CMPE 281 - LAB #5 - AWS NoSQL MongoDB Cluster

In this Lab, you will be deploying a Mongo DB Replica Set Cluster and running some DB Queries against the Cluster.

Lab Files

- https://github.com/paulnguyen/cmpe281/tree/master/labs/lab5

References

- https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/
- https://docs.mongodb.com/manual/tutorial/deploy-replica-set/
- https://docs.mongodb.com/manual/tutorial/deploy-replica-set-with-keyfile-access-control/#deploy-repl-set-with-auth

Part 1 - Setup MongoDB AMI

https://gist.github.com/calvinh8/c99e198ce5df3d8b1f1e42c1b984d7a4

Launch Ubuntu Server 16.04 LTS

1. AMI:	Ubuntu Server 16.04 LTS (HVM)
2. Instance Type:	t2.micro
3. VPC:	cmpe281
4. Network:	public subnet
5. Auto Public IP:	no
6. Security Group:	mongodb-cluster
7. SG Open Ports:	22, 27017
8. Key Pair:	your key pair (i.e. cmpe281-us-west-2 or cmpe281-us-east-1)

Allocate & Assign an Elastic IP to Mongo Instance

1. Allocate Elastic IP:	Scope VPC
2. Name Elastic IP:	mongodb
3. Associate Elastic IP:	Instance = Mongo EC2 Instance

SSH into Mongo Instance

```
ssh -i <key>.pem ubuntu@<public ip>
```

Install MongoDB

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
9DA31620334BD75D9DCB49F368818C72E52529D4

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/
4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb.list

sudo apt update
sudo apt install mongodb-org
```

MongoDB Keyfile

```
openssl rand -base64 741 > keyFile
sudo mkdir -p /opt/mongodb
sudo cp keyFile /opt/mongodb
sudo chown mongodb:mongodb /opt/mongodb/keyFile
sudo chmod 0600 /opt/mongodb/keyFile
```

Config mongod.conf

```
sudo vi /etc/mongod.conf

1.  remove or comment out bindIp: 127.0.0.1
    replace with bindIp: 0.0.0.0 (binds on all ips)

    # network interfaces
    net:
        port: 27017
        bindIp: 0.0.0.0

2. Uncomment security section & add key file

security:
    keyFile: /opt/mongodb/keyFile

3. Uncomment Replication section. Name Replica Set = cmpe281

replication:
    replSetName: cmpe281

4. Create mongod.service

sudo vi /etc/systemd/system/mongod.service

    [Unit]
        Description=High-performance, schema-free document-oriented database
        After=network.target

    [Service]
        User=mongodb
        ExecStart=/usr/bin/mongod --quiet --config /etc/mongod.conf

    [Install]
        WantedBy=multi-user.target
```

5. Enable Mongo Service

```
    sudo systemctl enable mongod.service
```

6. Restart MongoDB to apply our changes

```
    sudo service mongod restart
    sudo service mongod status
```

Save to AMI Image

AMI: **mongo-ami**
Initialize the replica set. Replace Host Names with Public IP or DNS Names.

Using **mongo-ami,** launch three free tier instances with their own Elastic IPs.

Edit /etc/hosts in each EC2 Instance adding local host names for Public IPs.
For example:

```
    54.183.205.52   primary
    54.241.133.243  secondary1
    52.53.242.223   secondary2
```

Set each node's hostname as follows:

```
   sudo hostnamectl set-hostname <hostname>

   For example (for primary node):
   sudo hostnamectl set-hostname primary
```

Test each instance using the following to make sure hostnames are correct:

```
    hostname -f
    telnet <host or ip> 27017 (i.e. primary, secondary1, secondary2)
```

Initialize the Replica Set

```
    mongo  (run as local client on primary)

    rs.initiate( {
       _id : "cmpe281",
       members: [
          { _id: 0, host: "primary:27017" },
          { _id: 1, host: "secondary1:27017" },
          { _id: 2, host: "secondary2:27017" }
       ]
    })

    rs.status()
```

Troubleshooting: If you have connectivity issues, check that mongo is up and running and use "telnet" to try to connect to sceondaries from primary (and vice versa)

```
    sudo service mongod restart
    sudo service mongod status

    telnet <host or ip> 27017
```

Create Admin Account

The default MongoDB configuration is wide open, meaning anyone can access the stored databases unless your network has firewall rules in place.

Create an admin user to access the database.

```
    mongo
```

Select admin database.

```
    use admin
```

Create admin account.

```
    db.createUser( {
        user: "admin",
        pwd: "*****",
        roles: [{ role: "root", db: "admin" }]
    });
```

Login to Primary as Admin:

```
    mongo -u <user> -p <password> --authenticationDatabase admin
```

Login to Mongo Remote

```
    mongo -u <user> -p <password> <mongo host ip> --authenticationDatabase admin
```

Connect to Primary and Test DB (I.E. - Commands from Your Desktop using RoboMongo)

```
db.test.save( { a : 1 } )   // save simple document
db.test.find()              // find document
```

Part 2 - Bios MySQL and MongoDB Data Queries

See: ***https://github.com/paulnguyen/cmpe281/tree/master/labs/lab5***

Write the Equivalent Mongo Queries for the following MySQL Queries:

1 - Count of Records/Documents

```
select count(*) from person
```

2 - Find Bios with Birth Date before 1950

```
select first_name, last_name, birth_date
from person
where birth_date < date('1950-01-01')
```

3 - Get a Unique Listing of all the Awards (in DB/Collection) granted

```
select distinct(a.award_name)
from person_awards pa, awards a
where pa.award_id = a.award_id
```

4 - Get a Sorted Listing of all the First Names (ascending order)

```
select first_name
from person
order by 1
```

5 - Get a Sorted Listing of all the First Names (descending order)

```
select first_name
from person
order by 1 desc
```

6 - Count the number of BIOS that don't yet have an award

```
select count(*) from person p
where not exists
    (select 1 from person_awards
     where person_id = p.person_id)
```

7 - Display the System ID (Primary Key) for the BIO in Query 6

```
select p.person_id from person p
where not exists
    (select 1 from person_awards
     where person_id = p.person_id)
```

8 - Display names (first and last) from BIOS with 1 Contribution AND 2 Awards

```
select p.first_name, p.last_name
from person p
where (select count(*) from contribs c where c.person_id = p.person_id) = 1
and (select count(*) from person_awards pa where pa.person_id = p.person_id) = 2
```

9 - Display names (first and last) from BIOS with 1 Contributions OR 2 Awards

```
select p.first_name, p.last_name
from person p
where (select count(*) from contribs c where c.person_id = p.person_id) = 1
or (select count(*) from person_awards pa where pa.person_id = p.person_id) = 2
```

10 - List all the Awards for a BIO

```
select p.first_name, p.last_name, a.award_name
from awards a, person_awards pa, person p
where a.award_id = pa.award_id
and p.person_id = pa.person_id
and p.person_id = 1
```

Part 3 (Optional) - Partition Tolerance Testing:

In this part, you will be testing the **Partition Tolerance** using the procedures described in the following article: https://www.infoq.com/articles/jepsen. Note: you don't have to follow the directions in the article "verbatim". Adjust the steps as you see necessary -- for example, in how you create a partition in AWS.

Notes:

- Use the steps in the article as guidance only.
- Feel free to diverge from those steps as needed to accomplish your testing goals.
- Using alternative testing programs, programming languages, tools and/or approaches to creating network partitions are allowed.
- Please document any steps you take that diverge from the steps in the article.
- ***Partition Testing Experiment:***
  - Set up your cluster as AWS EC2 Instances. (# of Nodes and Topology is open per your design)
    - Make sure to note your approach to creating a "network partition" for experiments.
  - Set up the Experiments (i.e. Test Cases) to answer the following questions:
    - CP (Mongo DB Cluster):
      - How does the system function during normal mode (i.e. no partition)
      - What happens to the master node during a partition?
      - Can stale data be read from a slave node during a partition?
      - What happens to the system during partition recovery?
  - Run the Experiments and Record results.