Simple Role Based Authentication and Authorization for cassandra keyspaces (RBAC)

Purpose: To create role based authentication and authorization for cassandra keyspaces.

We need to follow following steps in order to implement RBAC in cassandra.

Step 1: Modify **conf/cassandra.yaml** file in cassandra distribution package.

Changes=>

1. authenticator: AllowAllAuthenticator => authenticator: PasswordAuthenticator
2. authorizer: AllowAllAuthorizer => authorizer: CassandraAuthorizer

Step 2: Create users and roles in database

In current system there is a need of creating role for each individual component.

Diagram: showing roles and their relationships.

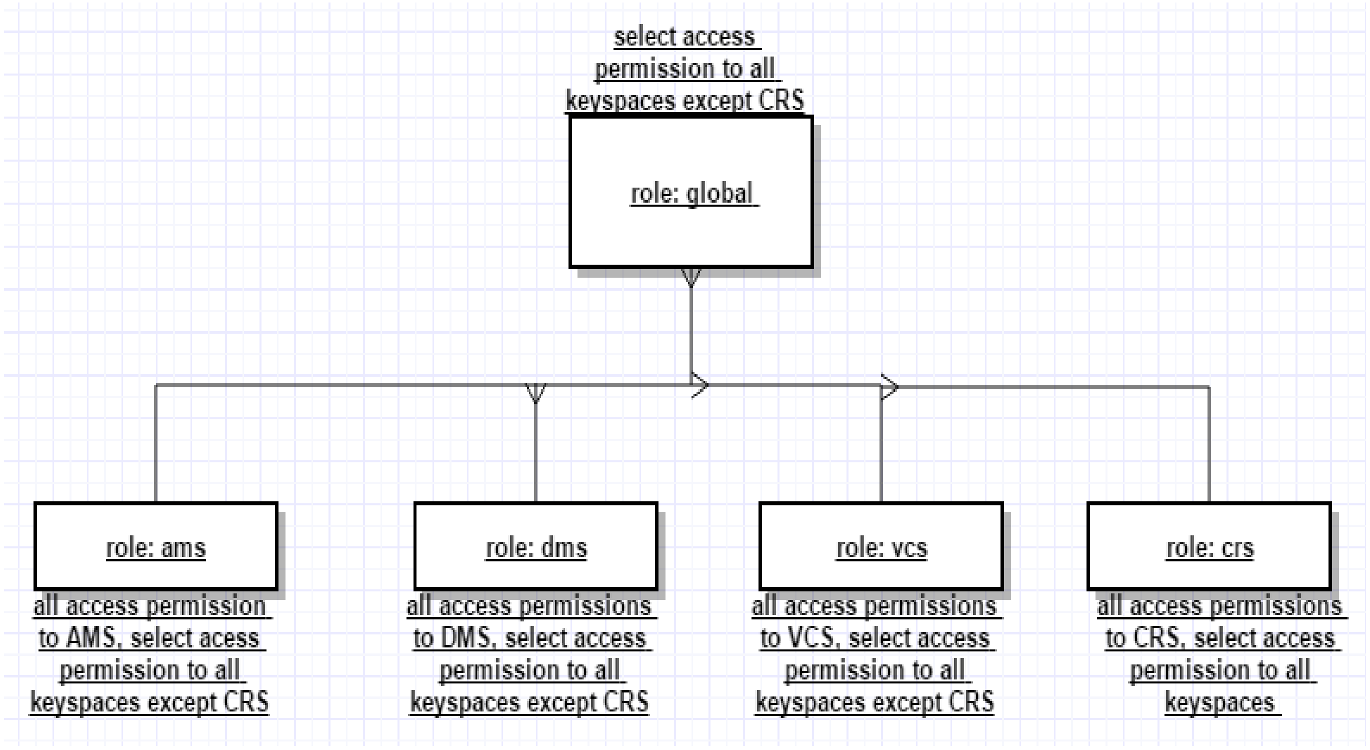

select access permission to all keyspaces except CRS
role: global
role: ams
all access permission to AMS, select acess permission to all keyspaces except CRS
role: dms
all access permissions to DMS, select access permission to all keyspaces except CRS
role: vcs
all access permissions to VCS, select access permission to all keyspaces except CRS
role: crs
all access permissions to CRS, select access permission to all keyspaces

Diagram 1:: roles and their relationships

Keyspace access permissions are mention in following table:

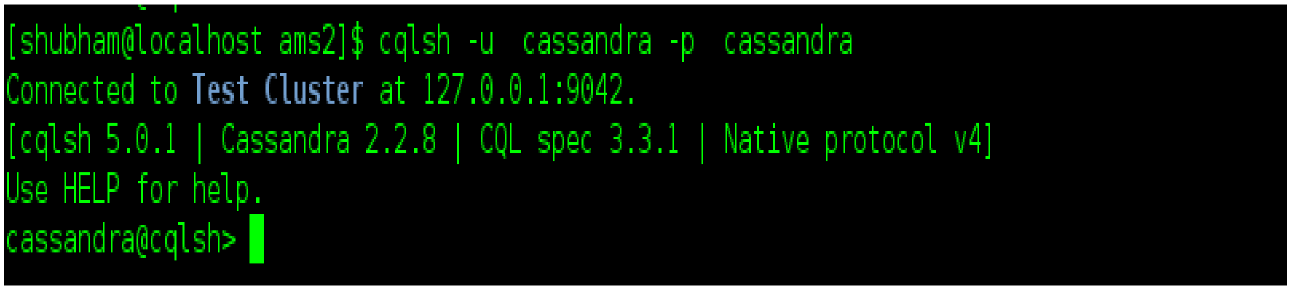
component/user(database)	**role**	**Access permission**
DMS	dms	**DMS** role will have access permission **ALL** to dms keyspace and **select** access permission to all key-spaces except **CRS**
AMS	ams	**AMS** role will have access permission **ALL** to ams key-space and **select** access permission to all key-spaces except **CRS**
VCS	vcs	**VCS** role will have access permission **ALL** to vcs key-space and **select** access permission to all keyspaces except **CRS**
CRS	crs	**CRS** role will have access permission **ALL** to crs key-space and **select** access permission to all keyspaces
MP	global	**SELECT** access permission to all key-spaces **except CRS**

Table no 1: Access permissions table

Implementation commands:

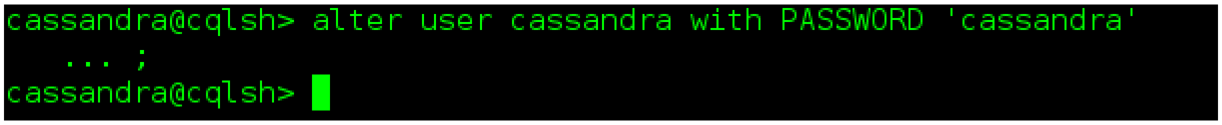
Follow following commands to implement role based authentication and authorization:

1. Enter into interactive shell by using **command:: cqlsh -u cassandra -p cassandra** these are default **superuser** credentials for cassandra.



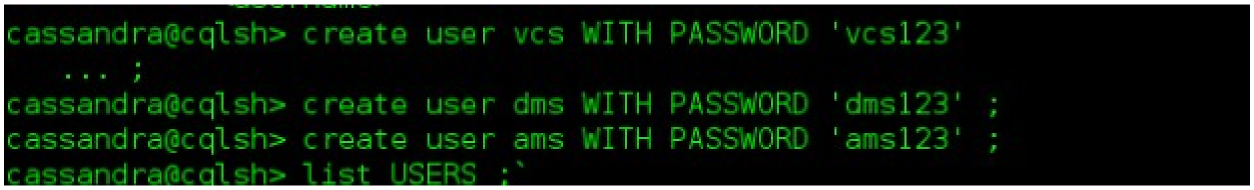
```
[shubham@localhost ams2]$ cqlsh -u  cassandra -p  cassandra
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.2.8 | CQL spec 3.3.1 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh>
```

2. Now change password of cassandra user to new_password: - **command:: cqlsh> alter user cassandra with password 'newpassword'**

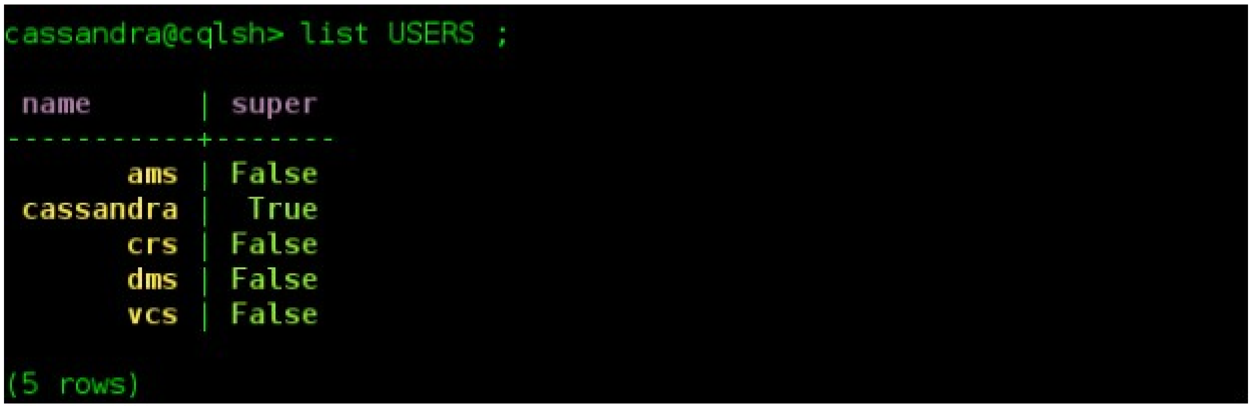


```
cassandra@cqlsh> alter user cassandra with PASSWORD 'cassandra'
   ... ;
cassandra@cqlsh>
```

3. **Create users** for all components DMS,VCS,AMS,CRS,MP:: use **command:: cqlsh> create user username with password 'pass'. Example: create user user1 with password 'pass1'.** By default user will not have any access permission on keyspaces.Further you can grant access permissions by login as superuser 'cassandra'

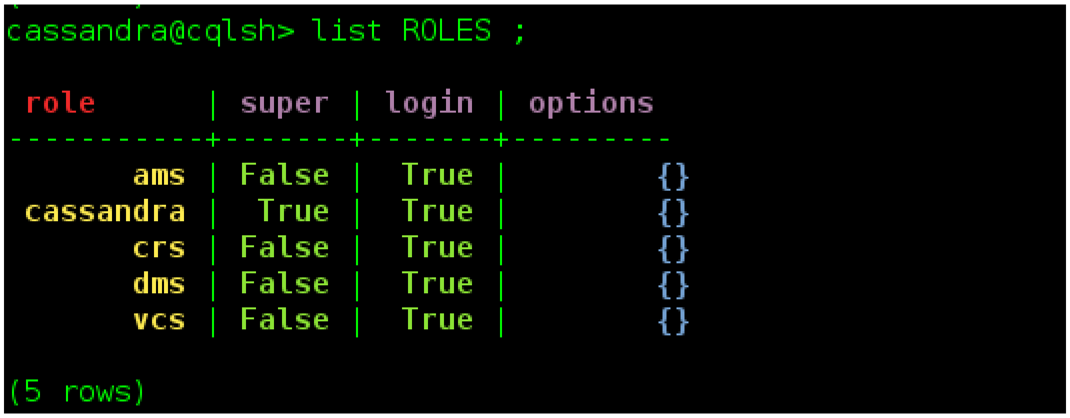


```
cassandra@cqlsh> create user vcs WITH PASSWORD 'vcs123'
   ... ;
cassandra@cqlsh> create user dms WITH PASSWORD 'dms123' ;
cassandra@cqlsh> create user ams WITH PASSWORD 'ams123' ;
cassandra@cqlsh> list USERS ;`
```



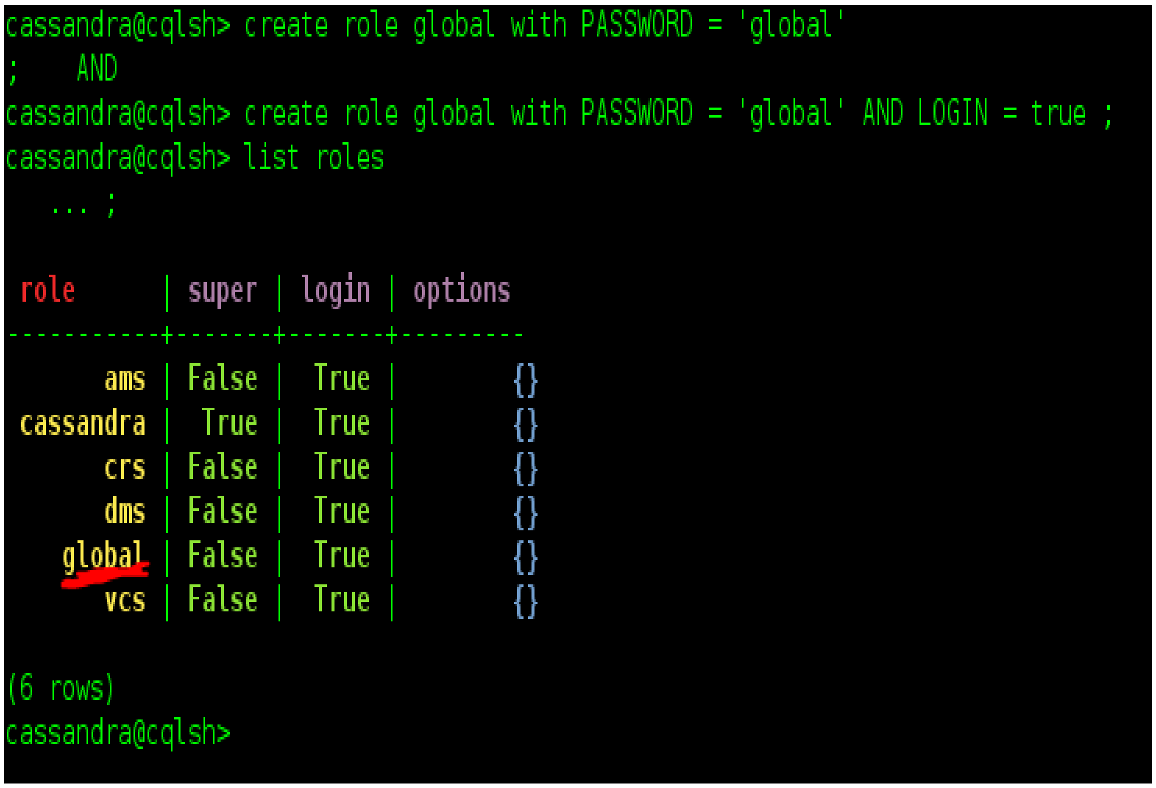
```
cassandra@cqlsh> list USERS ;

 name      | super
-----------+-------
       ams | False
 cassandra |  True
       crs | False
       dms | False
       vcs | False

(5 rows)
```

4. **Create roles::** When we create user, it will automatically creates role with same name. You can see roles that are created automatically



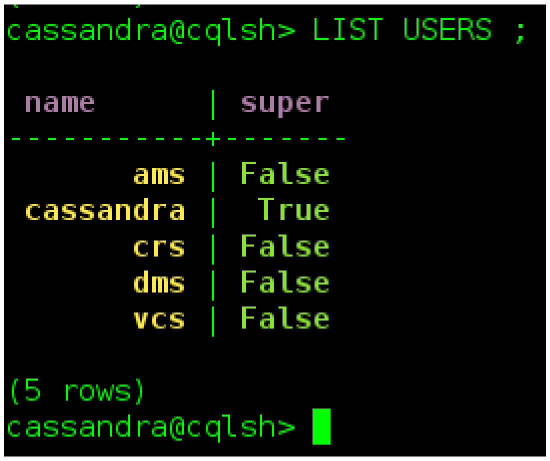
```
cassandra@cqlsh> list ROLES ;

 role      | super | login | options
-----------+-------+-------+---------
       ams | False |  True |        {}
 cassandra |  True |  True |        {}
       crs | False |  True |        {}
       dms | False |  True |        {}
       vcs | False |  True |        {}

(5 rows)
```

You can create role irrespective of user by **command:: create role global with password 'global' AND login=true**;



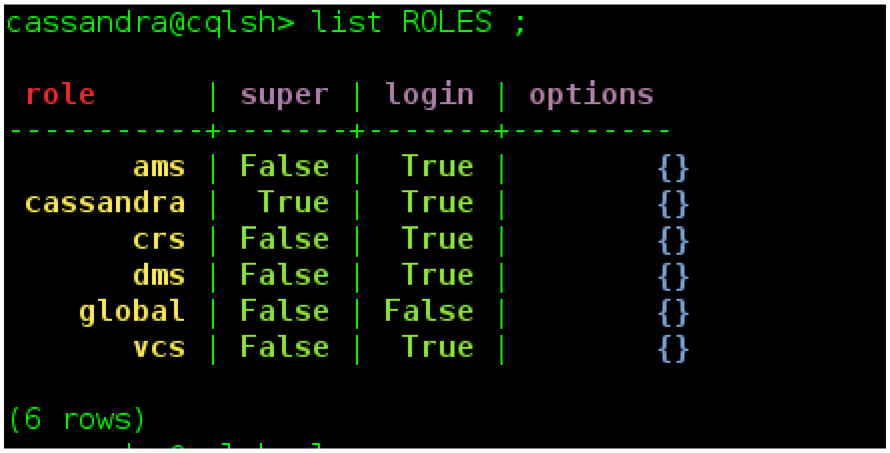
```
cassandra@cqlsh> create role global with PASSWORD = 'global'
;    AND
cassandra@cqlsh> create role global with PASSWORD = 'global' AND LOGIN = true ;
cassandra@cqlsh> list roles
   ... ;

 role      | super | login | options
-----------+-------+-------+---------
       ams | False |  True |        {}
 cassandra |  True |  True |        {}
       crs | False |  True |        {}
       dms | False |  True |        {}
    global | False |  True |        {}
       vcs | False |  True |        {}

(6 rows)
cassandra@cqlsh>
```

5. Verify user creation from cqlsh command:: **list users**;



```
cassandra@cqlsh> LIST USERS ;

 name      | super
-----------+-------
       ams | False
 cassandra |  True
       crs | False
       dms | False
       vcs | False

(5 rows)
cassandra@cqlsh>
```

6. Verify role creation by cqlsh command :: **List roles**

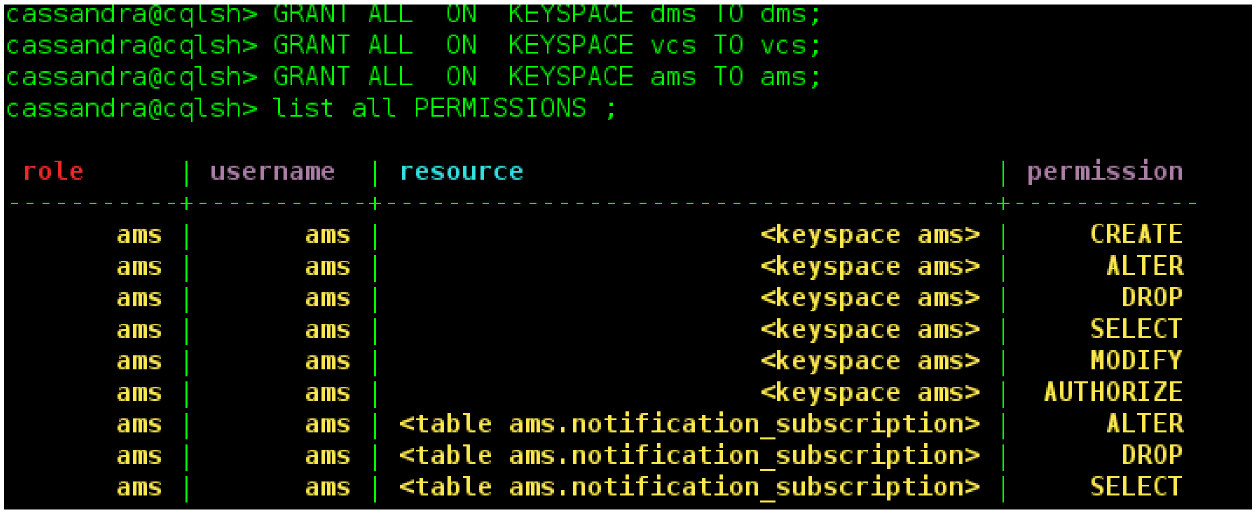


```
cassandra@cqlsh> list ROLES ;

 role      | super | login | options
-----------+-------+-------+---------
       ams | False |  True |        {}
 cassandra |  True |  True |        {}
       crs | False |  True |        {}
       dms | False |  True |        {}
    global | False | False |        {}
       vcs | False |  True |        {}

(6 rows)
```

7. Now it requires to give necessary access permission to roles to limit key-space access. **Command:: GRANT permission 'permission_name' ON KEYSPACE 'keyspace_name' TO role_name.** There are a different types of access permissions for keyspaces these are:

- ALL
- ALTER
- AUTHORIZE
- CREATE
- DROP
- MODIFY
- SELECT

Example:



```
cassandra@cqlsh> GRANT ALL  ON  KEYSPACE dms TO dms;
cassandra@cqlsh> GRANT ALL  ON  KEYSPACE vcs TO vcs;
cassandra@cqlsh> GRANT ALL  ON  KEYSPACE ams TO ams;
cassandra@cqlsh> list all PERMISSIONS ;

 role       | username  | resource                                | permission
------------+-----------+-----------------------------------------+------------
       ams  |       ams |                          <keyspace ams> |     CREATE
       ams  |       ams |                          <keyspace ams> |      ALTER
       ams  |       ams |                          <keyspace ams> |       DROP
       ams  |       ams |                          <keyspace ams> |     SELECT
       ams  |       ams |                          <keyspace ams> |     MODIFY
       ams  |       ams |                          <keyspace ams> |  AUTHORIZE
       ams  |       ams | <table ams.notification_subscription> |      ALTER
       ams  |       ams | <table ams.notification_subscription> |       DROP
       ams  |       ams | <table ams.notification_subscription> |     SELECT
```

In our case:

We are going to make separate users for all components. We are achieving authorization by assigning necessary roles to users.

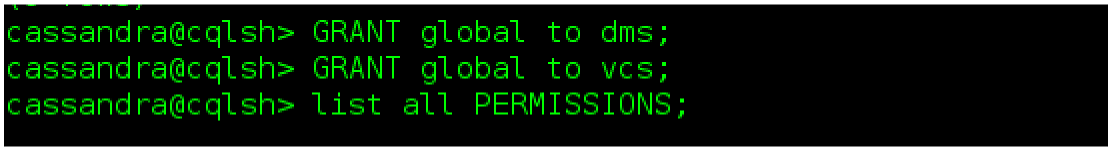
global role:: global role will have SELECT access permission on vcs, dms, ams keyspaces.

CRS,DMS,AMS,VCS,MP roles: access permissions are as per Table No 1.

cassandra@cqlsh> GRANT ALL ON KEYSPACE dms TO dms; (To grant access permission ALL on keyspace dms to role dms)

cassandra@cqlsh>GRANT global to ams; (to GRANT select on all keyspaces except CRS. This is done by simply assigning global role to dms**).**

Apply the same for ams, vcs, mp and crs roles.



```
cassandra@cqlsh> GRANT global to dms;
cassandra@cqlsh> GRANT global to vcs;
cassandra@cqlsh> list all PERMISSIONS;
```

As MP do not have its own keyspace just grant role global to MP. Execute second command.Now every role has access to required keyspaces

Approach to make common user for all components except crs.(Alternate)

Create user COMMON_USER by default COMMON_USER role will get generate. Give this role All access permission to ams,dms,vcs keyspaces. We can use this user credentials to log in from other components.

We can use CRS user as it is for crs keyspace.

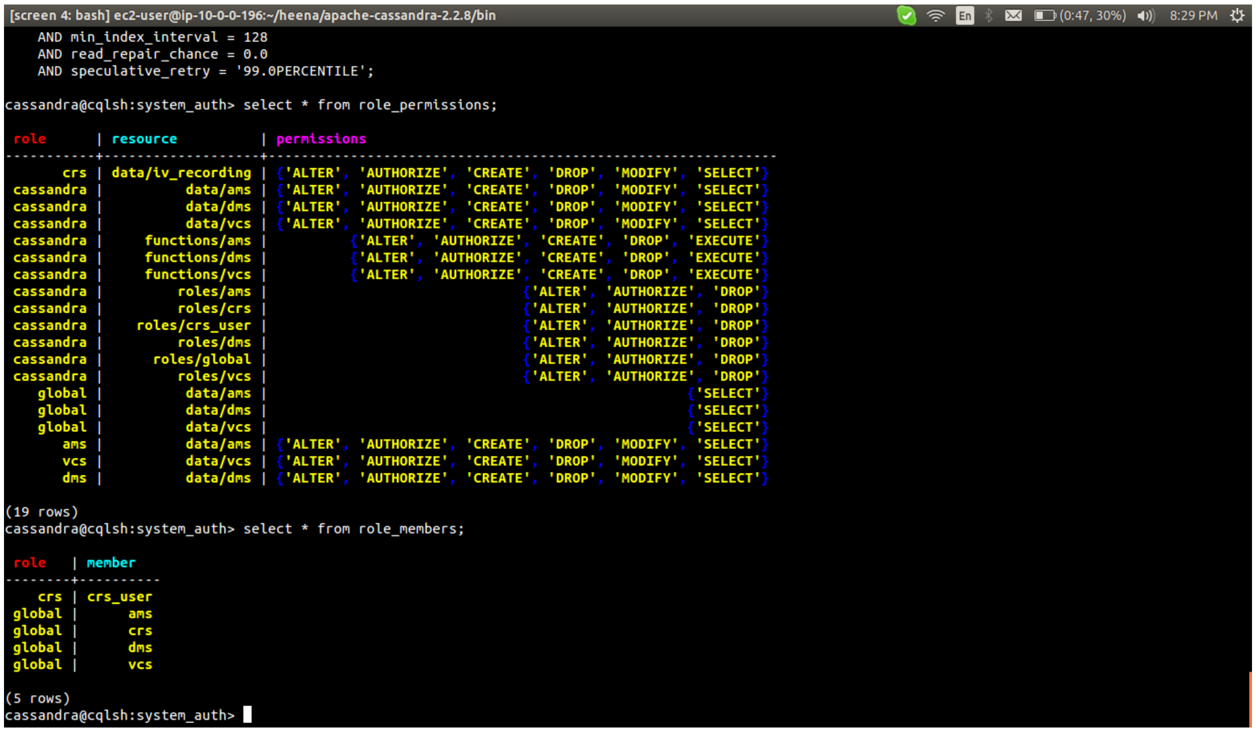
Step 3: Code change:: Need to change code for cassandra connection in all components with proper login credentials specific to component user.Global user credentials for ams, dms, vcs, mp

FROM: private val cluster = Cluster.builder()..addContactPoints(nodes: _*).withPort(port.trim.toInt).withReconnectionPolicy(new ConstantReconnectionPolicy(1000L)).build()

TO:
private val cluster =
Cluster.builder().**withCredentials("username","password")**.addContactPoints(nodes: _*).withPort(port.trim.toInt).withReconnectionPolicy(new ConstantReconnectionPolicy(1000L)).build()

Step 4: Re-deployment of all components.

All roles and related access permissions will look like:



```
[screen 4: bash] ec2-user@ip-10-0-0-196:~/heena/apache-cassandra-2.2.8/bin                    (0:47, 30%)   8:29 PM
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99.0PERCENTILE';

cassandra@cqlsh:system_auth> select * from role_permissions;

 role      | resource          | permissions
-----------+-------------------+-------------------------------------------------------------
       crs | data/iv_recording | {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'MODIFY', 'SELECT'}
 cassandra |          data/ams | {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'MODIFY', 'SELECT'}
 cassandra |          data/dms | {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'MODIFY', 'SELECT'}
 cassandra |          data/vcs | {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'MODIFY', 'SELECT'}
 cassandra |     functions/ams |          {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'EXECUTE'}
 cassandra |     functions/dms |          {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'EXECUTE'}
 cassandra |     functions/vcs |          {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'EXECUTE'}
 cassandra |         roles/ams |                                {'ALTER', 'AUTHORIZE', 'DROP'}
 cassandra |         roles/crs |                                {'ALTER', 'AUTHORIZE', 'DROP'}
 cassandra |    roles/crs_user |                                {'ALTER', 'AUTHORIZE', 'DROP'}
 cassandra |         roles/dms |                                {'ALTER', 'AUTHORIZE', 'DROP'}
 cassandra |      roles/global |                                {'ALTER', 'AUTHORIZE', 'DROP'}
 cassandra |         roles/vcs |                                {'ALTER', 'AUTHORIZE', 'DROP'}
    global |          data/ams |                                                    {'SELECT'}
    global |          data/dms |                                                    {'SELECT'}
    global |          data/vcs |                                                    {'SELECT'}
       ams |          data/ams | {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'MODIFY', 'SELECT'}
       vcs |          data/vcs | {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'MODIFY', 'SELECT'}
       dms |          data/dms | {'ALTER', 'AUTHORIZE', 'CREATE', 'DROP', 'MODIFY', 'SELECT'}

(19 rows)
cassandra@cqlsh:system_auth> select * from role_members;

 role   | member
--------+----------
    crs | crs_user
 global |      ams
 global |      crs
 global |      dms
 global |      vcs

(5 rows)
cassandra@cqlsh:system_auth>
```

References :

Role based Access Control cassandra documentation.

http://www.datastax.com/dev/blog/a-quick-tour-of-internal-authentication-and-authorization-security-in-datastax-enterprise-and-apache-cassandra