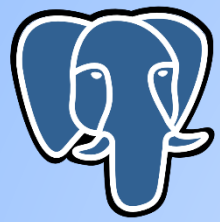
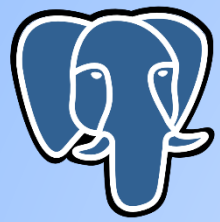


PostgreSQL

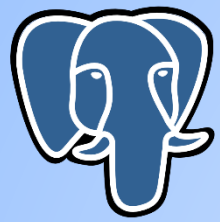
Module 13 (Part 2)

Replication



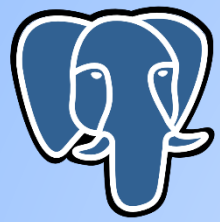
Terminology

- Fail over: A term which refers to using your secondary site in case there is a major failure at your primary site.
- Switchover: A term which refers to switching the roles of your Primary and seconday sites either because Primary site has come up again after a recent major failure or due to some maintenance work for which you have to switch the roles
- Warm standby: A standby server which is an exactly (almost real time) replica of your primary server and is always available in recovery mode and cannot be accessed until a recovery is triggered
- Hot Standby: A standby server which is an exactly (almost real time) replica of your primary server and is always available in recovery mode and cannot be accessed for write commands. But for read queries
- WAL Archive Replication: A replication process where the WAL segments which have been archived are copied and replayed on the standby server
- Streaming Replication: A replication process where the WAL segments are copied directly to the standby servers

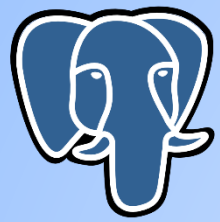


Replication Concepts

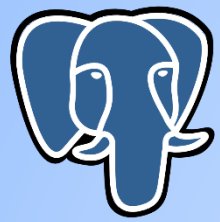
- Replication requires understanding, effort, and patience
- Database Replication is the technology used to maintain a copy of a set of data on a remote system
- Two main reasons for doing replication:
  - High Availability – Reducing the chances of data unavailability by having multiple systems, each holding a full copy of data
  - Data Movement - Allowing data to be used by additional applications or workload on additional hardware
- Focus is on High availability, where there is no transformation of the data – Copy data from one PostgreSQL database server to another



- Database servers are referred as nodes
- Whole group of database servers involved in replication is known as cluster, but be careful, as term “cluster” is used for separate meaning elsewhere in PostgreSQL
- A database server that allows a user to make changes is known as Master or Primary
- A database server that allows read-only access is known as Hot Standby or Slave

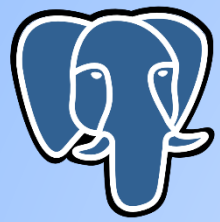


- **Physical Streaming Replication (PSR)**
  - Take the transaction log and ship that data to the remote node
  - PSR requires us to have only a single master node, though it allows multiple standbys
- **Logical Streaming Replication (LSR)**
  - An efficient mechanism for reading the transaction log (WAL) and transforming it into a stream of changes, a process known as logical decoding
  - LSR can be used as the basis of multimaster clusters



Practical aspects

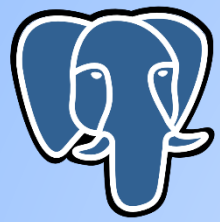
- Transfer of replicated data as "streaming“
- It may have downstream nodes that receive replicated data from it and/or upstream nodes that send data to it
- The time taken to transfer data changes to a remote node is usually referred to as the **latency**, or **replication delay**
- Changes must then be applied to the remote node, which takes an amount of time known as the **apply delay**
- The total time a record takes from the master to a downstream node is the replication delay plus the apply delay
- Either replication will copy all tables, or in some cases, we can copy a subset of tables, in which case we call it **selective replication**



Replication Best Practices

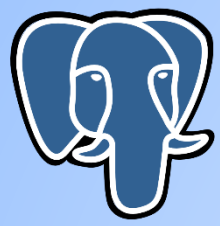
- Use similar hardware and OS on all systems – We may get performance issues, if we failover to different hardware
- Configure all systems identically as far as possible – Keep everything possible the same
  - Use same mount points
  - Use same directory names
  - Use same users
- Don't make one system more important than others in some way
- Give systems/servers good names to reduce confusion
- If one system fails, never reuse the same name of the old system i.e., pick another name

- Set the application_name parameter to be the server name in the replication connection string
- Keep the system clock synchronized
- Use single time-zone – Use Co-ordinated Universal Time (UTC)
  - Don't use the time with Daylight Saving Time
- Monitor each of database servers
- Monitor replication delay between servers
- Replications are useful if data is flowing correctly between servers
- Monitor the time it takes for data to go from one server to other



Hot Standby and Read Scalability

- Hot Standby - allows us to connect to a standby node and execute read-only queries
- Hot Standby allows to run queries while being continuously updated through streaming replication
- Multiple Hot Standby nodes can be added to scale the read-only workload. There is no hard limit, 10, 20 or more…
- Two main capabilities provided by Hot Standby node
  - It provides a secondary node in case the primary node fails
  - We can run queries on that node
- If we have more than one Hot Standby node, it may be possible to have one node nominated as standby and others dedicated to serving queries



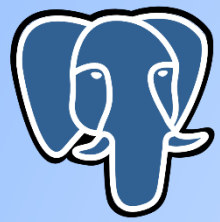
PostgreSQL

- Hot Standby is usable with following:
  - Streaming Replication
  - While performing a point in time recovery
- You need to configure additional parameters

```
On the master, set the following in postgresql.conf
wal_level = 'replica'

On the standby, set the following in postgresql.conf
hot_standby = on
```

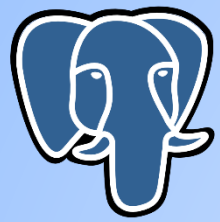
  - Now clean restart your database server on the master
  - Wait a few seconds, and restart the standby for these changes to take effect
  - Do not restart standby too quickly
- Configure this once, not every time you restart



Delaying, Pausing and Synchronizing Replication

- Delaying
  - In case of multiple standby servers – To have one or more servers operating in a delayed apply state e.g., 1 hour behind the master
- How to do it:
  - When you set the recovery_min_apply_delay parameter in recovery.conf, the application of commit records will be delayed by the specified duration
  - Only commit records are delayed
  - If some bad happen , Hot Standby allows you to pause and resume replay of changes

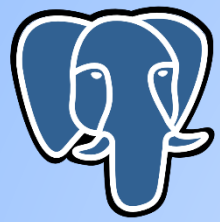
```
SELECT pg_xlog_replay_pause();
```



- To resume (un-pause) processing, use this query:

```
SELECT pg_xlog_replay_resume();
```

- Do not promote a delayed standby
- If your delayed standby is the last server available, you should reset recovery_min_apply_delay, then restart the server



Setting-Up Streaming Replication

- Streaming replication transfers WAL data directly from master to the standby
  - It integrates security, and
  - Reduces replication delay
- Streaming replication refers to the master node as the primary node; terms can be used interchangeably
- Two main ways to set up streaming replication
  - With additional archive
  - Without additional archive

How to do it…

- Carry out the following steps:

1. Identify master and standby nodes
2. Configure replication security: Create/Confirm replication user existence on master node

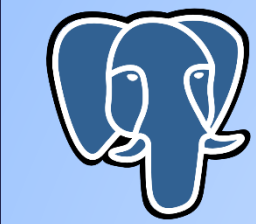
```
CREATE USER repuser SUPERUSER
        CONNECTION LIMIT 1
        ENCRYPTED PASSWORD 'root';
```

3. Allow replication user to authenticate. The following will allow access from any IP Address using md5-encrypted password authentication though pg_hba.conf file

```
Host    replication    repuser    127.0.0.1/0    md5
```

4. Set logging options in postgresql.conf on both master and standby nodes

```
log_connections = on
```



5. Set following parameters on master in postgresql.conf:

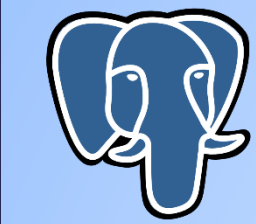
```
Max_wal_Senders = 2
Wal_level = 'replica'
Archive_mode = on
Archive_command = 'cd .'
```

6. Adjust wal_keep_segments on master in postgresql.conf

```
wal_keep_segments = 10000
```

7. Adjust Hot Standby parameters if required
8. Take a base backup; physical backup
   - Start the backup

```
psql -c "select pg_start_backup('base backup for streaming rep')"
```



- Copy the data files

```
rsync -cva --inplace --exclude=*pg_xlog* ${PGDATA}
$STANDBYNODE:$PGDATA
```

- Stop the backup

```
psql -c "select pg_stop_backup(), current_timestamp"
```

9. Set recovery.conf parameters on standby

```
standby_mode = 'on'
primary_conninfo = 'host=alpha user=repuser'
```

10. Start the standby server

11. Carefully monitor the replication delay until the catch up period is over

Setting-up Streaming Replication Security

- Streaming replication is at least as secure as normal user connections to PostgreSQL
- Standbys are identical copies of the master, so all users exist on all nodes with identical passwords
- All of the data is identical and all the permissions are the same too
- On the master, perform these steps:
  - Enable replication by setting a specific host access rule in pg_hba.conf
  - Give the selected replication user/role the REPLICATION attribute

```
ALTER ROLE replogin REPLICATION;
Alternatively, create it using this command:
CREATE ROLE replogin WITH REPLICATION;
```

- On the standby, perform these steps:
  - Request replication by setting primary_conninfo
  - Enable per-server rules, if any, for this server in pg_hba.conf

STREAMING REPLICATION-HOT STANDBY

- In this example we're going to be dealing with two hosts:
  Master 192.168.0.1

  Slave 192.168.0.2
- Step 1 : THE MASTER SETUP
  - The following are the minimum settings to get the master host ready for streaming replication:

```
-bash-4.2$ vi /9.4/data/postgresql.conf

#listen_addresses = 'localhost'                    # what IP address(es) to listen
on;
listen_addresses = '*'

#wal_level = minimal                               # minimal, archive, or hot_standby
wal_level = hot_standby

#max_wal_senders = 0
max_wal_senders = 3
```

STREAMING REPLICATION-HOT STANDBY

- Step 2
- Once you've made those changes you'll also need to create a replication user:

```
-bash-4.2$ psql
```

```
Postgres=# CREATE USER repuser WITH REPLICATION PASSWORD 'root';
```

- Step 3
- After the replication user has been created you'll need to allow it to connect:

```
-bash-4.2$ vi /9.4/data/pg_hba.conf
```

```
#rep
host    replication     repuser         192.168.0.2/32          md5
```

STREAMING REPLICATION-HOT STANDBY

- Step 4
- Now stop the service:

```
-bash-4.2$ /usr/pgsql-9.4/bin/pg_ctl stop
```

- SLAVE CONFIGURATION
- Step 1
- Again we'll be editing the configuration file:

```
-bash-4.2$ vi /9.4/data/postgresql.conf
```

```
#hot_standby = off
hot_standby = on
```

Note: set password for user postgres(System user)-- passwd postgres

STREAMING REPLICATION-HOT STANDBY

- Step 2

```
-bash-4.2$ vi /9.4/data/recovery.conf
```

```
primary_conninfo = 'host=192.168.0.1 port=5432 user=repuser password=root'
standby_mode = on
```

- Step 3
- Now stop the service:

```
bash-4.2$ /usr/pgsql-9.4/bin/pg_ctl stop
```

- Connect to master:
  - Copy all of your data from the master to the slave host. You could do that by running this on the master:

STREAMING REPLICATION-HOT STANDBY

```
bash-4.2$ pwd
bash-4.2$ var/lib/pgsql
bash-4.2$ rsync -av -exclude pg_xlog -exclude postgresql.conf -exclude
pg_hba.conf 9.4/data/* 192.168.0.2:/var/lib/pgsql/9.4/data/
```

- On both the master and the slave, add the postgres service in firewall settings:

```
root@localhost ~]# firewall-cmd -permanent -add-service=postgresql
```

- On both the master and the slave you can now start the service:

```
bash-4.2$ /usr/pgsql-9.4/bin/pg_ctl start
```

STREAMING REPLICATION-HOT STANDBY

- With both hosts running the service you can now check on the status of the replication on master machine:

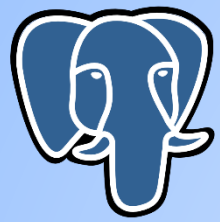
```
-bash-4.2$ psql
```

```
Postgres=# select * from pg_stat_replication;
```

- To pause replication, run following command on Slave machine:

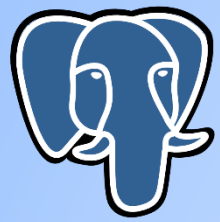
```
bash-4.2$ psql
```

```
Postgres=# select pg_xlog_replay_pause();
Postgres=# select pg_xlog_replay_resume(); --- To resume
```



Upgrading to synchronous replication

- A synchronous replication - commit on the slave is after commit on master, data might be lost
- A commit has to be flushed to disk by at least one replica, reduces odds of data loss
- Two things have to be done
  - SET synchronous_commit = on;
  - synchronous_standby_names on master
  - application_name to primary_conninfo in recovery.conf in replica
- Example:
  - synchronous_standby_names = 'slave1, slave2, slave3'
  - On slave:
    - primary_conninfo = '... application_name=slave2'



Logical replication

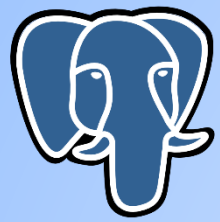
- One of the new features of PostgreSQL 10 is logical replication
- Replicating data objects and their changes, based upon their replication identity
- PostgreSQL supports both mechanisms concurrently – Logical and Physical
- Logical replication uses a publish and subscribe model
- Subscribers pull data from the publications they subscribe to
  - May re-publish data to allow cascading replication

Logical replication

- On the master (Node 1),
  - Set wal_level to logical on postgresql.conf
    - **show wal_level;**
  - Create the role REPLICATION
    - **create ROLE replicator REPLICATION LOGIN PASSWORD 'linux';**
  - Give the right permissions to the tables that we want to be replicated
    - **create table mynames (id int not null primary key, name text);**
    - **grant ALL ON mynames to replicator;**
  - Modify the pg_hba.conf
    - **Host all replicator node2/32 md5**

Logical replication

- On the master (Node 1),
  - Create the publication
    - **create publication mynames_pub for table mynames;**
- On the (Node 2),
  - Modify pg_hba.conf
  - Create the structure of the table we want to replicate
    - **create table mynames (id int not null primary key, name text);**
  - Create the subscription
    - **create subscription mynames_sub CONNECTION 'dbname=db1 host=node1 user=replicator password=password'**



Logical replication

- On the master (Node 1),
  - **insert into mynames values(1,'micky mouse');**
- On the (Node 2),
  - **select * from mynames ;**
  - **insert into mynames values(2,'minni');**
  - **select * from mynames ;**
- On the master (Node 1),
  - **select * from mynames ;**
- On Node 2
  - **drop table mynames ;**
  - **create table mynames (id int not null primary key, name text);**
  - **alter subscription mynames_sub refresh publication;**

Managing Streaming Replication

- Replication works well if it's understood
- Switchover – It is a controlled switch from the master to the standby
  - If performed correctly, there will be no data loss
  - To be safe, simply shut down the master node cleanly, using either the smart or fast shutdown modes
- Failover – It is a forced switch from the master node to a standby because of the loss of the master
  - There is no action to perform on the master; we presume it is not there anymore
  - Now, we need to promote one of the standby nodes to be the new master