PG_DUMP

The pg_dump tool is a command-line utility that you can use to create a logical backup of a PostgreSQL database. Specifically, it can dump an entire database or specific parts of it, such as individual tables or schemas. The output of the utility can be:

An SQL script: A plain-text file containing the SQL commands required to reconstruct the database to the state it was in at the time of the backup. To execute the script, you can use psql.

A directory-based archive file: The resulting format is a set of folders, and it is designed to be portable across different architectures. To rebuild the database, you must import these archive files with pg_restore.

Here is a list of the most important and commonly used options for the pg_dump command in PostgreSQL:-

U <username> or --username=<username>: Specifies the PostgreSQL username to connect with.

h <hostname> or --host=<hostname> Specifies the host where the database server is running. If set, the default is taken from the PGHOST environment variable.

p <port> or --port=<port> Specifies the port to use for the connection to the database. If set, the default value is read from the PGPORT environment variable.

d <dbname> or --dbname=*dbname*: Specifies the name of the database to dump. <dbname> can also be a connection string.

F <format> or --format=<format>: Specifies the file format for the export. Common values include:

t or text: For an SQL-like plain text script (default option).

c or custom: For a custom format.

d or directory: For the directory format suitable for input into pg_restore.

t or tar: For the tar archive directory format suitable for input into pg_restore.

f <filename> or --file=<filename>: Specifies the name of the output file.

t <pattern> or --table=<pattern>: Dumps only the tables specified by the given pattern.

T <pattern> or --exclude-table=<*pattern*>: Excludes the tables identified by the specified pattern from the dump.

n <pattern> or --schema=<pattern>: Dumps only the schemas matching the pattern.

a or --data-only: Dumps only the data, not the schema.

s or --schema-only: Exports only the schema, not the data.

c or --clean: Adds SQL commands to drop database objects before recreating them.

C or --create: Adds SQL commands to create the database.

-inserts: Dumps data as INSERT commands rather than COPY instructions.

-no-password: Skips the password prompt by assuming that no password is required.

Example :-

```
pg_dump -U admin -d company -f company_backup.sql
```

Dump a Database Into a Directory-Format Archive:

```
pg_dump -U admin -d company -F d -f company_backup
```

To generate a .tar file with the same structure, run instead:

```
pg_dump -U admin -d company -F t -f company_backup.tar
```

Export Schema Only:

```
pg_dump -U admin -d company -f company_backup.sql --schema-only
```

Include Only a Few Tables :-

Assume you only want to dump tables that contain the word “order.” Achieve that goal with:

```
pg_dump -U admin -d company -t '*order*' -f company_backup.sql
```

Is it possible to run pg_dump from remote server?

Yes, you can run pg_dump from remote server.Use the -h option to specify the hostname or IP address of the PostgreSQL server.

Provide the -U option to specify the PostgreSQL username.

Include the -d option to specify the name of the remote database.

How to perform a parallel dump with pg_dump?

To perform a parallel dump with pg_dump in PostgreSQL, use the -j or --jobs option followed by the number of parallel jobs. For example, to run two parallel jobs, add -j 2 to your pg_dump command. Bear in mind that parallel dumping can significantly speed up the backup process, especially for large databases. At the same time, it usually takes more memory resources.

pgdump_all

`pg_dumpall` in PostgreSQL is used to back up an entire PostgreSQL cluster, including all databases, roles, and tablespaces. `pg_dumpall` always creates a **plain-text SQL** file. Unlike the pg_dump tool which backups individual databases or objects, the pg_dumpall tool offers a convenient way to make a backup of all databases in an PostgreSQL cluster (instance) in a single operation.

Usage:

pg_dumpall [OPTION]...

General options:

-f, --file=FILENAME output file name

-v, --verbose verbose mode

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

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

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

Options controlling the output content:

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

-c, --clean clean (drop) databases before recreating

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

-g, --globals-only dump only global objects, no databases

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

-r, --roles-only dump only roles, no databases or tablespaces

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

-S, --superuser=NAME superuser user name to use in the dump

-t, --tablespaces-only dump only tablespaces, no databases or roles

-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

--exclude-database=PATTERN exclude databases whose name matches PATTERN

--extra-float-digits=NUM override default setting for extra_float_digits

--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-role-passwords do not dump passwords for roles

```
  --no-security-labels        do not dump security label assignments
  --no-subscriptions          do not dump subscriptions
  --no-sync                   do not wait for changes to be written safely to disk
  --no-table-access-method    do not dump table access methods
  --no-tablespaces            do not dump tablespace assignments
  --no-toast-compression      do not dump TOAST compression methods
  --no-unlogged-table-data    do not dump unlogged table data
  --on-conflict-do-nothing    add ON CONFLICT DO NOTHING to INSERT commands
  --quote-all-identifiers     quote all identifiers, even if not key words
  --rows-per-insert=NROWS     number of rows per INSERT; implies --inserts
  --use-set-session-authorization
                              use SET SESSION AUTHORIZATION commands instead of
                              ALTER OWNER commands to set ownership

Connection options:
  -d, --dbname=CONNSTR        connect using connection string
  -h, --host=HOSTNAME         database server host or socket directory
  -l, --database=DBNAME       alternative default database
  -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
```

Example :-

The basic syntax of the pgdump_all command is shown below:

```
pg_dumpall -f backupfile_name.sql
```

Backing up all databases using the pg_dumpall utility:

```
pg_dumpall -U postgres > D:\backup\all_databases.sql
```

If you want to back up role definition only, use the following command:

pg_dumpall --roles-only > D:\backup\roles.sql

If you want to back up tablespaces definition, use the following command:

pg_dumpall --tablespaces-only > d:\backup\tablespaces.sql

Backup with a Specific User:

```
pg_dumpall -U postgres -f /path/to/backup.sql
```

Explanation: This command uses the -U option to specify the user (in this case, postgres) for running the backup. The user must have superuser privileges to dump all databases. If not specified, it uses the current OS user.

Backup and Compress in Real Time :-

```
pg_dumpall | gzip > /path/to/backup.sql.gz
```

Explanation: This command pipes the output of pg_dumpall directly into the gzip command, which compresses the SQL backup file. The result is a smaller .gz file, useful for saving storage space and reducing transfer times for backups.

Backup and Output to Standard Output (stdout) :-

Explanation: Without the -f option, pg_dumpall writes the output to standard output (your terminal or console). This is useful if you want to inspect the SQL dump or pipe it into another command or tool.

Restore Example

To restore the backup created by `pg_dumpall`, use the `psql` command:

1. **Restoring from a Plain SQL Backup:**

   ```
   bash
   Copy code
   psql -U postgres -f /path/to/backup.sql
   ```

2. **Restoring from a Compressed Backup:**

   ```
   bash
   Copy code
   gunzip -c /path/to/backup.sql.gz | psql -U postgres
   ```

   - **Explanation**: If the backup file is compressed (`.gz`), you need to decompress it using `gunzip` before passing it to `psql` for restoring.

Important Notes:

- `pg_dumpall` always creates a **plain-text SQL** file. You can restore it using `psql`, not `pg_restore`, because it's not in a custom format.
- Make sure to back up and restore **global objects** (like roles and tablespaces) separately if needed.
- Since `pg_dumpall` includes global objects, you typically need superuser privileges to run it.