# Overview

## Architecture and Concept

# Installation

## Installation and Configuration

***Before Installing the Postgres-XL, remember the few points***

* Do the proper passwordless ssh between users created for the Postgres-XL
* Configure, make of Postgresxl and make of pgxc\_ctl should be done in user not root
* After make, make install has to be done in root, Installation all from the root
* Data and coordinator directories are created in $HOME/serveradm/postgres(pgxcl), u can change this directory

**Install Common Utilities(this need to be done for all the postgresxl machines):**

Install the basic utilities and dependencies for postgresxl

* yum -y update
* yum install -y tar openssh-server openssh-clients openssh wget vim sudo

Do Yum for following packages, which install all the dependencies for postgresxl

* yum groupinstall -y "Development tools"
* yum -y install readline-devel zlib-devel

Create the user for the Postgres for simplicity we are creating the user as ‘postgres’

* adduser postgres
* passwd postgres
* usermod -aG wheel postgres

**Creating the folder structure for the installation**

Create the folder installation, let’s create the folder Damocles folder and install the postgres into that folder

* cd /opt/
* mkdir damocles
* cd damocles

Download and extract the postgresxl in this folder

* wget <https://nchc.dl.sourceforge.net/project/postgres-xl/Releases/Version_9.5r1/postgres-xl-9.5r1.4.tar.gz>
* tar -zxvf postgres-xl-9.5r1.4.tar.gz

Installation of the postgresxl - Now login as postgres user

* su postgres
* cd postgres-xl-9.5r1.4/
* ./configure
* make

Switch as root user

* make install

Switch back to postgres user

* cd /opt/damocles/postgres-xl-9.5r1.4/contrib/pgxc\_ctl
* make

Switch to root

* make install

This completes the installation of the postgresxl

Switch to postgres user

Add the environmental variables for the postgresxl in bashrc. Postgresxl is installed in /usr/local/pgsql/ open bashrc and the environmental variables

* vim ~/.bashrc
* #Postgres-XL enviromental variables starts
* export POST\_HOME=/usr/local/pgsql
* export PATH=$POST\_HOME/bin:$PATH
* #Postgres-xl enviromental variables ends
* source ~/.bashrc

Verify the environment variable

* echo $POST\_HOME

*output>>* /usr/local/pgsql

***Please repeat these procedure/installation in all machines***

**Passwordless ssh between the machines**

This will guide you to do the passwordless ssh between machine because postgres communicates between nodes by ssh so make sure when you do ssh it should not ask for password

* ssh-keygen -t rsa
* cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys
* chmod 0600 ~/.ssh/authorized\_keys
* service sshd start
* ssh localhost
* **Passwordless ssh between machines(this is demonstration between** [**serveradm@10.138.32.25**](mailto:serveradm@10.138.32.25) **to** [**serveradm@10.138.32.232**](mailto:serveradm@10.138.32.232)**) change the user and ip address for rest of machines**
* ssh serveradm@10.138.32.232 mkdir -p .ssh
* cat .ssh/id\_rsa.pub | ssh serveradm@10.138.32.232 'cat >> .ssh/authorized\_keys'
* ssh serveradm@10.138.32.232 "chmod 700 .ssh; chmod 640 .ssh/authorized\_keys"
* ssh [serveradm@10.138.32.232](mailto:serveradm@10.138.32.232)

Once the passwordless ssh is done, lets start the configuration of the postgresxl

**Configuring the Postgresxl for (GTM master, GTM Slave, GTM Proxy, Coordinator and Datanode)**

This configuration has to be done in GTM master not in all machine

* cd /opt/damocles/postgres-xl-9.5r1.4/contrib/pgxc\_ctl
* cp pgxc\_ctl\_conf\_part\_minimal pgxc\_ctl.conf
* vim pgxc\_ctl.conf

Add the below configuration lines

***#Data, Location and Other Configuration***

**pgxcOwner=postgres** #give user\_name created for postgres during the installation

**pgxcUser=$pgxcOwner**

**tmpDir=/tmp**

**localTmpDir=$tmpDir**

**configBackup=n**

**configBackupHost=pgxc-linker**

**configBackupDir=/opt/damocles/data/pgxc**

**configBackupFile=$configBackupDir/pgxc\_ctl.bak**

#give the path the data directory where data and coordinate nodes to be created.

Please create this folder structure before you make this configuration in all the machines

**dataDirRoot=/opt/damocles/data/DATA/pgxl/nodes**

***#GTM Master Configuration***

**gtmName=gtm** #gtm master name

**gtmMasterServer=10.138.32.25**

**gtmMasterPort=8080**

**gtmMasterDir=$dataDirRoot/gtm**

**gtmExtraConfig=none**

**gtmMasterSpecificExtraConfig=none**

***#GTM Slave Configuration***

**gtmSlave=y**

**gtmSlaveName=gtmSlave**

**gtmSlaveServer=10.138.32.27**

**gtmSlavePort=20002**

**gtmSlaveDir=$dataDirRoot/gtm\_slv**

***#GTM Proxy***

**gtmProxyDir=$dataDirRoot/gtm\_pxy**

**gtmProxy=y**

**gtmProxyNames=(gtm\_pxy1)**

**gtmProxyServers=(10.138.32.25)**

**gtmProxyPorts=(20101)**

**gtmProxyDirs=($gtmProxyDir.1)**

**gtmPxyExtraConfig=n**

***#Coordinators Configuration***

**coordMasterDir=$dataDirRoot/coord\_master**

**coordSlaveDir=/opt/damocles/data/coord\_slave**

**coordArchLogDir=/opt/damocles/data/coord\_archlog**

**coordNames=(coord1 coord2)** #give the name of the coordinators with space separated

**coordPosrts=(30001 30002)** #give the ports of the coordinators with space separated

**poolerPorts=(30011 30012)** #give the poolerports of the coordinators with space separated

**coordPgHbaEntries=(::1/128)**

**coordPgHbaEntries=(10.138.32.25/16)** #mention the GTM, GTM proxy, GTM Slave IP series

**coordMasterServers=(10.138.32.232 10.138.32.233)** #give the IP of the coordinators with space separated

**coordMasterDirs=($coordMasterDir.1 $coordMasterDir.2)** #give the directory of the coordinators with space separated

***#Datanode Configuration***

**datanodeMasterDir=$dataDirRoot/dn\_master**

**datanodeSlaveDir=$dataDirRoot/dn\_slave**

**datanodeArchLogDir=$dataDirRoot/datanode\_archlog**

**primaryDatanode=datanode\_1**

**datanodeNames=(datanode\_1 datanode\_2)**

**datanodePorts=(40001 40002)**

**datanodePoolerPorts=(40011 40012)**

**datanodePgHbaEntries=(::1/128)**

**datanodePgHbaEntries=(10.138.32.25/16)**

**datanodeMasterServers=(10.138.32.232 10.138.32.233)**

**datanodeMasterDirs=($datanodeMasterDir.1 $datanodeMasterDir.2)**

Save and run the postgresxl so that it will create the node folders for all GTMs.

Running the postgresxl cluster

* Pgx\_ctl
* init all

Once its folders are created move the configuration pg\_ctl.conf to $HOME/pgxc\_ctl

* cd /opt/damocles/postgres-xl-9.5r1.4/contrib/pgxc\_ctl
* mv pgxc\_ctl.conf /home/serveradm/pgxc\_ctl

Do the above configuration in GTM master machine only

GTM master configuration

* cd /opt/damocles/postgres-xl-9.5r1.4/src/gtm/main/
* cp gtm.conf.sample gtm.conf
* vim gtm.conf

**nodename = 'gtm'**

**listen\_addresses = '10.138.32.25'**

**port = 8080**

**startup = ACT**

**active\_host = '10.138.32.27'**

**active\_port = 20002**

* mv opt/damocles/postgres-xl-9.5r1.4/src/gtm/main/ gtm.conf /opt/damocles/data/DATA/pgxl/nodes/gtm

GTM proxy Configuration

* cd /opt/damocles/postgres-xl-9.5r1.4/src/gtm/proxy/
* cp gtm\_proxy.conf.sample gtm\_proxy.conf
* vim gtm\_proxy.conf

**nodename = 'gtm\_pxy1'**

**listen\_addresses = '10.138.32.25'**

**port = 20101**

**gtm\_host = '10.138.32.27'**

**gtm\_port = 20002**

* mv opt/damocles/postgres-xl-9.5r1.4/src/gtm/ proxy/ gtm\_proxy.conf /opt/damocles/data/DATA/pgxl/nodes/gtm\_pxy.1/

GTM Slave configuration

* cd /opt/damocles/postgres-xl-9.5r1.4/src/gtm/main/
* cp gtm.conf.sample gtm.conf
* vim gtm.conf

**nodename = 'gtm\_slv'**

**listen\_addresses = '10.138.32.27’**

**port = 20001**

* mv opt/damocles/postgres-xl-9.5r1.4/src/gtm/main/ gtm.conf /opt/damocles/data/DATA/pgxl/nodes/gtm

Creating the Master and Datanode Cluster

Create the for structure in the master and datanode, for instance lets consider below folder structure

/opt/damocles/data/DATA/pgxl/nodes/ coord\_master.1 (This for coordinator 1)

/opt/damocles/data/DATA/pgxl/nodes/dn\_master.1/(This for datanode 1)

Then run the below command

* initdb --nodename=coord1 -D /opt/damocles/data/DATA/pgxl/nodes/coord\_master.1/
* initdb --nodename= datanode\_1 -D /opt/damocles/data/DATA/pgxl/nodes/ dn\_master.1 /

Once the all configuration are in proper place, start the cluster again from GTM Master

* Pgx\_ctl
* init all

It will take some time to at the initialization. It will form supporting schema, tables during the initialization once it done use below command to check the status of the nodes

* monitor all

It will show the status as below

Running: gtm master

Running: gtm slave

Running: gtm proxy gtm\_pxy1

Running: coordinator master coord1

Running: coordinator master coord2

Running: datanode master datanode\_1

Running: datanode master datanode\_2

## Software path architecture

# Configuration

**Configuring the Postgresxl for (GTM master, GTM Slave, GTM Proxy, Coordinator and Datanode)**

This configuration has to be done in GTM master not in all machine

* cd /opt/damocles/postgres-xl-9.5r1.4/contrib/pgxc\_ctl
* cp pgxc\_ctl\_conf\_part\_minimal pgxc\_ctl.conf
* vim pgxc\_ctl.conf

Add the below configuration lines

***#Data, Location and Other Configuration***

**pgxcOwner=postgres** #give user\_name created for postgres during the installation

**pgxcUser=$pgxcOwner**

**tmpDir=/tmp**

**localTmpDir=$tmpDir**

**configBackup=n**

**configBackupHost=pgxc-linker**

**configBackupDir=/opt/damocles/data/pgxc**

**configBackupFile=$configBackupDir/pgxc\_ctl.bak**

#give the path the data directory where data and coordinate nodes to be created.

Please create this folder structure before you make this configuration in all the machines

**dataDirRoot=/opt/damocles/data/DATA/pgxl/nodes**

***#GTM Master Configuration***

**gtmName=gtm** #gtm master name

**gtmMasterServer=10.138.32.25**

**gtmMasterPort=8080**

**gtmMasterDir=$dataDirRoot/gtm**

**gtmExtraConfig=none**

**gtmMasterSpecificExtraConfig=none**

***#GTM Slave Configuration***

**gtmSlave=y**

**gtmSlaveName=gtmSlave**

**gtmSlaveServer=10.138.32.27**

**gtmSlavePort=20002**

**gtmSlaveDir=$dataDirRoot/gtm\_slv**

***#GTM Proxy***

**gtmProxyDir=$dataDirRoot/gtm\_pxy**

**gtmProxy=y**

**gtmProxyNames=(gtm\_pxy1)**

**gtmProxyServers=(10.138.32.25)**

**gtmProxyPorts=(20101)**

**gtmProxyDirs=($gtmProxyDir.1)**

**gtmPxyExtraConfig=n**

***#Coordinators Configuration***

**coordMasterDir=$dataDirRoot/coord\_master**

**coordSlaveDir=/opt/damocles/data/coord\_slave**

**coordArchLogDir=/opt/damocles/data/coord\_archlog**

**coordNames=(coord1 coord2)** #give the name of the coordinators with space separated

**coordPosrts=(30001 30002)** #give the ports of the coordinators with space separated

**poolerPorts=(30011 30012)** #give the poolerports of the coordinators with space separated

**coordPgHbaEntries=(::1/128)**

**coordPgHbaEntries=(10.138.32.25/16)** #mention the GTM, GTM proxy, GTM Slave IP series

**coordMasterServers=(10.138.32.232 10.138.32.233)** #give the IP of the coordinators with space separated

**coordMasterDirs=($coordMasterDir.1 $coordMasterDir.2)** #give the directory of the coordinators with space separated

***#Datanode Configuration***

**datanodeMasterDir=$dataDirRoot/dn\_master**

**datanodeSlaveDir=$dataDirRoot/dn\_slave**

**datanodeArchLogDir=$dataDirRoot/datanode\_archlog**

**primaryDatanode=datanode\_1**

**datanodeNames=(datanode\_1 datanode\_2)**

**datanodePorts=(40001 40002)**

**datanodePoolerPorts=(40011 40012)**

**datanodePgHbaEntries=(::1/128)**

**datanodePgHbaEntries=(10.138.32.25/16)**

**datanodeMasterServers=(10.138.32.232 10.138.32.233)**

**datanodeMasterDirs=($datanodeMasterDir.1 $datanodeMasterDir.2)**

# Connection Check

## How to check network connection between two components?

## How to check network session?

## How to check service port?

## How to check software component and architecture?

# Operation and Maintenance

## Operation Commands

Below are the list of commands for basic operation:

* ***psql -p*** <port\_no> ***-h*** <host\_ip> ***-U*** </user\_name> ***-W***<password> ***-d*** <dbanme>: to connect to the database
* ***\l*** : List the database
* ***\c*** dbname : Connect to the database dbname
* ***\dt*** : List only tables
* ***\d*** : List all tables and sequences
* ***\d+***: List all tables and sequences with size
* ***\d*** : tablename : Describe the table tablename
* ***\h*** : List all the commands available
* ***\n*** : command : Give the description, syntax of that command
* ***\?*** : list all the operations commands with description
* ***\q***: quit the database

## Operation Tools

**--SQL Dump**

**pg\_dump**

* Dump is used to create the file with all the psql commands, when fed back to the server, will recreate the database in the same state as it was while taking the dump. Basically dump is used to create the backup of the server.
* pg\_dump doesn't block the other operations of the database
* psql programs are used to restore
* psql will execute even SQL error is encountered we can set the variable for script to stop when error is encountered

***psql -p <port\_no> postgres --set ON\_ERROR\_STOP=on dbname < outputfile.psql***

* Alternatively you can restore the whole dump as one single transaction by setting -1 or --single-transaction

***psql -p <port\_no> postgres -1 < outputfile.psql OR psql -p <port\_no> postgres --single-transaction < outputfile.psql***

**But this will rollback entire database restore, if very small error occurs**

* You take the dump and directly write into other server by using the pipes

pg\_dump -p <port\_no> postgres | psql -p <port\_no> postgres

***Important:*** The dumps produced by pg\_dump are relative to template0. This means that any languages, procedures, etc. added via template1 will also be dumped by pg\_dump. As a result, when restoring, if you are using a customized template1, you must create the empty database from template0, as in the example above.

* After restoring, run the ANALYZE on each database so the query optimizer has the useful statistics

**pg\_dumpall**

* pg\_dump is used dump only one database at single time, and doesn't dump the information aboubt the roles and tablespace
* For convenient dumping of all the entire contents of a database cluster we use pg\_dumpall

***pg\_dumpall -p <port\_no> > outputfile.psql***

**Handling the larger database**

* Use compressed dumps

***pg\_dump -p <port\_no> postgres | gzip > outputfile.psql.gz***

***pg\_dumpall -p <port\_no> | gzip > outputfile.psql.gz***

* Restoring

***gunzip -c outputfile.psql.gz | psql -p <port\_no> postgres***

**Using splits**

***pg\_dumpall -p <port\_no> | split -b 1m - outputfile***

This will split outputfile in 1Mb

Restoring

***cat outputfile\* | psql -p <port\_no>***

**Parallel dump Feature**

* To speed up the dump for larger database, you can use pg\_dump parallel mode. This will dump multiple tables at a same time
* You can control the degres of parallelism with -j parameter. Parallel dumps are only supported for the "directory" archive format

***pg\_dump -p <port\_no> -j num -F d -f outputfile.psql postgres***

* To restore you can use the

***pg\_restore -j***

**--Taking the backup of the Cluster and the Database**

* The better approach for taking the backup is to stop all the request for comming to the cluster and use pg\_dumpall.
* One can still take the backup with request comming to database but dump will not update the database.
* One can edit pg\_hba conf and change the permission of the other users, except yours.

***syntax:***

***pg\_dumpall -p <port\_no> postgres > outputfile.psql***

***Example:***

***pg\_dumpall -p 30001 postgres > alltradebackup.psql***

* This is advisable to install the new version of the Postgresxl and take the dump parallely, this decrease the downtime
* Complete the installation and transfer the data normally.
* Stop the older version of the cluster.

***pg\_ctl stop***

* While restoring the data into new version, rename the previous installation directory or you can delete the previous version, recommended to rename the directory

***mv /usr/local/pgsql /usr/local/pgsql.old***

* Use the command to restore the older version data into new one

***psql -p <port\_no> postgres -f outputfile.psql***

Using new psql

**Second approach**

The least downtime can be achieved by installing the new version in different directory and running the older and new server both in parallely

***pg\_dumpall -p 30001 | psql -p 16633 postgres***

to transfer the data

**Check the database size, table size**

* The query below gives the size of all databases

***SELECT***

***pg\_database.datname,***

***pg\_size\_pretty(pg\_database\_size(pg\_database.datname)) AS size***

***FROM pg\_database;***

* The query below gives the size of the all\_trade bases

***SELECT***

***pg\_database.datname,***

***pg\_size\_pretty(pg\_database\_size(pg\_database.datname)) AS size***

***FROM pg\_database where pg\_database.datname = ‘all\_trade’ ;***

* The query below gives the size of all tables

***SELECT***

***relname as "Table",***

***pg\_size\_pretty(pg\_total\_relation\_size(relid)) As "Size",***

***pg\_size\_pretty(pg\_total\_relation\_size(relid) - pg\_relation\_size(relid)) as "External Size"***

***FROM pg\_catalog.pg\_statio\_user\_tables ' ORDER BY pg\_total\_relation\_size(relid) DESC;***

* The below query will give the size of the local\_service\_requests\_new8 table

***SELECT***

***relname as "Table",***

***pg\_size\_pretty(pg\_total\_relation\_size(relid)) As "Size",***

***pg\_size\_pretty(pg\_total\_relation\_size(relid) - pg\_relation\_size(relid)) as "External Size"***

***FROM pg\_catalog.pg\_statio\_user\_tables where relname = 'local\_service\_requests\_new8' ORDER BY pg\_total\_relation\_size(relid) DESC;***

**Operation Commands:**