Postgres repmgr setup with a witness

Dear Readers,

Automatic failover tool is built into PG's core but we have other open-source tools, like Patroni and repmgr. So, in this post, we will see

1. how to set repmgr with Postgres 17 for streaming replication
2. how to set up a witness
3. how to perform switchover
4. how to perform failover
5. how to setup synchronous replication
6. how to add a custom/user-managed tablespace in a replication setup

Contents

Terminologies 2

High level steps 2

Setup 2

- Configuring the primary node 3
- Configuring the secondary node 8
- Start repmgrd daemon 11
- Configuring the witness node 12

Performing switchover 15

Performing failover 17

Setting Synchronous Replication 20

Creating a custom/user managed tablespace 22

Terminologies

Before we start, let's understand a few terminologies related to repmgr. You can even refer this link:

- **Node**: A single PG server
- **Cluster**: Group of nodes among which replication is happening
- **repmgr**: A tool to manage PG replication. It's used to
  - set up a replication server(s)
  - performing switchover and failover
  - displaying the status, events of the replicaiton cluster and it's nodes
- **repmgrd**: It's a linux daemon that actively monitors nodes in the cluster and performs
  - recording replication performance
  - performs automatic failover
  - to setup monitoring alerts/notifications
- **Upstream node**: A source node from which the target node receives data to be replicated. It could be a primary node or replica node (in the case of cascading replication)
- **Switchover**: Selecting a new primary node and making the existing primary node a secondary one.
- **Failover**: action which occurs if a primary server fails and a suitable standby is promoted as the new primary.
- **Witness**: A node that doesn't participate in replication but it's being used to elect a new primary node and even to avoid split brain (multiple primaries) type of situations.

High level steps

1. Setup 3 Liux machines with PG and repmgr packages
2. Setup passwordless ssh among these servers
3. Setup PG and repmgr in a primary node and register that node in the repmgr
4. Setup PG and repmgr in a secondary node. Clone primary's PGDATA into it and register that node in the repmgr
5. Setup and add witness node in repmgr
6. Perform various operations e.g. switchover, failover etc.

So let's begin,

Setup

In our setup, we'll have three Linux machines. One will be PG primary, the other a replica, and the last one a witness node.

I have 3 t2.micro (1 vCPU and GB RAM) machines running on CentOS 9. **I have set passwordless ssh among them for postgres** user.

Node name	Node IP	Role	Running daemons
Original primary	54.173.215.225	PG primary node	PG 17 + repmgrd
Original secondary	54.224.11.184	PG secondary node	PG 17 + repmgrd
Witness	3.91.195.224	repmgr witness node	PG 17 + repmgrd

Configuring the primary node

Set the following parameters in the PG cluster:
wal_level=replica : determines how much information is written to the WAL. The default value is replica, which writes enough data to support WAL archiving and replication, including running read-only queries on a standby server.

wal_log_hints=on : A configuration parameter determining whether the entire content of each disk page is written to WAL after a checkpoint. It's needed for pg_rewind when we want to start a stopped/failed node in the cluster.

hot_standby=on : Standby can serve read traffic.

listen_addresses='*' : Allowing PG to listen to all incoming TCP/IP connections.

shared_preload_libraries ='repmgr' : Some libraries need to perform certain operations that can only take place at postmaster start, such as allocating shared memory, reserving light-weight locks, or starting background workers. Those libraries must be loaded at server start through this parameter.

archive_mode=on : To enable WAL archiveing. Technically, we don't need archive mode for streaming replication.

archive_command ='/bin/true' : The script/command via which WALs will be archived.

```
postgres=# alter system set wal_level =replica;
ALTER SYSTEM
postgres=# alter system set wal_log_hints =on;
ALTER SYSTEM
postgres=# alter system set hot_standby=on;
ALTER SYSTEM
postgres=# alter system set listen_addresses ='*';
ALTER SYSTEM
postgres=# alter system set shared_preload_libraries ='repmgr';
ALTER SYSTEM
postgres=# alter system set archive_mode=on;
ALTER SYSTEM
postgres=# alter system set archive_command ='/bin/true';
ALTER SYSTEM
postgres=# show max_wal_senders;
max_wal_senders
-----------------
10
(1 row)

postgres=# show max_replication_slots;
max_replication_slots
----------------------
```

```
10
(1 row)
```

As you can see, I've not max_wal_senders and max_replication_slots in my configuration because their existing values are sufficient for my tests. If needed, you may want to set it to desired values.

As a root user, install the repmgr package

```
 [root@ip-172-31-82-57 ~]# yum install repmgr*17* -y
Last metadata expiration check: 0:45:38 ago on Sat 11 Jan 2025 10:06:01 AM UTC.
Dependencies resolved.
================================================================================
================================================================================
Package Architecture Version Repository Size
================================================================================
================================================================================
Installing:
repmgr_17 x86_64 5.5.0-1PGDG.rhel9 pgdg17 268 k
repmgr_17-devel x86_64 5.5.0-1PGDG.rhel9 pgdg17 7.4 k
repmgr_17-llvmjit x86_64 5.5.0-1PGDG.rhel9 pgdg17 20 k
Installing dependencies:
llvm x86_64 19.1.5-2.el9 appstream 21 M
llvm-libs x86_64 19.1.5-2.el9 appstream 57 M

Transaction Summary
================================================================================
================================================================================
Install 5 Packages

Total download size: 78 M
Installed size: 297 M
```

As postgres DB user, create a repmgr user and repmgr database. Using this user replication will happen and replication configuration/status will be saved in this repmgr database.

```
postgres=# CREATE USER repmgr WITH SUPERUSER ENCRYPTED PASSWORD
'Oracle123';
CREATE ROLE
postgres=# CREATE DATABASE repmgr WITH OWNER=repmgr;
CREATE DATABASE
```

Add the below repmgr-related entries in pg_hba.conf. These will help repmgr user to connect to repmgr DB to get/set status/configuration. Also, repmgr user will be able to fetch WAL segments for replication purpose. To set up a more secure connection, you may want to avoid trust.

```
# "local" is for Unix domain socket connections only
local all all peer
```

```
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
host repmgr repmgr 54.173.215.225/32 trust
host repmgr repmgr 54.224.11.184/32 trust
host repmgr repmgr 3.91.195.224/32 trust
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256
host replication repmgr 54.173.215.225/32 trust
host replication repmgr 54.224.11.184/32 trust
host replication repmgr 3.91.195.224/32 trust
```

Restart PG cluster so that the settings will take effect

```
[postgres@original_primary ~]$ pg_ctl -D $PGDATA restart
```

Now we'll create repmgr's config directory and file but before that it's important to make yourself familiarize with repmgr terminologies .To understand in-depth about the above parameter's, you can refer link and for a sample repmgr configuration.

node_id: It's an integer and should be unique throughout the cluster. It's a number assigned to a node
node_name: A string that uniquely identifies a node in the cluster.

conninfo: Connection string using which other nodes in the cluster will connect to the current node.
connect_timeout: It's a recommended setting as we'll use repmgrd. It is to determine the length of time which elapses before a network connection attempt is abandoned.

data_directory: PGDATA location of the node. It's needed when the postgres process is not running and repmgr wants to perform some operations e.g. clone, run pg_rewind etc.

pg_bindir: Path to the PostgreSQL binary directory (location of pg_ctl, pg_basebackup etc.). Only required if these are not in the system PATH.

log_file: If log_facility is set to STDERR, log output can be redirected to the specified file.
promote_command: The program or script defined in promote_command will be executed in a failover situation when repmgrd determines that the current node is to become the new primary node.

follow_command: The program or script defined in follow_command will be executed in a failover situation when repmgrd determines that the current node is to follow the new primary node.

```
[postgres@original_primary ~]$ cd
[postgres@original_primary ~]$ mkdir repmgr
```

```
[postgres@original_primary ~]$ cd repmgr/
[postgres@original_primary repmgr]$ vi repmgr.conf
[postgres@original_primary repmgr]$ cat repmgr.conf
node_id=1
node_name=original_primary
conninfo='host=3.88.202.217 user=repmgr dbname=repmgr connect_timeout=2'
data_directory='/var/lib/pgsql/17/data/'
use_replication_slots=true
failover=automatic
pg_bindir='/usr/pgsql-17/bin'
log_file='/var/lib/pgsql/repmgr/log_repmgr.log'
promote_command='/usr/pgsql-17/bin/repmgr standby promote -f
/var/lib/pgsql/repmgr/repmgr.conf --log-to-file'
follow_command='/usr/pgsql-17/bin/repmgr standby follow -f /var/lib/pgsql/repmgr/repmgr.conf
--log-to-file --upstream_node_id=%n'
```

Register the primary node in the repmgr's configuration.
Dry-run

```
[postgres@original_primary repmgr]$ repmgr -f repmgr.conf primary register --dry-run
INFO: connecting to primary database...
NOTICE: would now attempt to install extension "repmgr"
```

Actual run
As you can see it has created an extension in repmgr DB

```
[postgres@original_primary repmgr]$ repmgr -f 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@original_primary repmgr]$ repmgr -f repmgr.conf primary register --dry-run
INFO: connecting to primary database...
NOTICE: would now attempt to install extension "repmgr"
[postgres@original_primary repmgr]$ repmgr -f 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@original_primary repmgr]$
[postgres@original_primary repmgr]$ psql
psql (17.2)
Type "help" for help.

postgres=# \c repmgr
You are now connected to database "repmgr" as user "postgres".
repmgr=# \c
You are now connected to database "repmgr" as user "postgres".
repmgr=# \dx
                    List of installed extensions
  Name   | Version |   Schema   |            Description
---------+---------+------------+-----------------------------------
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
 repmgr  | 5.5     | repmgr     | Replication manager for PostgreSQL
(2 rows)

repmgr=# \dt repmgr.*
              List of relations
 Schema |        Name        | Type  | Owner
--------+--------------------+-------+--------
 repmgr | events             | table | repmgr
 repmgr | monitoring_history | table | repmgr
 repmgr | nodes              | table | repmgr
 repmgr | voting_term        | table | repmgr
(4 rows)
```

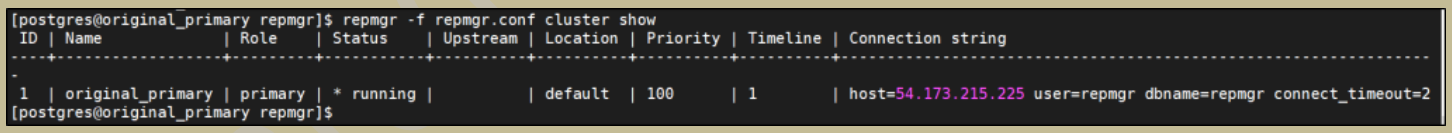
Verification
From psql

```
[postgres@original_primary repmgr]$ psql
psql (17.2)
Type "help" for help.

postgres=# \c repmgr
You are now connected to database "repmgr" as user "postgres".
repmgr=# \c
You are now connected to database "repmgr" as user "postgres".
repmgr=# \dx
List of installed extensions
Name | Version | Schema | Description
---------+---------+------------+-----------------------------------
plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
repmgr | 5.5 | repmgr | Replication manager for PostgreSQL
(2 rows)

repmgr=# \dt repmgr.*
List of relations
```

```
Schema | Name | Type | Owner
--------+-------------------+-------+--------
repmgr | events | table | repmgr
repmgr | monitoring_history | table | repmgr
repmgr | nodes | table | repmgr
repmgr | voting_term | table | repmgr
(4 rows)

repmgr=# select * from repmgr.nodes;
-[ RECORD 1 ]----+----------------------------------------------------------------
node_id | 1
upstream_node_id |
active | t
node_name | original_primary
type | primary
location | default
priority | 100
conninfo | host=54.173.215.225 user=repmgr dbname=repmgr connect_timeout=2
repluser | repmgr
slot_name | repmgr_slot_1
config_file | /var/lib/pgsql/repmgr/repmgr.conf
```

From repmgr

```
[postgres@original_primary repmgr]$ repmgr -f repmgr.conf cluster show
ID | Name | Role | Status | Upstream | Location | Priority | Timeline | Connection string
----+------------------+---------+-----------+----------+----------+----------+-----------+-----------------------------
-----------------------------------
1 | original_primary | primary | * running | | default | 100 | 1 | host=54.173.215.225 user=repmgr
dbname=repmgr connect_timeout=2
```



Configuring the secondary node

As the root user, install repmgr Linux package

```
[postgres@original_secondary ~]$ cd
[postgres@original_secondary ~]$ mkdir repmgr
[postgres@original_secondary ~]$ cd repmgr
[postgres@original_secondary repmgr]$ vi repmgr.conf
[postgres@original_secondary repmgr]$ cat repmgr.conf
node_id=2
node_name=original_secondary
```

```
conninfo='host=54.90.90.159 user=repmgr dbname=repmgr connect_timeout=2'
data_directory='/var/lib/pgsql/17/data/'
use_replication_slots=true
failover=automatic
pg_bindir='/usr/pgsql-17/bin'
log_file='/var/lib/pgsql/repmgr/log_repmgr.log'
promote_command='/usr/pgsql-17/bin/repmgr standby promote -f
/var/lib/pgsql/repmgr/repmgr.conf --log-to-file'
follow_command='/usr/pgsql-17/bin/repmgr standby follow -f /var/lib/pgsql/repmgr/repmgr.conf
--log-to-file --upstream_node_id=%n'
```

Clone primary PG node into the secondary

We are using --fast-checkpoint so that cloning won't wait for the next checkpoint to complete. We are forcing a checkpoint.

Dry-run

```
[postgres@original_secondary repmgr]$ repmgr -h 54.173.215.225 -U repmgr -f repmgr.conf
standby clone --fast-checkpoint --dry-run
NOTICE: destination directory "/var/lib/pgsql/17/data" provided
INFO: connecting to source node
DETAIL: connection string is: host=54.173.215.225 user=repmgr
DETAIL: current installation size is 29 MB
INFO: "repmgr" extension is installed in database "(null)"
INFO: replication slot usage not requested; no replication slot will be set up for this standby
INFO: parameter "max_wal_senders" set to 10
NOTICE: checking for available walsenders on the source node (2 required)
INFO: sufficient walsenders available on the source node
DETAIL: 2 required, 10 available
NOTICE: checking replication connections can be made to the source server (2 required)
INFO: required number of replication connections could be made to the source server
DETAIL: 2 replication connections required
INFO: replication slots will be created by user "repmgr"
NOTICE: standby will attach to upstream node 1
INFO: would execute:
/usr/pgsql-17/bin/pg_basebackup -l "repmgr base backup" -D /var/lib/pgsql/17/data -h
54.173.215.225 -p 5432 -U repmgr -c fast -X stream
INFO: all prerequisites for "standby clone" are met
```

Actual run

```
[postgres@original_secondary repmgr]$ repmgr -h 54.173.215.225 -U repmgr -f repmgr.conf
standby clone --fast-checkpoint
NOTICE: destination directory "/var/lib/pgsql/17/data" provided
INFO: connecting to source node
DETAIL: connection string is: host=54.173.215.225 user=repmgr
DETAIL: current installation size is 29 MB
INFO: replication slot usage not requested; no replication slot will be set up for this standby
```

```
NOTICE: checking for available walsenders on the source node (2 required)
NOTICE: checking replication connections can be made to the source server (2 required)
INFO: checking and correcting permissions on existing directory "/var/lib/pgsql/17/data"
NOTICE: starting backup (using pg_basebackup)...
INFO: executing:
/usr/pgsql-17/bin/pg_basebackup -l "repmgr base backup" -D /var/lib/pgsql/17/data -h
54.173.215.225 -p 5432 -U repmgr -c fast -X stream
NOTICE: standby clone (using pg_basebackup) complete
NOTICE: you can now start your PostgreSQL server
HINT: for example: pg_ctl -D /var/lib/pgsql/17/data start
HINT: after starting the server, you need to register this standby with "repmgr standby register"
```

Start the secondary cluster

```
[postgres@original_secondary repmgr]$ pg_ctl -D $PGDATA/ start
```

Register the secondary node

Dry-run

```
[postgres@original_secondary repmgr]$ repmgr -f repmgr.conf standby register --dry-run
INFO: connecting to local node "original_secondary" (ID: 2)
INFO: connecting to primary database
WARNING: --upstream-node-id not supplied, assuming upstream node is primary (node ID: 1)
INFO: all prerequisites for "standby register" are met
```

Actual run

```
[postgres@original_secondary repmgr]$ repmgr -f repmgr.conf standby register

INFO: connecting to local node "original_secondary" (ID: 2)
INFO: connecting to primary database
WARNING: --upstream-node-id not supplied, assuming upstream node is primary (node ID: 1)
INFO: standby registration complete
NOTICE: standby node "original_secondary" (ID: 2) successfully registered
```

Verification

From PG

```
repmgr=# select * from repmgr.nodes;
-[ RECORD 1 ]----+-----------------------------------------------------------
node_id | 1
upstream_node_id |
active | t
node_name | original_primary
```

```
type | primary
location | default
priority | 100
conninfo | host=54.173.215.225 user=repmgr dbname=repmgr connect_timeout=2
repluser | repmgr
slot_name | repmgr_slot_1
config_file | /var/lib/pgsql/repmgr/repmgr.conf
-[ RECORD 2 ]----+---------------------------------------------------------
node_id | 2
upstream_node_id | 1
active | t
node_name | original_secondary
type | standby
location | default
priority | 100
conninfo | host=54.224.11.184 user=repmgr dbname=repmgr connect_timeout=2
repluser | repmgr
slot_name | repmgr_slot_2
config_file | /var/lib/pgsql/repmgr/repmgr.conf
```

From repmgr

```
[postgres@original_primary repmgr]$ repmgr -f repmgr.conf cluster show
ID | Name | Role | Status | Upstream | Location | Priority | Timeline | Connection string
----+--------------------+---------+-----------+--------------------+----------+----------+----------+--------------------
-----------------------------------------
1 | original_primary | primary | * running | | default | 100 | 1 | host=54.173.215.225 user=repmgr
dbname=repmgr connect_timeout=2
2 | original_secondary | standby | running | original_primary | default | 100 | 1 |
host=54.224.11.184 user=repmgr dbname=repmgr connect_timeout=2
```

```
[postgres@original_primary repmgr]$ repmgr -f repmgr.conf cluster show
 ID | Name               | Role    | Status    | Upstream         | Location | Priority | Timeline | Connection string

----+--------------------+---------+-----------+------------------+----------+----------+----------+-------------------------------------------------
-----------
 1  | original_primary   | primary | * running |                  | default  | 100      | 1        | host=54.173.215.225 user=repmgr dbname=repmgr connect
_timeout=2
 2  | original_secondary | standby |   running | original_primary | default  | 100      | 1        | host=54.224.11.184 user=repmgr dbname=repmgr connect_
timeout=2
[postgres@original_primary repmgr]$
```

Start repmgrd daemon

On both primary and secondary servers start repmgrd daemon

```
[postgres@original_secondary repmgr]$ repmgrd -f repmgr.conf
[2025-01-11 11:46:57] [NOTICE] redirecting logging output to
"/var/lib/pgsql/repmgr/log_repmgr.log"

[postgres@original_secondary repmgr]$ ps -ef | grep repmgr
postgres 22283 20948 0 11:46 ? 00:00:00 postgres: repmgr repmgr 54.224.11.184(35832) idle
postgres 22285 1 0 11:46 ? 00:00:00 repmgrd -f repmgr.conf
```

```
postgres 22355 4931 0 11:47 pts/0 00:00:00 grep --color=auto repmgr
```

Configuring the witness node

Install the repmgr binaries

```
[root@ip-172-31-88-201 ~]# yum install repmgr*17* -y
```

Set the below parameters in postgres DB

```
postgres=# alter system set listen_addresses ='*';
ALTER SYSTEM
postgres=# alter system set shared_preload_libraries ='repmgr';
ALTER SYSTEM
postgres=# CREATE USER repmgr WITH SUPERUSER ENCRYPTED PASSWORD
'Oracle123';
CREATE ROLE
postgres=# CREATE DATABASE repmgr WITH OWNER=repmgr;
CREATE DATABASE
```

Add the below entries in pg_hba.conf

```
[postgres@ip-172-31-88-201 ~]$ grep -i repmgr $PGDATA/pg_hba.conf
host repmgr repmgr 54.173.215.225/32 trust
host repmgr repmgr 54.224.11.184/32 trust
host repmgr repmgr 3.91.195.224/32 trust
```

Create repmgr configuration

```
[postgres@witness ~]$ cd
[postgres@witness ~]$ mkdir repmgr
[postgres@witness ~]$ cd repmgr/
[postgres@witness ~]$ cat repmgr.conf
[postgres@witness repmgr]$ vi repmgr.conf
[postgres@witness repmgr]$ cat repmgr.conf
node_id=3
node_name=witness_node
conninfo='host=3.91.195.224 user=repmgr dbname=repmgr connect_timeout=2'
data_directory='/var/lib/pgsql/17/data/'
pg_bindir='/usr/pgsql-17/bin'
log_file='/var/lib/pgsql/repmgr/log_repmgr.log'
```

Restart the witness PG DB cluster

```
[postgres@ip-172-31-88-201 repmgr]$ pg_ctl -D $PGDATA/ restart
```

Register the witness node

Dry-run

```
[postgres@ip-172-31-88-201 repmgr]$ repmgr -f repmgr.conf witness register -h
54.173.215.225 --dry-run
INFO: connecting to witness node "witness_node" (ID: 3)
INFO: connecting to primary node
INFO: prerequisites for registering the witness node are met
```

Actual run

```
[postgres@ip-172-31-88-201 repmgr]$ repmgr -f repmgr.conf witness register -h
54.173.215.225
INFO: connecting to witness node "witness_node" (ID: 3)
INFO: connecting to primary node
NOTICE: attempting to install extension "repmgr"
NOTICE: "repmgr" extension successfully installed
INFO: witness registration complete
NOTICE: witness node "witness_node" (ID: 3) successfully registered
```

Verification

From PG

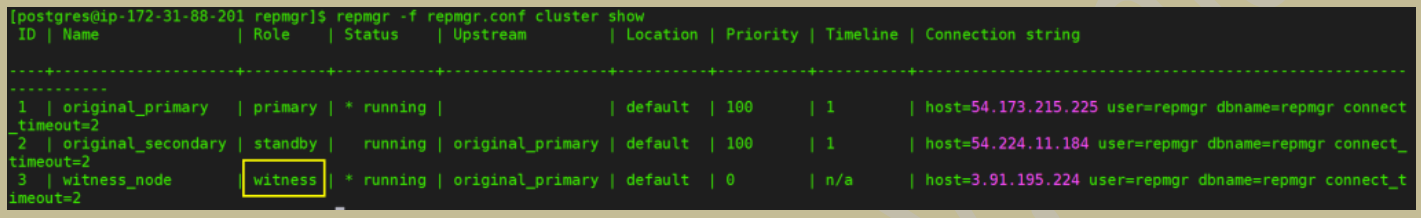
```
repmgr=# select * from repmgr.nodes;
-[ RECORD 1 ]----+-----------------------------------------------------------
node_id | 1
upstream_node_id |
active | t
node_name | original_primary
type | primary
location | default
priority | 100
conninfo | host=54.173.215.225 user=repmgr dbname=repmgr connect_timeout=2
repluser | repmgr
slot_name |
config_file | /var/lib/pgsql/repmgr/repmgr.conf
-[ RECORD 2 ]----+-----------------------------------------------------------
node_id | 2
upstream_node_id | 1
active | t
node_name | original_secondary
```

```
type | standby
location | default
priority | 100
conninfo | host=54.224.11.184 user=repmgr dbname=repmgr connect_timeout=2
repluser | repmgr
slot_name |
config_file | /var/lib/pgsql/repmgr/repmgr.conf
-[ RECORD 3 ]----+-----------------------------------------------------------
node_id | 3
upstream_node_id | 1
active | t
node_name | witness_node
type | witness
location | default
priority | 0
conninfo | host=3.91.195.224 user=repmgr dbname=repmgr connect_timeout=2
repluser | repmgr
slot_name |
config_file | /var/lib/pgsql/repmgr/repmgr.conf
```

```
repmgr=# \dt repmgr.*
             List of relations
 Schema |        Name        | Type  | Owner
--------+--------------------+-------+--------
 repmgr | events             | table | repmgr
 repmgr | monitoring_history | table | repmgr
 repmgr | nodes              | table | repmgr
 repmgr | voting_term        | table | repmgr
(4 rows)

repmgr=# SELECT node_id, upstream_node_id, active, node_name, type, priority, slot_name
              FROM repmgr.nodes ORDER BY node_id;
 node_id | upstream_node_id | active |     node_name      |  type   | priority |   slot_name
---------+------------------+--------+--------------------+---------+----------+---------------
       1 |                  | t      | original_primary   | primary |      100 | repmgr_slot_1
       2 |                1 | t      | original_secondary | standby |      100 | repmgr_slot_2
       3 |                1 | t      | witness_node       | witness |        0 |
(3 rows)

repmgr=# SELECT slot_name, slot_type, active, active_pid FROM pg_replication_slots ;
   slot_name   | slot_type | active | active_pid
---------------+-----------+--------+------------
 repmgr_slot_2 | physical  | t      |      23394
(1 row)

repmgr=# \c postgres
You are now connected to database "postgres" as user "postgres".
postgres=# SELECT slot_name, slot_type, active, active_pid FROM pg_replication_slots ;
   slot_name   | slot_type | active | active_pid
---------------+-----------+--------+------------
 repmgr_slot_2 | physical  | t      |      23394
(1 row)
```

From repmgr

```
[postgres@ip-172-31-88-201 repmgr]$ repmgr -f repmgr.conf cluster show
ID | Name | Role | Status | Upstream | Location | Priority | Timeline | Connection string
----+-------------------+---------+-----------+-----------------+----------+----------+----------+-------------------
-----------------------------------------
1 | original_primary | primary | * running | | default | 100 | 1 | host=54.173.215.225 user=repmgr
dbname=repmgr connect_timeout=2
2 | original_secondary | standby | running | original_primary | default | 100 | 1 |
host=54.224.11.184 user=repmgr dbname=repmgr connect_timeout=2
3 | witness_node | witness | * running | original_primary | default | 0 | n/a | host=3.91.195.224
user=repmgr dbname=repmgr connect_timeout=2
```



Start the repmgrd

```
[postgres@ip-172-31-88-201 repmgr]$ repmgrd -f repmgr.conf
```

Performing switchover

From the standby node run
Dry-run

```
[postgres@original_secondary repmgr]$ repmgr -f repmgr.conf standby switchover --dry-run
NOTICE: checking switchover on node "original_secondary" (ID: 2) in --dry-run mode
INFO: SSH connection to host "54.173.215.225" succeeded
INFO: able to execute "repmgr" on remote host "54.173.215.225"
WARNING: 1 sibling nodes found, but option "--siblings-follow" not specified
DETAIL: these nodes will remain attached to the current primary:
witness_node (node ID: 3, witness server)
INFO: 1 walsenders required, 10 available
INFO: demotion candidate is able to make replication connection to promotion candidate
INFO: archive mode is "off"
INFO: replication lag on this standby is 0 seconds
NOTICE: attempting to pause repmgrd on 3 nodes
INFO: would pause repmgrd on node "original_primary" (ID: 1)
INFO: would pause repmgrd on node "original_secondary" (ID: 2)
INFO: would pause repmgrd on node "witness_node" (ID: 3)
NOTICE: local node "original_secondary" (ID: 2) would be promoted to primary; current
primary "original_primary" (ID: 1) would be demoted to standby
INFO: following shutdown command would be run on node "original_primary":
"/usr/pgsql-17/bin/pg_ctl -D '/var/lib/pgsql/17/data' -W -m fast stop"
```

```
INFO: parameter "shutdown_check_timeout" is set to 60 seconds
INFO: prerequisites for executing STANDBY SWITCHOVER are met
```

Actual Run

```
[postgres@original_secondary repmgr]$ repmgr -f repmgr.conf standby switchover
NOTICE: executing switchover on node "original_secondary" (ID: 2)
WARNING: 1 sibling nodes found, but option "--siblings-follow" not specified
DETAIL: these nodes will remain attached to the current primary:
witness_node (node ID: 3, witness server)
NOTICE: attempting to pause repmgrd on 3 nodes
NOTICE: local node "original_secondary" (ID: 2) will be promoted to primary; current primary
"original_primary" (ID: 1) will be demoted to standby
NOTICE: stopping current primary node "original_primary" (ID: 1)
NOTICE: issuing CHECKPOINT on node "original_primary" (ID: 1)
DETAIL: executing server command "/usr/pgsql-17/bin/pg_ctl -D '/var/lib/pgsql/17/data' -W -m
fast stop"
INFO: checking for primary shutdown; 1 of 60 attempts ("shutdown_check_timeout")
INFO: checking for primary shutdown; 2 of 60 attempts ("shutdown_check_timeout")
NOTICE: current primary has been cleanly shut down at location 0/3011BD8
NOTICE: promoting standby to primary
DETAIL: promoting server "original_secondary" (ID: 2) using pg_promote()
NOTICE: waiting up to 60 seconds (parameter "promote_check_timeout") for promotion to
complete
NOTICE: STANDBY PROMOTE successful
DETAIL: server "original_secondary" (ID: 2) was successfully promoted to primary
NOTICE: node "original_secondary" (ID: 2) promoted to primary, node "original_primary" (ID:
1) demoted to standby
NOTICE: switchover was successful
DETAIL: node "original_secondary" is now primary and node "original_primary" is attached as
standby
NOTICE: STANDBY SWITCHOVER has completed successfully
```

Logs from the witness

```
[2025-01-11 12:02:04] [WARNING] unable to ping "host=54.173.215.225 user=repmgr
dbname=repmgr connect_timeout=2"
[2025-01-11 12:02:04] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:02:04] [INFO] checking state of node 1, 1 of 6 attempts
[2025-01-11 12:02:04] [WARNING] unable to ping "user=repmgr connect_timeout=2
dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr"
[2025-01-11 12:02:04] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:02:04] [INFO] sleeping 10 seconds until next reconnection attempt
[2025-01-11 12:02:14] [INFO] checking state of node 1, 2 of 6 attempts
[2025-01-11 12:02:14] [NOTICE] node 1 has recovered, reconnecting
[2025-01-11 12:02:14] [INFO] connection to node 1 succeeded
[2025-01-11 12:02:14] [INFO] original connection no longer available, using new connection
[2025-01-11 12:02:14] [NOTICE] reconnected to upstream node after 10 seconds
```

```
[2025-01-11 12:02:14] [NOTICE] current upstream node "original_primary" (ID: 1) is not primary, restarting monitoring
[2025-01-11 12:02:14] [INFO] witness monitoring connection to primary node "original_secondary" (ID: 2)
```

```
[postgres@original_primary repmgr]$ repmgr -f repmgr.conf cluster show
 ID | Name              | Role    | Status    | Upstream          | Location | Priority | Timeline | Connection string
----+-------------------+---------+-----------+-------------------+----------+----------+----------+-----------------------------------------------------------------------
 1  | original_primary   | standby |   running | original_secondary | default  | 100      | 1        | host=54.173.215.225 user=repmgr dbname=repmgr connect_timeout=2
 2  | original_secondary | primary | * running |                   | default  | 100      | 2        | host=54.224.11.184 user=repmgr dbname=repmgr connect_timeout=2
 3  | witness_node       | witness | * running | original_secondary | default  | 0        | n/a      | host=3.91.195.224 user=repmgr dbname=repmgr connect_timeout=2
[postgres@original_primary repmgr]$
```

Performing failover

Before performing a failover, from the pervious state I did switchover (just to bring the cluster in the original state)

Current state

```
[postgres@original_primary repmgr]$ repmgr -f repmgr.conf cluster show
 ID | Name              | Role    | Status    | Upstream         | Location | Priority | Timeline | Connection string
----+-------------------+---------+-----------+------------------+----------+----------+----------+-----------------------------------------------------------------------
 1  | original_primary   | primary | * running |                  | default  | 100      | 3        | host=54.173.215.225 user=repmgr dbname=repmgr connect_timeout=2
 2  | original_secondary | standby |   running | original_primary | default  | 100      | 2        | host=54.224.11.184 user=repmgr dbname=repmgr connect_timeout=2
 3  | witness_node       | witness | * running | original_primary | default  | 0        | n/a      | host=3.91.195.224 user=repmgr dbname=repmgr connect_timeout=2
[postgres@original_primary repmgr]$
```

Now I shut the PG cluster from the primary node (you can even turn off the Linux machine if you want).

```
[postgres@original_primary repmgr]$ pg_ctl -D $PGDATA/ stop
```

Logs from the witness

```
[2025-01-11 12:12:32] [WARNING] unable to ping "host=54.173.215.225 user=repmgr dbname=repmgr connect_timeout=2"
[2025-01-11 12:12:32] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:12:32] [INFO] checking state of node 1, 1 of 6 attempts
[2025-01-11 12:12:32] [WARNING] unable to ping "user=repmgr connect_timeout=2 dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr"
[2025-01-11 12:12:32] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:12:32] [INFO] sleeping 10 seconds until next reconnection attempt
[2025-01-11 12:12:42] [INFO] checking state of node 1, 2 of 6 attempts
[2025-01-11 12:12:42] [WARNING] unable to ping "user=repmgr connect_timeout=2 dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr"
[2025-01-11 12:12:42] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:12:42] [INFO] sleeping 10 seconds until next reconnection attempt
[2025-01-11 12:12:52] [INFO] checking state of node 1, 3 of 6 attempts
[2025-01-11 12:12:52] [WARNING] unable to ping "user=repmgr connect_timeout=2 dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr"
```

```
[2025-01-11 12:12:52] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:12:52] [INFO] sleeping 10 seconds until next reconnection attempt
[2025-01-11 12:13:02] [INFO] checking state of node 1, 4 of 6 attempts
[2025-01-11 12:13:02] [WARNING] unable to ping "user=repmgr connect_timeout=2
dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr"
[2025-01-11 12:13:02] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:13:02] [INFO] sleeping 10 seconds until next reconnection attempt
[2025-01-11 12:13:12] [INFO] checking state of node 1, 5 of 6 attempts
[2025-01-11 12:13:12] [WARNING] unable to ping "user=repmgr connect_timeout=2
dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr"
[2025-01-11 12:13:12] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:13:12] [INFO] sleeping 10 seconds until next reconnection attempt
[2025-01-11 12:13:22] [INFO] checking state of node 1, 6 of 6 attempts
[2025-01-11 12:13:22] [WARNING] unable to ping "user=repmgr connect_timeout=2
dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr"
[2025-01-11 12:13:22] [DETAIL] PQping() returned "PQPING_NO_RESPONSE"
[2025-01-11 12:13:22] [WARNING] unable to reconnect to node 1 after 6 attempts
[2025-01-11 12:13:23] [NOTICE] witness node "witness_node" (ID: 3) now following new primary
node "original_secondary" (ID: 2)
[2025-01-11 12:13:23] [INFO] resuming witness monitoring mode
[2025-01-11 12:13:23] [DETAIL] following new primary "original_secondary" (ID: 2)
[2025-01-11 12:13:23] [INFO] witness monitoring connection to primary node "original_secondary"
(ID: 2)
[2025-01-11 12:18:24] [INFO] witness node "witness_node" (ID: 3) monitoring primary node
"original_secondary" (ID: 2) in normal state
```

```
[postgres@original_secondary repmgr]$ repmgr -f repmgr.conf cluster show --verbose
NOTICE: using provided configuration file "repmgr.conf"
INFO: connecting to database
ERROR: connection to database failed
DETAIL:
connection to server at "54.173.215.225", port 5432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?

DETAIL: attempted to connect using:
  user=repmgr connect_timeout=2 dbname=repmgr host=54.173.215.225 fallback_application_name=repmgr options=-csearch_path=
 ID | Name              | Role    | Status    | Upstream           | Location | Priority | Timeline | Connection string

----+-------------------+---------+-----------+--------------------+----------+----------+----------+-----------------------------------------------
-------------
 1  | original_primary  | primary | - failed  | ?                  | default  | 100      |          | host=54.173.215.225 user=repmgr dbname=repmgr conne
ct_timeout=2
 2  | original_secondary | primary | * running |                   | default  | 100      | 4        | host=54.224.11.184 user=repmgr dbname=repmgr connec
t_timeout=2
 3  | witness_node      | witness | * running | original_secondary | default  | 0        | n/a      | host=3.91.195.224 user=repmgr dbname=repmgr connect
_timeout=2

WARNING: following issues were detected
  - when attempting to connect to node "original_primary" (ID: 1), following error encountered :
"connection to server at "54.173.215.225", port 5432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?"

[postgres@original_secondary repmgr]$
```

Recovering the primary from the failover
Connecting to the secondary and creating some tables, so that we can verify the recovery.

```
postgres=# select pg_is_in_recovery();
pg_is_in_recovery
-------------------
f
(1 row)
```

```
postgres=# create database amol;
CREATE DATABASE
postgres=# \c amol
You are now connected to database "amol" as user "postgres".
amol=# create table table_1 as select now() c1;
SELECT 1
amol=# select count(*) from table_1 ;
count
-------
1
(1 row)

amol=# select * from table_1;
c1
-------------------------------
2025-01-11 12:19:47.254665+00
(1 row)
```

Perform checkpoint in the current primary (original_secondary) node:

```
[postgres@original_secondary repmgr]$ repmgr -f repmgr.conf node service --checkpoint
```

Run pg_rewind in the stopped cluster (original_primary) node.

Dry-run

```
[postgres@original_primary repmgr]$ repmgr node rejoin -f repmgr.conf -d
'host=54.224.11.184 user=repmgr dbname=repmgr' --force-rewind --dry-run
NOTICE: rejoin target is node "original_secondary" (ID: 2)
INFO: replication connection to the rejoin target node was successful
INFO: local and rejoin target system identifiers match
DETAIL: system identifier is 7458597170689229097
INFO: prerequisites for using pg_rewind are met
INFO: pg_rewind would now be executed
DETAIL: pg_rewind command is:
/usr/pgsql-17/bin/pg_rewind -D '/var/lib/pgsql/17/data' --source-server='host=54.224.11.184
user=repmgr dbname=repmgr connect_timeout=2'
INFO: prerequisites for executing NODE REJOIN are met
```

Actual run

```
[postgres@original_primary repmgr]$ repmgr node rejoin -f repmgr.conf -d
'host=54.224.11.184 user=repmgr dbname=repmgr' --force-rewind
NOTICE: rejoin target is node "original_secondary" (ID: 2)
NOTICE: executing pg_rewind
DETAIL: pg_rewind command is "/usr/pgsql-17/bin/pg_rewind -D '/var/lib/pgsql/17/data' --
source-server='host=54.224.11.184 user=repmgr dbname=repmgr connect_timeout=2'"
NOTICE: 0 files copied to /var/lib/pgsql/17/data
```

```
NOTICE: setting node 1's upstream to node 2
WARNING: unable to ping "host=54.173.215.225 user=repmgr dbname=repmgr
connect_timeout=2"
DETAIL: PQping() returned "PQPING_NO_RESPONSE"
NOTICE: starting server using "/usr/pgsql-17/bin/pg_ctl -w -D '/var/lib/pgsql/17/data' start"
NOTICE: NODE REJOIN successful
DETAIL: node 1 is now attached to node 2
```

```
[postgres@original_primary repmgr]$ repmgr -f repmgr.conf cluster show
 ID | Name              | Role    | Status    | Upstream           | Location | Priority | Timeline | Connection string
----+-------------------+---------+-----------+--------------------+----------+----------+----------+-----------------------------------------------
------------
 1  | original_primary  | standby |   running | original_secondary | default  | 100      | 3        | host=54.173.215.225 user=repmgr dbname=repmgr conne
ct_timeout=2
 2  | original_secondary | primary | * running |                    | default  | 100      | 4        | host=54.224.11.184 user=repmgr dbname=repmgr connec
t_timeout=2
 3  | witness_node      | witness | * running | original_secondary | default  | 0        | n/a      | host=3.91.195.224 user=repmgr dbname=repmgr connect
_timeout=2
```

Verification

```
postgres=# \l
                                                 List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+-------------+-------------+--------+-----------+-----------------------
 amol      | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |           |
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |           |
 repmgr    | repmgr   | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |           |
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |           | =c/postgres          +
           |          |          |                 |             |             |        |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |           | =c/postgres          +
           |          |          |                 |             |             |        |           | postgres=CTc/postgres
(5 rows)

postgres=# \c amol
You are now connected to database "amol" as user "postgres".
amol=# select pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 t
(1 row)

amol=#  select * from table_1;
             c1
-------------------------------
 2025-01-11 12:19:47.254665+00
(1 row)

amol=#  select count(*) from table_1;
 count
-------
     1
(1 row)
```

Setting Synchronous Replication

In the primary

```
postgres=# alter system set synchronous_standby_names='original_secondary';
ALTER SYSTEM

[postgres@original_primary repmgr]$ pg_ctl -D $PGDATA/ restart
```

In the secondary

```
postgres=# alter system set synchronous_standby_names='original_primary';
ALTER SYSTEM

[postgres@original_secondary repmgr]$ pg_ctl -D $PGDATA/ restart
```

Now, stop the secondary

```
[postgres@original_secondary repmgr]$ pg_ctl -D $PGDATA/ stop
```

Create a table/TX in the primary

```
postgres=# \timing
Timing is on.
postgres=# create table tab_sync_check as select 1;
```

The above statement will be in a hung state.
Now there are 2 conditions:
1. If you start the secondary, a table will be created

```
[postgres@original_secondary repmgr]$ pg_ctl -D $PGDATA/ start
```

```
postgres=# create table tab_sync_check as select 1;
SELECT 1
Time: 147543.125 ms (02:27.543)
```

2. If you hit control C i.e. canceled the TX, TX will get committed locally but not in the secondary

```
postgres=# create table tab_sync as select 1;
^CCancel request sent
WARNING:  canceling wait for synchronous replication due to user request
DETAIL:  The transaction has already committed locally, but might not have been replicated to the standby.
SELECT 1
```

We can verify the table creation if we try to recreate it once secondary comes back, we get an error:

```
postgres=# create table tab_sync as select 1;
ERROR:  relation "tab_sync" already exists
Time: 0.299 ms
```

Creating a custom/user managed tablespace

In the primary

```
[postgres@original_primary repmgr]$ mkdir $HOME/custom_ts_dir
[postgres@original_primary repmgr]$ chown postgres: $HOME/custom_ts_dir
[postgres@original_primary repmgr]$ ls -l $HOME/
total 0
drwx------. 4 postgres postgres 51 Jan 11 10:06 17
drwxr-xr-x. 2 postgres postgres 6 Jan 11 12:54 custom_ts_dir
drwxr-xr-x. 2 postgres postgres 47 Jan 11 12:29 repmgr
[postgres@original_primary repmgr]$ ls -l $HOME/custom_ts_dir/
total 0
```

In the secondary run the same above commands. Basically, we are creating a same directory structure in primary and secondary nodes.

```
[postgres@original_secondary repmgr]$ mkdir $HOME/custom_ts_dir
[postgres@original_secondary repmgr]$ chown postgres: $HOME/custom_ts_dir
[postgres@original_secondary repmgr]$ ls -l $HOME/
total 0
drwx------. 4 postgres postgres 51 Jan 11 10:06 17
drwxr-xr-x. 2 postgres postgres 6 Jan 11 12:56 custom_ts_dir
drwxr-xr-x. 2 postgres postgres 47 Jan 11 12:04 repmgr
[postgres@original_secondary repmgr]$ ls -l $HOME/custom_ts_dir/
total 0
```

In the primary, create tablespace, a database and table on that TS

```
postgres=# \db
List of tablespaces
Name | Owner | Location
------------+----------+----------
pg_default | postgres |
pg_global | postgres |
(2 rows)
postgres=# CREATE TABLESPACE custom_ts OWNER postgres LOCATION
'/var/lib/pgsql/custom_ts_dir';
CREATE TABLESPACE
postgres=# create database application_db WITH OWNER = postgres TABLESPACE =
custom_ts;
CREATE DATABASE
postgres=# create table app_tab_1 (c1 int) TABLESPACE custom_ts;
CREATE TABLE
postgres=# \l+ application_db
List of databases
Name | Owner | Encoding | Locale Provider | Collate | Ctype | Locale | ICU Rules | Access
privileges | Size | Tablespace | Desc
ription
```

```
---------------+----------+----------+----------------+------------+------------+--------+-----------+-------------
-----+---------+------------+-----
--------
application_db | postgres | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 | | | | 7409 kB |
custom_ts |
(1 row)

postgres=# \db+ custom_ts
List of tablespaces
Name | Owner | Location | Access privileges | Options | Size | Description
-----------+----------+----------------------------+-------------------+---------+---------+-------------
custom_ts | postgres | /var/lib/pgsql/custom_ts_dir | | | 7417 kB |
(1 row)
postgres=# \d+ app_tab_1
Table "public.app_tab_1"
Column | Type | Collation | Nullable | Default | Storage | Compression | Stats target |
Description
--------+---------+-----------+----------+---------+---------+-------------+--------------+-------------
c1 | integer | | | | plain | | |
Tablespace: "custom_ts"
Access method: heap
```

```
postgres=# \l+ application_db
                                                                    List of databases
      Name      |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | Locale | ICU Rules | Access privileges |  Size   | Tablespace | Desc
ription
----------------+----------+----------+-----------------+-------------+-------------+--------+-----------+-------------------+---------+------------+-----
--------
 application_db | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |        |           |                   | 7409 kB | custom_ts  |
(1 row)

postgres=# \db+ custom_ts
                                    List of tablespaces
   Name    |  Owner   |          Location          | Access privileges | Options |  Size   | Description
-----------+----------+----------------------------+-------------------+---------+---------+-------------
 custom_ts | postgres | /var/lib/pgsql/custom_ts_dir |                   |         | 7417 kB |
(1 row)

postgres=# \d+ app_tab_1
                                Table "public.app_tab_1"
 Column |  Type   | Collation | Nullable | Default | Storage | Compression | Stats target | Description
--------+---------+-----------+----------+---------+---------+-------------+--------------+-------------
 c1     | integer |           |          |         | plain   |             |              |
Tablespace: "custom_ts"
Access method: heap
```


We can see that the below directory was created automatically in the secondary

```
[postgres@original_secondary repmgr]$ ls -l $HOME/custom_ts_dir/
total 0
drwx------. 2 postgres postgres 6 Jan 11 12:59 PG_17_202406281
[postgres@original_secondary repmgr]$ ls -l $HOME/custom_ts_dir/PG_17_202406281/
total 12
drwx------. 2 postgres postgres 8192 Jan 11 13:00 16468
drwx------. 2 postgres postgres 19 Jan 11 13:02 5
```