Getting Started with ElastiCache for Redis
Hands-on Lab

1. Create an Amazon EC2 t2.micro instance with Amazon Linux
   - See documentation at
   http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html
   - Create and use a security group that allows inbound TCP access using SSH (port 22) and MYSQL (port 3306) plus a custom rule to allow access from Redis (port 6379)

   You might get an automated warning that your EC2 instance is "open to the world", because we're not limiting the source range for SSH. This is expected. In a production system, you'll want to provide a limited IP range for allowed SSH access. For this lab, 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.
2. 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 ~]$
```

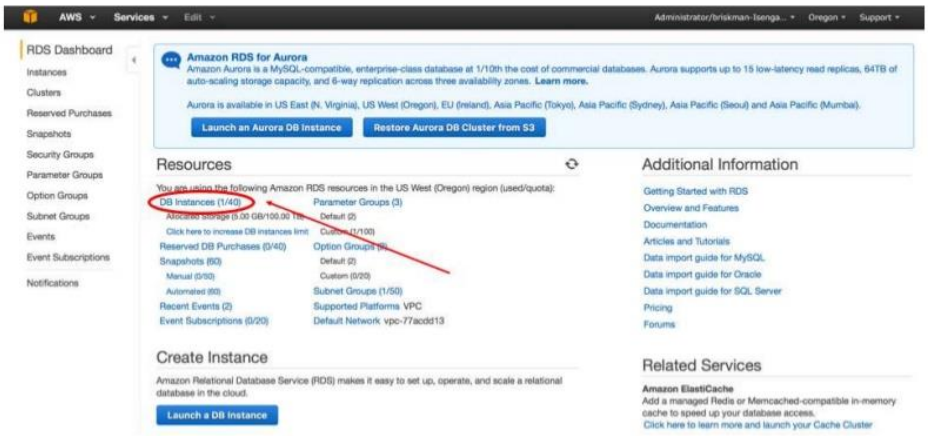
   - Install MySql and Redis development clients
   $ sudo yum install mysql-devel
   $ sudo yum install gcc
   $ sudo pip install MySQL-python
   $ sudo pip install redis

© 2018 CHAITANYA GAAJULA - ALL RIGHTS RESERVED

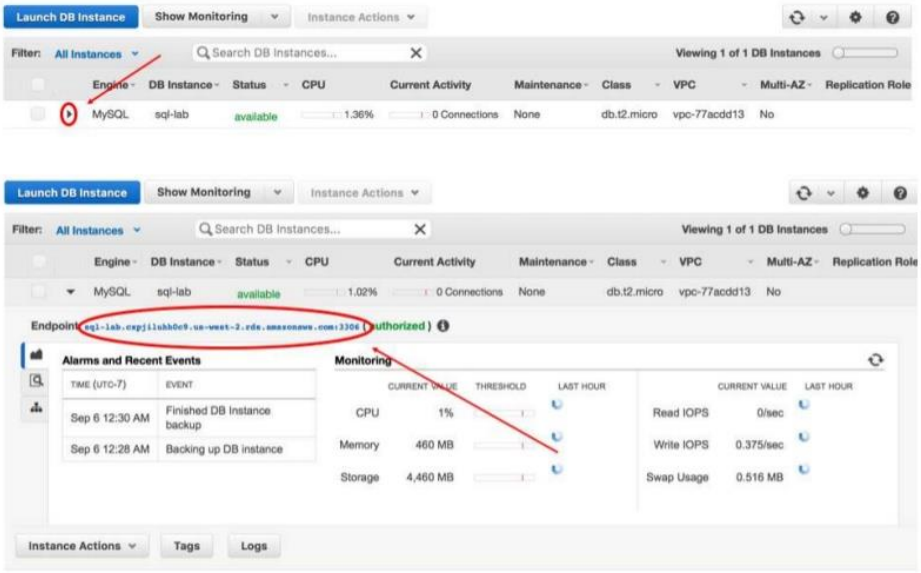
- Install Redis command line interface
  $ sudo yum --enablerepo=epel install redis

3. Create an Amazon RDS MySQL db.t2.micro instance
   - See documentation at
     http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html
   - 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)e
   - Once the instance is created, find your mysql node name
     i. On the AWS Console, choose Services, then RDS

ii. On the RDS dashboard, choose DB Instances



iii. Select the ▸ next to your DB Instance



iv. Note your endpoint name. You will need it later!
When using the endpoint name, you usually should not use the port extension (:3306), just the name.

- Verify you can access the mysql> console from your EC2 instance
  $ mysql –h <mysql node name> -u <user name> -p

Example:

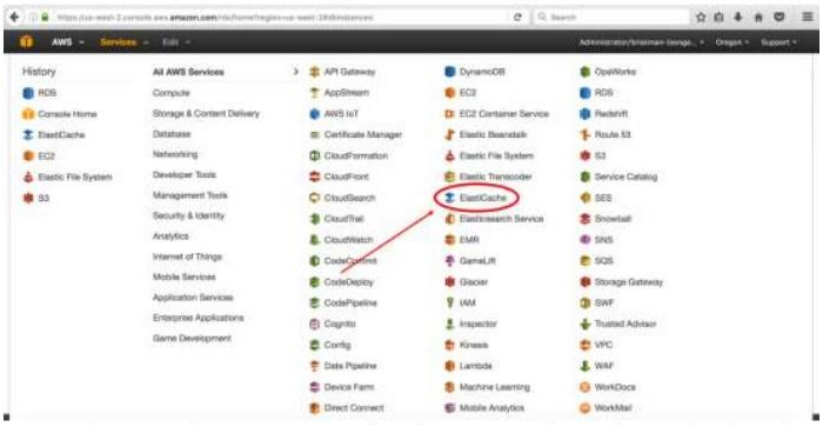
```
$ mysql -h sql-lab.cxpjiluqq0c9.us-west-2.rds.amazonaws.com  -u awsuser  -p

Enter password:

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9999
Server version: 5.6.27-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

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

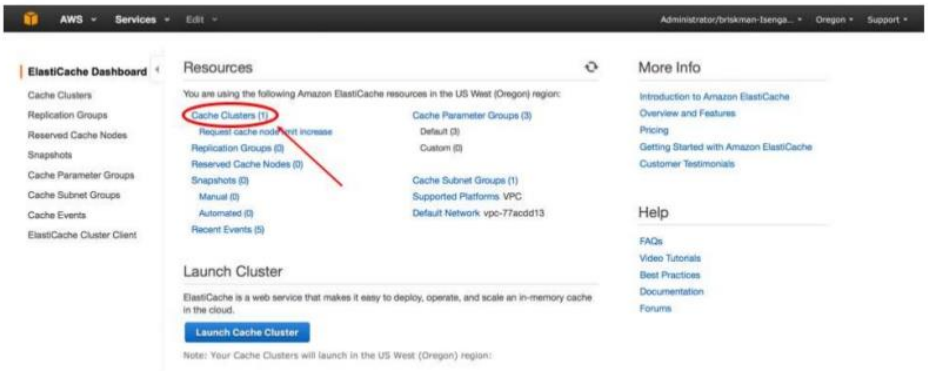
4. Create an Amazon ElastiCache cache.t2.micro instance with Redis
   - See documentation at

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

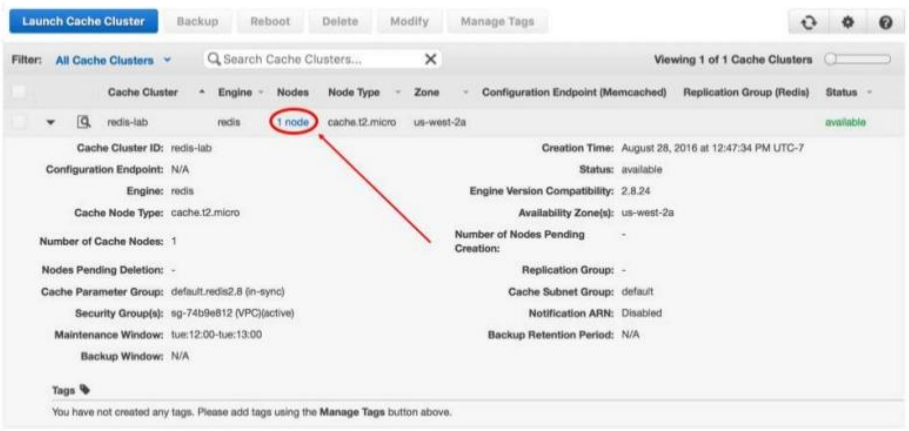
- Name the instance redis-lab. Be sure to choose the cache.t2.micro instance size
- Disable replication (just a single node)
- Once the instance is created, find your ElastiCache for Redis node name



i. On the AWS Console, choose Services, then ElastiCache
ii. On the ElastiCache dashboard, choose Cache Clusters



iii. Select the ▸ next to your Cache Cluster name, then click



on the Nodes

iv. Note your endpoint name. You will need it later!

- 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

5. 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 http://landsat-pds.s3.amazonaws.com/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
6. Load to MySQL
   - Log into the mysql> console
   - Create a landsat database
     mysql> CREATE DATABASE landsat;
     mysql> USE landsat;
   - Create the scene_list table
     mysql> CREATE TABLE scene_list (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));
   - Load the landsat data
     mysql> LOAD DATA LOCAL INFILE 'scene_list' INTO TABLE scene_list FIELDS TERMINATED BY ',';

7. 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 entityId, UNIX_TIMESTAMP(acquisitionDate), cloudCover,
processingLevel, path, row, min_lat, min_lon, max_lat, max_lon, download_url FROM
scene_list'
cursor.execute(select)
data = cursor.fetchall()

for row in data:
        r.hmset(row[0],{'acquisitionDate':row[1],'cloudCover':row[2],'processingLevel':row[
        3],'path':row[4],'row':row[5],'min_lat':row[6],'min_lon':row[7],'max_lat':row[8],'max_
        lon':row[9],'download_url':row[10]})
        r.zadd('cCov',row[2],row[0])
        r.zadd('acqDate',row[1],row[0])
cursor.close()
database.close()
```

   - 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> DBSIZE

8. Run SQL query
    - Log in to your MySQL node and run a query

```
mysql> SELECT DISTINCT(a.entityId) AS Id, a.cloudCover
FROM scene_list a
INNER JOIN (
        SELECT entityId, acquisitionDate
        FROM scene_list
        WHERE acquisitionDate > (
                SELECT MAX(acquisitionDate)
                FROM scene_list
                WHERE acquisitionDate < CURDATE() - INTERVAL 1 YEAR
        )
) b ON a.entityId = b.entityId AND a.acquisitionDate = b.acquisitionDate
        WHERE cloudCover < 50
        ORDER BY Id;
```

    - This generates a list of all the satellite images during the last year which have less than 50% cloud cover
    - Note how long it takes to get an answer

9. Run Redis query
    - 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 1465862400
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
    - Note how long it takes to get an answer