NAME

pg_dumpall - extract a PostgreSQL database cluster into a script file

SYNOPSIS

pg_dumpall [*connection-option*...] [*option*...]

DESCRIPTION

pg_dumpall is a utility for writing out ("dumping") all PostgreSQL databases of a cluster into one script file. The script file contains SQL commands that can be used as input to **psql**(1) to restore the databases. It does this by calling **pg_dump**(1) for each database in the cluster. pg_dumpall also dumps global objects that are common to all databases, namely database roles, tablespaces, and privilege grants for configuration parameters. (pg_dump does not save these objects.)

Since pg_dumpall reads tables from all databases you will most likely have to connect as a database superuser in order to produce a complete dump. Also you will need superuser privileges to execute the saved script in order to be allowed to add roles and create databases.

The SQL script will be written to the standard output. Use the **-f**/**--file** option or shell operators to redirect it into a file.

pg_dumpall needs to connect several times to the PostgreSQL server (once per database). If you use password authentication it will ask for a password each time. It is convenient to have a ~/.pgpass file in such cases. See Section 34.16 for more information.

OPTIONS

The following command-line options control the content and format of the output.

-a
--data-only
Dump only the data, not the schema (data definitions).

-c
--clean
Include SQL commands to clean (drop) databases before recreating them. **DROP** commands for roles and tablespaces are added as well.

-E *encoding*
--encoding=*encoding*
Create the dump in the specified character set encoding. By default, the dump is created in the database encoding. (Another way to get the same result is to set the **PGCLIENTENCODING**

environment variable to the desired dump encoding.)

-f *filename*
--file=*filename*
Send output to the specified file. If this is omitted, the standard output is used.

-g
--globals-only
Dump only global objects (roles and tablespaces), no databases.

-O
--no-owner
Do not output commands to set ownership of objects to match the original database. By default, pg_dumpall issues **ALTER OWNER** or **SET SESSION AUTHORIZATION** statements to set ownership of created schema elements. These statements will fail when the script is run unless it is started by a superuser (or the same user that owns all of the objects in the script). To make a script that can be restored by any user, but will give that user ownership of all the objects, specify **-O**.

-r
--roles-only
Dump only roles, no databases or tablespaces.

-s
--schema-only
Dump only the object definitions (schema), not data.

-S *username*
--superuser=*username*
Specify the superuser user name to use when disabling triggers. This is relevant only if **--disable-triggers** is used. (Usually, it’s better to leave this out, and instead start the resulting script as superuser.)

-t
--tablespaces-only
Dump only tablespaces, no databases or roles.

-v
--verbose
Specifies verbose mode. This will cause pg_dumpall to output start/stop times to the dump file, and progress messages to standard error. Repeating the option causes additional debug-level

messages to appear on standard error. The option is also passed down to pg_dump.

-V
--version
Print the pg_dumpall version and exit.

-x
--no-privileges
--no-acl
Prevent dumping of access privileges (grant/revoke commands).

--binary-upgrade
This option is for use by in-place upgrade utilities. Its use for other purposes is not recommended or supported. The behavior of the option may change in future releases without notice.

--column-inserts
--attribute-inserts
Dump data as **INSERT** commands with explicit column names (INSERT INTO *table* (*column*, ...) VALUES ...). This will make restoration very slow; it is mainly useful for making dumps that can be loaded into non-PostgreSQL databases.

--disable-dollar-quoting
This option disables the use of dollar quoting for function bodies, and forces them to be quoted using SQL standard string syntax.

--disable-triggers
This option is relevant only when creating a data-only dump. It instructs pg_dumpall to include commands to temporarily disable triggers on the target tables while the data is restored. Use this if you have referential integrity checks or other triggers on the tables that you do not want to invoke during data restore.

Presently, the commands emitted for **--disable-triggers** must be done as superuser. So, you should also specify a superuser name with **-S**, or preferably be careful to start the resulting script as a superuser.

--exclude-database=*pattern*
Do not dump databases whose name matches *pattern*. Multiple patterns can be excluded by writing multiple **--exclude-database** switches. The *pattern* parameter is interpreted as a pattern according to the same rules used by psql’s \d commands (see Patterns below), so multiple databases can also be excluded by writing wildcard characters in the pattern. When using

wildcards, be careful to quote the pattern if needed to prevent shell wildcard expansion.

--extra-float-digits=*ndigits*
Use the specified value of extra_float_digits when dumping floating-point data, instead of the maximum available precision. Routine dumps made for backup purposes should not use this option.

--if-exists
Use conditional commands (i.e., add an IF EXISTS clause) to drop databases and other objects. This option is not valid unless **--clean** is also specified.

--inserts
Dump data as **INSERT** commands (rather than **COPY**). This will make restoration very slow; it is mainly useful for making dumps that can be loaded into non-PostgreSQL databases. Note that the restore might fail altogether if you have rearranged column order. The **--column-inserts** option is safer, though even slower.

--load-via-partition-root
When dumping data for a table partition, make the **COPY** or **INSERT** statements target the root of the partitioning hierarchy that contains it, rather than the partition itself. This causes the appropriate partition to be re-determined for each row when the data is loaded. This may be useful when restoring data on a server where rows do not always fall into the same partitions as they did on the original server. That could happen, for example, if the partitioning column is of type text and the two systems have different definitions of the collation used to sort the partitioning column.

--lock-wait-timeout=*timeout*
Do not wait forever to acquire shared table locks at the beginning of the dump. Instead, fail if unable to lock a table within the specified *timeout*. The timeout may be specified in any of the formats accepted by **SET statement_timeout**.

--no-comments
Do not dump comments.

--no-publications
Do not dump publications.

--no-role-passwords
Do not dump passwords for roles. When restored, roles will have a null password, and password authentication will always fail until the password is set. Since password values aren’t needed when this option is specified, the role information is read from the catalog view pg_roles instead

of pg_authid. Therefore, this option also helps if access to pg_authid is restricted by some security policy.

--no-security-labels
Do not dump security labels.

--no-subscriptions
Do not dump subscriptions.

--no-sync
By default, **pg_dumpall** will wait for all files to be written safely to disk. This option causes **pg_dumpall** to return without waiting, which is faster, but means that a subsequent operating system crash can leave the dump corrupt. Generally, this option is useful for testing but should not be used when dumping data from production installation.

--no-table-access-method
Do not output commands to select table access methods. With this option, all objects will be created with whichever table access method is the default during restore.

--no-tablespaces
Do not output commands to create tablespaces nor select tablespaces for objects. With this option, all objects will be created in whichever tablespace is the default during restore.

--no-toast-compression
Do not output commands to set TOAST compression methods. With this option, all columns will be restored with the default compression setting.

--no-unlogged-table-data
Do not dump the contents of unlogged tables. This option has no effect on whether or not the table definitions (schema) are dumped; it only suppresses dumping the table data.

--on-conflict-do-nothing
Add ON CONFLICT DO NOTHING to **INSERT** commands. This option is not valid unless **--inserts** or **--column-inserts** is also specified.

--quote-all-identifiers
Force quoting of all identifiers. This option is recommended when dumping a database from a server whose PostgreSQL major version is different from pg_dumpall's, or when the output is intended to be loaded into a server of a different major version. By default, pg_dumpall quotes only identifiers that are reserved words in its own major version. This sometimes results in

compatibility issues when dealing with servers of other versions that may have slightly different sets of reserved words. Using **--quote-all-identifiers** prevents such issues, at the price of a harder-to-read dump script.

--rows-per-insert=*nrows*

Dump data as **INSERT** commands (rather than **COPY**). Controls the maximum number of rows per **INSERT** command. The value specified must be a number greater than zero. Any error during restoring will cause only rows that are part of the problematic **INSERT** to be lost, rather than the entire table contents.

--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, might not restore properly.

-?
--help

Show help about pg_dumpall command line arguments, and exit.

The following command-line options control the database connection parameters.

-d *connstr*
--dbname=*connstr*

Specifies parameters used to connect to the server, as a connection string; these will override any conflicting command line options.

The option is called --dbname for consistency with other client applications, but because pg_dumpall needs to connect to many databases, the database name in the connection string will be ignored. Use the -l option to specify the name of the database used for the initial connection, which will dump global objects and discover what other databases should be dumped.

-h *host*
--host=*host*

Specifies the host name of the machine on which the database 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.

-l *dbname*
--database=*dbname*

Specifies the name of the database to connect to for dumping global objects and discovering what other databases should be dumped. If not specified, the postgres database will be used, and if that does not exist, template1 will be used.

-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*
--username=*username*

User name to connect as.

-w
--no-password

Never issue a password prompt. If the server requires password authentication and a password is not available by other means such as a .pgpass file, the connection attempt will fail. This option can be useful in batch jobs and scripts where no user is present to enter a password.

-W
--password

Force pg_dumpall to prompt for a password before connecting to a database.

This option is never essential, since pg_dumpall will automatically prompt for a password if the server demands password authentication. However, pg_dumpall will waste a connection attempt finding out that the server wants a password. In some cases it is worth typing **-W** to avoid the extra connection attempt.

Note that the password prompt will occur again for each database to be dumped. Usually, it’s better to set up a ~/.pgpass file than to rely on manual password entry.

--role=*rolename*

Specifies a role name to be used to create the dump. This option causes pg_dumpall to issue a **SET ROLE** *rolename* command after connecting to the database. It is useful when the authenticated user (specified by **-U**) lacks privileges needed by pg_dumpall, but can switch to a role with the required rights. Some installations have a policy against logging in directly as a superuser, and use of this option allows dumps to be made without violating the policy.

ENVIRONMENT

PGHOST

PGOPTIONS
PGPORT
PGUSER

Default connection parameters

PG_COLOR

Specifies whether to use color in diagnostic messages. Possible values are always, auto and never.

This utility, like most other PostgreSQL utilities, also uses the environment variables supported by libpq (see Section 34.15).

NOTES

Since pg_dumpall calls pg_dump internally, some diagnostic messages will refer to pg_dump.

The **--clean** option can be useful even when your intention is to restore the dump script into a fresh cluster. Use of **--clean** authorizes the script to drop and re-create the built-in postgres and template1 databases, ensuring that those databases will retain the same properties (for instance, locale and encoding) that they had in the source cluster. Without the option, those databases will retain their existing database-level properties, as well as any pre-existing contents.

Once restored, it is wise to run **ANALYZE** on each database so the optimizer has useful statistics. You can also run **vacuumdb -a -z** to analyze all databases.

The dump script should not be expected to run completely without errors. In particular, because the script will issue **CREATE ROLE** for every role existing in the source cluster, it is certain to get a "role already exists" error for the bootstrap superuser, unless the destination cluster was initialized with a different bootstrap superuser name. This error is harmless and should be ignored. Use of the **--clean** option is likely to produce additional harmless error messages about non-existent objects, although you can minimize those by adding **--if-exists**.

pg_dumpall requires all needed tablespace directories to exist before the restore; otherwise, database creation will fail for databases in non-default locations.

EXAMPLES

To dump all databases:

```
$ pg_dumpall > db.out
```

To restore database(s) from this file, you can use:

```
$ psql -f db.out postgres
```

It is not important to which database you connect here since the script file created by pg_dumpall will contain the appropriate commands to create and connect to the saved databases. An exception is that if you specified **--clean**, you must connect to the postgres database initially; the script will attempt to drop other databases immediately, and that will fail for the database you are connected to.

SEE ALSO

Check **pg_dump**(1) for details on possible error conditions.