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

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

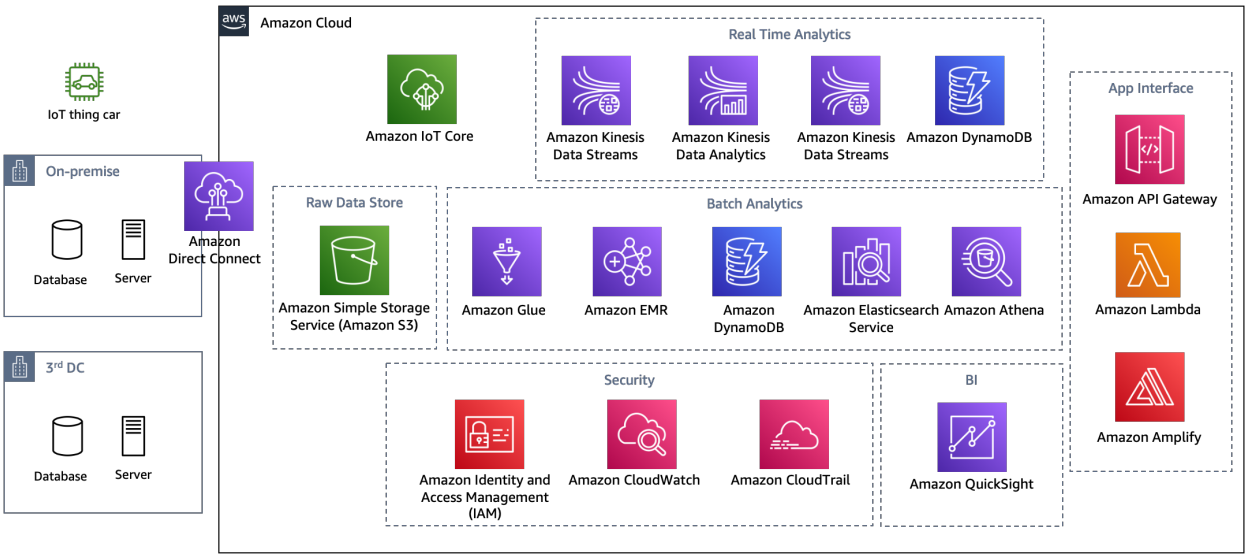
CMS – Data Lake is made up of four parts, which are respectively car fleet with IoT thing car components, on-premise data centers, 3rd party data centers and AWS cloud. Car fleet with IoT thing car components communicate with AWS cloud via AWS IoT Core[i] based on MQTT protocol. On-premise and 3rd party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

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 Glue[iii], 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

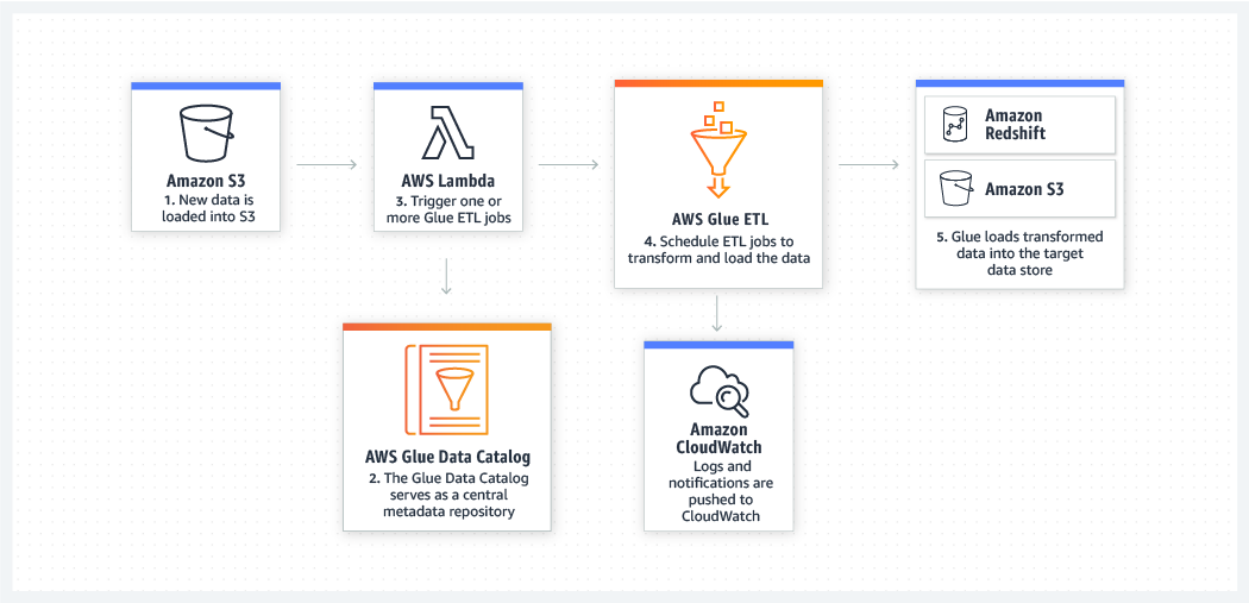
data lake, users can access the data through their choice of analytics services, including Amazon Athena, Amazon Redshift[xvii], and Amazon EMR.

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

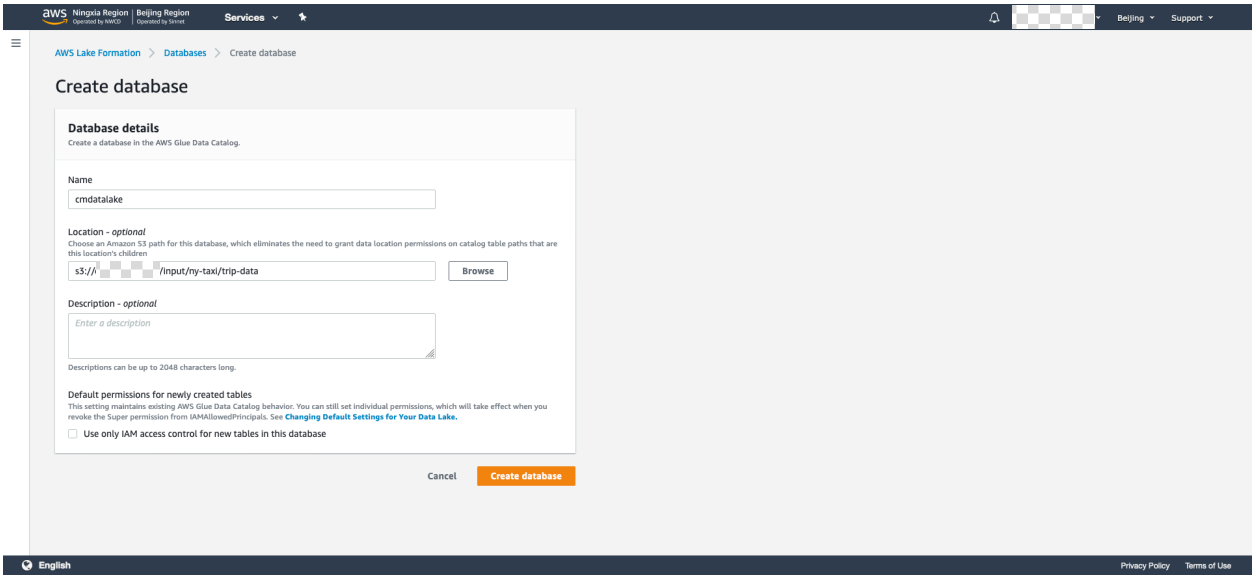
Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run.

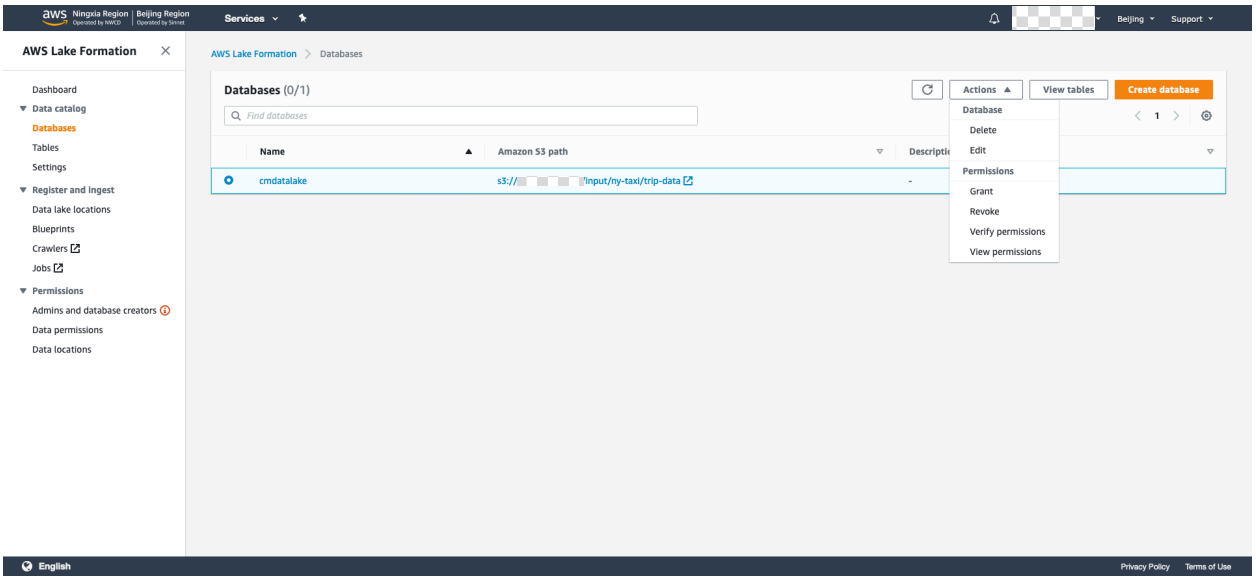
Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there's no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly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

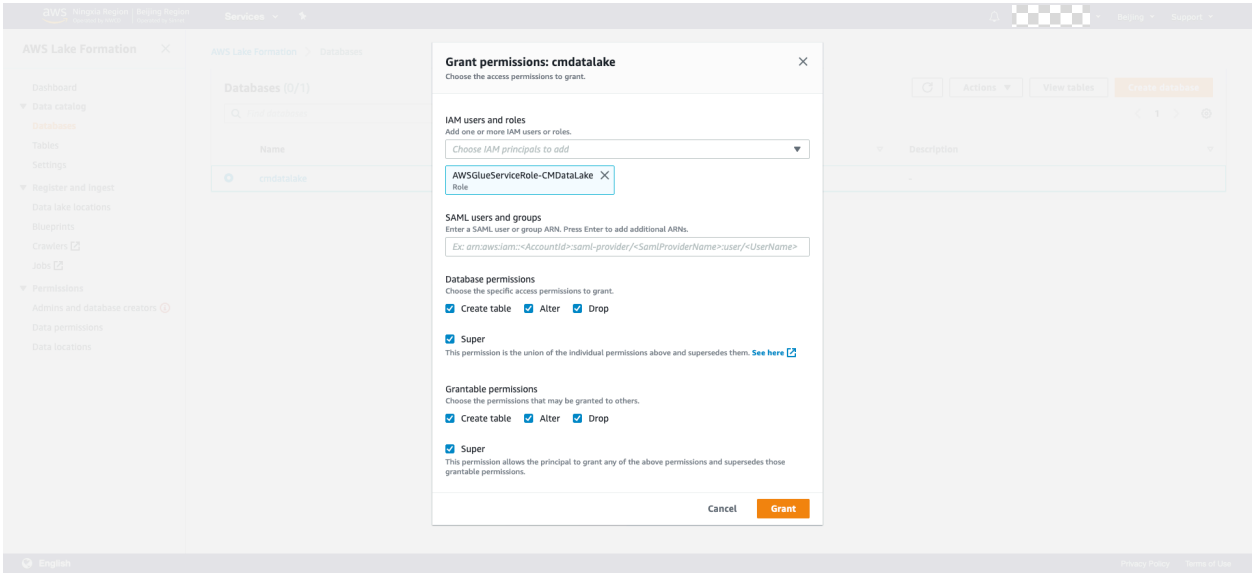
4.1.4. Data Catalog and Query

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.

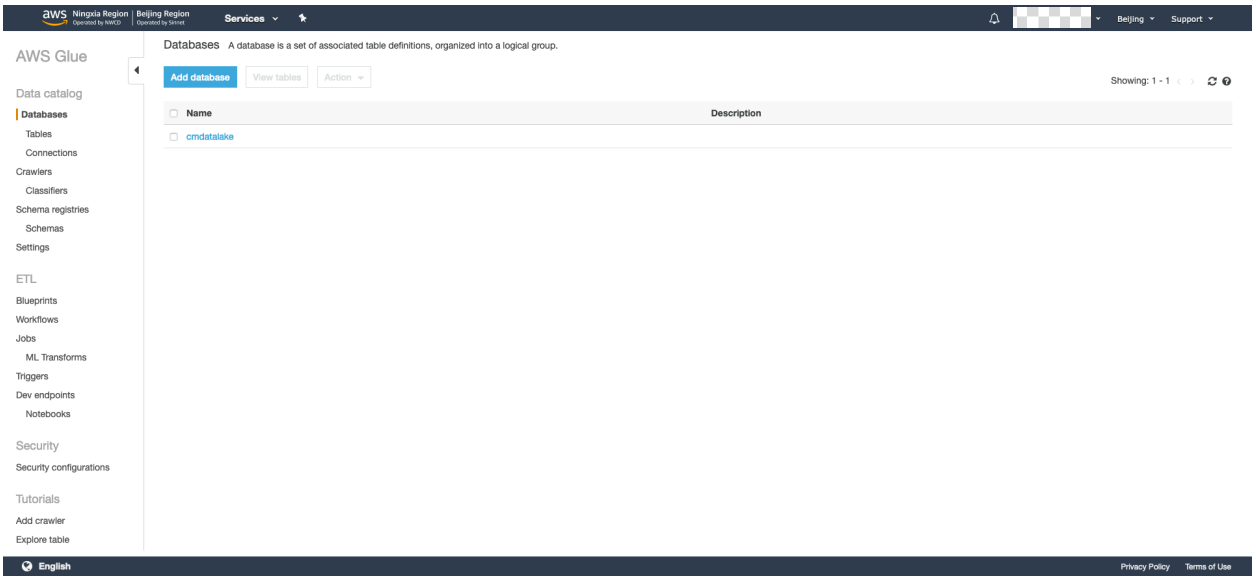




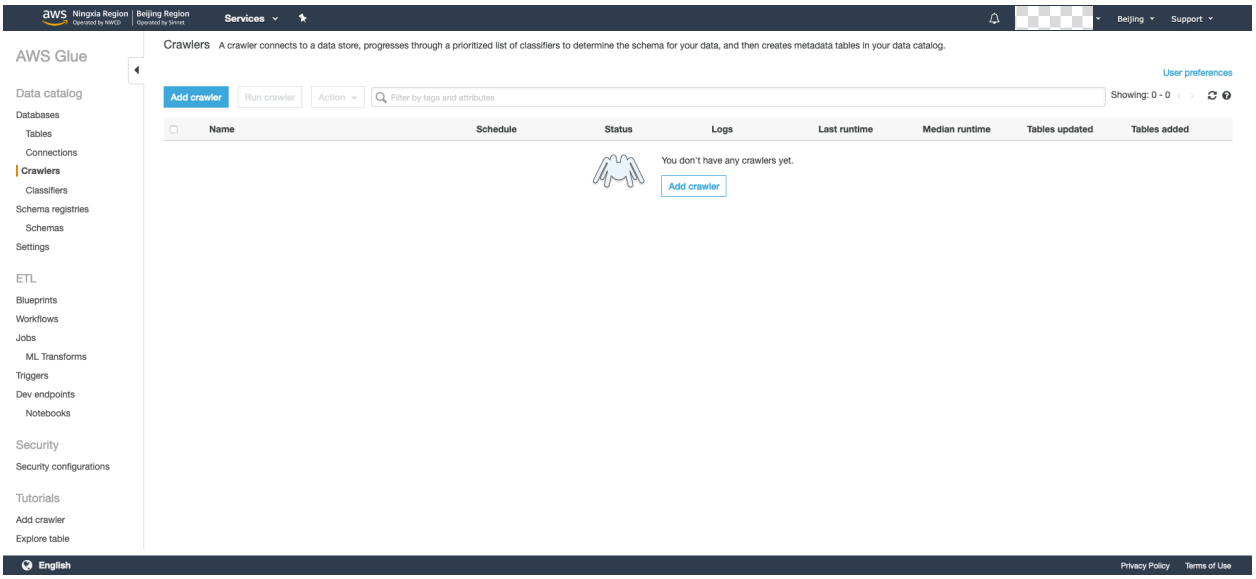
Grant access permissions to your AWS IAM role AWSGlueServiceRole-CMDataLake, which is created by yourself and will be introduced later.

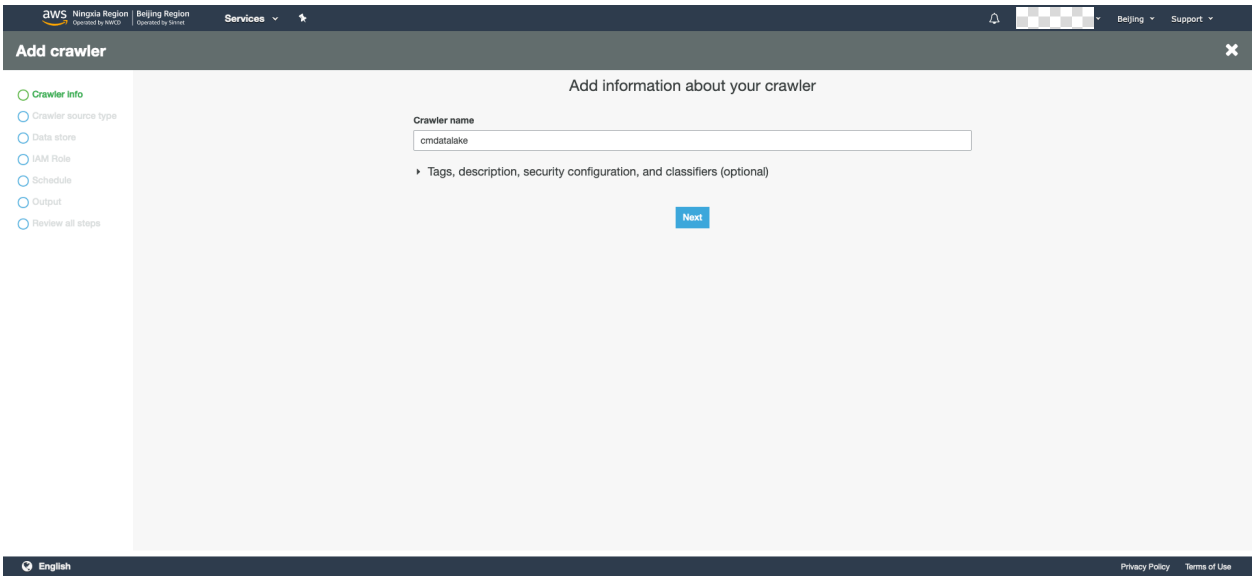


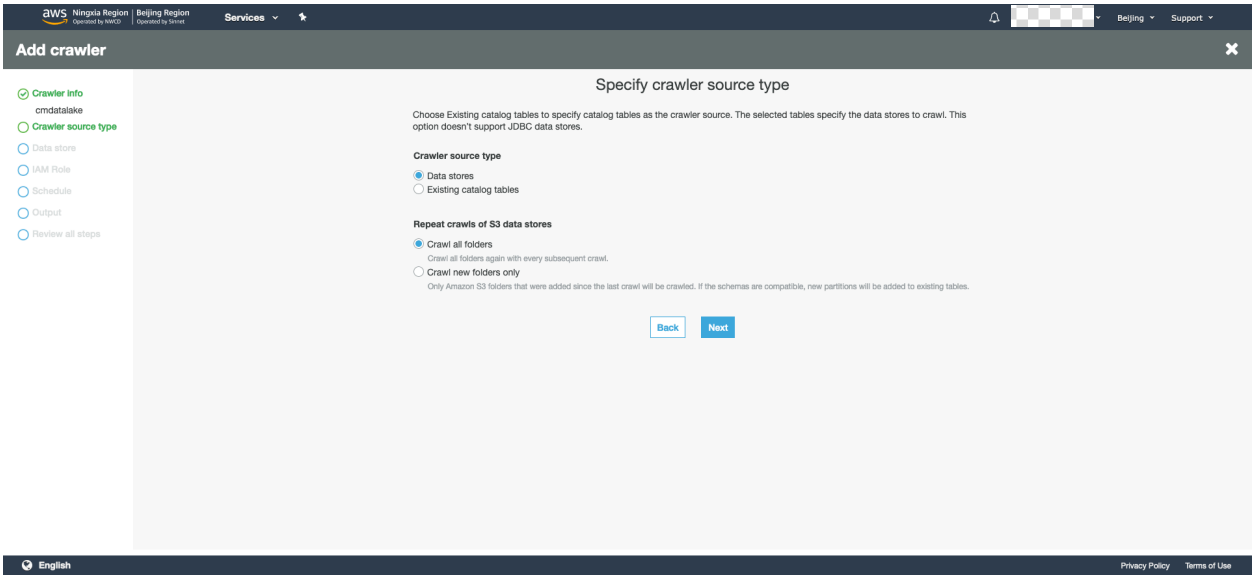
The database created in Lake Formation can also be viewed in the console of AWS Glue.

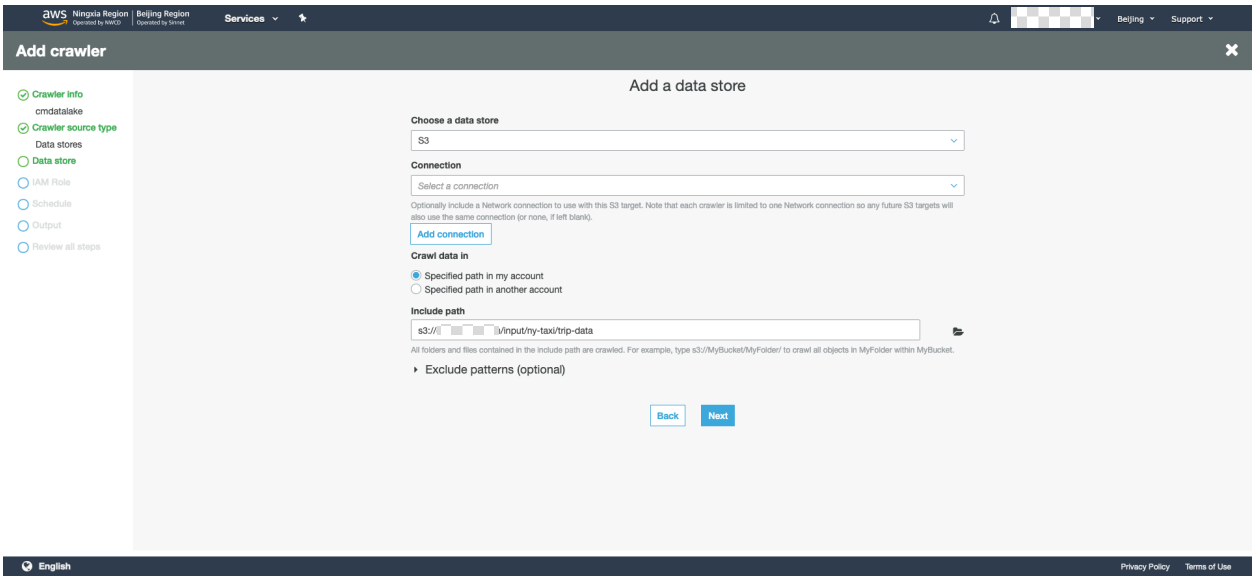


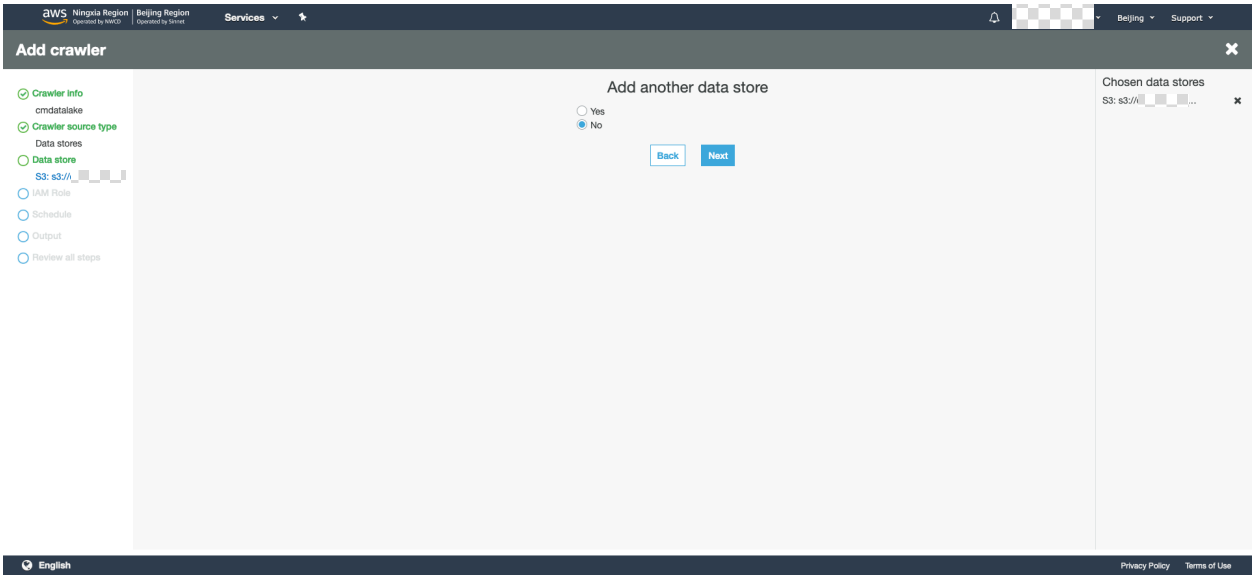
Then a crawler can be created in the console of AWS Glue. The following diagrams illustrate how to create a crawler and craw the data from the CSV files in S3, build relevant data catalogs.



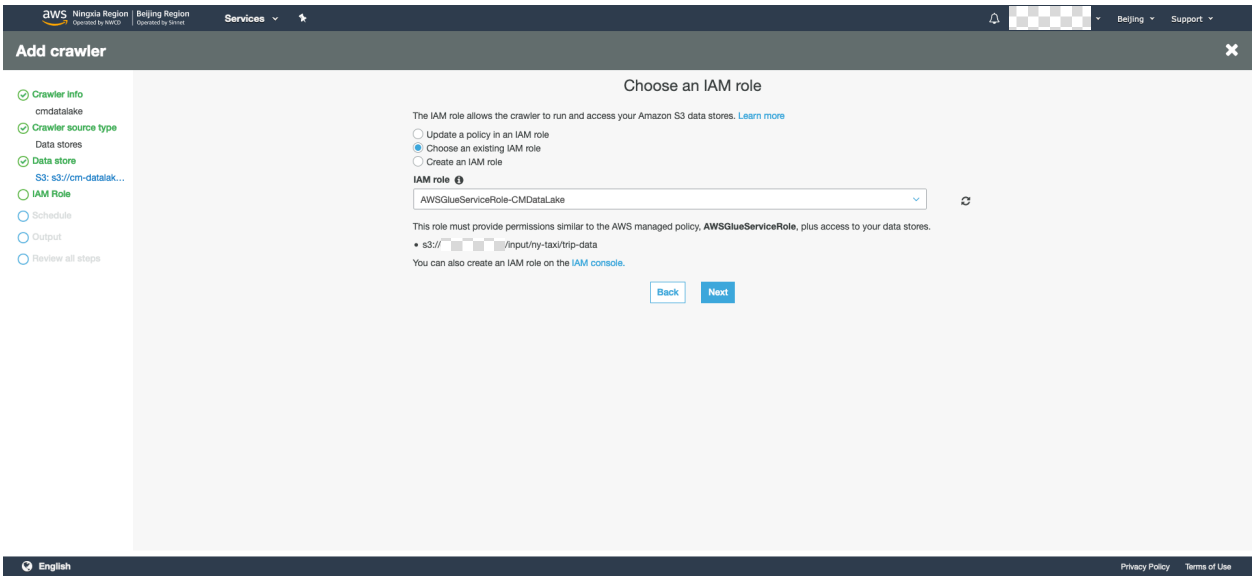

Ningxia Region | Beijing Region
Services
Beijing
Support
Add crawler
Crawler info
Crawler source type
Data store
IAM Role
Schedule
Output
Review all steps
Add information about your crawler
Crawler name
cmdatalake
Tags, description, security configuration, and classifiers (optional)
Next
English
Privacy Policy
Terms of Use

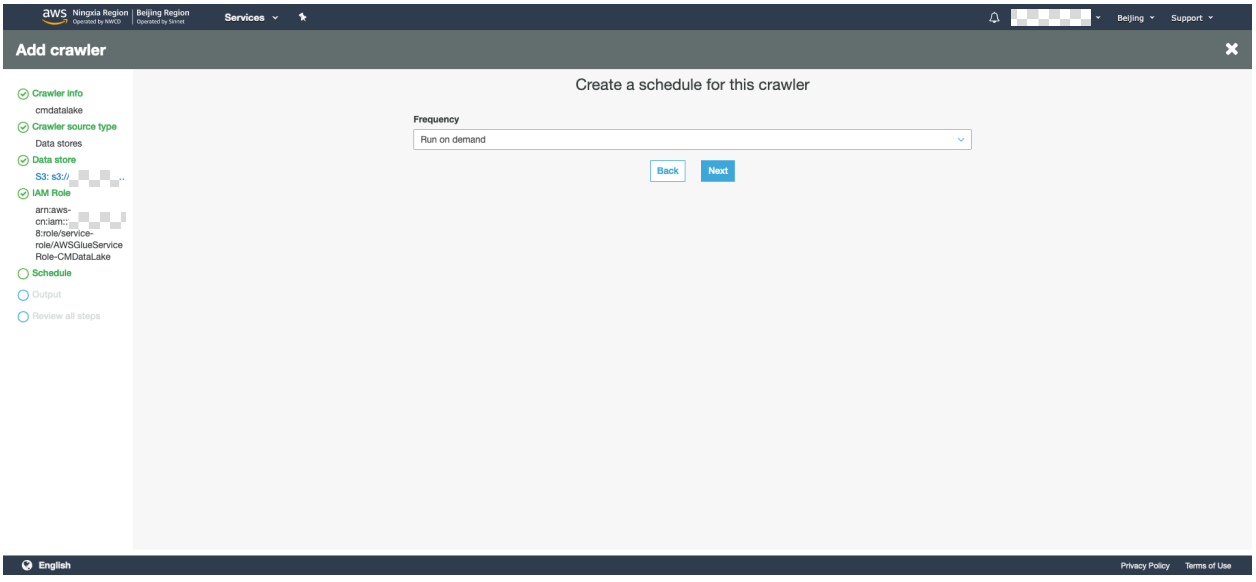

Ningxia Region | Beijing Region
Services
Beijing
Support
Add crawler
Crawler info
cmdatalake
Crawler source type
Data store
IAM Role
Schedule
Output
Review all steps
Specify crawler source type
Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify 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 all folders again with every subsequent crawl.
Crawl new folders only
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



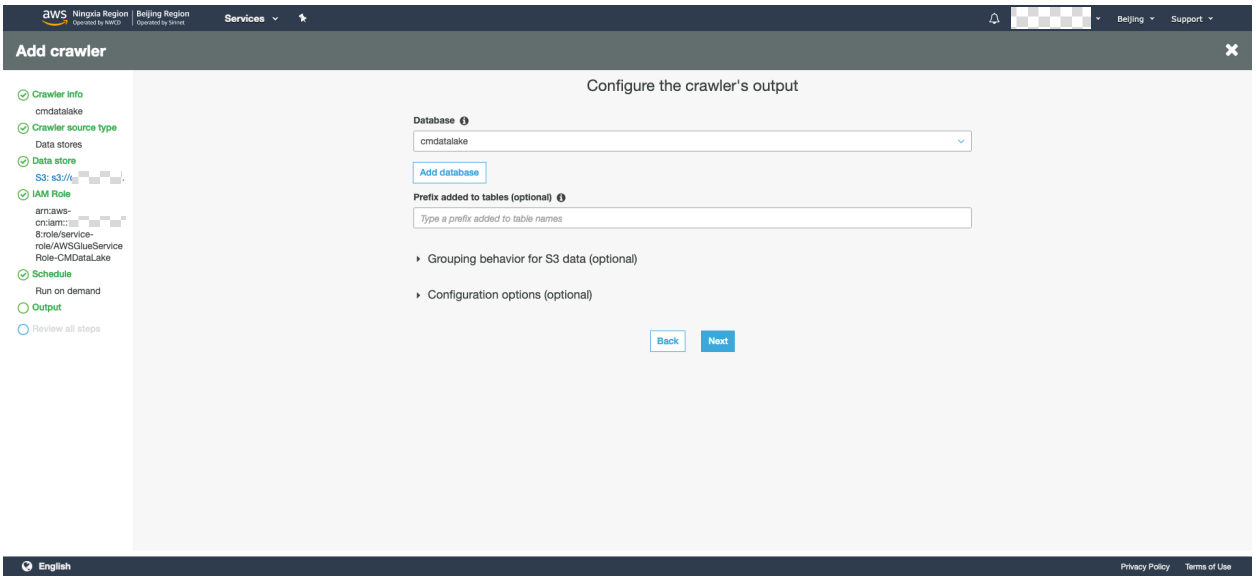


For first time creation, choose the option “Create an IAM role”, otherwise choose the option “Choose an existing IAM role” to specify the role that you have ever created. Here the solution uses the IAM role “AWSGlueServiceRole-CMDataLake” which has already been created ealier.

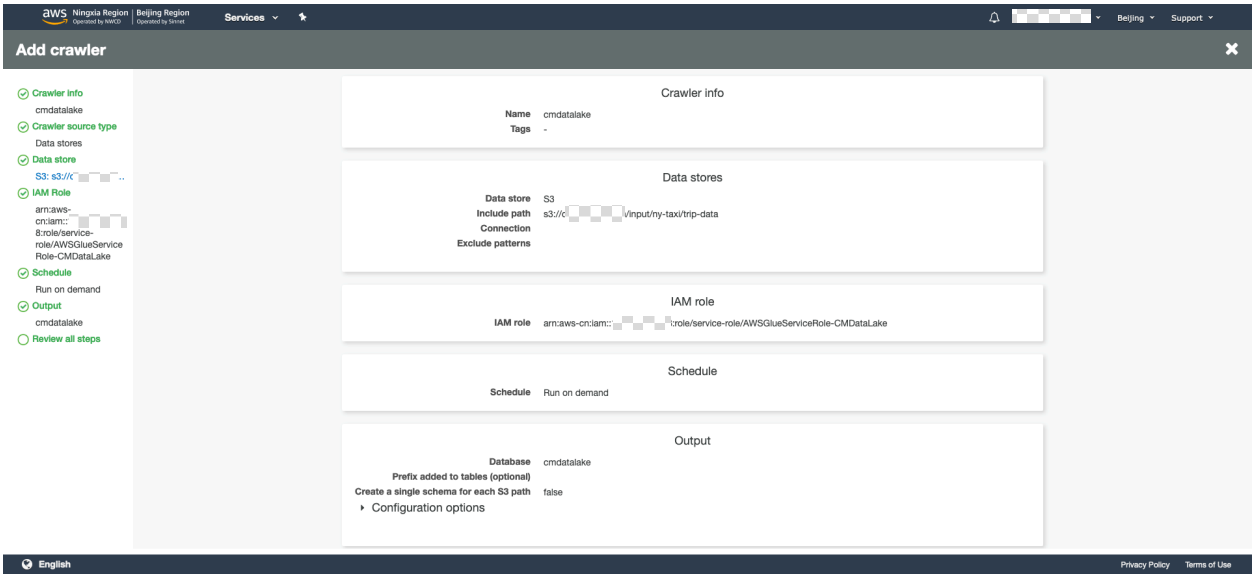




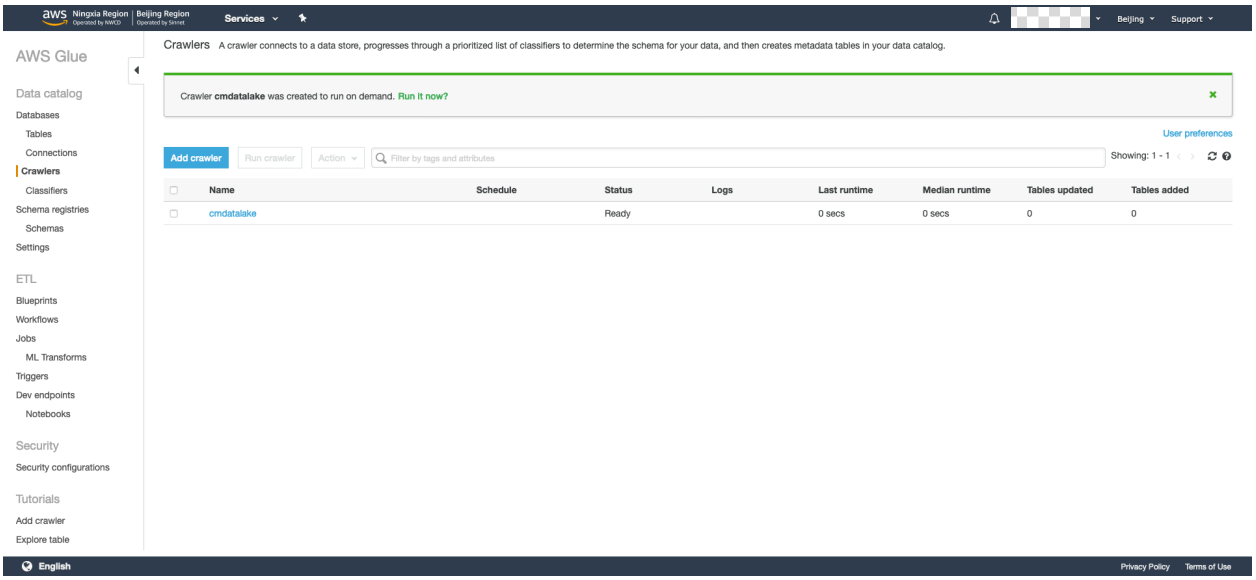
Specify the database that you have created in Lake Formation.



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

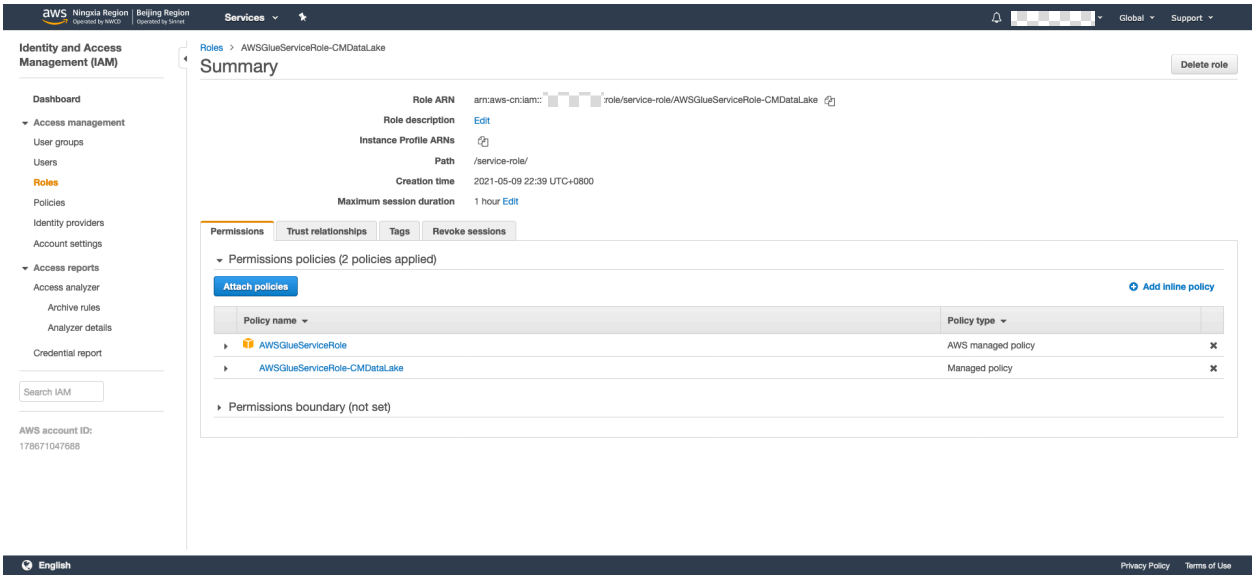


Click the “Finish” button to create a crawler.

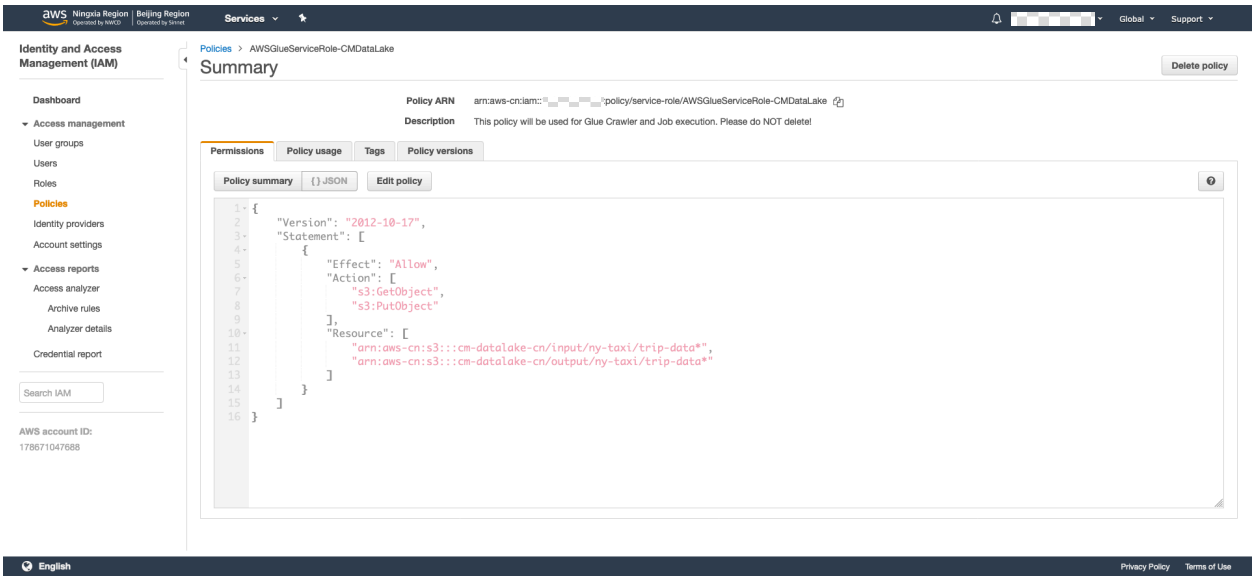


Click the hyper link “Run it now” to launch the crawler.

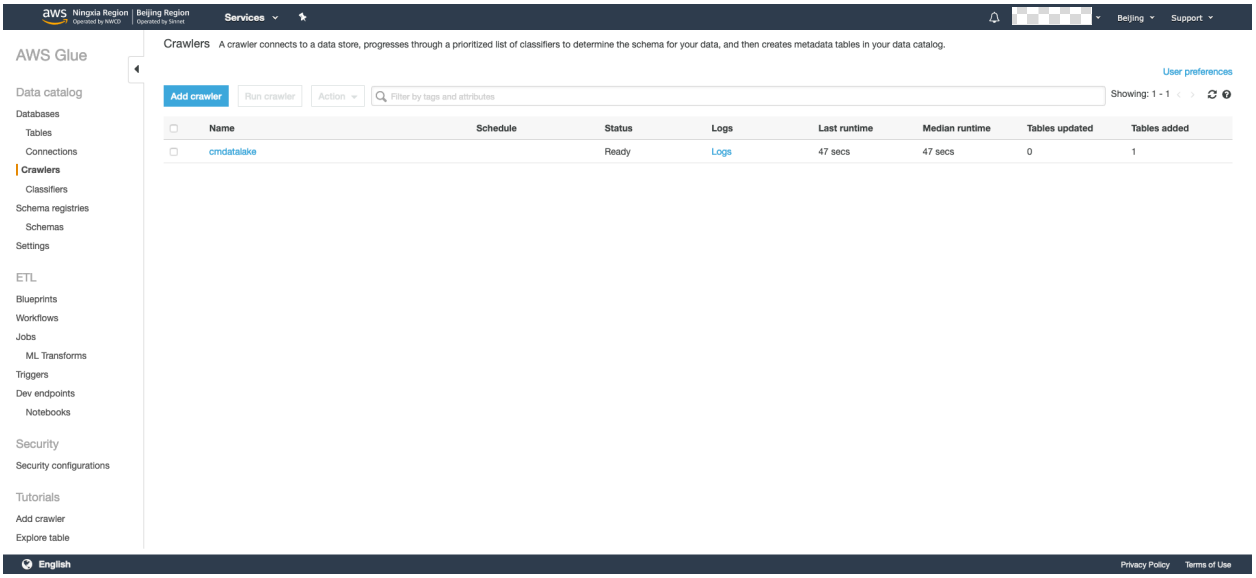
You can go to the IAM console to check the information of the IAM role “AWSGlueServiceRole-CMDataLake”.



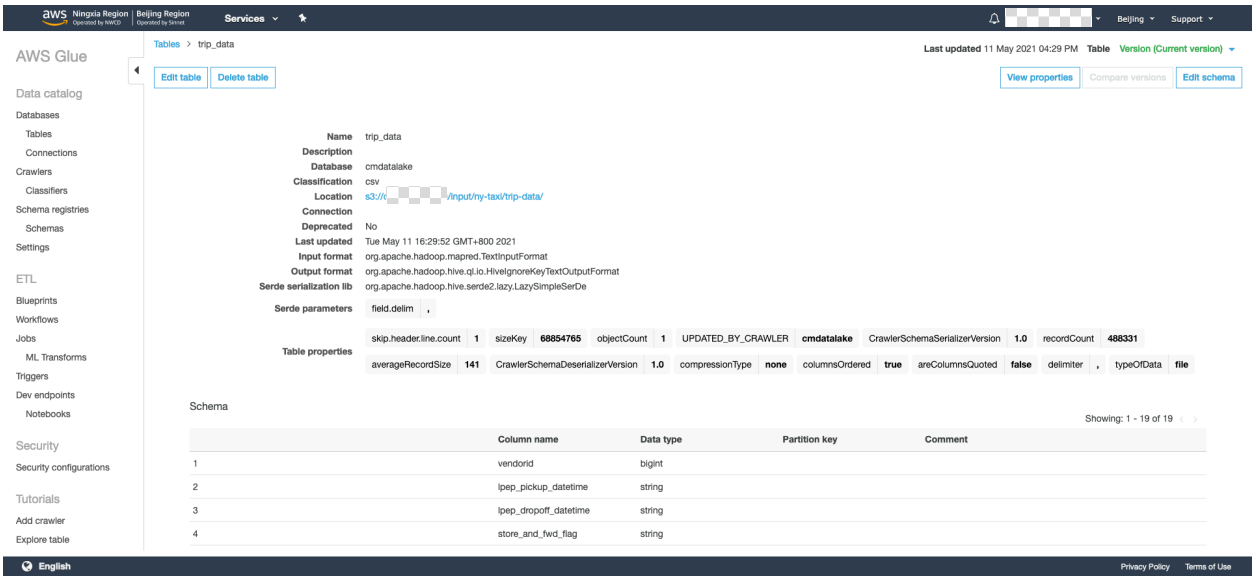
The detailed information for the IAM role “AWSGlueServiceRole-CMDataLake” is as following. Please be noted that the bucket “arn:aws-cn:s3:::[your-bucket-name]/input/ny-taxi/trip-data*” and “arn:aws-cn:s3:::[your-bucket-name]/output/ny-taxi/trip-data*” specify you can only access these URLs in s3.

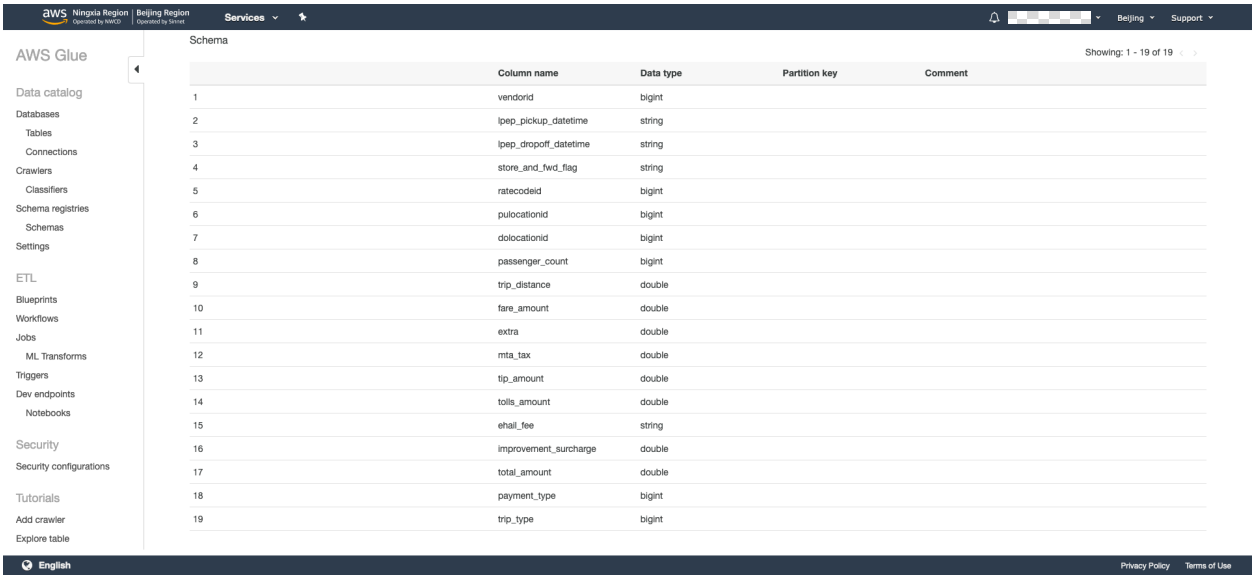


The crawler completed its task and created a table.

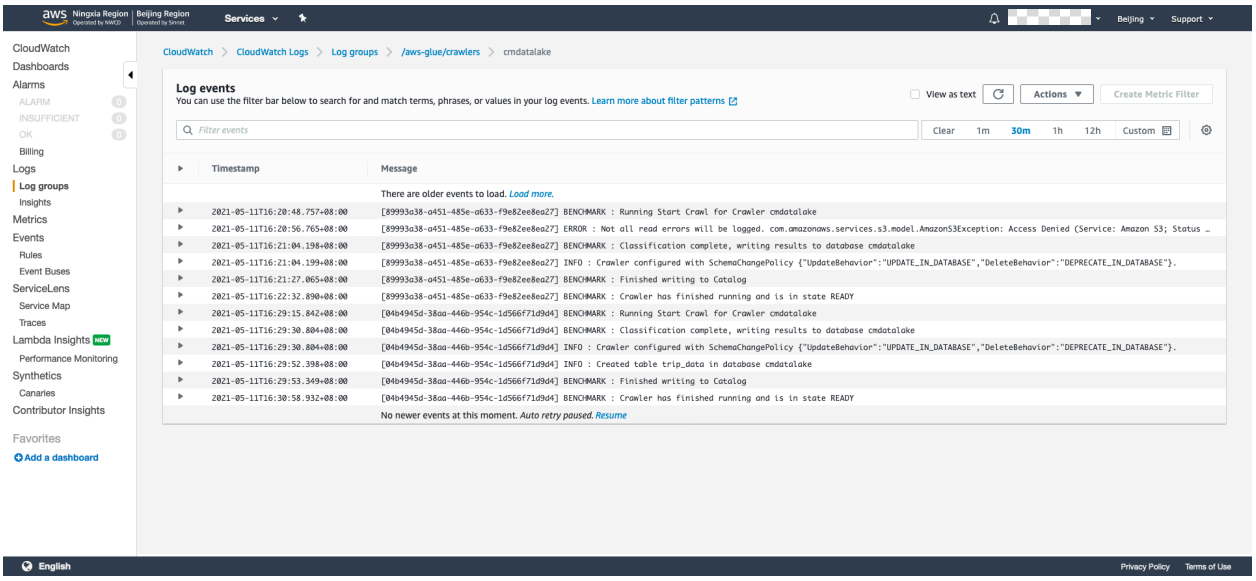


You can view the detailed information of the table as following.

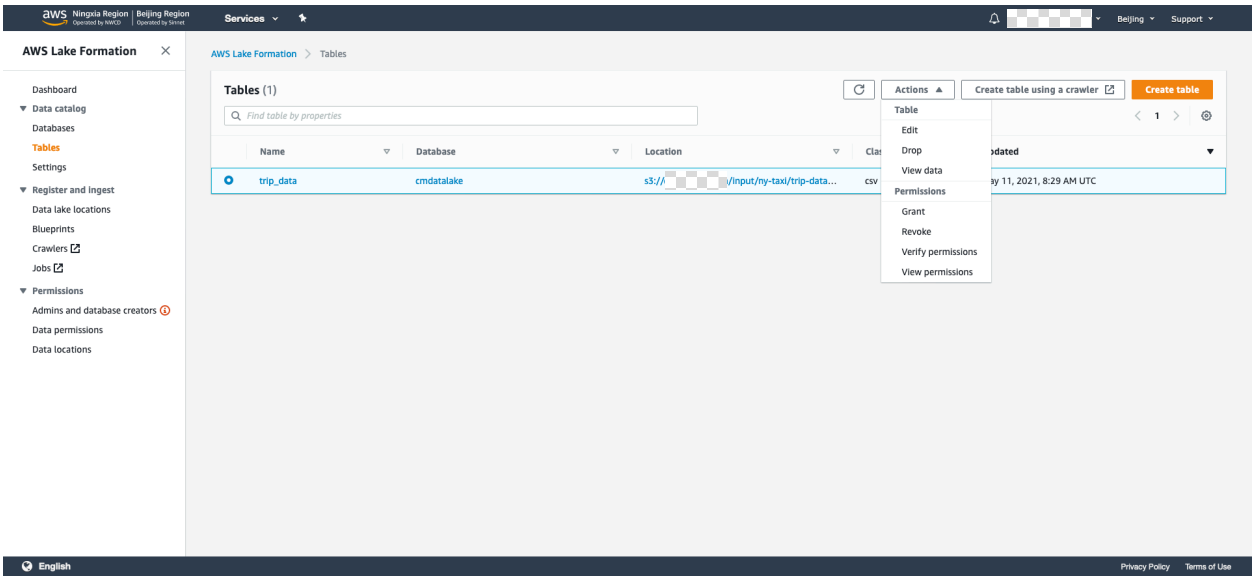




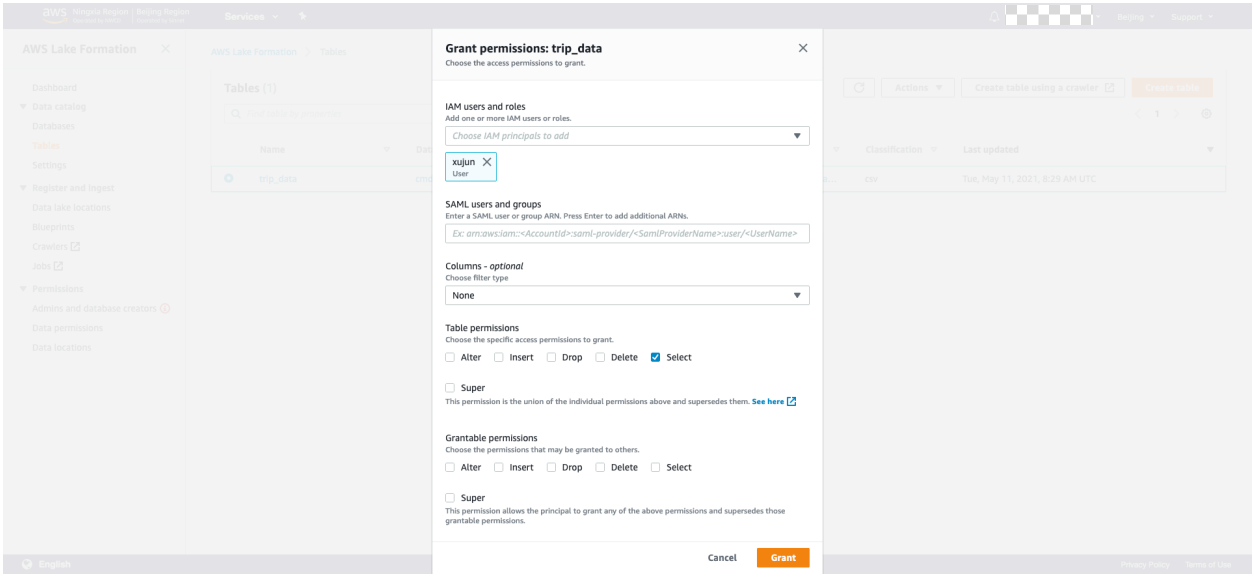
You can also view the logs in Amazon CloudWatch[xviii] for troubleshooting and debugging if you do not get the expected result.



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.

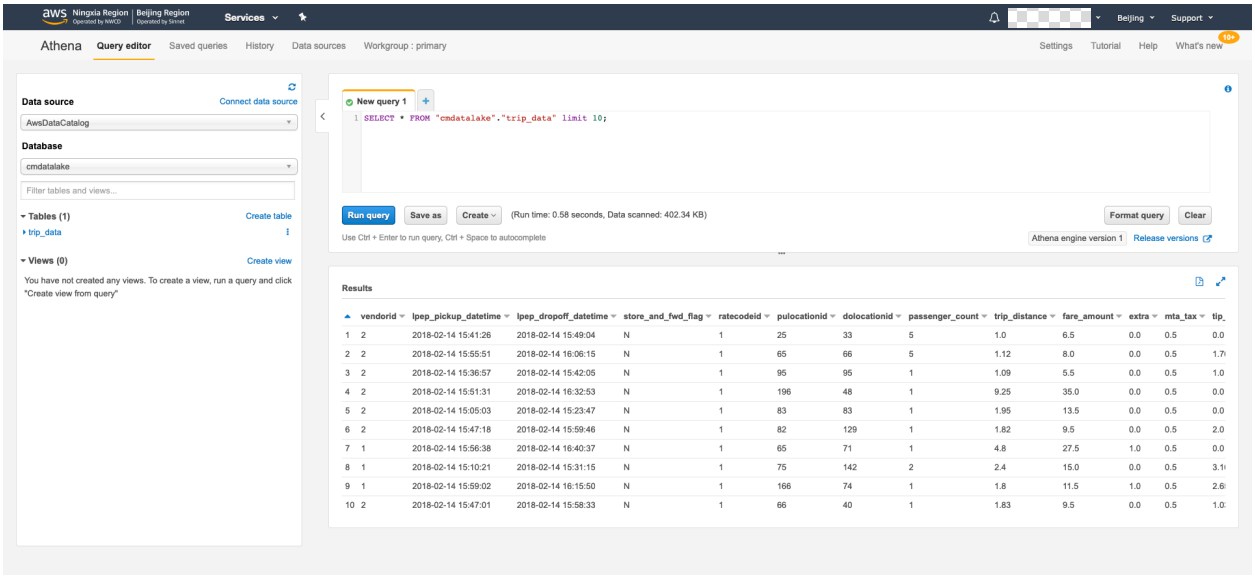


Here the solution logged as an IAM user “xujun”, just grant “Select” permissions to it.



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 the the console of Amazon Athena, select the database “cmdatalake”, then you can query data from the editor window.



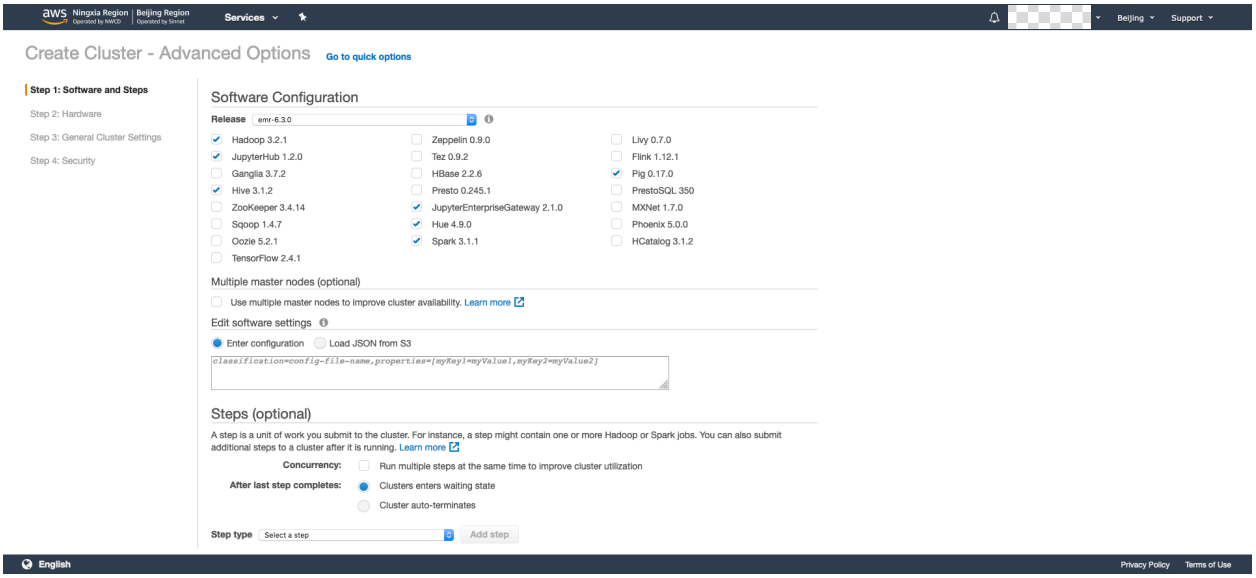
4.1.5. Amazon EMR

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

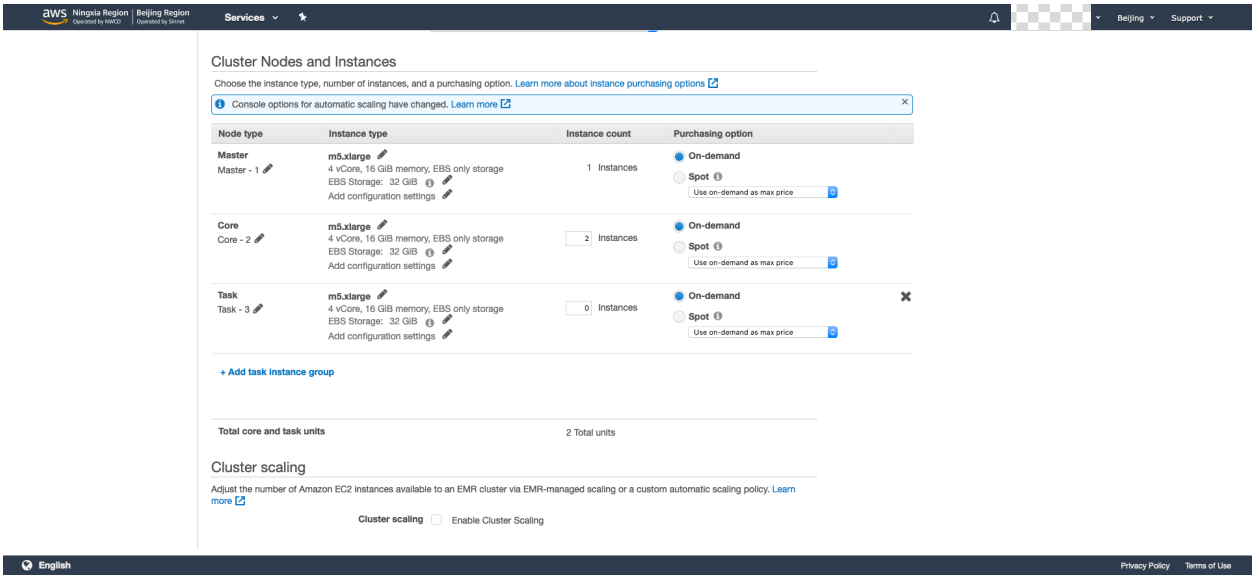
customize your business logic in the Spark job in AWS Glue. A more general method is to use Amazon EMR.

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

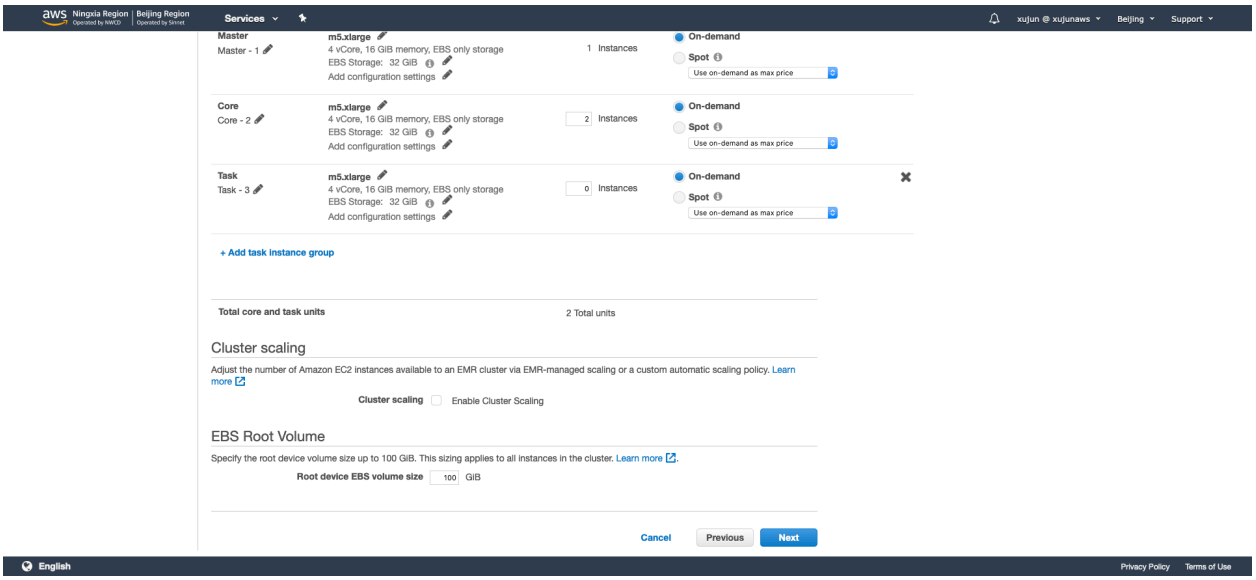
In the console of Amazon EMR, choose advanced options and specify the latest EMR release edition.



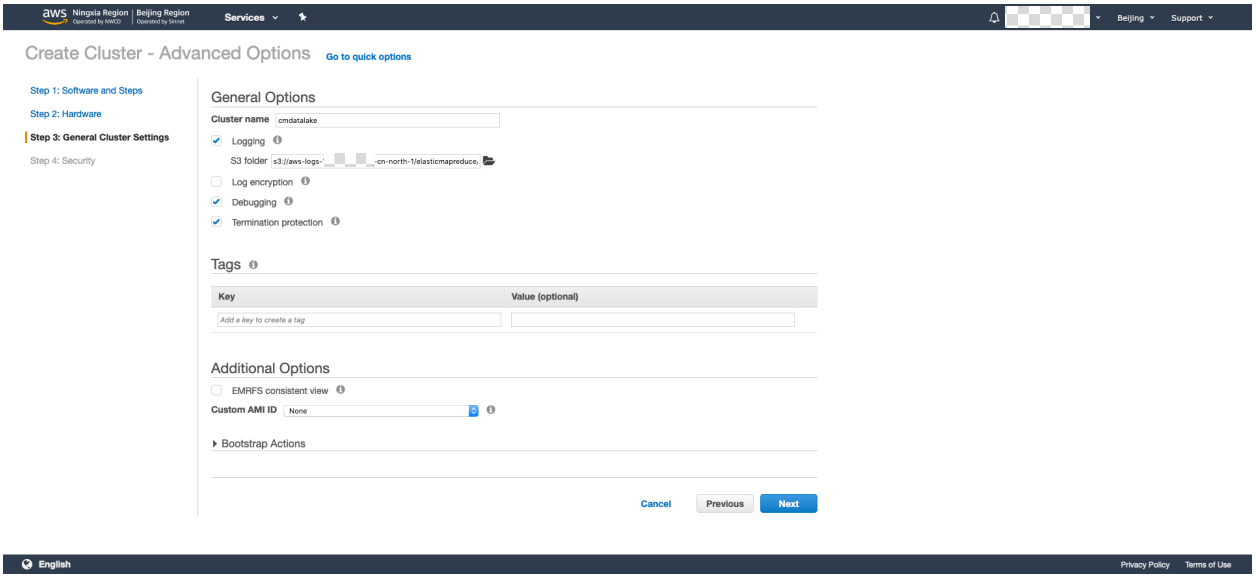
Set the options in Cluster Nodes and Instances page as following.



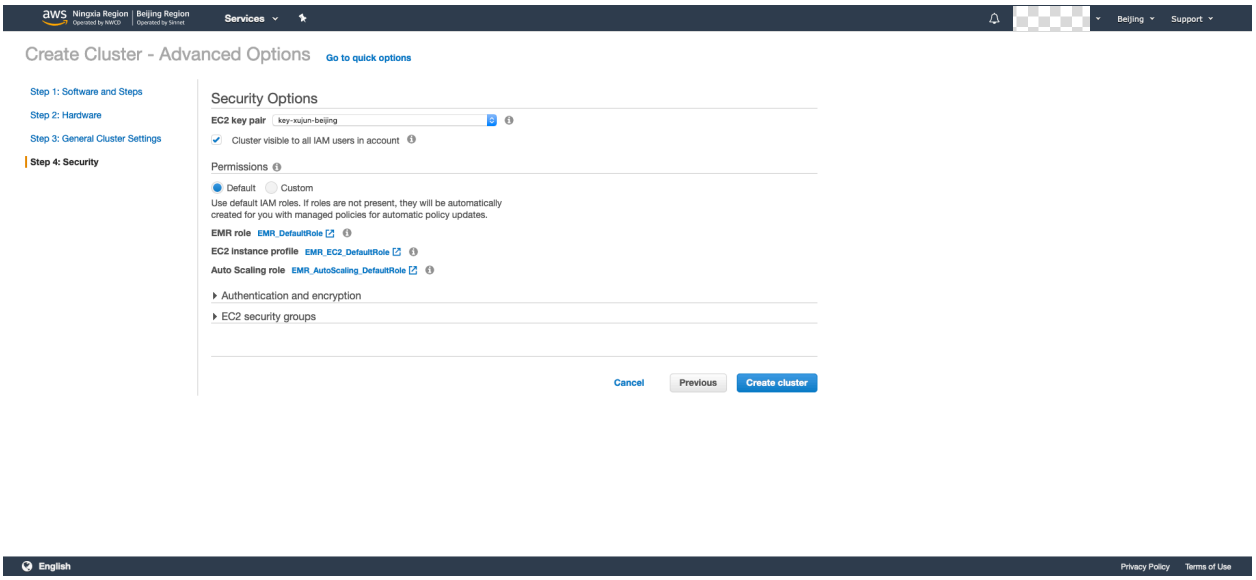
Please set the EBS root volume to 100GB for your data storage capacity.



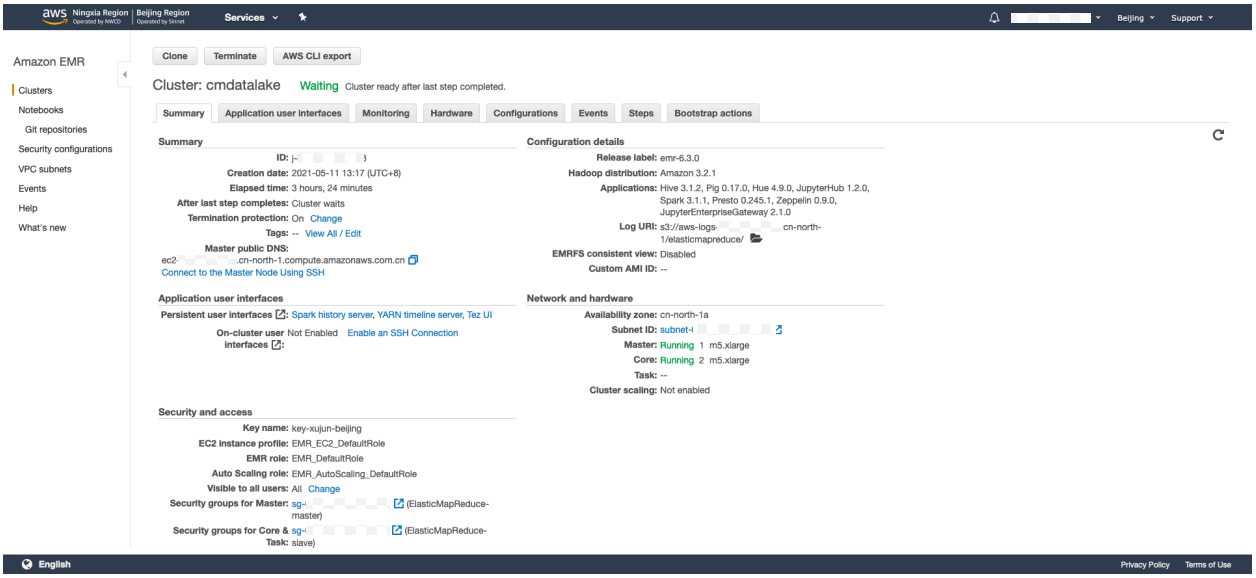
Set the S3 folder for logging.



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.



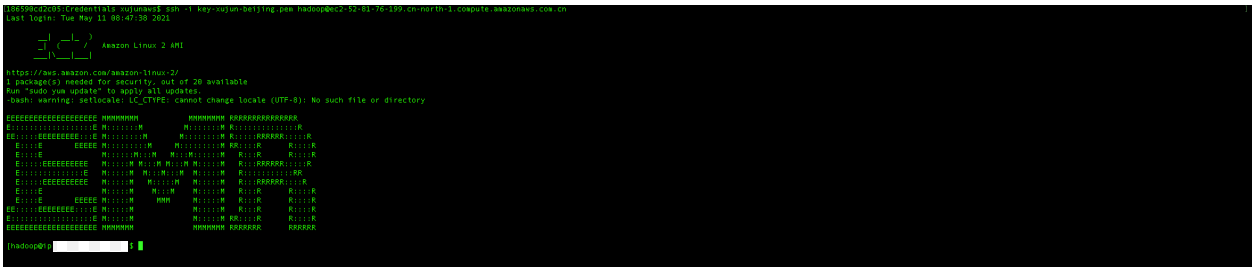
The detailed information for the EMR is as following.



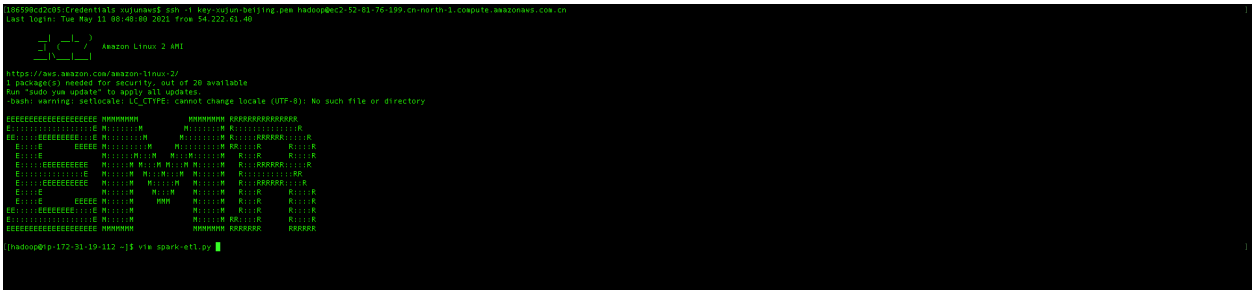
4.1.6. Run Spark Job in Amazon EMR Master Node

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
```



Create a Python file named “spark-etl.py”.



Copy the following code into the Python file.

```
import sys
from datetime import datetime

from pyspark.sql import SparkSession
from pyspark.sql.functions import *

if __name__ == "__main__":

    print(len(sys.argv))
    if (len(sys.argv) != 3):
        print("Usage: spark-etl [input-folder] [output-folder]")
        sys.exit(0)

    spark = SparkSession\
        .builder\
        .appName("SparkETL")\
        .getOrCreate()

    nyTaxi = spark.read.option("inferSchema", "true").option("header",
"true").csv(sys.argv[1])

    updatedNYTaxi = nyTaxi.withColumn("current_date", lit(datetime.now()))

    updatedNYTaxi.printSchema()
```

```
    print(updatedNYTaxi.show())

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

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

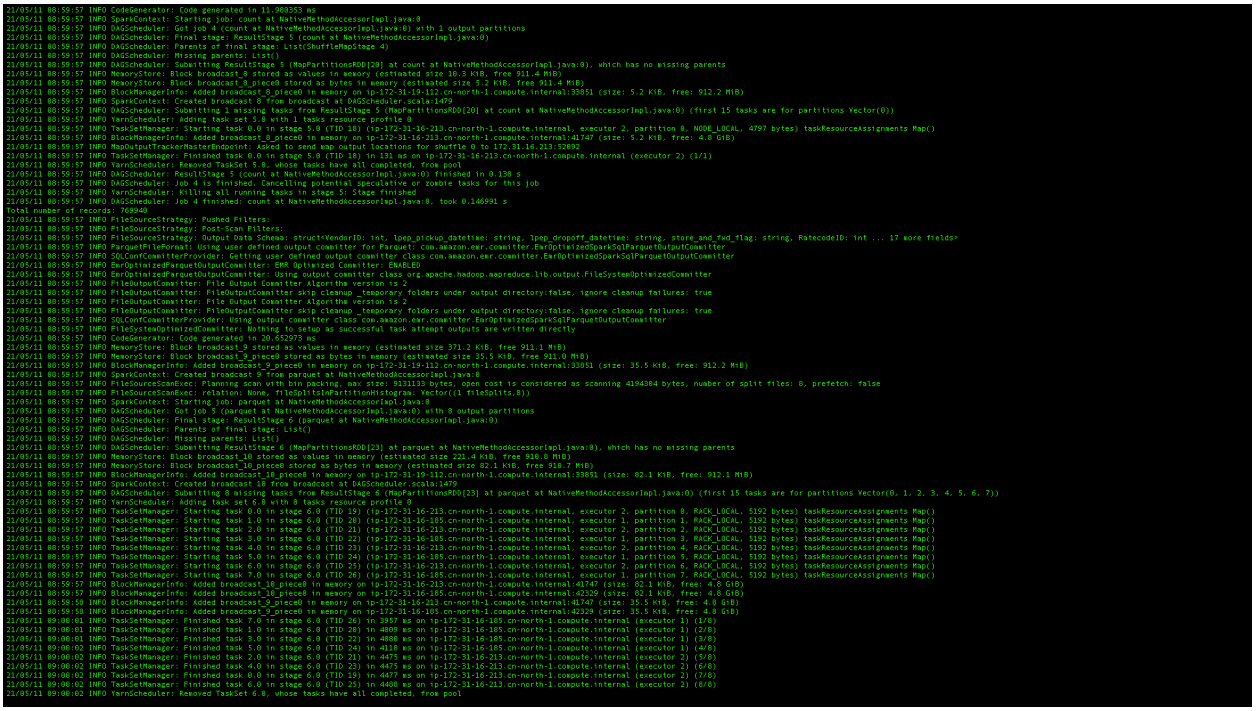
Or download the file from your S3 bucket.

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

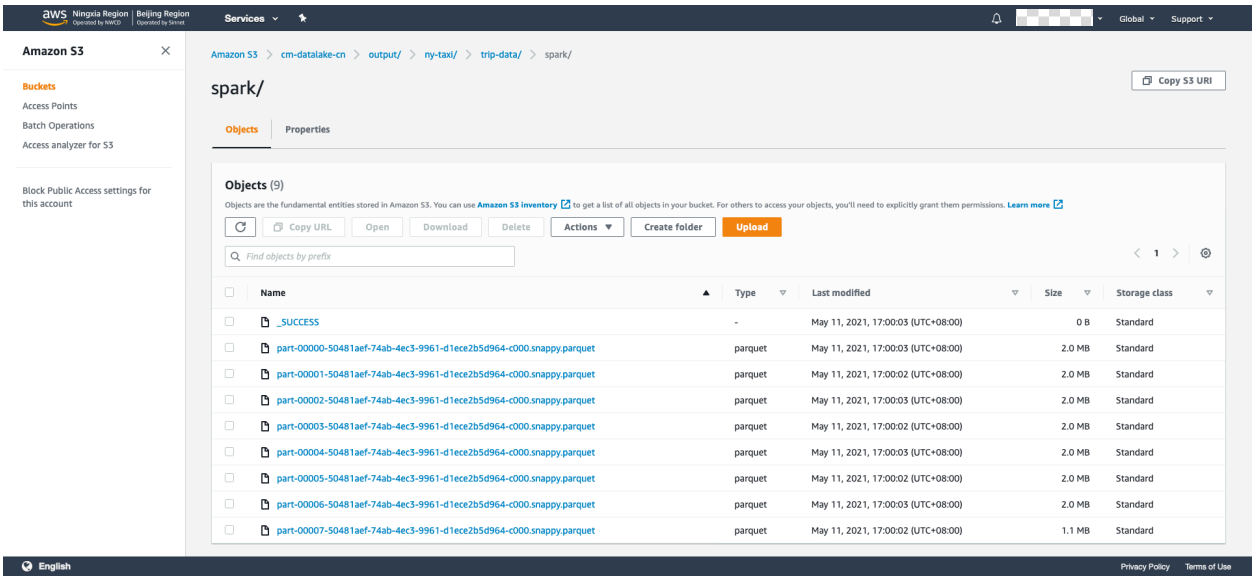
Submit your Spark ETL job.

```
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.

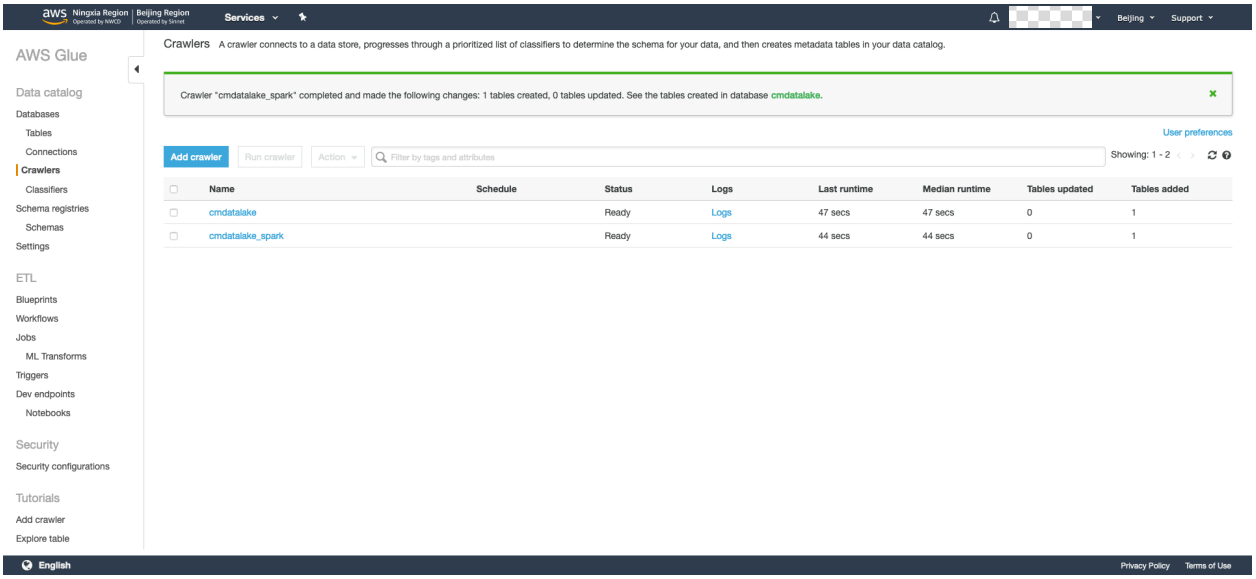


On the console of Amazon S3, you can see the file has been transformed from CSV to parquet format, the total file size decrease from approximately 65 MB to the 15MB in total.

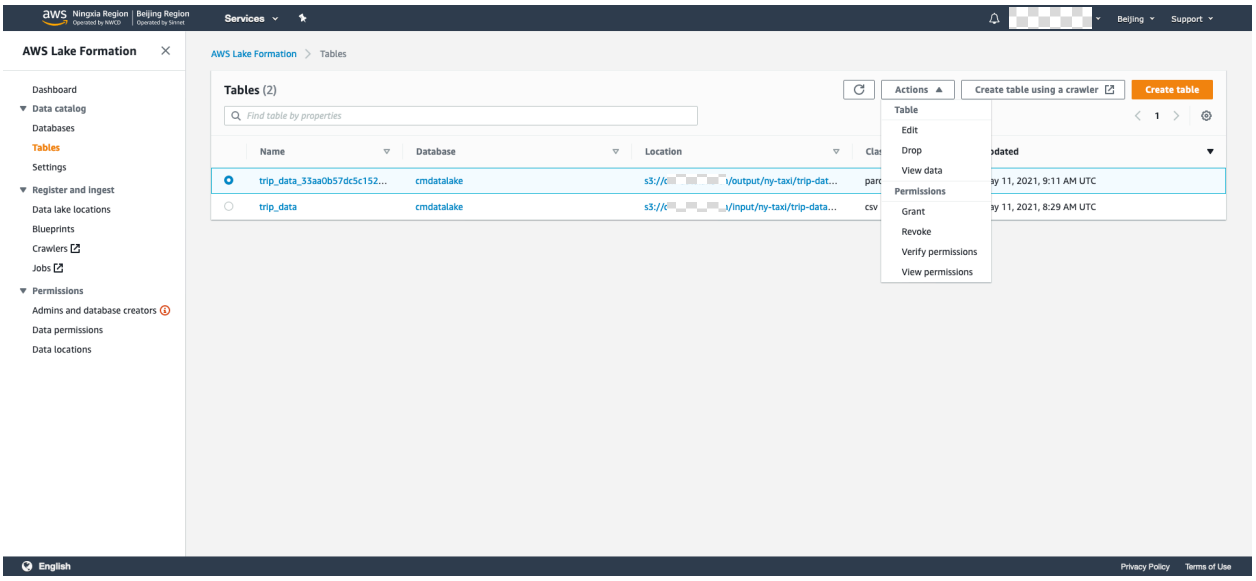


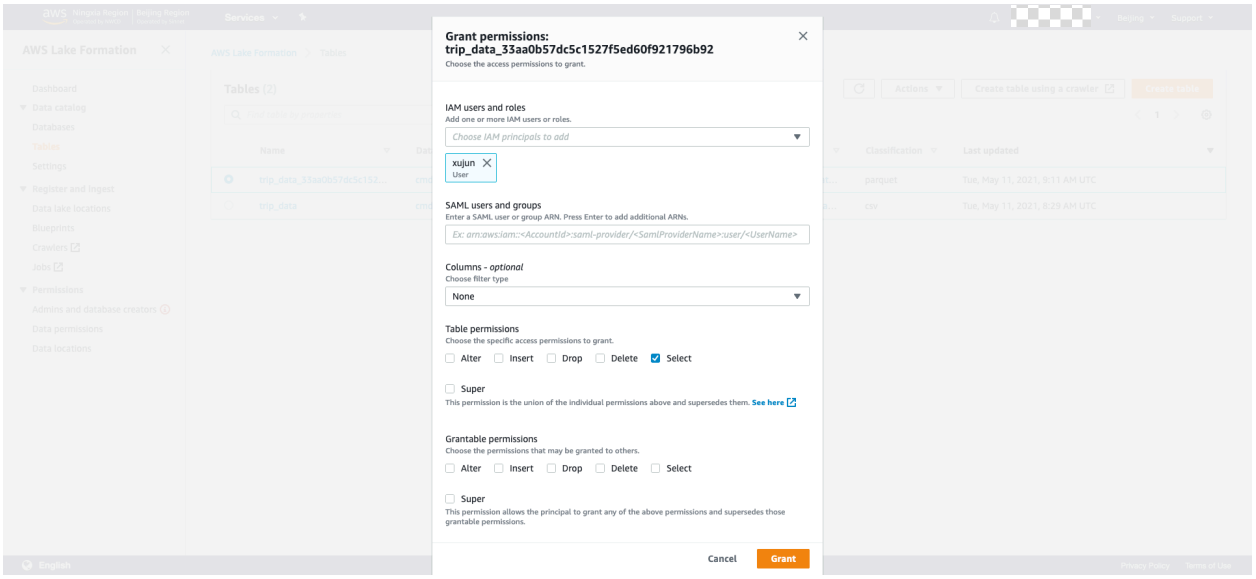
Create another crawler named “cmdatalake_spark” to craw the data output by above Spark ETL job, which is located in the URL “s3://[your-s3-bucket]/output/ny-taxi/trip-data/spark”.

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

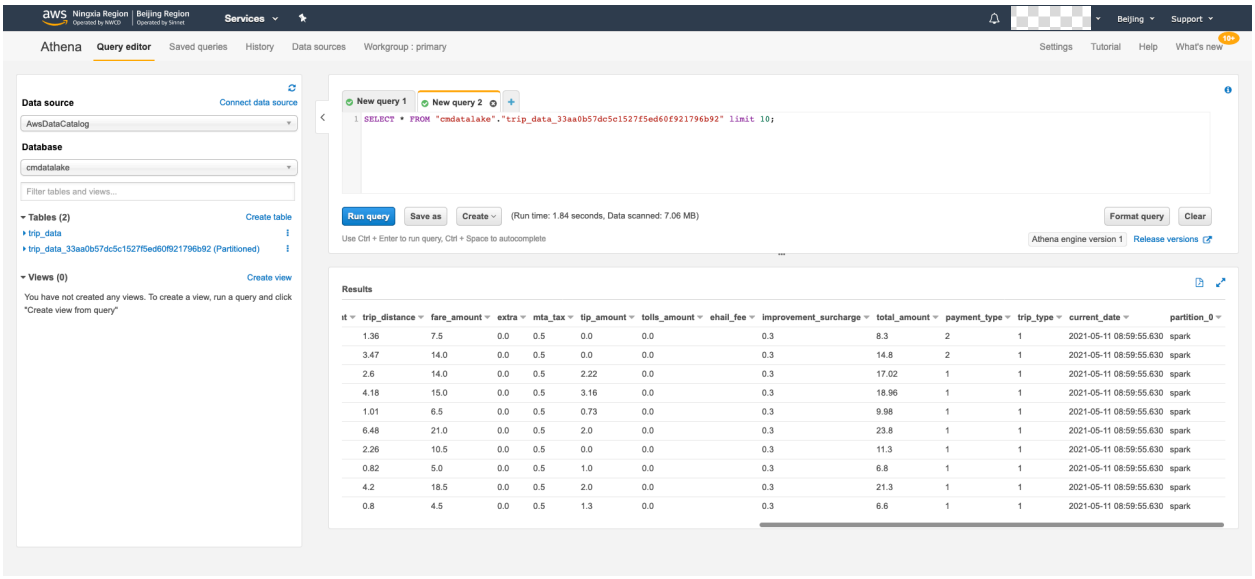


Grant access permissions for the newly created table.





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

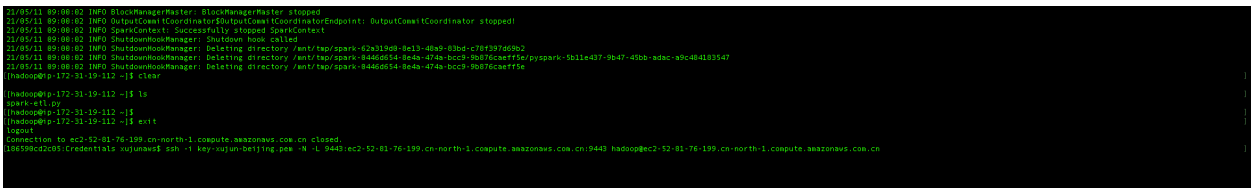
Open another command line terminal on your laptop and launch the following SSH tunneling command.

```
# 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.

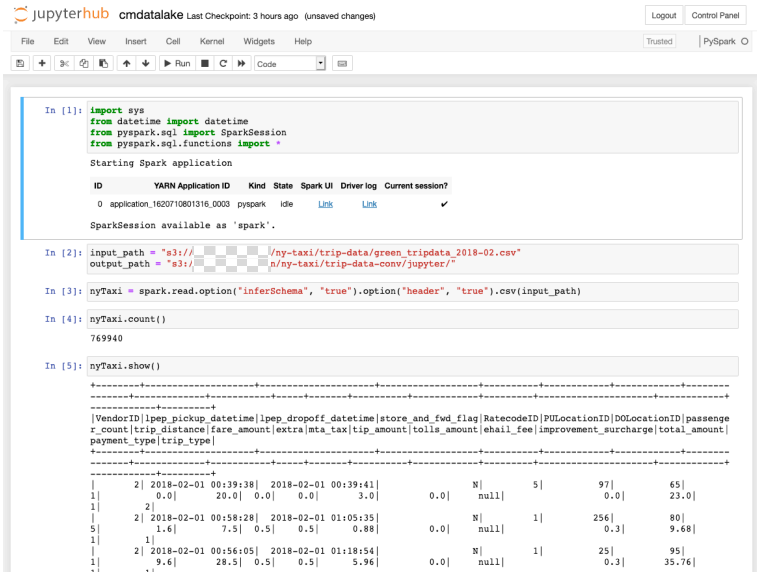
Please login via the default credentials listed as following:

- Username: jovyan
- Password: jupyter



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.

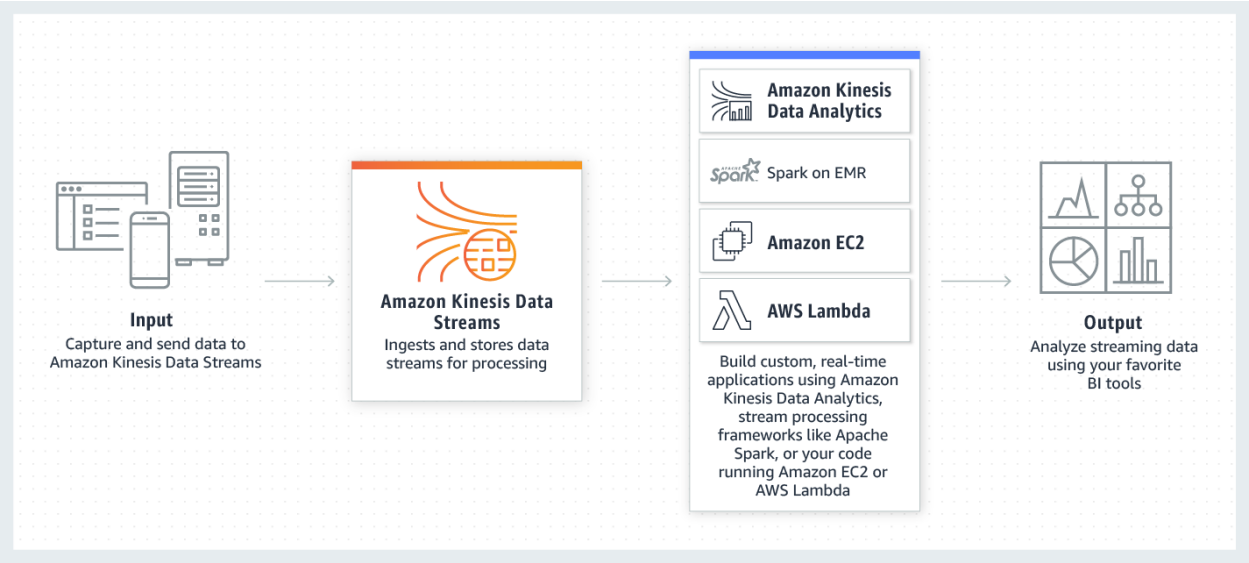
4.2. Streaming Mode

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

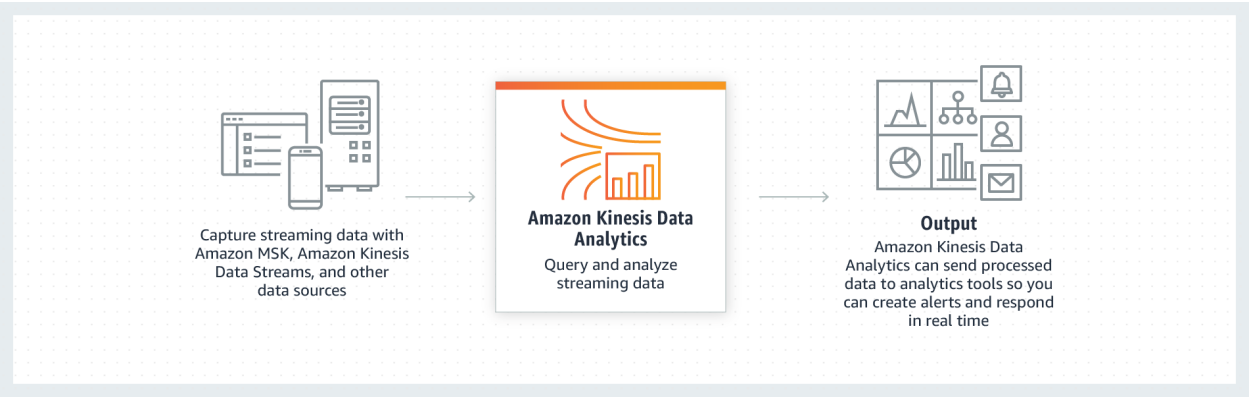
available in milliseconds to enable real-time analytics use cases such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more.



4.2.2. Amazon Kinesis Data Analytics

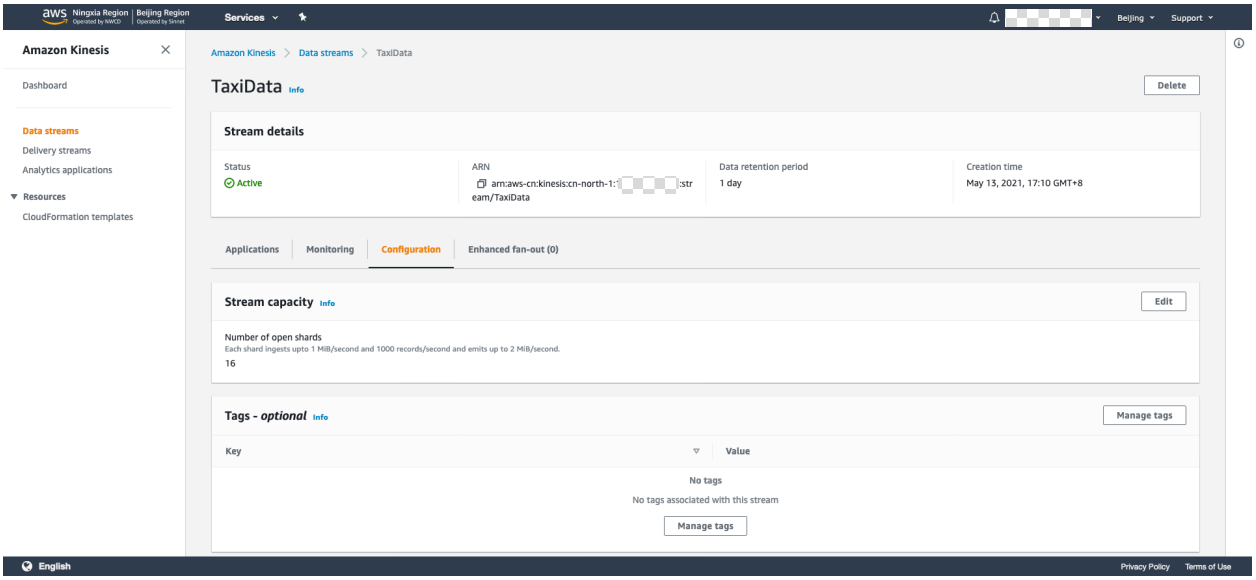
Amazon Kinesis Data Analytics is the easiest way to transform and analyze streaming data in real time with Apache Flink. Apache Flink is an open source framework and engine for processing data streams. Amazon Kinesis Data Analytics reduces the complexity of building, managing, and integrating Apache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

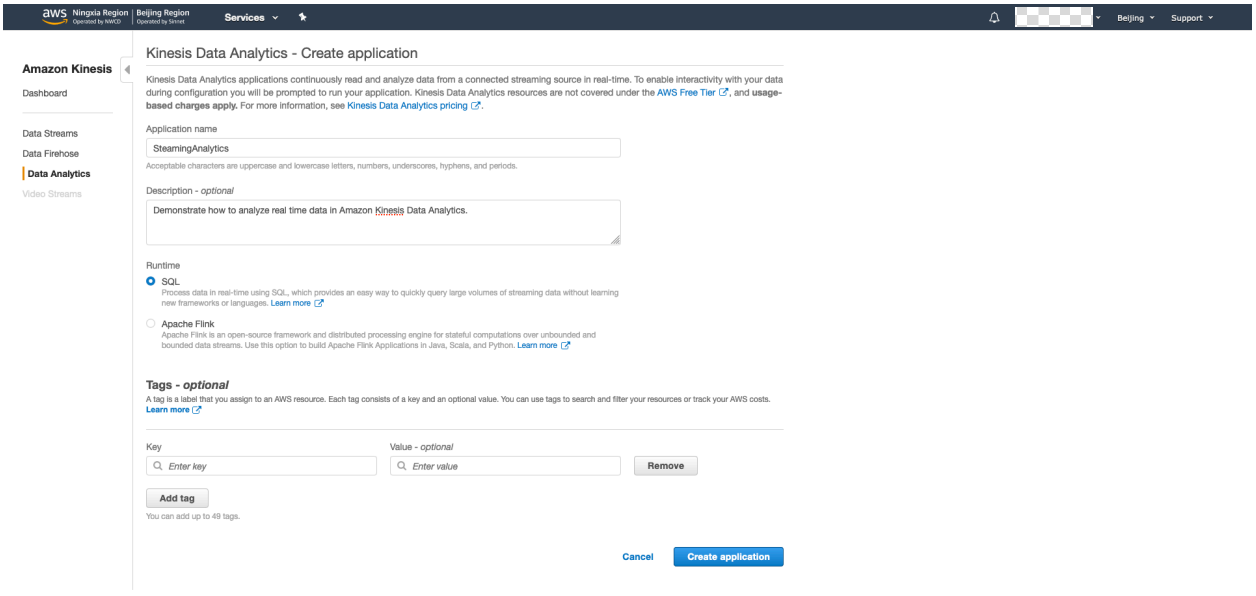
```
Ssh -I “key-xujun-beijing.pem” ec2-user@ec2-#-#-#-#.cn-north-1.compute.amazonaws.com.cn
```

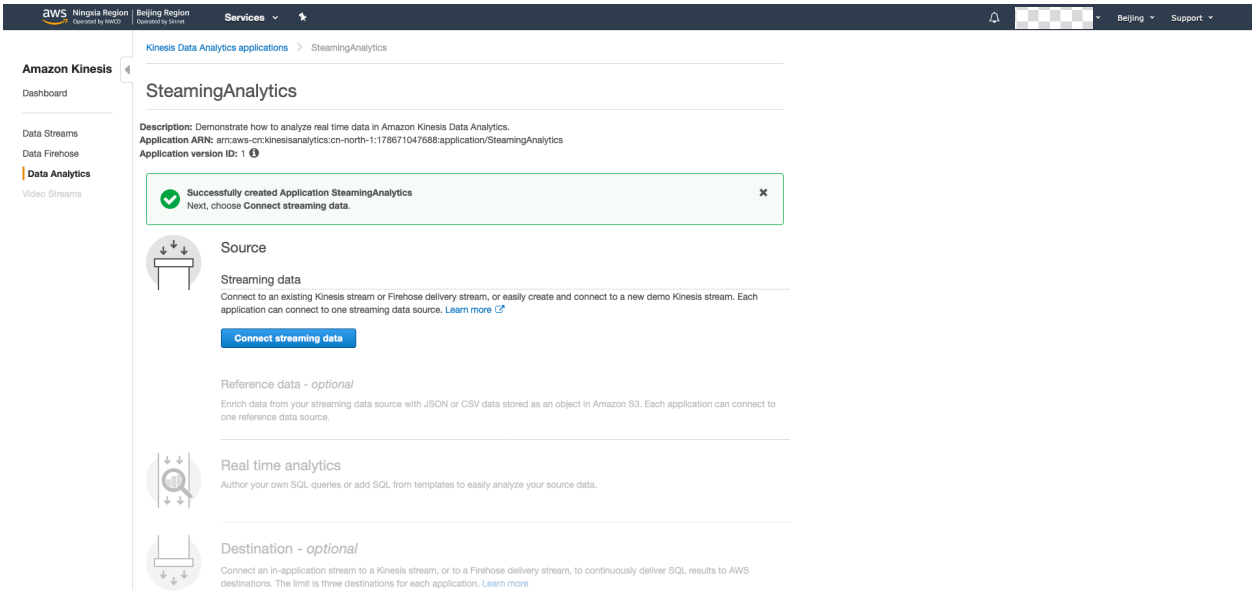
Download the data file and streaming script file from your S3 bucket, run the streaming script for synthesizing data streams.

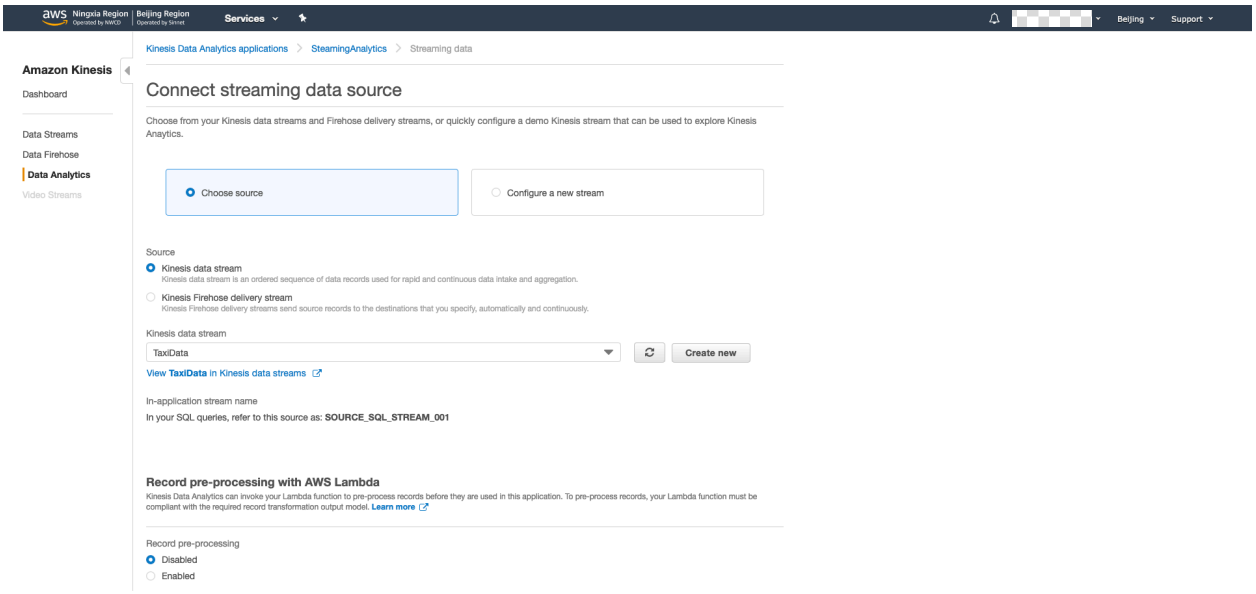
```
Wget https:// [your-s3-bucket].s3.cn-north-1.amazonaws.com.cn/files/kinesis/python/streaming.py

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.

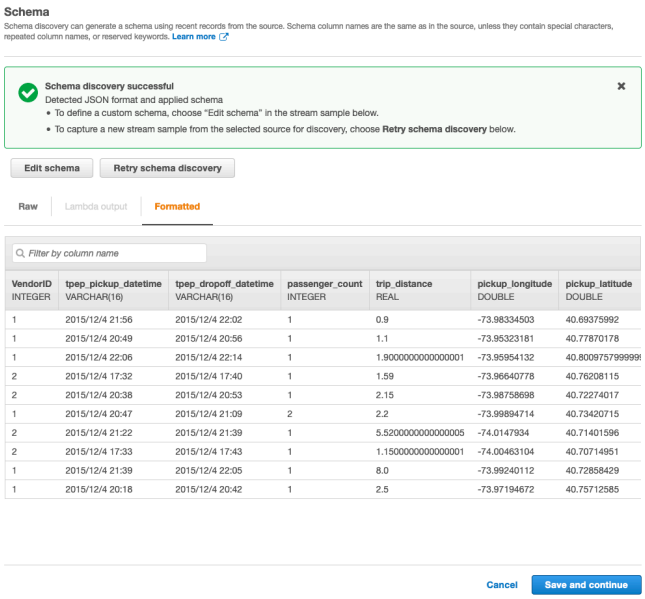
Once the data stream is generated, you can use Amazon Kinesis Data Analytics for following analytics.

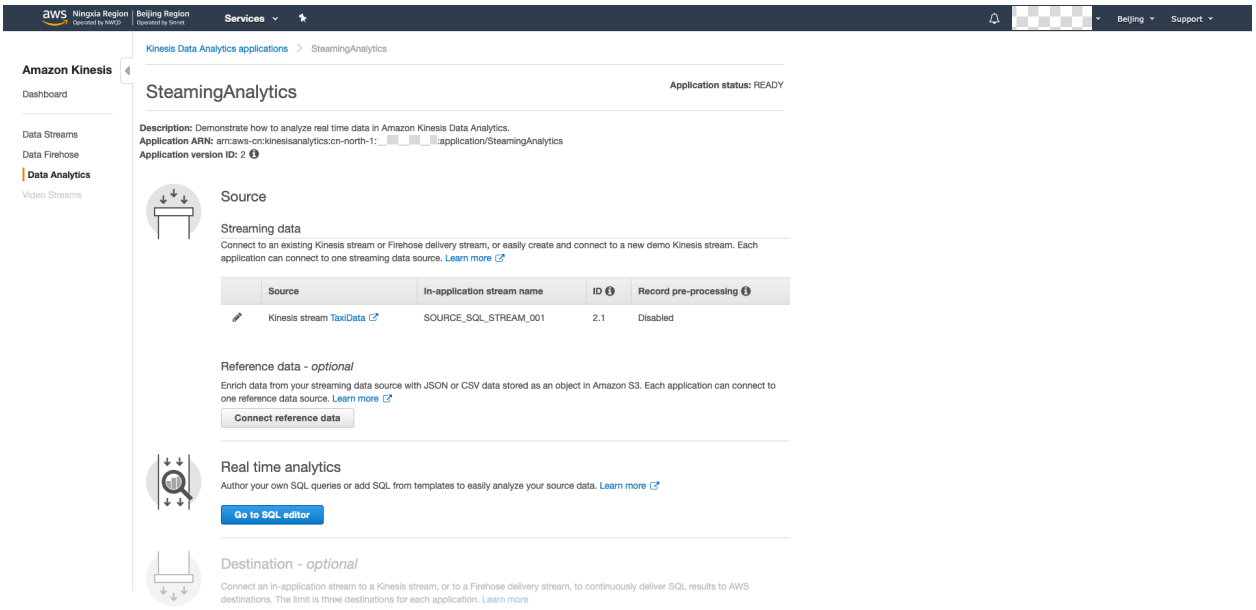






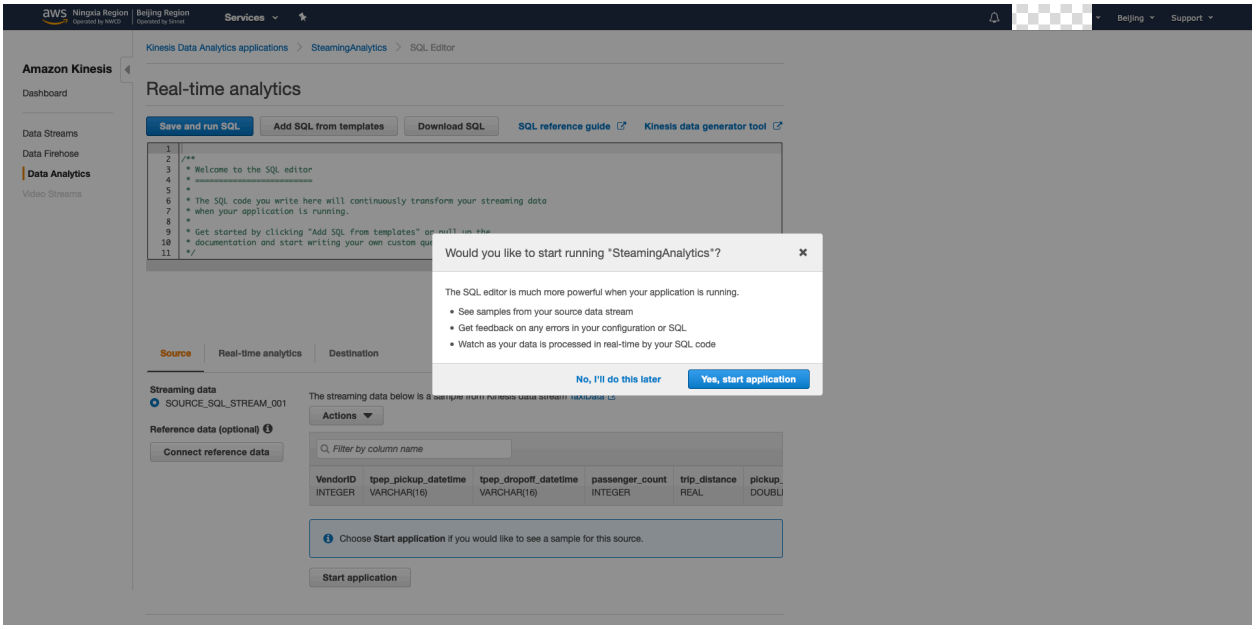
The following diagram illustrates the schema discovery succeeded.

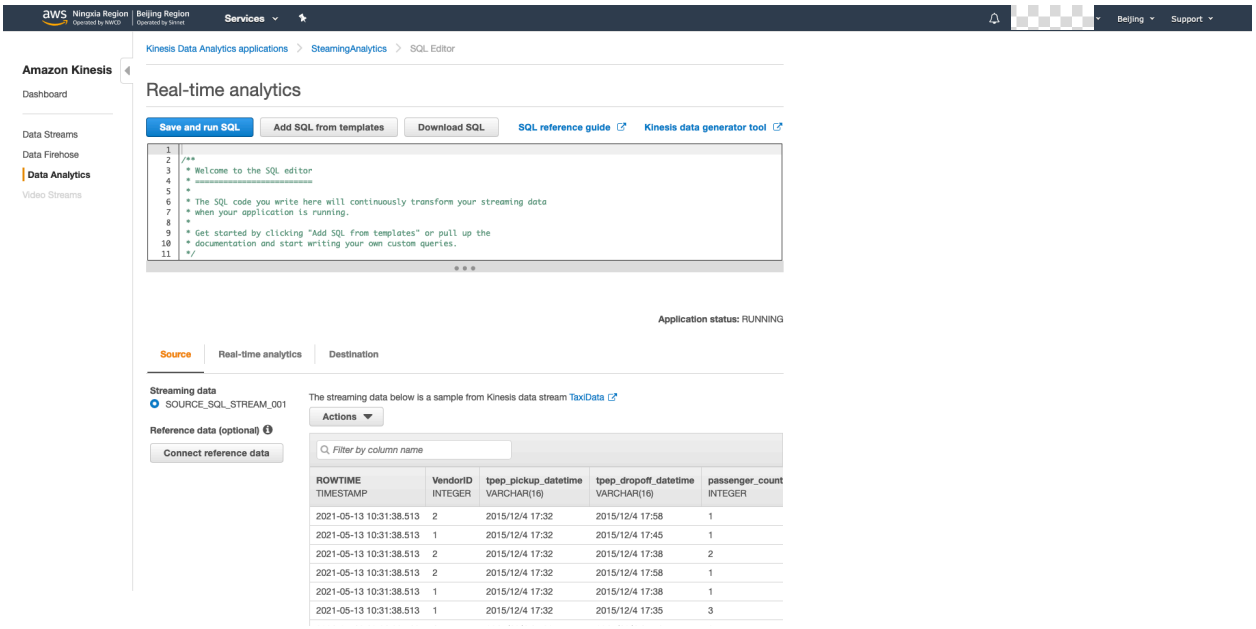




4.2.4. Real Time Analytics

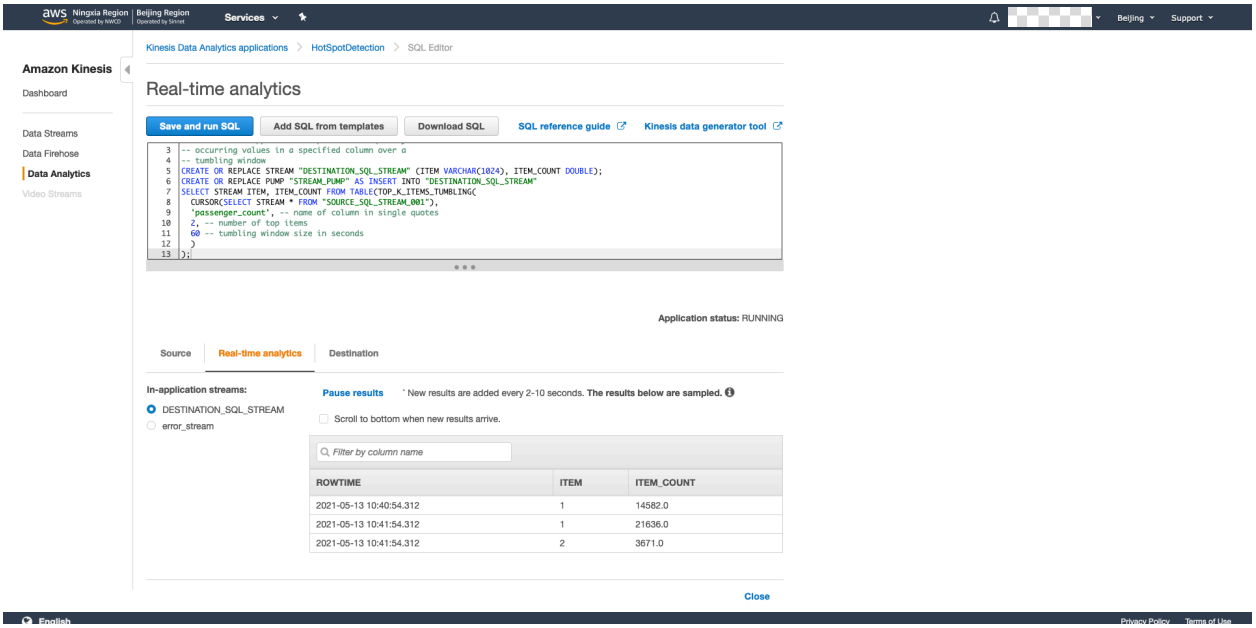
In the real-time analytics panel, the SQL editor helps you to see samples from your source data stream, get feedback on any errors in your configuration or SQL, watch as you data is processed in real-time by your SQL code.





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
  )
);
```



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/