Repmgr Guide

Support and Operation Documents

Roya Anooshe

Exported on 08/29/2021

Table of Contents

1 Preface

This document describes the installation process of repmgr service as a PostgreSQL replication and failover management.

Moving on, a summary definition of repmgr service is will be provided, then this document continues to the installation process step by step.

2 A brief explanation

repmgr is a popular tool for PostgreSQL replication and failover management.

this service greatly simplifies the process of setting up and managing PostgreSQL databases with high availability (HA) and scalability requirements. It also helps us manage a cluster of PostgreSQL databases by taking advantage of the Hot Standby capability.

without the repmgr service we are likely to do these steps manually:

Configuring replication parameters in both primary and each standby node
Backing up primary node data with pg_basebackup from each standby node and restoring it there
Restarting the standby node(s)

Also from an operational side, a few tasks include:

Checking replication status using SQL statements
Promoting a standby node when a switchover is necessary or when the primary is unavailable
Fencing off failed or stopped primary node
Recreating replication from the new read/write node to existing or new standby nodes

With repmgr service most of these tasks can be automated, saving us both time and effort.

The repmgr project is hosted at https://github.com/2ndQuadrant/repmgr

Source code downloads are available from repmgr.org[1]

If interested you can take a look at the links mentioned above.

1 http://repmgr.org

3 Repmgr Installation Guide

3.1 Requirements

For this installation, we need at least 2 servers (virtual machines).

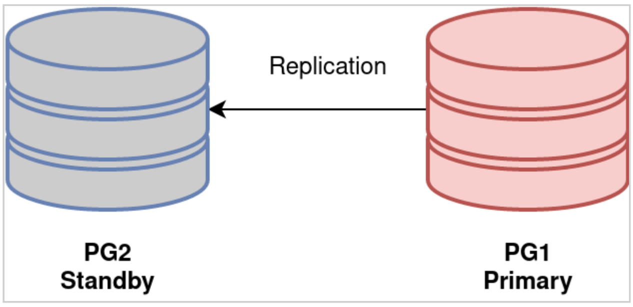
One server as the Master node and the other one as the Standby node.

notice in most cases for broadband customers 2 servers as mentioned before is efficient. So in this case we will have 2 servers with roles defined.

Our scenario information and requirements are described in the table below:

Node Name	Role	OS	IP	Apps
PG1	Master	Debian 9	192.168.56.101	PostgreSQL 12 and repmgr
PG2	Standby	Debian 9	192.168.56.102	PostgreSQL 12 and repmgr

Also here is the diagram:



3.2 Installation process:

in order to install rempgr service we need to do the following steps on both master and standby nodes:

Serivce Not Available

If there were errors during the installation of packages or updating them, you might consider using "shecan DNS" in order to install them completely.

just edit the **/etc/resolv.conf** file on the server and set it as follows:

/etc/resolv.conf

nameserver 178.22.122.100

nameserver 185.51.200.2

nameserver 8.8.8.8

after installation, you might want to set everything the way there were before.

*preference: https://shecan.ir/

On both master and standby nodes, install repmgr service using the following commands:

Installing repmgr and required services on debian9

```
#apt-get install apt-transport-https
#sh -c 'echo "deb https://apt.2ndquadrant.com/ $(lsb_release -cs)-2ndquadrant
main" > /etc/apt/sources.list.d/2ndquadrant.list'
#apt-get install curl ca-certificates
#apt-get install curl ca-certificates
#curl https://apt.2ndquadrant.com/site/keys/9904CD4BD6BAF0C3.asc | apt-key
add -
#apt update
#apt-get install postgresql-12-repmgr
#systemctl status repmgrd
```

If postgresql service is not installed

Normally when we begin to deploy replications or clustering the main service (in this case postgresql) is already installed. in case that postgresql didn't exist on any of the servers, you can install It with these commands:

```
#sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release
-cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
#wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |
 apt-key add -
#apt-get update
#apt-get install postgresql-12
#apt-get install postgresql
```

Backups

It is strongly recommended to save backups from configuration files. Such as postgres.conf and pg_hba.conf.

then on the **primary node** edit /etc/postgresql/12/main/postgresql.conf file with your preferred editor and make the following changes:

/etc/postgresql/12/main/postgresql.conf

```
listen_addresses = '*'          # what IP address(es) to listen on;
Port=5432
wal_level =replica              # minimal, replica, or logical
max_wal_size = 1GB
min_wal_size = 80MB
archive_mode = on               # enables archiving; off, on, or always
archive_command = '/bin/true'   # command to use to archive a logfile segment
max_wal_senders = 10            # max number of walsender processes
max_replication_slots = 10      # max number of replication slots
hot_standby = on                # "off" disallows queries during recovery
```

Now we need to restart the postgresql service:

```
#systemctl restart postgresql
```

then on the **standby node** edit /etc/postgresql/12/main/postgresql.conf file with your preferred editor and make the following changes:

```
listen_addresses = '*'          # what IP address(es) to listen on;
Port=5432
max_wal_size = 1GB
min_wal_size = 80MB
max_replication_slots = 10      # max number of replication slots
hot_standby = on                # "off" disallows queries during recovery
```

Now we need to restart the postgresql service:

```
#systemctl restart postgresql
```

Now we need to create a superuser for repmgr. Do the following steps on both nodes:

```
#su - postgres
$ createuser --superuser repmgr
$ createdb --owner=repmgr repmgr
$ psql -c "ALTER USER repmgr SET search_path TO repmgr, public;"
```

After creating the repmgr user, we need to edit postgres.conf file and add the following line:

```
shared_preload_libraries = 'repmgr'
```

Then restart postgres service again:

```
#systemctl restart postgresql
```

For each nodes we need set connection configurations. do the following steps if haven't already:

First set server's hostname

- Primary node: edit /etc/hostname and set it to PG1
- Standby node: edit /etc/hostname and set it to PG2

Then set hosts for both servers. add the following lines to /etc/hosts:

/etc/hosts

```
192.168.56.101    PG1
192.168.56.102    PG2
```

in order to make the servers available for each other, we need to configure the pg_hba.conf.

on the **primary node** please add the following lines to pg_hba.conf (please notice the nodes IPs and don't make mistakes):

/etc/postgresql/12/main/pg_hba.conf

```
# Database administrative login by Unix domain socket
local   all            postgres                                    peer
local   replication   repmgr                                      trust
host    replication   repmgr      127.0.0.1/32                    trust
host    replication   repmgr      192.168.56.102/32               trust
host    replication   repmgr      192.168.56.101/32               trust

local   repmgr        repmgr                                      trust
host    repmgr        repmgr      127.0.0.1/32                    trust
host    repmgr        repmgr      192.168.56.101/32               trust
host    repmgr        repmgr      192.168.56.102/32               trust
```

on the **standby node** please add the following lines to pg_hba.conf (please notice the nodes IPs and don't make mistakes):

/etc/postgresql/12/main/pg_hba.conf

```
# Database administrative login by Unix domain socket
local   all           postgres                       peer
local   replication   repmgr                         trust
host    replication   repmgr     127.0.0.1/32        trust
host    replication   repmgr     192.168.56.102/32   trust
host    replication   repmgr     192.168.56.101/32   trust

local   repmgr        repmgr                         trust
host    repmgr        repmgr     127.0.0.1/32        trust
host    repmgr        repmgr     192.168.56.102/32   trust
host    repmgr        repmgr     192.168.56.101/32   trust
```

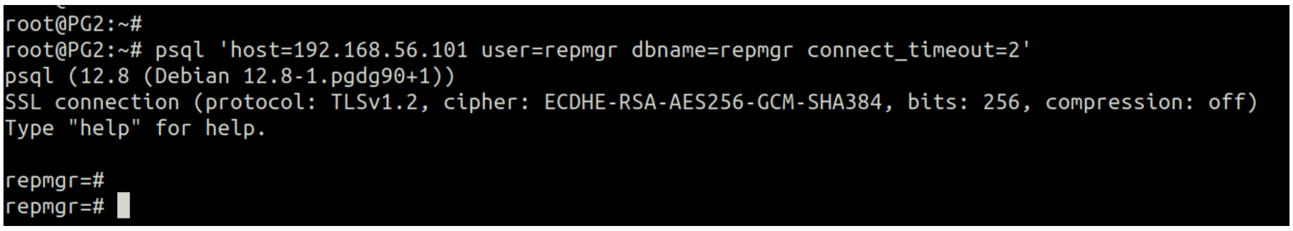
We need to restart or reload the postgres service:

```
#systemctl restart postgresql
```

Now test the connection from PG2 to PG1 by the command below. with this command you should be able to connect to repmgr database in PG1:

test connection

```
#psql 'host=PG1 user=repmgr dbname=repmgr connect_timeout=2'
```



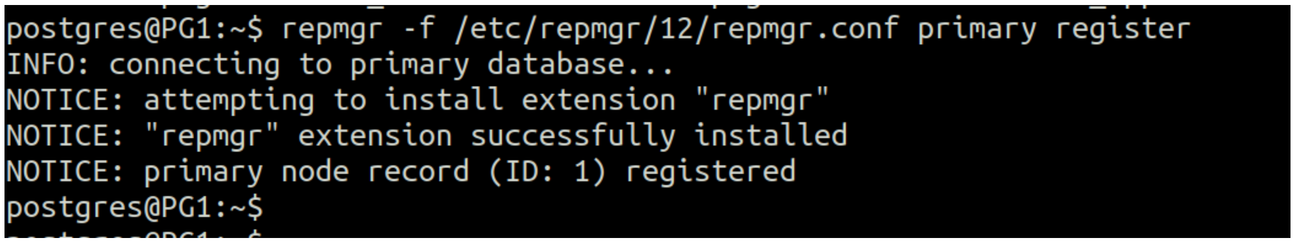
```
root@PG2:~#
root@PG2:~# psql 'host=192.168.56.101 user=repmgr dbname=repmgr connect_timeout=2'
psql (12.8 (Debian 12.8-1.pgdg90+1))
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

repmgr=#
repmgr=#
```

Now we need to register PG1 as the primary node:

Master Node Registering

```
#su - postgres
$repmgr -f /etc/repmgr/12/repmgr.conf primary register
```

the output will be like:



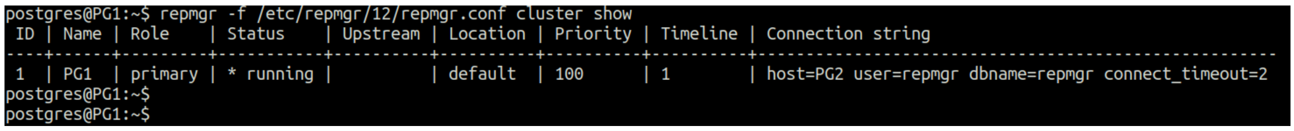
```
postgres@PG1:~$ repmgr -f /etc/repmgr/12/repmgr.conf primary register
INFO: connecting to primary database...
NOTICE: attempting to install extension "repmgr"
NOTICE: "repmgr" extension successfully installed
NOTICE: primary node record (ID: 1) registered
postgres@PG1:~$
```

Now we check the cluster status:

Cluste Status

```
$repmgr -f /etc/repmgr/12/repmgr.conf cluster show
```

you should see the table as:



```
postgres@PG1:~$ repmgr -f /etc/repmgr/12/repmgr.conf cluster show
 ID | Name | Role    | Status    | Upstream | Location | Priority | Timeline | Connection string
----+------+---------+-----------+----------+----------+----------+----------+------------------------------------------------------
 1  | PG1  | primary | * running |          | default  | 100      | 1        | host=PG2 user=repmgr dbname=repmgr connect_timeout=2
postgres@PG1:~$
postgres@PG1:~$
```

Here we need to test if the cloning process from the primary node is ok to be done.

Notice that you need to stop postgres service first during the cloning.

on the standby node enter the following command:

Testing Clone Process on Standby

```
#systemctl stop postgresql#su - postgres
$repmgr -h 192.168.56.101 -U repmgr -d repmgr -f /etc/repmgr/12/repmgr.conf
standby clone --dry-run
```

If you see the "**INFO: all prerequisites for "standby clone" are met**" message on the screen then the cloning process is ok. so now we need to clone:

Clone

```
$repmgr -h 192.168.56.101  -U repmgr -d repmgr -F -f /etc/repmgr/12/
repmgr.conf standby clone
```

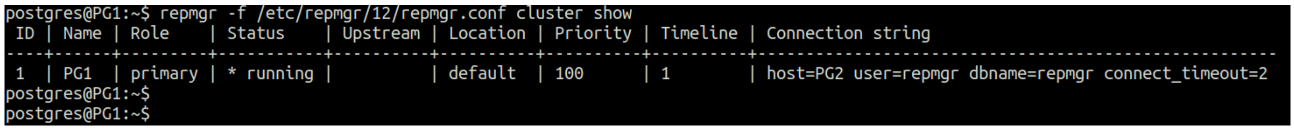
if the output's last line is **HINT: after starting the server, you need to register this standby with "repmgr standby register"**, which means cloning was successful.

then start the Postgres service and register PG2 as Standby:

Stand by Registration

```
#systemctl restart postgresql
$repmgr -f /etc/repmgr/12/repmgr.conf standby register$repmgr -f /etc/repmgr/
12/repmgr.conf cluster show
```

the output should look like:



```
postgres@PG1:~$ repmgr -f /etc/repmgr/12/repmgr.conf cluster show
 ID | Name | Role    | Status    | Upstream | Location | Priority | Timeline | Connection string
----+------+---------+-----------+----------+----------+----------+----------+-----------------------------------------------------
 1  | PG1  | primary | * running |          | default  | 100      | 1        | host=PG2 user=repmgr dbname=repmgr connect_timeout=2
postgres@PG1:~$
postgres@PG1:~$
```

Now rempgr service is set and running.

Thank you for reviewing this document. Please let me know if there are any errors

Contact me:
Email: anooshe.roya@gmail.com
LinkedIn : www.linkedin.com/in/roya-anooshe1995