Moving Redshift Data to Salesforce

using Heroku and AWS Glue

J. Dority
Platform Solutions Engineer

March 2018

V1

Introduction

Customers who leverage Amazon Redshift as a data warehouse platform often have the need to surface this data in Salesforce. Typically, customers want to expose summary information from their data warehouse into Salesforce either by:

- copying Redshift data directly into Salesforce Objects and Fields

  or

- creating an endpoint which allows data to be accessed virtually without copying data

While there are likely many strategies that could be architected to meet this objective, **Heroku coupled with AWS Services and Salesforce Connect offer unique value**. ***Middleware solutions such as Informatica*** could be used, but these tools are likely to consume significant Salesforce API resource which may cause the customer's Salesforce org to **hit its *API limits***. In addition, Informatica also requires specialized ETL administrator/developer skills. With the AWS Services, Heroku, and Salesforce solution outlined in this document, a majority of Salesforce Administrators would be capable of configuring this environment without specialized.

To accomplish this integration, data must first be extracted from Redshift (OLAP) and transformed into a data stream that can be written to an OLTP consumable format. For this Redshift use case, ***AWS Glue*** is the preferred solution to perform the Extract-Transform- Load of the Redshift source data.

Once this transformation is complete, the data is then written to an ***OLTP target database (Heroku Postgres)*** where ***Heroku Connect (unique data sync service to Salesforce***) can automatically sync the data into Salesforce. The entire work stream can be automated to refresh the data at any frequency. The Amazon Redshift instance is a part of an Amazon VPC which connects via a Trusted IP Link to a Heroku Private Space (VPC) containing Heroku Postgres.

Note: Heroku Connect only operates with a *Heroku Postgres database* which is why Heroku Postgres is the required target database.

(NOTE: YOU MUST REQUEST TRUSTED IP DATA SERVICE BE ENABLED – EMAIL postgres@heroku.com with App Name)**

The following outlines high level functionality of the various solutions:

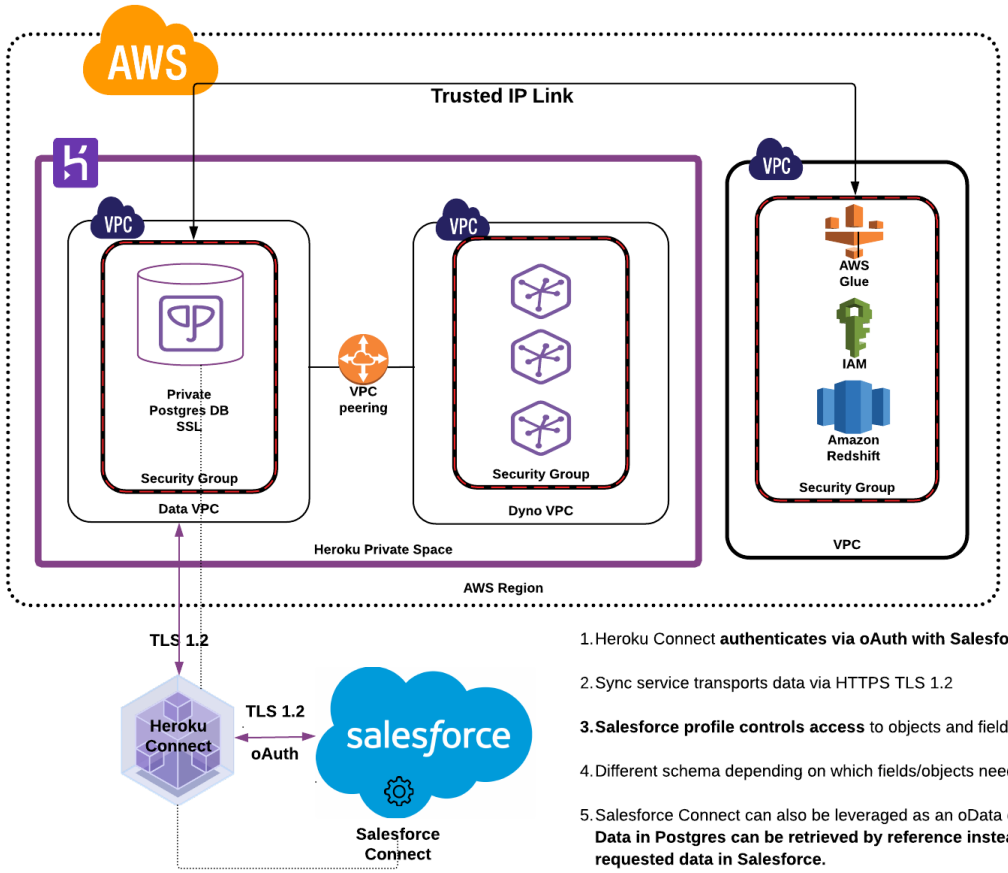
	AWS Glue (ETL)	AWS Data Migration Services	Amazon Redshift	Heroku Postgres/Connect	Salesforce Connect
OLAP to OLTP transformation	Yes Redshift (source) Heroku Postgres (target)	No	N/A	N/A	N/A
Declarative Interface	Yes	Yes	N/A	Yes	Yes
Scheduled jobs	Yes	Yes		Yes (polling)	N/A
oData endpoint available	No	No	No	Yes (producer)	Yes (consumer)
Virtualize data in Salesforce	No	No	No	No	Yes

The remainder of this document provides a basic overview of the configuration steps used to prepare a demonstration environment, and is not intended to be used in place of vendor documentation or best practice recommendations. This document covers:

- Configuration of AWS VPC / Trusted IP Like to Heroku Private Space
- Amazon Glue Configuration
- Configuring Heroku (Postgres and Connect) is covered in a separate guide

Reference Architecture

Trusted IP Link



1. Connections to the Postgres DB inside the Heroku Private Space are made using the database's public hostname. However, **AWS keeps the traffic internal to the AWS Region rather than routing it over the public Internet.**

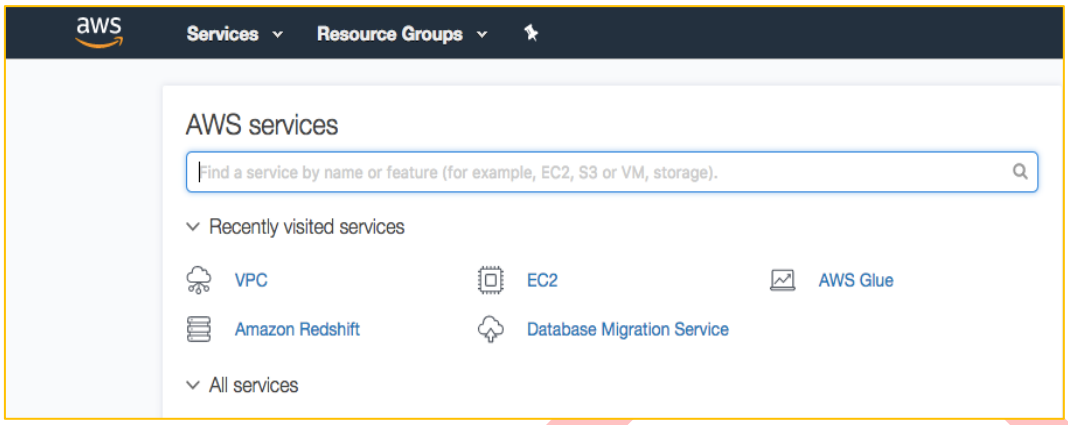
   The connection to the database via the trusted IP entry point is completely within the AWS network (if you are connecting from AWS) so this limits the concern about traffic going over the public Internet. Traffic is encrypted and the AWS firewall setup is such that an interception/MTM attack is not possible.

2. **Heroku PostgreSQL servers require SSL connections**

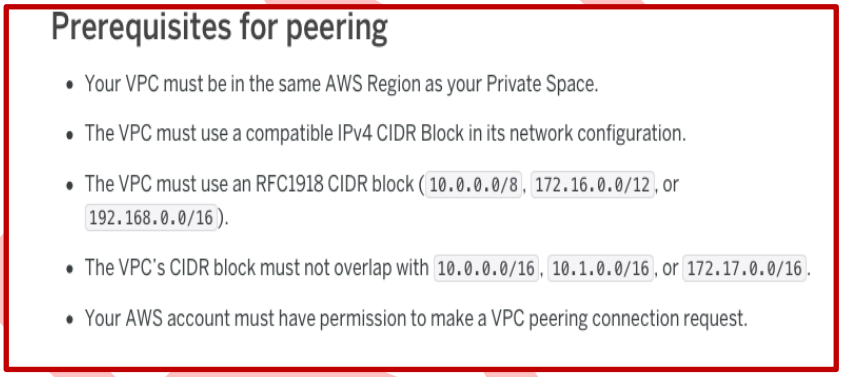
1. Heroku Connect **authenticates via oAuth with Salesforce username/password/profile**
2. Sync service transports data via HTTPS TLS 1.2
3. **Salesforce profile controls access** to objects and fields
4. Different schema depending on which fields/objects needed
5. Salesforce Connect can also be leveraged as an oData connection to Heroku Connect. **Data in Postgres can be retrieved by reference instead of actually writing/storing the requested data in Salesforce.**

VPC / Trusted IP Link Configuration:

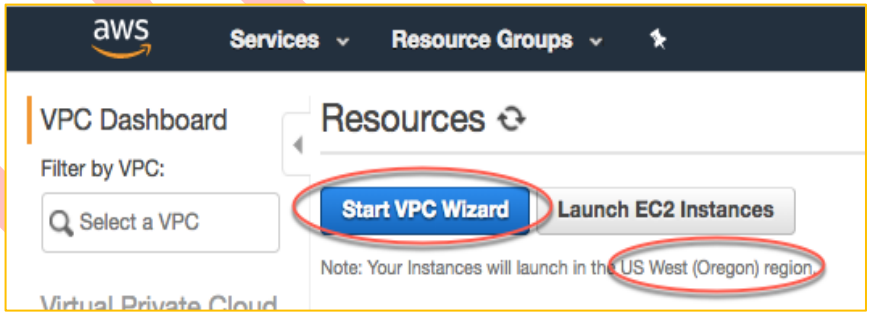
1. From the AWS Console configure an AWS VPC. Select VPC to get started:



2. *Note: When configuring the VPC, attention must be given to the region, CIDR block, etc. that you select. See Heroku prerequisites for peering before you begin. https://devcenter.heroku.com/articles/private-space-peering



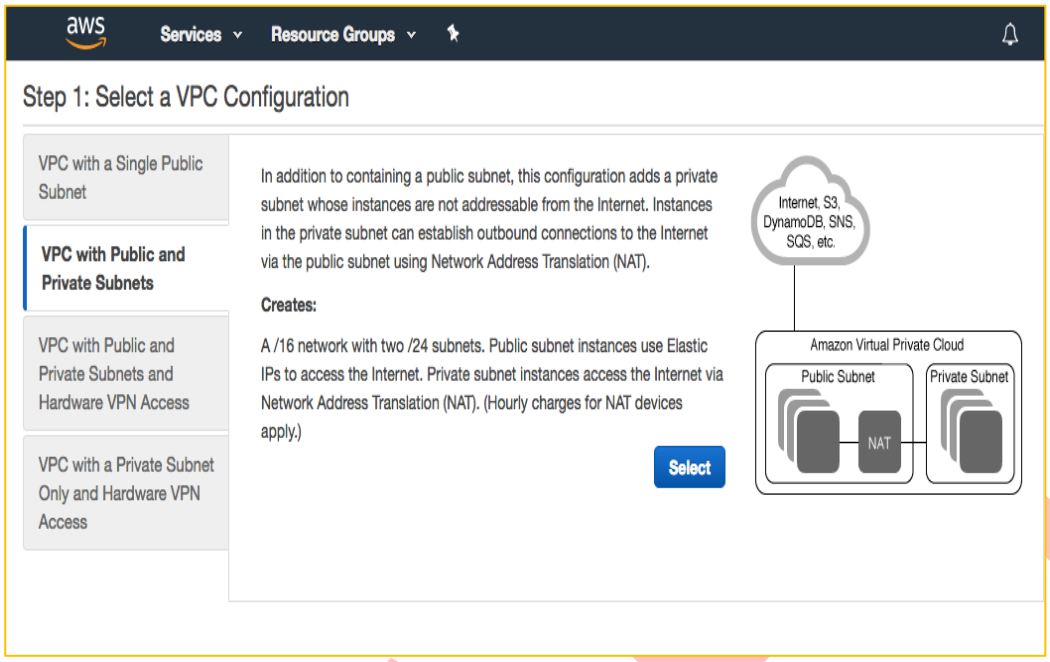
3. **Create an AWS VPC using the VPC Wizard:**
   a. Click "Start VPC Wizard"



***Region must be the same as your Heroku Private Space**

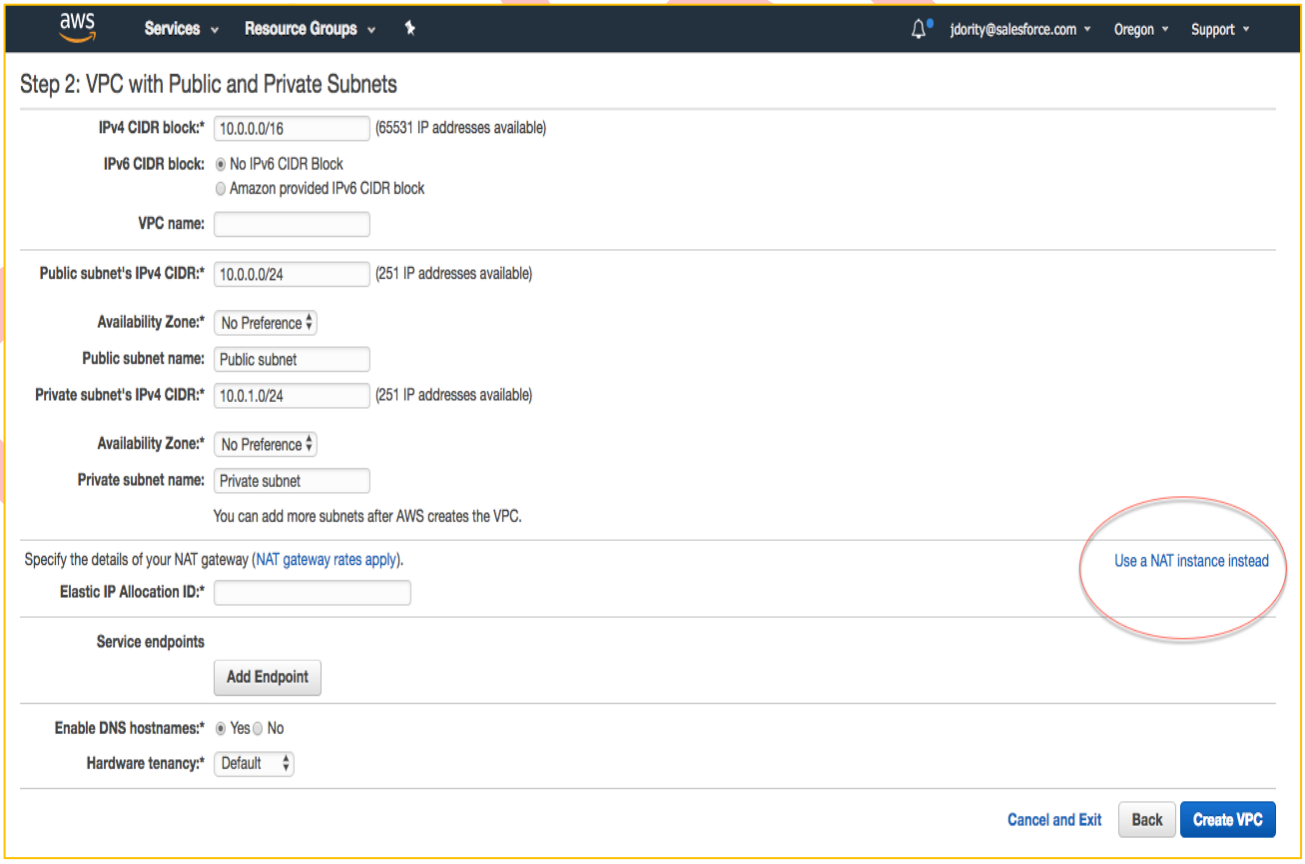
https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario2.html

b. Select "***with Public and Private Subnets"***



c. Be sure to follow the Peering Prerequisites (i.e. CIDR block must be in the supported range.)

d. **Select NAT Instance in the VPC Wizard;** this will be changed later.

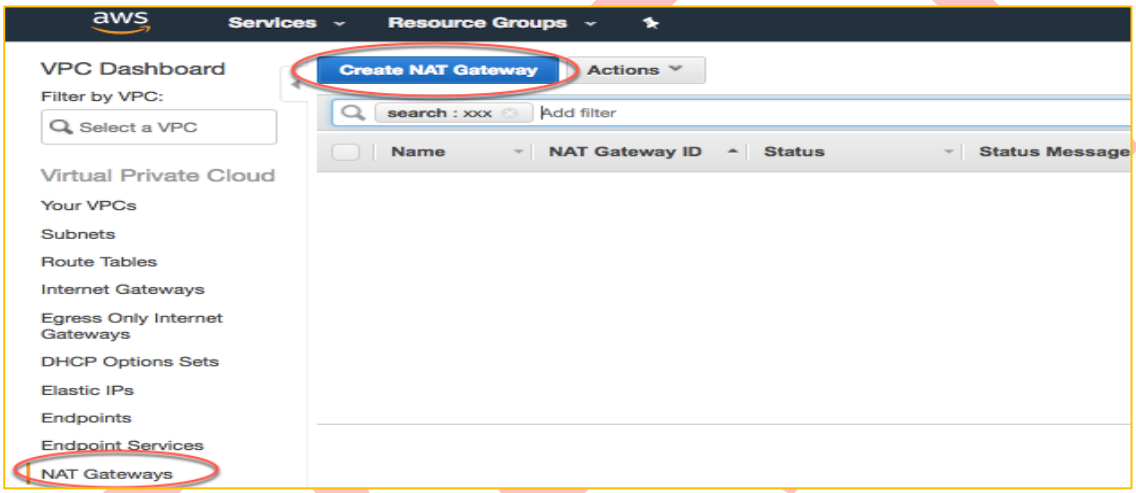


e. If you are creating a new VPC with the VPC Wizard you must create the NAT Gateway *after you launch the VPC*. A NAT Gateway enables instances in a private subnet to connect to the internet or other AWS services such as AMZ Glue.

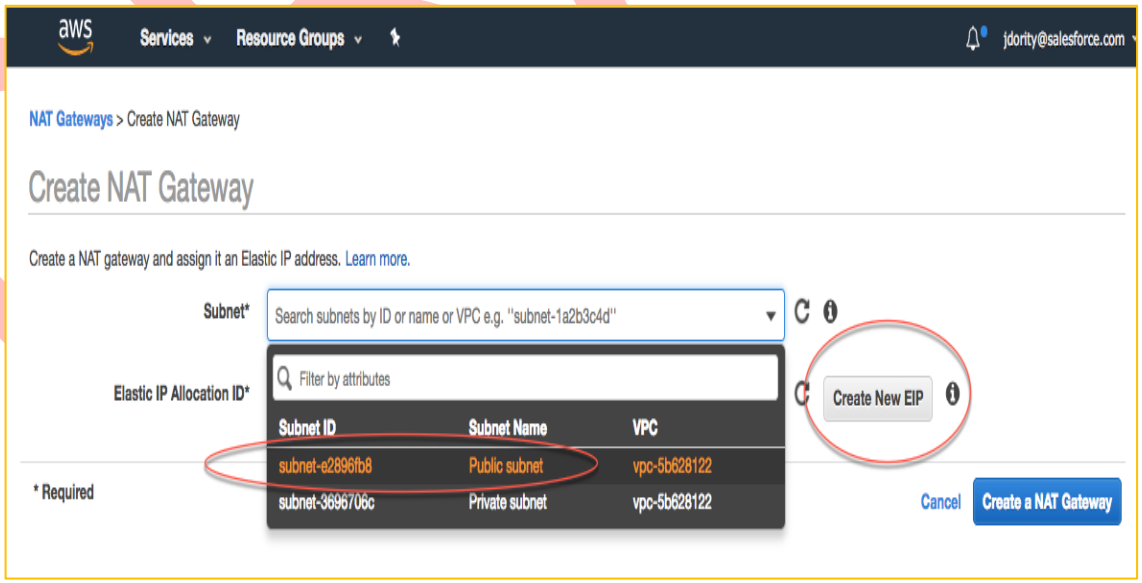
A Private Subnet using a NAT Gateway will BLOCK internet traffic such as a SQL client from connecting to the databases. If the Private Subnet uses the IGW access is then access is possible.

https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-nat-gateway.html

4. **Create a NAT Gateway from the VPC Dashboard:**
   a. Select NAT Gateways and Create NAT Gateway
   b. Select your Public Subnet (more about subnets later)
   c. Click Create EIP (Elastic IP)
   d. Click Create NAT Gateway



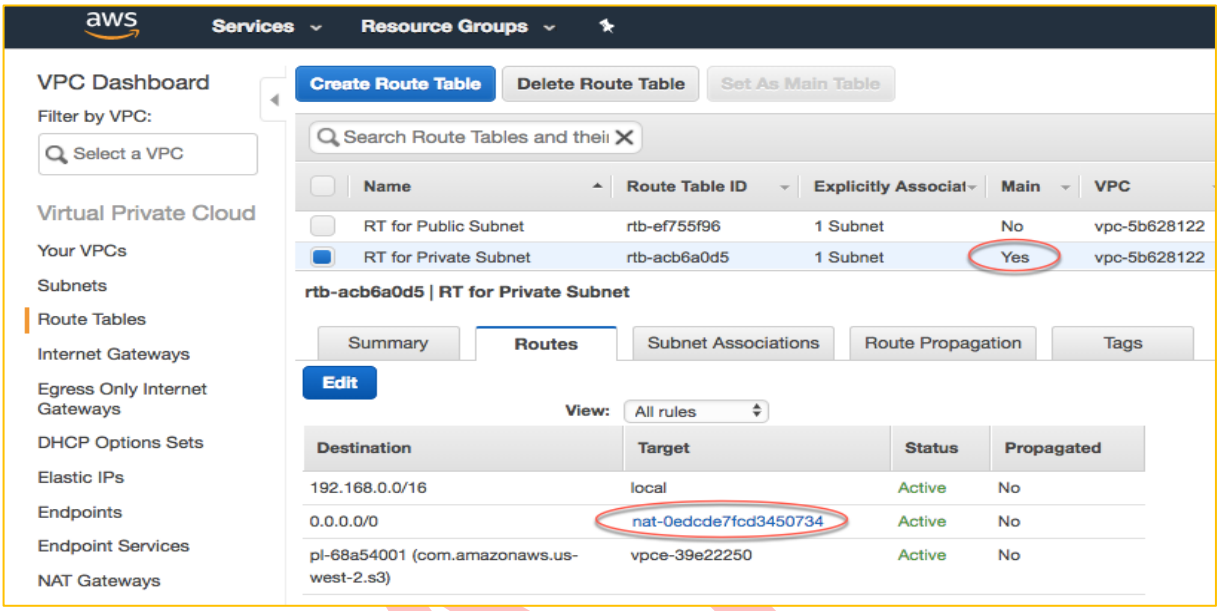
Specify the ***public subnet*** from your VPC in which the NAT gateway will reside. You must also specify an Elastic IP address to associate with the NAT gateway when you create it.

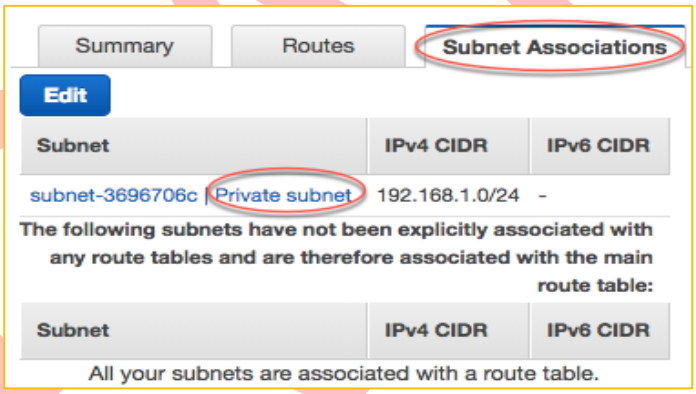


After you've created a NAT gateway, **you must update the route table associated with one or more of your private subnets to point Internet-bound traffic to the NAT gateway.** This enables instances in your private subnets to communicate with the internet.

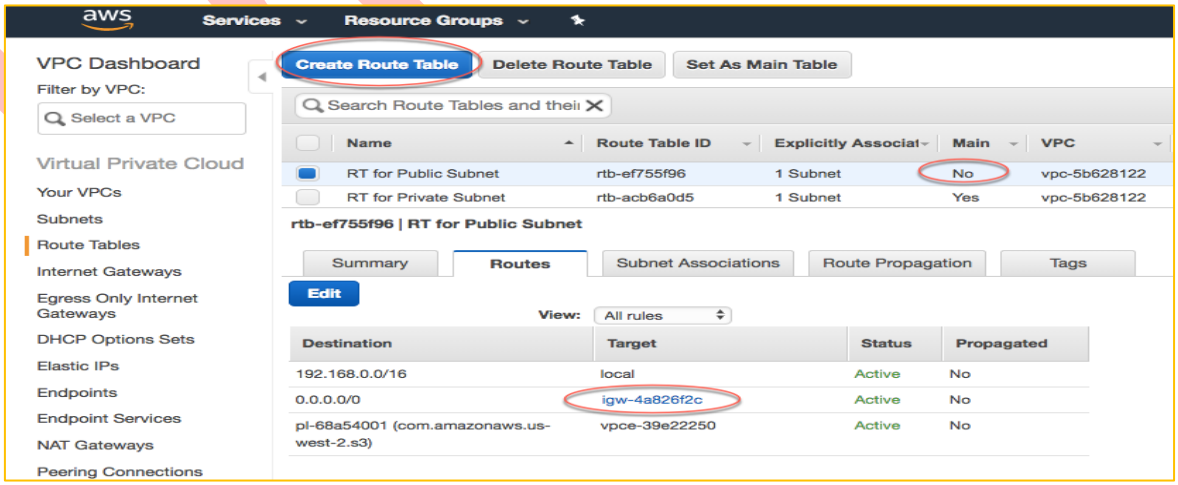
5. **Configure Route tables for your Subnets:**

   a. **PRIVATE SUBNET:** Associate the **private subnet to the main route table**
   b. **Edit the route table for the private subnet and point all traffic to a target of your newly created NAT Gateway**

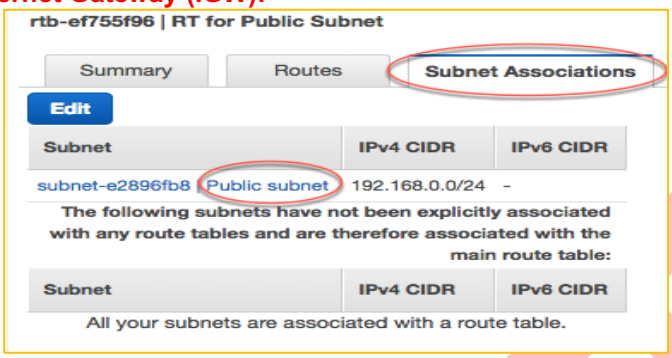




   c. **PUBLIC SUBNET:** **If a second route table does not exist, create a new route table**



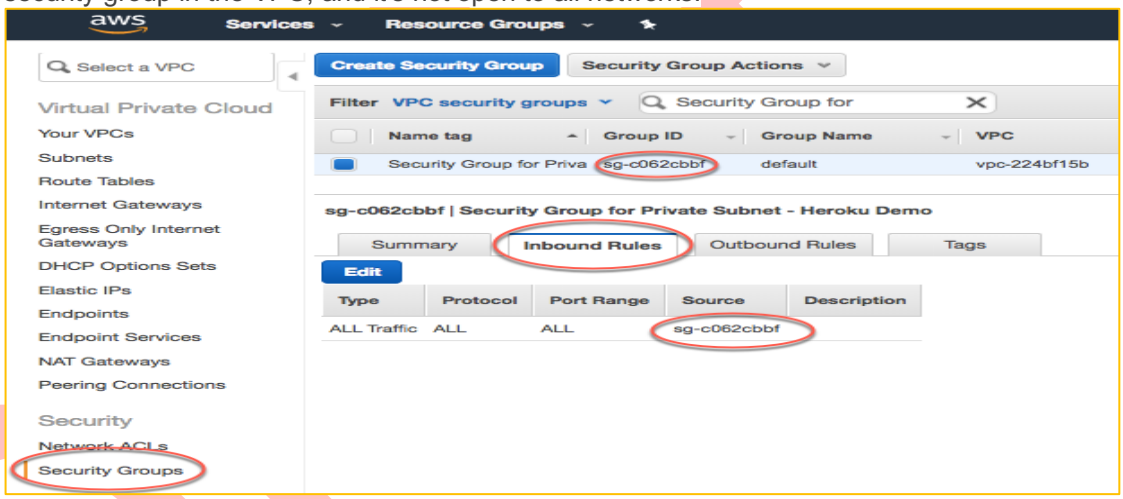
a. Associate the **public subnet to this new route table and edit the route so that all traffic goes through the Internet Gateway (IGW):**



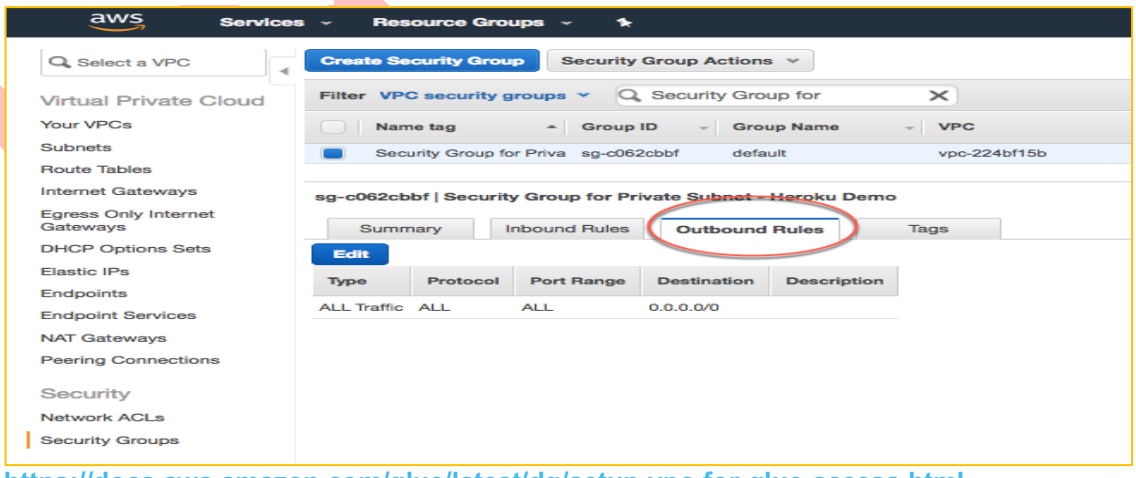
Instances in the public subnet can send outbound traffic directly to the Internet, whereas the instances in the private subnet can't. The database servers can connect to the Internet for software updates using the NAT gateway, **but the Internet cannot establish connections to the database servers.**
https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-nat-gateway.html

6. **Security Groups** control both inbound and outbound traffic at the **instance level**, ***and specifically control traffic for a private subnet.***

   a. **I**nbound traffic should be self-referencing. self-referencing rule, you can restrict the source to the same security group in the VPC, and it's not open to all networks.

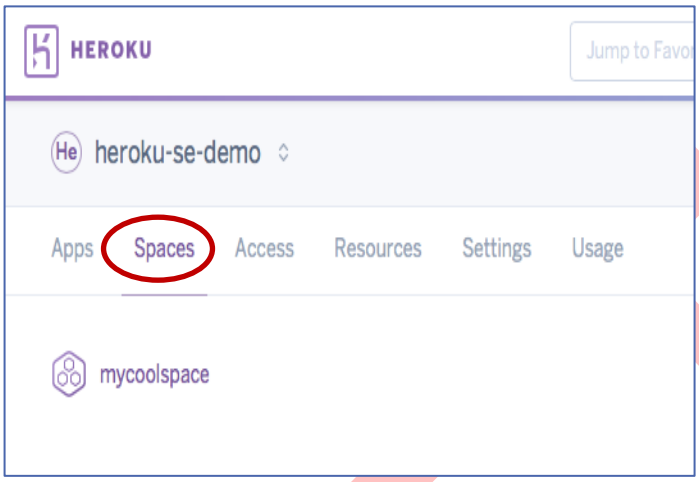


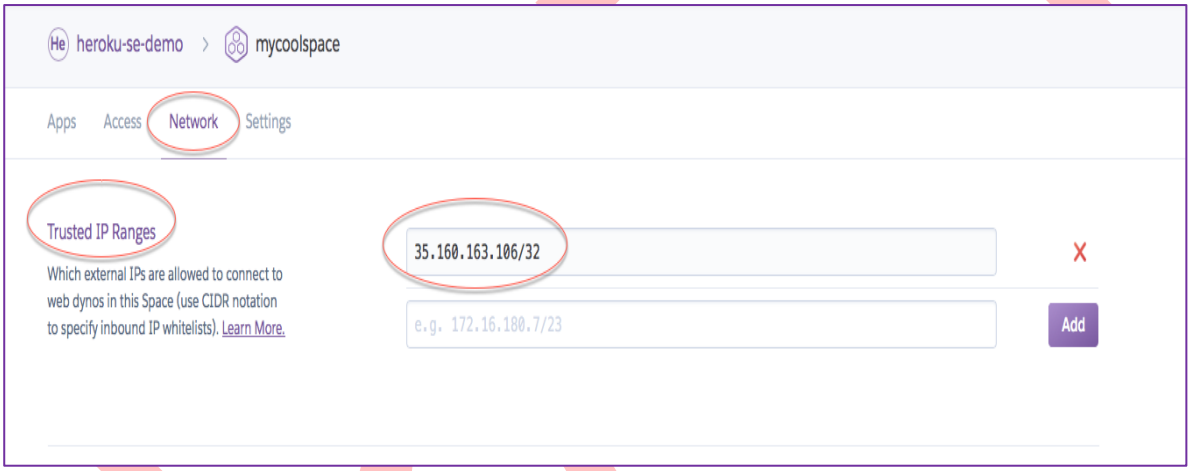
   b. Outbound traffic is set to all.



https://docs.aws.amazon.com/glue/latest/dg/setup-vpc-for-glue-access.html

7. After configuring a Private Space and Private Postgres Database, white list the Elastic IP that was configured in step 4 (Creating the NAT Gateway).

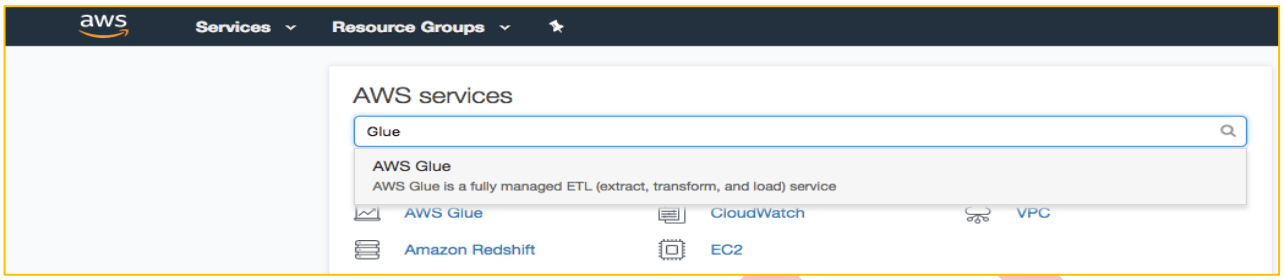




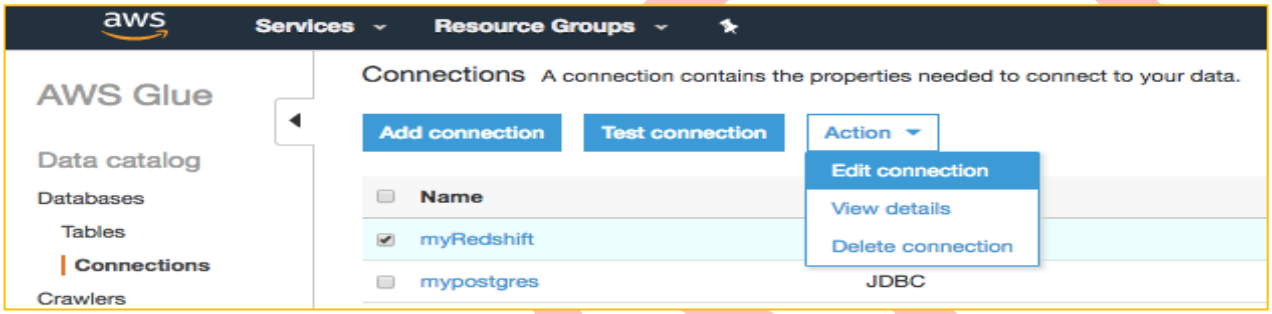
Amazon Glue Example Configuration:

In this example, we will set up connections to both Redshift and Heroku Postgres, collect metadata from both data sources, create a job, and finally we will execute the job and copy the data to Heroku Postgres.

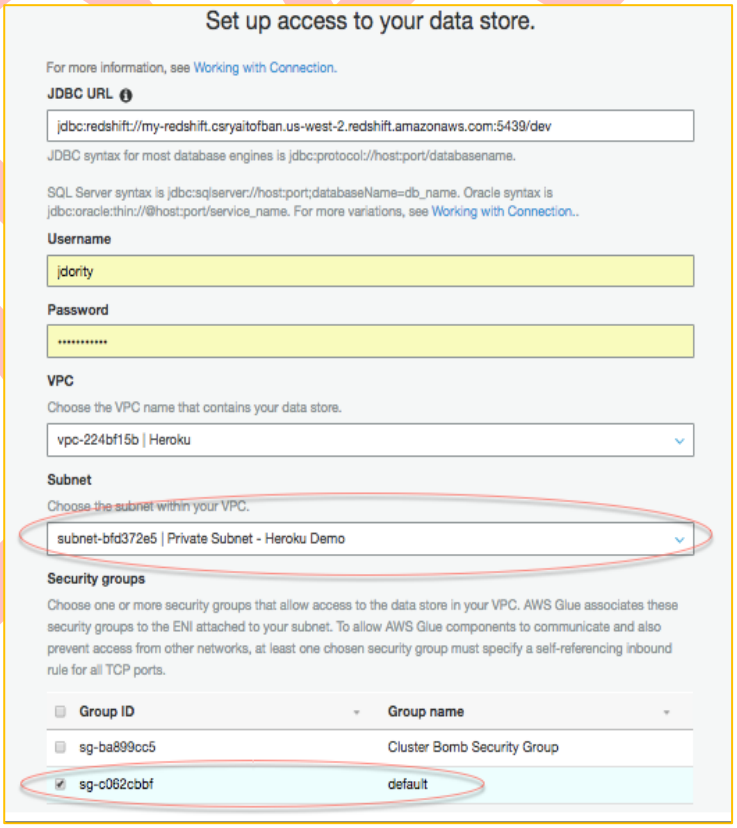
1. From the AWS Console, select AWS Glue:



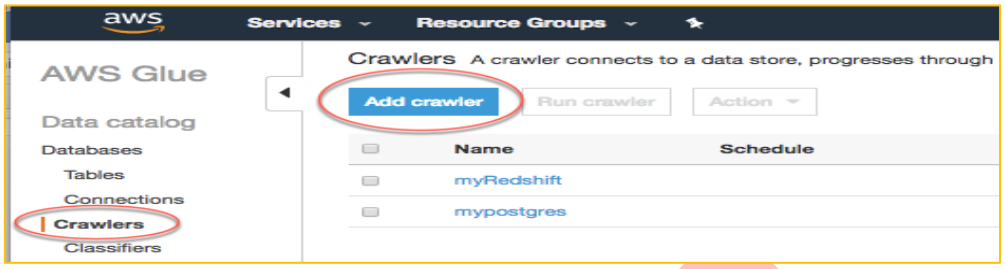
2. Set up connections to your data source(s) and target(s) by "Adding connection":



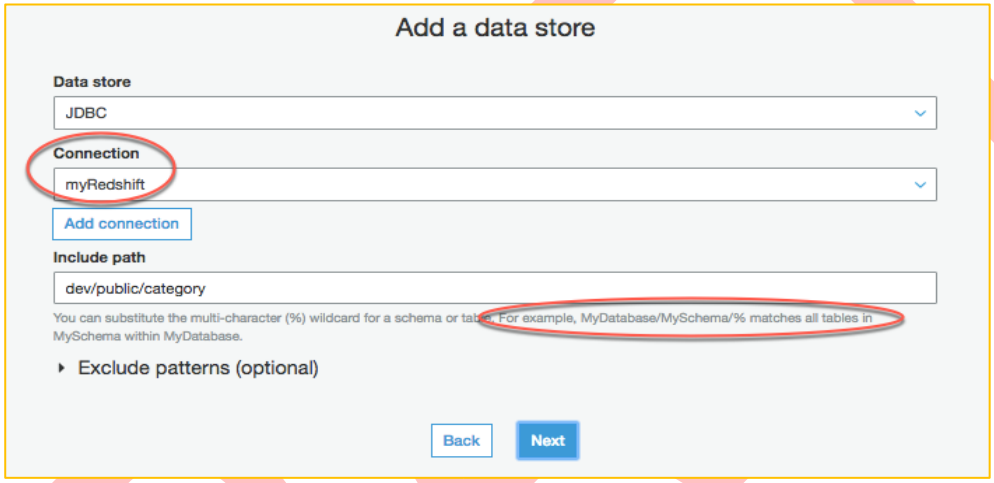
Be sure to select the PRIVATE SUBNET in your VPC and the correct security that has self-referencing routing.



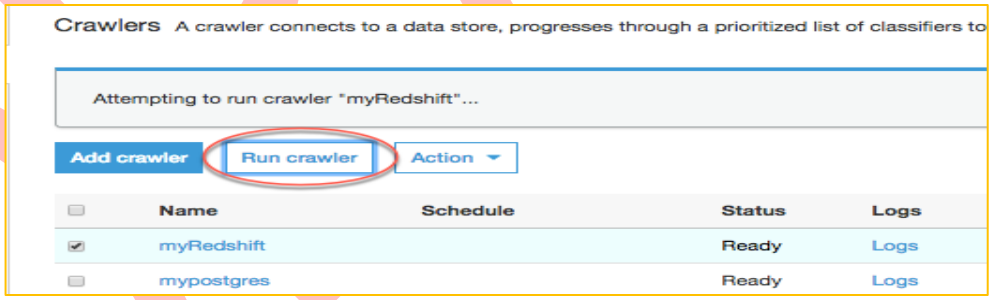
3. After your connections are configured, add a metadata crawler to collect metadata from your source and target tables.



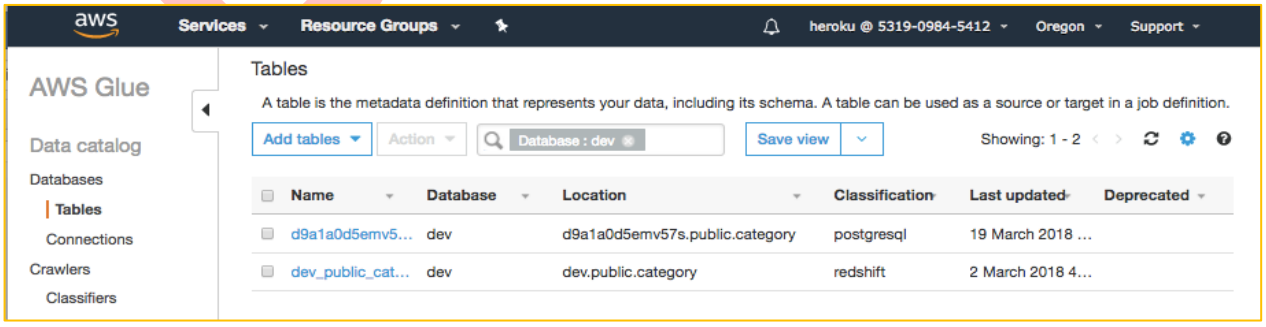
4. Provide the connection you created in Step 2 and specify the database/schema/tables you wish to collect metadata from:



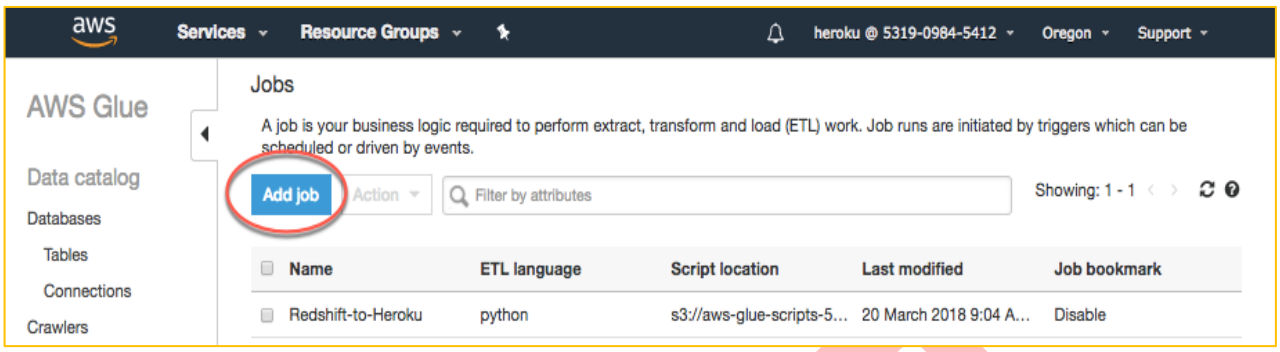
After configuring the crawler, "Run" it:



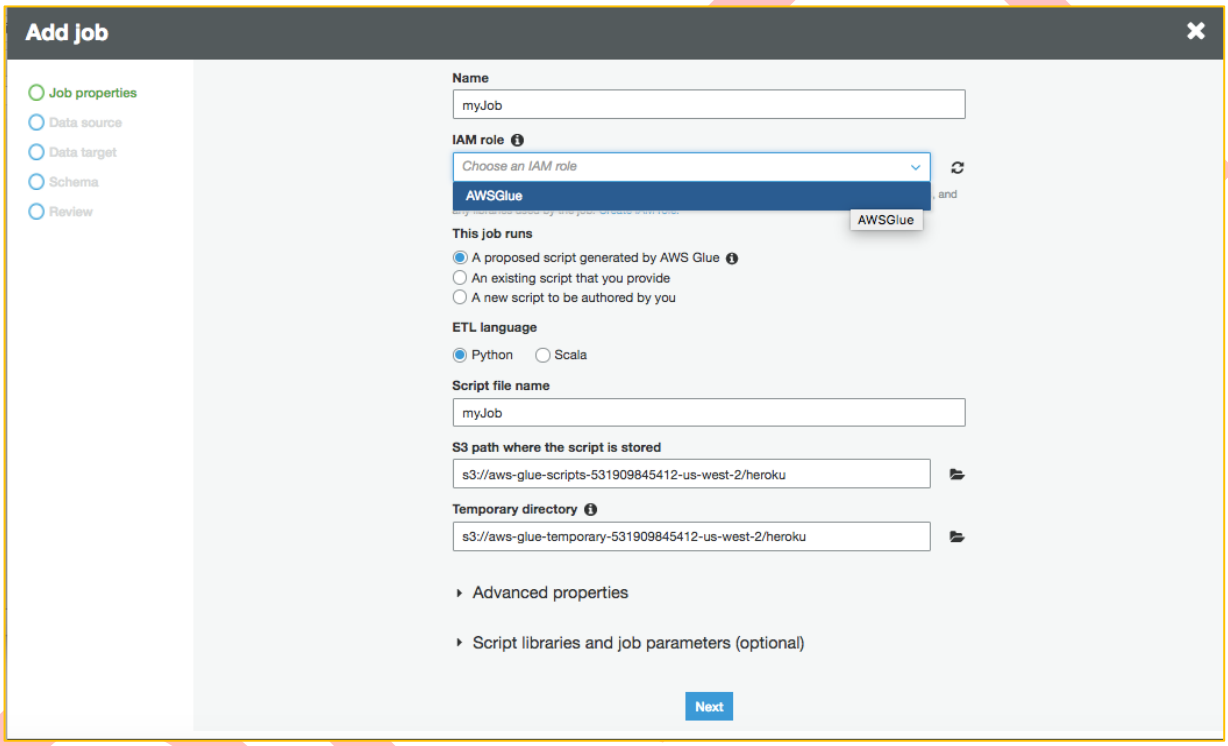
5. When the crawler has finished running, you can see the table metadata in the Database-→Tables section:



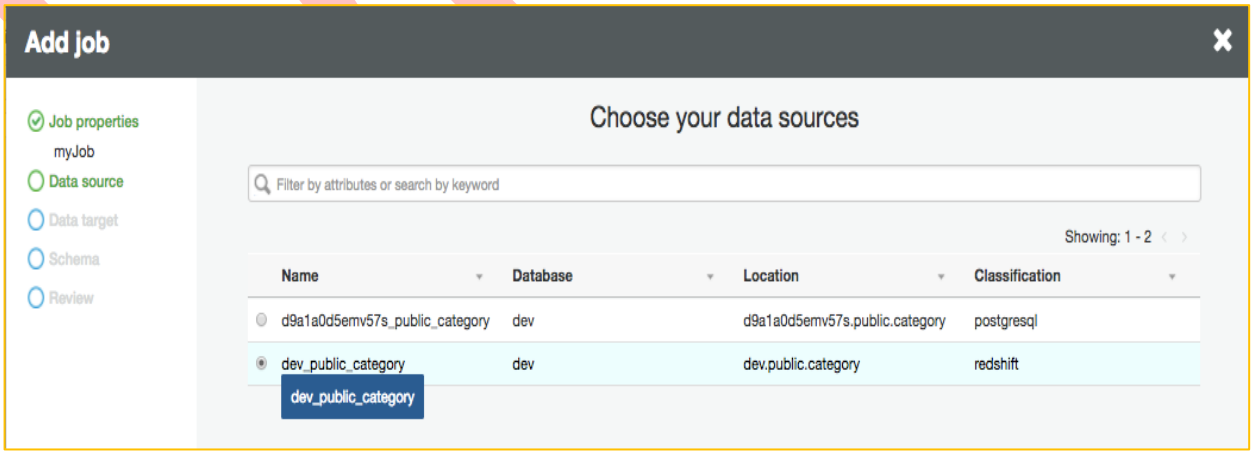
6. Once you have the table metatdata, you are ready to create a Job:



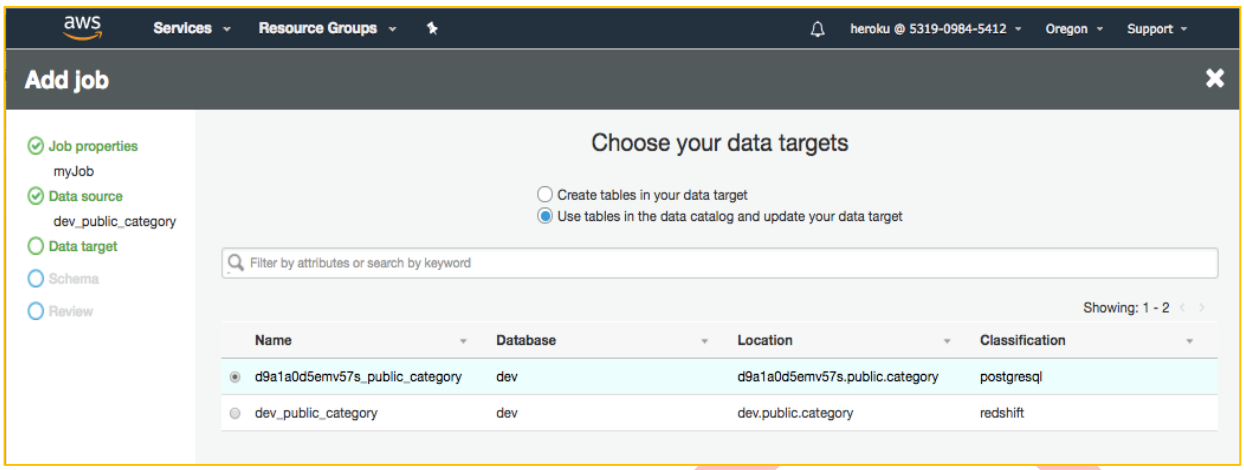
You'll need to specify an IAM Role (Identity and Access Management Role) that is configured outside of Glue.



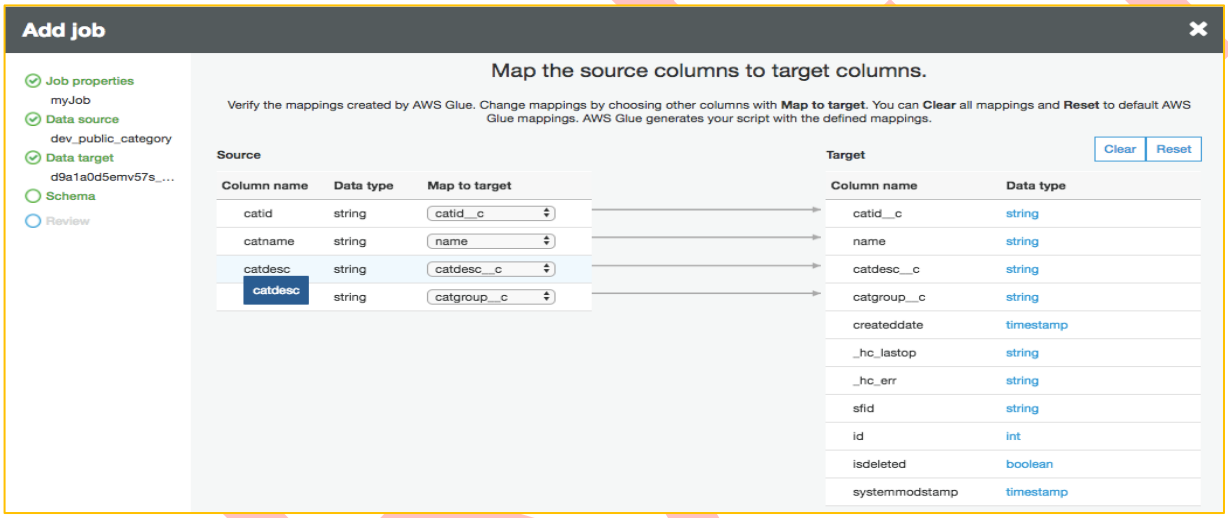
7. Select your data source table, in this case "category" in redshift:



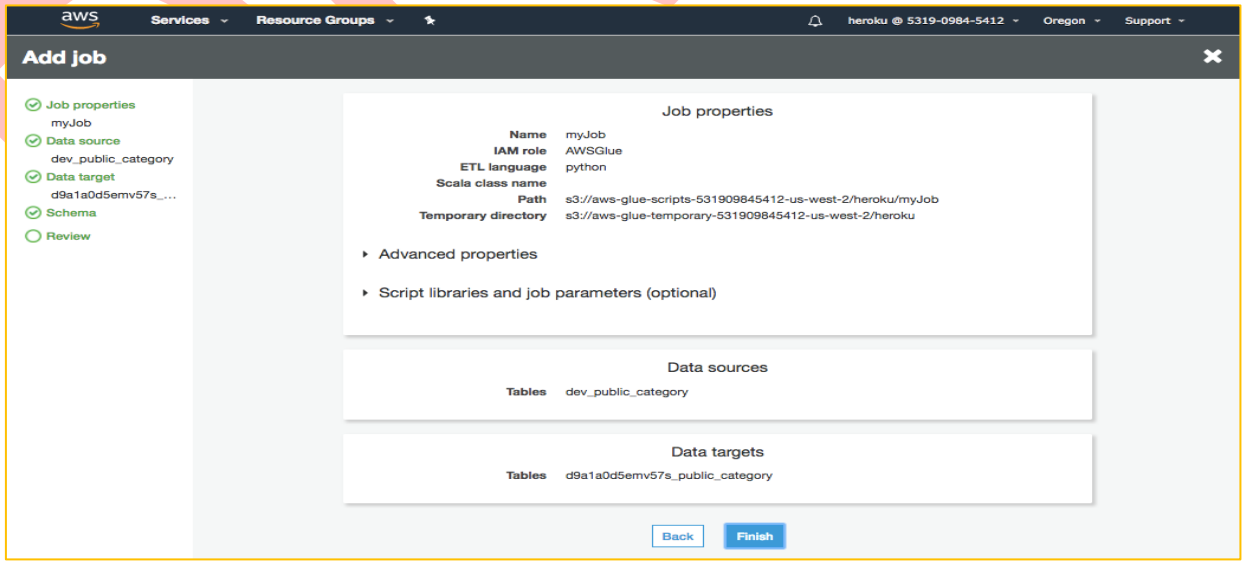
8. Select your data target table, in this case "category__c" in postgresql:



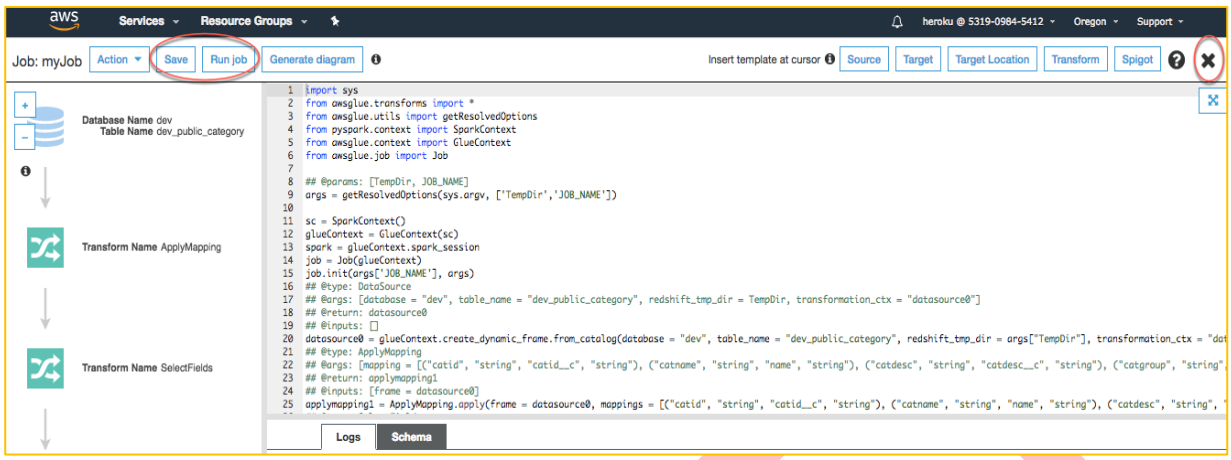
9. Glue will attempt to "auto-map" column names that match:



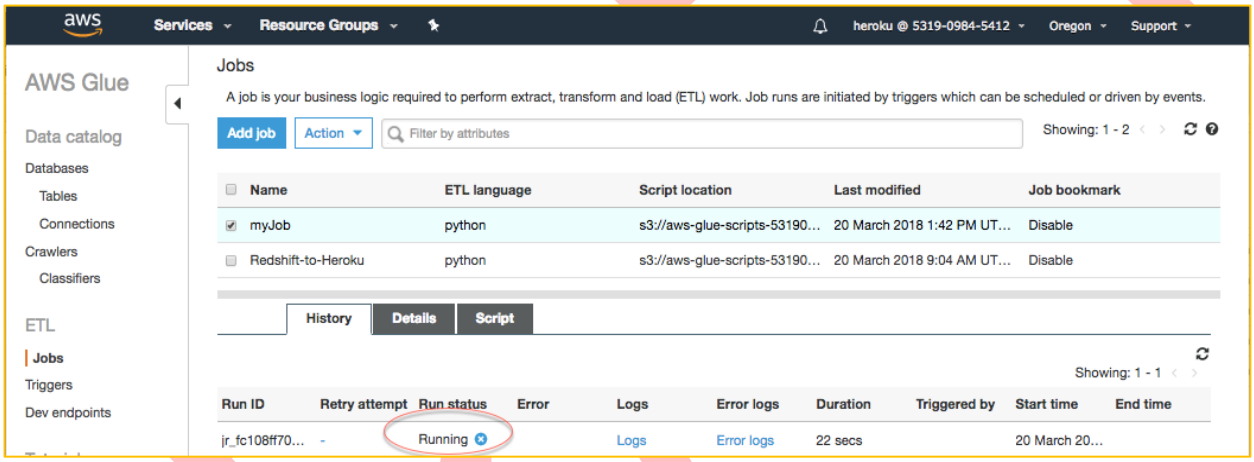
10. Once you finish mapping (and adding transformations), click "Finish":



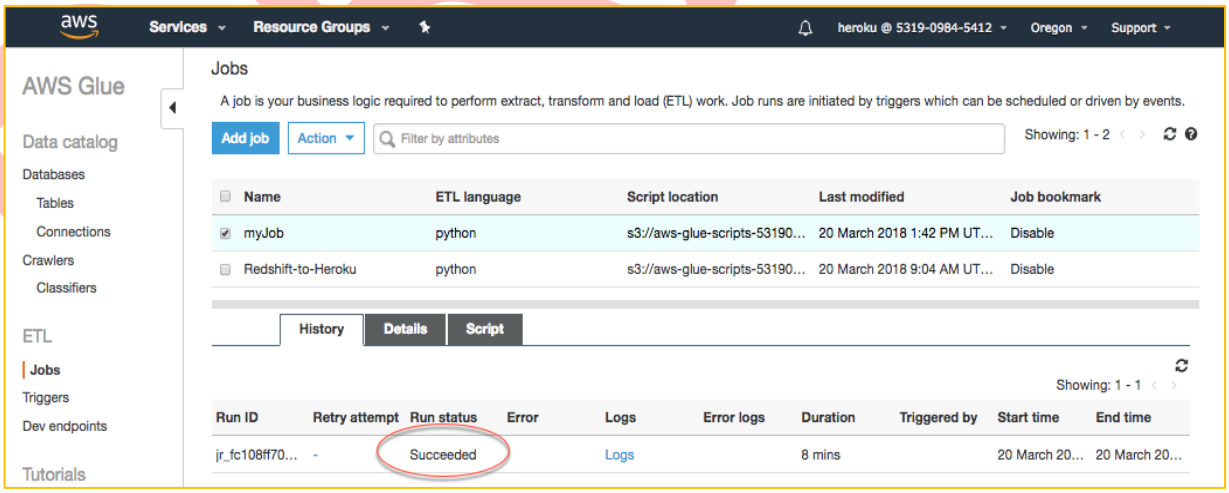
11. Click "Save" and then "Run Job". To exit the script/diagram screen click the "x" in the right upper hand corner.



12. You should see the job with a status of "running":



13. Once the job completes, you should see a status of "Succeeded".



Note: Amazon Glue does not currently have the ability to TRUNCATE a table or to UPSERT/REPLACE a row. As a workaround, I am using a Heroku Postgres function/trigger that manages the upsert. AWS Glue writes to a staging table in Postgres, and this table has the function/trigger configured, which inserts/upserts the Redshift rows into the Heroku Connect Schema. Please see Github for an example. https://github.com/jdority/redshift-to-salesforce/blob/master/category_setup.sql