*********Backup and Restore****************

pg_dump` and `pg_dumpall` are two utilities provided by PostgreSQL for creating logical backups of databases.

1. Logical Backup:-

- pg_dump,pg_dumpall and pg_restore utilities are used for logical backups.
- pg_dump will help in taking backups for Database level, Schema level and Table Level.
- Pg_dumpall used for cluster level dump.

A. pg_dump:-

- pg_dump will help in taking backups for Database level, Schema level and Table Level.
- pg_dump` is used to back up a single PostgreSQL database into a script or archive file. The backup can be restored later using `psql` or `pg_restore`.

Syntax:-

 Pg_dump -U user_name -d database_name > location(/home/gaurav)

Step01:- Make directory and owner of this directory is postgres

```
[root@standby /]# mkdir backup
[root@standby /]# chown 'postgres:postgres' backup
```

Step 02:-

```
[root@standby /]# su - postgres
```

Step 03:-

```
bash-4.2$ pg_dump -U postgres -d gaurav > /backup/pgbackup
```

```
$ pg_dump -U username -Fp dbname  >  filename.txt
$ pg_dump -U username dbname -f  filename.txt
$ pg_dump -Fp -U username dbname -f  filename.txt
```

-U =user_name

-d=databasename

-F = format

1. Fp =plain

2. Fc=custom

3. Ft = tar

1. **Plain SQL Script:**

pg_dump -U username -W -F p -f output_file.sql database_name

— `-U username`: Specifies the database user.
— `-W`: Prompts for the password.
— `-F p`: Specifies the format as plain SQL.
— `-f output_file.sql`: Output file.
— `database_name`: Name of the database to back up.

2. **Custom Archive Format:**

pg_dump -U username -W -F c -f output_file.dump database_name

— `-F c`: Specifies the format as a custom archive.
— Custom format allows for more flexible restoration options using `pg_restore`.

3. **Directory Format:**

pg_dump -U username -W -F d -f output_directory database_name

— `-F d`: Specifies the format as a directory.
— The output is a directory with one file per table, allowing for parallel restoration.

4. **Compressing Output:**

pg_dump -U username -W -F c -Z 9 -f output_file.dump database_name

— `-Z 9`: Compresses the output with maximum compression.

5. **Backing Up Specific Tables:**

pg_dump -U username -W -F c -f output_file.dump -t table_name database_name

— `-t table_name`: Specifies a table to back up.

6. **Exclude ownership and Privliage**

pg_dump -U smile_mod -h 10.189.8.160 -p 54032 -d polc_claims_db --schema claims -v -x -O -f /backup_25/calmis.sql > backup.log 2>&1

-v :- Verbose mode — shows detailed progress output.
-x :- Exclude privileges (GRANT/REVOKE) from the dump.
-O :- Exclude ownership information (i.e., no OWNER TO ... lines).

-Fc :- Dump in custom format, useful for selective restoration via pg_restore.

7.   Parallel cpu:-

pg_dump -U postgres -h localhost -d mydatabase -Fd -j 4 -Z 9 -f mydatabase_backup_dir

******Compress backup file************

Synatx:-
Pg_dump -U postgres -d gaurav | gzip > /backup/pgback

B. pg_dumpall:-
- It is used to take backup of entire cluster

- `pg_dumpall` is used to back up an entire PostgreSQL database cluster, including all databases, roles, and tablespaces.

Step 01:-

```
[root@standby /]# mkdir backup
[root@standby /]# chown 'postgres:postgres' backup
```

Step 02:-

```
[root@standby /]# su - postgres
```

Step 03:-

```
-bash-4.2$ pg_dumpall -U postgers > /backup/all
```

1. Plain SQL Script:

pg_dumpall -U username -W -f output_file.sql
— Backs up all databases, roles, and tablespaces into a single plain SQL script file.
— This file can be restored using `psql`.

2. Backing Up Globals Only:

pg_dumpall -U username -W - globals-only -f globals.sql
— `— globals-only`: Backs up only the global objects such as roles and tablespaces.

Differences Between pg_dump ,pg_dumpall and pg_basebackup

1. Scope:

— `pg_dump`: Backs up a single database.
— `pg_dumpall`: Backs up all databases in a cluster, including roles and tablespaces.

2. Output Formats:

— `pg_dump`: Supports multiple output formats (plain, custom, directory, tar).
— `pg_dumpall`: Only outputs plain SQL scripts.

3. Restoration:

— `pg_dump`: The backup can be restored using `psql` (for plain SQL format) or `pg_restore` (for other formats).
— `pg_dumpall`: The backup is restored using `psql`.

4. Use Cases:

— `pg_dump`: Used for backing up individual databases, especially when specific customization or flexibility in restoration is required.
— `pg_dumpall`: Used for comprehensive backups of the entire PostgreSQL instance, including all databases, roles, and tablespaces.

**********<mark>Restore (pg_restore or psql)</mark>***********

- If your dumps(backup) are Custom or Tar Format you need to use only **pg_restore** utility
- If your dumps(backup) are Plain SQL format you need to use **psql** utility

Syntax:-

<mark>**1. psql:-**</mark>
Psql -U postgres -d jk < /backup/pgbackup

```
CREATE DATABASE
postgres=# \q
-bash-4.2$ psql -U postgres -d jk < /backup/pgbackup
```

<mark style="background-color:red">It is used when backup file in plain format</mark>

```
all   bk.tar   pgbackup   tar
-bash-4.2$ pg_restore -U postgres -d kh /backup/bk.tar
```

It is used when backup in tar format

*********************pg_basebackup*********************

- Pg_basebackup is used to take the base backup of running PostgreSQL database cluster.
- This backup can be used for PITR or replication.
- Bckups are always taken of the entire cluster and cannot be used for single database or objects

Syntax:-

Pg_basebackup -h <ip address> -p <port number> -U <user_name> -D <directory location> -Ft -Xs -R -P

-h =hostname

-p = port number

-U =username

-D = directory location

-F = format

-Xs = wal method - stream

-R = write configuration parameter for replication

-P = postures information

```
-bash-4.2$ pg_basebackup -U postgres -D /backup/pgbase -Fp -P -Xs -R
53283/53283 kB (100%), 1/1 tablespace
-bash-4.2$ cd /
```

3.   **Backup command :-**

/usr/pgsql-15/bin/pg_basebackup -U postgres -p 5432 -D
$backup_dir/PostgreSQL_Base_Backup_$(date +"%d-%m-%y-%H%M%S") -l "`date`" -P -Ft -R -z

-U postgres → Use PostgreSQL user postgres.

-h /tmp → Connect to PostgreSQL using the Unix socket in /tmp.

-p 4702 → Port number = 4702.

-D $backup_dir/... → Destination directory for the backup.

Here it's timestamped → PostgreSQL_Base_Backup_DD-MM-YY-HHMMSS.

-l "$(date)" → Backup label, will be current date/time.

-P → Show progress during backup.

-Ft → Write output as a tar archive.

-z → Compress the backup with gzip.

-R → Write a standby.signal file and a correct postgresql.auto.conf with primary_conninfo.

This makes the backup ready to start as a replica (standby server).

4. controll bandwidth:-

pg_basebackup -h <server_ip> -U <replication_user> -D /backup_dir --progress --checkpoint-timeout=300

5. Replication:-

With checkpoint option

pg_basebackup   -D /data/pgdata --checkpoint=fast -h 100.127.129.45 -p 5432 -Xs -R -P > backup.log 2>&1

Without option

pg_basebackup   -D /data/pgdata -h 100.127.129.45 -p 5432 -Xs -R -P > backup.log 2>&1

-D /var/lib/pgsql/15/data → Target data directory for restore.

-h 192.168.29.193 → Source PostgreSQL server IP.

-p 5432 → Port number.

-Xs → Include WAL files in tar or plain format.

-R → Automatically write standby.signal and primary_conninfo (for streaming replication).

-P → Show progress.

> backup.log 2>&1 → Logs all output (stdout + stderr) to backup.log.

******************role******************

The following command will export the roles only:_

```
pg_dumpall --roles-only -U postgres > all_db_roles.sql
```

The following command will export schemas only:-
```
pg_dumpall --schema-only -U postgres > all_db_schema.sql
```

The following command will export only the tablespace definitions:-
```
pg_dumpall --tablespaces-only -U postgres >
all_db_tablespaces.sql
```

****** Backup and Restortion****

1. First we take full backup on server:- 20.243.32.40 With the help pg_dumpall command.
   Command:- pg_dumpall -U postgres > full.bkp.sql

2. Than we transfer this file on destination server 40.81.213.8.

3. We restore this file with help psql command.

Command: - psql -U postgres < full_bkp.sql