

# HOW TO CONNECT AMAZON GLUE TO A JDBC DATABASE



Here we explain how to connect Amazon Glue to a Java Database Connectivity (JDBC) database.

The reason you would do this is to be able to run [ETL jobs](#) on data stored in various systems. For example, you could:

- Read .CSV files stored in S3 and write those to a JDBC database.
- Write database data to Amazon Redshift, JSON, CSV, ORC, Parquet, or Avro files in S3.
- Once the JDBC database metadata is created, you can write Python or Scala scripts and create Spark dataframes and Glue dynamic frames to do ETL transformations and then save the results.
- Since a Glue Crawler can span multiple data sources, you can bring disparate data together and join it for purposes of preparing data for machine learning, running other analytics, deduping a file, and doing other data cleansing. However, that is limited by the number of Python packages installed in Glue (you cannot add more) in GluePYSpark.

In this tutorial, we use [PostgreSQL](#) running on an EC2 instance. Glue supports Postgres, MySQL, Redshift, and Aurora databases. To use other databases, you would have to provide your own JDBC jar file.

## Amazon VPC

Unfortunately, configuring Glue to crawl a JDBC database requires that you understand how to work

with Amazon VPC (virtual private clouds). I say unfortunately because application programmers don't tend to understand networking. Amazon requires this so that your traffic does not go over the public internet.

Fortunately, EC2 creates these network gateways (VPC and subnet) for you when you spin up [virtual machines](#). All you need to do is set the firewall rules in the **default** security group for your virtual machine.

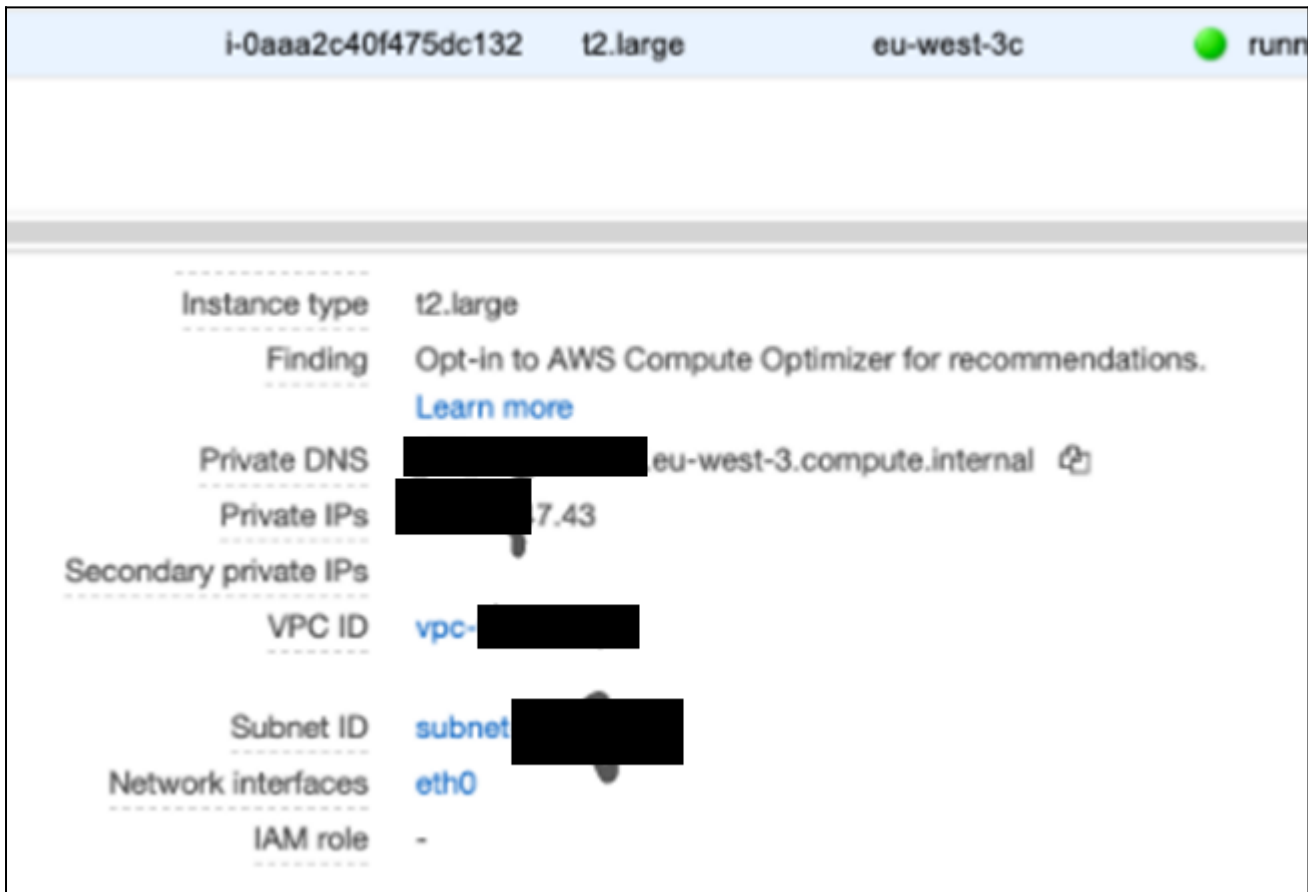
If you do this step wrong, or skip it entirely, you will get the error:

**ERROR : At least one security group must open all ingress ports. To limit traffic, the source security group in your inbound rule can be restricted to the same security group**

Glue can only crawl networks in the same AWS region—unless you create your own NAT gateway.

## Configure firewall rule

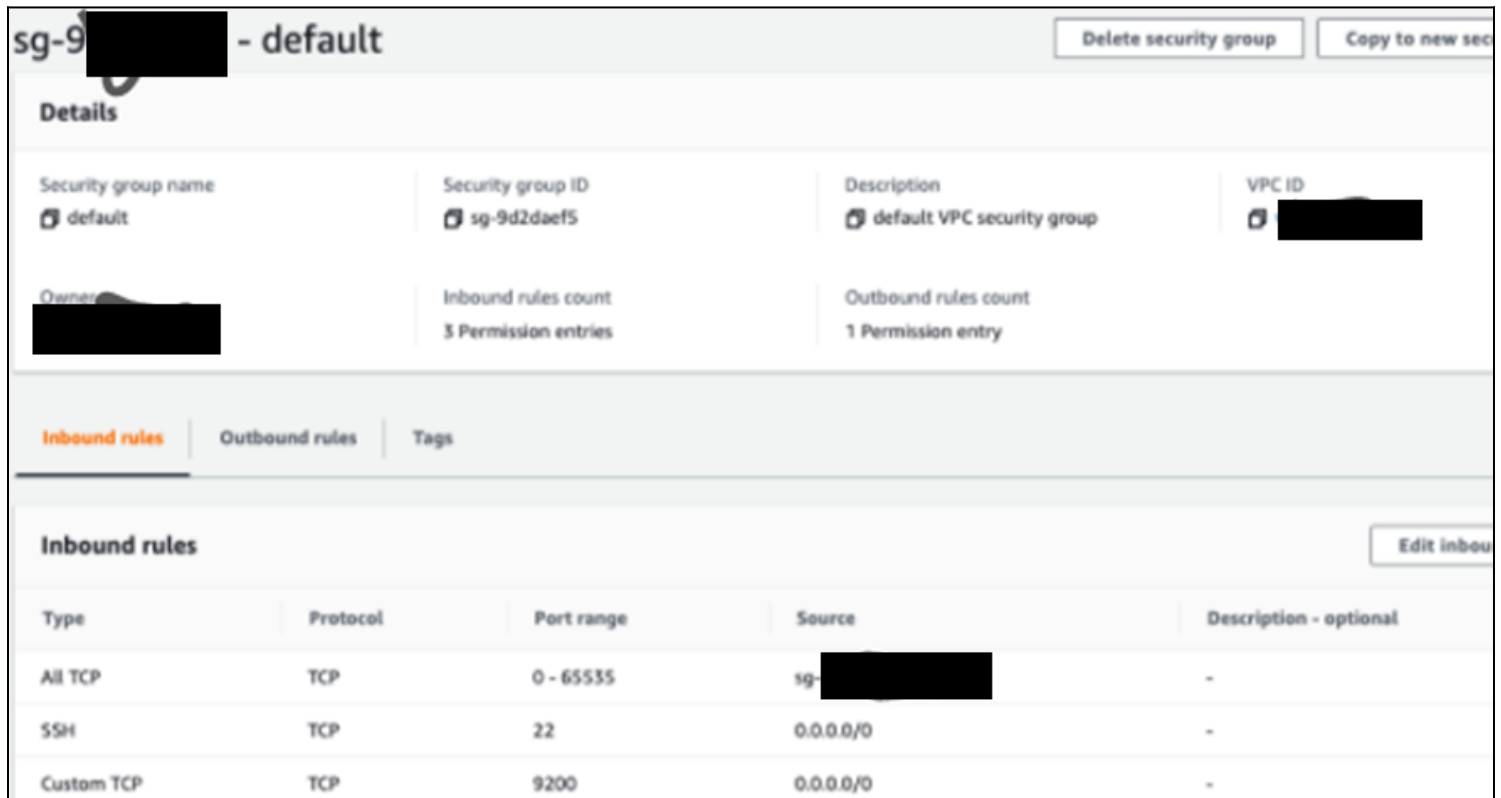
Look at the EC2 instance where your database is running and note the **VPC ID** and **Subnet ID**.



Go to

**Security Groups** and pick the **default** one. You might have to clear out the filter at the top of the screen to find that.

Add an **All TCP** inbound firewall rule. Then attach the **default** security group ID.



## Amazon Glue security groups

Don't use your Amazon console root login. Use an IAM user. For all Glue operations they will need: **AWSGlueServiceRole** and **AmazonS3FullAccess** or some subset thereof.

Your Glue security rule will look something like this:

```
arn:aws:iam::(XXXX):role/service-role/AWSGlueServiceRole-S3IAMRole
```

## Create a JDBC connection

In Amazon Glue, create a JDBC connection. It should look something like this:

```
Type      JDBC
JDBC URL   jdbc:postgresql://xxxxxx:5432/inventory
VPC Id     vpc-xxxxxxx
Subnet     subnet-xxxxxx
Security groups sg-xxxxxx
Require SSL connection false
Description -
Username    xxxxxxxx
Created    30 August 2020 9:37 AM UTC+3
Last modified 30 August 2020 4:01 PM UTC+3
```

## Define crawler

Create a Glue database. This is basically just a name with no other parameters, in Glue, so it's not really a database.

Next, define a [crawler to run](#) against the JDBC database. The **include path** is the **database/table** in the case of PostgreSQL.

For other databases, look up the JDBC connection string.



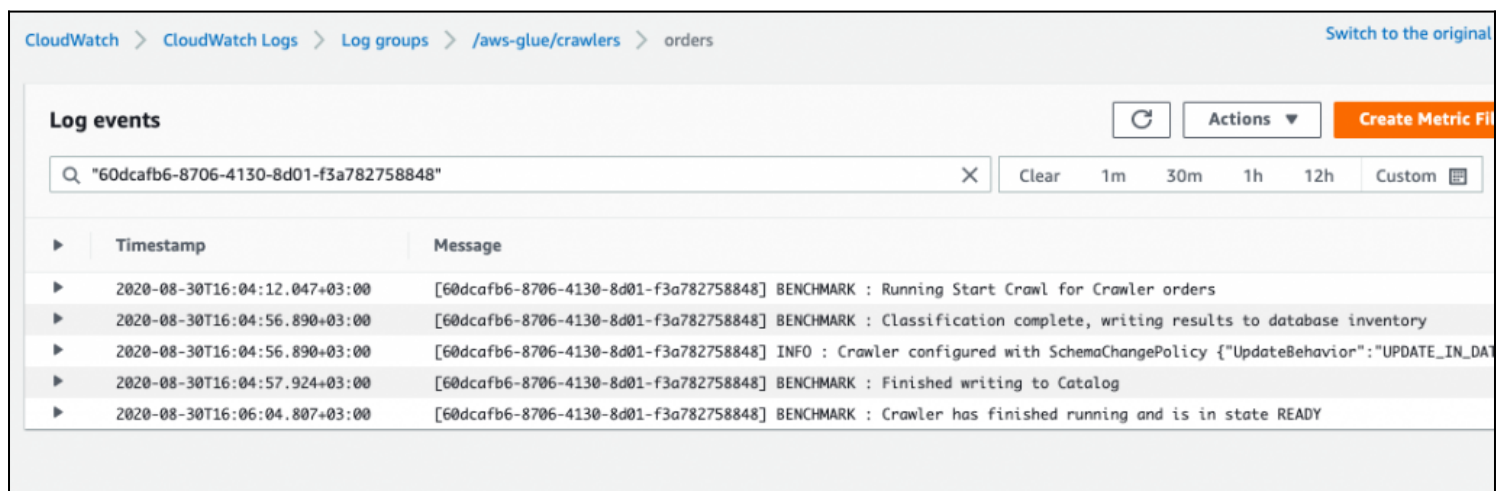
The screenshot shows the AWS Glue console configuration for a crawler named 'orders'. At the top, there are 'Run crawler' and 'Edit' buttons. Below is a list of configuration options:

<b>Name</b>	orders
<b>Description</b>	
<b>Create a single schema for each S3 path</b>	false
<b>Security configuration</b>	
<b>Tags</b>	-
<b>State</b>	Ready
<b>Schedule</b>	
<b>Last updated</b>	Sun Aug 30 16:03:48 GMT+300 2020
<b>Date created</b>	Sun Aug 30 09:38:16 GMT+300 2020
<b>Database</b>	inventory
<b>Service role</b>	service-role/AWSGlueServiceRole-S3IAMRole
<b>Selected classifiers</b>	
<b>Data store</b>	JDBC
<b>Connection</b>	orders postgresSQL
<b>Include path</b>	inventory/customers
<b>Exclude paths</b>	

Configuration options

## Run the crawler

Then you run the crawler, it provides a link to the logs stored in CloudWatch. Look there for errors or success.



The screenshot shows the AWS CloudWatch console displaying log events for the crawler. The breadcrumb navigation is 'CloudWatch > CloudWatch Logs > Log groups > /aws-glue/crawlers > orders'. The search filter is '60dcafb6-8706-4130-8d01-f3a782758848'. The log events are as follows:

Timestamp	Message
2020-08-30T16:04:12.047+03:00	[60dcafb6-8706-4130-8d01-f3a782758848] BENCHMARK : Running Start Crawl for Crawler orders
2020-08-30T16:04:56.890+03:00	[60dcafb6-8706-4130-8d01-f3a782758848] BENCHMARK : Classification complete, writing results to database inventory
2020-08-30T16:04:56.890+03:00	[60dcafb6-8706-4130-8d01-f3a782758848] INFO : Crawler configured with SchemaChangePolicy {"UpdateBehavior":"UPDATE_IN_DAT
2020-08-30T16:04:57.924+03:00	[60dcafb6-8706-4130-8d01-f3a782758848] BENCHMARK : Finished writing to Catalog
2020-08-30T16:06:04.807+03:00	[60dcafb6-8706-4130-8d01-f3a782758848] BENCHMARK : Crawler has finished running and is in state READY

If you have done everything correctly, it will generate metadata in tables in the database. This is not data. It's just a schema for your tables.

# Additional resources

For more tutorials like this, explore these resources:

- [BMC Machine Learning & Big Data Blog](#)
- [Apache Spark Guide](#), with 15 articles and tutorials
- [AWS Guide](#)
- [Amazon Braket Quantum Computing: How To Get Started](#)