ASSIGNMENT 2: CASSANDRA DATABASE AND LOAD BALANCING ON CLOUD

a. TASK DESCRIPTION

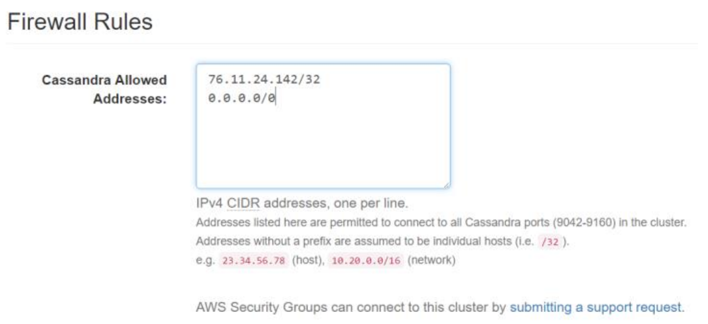
- ✓ To create an application to query data from a Cassandra database uploaded on the cloud.
- ✓ To check the response time taken for the queries formed based on the application scenario.
- ✓ To convert the application created into a Web Service application and to upload it in the cloud web service using Elastic beanstalk.
- ✓ To create another web service replicating the previous one.
- ✓ Creating a load balancer to perform load balancing in both of the created web services for the application created which also hits Cassandra on the cloud.
- ✓ Using JMeter to test the results between the load balancer and the instance.
- ✓ Decreasing the number of instances on the load balancer and doing a test on JMeter
- ✓ Another test on JMeter by stopping the load balancer.

Application description

This application is created to fetch crime records based on various scenarios like crime type, hour, year-month and a combined query for all the mentioned before.

b. CASSANDRA DATABASE DESIGN

Creating Cassandra Cluster

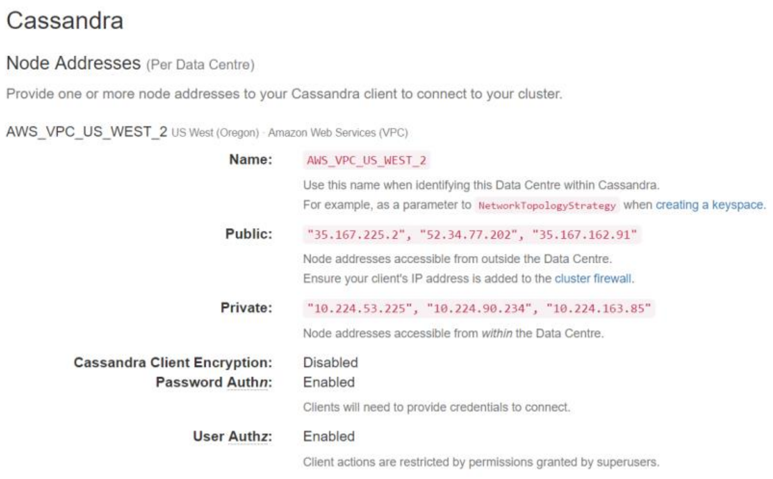


Firewall Rules

Cassandra Allowed Addresses:
```
76.11.24.142/32
0.0.0.0/0
```

IPv4 CIDR addresses, one per line.
Addresses listed here are permitted to connect to all Cassandra ports (9042-9160) in the cluster.
Addresses without a prefix are assumed to be individual hosts (i.e. /32).
e.g. 23.34.56.78 (host), 10.20.0.0/16 (network)

AWS Security Groups can connect to this cluster by submitting a support request.



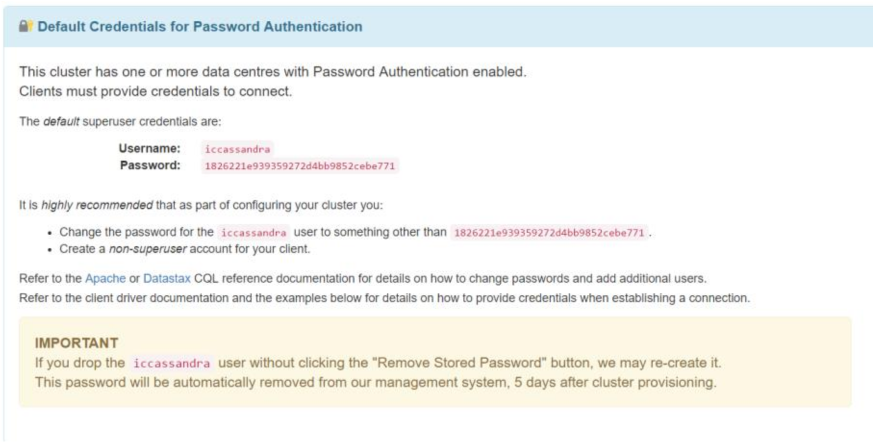
Cassandra

Node Addresses (Per Data Centre)

Provide one or more node addresses to your Cassandra client to connect to your cluster.

AWS_VPC_US_WEST_2 US West (Oregon) · Amazon Web Services (VPC)

Name:	AWS_VPC_US_WEST_2 Use this name when identifying this Data Centre within Cassandra. For example, as a parameter to NetworkTopologyStrategy when creating a keyspace.
Public:	"35.167.225.2", "52.34.77.202", "35.167.162.91" Node addresses accessible from outside the Data Centre. Ensure your client's IP address is added to the cluster firewall.
Private:	"10.224.53.225", "10.224.90.234", "10.224.163.85" Node addresses accessible from *within* the Data Centre.
Cassandra Client Encryption:	Disabled
Password Authn:	Enabled Clients will need to provide credentials to connect.
User Authz:	Enabled Client actions are restricted by permissions granted by superusers.



Default Credentials for Password Authentication

This cluster has one or more data centres with Password Authentication enabled.
Clients must provide credentials to connect.

The *default* superuser credentials are:

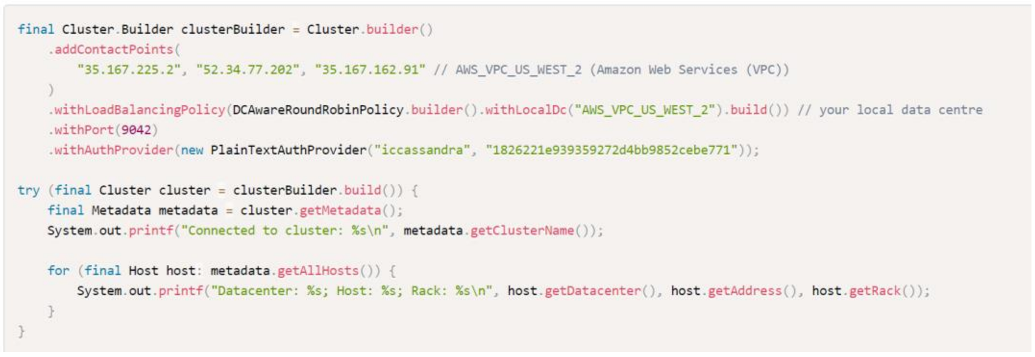
Username:	`iccassandra`
Password:	`1826221e939359272d4bb9852cebe771`

It is *highly recommended* that as part of configuring your cluster you:

- Change the password for the `iccassandra` user to something other than `1826221e939359272d4bb9852cebe771`.
- Create a *non-superuser* account for your client.

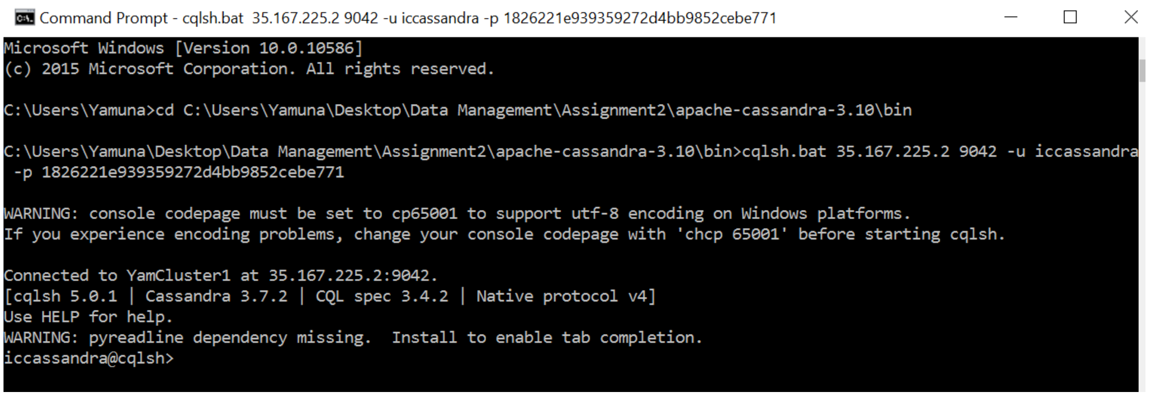
Refer to the Apache or Datastax CQL reference documentation for details on how to change passwords and add additional users.
Refer to the client driver documentation and the examples below for details on how to provide credentials when establishing a connection.

IMPORTANT
If you drop the `iccassandra` user without clicking the "Remove Stored Password" button, we may re-create it.
This password will be automatically removed from our management system, 5 days after cluster provisioning.



```
final Cluster.Builder clusterBuilder = Cluster.builder()
    .addContactPoints(
        "35.167.225.2", "52.34.77.202", "35.167.162.91" // AWS_VPC_US_WEST_2 (Amazon Web Services (VPC))
    )
    .withLoadBalancingPolicy(DCAwareRoundRobinPolicy.builder().withLocalDc("AWS_VPC_US_WEST_2").build()) // your local data centre
    .withPort(9042)
    .withAuthProvider(new PlainTextAuthProvider("iccassandra", "1826221e939359272d4bb9852cebe771"));

try (final Cluster cluster = clusterBuilder.build()) {
    final Metadata metadata = cluster.getMetadata();
    System.out.printf("Connected to cluster: %s\n", metadata.getClusterName());

    for (final Host host: metadata.getAllHosts()) {
        System.out.printf("Datacenter: %s; Host: %s; Rack: %s\n", host.getDatacenter(), host.getAddress(), host.getRack());
    }
}
```

Connecting to Cassandra Cluster



Command Prompt - cqlsh.bat 35.167.225.2 9042 -u iccassandra -p 1826221e939359272d4bb9852cebe771

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Yamuna>cd C:\Users\Yamuna\Desktop\Data Management\Assignment2\apache-cassandra-3.10\bin

C:\Users\Yamuna\Desktop\Data Management\Assignment2\apache-cassandra-3.10\bin>cqlsh.bat 35.167.225.2 9042 -u iccassandra
 -p 1826221e939359272d4bb9852cebe771

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to YamCluster1 at 35.167.225.2:9042.
[cqlsh 5.0.1 | Cassandra 3.7.2 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
iccassandra@cqlsh>
```

Create Table Schema

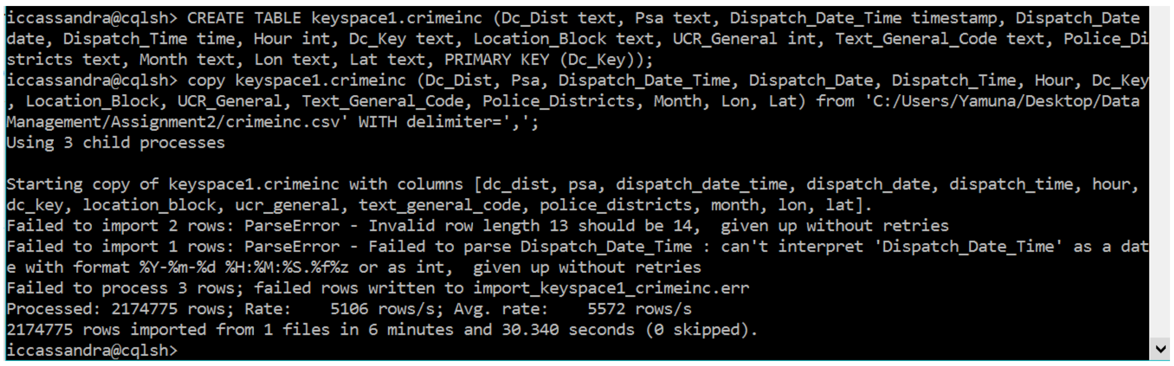
```
CREATE TABLE keyspace1.crimeinc (
        Dc_Dist text,
        Psa text,
        Dispatch_Date_Time timestamp,
        Dispatch_Date date,
        Dispatch_Time time,
        Hour int,
        Dc_Key text,
        Location_Block text,
        UCR_General int,
        Text_General_Code text,
        Police_Districts text,
        Month text,
        Lon text,
        Lat text,
        PRIMARY KEY (Dc_Key)
);
```

Database Description

Field Name	Description
DC_Dist	This is a two character field that names the District boundary.
DC_Key	A unique identifier of the crime that consists of Year |District | Unique ID.
Dispatch_Date_Time	Date and time that the officer was dispatched to the scene.
Hour	Generalized hour of the dispatched time.
Location_Block	Location of crime generalized by street block.
Sector	Field that names the Police Service Area boundary.
Text_General_Code	Generalized text for the crime code.
UCR_General	Rounded crime code, i.e. 614 to 600.

Copy command

copy keyspace1.crimeinc (Dc_Dist, Psa, Dispatch_Date_Time, Dispatch_Date, Dispatch_Time, Hour, Dc_Key, Location_Block, UCR_General, Text_General_Code, Police_Districts, Month, Lon, Lat) from 'C:/Users/Yamuna/Desktop/Data Management/Assignment2/crimeinc.csv' WITH delimiter=',';



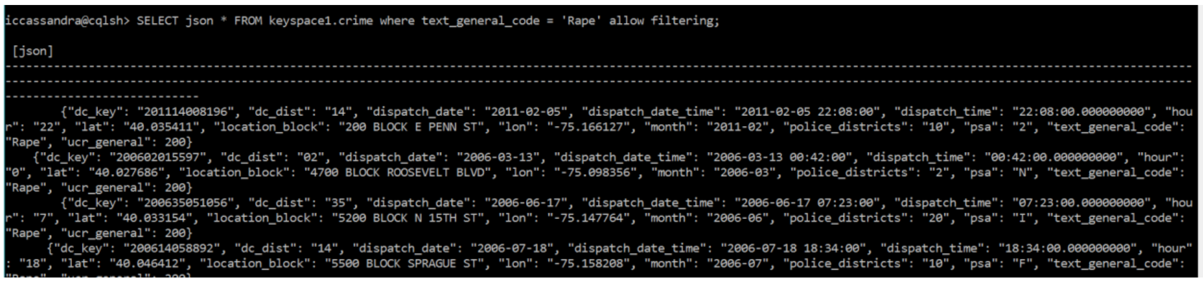
```
iccassandra@cqlsh> CREATE TABLE keyspace1.crimeinc (Dc_Dist text, Psa text, Dispatch_Date_Time timestamp, Dispatch_Date
date, Dispatch_Time time, Hour int, Dc_Key text, Location_Block text, UCR_General int, Text_General_Code text, Police_Di
stricts text, Month text, Lon text, Lat text, PRIMARY KEY (Dc_Key));
iccassandra@cqlsh> copy keyspace1.crimeinc (Dc_Dist, Psa, Dispatch_Date_Time, Dispatch_Date, Dispatch_Time, Hour, Dc_Key
, Location_Block, UCR_General, Text_General_Code, Police_Districts, Month, Lon, Lat) from 'C:/Users/Yamuna/Desktop/Data
Management/Assignment2/crimeinc.csv' WITH delimiter=',';
Using 3 child processes

Starting copy of keyspace1.crimeinc with columns [dc_dist, psa, dispatch_date_time, dispatch_date, dispatch_time, hour,
dc_key, location_block, ucr_general, text_general_code, police_districts, month, lon, lat].
Failed to import 2 rows: ParseError - Invalid row length 13 should be 14,  given up without retries
Failed to import 1 rows: ParseError - Failed to parse Dispatch_Date_Time : can't interpret 'Dispatch_Date_Time' as a dat
e with format %Y-%m-%d %H:%M:%S.%f%z or as int,  given up without retries
Failed to process 3 rows; failed rows written to import_keyspace1_crimeinc.err
Processed: 2174775 rows; Rate:    5106 rows/s; Avg. rate:    5572 rows/s
2174775 rows imported from 1 files in 6 minutes and 30.340 seconds (0 skipped).
iccassandra@cqlsh>
```

Query 1

This query is used to fetch the crime data based on the crime type. Execution time is 1.2 seconds approximately

SELECT json * FROM keyspace1.crime where text_general_code = 'Rape' allow filtering;

Sample JSON format output.

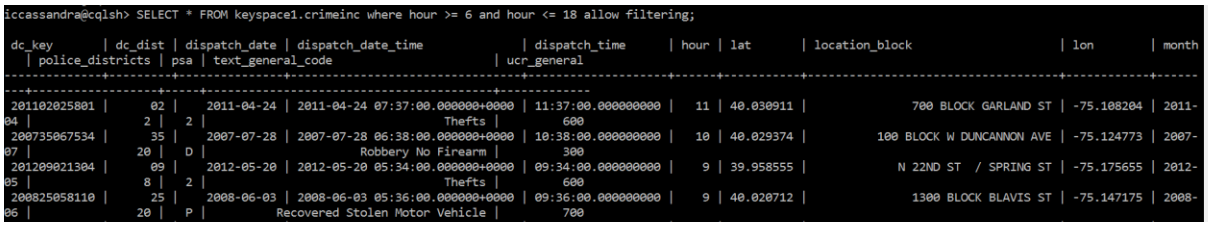


```
iccassandra@cqlsh> SELECT json * FROM keyspace1.crime where text_general_code = 'Rape' allow filtering;

 [json]
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------
       {"dc_key": "201114008196", "dc_dist": "14", "dispatch_date": "2011-02-05", "dispatch_date_time": "2011-02-05 22:08:00", "dispatch_time": "22:08:00.000000000", "hou
r": "22", "lat": "40.035411", "location_block": "200 BLOCK E PENN ST", "lon": "-75.166127", "month": "2011-02", "police_districts": "10", "psa": "2", "text_general_code":
"Rape", "ucr_general": 200}
   {"dc_key": "200602015597", "dc_dist": "02", "dispatch_date": "2006-03-13", "dispatch_date_time": "2006-03-13 00:42:00", "dispatch_time": "00:42:00.000000000", "hour":
"0", "lat": "40.027686", "location_block": "4700 BLOCK ROOSEVELT BLVD", "lon": "-75.098356", "month": "2006-03", "police_districts": "2", "psa": "N", "text_general_code":
"Rape", "ucr_general": 200}
       {"dc_key": "200635051056", "dc_dist": "35", "dispatch_date": "2006-06-17", "dispatch_date_time": "2006-06-17 07:23:00", "dispatch_time": "07:23:00.000000000", "hou
r": "7", "lat": "40.033154", "location_block": "5200 BLOCK N 15TH ST", "lon": "-75.147764", "month": "2006-06", "police_districts": "20", "psa": "I", "text_general_code":
"Rape", "ucr_general": 200}
     {"dc_key": "200614058892", "dc_dist": "14", "dispatch_date": "2006-07-18", "dispatch_date_time": "2006-07-18 18:34:00", "dispatch_time": "18:34:00.000000000", "hour"
: "18", "lat": "40.046412", "location_block": "5500 BLOCK SPRAGUE ST", "lon": "-75.158208", "month": "2006-07", "police_districts": "10", "psa": "F", "text_general_code":
```

Query 2

This query is used to fetch crime records based on the hour of a day. In the below example the crime happening before 6 AM and after 6 PM is displayed. That is the crimes that takes place in dark time of a day. Execution time for this query is 1.6 seconds approximately.

SELECT * FROM keyspace1.crimeinc where hour >= 6 and hour <= 18 allow filtering;

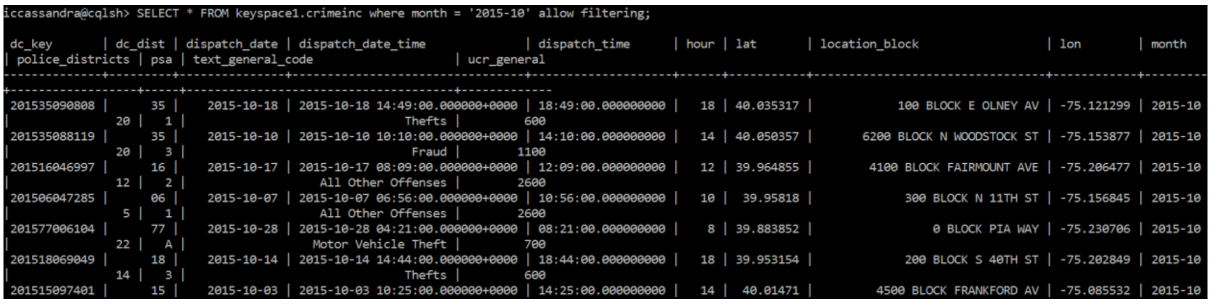


```
iccassandra@cqlsh> SELECT * FROM keyspace1.crimeinc where hour >= 6 and hour <= 18 allow filtering;

 dc_key       | dc_dist | dispatch_date | dispatch_date_time              | dispatch_time      | hour | lat       | location_block                   | lon        | month
   | police_districts | psa | text_general_code                  | ucr_general
-------------+---------+---------------+---------------------------------+--------------------+------+-----------+----------------------------------+------------+------
---+------------------+-----+------------------------------------+------------
 201102025801 |      02 |    2011-04-24 | 2011-04-24 07:37:00.000000+0000 | 11:37:00.000000000 |   11 | 40.030911 |              700 BLOCK GARLAND ST | -75.108204 | 2011-
04 |                2 |   2 |                             Thefts |         600
 200735067534 |      35 |    2007-07-28 | 2007-07-28 06:38:00.000000+0000 | 10:38:00.000000000 |   10 | 40.029374 |         100 BLOCK W DUNCANNON AVE | -75.124773 | 2007-
07 |               20 |   D |                 Robbery No Firearm |         300
 201209021304 |      09 |    2012-05-20 | 2012-05-20 05:34:00.000000+0000 | 09:34:00.000000000 |    9 | 39.958555 |           N 22ND ST  / SPRING ST | -75.175655 | 2012-
05 |                8 |   2 |                             Thefts |         600
 200825058110 |      25 |    2008-06-03 | 2008-06-03 05:36:00.000000+0000 | 09:36:00.000000000 |    9 | 40.020712 |              1300 BLOCK BLAVIS ST | -75.147175 | 2008-
06 |               20 |   P |          Recovered Stolen Motor Vehicle |         700
```

Query 3

This query is used to fetch the crime records based on the year-month filtering. The execution time for this query is 4 seconds approximately.

SELECT * FROM keyspace1.crimeinc where month = '2015-10' allow filtering;

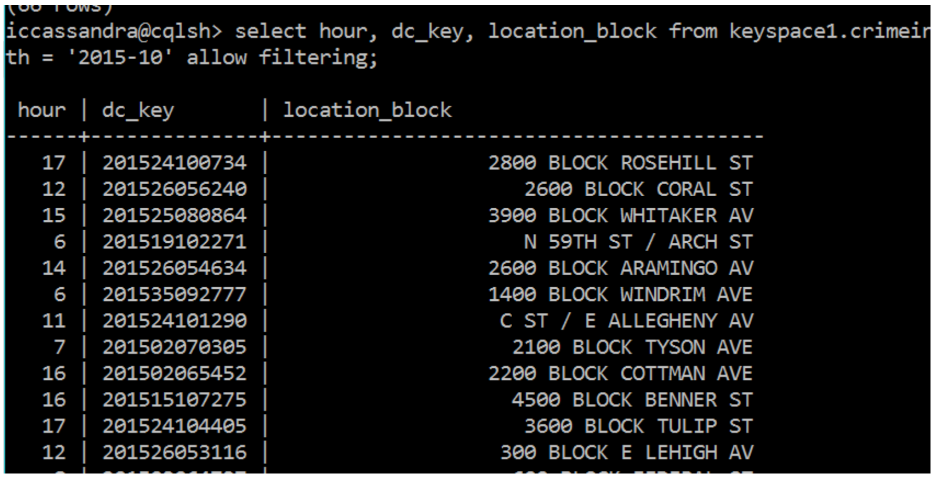


```
iccassandra@cqlsh> SELECT * FROM keyspace1.crimeinc where month = '2015-10' allow filtering;

 dc_key       | dc_dist | dispatch_date | dispatch_date_time              | dispatch_time      | hour | lat       | location_block                   | lon        | month
| police_districts | psa | text_general_code                  | ucr_general
-------------+---------+---------------+---------------------------------+--------------------+------+-----------+----------------------------------+------------+---------
+------------------+-----+------------------------------------+------------
 201535090808 |      35 |    2015-10-18 | 2015-10-18 14:49:00.000000+0000 | 18:49:00.000000000 |   18 | 40.035317 |          100 BLOCK E OLNEY AV | -75.121299 | 2015-10
|               20 |   1 |                             Thefts |         600
 201535088119 |      35 |    2015-10-10 | 2015-10-10 10:10:00.000000+0000 | 14:10:00.000000000 |   14 | 40.050357 |       6200 BLOCK N WOODSTOCK ST | -75.153877 | 2015-10
|               20 |   3 |                              Fraud |        1100
 201516046997 |      16 |    2015-10-17 | 2015-10-17 08:09:00.000000+0000 | 12:09:00.000000000 |   12 | 39.964855 |        4100 BLOCK FAIRMOUNT AVE | -75.206477 | 2015-10
|               12 |   2 |                 All Other Offenses |        2600
 201506047285 |      06 |    2015-10-07 | 2015-10-07 06:56:00.000000+0000 | 10:56:00.000000000 |   10 |  39.95818 |          300 BLOCK N 11TH ST | -75.156845 | 2015-10
|                5 |   1 |                 All Other Offenses |        2600
 201577006104 |      77 |    2015-10-28 | 2015-10-28 04:21:00.000000+0000 | 08:21:00.000000000 |    8 | 39.883852 |              0 BLOCK PIA WAY | -75.230706 | 2015-10
|               22 |   A |                Motor Vehicle Theft |         700
 201518069049 |      18 |    2015-10-14 | 2015-10-14 14:44:00.000000+0000 | 18:44:00.000000000 |   18 | 39.953154 |          200 BLOCK S 40TH ST | -75.202849 | 2015-10
|               14 |   3 |                             Thefts |         600
 201515097401 |      15 |    2015-10-03 | 2015-10-03 10:25:00.000000+0000 | 14:25:00.000000000 |   14 |  40.01471 |        4500 BLOCK FRANKFORD AV | -75.085532 | 2015-10
```

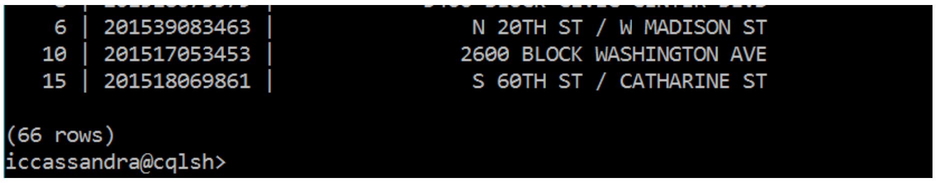
Query 4

This query is fetching crime records based on crime type, hour of day and year month. This is basically the combined result of all the above options. The execution time for this query is 7.5 seconds approximately.

select hour, dc_key, location_block from keyspace1.crimeinc where text_general_code = 'DRIVING UNDER THE INFLUENCE' and hour >= 6 and hour <= 18 and month = '2015-10' allow filtering;

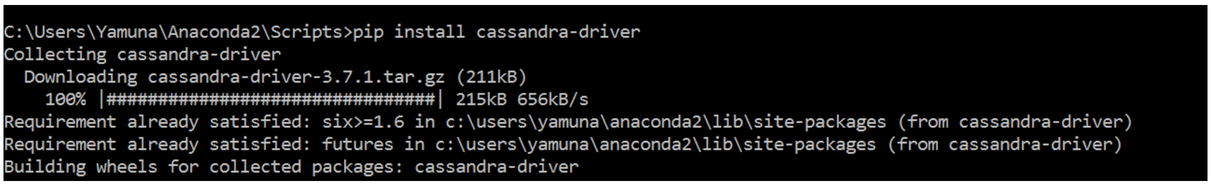


```
iccassandra@cqlsh> select hour, dc_key, location_block from keyspace1.crimeir
th = '2015-10' allow filtering;

 hour | dc_key       | location_block
------+--------------+---------------------------------------
   17 | 201524100734 |                    2800 BLOCK ROSEHILL ST
   12 | 201526056240 |                       2600 BLOCK CORAL ST
   15 | 201525080864 |                    3900 BLOCK WHITAKER AV
    6 | 201519102271 |                       N 59TH ST / ARCH ST
   14 | 201526054634 |                    2600 BLOCK ARAMINGO AV
    6 | 201535092777 |                    1400 BLOCK WINDRIM AVE
   11 | 201524101290 |                     C ST / E ALLEGHENY AV
    7 | 201502070305 |                      2100 BLOCK TYSON AVE
   16 | 201502065452 |                    2200 BLOCK COTTMAN AVE
   16 | 201515107275 |                      4500 BLOCK BENNER ST
   17 | 201524104405 |                       3600 BLOCK TULIP ST
   12 | 201526053116 |                      300 BLOCK E LEHIGH AV
```



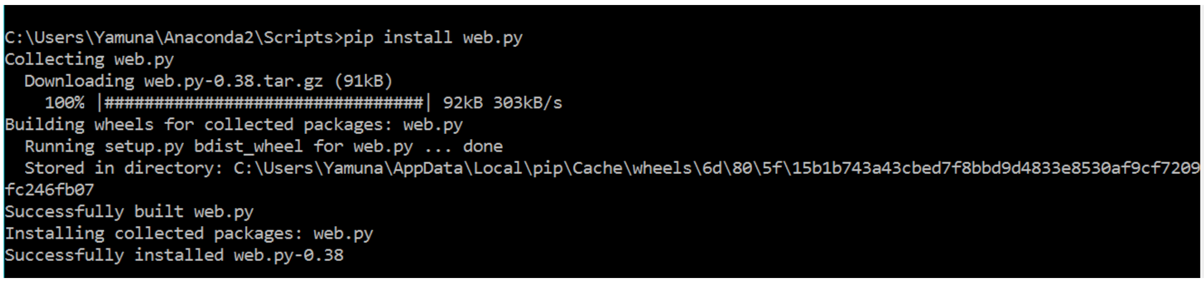
```
    6 | 201539083463 |                    N 20TH ST / W MADISON ST
   10 | 201517053453 |                  2600 BLOCK WASHINGTON AVE
   15 | 201518069861 |                    S 60TH ST / CATHARINE ST

(66 rows)
iccassandra@cqlsh>
```

Installing Cassandra-driver package for Python



```
C:\Users\Yamuna\Anaconda2\Scripts>pip install cassandra-driver
Collecting cassandra-driver
  Downloading cassandra-driver-3.7.1.tar.gz (211kB)
    100% |################################| 215kB 656kB/s
Requirement already satisfied: six>=1.6 in c:\users\yamuna\anaconda2\lib\site-packages (from cassandra-driver)
Requirement already satisfied: futures in c:\users\yamuna\anaconda2\lib\site-packages (from cassandra-driver)
Building wheels for collected packages: cassandra-driver
```

Installing web.py package for python Web Services

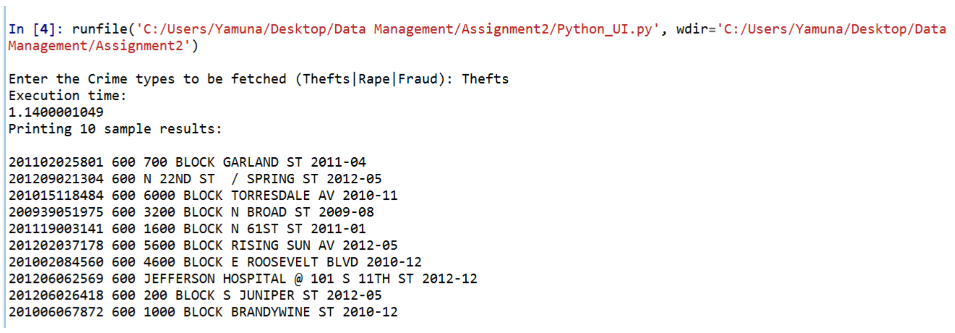


```
C:\Users\Yamuna\Anaconda2\Scripts>pip install web.py
Collecting web.py
  Downloading web.py-0.38.tar.gz (91kB)
    100% |################################| 92kB 303kB/s
Building wheels for collected packages: web.py
  Running setup.py bdist_wheel for web.py ... done
  Stored in directory: C:\Users\Yamuna\AppData\Local\pip\Cache\wheels\6d\80\5f\15b1b743a43cbed7f8bbd9d4833e8530af9cf7209
fc246fb07
Successfully built web.py
Installing collected packages: web.py
Successfully installed web.py-0.38
```

c. APPLICATION QUERIES

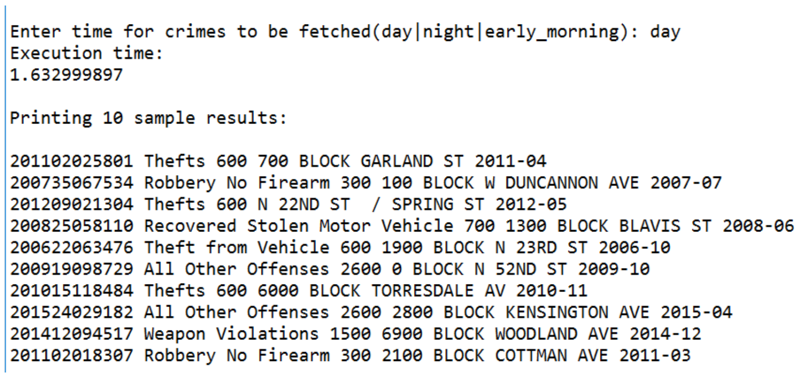
Python UI Output – Code Attached Python_UI.py

Query 1 – Fetching crime records based on Crime type



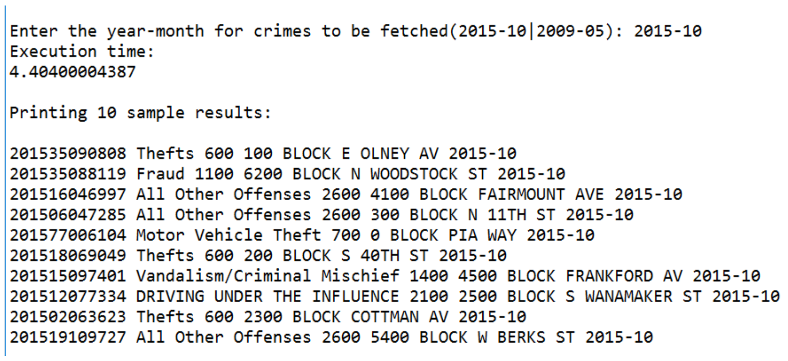
```
In [4]: runfile('C:/Users/Yamuna/Desktop/Data Management/Assignment2/Python_UI.py', wdir='C:/Users/Yamuna/Desktop/Data
Management/Assignment2')

Enter the Crime types to be fetched (Thefts|Rape|Fraud): Thefts
Execution time:
1.1400001049
Printing 10 sample results:

201102025801 600 700 BLOCK GARLAND ST 2011-04
201209021304 600 N 22ND ST  / SPRING ST 2012-05
201015118484 600 6000 BLOCK TORRESDALE AV 2010-11
200939051975 600 3200 BLOCK N BROAD ST 2009-08
201119003141 600 1600 BLOCK N 61ST ST 2011-01
201202037178 600 5600 BLOCK RISING SUN AV 2012-05
201002084560 600 4600 BLOCK E ROOSEVELT BLVD 2010-12
201206062569 600 JEFFERSON HOSPITAL @ 101 S 11TH ST 2012-12
201206026418 600 200 BLOCK S JUNIPER ST 2012-05
201006067872 600 1000 BLOCK BRANDYWINE ST 2010-12
```

Query 2 – Fetching crime records based on hour of day



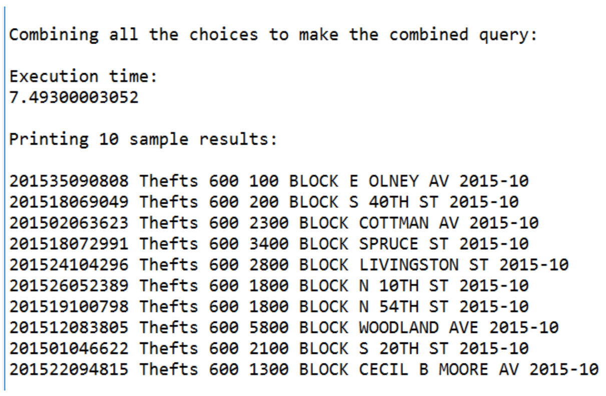
```
Enter time for crimes to be fetched(day|night|early_morning): day
Execution time:
1.632999897

Printing 10 sample results:

201102025801 Thefts 600 700 BLOCK GARLAND ST 2011-04
200735067534 Robbery No Firearm 300 100 BLOCK W DUNCANNON AVE 2007-07
201209021304 Thefts 600 N 22ND ST  / SPRING ST 2012-05
200825058110 Recovered Stolen Motor Vehicle 700 1300 BLOCK BLAVIS ST 2008-06
200622063476 Theft from Vehicle 600 1900 BLOCK N 23RD ST 2006-10
200919098729 All Other Offenses 2600 0 BLOCK N 52ND ST 2009-10
201015118484 Thefts 600 6000 BLOCK TORRESDALE AV 2010-11
201524029182 All Other Offenses 2600 2800 BLOCK KENSINGTON AVE 2015-04
201412094517 Weapon Violations 1500 6900 BLOCK WOODLAND AVE 2014-12
201102018307 Robbery No Firearm 300 2100 BLOCK COTTMAN AVE 2011-03
```

Query 3 – Fetching crime records based on year and month



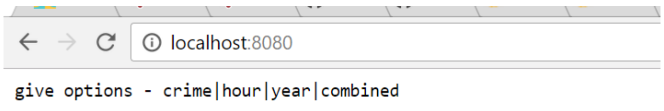
```
Enter the year-month for crimes to be fetched(2015-10|2009-05): 2015-10
Execution time:
4.40400004387

Printing 10 sample results:

201535090808 Thefts 600 100 BLOCK E OLNEY AV 2015-10
201535088119 Fraud 1100 6200 BLOCK N WOODSTOCK ST 2015-10
201516046997 All Other Offenses 2600 4100 BLOCK FAIRMOUNT AVE 2015-10
201506047285 All Other Offenses 2600 300 BLOCK N 11TH ST 2015-10
201577006104 Motor Vehicle Theft 700 0 BLOCK PIA WAY 2015-10
201518069049 Thefts 600 200 BLOCK S 40TH ST 2015-10
201515097401 Vandalism/Criminal Mischief 1400 4500 BLOCK FRANKFORD AV 2015-10
201512077334 DRIVING UNDER THE INFLUENCE 2100 2500 BLOCK S WANAMAKER ST 2015-10
201502063623 Thefts 600 2300 BLOCK COTTMAN AV 2015-10
201519109727 All Other Offenses 2600 5400 BLOCK W BERKS ST 2015-10
```

Query 4 – Fetching crime records based on crime type, hour of day and year month



```
Combining all the choices to make the combined query:

Execution time:
7.49300003052

Printing 10 sample results:

201535090808 Thefts 600 100 BLOCK E OLNEY AV 2015-10
201518069049 Thefts 600 200 BLOCK S 40TH ST 2015-10
201502063623 Thefts 600 2300 BLOCK COTTMAN AV 2015-10
201518072991 Thefts 600 3400 BLOCK SPRUCE ST 2015-10
201524104296 Thefts 600 2800 BLOCK LIVINGSTON ST 2015-10
201526052389 Thefts 600 1800 BLOCK N 10TH ST 2015-10
201519100798 Thefts 600 1800 BLOCK N 54TH ST 2015-10
201512083805 Thefts 600 5800 BLOCK WOODLAND AVE 2015-10
201501046622 Thefts 600 2100 BLOCK S 20TH ST 2015-10
201522094815 Thefts 600 1300 BLOCK CECIL B MOORE AV 2015-10
```

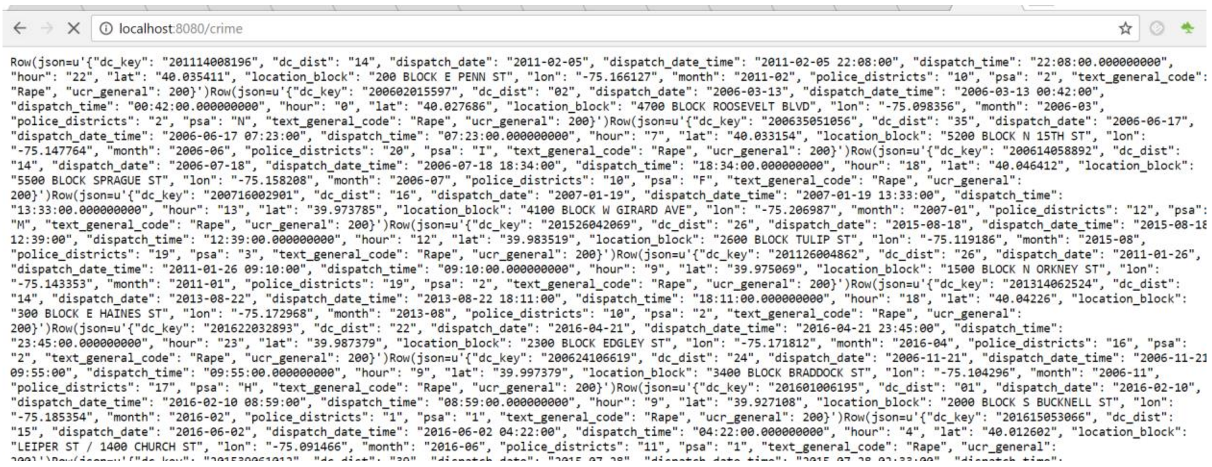
Implementing Web Services using Python – Code Attached Cassandra.py



localhost:8080

```
give options - crime|hour|year|combined
```

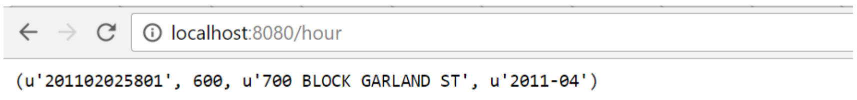
Query 1 – Fetching crime records based on Crime type



localhost:8080/crime

```
Row(json=u'{"dc_key": "201114008196", "dc_dist": "14", "dispatch_date": "2011-02-05", "dispatch_date_time": "2011-02-05 22:08:00", "dispatch_time": "22:08:00.000000000",
"hour": "22", "lat": "40.035411", "location_block": "200 BLOCK E PENN ST", "lon": "-75.166127", "month": "2011-02", "police_districts": "10", "psa": "2", "text_general_code":
"Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "200602015597", "dc_dist": "02", "dispatch_date": "2006-03-13", "dispatch_date_time": "2006-03-13 00:42:00",
"dispatch_time": "00:42:00.000000000", "hour": "0", "lat": "40.027686", "location_block": "4700 BLOCK ROOSEVELT BLVD", "lon": "-75.098356", "month": "2006-03",
"police_districts": "2", "psa": "N", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "200635051056", "dc_dist": "35", "dispatch_date": "2006-06-17",
"dispatch_date_time": "2006-06-17 07:23:00", "dispatch_time": "07:23:00.000000000", "hour": "7", "lat": "40.033154", "location_block": "5200 BLOCK N 15TH ST", "lon":
"-75.147764", "month": "2006-06", "police_districts": "20", "psa": "I", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "200614058892", "dc_dist":
"14", "dispatch_date": "2006-07-18", "dispatch_date_time": "2006-07-18 18:34:00", "dispatch_time": "18:34:00.000000000", "hour": "18", "lat": "40.046412", "location_block":
"5500 BLOCK SPRAGUE ST", "lon": "-75.158208", "month": "2006-07", "police_districts": "10", "psa": "F", "text_general_code": "Rape", "ucr_general":
200}')Row(json=u'{"dc_key": "200716002901", "dc_dist": "16", "dispatch_date": "2007-01-19", "dispatch_date_time": "2007-01-19 13:33:00", "dispatch_time":
"13:33:00.000000000", "hour": "13", "lat": "39.973785", "location_block": "4100 BLOCK W GIRARD AVE", "lon": "-75.206987", "month": "2007-01", "police_districts": "12", "psa":
"M", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "201526042069", "dc_dist": "26", "dispatch_date": "2015-08-18", "dispatch_date_time": "2015-08-18
12:39:00", "dispatch_time": "12:39:00.000000000", "hour": "12", "lat": "39.983519", "location_block": "2600 BLOCK TULIP ST", "lon": "-75.119186", "month": "2015-08",
"police_districts": "19", "psa": "3", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "201126004862", "dc_dist": "26", "dispatch_date": "2011-01-26",
"dispatch_date_time": "2011-01-26 09:10:00", "dispatch_time": "09:10:00.000000000", "hour": "9", "lat": "39.975069", "location_block": "1500 BLOCK N ORKNEY ST", "lon":
"-75.143353", "month": "2011-01", "police_districts": "19", "psa": "2", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "201314062524", "dc_dist":
"14", "dispatch_date": "2013-08-22", "dispatch_date_time": "2013-08-22 18:11:00", "dispatch_time": "18:11:00.000000000", "hour": "18", "lat": "40.04226", "location_block":
"300 BLOCK E HAINES ST", "lon": "-75.172968", "month": "2013-08", "police_districts": "10", "psa": "2", "text_general_code": "Rape", "ucr_general":
200}')Row(json=u'{"dc_key": "201622032893", "dc_dist": "22", "dispatch_date": "2016-04-21", "dispatch_date_time": "2016-04-21 23:45:00", "dispatch_time":
"23:45:00.000000000", "hour": "23", "lat": "39.987379", "location_block": "2300 BLOCK EDGLEY ST", "lon": "-75.171812", "month": "2016-04", "police_districts": "16", "psa":
"2", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "200624106619", "dc_dist": "24", "dispatch_date": "2006-11-21", "dispatch_date_time": "2006-11-21
09:55:00", "dispatch_time": "09:55:00.000000000", "hour": "9", "lat": "39.997379", "location_block": "3400 BLOCK BRADDOCK ST", "lon": "-75.104296", "month": "2006-11",
"police_districts": "17", "psa": "H", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "201601006195", "dc_dist": "01", "dispatch_date": "2016-02-10",
"dispatch_date_time": "2016-02-10 08:59:00", "dispatch_time": "08:59:00.000000000", "hour": "9", "lat": "39.927108", "location_block": "2000 BLOCK S BUCKNELL ST", "lon":
"-75.185354", "month": "2016-02", "police_districts": "1", "psa": "1", "text_general_code": "Rape", "ucr_general": 200}')Row(json=u'{"dc_key": "201615053066", "dc_dist":
"15", "dispatch_date": "2016-06-02", "dispatch_date_time": "2016-06-02 04:22:00", "dispatch_time": "04:22:00.000000000", "hour": "4", "lat": "40.012602", "location_block":
"LEIPER ST / 1400 CHURCH ST", "lon": "-75.091466", "month": "2016-06", "police_districts": "11", "psa": "1", "text_general_code": "Rape", "ucr_general":
```

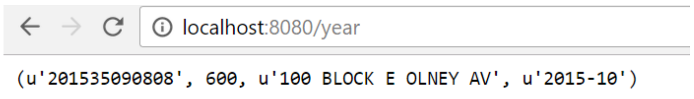
Query 2 – Fetching crime records based on hour of day



localhost:8080/hour

```
(u'201102025801', 600, u'700 BLOCK GARLAND ST', u'2011-04')
```

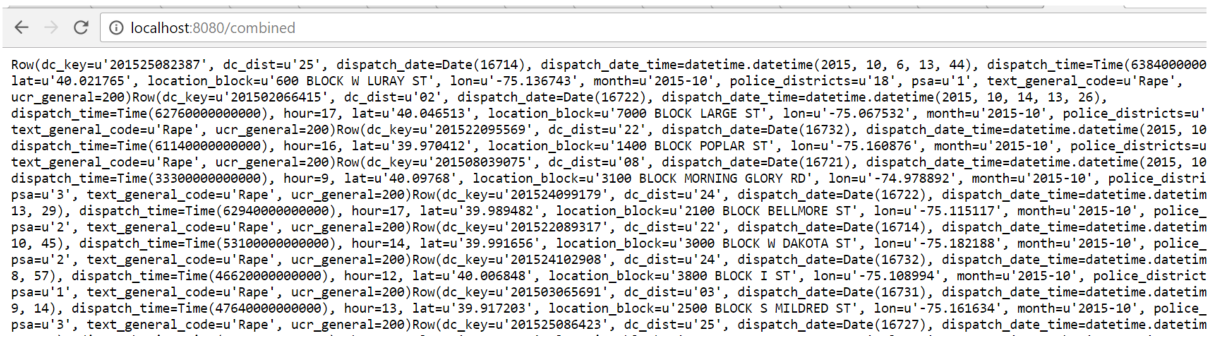
Query 3 – Fetching crime records based on year and month



localhost:8080/year

```
(u'201535090808', 600, u'100 BLOCK E OLNEY AV', u'2015-10')
```

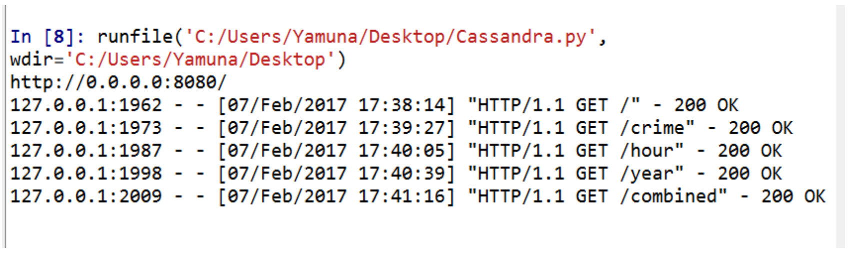
Query 4 – Fetching crime records based on crime type, hour of day and year month



localhost:8080/combined

```
Row(dc_key=u'201525082387', dc_dist=u'25', dispatch_date=Date(16714), dispatch_date_time=datetime.datetime(2015, 10, 6, 13, 44), dispatch_time=Time(6384000000
lat=u'40.021765', location_block=u'600 BLOCK W LURAY ST', lon=u'-75.136743', month=u'2015-10', police_districts=u'18', psa=u'1', text_general_code=u'Rape',
ucr_general=200)Row(dc_key=u'201502066415', dc_dist=u'02', dispatch_date=Date(16722), dispatch_date_time=datetime.datetime(2015, 10, 14, 13, 26),
dispatch_time=Time(62760000000000), hour=17, lat=u'40.046513', location_block=u'7000 BLOCK LARGE ST', lon=u'-75.067532', month=u'2015-10', police_districts=u'
text_general_code=u'Rape', ucr_general=200)Row(dc_key=u'201522095569', dc_dist=u'22', dispatch_date=Date(16732), dispatch_date_time=datetime.datetime(2015, 10
dispatch_time=Time(61140000000000), hour=16, lat=u'39.970412', location_block=u'1400 BLOCK POPLAR ST', lon=u'-75.160876', month=u'2015-10', police_districts=u
text_general_code=u'Rape', ucr_general=200)Row(dc_key=u'201508039075', dc_dist=u'08', dispatch_date=Date(16721), dispatch_date_time=datetime.datetime(2015, 10
dispatch_time=Time(33300000000000), hour=9, lat=u'40.09768', location_block=u'3100 BLOCK MORNING GLORY RD', lon=u'-74.978892', month=u'2015-10', police_distri
psa=u'3', text_general_code=u'Rape', ucr_general=200)Row(dc_key=u'201524099179', dc_dist=u'24', dispatch_date=Date(16722), dispatch_date_time=datetime.datetim
13, 29), dispatch_time=Time(62940000000000), hour=17, lat=u'39.989482', location_block=u'2100 BLOCK BELLMORE ST', lon=u'-75.115117', month=u'2015-10', police_
psa=u'2', text_general_code=u'Rape', ucr_general=200)Row(dc_key=u'201522089317', dc_dist=u'22', dispatch_date=Date(16714), dispatch_date_time=datetime.datetim
10, 45), dispatch_time=Time(53100000000000), hour=14, lat=u'39.991656', location_block=u'3000 BLOCK W DAKOTA ST', lon=u'-75.182188', month=u'2015-10', police_
psa=u'2', text_general_code=u'Rape', ucr_general=200)Row(dc_key=u'201524102908', dc_dist=u'24', dispatch_date=Date(16732), dispatch_date_time=datetime.datetim
8, 57), dispatch_time=Time(46620000000000), hour=12, lat=u'40.006848', location_block=u'3800 BLOCK I ST', lon=u'-75.108994', month=u'2015-10', police_district
psa=u'1', text_general_code=u'Rape', ucr_general=200)Row(dc_key=u'201503065691', dc_dist=u'03', dispatch_date=Date(16731), dispatch_date_time=datetime.datetim
9, 14), dispatch_time=Time(47640000000000), hour=13, lat=u'39.917203', location_block=u'2500 BLOCK S MILDRED ST', lon=u'-75.161634', month=u'2015-10', police_
psa=u'3', text_general_code=u'Rape', ucr_general=200)Row(dc_key=u'201525086423', dc_dist=u'25', dispatch_date=Date(16727), dispatch_date_time=datetime.datetim
```

HTTP response code outputs captured for all 4 queries

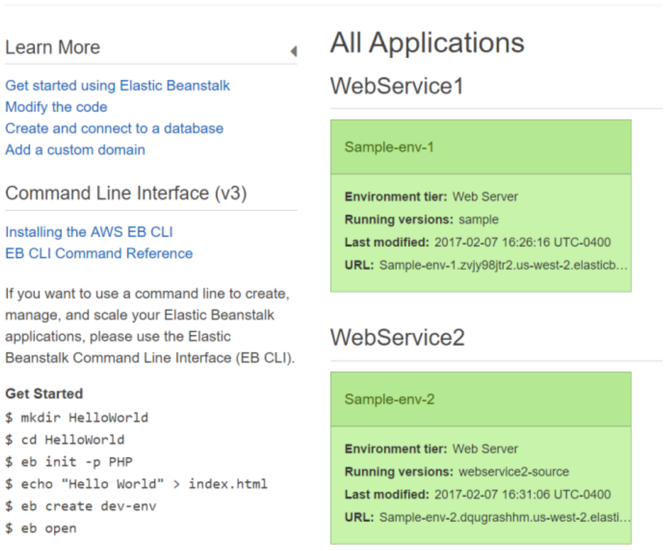


```
In [8]: runfile('C:/Users/Yamuna/Desktop/Cassandra.py',
wdir='C:/Users/Yamuna/Desktop')
http://0.0.0.0:8080/
127.0.0.1:1962 - - [07/Feb/2017 17:38:14] "HTTP/1.1 GET /" - 200 OK
127.0.0.1:1973 - - [07/Feb/2017 17:39:27] "HTTP/1.1 GET /crime" - 200 OK
127.0.0.1:1987 - - [07/Feb/2017 17:40:05] "HTTP/1.1 GET /hour" - 200 OK
127.0.0.1:1998 - - [07/Feb/2017 17:40:39] "HTTP/1.1 GET /year" - 200 OK
127.0.0.1:2009 - - [07/Feb/2017 17:41:16] "HTTP/1.1 GET /combined" - 200 OK
```

Creating Two Web Services

sample.war file attached with the submission

File Downloaded from: https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/



Learn More

Get started using Elastic Beanstalk
Modify the code
Create and connect to a database
Add a custom domain

Command Line Interface (v3)

Installing the AWS EB CLI
EB CLI Command Reference

If you want to use a command line to create, manage, and scale your Elastic Beanstalk applications, please use the Elastic Beanstalk Command Line Interface (EB CLI).

Get Started

```
$ mkdir HelloWorld
$ cd HelloWorld
$ eb init -p PHP
$ echo "Hello World" > index.html
$ eb create dev-env
$ eb open
```

All Applications

WebService1

Sample-env-1

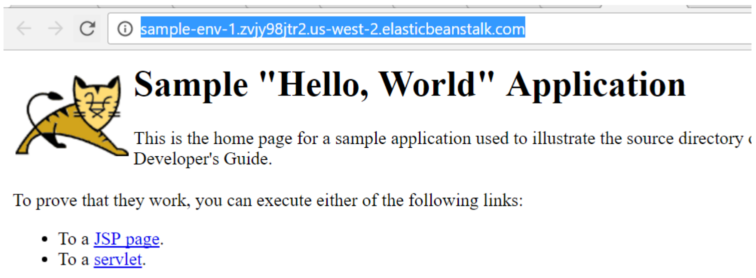
Environment tier: Web Server
Running versions: sample
Last modified: 2017-02-07 16:26:16 UTC-0400
URL: Sample-env-1.zvjy98jtr2.us-west-2.elasticb...

WebService2

Sample-env-2

Environment tier: Web Server
Running versions: webservice2-source
Last modified: 2017-02-07 16:31:06 UTC-0400
URL: Sample-env-2.dqugrashhm.us-west-2.elasti...

http://sample-env-1.zvjy98jtr2.us-west-2.elasticbeanstalk.com/



sample-env-1.zvjy98jtr2.us-west-2.elasticbeanstalk.com

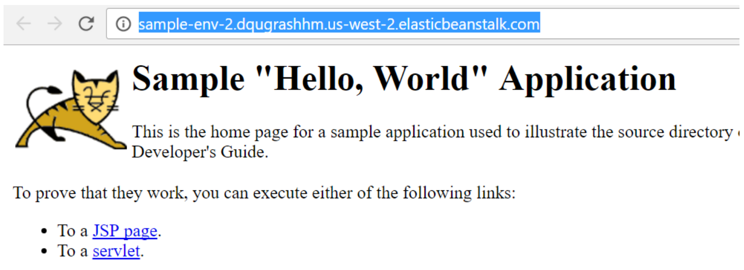
Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory
Developer's Guide.

To prove that they work, you can execute either of the following links:

- To a JSP page.
- To a servlet.

http://sample-env-2.dqugrashhm.us-west-2.elasticbeanstalk.com/



sample-env-2.dqugrashhm.us-west-2.elasticbeanstalk.com

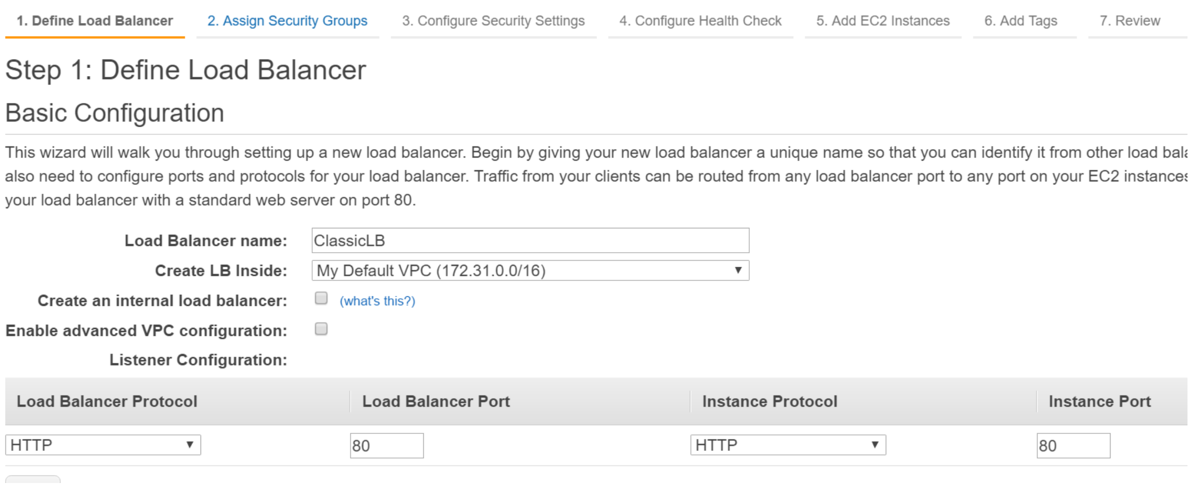
Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory
Developer's Guide.

To prove that they work, you can execute either of the following links:

- To a JSP page.
- To a servlet.

Load Balancer Creation



1. Define Load Balancer
2. Assign Security Groups
3. Configure Security Settings
4. Configure Health Check
5. Add EC2 Instances
6. Add Tags
7. Review

Step 1: Define Load Balancer

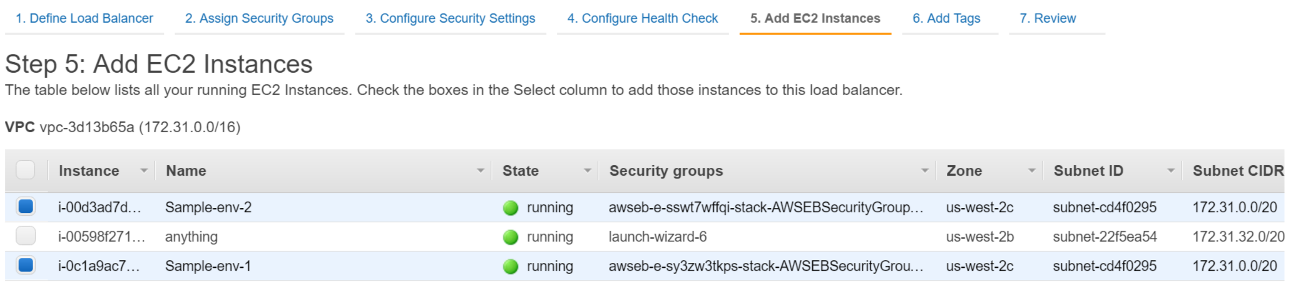
Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load bal
also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instance
your load balancer with a standard web server on port 80.

Load Balancer name:	ClassicLB
Create LB Inside:	My Default VPC (172.31.0.0/16)
Create an internal load balancer:	☐ (what's this?)
Enable advanced VPC configuration:	☐
Listener Configuration:	

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

Load Balancer – Adding EC2 Instances



1. Define Load Balancer
2. Assign Security Groups
3. Configure Security Settings
4. Configure Health Check
5. Add EC2 Instances
6. Add Tags
7. Review

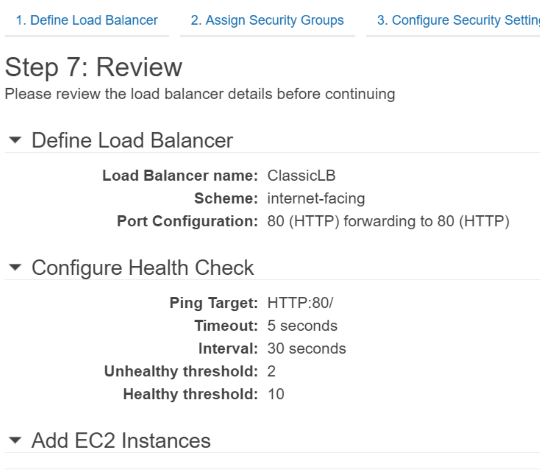
Step 5: Add EC2 Instances

The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.

VPC vpc-3d13b65a (172.31.0.0/16)

☐	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
☑	i-00d3ad7d…	Sample-env-2	● running	awseb-e-sswt7wffqi-stack-AWSEBSecurityGroup…	us-west-2c	subnet-cd4f0295	172.31.0.0/20
☐	i-00598f271…	anything	● running	launch-wizard-6	us-west-2b	subnet-22f5ea54	172.31.32.0/20
☑	i-0c1a9ac7…	Sample-env-1	● running	awseb-e-sy3zw3tkps-stack-AWSEBSecurityGrou…	us-west-2c	subnet-cd4f0295	172.31.0.0/20

Review



Step 7: Review

Please review the load balancer details before continuing

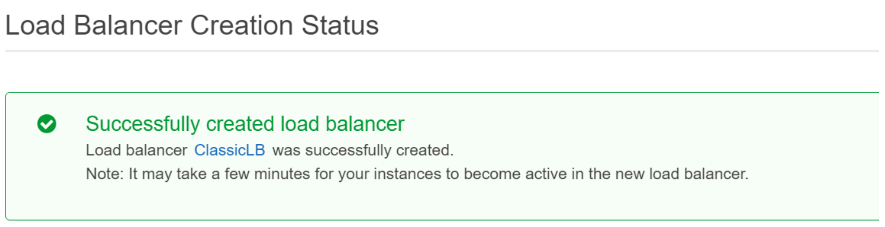
▾ Define Load Balancer

Load Balancer name:	ClassicLB
Scheme:	internet-facing
Port Configuration:	80 (HTTP) forwarding to 80 (HTTP)

▾ Configure Health Check

Ping Target:	HTTP:80/
Timeout:	5 seconds
Interval:	30 seconds
Unhealthy threshold:	2
Healthy threshold:	10

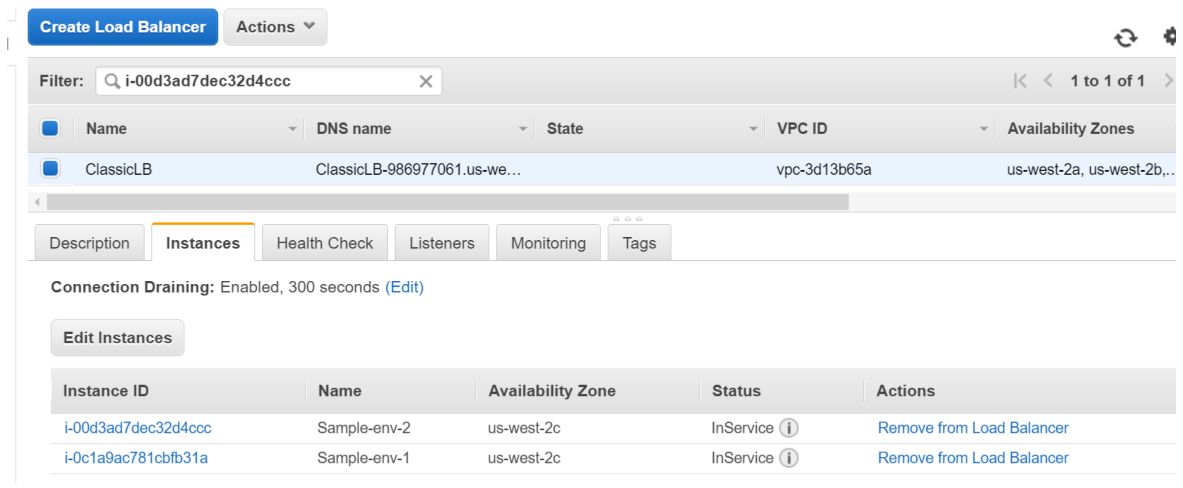
▾ Add EC2 Instances


Load Balancer Creation Status

Successfully created load balancer

Load balancer ClassicLB was successfully created.

Note: It may take a few minutes for your instances to become active in the new load balancer.

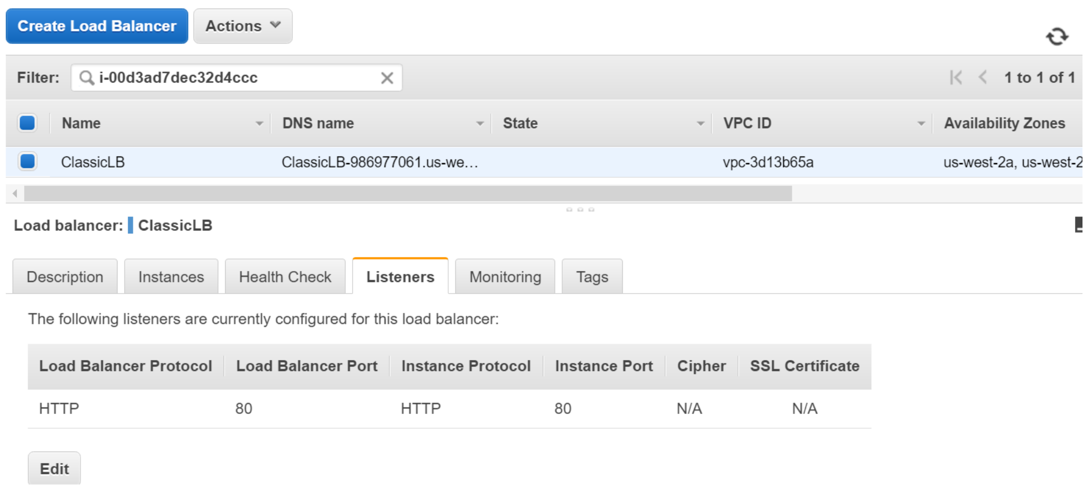


Filter: i-00d3ad7dec32d4ccc — 1 to 1 of 1

Name	DNS name	State	VPC ID	Availability Zones
ClassicLB	ClassicLB-986977061.us-we…		vpc-3d13b65a	us-west-2a, us-west-2b,…

Connection Draining: Enabled, 300 seconds (Edit)

Instance ID	Name	Availability Zone	Status	Actions
i-00d3ad7dec32d4ccc	Sample-env-2	us-west-2c	InService	Remove from Load Balancer
i-0c1a9ac781cbfb31a	Sample-env-1	us-west-2c	InService	Remove from Load Balancer



Filter: i-00d3ad7dec32d4ccc — 1 to 1 of 1

Name	DNS name	State	VPC ID	Availability Zones
ClassicLB	ClassicLB-986977061.us-we…		vpc-3d13b65a	us-west-2a, us-west-2

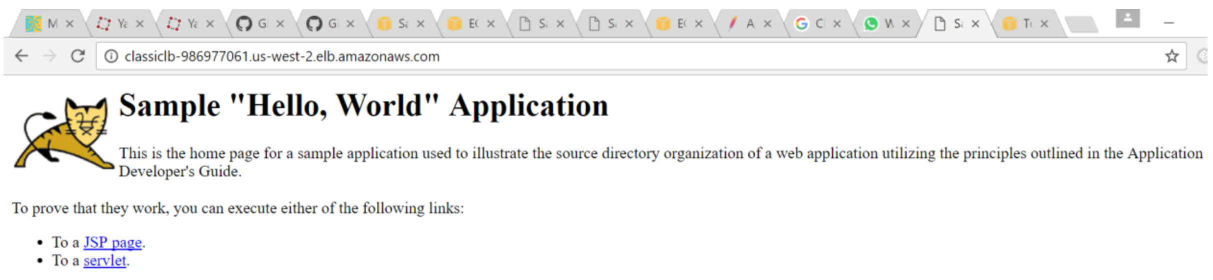
Load balancer: ClassicLB

The following listeners are currently configured for this load balancer:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate
HTTP	80	HTTP	80	N/A	N/A

Load Balancer – Working

URL - ClassicLB-986977061.us-west-2.elb.amazonaws.com



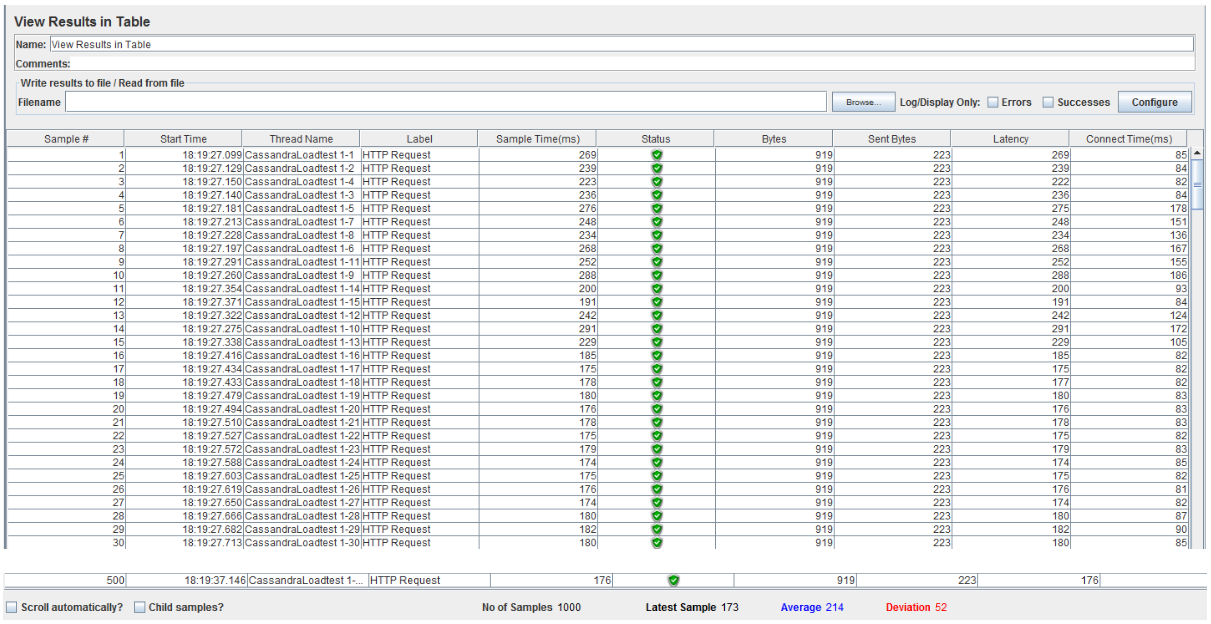
Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory organization of a web application utilizing the principles outlined in the Application Developer's Guide.

To prove that they work, you can execute either of the following links:

- To a JSP page.
- To a servlet.

d. JMETER TEST RESULTS



View Results in Table

Name: View Results in Table

Comments:

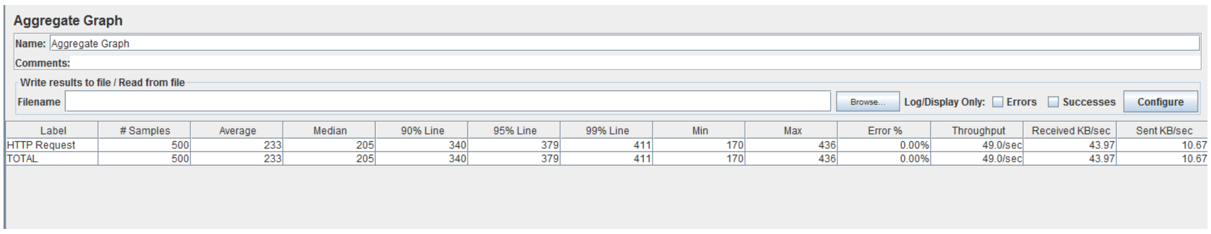
Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	18:19:27.099	CassandraLoadtest 1-1	HTTP Request	269	✓	919	223	269	85
2	18:19:27.129	CassandraLoadtest 1-2	HTTP Request	239	✓	919	223	239	84
3	18:19:27.150	CassandraLoadtest 1-4	HTTP Request	223	✓	919	223	222	82
4	18:19:27.140	CassandraLoadtest 1-3	HTTP Request	236	✓	919	223	236	84
5	18:19:27.181	CassandraLoadtest 1-5	HTTP Request	276	✓	919	223	275	178
6	18:19:27.213	CassandraLoadtest 1-7	HTTP Request	248	✓	919	223	248	151
7	18:19:27.228	CassandraLoadtest 1-8	HTTP Request	234	✓	919	223	234	136
8	18:19:27.197	CassandraLoadtest 1-6	HTTP Request	268	✓	919	223	268	167
9	18:19:27.291	CassandraLoadtest 1-11	HTTP Request	252	✓	919	223	252	155
10	18:19:27.260	CassandraLoadtest 1-9	HTTP Request	288	✓	919	223	288	186
11	18:19:27.354	CassandraLoadtest 1-14	HTTP Request	200	✓	919	223	200	93
12	18:19:27.371	CassandraLoadtest 1-15	HTTP Request	191	✓	919	223	191	84
13	18:19:27.322	CassandraLoadtest 1-12	HTTP Request	242	✓	919	223	242	124
14	18:19:27.275	CassandraLoadtest 1-10	HTTP Request	291	✓	919	223	291	172
15	18:19:27.338	CassandraLoadtest 1-13	HTTP Request	229	✓	919	223	229	105
16	18:19:27.416	CassandraLoadtest 1-16	HTTP Request	185	✓	919	223	185	82
17	18:19:27.434	CassandraLoadtest 1-17	HTTP Request	175	✓	919	223	175	82
18	18:19:27.433	CassandraLoadtest 1-18	HTTP Request	178	✓	919	223	177	82
19	18:19:27.479	CassandraLoadtest 1-19	HTTP Request	180	✓	919	223	180	83
20	18:19:27.494	CassandraLoadtest 1-20	HTTP Request	176	✓	919	223	176	83
21	18:19:27.510	CassandraLoadtest 1-21	HTTP Request	178	✓	919	223	178	83
22	18:19:27.527	CassandraLoadtest 1-22	HTTP Request	175	✓	919	223	175	82
23	18:19:27.572	CassandraLoadtest 1-23	HTTP Request	179	✓	919	223	179	83
24	18:19:27.588	CassandraLoadtest 1-24	HTTP Request	174	✓	919	223	174	85
25	18:19:27.603	CassandraLoadtest 1-25	HTTP Request	175	✓	919	223	175	82
26	18:19:27.619	CassandraLoadtest 1-26	HTTP Request	176	✓	919	223	176	81
27	18:19:27.650	CassandraLoadtest 1-27	HTTP Request	174	✓	919	223	174	82
28	18:19:27.666	CassandraLoadtest 1-28	HTTP Request	180	✓	919	223	180	87
29	18:19:27.682	CassandraLoadtest 1-29	HTTP Request	182	✓	919	223	182	90
30	18:19:27.713	CassandraLoadtest 1-30	HTTP Request	180	✓	919	223	180	85
500	18:19:37.146	CassandraLoadtest 1-...	HTTP Request	176	✓	919	223	176	

☐ Scroll automatically? ☐ Child samples? No of Samples 1000 Latest Sample 173 Average 214 Deviation 52



Aggregate Graph

Name: Aggregate Graph

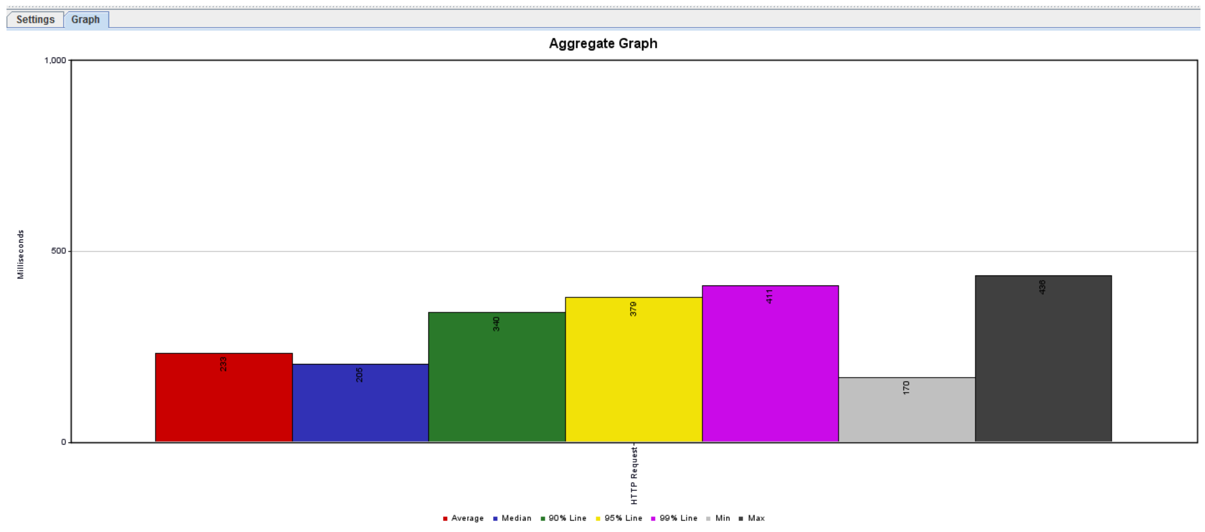
Comments:

Write results to file / Read from file

Filename

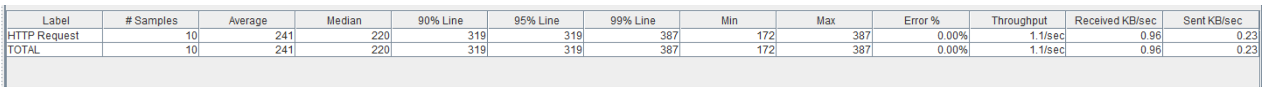
Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	500	233	205	340	379	411	170	436	0.00%	49.0/sec	43.97	10.67
TOTAL	500	233	205	340	379	411	170	436	0.00%	49.0/sec	43.97	10.67



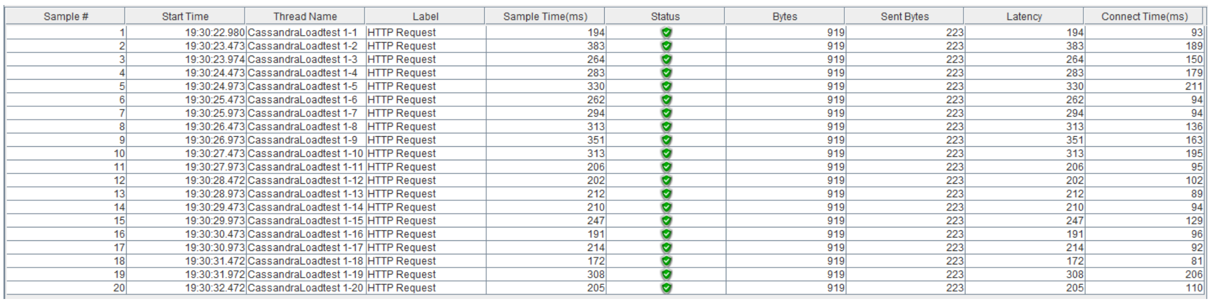
Load balancer with two instances

Sample - 10

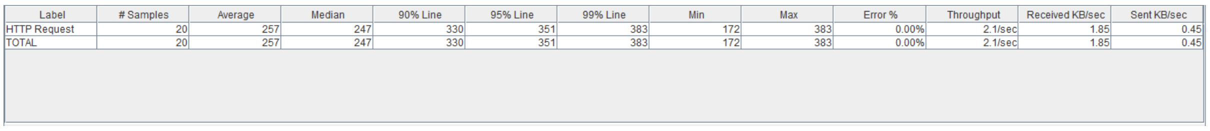


Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	10	241	220	319	319	387	172	387	0.00%	1.1/sec	0.96	0.23
TOTAL	10	241	220	319	319	387	172	387	0.00%	1.1/sec	0.96	0.23

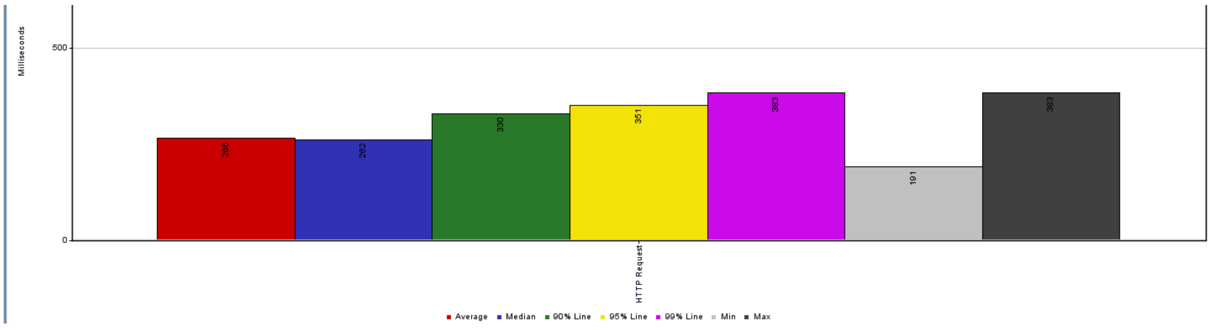
Sample – 20



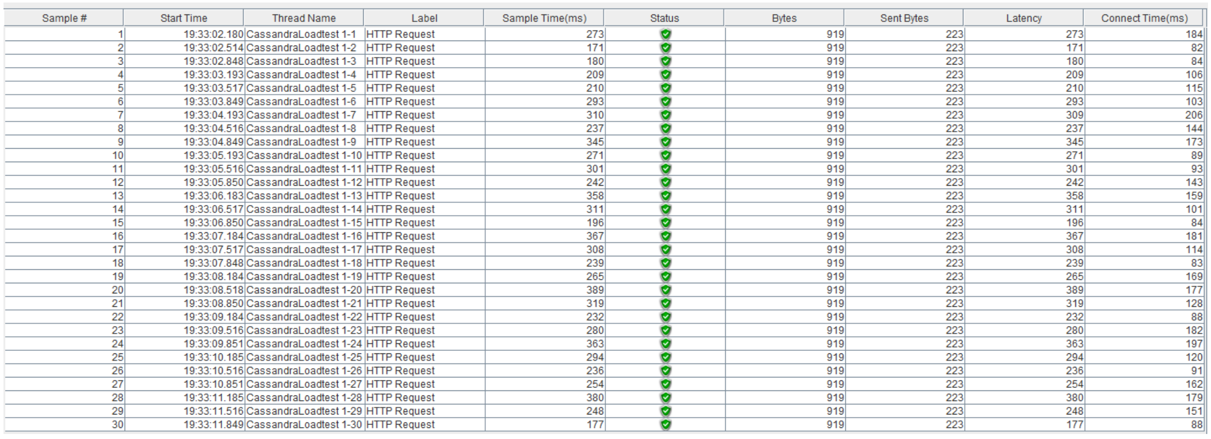
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	19:30:22.980	CassandraLoadtest 1-1	HTTP Request	194	✓	919	223	194	93
2	19:30:23.473	CassandraLoadtest 1-2	HTTP Request	383	✓	919	223	383	189
3	19:30:23.974	CassandraLoadtest 1-3	HTTP Request	264	✓	919	223	264	150
4	19:30:24.473	CassandraLoadtest 1-4	HTTP Request	283	✓	919	223	283	179
5	19:30:24.973	CassandraLoadtest 1-5	HTTP Request	330	✓	919	223	330	211
6	19:30:25.473	CassandraLoadtest 1-6	HTTP Request	262	✓	919	223	262	94
7	19:30:25.973	CassandraLoadtest 1-7	HTTP Request	294	✓	919	223	294	94
8	19:30:26.473	CassandraLoadtest 1-8	HTTP Request	313	✓	919	223	313	136
9	19:30:26.973	CassandraLoadtest 1-9	HTTP Request	351	✓	919	223	351	163
10	19:30:27.473	CassandraLoadtest 1-10	HTTP Request	313	✓	919	223	313	195
11	19:30:27.973	CassandraLoadtest 1-11	HTTP Request	206	✓	919	223	206	95
12	19:30:28.472	CassandraLoadtest 1-12	HTTP Request	202	✓	919	223	202	102
13	19:30:28.973	CassandraLoadtest 1-13	HTTP Request	212	✓	919	223	212	89
14	19:30:29.473	CassandraLoadtest 1-14	HTTP Request	210	✓	919	223	210	94
15	19:30:29.973	CassandraLoadtest 1-15	HTTP Request	247	✓	919	223	247	129
16	19:30:30.473	CassandraLoadtest 1-16	HTTP Request	191	✓	919	223	191	96
17	19:30:30.973	CassandraLoadtest 1-17	HTTP Request	214	✓	919	223	214	92
18	19:30:31.472	CassandraLoadtest 1-18	HTTP Request	172	✓	919	223	172	81
19	19:30:31.972	CassandraLoadtest 1-19	HTTP Request	308	✓	919	223	308	206
20	19:30:32.472	CassandraLoadtest 1-20	HTTP Request	205	✓	919	223	205	110



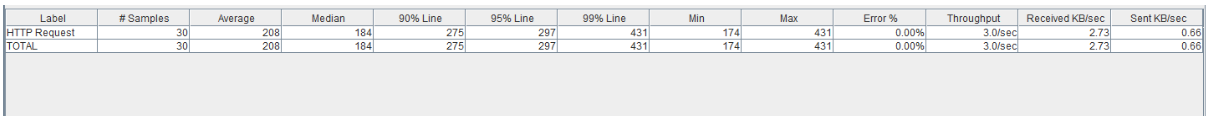
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	20	257	247	330	351	383	172	383	0.00%	2.1/sec	1.85	0.45
TOTAL	20	257	247	330	351	383	172	383	0.00%	2.1/sec	1.85	0.45



Sample 30

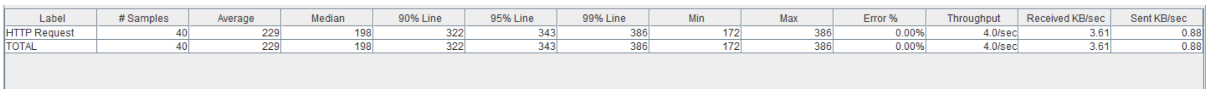


Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	19:33:02.180	CassandraLoadtest 1-1	HTTP Request	273	✓	919	223	273	184
2	19:33:02.514	CassandraLoadtest 1-2	HTTP Request	171	✓	919	223	171	82
3	19:33:02.848	CassandraLoadtest 1-3	HTTP Request	180	✓	919	223	180	84
4	19:33:03.193	CassandraLoadtest 1-4	HTTP Request	209	✓	919	223	209	106
5	19:33:03.517	CassandraLoadtest 1-5	HTTP Request	210	✓	919	223	210	115
6	19:33:03.849	CassandraLoadtest 1-6	HTTP Request	293	✓	919	223	293	103
7	19:33:04.193	CassandraLoadtest 1-7	HTTP Request	310	✓	919	223	309	206
8	19:33:04.516	CassandraLoadtest 1-8	HTTP Request	237	✓	919	223	237	144
9	19:33:04.849	CassandraLoadtest 1-9	HTTP Request	345	✓	919	223	345	173
10	19:33:05.193	CassandraLoadtest 1-10	HTTP Request	271	✓	919	223	271	89
11	19:33:05.516	CassandraLoadtest 1-11	HTTP Request	301	✓	919	223	301	93
12	19:33:05.850	CassandraLoadtest 1-12	HTTP Request	242	✓	919	223	242	143
13	19:33:06.183	CassandraLoadtest 1-13	HTTP Request	358	✓	919	223	358	159
14	19:33:06.517	CassandraLoadtest 1-14	HTTP Request	311	✓	919	223	311	101
15	19:33:06.850	CassandraLoadtest 1-15	HTTP Request	196	✓	919	223	196	84
16	19:33:07.184	CassandraLoadtest 1-16	HTTP Request	367	✓	919	223	367	181
17	19:33:07.517	CassandraLoadtest 1-17	HTTP Request	308	✓	919	223	308	114
18	19:33:07.848	CassandraLoadtest 1-18	HTTP Request	239	✓	919	223	239	83
19	19:33:08.184	CassandraLoadtest 1-19	HTTP Request	265	✓	919	223	265	169
20	19:33:08.518	CassandraLoadtest 1-20	HTTP Request	389	✓	919	223	389	177
21	19:33:08.850	CassandraLoadtest 1-21	HTTP Request	319	✓	919	223	319	128
22	19:33:09.184	CassandraLoadtest 1-22	HTTP Request	232	✓	919	223	232	88
23	19:33:09.516	CassandraLoadtest 1-23	HTTP Request	280	✓	919	223	280	182
24	19:33:09.851	CassandraLoadtest 1-24	HTTP Request	363	✓	919	223	363	197
25	19:33:10.185	CassandraLoadtest 1-25	HTTP Request	294	✓	919	223	294	120
26	19:33:10.516	CassandraLoadtest 1-26	HTTP Request	236	✓	919	223	236	91
27	19:33:10.851	CassandraLoadtest 1-27	HTTP Request	254	✓	919	223	254	162
28	19:33:11.185	CassandraLoadtest 1-28	HTTP Request	380	✓	919	223	380	179
29	19:33:11.516	CassandraLoadtest 1-29	HTTP Request	248	✓	919	223	248	151
30	19:33:11.849	CassandraLoadtest 1-30	HTTP Request	177	✓	919	223	177	88



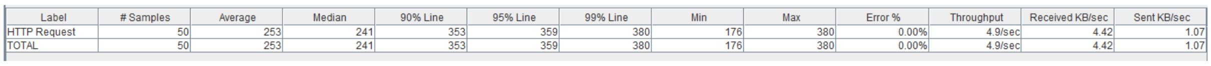
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	30	208	184	275	297	431	174	431	0.00%	3.0/sec	2.73	0.66
TOTAL	30	208	184	275	297	431	174	431	0.00%	3.0/sec	2.73	0.66

Sample 40



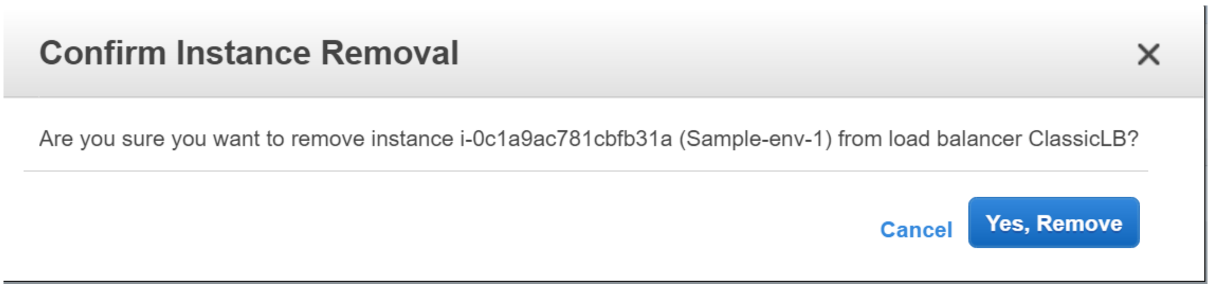
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	40	229	198	322	343	386	172	386	0.00%	4.0/sec	3.61	0.88
TOTAL	40	229	198	322	343	386	172	386	0.00%	4.0/sec	3.61	0.88

Sample 50



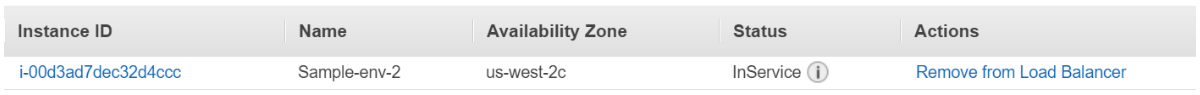
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	50	253	241	353	359	380	176	380	0.00%	4.9/sec	4.42	1.07
TOTAL	50	253	241	353	359	380	176	380	0.00%	4.9/sec	4.42	1.07

Removing one instance from load balancer



Confirm Instance Removal

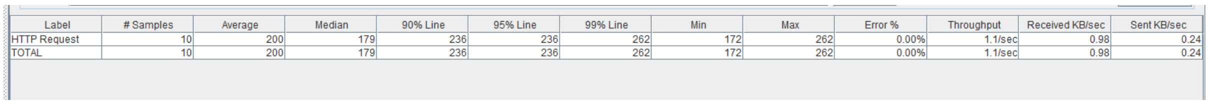
Are you sure you want to remove instance i-0c1a9ac781cbfb31a (Sample-env-1) from load balancer ClassicLB?



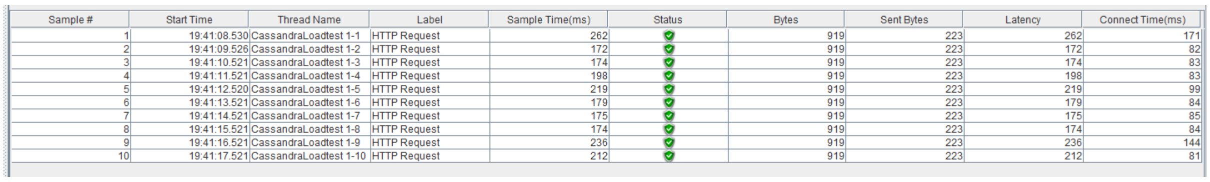
Instance ID	Name	Availability Zone	Status	Actions
i-00d3ad7dec32d4ccc	Sample-env-2	us-west-2c	InService	Remove from Load Balancer

Load balancer with one instance

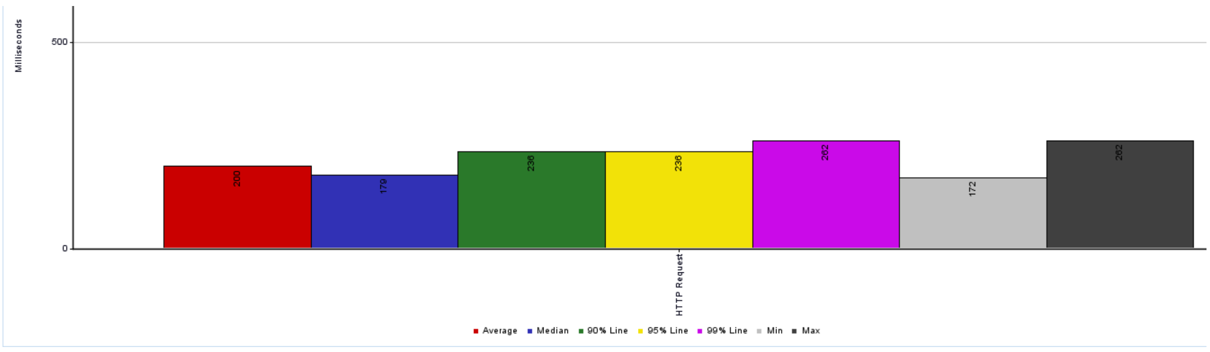
Sample 10



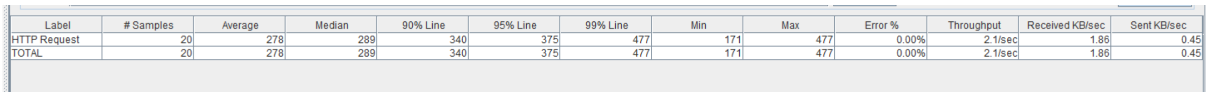
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	10	200	179	236	236	262	172	262	0.00%	1.1/sec	0.98	0.24
TOTAL	10	200	179	236	236	262	172	262	0.00%	1.1/sec	0.98	0.24



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	19:41:08.530	CassandraLoadtest 1-1	HTTP Request	262	✓	919	223	262	171
2	19:41:09.526	CassandraLoadtest 1-2	HTTP Request	172	✓	919	223	172	82
3	19:41:10.521	CassandraLoadtest 1-3	HTTP Request	174	✓	919	223	174	83
4	19:41:11.521	CassandraLoadtest 1-4	HTTP Request	198	✓	919	223	198	83
5	19:41:12.520	CassandraLoadtest 1-5	HTTP Request	219	✓	919	223	219	99
6	19:41:13.521	CassandraLoadtest 1-6	HTTP Request	179	✓	919	223	179	84
7	19:41:14.521	CassandraLoadtest 1-7	HTTP Request	175	✓	919	223	175	85
8	19:41:15.521	CassandraLoadtest 1-8	HTTP Request	174	✓	919	223	174	84
9	19:41:16.521	CassandraLoadtest 1-9	HTTP Request	236	✓	919	223	236	144
10	19:41:17.521	CassandraLoadtest 1-10	HTTP Request	212	✓	919	223	212	81

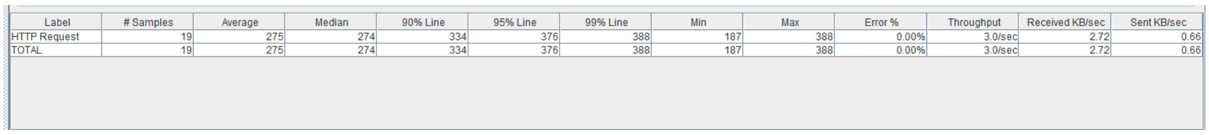


Sample 20



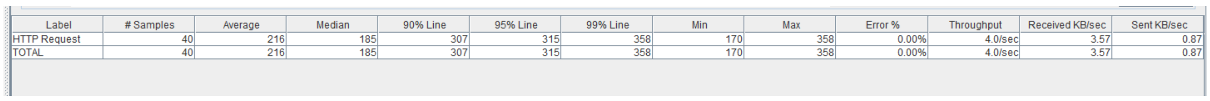
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	20	278	289	340	375	477	171	477	0.00%	2.1/sec	1.86	0.45
TOTAL	20	278	289	340	375	477	171	477	0.00%	2.1/sec	1.86	0.45

Sample 30



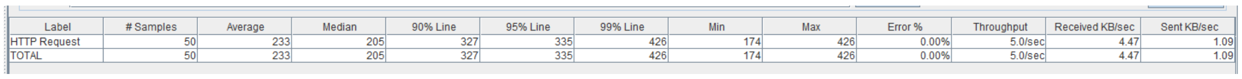
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	19	275	274	334	376	388	187	388	0.00%	3.0/sec	2.72	0.66
TOTAL	19	275	274	334	376	388	187	388	0.00%	3.0/sec	2.72	0.66

Sample 40



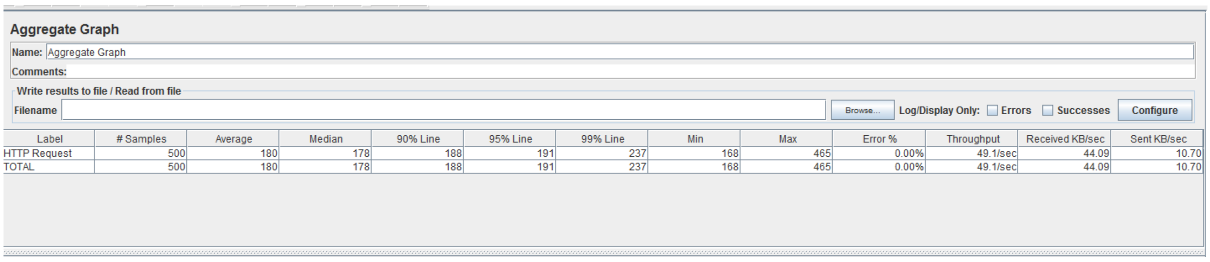
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	40	216	185	307	315	358	170	358	0.00%	4.0/sec	3.57	0.87
TOTAL	40	216	185	307	315	358	170	358	0.00%	4.0/sec	3.57	0.87

Sample 50



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	50	233	205	327	335	426	174	426	0.00%	5.0/sec	4.47	1.09
TOTAL	50	233	205	327	335	426	174	426	0.00%	5.0/sec	4.47	1.09

JMeter – After stopping load Balancer



Aggregate Graph

Name: Aggregate Graph

Comments:

Write results to file / Read from file

Filename

Log/Display Only: Errors Successes

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	500	180	178	188	191	237	168	465	0.00%	49.1/sec	44.09	10.70
TOTAL	500	180	178	188	191	237	168	465	0.00%	49.1/sec	44.09	10.70

The throughput, ReceivedKB/sec and SentKB/sec for the load balancer with single and two instances are almost same. No significant difference is observed. The JMeter continues to run even after the load balancer is stopped. However, the Average seconds are also very random based on the observation.

	Average seconds	
Number of users	LB - Two instances	LB - One instance
10	241	200
20	257	278
30	208	275
40	229	216
50	253	233

e. SUMMARY

The instances created on the InstaClustr was up and running the Cassandra database without any interruption and the performance can be considered really very good. The Web services implementation was pretty complicated for python. The load balancer is not showing much difference when measured with JMeter for two and single instance.

f. REFERENCES

https://datastax.github.io/python-driver/

https://datastax.github.io/python-driver/getting_started.html

https://datastax.github.io/python-driver/api/cassandra/cluster.html

https://datastax.github.io/python-driver/security.html

https://academy.datastax.com/resources/getting-started-apache-cassandra-and-python-part-i

http://stackoverflow.com/questions/415511/how-to-get-current-time-in-python http://www.datastax.com/dev/blog/a-deep-look-to-the-cql-where-clause

http://metadata.phila.gov/#home/datasetdetails/5543868920583086178c4f8e/representationdetails/570e7621c03327dc14f4b68d/