

Quick Start Guide

Overview

This document should get you up and running with a replicated PostgreSQL database in just a few steps. We'll gloss over many details in the interest of getting you up and running as quickly as possible. To make things simple, this document will use the sample cluster configuration found in the "bruce/sample" directory of the distribution. The example uses a single PostgreSQL installation with 4 databases.

- `bruce_config` - This database contains the cluster configuration metadata. This includes cluster names and identifiers, master database uri's and descriptive names, slave database uri's and descriptive names, and regular expression patterns to determine which tables are replicated.
- `bruce_master` - The master database. All modifications to replicated tables are made to this database and subsequently replicated to each slave.
- `bruce_slave_1` - A slave database. All modifications to the master database are replicated here.
- `bruce_slave_2` - A slave database. All modifications to the master database are replicated here.

The master and each slave have a `replication_test` schema and a `replication_test.replicate_this` table that we will replicate across the cluster.

Note: *This demonstration makes the assumption that all replicated clusters in the schema contain identical data.*

Automated Test

The steps in this guide are loosely followed in `test/src/com/netblue/bruce/SetupClusterFromExistingDbAcceptanceTest.java`

Prerequisites

- Download latest build: [insert build link here](#)
- `tar xvzf bruce-0.1a.tgz`
- Change scripts to be executable:
- `cd release/bin`
- `chmod 755 *.sh`
- Ensure you have database set up properly:
- create db `bruce_config`
- create db `bruce_master`
- create db `bruce_slave_1`
- create db `bruce_slave_2`
- create user `bruce` with access to all databases and ability to create database objects (NB: need to think about security in real life applications)
- create schema `replication_test` in `bruce_master`, `bruce_slave_1` and `bruce_slave_2`
- create table `replication_test.replicate_this` in `bruce_master`, `bruce_slave_1` and `bruce_slave_2`

Configure the node topology database urls:

- `cd $BRUCE/sample`
- `vi config.xml` (change db urls to point to your databases for each node)

Setup the node topology

- `cd $BRUCE/bin`
- `./admin.sh -data ../sample/config.xml -initnodeschema -initsnapshots MASTER`

```
-loadschema -operation CLEAN_INSERT -url  
jdbc:postgresql://localhost:5432/bruce_config?user=bruce -pass bruce
```

Command line notes:

-data ../sample/config.xml tells us what data file to load. the sample file contains 1 master and 2 slaves with replication patterns to recognize the replication_test.* tables.

-initnodeschema tells us to install the replication schema on each node, along with the triggers for each replicated table. If the schema is already installed, then only the replication triggers are installed for each replicated table. For master nodes, if there is no data in the snapshotlog table (as in this case) a snapshot of the db is taken.

-initsnapshots MASTER tells us to set the slave snapshot status to the last snapshot in the master database. Since the -initnodeschema option initializes this for us, we'll use that.

-loadschema tells us to install the replication node topology schema on the configuration database. This is only really useful the first time you run the admin tool for a given configuration database.

-operation CLEAN_INSERT tells us to drop everything from the node topology configuration database before inserting these new nodes

-url is the URL for the configuration database

-pass is the password for the configuration database

Start up replication for the new cluster

- cd \$BRUCE/bin
- ./startup.sh ClusterOne

Insert rows in master, confirm slave is updated

- psql bruce_master
- insert into replication_test.replicate_this (name, value) values ('One', 1);
- \c bruce_slave_1
- select * from replication_test.replicate_this
- \c bruce_slave_2
- select * from replication_test.replicate_this