Rashmi Banthia

# CSCI E63 - Final Project
# Redis - No SQL Database

# Table of Contents

# Redis - NoSQL Database

## Objectives

- Install ***Redis Cluster*** 3.0.0 (Released April 2015) on AWS EC2 (3 Node cluster) and configure a ***Master Slave replication***.
- Read MovieLens dataset and store it in REDIS (Sets) - Run queries - Using Java Client.
- REDIS Mass Insertion Protocol (with Redis Pipeline) - 21 Million Rows Bulk Insert

## Redis (Remote Dictionary Server)

- Advanced Key Value Cache and Store (Persistence)
- Fast (In Memory)
- Open Source (BSD Licensed)
- Simple and Flexible
- Currently most popular key-value database

51 systems in ranking, May 2015

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| May 2015 | Apr 2015 | May 2014 | | | May 2015 | Apr 2015 | May 2014 |
| 1. | 1. | 1. | Redis | Key-value store | 94.73 | +0.17 | +32.69 |
| 2. | 2. | 2. | Memcached | Key-value store | 33.11 | -0.93 | +1.37 |
| 3. | 3. | ↑ 4. | Amazon DynamoDB | Multi-model | 14.87 | +0.29 | +5.60 |
| 4. | 4. | ↓ 3. | Riak | Key-value store | 12.89 | -0.20 | +2.36 |
| 5. | 5. | 5. | Ehcache | Key-value store | 7.92 | +0.21 | +1.47 |
| 6. | 6. | 6. | Hazelcast | Key-value store | 5.82 | +0.14 | +1.83 |
| 7. | ↑ 8. | ↑ 10. | OrientDB | Multi-model | 3.99 | +0.62 | +2.31 |
| 8. | ↓ 7. | ↓ 7. | Berkeley DB | Key-value store | 3.60 | -0.32 | +0.53 |
| 9. | 9. | 9. | Oracle Coherence | Key-value store | 3.30 | -0.03 | +0.72 |
| 10. | 10. | ↓ 8. | Amazon SimpleDB | Key-value store | 2.96 | -0.24 | +0.00 |

http://db-engines.com/en/ranking/key-value+store

# Super Quick Installation

(Follow this if you just want to get started, Redis Cluster on EC2 - see next page)

**Installation** (On Mac / *nix)

-Couldn't be any simpler (http://redis.io/download) :

```
$ wget http://download.redis.io/releases/redis-3.0.0.tar.gz
$ tar xzf redis-3.0.0.tar.gz
$ cd redis-3.0.0
$ make
```

```
$ src/redis-server
$ src/redis-cli
redis> set foo bar
OK
redis> get foo
"bar"
```

**Windows** - see http://redis.io/download and https://github.com/MSOpenTech/redis

**Redis Cluster (Redis Version 3.0.0 - Released April 2015)**

**Redis Cluster 101**

- Redis Cluster provides a way to run a Redis installation where data is automatically sharded across multiple Redis nodes.
- Redis Cluster  has the ability to automatically split your dataset among multiple nodes.
- Redis Cluster has the ability to continue operations when a subset of the nodes are experiencing failures or are unable to communicate with the rest of the cluster.

**Installation Overview**

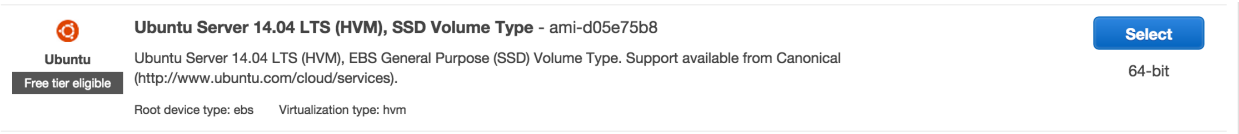
- 3 EC2 Instances + 1 EC2 Instances for Master Slave
- Do the following on all 3 Instances (We'll setup master slave later)
- Download and Extract Redis
- Install build-essential, make
- Make Redis libraries
- Install tk8.5 tcl8.5 - (C functions/Libraries)
- Update configuration (redis.conf)
- Start Redis Server
- Install Ruby and Ruby Redis gems
- Create the cluster and test it!!

# **Redis Cluster Installation** (3 EC2 Instances)

Redis Cluster - Redis Cluster is the mechanism to distribute shards to multiple Redis instances

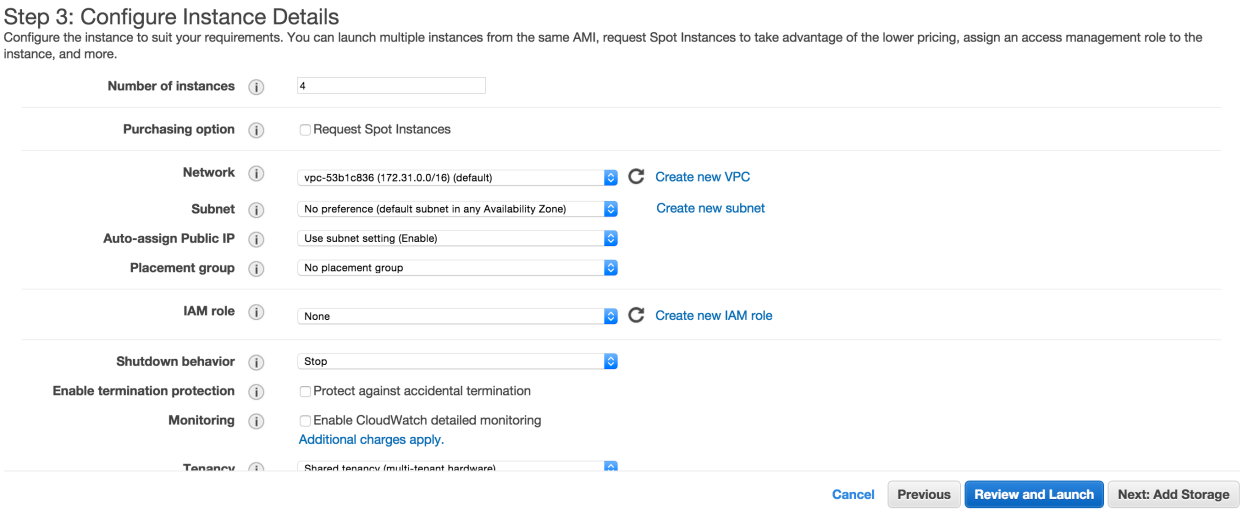
(Before moving on make sure you have a Key Pair set up)
Watch Video here -> https://www.youtube.com/watch?v=s4YpCA2Y_-Q

**Step 1:** AWS EC2 -> Instances -> Launch an Instance -> Select Ubuntu Server 14.04




**Step 2:** Choose an Instance Type
Select Memory Optimized r3.large
(t2.micro works well too)

**Step 3:** Configure Instance Details
Select Number of Instances: 4 *(We will use 1 for Master Slave replication)*

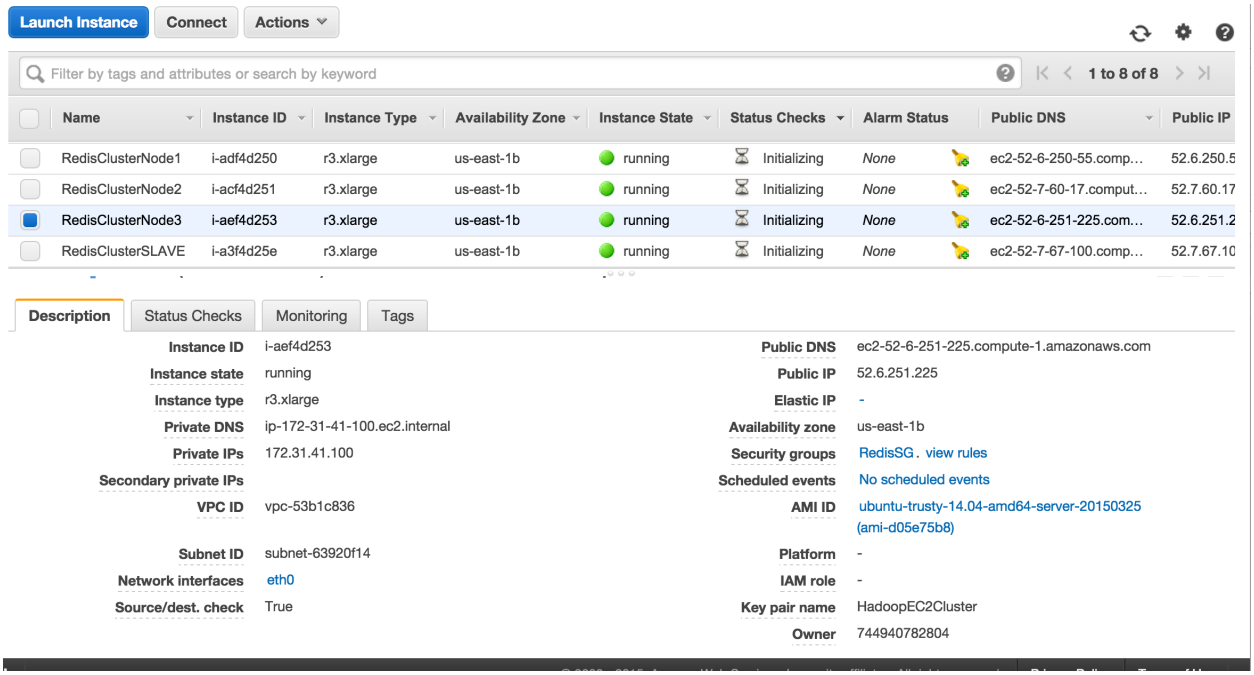



**Step:4** Configure Security Group

| Type | Protocol | Port Range | Source |
|---|---|---|---|
| SSH | TCP | 22 | 0.0.0.0/0 |
| All TCP | TCP | 0 - 65535 | 0.0.0.0/0 |
| Custom TCP Rule | TCP | 6379 | 0.0.0.0/0 |
| HTTP | TCP | 80 | 0.0.0.0/0 |

**Review and Launch** (Use the existing key pair)

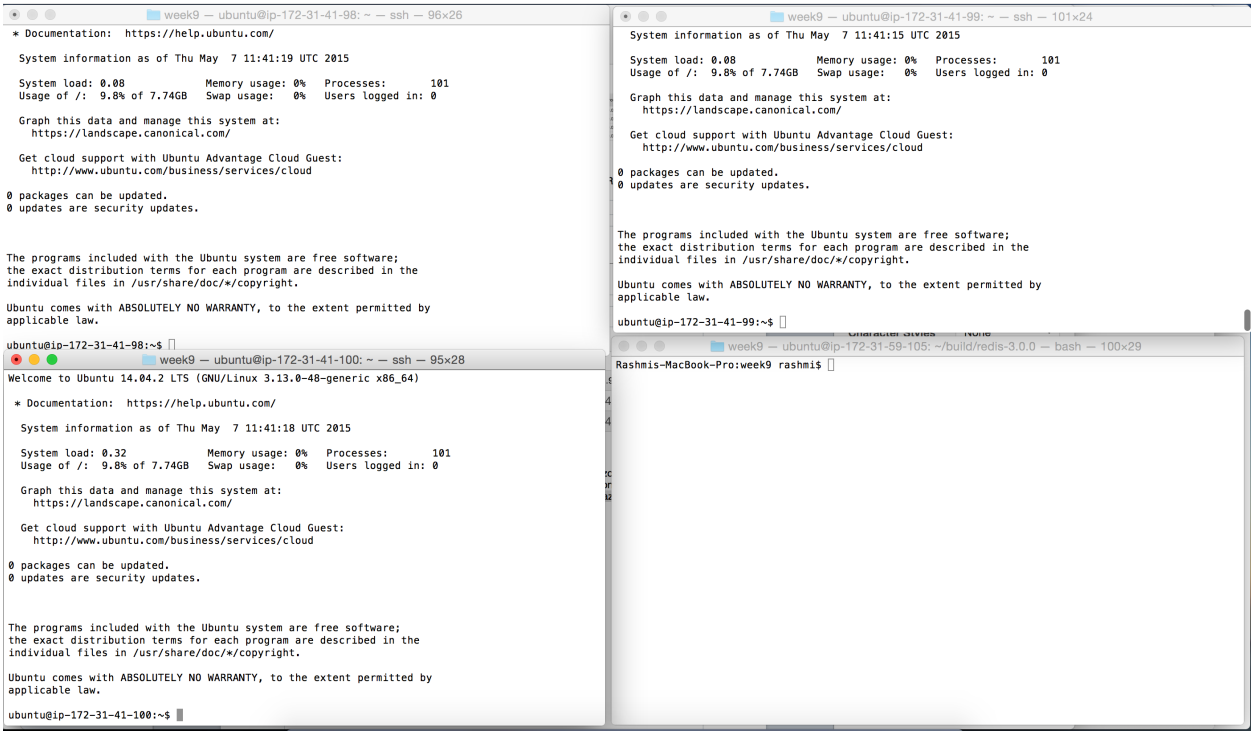
- Rename the instances as RedisClusterNode1, RedisClusterNode2 and RedisClusterNode3




Lets fill up the following table:

| | Public IP | Private IP |
|---|---|---|
| RedisClusterNode1 | 52.6.250.55 | 172.31.41.98 |
| RedisClusterNode2 | 52.7.60.17 | 172.31.41.99 |
| RedisClusterNode3 | 52.6.251.225 | 172.31.41.100 |

**SSH to all three instances**
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-6-250-55.compute-1.amazonaws.com
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-7-60-17.compute-1.amazonaws.com
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-6-251-225.compute-1.amazonaws.com




## Download and Extract Redis on all 3 instances

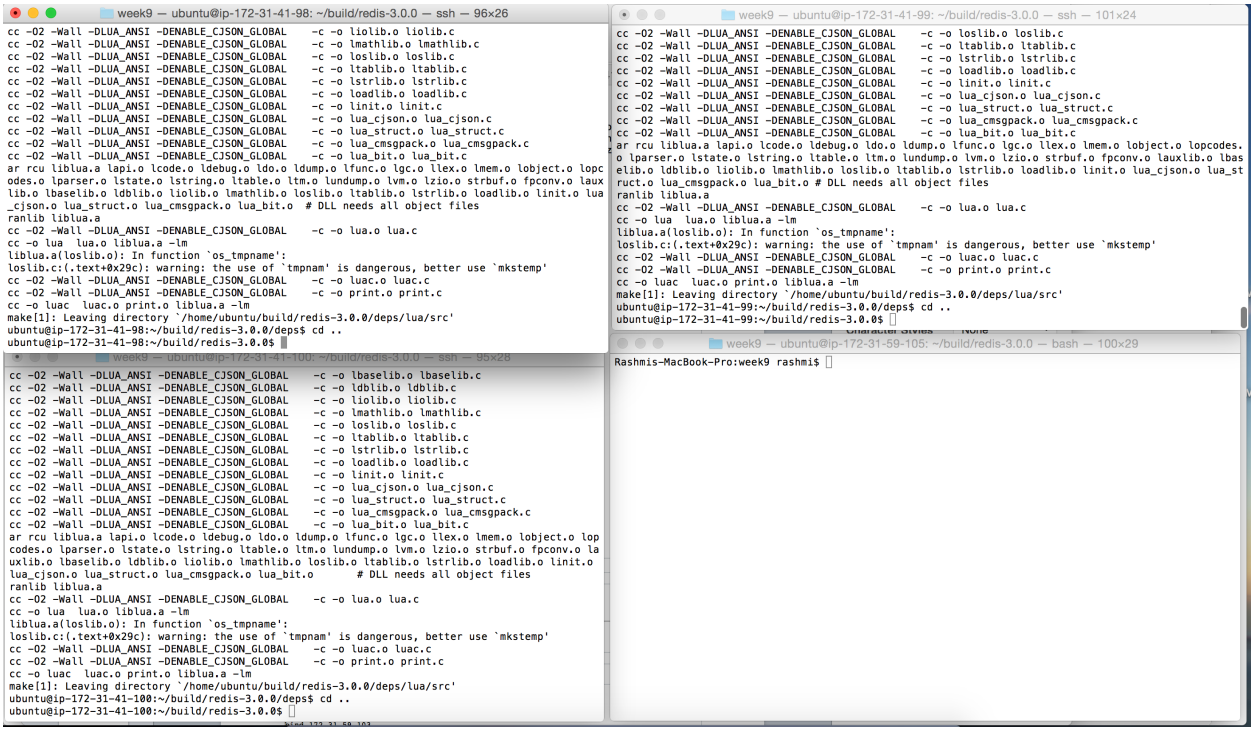
```
mkdir build && cd build
wget http://download.redis.io/releases/redis-3.0.0.tar.gz
tar -xvzf redis-3.0.0.tar.gz
cd redis-3.0.0/
```

## Now we can build it

```
sudo apt-get install -y make gcc build-essential
sudo apt-get update
sudo apt-get install make
sudo apt-get install gcc
cd deps
make hiredis jemalloc linenoise lua
cd ..
```

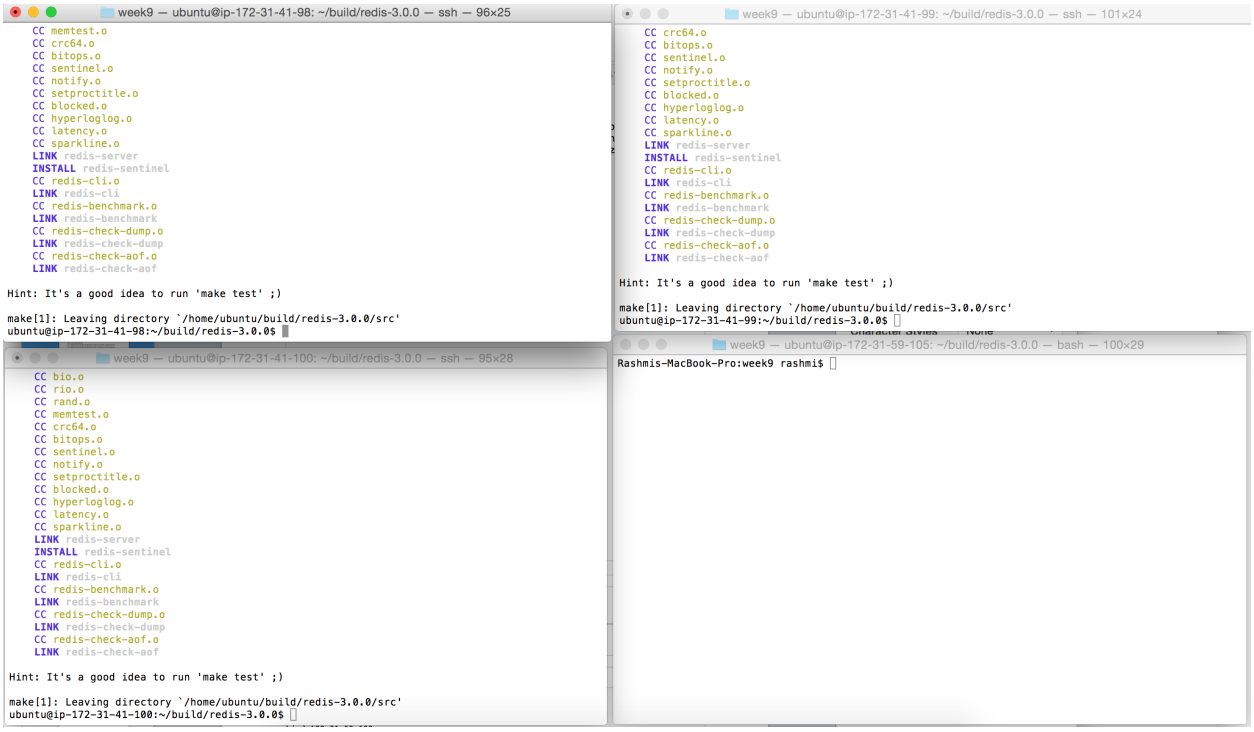
(Watch for any error messages in any of the above steps)

Now my shell looks like this:



Now Run the following command:

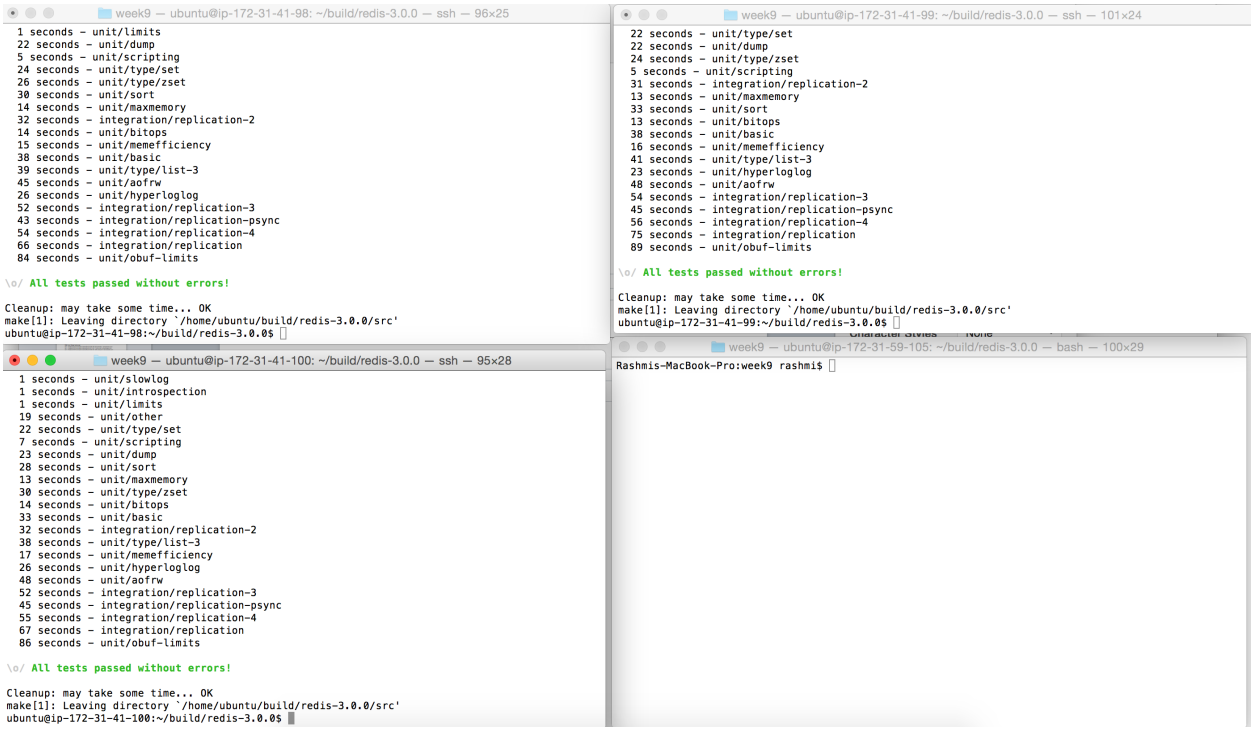
```
make MALLOC=libc
```



Now let's run tests

```
sudo  apt-get install -y tk8.5 tcl8.5
make test
```

(You should see lot of green [ok] messages)




Now lets update the configuration for cluster:

```
sudo vi redis.conf
```

Modify the following:

#On RedisClusterNode1 - redis.conf

```
bind 172.31.41.98
cluster-enabled yes
cluster-config-file nodes-6379.conf
cluster-node-timeout 5000
```

#On RedisClusterNode2 - redis.conf

```
bind 172.31.41.99
cluster-enabled yes
cluster-config-file nodes-6379.conf
cluster-node-timeout 5000
```

#On RedisClusterNode3 - redis.conf

```
bind 172.31.41.100
cluster-enabled yes
cluster-config-file nodes-6379.conf
cluster-node-timeout 5000
```

Next start Redis Server On all 3 nodes

```
src/redis-server ./redis.conf
```

Note: At any point if you want to **shutdown** redis-server:
connect to redis client -

```
ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$ src/redis-cli -h 172.31.12.238
172.31.12.238:6379>shutdown
```

Now we have 3 nodes and we need to connect them: (Open another tab on each of the terminal)

Install Ruby Redis gems

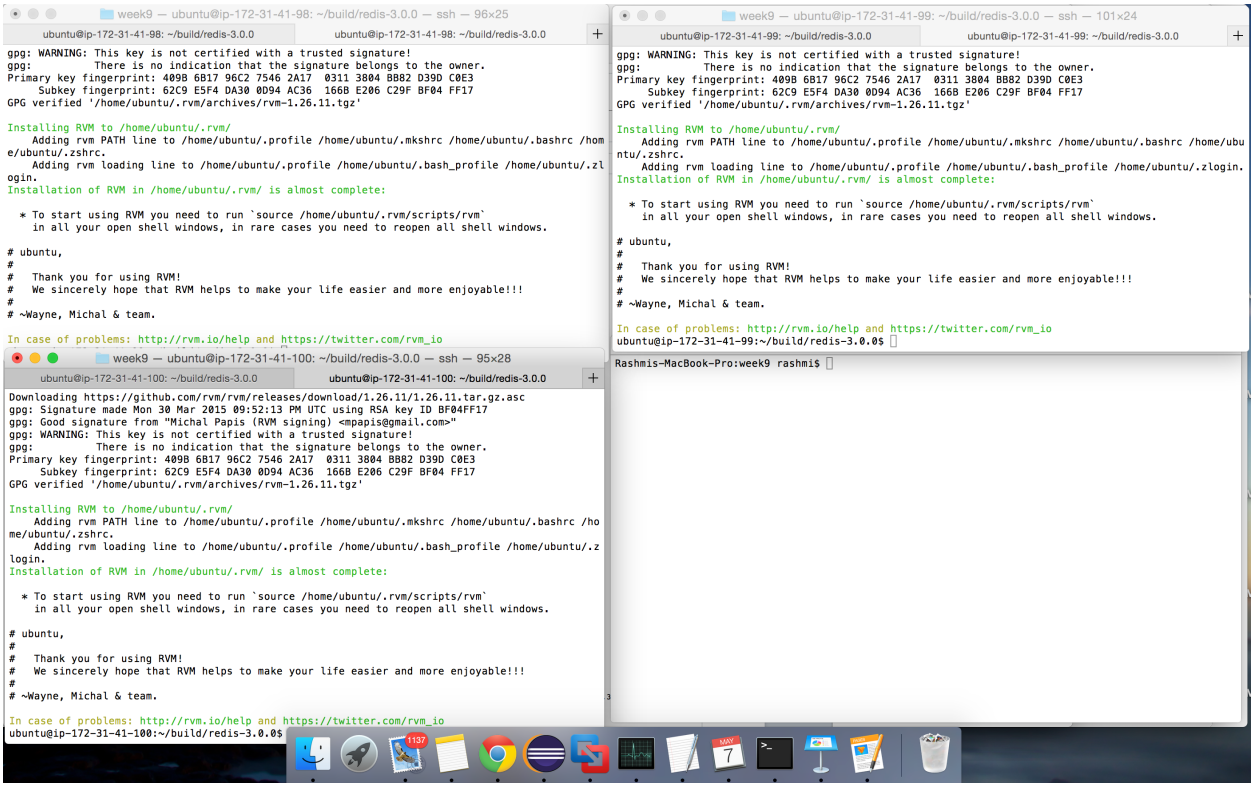
```
cd build/redis-3.0.0/
\curl -L https://get.rvm.io | bash -s stable
```

-This command gives error GPG signature verification failed
Run the following command (Copy it from your terminal)

```
gpg --keyserver hkp://keys.gnupg.net --recv-keys
409B6B1796C275462A1703113804BB82D39DC0E3
```

Run this again

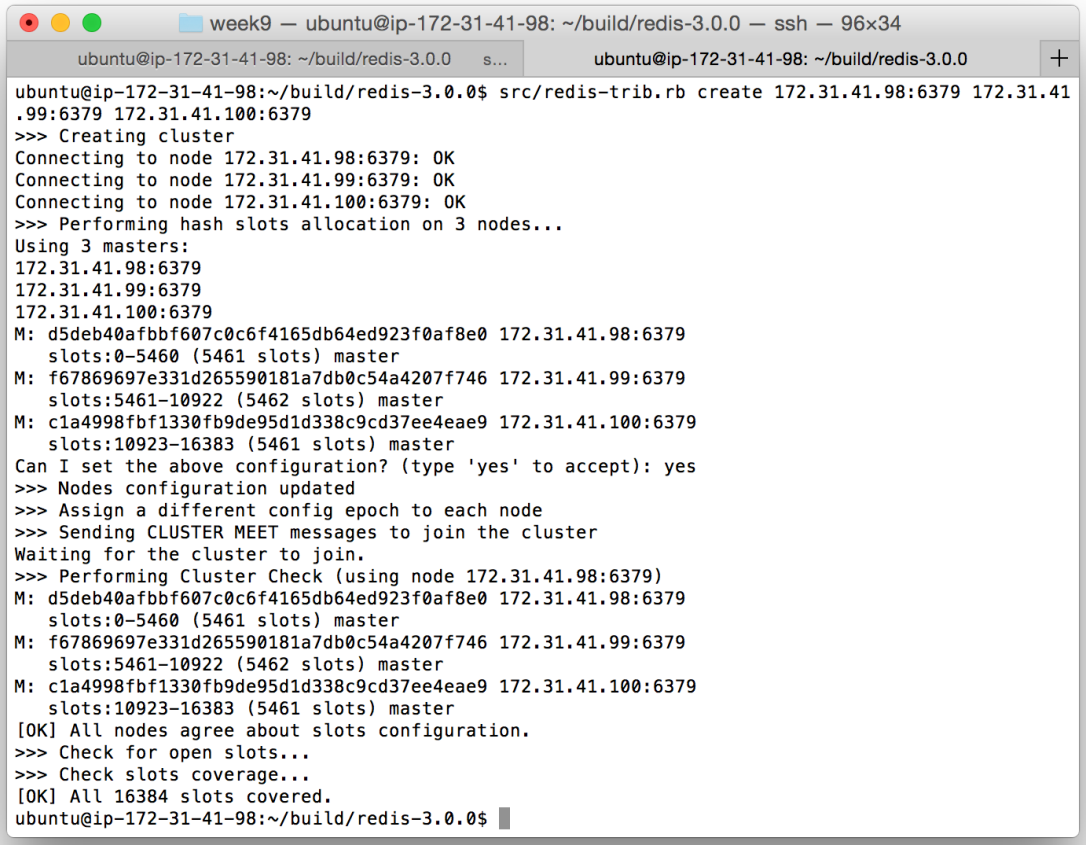
```
\curl -L https://get.rvm.io | bash -s stable
```

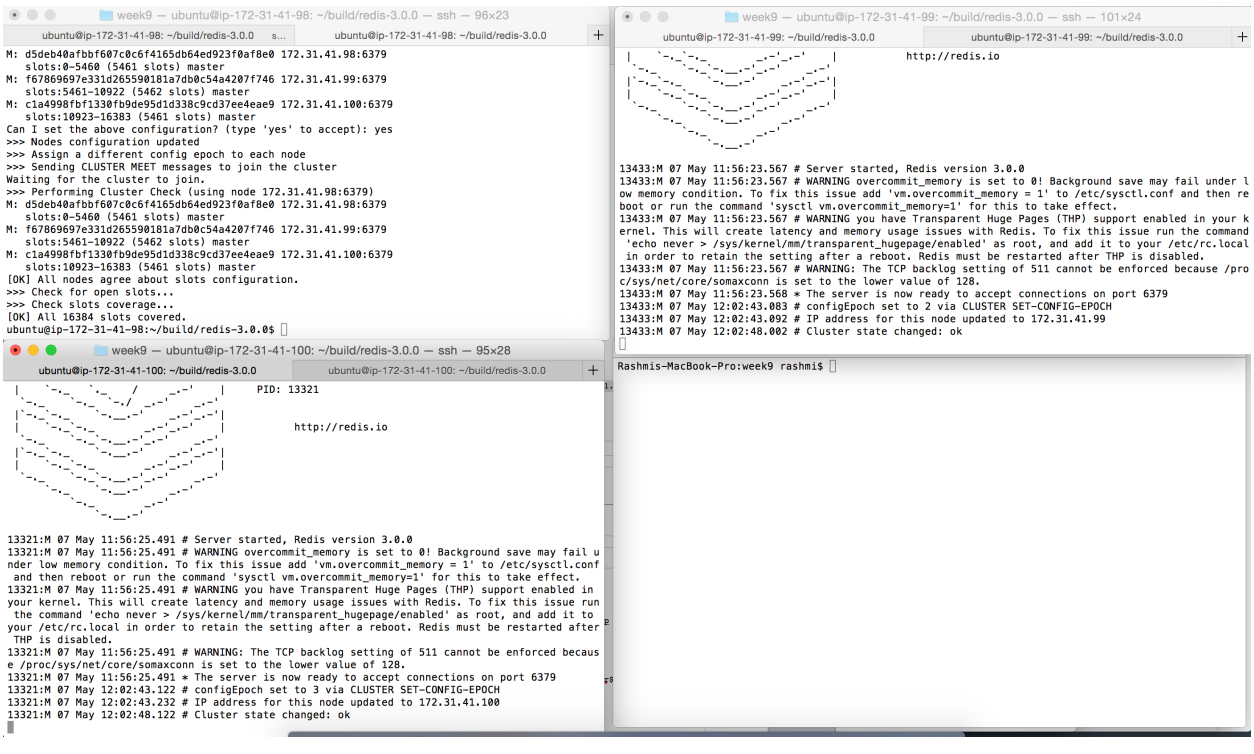



```
source ~/.rvm/scripts/rvm
rvm requirements
rvm install ruby
gem install redis
```

Finally:

```
src/redis-trib.rb create 172.31.41.98:6379 172.31.41.99:6379 172.31.41.100:6379
```



Test the cluster:
On RedisClusterNode2

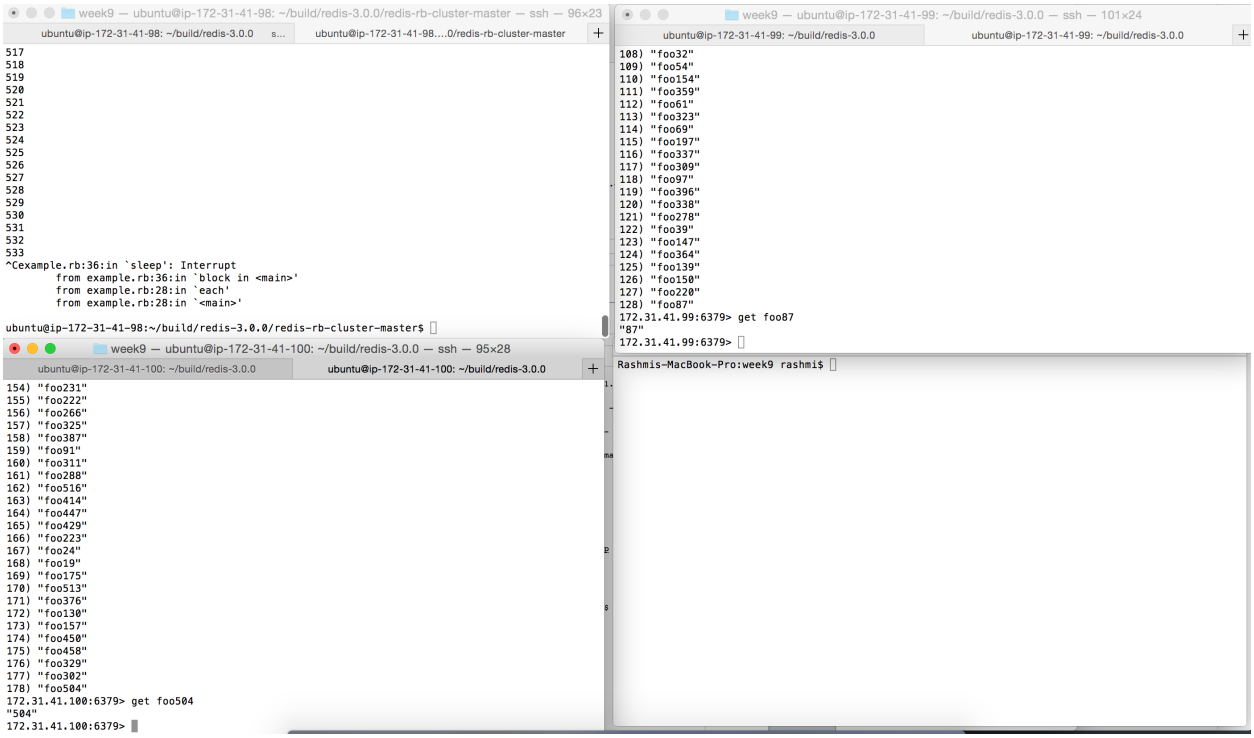
```
src/redis-cli -h 172.31.41.98 cluster nodes
```

```
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$ src/redis-cli -h 172.31.41.98 cluster
nodes
c1a4998fbf1330fb9de95d1d338c9cd37ee4eae9 172.31.41.100:6379 master - 0 1431000234118 3
connected 10923-16383
f67869697e331d265590181a7db0c54a4207f746 172.31.41.99:6379 master - 0 1431000235121 2
connected 5461-10922
d5deb40afbbf607c0c6f4165db64ed923f0af8e0 172.31.41.98:6379 myself,master - 0 0 1
connected 0-5460
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$
```

Benchmarks -

```
wget https://github.com/antirez/redis-rb-cluster/archive/master.zip
sudo apt-get install unzip
unzip master.zip
cd redis-rb-cluster-master

ubuntu@ip-172-31-41-98:~/build/redis-3.0.0/redis-rb-cluster-master$ ruby example.rb
172.31.41.98 6379
```

Interrupt the program. Use **`info`** to check keys on all the clients. To delete all keys run **`flushdb`** on `redis-cli.`



Now,lets setup the Master Slave Replication

# Master Slave Replication 101

- This replication allows for a database service to replicate data written to the master instance to a slave instance. This slave instance could be located within the same facility for scaling out requests or in another facility in order to mitigate a disaster recovery scenario.

- Master-slave replication allows slave Redis servers to be exact copies of master servers.

- Replication is asynchronous because of performance concerns. It means that Redis HA (High Availability) does not guarantee strong data consistency, i.e. It is not possible to ensure the slave actually received a given write, so there is always a window for data loss.

**Add a Slave**
(We are only replicating RedisClusterNode2 - procedure remains the same for other Nodes)

ec2-52-7-67-100.compute-1.amazonaws.com
52.7.67.100

```
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-7-67-100.compute-1.amazonaws.com
sudo apt-get install redis-server
sudo apt-get install redis-tools
```

Configure the Slave

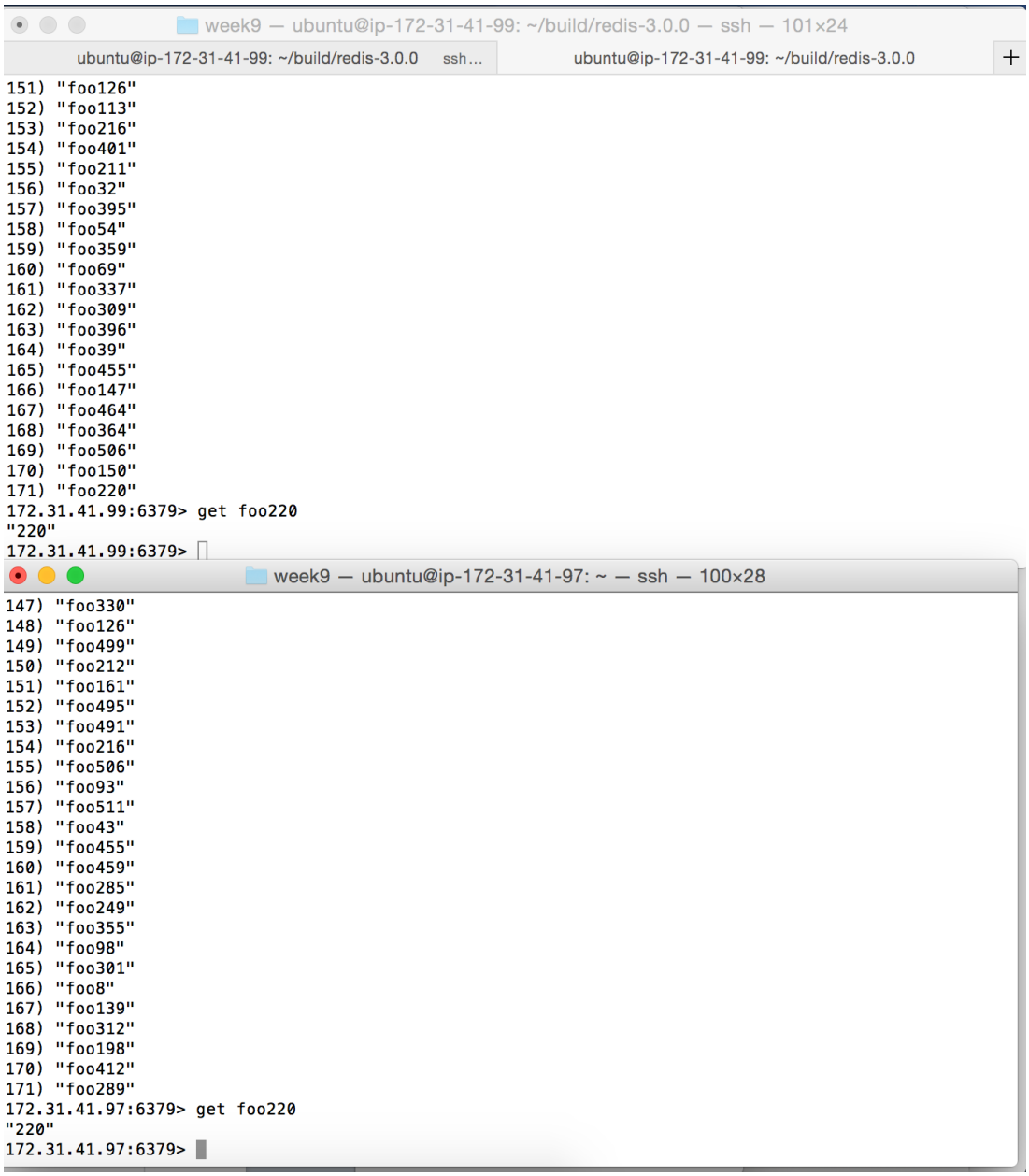
```
sudo vi /etc/redis/redis.conf

bind 172.31.41.97 #Private IP
slaveof 172.31.41.99 6379  #RedisClusterNode2
```

Start the server

```
sudo redis-server /etc/redis/redis.conf
```

Test

```
ubuntu@ip-172-31-41-97:~$ redis-cli -h 172.31.41.97
```


week9 — ubuntu@ip-172-31-41-99: ~/build/redis-3.0.0 — ssh — 101×24
ubuntu@ip-172-31-41-99: ~/build/redis-3.0.0 ssh...
ubuntu@ip-172-31-41-99: ~/build/redis-3.0.0
151) "foo126"
152) "foo113"
153) "foo216"
154) "foo401"
155) "foo211"
156) "foo32"
157) "foo395"
158) "foo54"
159) "foo359"
160) "foo69"
161) "foo337"
162) "foo309"
163) "foo396"
164) "foo39"
165) "foo455"
166) "foo147"
167) "foo464"
168) "foo364"
169) "foo506"
170) "foo150"
171) "foo220"
172.31.41.99:6379> get foo220
"220"
172.31.41.99:6379>
week9 — ubuntu@ip-172-31-41-97: ~ — ssh — 100×28
147) "foo330"
148) "foo126"
149) "foo499"
150) "foo212"
151) "foo161"
152) "foo495"
153) "foo491"
154) "foo216"
155) "foo506"
156) "foo93"
157) "foo511"
158) "foo43"
159) "foo455"
160) "foo459"
161) "foo285"
162) "foo249"
163) "foo355"
164) "foo98"
165) "foo301"
166) "foo8"
167) "foo139"
168) "foo312"
169) "foo198"
170) "foo412"
171) "foo289"
172.31.41.97:6379> get foo220
"220"
172.31.41.97:6379>
```
```

# MovieLens Dataset - Load Users in Cluster

**(Use Java Client - Jedis to load 230,000 users and query)**
On RedisClusterNode1

### Install JAVA

```
sudo apt-get install openjdk-7-jdk
sudo apt-get install unzip
```

### Create directories on RedisClusterNode1:

```
cd ;mkdir -p RedisProject/src/org/hu/e63/
cd RedisProject; mkdir classes; mkdir input; mkdir jars
```

### Download and unzip latest MovieLens Dataset (21million ratings and 132 MB zip file)

```
wget http://files.grouplens.org/datasets/movielens/ml-latest.zip
unzip ml-latest.zip
rm ml-latest.zip
mv ml-latest/ratings.csv input/ratings.csv
rm -R ml-latest/
```

### MovieLens - ratings.csv

```
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv  | head
userId,movieId,rating,timestamp
1,253,3.0,900660748
1,1387,3.0,900660748
1,1407,5.0,900660871
1,1717,5.0,900660871
1,1876,3.0,900660543
1,1882,3.0,900660584
1,1895,4.0,900660518
1,1909,3.0,900660468
1,1911,3.0,900660543
ubuntu@ip-172-31-41-98:~/RedisProject/input$

ubuntu@ip-172-31-41-98:~/RedisProject/input$ ls -l
total 554976
-rw-r--r-- 1 ubuntu ubuntu 568287348 May  7 12:35 ratings.csv
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv  | wc -l
21063129
ubuntu@ip-172-31-41-98:~/RedisProject/input$
```

Now we will load ratings.cv to Redis, Our goal is to be able get query by user. For eg: **User X has rated which movies ?** We will look at other queries later.

Redis is very flexible (as there is so precise schema required), I am storing ratings.csv in Redis sets, Eg:
SADD u:1:m 253
SADD u:1:m 1387
#Above statements create a set with key u:1:m (userid 1) and values [253, 1387] (movie nos.)

Following is MovieLensJedis.java program - which creates SADD commands:
MovieLensJedis.java

```
package org.hu.e63;

import java.io.File;
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisCluster;

public class MovieLensJedis {
	public static void main(String[] args) {
		Jedis jedis = new Jedis("172.31.41.98", 6379);
		System.out.println("Connection to server sucessfully hello!!");
	    //check whether server is running or not
	    System.out.println("Server is running: "+jedis.ping());

		Set<HostAndPort> jedisClusterNodes = new HashSet<HostAndPort>();
		//Jedis Cluster will attempt to discover cluster nodes automatically
		jedisClusterNodes.add(new HostAndPort("172.31.41.98", 6379));
		jedisClusterNodes.add(new HostAndPort("172.31.41.99", 6379));
		jedisClusterNodes.add(new HostAndPort("172.31.41.100", 6379));
		JedisCluster jc = new JedisCluster(jedisClusterNodes);
		try {
			Scanner s = new Scanner(new File("input/ratings.csv"));
		   String s1 = s.nextLine(); //Skip first line
		 String key = "";
		 String member = "";
		 while (s.hasNextLine()){
			s1 = s.nextLine();
			String[] spl = s1.split(",");
			key = "u:"+spl[0]+":m";
			member = spl[1];
			jc.sadd(key, member);
		 }
		}
	  catch(Exception e)
		{
		System.out.println("bye" + e.getMessage());
		}
		jedis.close();
	}
}
```

(I wrote the code on Eclipse on my macbook - then copied over the JAVA, JAR and input files to EC2 instance)

### Copy java and jar files

Remember to change the IP address to connect in .JAVA file

#### Copy JAVA source file

```
scp -i HadoopEC2Cluster.pem ../../workspace/RedisProject/src/org/hu/e63/
MovieLensJedis.java ubuntu@ec2-52-6-250-55.compute-1.amazonaws.com:RedisProject/src/
org/hu/e63/MovieLensJedis.java
```

### Copy required JARS

Required Jar - jedis-2.4.2.jar, commons-pool2-2.0.jar

```
scp -i HadoopEC2Cluster.pem  ../FinalProject/Redis/jedis-2.4.2.jar ../FinalProject/
Redis/commons-pool2-2.0.jar
ubuntu@ec2-52-6-250-55.compute-1.amazonaws.com:RedisProject/jars
```

### Compile

```
javac -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes -d classes
src/org/hu/e63/MovieLensJedis.java
```

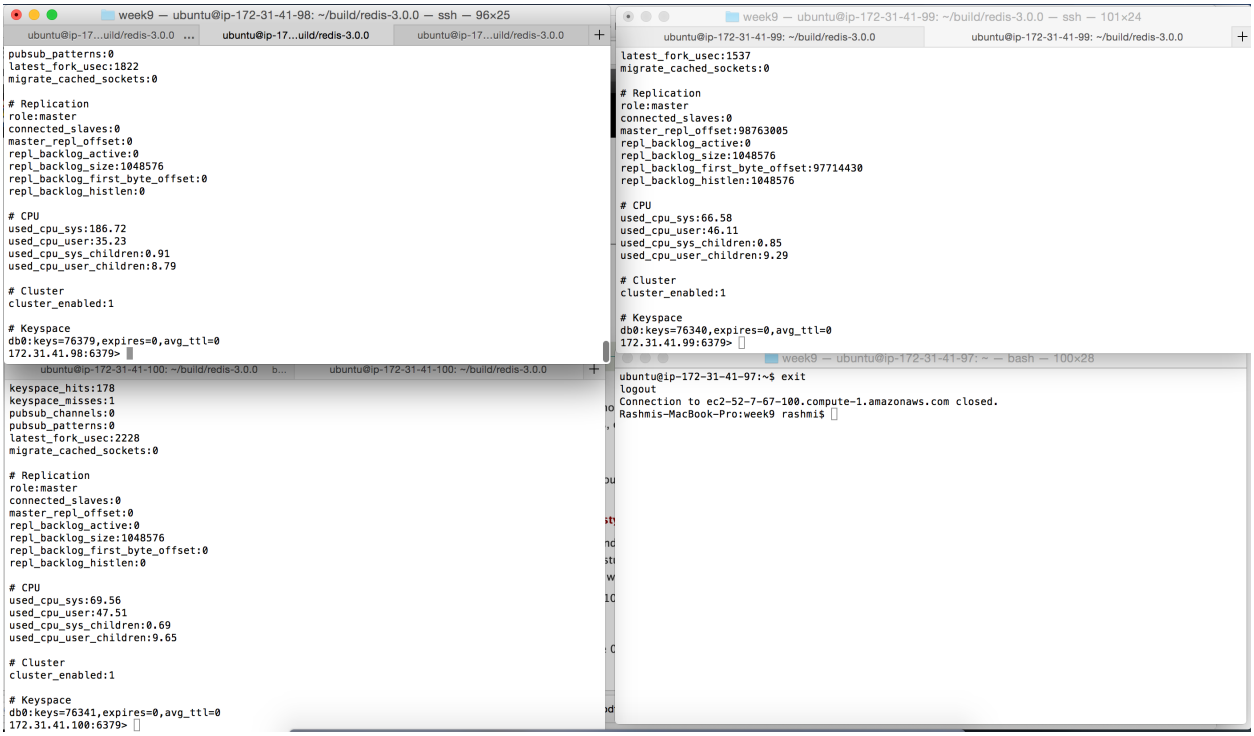
### Run

```
java -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes
org.hu.e63.MovieLensJedis
```

After copying over and compiling my directory :

```
ubuntu@ip-172-31-59-103:~$ tree RedisProject/
RedisProject/
├── classes
|   └── org
|       └── hu
|           └── e63
|               └── MovieLensJedis.class
├── input
|   └── ratings.csv
├── jars
|   ├── commons-pool2-2.0.jar
|   └── jedis-2.4.2.jar
└── src
    └── org
        └── hu
            └── e63
                └── MovieLensJedis.java

10 directories, 5 files
ubuntu@ip-172-31-59-103:~$
```

Following screenshot shows about 230,000 keys on Redis Cluster (about 76000 in each node)



# MovieLensJedisQuery

Now we write a program to query above stored data:

```
package org.hu.e63;

import java.util.HashSet;
import java.util.Set;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.JedisCluster;

public class MovieLensJedisQuery {
	public static void main(String[] args) {

		Set<HostAndPort> jedisClusterNodes = new HashSet<HostAndPort>();
		//Jedis Cluster will attempt to discover cluster nodes automatically
		jedisClusterNodes.add(new HostAndPort("172.31.2.131", 6379));
		JedisCluster jc = new JedisCluster(jedisClusterNodes);
		try {
			System.out.println(jc.smembers("u:1:m"));
		}
	  catch(Exception e)
	     {
		System.out.println("ERROR:" + e.getMessage());
	     }
	}
}
```

### Copy java and jar files

Remember to change the IP address to connect in .JAVA file

### Copy JAVA source file

```
scp -i HadoopEC2Cluster.pem ../../workspace/RedisProject/src/org/hu/e63/
MovieLensJedisQuery.java ubuntu@ec2-52-0-211-157.compute-1.amazonaws.com:RedisProject/
src/org/hu/e63/MovieLensJedisQuery.java
```

### Copy required JARS

Required Jar - jedis-2.4.2.jar, commons-pool2-2.0.jar

```
scp -i HadoopEC2Cluster.pem  ../FinalProject/Redis/jedis-2.4.2.jar ../FinalProject/
Redis/commons-pool2-2.0.jar
ubuntu@ec2-52-0-211-157.compute-1.amazonaws.com:RedisProject/jars
```

### Compile

```
javac -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes -d classes
src/org/hu/e63/MovieLensJedisQuery.java
```

**Run**

```
java -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes
org.hu.e63.MovieLensJedisQuery
[1895, 2002, 1407, 2003, 1917, 1918, 1876, 1882, 1911, 1982, 1909, 1717, 1983, 1387,
253, 2001, 2000]
```

After copying over and compiling my directory :

```
ubuntu@ip-172-31-2-131:~/RedisProject$ tree
.
├── classes
|   └── org
|       └── hu
|           └── e63
|               ├── MovieLensJedis.class
|               └── MovieLensJedisQuery.class
├── input
|   └── ratings.csv
├── jars
|   ├── commons-pool2-2.0.jar
|   └── jedis-2.4.2.jar
└── src
    └── org
        └── hu
            └── e63
                ├── MovieLensJedis.java
                └── MovieLensJedisQuery.java

10 directories, 7 files
```

To retrieve **User 706 has rated which movies?** run the following on Redis Client

```
172.31.41.100:6379> smembers u:706:m
 1) "170"
 2) "355"
 3) "1129"
 4) "1339"
 5) "1381"
 6) "1892"
 7) "2454"
 8) "2605"
 9) "2724"
10) "2763"
11) "2881"
```

```
12) "3476"
13) "3686"
14) "3798"
15) "3827"
172.31.41.100:6379>
```

# Mass Insertion Protocol (Pipeline)

(Load 21M records on Redis using Mass Insertion Protocol)

Do the following on a node that is not part of Cluster or your Virtual Machine (Redis Cluster doesn't support pipeline)

**Install Redis and Java**

**Install JAVA**

```
sudo apt-get install openjdk-7-jdk
sudo apt-get install unzip
```

**Create directories on RedisClusterNode1:**

```
mkdir -p RedisProject/src/org/hu/e63/
cd RedisProject; mkdir classes; mkdir input; mkdir jars
```

**Download and unzip latest MovieLens Dataset (21million ratings and 132 MB zip file)**

```
wget http://files.grouplens.org/datasets/movielens/ml-latest.zip
unzip ml-latest.zip
rm ml-latest.zip
mv ml-latest/ratings.csv input/ratings.csv
rm -R ml-latest/
```

**MovieLens - ratings.csv**

```
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv  | head
userId,movieId,rating,timestamp
1,253,3.0,900660748
1,1387,3.0,900660748
1,1407,5.0,900660871
1,1717,5.0,900660871
1,1876,3.0,900660543
1,1882,3.0,900660584
1,1895,4.0,900660518
1,1909,3.0,900660468
1,1911,3.0,900660543
ubuntu@ip-172-31-41-98:~/RedisProject/input$

ubuntu@ip-172-31-41-98:~/RedisProject/input$ ls -l
total 554976
-rw-r--r-- 1 ubuntu ubuntu 568287348 May  7 12:35 ratings.csv
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv  | wc -l
21063129
ubuntu@ip-172-31-41-98:~/RedisProject/input$
```

**Now We will try to add 21M movie ratings on to Node1 (not in cluster mode)**

For eg:

```
HMSET u:1:m:253 RATING 3.0 TIMESTAMP 900660748
HMSET u:1:m:1387 RATING 3.0 TIMESTAMP 900660748
```

(I wrote the code on Eclipse on my macbook - then copied over the JAVA, JAR and input files to EC2 instance)

### Source Code (MovieLens21M.java)

```
package org.hu.e63;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.util.Scanner;

public class MovieLens21M {
       public static void main(String[] args) {
       try{
           Scanner s = new Scanner(new File("input/ratings.csv"));
           File fileDir = new File("outputfile1.txt");
        Writer out = new BufferedWriter(new OutputStreamWriter(
                    new FileOutputStream(fileDir), "UTF8"));
        String resp = "";
        String s1 = s.nextLine(); //Skip first line
        while (s.hasNextLine()){
             s1 = s.nextLine();
             String[] spl = s1.split(",");
             int ln = 0;

             ln = 5 + spl[0].length()+spl[1].length();
             resp = "*" + 6 + "\r\n" +
                                  "$5\r\n" +
                                  "HMSET\r\n" +
                                  "$" + ln + "\r\n" +
                                  "u:" + spl[0] + ":m:" + spl[1] + "\r\n" +
                                  "$6\r\n"+
                                  "RATING\r\n"+
                                  "$"+spl[2].length()+"\r\n"+
                                  spl[2]+"\r\n"+
                                  "$9\r\n" +
                                  "TIMESTAMP\r\n" +
                                  "$" + spl[3].length()+ "\r\n" +
                                  spl[3] + "\r\n"
                                  ;
             out.append(resp);

        }

           s.close();
           out.flush();
        out.close();
       }
       catch(Exception e)
    {
       System.out.println("bye" + e.getMessage());
    }

}
}
```

### Copy JAVA source file

From the terminal run the following command

```
scp -i HadoopEC2Cluster.pem ../../workspace/RedisProject/src/org/hu/e63/
MovieLens21M.java ubuntu@ec2-52-7-171-102.compute-1.amazonaws.com:RedisProject/src/
org/hu/e63/MovieLens21M.java
```

**Copy required JARS**
(Required Jar - jedis-2.4.2.jar, commons-pool2-2.0.jar)
From the terminal run the following command

```
scp -i HadoopEC2Cluster.pem  ../FinalProject/Redis/jedis-2.4.2.jar ../FinalProject/
Redis/commons-pool2-2.0.jar
ubuntu@ec2-52-7-171-102.compute-1.amazonaws.com:RedisProject/jars
```

**Compile** (on RedisClusterNode1)

```
javac -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes -d classes
src/org/hu/e63/MovieLens21M.java
```

**Run** (on RedisClusterNode1)

```
java -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes
org.hu.e63.MovieLens21M
```

**Directory Structure** (After Run my directory looks like this:)

```
ubuntu@ip-172-31-12-238:~/RedisProject$ tree
.
├── classes
│   └── org
│       └── hu
│           └── e63
│               └── MovieLens21M.class
├── input
│   └── ratings.csv
├── jars
│   ├── commons-pool2-2.0.jar
│   └── jedis-2.4.2.jar
├── outputfile1.txt
└── src
    └── org
        └── hu
            └── e63
                └── MovieLens21M.java

10 directories, 6 files
ubuntu@ip-172-31-12-238:~/RedisProject$
```

**Load 21M records on Redis:**

```
cd; cd build/redis-3.0.0/
cat ../../RedisProject/outputfile1.txt | src/redis-cli -h 172.31.12.238 --pipe
```

```
ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$ date;cat ../../RedisProject/
outputfile1.txt | src/redis-cli -h 172.31.12.238 --pipe;date
Mon May 11 15:02:59 UTC 2015
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 21063128
Mon May 11 15:03:59 UTC 2015
ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$
```

**Note that It took 1 min. to load 21M keys.**

**Verification**

```
ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$ src/redis-cli -h 172.31.12.238
172.31.12.238:6379> HMGET u:1:m:253 RATING
1) "3.0"
172.31.12.238:6379> HMGET u:1:m:253 TIMESTAMP
1) "900660748"
172.31.12.238:6379> HGETALL u:1:m:253
1) "RATING"
2) "3.0"
3) "TIMESTAMP"
4) "900660748"
172.31.12.238:6379>
```

Partial output of info command on redis-cli

```
# Keyspace
db0:keys=21063128,expires=0,avg_ttl=0
```

# YouTube Video Links

Short Video
https://www.youtube.com/watch?v=eYFnniP-ELk

Full Video
https://www.youtube.com/watch?v=fk3_5U3Tl9c

Redis Cluster Installation (Extra Video)
https://www.youtube.com/watch?v=s4YpCA2Y_-Q

# References


http://ilyabylich.svbtle.com/redis-cluster-quick-overview

http://redis.io/topics/cluster-tutorial

http://redis.io/topics/mass-insert

http://bencane.com/2013/11/12/installing-redis-and-setting-up-master-slave-replication/

http://redis.io/topics/replication

https://www.digitalocean.com/community/tutorials/how-to-install-ruby-on-rails-on-ubuntu-12-04-lts-precise-pangolin-with-rvm