pg_dump and pg_restore PostgreSQL backups

START HERE

`pg_dump <db> > db.sql`
Plain SQL dump

`pg_dump -Fc -f <file> <db>`
Compressed archive

`pg_dump -Fd -j <n> -f <dir> <db>`
Parallel dump

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

`pg_restore -d <db> <file>`
Restore an archive

`pg_restore -l <file>`
List archive contents

`pg_dump -s <db>`
Schema only

`pg_dumpall -g > globals.sql`
Roles and tablespaces

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

CONCEPTS

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

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
Restore it with psql, not pg_restore.

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

`pg_dump -Fc -f <file> <db>`
Custom-format archive
Compressed, and pg_restore can pick parts out of it.

`pg_dump -Fd -f <dir> <db>`
Directory-format archive
The directory must not exist yet.

`pg_dump -Fd -j <n> -f <dir> <db>`
Dump n tables in parallel
Only the directory format supports -j on dump.

`pg_dump -Ft -f <file> <db>`
Tar-format archive
Uncompressed; prefer -Fc or -Fd.

`pg_dump -Z zstd:<level> -Fc -f <file> <db>`
Choose compression method
Methods: gzip, lz4, zstd, none.

`pg_dump -v -Fc -f <file> <db>`
Show progress while dumping

CHOOSE WHAT TO DUMP

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

`pg_dump -a <db>`
Data only, no schema
Restore into a database that already has the tables.

`pg_dump -t <table> <db>`
Only matching tables
Quote mixed-case names: -t '"MyTab"'.

CHOOSE WHAT TO DUMP (cont.)

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

`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
*Patterns take * and ?, and a schema prefix.*

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

`pg_dump --filter=<file> <db>`
Read include/exclude rules from file
Lines like: include table mytab, exclude table logs.*

`pg_dump --section=pre-data <db>`
Only one section
Sections: pre-data, data, post-data.

`pg_dump --no-owner --no-privileges <db>`
Omit owners and GRANTs
Handy when restoring as a different user.

`pg_dump --inserts <db>`
INSERT statements instead of COPY
Slower to load, but portable to other databases.

RESTORE A DUMP

`psql -X -d <db> -f db.sql`
Restore a plain SQL dump
Add -v ON_ERROR_STOP=1 to stop at the first error.

`pg_restore -d <db> <file>`
Restore an archive into a database
The database must already exist unless you use -C.

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

`pg_restore -j <n> -d <db> <file>`
Restore with n parallel jobs
Needs a file or directory; a pipe will not do.

RESTORE A DUMP (cont.)

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

`pg_restore -C -d postgres <file>`
Create the database from the archive
-d only names the database used to connect.

`pg_restore -c --if-exists -d <db> <file>`
Drop objects before recreating
Overwrites what is there; --if-exists silences missing-object errors.

`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

`pg_restore -1 -d <db> <file>`
All or nothing in one transaction
Cannot be combined with -j.

`pg_restore -e -d <db> <file>`
Stop at the first error
By default it carries on and reports a count.

`pg_restore -l <file> > list.txt`
Save the contents list to edit
Comment out lines with ; to skip them.

`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
Use it to inspect a dump you do not trust.

`pg_restore --section=post-data -d <db> <file>`
Restore one section

`pg_restore --transaction-size=<n> -d <db> <file>`
Commit every n objects

`pg_restore --no-owner --role=<role> -d <db> <file>`
Restore with different ownership

WHOLE CLUSTER

`pg_dumpall > all.sql`
Every database, roles, tablespaces
Always plain SQL; restore with psql.

`pg_dumpall -g > globals.sql`
Roles and tablespaces only

`pg_dumpall -r > roles.sql`
Roles only

`psql -X -f all.sql postgres`
Restore a pg_dumpall script
Connect to postgres as a superuser.

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>
```

Copy schema first, data later

```
pg_dump -Fc -f db.dump <db>
pg_restore --section=pre-data -
d <new> db.dump
pg_restore --section=data -j 4
-d <new> db.dump
pg_restore --section=post-data
-j 4 -d <new> db.dump
```

Restore only a few objects

```
pg_restore -l db.dump >
list.txt
# comment out unwanted lines
with ;
pg_restore -L list.txt -d <db>
db.dump
```