Install and configure pgbouncer for connection pooling.

1/ Installation of pgbouncer by Sly

2/ After Installing pgbouncer by Sly , we have two important files to work on,

* pgbouncer.ini
* userlist.txt

The pgbouncer.ini file is used to configure all our parameters for connection pooling. The userlist.txt is the authentication file used to whitelist all the users already available in the database instance. In the userlist.txt you will have to add every user name and password from the database for pgbouncer to verify that the credentials are correct before establishing a connection. But this is a lot of administrative work because everytime a user is added th PG, you also need to add it in the userlist.txt file and everytime you change password it has to also be changed in the uselist.txt file as well. To avoid this we will have to make use of auth\_user and auth\_query to centralize the password management. When this is provided pgbouncer will have to read credentials from connection string, but instead of comparing against userlist.txt, it logs in to postgres with the specified auth\_user (non-superuser) and runs auth\_query to pull the corresponding md5 password hash for the desired user and if correct then user is allowed to log in.

3/ Create a postgres user to use as auth\_user ex

* CREATE USER pgbounceruser WITH PASSWORD ‘medimpact123’;
* GRANT SELECT to pgbounceruser

4/ Create the user/password lookup function in postgres . This is because non-superuser doesn't have permission to read pg\_shadow so it means that if authuser is not superuser, we will have some permission denied errors in the Postgres log. So creating the function bellow allows non-superuser (authuser) to query pg\_authid because it is a security definer function. In the pgbouncer.ini adjust the auth\_query parameter accordingly:

auth\_query = SELECT rolename, rolepassword FROM user\_credentials($1)

* CREATE OR REPLACE FUNCTION user\_credentials(IN vusername text, OUT rolename text, OUT rolepassword text)

RETURNS record AS $$

BEGIN

SELECT rolname, rolpassword FROM pg\_authid WHERE rolname = vusername INTO rolename, rolepassword;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

* REVOKE ALL PRIVILEGES ON FUNCTION user\_credentials FROM PUBLIC;
* GRANT EXECUTE ON FUNCTION user\_credentials TO pgbounceruser;

5/ Configure pgbouncer.ini by making parameter changes

[databases]

citus = port=5432 auth\_user=pgbounceruser pool\_size=100

[pgbouncer]

listen\_addr = \* or 0.0.0.0

listen\_port = 6433

auth\_type = md5

auth\_file = /opt/app/pgbouncer-data/userlist.txt

admin\_users = admin\_user

pool\_mode = transaction

max\_client\_conn = 300 ( It sets the maximum num of conn’s for all databases )

default\_pool\_size = 100 ( Is a per database parameter that sets default size for specific DB)

logfile = /var/log/pgbouncer/pgbouncer.log

pidfile = /var/run/pgbouncer/pgbouncer.pid

auth\_query = auth\_query = SELECT rolename, rolepassword FROM user\_credentials($1)

6/ Get pgbouncer to start on reboot

* Sudo chkconfig pgbouncer on
* Sudo chkconfig –list | grep pgbouncer

7/ Start pgbouncer

* Pgbouncer /opt/app/pgbouncer-data/pgbouncer.ini -v

8/ In the postgres HBA whitelist bellow

host all pgbounceruser ::1/128 md5

host all pgbounceruser 127.0.0.1/32 md5

host all citus ::1/128 md5

host all citus 127.0.0.1/32 md5

Step 8 whitelisting in pg\_hba.conf is done because pgbouncer first does a connection using auth\_user called pgbounceruser to execute the auth\_query and obtains the credentials. After that, pgbouncer does the real connection using the credential (it gets in the previous step) and the connection string that is why we have above whitelist in step 8.