pg_restore

https://www.postgresql.org/docs/8.1/static/app-pgrestore.html

Name

pg_restore -- restore a PostgreSQL database from an archive file created by pg_dump

Synopsis

`pg_restore` [`option`...] [`filename`]

Description

pg_restore is a utility for restoring a PostgreSQL database from an archive created by pg_dump in one of the non-plain-text formats. It will issue the commands necessary to reconstruct the database to the state it was in at the time it was saved. The archive files also allow pg_restore to be selective about what is restored, or even to reorder the items prior to being restored. The archive files are designed to be portable across architectures.

pg_restore can operate in two modes. If a database name is specified, pg_restore connects to that database and restores archive contents directly into the database. Otherwise, a script containing the SQL commands necessary to rebuild the database is created and written to a file or standard output. This script output is equivalent to the plain text output format of pg_dump. Some of the options controlling the output are therefore analogous to pg_dump options.

Obviously, pg_restore cannot restore information that is not present in the archive file. For instance, if the archive was made using the "dump data as `INSERT` commands" option, pg_restore will not be able to load the data using `COPY` statements.

Options

pg_restore accepts the following command line arguments.

`filename`

Specifies the location of the archive file to be restored. If not specified, the standard input is used.

`-a`
`--data-only`

Restore only the data, not the schema (data definitions).

`-c`
`--clean`

Clean (drop) database objects before recreating them.

`-C`
`--create`

Create the database before restoring into it. (When this option is used, the database named with `-d` is used only to issue the initial `CREATE DATABASE` command. All data is restored into the database name that appears in the archive.)

`-d dbname`
`--dbname=dbname`

Connect to database `dbname` and restore directly into the database.

`-e`
`--exit-on-error`

Exit if an error is encountered while sending SQL commands to the database. The default is to continue and to display a count of errors at the end of the restoration.

`-f filename`
`--file=filename`

Specify output file for generated script, or for the listing when used with `-l`. Default is the standard output.

`-F format`
`--format=format`

Specify format of the archive. It is not necessary to specify the format, since pg_restore will determine the format automatically. If specified, it can be one of the following:

`t`

The archive is a `tar` archive.

`c`

The archive is in the custom format of pg_dump.

`-i`
`--ignore-version`

Ignore database version checks.

`-I index`
`--index=index`

Restore definition of named index only.

`-l`
`--list`

List the contents of the archive. The output of this operation can be used as input to the `-L` option. Note that if filtering switches such as `-n` or `-t` are used with `-l`, they will restrict the items listed.

`-L list-file`
`--use-list=list-file`

Restore only those archive elements that are listed in `list-file`, and restore them in the order they appear in the file. Note that if filtering switches such as `-n` or `-t` are used with `-L`, they will further restrict the items restored.

`list-file` is normally created by editing the output of a previous `-l` operation. Lines can be moved or removed, and can also be commented out by placing a semicolon (`;`) at the start of the line. See below for examples.

`-n namespace`
`--schema=schema`

Restore only objects that are in the named schema. This can be combined with the `-t` option to restore just a specific table.

`-O`
`--no-owner`

Do not output commands to set ownership of objects to match the original database. By default, pg_restore issues `ALTER OWNER` or `SET SESSION AUTHORIZATION` statements to set ownership of created schema elements. These statements will fail unless the initial connection to the database is made by a superuser (or the same user that owns all of the objects in the script). With `-O`, any user name can be used for the initial connection, and this user will own all the created objects.

`-P function-name(argtype [, ...])`
`--function=function-name(argtype [, ...])`

Restore the named function only. Be careful to spell the function name and arguments exactly as they appear in the dump file's table of contents.

`-R`
`--no-reconnect`

This option is obsolete but still accepted for backwards compatibility.

`-s`
`--schema-only`

Restore only the schema (data definitions), not the data (table contents). Sequence current values will not be restored, either. (Do not confuse this with the `--schema` option, which uses the word "schema" in a different meaning.)

`-S username`
`--superuser=username`

Specify the superuser user name to use when disabling triggers. This is only relevant if `--disable-triggers` is used.

`-t table`
`--table=table`

Restore definition and/or data of named table only.

`-T trigger`
`--trigger=trigger`

Restore named trigger only.

`-v`
`--verbose`

Specifies verbose mode.

`-x`
`--no-privileges`
`--no-acl`

Prevent restoration of access privileges (grant/revoke commands).

`-X use-set-session-authorization`
`--use-set-session-authorization`

Output SQL-standard `SET SESSION AUTHORIZATION` commands instead of `ALTER OWNER` commands to determine object ownership. This makes the dump more standards compatible, but depending on the history of the objects in the dump, may not restore properly.

`-X disable-triggers`
`--disable-triggers`

This option is only relevant when performing a data-only restore. It instructs pg_restore to execute commands to temporarily disable triggers on the target tables while the data is reloaded. Use this if you have referential integrity checks or other triggers on the tables that you do not want to invoke during data reload.

Presently, the commands emitted for `--disable-triggers` must be done as superuser. So, you should also specify a superuser name with `-S`, or preferably run pg_restore as a PostgreSQL superuser.

pg_restore also accepts the following command line arguments for connection parameters:

`-h host`
`--host=host`

Specifies the host name of the machine on which the server is running. If the value begins with a slash, it is used as the directory for the Unix domain socket. The default is taken from the `PGHOST` environment variable, if set, else a Unix domain socket connection is attempted.

`-p port`
`--port=port`

Specifies the TCP port or local Unix domain socket file extension on which the server is listening for connections. Defaults to the `PGPORT` environment variable, if set, or a compiled-in default.

`-U username`

Connect as the given user

`-W`

Force a password prompt. This should happen automatically if the server requires password authentication.

Environment

`PGHOST`
`PGPORT`
`PGUSER`

Default connection parameters

Diagnostics

When a direct database connection is specified using the `-d` option, pg_restore internally executes SQL statements. If you have problems running pg_restore, make sure you are able to select information from the database using, for example, psql.

Notes

If your installation has any local additions to the `template1` database, be careful to load the output of pg_restore into a truly empty database; otherwise you are likely to get errors due to duplicate definitions of the added objects. To make an empty database without any local additions, copy from `template0` not `template1`, for example:

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

The limitations of pg_restore are detailed below.

- When restoring data to a pre-existing table and the option `--disable-triggers` is used, pg_restore emits commands to disable triggers on user tables before inserting the data then emits commands to re-enable them after the data has been inserted. If the restore is stopped in the middle, the system catalogs may be left in the wrong state.
- pg_restore will not restore large objects for a single table. If an archive contains large objects, then all large objects will be restored.

See also the pg_dump documentation for details on limitations of pg_dump.

Once restored, it is wise to run `ANALYZE` on each restored table so the optimizer has useful statistics.

Examples

To dump a database called `mydb` to a `tar` file:

```
$ pg_dump -Ft mydb > db.tar
```

To reload this dump into an existing database called `newdb`:

```
$ pg_restore -d newdb db.tar
```

To reorder database items, it is first necessary to dump the table of contents of the archive:

```
$ pg_restore -l archive.file > archive.list
```

The listing file consists of a header and one line for each item, e.g.,

```
;
; Archive created at Fri Jul 28 22:28:36 2000
;     dbname: birds
;     TOC Entries: 74
;     Compression: 0
;     Dump Version: 1.4-0
;     Format: CUSTOM
;
;
; Selected TOC Entries:
;
2; 145344 TABLE species postgres
3; 145344 ACL species
4; 145359 TABLE nt_header postgres
5; 145359 ACL nt_header
6; 145402 TABLE species_records postgres
7; 145402 ACL species_records
8; 145416 TABLE ss_old postgres
9; 145416 ACL ss_old
10; 145433 TABLE map_resolutions postgres
11; 145433 ACL map_resolutions
12; 145443 TABLE hs_old postgres
13; 145443 ACL hs_old
```

Semicolons start a comment, and the numbers at the start of lines refer to the internal archive ID assigned to each item.

Lines in the file can be commented out, deleted, and reordered. For example,

```
10; 145433 TABLE map_resolutions postgres
;2; 145344 TABLE species postgres
;4; 145359 TABLE nt_header postgres
6; 145402 TABLE species_records postgres
;8; 145416 TABLE ss_old postgres
```

could be used as input to pg_restore and would only restore items 10 and 6, in that order:

```
$ pg_restore -L archive.list archive.file
```

History

The pg_restore utility first appeared in PostgreSQL 7.1.

See Also

pg_dump, pg_dumpall, psql, Environment Variables (Section 28.11)
