**AWS Loft – Database Week**

Comparing Redis with Relational Hands-on Lab

In this lab you will learn the following:

* How to create an EC2 instance for use as a SQL and Redis client
* How to create an RDS instance using MySQL
* How to connect your SQL client instance to your RDS instance
* How to create a database and load a public Landsat dataset into it
* How to query your database using SQL
* How to create an ElastiCache for Redis cluster
* How to connect your Redis client instance to your ElastiCache cluster
* How to query your ElastiCache cluster using Redis commands

1. Create an Amazon EC2 t2.micro instance running the most recent version of Amazon Linux 2.

You will use this EC2 instance as a SQL client to access your RDS/MySQL instance.

Follow the documentation at:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html>

1. Create and use an EC2 Security Group that allows inbound access as follows:
   * SSH using TCP on port 22 from Anywhere\* (0.0.0.0/0)
   * MYSQL/Aurora using TCP on port 3306 from MyIP
   * Custom protocol using TCP on port 6379 (for Redis) from MyIP (for Redis)

* + \*You might get an automated warning that your EC2 instance is “open to the world”, because we’re not limiting the source IP range for SSH. This is expected. In a production system, as a security best practice, you must only provide a limited IP range for allowed SSH access. For this lab however, you can disregard the warning.
  + We’ll be accessing both MySQL and Redis from this EC2 instance. **Once the instance is created, go back to the security group and update the “Source” for both services to use the IP address of your EC2 instance**.

Access the linux console. See documentation at

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html

Use ssh for Linux or Mac; use PuTTY for Windows  
Example:

ssh –i ~/Downloads/key.pem ec2-user@ec2-01-02-03-99.us-west-2.compute.amazonaws.com

[ec2-user@ip-192-168-0-1 ~]$

* + Install MySql and Redis prerequisites

$ sudo yum install mysql

$ sudo yum install mysql-devel

$ sudo yum install gcc

$ sudo yum install python-pip

$ sudo pip install MySQL-python

* + Install Redis command line interface and tools

$ sudo amazon-linux-extras install redis4.0

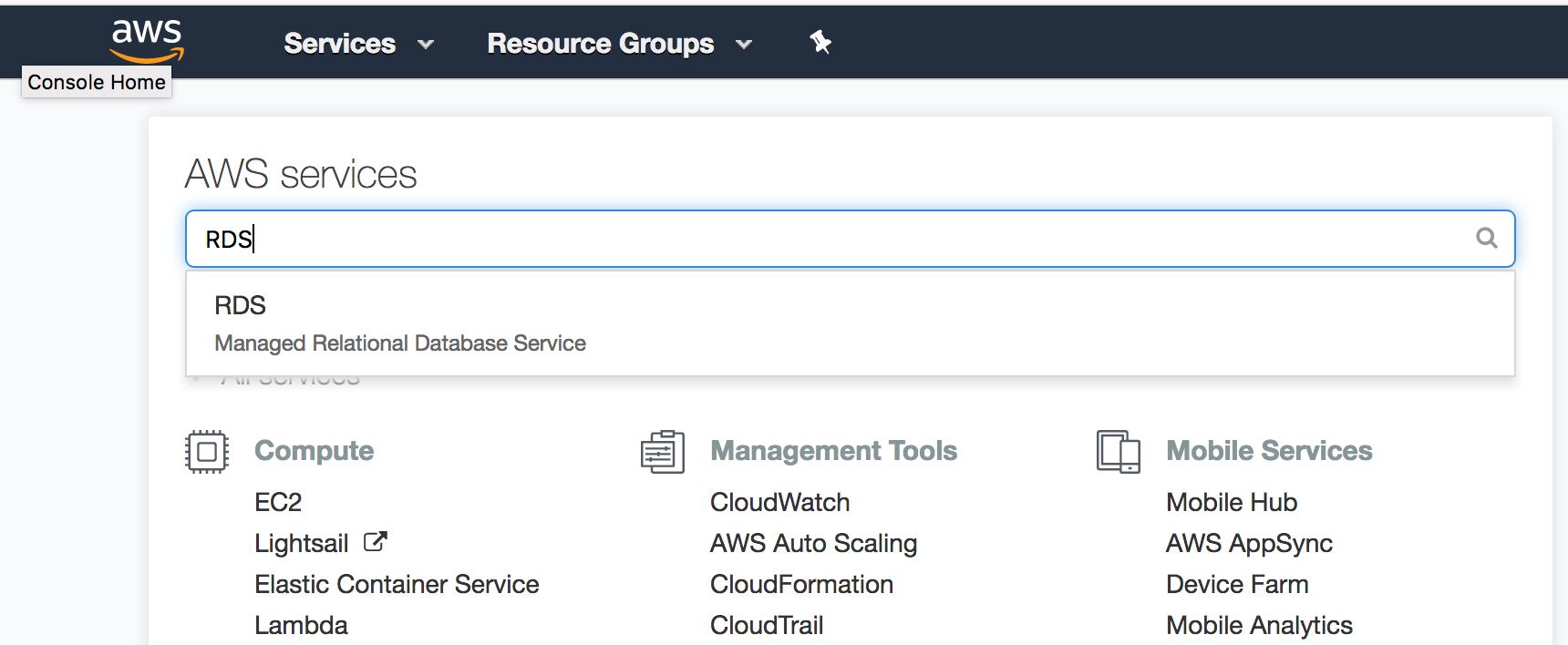
$ sudo pip install redis

1. Create an Amazon RDS MySQL db.t2.small instance

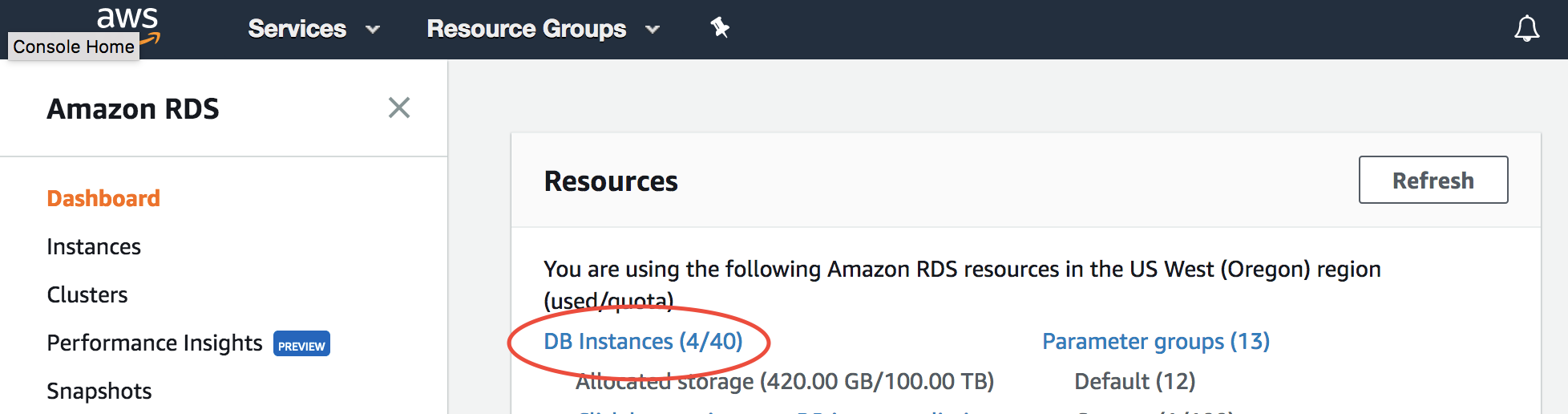
See documentation at:

<http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html>

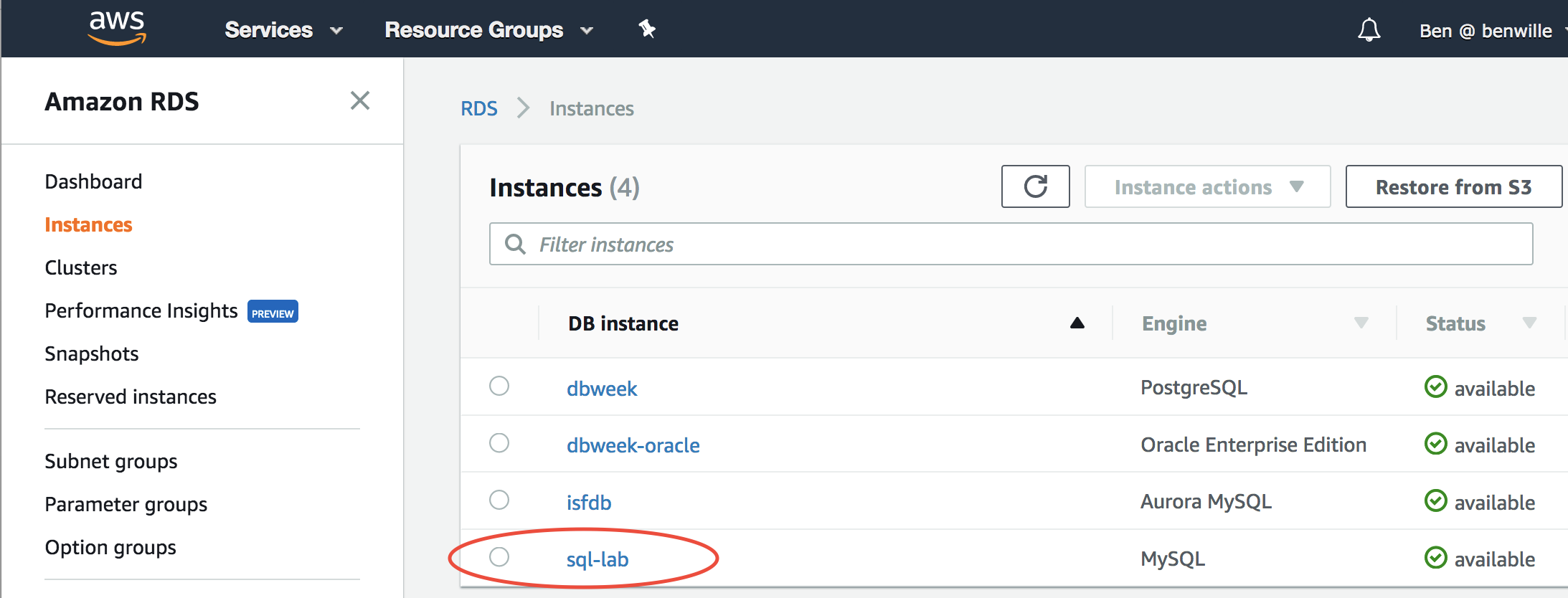
* + Use the MySQL engine and a Dev/Test use case
  + Use the general public license, the most current version of MySQL 5.7, and a db.t2.small instance. Choose No for Multi-AZ deployment and configure 20GB of General Purpose SSD storage
  + Name the instance **sql-lab**. Be sure to choose MySQL (not Aurora and not MariaDB) and the db.t2.micro instance size
  + Choose a username and a password (and don’t forget them!)
  + Do not create a database (we will do that later). Keep all other default settings, and Create database
  + Wait a few minutes until the RDS instance status is ‘Available’.
  + Once the RDS instance is available, find your instance’s DNS endpoint. We’ll use this endpoint to connect the MySQL client to your RDS instance.
    1. On the AWS Console, in the AWS Services box, type ‘RDS’ and press enter.



* + 1. On the RDS Dashboard, choose DB Instances



* + 1. Select your DB Instance



* + 1. Scroll down and find your endpoint name. You will need it later!

Note: When using the endpoint name, you usually should not use the

port extension (:3306), just the name.

* + 1. Modify your RDS Security Group (using the Modify button) to allow inbound access from your EC2 instance. Select the Security Group and update the source IP with the IP address of your EC2 instance (MyIP from step 2). Be sure to choose “Apply Immediately”  
         
       Also modify your EC2 Security Group to allow inbound access from your EC2 instance for both MySQL and Redis

1. Create an Amazon ElastiCache cluster with a single cache.t2.micro instance with Redis
   * See documentation at

<http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/GettingStarted.html>

* + Name the instance redis-lab and use the most currest version of Redis 4.x. Be sure to choose the cache.t2.micro instance size
  + Set the number of replicas to **none**
  + Choose a subnet in the same Availability Zone as your EC2 instance
  + Use the Security Group you created earlier
  + Leave other options on their default settings and **Create**
  + Once the cluster is created, find your ElastiCache for Redis node name
    1. On the AWS Console, choose Services, then ElastiCache
    2. On the ElastiCache dashboard, choose Redis, then click on your cluster and look at the Description tab
    3. Note your endpoint name. You will need it later!
    4. modify your ElastiCache Security Group to allow inbound access from your EC2 instance Redis (TCP port 6379)

* + Verify you can access the redis> console from your EC2 instance

$ redis-cli –h <redis node name>

Example:   
$ $ redis-cli -h redis-lab.qwe34ytz.0001.usw2.cache.amazonaws.com

redis-lab.sjyq2g.0001.usw2.cache.amazonaws.com:6379>

To exit from the redis> prompt, use CTRL-D

1. Download and Prepare Landsat scenes

See documentation at

<https://aws.amazon.com/public-data-sets/landsat/>

From your EC2 instance, download the Landsat scenes

$ wget https://landsat-pds.s3.amazonaws.com/c1/L8/scene\_list.gz

* + Unzip the scene list  
    $ gunzip scene\_list.gz
  + Trim the list to the last 250,000 scenes  
    $ cp scene\_list scene\_list.orig  
    $ tail -n 250000 scene\_list.orig > scene\_list

1. Load to MySQL

* Log into the mysql> console

$ mysql –h <mysql endpoint> -u <user name> -p

Example:   
$ mysql -h sql-lab.cxpjiluqq0c9.us-west-2.rds.amazonaws.com -u awsuser -p

* + Create a Landsat database  
      
    mysql> CREATE DATABASE landsat;  
    mysql> USE landsat;
  + Create the scene\_list table  
      
    mysql> CREATE TABLE scene\_list (productId VARCHAR(64),entityId VARCHAR(64), acquisitionDate DATETIME,cloudCover DECIMAL(5,2),processingLevel VARCHAR(8),path INT,row INT,min\_lat DECIMAL(8,5),min\_lon DECIMAL(8,5),max\_lat DECIMAL(8,5),max\_lon DECIMAL(8,5),download\_url VARCHAR(128));
  + Add an index

mysql> CREATE INDEX idx\_entityId ON scene\_list (entityId);

* + Load the landsat data  
    mysql> LOAD DATA LOCAL INFILE 'scene\_list' INTO TABLE scene\_list FIELDS TERMINATED BY ',';

1. Load to Redis
   * Create the file sql2redis.py, containing the following code. Be sure to replace items in red with your own endpoints, user name and password.

#!/usr/bin/python

import redis

import MySQLdb

from collections import Counter

r = redis.StrictRedis('<your redis node>',port=6379,db=0)

database = MySQLdb.connect("<your MySQL node>","<your username>","<your password>","landsat")

cursor = database.cursor()

select = 'SELECT productId, entityId, UNIX\_TIMESTAMP(acquisitionDate), cloudCover, processingLevel, path, row, min\_lat, min\_lon, max\_lat, max\_lon, download\_url, CONCAT(productId, entityId) AS Id FROM scene\_list'

cursor.execute(select)

data = cursor.fetchall()

for row in data:

r.zadd('cCov',row[3],row[12])

r.zadd('acqDate',row[2],row[12])

cursor.close()

database.close()

* This script copies from the MySQL database to two sorted sets: cCov (with the Cloud Cover information) and acqDate (with the acquisition date information)
* From the linux console, run the python script to cache data in Redis  
  $ python sql2redis.py
  + this will take a few minutes to run. To check progress, log into your Redis node and run  
    redis> ZCOUNT cCov -inf +inf

1. Run SQL query

* Log in to your MySQL node and run a query

mysql> USE landsat;

mysql> SELECT DISTINCT(a.entityId) AS Id, a.productId, a.acquisitionDate, a.cloudCover

FROM scene\_list a

INNER JOIN (

SELECT productid, entityId, acquisitionDate

FROM scene\_list

WHERE acquisitionDate > (

SELECT MAX(acquisitionDate)

FROM scene\_list

WHERE acquisitionDate < CURDATE() - INTERVAL 1 YEAR)

) b ON a.productId = b.productId AND a.entityId = b.entityId AND a.acquisitionDate = b.acquisitionDate

WHERE cloudCover < 50

ORDER BY acquisitionDate;

This generates a list of all the satellite images during the last year which have less than 50% cloud cover

1. Run Redis query
   * Calculate the Unix Epoch for **one year ago today**

$ date --date="<DD-MMM-YY>" +%s  
  
Example: if today is August 29, 2018, then:  
$ date --date="29-AUG-17" +%s

1503964800

* + Log in to your Redis node and run

redis> zunionstore temp:cCov 1 cCov

redis> zremrangebyscore temp:cCov 50 inf

redis> zunionstore temp:acqDate 1 acqDate

redis> zremrangebyscore temp:acqDate 0 <epoch date of 1 year ago today>

redis> zinterstore out 2 temp:cCov temp:acqDate WEIGHTS 1 0

* Again, this generates a list of all the satellite images during the last year which have less than 50% cloud cover
  + Redis commands that start with a “z” are using sorted sets
  + You used zunionstore to copy the cCov sorted set to temp:cCov, then used zremrangebyscore to remove all items in temp:cCov with 50% or more cloud cover
  + You used zunionstore to copy the acqDate sorted set to temp:acqDate, then used zremrangebyscore to remove all items in temp:acqDate from before today
  + You used zinterstore to find items that appear in both sorted sets, finding only satellite images with less than 50% cloud cover
* Is this the same value from the SQL query?