

# How to Create Your Own Extension for Setting Up PostgreSQL Replication Using C Functions.

1. Install PostgreSQL v16 and perform initDB

```
1 ./initdb /var/lib/pgsql/16/data/
```

2. Install these additional packages via root user.

```
1 dnf install epel-release -y
2 dnf --enablerepo=powertools install perl-IPC-Run -y
3 sudo dnf install postgresql16-devel
4 sudo dnf install wget
5 dnf install make
6 sudo dnf groupinstall "Development Tools"
7 sudo dnf install readline-devel
8 dnf install passwd
9
```

3. Change the password of postgres user and add it in sudo group

```
1 sudo passwd postgres
2 sudo usermod -aG wheel postgres
```

4. Set the following environment variables.

```
1 su - postgres
2 export PATH=/usr/pgsql-16/bin:$PATH
3 export pg_config=/usr/pgsql-16/bin/pg_config
```

5. Download and extract the source code for PostgreSQL 16.4.

```
1 cd /var/lib/pgsql/16
2 mkdir src
3 cd src/
```

```
1 wget https://ftp.postgresql.org/pub/source/v16.4/postgresql-16.4.tar.gz
```

```
1 tar -xvf postgresql-16.4.tar.gz
2 cd postgresql-16.4
```

6. Execute the configure and make commands in the postgresql-16.4 directory.

```
1 ./configure --prefix=/usr/pgsql-16/
2 make
```

7. Create a directory and add the replication setup functions from the file replication.c with the specified contents.

```
1 cd /var/lib/pgsql/16/
2 mkdir repsetup
3 cd /var/lib/pgsql/16/repsetup
```

```
1 vi replication.c
```

```
1 #include "postgres.h"
2 #include "fmgr.h"
```

```

3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <unistd.h>
6 #include <fcntl.h>
7
8 #ifdef PG_MODULE_MAGIC
9 PG_MODULE_MAGIC;
10 #endif
11
12 // Function to execute shell commands
13 static int exec_command(const char *command) {
14     int result = system(command);
15     if (result != 0) {
16         elog(ERROR, "Command failed: %s", command);
17     }
18     return result;
19 }
20
21 // Function to configure the primary node
22 void configure_primary() {
23     FILE *conf_file = fopen("/var/lib/pgsql/16/data/postgresql.conf", "a");
24     if (conf_file == NULL) {
25         elog(ERROR, "Failed to open postgresql.conf for primary. Check file path and permissions.");
26         return;
27     }
28     fprintf(conf_file, "\nwal_level = replica\n");
29     fprintf(conf_file, "max_wal_senders = 10\n");
30     fprintf(conf_file, "hot_standby = on\n");
31
32     // Ensure the changes are written to disk
33     fflush(conf_file);
34     fsync(fileno(conf_file));
35     fclose(conf_file);
36
37     // Reload PostgreSQL to apply the new configuration
38     exec_command("/usr/pgsql-16/bin/pg_ctl reload -D /var/lib/pgsql/16/data/");
39 }
40
41 // Function to perform base backup for standby node
42 void perform_base_backup(const char *standby_dir) {
43     char command[256];
44     snprintf(command, sizeof(command), "/usr/pgsql-16/bin/pg_basebackup -D %s -Fp -Xs -P -R", standby_dir);
45     exec_command(command);
46 }
47
48 // Function to configure a standby node with a specific port
49 void configure_standby(const char *standby_dir, const char *standby_name, int port, int primary_port) {
50     char conf_path[256];
51     snprintf(conf_path, sizeof(conf_path), "%s/postgresql.conf", standby_dir);
52
53     FILE *conf_file = fopen(conf_path, "a");
54     if (conf_file == NULL) {
55         elog(ERROR, "Failed to open postgresql.conf for standby %s. Check file path and permissions.",
56 standby_name);
57         return;
58     }
59     fprintf(conf_file, "\nport = %d\n", port);

```

```

59     fprintf(conf_file, "primary_conninfo = 'host=localhost port=%d user=postgres password=test'\n",
primary_port);
60
61     fflush(conf_file);
62     fsync(fileno(conf_file));
63     fclose(conf_file);
64
65     // Start the standby node
66     char command[256];
67     snprintf(command, sizeof(command), "/usr/pgsql-16/bin/pg_ctl -D %s start", standby_dir);
68     exec_command(command);
69 }
70
71 // Main function to set up replication with 3 nodes and different ports
72 PG_FUNCTION_INFO_V1(setup_replication);
73
74 Datum
75 setup_replication(PG_FUNCTION_ARGS) {
76     int primary_port = 5432; // Port for the primary
77     int standby1_port = 5433; // Port for standby node 1
78     int standby2_port = 5434; // Port for standby node 2
79     elog(INFO, "Configuring primary node on port %d", primary_port);
80     configure_primary();
81
82     elog(INFO, "Performing base backup for standby node 1...");
83     perform_base_backup("/var/lib/pgsql/standby_data_1");
84     elog(INFO, "Performing base backup for standby node 2...");
85     perform_base_backup("/var/lib/pgsql/standby_data_2");
86
87     elog(INFO, psprintf("Configuring standby node 1 on port %d...", standby1_port));
88     configure_standby("/var/lib/pgsql/standby_data_1", "standby_1", standby1_port, primary_port);
89
90     elog(INFO, psprintf("Configuring standby node 2 on port %d...", standby2_port));
91     configure_standby("/var/lib/pgsql/standby_data_2", "standby_2", standby2_port, primary_port);
92
93     elog(INFO, "Replication setup complete for 1 primary and 2 standby nodes!");
94     PG_RETURN_VOID();
95 }

```

8. Create a Makefile with the specified contents in the C function directory.

```

1 vi Makefile
2
3 MODULES = replication
4 EXTENSION = replication
5 DATA = replication--1.0.sql
6 PG_CONFIG = /usr/pgsql-16/bin/pg_config
7 PGXS = /var/lib/pgsql/16/src/postgresql-16.4/src/makefiles/pgxs.mk
8 include $(PGXS)

```

9. Create an Extension Control File

```

1 vi replication.control
2
3 # replication.control
4 comment = 'Extension to set up PostgreSQL replication using C'
5 default_version = '1.0'
6 relocatable = false

```

```
5 module_pathname = 'replication'
```

#### 10. Create an SQL File for the Extension

```
1 vi replication--1.0.sql
```

```
1 -- replication--1.0.sql
2 CREATE FUNCTION setup_replication()
3 RETURNS void
4 LANGUAGE c
5 AS 'replication', 'setup_replication';
```

#### 11. Execute the Make command in the C function directory.

```
1 make
```

#### 12. Ensure that the `replication.so` file is generated after running the make command.

```
1 ls
2 Makefile replication--1.0.sql replication.c replication.control replication.o replication.so
```

#### 13. Now execute the make install command

```
1 sudo -E make install
```

#### 14. Connect to the PostgreSQL database and create the extension.

```
1 psql -U postgres -d postgres
2 psql (16.4)
3 Type "help" for help.
```

```
1 create extension replication;
```

```
1 \dx
2
3          List of installed extensions
4
5  Name      | Version | Schema  | Description
6  -----+-----+-----+-----
7  plpgsql   | 1.0     | pg_catalog | PL/pgSQL procedural language
8  replication | 1.0     | public   | Extension to set up PostgreSQL replication using C
9  (2 rows)
```

#### 15. Call the extension's function now.

```
1 SELECT setup_replication();
2 INFO: Configuring primary node on port 5432
3 INFO: Performing base backup for standby node 1...
4 INFO: Performing base backup for standby node 2...
5 INFO: Configuring standby node 1 on port 5433...
6 INFO: Configuring standby node 2 on port 5434...
7 INFO: Replication setup complete for 1 primary and 2 standby nodes!
8 setup_replication
9 -----
10
11 (1 row)
```

#### 16. Verify replication.

Create the table and data on the primary node.

```
1 CREATE TABLE cities (
```

```

2      id SERIAL PRIMARY KEY,
3      name VARCHAR(50),
4      population INTEGER
5  );
6  INSERT INTO cities (name, population)
7  VALUES
8  ('New York', 8419000),
9  ('Los Angeles', 3980000),
10 ('Chicago', 2716000);

```

Verify data on standby node 1.

```

1  psql -U postgres -p 5433
2  psql (16.4)
3  Type "help" for help.
4
5  postgres=# select * from cities;
6  id |   name   | population
7  ----+-----+-----
8   1 | New York |   8419000
9   2 | Los Angeles | 3980000
10  3 | Chicago  | 2716000
11 (3 rows)

```

Verify data on standby node 2.

```

1  psql -U postgres -p 5434
2  psql (16.4)
3  Type "help" for help.
4
5  postgres=# select * from cities;
6  id |   name   | population
7  ----+-----+-----
8   1 | New York |   8419000
9   2 | Los Angeles | 3980000
10  3 | Chicago  | 2716000
11 (3 rows)

```