%title: Traffic Control Summit -- Postgres High Availability %author: Dewayne Richardson %date: 2017-05-10

-> Postgres High Availability <-

Write-Ahead Logging (WAL)

A standard method for ensuring data integrity. Briefly, WAL's central concept is that changes to data files (where tables and indexes reside) must be written only after those changes have been logged, that is, after log records describing the changes have been flushed to permanent storage.

This is roll-forward recovery, also known as REDO.

WAL Segments

The system physically divides this sequence into WAL segment files, which are normally 16MB apiece (although the segment size can be altered when building PostgreSQL). The segment files are given numeric names that reflect their position in the abstract WAL sequence.

Streaming Replication (SR)

Provides the capability to continuously ship and apply the WAL XLOG records to some number of standby servers in order to keep them current.

-> Postgres Configuration Parameters

/data/pgdata/postgresql.conf

```
#------------------------------------------------------------------------------
# Connection Settings
#------------------------------------------------------------------------------
listen_addresses = '*'             # comma-separated list of addresses;
                                   # defaults to 'localhost'; use '*' for all
                                   # (change requires restart)
port = 5432                        # (change requires restart)
max_connections = 500              # (change requires restart)

# - Memory -
shared_buffers = 512MB             # min 128kb
dynamic_shared_memory_type = posix # the default is the first option
#------------------------------------------------------------------------------
# WRITE AHEAD LOG
#------------------------------------------------------------------------------
wal_level = replica                # minimal, replica, or logical
archive_mode = off                 # enables archiving; off, on, or always
#------------------------------------------------------------------------------
# REPLICATION
#------------------------------------------------------------------------------
# - Sending Server(s) -
# Set these on the master and on any standby that will send replication data.
max_wal_senders = 10               # max number of walsender processes # (change requires restart)
wal_keep_segments = 32             # in logfile segments, 16MB each; 0 disables
hot_standby = on                   # "on" allows queries during recovery
hot_standby_feedback = on          # send info from standby to prevent
#------------------------------------------------------------------------------
# QUERY TUNING
#------------------------------------------------------------------------------
effective_cache_size = 6GB         # This is used when logging to stderr:
logging_collector = on             # Enable capturing of stderr and csvlog
#------------------------------------------------------------------------------
# LOGGING
#------------------------------------------------------------------------------
log_directory = '/var/log/postgresql' # directory where log files are written,
                                      # can be absolute or relative to PGDATA
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'    # log file name pattern, can include strftime() escapes
log_file_mode = 0600               # creation mode for log files, begin with 0 to use octal notation
log_truncate_on_rotation = on      # If on, an existing log file with the
log_rotation_age = 1d              # Automatic rotation of logfiles will
log_rotation_size = 10MB           # Automatic rotation of logfiles will
log_connections = on
log_error_verbosity = verbose      # terse, default, or verbose messages
log_statement = 'none'             # none, ddl, mod, all
log_timezone = 'UTC'
```

-> Postgres Client authentication

Host-Based Authentication (pg_hba)

Client Authentication Types:

- trust
- reject
- md5
- password (unencrypted)
- gss (kerberos)

- sspi (windows)
- ident
- peer
- ldap
- radius
- cert
- pam
- bsd

-> Postgres Client authentication

/data/pgdata/pg_hba.conf

```
# TYPE  DATABASE        USER            ADDRESS             METHOD
host    all             postgres        127.0.0.1/32        trust
host    all             postgres        172.18.0.1/32       trust

host    traffic_ops     traffic_ops     172.18.0.1/32       md5

# -------  CLIENTS ------
# --  to-01.central.apache.org
host    traffic_ops     traffic_ops     192.168.0.1/32      md5

# --  to-02.east.apache.org
host    traffic_ops     traffic_ops     192.168.0.2/32      md5

# --  to-03.west.apache.org
host    traffic_ops     traffic_ops     192.168.0.3/32      md5

# -------  DATABASES ------
host    replication     to_replication  192.168.0.100/32    md5    # Primary
host    replication     to_replication  192.168.0.101/32    md5    # Secondary
host    replication     to_replication  192.168.0.102/32    md5    # Replicas



 FATAL:  no pg_hba.conf entry for host "192.168.0.103", user "traffic_ops", database "traffic_ops", SSL off
 Error connecting to database
```

-> Postgres Standby Server Settings/Recovery Settings

/data/pgdata/recovery.conf

```
standby_mode = 'on'

# Streaming replication source settings
primary_conninfo = 'host=192.168.0.101 port=5432 user=to_replication password=thepass'

# Touch this file to turn this instance into Read/Write mode
trigger_file = '/data/readwrite_mode'
```

```
postgres=# \x
Expanded display is on.



# Replication Enabled

postgres=# SELECT * FROM pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 25
usesysid         | 16386
usename          | to_replication
application_name | walreceiver
client_addr      | 192.168.0.102
client_hostname  |
client_port      | 59713
backend_start    | 2017-05-10 16:33:56.32385+00
backend_xmin     | 452639
state            | streaming
sent_location    | 3/E0B8648
write_location   | 3/E0B8648
flush_location   | 3/E0B8648
replay_location  | 3/E0B8648
sync_priority    | 0
sync_state       | async



# Replication NOT Enabled

postgres=# SELECT * FROM pg_stat_replication;
(0 rows)
```

```
# Replication To Multiple Children

postgres=# SELECT * FROM pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 25
usesysid         | 16386
usename          | to_replication
application_name | walreceiver
```

```
client_addr       | 192.168.0.102
client_hostname   |
client_port       | 59713
backend_start     | 2017-05-10 16:33:56.32385+00
backend_xmin      | 452639
state             | streaming
sent_location     | 3/E0B8648
write_location    | 3/E0B8648
flush_location    | 3/E0B8648
replay_location   | 3/E0B8648
sync_priority     | 0
sync_state        | async
-[ RECORD 2 ]----+------------------------------
pid               | 26
usesysid          | 16387
usename           | to_replication
application_name  | walreceiver
client_addr       | 192.168.0.103
client_hostname   |
client_port       | 59713
backend_start     | 2017-05-10 16:33:56.32385+00
backend_xmin      | 452639
state             | streaming
sent_location     | 3/E0B8648
write_location    | 3/E0B8648
flush_location    | 3/E0B8648
replay_location   | 3/E0B8648
sync_priority     | 0
sync_state        | async
```

Backups

To create a base backup of the server at mydbserver and store it in the local directory /usr/local/pgsql/data

pg_basebackup -h mydbserver -D /usr/local/pgsql/data -X stream

To create a backup of a single-tablespace local database and compress this with bzip2

pg_basebackup -D backup -Ft -z -P

To create a backup of a local database where the tablespace in /opt/ts is relocated to ./backup/ts

pg_basebackup -D backup/data -T /backup/archive

Archived WAL files

Postgres pgdata directory: */backup*

Postgres Admin user

$ sudo su - postgres

Postgres default password file

/home/postgres/.pgpass

Postgres Login

$ psql -U traffic_ops -h localhost

Postgres Help

traffic_ops=# \?

List Databases

traffic_ops=# \l

List Tables

traffic_ops=# \d

Command History

traffic_ops=# \s

Search Command History

traffic_ops=# ^R

Search Command Watching

traffic*ops=# SELECT * from cdn; traffic*ops=# \watch 2

Describe a Single table

traffic_ops=# \d cdn

```
                                    Table "public.cdn"
     Column     |           Type           |                    Modifiers
----------------+--------------------------+-------------------------------------------------
 id             | bigint                   | not null default nextval('cdn_id_seq'::regclass)
 name           | text                     |
 last_updated   | timestamp with time zone | not null default now()
 dnssec_enabled | boolean                  | not null default false
Indexes:
    "idx_57376_primary" PRIMARY KEY, btree (id)
    "idx_57376_cdn_cdn_unique" UNIQUE, btree (name)
Referenced by:
    TABLE "deliveryservice" CONSTRAINT "fk_cdn1" FOREIGN KEY (cdn_id) REFERENCES cdn(id) ON UPDATE RESTRICT ON DELETE RESTRICT
    TABLE "server" CONSTRAINT "fk_cdn2" FOREIGN KEY (cdn_id) REFERENCES cdn(id) ON UPDATE RESTRICT ON DELETE RESTRICT
```

Replication Enabled

traffic*ops=# SELECT * from pg*stat_replication;

Get time stamp of last transaction replayed during recovery. (Streaming Replication Lag)

traffic*ops=# SELECT EXTRACT(EPOCH FROM (now() - pg*last*xact*replay_timestamp()))::INT;

Get the current timeline of the Postgres cluster

traffic*ops=# SELECT substr(pg*xlogfile*name(pg*current*xlog*location()), 1, 8);

Quit Console

traffic_ops=# \q

-> Grafana Graphs <-
