# Upgrading PostgreSQL major version using `pg_upgrade`

Upgrading PostgreSQL from version 14 to 15 can be done using `pg_upgrade`, which is a utility that allows for fast upgrades by reusing the existing data files.

Here's a brief overview of the **` — link` and ` — clone`** options and how to use them:

**`pg_upgrade` Overview**

`pg_upgrade` works by leveraging the existing data files in your old PostgreSQL cluster to speed up the upgrade process. Instead of copying data, it can either link to the existing files or clone them.

**Why upgrade is required?**

- Security fixes
- Bugfixes
- Performance improvements
- New features

**Types of Upgrade:**
- Minor upgrade: 13.1 -> 13.5
- Major upgrade: 13.1 -> 15.2

**pg_upgrade workflow:**

1. install new major binaries
2. initdb — initialize the new cluster
3. shut down the old cluster
4. run pg_upgrade
5. start the new cluster

**Before upgrade:**

- Read release notes: incompatibilities must be addressed before pg_upgrade.
- Try pg_upgrade — check: if there are any problems reported — fix them.
- Make a backup database using barman/pgbackrest/pg_dump/pg_dumpall and also restored test.
- Check extentions whether it is compatible.

| Option | Description |
|---|---|
| --old-datadir=DIR | Path to the **old PostgreSQL data directory** (e.g., /var/lib/pgsql/13/data). |
| --new-datadir=DIR | Path to the **new PostgreSQL data directory** (e.g., /var/lib/pgsql/15/data). |
| --old-bindir=DIR | Path to the **old PostgreSQL binaries** (e.g., /usr/pgsql-13/bin). |
| --new-bindir=DIR | Path to the **new PostgreSQL binaries** (e.g., /usr/pgsql-15/bin). |
| --check | Run a **dry-run check** to see if upgrade is possible. No actual changes are made. |
| --link | Use **hard links** instead of copying files, which makes the upgrade faster and saves disk space. **Risk**: corrupts old cluster if new one is modified. |
| --clone | Like --link, but uses copy_file_range() syscall or reflinks (safer on supported systems). |
| --jobs=N | Use **parallel jobs** to speed up upgrade. Good for large databases (e.g., --jobs=4). |
| --verbose | Outputs detailed logs for debugging or audit purposes. |
| --username=NAME | Database superuser to use for connecting (default is postgres). |
| --socketdir=DIR | Location of the Unix socket file (if not the default). |
| --old-options='OPTIONS' | Additional options passed to the **old cluster's postgres process**. |
| --new-options='OPTIONS' | Additional options passed to the **new cluster's postgres process**. |
| --retain | Retain temporary files (e.g., pg_upgrade_dump.*) generated during upgrade. Useful for debugging. |
| --debug | Enable debugging mode — forces pg_upgrade to pause at key steps so you can manually inspect. |
| --disable-link | Explicitly disables hard linking even if the system supports it. |
| --no-sync | Skips syncing data to disk after upgrade. **Faster** but **risky** if system crashes. |
| --check | Repeated here to emphasize: it's the **safe pre-upgrade validation** step. |
| | |

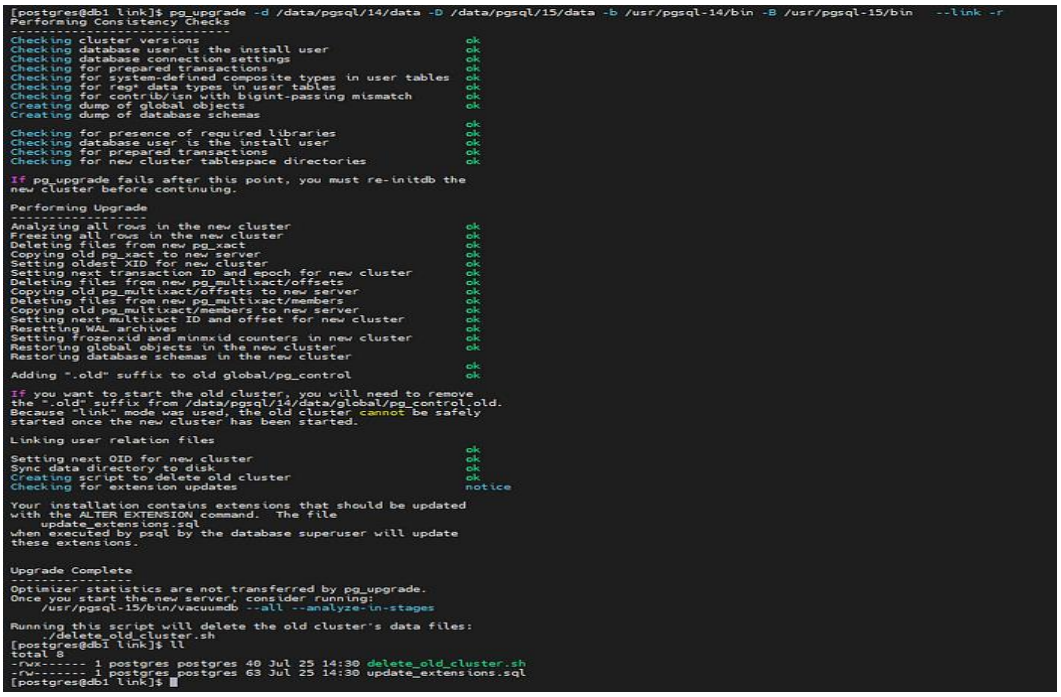
| Feature/Aspect | --copy (Default) | --link | --clone |
|---|---|---|---|
| **How it works** | Copies data files | Creates hard links to data files | Uses copy-on-write (CoW) or reflinks |
| **Speed** | ✗Slow | ✓Fast | ✓Fast |
| **Disk usage** | ✗High (requires double storage) | ✓Low (no extra space used) | ✓Low (depends on filesystem) |
| **Old cluster safety** | ✓Safe (files not touched) | ✗Unsafe (modifying new affects old) | ✓Safe (files logically separate) |
| **Filesystem requirements** | ✗None | ✗None | ✓Requires CoW/reflink support (e.g., Btrfs, XFS with reflinks) |
| **Risk** | ✓Very safe | ⚠ Risky if new cluster modified | ✓Safer than --link |
| **Use case** | Production environments | DR/testing where speed is key | Hybrid: Fast and safe on modern filesystems |
| **Command line flag** | *(default if no flag)* | --link | --clone |

| Environment | Recommended Option |
|---|---|
| **Production** | --copy (default) |
| **Testing/DR** | --link |
| **Modern Linux with Btrfs/XFS** | --clone |

| Feature | --copy (default) | --link | --clone |
|---|---|---|---|
| **Speed** | ✗Slow | ✓Fast | ✓Fast |
| **Disk Usage** | ✗High | ✓Low | ✓Low (if FS supports) |
| **Old Cluster Safety** | ✓Safe | ✗Unsafe | ✓Safe |
| **Filesystem Requirement** | ✗None | ✗None | ✓Special (e.g., Btrfs, XFS with reflink) |
| **Best For** | Production | Testing/DR | Modern systems with COW support |

**--clone only works on filesystems that support copy-on-write or efficient cloning.**

## Using `pg_upgrade --link`



**What it does:**

- **- With the ` — link` option**, `pg_upgrade` creates hard links between the old and new data directories. This means that both the old and new clusters share the same data files, avoiding the need to duplicate data on disk.

**Advantages:**

- - Speed: Linking is faster because it avoids copying data files
- .
  - Disk Space: It conserves disk space since the data files are not duplicated.

**Disadvantages:**

- - Limited Use Case: This option is only available if the data directories are on the same filesystem. If the new cluster is on a different filesystem, this option cannot be used.
- - Risk: If the new cluster or the old cluster's data files are corrupted or deleted, it could affect both clusters due to the shared data.

- --link tells pg_upgrade not to copy the data files, but instead to create hard links from the old cluster's data directory to the new cluster's data directory.

- A hard link means both directories point to the same physical file on disk. There's no duplicate — just two directory entries for the same underlying data.

- If you delete the old cluster's data directory after upgrading with --link, the files themselves will be deleted from disk (because the last hard link will be removed).

- That will cause corruption or complete data loss in the new cluster, because the files it depends on will be gone.

**How to use it:**

1. Stop the PostgreSQL server for version 14.

2. Initialize the new PostgreSQL 15 cluster.

3. Run `pg_upgrade` with the ` — link` option also with version 15, it means pg_upgrade verson should be higher:

mkdir -p /data/pgsql/15/data # Create a data directory for postgresql 15

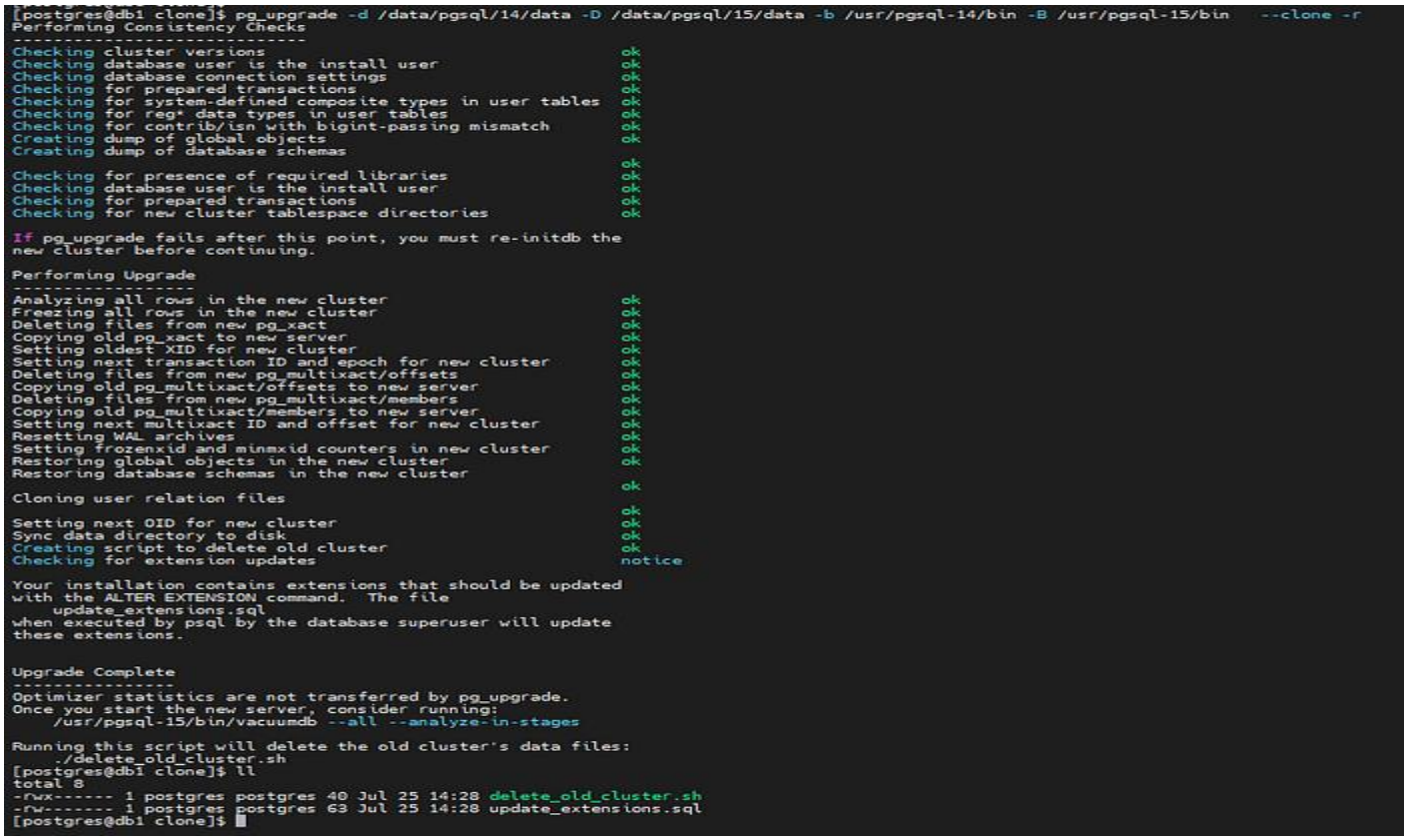
sudo /usr/pgsql-15/bin/postgresql-15-setup initdb # Should have a initialized data files in

/data/pgsql/15/data, and this path should be configured in service file

pg_upgrade --link -r -d /data/pgsql/14/data -D /data/pgsql/15/data -b /usr/pgsql-14/bin -B /usr/pgsql-15/bin --check # Used for dry run

pg_upgrade --link -r -d /data/pgsql/14/data -D /data/pgsql/15/data -b /usr/pgsql-14/bin -B /usr/pgsql-15/bin # Used for final execution

**Using `pg_upgrade — clone`**




**What it does:**

- With the ` — clone` option, `pg_upgrade` creates a copy of the old data directory for the new cluster. This means the new cluster will have its own separate set of data files.

**Advantages:**

- Independence: The new cluster's data files are independent of the old cluster's files, which reduces the risk of data corruption affecting both clusters.

- Filesystem Flexibility: You can use this option even if the old and new clusters are on different filesystems.

**Disadvantages:**
- Speed: Cloning takes longer than linking because it involves copying data files.

- Disk Space: It requires more disk space as data files are duplicated.

**How to use it:**

1. Stop the PostgreSQL server for version 14.
2. Initialize the new PostgreSQL 15 cluster.
3. Run `pg_upgrade` with the ` — clone` option also with version 15, it means pg_upgrade verson should be higher:

mkdir -p /data/pgsql/15/data  # Create a data directory for postgresql 15

sudo  /usr/pgsql-15/bin/postgresql-15-setup initdb  # Should have a initialized data files in

/data/pgsql/15/data, and this path should be configured in service file

pg_upgrade --clone -r -d /data/pgsql/14/data -D /data/pgsql/15/data -b /usr/pgsql-14/bin -B /usr/pgsql-15/bin --check  # Used for dry run

pg_upgrade --clone -r -d /data/pgsql/14/data -D /data/pgsql/15/data -b /usr/pgsql-14/bin -B /usr/pgsql-15/bin  # Used for final execution

**Summary:**

Use ` — link` if you want a faster upgrade and are okay with the old and new clusters sharing the same filesystem.

- Use ` — clone` if you need the new cluster to have its own set of data files or if the old and new data directories are on different filesystems.

**Note:**

Always make sure to perform a full backup of your data before starting the upgrade process to protect against any unexpected issues.

2. For dry run pelase try with — check option before execuition above command.

| Option | Description |
|---|---|
| --old-datadir=DIR | Path to the **old PostgreSQL data directory** (e.g., /var/lib/pgsql/13/data). |
| --new-datadir=DIR | Path to the **new PostgreSQL data directory** (e.g., /var/lib/pgsql/15/data). |
| --old-bindir=DIR | Path to the **old PostgreSQL binaries** (e.g., /usr/pgsql-13/bin). |
| --new-bindir=DIR | Path to the **new PostgreSQL binaries** (e.g., /usr/pgsql-15/bin). |
| --check | Run a **dry-run check** to see if upgrade is possible. No actual changes are made. |
| --link | Use **hard links** instead of copying files, which makes the upgrade faster and saves disk space. **Risk**: corrupts old cluster if new one is modified. |
| --clone | Like --link, but uses copy_file_range() syscall or reflinks (safer on supported systems). |
| --jobs=N | Use **parallel jobs** to speed up upgrade. Good for large databases (e.g., --jobs=4). |
| --verbose | Outputs detailed logs for debugging or audit purposes. |
| --username=NAME | Database superuser to use for connecting (default is postgres). |
| --socketdir=DIR | Location of the Unix socket file (if not the default). |
| --old-options='OPTIONS' | Additional options passed to the **old cluster's postgres process**. |
| --new-options='OPTIONS' | Additional options passed to the **new cluster's postgres process**. |
| --retain | Retain temporary files (e.g., pg_upgrade_dump.*) generated during upgrade. Useful for debugging. |
| --debug | Enable debugging mode — forces pg_upgrade to pause at key steps so you can manually inspect. |
| --disable-link | Explicitly disables hard linking even if the system supports it. |
| --no-sync | Skips syncing data to disk after upgrade. **Faster** but **risky** if system crashes. |
| --check | Repeated here to emphasize: it's the **safe pre-upgrade validation** step. |
| | |