**Purpose**

There are several methods to update a PostgreSQL database cluster. A table of advantages and disadvantages of each are (from *https://severalnines.com/blog/upgrading-your-database-to-postgresql-version-10* ):

| **Method** | **Advantages** | **Disadvantages** | **Comment** |
| --- | --- | --- | --- |
| Traditional dump/restore | * Easy, almost foolproof * If the restoration process finishes without error, you are pretty much there. If not, you have to modify the existing old cluster to eliminate any errors and start the process over again. | * Requires downtime * longest time to complete | * Takes a logical backup and restoring it to a new, higher version of the database * popular choice for those who can afford the downtime * Using this method is the oldest way to upgrade and has been verified by MANY people |
| Traditional pg\_upgrade | * The same try-and-fix approach as the previous method. You save the actions taken against the test database in a script, until you successfully pg\_upgrade * Shorter downtime than logical backup/restore * neater process - pg\_upgrade upgrades the original database with existing data and structure | * Requires downtime | * created to shorten the time it takes to upgrade to a major version, possibly to seconds or minutes * Has been used a lot in past and still would be the preference for the most DBAs running version below 9.4 (which allows using pglogical) |
| Trigger Based Replication | * no backups need to be taken and no downtime required | * High Complexity of setup * Lack of structured documentation | * Most exotic * Not very popular - less user cases to study (and share) |
| Logical Replication With pglogical | * Basically no downtime * much less effort than trigger based replication | * highly complex to setup (especially for older versions) * Lack of structured documentation | * Not very popular - less user cases to study (and share) |

This process documents the traditional dump and restore approach using Dnf. The process is adapted from those outlined by:

* Leo Hsu and Regina Obe (2014; *An almost idiot's guide to install* *PostgreSQL 9.3, PostGIS 2.1 and pgRouting with Dnf*, followed closely in process 315-10), and by
* Vladimir Svedov (2018; *Upgrading your database to PostgreSQL version 10 – what you should know*).
* Paul Ramsay (2019; *Upgrading PostGIS on CentOS 7*)

The following process references Postgresql-9.3 and PostGIS-2.1, but the dump/restore process applies to all recent versions.

| **Step** | **Major Activity** | **References, Forms and Details** |
| --- | --- | --- |
| **1** | Determine your version of Linux   * CentOS/Red Hat:   uname -a   * Others:   lsb release -a | * Use the relevant commands for the appropriate Linux OS in following steps * As one possible response:   CentOS Linux release 7.6.1810 (Core) |
| **2** | Ensure you have the epel repo   * sudo dnf list | grep epel   If not, then install:  sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm | * the *Enterprise Linux Extra Packages* (epel) package is required for PostGIS |
| **3** | Install helper applications  sudo subscription-manager repos --enable codeready-builder-for-rhel-8-x86\_64-rpms | * Also required for PostGIS * note that the RHEL repository is different than CentOS, so “Power Tools” is only CentOS |
| **4** | Perform a database cluster dump   * pg\_dumpall –f <location>/<file>.sql   Or a dump of each database   * pg\_dump –f <location>/<file>.sql <database> | * pg\_dumpall dumps the database cluster, including roles, schemas and users. This is often the best approach. * for pg\_dump, need to dump each database. Note that the dumps should be checked for errors by test-loading them elsewhere * Plain text sql commands result. It is possible to dump compressed files, but then need to use pg\_restore. |
| **5** | Attempt to restore the dump file on a test database cluster   * If there are errors, fix the errors until a clean copy is obtained | * This can take significant time and repetitions, but it is essential to create a clean copy |
| **6** | Stop your existing PostgreSQL service   * RHEL/CentOS-6:   service postgresql-9.3 stop   * Other systems using systemctl:   systemctl stop postgresql-9.3 | * Should also be able to use pg\_ctl stop command |
| **7** | Switch to user **root**  sudo -i | * This places you in the required root shell for the following installation steps |
| **8** | Update the Dnf repositiory:   * Go to   [*https://dnf.postgresql.org/repopackages.php*](https://yum.postgresql.org/repopackages.php)   * Copy the link corresponding to your OS version and PostgreSQL version. For example,   *rpm -ivh https://download.postgresql.org/pub/repos/dnf/11/redhat/rhel-7-x86\_64/pgdg-centos11-11-2.noarch.rpm* | * The chosen postgresql version must be compatible with the operating system and with the postgis spatial extension package |
| **9** | Confirm there is a compatible and desired postgis version for the desired postgresql version and operating system:  dnf list | grep postgis | Typical output   * *postgis2\_93.x86\_64 2.1.8-1.rhel7 @pgdg93* * *postgis.x86\_64 2.0.7-1.el7 epel* * *postgis-docs.x86\_64 2.0.7-1.el7 epel*   etcetera |
| **10** | Install the new version of PostgreSQL  *e.g. dnf install –y postgresql11-server* | * this could be any PostgreSQL version that is compatible with the operating system * there will be verbose output hopefully confirming completion and successful installation |
| **11** | Install PostGIS 25 for PostgreSQL 11  *e.g. dnf install –y postgis25\_11 postgis25\_11-client* | * this could be any PostGIS version that is compatible with the PostgreSQL version and the OS, with similar completion messages |
| **12** | Switch to user postgres  su postgres | * Perform following steps as user postgres so data will be accessible |
| **13** | (Optional) Create a location for the new database  mkdir –p {path/data} | * if not implemented, data will be installed in default location, which is often undesired. * location needs to be owned by user postgres * as an example:   mkdir –p /data/postgresql/11/data |
| **14** | Initialize your PostgreSQL 11 data cluster  *e.g. /usr/pgsql-11/bin/initdb –D {path//data/}* | * The “-D” overrides any other data directory command and location. It is required if the data directory is not at the default location (generally preferred). * There should be feedback indicating successful installation, and guidance to start the cluster. Don’t plan to use the suggested –l option (may cause an error). |
| **15** | Type  exit | * exit from user ***postgres*** to user ***root*** |
| **16** | Start the server   * /usr/pgsql-11/bin/pg\_ctl start –D  *{path//data/}* | * There should be feedback describing listening IP address and port, and a “server started” comment |
| **17** | Enable server to start on reboot   * chkconfig postgresql-11 on   Or, newer versions using system V   * systemctl enable postgresql-11 | * “chkconfig on” so database cluster will auto-restart on power-on |
| **18** | Revert to user postgres   * su postgres | * was user **root** |
| **19** | Load database cluster.   * If used pg\_dumpall,   *psql –f {path-to-file}/{file}.sql*   * Confirm postgres and postgis are working:   *sudo -u postgres psql*  *SELECT \* from postgis\_full\_version();* | Should see something like:   * POSTGIS="2.5.1 r17027" [EXTENSION] PGSQL="110" * GEOS="3.7.0-CAPI-1.11.0 673b9939" PROJ="Rel. 4.9.3, 15 August 2016" * GDAL="GDAL 1.11.4, released 2016/01/25" LIBXML="2.9.1" LIBJSON="0.11" RASTER * (1 row) |
| **20** | Edit .bashrc and save:  Insert:  export PGHOST=localhost  export PGUSER=postgres  export PGPORT={port}  export PGDATA={path-to-data/data}  Modify PATH to include:  /bin:$JAVA\_HOME/bin:/usr/pgsql-11/bin:$PATH | * Normally, PGPORT is 5432 * In path statement, change any reference to former pgsql-x and point to new database applications * Upgrade is normally complete after this step |
| **21** | Only if needed – see note. For each database saved using pg\_dump   * psql –f <location/file> | * These and next steps are only required if you created a dump for each database, or a schema-only dump. You may need to create users (roles) and schemas. * load each database |
| **22** | Only if needed – see note. Create production databases and users. For example   * sudo -u postgres psql * CREATE USER geonetwork   WITH PASSWORD ‘secret’ LOGIN;   * \du (to view and confirm owner) * CREATE DATABASE gndb4 OWNER geonetwork; | * **Sudo -u postgres psql** is the standard logon * superuser **postgres** owns the functions * owner geonetwork will own the database. It is best to separate the two users for security reasons |
|  | Only if needed – see note. Connect to database and create extensions. For example   * \c gndb4 <superuser> * (enter password) * CREATE EXTENSION postgis; * CREATE EXTENSION postgis\_topology; * \dx (to view and confirm extensions) | * Connect to gndb4, or metadata database name of your choosing AS superuser * Create two needed extensions * \dx should display three extensions |

**RESULT OF initdb**

# sudo -u postgres ./initdb -D /data/postgresql/13/data

The files belonging to this database system will be owned by user "postgres".

This user must also own the server process.

The database cluster will be initialized with locale "en\_CA.UTF-8".

The default database encoding has accordingly been set to "UTF8".

The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /data/postgresql/13/data ... ok

creating subdirectories ... ok

selecting dynamic shared memory implementation ... posix

selecting default max\_connections ... 100

selecting default shared\_buffers ... 128MB

selecting default time zone ... America/Vancouver

creating configuration files ... ok

running bootstrap script ... ok

performing post-bootstrap initialization ... ok

syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections

You can change this by editing pg\_hba.conf or using the option -A, or

--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

./pg\_ctl -D /data/postgresql/13/data -l logfile start

***# Optionally initialize the database and enable automatic start:***

***sudo /usr/pgsql-13/bin/postgresql-13-setup initdb***

***sudo systemctl enable postgresql-13***

***sudo systemctl start postgresql-13***

[root@soggy2 bin] 19:39:45

# sudo -u postgres psql

psql: error: could not connect to server: No such file or directory

Is the server running locally and accepting

connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"?

[root@soggy2 bin] 19:40:48