pg_dump and pg_restore PostgreSQL backups

CONCEPTS

pg_dump saves one database; roles and tablespaces live cluster-wide, so add pg_dumpall -g.

Pick a format (-F)

Flag	Format	Restore with	Parallel -j
`-Fp`	plain SQL (default)	psql	no
`-Fc`	custom archive	pg_restore	restore only
`-Fd`	directory	pg_restore	dump and restore
`-Ft`	tar	pg_restore	no

Database live, consistent
↓ pg_dump
Dump SQL or archive
↓ psql / pg_restore
New database empty target

Dumps are consistent snapshots and do not block writers, so you can dump a live database.

Restore as a superuser into a fresh database created from template0, or extensions and ownership may fail.

DUMP A DATABASE

pg_dump *<db>* **> db.sql**
Plain SQL script to stdout

pg_dump -f db.sql *<db>*
Write the script to a file

pg_dump -Fc -f *<file>* *<db>*
Custom-format archive

pg_dump -Fd -f *<dir>* *<db>*
Directory-format archive

pg_dump -Fd -j *<n>* **-f** *<dir>* *<db>*
Dump n tables in parallel

CHOOSE WHAT TO DUMP

pg_dump -s *<db>*
Schema only, no data

pg_dump -a *<db>*
Data only, no schema

pg_dump -t *<table>* *<db>*
Only matching tables

pg_dump -T *<table>* *<db>*
Skip matching tables

CHOOSE WHAT TO DUMP (cont.)

pg_dump -n *<schema>* *<db>*
Only matching schemas

pg_dump -N *<schema>* *<db>*
Skip matching schemas

pg_dump --exclude-table-data=*<table>* *<db>*
Keep table definition, drop rows

pg_dump -t '*<schema>***.***<prefix>****'** *<db>*
Wildcard table pattern

pg_dump --table-and-children=*<table>* *<db>*
Table plus its partitions

pg_dump --filter=*<file>* *<db>*
Read include/exclude rules from file

pg_dump --no-owner --no-privileges *<db>*
Omit owners and GRANTs

RESTORE A DUMP

psql -X -d *<db>* **-f db.sql**
Restore a plain SQL dump

pg_restore -d *<db>* *<file>*
Restore an archive into a database

pg_restore -l *<file>*
List the archive's contents

pg_restore -j *<n>* **-d** *<db>* *<file>*
Restore with n parallel jobs

createdb -T template0 *<db>*
Create an empty target first

pg_restore -C -d postgres *<file>*
Create the database from the archive

pg_restore -c --if-exists -d *<db>* *<file>*
Drop objects before recreating

pg_restore -s -d *<db>* *<file>*
Schema only

pg_restore -a -d *<db>* *<file>*
Data only

pg_restore -t *<table>* **-d** *<db>* *<file>*
Restore a single table

pg_restore -n *<schema>* **-d** *<db>* *<file>*
Restore a single schema

RESTORE A DUMP (cont.)

pg_restore -1 -d *<db>* *<file>*
All or nothing in one transaction

pg_restore -e -d *<db>* *<file>*
Stop at the first error

pg_restore -l *<file>* **> list.txt**
Save the contents list to edit

pg_restore -L list.txt -d *<db>* *<file>*
Restore only the edited list

pg_restore -f script.sql *<file>*
Convert an archive to SQL text

WHOLE CLUSTER

pg_dumpall > all.sql
Every database, roles, tablespaces

pg_dumpall -g > globals.sql
Roles and tablespaces only

psql -X -f all.sql postgres
Restore a pg_dumpall script

CONNECTION

pg_dump -h *<host>* **-p** *<port>* **-U** *<user>* *<db>*
Connect to another server

pg_dump -d "*<connstring>***"**
Use a connection string or URI

Environment variables

Variable	Meaning
`PGHOST`	server host
`PGPORT`	server port
`PGUSER`	user name
`PGDATABASE`	default database (pg_dump)
`PGPASSFILE`	password file path

pg_restore does not read PGDATABASE. Without -d it prints SQL instead of restoring.

RECIPES

Dated backup, parallel

```
pg_dump -Fd -j 4 -f
"backup_$(date +%F)" <db>
pg_dumpall -g > "globals_$(date +
%F).sql"
```

Clone a database to a new server

```
createdb -h <newhost> -T
template0 <db>
pg_dump -Fc <db> | pg_restore -h
<newhost> -d <db>
```