s3:::[your-bucket-name]/input/ny-taxi/trip-data*” and “arn:aws-cn:s3:::[your-bucket-name]/output/ny-taxi/trip-data*” specify you can only access these URLs in s3.



The crawler completed its task and created a table.



You can view the detailed information of the table as following.

aws Ningxia Region Beijing Region Services

Tables > trip_data

Last updated 11 May 2021 04:29 PM Table Version (Current version)


Edit table Delete table View properties Compare versions Edit schema

Name trip_data

Description

Database cndatalake

Classification csv

Location s3:///input/ny-taxi/trip-data/

Connection No

Deprecated No

Last updated Tue May 11 16:29:52 GMT+800 2021

Input format org.apache.hadoop.mapred.TextInputFormat

Output format org.apache.hadoop.hive ql.io.HiveHadoopKeyTextOutputFormat

Serde serialization lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

Serde parameters field.delim ,

Table properties

skip.header.line.count	1	sizeKey	68654765	objectCount	1	UPDATED_BY_CRAWLER	cndatalake	CrawlerSchemaSerializerVersion	1.0	recordCount	488331
averageRecordSize	141	CrawlerSchemaDeserializerVersion	1.0	compressionType	none	columnsOrdered	true	areColumnsQuoted	false	delimiter	,
										typeOfData	file

Schema

Showing: 1 - 19 of 19

	Column name	Data type	Partition key	Comment
1	vendorid	bigint		
2	lpep_pickup_datetime	string		
3	lpep_dropoff_datetime	string		
4	store_and_fwd_flag	string		

English Privacy Policy Terms of Use

aws Ningxia Region Beijing Region Services

Schema

Showing: 1 - 19 of 19

	Column name	Data type	Partition key	Comment
1	vendorid	bigint		
2	lpep_pickup_datetime	string		
3	lpep_dropoff_datetime	string		
4	store_and_fwd_flag	string		
5	ratecodeid	bigint		
6	pulocationid	bigint		
7	dolocationid	bigint		
8	passenger_count	bigint		
9	trip_distance	double		
10	fare_amount	double		
11	extra	double		
12	mta_tax	double		
13	tip_amount	double		
14	tolls_amount	double		
15	ehail_fee	string		
16	improvement_surcharge	double		
17	total_amount	double		
18	payment_type	bigint		
19	trip_type	bigint		

English Privacy Policy Terms of Use

You can also view the logs in Amazon CloudWatch^{xviii} for troubleshooting and debugging if you do not get the expected result.

The screenshot shows the AWS CloudWatch Logs console for the crawler 'cndatalake'. The left sidebar contains navigation options like CloudWatch, Dashboards, Alarms, Billing, Logs, Log groups, Metrics, Events, Rules, Event Buses, ServiceLens, Service Map, Traces, Lambda Insights, Performance Monitoring, Synthetics, Canaries, and Contributor Insights. The main panel displays 'Log events' for the log group '/aws-glue/crawlers /cndatalake'. A search bar is at the top, and a table of log events is shown below. The events include messages about running the crawler, classification complete, and finished writing to the catalog.

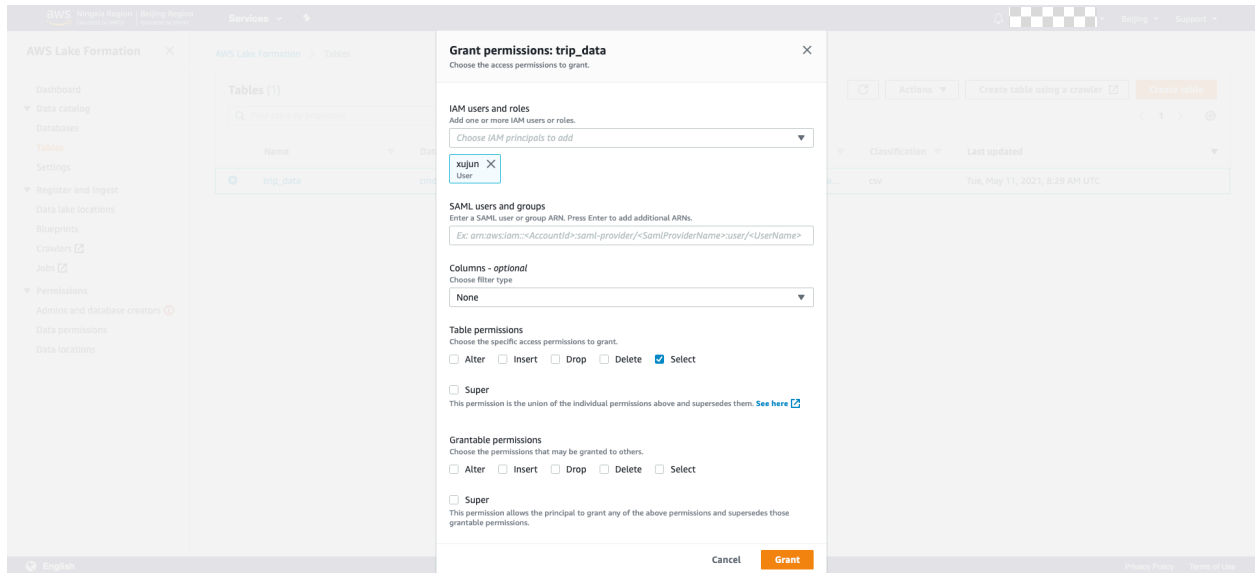
Timestamp	Message
2021-05-11T16:20:48.757+08:00	[89993a38-0451-485e-0633-f9e82ee8e27] BENCHMARK : Running Start Crawl for Crawler cndatalake
2021-05-11T16:20:56.765+08:00	[89993a38-0451-485e-0633-f9e82ee8e27] ERROR : Not all read errors will be logged. com.amazonaws.services.s3.model.AmazonS3Exception: Access Denied (Service: Amazon S3; Status ...
2021-05-11T16:21:04.198+08:00	[89993a38-0451-485e-0633-f9e82ee8e27] BENCHMARK : Classification complete, writing results to database cndatalake
2021-05-11T16:21:04.199+08:00	[89993a38-0451-485e-0633-f9e82ee8e27] INFO : Crawler configured with SchemaChangePolicy {"updateBehavior":"UPDATE_IN_DATABASE","deleteBehavior":"DEPRECATE_IN_DATABASE"}.
2021-05-11T16:21:27.065+08:00	[89993a38-0451-485e-0633-f9e82ee8e27] BENCHMARK : Finished writing to Catalog
2021-05-11T16:22:32.890+08:00	[89993a38-0451-485e-0633-f9e82ee8e27] BENCHMARK : Crawler has finished running and is in state READY
2021-05-11T16:29:15.842+08:00	[04b4945d-380a-446b-954c-1d566f71d9d4] BENCHMARK : Running Start Crawl for Crawler cndatalake
2021-05-11T16:29:30.804+08:00	[04b4945d-380a-446b-954c-1d566f71d9d4] BENCHMARK : Classification complete, writing results to database cndatalake
2021-05-11T16:29:30.804+08:00	[04b4945d-380a-446b-954c-1d566f71d9d4] INFO : Crawler configured with SchemaChangePolicy {"updateBehavior":"UPDATE_IN_DATABASE","deleteBehavior":"DEPRECATE_IN_DATABASE"}.
2021-05-11T16:29:52.398+08:00	[04b4945d-380a-446b-954c-1d566f71d9d4] INFO : Created table trip_data in database cndatalake
2021-05-11T16:29:53.349+08:00	[04b4945d-380a-446b-954c-1d566f71d9d4] BENCHMARK : Finished writing to Catalog
2021-05-11T16:30:58.932+08:00	[04b4945d-380a-446b-954c-1d566f71d9d4] BENCHMARK : Crawler has finished running and is in state READY

To query data in Amazon Athena, you need to grant access permissions for the crawled table in the database you have created. The target user is the IAM user that you have logged into the AWS console. In the console of Lake Formation, choose the table and grant select permissions for the logged IAM user.

The screenshot shows the AWS Lake Formation console. The left sidebar contains navigation options like Dashboard, Data catalog, Databases, Tables, Settings, Register and Ingest, and Permissions. The main panel displays 'Tables (1)' for the database 'cndatalake'. A table lists the 'trip_data' table. A context menu is open over the 'trip_data' table, showing options like Table, Edit, Drop, View data, Permissions, Grant, Revoke, Verify permissions, and View permissions.

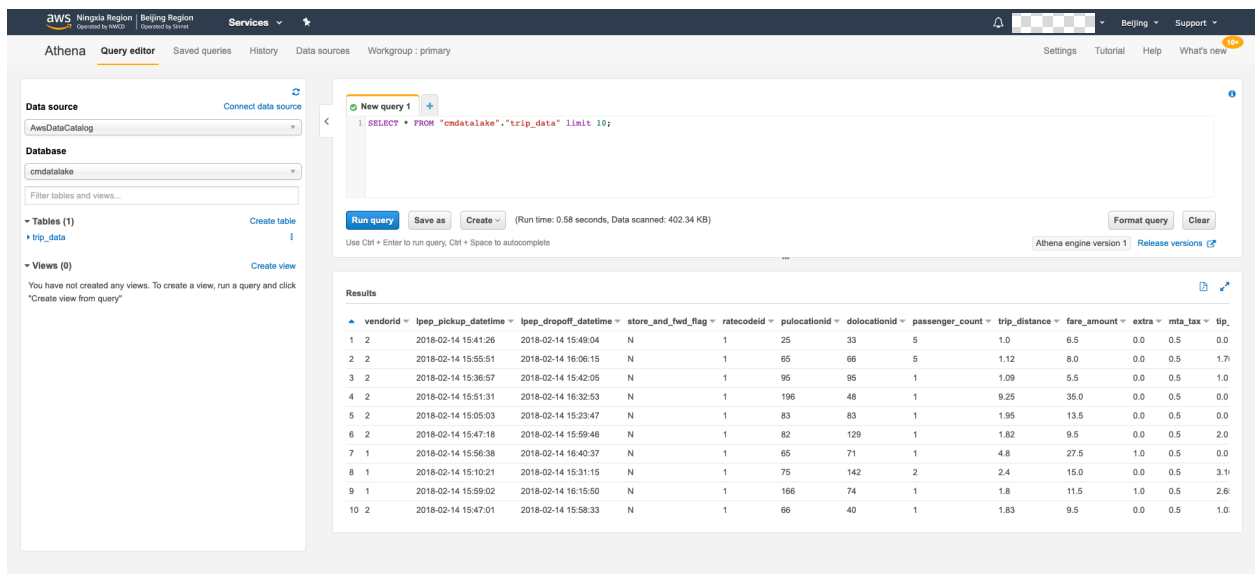
Name	Database	Location	Class	Updated
trip_data	cndatalake	s3:///input/ny-taxi/trip-data...	csv	May 11, 2021, 8:29 AM UTC

Here the solution logged as an IAM user "xujun", just grant "Select" permissions to it.



Lake formation supports more fine-grained access control, which means you can specify the access right to a column in a table. This is a very useful feature for production environment.

Go to the console of Amazon Athena, select the database "cmdatalake", then you can query data from the editor window.



4.1.5. Amazon EMR

In previous chapter, you can crawl data from S3 and query it. However, in the production scenario of connected mobility, the raw data usually cannot be directly used. You need to do a series of ETL work. AWS Glue can be used for this job if you

customize your business logic in the Spark job in AWS Glue. A more general method is to use Amazon EMR.

The following diagrams illustrate how to create an Amazon EMR cluster for such kind of work.

In the console of Amazon EMR, choose advanced options and specify the latest EMR release edition.

aws Ningxia Region Beijing Region Services

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security

Software Configuration

Release: **emr-6.3.0**

<input checked="" type="checkbox"/> Hadoop 3.2.1	<input type="checkbox"/> Zeppelin 0.9.0	<input type="checkbox"/> Livy 0.7.0
<input checked="" type="checkbox"/> JupyterHub 1.2.0	<input type="checkbox"/> Tez 0.9.2	<input type="checkbox"/> Flink 1.12.1
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 2.2.6	<input checked="" type="checkbox"/> Pig 0.17.0
<input checked="" type="checkbox"/> Hive 3.1.2	<input type="checkbox"/> Presto 0.245.1	<input type="checkbox"/> PrestoSQL 350
<input type="checkbox"/> ZooKeeper 3.4.14	<input checked="" type="checkbox"/> JupyterEnterpriseGateway 2.1.0	<input type="checkbox"/> MXNet 1.7.0
<input checked="" type="checkbox"/> Sqoop 1.4.7	<input type="checkbox"/> Hue 4.9.0	<input type="checkbox"/> Phoenix 5.0.0
<input type="checkbox"/> Oozie 5.2.1	<input checked="" type="checkbox"/> Spark 3.1.1	<input type="checkbox"/> HCatalog 3.1.2
<input type="checkbox"/> TensorFlow 2.4.1		

Multiple master nodes (optional)
☐ Use multiple master nodes to improve cluster availability. [Learn more](#)

Edit software settings
☒ Enter configuration ☐ Load JSON from S3

```
classification=conf-file-name,property=anyValue1,anyValue2
```

Steps (optional)
A step is a unit of work you submit to the cluster. For instance, a step might contain one or more Hadoop or Spark jobs. You can also submit additional steps to a cluster after it is running. [Learn more](#)

Concurrency: ☐ Run multiple steps at the same time to improve cluster utilization
☒ Clusters enters waiting state
☐ Cluster auto-terminates

Step type: **Select a step** [Add step](#)

English Privacy Policy Terms of Use

Set the options in Cluster Nodes and Instances page as following.

aws Ningxia Region Beijing Region Services

Cluster Nodes and Instances

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

Node type	Instance type	Instance count	Purchasing option
Master Master - 1	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price
Core Core - 2	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	2 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price
Task Task - 3	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	0 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price

[+ Add task instance group](#)

Total core and task units: 2 Total units

Cluster scaling

Adjust the number of Amazon EC2 instances available to an EMR cluster via EMR-managed scaling or a custom automatic scaling policy. [Learn more](#)

Cluster scaling ☒ Enable Cluster Scaling

English Privacy Policy Terms of Use

Please set the EBS root volume to 100GB for your data storage capacity.



Services

Master - 1 **m5.xlarge** 4 vCores, 16 GiB memory, EBS only storage EBS Storage: 32 GiB 1 Instances **On-demand** Use on-demand as max price

Core - 2 **m5.xlarge** 4 vCores, 16 GiB memory, EBS only storage EBS Storage: 32 GiB 2 Instances **On-demand** Use on-demand as max price

Task - 3 **m5.xlarge** 4 vCores, 16 GiB memory, EBS only storage EBS Storage: 32 GiB 0 Instances **On-demand** Use on-demand as max price

+ Add task instance group

Total core and task units 2 Total units

Cluster scaling
Adjust the number of Amazon EC2 instances available to an EMR cluster via EMR-managed scaling or a custom automatic scaling policy. [Learn more](#)
Cluster scaling ☐ Enable Cluster Scaling

EBS Root Volume
Specify the root device volume size up to 100 GiB. This sizing applies to all instances in the cluster. [Learn more](#)
Root device EBS volume size 100 GiB

Buttons: Cancel Previous Next

Set the S3 folder for logging.

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security

General Options

Cluster name emdatalake

☒ **Logging**
S3 folder s3://aws-logs-...cn-north-1/elasticmapreduce/

☐ **Log encryption**

☒ **Debugging**

☒ **Termination protection**

Tags

Key	Value (optional)
Add a key to create a tag	

Additional Options

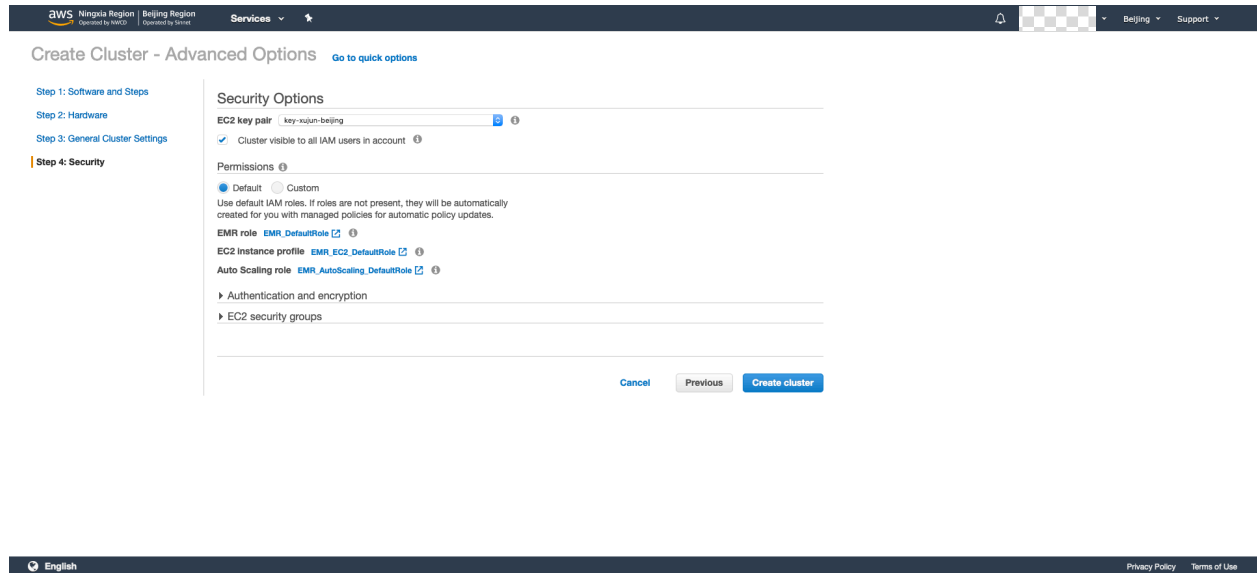
☐ **EMRFS consistent view**

Custom AMI ID None

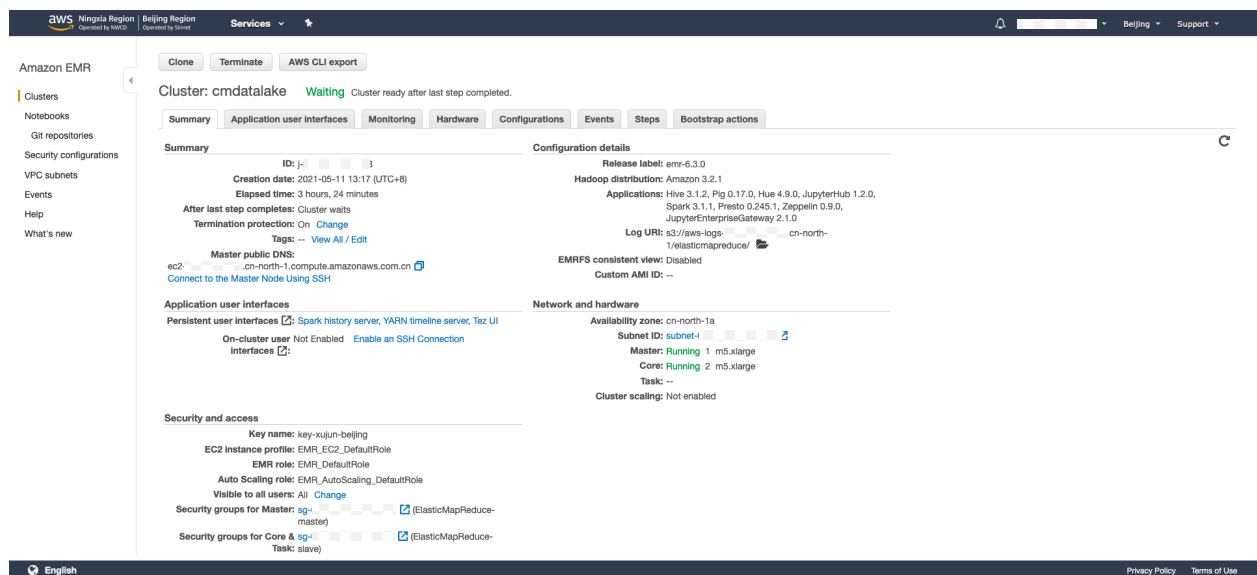
Bootstrap Actions

Buttons: Cancel Previous Next

Choose an EC2 key pair you have already created or create a new one. This EC2 key pair will be used for logging to EMR master node.



The detailed information for the EMR is as following.



4.1.6. Run Spark Job in Amazon EMR Master Node

On your laptop, use a command line terminal to log into the EMR master node.

SSH Login

```
ssh -i key-xujun-beijing.pem hadoop@ec2-#-#-#.cn-north-1.compute.amazonaws.com.cn
```



```
print(updatedNYTaxi.show())

print("Total number of records: " + str(updatedNYTaxi.count()))

updatedNYTaxi.write.parquet(sys.argv[2])
```

Or download the file from your S3 bucket.

wget [https://\[your-s3-bucket\].s3.cn-north-1.amazonaws.com.cn/files/spark/spark-etl.py](https://[your-s3-bucket].s3.cn-north-1.amazonaws.com.cn/files/spark/spark-etl.py)

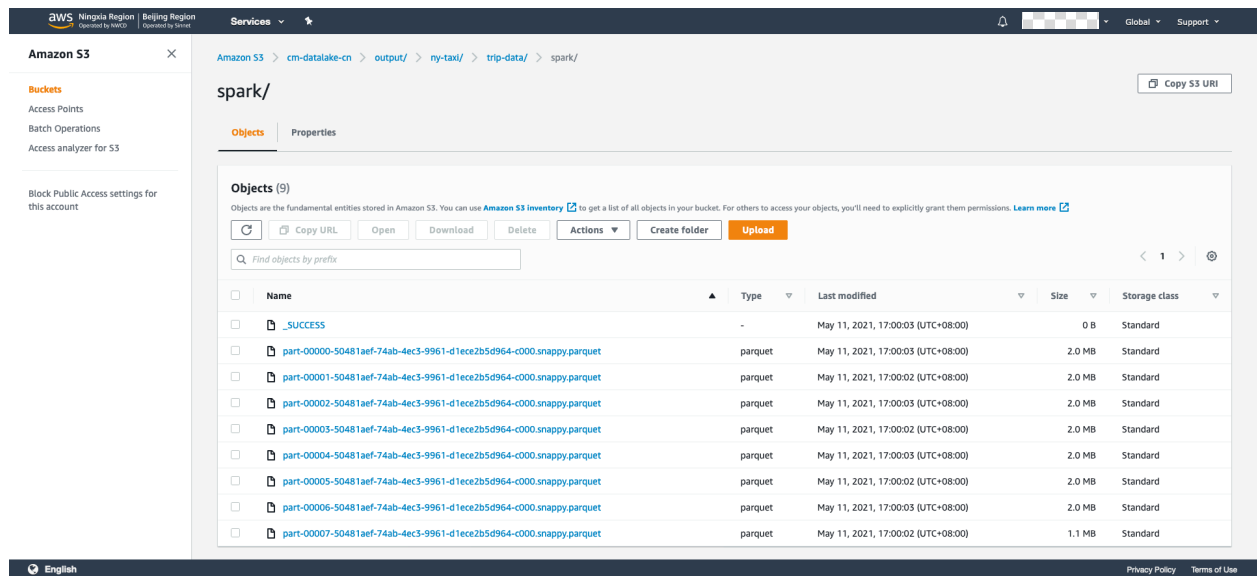
Submit your Spark ETL job.

spark-submit spark-etl.py s3://[your-s3-bucket]/input/ny-taxi/trip-data/ s3://[your-s3-bucket]/output/ny-taxi/trip-data/spark

You can view the output from your local command line window. If the output looks like following, you are on the right track. If not, please check your configurations for troubleshooting.

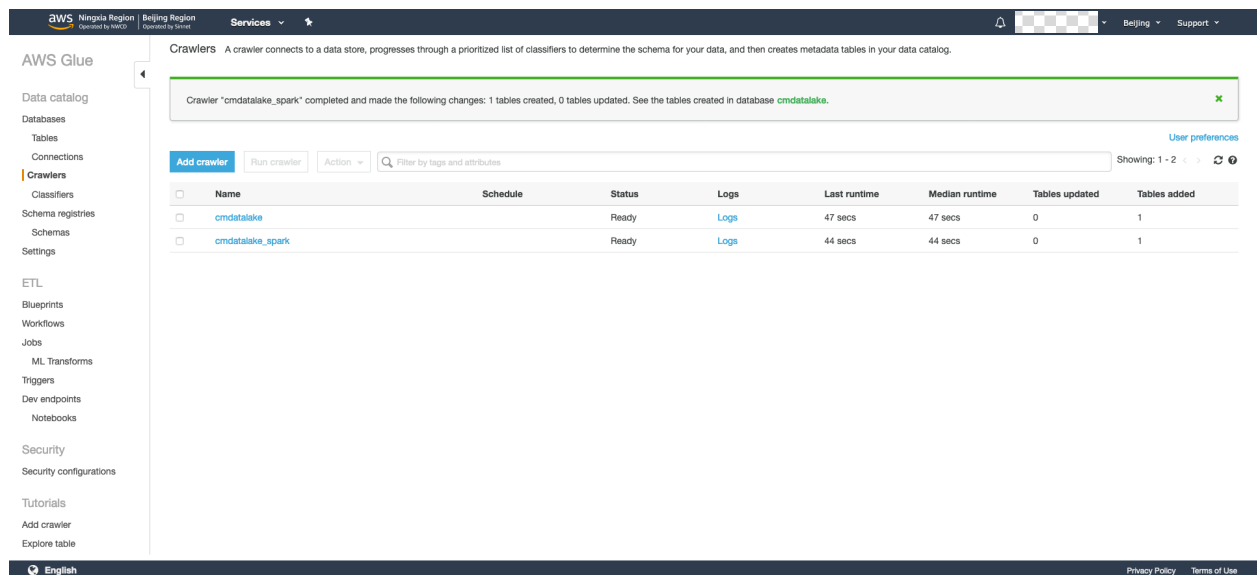
```
20/05/11 00:59:57 INFO CodeGenerator: Code generated in 11.900533 s
20/05/11 00:59:57 INFO SparkContext: Starting job: count at NativeMethodAccessorImpl.java:0
20/05/11 00:59:57 INFO DAGScheduler: getJob() count at NativeMethodAccessorImpl.java:0 with 1 output partitions
20/05/11 00:59:57 INFO DAGScheduler: Final stage: ResultStage 5 (count at NativeMethodAccessorImpl.java:0)
20/05/11 00:59:57 INFO DAGScheduler: Parents of final stage: List()
20/05/11 00:59:57 INFO DAGScheduler: Missing parents: List()
20/05/11 00:59:57 INFO DAGScheduler: Submitting ResultStage 5 (MapPartitionsMapOutput) at count at NativeMethodAccessorImpl.java:0, which has no missing parents
20/05/11 00:59:57 INFO MemoryStore: Block broadcast_8_piece0 stored as bytes in memory (estimated size 5.2 KiB, free 911.4 MiB)
20/05/11 00:59:57 INFO MemoryStore: Added broadcast_8_piece0 in memory on ip-172-31-16-213.cn-north-1.compute.internal:4147 (size: 5.2 KiB, free: 912.2 MiB)
20/05/11 00:59:57 INFO SparkContext: Created broadcast 0 from broadcast at DAGScheduler.scala:1479
20/05/11 00:59:57 INFO DAGScheduler: Submitting 2 waiting tasks from ResultStage 5 (MapPartitionsMapOutput) at count at NativeMethodAccessorImpl.java:0 (first 15 tasks are for partitions Vector(0))
20/05/11 00:59:57 INFO TaskScheduler: Adding task set 5.8 with 1 tasks resource profile 0
20/05/11 00:59:57 INFO TaskSetManager: Starting task 0.0 in stage 5.0 (TID 28) (ip-172-31-16-213.cn-north-1.compute.internal, executor 2, partition 0, RACK_LOCAL, 4797 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO TaskSetManager: Added broadcast_8_piece0 in memory on ip-172-31-16-213.cn-north-1.compute.internal:4147 (size: 5.2 KiB, free: 911.4 MiB)
20/05/11 00:59:57 INFO TaskSetManager: Finished task 0.0 in stage 5.0 (TID 28) on ip-172-31-16-213.cn-north-1.compute.internal (executor 2) (3/1)
20/05/11 00:59:57 INFO TaskScheduler: Removed TaskSet 5.8, whose tasks have all completed, from pool
20/05/11 00:59:57 INFO DAGScheduler: Job 4 is finished. Cancelling potential speculative or zombie tasks for this job
20/05/11 00:59:57 INFO TaskScheduler: Killing all running tasks in stage 5: Stage finished
20/05/11 00:59:57 INFO DAGScheduler: Job 4 finished: count at NativeMethodAccessorImpl.java:0, took 0.146993 s
Total number of records: 76940
20/05/11 00:59:57 INFO FileSourceStrategy: Pushed Filters:
20/05/11 00:59:57 INFO FileSourceStrategy: Post-Scan Filters:
20/05/11 00:59:57 INFO ParquetFileFormat: Using user defined output codec for Parquet: com.amazonaws.co.connector.EarOptimizedSparkSqlParquetOutputCodecWriter
20/05/11 00:59:57 INFO EarOptimizedParquetOutputCodecWriter: EarOptimizedParquetOutputCodecWriter: com.amazonaws.co.connector.EarOptimizedSparkSqlParquetOutputCodecWriter
20/05/11 00:59:57 INFO EarOptimizedParquetOutputCodecWriter: Using output codec class org.apache.hadoop.mapreduce.lib.output.FileSystemOptimizedCodecWriter
20/05/11 00:59:57 INFO FileOutputCommitter: File Output Committer Algorithm version is 2
20/05/11 00:59:57 INFO FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: true
20/05/11 00:59:57 INFO FileOutputCommitter: Using output codec class com.amazonaws.co.connector.EarOptimizedSparkSqlParquetOutputCodecWriter
20/05/11 00:59:57 INFO FileOutputCommitter: Nothing to setup as successful task attempt outputs are written directly
20/05/11 00:59:57 INFO CodeGenerator: Code generated in 20.552973 s
20/05/11 00:59:57 INFO MemoryStore: Block broadcast_9 stored as values in memory (estimated size 371.2 KiB, free 911.1 MiB)
20/05/11 00:59:57 INFO MemoryStore: Block broadcast_9_piece0 stored as bytes in memory (estimated size 35.5 KiB, free 911.0 MiB)
20/05/11 00:59:57 INFO MemoryStore: Added broadcast_9_piece0 in memory on ip-172-31-16-213.cn-north-1.compute.internal:4147 (size: 35.5 KiB, free: 912.2 MiB)
20/05/11 00:59:57 INFO SparkContext: Created broadcast 9 from parquet at NativeMethodAccessorImpl.java:0
20/05/11 00:59:57 INFO FileSourceCodec: relation Name, fileSplitFilePartitionsHistogram: Vector(1 fileSplitFile)
20/05/11 00:59:57 INFO SparkContext: Starting job: parquet at NativeMethodAccessorImpl.java:0
20/05/11 00:59:57 INFO DAGScheduler: getJob() parquet at NativeMethodAccessorImpl.java:0 with 8 output partitions
20/05/11 00:59:57 INFO DAGScheduler: Final stage: ResultStage 6 (parquet at NativeMethodAccessorImpl.java:0)
20/05/11 00:59:57 INFO DAGScheduler: Parents of final stage: List()
20/05/11 00:59:57 INFO DAGScheduler: Missing parents: List()
20/05/11 00:59:57 INFO DAGScheduler: Submitting ResultStage 6 (MapPartitionsMapOutput) at parquet at NativeMethodAccessorImpl.java:0, which has no missing parents
20/05/11 00:59:57 INFO MemoryStore: Block broadcast_10 stored as values in memory (estimated size 221.4 KiB, free 910.8 MiB)
20/05/11 00:59:57 INFO MemoryStore: Block broadcast_10_piece0 stored as bytes in memory (estimated size 82.1 KiB, free 910.7 MiB)
20/05/11 00:59:57 INFO MemoryStore: Added broadcast_10_piece0 in memory on ip-172-31-16-213.cn-north-1.compute.internal:4147 (size: 82.1 KiB, free: 912.3 MiB)
20/05/11 00:59:57 INFO SparkContext: Created broadcast 10 from broadcast at DAGScheduler.scala:1479
20/05/11 00:59:57 INFO DAGScheduler: Submitting 8 waiting tasks from ResultStage 6 (MapPartitionsMapOutput) at parquet at NativeMethodAccessorImpl.java:0 (first 15 tasks are for partitions Vector(0, 1, 2, 3, 4, 5, 6, 7))
20/05/11 00:59:57 INFO TaskScheduler: Adding task set 6.8 with 8 tasks resource profile 0
20/05/11 00:59:57 INFO TaskSetManager: Starting task 1.0 in stage 6.0 (TID 29) (ip-172-31-16-185.cn-north-1.compute.internal, executor 1, partition 1, RACK_LOCAL, 3192 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO TaskSetManager: Starting task 2.0 in stage 6.0 (TID 29) (ip-172-31-16-213.cn-north-1.compute.internal, executor 2, partition 2, RACK_LOCAL, 3192 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO TaskSetManager: Starting task 3.0 in stage 6.0 (TID 29) (ip-172-31-16-185.cn-north-1.compute.internal, executor 1, partition 3, RACK_LOCAL, 3192 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO TaskSetManager: Starting task 4.0 in stage 6.0 (TID 29) (ip-172-31-16-213.cn-north-1.compute.internal, executor 2, partition 4, RACK_LOCAL, 3192 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO TaskSetManager: Starting task 5.0 in stage 6.0 (TID 29) (ip-172-31-16-185.cn-north-1.compute.internal, executor 1, partition 5, RACK_LOCAL, 3192 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO TaskSetManager: Starting task 6.0 in stage 6.0 (TID 29) (ip-172-31-16-213.cn-north-1.compute.internal, executor 2, partition 6, RACK_LOCAL, 3192 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO TaskSetManager: Starting task 7.0 in stage 6.0 (TID 29) (ip-172-31-16-185.cn-north-1.compute.internal, executor 1, partition 7, RACK_LOCAL, 3192 bytes) taskResourceAssignments Map()
20/05/11 00:59:57 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on ip-172-31-16-213.cn-north-1.compute.internal:4147 (size: 82.1 KiB, free: 4.0 GiB)
20/05/11 00:59:58 INFO BlockManagerInfo: Added broadcast_5_piece0 in memory on ip-172-31-16-213.cn-north-1.compute.internal:4147 (size: 35.5 KiB, free: 4.0 GiB)
20/05/11 00:59:58 INFO BlockManagerInfo: Added broadcast_5_piece0 in memory on ip-172-31-16-185.cn-north-1.compute.internal:42329 (size: 35.5 KiB, free: 4.0 GiB)
20/05/11 00:59:58 INFO TaskSetManager: Finished task 1.0 in stage 6.0 (TID 29) on ip-172-31-16-185.cn-north-1.compute.internal (executor 1) (2/8)
20/05/11 00:59:58 INFO TaskSetManager: Finished task 2.0 in stage 6.0 (TID 29) on ip-172-31-16-213.cn-north-1.compute.internal (executor 2) (2/8)
20/05/11 00:59:58 INFO TaskSetManager: Finished task 3.0 in stage 6.0 (TID 29) on ip-172-31-16-185.cn-north-1.compute.internal (executor 1) (4/8)
20/05/11 00:59:58 INFO TaskSetManager: Finished task 4.0 in stage 6.0 (TID 29) on ip-172-31-16-185.cn-north-1.compute.internal (executor 1) (4/8)
20/05/11 00:59:58 INFO TaskSetManager: Finished task 5.0 in stage 6.0 (TID 29) on ip-172-31-16-185.cn-north-1.compute.internal (executor 1) (4/8)
20/05/11 00:59:58 INFO TaskSetManager: Finished task 6.0 in stage 6.0 (TID 29) on ip-172-31-16-213.cn-north-1.compute.internal (executor 2) (17/8)
20/05/11 00:59:58 INFO TaskSetManager: Finished task 7.0 in stage 6.0 (TID 29) on ip-172-31-16-213.cn-north-1.compute.internal (executor 2) (8/8)
20/05/11 00:59:58 INFO TaskScheduler: Removed TaskSet 6.8, whose tasks have all completed, from pool
```

On the console of Amazon S3, you can see the file has been transformed from CSV to parquet format, the total file size decrease from approximately 65 MB to the 15MB in total.

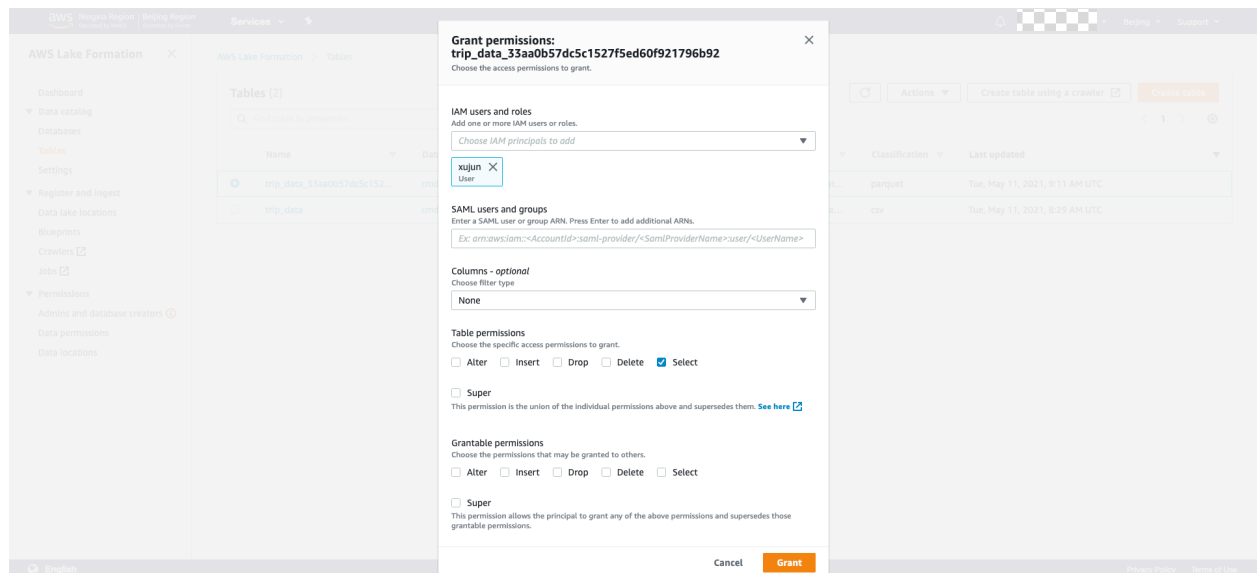
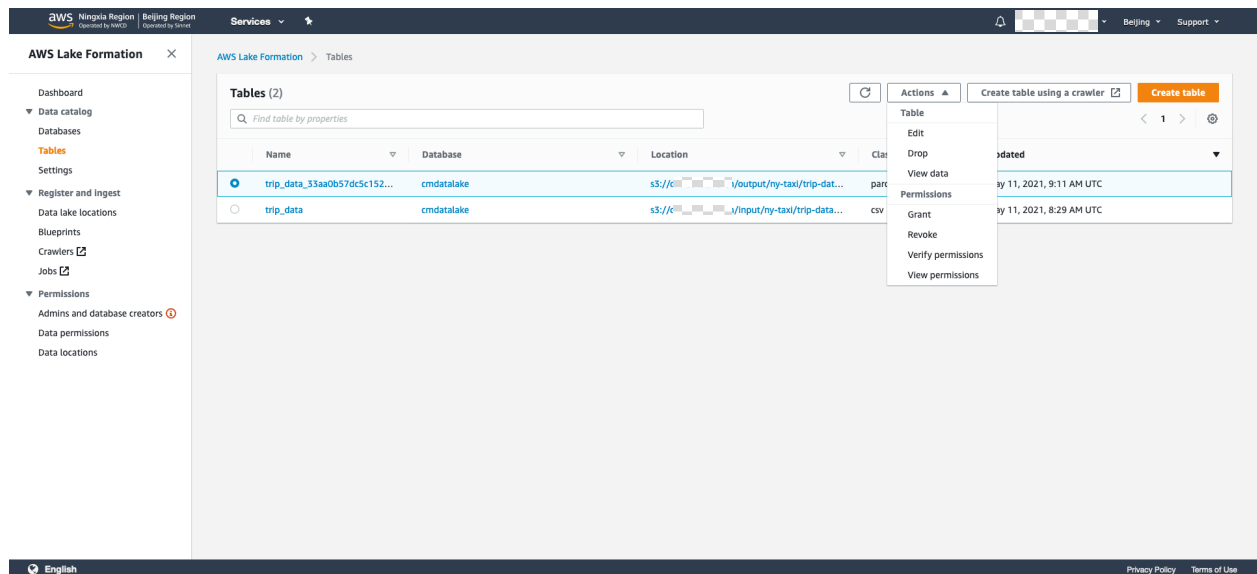


Create another crawler named “cmdatalake_spark” to crawl the data output by above Spark ETL job, which is located in the URL “s3://[your-s3-bucket]/output/ny-taxi/trip-data/spark”.

Crawler “cmdatalake_spark” worked as expected and added a table.



Grant access permissions for the newly created table.



Query data in Amazon Athena. You can find a new table column named “current_date” is added. In this way, you can enrich your information for the original data. It is very useful for connected mobility using scenarios for adding more sensor fusion data.

4.1.7. Run Spark Job in Amazon EMR Jupyter Notebook

Open another command line terminal on your laptop and launch the following SSH tunneling command.

```
# SSH Tunnel
ssh -i key-xujun-beijing.pem -N -L 9443:ec2-#-#-#.cn-north-1.compute.amazonaws.com.cn:9443 hadoop@ec2-#-#-#.cn-north-1.compute.amazonaws.com.cn
```

Use <https://localhost:9443> (Please be noted that it is https not http.) in your local web browser to visit your EMR Jupyter Notebook console.

Please login via the default credentials listed as following:

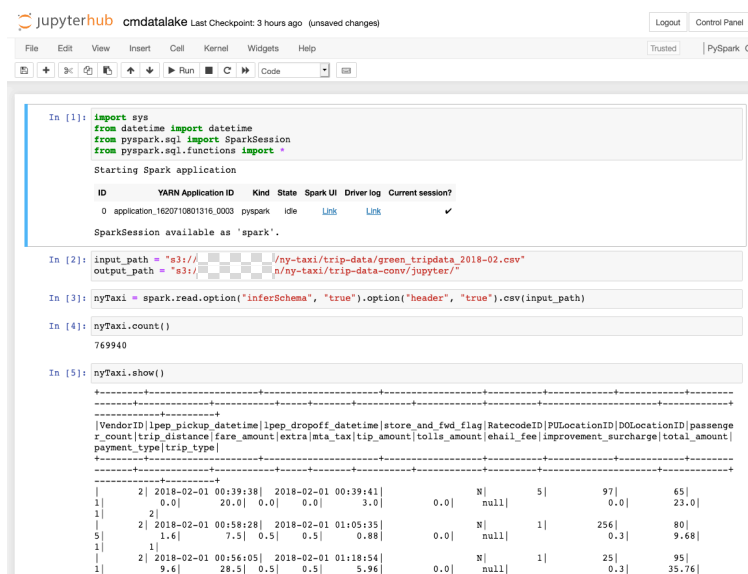
- Username: jovyan
- Password: jupyter

```
21/05/11 09:06:02 INFO SparkManagerFactory: ShutdownManagerFactory: stopped
21/05/11 09:06:02 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/05/11 09:06:02 INFO SparkContext: Successfully stopped SparkContext
21/05/11 09:06:02 INFO ShutdownHookManager: Shutdown hook called
21/05/11 09:06:02 INFO ShutdownHookManager: Deleting directory /mnt/rtp/spark-62a319d8-8e13-4ba9-03ba-c70f39f609d2
21/05/11 09:06:02 INFO ShutdownHookManager: Deleting directory /mnt/rtp/spark-844d654-8eda-47da-bcc9-9b876c0e7ffe
21/05/11 09:06:02 INFO ShutdownHookManager: Deleting directory /mnt/rtp/spark-844d654-8eda-47da-bcc9-9b876c0e7ffe
hadoop@ip-172-31-19-112 ~$ clear
hadoop@ip-172-31-19-112 ~$ ls
spark-ctl.py
hadoop@ip-172-31-19-112 ~$
hadoop@ip-172-31-19-112 ~$ exit
logout
Connection to ec2-52-81-76-199.cn-north-1.compute.amazonaws.com.cn closed.
196599d2c05:Credentials xujunaws ssh -i key-xujun-beijing.pem -N -L 9443:ec2-52-81-76-199.cn-north-1.compute.amazonaws.com.cn:9443 hadoop@ec2-52-81-76-199.cn-north-1.compute.amazonaws.com.cn
```

Open the Jupyterhub console and upload the cmdatalake.ipynb in the deployment guide bundle. The directory for this file is “HOME_DIR_DEPLOYMENT_GUIDE/emr/jupyter/cmdatalake.ipynb”.



Select Kernel as PySpark, run the codes step-by-step as shown in the following diagram.



In this way, data can also be processed and transformed.

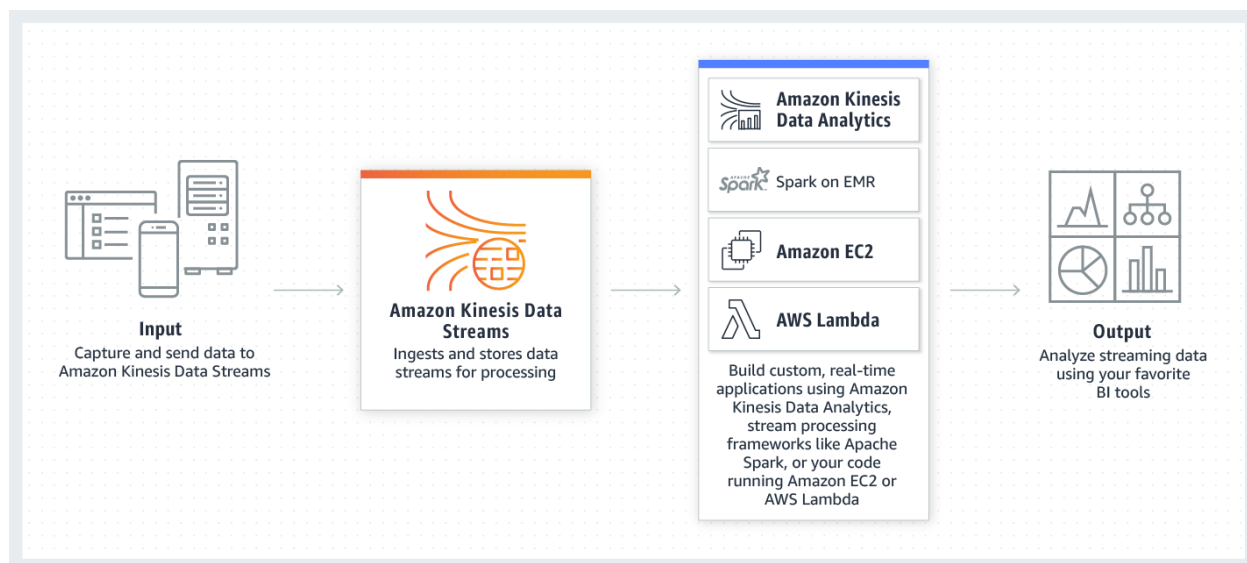
4.2. Streaming Mode

In the using scenario of connected mobility, a more common and challenging using scenario is real time, i.e., streaming mode. Before diving deep into the details of streaming mode, it is necessary to introduce the related Amazon services.

4.2.1. Amazon Kinesis Data Stream

Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. The data collected is

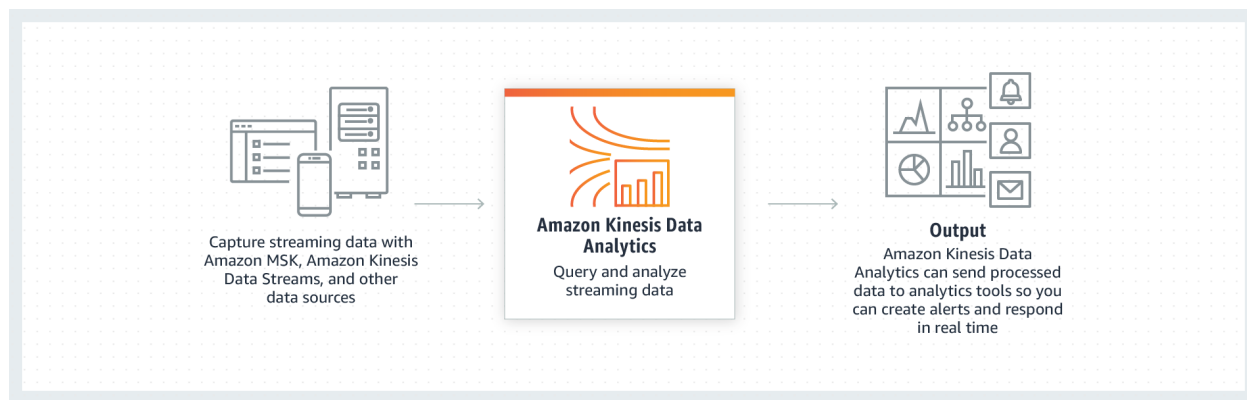
available in milliseconds to enable real-time analytics use cases such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more.



4.2.2. Amazon Kinesis Data Analytics

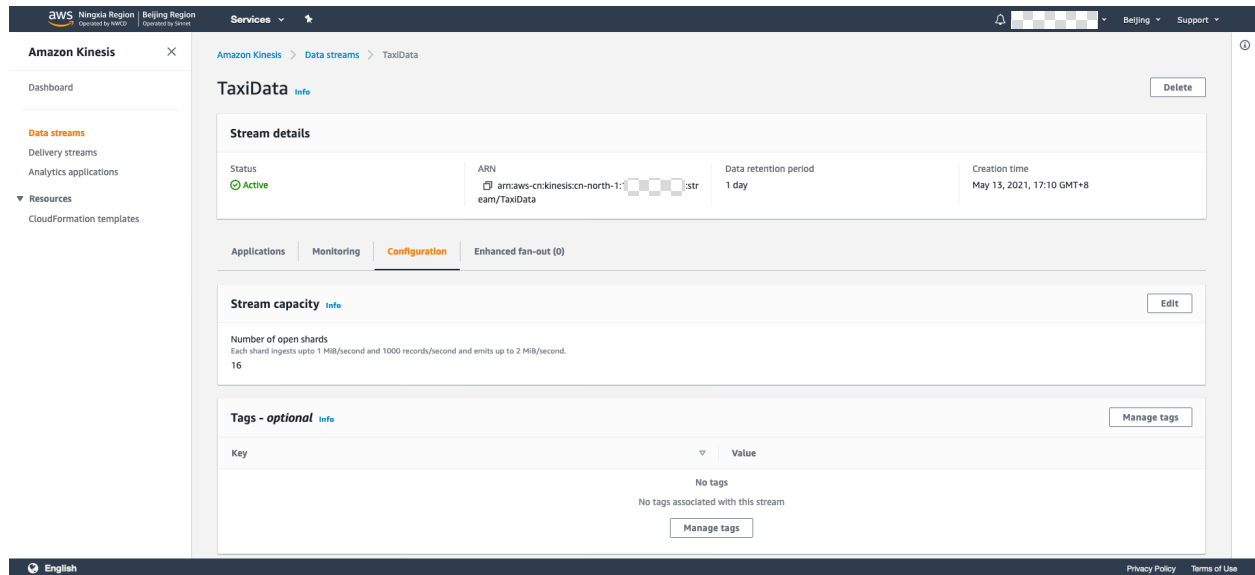
Amazon Kinesis Data Analytics is the easiest way to transform and analyze streaming data in real time with Apache Flink. Apache Flink is an open source framework and engine for processing data streams. Amazon Kinesis Data Analytics reduces the complexity of building, managing, and integrating Apache Flink applications with other AWS services.

Amazon Kinesis Data Analytics takes care of everything required to run streaming applications continuously, and scales automatically to match the volume and throughput of your incoming data. With Amazon Kinesis Data Analytics, there are no servers to manage, no minimum fee or setup cost, and you only pay for the resources your streaming applications consume.



4.2.3. Data Streaming and Analytics

In this guide, create an Amazon Kinesis Data Stream named “TaxiData” with stream capacity of 16 open shards. You can set the number of shards to be 8 or lower according to your specific using requirements.



This solution uses an Amazon EC2 instance for data stream synthesizing. You can log into your EC2 instance as following by specifying the .pem file and EC2 IP address to be your own.

```
Ssh -I "key-xujun-beijing.pem" ec2-user@ec2-#-#-#.cn-north-1.compute.amazonaws.com.cn
```

Download the data file and streaming script file from your S3 bucket, run the streaming script for synthesizing data streams.

```
Wget https://[your-s3-bucket].s3.cn-north-1.amazonaws.com.cn/files/kinesis/python/streaming.py
```

```
wget https://[your-s3-bucket].s3.cn-north-1.amazonaws.com.cn/input/ny-taxi/trip-data/yellow_tripdata_2015-12.csv
```

```
nohup python3 streaming.py >> my.log 2>&1 &
```

The data streams will be sent to Amazon Kinesis Data Analytics for subsequent processing. The above instruction will output the streaming process id for tracking. You can kill the streaming process for debugging or specific requirements.

Sudo kill -9 process-id

Once the data stream is generated, you can use Amazon Kinesis Data Analytics for following analytics.

The screenshot shows the 'Kinesis Data Analytics - Create application' page in the AWS console. The left sidebar shows the 'Amazon Kinesis' menu with options for 'Dashboard', 'Data Streams', 'Data Firehose', and 'Data Analytics' (selected). The main content area is titled 'Kinesis Data Analytics - Create application' and includes a description of the service. Below the description, there are fields for 'Application name' (filled with 'SteamingAnalytics') and 'Description - optional' (filled with 'Demonstrate how to analyze real time data in Amazon Kinesis Data Analytics.'). The 'Runtime' section has two options: 'SQL' (selected) and 'Apache Flink'. Below this is the 'Tags - optional' section, which includes a table for adding tags with columns for 'Key' and 'Value - optional'. At the bottom right, there are 'Cancel' and 'Create application' buttons.

aws Ningxia Region Beijing Region Services

Amazon Kinesis

Dashboard

Data Streams

Data Firehose

Data Analytics

Video Streams

Kinesis Data Analytics - Create application

Kinesis Data Analytics applications continuously read and analyze data from a connected streaming source in real-time. To enable interactivity with your data during configuration you will be prompted to run your application. Kinesis Data Analytics resources are not covered under the [AWS Free Tier](#), and usage-based charges apply. For more information, see [Kinesis Data Analytics pricing](#).

Application name
SteamingAnalytics

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

Description - optional
Demonstrate how to analyze real time data in Amazon Kinesis Data Analytics.

Runtime

☒ SQL
Process data in real-time using SQL, which provides an easy way to quickly query large volumes of streaming data without learning new frameworks or languages. [Learn more](#)

☐ Apache Flink
Apache Flink is an open-source framework and distributed processing engine for stateful computations over unbounded and bounded data streams. Use this option to build Apache Flink Applications in Java, Scala, and Python. [Learn more](#)

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs. [Learn more](#)

Key	Value - optional	
<input type="text" value="Enter key"/>	<input type="text" value="Enter value"/>	Remove

Add tag

You can add up to 49 tags.

Cancel Create application

English Privacy Policy Terms of Use

The screenshot shows the 'SteamingAnalytics' application configuration page in the AWS console. The left sidebar shows the 'Amazon Kinesis' menu with options for 'Dashboard', 'Data Streams', 'Data Firehose', and 'Data Analytics' (selected). The main content area is titled 'SteamingAnalytics' and includes a description of the application. Below the description, there is a success message: 'Successfully created Application SteamingAnalytics. Next, choose Connect streaming data.' The 'Source' section has a 'Streaming data' icon and a 'Connect streaming data' button. The 'Reference data - optional' section has a 'Reference data' icon and a 'Connect reference data' button. The 'Real time analytics' section has a 'Real time analytics' icon and a 'Connect real time analytics' button. The 'Destination - optional' section has a 'Destination' icon and a 'Connect destination' button.

aws Ningxia Region Beijing Region Services

Amazon Kinesis

Dashboard

Data Streams

Data Firehose

Data Analytics

Video Streams

SteamingAnalytics

Description: Demonstrate how to analyze real time data in Amazon Kinesis Data Analytics.

Application ARN: arn:aws-cn:kinesisanalytics:cn-north-1:178671047688:application/SteamingAnalytics

Application version ID: 1

Successfully created Application SteamingAnalytics
Next, choose Connect streaming data.

Source

Streaming data
Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. Each application can connect to one streaming data source. [Learn more](#)

Connect streaming data

Reference data - optional
Enrich data from your streaming data source with JSON or CSV data stored as an object in Amazon S3. Each application can connect to one reference data source.

Real time analytics
Author your own SQL queries or add SQL from templates to easily analyze your source data.

Destination - optional
Connect an in-application stream to a Kinesis stream, or to a Firehose delivery stream, to continuously deliver SQL results to AWS destinations. The limit is three destinations for each application. [Learn more](#)

aws

Ningxia Region

Beijing Region

Services

Beijing

Support

Amazon Kinesis

Dashboard

Data Streams

Data Firehose

Data Analytics

Video Streams

Kinesis Data Analytics applications

StreamingAnalytics

Streaming data

Connect streaming data source

Choose from your Kinesis data streams and Firehose delivery streams, or quickly configure a demo Kinesis stream that can be used to explore Kinesis Analytics.

Choose source

Configure a new stream

Source

Kinesis data stream

Kinesis data stream is an ordered sequence of data records used for rapid and continuous data intake and aggregation.

Kinesis Firehose delivery stream

Kinesis Firehose delivery streams send source records to the destinations that you specify, automatically and continuously.

Kinesis data stream

TaxiData

Create new

View TaxiData in Kinesis data streams

In-application stream name

In your SQL queries, refer to this source as: **SOURCE_SQL_STREAM_001**

Record pre-processing with AWS Lambda

Kinesis Data Analytics can invoke your Lambda function to pre-process records before they are used in this application. To pre-process records, your Lambda function must be compliant with the required record transformation output model. [Learn more](#)

Record pre-processing

Disabled

Enabled

The following diagram illustrates the schema discovery succeeded.

Schema

Schema discovery can generate a schema using recent records from the source. Schema column names are the same as in the source, unless they contain special characters, repeated column names, or reserved keywords. [Learn more](#)

Schema discovery successful

Detected JSON format and applied schema

To define a custom schema, choose "Edit schema" in the stream sample below.

To capture a new stream sample from the selected source for discovery, choose **Retry schema discovery** below.

Edit schema

Retry schema discovery

Raw

Lambda output

Formatted

Filter by column name

VendorID	VendorID	VendorID	VendorID	VendorID	VendorID	VendorID
INTEGER	INTEGER	INTEGER	INTEGER	INTEGER	INTEGER	INTEGER
1	2015/12/4 21:56	2015/12/4 22:02	1	0.9	-73.98334503	40.69375992
1	2015/12/4 20:49	2015/12/4 20:56	1	1.1	-73.95323181	40.77870178
1	2015/12/4 22:06	2015/12/4 22:14	1	1.9000000000000001	-73.95954132	40.800975799999
2	2015/12/4 17:32	2015/12/4 17:40	1	1.59	-73.96640776	40.76208115
2	2015/12/4 20:38	2015/12/4 20:53	1	2.15	-73.98758698	40.72274017
1	2015/12/4 20:47	2015/12/4 21:09	2	2.2	-73.98984714	40.73420715
2	2015/12/4 21:22	2015/12/4 21:39	1	5.5200000000000005	-74.0147834	40.71401596
2	2015/12/4 17:33	2015/12/4 17:43	1	1.1500000000000001	-74.00463104	40.70714951
1	2015/12/4 21:39	2015/12/4 22:05	1	8.0	-73.99240112	40.72858429
1	2015/12/4 20:18	2015/12/4 20:42	1	2.5	-73.97194672	40.75712585

Cancel

Save and continue

31

Amazon Kinesis | Kinesis Data Analytics applications > SteamingAnalytics

Application status: READY

Description: Demonstrate how to analyze real time data in Amazon Kinesis Data Analytics.
Application ARN: arn:aws-cn:kinesisanalytics:cn-north-1:123456789012:application/SteamingAnalytics
Application version ID: 2

Source
 Streaming data
 Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. Each application can connect to one streaming data source. [Learn more](#)

Source	In-application stream name	ID	Record pre-processing
Kinesis stream TaxiData	SOURCE_SQL_STREAM_001	2.1	Disabled

Reference data - optional
 Enrich data from your streaming data source with JSON or CSV data stored as an object in Amazon S3. Each application can connect to one reference data source. [Learn more](#)
[Connect reference data](#)

Real time analytics
 Author your own SQL queries or add SQL from templates to easily analyze your source data. [Learn more](#)
[Go to SQL editor](#)

Destination - optional
 Connect an in-application stream to a Kinesis stream, or to a Firehose delivery stream, to continuously deliver SQL results to AWS destinations. The limit is three destinations for each application. [Learn more](#)

4.2.4. Real Time Analytics

In the real-time analytics panel, the SQL editor helps you to see samples from your source data stream, get feedback on any errors in your configuration or SQL, watch as you data is processed in real-time by your SQL code.

Amazon Kinesis | Kinesis Data Analytics applications > SteamingAnalytics > SQL Editor

Real-time analytics

[Save and run SQL](#) [Add SQL from templates](#) [Download SQL](#) [SQL reference guide](#) [Kinesis data generator tool](#)

```

1  /**
2  * Welcome to the SQL editor
3  *
4  *
5  * The SQL code you write here will continuously transform your streaming data
6  * when your application is running.
7  *
8  * Get started by clicking "Add SQL from templates" or write your own
9  * documentation and start writing your own custom queries.
10 *
11 */
  
```

Source | Real-time analytics | Destination

Streaming data
 SOURCE_SQL_STREAM_001

Reference data (optional)
[Connect reference data](#)

The streaming data below is a sample of the data currently being processed by this application.

[Filter by column name](#)

VendorID	tpcp_pickup_datetime	tpcp_dropoff_datetime	passenger_count	trip_distance	pickup_location
INTEGER	VARCHAR(16)	VARCHAR(16)	INTEGER	REAL	DOUBLE

[Choose Start application if you would like to see a sample for this source.](#)

[Start application](#)

Would you like to start running "SteamingAnalytics"?

The SQL editor is much more powerful when your application is running.

- See samples from your source data stream
- Get feedback on any errors in your configuration or SQL
- Watch as your data is processed in real-time by your SQL code

[No, I'll do this later](#) [Yes, start application](#)

Amazon Kinesis

Dashboard

Data Streams

Data Firehose

Data Analytics

Video Streams

Kinesis Data Analytics applications > StreamingAnalytics > SQL Editor

Real-time analytics

Save and run SQL Add SQL from templates Download SQL SQL reference guide Kinesis data generator tool

```

1  /**
2  *
3  * Welcome to the SQL editor
4  *
5  * =====
6  * The SQL code you write here will continuously transform your streaming data
7  * when your application is running.
8  *
9  * Get started by clicking "Add SQL from templates" or pull up the
10 * documentation and start writing your own custom queries.
11 */

```

Application status: RUNNING

Source Real-time analytics Destination

Streaming data

SOURCE_SQL_STREAM_001

Reference data (optional)

Connect reference data

The streaming data below is a sample from Kinesis data stream [TaxiData](#)

Actions

Filter by column name

ROWTIME TIMESTAMP	VendorID INTEGER	tpep_pickup_datetime VARCHAR(16)	tpep_dropoff_datetime VARCHAR(16)	passenger_count INTEGER
2021-05-13 10:31:38.513	2	2015/12/4 17:32	2015/12/4 17:58	1
2021-05-13 10:31:38.513	1	2015/12/4 17:32	2015/12/4 17:45	1
2021-05-13 10:31:38.513	2	2015/12/4 17:32	2015/12/4 17:38	2
2021-05-13 10:31:38.513	2	2015/12/4 17:32	2015/12/4 17:58	1
2021-05-13 10:31:38.513	1	2015/12/4 17:32	2015/12/4 17:38	1
2021-05-13 10:31:38.513	1	2015/12/4 17:32	2015/12/4 17:35	3

You can author your own SQL queries or add SQL from templates to easily analyze your source data. The following is an example for real-time analyzing Kinesis data stream "TaxiData".

```

-- Approximate top-K items - Finds the most frequently occurring values in
-- a stream using the Space Saving algorithm.
-- Returns the approximate top-K most frequently
-- occurring values in a specified column over a
-- tumbling window
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ITEM VARCHAR(1024), ITEM_COUNT
DOUBLE);
CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM ITEM, ITEM_COUNT FROM TABLE(TOP_K_ITEMS_TUMBLING(
    CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001"),
    'passenger_count', -- name of column in single quotes
    2, -- number of top items
    60 -- tumbling window size in seconds
))
);

```

The screenshot shows the Amazon Kinesis Real-time analytics console. The top navigation bar includes the AWS logo, region (Ningxia Region), and services menu. The left sidebar shows the navigation menu with options like Dashboard, Data Streams, Data Firehose, and Data Analytics. The main content area is titled 'Real-time analytics' and contains a SQL editor with a query, a 'Save and run SQL' button, and a 'Download SQL' button. Below the editor, the application status is 'RUNNING'. The 'Source' tab is selected, showing 'In-application streams' with 'DESTINATION_SQL_STREAM' selected. The 'Destination' tab is also visible. A table of results is displayed, showing 'ROWTIME', 'ITEM', and 'ITEM_COUNT'.

SQL Query:

```

3 -- occurring values in a specified column over a
4 -- tumbling window
5 CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ITEM VARCHAR(1024), ITEM_COUNT DOUBLE);
6 CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
7 SELECT STREAM ITEM, ITEM_COUNT FROM TABLE(TOP_A.ITENS,TUMBLING
8 CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001"),
9 'passenger_count', -- name of column in single quotes
10 2, -- number of top items
11 60 -- tumbling window size in seconds
12 );
13 ;

```

Application status: RUNNING

Source: Real-time analytics Destination

In-application streams:

- ☒ DESTINATION_SQL_STREAM
- ☐ error_stream

Pause results * New results are added every 2-10 seconds. The results below are sampled. ⓘ

☐ Scroll to bottom when new results arrive.

Filter by column name

ROWTIME	ITEM	ITEM_COUNT
2021-05-13 10:40:54.312	1	14582.0
2021-05-13 10:41:54.312	1	21636.0
2021-05-13 10:41:54.312	2	3671.0

Close

5. Business Intelligence

It is based on the connected mobility demo for the AWS Beijing/Shanghai Summit automotive track. It has been verified in AWS Northeast Virginia region. The original document can be found at [here](#).

6. Firmware-Over-The-Air

Please see the appendix [FOTA](#).

7. Reference

- i <https://aws.amazon.com/iot-core/>
- ii <https://aws.amazon.com/directconnect/>
- iii <https://aws.amazon.com/glue/>
- iv <https://aws.amazon.com/emr/>
- v <https://aws.amazon.com/dynamodb/>
- vi <https://aws.amazon.com/elasticsearch-service/>
- vii <https://aws.amazon.com/athena/>
- viii <https://aws.amazon.com/s3/>
- ix <https://aws.amazon.com/kinesis/data-streams/>
- x <https://aws.amazon.com/kinesis/data-analytics/>
- xi <https://aws.amazon.com/ec2/>
- xii <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
- xiii <https://aws.amazon.com/api-gateway/>
- xiv <https://aws.amazon.com/lambda/>
- xv <https://aws.amazon.com/amplify/>
- xvi <https://aws.amazon.com/lake-formation/>
- xvii <https://aws.amazon.com/redshift/>
- xviii <https://aws.amazon.com/cloudwatch/>