|  |
| --- |
| Document Privacy Level : Low |
| PostgreSQL 11 Backup and Restore Plan – CentOS 7 |
| PG Backup and Restore Plan Version PGBKREV1.0 |

|  |
| --- |
| Torrance TEAM  6/19/2020 |

Contents

[Overview 2](#_Toc43943424)

[Pre-Requirements 2](#_Toc43943425)

[Logical Backup 2](#_Toc43943426)

[Dump Method: 3](#_Toc43943427)

[Please follow the Methods to backup your Database 5](#_Toc43943428)

[Four backup formats: 6](#_Toc43943429)

[How to backup only one table? 6](#_Toc43943430)

[How to back up in insert file format? 6](#_Toc43943431)

[How to take back up of Entire Database? 7](#_Toc43943432)

[How to backup database in custom file format? 7](#_Toc43943433)

[How to take backup database in directory format? 7](#_Toc43943434)

[Restore Database 7](#_Toc43943435)

[Essential Step before Restore 7](#_Toc43943436)

[How to restore the database from plan text file format? 8](#_Toc43943437)

[How to create new database and restore existing database in custom format? 8](#_Toc43943438)

[How to restore database from directory format? 9](#_Toc43943439)

[Objects Definitions Backup 10](#_Toc43943440)

[Physical Backup 10](#_Toc43943441)

[File System Level Backup 10](#_Toc43943442)

[Point in time recovery PITR 13](#_Toc43943443)

[Lab Exercises 14](#_Toc43943444)

[Lab Exercise -1 14](#_Toc43943445)

[Lab Exercise – 2 14](#_Toc43943446)

[Lab Exercise – 3 14](#_Toc43943447)

[Lab Exercise – 4 15](#_Toc43943448)

[Lab Exercise -5 15](#_Toc43943449)

[Lab Exercise – 6 15](#_Toc43943450)

[Lab Exercise – 7 15](#_Toc43943451)

[Lab Exercise – 8 15](#_Toc43943452)

# Overview

This tutorial will help you to take logical and physical backups efficiently. In addition, it is providing live exercises for hand on experience for successful backups. This tutorial covers following topics:

1) PostgreSQL Logical backup

2) PostgreSQL Logical restores

3) Setting up CRON jobs with PostgreSQL Logical Backups

4) PostgreSQL Physical Backups

5) Setting up CRON jobs with PostgreSQL Physical Backups

6) PostgreSQL Physical Restores

7) Lab exercises covering Logical Backup & Restores

8) Lab exercise covering Physical Backup & Restores

This document is straight hit on goal of logical, physical backups and scheduling of jobs.

## Pre-Requirements

It is being considered below mention technologies install before implementation

1. CentOS 7
2. Postgresql 11
3. First Restore the provided database.

## Logical Backup

In Postgres, we have three fundamental approaches to make backup of Data and Database objects

1. Logical Backup (SQL Dump)
2. Physical Backup (File System level Backup)
3. Point in time Recovery (Continues Archiving)

In Postgres SQL Dump back is considered as logical back and it uses **pg\_dump** utility for this purpose. The advantage of that backup is very life easing because it could be compatible with upgrading versions of Postgres or on machine with different architecture. On the other hand, file system level backup and Point recovery are bound to Postgres Server Version and System architecture.

## Dump Method:

This method is to generate a file with the help of SQL commands and these files can easily feedback to the server. It will create the database in same state as it was at that time of dump creation. Pg\_dump or pg\_dumpall are the utilities provided by PostgreSQL fir this purpose.

Syntax:

Pg\_dump Database\_name > “Path of Foler”\Name of file.sql

g\_dump dumps a database as a text file or to other formats.

Usage:

pg\_dump [OPTION]... [DBNAME]

General options:

-f, --file=FILENAME output file or directory name

-F, --format=c|d|t|p output file format (custom, directory, tar,

plain text (default))

-j, --jobs=NUM use this many parallel jobs to dump

-v, --verbose verbose mode

-V, --version output version information, then exit

-Z, --compress=0-9 compression level for compressed formats

--lock-wait-timeout=TIMEOUT fail after waiting TIMEOUT for a table lock

--no-sync do not wait for changes to be written safely to disk

-?, --help show this help, then exit

Options controlling the output content:

-a, --data-only dump only the data, not the schema

-b, --blobs include large objects in dump

-B, --no-blobs exclude large objects in dump

-c, --clean clean (drop) database objects before recreating

-C, --create include commands to create database in dump

-E, --encoding=ENCODING dump the data in encoding ENCODING

-n, --schema=SCHEMA dump the named schema(s) only

-N, --exclude-schema=SCHEMA do NOT dump the named schema(s)

-o, --oids include OIDs in dump

-O, --no-owner skip restoration of object ownership in

plain-text format

-s, --schema-only dump only the schema, no data

-S, --superuser=NAME superuser user name to use in plain-text format

-t, --table=TABLE dump the named table(s) only

-T, --exclude-table=TABLE do NOT dump the named table(s)

-x, --no-privileges do not dump privileges (grant/revoke)

--binary-upgrade for use by upgrade utilities only

--column-inserts dump data as INSERT commands with column names

--disable-dollar-quoting disable dollar quoting, use SQL standard quoting

--disable-triggers disable triggers during data-only restore

--enable-row-security enable row security (dump only content user has

access to)

--exclude-table-data=TABLE do NOT dump data for the named table(s)

--if-exists use IF EXISTS when dropping objects

--inserts dump data as INSERT commands, rather than COPY

--load-via-partition-root load partitions via the root table

--no-comments do not dump comments

--no-publications do not dump publications

--no-security-labels do not dump security label assignments

--no-subscriptions do not dump subscriptions

--no-synchronized-snapshots do not use synchronized snapshots in parallel jobs

--no-tablespaces do not dump tablespace assignments

--no-unlogged-table-data do not dump unlogged table data

--quote-all-identifiers quote all identifiers, even if not key words

--section=SECTION dump named section (pre-data, data, or post-data)

--serializable-deferrable wait until the dump can run without anomalies

--snapshot=SNAPSHOT use given snapshot for the dump

--strict-names require table and/or schema include patterns to

match at least one entity each

--use-set-session-authorization

use SET SESSION AUTHORIZATION commands instead of

ALTER OWNER commands to set ownership

Connection options:

-d, --dbname=DBNAME database to dump

-h, --host=HOSTNAME database server host or socket directory

-p, --port=PORT database server port number

-U, --username=NAME connect as specified database user

-w, --no-password never prompt for password

-W, --password force password prompt (should happen automatically)

--role=ROLENAME do SET ROLE before dump

If no database name is supplied, then the PGDATABASE environment

variable value is used.

Report bugs to <pgsql-bugs@postgresql.org>.

# Please follow the Methods to backup your Database

Open the terminal and follow these commands.

[umair@localhost ~]$ ## connect to postgres super user

[umair@localhost ~]$ ##################################

[umair@localhost ~]$ sudo su - postgres

[sudo] password for umair:

Last login: Tue Jun 23 01:16:02 PKT 2020 on pts/0

-bash-4.2$ psql

psql (11.8)

Type "help" for help.

Here we will check the list of all databases already exist in server.

postgres=# ## check how many databases already exist.

postgres-# \l

Lists of database will look like this

List of databases

Name | Owner | Encoding | Collate | Ctype | Acces

s privileges

-----------+----------+----------+-------------+-------------+--------

---------------

dirdb | postgres | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

mydb2 | myuser2 | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

newdb | testuser | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

postgres | postgres | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

template0 | postgres | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 | =c/post

gres +

| | | | | postgre

s=CTc/postgres

template1 | postgres | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 | =c/post

gres +

| | | | | postgre

s=CTc/postgres

tutdb | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 |

(7 rows)

## Four backup formats:

* \*.bak: compressed binary format
* \*.sql: plaintext dump
* \*.tar: tarball
* Directory Format

# How to backup only one table?

-bash-4.2$ ### we will backup in plan text format.

-bash-4.2$ pg\_dump -d tutdb -p 5432 -U postgres -t customer -f /tmp/tempfile/custtab\_20200623.txt

-bash-4.2$ ### Its created

# How to back up in insert file format?

-bash-4.2$ #### if you want to restore the plan text for format backup use psql utility

-bash-4.2$ ### now we take backup in inserts format.

-bash-4.2$ pg\_dump -d tutdb -p 5432 -U postgres -t customer --inserts -f /tmp/tempfile/custtab\_20200623.txt

-bash-4.2$ ## its created please check the directory path.

How to take back up of Entire Database?

-bash-4.2$ ###Now we will take the backup of entire database.

-bash-4.2$ pg\_dump -d tutdb -p 5432 -U postgres --inserts -v -f /tmp/tempfile/entiretutdb\_inserts\_20200623.txt

-bash-4.2$ ## you see here we add -v in pervious command. it verbose which shows about the all transection details and show you on screen

# How to backup database in custom file format?

-bash-4.2$ ###Lets take the backup in custom format. Here we will add -F c and this format is not human readable. This is the only format use for pg\_restore for restore of database.

-bash-4.2$ pg\_dump -d tutdb -p 5432 -U postgres -F c -v -f /tmp/tempfile/entiretutdb\_custom\_20200623.txt

# How to take backup database in directory format?

Before entering this command, please make sure particular directory has been created with full permission for user.

-bash-4.2$ ## lets take the backup in diecctory format

-bash-4.2$ ## first create the directory and give permissions for postgres or group users.

-bash-4.2$ ## i have created the directory /tmp/tempfile/bkdir

-bash-4.2$ ##here we will use directory name instead of file name

-bash-4.2$ pg\_dump -d tutdb -p 5432 -U postgres -F d -v -f /tmp/tempfile/bkdir

You can use –j at the end of above command to make threads of backup process and catalyze the process. Here I am creating three threads.

-bash-4.2$ pg\_dump -d tutdb -p 5432 -U postgres -F d -v -f /tmp/tempfile/bkdir -j 3

# Restore Database

# Essential Step before Restore

Since the text files generated by pg\_dump contain a set of SQL commands, they can be fed to the psql utility. The database itself will not be created by psql, so you must create it yourself from template0 first. So, the general command form to restore a dump is:

createdb -T template0 **database\_name** psql **database\_name** < **database.sql**

Before starting to restore an SQL dump and the recreation of the objects with the original ownership and/or permissions, it is crucial to make sure that all users who have been granted permissions on objects, or who own objects in the uploaded database, already exist. Otherwise, the restoration process will fail.

# How to restore the database from plan text file format?

Here we will restore only one table. Please use psql utility for restore of plan text file format backup.

-bash-4.2$ ## first we will restore the plan text format. As we discussed we will use psql.

-bash-4.2$ psql -d tutdb -p 5432 -U postgres -f /tmp/tempfile/custtab\_20200623.txt

# How to create new database and restore existing database in custom format?

-bash-4.2$ ### Now we will restore full data . first we drop tutdb and then create again/

-bash-4.2$ psql

psql (11.8)

Type "help" for help.

postgres=# drop database tutdb;

DROP DATABASE

postgres=# create database tutdb;

CREATE DATABASE

postgres=# \l

postgres=# \q

-bash-4.2$ ### now we restore full database.

-bash-4.2$ ## here i will use custom format.

-bash-4.2$ pg\_restore -d tutdb -U postgres -p 5432 /tmp/tempfile/entiretutdb\_custom\_20200623.txt

-bash-4.2$ ## database populated lets see

-bash-4.2$ psql

psql (11.8)

Type "help" for help.

postgres=# \c tutdb

You are now connected to database "tutdb" as user "postgres".

tutdb=#

tutdb=# \dt

tutdb=# select \* from actor;

tutdb=# \q

-bash-4.2$ ####it perfectly done

-bash-4.2$ #### lets create an other user and try to restore database from directory dump formate.

-bash-4.2$ ### lets do it.

# How to restore database from directory format?

postgres=# drop database dirdb;

DROP DATABASE

postgres=# create database dirdb;

CREATE DATABASE

postgres=# \q

-bash-4.2$ pg\_restore -d dirdb -U postgres -p 5432 /tmp/tempfile/bkdir

-bash-4.2$ ### its restores lets see

-bash-4.2$ psql

psql (11.8)

Type "help" for help.

postgres=# \c dirdb

You are now connected to database "dirdb" as user "postgres".

dirdb=# \dt

### its perfectly done.

## Objects Definitions Backup

Use the following command to backup all objects in all databases, including roles, databases, tablespaces, tables, schemas, indexes, functions, triggers, constraints, privileges, views, and ownerships:

pg\_dumpall –U postgres --schema-only > “MyPATH”\definitions.sql

Use the following command to backup the role definition only:

pg\_dumpall –U postgres --roles-only > “MyPATH”\roles.sql

Use the following command to backup the [tablespaces](https://www.postgresql.org/docs/9.1/manage-ag-tablespaces.html) definition:

pg\_dumpall –U postgres --tablespaces-only > “MyPATH”\tablespaces.sql

# Physical Backup

All physical backups are dependent on source and target architecture, version, compiler flags and paths. Source and target environment must be equivalent.

# File System Level Backup

[umair@localhost Documents]$ ### create directory for archive logs

[umair@localhost Documents]$ sudo -H -u postgres mkdir /var/lib/postgresql/pg\_log\_archive

[sudo] [umair@localhost Documents]$ ## enable archive logging

[umair@localhost Documents]$ sudo nano /var/lib/pgsql/11/data/postgresql.conf

[umair@localhost Documents]$ systemctl restart postgresql-11

[umair@localhost Documents]$ ## create a database with insert data

[umair@localhost Documents]$ sudo su - postgres

Last login: Thu Jun 25 00:01:24 PKT 2020 on pts/1

-bash-4.2$ psql -c "create database testdb"

CREATE DATABASE

password for umair:

psql testdb -c "

> create table posts (

> id integer,

> title character varying(100),

> content text,

> published\_at timestamp without time zone,

> type character varying(100)

> );

>

> insert into posts (id, title, content, published\_at, type) values

> (100, 'Intro to SQL', 'Epic SQL Content', '2018-01-01', 'SQL'),

> (101, 'Intro to PostgreSQL', 'PostgreSQL is awesome!', now(), 'PostgreSQL');

> "

INSERT 0 2

-bash-4.2$ ## achive the logs

-bash-4.2$ psql -c "select pg\_switch\_wal();"

pg\_switch\_wal

---------------

0/C01E1E0

(1 row) -bash-4.2$ ##backup database

-bash-4.2$ pg\_basebackup -Ft -D /var/lib/postgresql/db\_file\_backup

-bash-4.2$ ##Stop database

-bash-4.2$ systemctl stop postgresql-11

-bash-4.2$ rm var/lib/pgsql/11/data/\* -r

rm: cannot remove ‘var/lib/pgsql/11/data/\*’: No such file or directory

-bash-4.2$ rm /var/lib/pgsql/11/data/\* -r

-bash-4.2$ ls /var/lib/pgsql/11/data/

-bash-4.2$ ls /var/lib/pgsql/11

backups data initdb.log

-bash-4.2$ ##restore files

-bash-4.2$ tar xvf /var/lib/postgresql/db\_file\_backup/base.tar -C /var/lib/pgsql/11/data/

backup\_label

tablespace\_map

pg\_wal/

./pg\_wal/archive\_status/

global/

global/1262

global/1262\_fsm

global/2964

global/1213

global/1213\_fsm

global/1136

global/1136\_fsm

global/1260

global/1260\_fsm

global/1261

global/1214

global/2396

global/6000

global/3592

global/6100

global/2846

global/2847

global/2966

global/2967

global/4060

…

…

…

bash-4.2$ tar xvf /var/lib/postgresql/db\_file\_backup/pg\_wal.tar -C /var/lib/pgsql/11/data/

00000001000000000000000E

-bash-4.2$ ##add recovery config

-bash-4.2$ nano /var/lib/pgsql/11/data/postgresql.conf

-bash-4.2$

-bash-4.2$

-bash-4.2$

-bash-4.2$ ###restore\_command = 'cp /var/lib/postgresql/pg\_log\_archive/%f %p'

-bash-4.2$ ###restore command added

-bash-4.2$ ##start db

-bash-4.2$ systemctl start postg[umair@localhost Documents]$ sudo su - postgres

Last login: Thu Jun 25 00:01:24 PKT 2020 on pts/1

-bash-4.2$ psql

-bash-4.2$ # verify restore was successful

# Point in time recovery PITR

[umair@localhost Documents]$ # backup database and gzip

[umair@localhost Documents]$ sudo su - postgres

Last login: Thu Jun 25 00:01:24 PKT 2020 on pts/1

-bash-4.2$ pg\_basebackup -Ft -X none -D - | gzip > /var/lib/postgresql/db\_file\_backup.tar.gz

# wait

-bash-4.2$ psql testdb -c "insert into posts (id, title, content, type) values

> (102, 'Intro to SQL Where Clause', 'Easy as pie!', 'SQL'),

> (103, 'Intro to SQL Order Clause', 'What comes first?', 'SQL');"

-bash-4.2$ # archive the logs

-bash-4.2$ psql -c "select pg\_switch\_wal();"

-bash-4.2$ # stop DB and destroy data

-bash-4.2$ systemctl stop postgresql-10

-bash-4.2$ rm /var/lib/pgsql/11/data/\* -r

-bash-4.2$ ls /var/lib/postgresql/10/main/

-bash-4.2$ # restore

-bash-4.2$ tar xvfz /var/lib/postgresql/db\_file\_backup.tar.gz -C /var/lib/pgsql/11/data/

-bash-4.2$ # add recovery.conf

-bash-4.2$ nano /var/lib/postgresql/10/main/recovery.conf

restore\_command = 'cp /var/lib/postgresql/pg\_log\_archive/%f %p'

recovery\_target\_time = '2020-06-25 15:20:00 EST'

-bash-4.2$ # start DB

-bash-4.2$ systemctl start postgresql-11

# verify restore was successful

-bash-4.2$ psql test -c "select \* from posts;

-bash-4.2$ # complete and enable database restore

-bash-4.2$ psql -c "select pg\_wal\_replay\_resume();"

# Lab Exercises

## Lab Exercise -1

The tutdb database is all setup and as a DBA you need to plan a proper backup strategy and implement it.

* As the root user, create a folder /bkupdir and assign ownership to the tutdb user using the chown utility.
* Take a full dataase dump of the tutdb database with the pg\_dump utility. The dump should be in plain text format.
* Nathe dump file as tutdb\_full\_Data.sql and store it in the /bkupdir directory.

## Lab Exercise – 2

1. Take the dump backup of tutdb schema and name the file as as tutdb\_schema.sql
2. Take a data only dump of the tutdb database, disable all triggers for faster restore, use the INSERT command instead of COPY, and name the fie as tutdb\_data.sql
3. Take a full dump of actors table and name the file as tutdb\_actors.sql.

## Lab Exercise – 3

1. Take a full database dump of tutdb in compressed format using the pd\_dump utility, name the file as tutdb\_full.dmp
2. Take a full database cluster dump using pg\_dumpall. Remember pg\_dumpall supports only plain text format; name the file tutdb.sql

## Lab Exercise – 4

In this exercise you will demonstrate your ability to restore a database.

1. Drop database tutdb
2. Create database tutdb with owner superuser
3. Restore the full dump from tutdb\_full.sql and verify all the objects and their ownership.
4. Drop database tutdb.
5. Create database tutdb with superuser owner
6. Restore the full dump from the compressed file tutdb\_full\_fc.dmp and verify all the objects and their ownership.

## Lab Exercise -5

1. Create a directory /opt/arch and give ownership to the superuser.
2. Configure your cluster to run in archive mode and set the archive log location to be /opt/arch.
3. Take a full online base backup of your cluster in the /pgbackup directory using the pg\_basebackup utility.

## Lab Exercise – 6

A database cluster can encounter different types of failures. Recover your database from a variety of simulated failures:

1. Recover from loss of the postgresql.conf file
2. Recover fom loss of an actors table data file.
3. Recover from mistakenly drop table actors.

## Lab Exercise – 7

1. Create new database name demodb and insert few records.
2. Check database archive log is enabling in postgresql.conf file. If you find not enable take necessary action.
3. Make file base backup using pg\_basebackup and restore

## Lab Exercise – 8

1. Create a new database pitrdb and insert records.
2. Take backup using pg\_basebackup() in .tar.gz format.
3. Set recovery target time.
4. Restore these backup and keep database running.