**PostgreSQL Scripts Documentation**

**1. promote\_PR.sh**

**Purpose:** This script promotes a standby (DR) site to become the new primary (PR) site.

**Description:**

* Stops the PostgreSQL service on the current primary (if needed).
* Updates the PostgreSQL configuration to reflect the new primary site.

**Dependencies:**

* The script should be run with appropriate permissions.
* Environmental variables to be updated as per setup

**Example:**

bash

Copy code

# Make sure the script is executable

chmod +x promote\_PR.sh

# Run the script

./promote\_PR.sh

**Script Content:**

bash

#!/bin/bash

# Configuration variables

PGDATA="/postgres/data/" # Path to PostgreSQL data directory

PGUSER="postgres" # PostgreSQL superuser

PGCTL="/usr/pgsql-16/bin/pg\_ctl" # Path to pg\_ctl command

PORT=6444

DBNAME=postgres

DBUSERNAME=postgres

# Function to promote standby to primary

promote\_standby() {

echo "Promoting standby to primary..."

$PGCTL promote -D "$PGDATA"

if [ $? -eq 0 ]; then

echo "Standby promotion successful."

else

echo "Standby promotion failed."

exit 1

fi

}

# Check if the server is currently in standby mode

is\_standby() {

local status

status=$($psql -p 6444 -t -c "SELECT pg\_is\_in\_recovery();" | tr -d '[:space:]')

if [ "$status" = "t" ]; then

echo "The server is in recovery mode."

else

echo "The server is not in recovery mode."

fi

}

# Main script execution

echo "Checking if the server is in standby mode..."

if is\_standby; then

promote\_standby

else

echo "The server is not in standby mode. Promotion is not necessary."

exit 0

fi

**2. standby\_DR.sh**

**Purpose:** This script converts the old primary site to a standby site and updates its configuration to sync with the new primary.

**Usage:**

bash

Copy code

./standby\_DR.sh

**Description:**

* Stops the PostgreSQL service on the old primary.
* Configures the old primary to act as a standby for the new primary.
* Updates postgresql.auto.conf with the new primary’s details.
* Starts the PostgreSQL service on the old primary in standby mode.

**Dependencies:**

* Ensure correct paths and configurations.
* Environmental variables to be updated as per setup

**Example:**

#!/bin/bash

# Variables

STANDBY\_DATA\_DIR="/postgres/data/"

NEW\_PRIMARY\_HOST="10.2.0.22"

REPLICATION\_USER="postgres"

PG\_CTL="/usr/pgsql-16/bin/pg\_ctl" # Path to pg\_ctl binary

PG\_BIN\_DIR="/usr/pgsql-16/bin/" # Path to PostgreSQL binaries

LOG\_FILE="/postgres/standby.log" # Path to PostgreSQL log file

PORT=6444

PG\_PORT=6444

DR\_HOST="10.2.0.21"

PRIMARY\_HOST="10.2.0.22"

# Function to check for errors

check\_error() {

if [ $? -ne 0 ]; then

echo "Error: $1" >&2

exit 1

fi

}

# Stop PostgreSQL on the old standby (now the standby)

echo "Stopping PostgreSQL service on the standby server..."

$PG\_CTL stop -D $STANDBY\_DATA\_DIR -s -m fast

check\_error "Failed to stop PostgreSQL service on the standby server."

# Start PostgreSQL in single-user mode to update configuration

echo "Updating replication settings in postgresql.auto.conf..."

$PG\_CTL start -D $STANDBY\_DATA\_DIR -l $LOG\_FILE -w

check\_error "Failed to start PostgreSQL service in single-user mode."

# Apply replication settings

echo "Applying replication settings using ALTER SYSTEM..."

PGPASSWORD=postgres@12345 $PG\_BIN\_DIR/psql -U $REPLICATION\_USER -p ${PORT} -d postgres -c "ALTER SYSTEM SET primary\_conninfo TO 'host=${NEW\_PRIMARY\_HOST} port=${PG\_PORT} user=${REPLICATION\_USER} password=postgres@123';"

check\_error "Failed to update primary\_conninfo."

# Reload configuration

echo "Reloading PostgreSQL configuration..."

PGPASSWORD=postgres@12345 $PG\_BIN\_DIR/psql -U $REPLICATION\_USER -p ${PORT} -d postgres -c "SELECT pg\_reload\_conf();"

check\_error "Failed to reload PostgreSQL configuration."

touch $STANDBY\_DATA\_DIR/standby.signal

echo "DR changes complete now execute promte script on new PR"

**3. replication\_verify.sh**

**Purpose:** This script checks the replication status on both the primary and standby sites.

**Usage:**

bash

Copy code

./replication\_verify.sh

**Description:**

* Connects to the primary server and retrieves replication status.
* Connects to the standby server and retrieves the WAL receiver status.

**Dependencies:**

* Ensure correct paths and configurations.
* Environmental variables to be updated as per setup

**Example:**

bash

Copy code

# Make sure the script is executable

chmod +x replication\_verify.sh

# Run the script

./replication\_verify.sh

**Script Content:**

# Define variables

PRIMARY\_HOST="10.2.0.22"

PORT="6444"

REPLICA\_USER="postgres"

REPLICA\_HOST="10.2.0.21"

OLD\_PRIMARY\_HOST="10.2.0.22"

OLD\_PORT="6444"

NEW\_PRIMARY\_HOST="10.2.0.21"

REPLICA\_USER="postgres"

PRIMARY\_DATA\_DIR="postgres/data"

REPLICA\_DATA\_DIR="/postgres/data"

# Connect to the primary server and check replication status

echo "Checking the replication on PR..."

psql -h "$PRIMARY\_HOST" -p "$PORT" -U "$REPLICA\_USER" -c \ "SELECT pid, application\_name, client\_addr, state, sync\_state, sent\_lsn, write\_lsn, flush\_lsn, replay\_lsn FROM pg\_stat\_replication pg\_stat\_replication;"

# Connect to the DR server and check replication status

echo "Checking the replication on DR..."

psql -h "$NEW\_PRIMARY\_HOST" -p "$PORT" -U "$REPLICA\_USER" -c \ "SELECT pid,status,receive\_start\_lsn,written\_lsn,last\_msg\_send\_time,last\_msg\_receipt\_time,latest\_end\_lsn,latest\_end\_time FROM pg\_stat\_wal\_receiver;"

**4. findprimary.sh**

**Purpose:** This script identifies the current primary site in a PostgreSQL replication setup.

**Usage:**

bash

Copy code

./findprimary.sh

**Description:**

* Queries the replication status to determine which site is currently acting as the primary.
* If not in recovery state creates standby. Signal file.

**Dependencies:**

* The script should be run with appropriate permissions.
* Environmental variables to be updated as per setup

**Example:**

bash

Copy code

# Make sure the script is executable

chmod +x findprimary.sh

# Run the script

./findprimary.sh

**Script Content:**

# Configuration variables

PGDATA="/postgres/data/" # Path to PostgreSQL data directory

PGUSER="postgres" # PostgreSQL superuser

PGCTL="/usr/pgsql-16/bin/pg\_ctl" # Path to pg\_ctl command

POSTGREBIN="/usr/pgsql-16/bin/"

POSTGREDATAPATH="/postgres/data/"

PORT="6444"

DBUSERNAME="postgres"

DBNAME="postgres"

echo "Verify the current active server ip"

RECOVERY\_STATUS=$(${POSTGREBIN}/psql -p ${PORT} -d ${DBNAME} -U ${DBUSERNAME} -w -c "select pg\_is\_in\_recovery();" -q)

value=$(echo $RECOVERY\_STATUS | cut -d " " -f3);

if [ $value == "f" ]

then

echo "Server is in Started State with master mode cannot able execute Standby.signal"

else

touch ${POSTGREDATAPATH}/standby.signal

fi

cat ${POSTGREDATAPATH}/postgresql.auto.conf | grep "^primary\_conninfo" | tr -d "'" | cut -d"=" -f3- | cut -d" " -f4