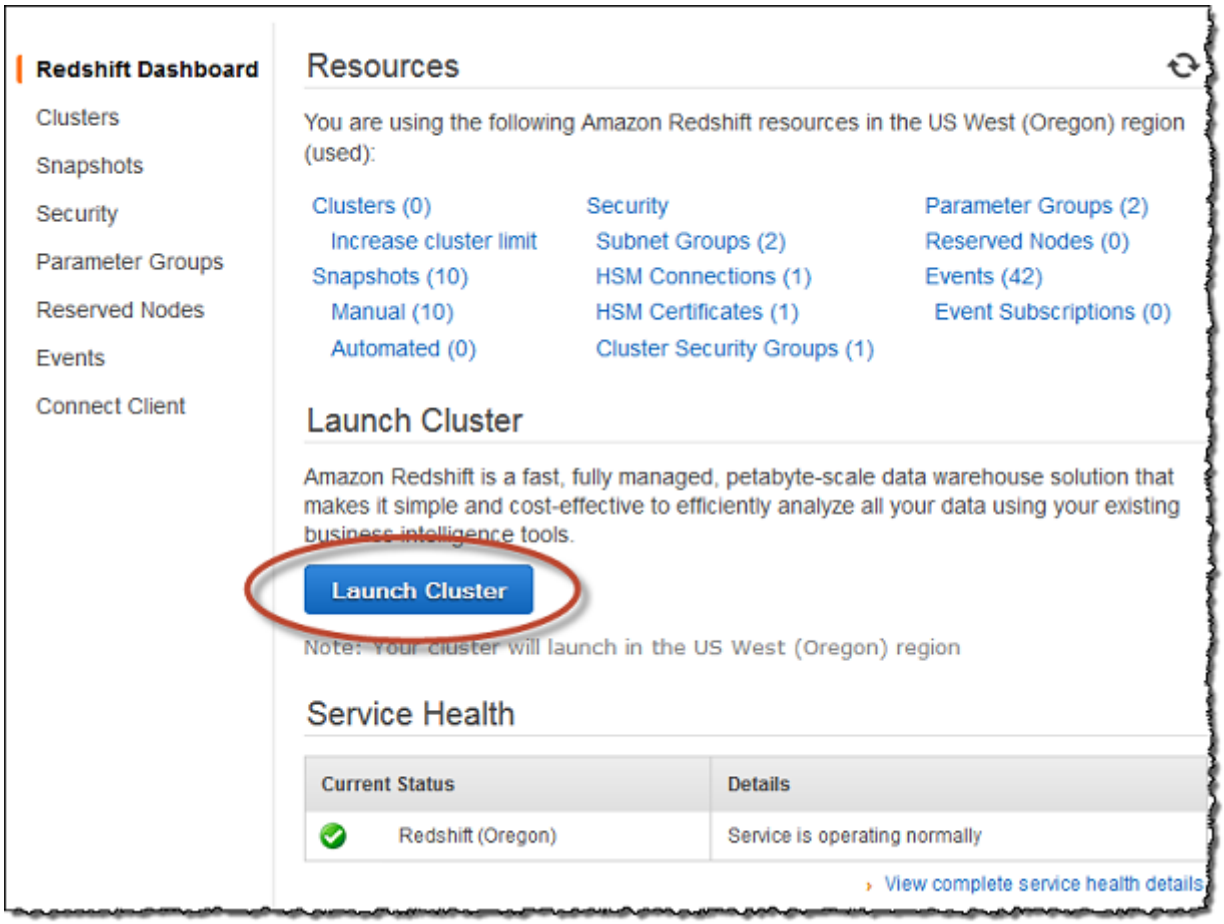
Lab 4: Getting Started with Amazon Redshift

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. An **Amazon Redshift** data warehouse is a collection of computing resources called *nodes*, which are organized into a group called a *cluster*. Each cluster runs an **Amazon Redshift** engine and contains one or more databases.

Step 1: Launch a Sample Amazon Redshift Cluster

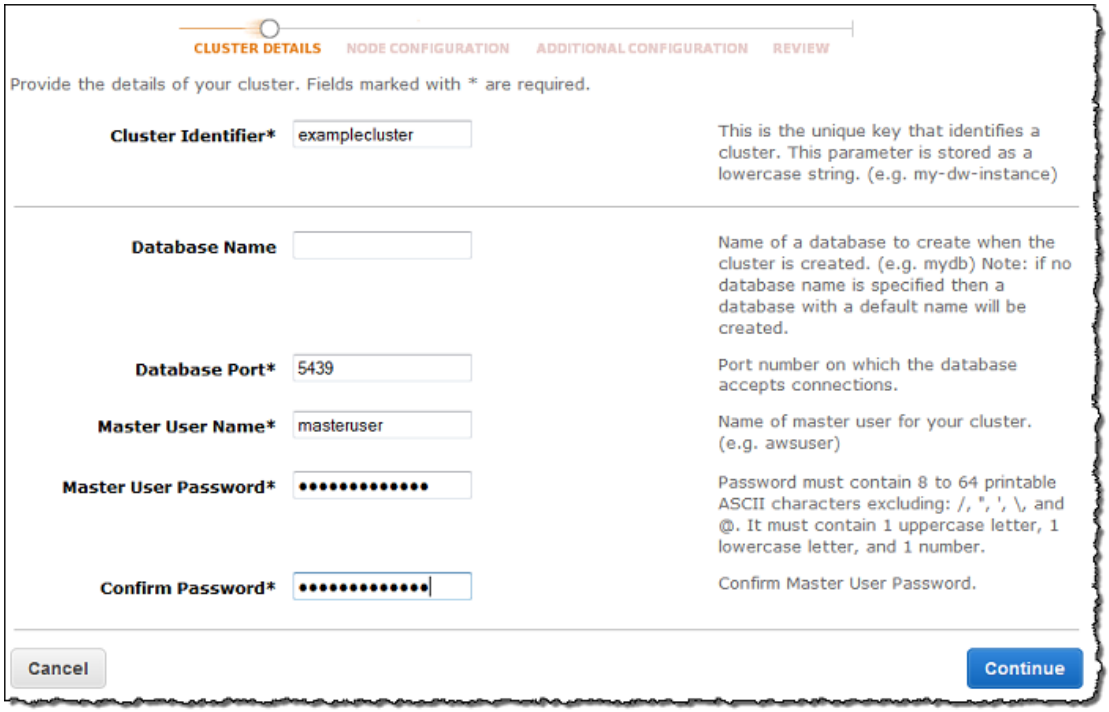
1. In the main menu, select the **N. Virginia region (us-east-1)**

2. On the Amazon **Redshift** Dashboard, click **Launch Cluster**.
   The Amazon **Redshift** Dashboard looks similar to the following:



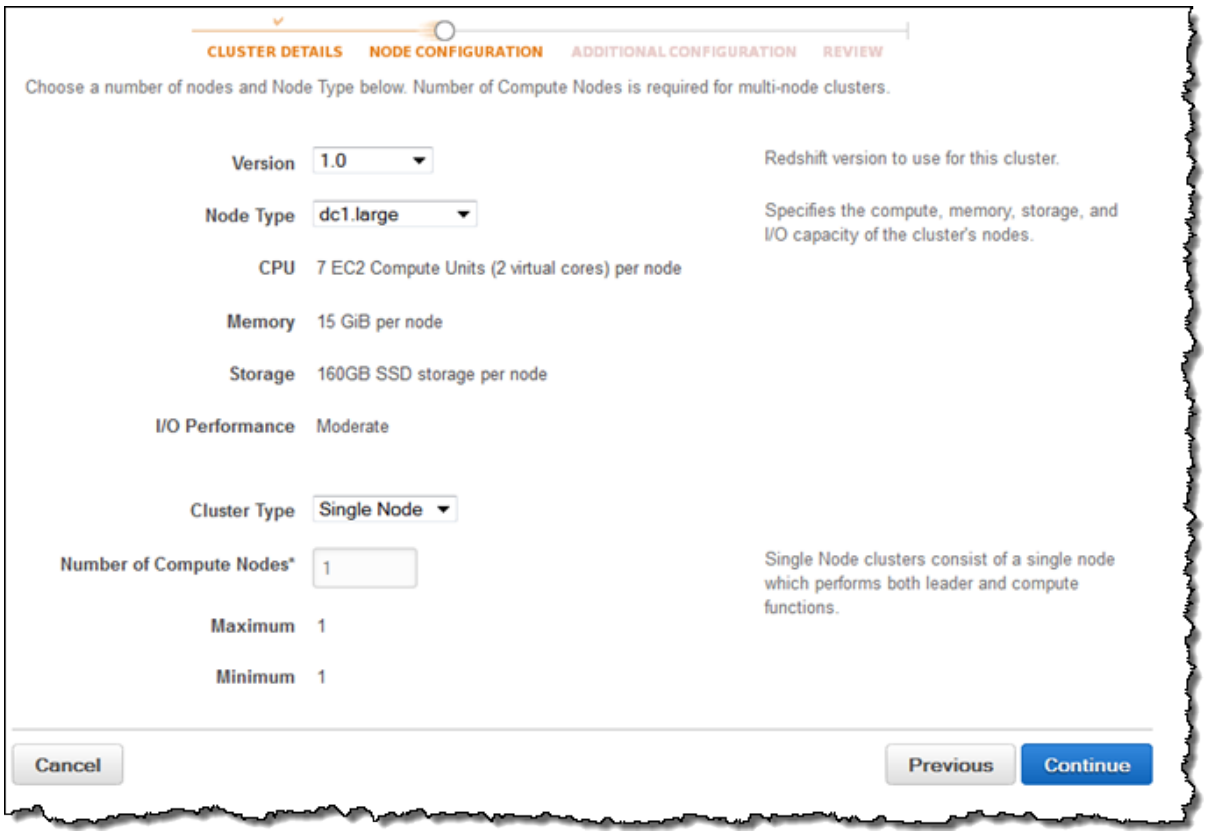
3. On the Cluster Details page, enter the following values and then click **Continue**:

- Cluster Identifier: type **<user>-redshift**.
- Database Name: **<user>**
- Database Port: **5439**
- Master User Name: **masteruser** (You will use this username and password to connect to your database after the cluster is available).
- Master User Password and Confirm Password: **type a password for the master user account.**

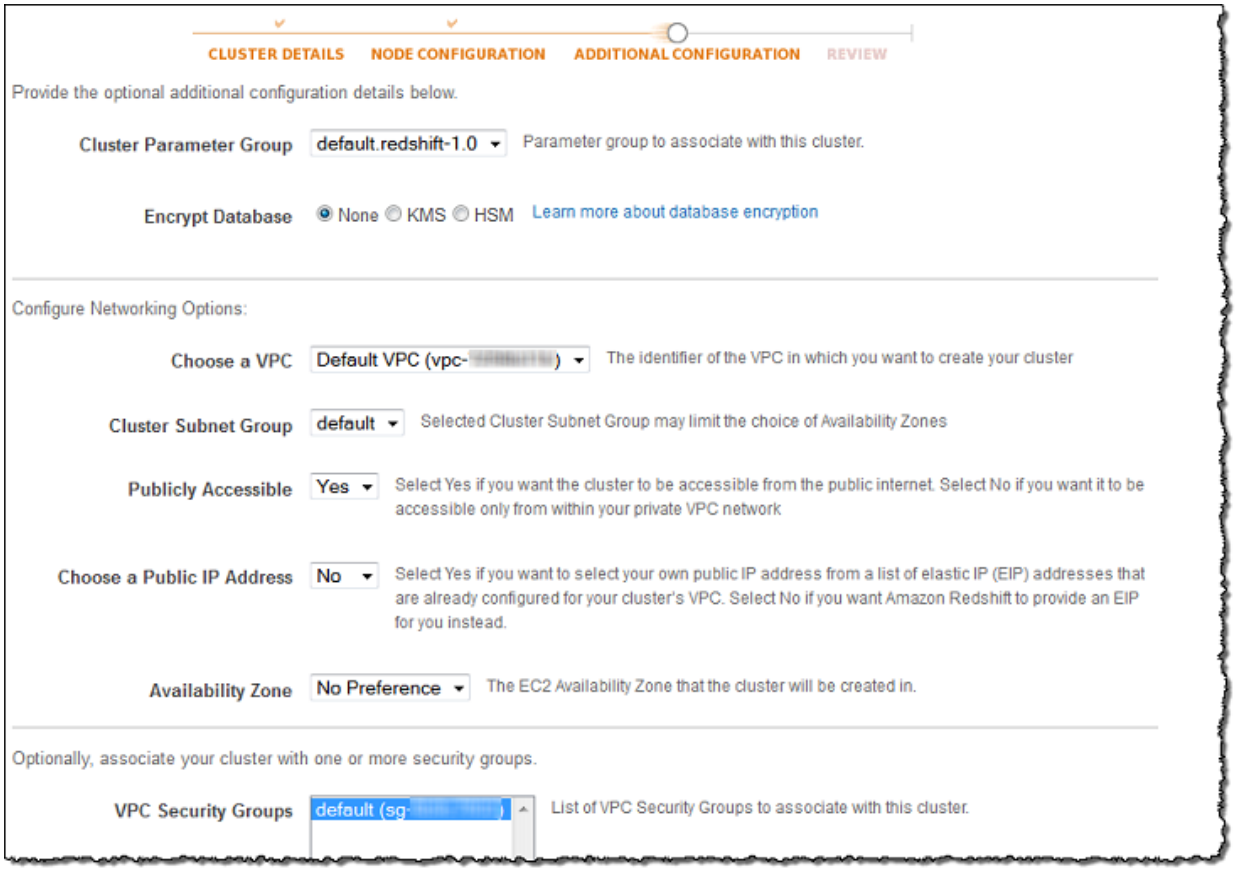


4. On the **Node Configuration** page, select the following values and then click **Continue**:

- Node Type: **dc1.large**
- Cluster Type: **Single Node**



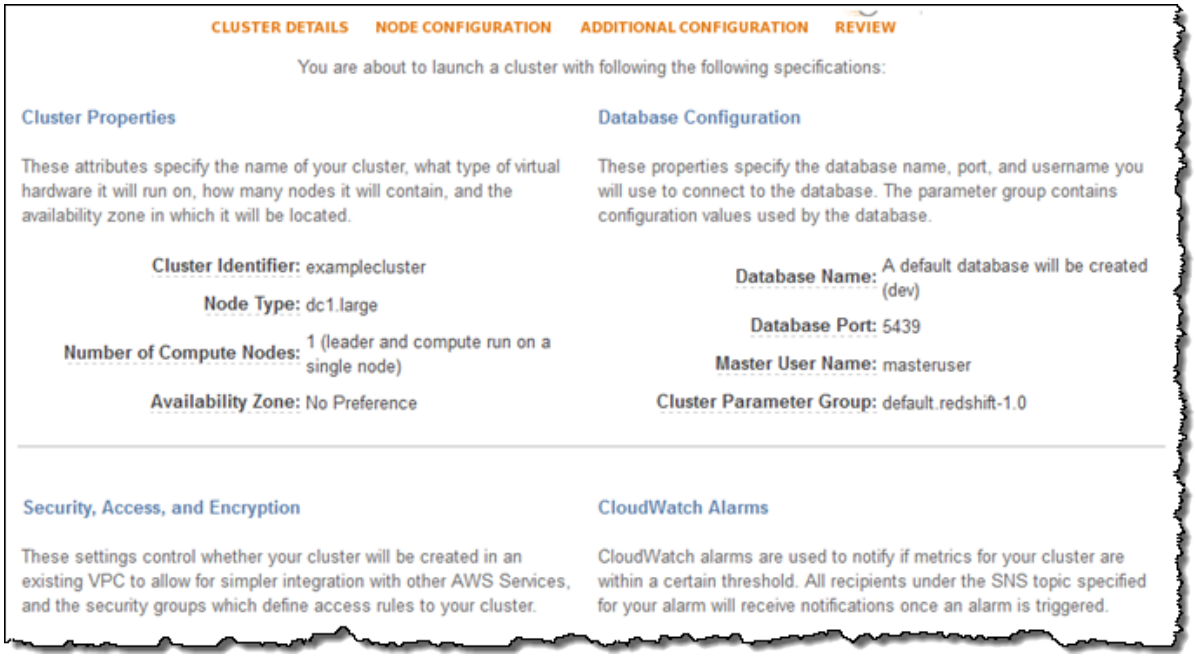
5. On the **Additional Configuration** page, you will see different options depending on your AWS account, which determines the type of platform the cluster uses.



Use the following values if you are launching your cluster in the EC2-VPC platform:

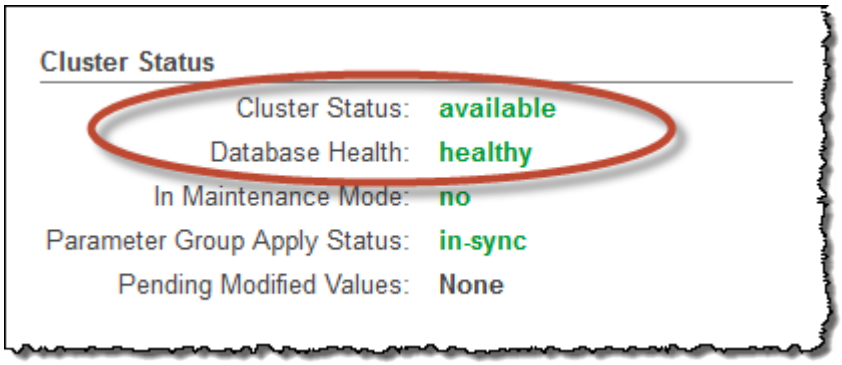
- Cluster Parameter Group: select the **default** parameter group.
- Encrypt Database: **None**.
- Choose a VPC: **Default VPC** (vpc-xxxxxxxx)
- Cluster Subnet Group: **default**
- Publicly Accessible: **Yes**
- Choose a Public IP Address: **No**
- Availability Zone: **us-east-1d**
- VPC Security Groups: **default** (sg-xxxxxxxx)
- Create CloudWatch Alarm: **No**
- Click **Continue**

6. On the Review page, review the selections that you've made and then click **Launch Cluster**.
   Your screen will look similar to the following:



7. A confirmation page appears and the cluster will take a few minutes to finish. Click **Close** to return to the list of clusters.

8. On the **Clusters** page, click the cluster that you just launched and review the **Cluster Status** information. Make sure that the Cluster Status is **available** and the Database Health is **healthy** before you try to connect to the database later in this tutorial.



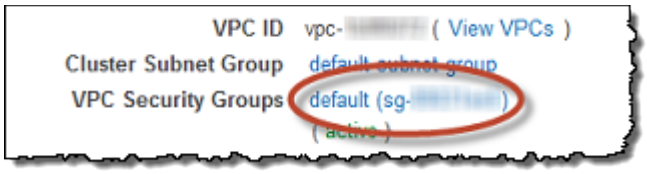
Step 3: Authorize Access to the Cluster

In the previous step, you launched your **Amazon Redshift** cluster. Before you can connect to the cluster, you need to configure a security group to authorize access.

To Configure the VPC Security Group

Note: This step doesn't need to be done if using a shared account.

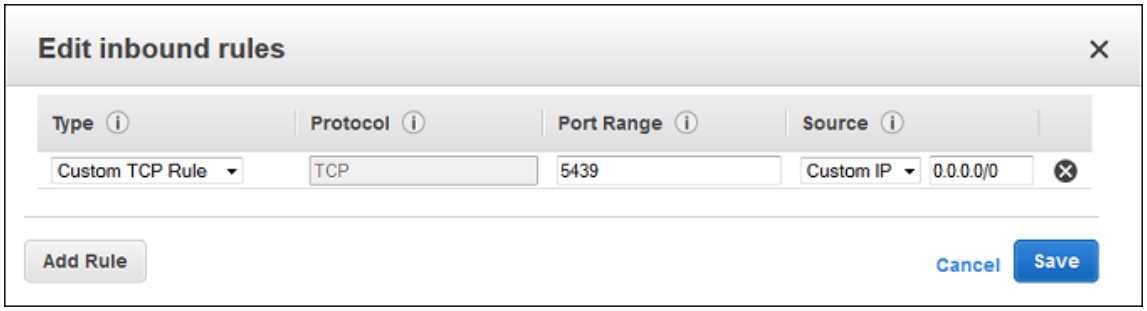
1. In the **Amazon Redshift** console, in the navigation pane, click **Clusters**.
2. Click **<user>-redshift** to open it, and make sure you are on the **Configuration tab**.
3. Under **Cluster Properties**, for **VPC Security Groups**, click your security group.



4. After your security group opens in the **Amazon EC2 console**, click the **Inbound** tab.

5. Click **Edit**, and enter the following, then click **Save**:
   - Type: **Custom TCP Rule**.
   - Protocol: **TCP**.
   - Port Range: type the same port number that you used when you launched the cluster. The default port for **Amazon Redshift** is **5439**, but your port might be different.
   - Source: select **Custom IP**, then type **0.0.0.0/0.**

   Important
   Using 0.0.0.0/0 is not recommended for anything other than demonstration purposes because it allows access from any computer on the internet. In a real environment, you would create inbound rules based on your own network settings.



Step 4: Install SQL Workbench/J on Your Client Computer

Note: If you have any other SQL client, you do not need to download SQL Workbench.

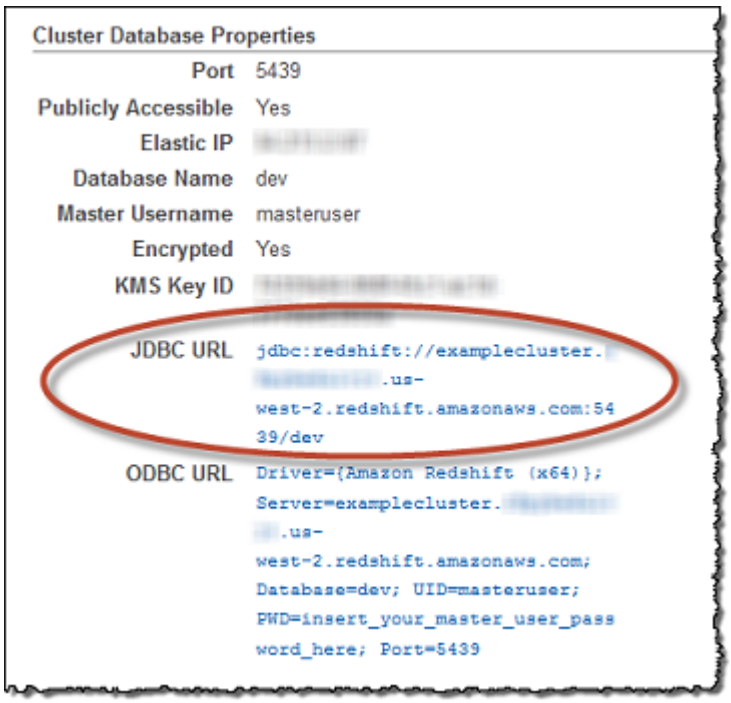
1. Go to the SQL Workbench/J website and download the appropriate package for your operating system.
2. Go to the Installing and starting SQL Workbench/J page and install SQL Workbench/J.

   Important
   Note the Java runtime version prerequisites for SQL Workbench/J and ensure you are using that version, otherwise, this client application will not run.

To Get Your Connection String

1. In the **Amazon Redshift** console, in the navigation pane, click **Clusters**.
2. Click **<user>-redshift** to open it, and make sure you are on the **Configuration** tab.
3. On the **Configuration** tab, under **Cluster Database Properties**, copy the **JDBC URL** of the cluster.

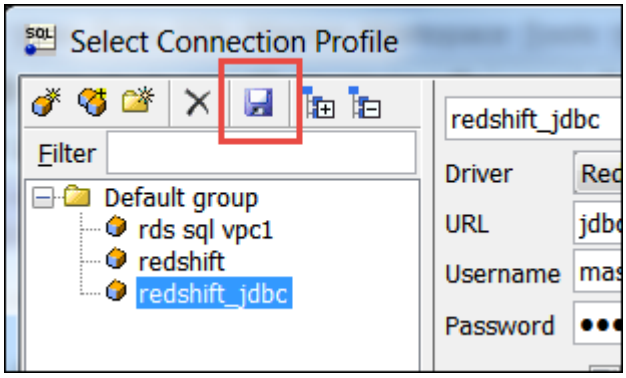
   Note
   The endpoint for your cluster is not available until the cluster is created and in the available state.

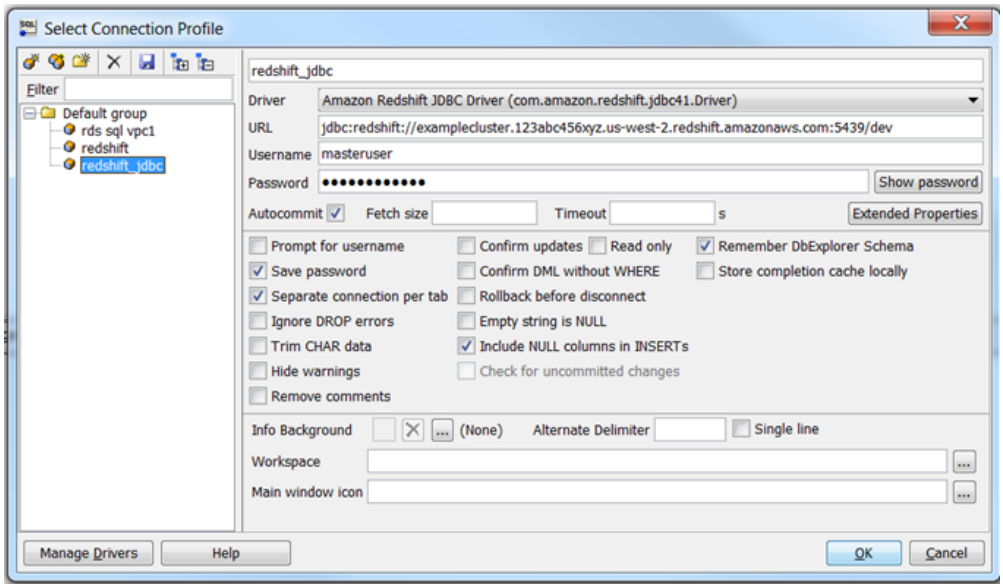


To Connect from SQL Workbench/J to Your Cluster

This step assumes you installed SQL Workbench/J in **Step 1: Set Up Prerequisites**.

1. In the Redshift console, go to **Connect Client** in the left menu.
2. Download the JDBC 4.2 driver.
3. Open SQL Workbench/J.
4. Choose **File**, and then choose **Connect window.**
5. Click on **Manage Drivers**
6. Choose **Amazon Redshift** and update the driver with the one downloaded from the Redshift console. Click Ok.
7. Create **New Connection Profile**
8. In the New profile text box, type a name for the profile.
9. Driver: **Amazon Redshift**
1. Username: **masteruser**.
2. In Password, type the password associated with the master user account.
3. Choose the **Autocommit** box.
4. Choose the **Save** profile list icon, as shown below:





Choose **OK**.

At this point you have a database called <user> and you are connected to it. We are going to query from Redshift the data stored in S3 our parquet_parquet table that is defined in our <user> database in the Amazon Catalogue.

First, we need to add a role to our **Redshift** cluster so that it has the right to query **S3** and the catalogue:

1. In the Redshift console, click on Clusters and pick your cluster: **<user>-redshift**.
2. Click on **Manage IAM Roles**
3. Assign the **RedshiftSpectrumRole**. (if you don't have it, create it first by attaching at least the policies **AmazonS3ReadOnlyAccess** and **AWSGlueConsoleFullAccess**)
4. Click **Apply**.

Run the following query in SQL Workbench to create an external schema called **spectrum_parquet** in Redshift from the <user> database in the Amazon Catalogue:

create external schema spectrum_parquet from data catalog

database '<user>'

iam_role 'arn:aws:iam::<ACCOUNT_ID>:role/RedshiftSpectrumRole'

region 'us-east-1';

Now we can query our data is **S3** by running this query:

*select * from spectrum_parquet.parquet_parquet;*

Step 5: Load Sample Data from Amazon S3

Now you will create some tables in the database, and upload data to the internal tables, and try a query. For your convenience, the sample data you will load is available in **Amazon S3** buckets. To copy this sample data, you will need your AWS account credentials (access key ID and secret access key). Only authenticated users can access this data.

> **Note**
> Before you proceed, ensure that your SQL Workbench/J client is connected to the cluster.

1. Create the tables.

Copy and execute the following create table statements to create tables in the database. For more information about the syntax, go to CREATE TABLE in the *Amazon Redshift Database Developer Guide*.

Copy the following data in the Statement section of SQLWorkbench and execute it.

```
create table users(
 userid integer not null distkey sortkey,
 username char(8),
 firstname varchar(30),
 lastname varchar(30),
 city varchar(30),
 state char(2),
 email varchar(100),
 phone char(14),
 likesports boolean,
 liketheatre boolean,
 likeconcerts boolean,
 likejazz boolean,
 likeclassical boolean,
 likeopera boolean,
 likerock boolean,
 likevegas boolean,
 likebroadway boolean,
 likemusicals boolean);

create table venue(
```

```
 venueid smallint not null distkey sortkey,
 venuename varchar(100),
 venuecity varchar(30),
 venuestate char(2),
 venueseats integer);

create table category(
 catid smallint not null distkey sortkey,
 catgroup varchar(10),
 catname varchar(10),
 catdesc varchar(50));

create table date(
 dateid smallint not null distkey sortkey,
 caldate date not null,
 day character(3) not null,
 week smallint not null,
 month character(5) not null,
 qtr character(5) not null,
 year smallint not null,
 holiday boolean default('N'));

create table event(
 eventid integer not null distkey,
 venueid smallint not null,
 catid smallint not null,
 dateid smallint not null sortkey,
 eventname varchar(200),
 starttime timestamp);

create table listing(
 listid integer not null distkey,
 sellerid integer not null,
 eventid integer not null,
 dateid smallint not null  sortkey,
 numtickets smallint not null,
 priceperticket decimal(8,2),
 totalprice decimal(8,2),
 listtime timestamp);

create table sales(
 salesid integer not null,
 listid integer not null distkey,
 sellerid integer not null,
 buyerid integer not null,
 eventid integer not null,
 dateid smallint not null sortkey,
 qtysold smallint not null,
 pricepaid decimal(8,2),
 commission decimal(8,2),
 saletime timestamp);
```

2. Load sample data from **Amazon S3** by using the COPY command.

   Note
   In **Amazon Redshift**, the COPY command is the recommended method to optimize performance during bulk loading of large datasets from **Amazon S3** or **DynamoDB**. For more information COPY syntax, go to COPY in the *Amazon Redshift Database Developer Guide*.

The sample data for this tutorial is provided in Amazon S3 buckets. The bucket permissions are configured to allow all authenticated AWS users read access to the sample data files. To load the sample data, make sure you have the following for your **IAM** user:

- Your access key and secret access key. If you do not know these, you can create new ones. For more information, go to Administering Access Keys for IAM Users in *IAM User Guide*.

To load the sample data by using the following COPY command, replace *<access-key-id>* and *<secret-access-key>* with the access key and secret access key for your **IAM** user. Then run the command in your SQL client tool.

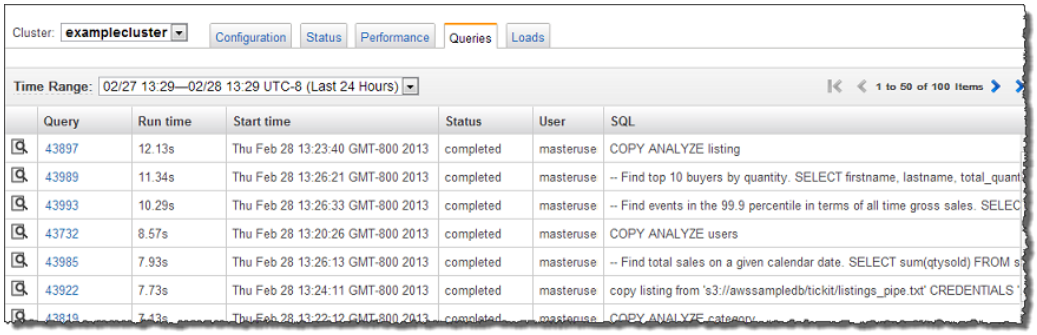
```
copy users from 's3://romerfra-
datasets/allusers_pipe.txt'
credentials 'aws_access_key_id=<access-key-
id>;aws_secret_access_key=<secret-access-key>'
delimiter '|' region 'us-east-1';

copy venue from 's3://romerfra-datasets/venue_pipe.txt'
credentials 'aws_access_key_id=<access-key-
id>;aws_secret_access_key=<secret-access-key>'
delimiter '|' region 'us-east-1';

copy category from 's3://romerfra-
datasets/category_pipe.txt'
credentials 'aws_access_key_id=<access-key-
id>;aws_secret_access_key=<secret-access-key>'
delimiter '|' region 'us-east-1';

copy date from 's3://romerfra-datasets/date2008_pipe.txt'
credentials 'aws_access_key_id=<access-key-
id>;aws_secret_access_key=<secret-access-key>'
delimiter '|' region 'us-east-1';

copy event from 's3://romerfra-
datasets/allevents_pipe.txt'
credentials 'aws_access_key_id=<access-key-
id>;aws_secret_access_key=<secret-access-key>'
delimiter '|' timeformat 'YYYY-MM-DD HH:MI:SS' region
'us-east-1';

copy listing from 's3://romerfra-
datasets/listings_pipe.txt'
credentials 'aws_access_key_id=<access-key-
id>;aws_secret_access_key=<secret-access-key>'
delimiter '|' region 'us-east-1';

copy sales from 's3://romerfra-datasets/sales_tab.txt'
credentials 'aws_access_key_id=<access-key-
id>;aws_secret_access_key=<secret-access-key>'
delimiter '\t' timeformat 'MM/DD/YYYY HH:MI:SS' region
'us-east-1';
```

3. Now try the example queries. For more information, go to SELECT in the *Amazon Redshift Developer Guide*.

```
-- Get definition for the sales table.
SELECT *
FROM pg_table_def
WHERE tablename = 'sales';

-- Find total sales on a given calendar date.
SELECT sum(qtysold)
FROM   sales, date
WHERE  sales.dateid = date.dateid
AND    caldate = '2008-01-05';

-- Find top 10 buyers by quantity.
SELECT firstname, lastname, total_quantity
FROM   (SELECT buyerid, sum(qtysold) total_quantity
        FROM  sales
        GROUP BY buyerid
        ORDER BY total_quantity desc limit 10) Q, users
WHERE Q.buyerid = userid
ORDER BY Q.total_quantity desc;

-- Find events in the 99.9 percentile in terms of all
time gross sales.
SELECT eventname, total_price
FROM  (SELECT eventid, total_price, ntile(1000)
over(order by total_price desc) as percentile
       FROM (SELECT eventid, sum(pricepaid) total_price
             FROM   sales
             GROUP BY eventid)) Q, event E
       WHERE Q.eventid = E.eventid
       AND percentile = 1
ORDER BY total_price desc;
```

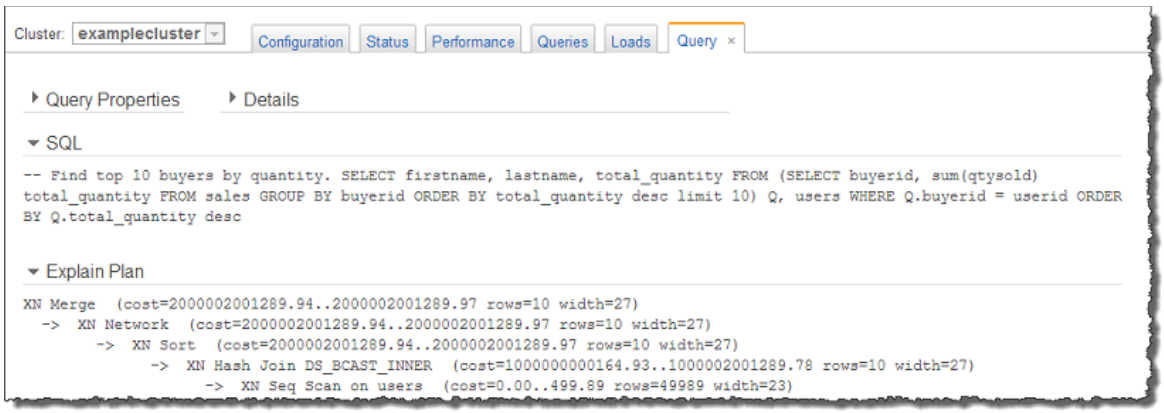
4. You can optionally go the **Amazon Redshift** console to review the queries you executed. The Queries tab shows a list of queries that you executed over a time period you specify. By default, the console displays queries that have executed in the last 24 hours, including currently executing queries.
   - Sign in to the **AWS Management Console** and open the **Amazon Redshift** console at https://console.aws.amazon.com/redshift/.
   - In the cluster list in the right pane, click **<user>-redshift**.
   - Click the **Queries** tab.

   The console displays list of queries you executed as shown in the example below.



   - In the list of queries, select a query to find more information about it.

The query information appears in a new **Query** tab. The following example shows the details of a query you ran in a previous step.



Step 6: Visualize data from Redshift with Quicksight

1. Open the **QuickSight console**
2. Click **New Analysis**
3. Click **New Dataset**
4. Choose **Redshift Auto-Discovered**
    a. Data source name: **<user>-redshift**
    b. Instance ID: **<user>-redshift**
    c. Database Name: **<user>**
    d. Username: **masteruser**
    e. Password**: <Password>**
5. Click **Create Data Source**
6. Choose **public** schema
7. Choose **venue** table

The following visualization shows the number of seats available per venue place: