"Iqra' bismi rabbika allathee khalaq, khalaqal insana min 'alaq."

Simple Guide to PostgreSQL Installation, Replication and Backup

Assumptions:

- PostgreSQL Version: 16 or newer
- Primary Server: 192.168.20.21 (hostname: pg-primary)
- Replica Server: 192.168.20.22 (hostname: pg-replica)
- OS: Ubuntu 24.04LTS (adjustable for RHEL/CentOS/Almalinux)
- Replication User: replicator
- Data Directory: /custom_pg_data
- Network: Both servers can communicate with port 5432
- Disk Setup: External disks /dev/sdb and /dev/sdc for LVM

Step 0: Prepare the Environment (Both Servers)

1. Set Hostnames
   On Primary:
   hostnamectl set-hostname pg-primary; bash

   On Replica:
   hostnamectl set-hostname pg-replica; bash

2. Update /etc/hosts
   On both servers:
   cat >> /etc/hosts <<EOF
   192.168.20.21 primary.yourdomain.com pg-primary
   192.168.20.22 replica.yourdomain.com pg-replica
   EOF

3. Set Up LVM for Custom Data Directory
   a. Partition External Disks:
      fdisk /dev/sdb
      - Press 'n' for new partition, 'p' for primary, accept defaults, then 'w' to write.

      fdisk /dev/sdc (repeat steps)

b. Verify Partitions:

root@pg-primary:~# lsblk

Expected output:

```
NAME                      MAJ:MIN  RM  SIZE  RO  TYPE  MOUNTPOINTS
sda                         8:0    0   50G   0  disk
├─sda1                      8:1    0    1M   0  part
├─sda2                      8:2    0    2G   0  part  /boot
└─sda3                      8:3    0   48G   0  part
  └─ubuntu--vg-ubuntu--lv 252:0    0   48G   0  lvm   /
sdb                         8:16   0   50G   0  disk
└─sdb1                      8:17   0   50G   0  part
sdc                         8:32   0   50G   0  disk
└─sdc1                      8:33   0   50G   0  part
sr0                         11:0   1  2.6G   0  rom
```

c. Create LVM:

```
sudo pvcreate /dev/sdb1 /dev/sdc1
sudo vgcreate pg_vg /dev/sdb1 /dev/sdc1
sudo lvcreate -n pg_lv -L 50G pg_vg
```

d. Format and Mount Logical Volume:

```
sudo mkfs.ext4 /dev/pg_vg/pg_lv
sudo mkdir -p /custom_pg_data
sudo mount /dev/pg_vg/pg_lv /custom_pg_data/
echo '/dev/pg_vg/pg_lv /custom_pg_data ext4        defaults    0      0' | sudo tee -a /etc/fstab
```

e. Verify LVM:

```
sudo lvscan
sudo blkid /dev/pg_vg/pg_lv
```

f. Activate Logical Volume (if inactive):

```
sudo lvchange -ay /dev/pg_vg/pg_lv
```

```
root@pg-primary:~# lsblk
NAME                      MAJ:MIN  RM  SIZE  RO  TYPE  MOUNTPOINTS
sda                         8:0    0   50G   0  disk
├─sda1                      8:1    0    1M   0  part
├─sda2                      8:2    0    2G   0  part  /boot
└─sda3                      8:3    0   48G   0  part
  └─ubuntu--vg-ubuntu--lv 252:0    0   48G   0  lvm   /
sdb                         8:16   0   50G   0  disk
└─sdb1                      8:17   0   50G   0  part
  └─pg_vg-pg_lv             252:1  0   50G   0  lvm   /custom_pg_data
sdc                         8:32   0   50G   0  disk
└─sdc1                      8:33   0   50G   0  part
  └─pg_vg-pg_lv             252:1  0   50G   0  lvm   /custom_pg_data
sr0                         11:0   1  2.6G   0  rom
```

Step 1: Install PostgreSQL (Both Servers)

1. Update and Install PostgreSQL:

```
sudo apt update -y

sudo apt upgrade -y

sudo apt install postgresql postgresql-contrib postgresql-client -y
```

2. Verify Installation:

```
pg_config --version
```

Expected output: PostgreSQL 16.x

3. Stop PostgreSQL:

```
sudo systemctl stop postgresql

sudo systemctl disable postgresql.service
```

4. Set Up Custom Data Directory:

```
sudo chown postgres:postgres /custom_pg_data

sudo chmod -R 700 /custom_pg_data

sudo rm -rf /custom_pg_data/*

sudo -u postgres /usr/lib/postgresql/16/bin/initdb  -D  /custom_pg_data
```

5. Update PostgreSQL Configuration:

```
sudo sed -i "s|^#*data_directory = .*|data_directory = '/custom_pg_data'|"
/etc/postgresql/16/main/postgresql.conf
```

6. Modify Authentication:

Edit /custom_pg_data/pg_hba.conf:

Change:

```
local    all    all    trust
```

To:

```
local    all    all    md5
```

7. Start PostgreSQL and Verify:

```
sudo systemctl start postgresql
sudo -u postgres psql -c "SHOW data_directory;"
```

Expected output:

```
data_directory
---------------------
/custom_pg_data
```

Step 2: Configure the Primary Server

1. Edit postgresql.conf:
   sudo vi /etc/postgresql/16/main/postgresql.conf
   Ensure:
   listen_addresses = '*'
   wal_level = replica
   max_wal_senders = 10
   wal_keep_size = 512MB
   max_replication_slots = 5
   hot_standby = on
   archive_mode = on
   archive_command = 'cp %p /var/lib/postgresql/wal_archive/%f'

Or use sed:
sudo sed -i 's/^#listen_addresses = .*/listen_addresses = "*"/' /etc/postgresql/16/main/postgresql.conf
sudo sed -i 's/^#wal_level = .*/wal_level = replica/' /etc/postgresql/16/main/postgresql.conf
sudo sed -i 's/^#max_wal_senders = .*/max_wal_senders = 10/' etc/postgresql/16/main/postgresql.conf
sudo sed -i 's/^#wal_keep_size = .*/wal_keep_size = 512MB/' /etc/postgresql/16/main/postgresql.conf
sudo sed -i 's/^#max_replication_slots = .*/max_replication_slots = 5/'
etc/postgresql/16/main/postgresql.conf
sudo sed -i 's/^#hot_standby = .*/hot_standby = on/' /etc/postgresql/16/main/postgresql.conf
sudo sed -i 's/^#archive_mode = .*/archive_mode = on/' /etc/postgresql/16/main/postgresql.conf
sudo sed -i "s|^#archive_command = .*|archive_command = 'cp %p
/var/lib/postgresql/wal_archive/%f'|" /etc/postgresql/16/main/postgresql.conf

2. Create WAL Archive Directory:

   sudo mkdir -p /var/lib/postgresql/wal_archive
   sudo chown postgres:postgres /var/lib/postgresql/wal_archive
   sudo chmod 700 /var/lib/postgresql/wal_archive

3. Edit pg_hba.conf:

   sudo vi /etc/postgresql/16/main/pg_hba.conf
   Add:
   host replication replicator 192.168.20.22/32 md5
   Or:
   echo "host replication replicator 192.168.20.22/32 md5" | sudo tee -a
/etc/postgresql/16/main/pg_hba.conf

4. Reload Configuration:

```
sudo systemctl reload postgresql
```

5. Create Replication User:

```
sudo -u postgres psql -c "CREATE USER replicator WITH REPLICATION ENCRYPTED PASSWORD 'replica_pass';"
```

6. Restart PostgreSQL:

```
sudo systemctl restart postgresql
```

7. Verify Replication Setup:

```
sudo -u postgres psql -c "SELECT * FROM pg_stat_replication;"
```

(No output expected yet)

Step 3: Set Up the Replica Server

1. Stop PostgreSQL:

```
sudo systemctl stop postgresql
```

2. Clear Data Directory:

```
sudo rm -rf /custom_pg_data/*
```

3. Take Base Backup from Primary:

```
PGPASSWORD='replica_pass' pg_basebackup -h 192.168.20.21 -D /custom_pg_data -U replicator -Fp -Xs -P -R
```

-R: Creates standby.signal and sets up primary_conninfo
-Xs: Includes WAL files
-Fp: Plain format

4. Set Permissions:

```
sudo chown -R postgres:postgres /custom_pg_data
sudo chown -R postgres:postgres /custom_pg_data/*
sudo chmod -R 700 /custom_pg_data
sudo chmod 600 /custom_pg_data/postgresql.auto.conf
```

5. Verify postgresql.auto.conf:

```
cat /custom_pg_data/postgresql.auto.conf
```

Ensure:

```
primary_conninfo = 'host=192.168.20.21 port=5432 user=replicator password=replica_pass'
```

6. Create WAL Archive Directory:

```
sudo mkdir -p /custom_pg_data/archive
```

sudo chown postgres:postgres /custom_pg_data/archive
sudo chmod 700 /custom_pg_data/archive

7. Update postgresql.conf:
sudo vi /custom_pg_data/postgresql.conf
Add/verify:
primary_conninfo = 'host=192.168.20.21 port=5432 user=replicator password=replica_pass'
restore_command = 'cp /custom_pg_data/archive/%f %p'
hot_standby = on

8. Start PostgreSQL:
sudo systemctl start postgresql
sudo systemctl enable postgresql

9. Check Logs:
sudo journalctl -u postgresql -f

Step 4: Monitoring and Verification

On Primary:
sudo -u postgres psql -c "SELECT pid, usename, client_addr, state, sync_state FROM pg_stat_replication;"
(Expected: Replica (192.168.20.22) with state=streaming)

```
 pid  | usename   | client_addr |   state   | sync_state
-------+------------+---------------+-----------+------------
 40179 | replicator | 192.168.20.41 | streaming | async
(1 row)
```

On Replica:
sudo -u postgres psql -c "SELECT status, receive_start_lsn, replay_lsn FROM pg_stat_wal_receiver;"
(Expected: status=streaming)

```
  status   | receive_start_lsn
-----------+-------------------
 streaming | 0/5000000
(1 row)
```

Step 5: Test Replication

1. On Primary:
   sudo -u postgres psql -c "CREATE DATABASE testdb;"
   sudo -u postgres psql -d testdb -c "CREATE TABLE test_replication (id SERIAL, name TEXT);"
   sudo -u postgres psql -d testdb -c "INSERT INTO test_replication (name) VALUES ('replicated\!');"

2. On Replica:
   sudo -u postgres psql -d testdb -c "SELECT * FROM test_replication;"
   Expected:

```
id |     name
----+--------------
 1 | replicated\!
(1 row)
```

Step 6: Configure Replication Slot (Prevents WAL loss)

1. On Primary:
   sudo -u postgres psql -c "SELECT * FROM pg_create_physical_replication_slot('replica_slot');"

2. On Replica:
   Edit /custom_pg_data/postgresql.conf:
   primary_slot_name = 'replica_slot'

3. Restart PostgreSQL on Replica:
   sudo systemctl restart postgresql

4. Verify Slot Usage on Primary:
   sudo -u postgres psql -c "SELECT slot_name, active FROM pg_replication_slots;"
   Expected: replica_slot with active=true

Final Step: Database Backup using bash Script

```
#!/bin/bash

# === CONFIGURATION ===
DB_NAME="yourdatabase"
BACKUP_DIR="/yourdirectory"
DATE=$(date +"%Y-%m-%d")
BACKUP_FILE="${BACKUP_DIR}/${DB_NAME}_backup_${DATE}.sql.gz"  # Adding compression
RETENTION_DAYS=15

# === CHECK IF BACKUP DIRECTORY EXISTS ===
if [ ! -d "$BACKUP_DIR" ]; then
   echo "[ERROR] Backup directory $BACKUP_DIR does not exist: $BACKUP_DIR" >&2
   exit 1
fi

# === CREATE BACKUP ===
echo "[INFO] $(date) - Starting backup for database: $DB_NAME"
su - postgres -c "pg_dump -d $DB_NAME -Fc | gzip > $BACKUP_FILE"  # Compress the backup with gzip

if [ $? -eq 0 ]; then
   echo "[INFO] $(date) - Backup successful: $BACKUP_FILE"
else
   echo "[ERROR] $(date) - Backup failed" >&2
   exit 1
fi

# === DELETE OLD BACKUPS (older than $RETENTION_DAYS days) ===
echo "[INFO] $(date) - Deleting backups older than $RETENTION_DAYS days from $BACKUP_DIR"
find "$BACKUP_DIR" -name "${DB_NAME}_backup_*.sql.gz" -type f -mtime +$RETENTION_DAYS -exec
rm -f {} \;

if [ $? -eq 0 ]; then
   echo "[INFO] $(date) - Cleanup complete."
else
   echo "[ERROR] $(date) - Failed to clean up old backups" >&2
   exit 1
fi
```

#Crontab Entry:

```
sudo nano /etc/cron.d/db_backup
0 2 * * * postgres /bin/bash /path/to/your/backup_script.sh >> /path/to/backup/logs/backup_log.log
2>&1
```