**Data Pipeline(2) Spark Scala Project – DSTi**

Professor: Jean-Luc Canela

Student: Abdullah Daqdoqa

Project template: <https://github.com/jlcanela/spark-sbt-template>

Project data source and instructions: <https://github.com/jlcanela/spark-hands-on/wiki>

Project final repository: <https://github.com/sqoor/spark-sbt-template.git>

**Project Overview**

This project build using spark scala to query information from the dataset access.log.gz which represents information about the access on a server.

The project delivered using sbt configuration to get .jar file to run with java on local server or aws sever using AWS Glue and AWS S3 as ETL automatic process.

To execute the code you need java and the .jar file you pass two arguments, the path of the input file (access.log.gz) and the output path, the end result in .json format, extracting:

The scala file containing a function

* def createReport(gzPath: String, outputPath: String): Unit // signature of the function
* the function find all the dates having too big number of connection (> 20000)
* for each date
* compute the list of number of access by URI for each URI
* compute the list of number of access per IP address for each IP address

**Excepted result:**

output result is .json file include the access number by URI and by Ips for the dates that has more than 20000 connections per day

{"date":"2020-29-10","countByIpAdresses":{"1.1.1.1":2,"2.2.3.4":1},"countByUri":{"/administator":23}}

Two files URIs and UPS on the selected output path parameter when execution .jar file

Will be within two in path prefixed with part-…etc.

**How to run the code and deploy**:

To package the project, run this command

* sbt assembly

To Deploy

Copy/Upload the fatjar to the destination:

> TARGET\_LOCATION=<location>

> cp target/scala-2.12/spark-sbt-template-assembly-1.0.jar $TARGET\_LOCATION

To run your project locally

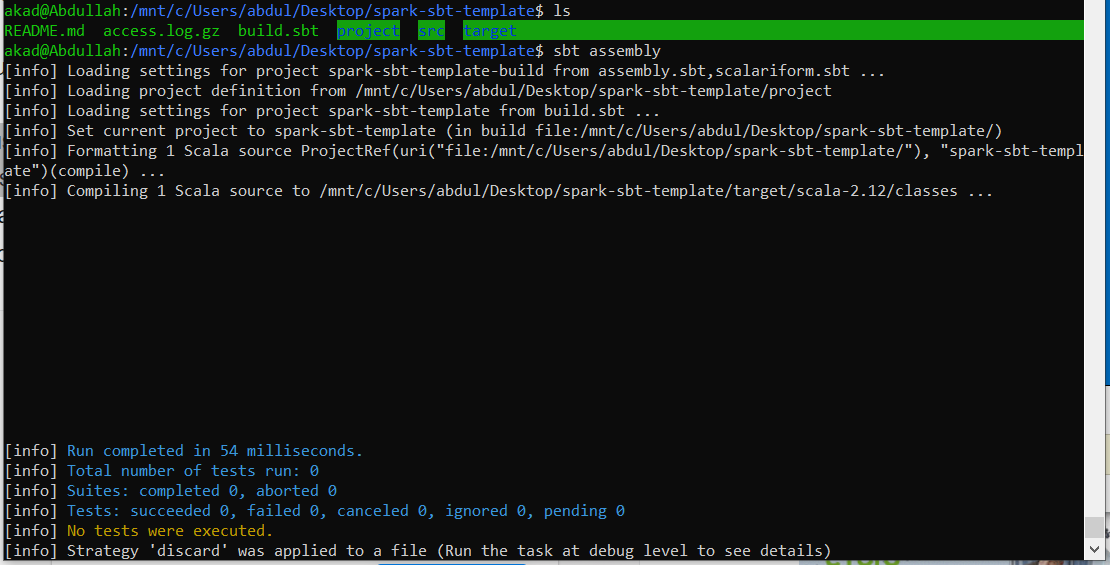
* JAR\_PATH=$(pwd)/target/scala-2.12/spark-sbt-template-assembly-1.0.jar
* spark-submit --master=local[\*] --deploy-mode client --class App $JAR\_PATH path\_input\_file path\_output\_file

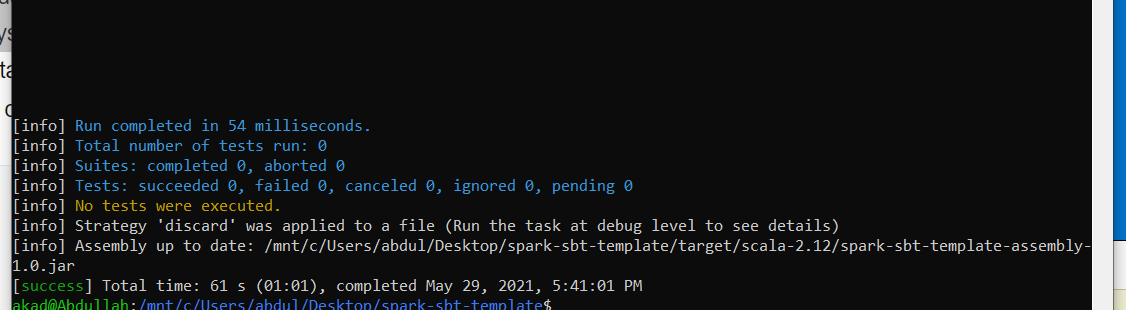
Environment:

* OS Ubuntu 18.04.5 LTS
* IDE IntelliJ IDEA 2021
* Spark shell version 3.1.1
* JDK 11.0.11
* Scala 2.11.12

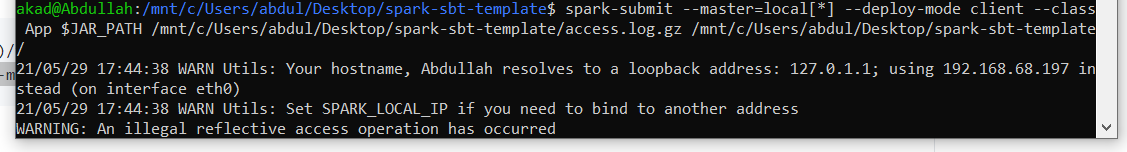
**Captures running the application locally:**

* Run the sbt assembly command on the project folder path, to get the .jar file

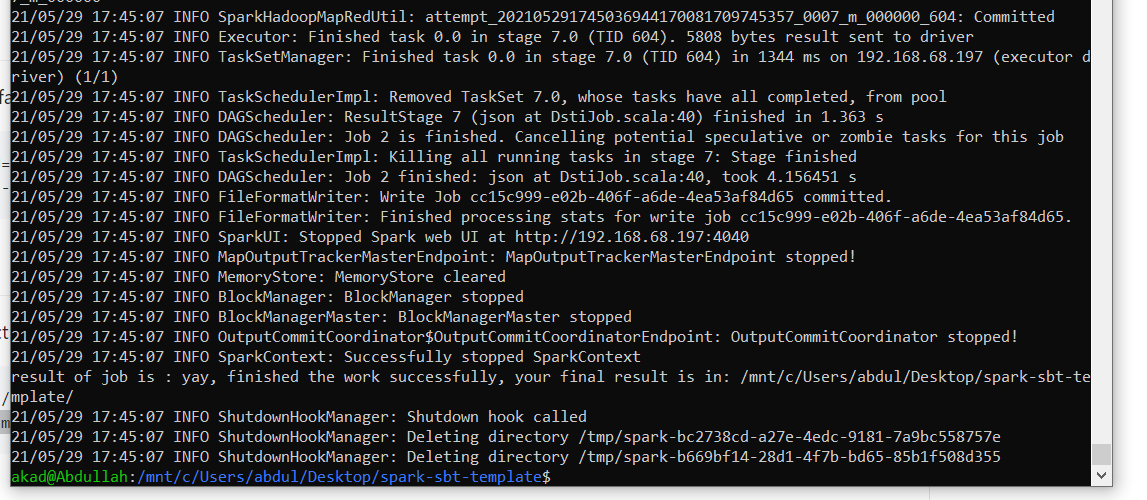




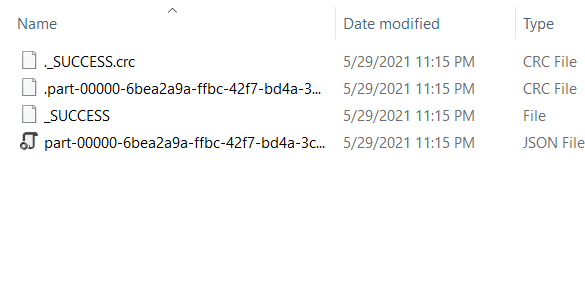
* Execute spark-submit with .jar file provides at the end the path to input files and output folder



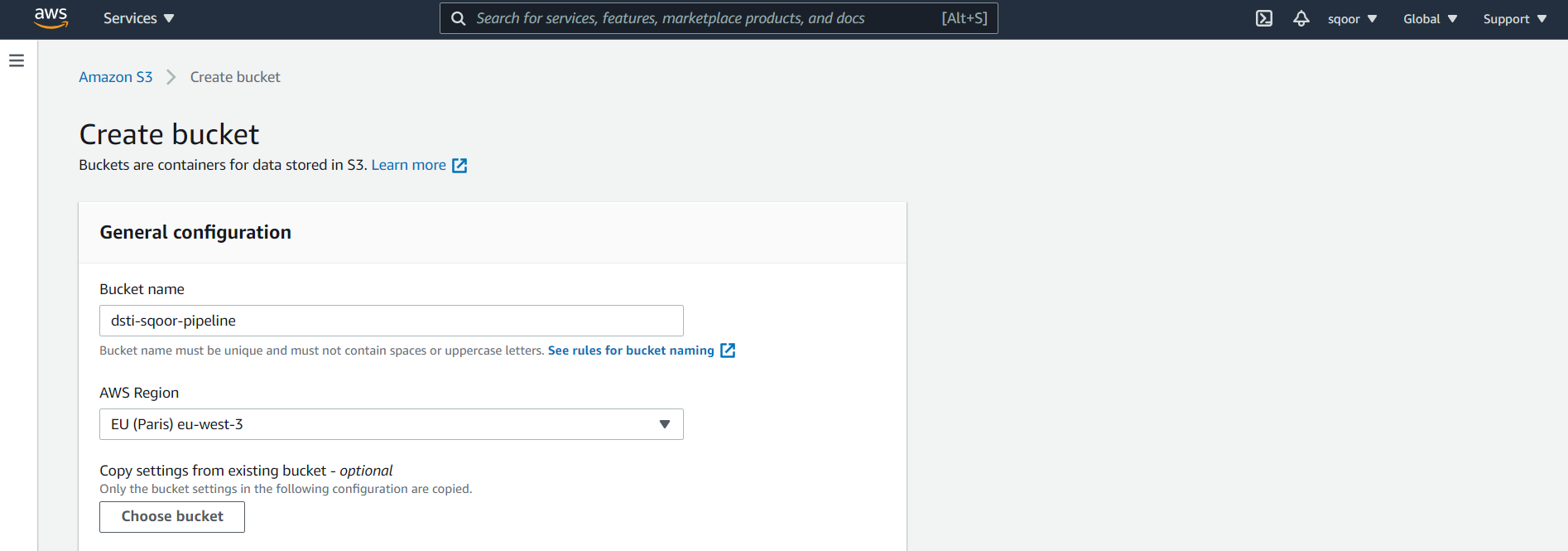
* Program finishes successfully

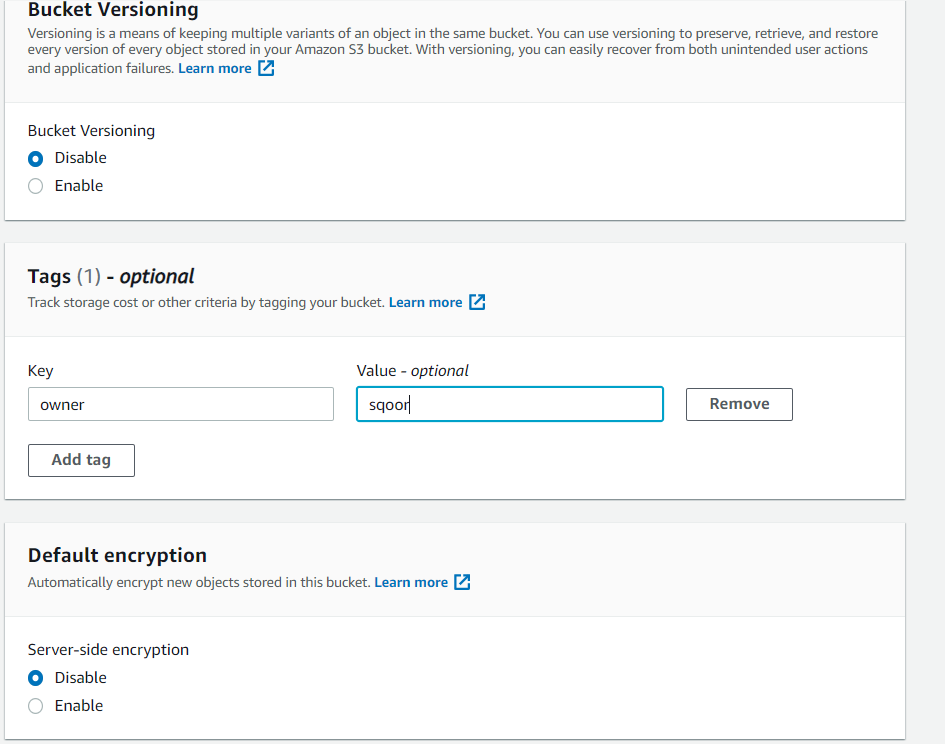


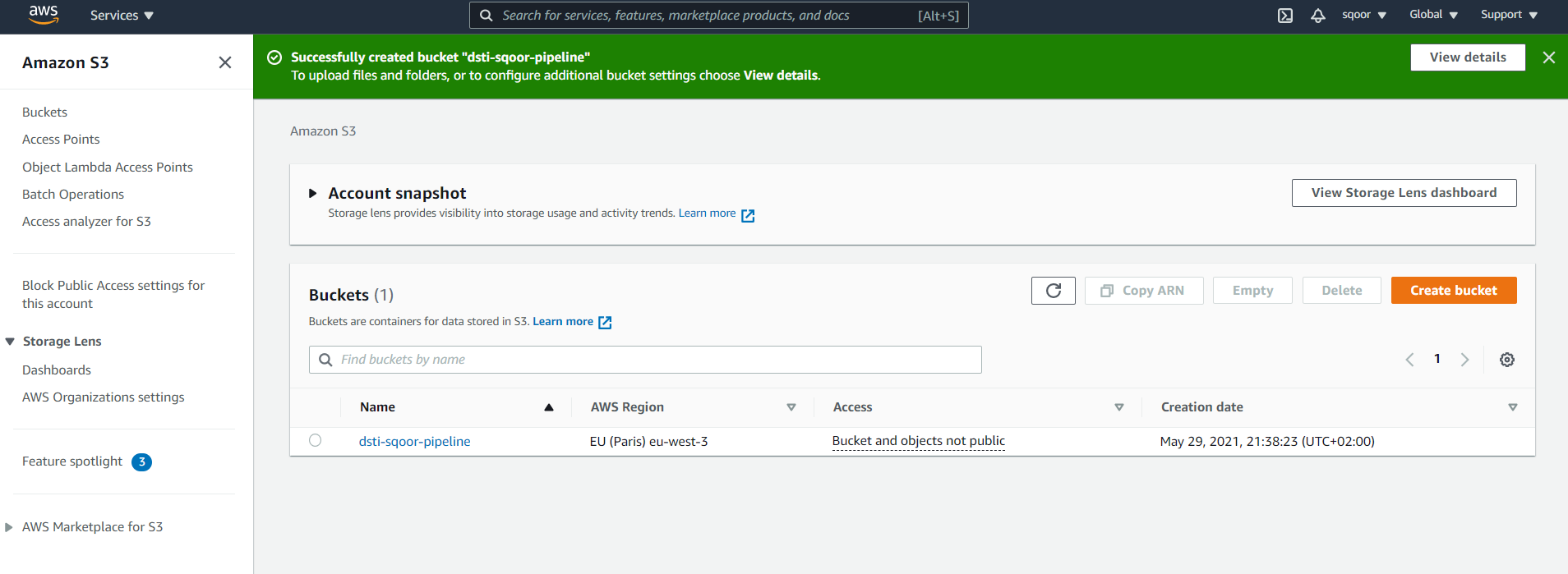
* Output files, .json is the target file

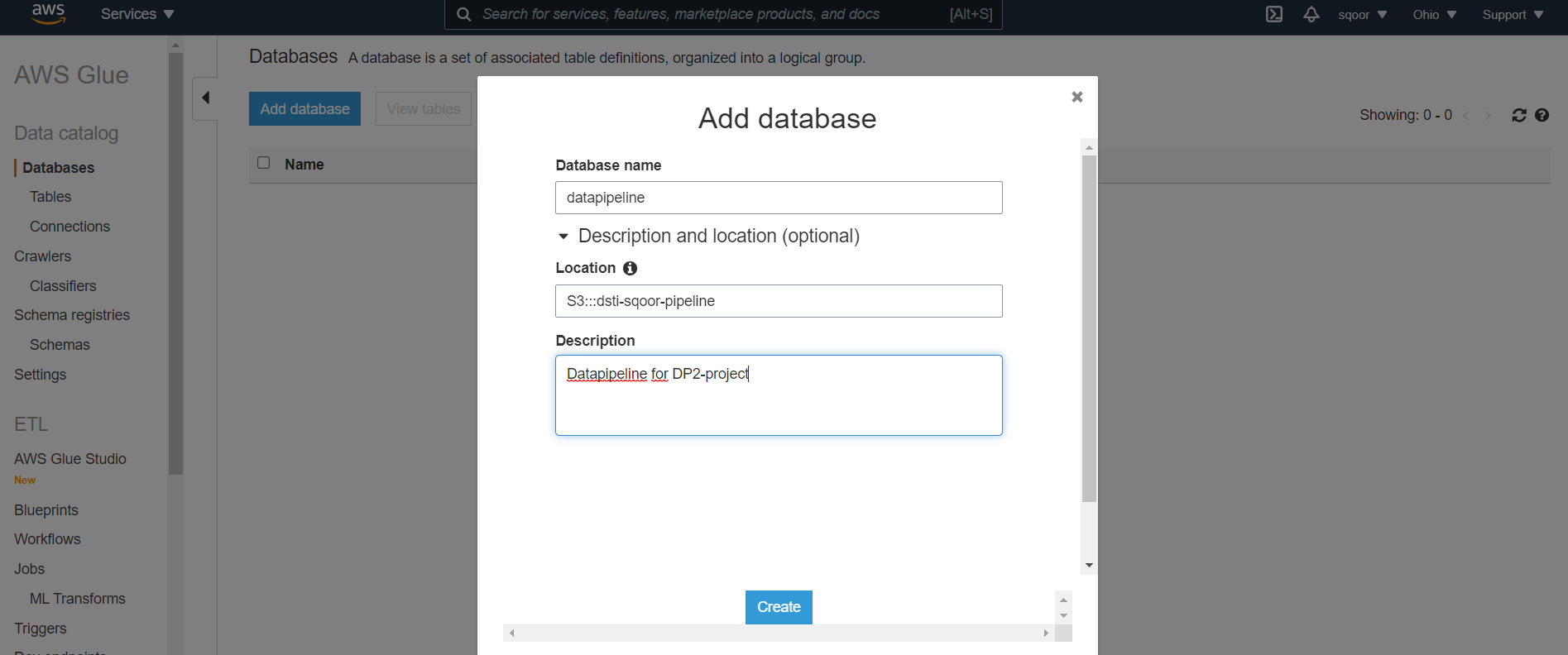


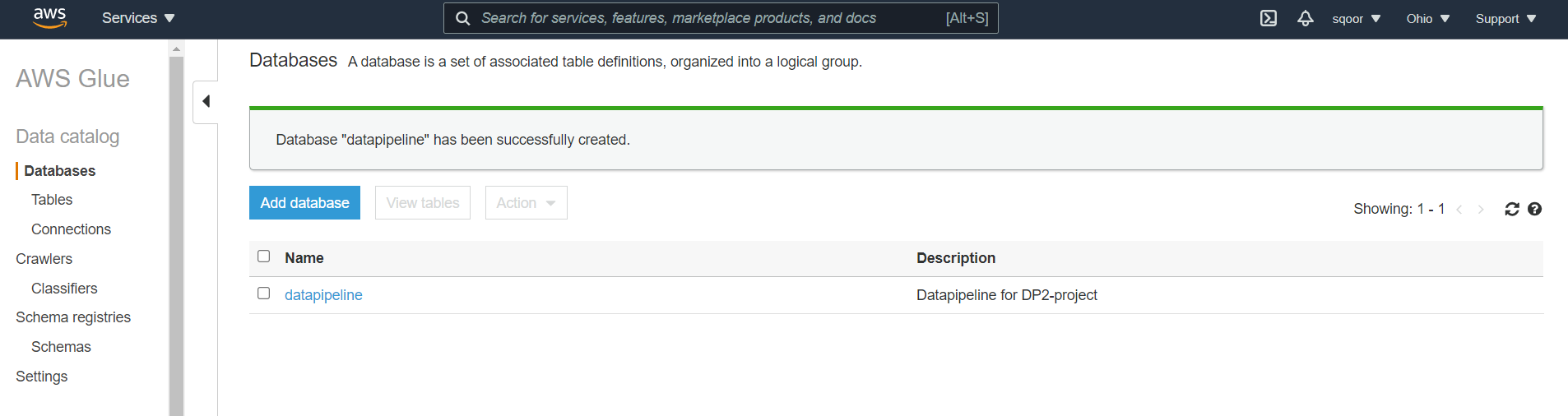
**Captures runing the application on AWS Amazon:**

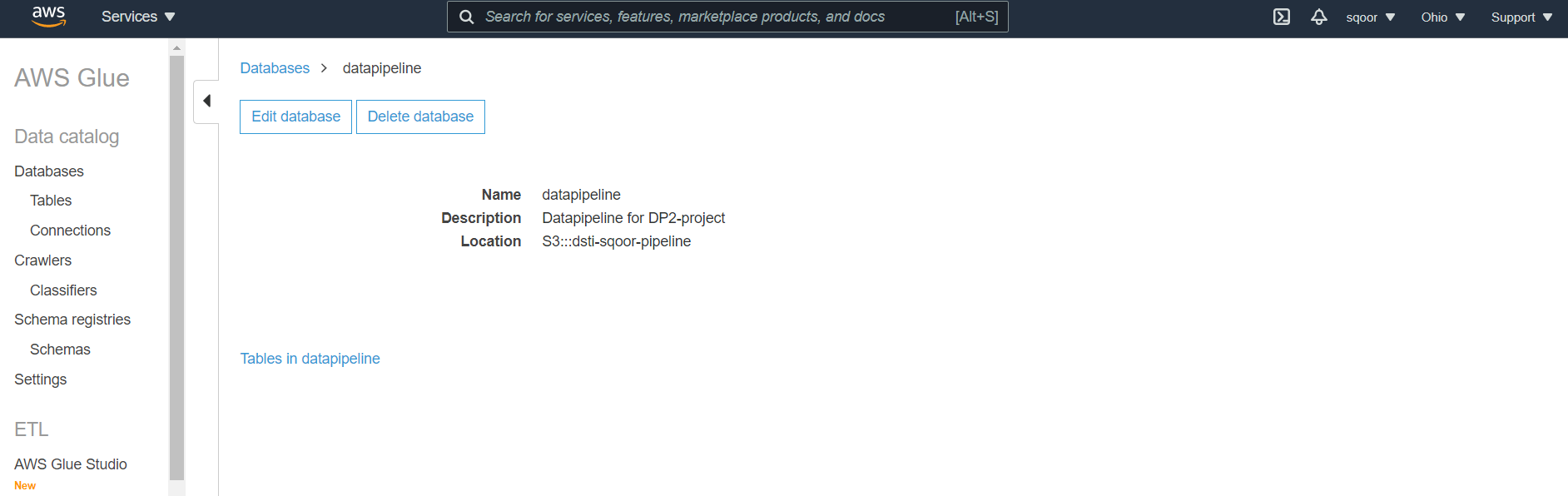




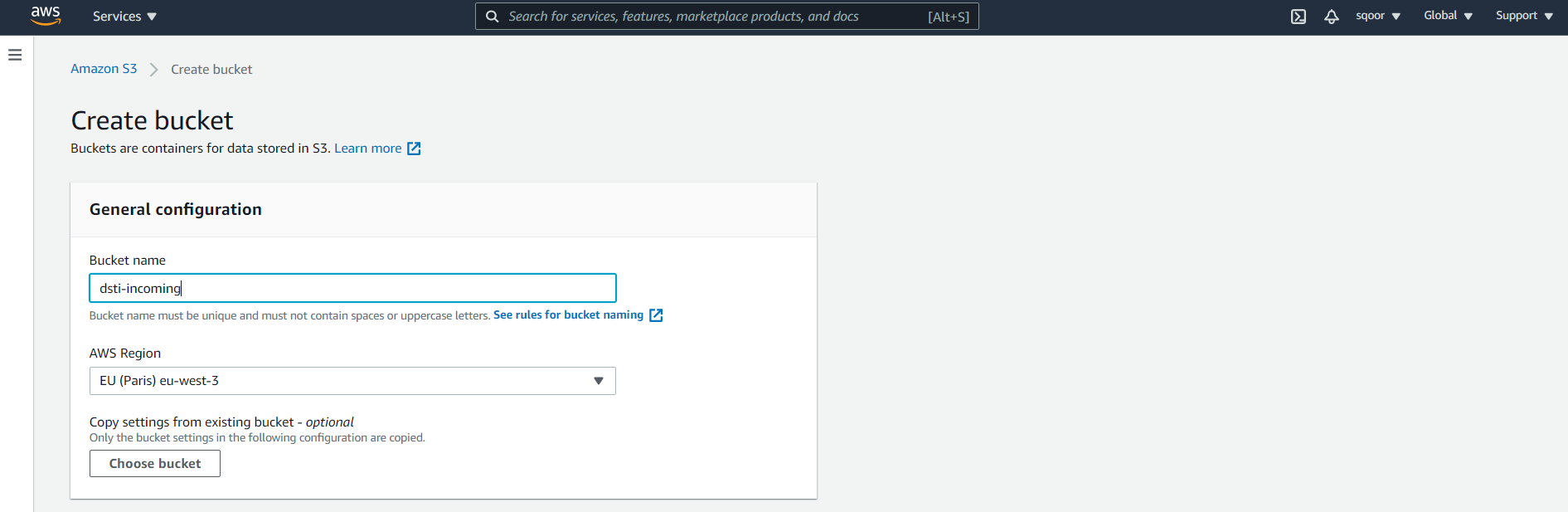


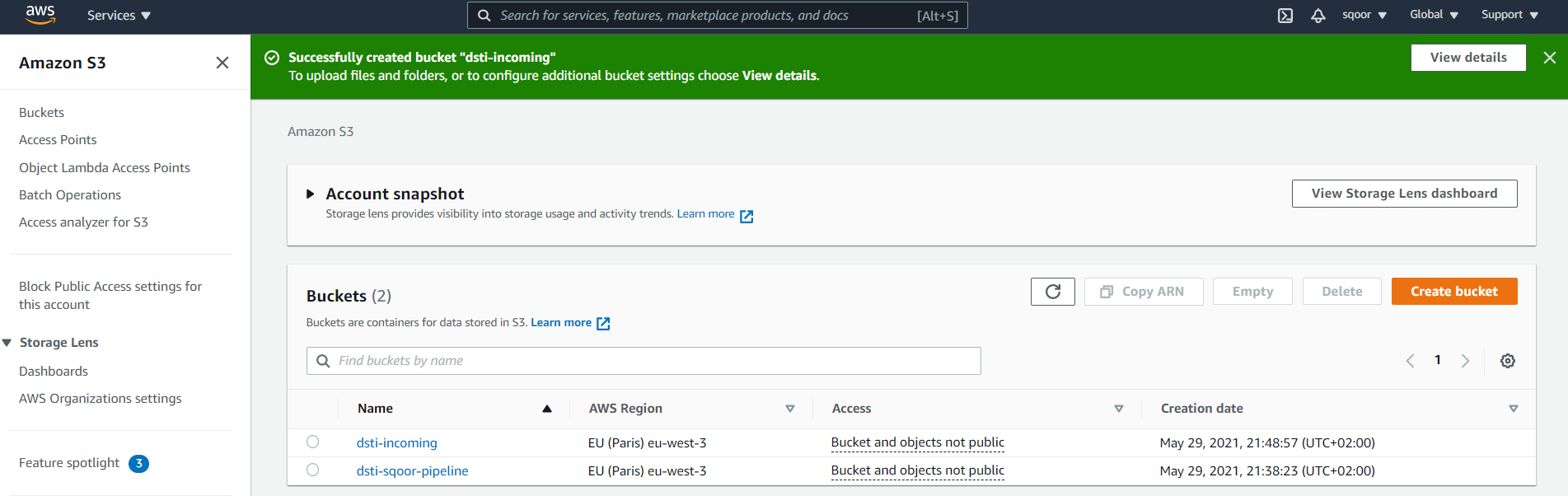


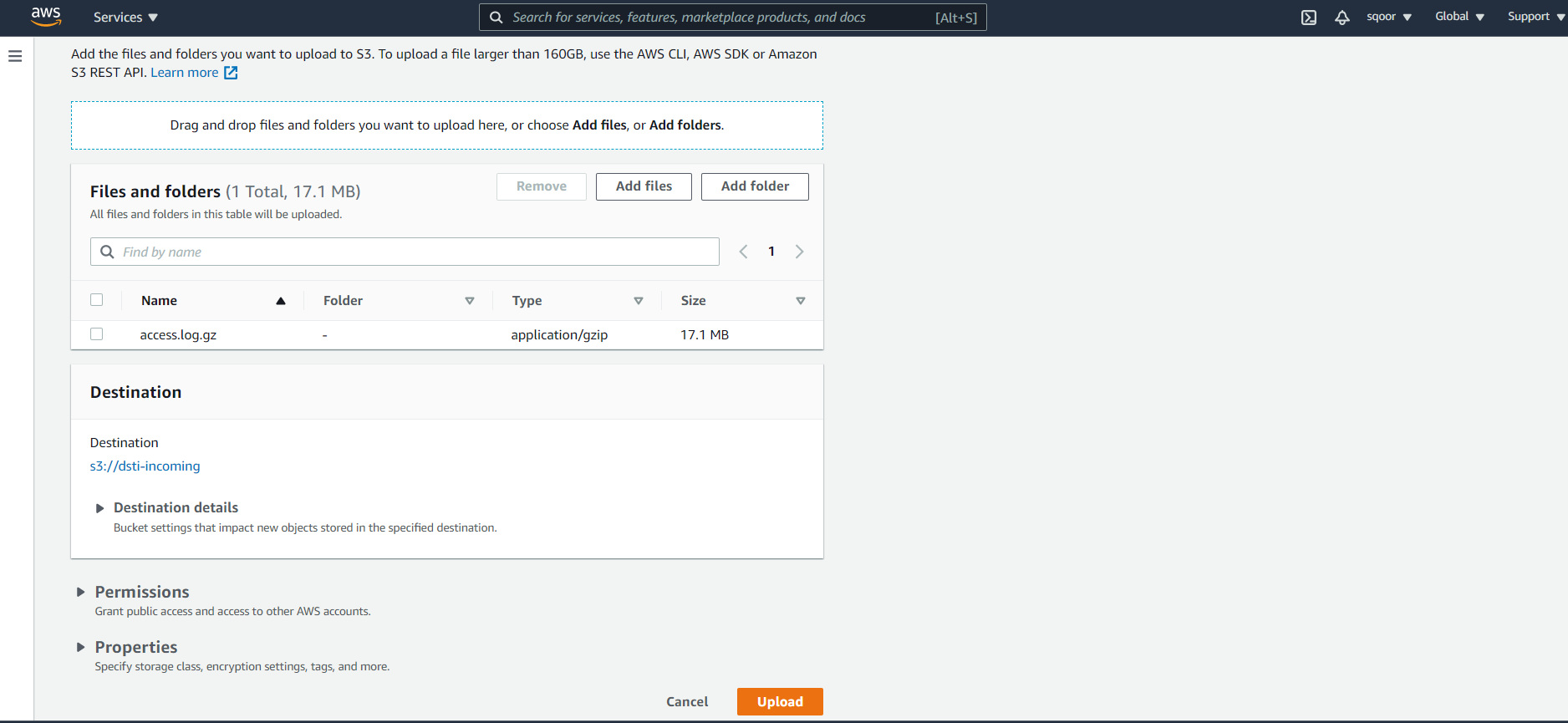


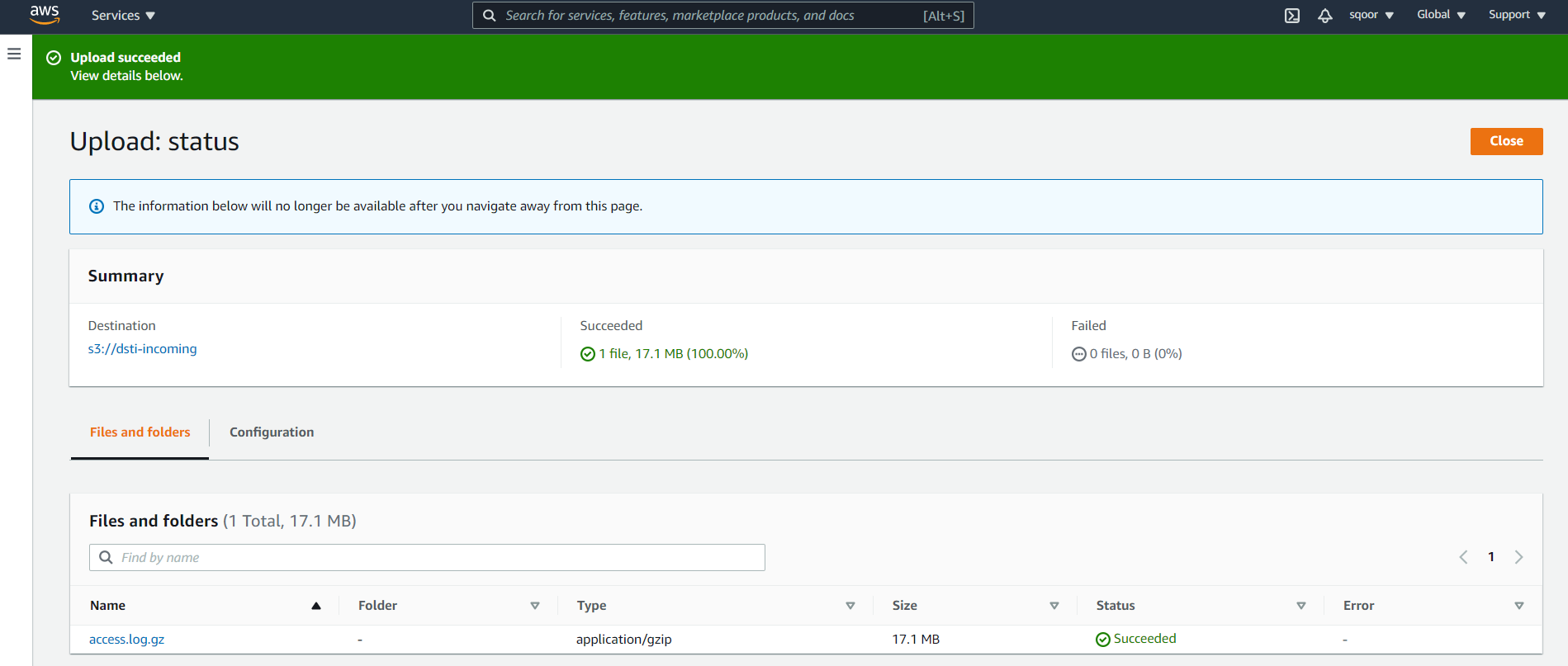


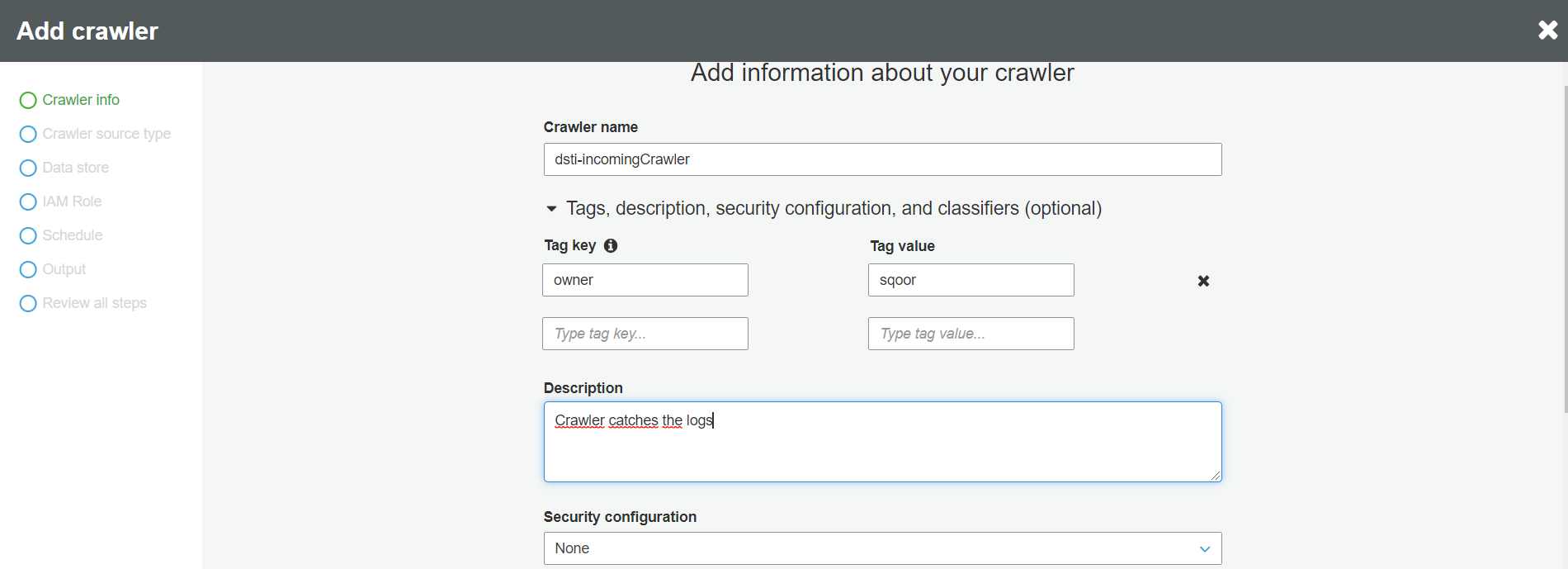
Now create one more S3 bucket for storing the input data:

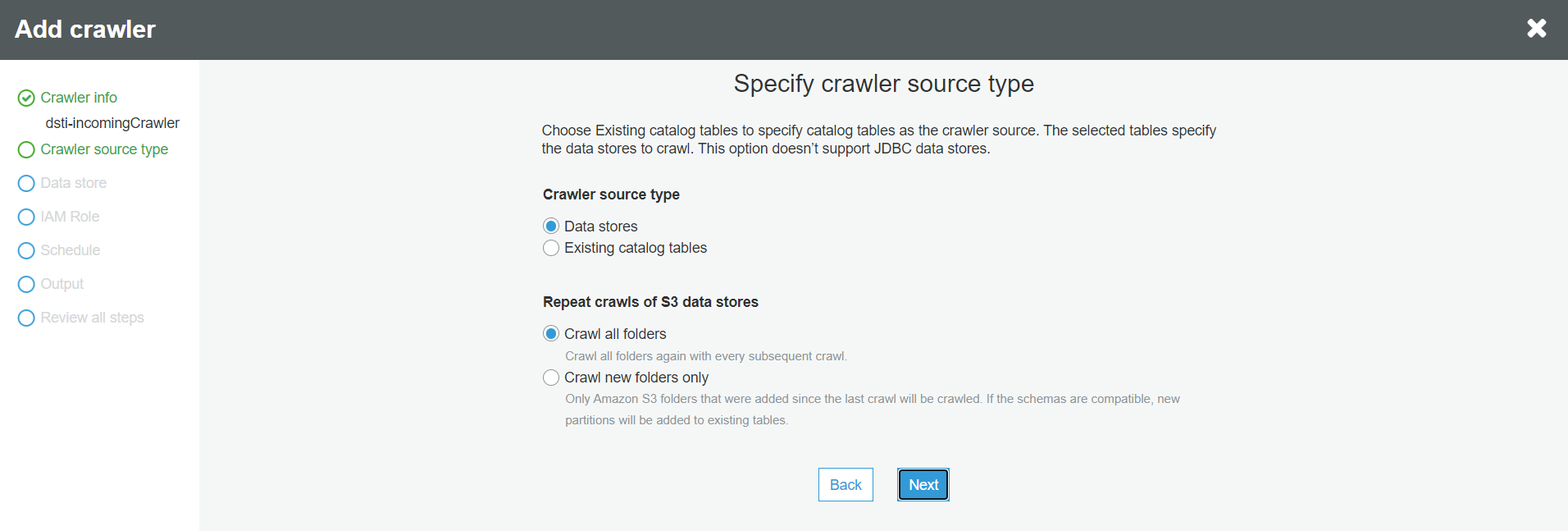


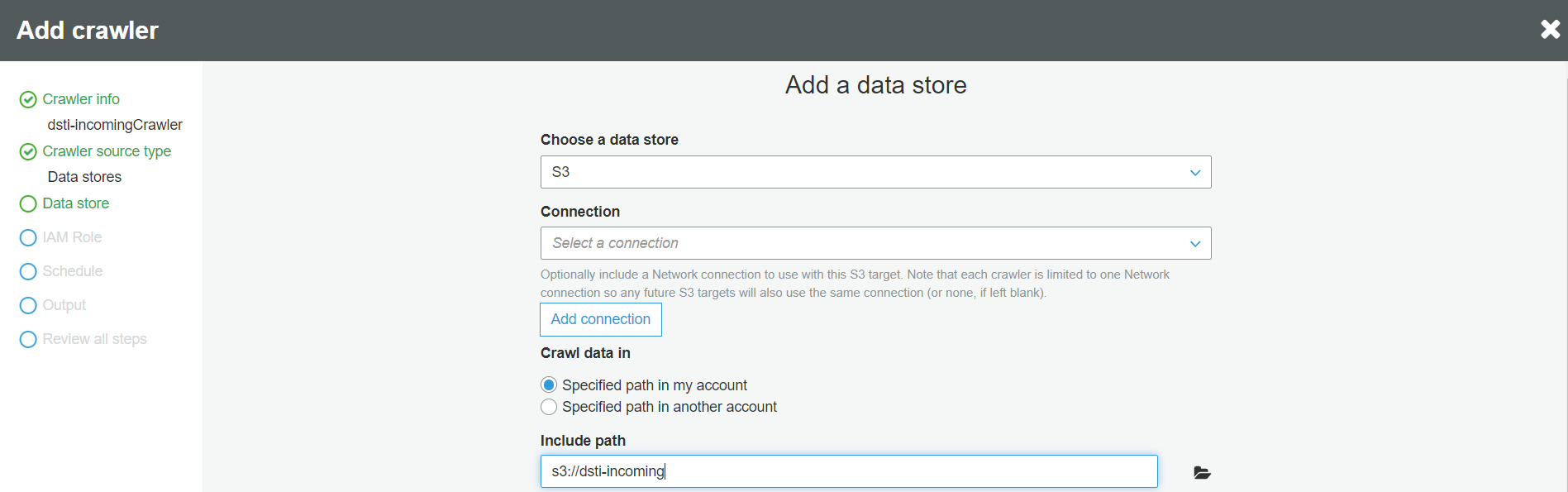


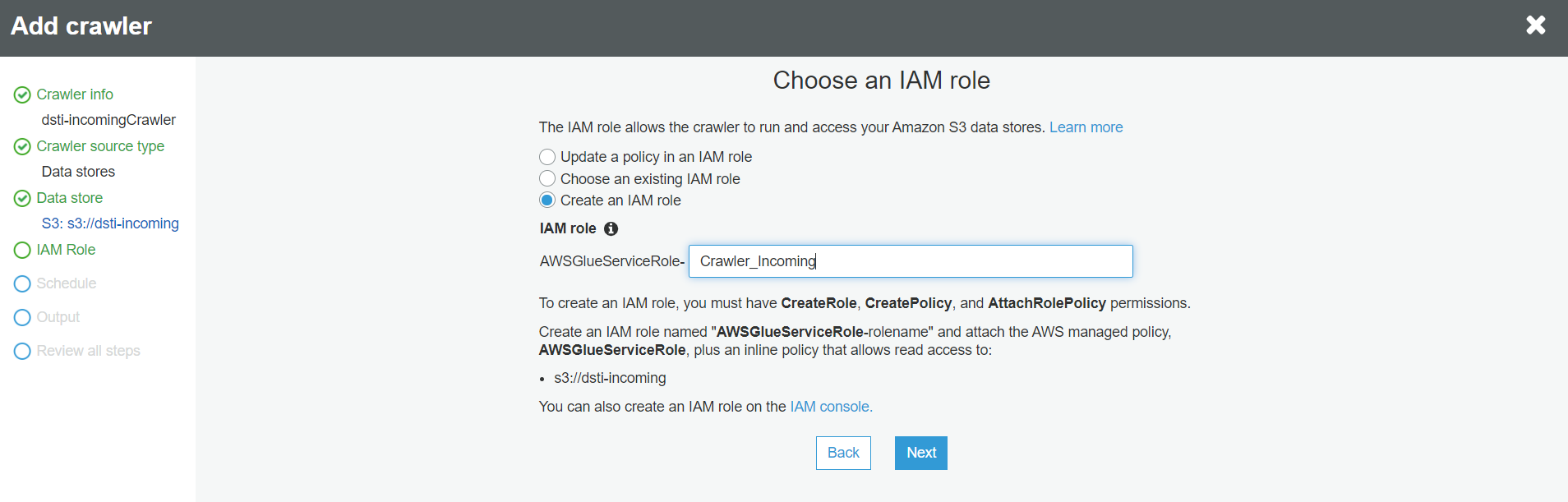


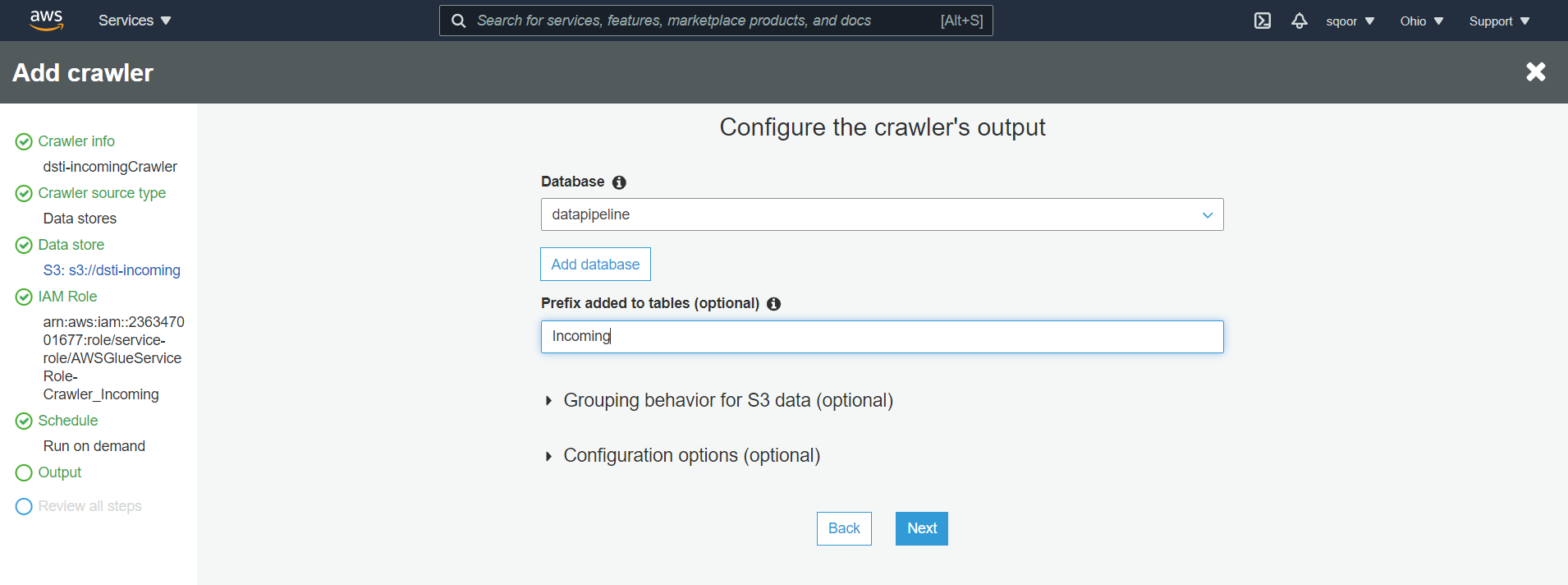


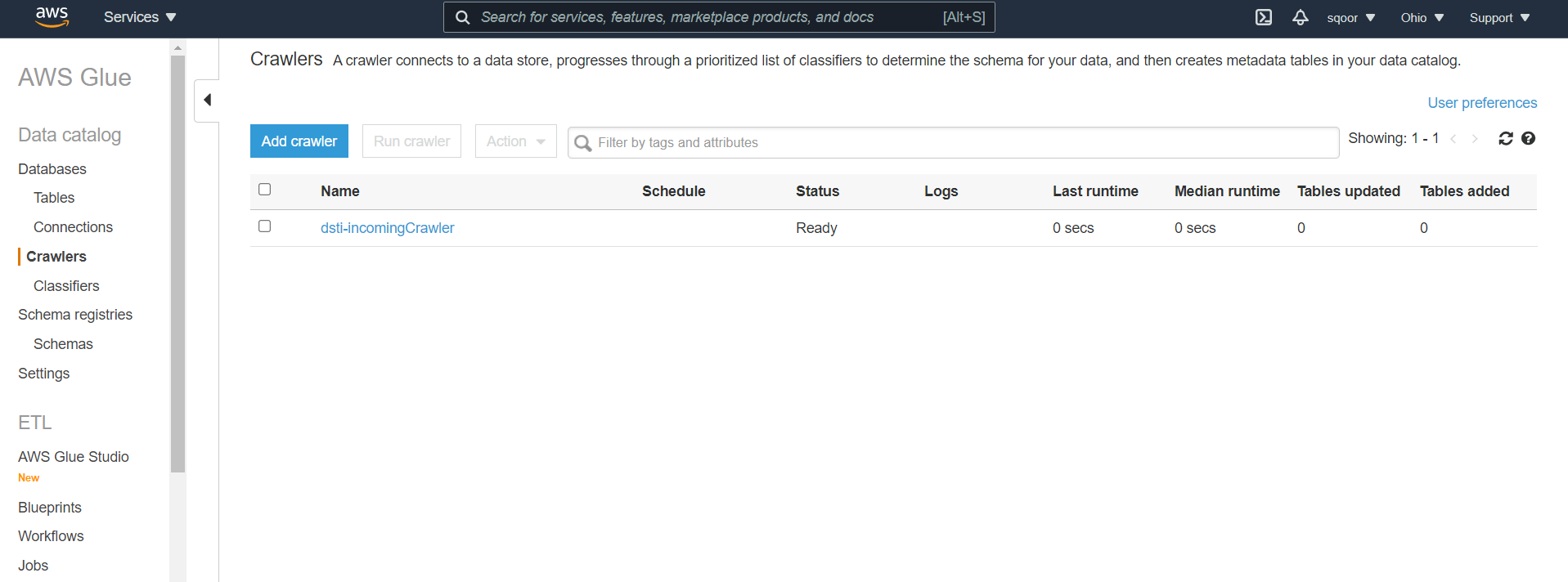




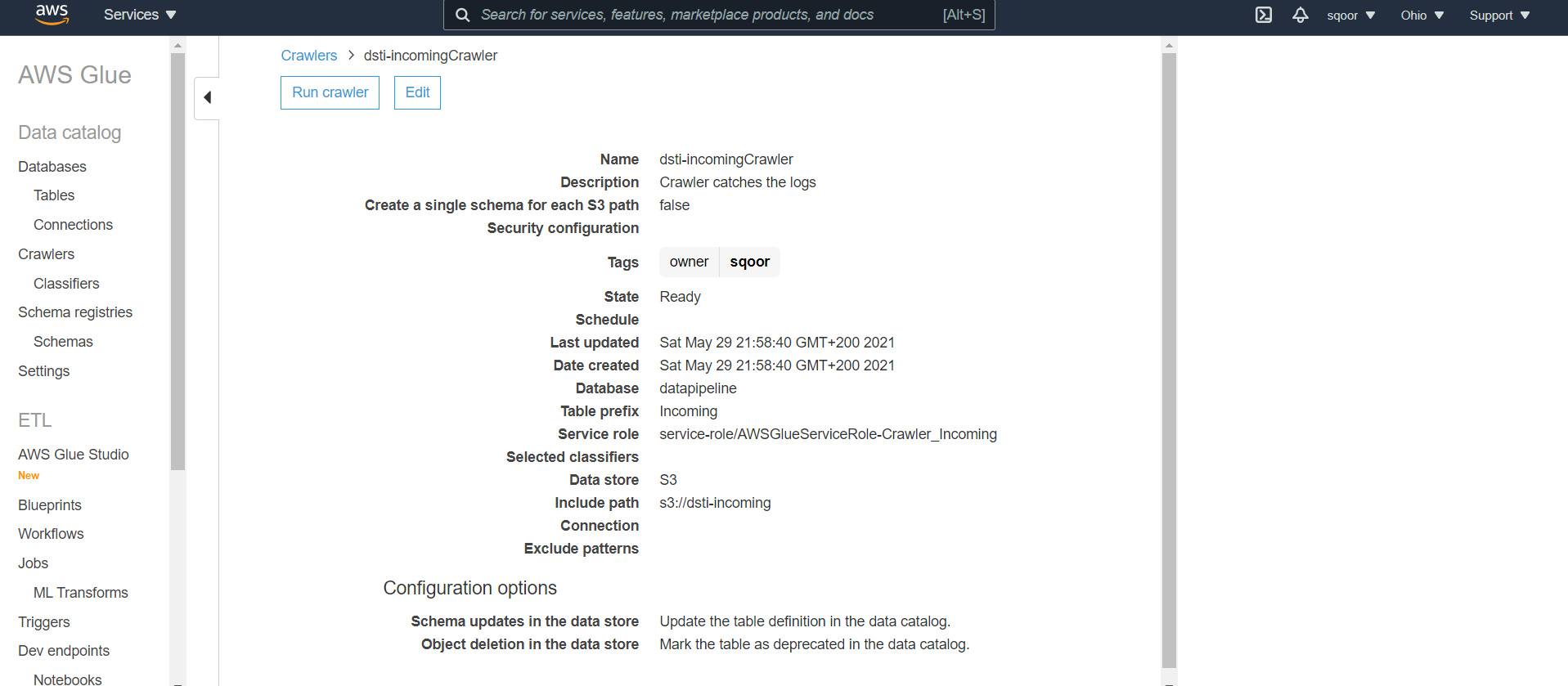




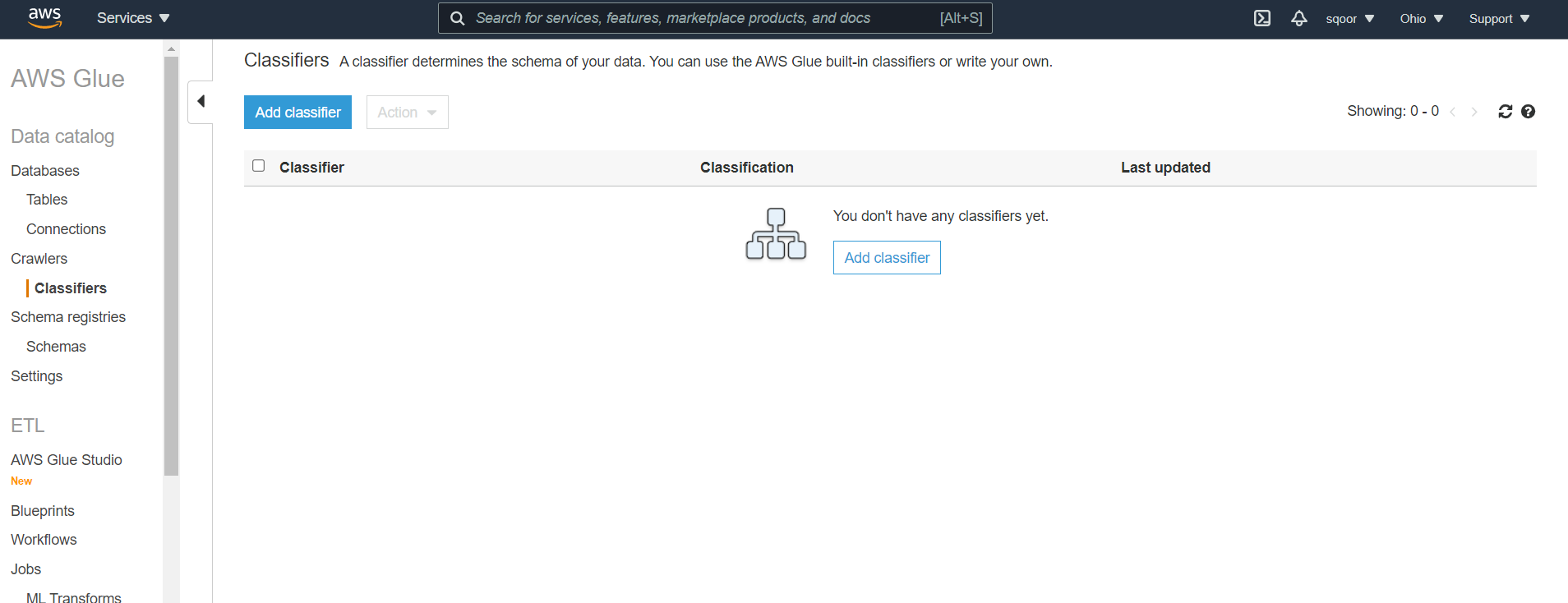


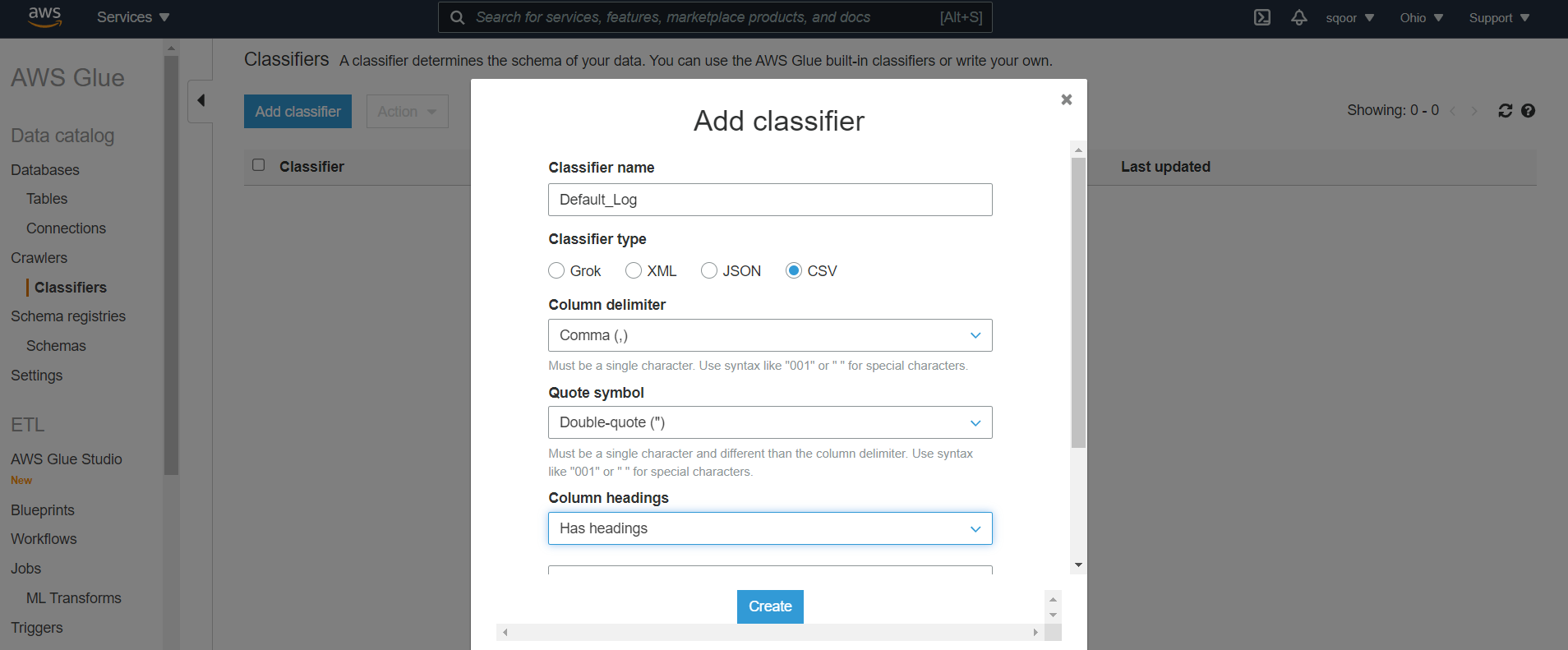


**Run the crawler:**

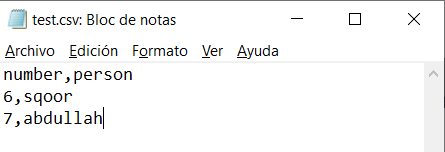


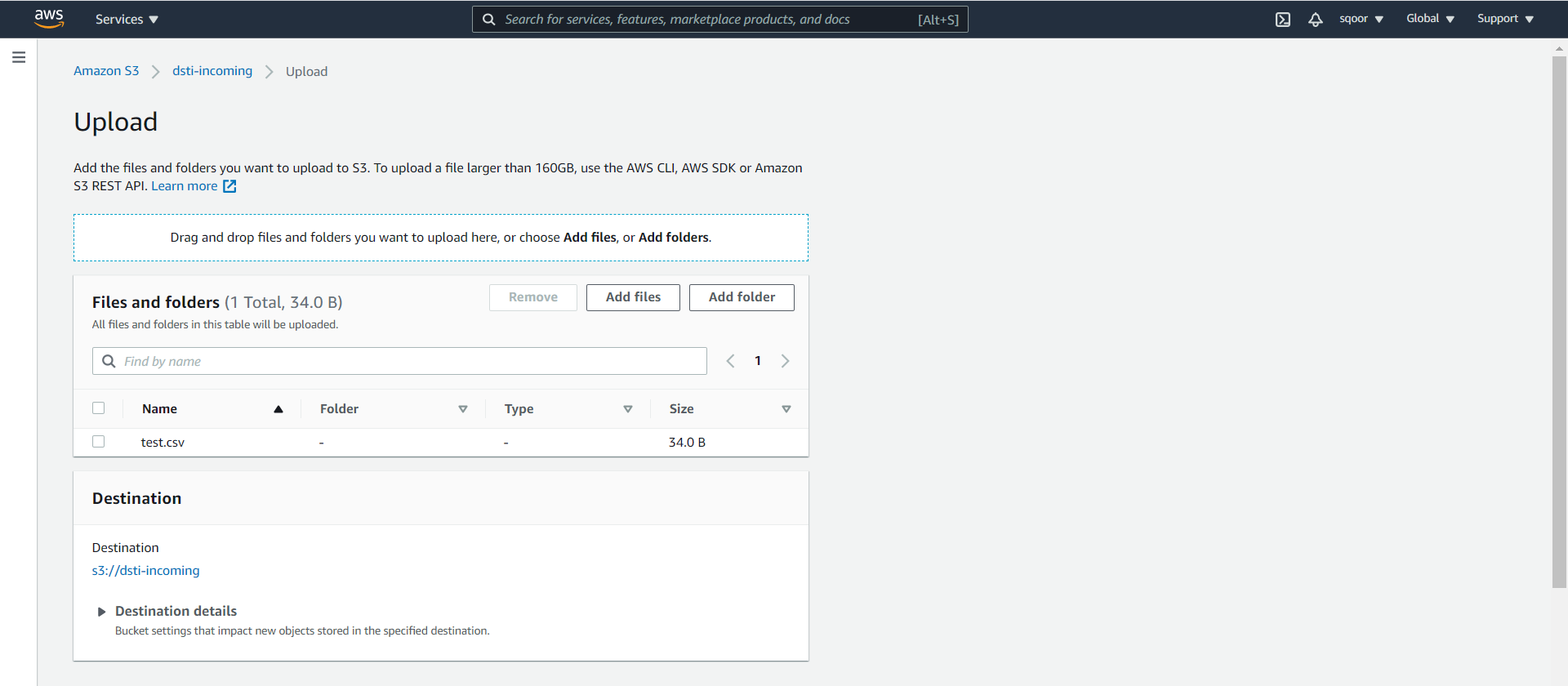
Add a classifier under crawlers:

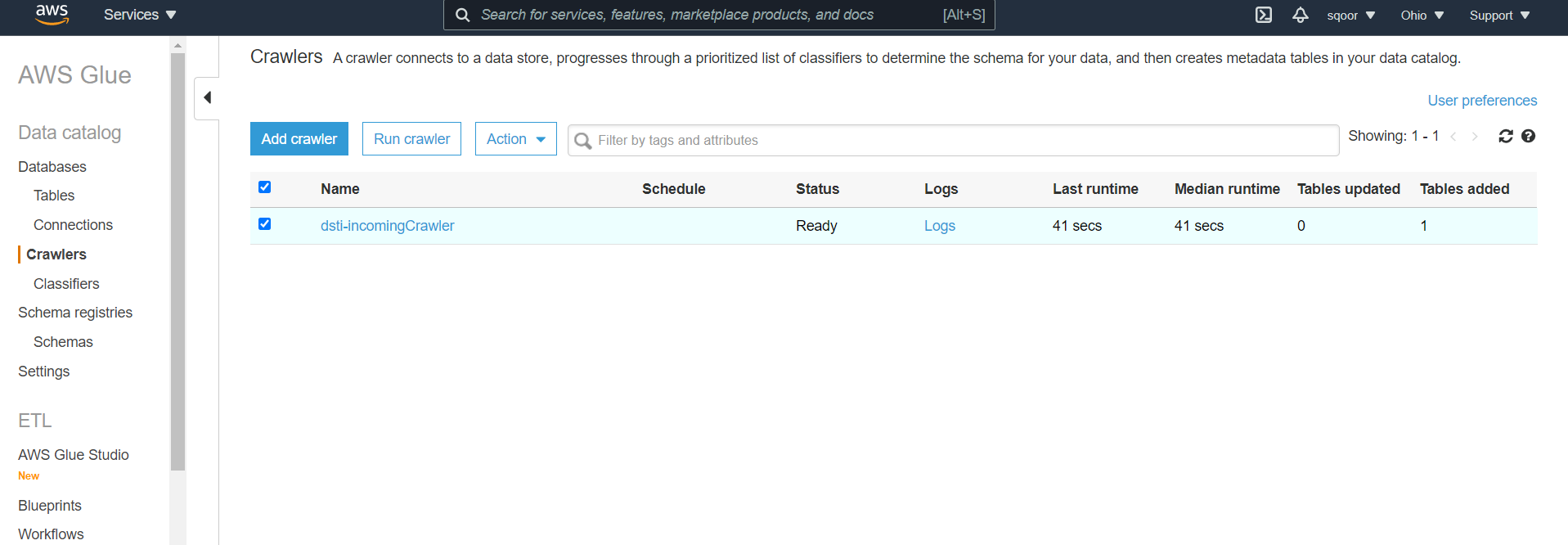


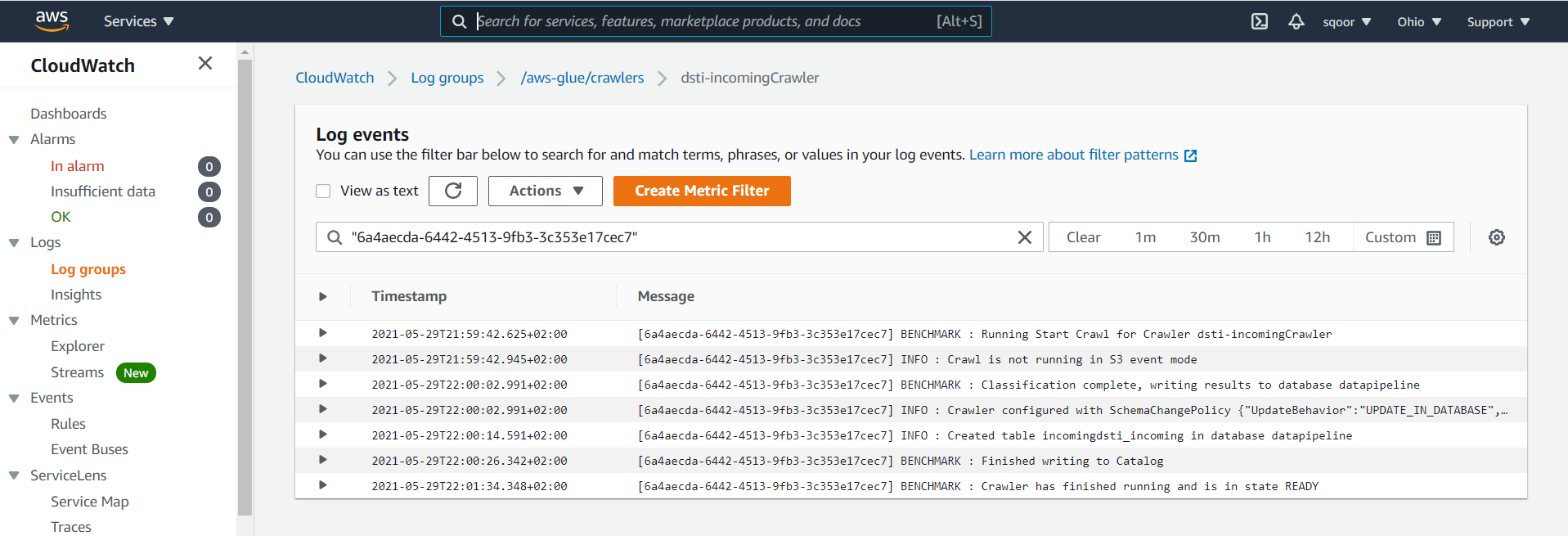


Test it with a .csv, by firstly uploading it to the incoming S3 bucket:

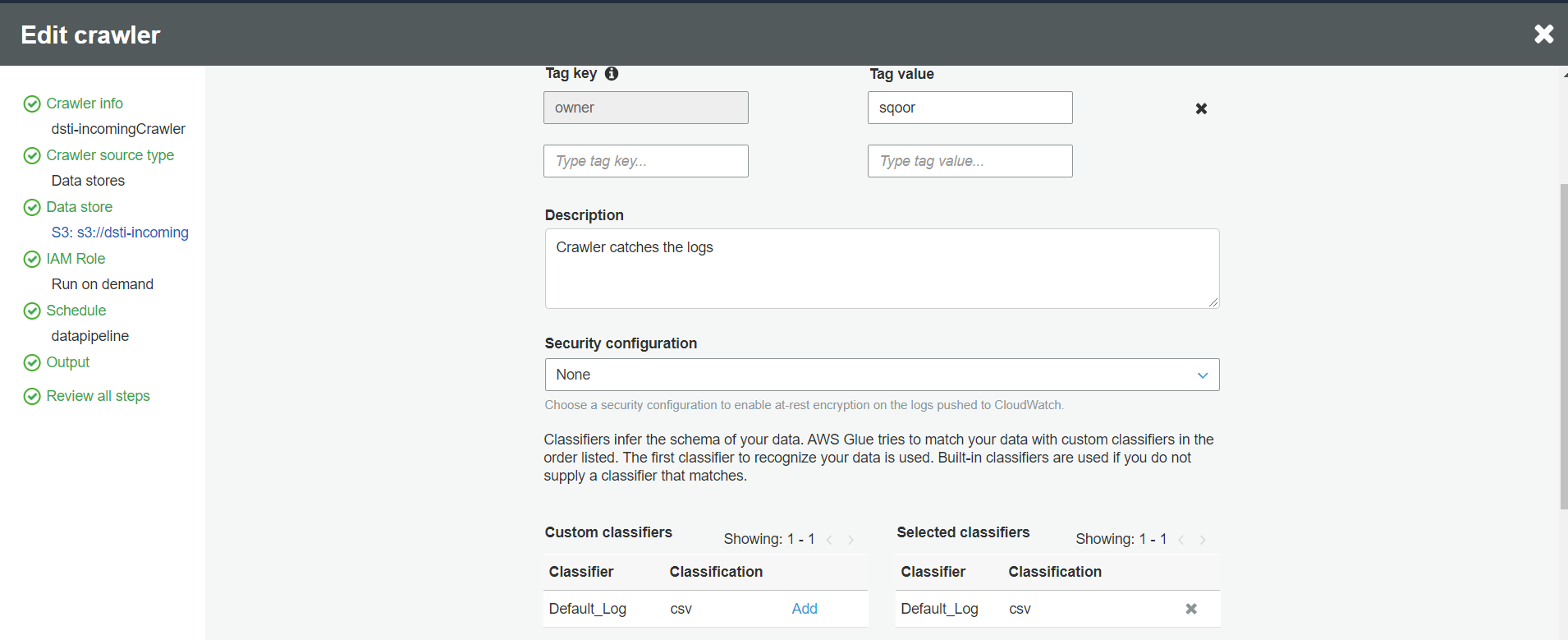


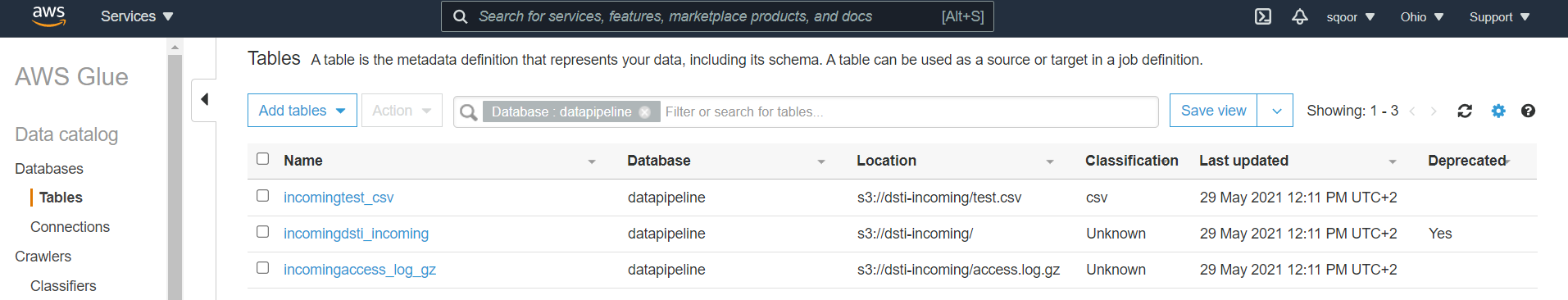




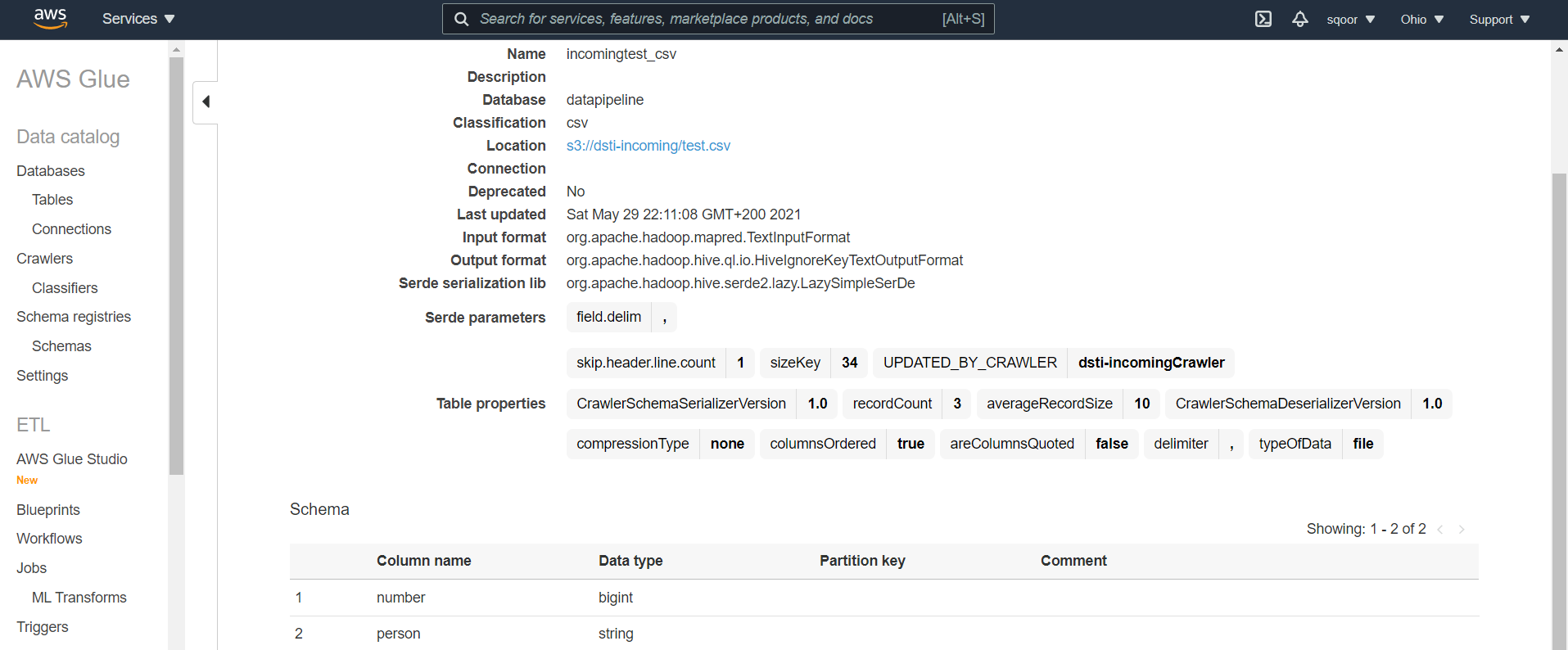


We now select our crawler and edit it by adding the just created Default\_Log.csv file. We keep the resto f the settings as they were.

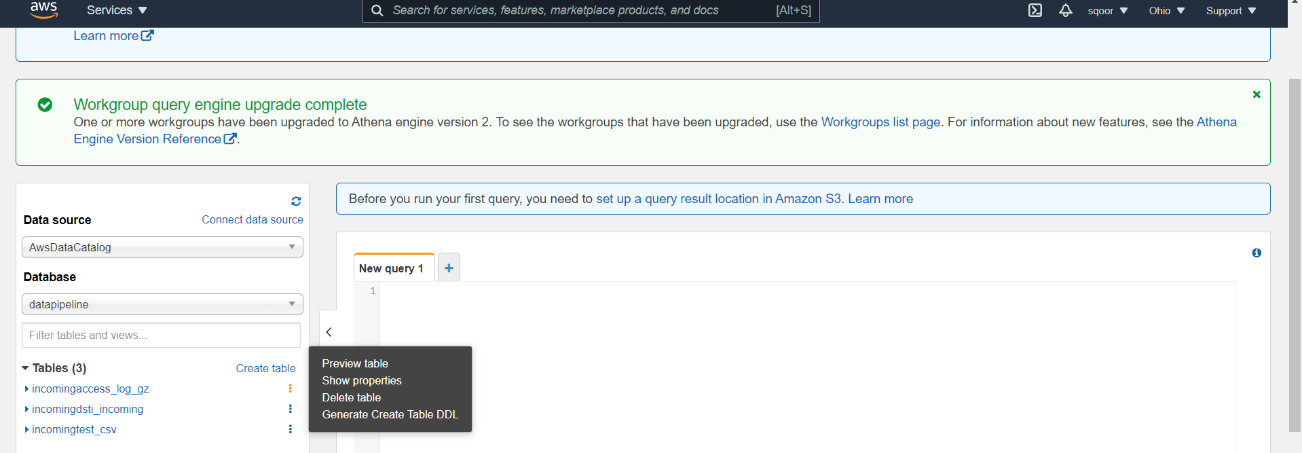


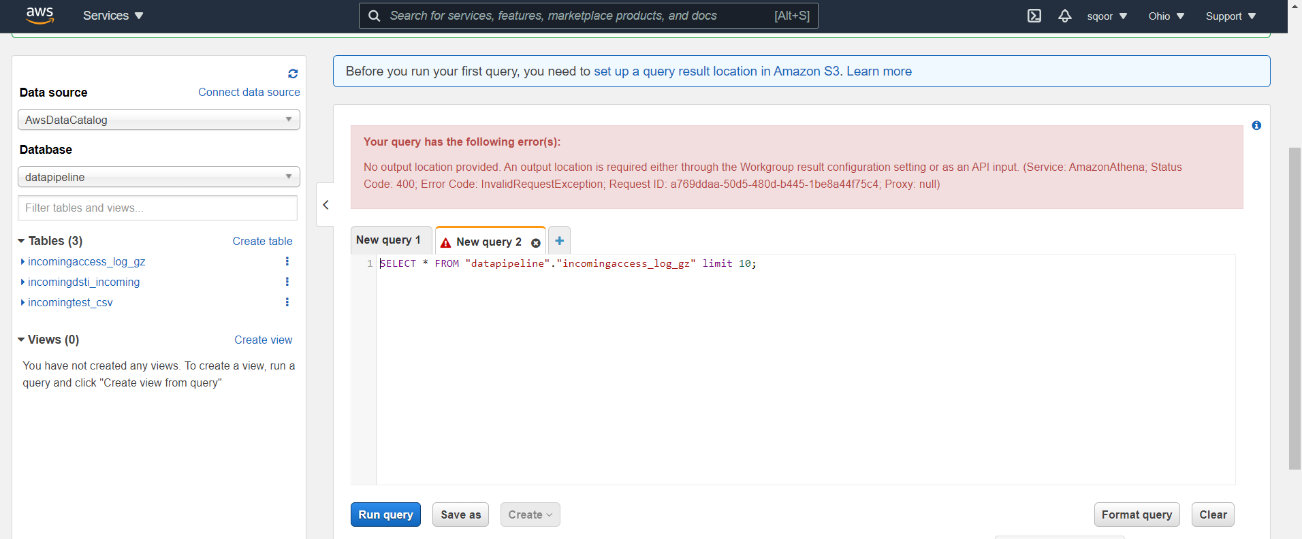


If we click on the csv table we can find that the data types of this file have been correctly interpreted:

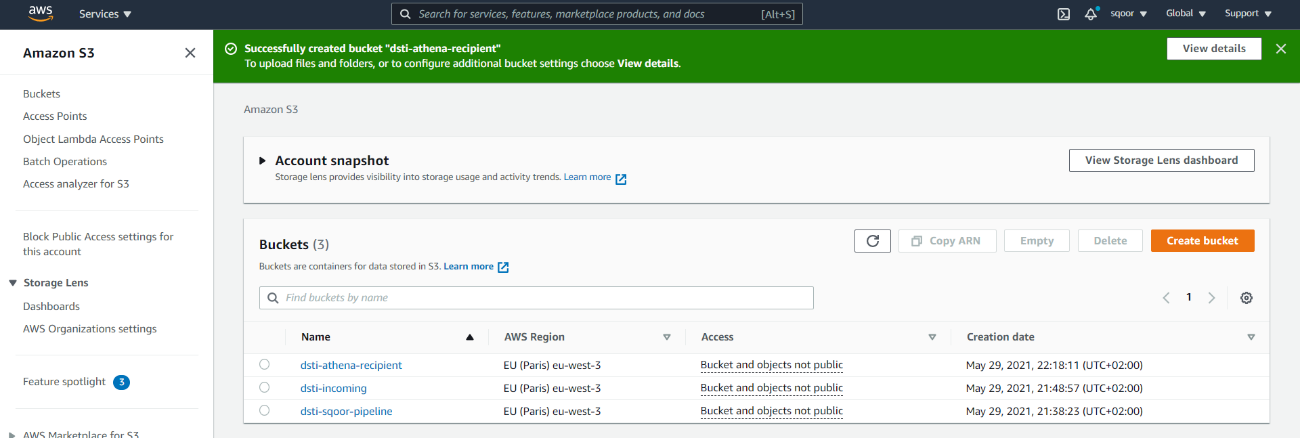
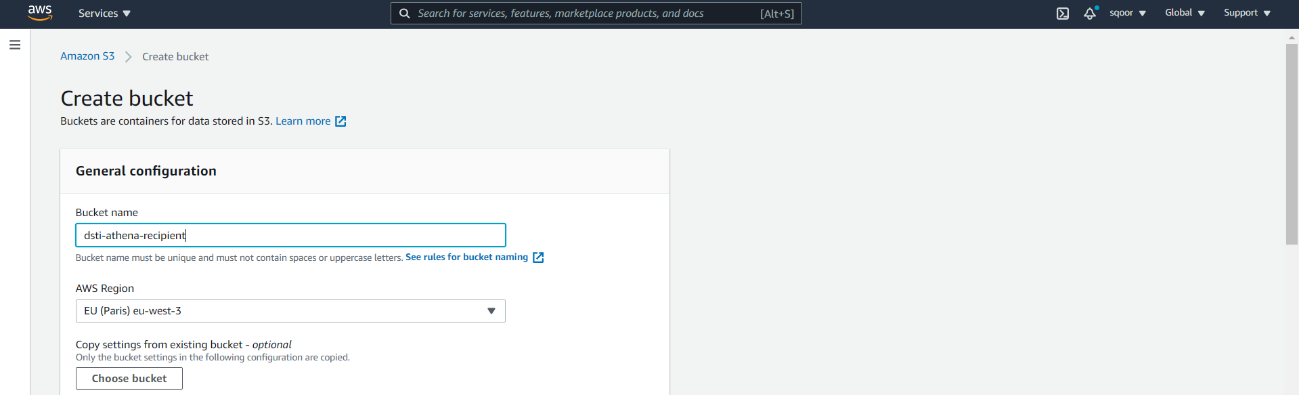


Now can go for the AWS Athena service, for querying. We select preview table from in

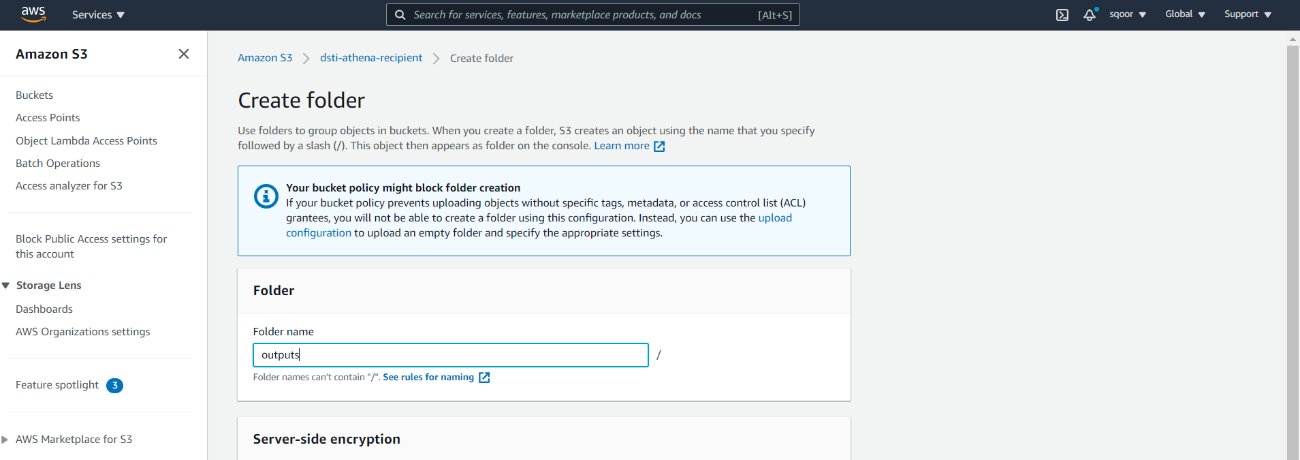
comingaccess\_log\_gz:



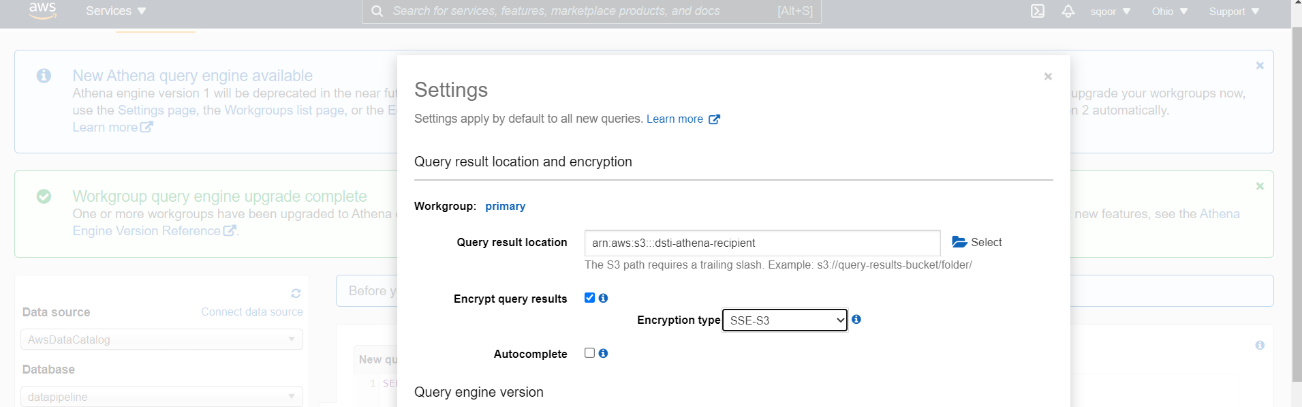
We need to create a final S3 bucket for becoming the recipient of the output of the queries:



We open it and add the folder outputs.



We copy the ARN newly S3 bucket created, go back to Athena and in settings we add the ouput destination:



After its creation, the following queries’ results will be outputted in that S3 folder.