Big Data Course 2019

Lab Session 5: Cloud SQL Shootout

Groups 1 - 34: on Monday we will handle in class new credentials for the lab session of this week. All the other groups (35 - 70) will continue using the old credentials to log into AWS.

Assignment (3 pts)

1. Compare Athena and RedShift for the TPC-H scale factor (SF) 1, 10 and 100. Load the datasets in the CSV format. In the RedShift, use a single node instance for SF1 and SF10 and mini cluster of 4 nodes (1 master and 3 workers) for SF100[1]. Execute the two given queries Q1 and Q5 at least three times. Discuss the following points (**1 pt**):
    a. Are the results between Athena and Redshift the same?
    b. Are the execution times similar between the two systems with respect to the given scale factors?
    c. Are the execution times per scale factor comparable between each execution?

    If any of the above points does not hold, discuss why that is not the case.
2. Measure the execution time with Spark and Athena with the Parquet data format (**0.5 pts**):
    a. Load the two datasets SF 1 and SF 10 in CSV format in Spark[2], execute the given queries and measure their execution time. Discuss whether it is comparable to what obtained in the first question by Athena and RedShift.
    b. Repeat the experiments this time with the data sets stored using the Parquet data format. We are providing the dataset in Parquet for SF1, but you have to perform the conversion on your own for SF10. Include the source code in the report. Discuss whether and why the execution time or the results are different with Parquet.
    c. Repeat the queries on Athena loading the data from Parquet SF1/SF10/SF100. Again, discuss whether the results or the execution times are different.
3. Explore the issue of partitioning your dataset for both Athena and RedShift. Can you identify a scheme capable of improving the performance of either Q1 or Q5? If so, what was your performance gain for SF10? Does your partitioning improve both queries or only one of them? Otherwise, why there are no advantageous partitioning schemes for the two queries? Include the source code to partition the data and load it in your final report. For the measurements with Athena, only use the Parquet data format. **(0.5 pts)**

---

[1] Loading the data set may take up about 20 - 30 minutes.
[2] SF100 is a bit too much to process for our Spark mini-clusters with 2 worker nodes.

4. Compare Athena, Redshift, Redshift Spectrum and Spark SQL in general terms. Discuss in which scenarios each system can be more beneficial than the others. Cover aspects such as query workload, data characteristics, ETL, overall performance, usability and cost. (**0.5 pt**)
5. Consider the use cases below. For each of them, describe if you would choose any of the systems considered, Athena, Redshift, Redshift Spectrum and Spark, possibly in combination, to tackle that particular scenario. Motivate your choice, describe how you envision it would be deployed, and estimate the cost per month your solution requires. **(0.5pt)**
    a. Data exploring. You have the availability of 16 TB of data in S3, in a variety of formats (csv, json, text files, sql dumps). You want to occasionally perform several ad-hoc queries per day. Multiple queries are often, but not always, executed in narrow sessions. Each session typically refers to only about 1% of the overall dataset, but different queries in the same session may touch part of the same data multiple times.
    b. Business reports. Your data set is organised in a star schema. It has one fact table, which grows of about 2GB of new homogenous data every day. More dimension tables are also present, their size is much smaller than the fact table and their content seldom changes. You want to perform several queries per hour. Each query may touch several gigabytes. About 90% of the overall data scanned in the main `fact' table is from the last month.
    c. Organisation. About 100 users in an organisation share the same data set of about 32 TB. The users run, on average, 8 analytical ad-hoc queries per day, scanning roughly 40GB of compressed data per query. Typically they cumulatively refer to only 4 TB of this dataset in a month, and even inside this subset, the data referred is highly skewed. Currently the whole data set is hosted in an in-house solution. Their system suffers of temporary slow-downs when multiple users operate concurrently, degrading the whole experience. The organisation is wondering whether it is suitable for them to migrate their in-house solution to the cloud, provisioning a (tentative) budget cap of $3800 per month.

Introduction

Today's business analyst demands SQL-like access to Big Data™. Your task today is to design a SQL-in-the-cloud data warehouse system. You will compare SparkSQL, Athena and Redshift, which is a hosted version of the parallel database system Actian Matrix (probably better known under its previous name ParAccel). As a benchmark, we are going to use TPC-H, an industry standard benchmark for analytical SQL systems.

Three data sets with "scale factors" (SF) 1, 10 and 100 have already been created for you and uploaded to S3.
In CSV format, they are stored at:

- SF1: s3://bigdatacourse2019/tpch_csv/SF1/
- SF10: s3://bigdatacourse2019/tpch_csv/SF10/
- SF100: s3://bigdatacourse2019/tpch_csv/SF100/

In Parquet format, they are stored at:

- SF1: s3://bigdatacourse2019/tpch_parquet/SF1/
- SF10: missing (!)
- SF100: s3://bigdatacourse2019/tpch_parquet/SF100/

RedShift

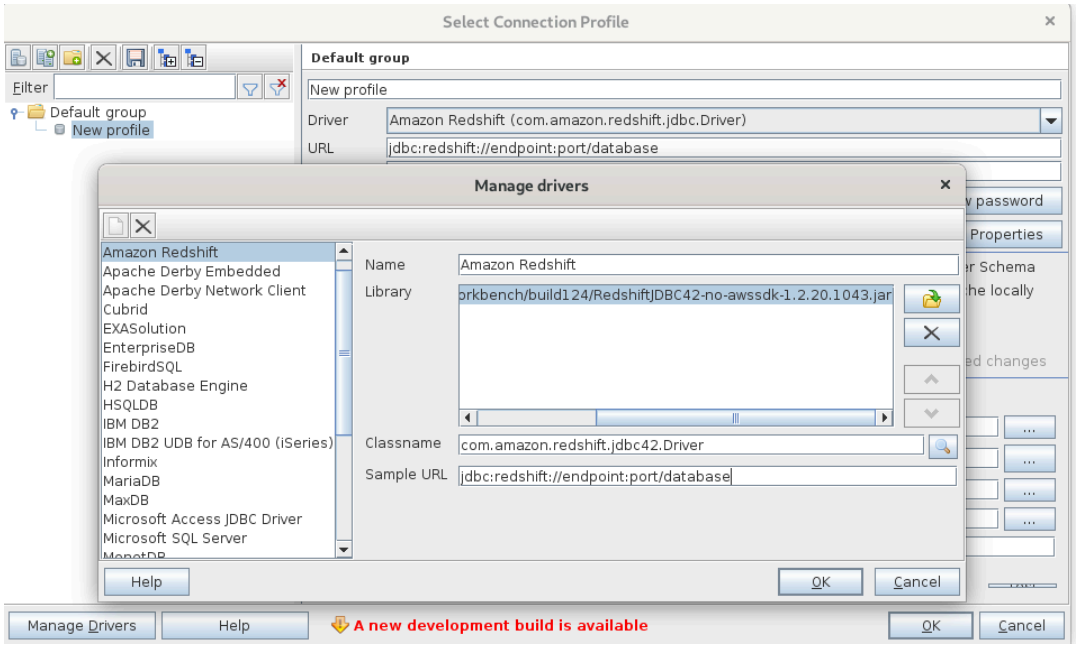
In this part we are going to set up Amazon RedShift, load the data from TPC-H Scale Factor 1 and run our two target TPC-H queries.

Configure SQL Workbench

We will interact with RedShift by an external tool, SQL Workbench. The tool needs to be downloaded and configured in your computer/laptop. Note, you may also have to install ~~evil~~ Oracle Java to run those.

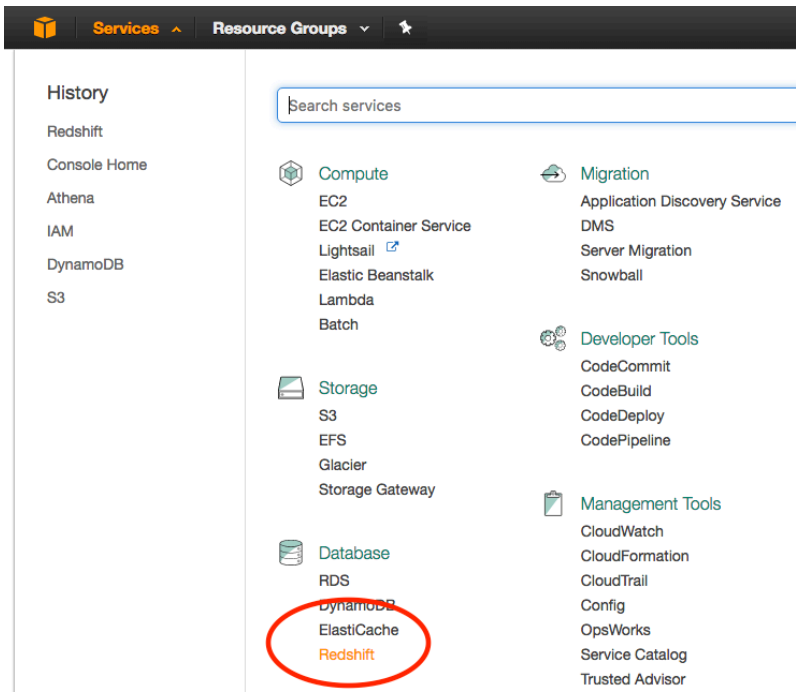
Download and install SQL Workbench from: http://www.sql-workbench.net/ . To connect to RedShift, you need to have its driver with the SQL Workbench. This can be downloaded from AWS: https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/1.2.20.1043/RedshiftJDBC42-no-awssdk-1.2.20.1043.jar

Once opened, register the driver from the button “Manage Drivers” on the bottom left, select Amazon Redshift, replace the library with the path of the driver just downloaded and set the class name to `com.amazon.redshift.jdbc42.Driver` :

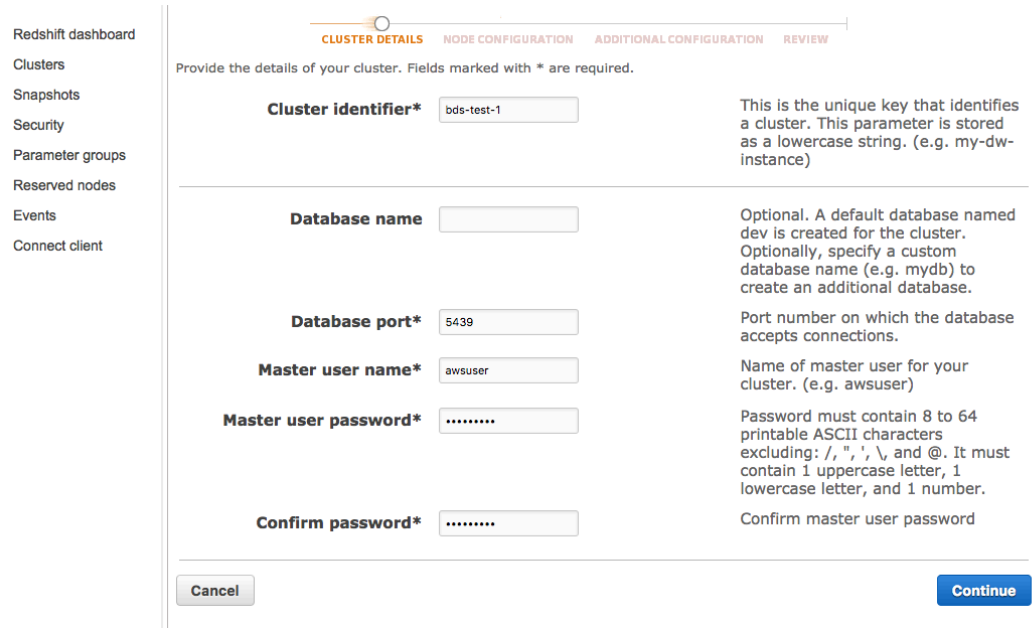


Start a Redshift Cluster

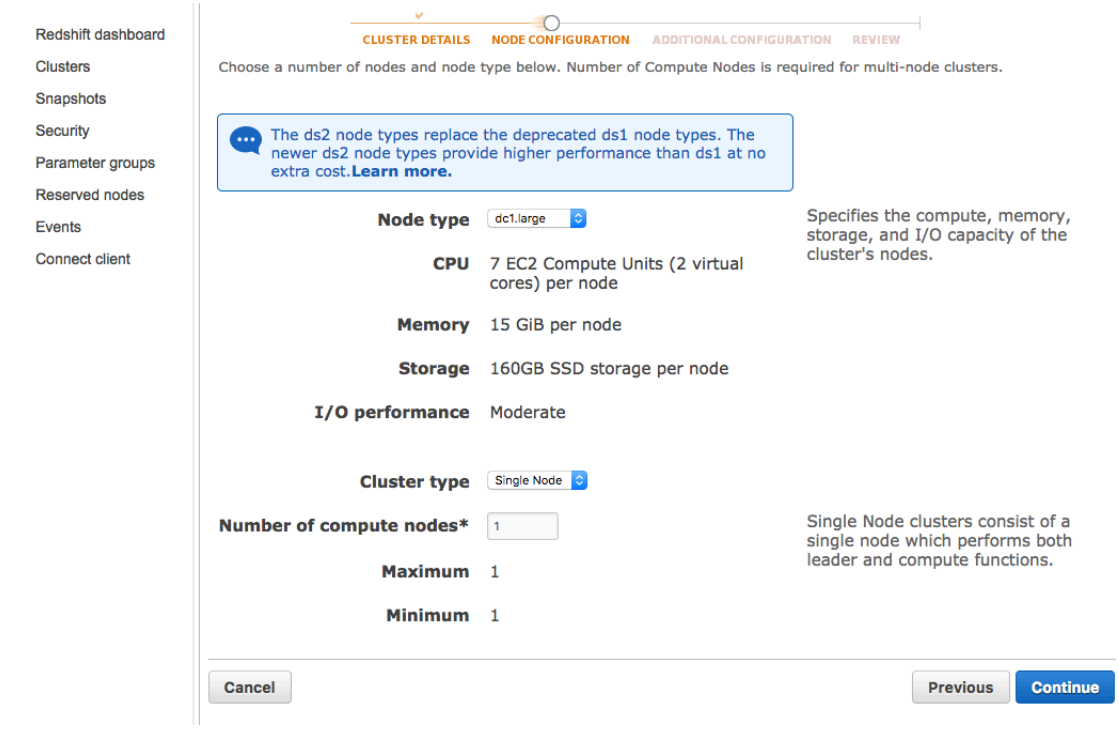
We strongly advise to read the rest of this tutorial before starting the cluster. These clusters are quite costly, please keep their runtime minimal.



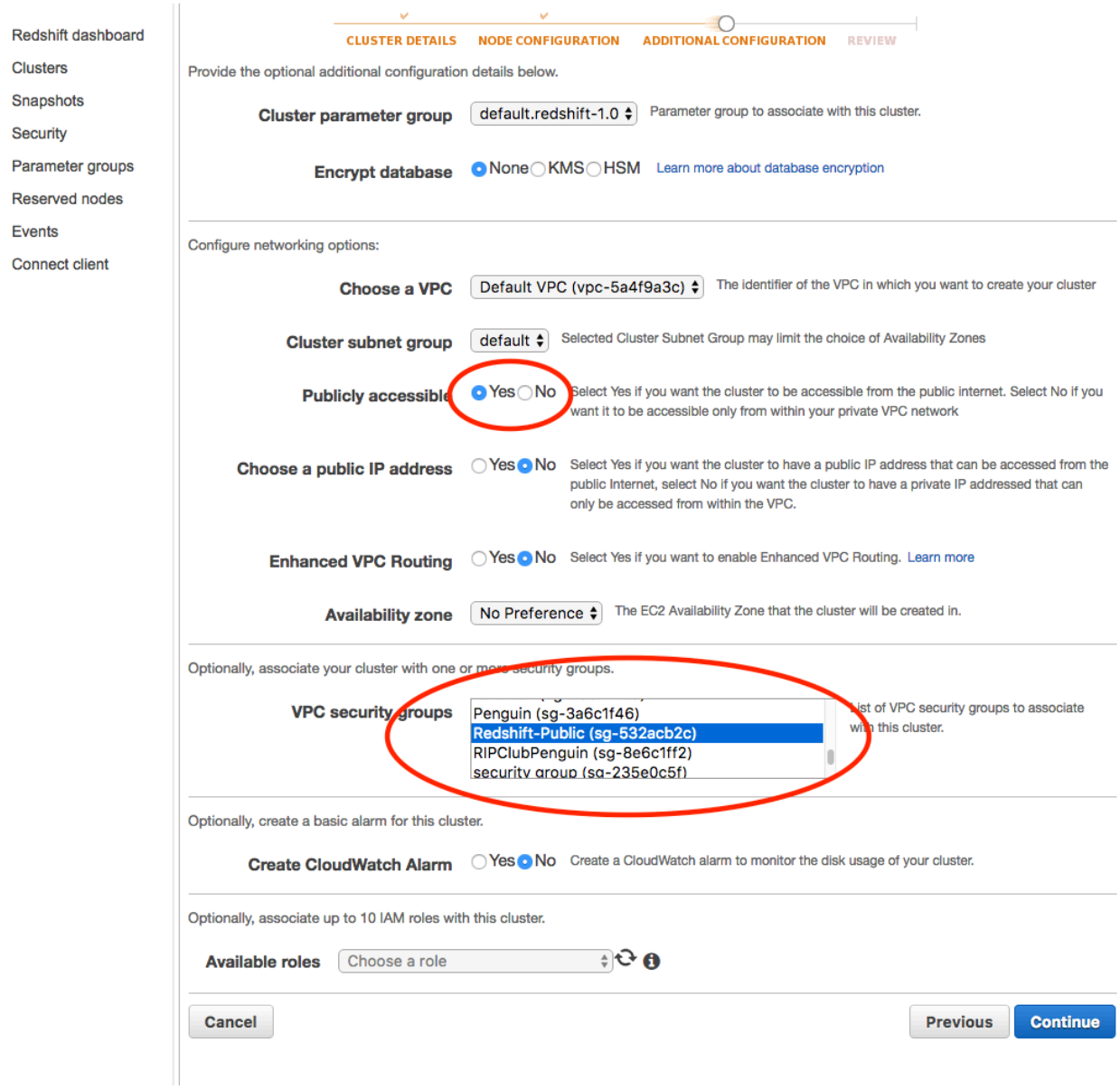
Create a new cluster. Use your user name as cluster name and choose an arbitrary username/password.



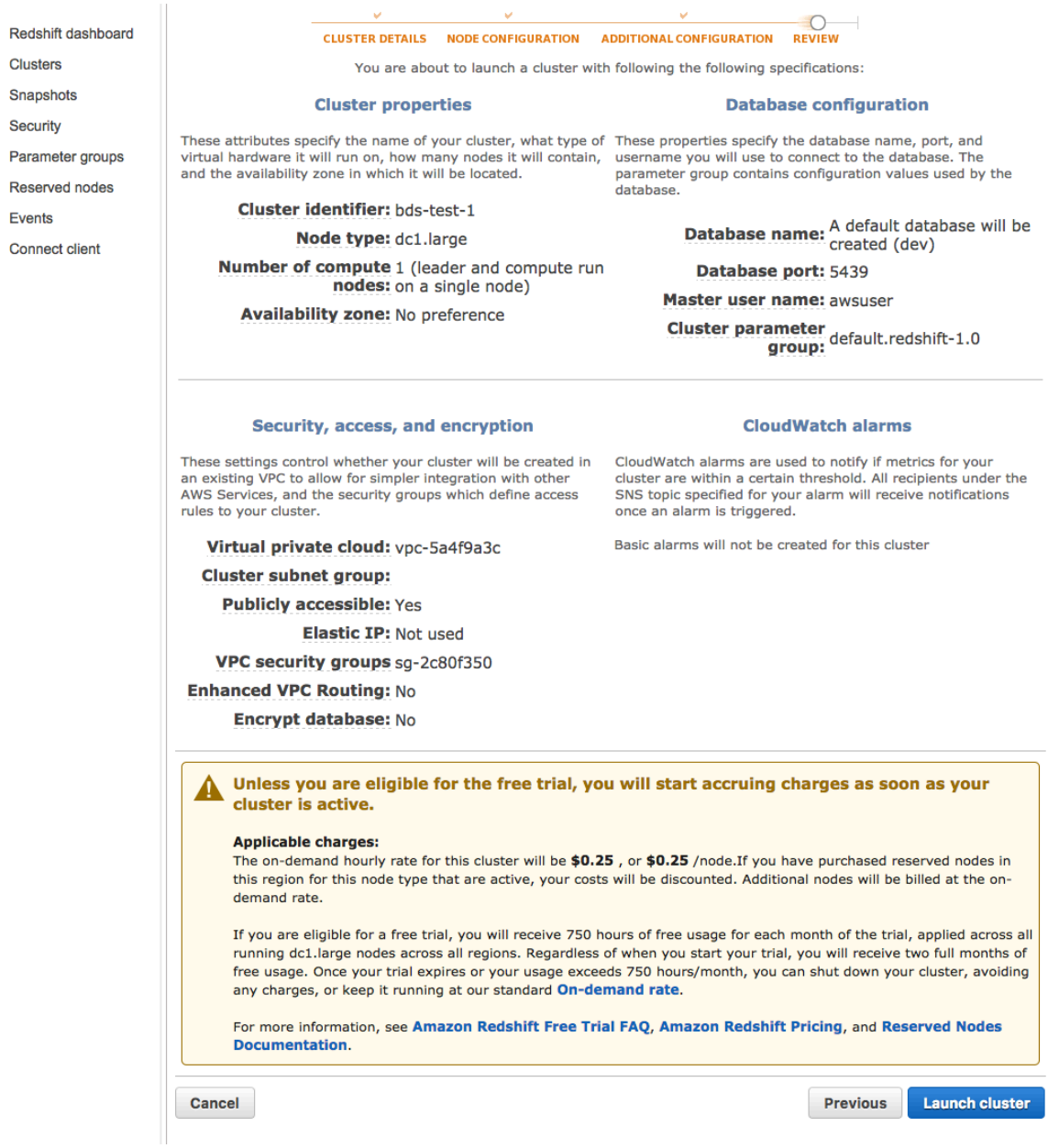
We will use a single-node cluster, either dc1.large or dc2.large:



Make sure your cluster is configured to be publicly accessible and use the "Redshift-Public" Security group

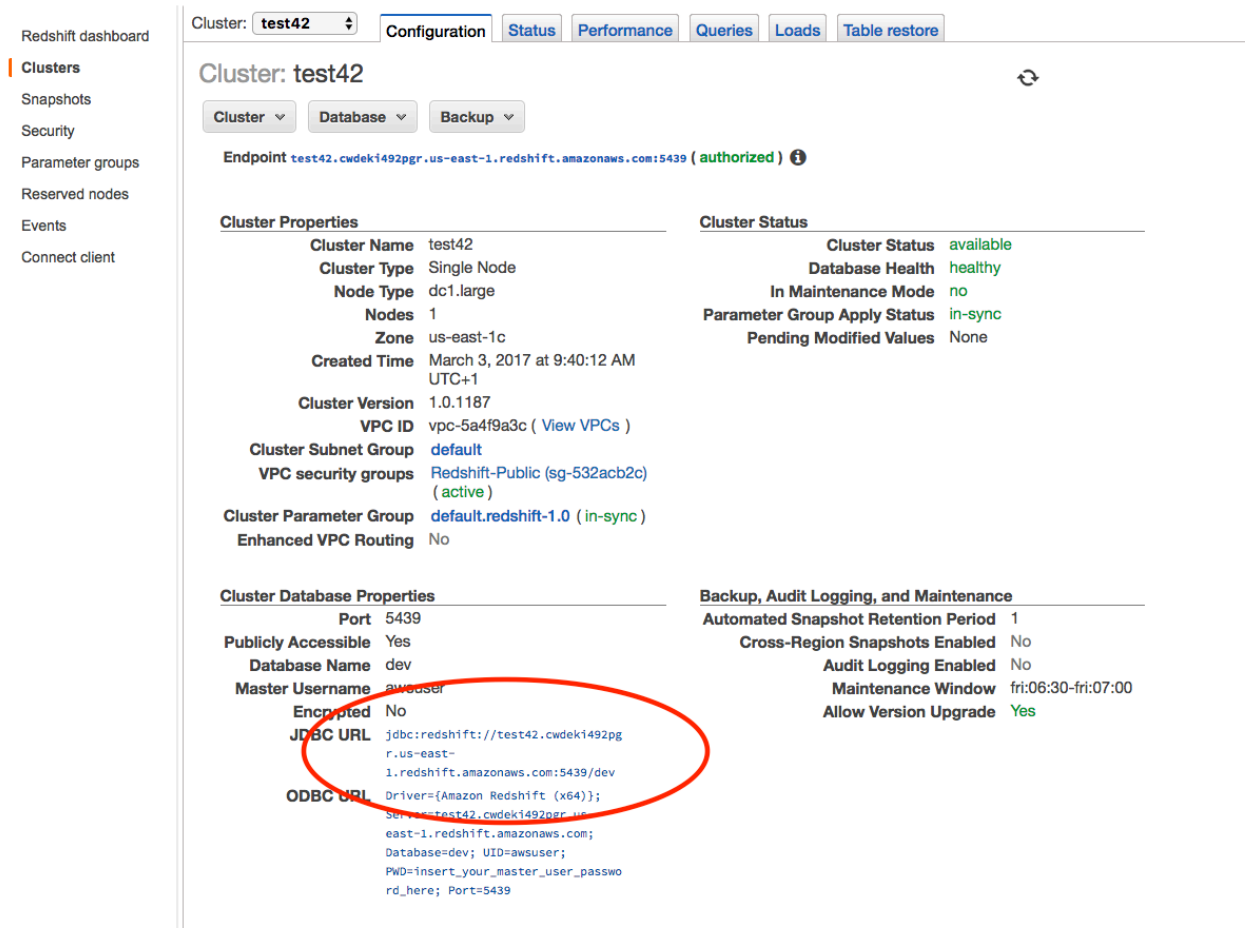

Redshift dashboard
Clusters
Snapshots
Security
Parameter groups
Reserved nodes
Events
Connect client
CLUSTER DETAILS
NODE CONFIGURATION
ADDITIONAL CONFIGURATION
REVIEW
Provide the optional additional configuration details below.
Cluster parameter group
default.redshift-1.0
Parameter group to associate with this cluster.
Encrypt database
None KMS HSM
Learn more about database encryption
Configure networking options:
Choose a VPC
Default VPC (vpc-5a4f9a3c)
The identifier of the VPC in which you want to create your cluster
Cluster subnet group
default
Selected Cluster Subnet Group may limit the choice of Availability Zones
Publicly accessible
Yes No
Select Yes if you want the cluster to be accessible from the public internet. Select No if you want it to be accessible only from within your private VPC network
Choose a public IP address
Yes No
Select Yes if you want the cluster to have a public IP address that can be accessed from the public Internet, select No if you want the cluster to have a private IP addressed that can only be accessed from within the VPC.
Enhanced VPC Routing
Yes No
Select Yes if you want to enable Enhanced VPC Routing. Learn more
Availability zone
No Preference
The EC2 Availability Zone that the cluster will be created in.
Optionally, associate your cluster with one or more security groups.
VPC security groups
Penguin (sg-3a6c1f46)
Redshift-Public (sg-532acb2c)
RIPClubPenguin (sg-8e6c1ff2)
security group (sg-235e0c5f)
List of VPC security groups to associate with this cluster.
Optionally, create a basic alarm for this cluster.
Create CloudWatch Alarm
Yes No
Create a CloudWatch alarm to monitor the disk usage of your cluster.
Optionally, associate up to 10 IAM roles with this cluster.
Available roles
Choose a role
Cancel
Previous
Continue

Launch!

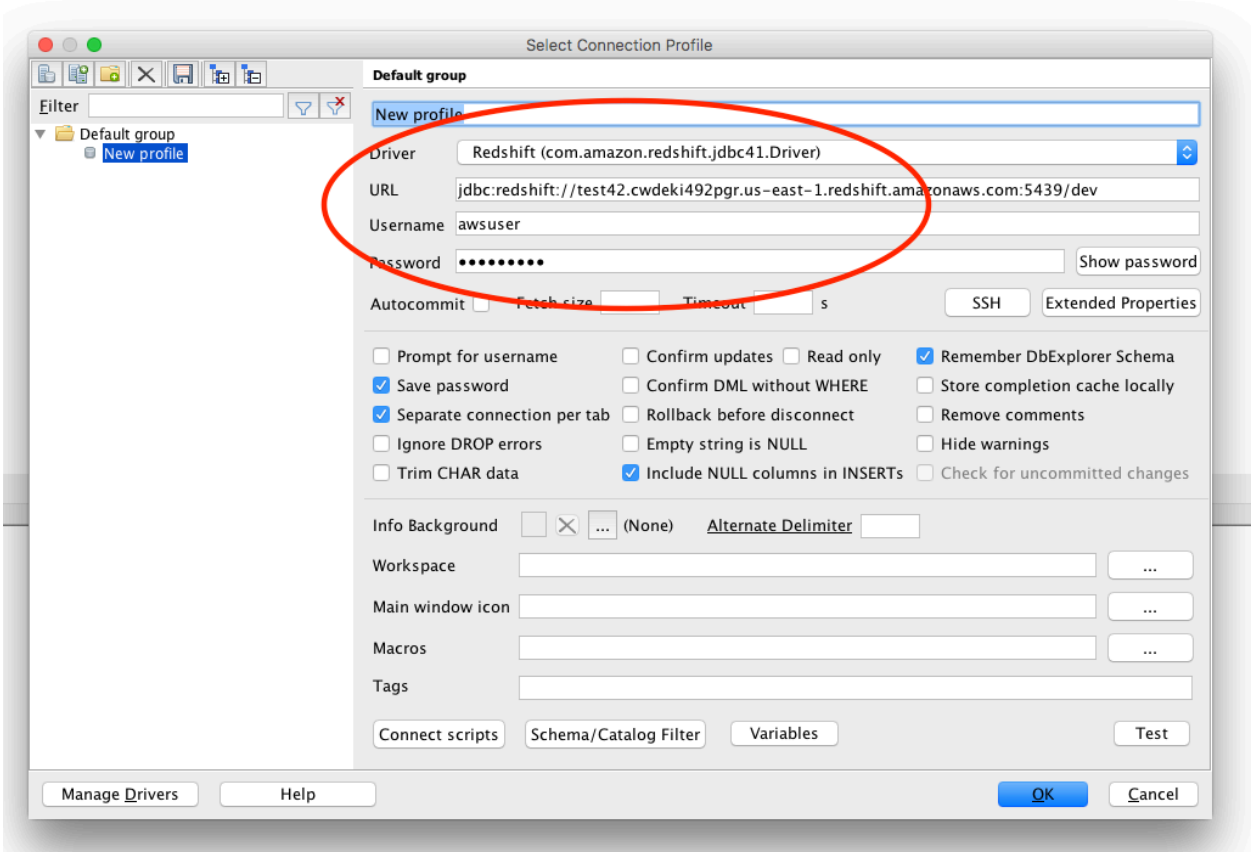


Connect to the cluster

We will use SQL Workbench to connect to the RedShift cluster. But first we need to find the JDBC endpoint url. You can retrieve it from the RedShift web console. Once the cluster becomes ready, you can see and copy its JDBC url:

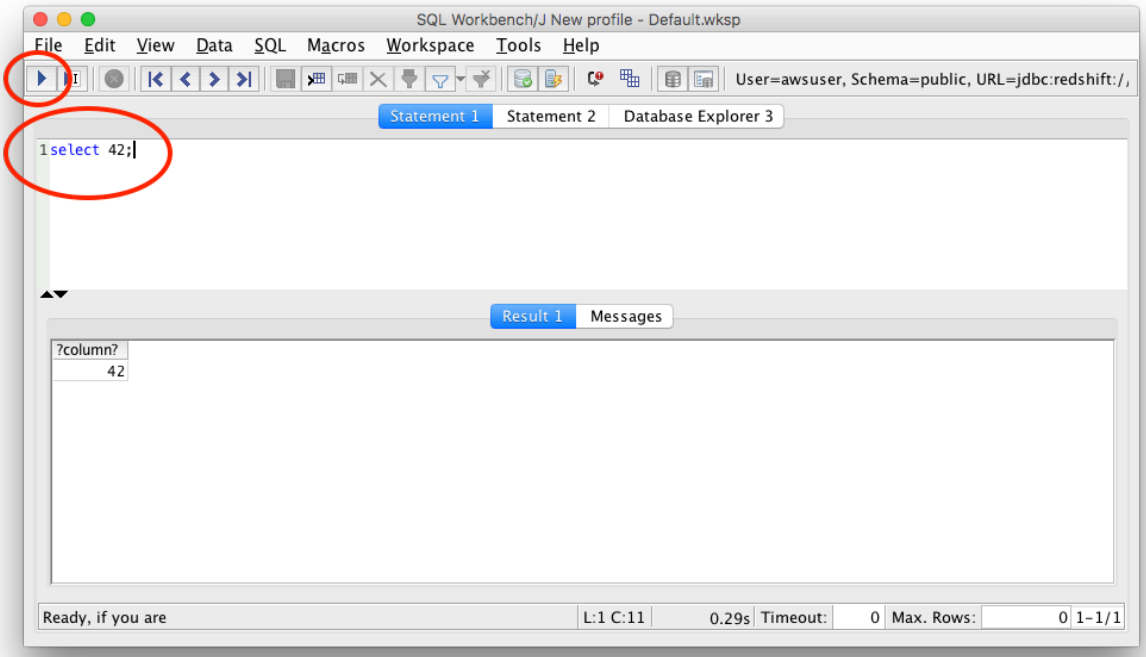


Open the SQL Workbench, and create a new connection (or profile, same thing). Set the driver to Amazon Redshift, the JDBC url to the string copied above, the username and password to the credentials set when the cluster was created:



You will be greeted by the Query/Query result screen. Run a simple query to check everything works.

```
SELECT 42;
```



Create the tables

Before importing the data into the RedShift database, we need to create a schema, that is the tables where the data will be stored.

The following SQL snippet should be valid for all datasets, SF1, SF10 and SF100:

```
CREATE TABLE region (r_regionkey INT NOT NULL, r_name VARCHAR(25) NOT NULL, r_comment
VARCHAR(152) NOT NULL, PRIMARY KEY (r_regionkey));

CREATE TABLE nation (n_nationkey INT NOT NULL, n_name VARCHAR(25) NOT NULL,
n_regionkey INT NOT NULL, n_comment VARCHAR(152) NOT NULL, PRIMARY KEY (n_nationkey));

CREATE TABLE supplier (s_suppkey INT NOT NULL, s_name VARCHAR(25) NOT NULL, s_address
VARCHAR(40) NOT NULL, s_nationkey INT NOT NULL, s_phone VARCHAR(15) NOT NULL,
s_acctbal DECIMAL(15,2) NOT NULL, s_comment VARCHAR(101) NOT NULL, PRIMARY KEY
(s_suppkey));

CREATE TABLE customer (c_custkey INT NOT NULL, c_name VARCHAR(25) NOT NULL, c_address
VARCHAR(40) NOT NULL, c_nationkey INT NOT NULL, c_phone VARCHAR(15) NOT NULL,
c_acctbal DECIMAL(15,2) NOT NULL, c_mktsegment VARCHAR(10) NOT NULL, c_comment
VARCHAR(117) NOT NULL, PRIMARY KEY (c_custkey));

CREATE TABLE part (p_partkey INT NOT NULL, p_name VARCHAR(55) NOT NULL, p_mfgr
VARCHAR(25) NOT NULL, p_brand VARCHAR(10) NOT NULL, p_type VARCHAR(25) NOT NULL,
p_size INT NOT NULL, p_container VARCHAR(10) NOT NULL, p_retailprice DECIMAL(15,2) NOT
NULL, p_comment VARCHAR(23) NOT NULL, PRIMARY KEY (p_partkey));
```

```
CREATE TABLE partsupp (ps_partkey INT NOT NULL, ps_suppkey INT NOT NULL, ps_availqty
INT NOT NULL, ps_supplycost DECIMAL(15,2) NOT NULL, ps_comment VARCHAR(199) NOT NULL,
PRIMARY KEY (ps_partkey, ps_suppkey), FOREIGN KEY (ps_partkey) REFERENCES part
(p_partkey), FOREIGN KEY (ps_suppkey) REFERENCES supplier (s_suppkey)) ;

CREATE TABLE orders (o_orderkey INT NOT NULL, o_custkey INT NOT NULL, o_orderstatus
VARCHAR(1) NOT NULL, o_totalprice DECIMAL(15,2) NOT NULL, o_orderdate DATE NOT NULL,
o_orderpriority VARCHAR(15) NOT NULL, o_clerk VARCHAR(15) NOT NULL, o_shippriority INT
NOT NULL, o_comment VARCHAR(79) NOT NULL, PRIMARY KEY (o_orderkey));

CREATE TABLE lineitem (l_orderkey INT NOT NULL, l_partkey INT NOT NULL, l_suppkey INT
NOT NULL, l_linenumber INT NOT NULL, l_quantity INTEGER NOT NULL, l_extendedprice
DECIMAL(15,2) NOT NULL, l_discount DECIMAL(15,2) NOT NULL, l_tax DECIMAL(15,2) NOT
NULL, l_returnflag VARCHAR(1) NOT NULL, l_linestatus VARCHAR(1) NOT NULL, l_shipdate
DATE NOT NULL, l_commitdate DATE NOT NULL, l_receiptdate DATE NOT NULL, l_shipinstruct
VARCHAR(25) NOT NULL, l_shipmode VARCHAR(10) NOT NULL, l_comment VARCHAR(44) NOT NULL,
PRIMARY KEY (l_orderkey,l_linenumber));

COMMIT;
```

Data loading (SF1)

The following statements load the data for Scale Factor 1 (SF1). For the other datasets, replace the value SF1 in the loading url with SF10 and SF100.
As we split the groups in two AWS accounts, replace the XXXXXXX according to your group number:

- Account 450296069091, groups 1 - 34:
  `aws_access_key_id=AKIAIFUHSX6B2GJGRSWQ;aws_secret_access_key=OGy6hKDeW+s25Ephxaxz6XOBSIhCsJbkp/geBPr5`
- Account 482531159440, groups 35 - 70, regenerated 10/Mar/2019 18:00:
  `aws_access_key_id=AKIAJLBMYOAAG373OOJA;aws_secret_access_key=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`

```
copy region from 's3://bigdatacourse2019/tpch_csv/SF1/region/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 100;

copy nation from 's3://bigdatacourse2019/tpch_csv/SF1/nation/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 100;

copy customer from 's3://bigdatacourse2019/tpch_csv/SF1/customer/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 100;

copy orders from 's3://bigdatacourse2019/tpch_csv/SF1/orders/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 100;

copy lineitem from 's3://bigdatacourse2019/tpch_csv/SF1/lineitem/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 10000;
```

```
copy part from 's3://bigdatacourse2019/tpch_csv/SF1/part/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 100;

copy partsupp from 's3://bigdatacourse2019/tpch_csv/SF1/partsupp/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 100;

copy supplier from 's3://bigdatacourse2019/tpch_csv/SF1/supplier/' delimiter '|' gzip
credentials 'XXXXXXX' MAXERROR 100;

COMMIT;
```

At the end check whether the data has been actually loaded, by running a `SELECT COUNT(*) FROM <table>` for all the loaded tables (replace <table> with an actual table name) after the COMMIT. The load may take a while. Also, if you get errors, it's always a good idea to run the command ROLLBACK and then try again.

Queries

Execute the following two queries through the SQL workbench. Note the execution time.

TPC-H Query 1:

```
select
	l_returnflag,
	l_linestatus,
	sum(l_quantity) as sum_qty,
	sum(l_extendedprice) as sum_base_price,
	sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
	sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
	avg(l_quantity) as avg_qty,
	avg(l_extendedprice) as avg_price,
	avg(l_discount) as avg_disc,
	count(*) as count_order
from
	lineitem
where
	l_shipdate <= date '1998-12-01' - interval '108' day
group by
	l_returnflag,
	l_linestatus
order by
	l_returnflag,
	l_linestatus;
```

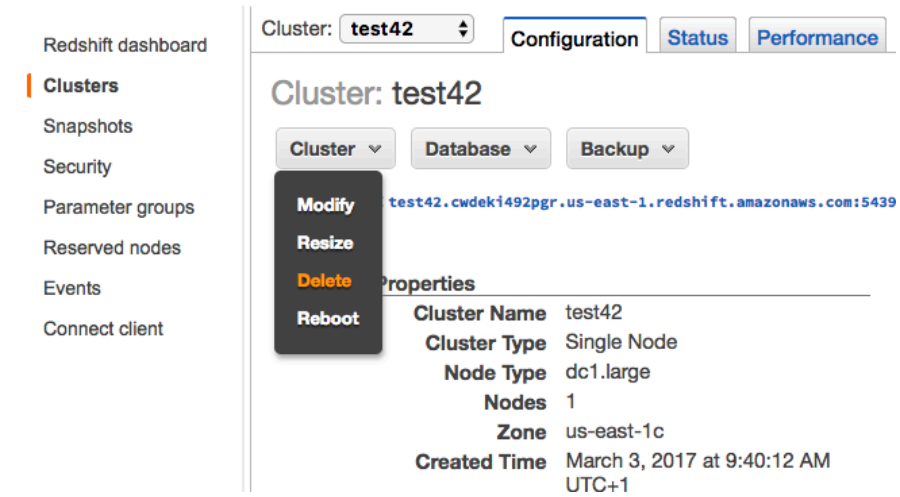
TPC-H Query 5

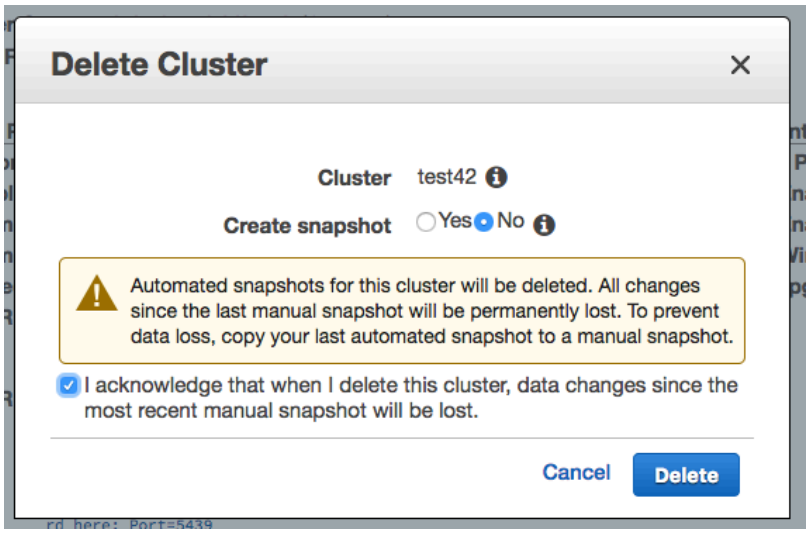
```
select
```

```
	n_name,
	sum(l_extendedprice * (1 - l_discount)) as revenue
from
	customer,
	orders,
	lineitem,
	supplier,
	nation,
	region
where
	c_custkey = o_custkey
	and l_orderkey = o_orderkey
	and l_suppkey = s_suppkey
	and c_nationkey = s_nationkey
	and s_nationkey = n_nationkey
	and n_regionkey = r_regionkey
	and r_name = 'MIDDLE EAST'
group by
	n_name
order by
	revenue desc;
```

Shutdown

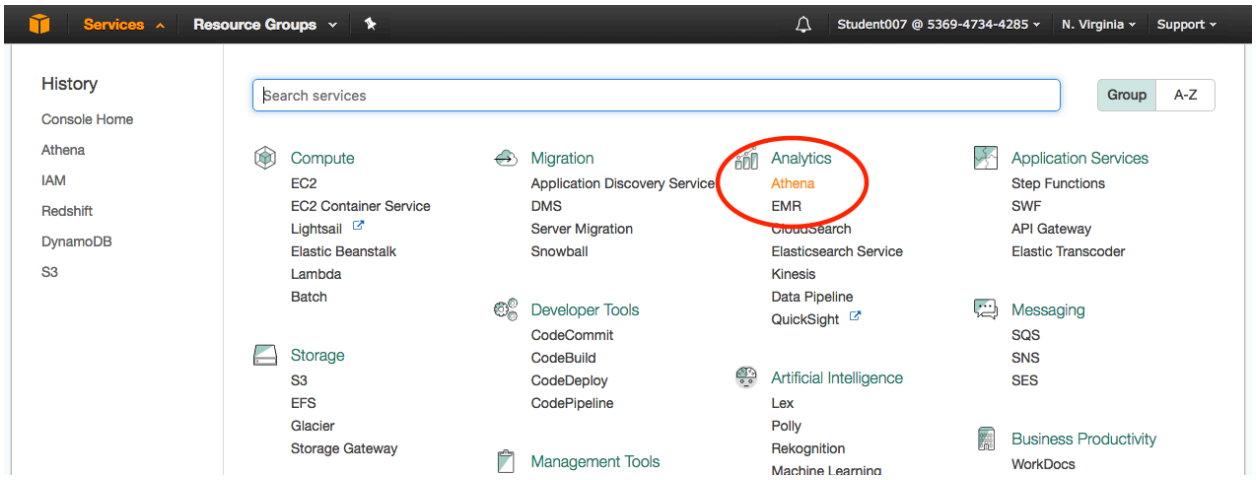
Those clusters tend to be rather expensive. It’s important that once you’ve finished your work, to shutdown your cluster:


Redshift dashboard
Clusters
Snapshots
Security
Parameter groups
Reserved nodes
Events
Connect client
Cluster: test42
Configuration
Status
Performance
Cluster: test42
Cluster
Database
Backup
Modify
Resize
Delete
Reboot
test42.cwdeki492pgr.us-east-1.redshift.amazonaws.com:5439
Properties
Cluster Name test42
Cluster Type Single Node
Node Type dc1.large
Nodes 1
Zone us-east-1c
Created Time March 3, 2017 at 9:40:12 AM UTC+1


Delete Cluster
Cluster test42
Create snapshot Yes No
Automated snapshots for this cluster will be deleted. All changes since the last manual snapshot will be permanently lost. To prevent data loss, copy your last automated snapshot to a manual snapshot.
I acknowledge that when I delete this cluster, data changes since the most recent manual snapshot will be lost.
Cancel
Delete

Athena

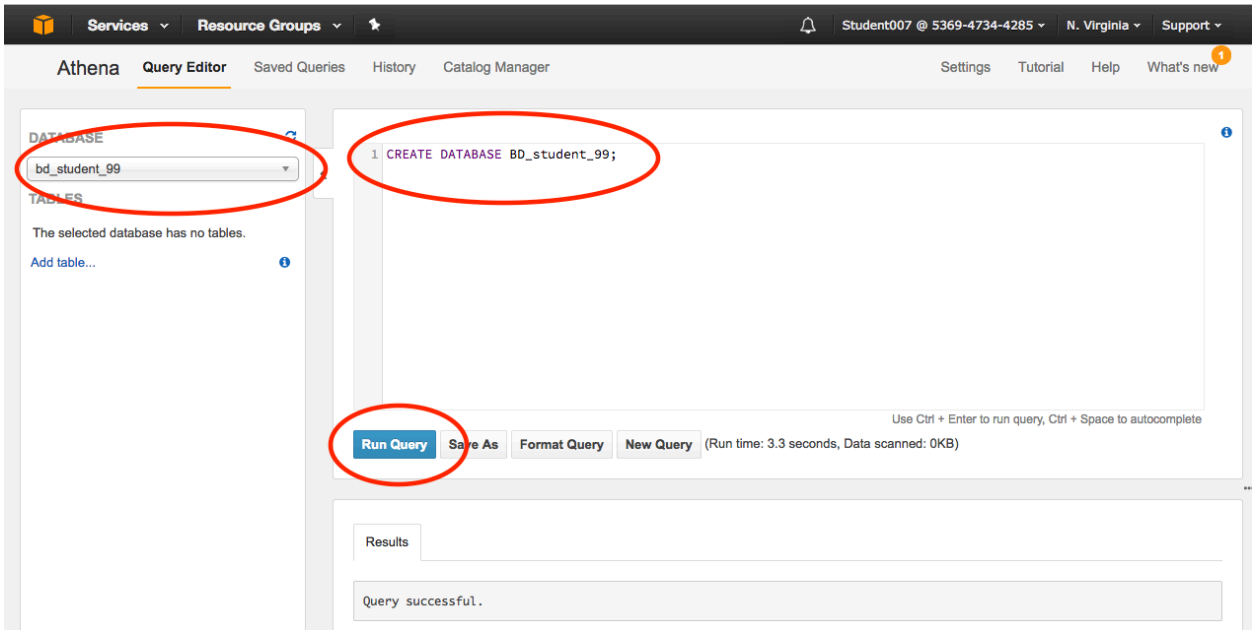
Athena is an AWS service to execute interactive SQL queries over S3. In this part, we are going to create a new database and perform the two TPC-H queries with the data sets already stored in S3. The following instructions apply to Scale Factor 1 (SF1). But first of all, reach the Athena console from the AWS page, click on Services, then look for Athena:



Creating the tables

We first need to create a database and the tables to operate within SQL.

Create a database for you to use, by running the following query:
`CREATE DATABASE groupXX;`
(replace XX with your group number), click "Run Query", finally select from the bar on the left side. In the screenshot below the database name is BD_student_99:



To create the tables, paste the following SQL statements in the query window and execute them. The SQL statements should be executed one by one (ouch):

```
CREATE EXTERNAL TABLE customer(
C_CustKey int ,
C_Name varchar(64) ,
C_Address varchar(64) ,
C_NationKey int ,
C_Phone varchar(64) ,
C_AcctBal decimal(13, 2) ,
C_MktSegment varchar(64) ,
C_Comment varchar(120) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/customer/';


CREATE EXTERNAL TABLE lineitem(
L_OrderKey int ,
L_PartKey int ,
L_SuppKey int ,
L_LineNumber int ,
L_Quantity int ,
L_ExtendedPrice decimal(13, 2) ,
L_Discount decimal(13, 2) ,
L_Tax decimal(13, 2) ,
L_ReturnFlag varchar(64) ,
L_LineStatus varchar(64) ,
L_ShipDate date ,
L_CommitDate date ,
```

```
L_ReceiptDate date ,
L_ShipInstruct varchar(64) ,
L_ShipMode varchar(64) ,
L_Comment varchar(64) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/lineitem/';

CREATE EXTERNAL TABLE nation(
N_NationKey int ,
N_Name varchar(64) ,
N_RegionKey int ,
N_Comment varchar(160) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/nation/';

CREATE EXTERNAL TABLE orders(
O_OrderKey int ,
O_CustKey int ,
O_OrderStatus varchar(64) ,
O_TotalPrice decimal(13, 2) ,
O_OrderDate date ,
O_OrderPriority varchar(15) ,
O_Clerk varchar(64) ,
O_ShipPriority int ,
O_Comment varchar(80) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/orders/';

CREATE EXTERNAL TABLE part(
P_PartKey int ,
P_Name varchar(64) ,
P_Mfgr varchar(64) ,
P_Brand varchar(64) ,
P_Type varchar(64) ,
P_Size int ,
P_Container varchar(64) ,
P_RetailPrice decimal(13, 2) ,
P_Comment varchar(64) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/part/';

CREATE EXTERNAL TABLE partsupp(
PS_PartKey int ,
PS_SuppKey int ,
PS_AvailQty int ,
PS_SupplyCost decimal(13, 2) ,
```

```
PS_Comment varchar(200) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/partsupp/';

CREATE EXTERNAL TABLE region(
R_RegionKey int ,
R_Name varchar(64) ,
R_Comment varchar(160) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/region/';

CREATE EXTERNAL TABLE supplier(
S_SuppKey int ,
S_Name varchar(64) ,
S_Address varchar(64) ,
S_NationKey int ,
S_Phone varchar(18) ,
S_AcctBal decimal(13, 2) ,
S_Comment varchar(105) ,
skip varchar(64)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
's3://bigdatacourse2019/tpch_csv/SF1/supplier/';
```

Make sure the tables have been created, e.g. they are in the tables list and all contain data. For instance, the record count for the table lineitem should be 6001215:

```
SELECT COUNT(*) FROM lineitem -- Expected 6001215
```

Queries

Execute the following two queries on the Athena console:

TPC-H Query 1:

```
SELECT
  L_RETURNFLAG, L_LINESTATUS, SUM(L_QUANTITY), SUM(L_EXTENDEDPRICE),
SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)), SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1+L_TAX)),
AVG(L_QUANTITY), AVG(L_EXTENDEDPRICE), AVG(L_DISCOUNT), COUNT(1)
FROM
  lineitem
WHERE
  L_SHIPDATE<= CAST ('1998-09-02' AS DATE)
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS;
```

TPC-H Query 5

```
select
  n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
from
  customer c join
    ( select n_name, l_extendedprice, l_discount, s_nationkey, o_custkey from orders o
join
      ( select n_name, l_extendedprice, l_discount, l_orderkey, s_nationkey from
lineitem l join
        ( select n_name, s_suppkey, s_nationkey from supplier s join
          ( select n_name, n_nationkey
            from nation n join region r
            on n.n_regionkey = r.r_regionkey and r.r_name = 'MIDDLE EAST'
          ) n1 on s.s_nationkey = n1.n_nationkey
        ) s1 on l.l_suppkey = s1.s_suppkey
      ) l1 on l1.l_orderkey = o.o_orderkey
) o1
on c.c_nationkey = o1.s_nationkey and c.c_custkey = o1.o_custkey
group by n_name
order by revenue desc;
```

Observe the **query runtime** and **amount of data read**.

Troubleshooting

Redshift

- `Invalid operation: current transaction is aborted, commands ignored until end of transaction block;` Run `rollback`


BIG
DATA