



# Moving Redshift Data to Salesforce

*using Heroku and Amazon Glue*

J. Dority  
Platform Solutions Engineer

March 2018

V1

DRAFT

## 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](mailto:postgres@heroku.com) with App Name)**

The following outlines high level functionality of the various solutions:

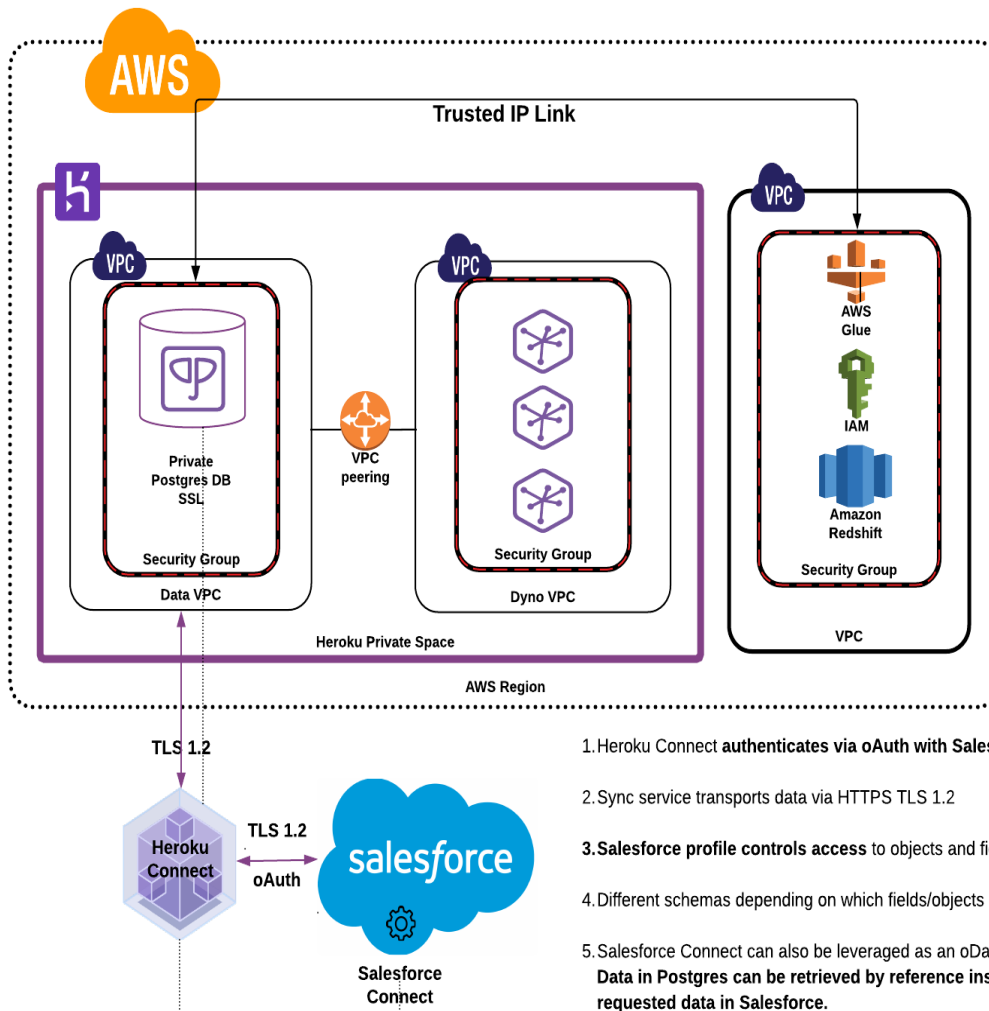
	<b>AWS Glue (ETL)</b>	<b>AWS Data Migration Services</b>	<b>Amazon Redshift</b>	<b>Heroku Postgres/Connect</b>	<b>Salesforce Connect</b>
OLAP to OLTP transformation	Yes Redshift (source) Heroku Postgres (target)	No	N/A	N/A	N/A
Declarative Interface	Yes	Yes	N/A	Yes	Yes
Scheduled jobs	Yes	Yes		Yes (polling)	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.**

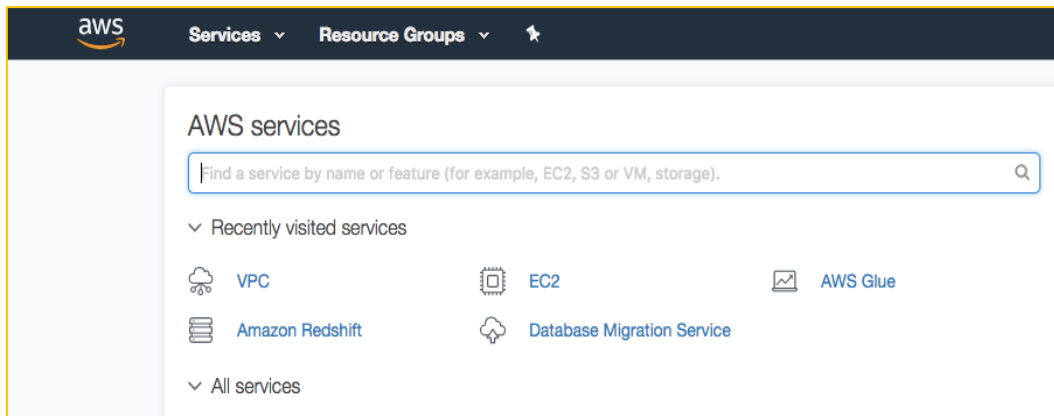
The connection to the database via the trusted IP entry point is completely within the AWS network (if you are connecting from AWS) so this limits the concern about traffic going over the public Internet. Traffic is encrypted and the AWS firewall setup is such that an interception/MTM attack is not possible.

2. Heroku PostgreSQL servers do 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 schemas 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:

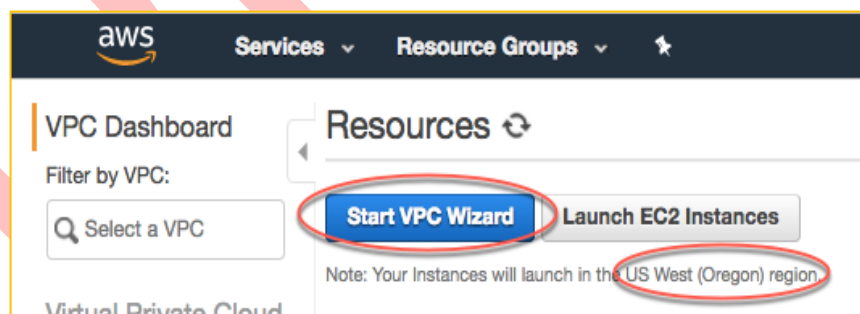


2. \*Note: When configuring the VPC, attention must be given to the region, CIDR block, etc. that you select. See Heroku prerequisites for peering before you begin. <https://devcenter.heroku.com/articles/private-space-peering>

### Prerequisites for peering

- Your VPC must be in the same AWS Region as your Private Space.
- The VPC must use a compatible IPv4 CIDR Block in its network configuration.
- The VPC must use an RFC1918 CIDR block ( `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16` ).
- The VPC's CIDR block must not overlap with `10.0.0.0/16`, `10.1.0.0/16`, or `172.17.0.0/16`.
- Your AWS account must have permission to make a VPC peering connection request.

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

b. Select **“with Public and Private Subnets”**

aws Services Resource Groups

### Step 1: Select a VPC Configuration

**VPC with a Single Public Subnet**

**VPC with Public and Private Subnets**

VPC with Public and Private Subnets and Hardware VPN Access

VPC with a Private Subnet Only and Hardware VPN Access

In addition to containing a public subnet, this configuration adds a private subnet whose instances are not addressable from the Internet. Instances in the private subnet can establish outbound connections to the Internet via the public subnet using Network Address Translation (NAT).

**Creates:**

A /16 network with two /24 subnets. Public subnet instances use Elastic IPs to access the Internet. Private subnet instances access the Internet via Network Address Translation (NAT). (Hourly charges for NAT devices apply.)

**Select**

Internet, S3, DynamoDB, SNS, SQS, etc.

Amazon Virtual Private Cloud

Public Subnet NAT Private Subnet

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.

aws Services Resource Groups jdority@salesforce.com Oregon Support

### Step 2: VPC with Public and Private Subnets

IPv4 CIDR block:\* 10.0.0.0/16 (65531 IP addresses available)

IPv6 CIDR block: ☒ No IPv6 CIDR Block ☐ Amazon provided IPv6 CIDR block

VPC name:

Public subnet's IPv4 CIDR:\* 10.0.0.0/24 (251 IP addresses available)

Availability Zone:\* No Preference

Public subnet name: Public subnet

Private subnet's IPv4 CIDR:\* 10.0.1.0/24 (251 IP addresses available)

Availability Zone:\* No Preference

Private subnet name: Private subnet

You can add more subnets after AWS creates the VPC.

Specify the details of your NAT gateway (NAT gateway rates apply).

Elastic IP Allocation ID:\*

Service endpoints

Add Endpoint

Enable DNS hostnames:\* ☒ Yes ☐ No

Hardware tenancy:\* Default

[Use a NAT instance instead](#)

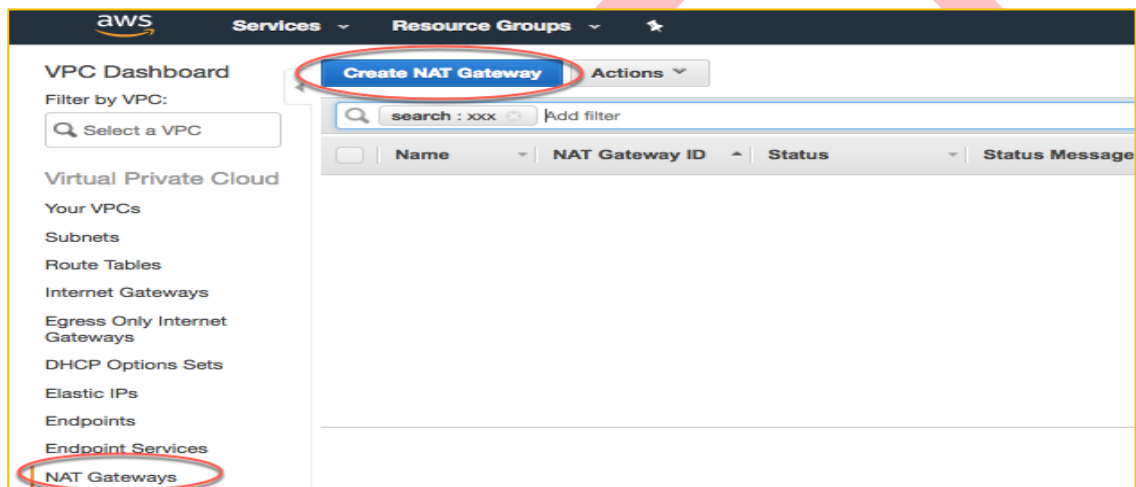
Cancel and Exit Back **Create VPC**

- 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.

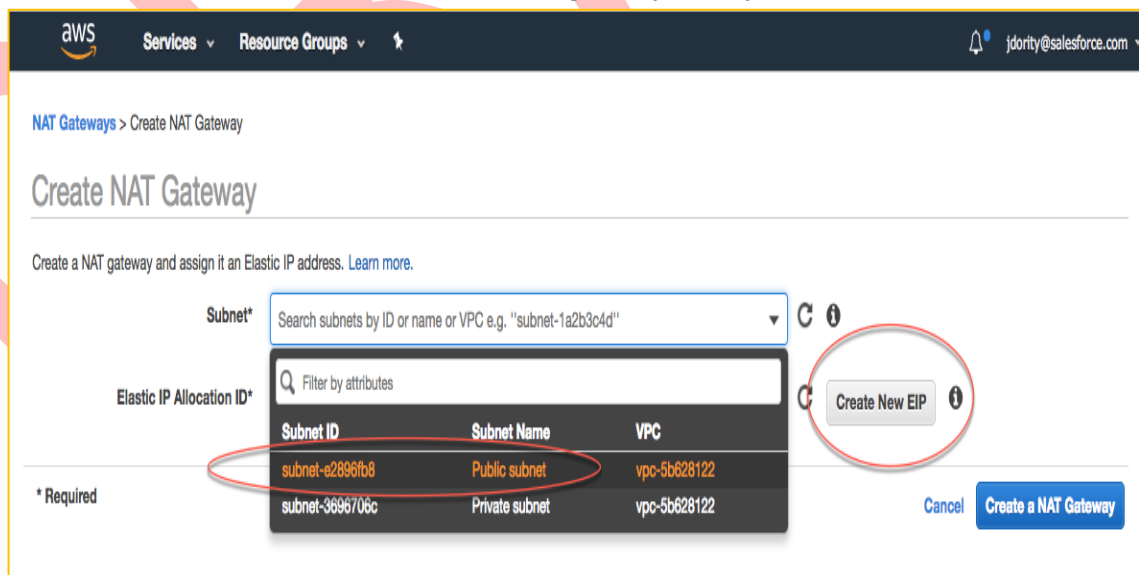
A Private Subnet using a NAT Gateway will BLOCK internet traffic such as a SQL client from connecting to the databases. If the Private Subnet uses the IGW access is then access is possible.

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

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

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

**VPC Dashboard**

Filter by VPC:

**Virtual Private Cloud**

- Your VPCs
- Subnets
- Route Tables**
- Internet Gateways
- Egress Only Internet Gateways
- DHCP Options Sets
- Elastic IPs
- Endpoints
- Endpoint Services
- NAT Gateways

**Create Route Table** **Delete Route Table** **Set As Main Table**

Search Route Tables and their X

Name	Route Table ID	Explicitly Associat	Main	VPC
RT for Public Subnet	rtb-ef755f96	1 Subnet	No	vpc-5b628122
<b>RT for Private Subnet</b>	<b>rtb-acb6a0d5</b>	1 Subnet	<b>Yes</b>	vpc-5b628122

**rtb-acb6a0d5 | RT for Private Subnet**

**Edit**

View: All rules

Destination	Target	Status	Propagated
192.168.0.0/16	local	Active	No
0.0.0.0/0	<b>nat-0edcde7fcd3450734</b>	Active	No
pl-68a54001 (com.amazonaws.us-west-2.s3)	vpce-39e22250	Active	No

**Summary** **Routes** **Subnet Associations**

**Edit**

Subnet	IPv4 CIDR	IPv6 CIDR
subnet-3696706c <b>Private subnet</b>	192.168.1.0/24	-

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Subnet	IPv4 CIDR	IPv6 CIDR
--------	-----------	-----------

All your subnets are associated with a route table.

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

**VPC Dashboard**

Filter by VPC:

**Virtual Private Cloud**

- Your VPCs
- Subnets
- Route Tables**
- Internet Gateways
- Egress Only Internet Gateways
- DHCP Options Sets
- Elastic IPs
- Endpoints
- Endpoint Services
- NAT Gateways
- Peering Connections

**Create Route Table** **Delete Route Table** **Set As Main Table**

Search Route Tables and their X

Name	Route Table ID	Explicitly Associat	Main	VPC
<b>RT for Public Subnet</b>	<b>rtb-ef755f96</b>	1 Subnet	<b>No</b>	vpc-5b628122
RT for Private Subnet	rtb-acb6a0d5	1 Subnet	Yes	vpc-5b628122

**rtb-ef755f96 | RT for Public Subnet**

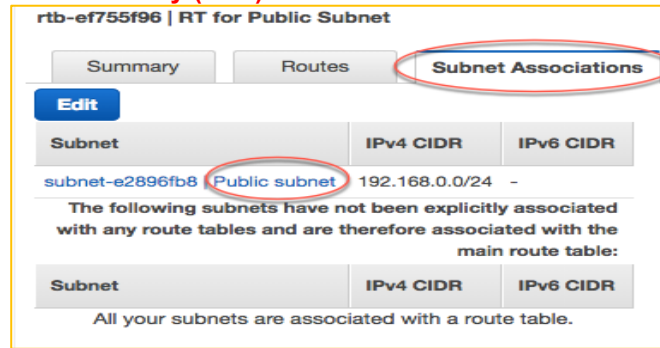
**Edit**

View: All rules

Destination	Target	Status	Propagated
192.168.0.0/16	local	Active	No
0.0.0.0/0	<b>igw-4a826f2c</b>	Active	No
pl-68a54001 (com.amazonaws.us-west-2.s3)	vpce-39e22250	Active	No



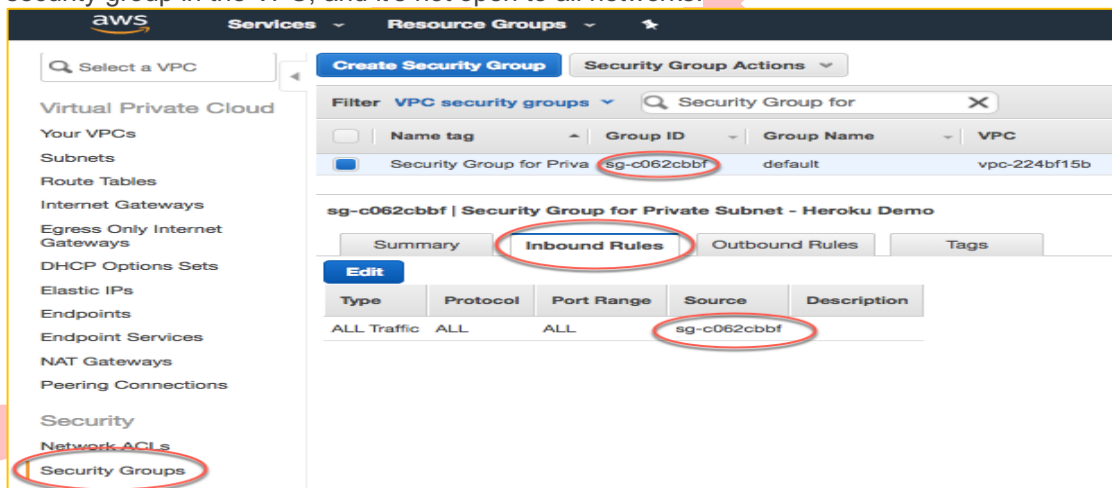
- a. Associate the **public subnet** to this new route table and edit the route so that all traffic goes through the Internet Gateway (IGW):



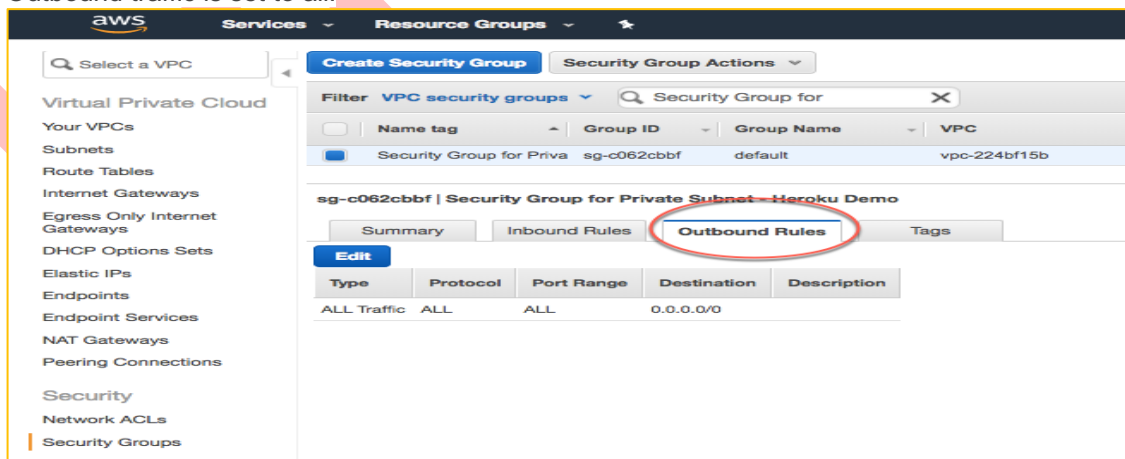
Instances in the public subnet can send outbound traffic directly to the Internet, whereas the instances in the private subnet can't. The database servers can connect to the Internet for software updates using the NAT gateway, **but the Internet cannot establish connections to the database servers.**  
<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. Inbound 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.

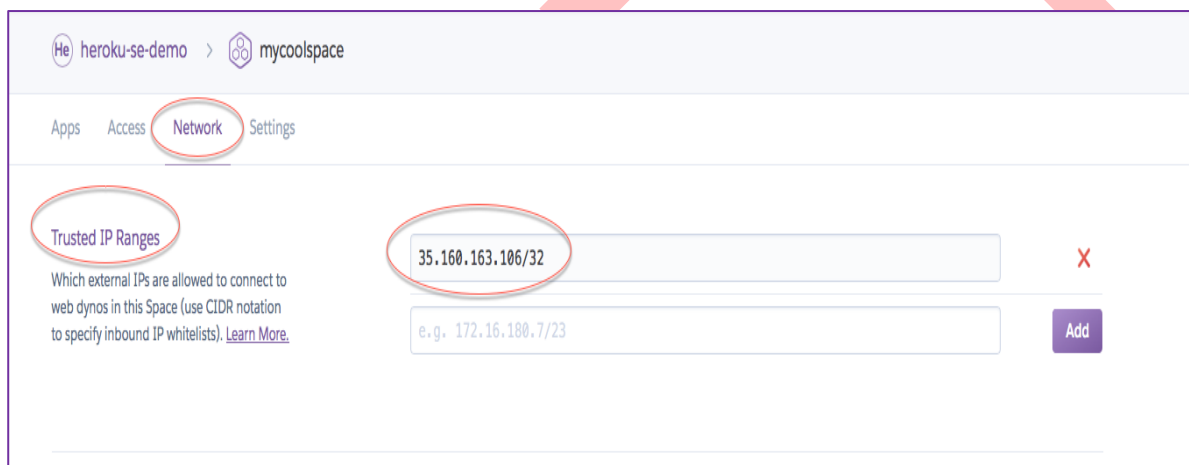
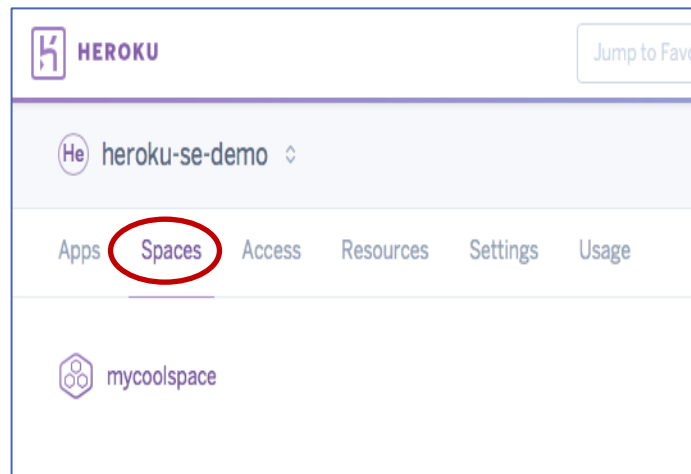


- b. Outbound traffic is set to all.



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

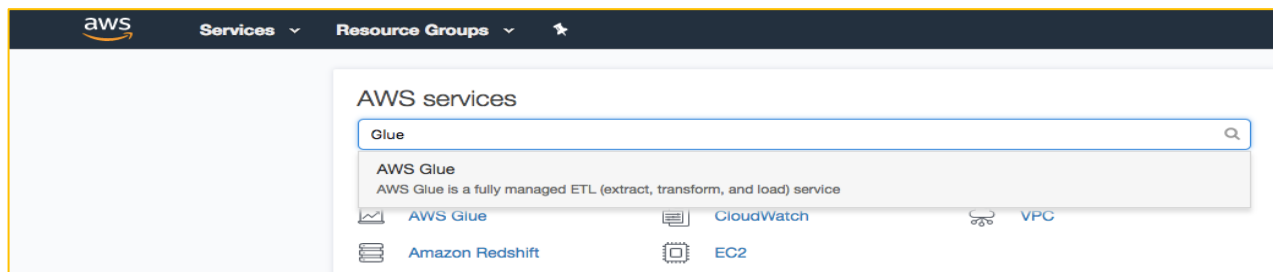
7. After configuring a Private Space and Private Postgres Database, white list the Elastic IP that was configured in step 4 (Creating the NAT Gateway).



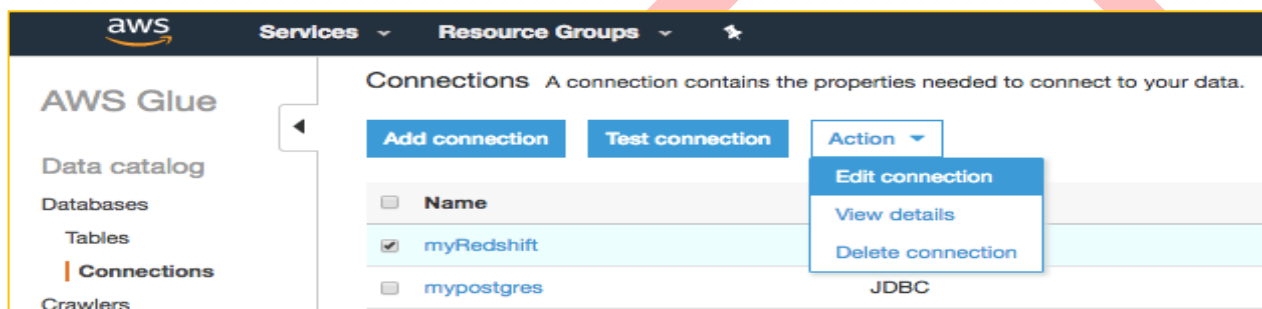
## Amazon Glue Example Configuration:

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.

**Set up access to your data store.**

For more information, see [Working with Connection](#).

**JDBC URL**

JDBC syntax for most database engines is jdbc:protocol://host:port/databasename.

SQL Server syntax is jdbc:sqlserver://host:port/databaseName=db\_name. Oracle syntax is jdbc:oracle:thin://@host:port/service\_name. For more variations, see [Working with Connection](#).

**Username**

**Password**

**VPC**

Choose the VPC name that contains your data store.

**Subnet**

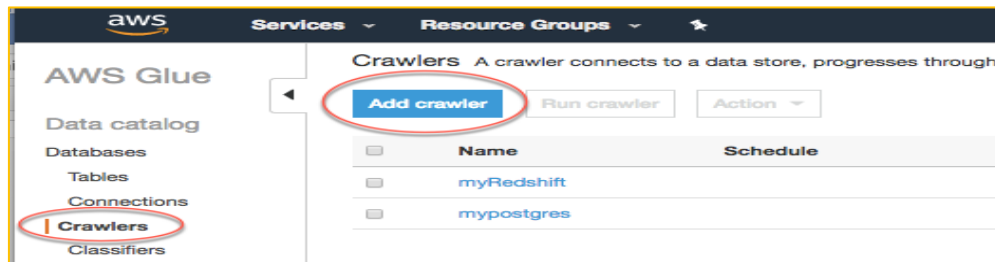
Choose the subnet within your VPC.

**Security groups**

Choose one or more security groups that allow access to the data store in your VPC. AWS Glue associates these security groups to the ENI attached to your subnet. To allow AWS Glue components to communicate and also prevent access from other networks, at least one chosen security group must specify a self-referencing inbound rule for all TCP ports.

Group ID	Group name
<input type="checkbox"/> sg-ba899cc5	Cluster Bomb Security Group
<input checked="" type="checkbox"/> sg-c062cbbf	default

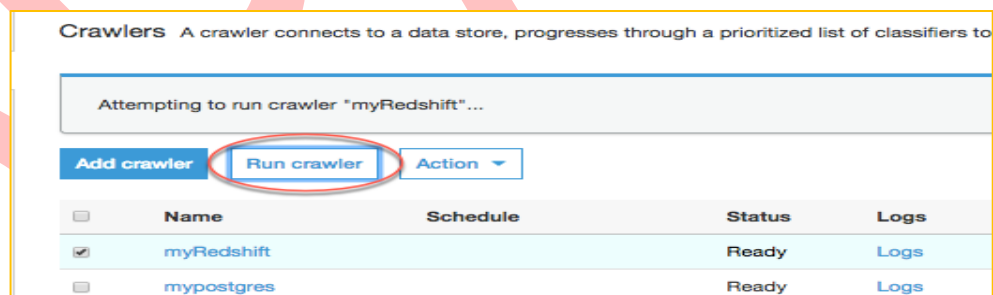
- After your connections are configured, add a metadata crawler to collect metadata from your source and target tables.



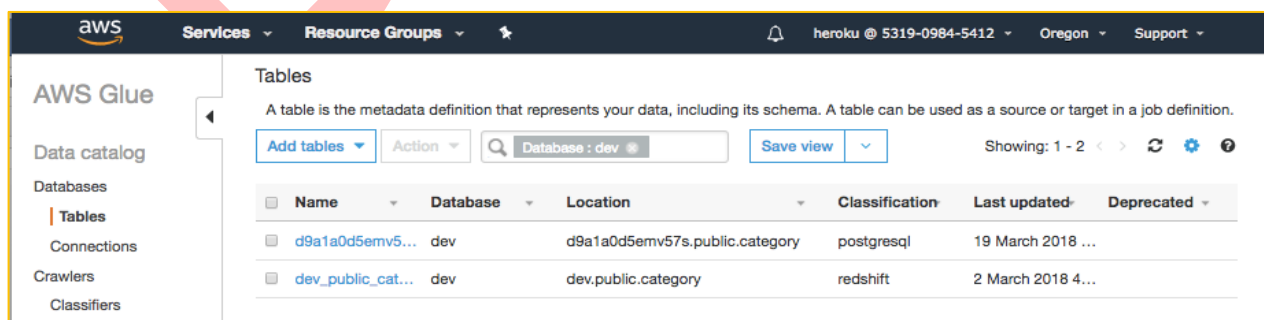
- Provide the connection you created in Step 2 and specify the database/schema/tables you wish to collect metadata from:

The screenshot shows the 'Add a data store' form. The 'Data store' dropdown is set to 'JDBC'. The 'Connection' dropdown is set to 'myRedshift' (circled in red). Below it is an 'Add connection' button. The 'Include path' field contains 'dev/public/category' (circled in red). A note below the field states: 'You can substitute the multi-character (%) wildcard for a schema or table. For example, MyDatabase/MySchema/% matches all tables in MySchema within MyDatabase.' There is an 'Exclude patterns (optional)' section. At the bottom are 'Back' and 'Next' buttons.

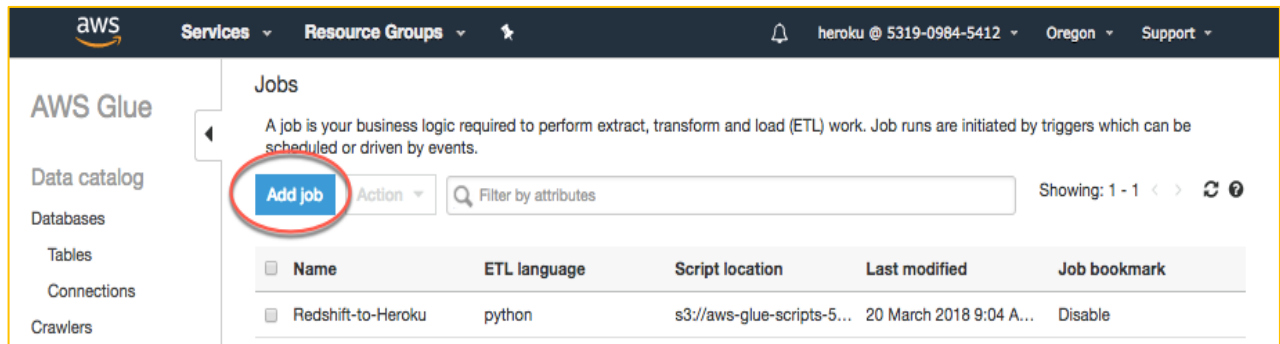
After configuring the crawler, "Run" it:



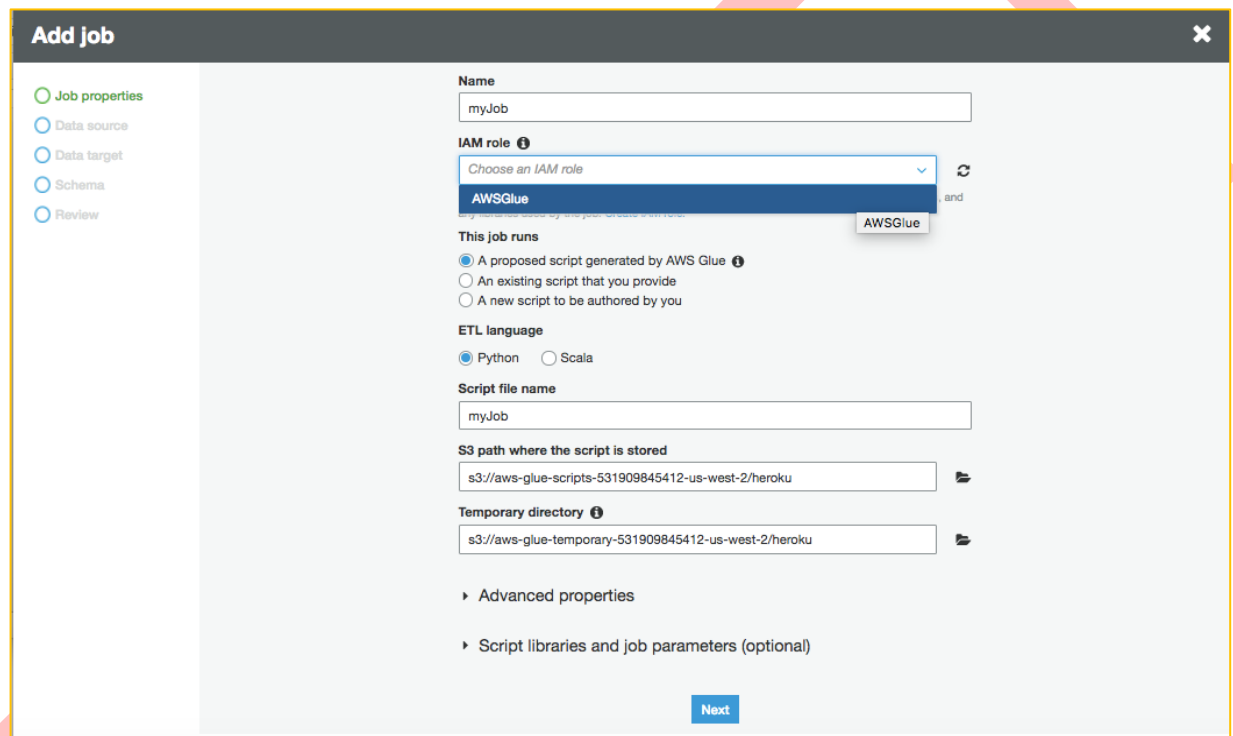
- When the crawler has finished running, you can see the table metadata in the Database-→Tables section:



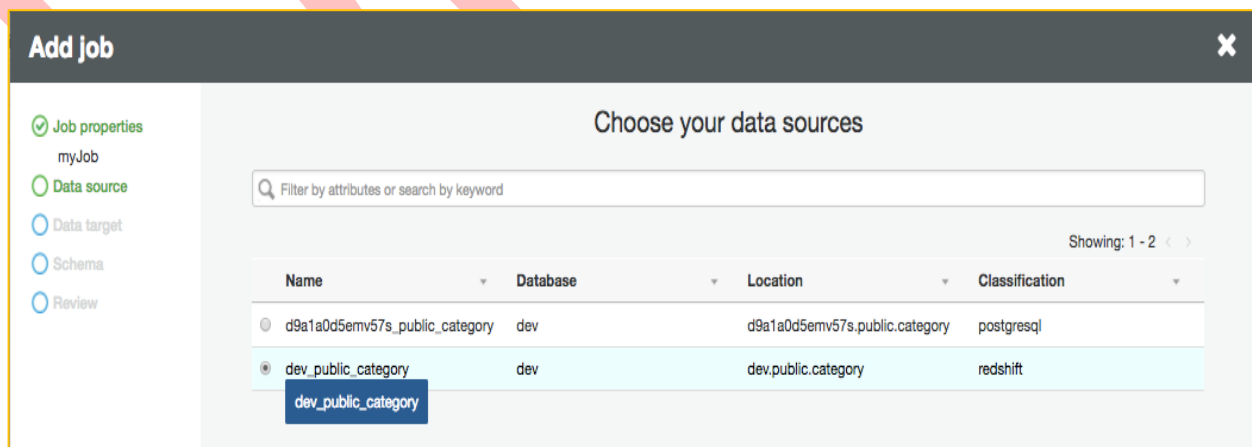
6. Once you have the table metadata, you are ready to create a Job:



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:

**Add job**

Choose your data targets

☐ Create tables in your data target  
☒ Use tables in the data catalog and update your data target

Filter by attributes or search by keyword

Showing: 1 - 2

Name	Database	Location	Classification
d9a1a0d5emv57s_public_category	dev	d9a1a0d5emv57s.public.category	postgresql
dev_public_category	dev	dev.public.category	redshift

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

**Add job**

Map the source columns to target columns.

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with **Map to target**. You can **Clear** all mappings and **Reset** to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

[Clear](#) [Reset](#)

Source Column name	Data type	Map to target
catid	string	catid__c
catname	string	name
catdesc	string	catdesc__c
catdesc	string	catgroup__c

Target Column name	Data type
catid__c	string
name	string
catdesc__c	string
catgroup__c	string
createddate	timestamp
_hc_lastop	string
_hc_err	string
sfid	string
id	int
isdeleted	boolean
systemmodstamp	timestamp

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

**Add job**

Job properties

Name: myJob  
IAM role: AWSGlue  
ETL language: python  
Scala class name: Path  
Temporary directory: s3://aws-glue-temporary-531909845412-us-west-2/heroku

Advanced properties

Script libraries and job parameters (optional)

Data sources

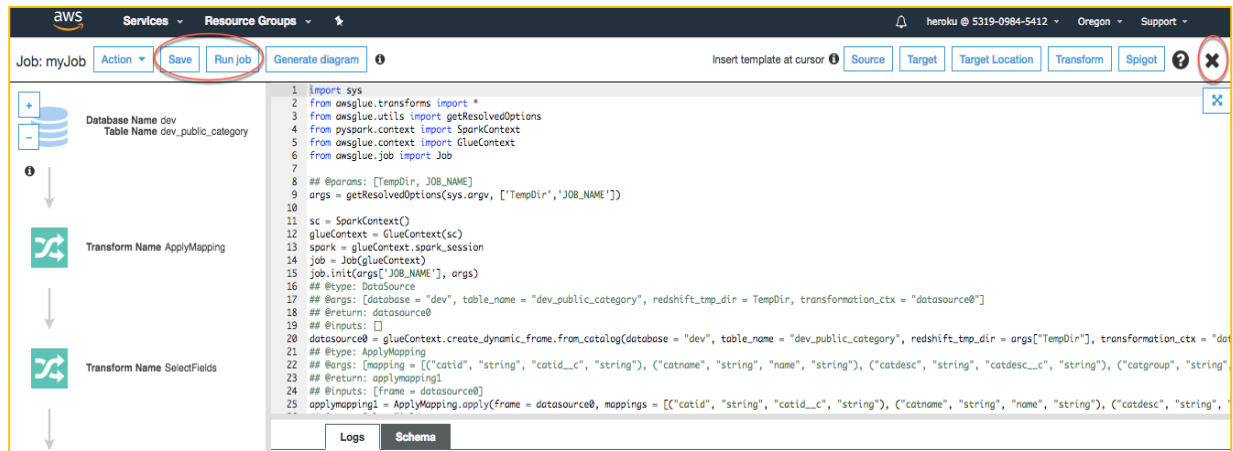
Tables: dev\_public\_category

Data targets

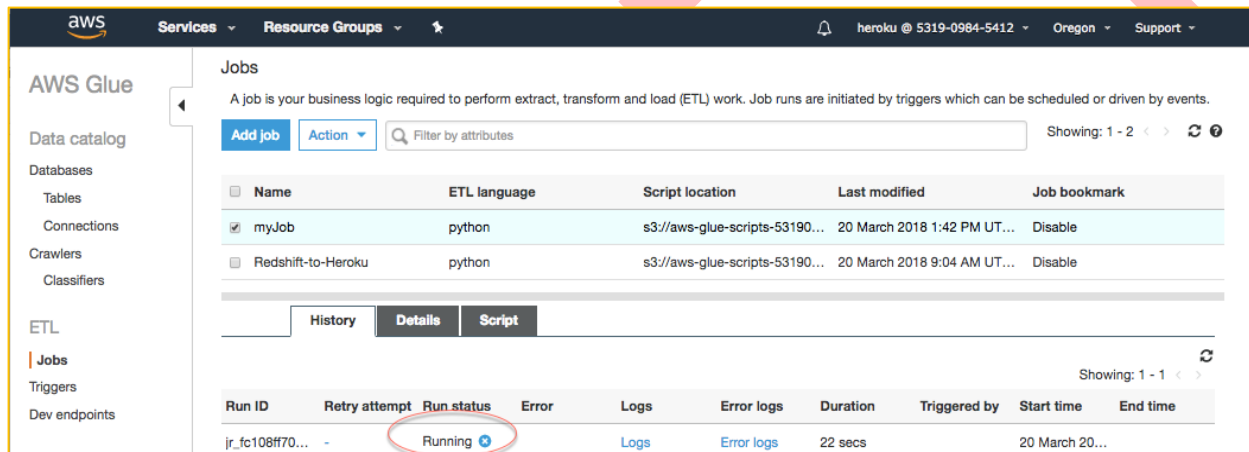
Tables: d9a1a0d5emv57s\_public\_category

[Back](#) [Finish](#)

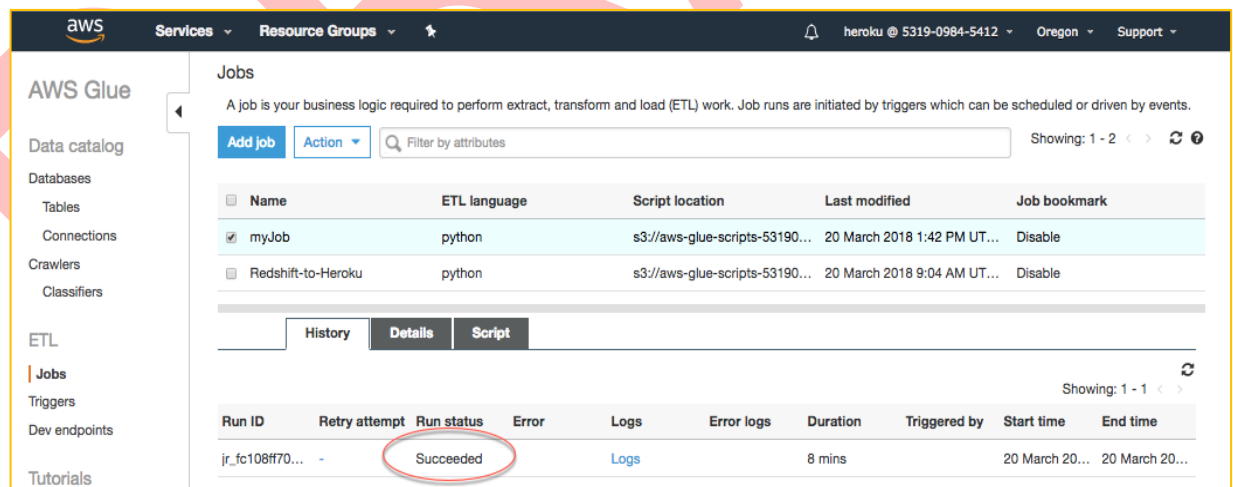
11. Click "Save" and then "Run Job". To exit the script/diagram screen click the "x" in the right upper hand corner.



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](https://github.com/jdority/redshift-to-salesforce/blob/master/category_setup.sql)