**Transform and Import a JSON file into Amazon Redshift with AWS Glue**

Let’s say we have JSON file containing the past year of temperature data collected from IoT Sensors. One million rows of JSON. Let’s analyse the sensor data to predict and simulate the tweaks to our cooling systems that will compensate for the overheating.

**Sample Data**

[{

"timestamp": 1519516800,

"temperature": 26.7,

"sensor": {

"number": 4,

"location": ["-75.5712", "-130.5355"],

"address": "123 Main St, LAX, CA"

}

}, {

"timestamp": 1519517100,

"temperature": 29.8,

"sensor": {

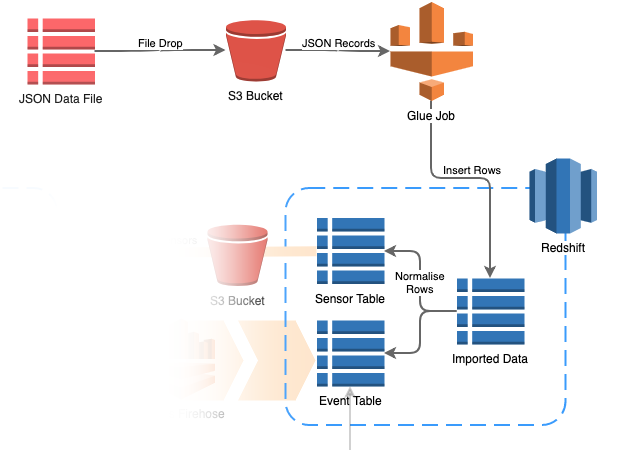
"number": 2,

"location": ["-48.8712", "-151.6866"],

"address": "456 Side St, SFO, CA"

}

}]



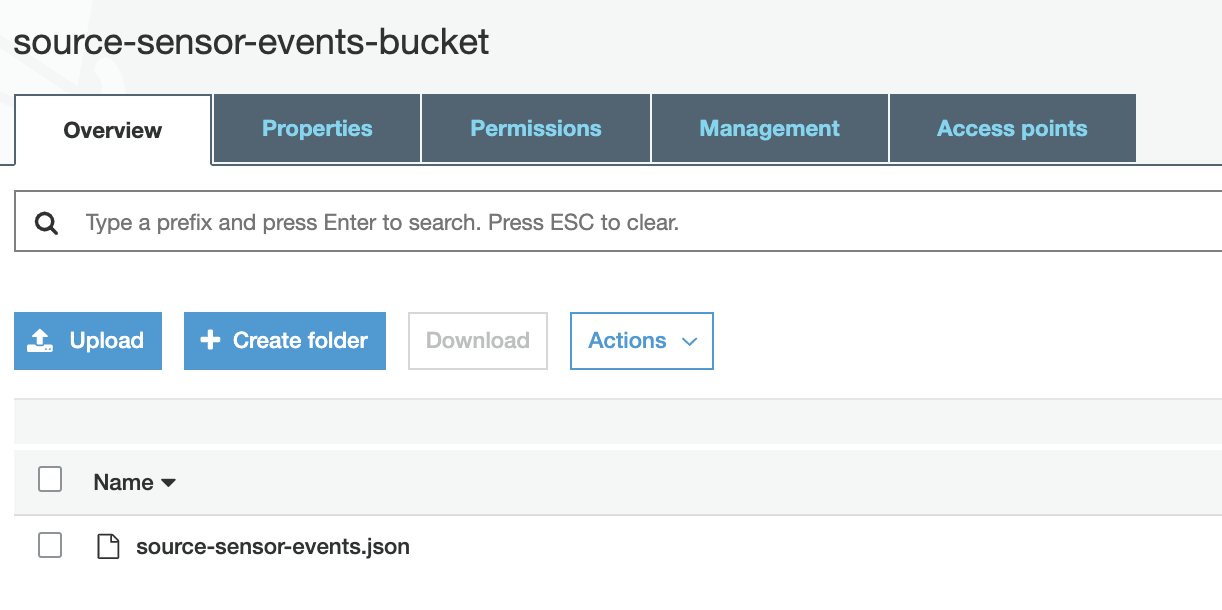
Loading a huge JSON file into Amazon Redshift doesn’t have to be difficult and disastrous. Just use AWS Glue and you’re all set to go!!!

1. **Build and maintain a JSON schema** automatically by crawling the JSON files.
2. **Create a Glue Job** in Python that maps JSON fields to Redshift columns.
3. Customize the Glue Job to **transform the columns**.

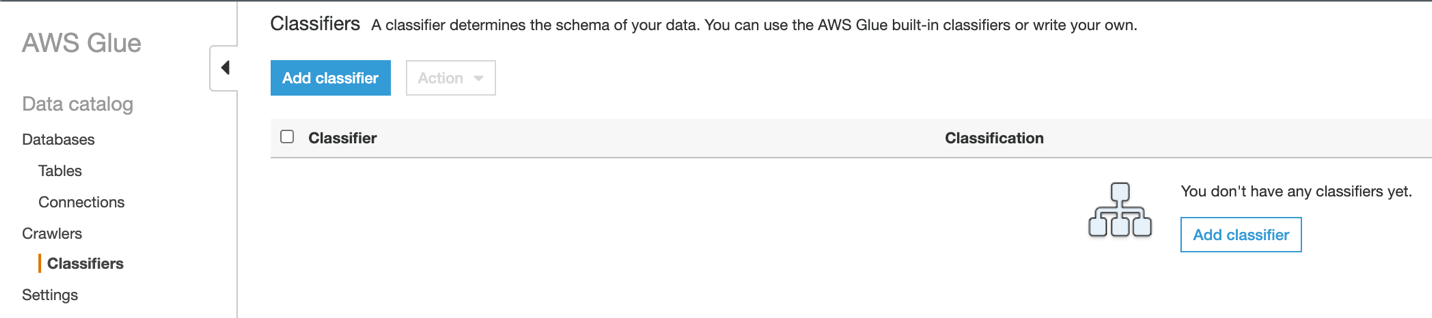
**Create a JSON Crawler**

We’ll start by creating a crawler for our JSON data file that will be stored in S3 bucket.

1. Browse to the S3 console. Create an S3 bucket (I called mine source-sensor-events-bucket)

****

1. Browse to the AWS Glue Console. In the AWS Glue Menu, click Classifiers 🡪 Add Classifier

****

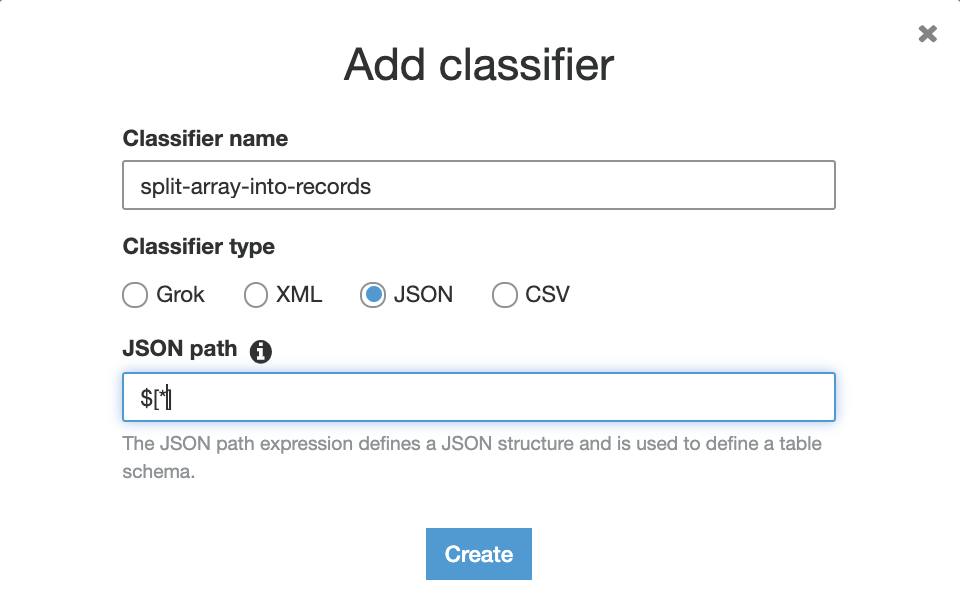
Set the Classifier Name as split-array-into-records

Set the JSON Path as $[\*]

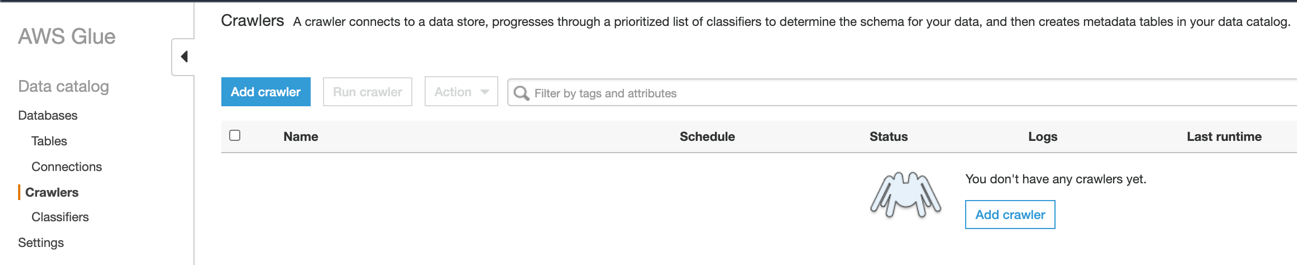
Click Create

Why do we need the Classifier? By default, AWS Glue loads the complete array of JSON records into a single Redshift row. Which isn’t really useful for analysis.

By using the Classifier $[\*] we lift the array elements up one level, so that each JSON record gets loaded into its own Redshift row.

****

1. In the AWS Glue menu, click Crawlers 🡪 Add Crawler

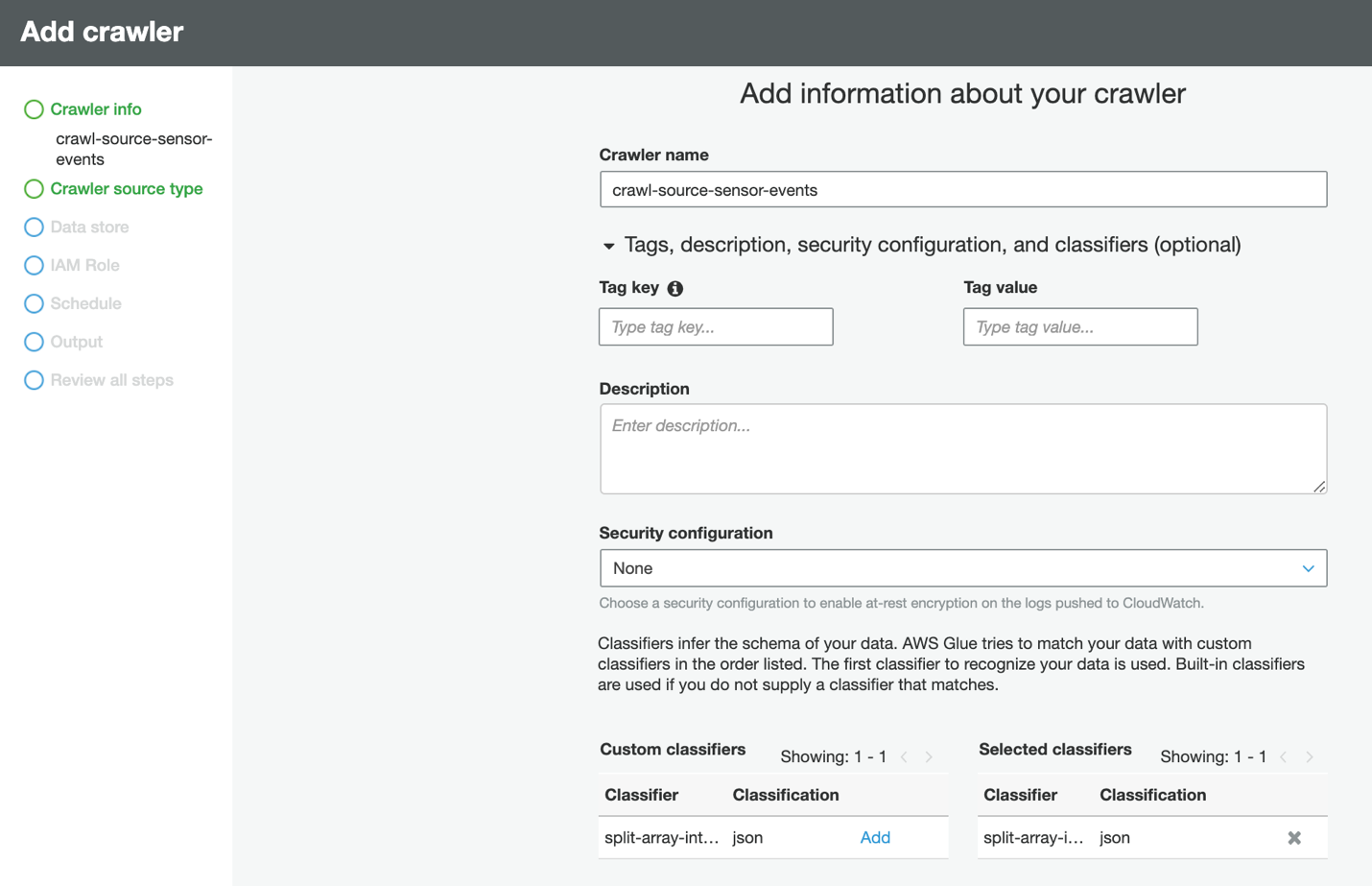


Set the Crawler Name to crawl-source-sensor-events

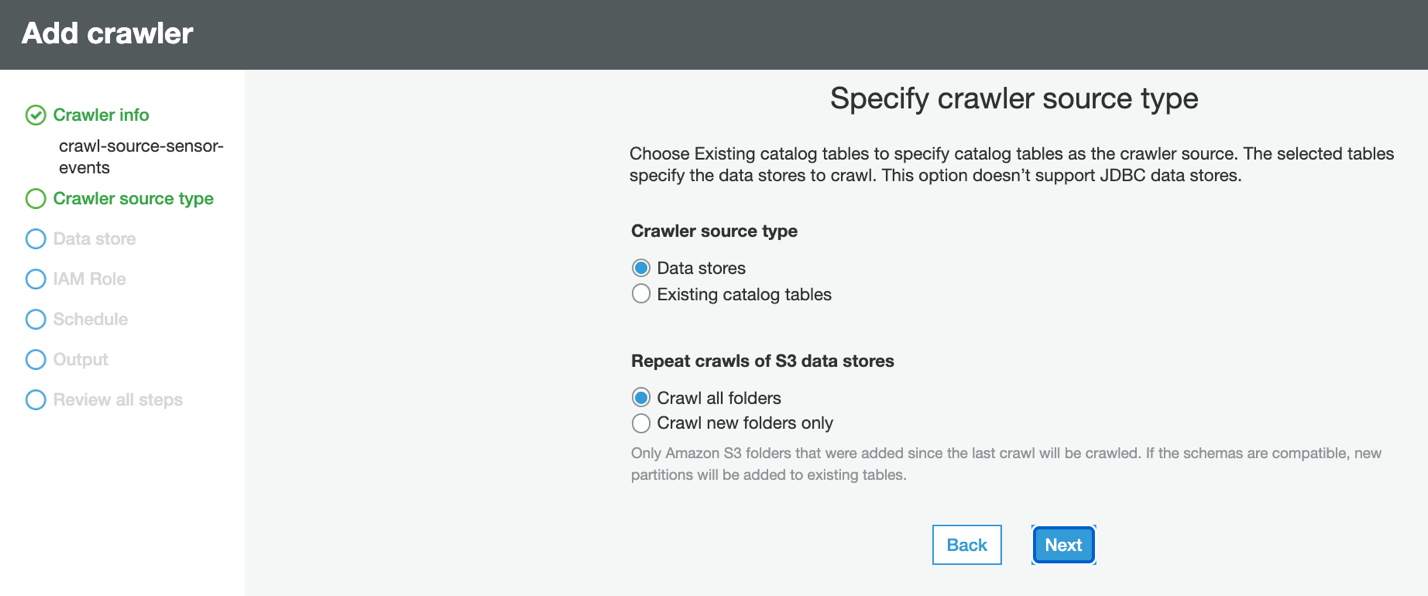
Under Custom Classifiers, click Add

split-array-into-records should appear in the Selected Classifiers column at lower right

Click Next



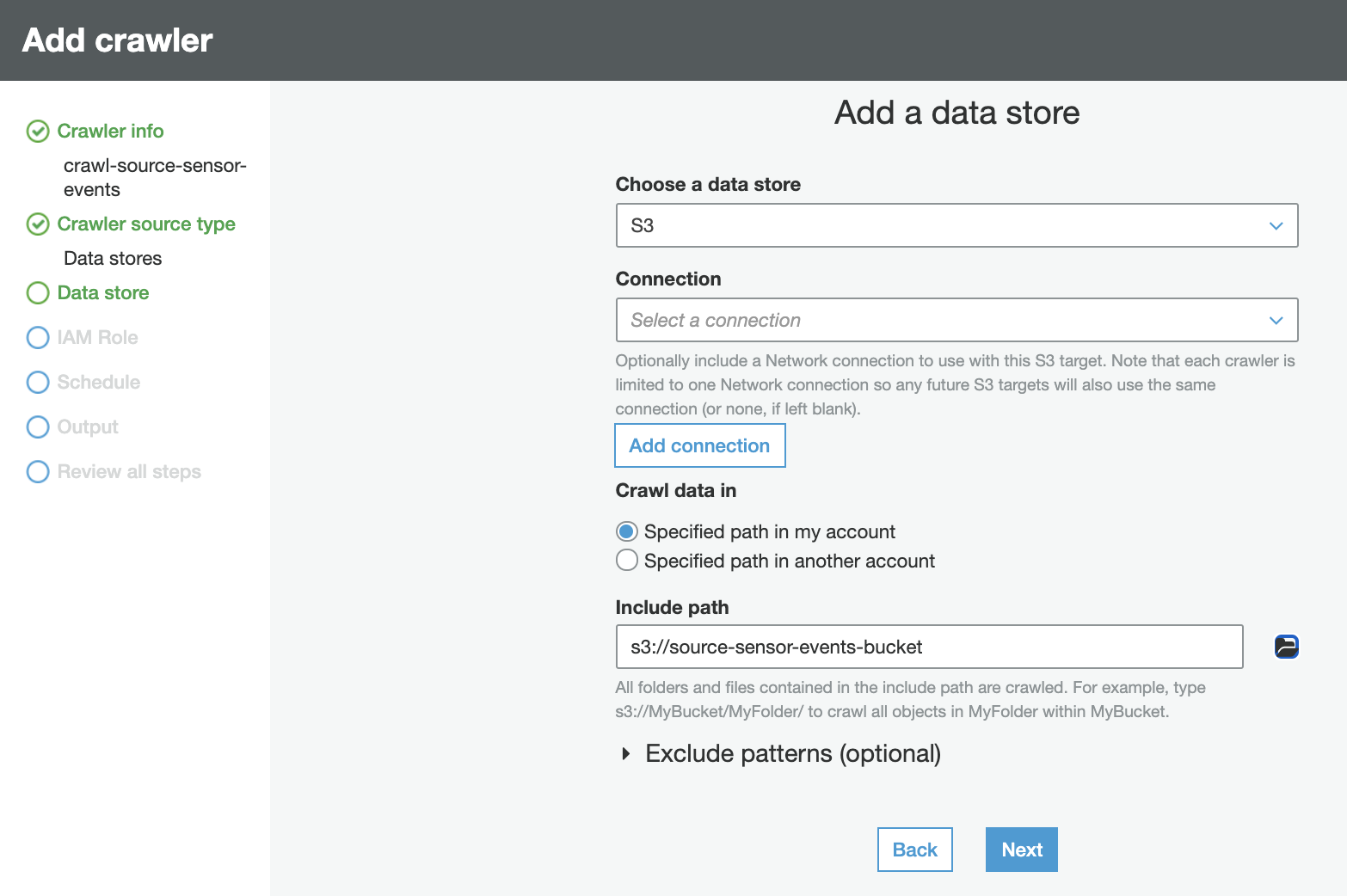
1. Specify crawler source type



1. Select S3 as the Data Store

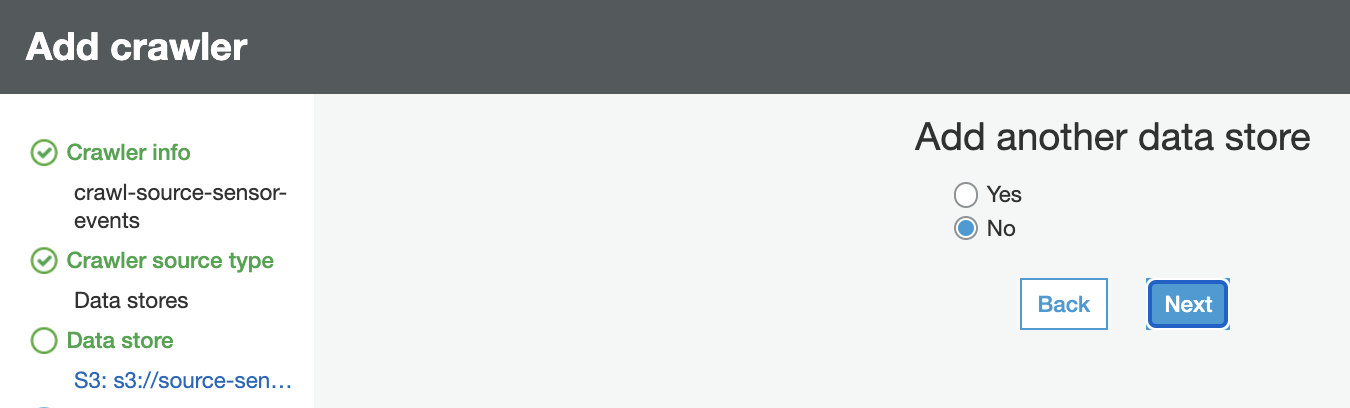
Set Include Path to the S3 bucket we created, e.g. source-sensor-events-bucket

Click Next



When prompted to add another data store, select No

Click Next

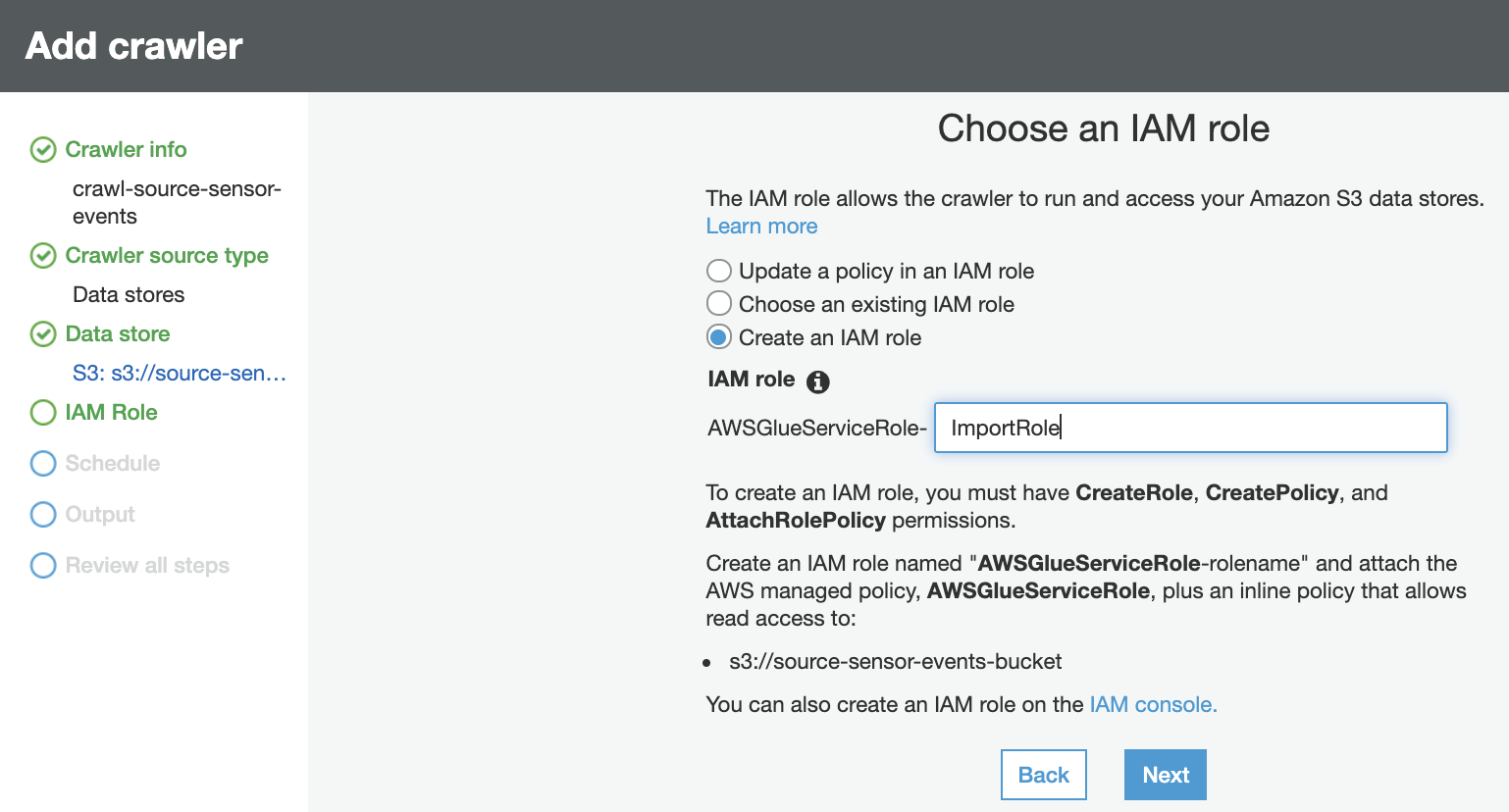


1. Select Create an IAM Role

Key in a name for the new Role, like AWSGlueServiceRole-ImportRole

AWS Glue will create this Role and grant it read permission for our bucket

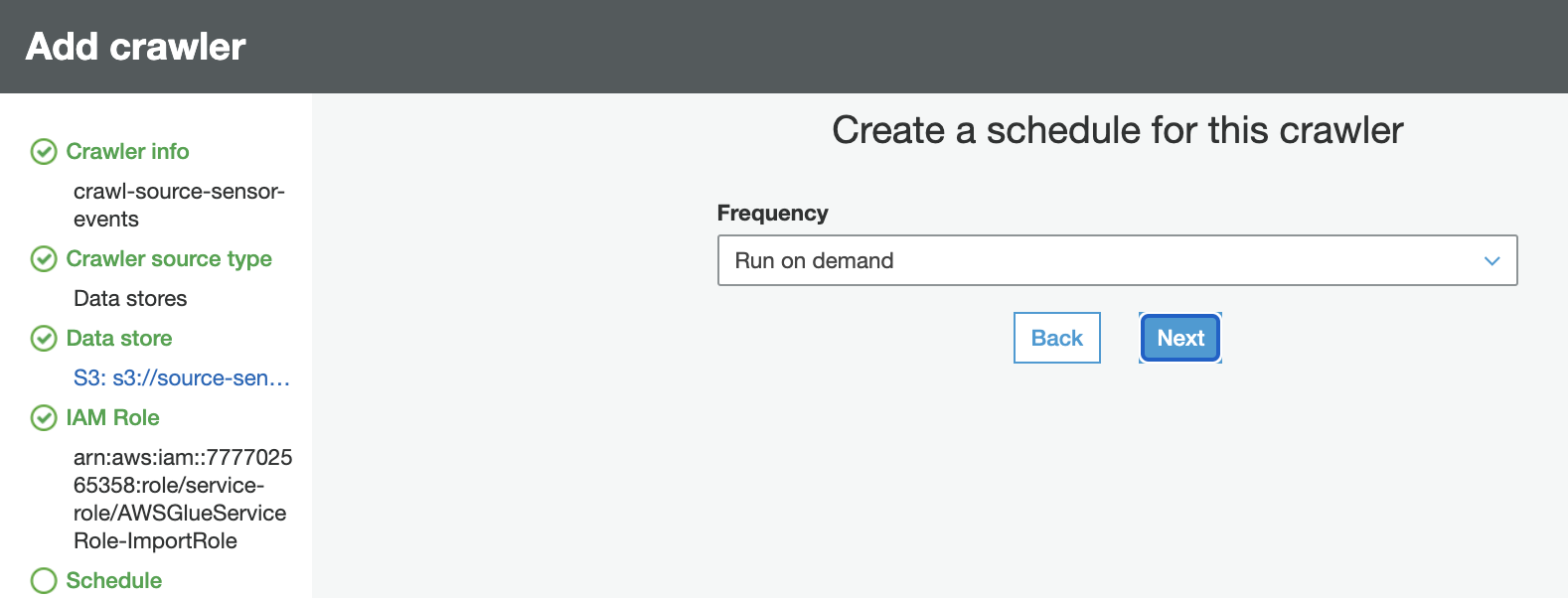
Click Next



1. Set the Crawler Frequency to Run on demand.

The Crawler runs only when we start it ourselves.

Click Next

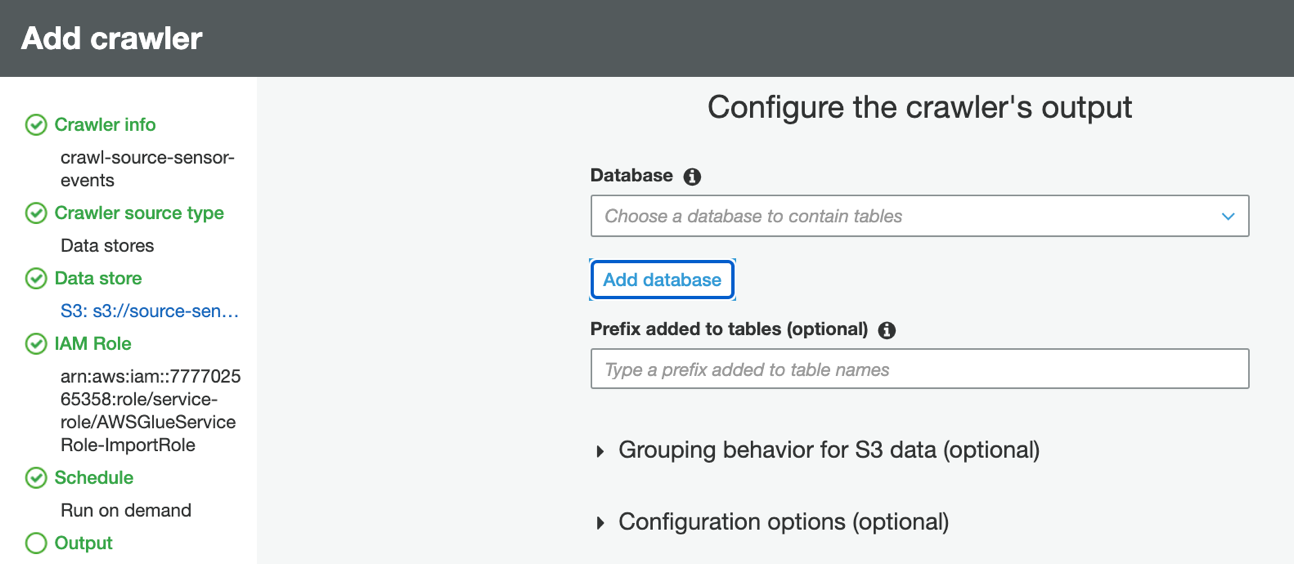


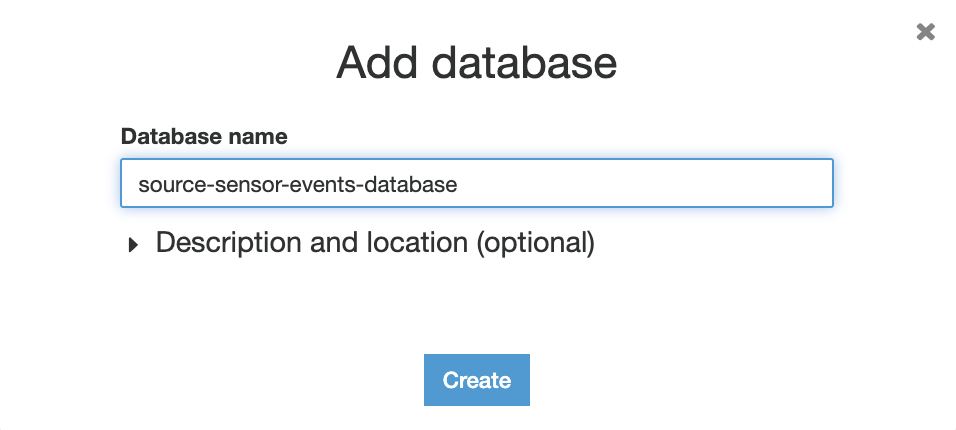
1. For the Crawler Output, click Add database

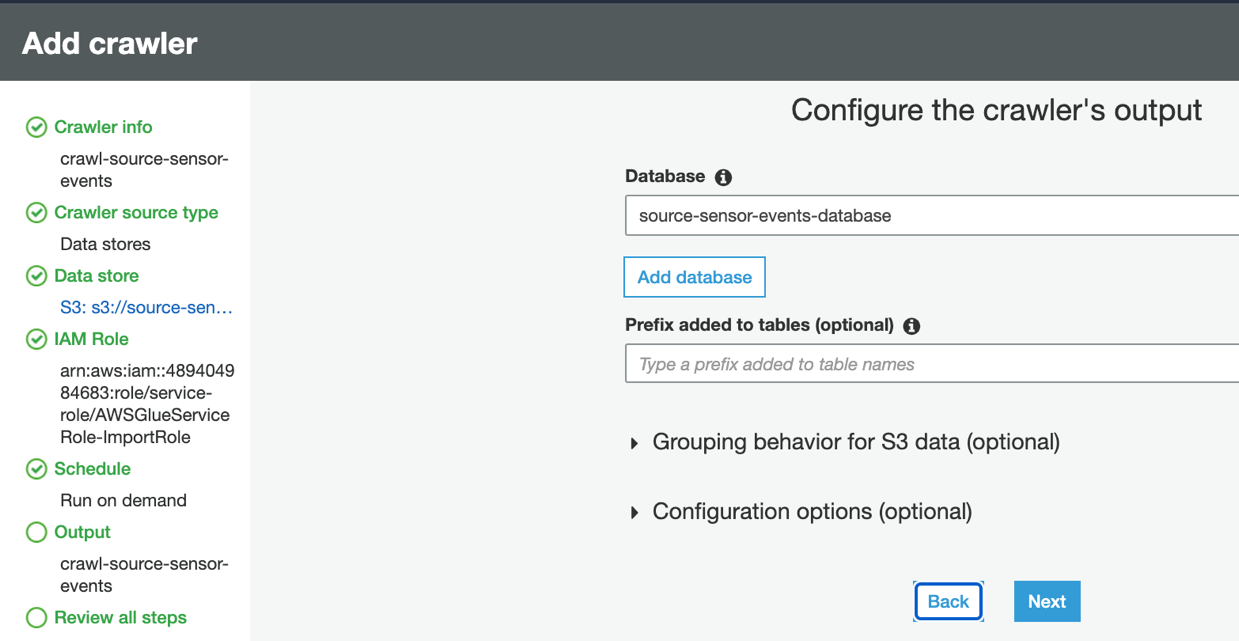
We’ll create a Glue Database that will store the schema deduced by the crawler from the JSON file.

Set the Database name as source-sensor-events-database

Click Create

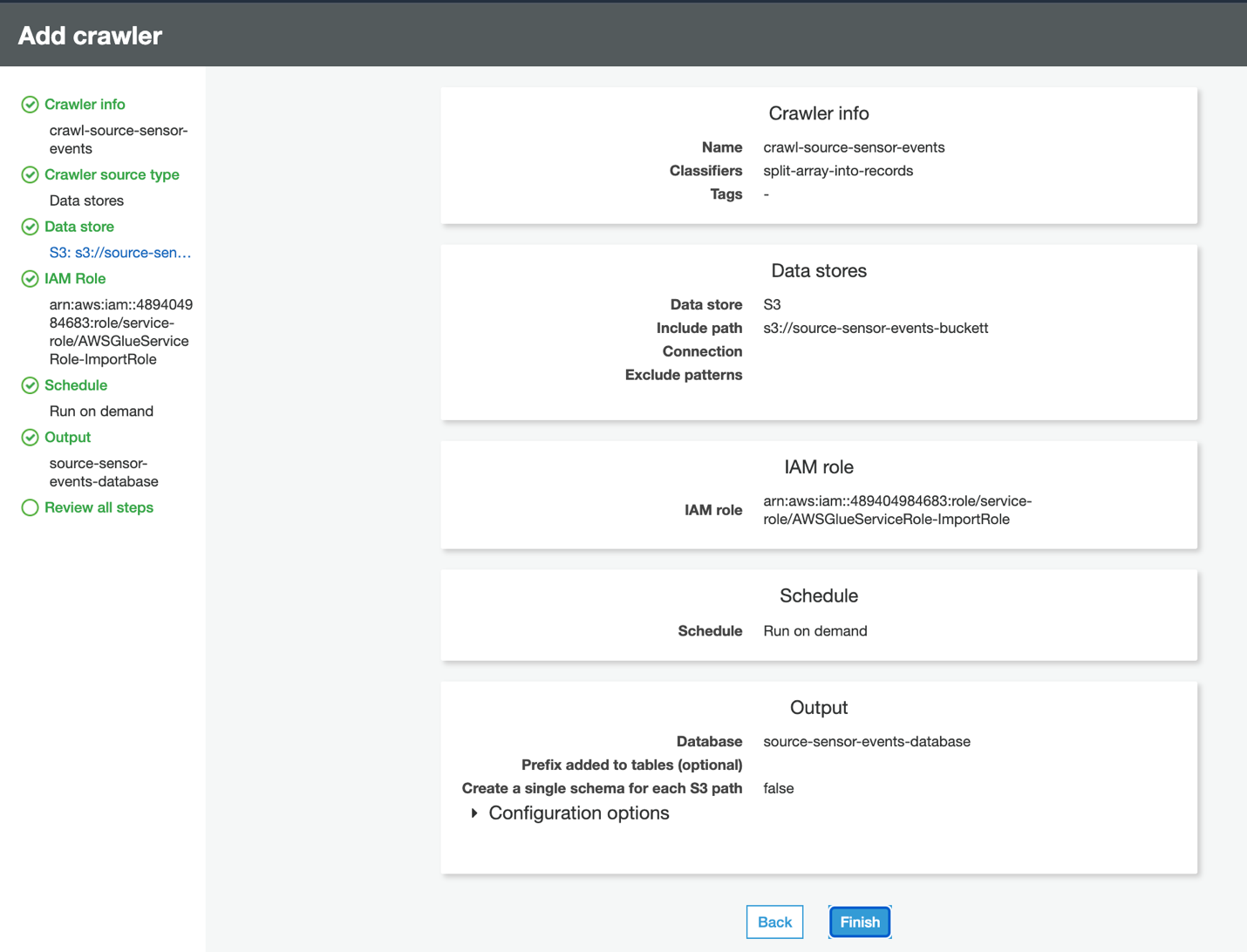






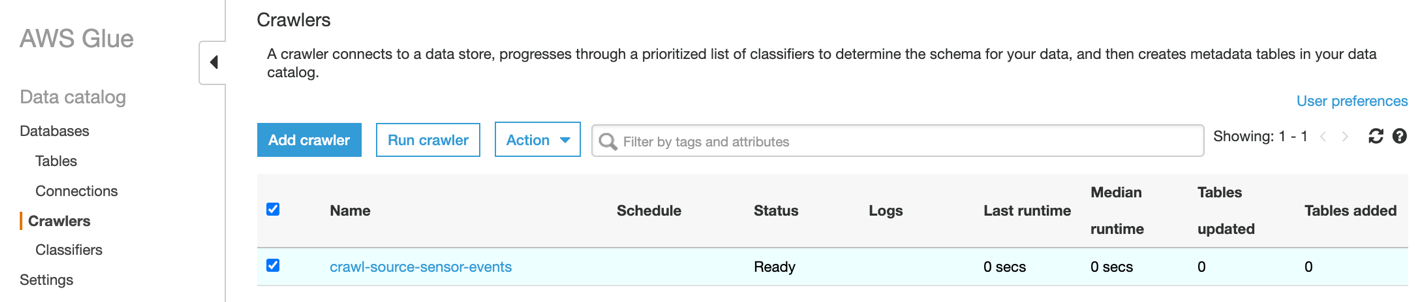
Review all the steps. Once reviewed click Finish.

AWS Glue Crawler has been created that will crawl the records in our JSON file and deduce the schema.



**Let’s test our JSON Crawler**

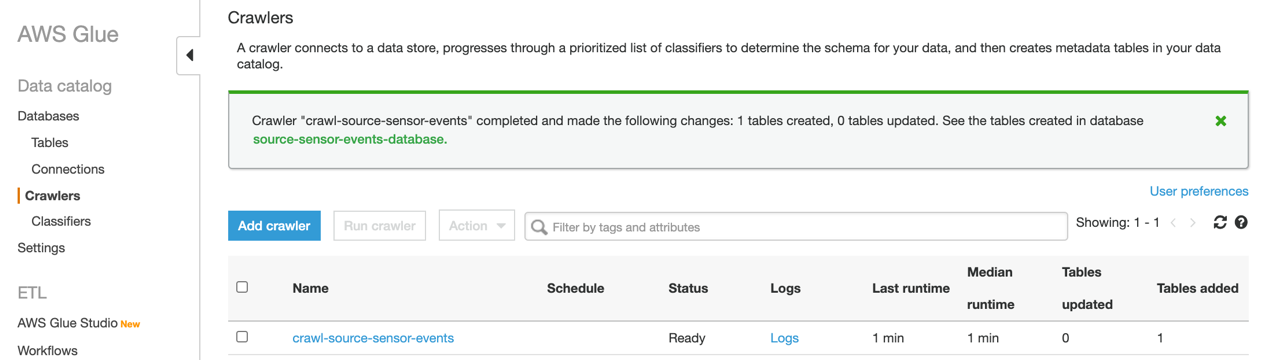
1. After creating the Crawler, select the crawler and click Run Crawler



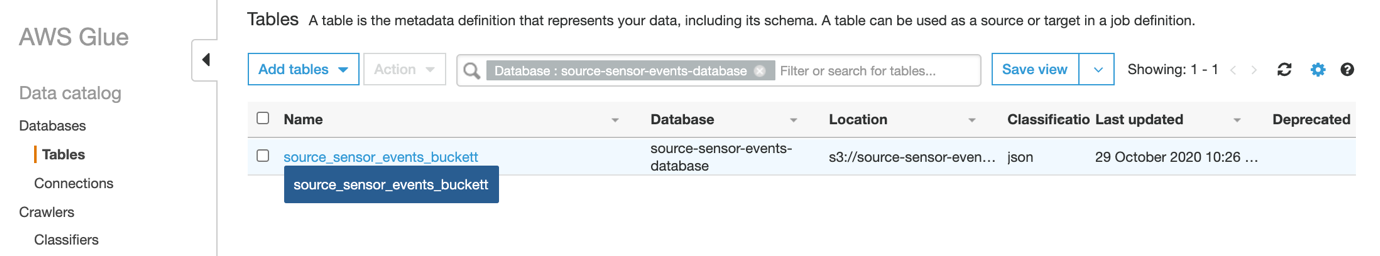
1. In a minute or two, AWS Glue will notify us that the crawling has been completed.

Click the link source-sensor-events-database

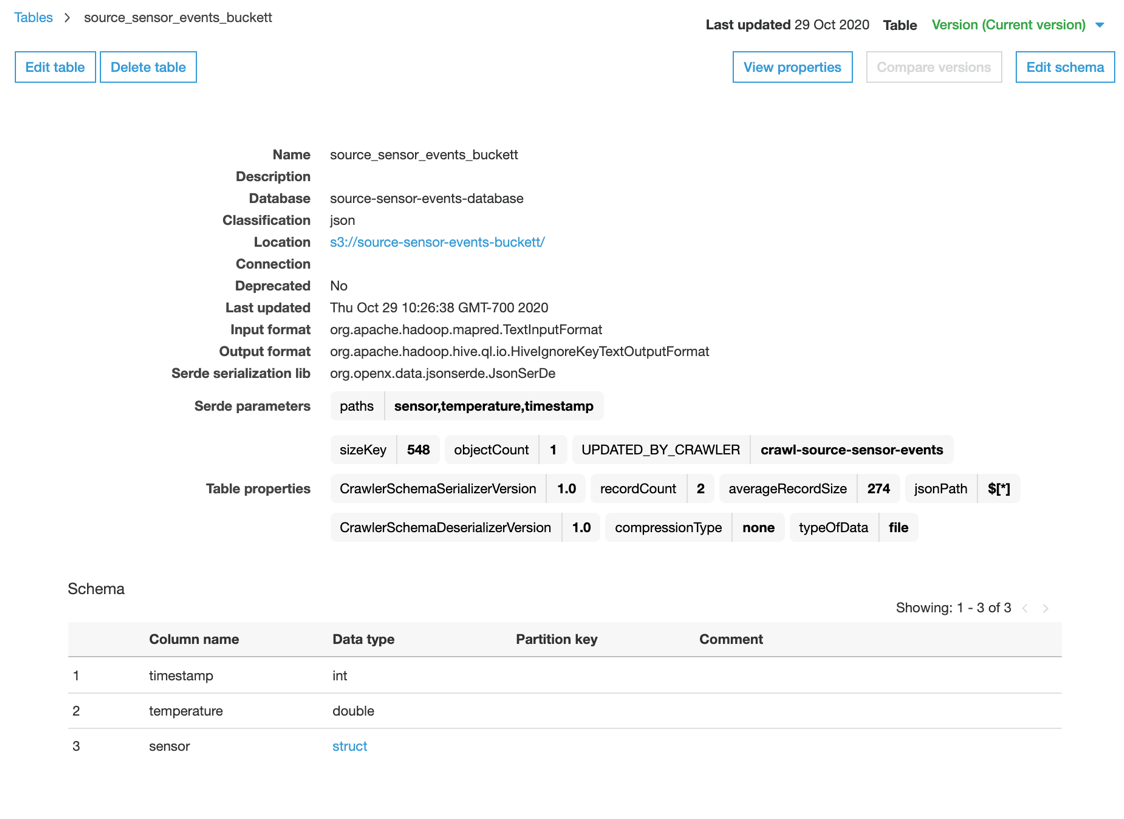
Will check whether the JSON fields were deduced correctly.



1. Click source\_sensor\_events\_bucket



This is the Glue Table that the Crawler has created for storing the schema of our JSON file.



1. This shows that 2 records have been crawled. The schema deduced by the crawler is shown at the bottom. The deduced schema matches our JSON file structure.

Note that the schema shows individual fields like timestamp and temperature at the top level.

If we see instead an array field, then most likely we have omitted the Classifier split-array-into-records from our crawler. Delete the Glue Database, fix the Crawler and crawl again.

[{

"timestamp": 1519516800,

"temperature": 26.7,

"sensor": {

"number": 4,

"location": ["-75.5712", "-130.5355"],

"address": "123 Main St, LAX, CA"

}

}, {

"timestamp": 1519517100,

"temperature": 29.8,

"sensor": {

"number": 2,

"location": ["-48.8712", "-151.6866"],

"address": "456 Side St, SFO, CA"

}

}]

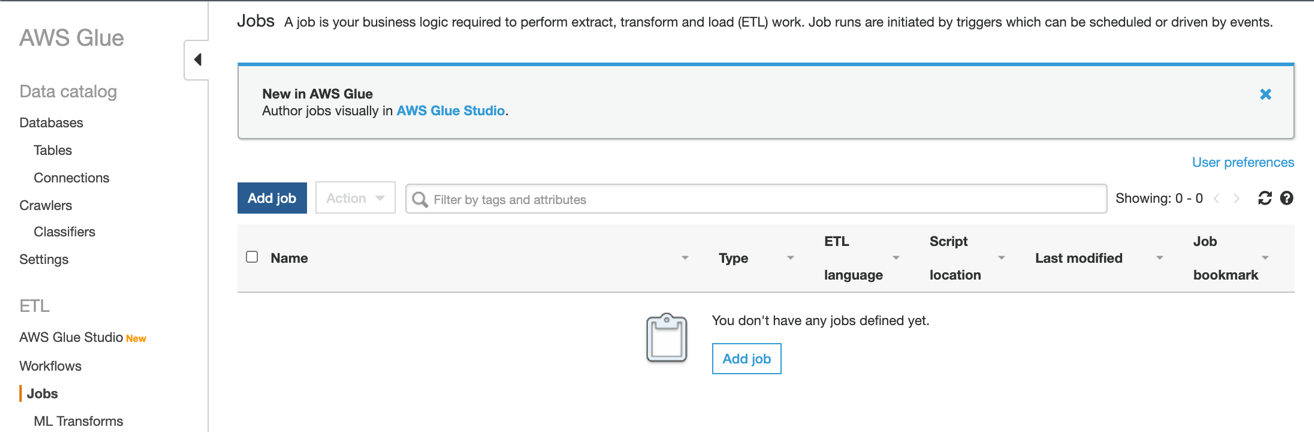
Let’s look closer at the JSON schema. At the top level of each JSON record we have the timestamp and the sensor value (temperature). Each record contains a nested object sensor that describes the sensor that recorded the value.

The sensor object may be duplicated in multiple JSON records. If we normalize the data, each JSON record should be split into two Redshift tables: Sensor Table and Event Table

But this kind of denormalized JSON data is common for distributed logging systems. We won’t be doing normalization as of now will create a Glue Job to read the JSON records and write them into single Redshift table.

Create an Import Job

1. In the AWS Glue menu, click Jobs 🡪 Add Job

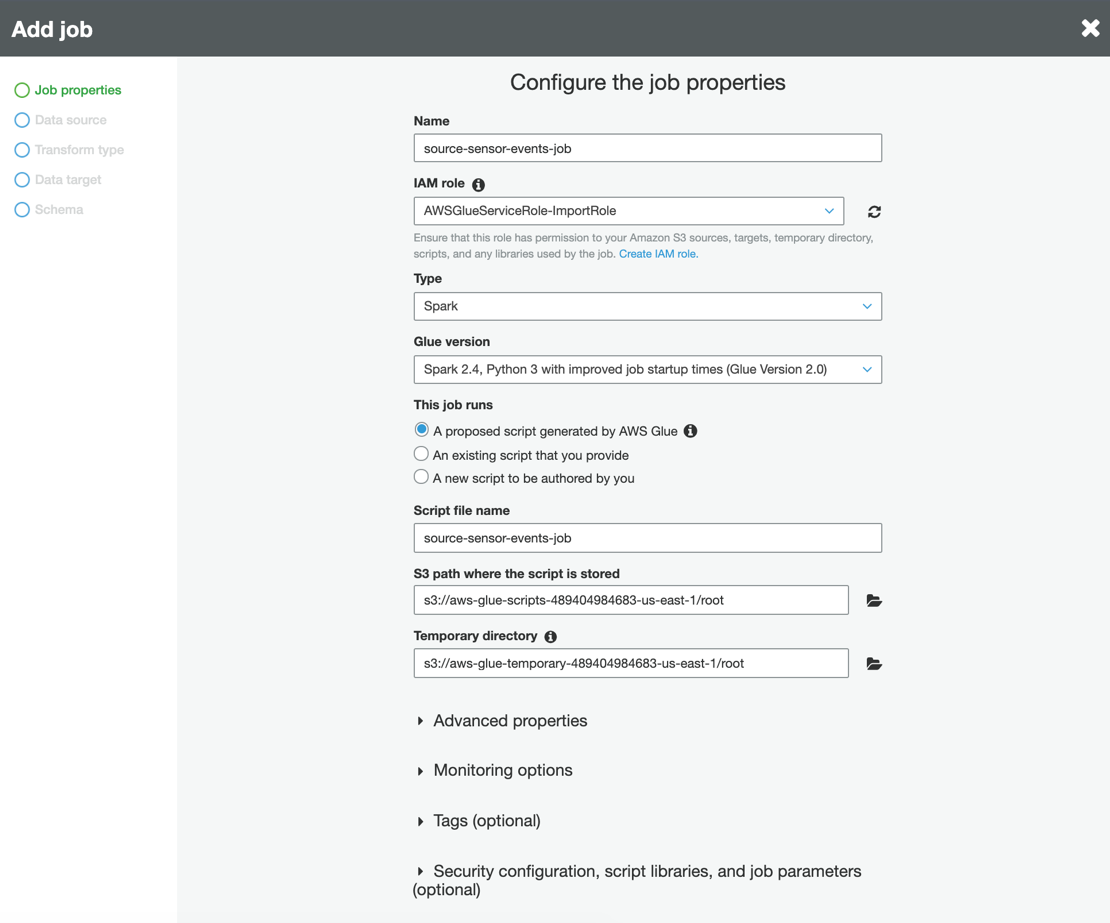


1. Set Name to source-sensor-events-job

Select the IAM Role that we created earlier: AWSGlueServiceRole-ImportRole

Click Next

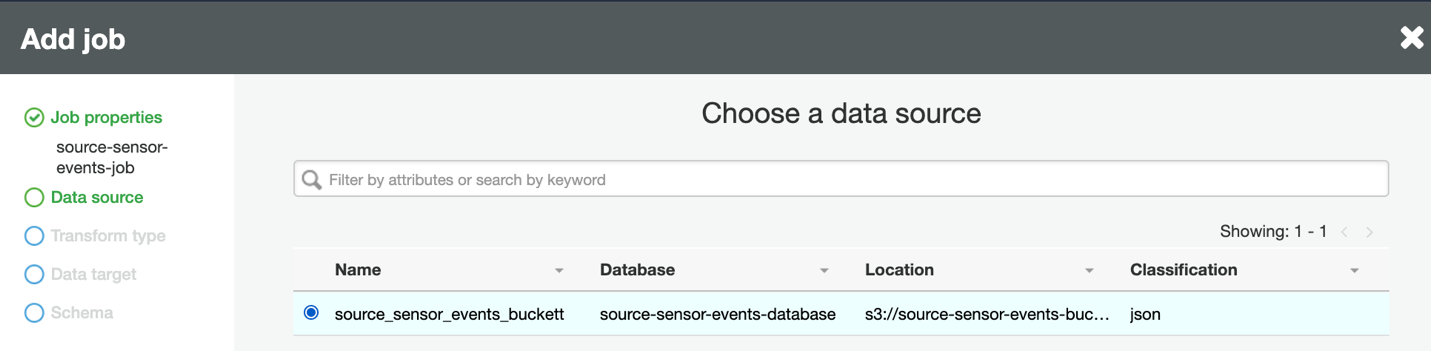
Note: Please make sure you’re selecting the correct version of python or the job would fail.



1. For the Data source, select the table that contains our crawled JSON schema:

source\_sensor\_events\_bucket

Click Next



1. For Transform type, select Change schema

Click Next

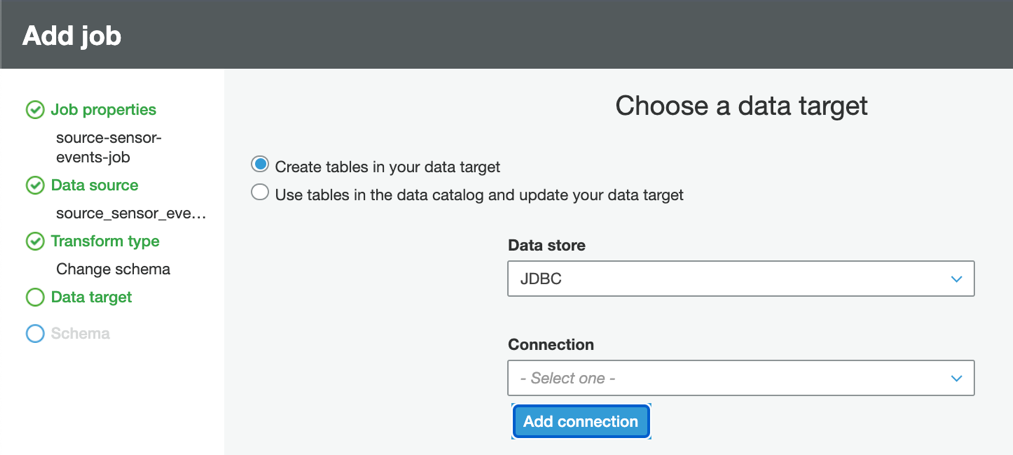


1. For Data target, select Create tables in your data target

Select Data store as JDBC

Click Add connection

We’ll now add the Redshift data connection



1. We assume that the Redshift cluster and database have been set up. We’ll enter the Redshift connection details, which may be obtained from the Redshift Console.

Set Name as redshift

Set Connection type as Amazon Redshift

Fill in the Redshift connection details for your Redshift cluster and database:

Cluster, Database Name, Username and Password

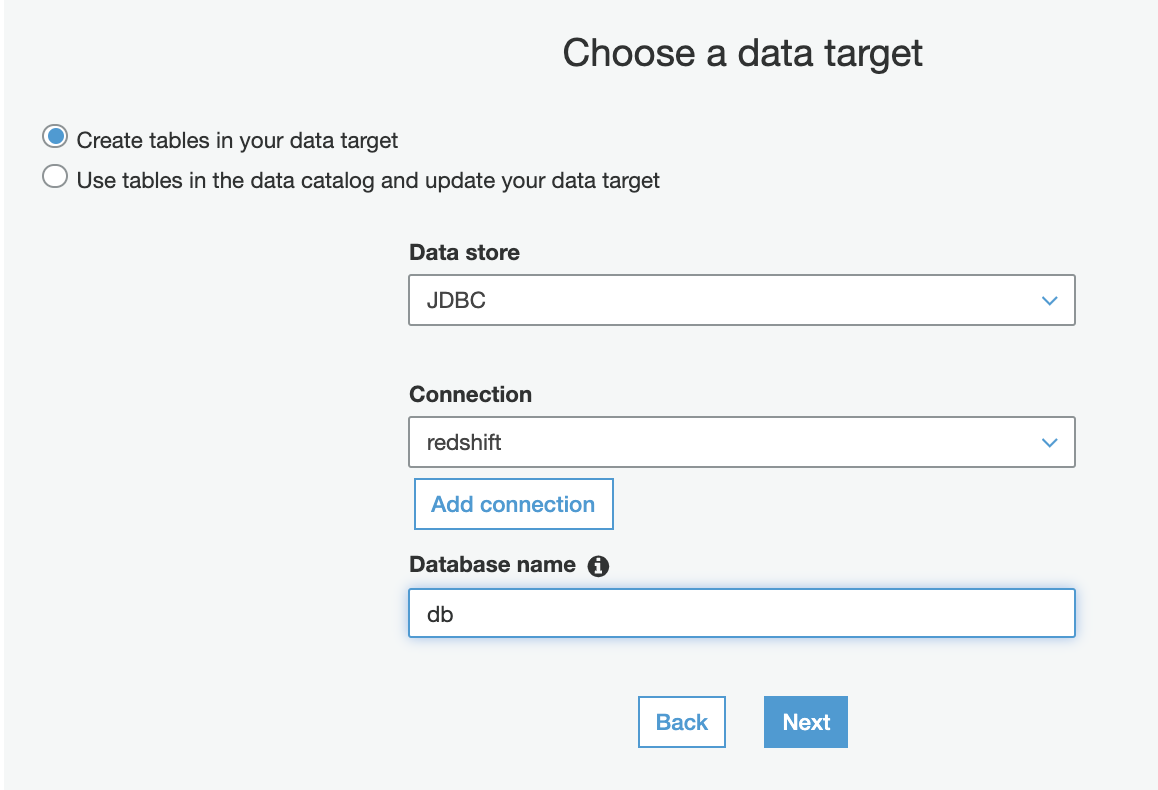
Click Add



1. Set Connection to redshift

Set Database name to the name of your Redshift database

Click Next



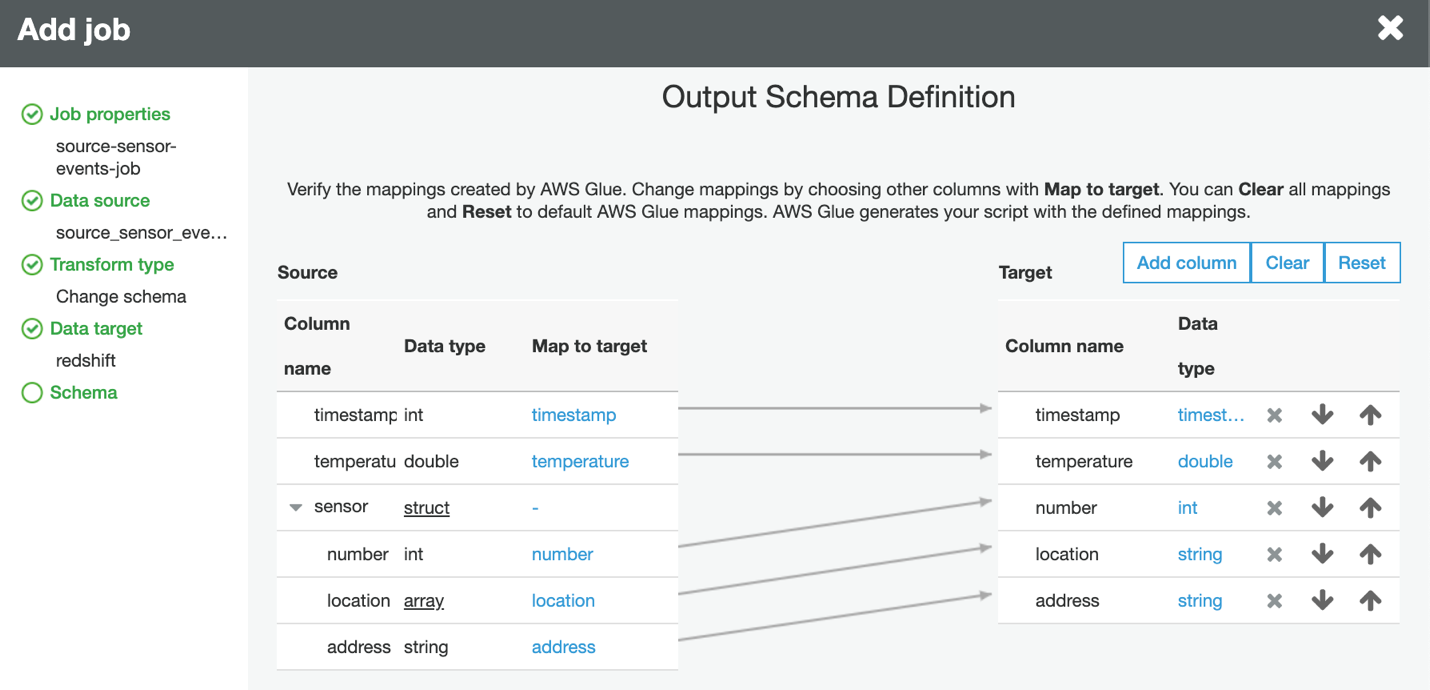
1. Let’s map the crawled JSON fields into the Redshift columns. At the right side of the screen (Target), click to edit the following mappings.

Timestamp column: Change the type from int to timestamp

sensor.number column: Rename as number

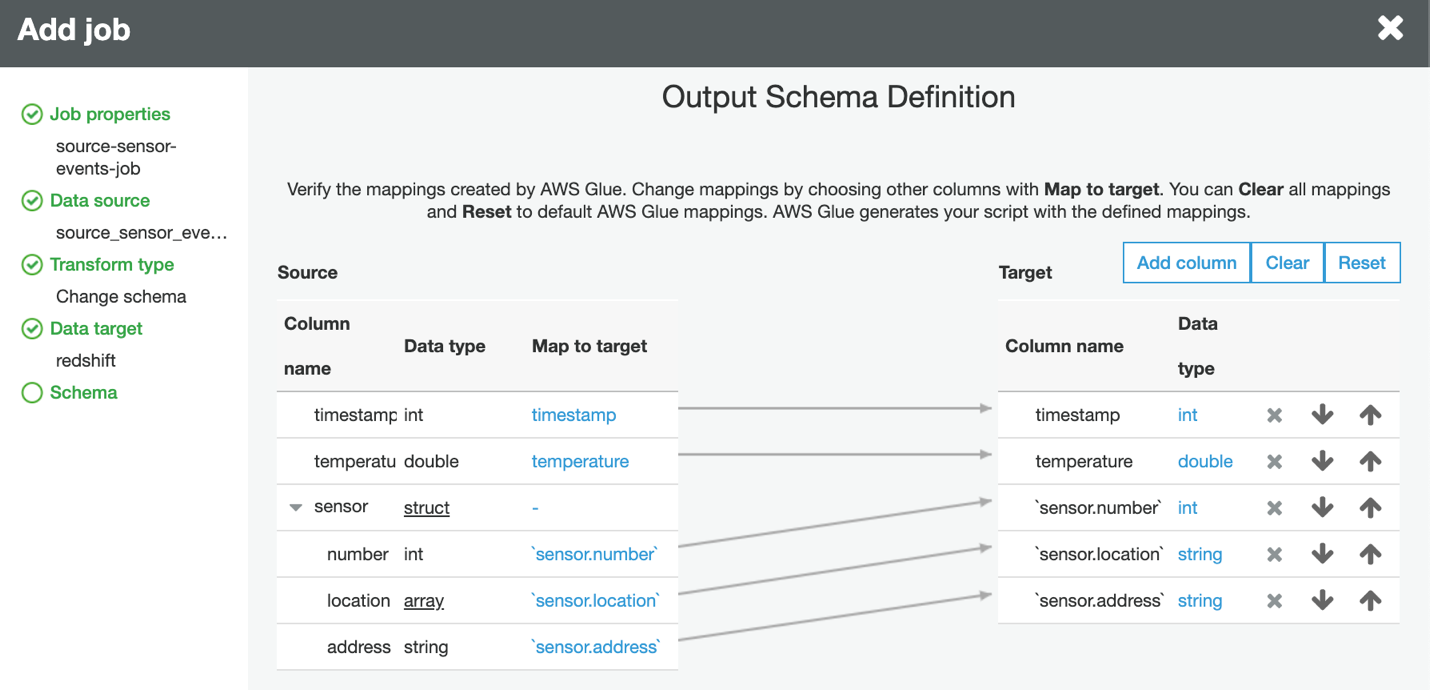
sensor.location column: Rename as location

sensor.address column: Rename as address



After editing, the mappings should look like below

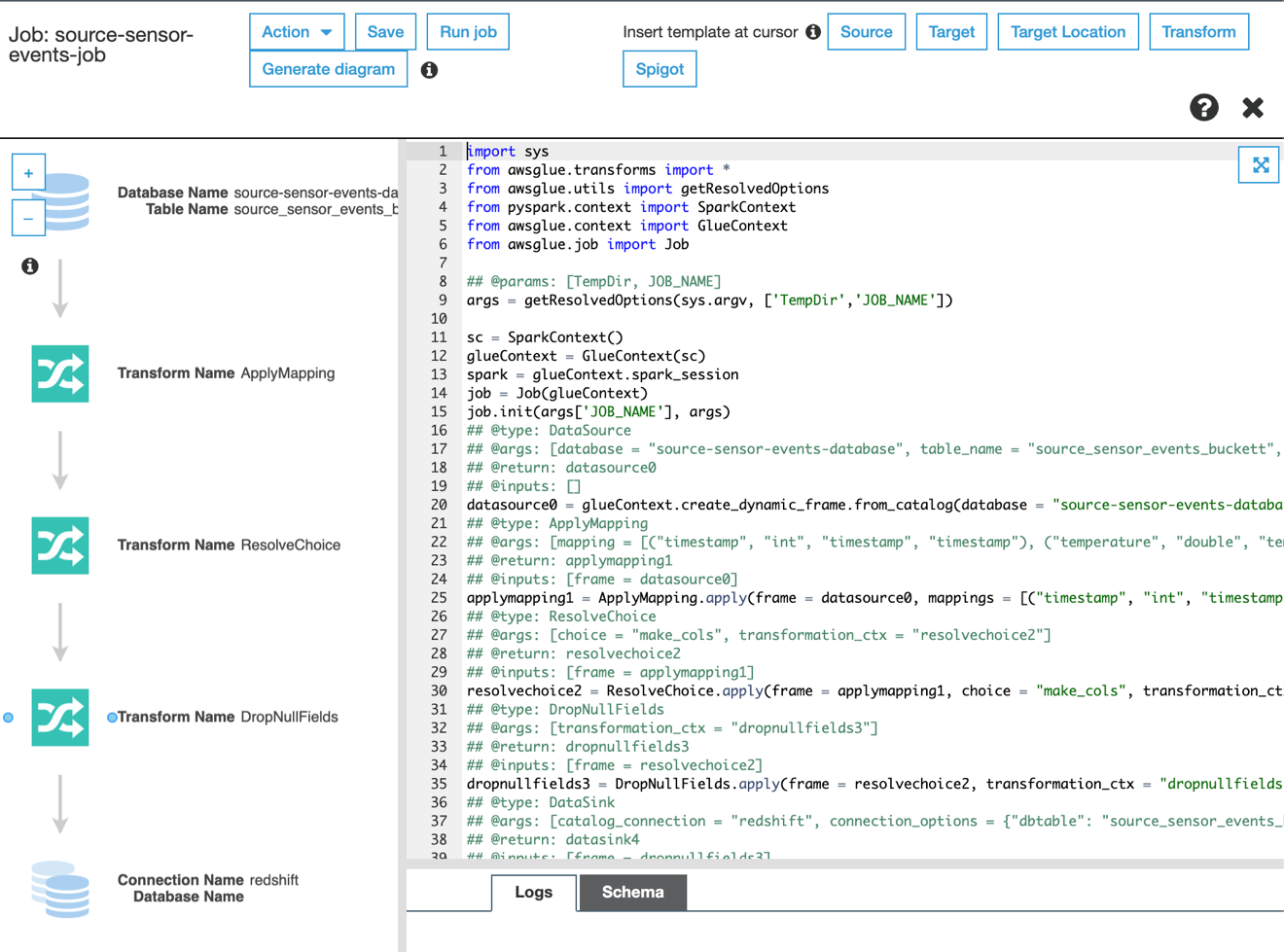
Click Save job and edit script



1. Our Import Job source-sensor-events-job has been created!

AWS Glue took all the inputs from the previous screens to generate this Python script, which loads our JSON file into Redshift.

In the real scenario, importing JSON data into Redshift isn’t straightforward. We may need to split some fields, aggregate some fields or perform some computation. Let’s customize the Import Job to try out.



**Customize our Import Job**

1. At the top of the Python script, right after the import section, insert the following import declarations.

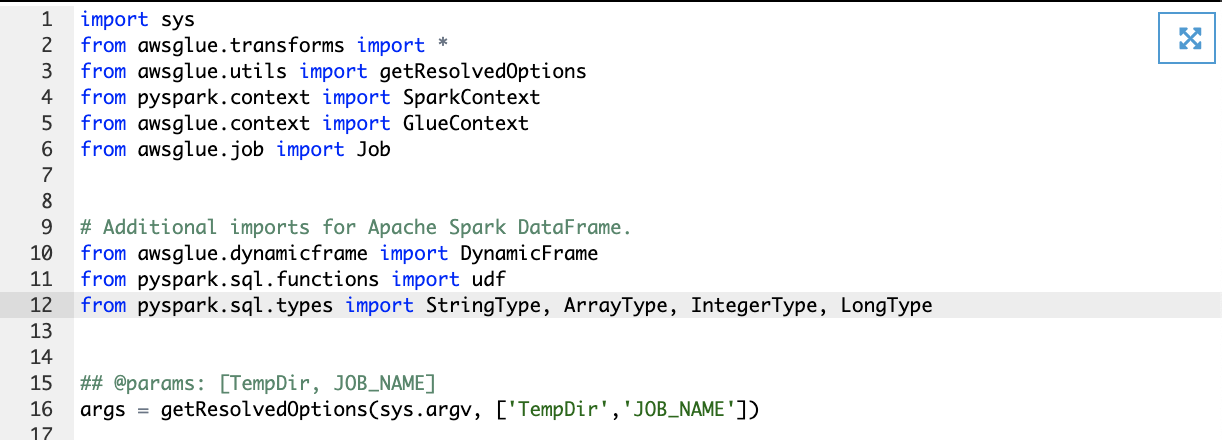
# Additional imports for Apache Spark DataFrame.

from awsglue.dynamicframe import DynamicFrame

from pyspark.sql.functions import udf

from pyspark.sql.types import StringType, ArrayType, IntegerType, LongType

After inserting the import declarations, the script should look like below



1. In the script, look for the ApplyMapping section



Replace the ApplyMapping section with this code applymapping.py

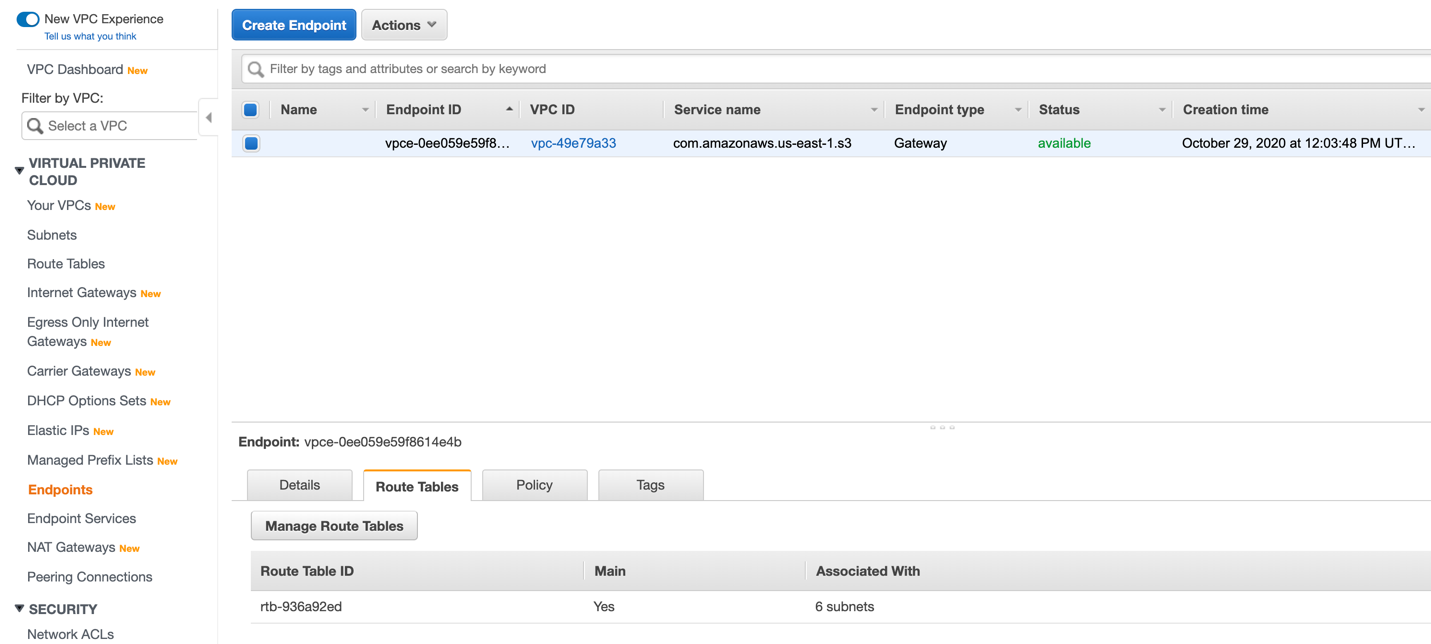
After replacing the ApplyMapping section, the script should look like this.

Click Save in the top button bar.

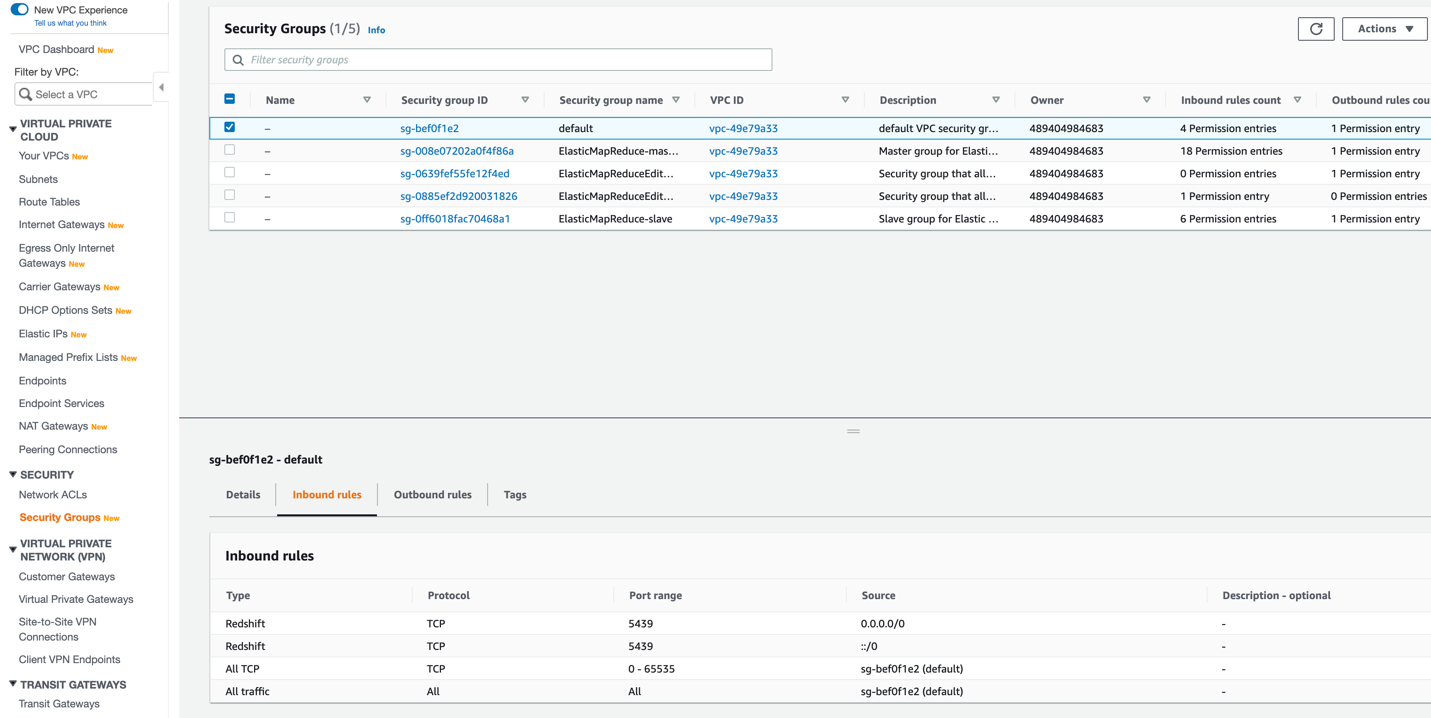
We have just added some transformations to the Import Job.

**Test our Import Job**

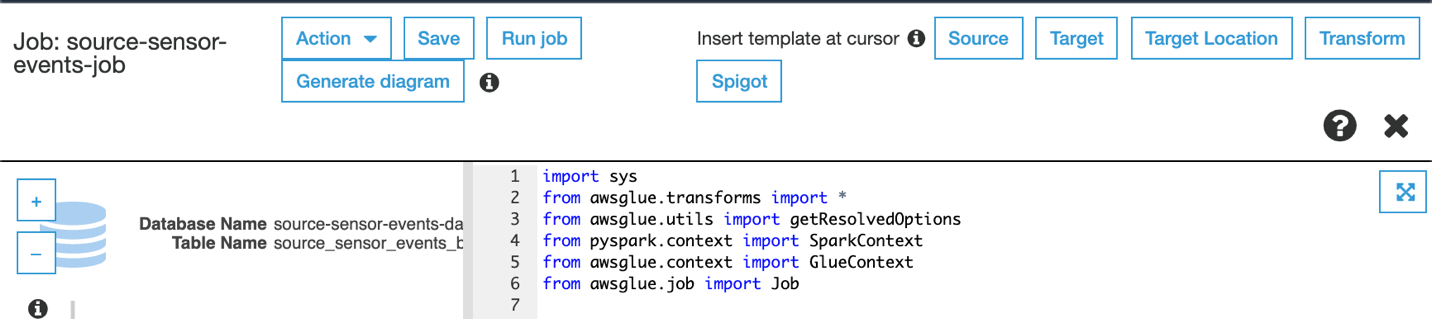
1. Before running our import job, we need to configure the AWS Virtual Private Cloud (VPC), Security Groups (i.e. which TCP ports to allow traffic netween networks) and Route Tables for AWS Glue and Redshift.



When configured, your VPC settings should look similar to the screens shown on this page.



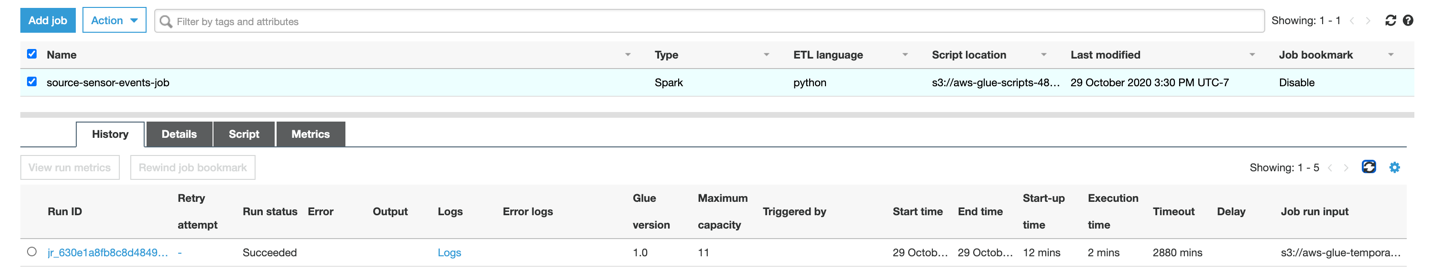
1. Click Save and Run Job in the top button bar



1. In the AWS Glue Menu, click Jobs 🡪 source-sensor-events-job

When the Import job has completed successfully, we should see Succeeded in the Run Status column.

The import job may take a couple of minutes to run.



1. In case of error, mouse over the ❗️ icon. The error message appears.

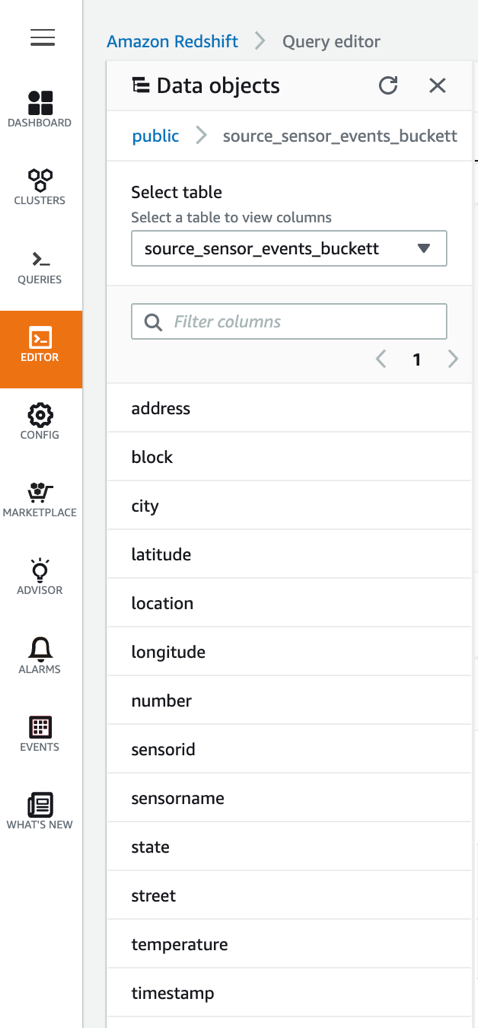


1. When the import job has completed successfully, head over to the Redshift console

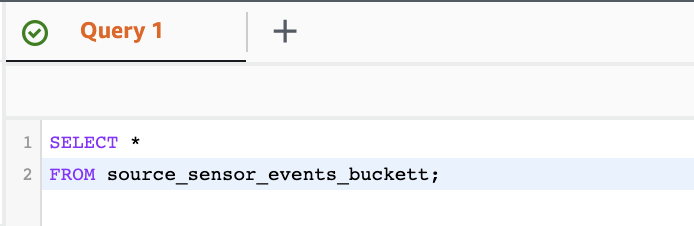
Click Query Editor

Look for the table source\_sensor\_events\_bucket

We should see the same columns as in the screenshot



1. Select the records from the table source\_sensor\_events\_bucket





We see the JSON fields loaded correctly into the timestamp, temperature, number, location and address columns.