

Linuxha.net
Step-by-step Guide:
PostgreSQL

Introduction

The purpose of this document is to suggest procedures for creating a clustered PostgreSQL configuration with Linuxha.net. These procedures were tested with the following configuration:

- Fedora Core 5
- DRBD version 0.7.20
- Linuxha.net version 1.2
- PostgreSQL version 8.1.4

The following conventions are used throughout this document:

# ls	Single-line command entered as root user.
# useradd \ > --home-dir /home2/xyz \ > --gid abc \ > xyz	Multi-line command entered as root user.
\$ ls	Single-line command entered as non-root user
\$ cut \ > --delimiter=":" \ > --fields=1,5 \ / etc/passwd	Multi-line command entered as non-root user.
127.0.0.1 localhost 192.168.1.32 fc5s1 192.168.1.33 fc5s2 192.168.100.32 fc5s1b 192.168.100.33 fc5s2b	Command output or file contents

User Account Creation

The PostgreSQL processes will run with the **postgres** user credentials. The **postgres** user and group must exist on both nodes and have the same user and groups ids.

All commands in this section are executed as **root** on both nodes unless indicated otherwise.

1. Check whether the **postgres** user exists.

```
# grep postgres /etc/passwd
```

If there is no result then continue to step 2. Otherwise, compare the user and group ids on both nodes. These are the third and fourth fields respectively of the user account record as shown below:

```
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
```

If the user and group ids are not the same on both nodes, proceed to step 6. Otherwise continue to the next section.

2. Identify next available group id.

```
# expr `cut --delimiter=":" --fields=3 /etc/group | \  
> grep --regexp="^[5-9][0-9][0-9]$" | sort --numeric-sort|tail -1` + 1
```

3. Create **postgres** group and assign the larger of the group ids identified in step 2, e.g. 506.

```
# groupadd --gid 506 mysql
```

4. Identify next available user id

```
# expr `cut --delimiter=":" --fields=3 /etc/passwd | \
> grep --regexp="^[5-9][0-9][0-9]$" | sort --numeric-sort|tail -1` + 1
```

5. Create **postgres** user and assign the larger of the user ids identified in step 4, e.g. 506.

```
# useradd --home-dir /pgsql/data --comment "PostgreSQL server" \
> --gid postgres --uid 506 postgres
```

Proceed to the next section.

6. Change **postgres** group id on **one of the nodes** to be the same as the other, e.g. 27. Ensure that the new group id is not already in use.

```
# grep 27 /etc/group # check that group id is not in use
# groupmod -g 27 mysql
```

7. Change **postgres** user id on **one of the nodes** to be the same as the other, e.g. 27. Ensure that the new user id is not already in use.

```
# grep 27 /etc/passwd # check that user id is not in use
# usermod --uid 27 --gid postgres postgres
```

8. Change user and group ownerships of files owned by **postgres** to reflect the new user / group id.

The following commands will find all files whose user / group ownership is 26 (original user / group id) and change their user / group ownership to **postgres**..

```
# find /usr -gid 26 -exec chgrp postgres {} \;
# find /usr -uid 26 -exec chown postgres {} \;
```

Installation

This section provides basic instructions for downloading and installing PostgreSQL version 8.1 from i386 binary RPM or source. Instructions for installing or upgrading using other package formats are beyond the scope of this guide.

All commands are executed as **root** on both nodes.

1. Check whether PostgreSQL is installed, by executing

```
# find /usr -name postmaster -print
```

If a result similar to the following is displayed, then PostgreSQL is installed and you may proceed to step 4.

```
/usr/bin/postmaster
```

2. Download PostgreSQL as follows (these are all **single-line** commands):

i386 binary RPM

```
# wget http://wwwmaster.postgresql.org/download/mirrors-
ftp?file=binary%2Fv8.1.4%2Flinux%2Frpms%2Ffedora%2Ffedora-core-5%2Fpostgresql-
8.1.4-3PGDG.i686.rpm
# wget http://wwwmaster.postgresql.org/download/mirrors-
ftp?file=binary%2Fv8.1.4%2Flinux%2Frpms%2Ffedora%2Ffedora-core-5%2Fpostgresql-
server-8.1.4-3PGDG.i686.rpm
```

or source

```
# cd /tmp
# wget ftp://ftp.postgresql.org/pub/source/v8.1.4/postgresql-8.1.4.tar.gz
```

3. Install PostgreSQL

RPM

```
# rpm --hash --install postgresql-8.1.4-3PGDG.i686.rpm
# rpm --hash --install postgresql-server-8.1.4-3PGDG.i686.rpm
```

or source

```
# cd /usr/local/src
# tar xvzf /tmp/postgresql-8.1.4.tar.gz
# cd postgresql-8.1.4
# ./configure
# make
# make install
```

4. Configure firewall

If the firewall is running, then it should be configured to accept connections on port 5432, the default PostgreSQL port.

Insert the following line into **/etc/sysconfig/iptables**.

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 5432 -j ACCEPT
```

Restart the firewall

```
# /etc/init.d/iptables/restart
```

5. Delete the PostgreSQL automatic shutdown scripts.

```
# rm /etc/rc3.d/S??postgresql /etc/rc0.d/K??postgresql
```

6. For a **source-based** installation, append **/usr/local/pgsql/bin** to the **postgres** PATH.

```
# inpath=`su - postgres -c "which pg_ctl 2>/dev/null|wc -l"`
# if [ $inpath -lt 1 ]
> then
> su - postgres -c \
> 'echo -e "PATH=\$PATH:/usr/local/pgsql/bin;export PATH" >> ~/.bash_profile'
> fi
```

Create Replicated File System

The commands in this section are to be executed on both nodes as **root**. Steps 5 and 6 may be omitted on the secondary node.

1. Use **fdisk** to create a partition (*/dev/sda6*) large enough to accommodate all PostgreSQL databases.

2. Initialize partition

```
# pvcreate /dev/sda6
```

3. Create application volume group (*postgreslvg*)

```
# vgcreate pgsqldvg /dev/sda6
```

4. Create mount point for replicated file system

```
# mkdir /pgsql
```

5. Create logical volume (*datav*) to store databases.

```
# lvcreate --size 512M --name datav pgsqsvg
```

6. Create file system on *datav*.

```
# mkfs -t ext3 /dev/pgsqsvg/datav
```

Populate Replicated File System

All commands are executed on the primary node as **root**, except where indicated otherwise.

1. Shutdown PostgreSQL (**both nodes**)

```
# su - postgres -c "pg_ctl stop"
```

2. Mount logical volume

```
# mount -t ext3 /dev/pgsqsvg/datav /pgsql
```

3. Create directory structure

```
# mkdir /pgsql/data
# chmod 0700 /pgsql/data
# chown postgres:postgres /pgsql/data
# mkdir /pgsql/scripts
# touch /pgsql/scripts/startapp /pgsql/scripts/stopapp
# chmod u+x /pgsql/scripts/*
```

4. Move or create database files.

If this is a new PostgreSQL installation, then initialize the database files as follows:

```
# su - postgres -c "initdb -D /pgsql/data"
```

Otherwise, copy existing files.

Existing RPM Installation

```
# cd /var/lib/pgsql/data
# find . -print|cpio --pass-through /pgsql/data
```

Existing Source Installation

```
# cd /usr/local/pgsql/data
# find . -print|cpio --pass-through /pgsql/data
```

Configuration

All commands are executed as **root** on the primary node.

1. Insert the following into **/pgsql/scripts/startapp**:

```
su - postgres -c "pg_ctl start -D /pgsql/data -w -l /var/log/pgsql"
if [ $? -eq 0 -a -f /pgsql/data/postmaster.pid ]
then mv /pgsql/data/postmaster.pid /var/run
fi
```

2. Insert the following into **/pgsql/scripts/stopapp**:

```
if [ -f /var/run/postmaster.pid ]
then mv /var/run/postmaster.pid /pgsql/data
fi
su - postgres -c "pg_ctl stop -D /pgsql/data"
```

3. Edit **/pgsql/data/postgres.conf** and set the *listen_addresses* parameter to the application virtual IP address, e.g. *192.168.1.60*.

```
listen_addresses = '192.168.1.60'
```

4. (Optional) Edit **/pgsql/data/pg_hba.conf** and add an entry to allow connection to the database from either cluster node.

The following line gives all hosts on the 192.168.1.0/24 network trusted access to all databases:

```
host all all 192.168.1.0/24 trust
```

Build Application

The commands in this section are to be executed as **root** on the **primary node**.

1. Create the **/etc/cluster/pgsql** directory.

```
# mkdir /etc/cluster/samba
```

2. Create **/etc/cluster/pgsql/appconf.xml** as shown:

```
<?xml version="1.0"?>
<appconf>
  <global>
    <version>0.1</version>
    <name>pgsql</name>
    <takeover>normal</takeover>
    <synccrate>2000</synccrate>
    <preferred_node>fc5s1</preferred_node>
  </global>
  <networks>
    <network net="main" ip="192.168.1.60" netmask="255.255.255.0" />
  </networks>
  <vg>
    <name>pgsqlvg</name>
    <type>filesystems</type>
  </vg>
  <application>
    <startscript>/pgsql/scripts/startapp</startscript>
    <stopscript>/pgsql/scripts/stopapp</stopscript>
    <maxstoptime>10</maxstoptime>
    <maxstarttime>20</maxstarttime>
  </application>
</appconf>
```

The bolded values are configuration-specific, as described in the following table:

Entry	Value
global/preferred_node	Host name of one of the nodes in the cluster, or LEAST_CPU_LOAD
networks/network.net	Same as one of <i>node/network.name</i> in /etc/cluster/clconf.xml
networks/network.ip	Virtual IP address of application
networks/network.netmask	Netmask corresponding to virtual IP address
vg/name	Name of application volume group

3. Create **/etc/cluster/pgsql/lems.local.xml** as shown:

```
<?xml version="1.0"?>
<lems_config>
  <globals modules="/sbin/cluster/lems/modules"
    programs="/sbin/cluster/lems/programs"
    logs="/var/log/cluster/lems"
  />
  <check>
    <name>flag_check</name>
    <type>internal</type>
    <module>flag_check postgres</module>
    <interval>5</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="%RCDATA%"/>
      <action rc="2" action="ABORT"/>
    </action_list>
  </check>
  <check>
    <name>postgres</name>
    <type>internal</type>
    <module>procmon /etc/cluster/pgsql/postgres.xml</module>
    <interval>10</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="STOP"/>
      <action rc="2" action="FAILOVER"/>
    </action_list>
  </check>
  <check>
    <name>ip</name>
    <type>internal</type>
    <module>ip_module postgres</module>
    <interval>10</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="RUN move_ip"/>
      <action rc="2" action="STOP"/>
    </action_list>
  </check>
  <check>
    <name>fsmonitor</name>
    <type>internal</type>
    <module>fsmon postgres</module>
    <interval>10</interval>
    <action_list>
      <action rc="0" action="NOP"/>
    </action_list>
  </check>
</lems_config>
```

```

        <action rc="1" action="PAUSE 30"/>
        <action rc="2" action="STOP"/>
        <action rc="3" action="FAILOVER"/>
        <action rc="10" action="PAUSE 60"/>
    </action_list>
</check>
</lems_config>

```

4. Create **/etc/cluster/pgsql/postgres.xml** as shown:

```

<?xml version="1.0"?>
<procmon>
  <global>
    <logdir>/var/log/cluster</logdir>
    <restarts>5</restarts>
    <resetwindow>3600</resetwindow>
    <restartcmd>
      /pgsql/scripts/stopapp ; /pgsql/scripts/startapp
    </restartcmd>
  </global>
  <process>
    <label>PostgreSQL postmaster</label>
    <user>postgres</user>
    <process_string>
      /usr(/|/local/pgsql/)bin/postmaster -D /pgsql/data
    </process_string>
    <min_count>1</min_count>
    <max_count>1</max_count>
  </process>
  <process>
    <label>postgres writer</label>
    <user>postgres</user>
    <process_string>postgres: writer process</process_string>
    <min_count>1</min_count>
    <max_count>1</max_count>
  </process>
  <process>
    <label>postgres stats buffer</label>
    <user>postgres</user>
    <process_string>
      postgres: stats buffer process
    </process_string>
    <min_count>1</min_count>
    <max_count>1</max_count>
  </process>
  <process>
    <label>postgres stats collector</label>
    <user>postgres</user>
    <process_string>
      postgres: stats collector process
    </process_string>
    <min_count>1</min_count>
    <max_count>1</max_count>
  </process>
</procmon>

```

5. Verify LEMS configuration by executing:

```

# lems.pl --config /etc/cluster/pgsql/lems.local.xml \
> --application postgres --verbose --check --file /dev/tty

```

If successful, a result similar to the following will be displayed.

```
INFO 03/09/2006 16:12:20 Using modules from : /sbin/cluster/lems/modules
INFO 03/09/2006 16:12:20 Using programs from : /sbin/cluster/lems/programs
INFO 03/09/2006 16:12:20 Writing logs to : /var/log/cluster/lems
INFO 03/09/2006 16:12:20 Listening on port : 9904
INFO 03/09/2006 16:12:20 Global initialisation complete.
INFO 03/09/2006 16:12:20 Started local server on port 9904
INFO 03/09/2006 16:12:20 Validating monitor entry ip...
INFO 03/09/2006 16:12:20 Validated monitor entry ip successfully.
INFO 03/09/2006 16:12:20 Validating monitor entry postgres...
INFO 03/09/2006 16:12:20 Validated monitor entry postgres successfully.
INFO 03/09/2006 16:12:20 Validating monitor entry fsmonitor...
INFO 03/09/2006 16:12:20 Validated monitor entry fsmonitor successfully.
INFO 03/09/2006 16:12:20 Validating monitor entry flag_check...
INFO 03/09/2006 16:12:20 Validated monitor entry flag_check successfully.
INFO 03/09/2006 16:12:20 Check mode - transferring validated config to remote node.
INFO 03/09/2006 16:12:20 Configuration transferred successfully.
INFO 03/09/2006 16:12:20 Calculated a check interval of 2.5 seconds.
```

6. Build **pgsql**.

```
# clbuildapp --application pgsql --sync
```

The output of a successful build is shown below:

```
INFO 02/09/2006 21:00:21 Backups directory defaulted to /clbackup
INFO 02/09/2006 21:00:21
INFO 02/09/2006 21:00:21 Validation of Application 'pgsql' started.
INFO 02/09/2006 21:00:21 ['/var/log/cluster/build/pgsql-check-300609022100.log']
INFO 02/09/2006 21:00:23 Initial Validation of Application successful.
INFO 02/09/2006 21:00:23
INFO 02/09/2006 21:00:23 NOTE: Build of new application is being performed.
INFO 02/09/2006 21:00:23
INFO 02/09/2006 21:00:23 Host Environment Validation started.
INFO 02/09/2006 21:00:23 ['/var/log/cluster/build/pgsql-envcheck-300609022100.log']
INFO 02/09/2006 21:00:30 Host Environment Validation successful.
INFO 02/09/2006 21:00:30
INFO 02/09/2006 21:00:30 Cluster state : RUNNING
INFO 02/09/2006 21:00:30 Application state: UNDEFINED
INFO 02/09/2006 21:00:30
INFO 02/09/2006 21:00:30 Volume Group Configuration started.
INFO 02/09/2006 21:00:30 ['/var/log/cluster/build/pgsql-lvm-300609022100.log']
INFO 02/09/2006 21:00:34 Volume Group Configuration successful.
INFO 02/09/2006 21:00:34
INFO 02/09/2006 21:00:34 Application Resource Allocation started.
INFO 02/09/2006 21:00:34 ['/var/log/cluster/build/pgsql-build-300609022100.log']
INFO 02/09/2006 21:00:41 Application Resource Allocation successful.
INFO 02/09/2006 21:00:41
INFO 02/09/2006 21:00:41 Application Data Synchronisation started.
INFO 02/09/2006 21:00:41 ['/var/log/cluster/build/pgsql-syncdata-300609022100.log']
Storage Syncing: 0Mb/ 0Mb [100 % Complete]
INFO 02/09/2006 21:03:06 Application Data Synchronisation successful.
INFO 02/09/2006 21:03:06
```

Run Application

The commands in this section are executed as **root** on either node except where indicated.

1. If necessary, form the cluster

```
# clform
```

2. Start **pgsql** on the primary node (*fc5s1*)

```
# clrunapp --application pgsql --node fc5s1
```

3. Check state of **pgsql**.

```
# clstat --application pgsql
```

A result similar to the following should be returned:

```
Cluster: cluster1 - UP

Application      Node      State  Runnig  Monitor  Stale  Fail-over?
      pgsql      fc5s1  STARTED 0:00:01 Running    0        Yes

File Systems

Mount Point      Valid  Type      State  % Complete  Completion
/pgsql           both   drbd      Sync

Process Monitors

      Name      Status  Restarts  Current  Reset at
      postgres  Running    5         0        N/A

General Monitors

      Type      Name      Status
      Flag Check  flag_check  Running
      FS Monitor  fsmonitor  Running
      IP Monitor  ip         Running
```

4. On the node where the application is running, list the PostgreSQL processes.

```
# ps -u postgres -f
```

A result similar to the following should be returned:

```
postgres 8800    1  0 15:49 ?        00:00:00 /usr/bin/postmaster
postgres 8803   8800  0 15:49 ?        00:00:00 postgres: writer process
postgres 8804   8800  0 15:49 ?        00:00:00 postgres: stats buffer process
postgres 8805   8804  0 15:49 ?        00:00:00 postgres: stats collector process
```

5. Test connection to database via virtual IP address (192.168.1.60), by executing on either node

```
# su - postgres -c "psql -h 192.168.1.60 -c '\\\dn\\'"
```

If PostgreSQL is listening on the virtual IP address and accepting remote connections, then a result similar to the following will be returned:

```
      List of schemas
      Name          | Owner
      -----+-----
      information_schema | postgres
```

```
pg_catalog      | postgres
pg_toast        | postgres
public          | postgres
(4 rows)
```

6. On the node where it is running, shut down **pgsql**.

```
# clhaltapp --application pgsql
```

7. Start the PostgreSQL server on the secondary node (*fc5s2*)

```
# clrunapp --application pgsql --node fc5s2
```

8. Repeat steps 3 to 6 to test **pgsql** on secondary node.