# 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)

### pgxc\_ctl.conf

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)**

#### Datanode Slave Configuration

**datanodeSlave=y**

**datanodeSlaveServers=(10.138.32.77 10.138.32.80)**

**datanodeSlavePorts=(40101 40102)**

**datanodeSlavePoolerPorts=(40111 40112)**

**datanodeSlaveSync=y**

**datanodeSlaveDirs=($datanodeSlaveDir.1 $datanodeSlaveDir.2)**

**datanodeArchLogDirs=( $datanodeArchLogDir.1 $datanodeArchLogDir.2)**

Save and exit the pgxc\_ctl.conf

#### Initializing the cluster

* pgxc\_ctl
* init all

#### Start the nodes

* start all

#### Verify the cluster

* monitor all

Running: gtm master

Running: gtm slave

Running: gtm\_proxy gtm\_proxy\_2

Running: gtm\_proxy gtm\_proxy\_1

Running: coordinator master coord1

Running: coordinator master coord2

Running: coordinator master coord3

Running: datanode master datanode\_1

Running: datanode slave datanode\_1

Running: datanode master datanode\_2

Running: datanode slave datanode\_2

Running: datanode master datanode\_3

Running: datanode slave datanode\_3

## Verifying the cluster

By creating the database, table, insert and select, in any one of the coordinator access the using ***psql***

***psql –p 30001 postgres***

***postgres=# select \* from pgxc\_node;***

***node\_name | node\_type | node\_port | node\_host | nodeis\_primary | nodeis\_preferred | node\_id***

***------------+-----------+-----------+---------------+----------------+------------------+-------------***

***coord1 | C | 30001 | 10.138.32.212 | f | f | 1885696643***

***coord2 | C | 30002 | 10.138.32.213 | f | f | -1197102633***

***coord3 | C | 30003 | 10.138.32.214 | f | f | 1638403545***

***datanode\_3 | D | 40003 | 10.138.32.214 | f | f | 1787525382***

***datanode\_1 | D | 40001 | 10.138.32.212 | f | f | -675012441***

***datanode\_2 | D | 40002 | 10.138.32.213 | f | t | -1047623914***

***postgres=# create database testcluster;***

***CREATE DATABASE***

***postgres=# \c testcluster***

***You are now connected to database "testcluster" as user "serveradm".***

***testcluster=# create table testdb (id int, name text);***

***CREATE TABLE***

***testcluster=# insert into testdb (id, name) select generate\_series(1,5),'foo';***

***INSERT 0 5***

***testcluster=# select \* from testdb ;***

***id | name***

***----+------***

***3 | foo***

***5 | foo***

***1 | foo***

***4 | foo***

***2 | foo***

***(5 rows)***

Here we go Cluster is up and running

# Cluster Management

Use pgxc\_ctl, the simple way to manage the postgres cluster below are the pgxc commands to add the nodes and remove the node

## Add GTM Master

***add gtm master gtm <ip> <port> <dataDir>*** //Syntax

***PGXC# add gtm master gtm 10.138.32.25 20001 $dataDirRoot/gtm***  //Example

## Add GTM Slave

***add gtm slave < gtm\_slave\_name> <ip> <port> <dataDir>***  //Syntax

***PGXC# add gtm slave gtm\_slave 10.138.32.25 20001 $dataDirRoot/gtm*** //Example

## Add GTM Proxy

***add gtm\_proxy <gtm\_proxy name> <ip> <port> <dataDir>*** //Syntax

***PGXC# add gtm master gtm 10.138.32.25 20001 $dataDirRoot/gtm*** //Example

## Add Coordinator Master

***add coordinator master <nodename> <nodeip> <nodeport> <nodepoolerport> <nodedatadir> extraconfig extrapghbaconfig*** //Syntax

***PGXC# add coordinator master coord2 localhost 30002 30012 $dataDirRoot/coord\_master.2 none none*** //Example

## Add Datanode Master

***add datanode master <nodename> <nodeip> <nodeport> <nodepoolerport> <nodedatadir> WALdir extraconfig extrapghbaconfig*** //Syntax

***PGXC# add datanode master dn3 localhost 40003 40013 $dataDirRoot/dn\_master.3 none none none*** //Example

## Add Datanode Slave

***add datanode slave <node\_name> <nodeip> <node\_port> <nodepoolerport> <nodedatadir> <WALArchdir> <nodearchlogdir>*** //Syntax

***PGXC# add datanode slave dn1 localhost 40101 40111 $dataDirRoot/dn\_slave.1 none $dataDirRoot/datanode\_archlog.1*** //Example

When Datanode Master is add to the cluster you add the table to the new datanode by simple alter query as below:

***<dbname>=# ALTER <TABLENAME> ADD NODE (dataNode\_master\_name);***

The above query will be distribute in the new datanode

***<dbname>=#SELECT xc\_node\_id, COUNT(\*) FROM <table\_name> GROUP BY xc\_node\_id;***

Run the above query to check the distribution of the table in the different data nodes

## Remove GTM Master

***remove gtm master <node\_name> clean***

This will delete the folder from node

## Remove GTM Slave

***remove gtm slave <node\_name> clean***

This will delete the folder from node

## Remove GTM Proxy

***remove gtm\_proxy <node\_name> clean***

This will delete the folder from node

## Remove Coordinator Master

***remove coordinator master <node\_name> clean***

This will delete the folder from node

## Remove Datanode Master

Before removing the any datanode master, first delete or remove the data from that particular node by running the below query  
***<dbname>=# ALTER <TABLENAME> DELETE NODE (dataNode\_master\_name);***The above query will redistribute the table into remaining datanode master  
***<dbname>=#SELECT xc\_node\_id, COUNT(\*) FROM <table\_name> GROUP BY xc\_node\_id;***

Run the above query to check the distribution of the table in the different data nodes ***remove datanode Master <node\_name> clean***This will delete the folder from node

## Remove Datanode Slave

***remove datanode Slave <node\_name> clean***

This will delete the folder from node

# Backup and Restoring

Use pg\_dump for taking the backup and pg\_restore to restore the dump

## Database

### Dump

***Syntax:***

***pg\_dump -p <port\_no> -U <user/role\_name> -F c -b -v -f "~FILE\_PATH/file\_name" <database\_name>***

***Example:***

***pg\_dump -p 30001 -U serveradm -F c -b -v -f "/tmp/alltrade" alltrade***

### Restoring

***Syntax:***

***pg\_restore -p <port\_no> -U <user/role\_name> -d <database\_name> -v "/tmp/alltrade"***

***Example:***

***pg\_restore -p 30001 -U serveradm -d pgresttest -v "/tmp/alltrade"***

## Node

Taking the dump of entire node(pg\_dumpall): pg\_dumpall uses pg\_dump internally to take the dump of database. You can take the dump of nodes. This will help to take the dump of entire node and create the node manually.

### Dump

***Syntax:***

***pg\_dumpall -p <port\_no> -s --include-nodes --dump-nodes > '~FILE\_PATH/file\_name'***

***Example:***

***pg\_dumpall -p 30001 -s --include-nodes --dump-nodes > '/tmp/coord1'***

### Restoring

***Syntax:***

***pg\_restore -p <port\_no> '~FILE\_PATH/file\_name'***

***Example:***

***pg\_restore -p <port\_no> '/tmp/coord1'***

## Data

Use the pg\_dump with the following options:

The below command give the dump of the full database with data

### Database

***Syntax:***

**pg\_dump -p <port\_no> --dbname=<database\_name> '~FILE\_PATH/file\_name'**

***Example:***

**pg\_dump -p 30001 --dbname=alltrade > '/tmp/delete.psql'**

### Table

The below command give the dump of the table

***Syntax:***

**pg\_dump -p <port\_no> --dbname=<database\_name> --table=<table\_name> > '~FILE\_PATH/file\_name'**

***Example:***

**pg\_dump -p 30001 --dbname=alltrade --table=app\_config > '/tmp/delete.psql'**

## Schema

### Database

The below command give the dump of the full schema without data

***Syntax:***

**pg\_dump -p <port\_no> --dbname=<database\_name> --schema-only > '~FILE\_PATH/file\_name'**

***Example:***

**pg\_dump -p 30001 --dbname=alltrade --schema-only > '/tmp/delete.psql'**

### Table

The below command give the dump of the full schema without data

***Syntax:***

**pg\_dump -p <port\_no> --dbname=<database\_name> --table=<table\_name> --schema-only > '~FILE\_PATH/file\_name'**

***Example:***

**pg\_dump -p 30001 --dbname=alltrade --table=app\_config --schema-only > '/tmp/delete.psql'**

## Restoring

**You can use pg\_restore as mentioned above or you can access the postgres and run the dump as script as shown below**

***postgres=#\i ‘*~FILE\_PATH/file\_name*’***

# Direct Access

Configure the ph\_hba.conf, before you access directly to the database for the client\_ip, user and database

## PSQL

***syntax:***

***psql -h <host\_ip> -p <port\_no> -U <user/role\_name> -W <database\_name>***

***Example***

***psql -h 10.138.32.212 -p 30001 -U admin\_alltrade -W all\_trade***

# Basic Operation with psql

***+ --> give the additional information***

***\l[+] --> list databases***

***\du[+] --> list users***

***\dv[+] --> list views***

***\df[+] --> list Store Procedure/Functions***

***\sf <function\_name> --> give the function definition***

***\dt[+] --> list the tables***

***\ds[+] --> list the sequences***

***\dL[+] --> list the procedural langiage***

***\di[+] --> list the indexes***

***\d[+] --> List of relations***

***\x [on|off|auto] --> toggle expanded output (currently off)***

***\a --> allign and unalign output mode***

***\f[STRING] --> separator between columns for unalign output***

## USER NAME and Password

### CREATE

***Syntax:***

***CREATE USER <username> <Attributes>;***

***Atribute:***

***ADMIN role\_name***

***The ADMIN clause is like ROLE, but the named roles are added to the new role WITH ADMIN OPTION, giving them the right to grant membership in this role to others.***

***NOSUPERUSER***

***SUPERUSER***

***These clauses determine whether the new role is a "superuser", who can override all access restrictions within the database. Superuser status is dangerous and should be used only when really needed. You must yourself be a superuser to create a new superuser. If not specified, NOSUPERUSER is the default.***

***CREATEROLE***

***NOCREATEROLE***

***These clauses determine whether a role will be permitted to create new roles (that is, execute CREATE ROLE). A role with CREATEROLE privilege can also alter and drop other roles. If not specified, NOCREATEROLE is the default.***

***REPLICATION***

***NOREPLICATION***

***These clauses determine whether a role is allowed to initiate streaming replication or put the system in and out of backup mode. A role having the REPLICATION attribute is a very highly privileged role, and should only be used on roles actually used for replication. If not specified, NOREPLICATION is the default for all roles except superusers.***

***CREATEDB***

***NOCREATEDB***

***These clauses define a role's ability to create databases. If CREATEDB is specified, the role being defined will be allowed to create new databases. Specifying NOCREATEDB will deny a role the ability to create databases. If not specified, NOCREATEDB is the default.***

***LOGIN***

***NOLOGIN***

***These clauses determine whether a role is allowed to log in; that is, whether the role can be given as the initial session authorization name during client connection. A role having the LOGIN attribute can be thought of as a user. Roles without this attribute are useful for managing database privileges, but are not users in the usual sense of the word. If not specified, NOLOGIN is the default, except when CREATE ROLE is invoked through its alternative spelling CREATE USER.***

***CREATEUSER***

***NOCREATEUSER***

***These clauses are an obsolete, but still accepted, spelling of SUPERUSER and NOSUPERUSER. Note that they are not equivalent to CREATEROLE as one might naively expect!***

***ENCRYPTED***

***UNENCRYPTED***

***These key words control whether the password is stored encrypted in the system catalogs. (If neither is specified, the default behavior is determined by the configuration parameter password\_encryption.) If the presented password string is already in MD5-encrypted format, then it is stored encrypted as-is, regardless of whether ENCRYPTED or UNENCRYPTED is specified (since the system cannot decrypt the specified encrypted password string). This allows reloading of encrypted passwords during dump/restore.***

***Note that older clients might lack support for the MD5 authentication mechanism that is needed to work with passwords that are stored encrypted.***

***ROLE role\_name***

***The ROLE clause lists one or more existing roles which are automatically added as members of the new role. (This in effect makes the new role a "group".)***

***VALID UNTIL 'timestamp'***

***The VALID UNTIL clause sets a date and time after which the role's password is no longer valid. If this clause is omitted the password will be valid for all time.***

***CONNECTION LIMIT connlimit***

***If role can log in, this specifies how many concurrent connections the role can make. -1 (the default) means no limit.***

***INHERIT***

***NOINHERIT***

***These clauses determine whether a role "inherits" the privileges of roles it is a member of. A role with the INHERIT attribute can automatically use whatever database privileges have been granted to all roles it is directly or indirectly a member of. Without INHERIT, membership in another role only grants the ability to SET ROLE to that other role; the privileges of the other role are only available after having done so. If not specified, INHERIT is the default.***

***PASSWORD***

***Sets the role's password. (A password is only of use for roles having the LOGIN attribute, but you can nonetheless define one for roles without it.) If you do not plan to use password authentication you can omit this option. If no password is specified, the password will be set to null and password authentication will always fail for that user. A null password can optionally be written explicitly as PASSWORD NULL.***

***Example:***

***create user mahantesh SUPERUSER LOGIN CREATEDB PASSWORD 'mahantesh'CREATEROLE ;***

### Change Password

***Change password for the user***

***Syntax:***

***\password <username>***

***Example:***

***all\_trade=# \password mahantesh***

***Enter new password:***

***Enter it again:***

### Alter

***Syntax:***

***alter role <user\_name> <attribute>;***

***Example:***

***all\_trade=# alter role mahantesh CREATEDB;***

***ALTER ROLE***

### Delete

***Deleting the user***

***Syntax:***

***DROP USER <user\_name>***

***Example:***

***all\_trade=# drop user mahantesh ;***

***DROP ROLE***