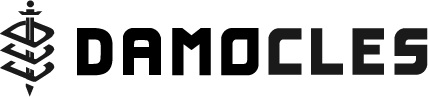
****

**DAMOCLES**

**O & M Guide**

# Contents

Contents

[Contents 2](#_Toc492037569)

[1. Overview 4](#_Toc492037570)

[Architecture & Concept 4](#_Toc492037571)

[2. Installation 5](#_Toc492037572)

[Installation & Configuration 5](#_Toc492037573)

[Software path architecture 6](#_Toc492037574)

[3. Configuration 7](#_Toc492037575)

[Configuration file and description 7](#_Toc492037576)

[Configuration update 8](#_Toc492037577)

[Configuration compare and verification 9](#_Toc492037578)

[4. Connection check 10](#_Toc492037579)

[How to check network connection between component 10](#_Toc492037580)

[How to check network session 11](#_Toc492037581)

[How to check service port 12](#_Toc492037582)

[How to check software component and architecture 13](#_Toc492037583)

[5. Operation & Maintenance 14](#_Toc492037584)

[Operation command 14](#_Toc492037585)

[Operation tools 15](#_Toc492037586)

[How to check status of each component 16](#_Toc492037587)

[How to check database size 17](#_Toc492037588)

[How to check performance of each component (respond time , transaction per second etc) 17](#_Toc492037589)

[How to check data sync between data node in the cluster 19](#_Toc492037590)

[How to get and verify data in the database 20](#_Toc492037591)

[Log description and analyzing 21](#_Toc492037592)

[Stat description and analyzing 22](#_Toc492037593)

[Alarm description and analyzing 23](#_Toc492037594)

[Binary version check 24](#_Toc492037595)

[Configuration version check 25](#_Toc492037596)

[How to monitor software component workload 26](#_Toc492037597)

[How to recovery node in case IP change 26](#_Toc492037598)

[How to change node member name in cluster 27](#_Toc492037599)

[Data management on D11 28](#_Toc492037600)

[Log file rotation and log management. 28](#_Toc492037601)

[6. Software upgrade 29](#_Toc492037602)

[How to check package version 29](#_Toc492037603)

[Package version update 30](#_Toc492037604)

[How to check compatible between binary and configuration file 30](#_Toc492037605)

[7. Schema 31](#_Toc492037606)

[Online Update Schema 31](#_Toc492037607)

[Offline Update Schema 32](#_Toc492037608)

[Schema compare and verification 33](#_Toc492037609)

[8. Database 34](#_Toc492037610)

[Header provisioning and verification 34](#_Toc492037611)

[How to restore database from other node in the same cluster 34](#_Toc492037612)

[Database backup and restore 35](#_Toc492037613)

[Database Daily backup 36](#_Toc492037614)

[Direct access to database 36](#_Toc492037615)

[Test query data via D01/D02 37](#_Toc492037616)

[9. Expansion 38](#_Toc492037617)

[Add new cluster 38](#_Toc492037618)

[Add new component 38](#_Toc492037619)

[Delete cluster 39](#_Toc492037620)

[Delete component 40](#_Toc492037621)

# Overview

## Architecture & Concept

|  |
| --- |
| Document Purpose: This document is created with the purpose:   * Brief Introduction to the Postgres XL * Architecture * Installation of the application * Configuring * Commonly faced issues  1.2 Brief Introduction : Postgres-XL is a horizontally scalable open source SQL database cluster, flexible enough to handle varying database workloads:   * OLTP write-intensive workloads * Business Intelligence requiring MPP parallelism * Operational data store * Key-value store * GIS Geospatial * Mixed-workload environments * Multi-tenant provider hosted environments    1.3 Product Architecture:GTM Keeps track of all the transactions and has responsibility of cluster wide tracking of database primitives like mutexes and transactions. This node is used by both Data Nodes and Coordinator nodes. This is a Single Point Of Failure of failure as it is only a single node. For this reason there is a GTM backup put in place. GTM Backup GTM Failover is a node that handles the switch between the GTM and GTM Backup in case of a failure condition. Coordinator Nodes These nodes are what the users connect to, to run the actual queries. These are essentially postgres instances, so any tool that can connect to a postgres database will also work on these. Users can connect to any coordinator node to run any queries on the postgresxl cluster.  A coordinator node keeps track of what data resides on what machine, makes the query plan and manages the execution of the query and returns the result. A coordinator node does not save any data by itself so even one of the nodes crashes, it does not impact the functioning of the cluster. Data Nodes These nodes save the actual data, these are also postgres instances but these would not be directly connected by anyone to query data, these nodes will be queried by the coordinator nodes. Distribute By Hash For tables that do not or in the future are expected to not fit into one machine will be distributed across several machines. Data of one row would go into one machine. To decide what machine the data would need to go into a hash of one of the pre-decided columns is taken. Based on the result of the hash the data is placed into a particular node. Distribute By Replication This is used for tables that can easily fit into one server and are not expected to grow substantially in the future. The same data is replicated to all the nodes. The benefit of this is that any node can be used to query this data. Certain use cases where the table is small but is updated very frequently is when you may not want to use this method. Replicated Slaves Because there is not built in redundancy like cassandra where 2 - 3 copies of data are maintained here the redundancy is maintained by maintaining replicated nodes for each data node. |

# Installation

## Installation & Configuration

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hardware Requirement**  The Damocles System are required the any CPU family that supported instruction set as below   * AMD64 is the 64 bit version of the x86 instruction set. * ASE-NI is a specification for the encryption of electronic data. * POPCNT is a population count instruction.   **Software Requirement**   |  |  |  |  | | --- | --- | --- | --- | | **Software Name** | **Version** | **Description** | **M/O** | | Linux kernel | 2.6.32 | Linux operation system (for el6) | M |   **Package Name: PostgresXl 9.5r1.4**  **Prerequisites: Install Development Tools, readline-devel, zlib-devel, Passwordless ssh between all Machines, Create the separate user for Postgres**  **Installation Steps:**  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 $POST\_HOME   *output>>* /usr/local/pgsql  ***Please repeat these procedure/installation in all machines*** Configuring the Postgresxl for (GTM master, GTM Slave, GTM Proxy, Coordinator and Datanode)pgxc\_ctl.conf This configuration has to be done in GTM master not in all machine   * cd /opt/damocles/postgres-xl-9.5r1.4/contrib/pgxc\_ctl * cp pgxc\_ctl\_conf\_part\_minimal pgxc\_ctl.conf * vim pgxc\_ctl.conf   Add the below configuration lines Data, Location and Other Configuration **pgxcOwner=postgres** #give user\_name created for postgres during the installation  **pgxcUser=$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 |

## Software path architecture

# Configuration

## Configuration file and description

**3.1 Configuration**

*Configuration path:* /opt/damocles/conf/

**Default configuration:**

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Example** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Configuration update

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Configuration compare and verification

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

# Connection check

## How to check network connection between component

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
| 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 |

**Post Test:**

|  |
| --- |
|  |

## How to check network session

**Objective:**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| Node should not ask the password when we do ssh, do telnet between the nodes to check the connection between nodes. Please configure the iptables with node IP and Port for secured connection between the nodes. |

**Post Test**

|  |
| --- |
|  |

## How to check service port

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to check software component and architecture

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| Use pgxc\_ctl to check the components,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  Run the normal linux command **ps aux | grep postgres** to check the components  After please check the entire cluster by creating the database, table, insert, select.  ***psql –p <node\_port> 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***  The above query will show the node details.  Now we create the database, table, insert and select  ***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 |

**Post Test**

|  |
| --- |
|  |

# Operation &Maintenance

## Operation command

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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  CREATING the 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*** |

**Post Test**

|  |
| --- |
|  |

## Operation tools

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | pg\_stat\_database | [One row per database, showing database-wide statistics. See pg\_stat\_database for details.](https://www.postgresql.org/docs/9.6/static/monitoring-stats.html#PG-STAT-DATABASE-VIEW) | | pg\_stat\_database\_conflicts | [One row per database, showing database-wide statistics about query cancels due to conflict with recovery on standby servers. See pg\_stat\_database\_conflicts for details.](https://www.postgresql.org/docs/9.6/static/monitoring-stats.html#PG-STAT-DATABASE-CONFLICTS-VIEW) | | pg\_stat\_user\_tables | Same as pg\_stat\_all\_tables, except that only user tables are shown. | | pg\_stat\_user\_indexes | Same as pg\_stat\_all\_indexes, except that only indexes on user tables are shown. | | pg\_statio\_user\_tables | Same as pg\_statio\_all\_tables, except that only user tables are shown. | | pg\_statio\_user\_indexes | Same as pg\_statio\_all\_indexes, except that only indexes on user tables are shown. | | pg\_statio\_user\_sequences | Same as pg\_statio\_all\_sequences, except that only user sequences are shown. |  |  |  |  | | --- | --- | --- | | **Column** | **Type** | **Description** | | datid | oid | OID of the database this backend is connected to | | datname | name | Name of the database this backend is connected to | | pid | integer | Process ID of this backend | | usesysid | oid | OID of the user logged into this backend | | usename | name | Name of the user logged into this backend | | application\_name | text | Name of the application that is connected to this backend | | client\_addr | inet | IP address of the client connected to this backend. If this field is null, it indicates either that the client is connected via a Unix socket on the server machine or that this is an internal process such as autovacuum. | | client\_hostname | text | [Host name of the connected client, as reported by a reverse DNS lookup of client\_addr. This field will only be non-null for IP connections, and only when log\_hostname is enabled.](https://www.postgresql.org/docs/9.6/static/runtime-config-logging.html#GUC-LOG-HOSTNAME) | | client\_port | integer | TCP port number that the client is using for communication with this backend, or -1 if a Unix socket is used | | backend\_start | timestamp with time zone | Time when this process was started, i.e., when the client connected to the server | | xact\_start | timestamp with time zone | Time when this process' current transaction was started, or null if no transaction is active. If the current query is the first of its transaction, this column is equal to the query\_start column. | | query\_start | timestamp with time zone | Time when the currently active query was started, or if state is not active, when the last query was started | | state\_change | timestamp with time zone | Time when the state was last changed | | wait\_event\_type | text | The type of event for which the backend is waiting, if any; otherwise NULL. Possible values are: | | LWLockNamed: The backend is waiting for a specific named lightweight lock. Each such lock protects a particular data structure in shared memory.wait\_event will contain the name of the lightweight lock. | |  | | LWLockTranche: The backend is waiting for one of a group of related lightweight locks. All locks in the group perform a similar function; wait\_eventwill identify the general purpose of locks in that group. | |  | | Lock: The backend is waiting for a heavyweight lock. Heavyweight locks, also known as lock manager locks or simply locks, primarily protect SQL-visible objects such as tables. However, they are also used to ensure mutual exclusion for certain internal operations such as relation extension. wait\_event will identify the type of lock awaited. | |  | | BufferPin: The server process is waiting to access to a data buffer during a period when no other process can be examining that buffer. Buffer pin waits can be protracted if another process holds an open cursor which last read data from the buffer in question. | |  | | wait\_event | text | [Wait event name if backend is currently waiting, otherwise NULL. See Table 28-4 for details.](https://www.postgresql.org/docs/9.6/static/monitoring-stats.html#WAIT-EVENT-TABLE) | | state | text | Current overall state of this backend. Possible values are: | | active: The backend is executing a query. | |  | | idle: The backend is waiting for a new client command. | |  | | idle in transaction: The backend is in a transaction, but is not currently executing a query. | |  | | idle in transaction (aborted): This state is similar to idle in transaction, except one of the statements in the transaction caused an error. | |  | | fastpath function call: The backend is executing a fast-path function. | |  | | [disabled: This state is reported if track\_activities is disabled in this backend.](https://www.postgresql.org/docs/9.6/static/runtime-config-statistics.html#GUC-TRACK-ACTIVITIES) | |  | | backend\_xid | xid | Top-level transaction identifier of this backend, if any. | | backend\_xmin | xid | The current backend's xmin horizon. | | query | text | Text of this backend's most recent query. If state is active this field shows the currently executing query. In all other states, it shows the last query that was executed. | |  |  |  |   Here are the few other tables gives stats   |  |  |  |  | | --- | --- | --- | --- | | pg\_aggregate | pg\_foreign\_table | pg\_seclabels | pg\_stat\_user\_functions | | pg\_am | pg\_group | pg\_settings | pg\_stat\_user\_indexes | | pg\_amop | pg\_index | pg\_shadow | pg\_stat\_user\_tables | | pg\_amproc | pg\_indexes | pg\_shdepend | pg\_stat\_xact\_all\_tables | | pg\_attrdef | pg\_inherits | pg\_shdescription | pg\_stat\_xact\_sys\_tables | | pg\_attribute | pg\_language | pg\_shseclabel | pg\_stat\_xact\_user\_functions | | pg\_authid | pg\_largeobject | pg\_stat\_activity | pg\_stat\_xact\_user\_tables | | pg\_auth\_members | pg\_largeobject\_metadata | pg\_stat\_all\_indexes | pg\_tables | | pg\_available\_extensions | pg\_locks | pg\_stat\_all\_tables | pg\_tablespace | | pg\_available\_extension\_versions | pg\_matviews | pg\_stat\_archiver | pg\_temp\_1. | | pg\_cast | pg\_namespace | pg\_stat\_bgwriter | pg\_timezone\_abbrevs | | pg\_catalog. | pg\_opclass | pg\_stat\_database | pg\_timezone\_names | | pg\_class | pg\_operator | pg\_stat\_database\_conflicts | pg\_toast. | | pg\_collation | pg\_opfamily | pg\_statio\_all\_indexes | pg\_toast\_temp\_1. | | pg\_constraint | pg\_pltemplate | pg\_statio\_all\_sequences | pg\_transform | | pg\_conversion | pg\_policies | pg\_statio\_all\_tables | pg\_trigger | | pg\_cursors | pg\_policy | pg\_statio\_sys\_indexes | pg\_ts\_config | | pg\_database | pg\_prepared\_statements | pg\_statio\_sys\_sequences | pg\_ts\_config\_map | | pg\_db\_role\_setting | pg\_prepared\_xacts | pg\_statio\_sys\_tables | pg\_ts\_dict | | pg\_default\_acl | pg\_proc | pg\_statio\_user\_indexes | pg\_ts\_parser | | pg\_depend | pg\_range | pg\_statio\_user\_sequences | pg\_ts\_template | | pg\_description | pg\_replication\_origin | pg\_statio\_user\_tables | pg\_type | | pg\_enum | pg\_replication\_origin\_status | pg\_statistic | pg\_user | | pg\_event\_trigger | pg\_replication\_slots | pg\_stat\_replication | pg\_user\_mapping | | pg\_extension | pg\_rewrite | pg\_stats | pg\_user\_mappings | | pg\_file\_settings | pg\_roles | pg\_stat\_ssl | pg\_views | | pg\_foreign\_data\_wrapper | pg\_rules | pg\_stat\_sys\_indexes |  | | pg\_foreign\_server | pg\_seclabel | pg\_stat\_sys\_tables |  |   https://www.postgresql.org/docs/9.6/static/monitoring-stats.html |

**Post Test**

|  |
| --- |
|  |

## How to check status of each component

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to check database size

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| To check the size of the 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 ORDER BY pg\_total\_relation\_size(relid) DESC;***  To check the size of the database  ***SELECT***  ***pg\_database.datname,***  ***pg\_size\_pretty(pg\_database\_size(pg\_database.datname)) AS size***  ***FROM pg\_database;*** |

**Post Test**

|  |
| --- |
|  |

## How to check performance of each component (respond time , transaction per second etc)

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to check data sync between data node in the cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to get and verify data in the database

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| To check the data sync and data distribution use  ***select count(\*), xc\_node\_id from <table\_name> group by xc\_node\_id;*** |

**Post Test**

|  |
| --- |
|  |

## Log description and analyzing

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| You can change the log setting of each node by editing the log parameters in postgressql.conf of each node |

**Post Test**

|  |
| --- |
|  |

## Stat description and analyzing

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Alarm description and analyzing

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Binary version check

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Configuration version check

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to monitor software component workload

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to recovery node in case IP change

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| First change the ip address in the pgxc\_ctl.conf and update the new ip on each node by  ***CREATE NODE <node\_name> WITH (TYPE='<node\_type>', HOST='<new\_ip>', PORT=<port\_no>);***  ***ALTER NODE <node\_name> WITH (TYPE='<node\_type>', HOST='<new\_ip>', PORT=<port\_no>);***  Second approach is remove the node before you change the ip address and after you change the ip, add the node with new configuration, this will create the nodes automatically during the initialization of the node. |

**Post Test**

|  |
| --- |
|  |

## How to change node member name in cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| Change the node name in the recently update configuration of the pgxc\_ctl.conf, start the pgxc\_ctl stop the node and start again |

**Post Test**

|  |
| --- |
|  |

## Data management on D11

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Log file rotation and log management.

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

# Software upgrade

## How to check package version

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Package version update

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## How to check compatible between binary and configuration file

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

# Schema

## Online Update Schema

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

## Offline Update Schema

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

## Schema compare and verification

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

# Database

## Header provisioning and verification

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
|  |

**Post Test:**

|  |
| --- |
|  |

## How to restore database from other node in the same cluster

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
| Use pg\_dump for taking the backup and pg\_restore to restore the 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 the dump  ***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"***  **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.**  **Taking 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'***  ***Syntax:***  ***pg\_restore -p <port\_no> '~FILE\_PATH/file\_name'***  ***Example:***  ***pg\_restore -p <port\_no> '/tmp/coord1'*** |

**Post Test**

|  |
| --- |
|  |

## Database backup and restore

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
| Use the pg\_dump with the following options:  The below command give the dump of the full database with data  ***Syntax:***  **pg\_dump -p <port\_no> --dbname=<database\_name> '~FILE\_PATH/file\_name'**  ***Example:***  **pg\_dump -p 30001 --dbname=alltrade > '/tmp/delete.psql'**  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'**  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'**  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'**  **Restoring the pg\_dump**  **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*’*** |

**Post Test:**

|  |
| --- |
|  |

## Database Daily backup

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
| **You can use any of the pg\_dump for taking the database backup** |

**Post Test:**

|  |
| --- |
|  |

## Direct access to database

**Objective:**

|  |
| --- |
| **1.** |

**Impact:**

|  |
| --- |
|  |

**Deployment Instruction:**

|  |
| --- |
| Configure the pg\_hba.conf, before you access directly to the database for the client\_ip, user and database  ***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***  ***Password for user admin\_alltrade:***  ***psql (PGXL 9.5r1.4, based on PG 9.5.5 (Postgres-XL 9.5r1.4))***  ***Type "help" for help.***  ***all\_trade=>*** |

**Post Test**

|  |
| --- |
|  |

## Test query data via D01/D02

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

# Expansion

## Add new cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Add new component

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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 |

**Post Test**

|  |
| --- |
|  |

## Delete cluster

**Objective**

|  |
| --- |
| **1.** |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
|  |

**Post Test**

|  |
| --- |
|  |

## Delete component

**Objective**

|  |
| --- |
|  |

**Impact**

|  |
| --- |
|  |

**Deployment Instruction**

|  |
| --- |
| 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 |

**Post Test**

|  |
| --- |
|  |