PostgreSQL Streaming Replication Cheat Sheet

Master setup

Primary GUC's (also needs postgresql restart)

wal_level = hot_standby – *required for hot-standby.*
max_wal_senders = 6 – *max number of concurrent replication connections (must be > 0).*

Secondary GUC's

wal_keep_segments = 256 – *(4GB on disk) or set more for high-write and huge databases.*
wal_sender_timeout = 60s – *ok by default.*

Standby setup

Primary GUC's

hot_standby = on – *if enabled standby can accept read-only queries.*

Secondary GUC's

max_standby_streaming_delay = 300s – *set more if recovery conflicts occurs.*
wal_receiver_status_interval = 10s – *ok by default.*
hot_standby_feedback = on – *enable it anyway.*
wal_receiver_timeout = 60s – *ok by default.*

Create dedicated role for replication (optional, need reload)

pg_hba.conf format

host replication repluser standby_ip/netmask authtype (trust, md5, etc.)

Add role with psql and reload configuration:

```
CREATE ROLE repluser WITH LOGIN REPLICATION PASSWORD 'lovesexgod';
SELECT pg_reload_conf();
```

Standby recovery.conf (streaming replication)

primary_conninfo = 'host=master port=… user=… password=…'
standby_mode = 'on' – *start as standby and don't stop recovery.*
trigger_file = '/path/to/file' – *create this file to switch postgres from standby mode to normal.*
recovery_min_apply_delay = 5min – *delay recovery by a fixed period of time (since 9.4).*

pg_basebackup

Main options

-h master, -p port, -U user, -D destdir – *(destdir must have 700 perms).*

Auxiliary options

-c fast | spread – *use 'fast' for start basebackup as soon as possible, 'spread' for minimize load.*
-X stream | fetch – *add WAL archives into backup. The 'stream' uses separate streaming connection (since 9.2).*
-R – *create minimal recovery.conf (since 9.3).*
-r – *limit network bandwidth in kB/s, or use suffix "k" or "M" (since 9.4).*
--xlogdir=dir – *set new pg_xlog location (since 9.4).*
-T olddir=newdir – *set new locations for tablspaces (since 9.4).*
-P – *show progress.*

Standby recovery.conf (point-in-time-recovery)

restore_command = 'cp /path/to/archives/%f "%p"' – *command for restore WAL.*
archive_cleanup_command = 'pg_archivecleanup /archivedir %r' – *clean up old WAL archives.*
recovery_target = 'immediate' – *stop recover when consistent state is reached.*
recovery_target_name = 'deploy' – *recover to point created by pg_create_restore_point().*
recovery_target_time = '2016-01-06 18:11:54.840563' – *recover to timestamp.*
recovery_target_xid = 1234567 – *recover to transaction ID.*
recovery_target_inclusive = true – *recover specified target (true) or stop before target (false).*
recovery_target_timeline = 'latest' – *recover to specified timeline number.*
recovery_end_command = 'cmd' – *run cmd after recovery.*
recovery_target_action = pause | promote | shutdown – *take action after recovery (since 9.5).*

Synchronous replication

Master postgresql.conf

synchronous_standby_names = 'main'

Standby recovery.conf

primary_conninfo = '... application_name = main ...'

Replication slots

Master postgresql.conf

max_worker_processes = max_replication_slots = 6 – *max number of replication slots.*

Create slot on master with psql

SELECT pg_create_physical_replication_slot('main');

Standby recovery.conf

primary_slot_name = 'main'

Monitoring on master

SELECT client_addr, pg_xlog_location_diff(pg_current_xlog_location(),sent_location) as pending from pg_stat_replication; – *get pending (not sent) amount of local WAL on master. Big values (more than 1GB) is bad and mean that the master under heavy load.*

SELECT client_addr, pg_xlog_location_diff(sent_location,replay_location) as lag from pg_stat_replication; – *get sent but not replayed amount of WAL on standby. Big values (more than 1GB) is bad and mean that the standby under heavy load or has network problems.*

Monitoring on standby

SELECT now() - pg_last_xact_replay_timestamp(); – *get lag in seconds;*