primary PostgreSQL stopped cleanly

Knowing whether your primary PostgreSQL stopped cleanly before a DR promotion is the key to avoiding timeline mismatches.
Let's go step-by-step — I'll show you how to verify clean shutdown, which logs and files to check, and best commands to use during DR drills.

🧠 1️⃣ Understand what a clean shutdown really means

- When PostgreSQL stops cleanly, it does 3 important things before exiting:

1. Writes all dirty buffers to disk (checkpoints everything)
2. Flushes all WAL records to disk
3. Updates pg_control with a "shutdown checkpoint" record

→ This tells PostgreSQL: "You can start normally next time, no crash recovery needed."

- If any of these 3 don't happen (for example, process killed, power loss, or forced stop), PostgreSQL will mark the last state as "in production", not "shut down".
- Then on next start, it runs crash recovery → leading to timeline mismatch risk if you promoted another node meanwhile.

🧠 2️⃣ How to Check if the Primary Shut Down Cleanly

✅(a) Use pg_controldata

- Run this command after stopping the primary but before promoting the DR node:
-

```
pg_controldata /var/lib/pgsql/15/data | egrep "Database cluster state|Latest checkpoint"
```

Example outputs:

Output	Meaning
Database cluster state: shut down	✅Clean shutdown (safe to promote DR)
Database cluster state: in production	❌PostgreSQL still running
Database cluster state: shut down in recovery	✅Clean standby shutdown
Database cluster state: in crash recovery	❌Previous stop was not clean (danger of mismatch)

☑(b) Check the PostgreSQL logs

- Look at /var/log/postgresql/postgresql-15-main.log (path may differ).
- You should see:

```
LOG:  received smart shutdown request
LOG:  checkpoint starting: shutdown immediate
LOG:  checkpoint complete: wrote 105 buffersLOG:  database system is shut down
```

- If you don't see "**database system is shut down**", then it wasn't cleanly stopped.
- If it ends abruptly like:

```
LOG:  received immediate shutdown request
```

- or no final "shut down" message, it means PostgreSQL was **terminated before writing the checkpoint** → not clean.

☑(c) Ensure the process fully stopped

- Run before promoting DR:

```
ps -ef | grep postgres
```

- Make sure no PostgreSQL backend processes are running.
- If processes still exist (like postmaster or checkpointer), the shutdown hasn't finished yet. Wait 2–3 seconds and re-check.

☑(d) Check service status

- If you're using systemd:

```
systemctl status postgresql-15
```

- You should see:

```
Active: inactive (dead)

If it says "stopping" or "failed," don't promote DR yet.
```

🔹 3️⃣ How to Stop PostgreSQL *Cleanly*

🔹 Recommended shutdown method for a primary during DR drill:

```
pg_ctl stop -D /var/lib/pgsql/15/data -m fast -w
```

Explanation:

-m fast → rolls back active transactions and checkpoints immediately

-w → waits until PostgreSQL confirms it's fully stopped

Avoid -m immediate unless absolutely needed (it skips checkpoint)

Or via systemd:

```
systemctl stop postgresql-15sleep 5
pg_controldata /var/lib/pgsql/15/data | grep "Database cluster state"
```

Wait until it reports shut down.

🔹 4️⃣ Optional: Verify Before Promotion (Best Practice)

- Make a quick safety check in your DR drill script:

```
state=$(pg_controldata /var/lib/pgsql/15/data | grep "Database cluster state" | awk
-F': ' '{print $2}')

if [[ "$state" == "shut down" ]]; then
  echo "✅Primary stopped cleanly. Safe to promote DR."
else
  echo "🔹 Primary not cleanly shut down. Do NOT promote yet!"
  exit 1
fi
```

- This one-liner check can **save you from timeline mismatch** every time.

5 Summary Table

Step	Check	Command	Clean?
1	Cluster state	pg_controldata /var/lib/pgsql/15/data	shut down ✓
2	Log message	Check last few lines in PostgreSQL log	"database system is shut down" ✓
3	Process	`ps -ef | grep postgres`	
4	Systemd	systemctl status postgresql	Inactive (dead) ✓

If any of these fail → **don't promote yet**.

6 Why This Prevents Timeline Mismatch

- If PostgreSQL stops cleanly:

1. The last WAL position is completely flushed
2. The replica already received all WAL up to that LSN
3. When you promote the DR, both share the same WAL end-point
4. So the DR's new timeline starts exactly where the old primary ended → no mismatch.

- If not clean:

1. Primary might have WAL not yet sent/applied on replica
2. DR promotion starts from older LSN → WAL history diverges → Timeline mismatch.

✓In short:

- Always ensure pg_controldata shows "Database cluster state: shut down" before promoting DR.
- That's your guarantee of a clean stop and no timeline ID mismatch.