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

1. Install PostgreSQL v16 and perform initDB

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

2. Install these additional packages via root user.

```
dnf install epel-release -y
dnf --enablerepo=powertools install perl-IPC-Run -y
sudo dnf install postgresql16-devel
sudo dnf install wget
dnf install make
sudo dnf groupinstall "Development Tools"
sudo dnf install readline-devel
dnf install passwd

```

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

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

4. Set the following environment variables.

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

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

```
cd /var/lib/pgsql/16
mkdir src
cd src/
```

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

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

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

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

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

```
cd /var/lib/pgsql/16/
mkdir repsetup
cd /var/lib/pgsql/16/repsetup
```

```
vi replication.c
```

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

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#ifdef PG_MODULE_MAGIC
PG_MODULE_MAGIC;
#endif

// Function to execute shell commands
static int exec_command(const char *command) {
    int result = system(command);
    if (result != 0) {
        elog(ERROR, "Command failed: %s", command);
    }
    return result;
}

// Function to configure the primary node
void configure_primary() {
    FILE *conf_file = fopen("/var/lib/pgsql/16/data/postgresql.conf", "a");
    if (conf_file == NULL) {
        elog(ERROR, "Failed to open postgresql.conf for primary. Check file path and permissions.");
        return;
    }
    fprintf(conf_file, "\nwal_level = replica\n");
    fprintf(conf_file, "max_wal_senders = 10\n");
    fprintf(conf_file, "hot_standby = on\n");

    // Ensure the changes are written to disk
    fflush(conf_file);
    fsync(fileno(conf_file));
    fclose(conf_file);

    // Reload PostgreSQL to apply the new configuration
    exec_command("/usr/pgsql-16/bin/pg_ctl reload -D /var/lib/pgsql/16/data/");
}

// Function to perform base backup for standby node
void perform_base_backup(const char *standby_dir) {
    char command[256];
    snprintf(command, sizeof(command), "/usr/pgsql-16/bin/pg_basebackup -D %s -Fp -Xs -P -R", standby_dir);
    exec_command(command);
}

// Function to configure a standby node with a specific port
void configure_standby(const char *standby_dir, const char *standby_name, int port, int primary_port) {
    char conf_path[256];
    snprintf(conf_path, sizeof(conf_path), "%s/postgresql.conf", standby_dir);

    FILE *conf_file = fopen(conf_path, "a");
    if (conf_file == NULL) {
        elog(ERROR, "Failed to open postgresql.conf for standby %s. Check file path and permissions.",
standby_name);
        return;
    }
    fprintf(conf_file, "\nport = %d\n", port);
```

```
    fprintf(conf_file, "primary_conninfo = 'host=localhost port=%d user=postgres password=test'\n",
primary_port);

    fflush(conf_file);
    fsync(fileno(conf_file));
    fclose(conf_file);

    // Start the standby node
    char command[256];
    snprintf(command, sizeof(command), "/usr/pgsql-16/bin/pg_ctl -D %s start", standby_dir);
    exec_command(command);
}

// Main function to set up replication with 3 nodes and different ports
PG_FUNCTION_INFO_V1(setup_replication);

Datum
setup_replication(PG_FUNCTION_ARGS) {
    int primary_port = 5432;  // Port for the primary
    int standby1_port = 5433; // Port for standby node 1
    int standby2_port = 5434; // Port for standby node 2
    elog(INFO, "Configuring primary node on port %d", primary_port);
    configure_primary();

    elog(INFO, "Performing base backup for standby node 1...");
    perform_base_backup("/var/lib/pgsql/standby_data_1");
    elog(INFO, "Performing base backup for standby node 2...");
    perform_base_backup("/var/lib/pgsql/standby_data_2");

    elog(INFO, psprintf("Configuring standby node 1 on port %d...", standby1_port));
    configure_standby("/var/lib/pgsql/standby_data_1", "standby_1", standby1_port, primary_port);

    elog(INFO, psprintf("Configuring standby node 2 on port %d...", standby2_port));
    configure_standby("/var/lib/pgsql/standby_data_2", "standby_2", standby2_port, primary_port);

    elog(INFO, "Replication setup complete for 1 primary and 2 standby nodes!");
    PG_RETURN_VOID();
}
```

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

```
vi Makefile
```

```
MODULES = replication
EXTENSION = replication
DATA = replication--1.0.sql
PG_CONFIG = /usr/pgsql-16/bin/pg_config
PGXS = /var/lib/pgsql/16/src/postgresql-16.4/src/makefiles/pgxs.mk
include $(PGXS)
```

9. Create an Extension Control File

```
vi replication.control
```

```
# replication.control
comment = 'Extension to set up PostgreSQL replication using C'
default_version = '1.0'
relocatable = false
```

```
module_pathname = 'replication'
```

10. Create an SQL File for the Extension

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

```
-- replication--1.0.sql
CREATE FUNCTION setup_replication()
RETURNS void
LANGUAGE c
AS 'replication', 'setup_replication';
```

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

```
make
```

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

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

13. Now execute the make install command

```
 sudo -E make install
```

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

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

```
create extension replication;
```

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

15. Call the extension's function now.

```
SELECT setup_replication();
INFO:  Configuring primary node on port 5432
INFO:  Performing base backup for standby node 1...
INFO:  Performing base backup for standby node 2...
INFO:  Configuring standby node 1 on port 5433...
INFO:  Configuring standby node 2 on port 5434...
INFO:  Replication setup complete for 1 primary and 2 standby nodes!
 setup_replication
-------------------

(1 row)
```

16. Verify replication.

Create the table and data on the primary node.

```
CREATE TABLE cities (
```

```
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    population INTEGER
);
INSERT INTO cities (name, population)
VALUES
('New York', 8419000),
('Los Angeles', 3980000),
('Chicago', 2716000);
```

Verify data on standby node 1.

```
 psql -U postgres -p 5433
psql (16.4)
Type "help" for help.

postgres=# select * from cities;
 id |    name     | population
----+-------------+------------
  1 | New York    |    8419000
  2 | Los Angeles |    3980000
  3 | Chicago     |    2716000
(3 rows)
```

Verify data on standby node 2.

```
psql -U postgres -p 5434
psql (16.4)
Type "help" for help.

postgres=# select * from cities;
 id |    name     | population
----+-------------+------------
  1 | New York    |    8419000
  2 | Los Angeles |    3980000
  3 | Chicago     |    2716000
(3 rows)
```