Exercises for PostgreSQL Replication Tutorial

version 0.5

Constants

```
$EDITOR below is the editor of your choice.  Options include:
* vi
* nano (recommended if you're unfamiliar with Linux editors)
* jmacs (emacs clone)
* joe

Lines in (parentheses) are comments and shouldn't be pasted/typed in.
The ^ carat symbol is short for the [Ctrl] key.  Thus ^c is "hold
down [Ctrl], press [c]".
```

If using Vagrant: Start Vagrant

```
vagrant up
vagrant ssh
sudo su -
```

If using Docker: run Docker

```
docker run -it pgreplicationtutorial
```

tmux

```
tmux
^b c
su - postgres
^b n
^b n
(you should be in "postgres" shell)
```

basic 2-node replication

```
cd 9.4
ls
pg_basebackup --help | less
pg_basebackup -x -P -D replica1
cp /setup/postgres/replica1/* replica1/
pg_ctl -D replica1 start
ps -aux | grep postgres
psql -p 5432 libdata
\dt
\q
psql -p 5433 libdata
\dt
```

```
\q
createdb -p 5432 -U bench bench
pgbench -i -s 10 -U bench bench
psql -p 5432 bench
\dt
\q
psql -p 5433 bench
\dt
delete from pgbench_history;
\q
```

Replication configuration

```
$EDITOR master/postgresql.conf
$EDITOR replica1/postgresql.conf
$EDITOR replica1/recovery.conf
(here we are just looking at the contents of these files, not
modifying them)
```

Administering replication

```
^b n
(root shell)
cd /setup/pgbench
./runbench.sh
^b n
psql -p 5432 bench
select count(*) from pgbench_history;
(repeat above several times)
\x
select * from pg_stat_activity;
select * from pg_stat_replication;
select pg_xlog_location_diff(write_location,replay_location) from
pg_stat_replication;
(repeat above 2-4 times)
\q
psql -p 5433 bench
select count(*) from pgbench_history;
select pg_is_in_recovery();
select pg_last_xlog_receive_location();
select pg_last_xlog_receive_location(),
pg_last_xlog_replay_location();
select pg_xlog_location_diff(pg_last_xlog_receive_location(),
pg_last_xlog_replay_location());
select pg_last_xact_replay_timestamp();
select now() - pg_last_xact_replay_timestamp();
select pg_xlog_replay_pause();
select pg_xlog_location_diff(pg_last_xlog_receive_location(),
pg_last_xlog_replay_location());
```

```
(repeat above twice)
select now() - pg_last_xact_replay_timestamp();
select pg_xlog_replay_resume();
\q
```

Replication and Security

```
psql -p 5432
create role replicator password 'replicate' login replication;
\du
\q
$EDITOR master/pg_hba.conf
(comment out postgres - replication lines)
$EDITOR replica1/recovery.conf
(change primary_conninfo to use replicator user)
cp /setup/postgres/.pgpass ~/
chmod 700 ~/.pgpass
$EDITOR ~/.pgpass
(check password info for replicator user)
pg_ctl -D master reload
pg_ctl -D replica1 restart
ps -aux | grep replicator
psql -p 5432
select * from pg_stat_replication;
\q
```

Cloning and Archiving

```
^b n
^c
#(pgbench killed)
^b n
pg_ctl -D replica1 stop
rm -rf replica1/*
$EDITOR master/postgresql.conf
(edit to enable archiving by uncommenting the "archive" lines)
pg_ctl -D master restart
ps aux | grep archiver
^b n
runbench.sh
^b n
psql -p 5432
select pg_start_backup('replica1');
select pg_is_in_backup();
\q
ls ~/wal_archive/
rsync -av --exclude="pg_xlog/*" --exclude="postmaster.pid" master/*
replica1/
```

```
psql -p 5432
select pg_stop_backup();
\q
cp /setup/postgres/replica1/* replica1/
cp replica1/recovery.conf.replica1.archiving replica1/recovery.conf
$EDITOR replica1/recovery.conf
pg_ctl -D replica1 start
ps aux | grep startup
psql -p 5432
create table test(test text);
select * from pg_stat_replication;
\q
tail -f /var/log/postgresql/postgresql-replica1
psql -p 5433
\dt
(repeat above until "test" table appears)
\c bench
select count(*) from pgbench_history;
(repeat above several times until result changes)
\q
```

Dual Replication

```
cp replica1/recovery.conf.replica1.dual replica1/recovery.conf
$EDITOR replica1/recovery.conf
^b n
pg_ctl -D replica1 restart
tail -f /var/log/postgresql/postgresql-replica1
^c
psql -p 5432
select * from pg_stat_replication;
\q
```

Failover and Failback

```
^b n
(check if runbench is still running, if not run the below)
./runbench.sh
^b n
pg_ctl -D master -m immediate stop
^b n
^b n
pg_ctl -D replica1 promote
ls replica1/
psql -p 5433
select pg_is_in_recovery();
create table test2(test text);
\q
^b n
```

```
./runbench_replica1.sh
^b n
rm -rf master/*
pg_basebackup -x -P -p 5433 -U replicator -D master
cp /setup/postgres/master/* master/
cp replica1/recovery.done master/recovery.conf
$EDITOR master/recovery.conf
(edit recovery.conf to connect to replica1)
pg_ctl -D master start
tail -f /var/log/postgresql/postgresql-master
^c
psql -p 5433
select * from pg_stat_replication;
\q
pg_ctl -D replica1 -m fast stop
touch master/PROMOTE
(wait)
psql -p 5432
select pg_is_in_recovery();
\q
mv replica1/recovery.done replica1/recovery.conf
pg_ctl -D replica1 start
psql -p 5432
select * from pg_stat_replication;
\q
```

Query Lag

```
$EDITOR replica1/postgresql.conf
(look at streaming_delay settings)
psql -p 5433 libdata
begin;
set transaction_isolation = 'repeatable read';
select * from copies;
q
^b n
psql -U postgres -p 5432 libdata
vacuum full copies;
select * from pg_stat_replication;
select pg_xlog_location_diff(write_location, replay_location) from
pg_stat_replication;
^b n
select * from copies;
( wait 10-20 sec. )
select * from copies;
( repeat above until error )
\q
^b n
\q
```

```
^b n (postgres shell)
```

Synchronous Replication

```
$EDITOR master/postgresql.conf
(add "replica1" to list of synchronous_standby_names, and turn on
hot_standby_feedback)
pg_ctl -D master reload
^b n
runbench.sh
^b n
psql -p 5432
select * from pg_stat_replication;
\timing
begin;
insert into test values ('test1');
commit;
begin;
set synchronous_commit='local';
insert into test values ('test2');
commit;
\q
pg_ctl -D replica1 stop
psql -p 5432
begin;
insert into test values ('test3');
commit;
^c
begin;
set synchronous_commit='local';
insert into test values ('test4');
commit;
\q
$EDITOR master/postgresql.conf
(remove synchronous replica)
pg_ctl -D master reload
pg_ctl -D replica1 start
```

Cascading Replication

```
pg_basebackup -p 5433 -x -P -U replicator -D replica2
cp /setup/postgres/replica2/* replica2/
pg_ctl -D replica2 start
psql -p 5432
select * from pg_stat_replication;
\q
psql -p 5433
```

```
select * from pg_stat_replication;
\q
psql -p 5434
select pg_is_in_recovery();
\q
pg_ctl -D replica1 promote
psql -p 5432
select * from pg_stat_replication;
\q
psql -p 5433
select * from pg_stat_replication;
\q
pg_ctl -D master stop
```

pgbouncer LB/Failover

```
^b n
service pgbouncer start
$EDITOR /etc/pgbouncer/pgbouncer.ini
^b n
psql -p 6432 bench
show port;
create table branches2 as select * from pgbench_branches;
\q
psql -p 6432 bench_ro
show port;
drop table branches2;
\q
pg_ctl -D replica1 -m fast stop
pg_ctl -D replica2 promote
psql -p 6432 bench
^b n
$EDITOR /etc/pgbouncer/pgbouncer.ini
(edit the rw connections so that the point to replica2)
service pgbouncer restart
^b n
psql -p 6432 bench
show port;
drop table branches2;
\q
```

replication slots

```
pg_ctl -D replica2 -m fast stop
pg_ctl -D master start
rm -rf replica1/*
pg_basebackup -x -P -D replica1
cp /setup/postgres/replica1/* replica1/
mv replica1/recovery.conf.replica1.slot replica1/recovery.conf
```

```
psql -p 5432
SELECT * FROM pg_create_physical_replication_slot('replica1');
\q
less replica1/recovery.conf
pg_ctl -D replica1 start
^b n
cd /setup/postgres/pgbench
./runbench.sh
^b n
psql -p 5432
select * from pg_replication_slots;
\q
pg_ctl -D replica1 stop
psql -p 5432
select * from pg_replication_slots;
# repeat above 3 times
\q
pg_ctl -D replica1 start
psql -p 5432
select * from pg_replication_slots;
# repeat above 3 times
\q
```

Exit

```
^b d
exit
exit
vagrant halt (or destroy)
```