https://www.linkedin.com/in/ahmed-mohamed-423583151

Day 2 Backup and Recovery Strategies

Backup and Restore in PostgreSQL

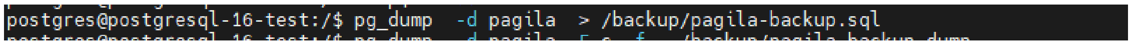
logical backup

logical backup consist of dumping sql command for whole database or table or schema
sql command consist of create and altering insert query that run on database for it current state when logical backup is taking
to take backup for database its pretty straight forward

for this section i have already deploy sample database pagila and will take backup of it

Taking DatabaseBackup

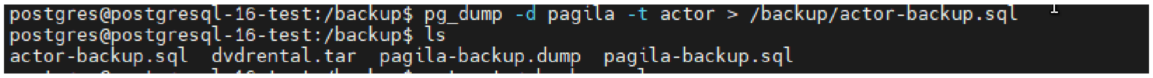
```
pg_dump  -d pagila  > /backup/pagila-backup.sql
```



```
postgres@postgresql-16-test:/$ pg_dump  -d pagila  > /backup/pagila-backup.sql
```

`-d` option to specify database you want the dump
`>` indication follow by location for storing pg_dump backup

```
pg_dump -d pagila -t actor > /backup/actor-backup.sql
```



```
postgres@postgresql-16-test:/backup$ pg_dump -d pagila -t actor > /backup/actor-backup.sql
postgres@postgresql-16-test:/backup$ ls
actor-backup.sql  dvdrental.tar  pagila-backup.dump  pagila-backup.sql
```

following command will take only backup for table you can mentation multilabel tables

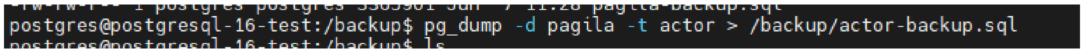
`-t` for specify table or multicable tables to take dump backup

Taking compress backup for pg_dump

pg_dump backup can get huge in size this way its good practices to compress backup

```
pg_dump  -d pagila -F c -f   /backup/pagila-backup.dump
```

`-F` used to mentioned format followed by `c` for compressed
`-f` used to specify the file name that you want to defined



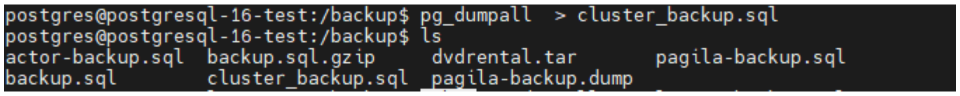
```
postgres@postgresql-16-test:/backup$ pg_dump -d pagila -t actor > /backup/actor-backup.sql
```

pg_dumpall

pg_dumpall is used to take full backup of the entire database cluster including roles and tablespaces
useful if you want to migrate you database to new server or just simply want to migrate it to upgraded version , you can restore pg_dumpall dump to any other PostgreSQL version

```
pg_dumpall  > cluster_backup.sql
```

you can use pg_dump and pg_dumpall to take backup from remote postgres you need to specify `-h` for host `-p` for pot if the remote host use port other than the default one `-U` for username



```
postgres@postgresql-16-test:/backup$ pg_dumpall  > cluster_backup.sql
postgres@postgresql-16-test:/backup$ ls
actor-backup.sql  backup.sql.gzip     dvdrental.tar       pagila-backup.sql
backup.sql        cluster_backup.sql  pagila-backup.dump
```

compressed backup

```
pg_dumpall  | gzip > dumpall.sql.gzip
```

Restore logical backup

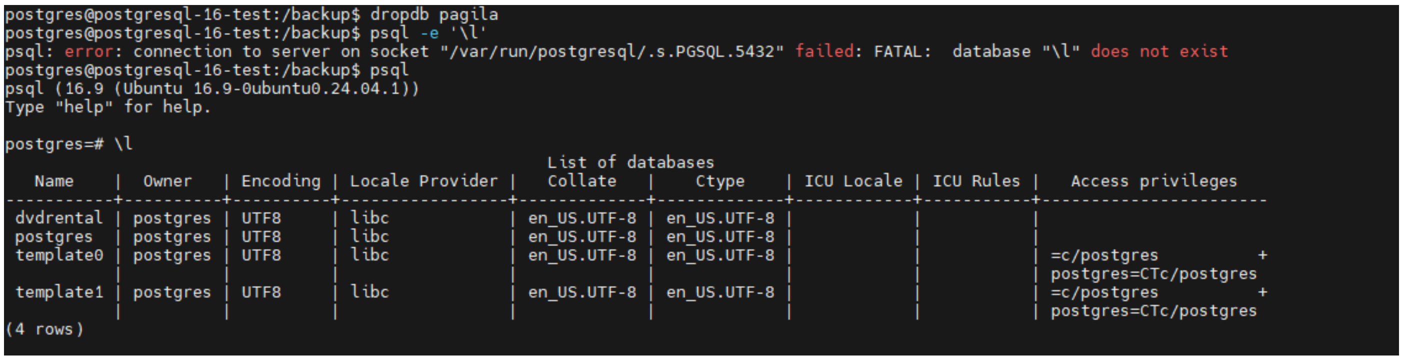
restore logical backup for both pg_dump and pg_dumpall is straight forward

for pg_dump i will drop database call pagila and recreate again then restore it with backup i have created
i will restore both the compresses one and normal dump

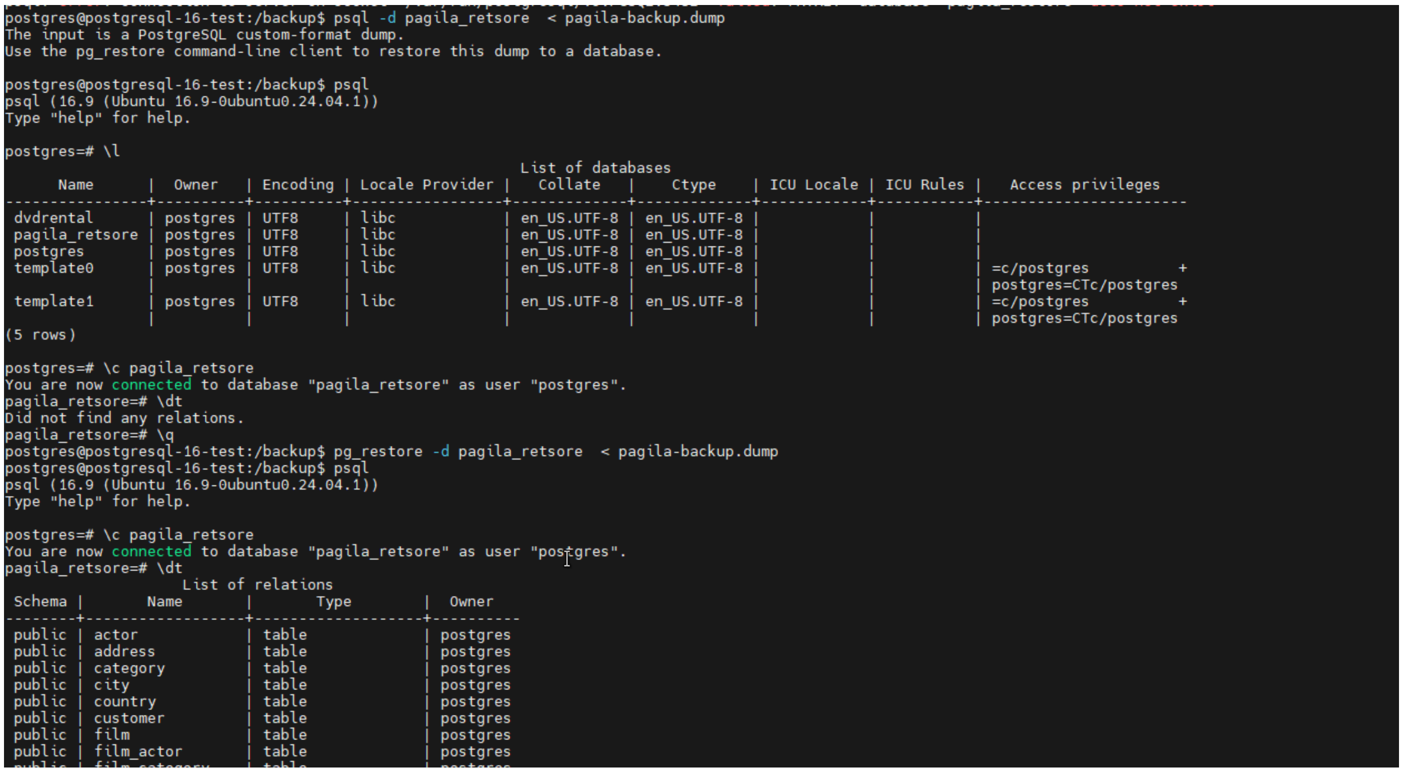
below command will restore dump compressed file , to restore compressed file best to use `pg_restore`

```
dropdb pagila
createdb pagila_retsore
pg_restore -d pagila_retsore  < pagila-backup.dump
```

the restoration is easy only you need to specify the database using option `-d` and make pointer `<` followed by backup path



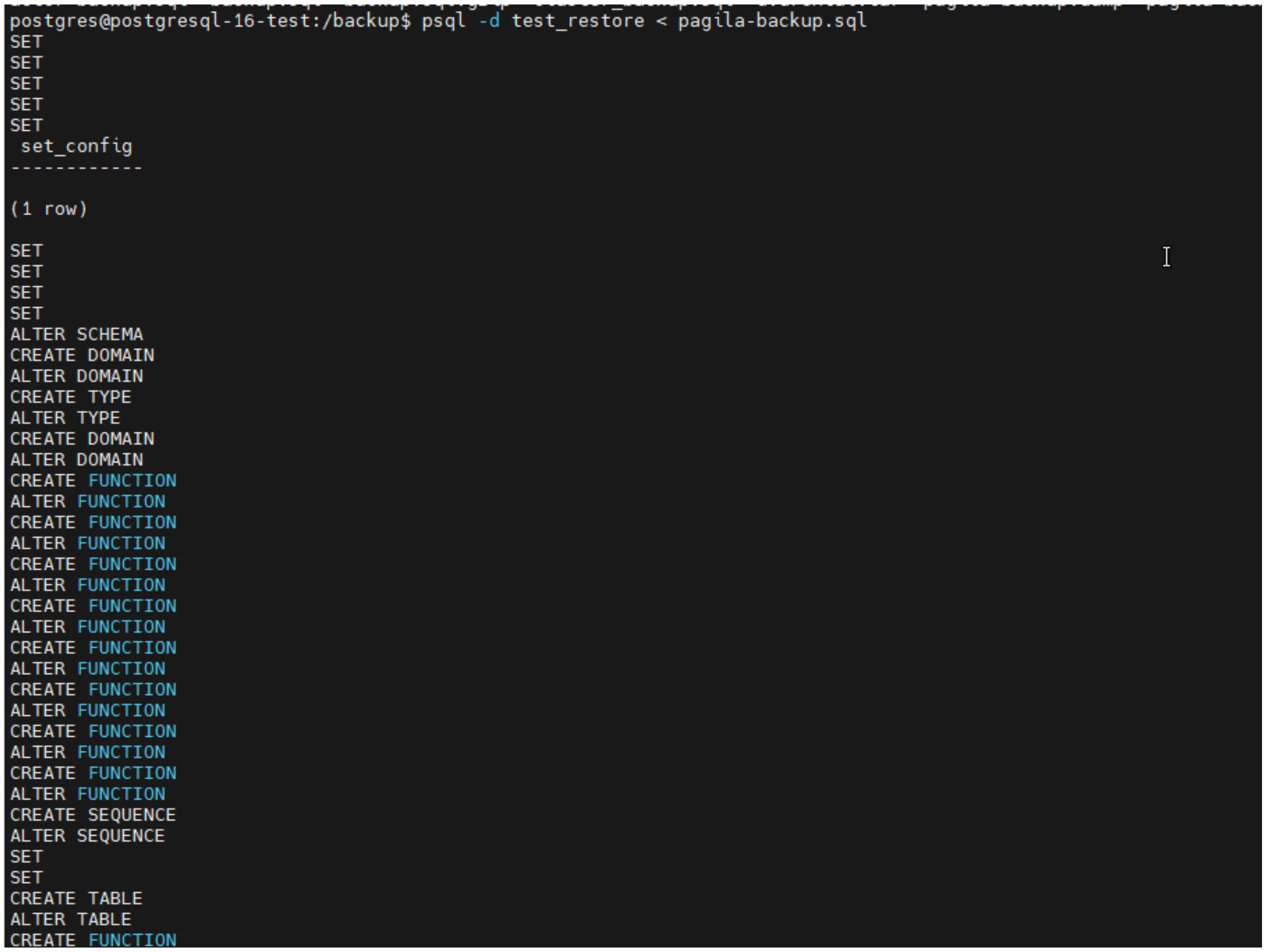
```
postgres@postgresql-16-test:/backup$ dropdb pagila
postgres@postgresql-16-test:/backup$ psql -e '\l'
psql: error: connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: FATAL:  database "\l" does not exist
postgres@postgresql-16-test:/backup$ psql
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))
Type "help" for help.

postgres=# \l
                                                     List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+-------------+-------------+------------+-----------+-----------------------
 dvdrental | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
(4 rows)
```



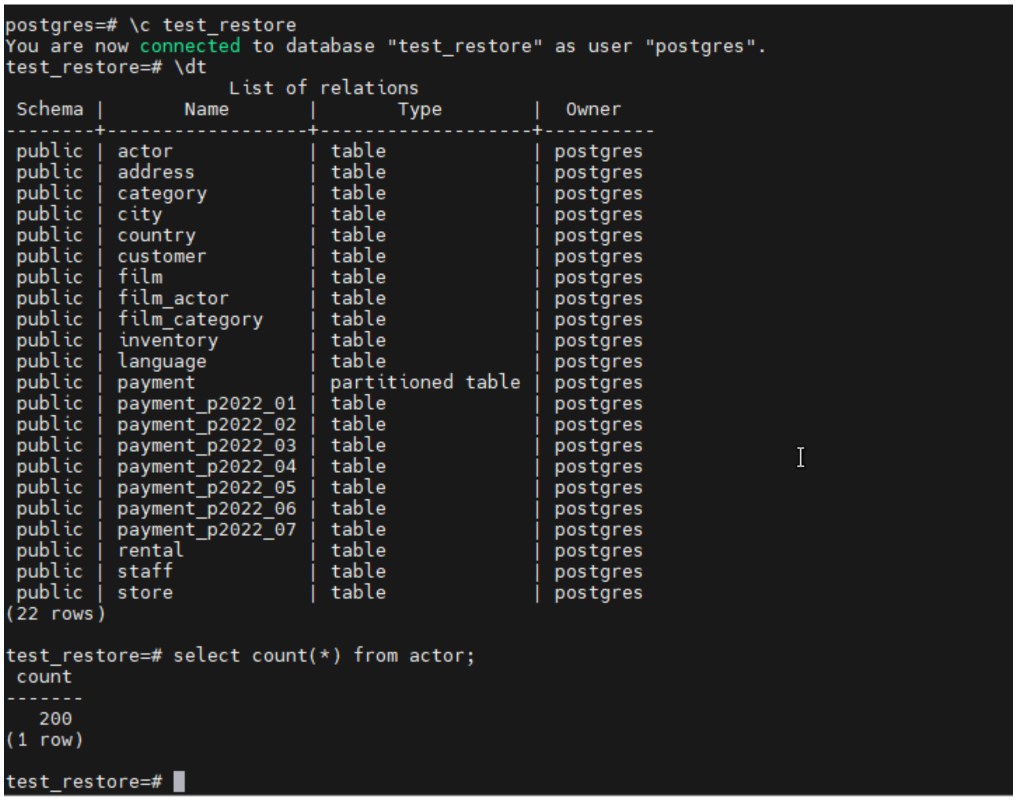
```
postgres@postgresql-16-test:/backup$ psql -d pagila_retsore  < pagila-backup.dump
The input is a PostgreSQL custom-format dump.
Use the pg_restore command-line client to restore this dump to a database.

postgres@postgresql-16-test:/backup$ psql
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))
Type "help" for help.

postgres=# \l
                                                       List of databases
      Name      |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access privileges
----------------+----------+----------+-----------------+-------------+-------------+------------+-----------+-----------------------
 dvdrental      | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 pagila_retsore | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 postgres       | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0      | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
                |          |          |                 |             |             |            |           | postgres=CTc/postgres
 template1      | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
                |          |          |                 |             |             |            |           | postgres=CTc/postgres
(5 rows)

postgres=# \c pagila_retsore
You are now connected to database "pagila_retsore" as user "postgres".
pagila_retsore=# \dt
Did not find any relations.
pagila_retsore=# \q
postgres@postgresql-16-test:/backup$ pg_restore -d pagila_retsore  < pagila-backup.dump
postgres@postgresql-16-test:/backup$ psql
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))
Type "help" for help.

postgres=# \c pagila_retsore
You are now connected to database "pagila_retsore" as user "postgres".
pagila_retsore=# \dt
                   List of relations
 Schema |       Name       |       Type        |  Owner
--------+------------------+-------------------+----------
 public | actor            | table             | postgres
 public | address          | table             | postgres
 public | category         | table             | postgres
 public | city             | table             | postgres
 public | country          | table             | postgres
 public | customer         | table             | postgres
 public | film             | table             | postgres
 public | film_actor       | table             | postgres
```

below command will restore normal dump with `.sql` extension for that you use `psql` for restore for that i will create new database and restore from normal uncompressed dump

```
createdb test_restore

psql -d test_restore < pagila-backup.sql
```



```
postgres@postgresql-16-test:/backup$ psql -d test_restore < pagila-backup.sql
SET
SET
SET
SET
SET
 set_config
------------

(1 row)

SET
SET
SET
SET
ALTER SCHEMA
CREATE DOMAIN
ALTER DOMAIN
CREATE TYPE
ALTER TYPE
CREATE DOMAIN
ALTER DOMAIN
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE FUNCTION
ALTER FUNCTION
CREATE SEQUENCE
ALTER SEQUENCE
SET
SET
CREATE TABLE
ALTER TABLE
CREATE FUNCTION
```



```
postgres=# \c test_restore
You are now connected to database "test_restore" as user "postgres".
test_restore=# \dt
                   List of relations
 Schema |       Name        |       Type        |  Owner
--------+-------------------+-------------------+----------
 public | actor             | table             | postgres
 public | address           | table             | postgres
 public | category          | table             | postgres
 public | city              | table             | postgres
 public | country           | table             | postgres
 public | customer          | table             | postgres
 public | film              | table             | postgres
 public | film_actor        | table             | postgres
 public | film_category     | table             | postgres
 public | inventory         | table             | postgres
 public | language          | table             | postgres
 public | payment           | partitioned table | postgres
 public | payment_p2022_01 | table             | postgres
 public | payment_p2022_02 | table             | postgres
 public | payment_p2022_03 | table             | postgres
 public | payment_p2022_04 | table             | postgres
 public | payment_p2022_05 | table             | postgres
 public | payment_p2022_06 | table             | postgres
 public | payment_p2022_07 | table             | postgres
 public | rental            | table             | postgres
 public | staff             | table             | postgres
 public | store             | table             | postgres
(22 rows)

test_restore=# select count(*) from actor;
 count
-------
   200
(1 row)

test_restore=#
```

Restore pg_dumpall

```
psql -f cluster_backup.sql
```

`-f` option to specify the pg_dumpall file name

For backups created in custom (-F c), directory (-F d), or tar (-F t) formats, use the pg_restore utility

Limitations of Logical Backups

- Performance Impact: Creating logical backups can be resource-intensive and may
- impact database performance.
- Restoration Time: Restoring from logical backups can be slower than physical backups, especially for large databases, as objects and indexes need to be recreated.
- No Point-in-Time Recovery: Logical backups alone don't support point-in-time recovery; they represent the database state at the time of backup.
  Best Practices for Logical Backups
- Regular Scheduling: Schedule regular backups based on your RPO requirements.
- Compression: Use compression to reduce storage requirements, especially for large databases.
- Backup Verification: Regularly verify the integrity of your backups by performing test restores.
- Backup Rotation: Implement a backup rotation strategy to maintain multiple backup copies while managing storage usage.
- Documentation: Document your backup procedures and ensure multiple team members know how to perform backups and restores.

Physical Backups with pg_basebackup

the recommend and most best practices way to take backup for production PostgreSQL is to use physical backup solution such as pg_basebackup or pg_backrest backup , it consider online backup meaning has very low impact on data base performance during initiation of backup , its support point in time recover and restoring is mush faster

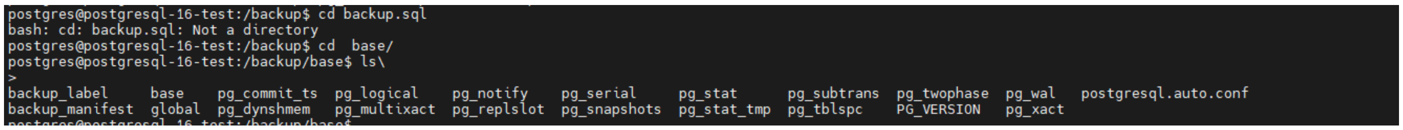
only drawback is for solution such as pg_basebackup you cannot take backup for single schema or single database pg_bsebackup will take full data dir backup

to take backup with point in time recovery we need to enable wal archiving

Normal backup

```
  pg_basebackup   -D /backup/base/
```

the directory where you specify using `-D` must be empty



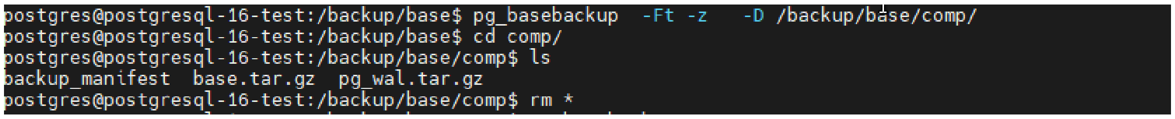
```
postgres@postgresql-16-test:/backup$ cd backup.sql
bash: cd: backup.sql: Not a directory
postgres@postgresql-16-test:/backup$ cd  base/
postgres@postgresql-16-test:/backup/base$ ls\
>
backup_label     base    pg_commit_ts  pg_logical    pg_notify    pg_serial     pg_stat      pg_subtrans  pg_twophase  pg_wal   postgresql.auto.conf
backup_manifest  global  pg_dynshmem   pg_multixact  pg_replslot  pg_snapshots  pg_stat_tmp  pg_tblspc    PG_VERSION   pg_xact
```

pg_basebackup from you can see take full data dir copy while database operational

compressed and wall stream backup

```
pg_basebackup  -Ft -z   -D /backup/base/
```

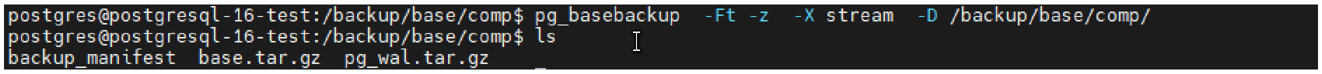
`-Ft` specify file format
`-Z` the file format will be `gzip` compressed



```
postgres@postgresql-16-test:/backup/base$ pg_basebackup  -Ft -z   -D /backup/base/comp/
postgres@postgresql-16-test:/backup/base$ cd comp/
postgres@postgresql-16-test:/backup/base/comp$ ls
backup_manifest  base.tar.gz  pg_wal.tar.gz
postgres@postgresql-16-test:/backup/base/comp$ rm *
```

compressed with wal stream

```
pg_basebackup  -Ft -z  -X stream  -D /backup/base/comp/
```



```
postgres@postgresql-16-test:/backup/base/comp$ pg_basebackup  -Ft -z  -X stream  -D /backup/base/comp/
postgres@postgresql-16-test:/backup/base/comp$ ls
backup_manifest  base.tar.gz  pg_wal.tar.gz
```

for this option to wok you need to update parameter in `postgresql.conf` file as follow

```
max_wal_senders = 10 # at least 2 is recommended
wal_level = replica # or logical
```

why you need to stream wall during backup

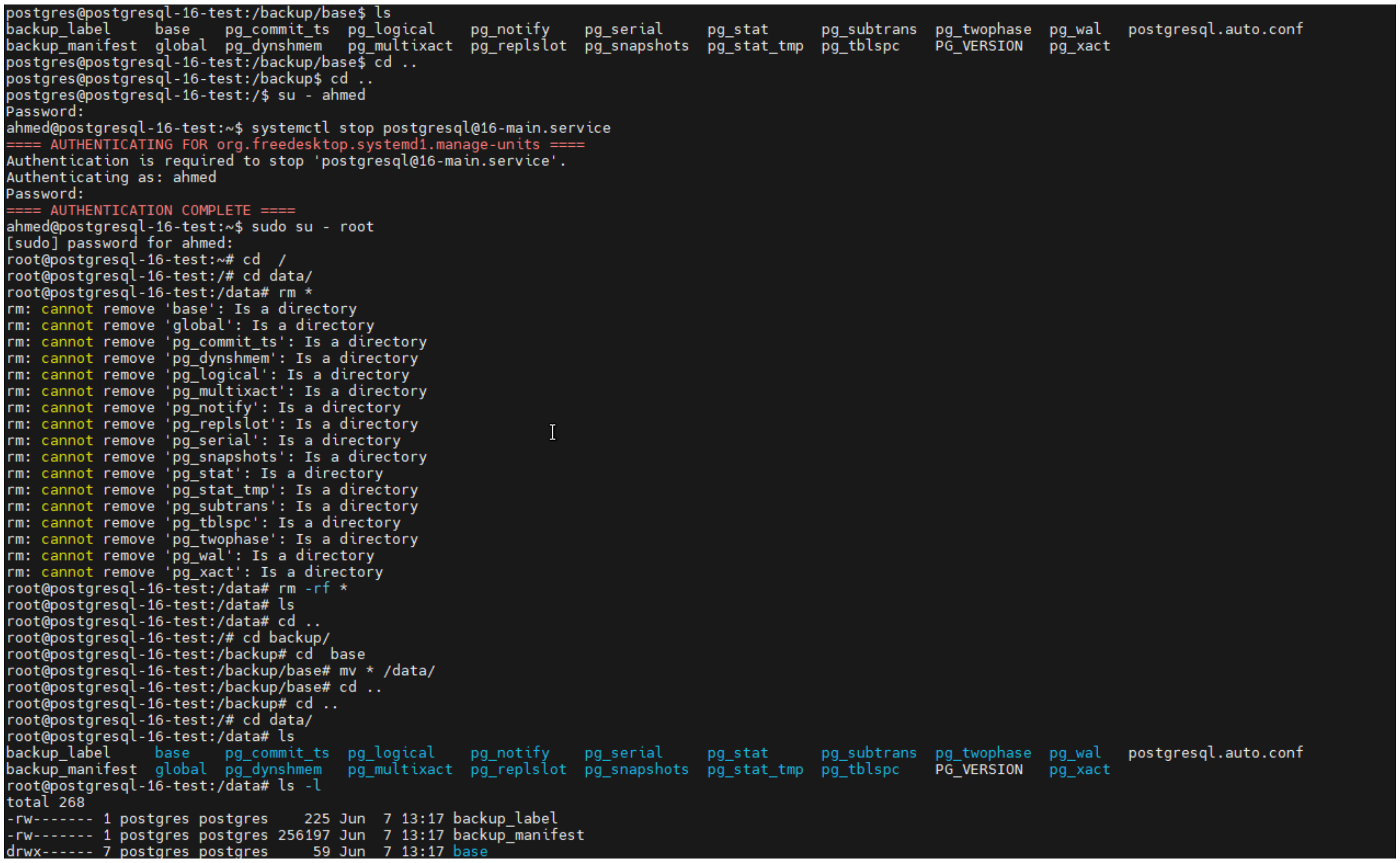
- pg_basebackup actually does by default `-X fetch` method, these WAL files pile up on your main server's disk and are only copied at the very end.
- When you take a backup of a large database, the process can take a long time (minutes or even hours). During this entire time, your database is still running and generating transaction logs (WAL files).
- By streaming the WAL files *as they are created*, you prevent this dangerous buildup. The backup process consumes the logs in real-time, putting no extra disk pressure on your primary server.

Restore pg_basebackup backup

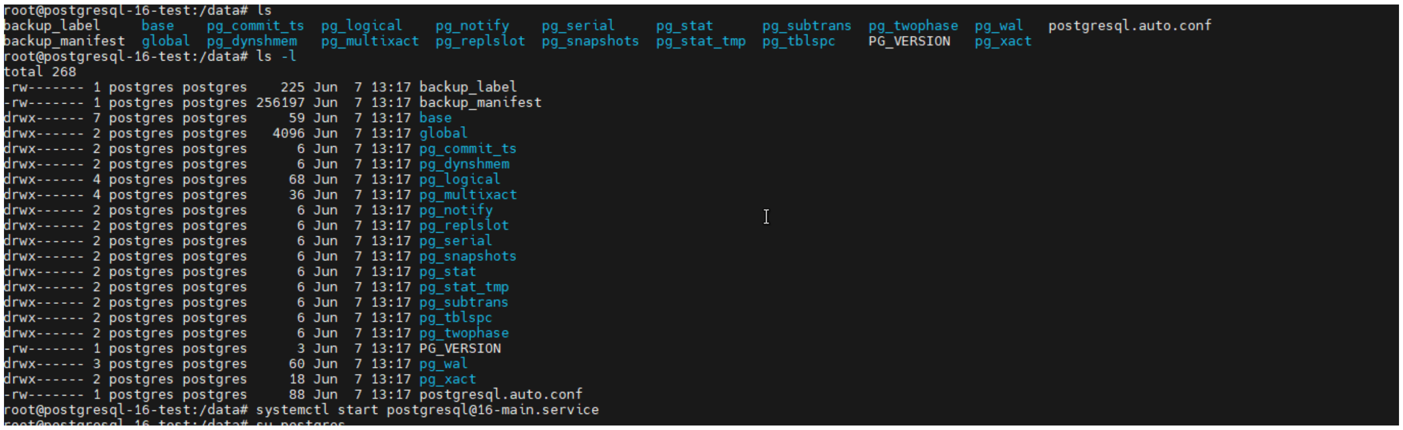
first to restore backup you need to disable PostgreSQL services and remove the data directory content and then copy backup content and ensure permission then start the PostgreSQL services

```
systemctl stop postgresql@16-main.service
cd  data # go to data dirtcory

rm -rf *  # remove all conent

cd  /backup/base # go to backup diretcory for basebackup
mv *  /data/ move the files to data diretcory

#confrim that file permisison are correct
systemctl start postgresql@16-main.service
```

https://www.linkedin.com/in/ahmed-mohamed-423583151



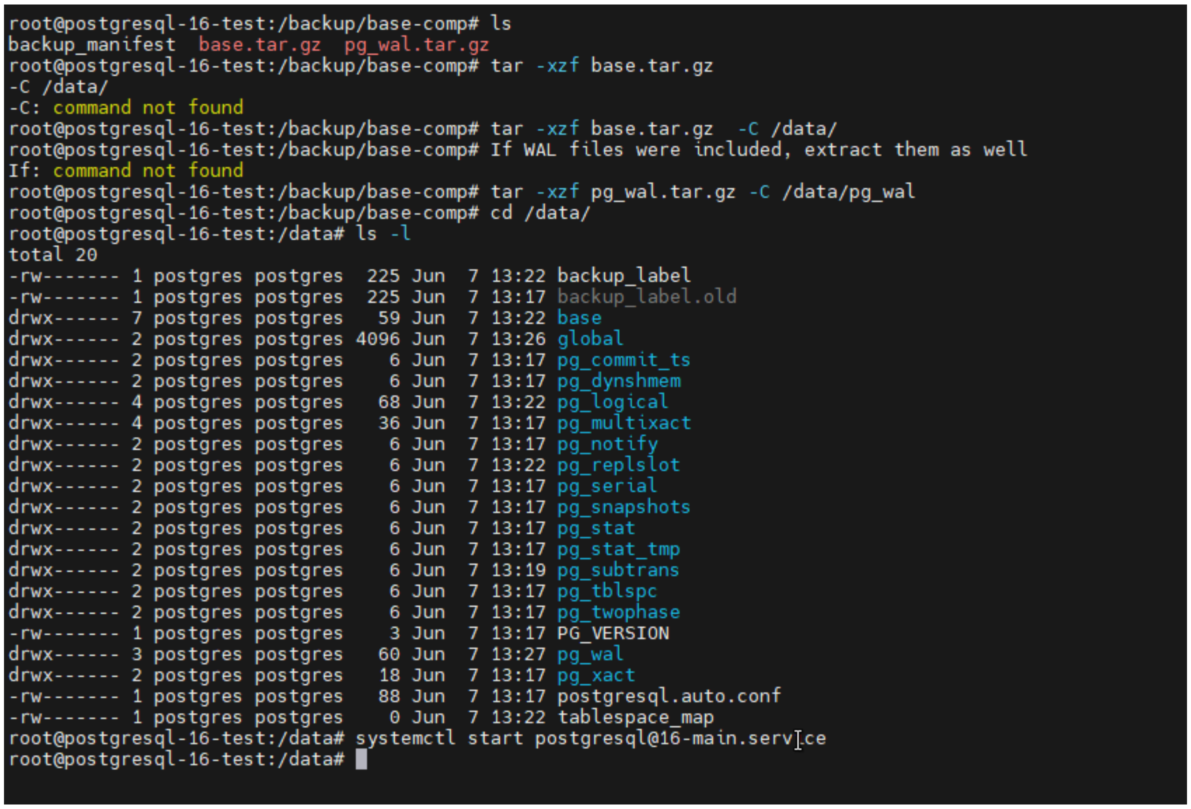
```
postgres@postgresql-16-test:/backup/base$ ls
backup_label     base    pg_commit_ts  pg_logical    pg_notify    pg_serial     pg_stat       pg_subtrans  pg_twophase  pg_wal    postgresql.auto.conf
backup_manifest  global  pg_dynshmem   pg_multixact  pg_replslot  pg_snapshots  pg_stat_tmp   pg_tblspc    PG_VERSION   pg_xact
postgres@postgresql-16-test:/backup/base$ cd ..
postgres@postgresql-16-test:/backup$ cd ..
postgres@postgresql-16-test:/$ su - ahmed
Password:
ahmed@postgresql-16-test:~$ systemctl stop postgresql@16-main.service
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to stop 'postgresql@16-main.service'.
Authenticating as: ahmed
Password:
==== AUTHENTICATION COMPLETE ====
ahmed@postgresql-16-test:~$ sudo su - root
[sudo] password for ahmed:
root@postgresql-16-test:~# cd  /
root@postgresql-16-test:/# cd data/
root@postgresql-16-test:/data# rm *
rm: cannot remove 'base': Is a directory
rm: cannot remove 'global': Is a directory
rm: cannot remove 'pg_commit_ts': Is a directory
rm: cannot remove 'pg_dynshmem': Is a directory
rm: cannot remove 'pg_logical': Is a directory
rm: cannot remove 'pg_multixact': Is a directory
rm: cannot remove 'pg_notify': Is a directory
rm: cannot remove 'pg_replslot': Is a directory
rm: cannot remove 'pg_serial': Is a directory
rm: cannot remove 'pg_snapshots': Is a directory
rm: cannot remove 'pg_stat': Is a directory
rm: cannot remove 'pg_stat_tmp': Is a directory
rm: cannot remove 'pg_subtrans': Is a directory
rm: cannot remove 'pg_tblspc': Is a directory
rm: cannot remove 'pg_twophase': Is a directory
rm: cannot remove 'pg_wal': Is a directory
rm: cannot remove 'pg_xact': Is a directory
root@postgresql-16-test:/data# rm -rf *
root@postgresql-16-test:/data# ls
root@postgresql-16-test:/data# cd ..
root@postgresql-16-test:/# cd backup/
root@postgresql-16-test:/backup# cd  base
root@postgresql-16-test:/backup/base# mv * /data/
root@postgresql-16-test:/backup/base# cd ..
root@postgresql-16-test:/backup# cd ..
root@postgresql-16-test:/# cd data/
root@postgresql-16-test:/data# ls
backup_label     base    pg_commit_ts  pg_logical    pg_notify    pg_serial     pg_stat       pg_subtrans  pg_twophase  pg_wal    postgresql.auto.conf
backup_manifest  global  pg_dynshmem   pg_multixact  pg_replslot  pg_snapshots  pg_stat_tmp   pg_tblspc    PG_VERSION   pg_xact
root@postgresql-16-test:/data# ls -l
total 268
-rw------- 1 postgres postgres    225 Jun  7 13:17 backup_label
-rw------- 1 postgres postgres 256197 Jun  7 13:17 backup_manifest
drwx------ 7 postgres postgres     59 Jun  7 13:17 base
```



```
root@postgresql-16-test:/data# ls
backup_label     base    pg_commit_ts  pg_logical    pg_notify    pg_serial     pg_stat       pg_subtrans  pg_twophase  pg_wal    postgresql.auto.conf
backup_manifest  global  pg_dynshmem   pg_multixact  pg_replslot  pg_snapshots  pg_stat_tmp   pg_tblspc    PG_VERSION   pg_xact
root@postgresql-16-test:/data# ls -l
total 268
-rw------- 1 postgres postgres    225 Jun  7 13:17 backup_label
-rw------- 1 postgres postgres 256197 Jun  7 13:17 backup_manifest
drwx------ 7 postgres postgres     59 Jun  7 13:17 base
drwx------ 2 postgres postgres   4096 Jun  7 13:17 global
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_commit_ts
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_dynshmem
drwx------ 4 postgres postgres     68 Jun  7 13:17 pg_logical
drwx------ 4 postgres postgres     36 Jun  7 13:17 pg_multixact
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_notify
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_replslot
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_serial
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_snapshots
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_stat
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_stat_tmp
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_subtrans
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_tblspc
drwx------ 2 postgres postgres      6 Jun  7 13:17 pg_twophase
-rw------- 1 postgres postgres      3 Jun  7 13:17 PG_VERSION
drwx------ 3 postgres postgres     60 Jun  7 13:17 pg_wal
drwx------ 2 postgres postgres     18 Jun  7 13:17 pg_xact
-rw------- 1 postgres postgres     88 Jun  7 13:17 postgresql.auto.conf
root@postgresql-16-test:/data# systemctl start postgresql@16-main.service
```

Restore compressed backup with wall

```
systemctl stop postgresql@16-main.service
cd  data # go to data dirtcory

rm -rf *  # remove all conent

cd  /backup/base-comp  # go to backup diretcory for basebackup
sudo tar -xzf base.tar.gz  -C /data/
# If WAL files were included, extract them as well
tar -xzf pg_wal.tar.gz -C /data/pg_wal
#confrim that file permisison are correct
systemctl start postgresql@16-main.service
```



```
root@postgresql-16-test:/backup/base-comp# ls
backup_manifest  base.tar.gz  pg_wal.tar.gz
root@postgresql-16-test:/backup/base-comp# tar -xzf base.tar.gz
-C /data/
-C: command not found
root@postgresql-16-test:/backup/base-comp# tar -xzf base.tar.gz  -C /data/
root@postgresql-16-test:/backup/base-comp# If WAL files were included, extract them as well
If: command not found
root@postgresql-16-test:/backup/base-comp# tar -xzf pg_wal.tar.gz -C /data/pg_wal
root@postgresql-16-test:/backup/base-comp# cd /data/
root@postgresql-16-test:/data# ls -l
total 20
-rw------- 1 postgres postgres  225 Jun  7 13:22 backup_label
-rw------- 1 postgres postgres  225 Jun  7 13:17 backup_label.old
drwx------ 7 postgres postgres   59 Jun  7 13:22 base
drwx------ 2 postgres postgres 4096 Jun  7 13:26 global
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_commit_ts
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_dynshmem
drwx------ 4 postgres postgres   68 Jun  7 13:22 pg_logical
drwx------ 4 postgres postgres   36 Jun  7 13:17 pg_multixact
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_notify
drwx------ 2 postgres postgres    6 Jun  7 13:22 pg_replslot
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_serial
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_snapshots
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_stat
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_stat_tmp
drwx------ 2 postgres postgres    6 Jun  7 13:19 pg_subtrans
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_tblspc
drwx------ 2 postgres postgres    6 Jun  7 13:17 pg_twophase
-rw------- 1 postgres postgres    3 Jun  7 13:17 PG_VERSION
drwx------ 3 postgres postgres   60 Jun  7 13:27 pg_wal
drwx------ 2 postgres postgres   18 Jun  7 13:17 pg_xact
-rw------- 1 postgres postgres   88 Jun  7 13:17 postgresql.auto.conf
-rw------- 1 postgres postgres    0 Jun  7 13:22 tablespace_map
root@postgresql-16-test:/data# systemctl start postgresql@16-main.service
root@postgresql-16-test:/data#
```

Advantages of Physical Backups

- Faster Restoration: Physical backups can be restored more quickly than logical backups, especially for large databases.
- Complete Backup: All databases and global objects are backed up together.
- Point-in-Time Recovery: When combined with WAL archiving, physical backups enable point-in-time recovery.
- Lower Impact: Physical backups can have less impact on database performance than logical backups.
  Limitations of Physical Backups
- Version Dependency: Physical backups are version-specific and cannot be used for upgrading to newer PostgreSQL versions.
- All or Nothing: You cannot selectively backup or restore specific databases or tables.
- Platform Dependency: Physical backups may not be portable across different operating systems or architectures.

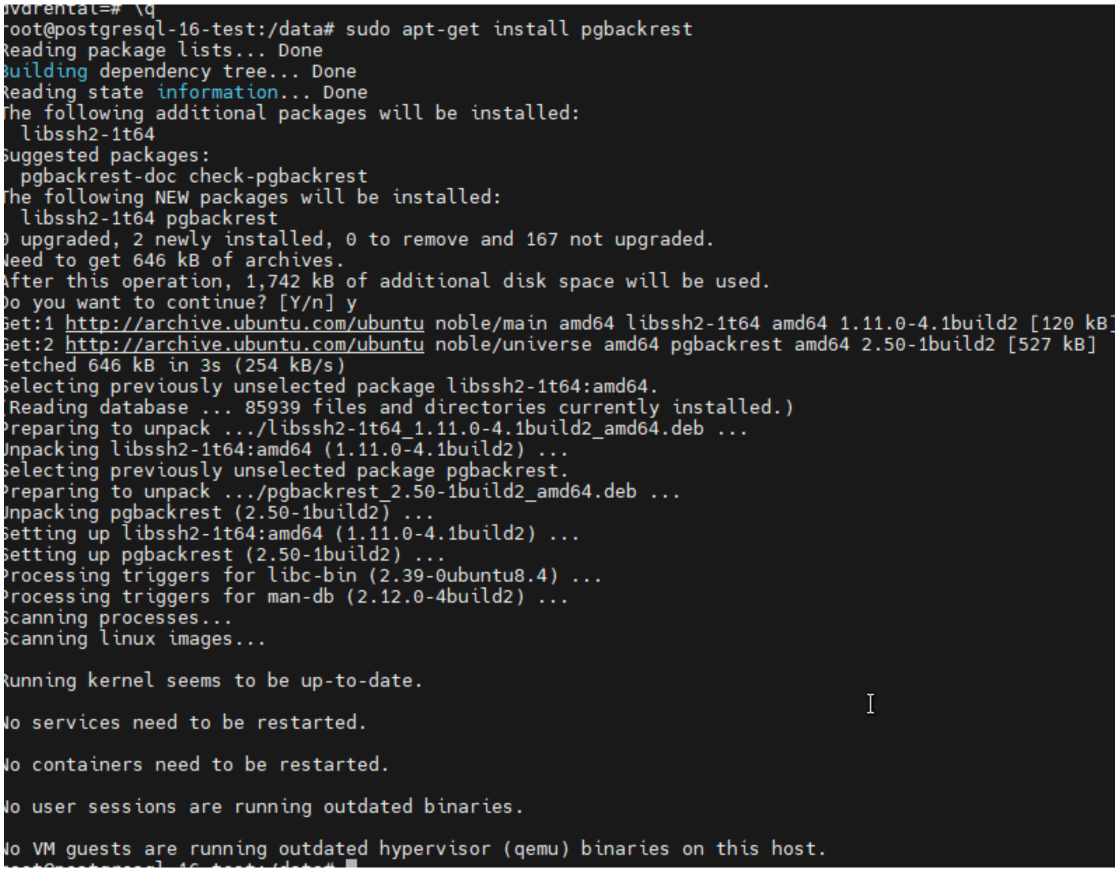
Advanced Backup Management with pgBackRest

this actually my prefer way of backing up PostgreSQL , it mush advanced with options such as incremental backup , you can defend the retention for backup and its more automated , you can also setup dedicated server for taking backup for multilabel PostgreSQL and you can also move the backup to cloud storage such as S3

Installing pgBackRest

i personally recommended that you install pgBackRest on sperate server and run the backup form there but its not bad idea to run pgBackRest on same PostgreSQL vm

```
sudo apt-get install pgbackrest
```



```
vdrental=# \q
oot@postgresql-16-test:/data# sudo apt-get install pgbackrest
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
he following additional packages will be installed:
 libssh2-1t64
uggested packages:
 pgbackrest-doc check-pgbackrest
he following NEW packages will be installed:
 libssh2-1t64 pgbackrest
 upgraded, 2 newly installed, 0 to remove and 167 not upgraded.
eed to get 646 kB of archives.
fter this operation, 1,742 kB of additional disk space will be used.
o you want to continue? [Y/n] y
et:1 http://archive.ubuntu.com/ubuntu noble/main amd64 libssh2-1t64 amd64 1.11.0-4.1build2 [120 kB]
et:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 pgbackrest amd64 2.50-1build2 [527 kB]
etched 646 kB in 3s (254 kB/s)
electing previously unselected package libssh2-1t64:amd64.
(Reading database ... 85939 files and directories currently installed.)
reparing to unpack .../libssh2-1t64_1.11.0-4.1build2_amd64.deb ...
npacking libssh2-1t64:amd64 (1.11.0-4.1build2) ...
electing previously unselected package pgbackrest.
reparing to unpack .../pgbackrest_2.50-1build2_amd64.deb ...
npacking pgbackrest (2.50-1build2) ...
etting up libssh2-1t64:amd64 (1.11.0-4.1build2) ...
etting up pgbackrest (2.50-1build2) ...
rocessing triggers for libc-bin (2.39-0ubuntu8.4) ...
rocessing triggers for man-db (2.12.0-4build2) ...
canning processes...
canning linux images...

unning kernel seems to be up-to-date.

o services need to be restarted.

o containers need to be restarted.

o user sessions are running outdated binaries.

o VM guests are running outdated hypervisor (qemu) binaries on this host.
```

go to `postgresql.conf` and update the below parameter and restart the services

```
#------------------------------------------------------------------------------
# WRITE-AHEAD LOG / Archiving
#------------------------------------------------------------------------------

# This is the main setting to fix the error.
archive_mode = on

# This tells PostgreSQL HOW to archive. pgBackRest provides its own command.
# Make sure 'demo' matches your --stanza name.
archive_command = 'pgbackrest --stanza=demo archive-push %p'

#------------------------------------------------------------------------------
# WRITE-AHEAD LOG / Settings
#------------------------------------------------------------------------------

# 'replica' is the minimum level required for pgBackRest backups.
wal_level = replica

#------------------------------------------------------------------------------
# REPLICATION / Sending Servers
#------------------------------------------------------------------------------

# pgBackRest needs at least one sender for differential/incremental backups.
# A value like 10 is a safe default.
max_wal_senders = 10
```

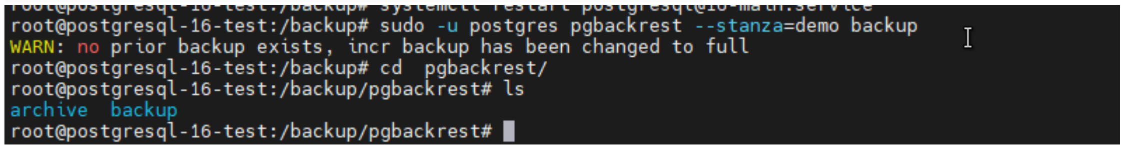
open the `/etc/pgbackrest.conf' and update backup location retention for backup

```
{global]
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
```

```
[demo]
pg1-path=/var/lib/postgresql/14/main
```

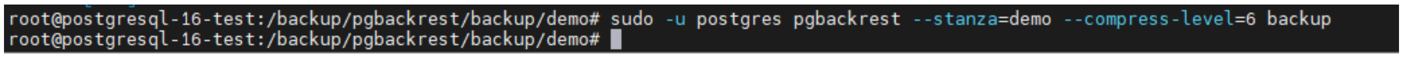
In this configuration: - [global] section defines global settings. - repo1-path specifies the location where backups will be stored. - repo1-retention-full specifies how many full backups to retain. - [demo] is a stanza name (a logical grouping of PostgreSQL databases). - pg1-path specifies the PostgreSQL data directory.

Creating a pgBackRest Stanza
Before performing backups, you need to create a stanza:
`sudo -u postgres pgbackrest --stanza=demo stanza-create`
Creating a full backup:
`sudo -u postgres pgbackrest --stanza=demo backup`

Creating a differential backup:
`sudo -u postgres pgbackrest --stanza=demo --type=diff backup`


```
root@postgresql-16-test:/backup# sudo -u postgres pgbackrest --stanza=demo backup
WARN: no prior backup exists, incr backup has been changed to full
root@postgresql-16-test:/backup# cd  pgbackrest/
root@postgresql-16-test:/backup/pgbackrest# ls
archive  backup
root@postgresql-16-test:/backup/pgbackrest#
```

Creating an incremental backup:
`sudo -u postgres pgbackrest --stanza=demo --type=incr backup`
Creating a backup with compression:
`sudo -u postgres pgbackrest --stanza=demo --compress-level=6 backup`



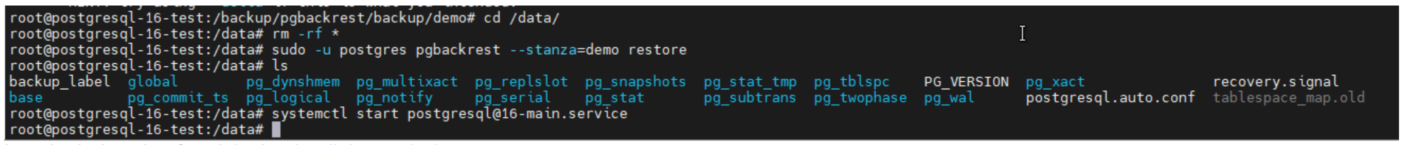
```
root@postgresql-16-test:/backup/pgbackrest/backup/demo# sudo -u postgres pgbackrest --stanza=demo --compress-level=6 backup
root@postgresql-16-test:/backup/pgbackrest/backup/demo#
```

Restoring with pgBackRest

to retsore backup is straight forward

first stop postgresql and then run the command to restore the latest backup

```
systemctl stop postgresql@16-main.service
rm -rf * # remove content of data dir
sudo -u postgres pgbackrest --stanza=demo restore
systemctl start  postgresql@16-main.service
```



```
root@postgresql-16-test:/backup/pgbackrest/backup/demo# cd /data/
root@postgresql-16-test:/data# rm -rf *
root@postgresql-16-test:/data# sudo -u postgres pgbackrest --stanza=demo restore
root@postgresql-16-test:/data# ls
backup_label  global        pg_dynshmem  pg_multixact  pg_replslot  pg_snapshots  pg_stat_tmp  pg_tblspc    PG_VERSION  pg_xact               recovery.signal
base          pg_commit_ts  pg_logical   pg_notify     pg_serial    pg_stat       pg_subtrans  pg_twophase  pg_wal      postgresql.auto.conf  tablespace_map.old
root@postgresql-16-test:/data# systemctl start postgresql@16-main.service
root@postgresql-16-test:/data#
```

What it Does: `pgBackRest` will automatically:

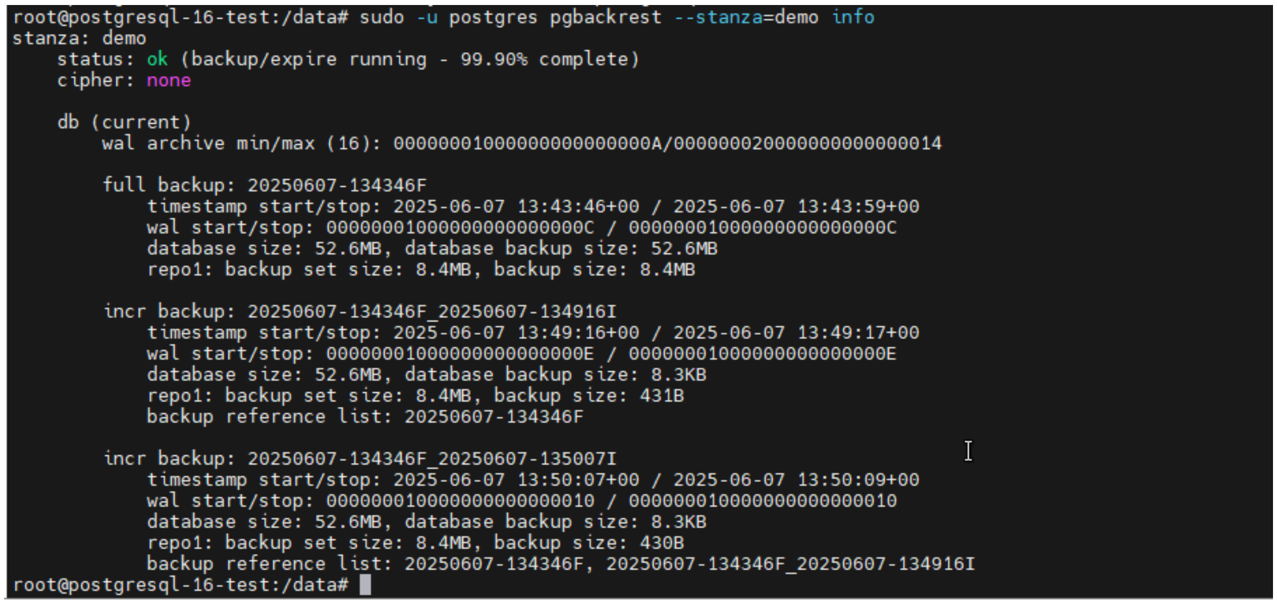
- Find the latest full backup.
- Apply any differential or incremental backups that were taken after it.
- Replay all the archived WAL files to bring the database to the most recent possible state.

restore certain full backup

for that you will first check what are backup there in pgbackrest usng the following command

```
sudo -u postgres pgbackrest --stanza=demo info
```

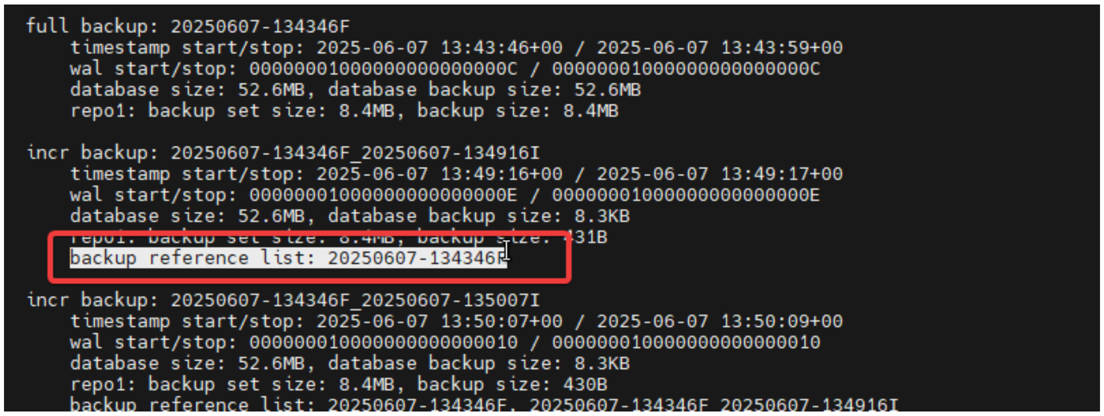
https://www.linkedin.com/in/ahmed-mohamed-423583151



```
root@postgresql-16-test:/data# sudo -u postgres pgbackrest --stanza=demo info
stanza: demo
    status: ok (backup/expire running - 99.90% complete)
    cipher: none

    db (current)
        wal archive min/max (16): 00000001000000000000000A/000000020000000000000014

        full backup: 20250607-134346F
            timestamp start/stop: 2025-06-07 13:43:46+00 / 2025-06-07 13:43:59+00
            wal start/stop: 00000001000000000000000C / 00000001000000000000000C
            database size: 52.6MB, database backup size: 52.6MB
            repo1: backup set size: 8.4MB, backup size: 8.4MB

        incr backup: 20250607-134346F_20250607-134916I
            timestamp start/stop: 2025-06-07 13:49:16+00 / 2025-06-07 13:49:17+00
            wal start/stop: 00000001000000000000000E / 00000001000000000000000E
            database size: 52.6MB, database backup size: 8.3KB
            repo1: backup set size: 8.4MB, backup size: 431B
            backup reference list: 20250607-134346F

        incr backup: 20250607-134346F_20250607-135007I
            timestamp start/stop: 2025-06-07 13:50:07+00 / 2025-06-07 13:50:09+00
            wal start/stop: 000000010000000000000010 / 000000010000000000000010
            database size: 52.6MB, database backup size: 8.3KB
            repo1: backup set size: 8.4MB, backup size: 430B
            backup reference list: 20250607-134346F, 20250607-134346F_20250607-134916I
root@postgresql-16-test:/data#
```

The output will show you a list of your full and incremental backups with their labels (e.g., `20250607-161005F` for a full backup).

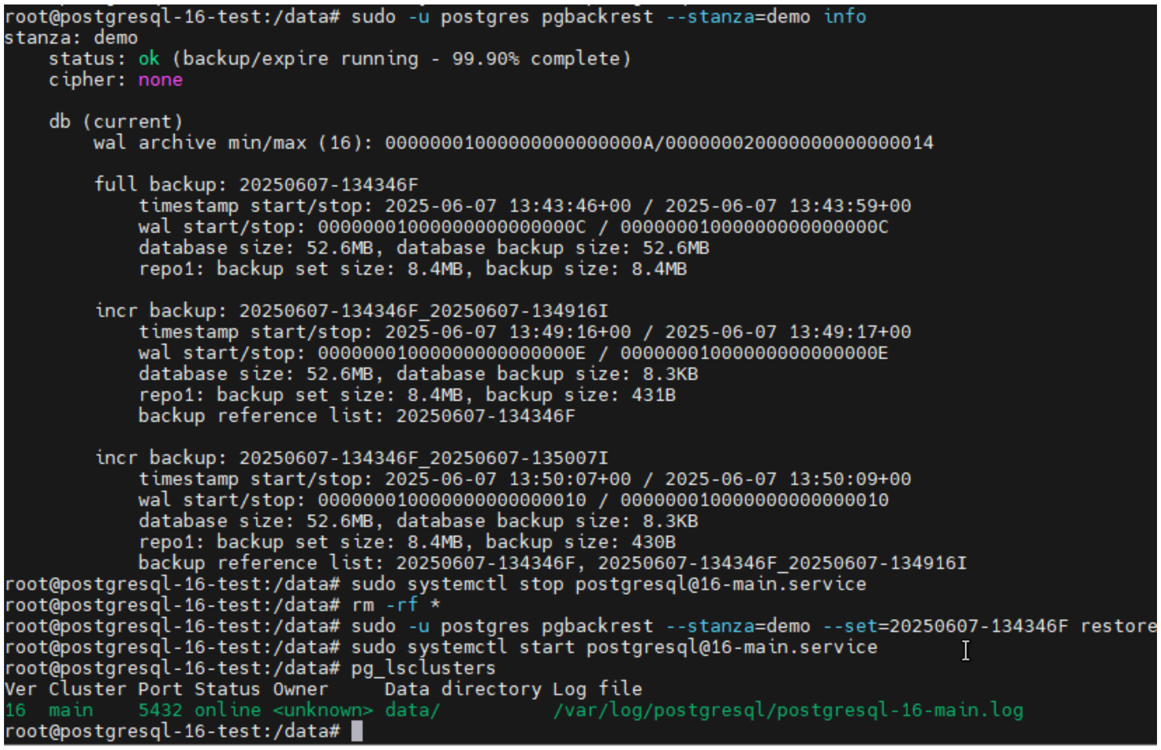
we will restore backup of first incremental using backup reference list



```
full backup: 20250607-134346F
    timestamp start/stop: 2025-06-07 13:43:46+00 / 2025-06-07 13:43:59+00
    wal start/stop: 00000001000000000000000C / 00000001000000000000000C
    database size: 52.6MB, database backup size: 52.6MB
    repo1: backup set size: 8.4MB, backup size: 8.4MB

incr backup: 20250607-134346F_20250607-134916I
    timestamp start/stop: 2025-06-07 13:49:16+00 / 2025-06-07 13:49:17+00
    wal start/stop: 00000001000000000000000E / 00000001000000000000000E
    database size: 52.6MB, database backup size: 8.3KB
    repo1: backup set size: 8.4MB, backup size: 431B
    backup reference list: 20250607-134346F

incr backup: 20250607-134346F_20250607-135007I
    timestamp start/stop: 2025-06-07 13:50:07+00 / 2025-06-07 13:50:09+00
    wal start/stop: 000000010000000000000010 / 000000010000000000000010
    database size: 52.6MB, database backup size: 8.3KB
    repo1: backup set size: 8.4MB, backup size: 430B
    backup reference list: 20250607-134346F, 20250607-134346F_20250607-134916I
```

```
# Stop PostgreSQL
sudo systemctl stop postgresql@16-main.service
rm -rf * # remove content of data dir
# Perform the restore
sudo -u postgres pgbackrest --stanza=demo --set=20250607-134346F restore
```

https://www.linkedin.com/in/ahmed-mohamed-423583151



```
root@postgresql-16-test:/data# sudo -u postgres pgbackrest --stanza=demo info
stanza: demo
    status: ok (backup/expire running - 99.90% complete)
    cipher: none

    db (current)
        wal archive min/max (16): 00000001000000000000000A/000000020000000000000014

        full backup: 20250607-134346F
            timestamp start/stop: 2025-06-07 13:43:46+00 / 2025-06-07 13:43:59+00
            wal start/stop: 00000001000000000000000C / 00000001000000000000000C
            database size: 52.6MB, database backup size: 52.6MB
            repo1: backup set size: 8.4MB, backup size: 8.4MB

        incr backup: 20250607-134346F_20250607-134916I
            timestamp start/stop: 2025-06-07 13:49:16+00 / 2025-06-07 13:49:17+00
            wal start/stop: 00000001000000000000000E / 00000001000000000000000E
            database size: 52.6MB, database backup size: 8.3KB
            repo1: backup set size: 8.4MB, backup size: 431B
            backup reference list: 20250607-134346F

        incr backup: 20250607-134346F_20250607-135007I
            timestamp start/stop: 2025-06-07 13:50:07+00 / 2025-06-07 13:50:09+00
            wal start/stop: 000000010000000000000010 / 000000010000000000000010
            database size: 52.6MB, database backup size: 8.3KB
            repo1: backup set size: 8.4MB, backup size: 430B
            backup reference list: 20250607-134346F, 20250607-134346F_20250607-134916I
root@postgresql-16-test:/data# sudo systemctl stop postgresql@16-main.service
root@postgresql-16-test:/data# rm -rf *
root@postgresql-16-test:/data# sudo -u postgres pgbackrest --stanza=demo --set=20250607-134346F restore
root@postgresql-16-test:/data# sudo systemctl start postgresql@16-main.service
root@postgresql-16-test:/data# pg_lsclusters
Ver Cluster Port Status Owner     Data directory Log file
16  main    5432 online <unknown> data/          /var/log/postgresql/postgresql-16-main.log
root@postgresql-16-test:/data#
```

Restore to a specific point in time:

```
# Stop PostgreSQL
sudo systemctl stop postgresql@16-main.service
rm -rf * # remove content of data dir
# Perform the restore

sudo -u postgres pgbackrest --stanza=demo --type=time --target="2025-06-07 16:45:00+03" restore
# Start PostgreSQL
sudo systemctl start postgresql@16-main.service
```

You need to use the `--type=time` and `--target` flags. The timestamp must be in a format PostgreSQL understands and should include your timezone.

Managing Backups with pgBackRest

to check the backup avaialble at pgbackrest use the following command

```
sudo -u postgres pgbackrest --stanza=demo info
```

Checking backup integrity:
`sudo -u postgres pgbackrest --stanza=demo check`

MySQL Backup Strategies

logical backup using mysqldump

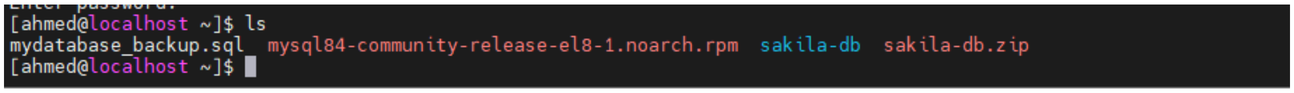
mysqldump is utility for taking dump backup similar to what pg_dump does
it can take all database backup or single database , it can also take specific table backup

taking backup using mysqldump

for this scenario we have sakila database sample already deployed on MySQL we will take dump backup and store in the dedicated `backup/` directory

Backing up a single database:

```
mysqldump -h localhost -u root -p --single-transaction --routines --triggers --events mydatabase >
mydatabase_backup.sql
```



```
Enter password:
[ahmed@localhost ~]$ ls
mydatabase_backup.sql  mysql84-community-release-el8-1.noarch.rpm  sakila-db  sakila-db.zip
[ahmed@localhost ~]$ 
```

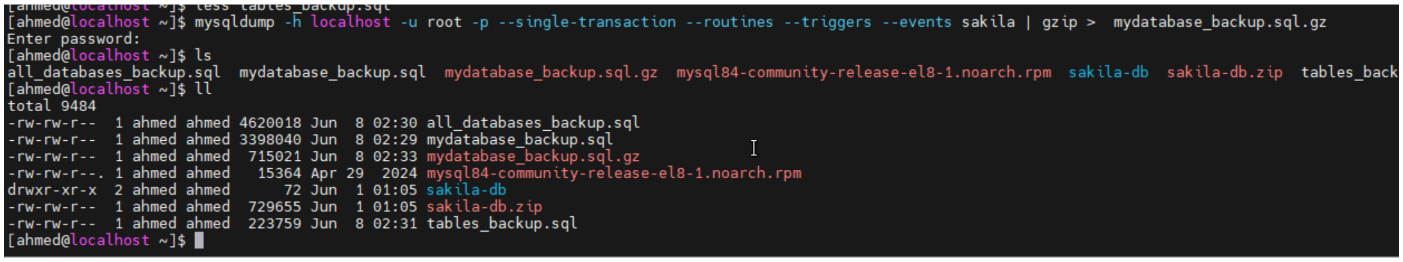
```
mysqldump -h localhost -u root -p --single-transaction --routines --triggers --events --all-databases >
all_databases_backup.sql
```

Backing up specific tables:

```
mysqldump -h localhost -u root -p --single-transaction  sakila  actor  film  > tables_backup.sql
```

Creating a compressed backup:

```
mysqldump -h localhost -u root -p --single-transaction --routines --triggers --events sakila | gzip >
mydatabase_backup.sql.gz
```



```
[ahmed@localhost ~]$ less tables_backup.sql
[ahmed@localhost ~]$ mysqldump -h localhost -u root -p --single-transaction --routines --triggers --events sakila | gzip >  mydatabase_backup.sql.gz
Enter password:
[ahmed@localhost ~]$ ls
all_databases_backup.sql  mydatabase_backup.sql  mydatabase_backup.sql.gz  mysql84-community-release-el8-1.noarch.rpm  sakila-db  sakila-db.zip  tables_back
[ahmed@localhost ~]$ ll
total 9484
-rw-rw-r--  1 ahmed ahmed 4620018 Jun  8 02:30 all_databases_backup.sql
-rw-rw-r--  1 ahmed ahmed 3398040 Jun  8 02:29 mydatabase_backup.sql
-rw-rw-r--  1 ahmed ahmed  715021 Jun  8 02:33 mydatabase_backup.sql.gz
-rw-rw-r--. 1 ahmed ahmed   15364 Apr 29  2024 mysql84-community-release-el8-1.noarch.rpm
drwxr-xr-x  2 ahmed ahmed      72 Jun  1 01:05 sakila-db
-rw-rw-r--  1 ahmed ahmed  729655 Jun  1 01:05 sakila-db.zip
-rw-rw-r--  1 ahmed ahmed  223759 Jun  8 02:31 tables_backup.sql
[ahmed@localhost ~]$ 
```

mysqldump options

- `--single-transaction` this recomneded to include in mysqldump command it will avoid locking the table while taking database dump
- `--flush-logs` Flushes the MySQL server log files before starting the dump
- `--master-data=2` Includes binary log position in the dump output as a commented CHANGE MASTER statement
- `--routines` Includes stored routines (procedures and functions) in the
- `--triggers` Includes triggers in the dump
- `--events` Includes events in the dump
- `--no-data` Dumps only the database structure, not the contents of the tables

Best Practices for mysqldump

- Use `--single-transaction`: For InnoDB tables, always use `--single-transaction` to create consistent backups without locking.
- Include Routines, Triggers, and Events: Use `--routines`, `--triggers`, and `--events` to ensure all database objects are backed up.
- Compression: Use compression to reduce storage requirements, especially for large databases.

Restoring mysqldump backup

Restoring a single database backup:

`mysql -h localhost -u root -p < mydatabase_backup.sql`

Restoring a compressed backup

`gunzip < mydatabase_backup.sql.gz | mysql -h localhost -u root -p`

Restoring to a specific database:

```
mysql -h localhost -u root -p mydatabase < mydatabase_backup.sql
```

Physical Backups with Percona XtraBackup

now this is my desire way of taking backup , it better for production has less effect on database during backup , and restoration is mush faster , it also support differential backup nd the backup and restore proses is mush more streamline than mysqldump
for production environment i highly recommended either use Percona XtraBackup or MySQL enterprise backup if you are using enterprise edition of MySQL

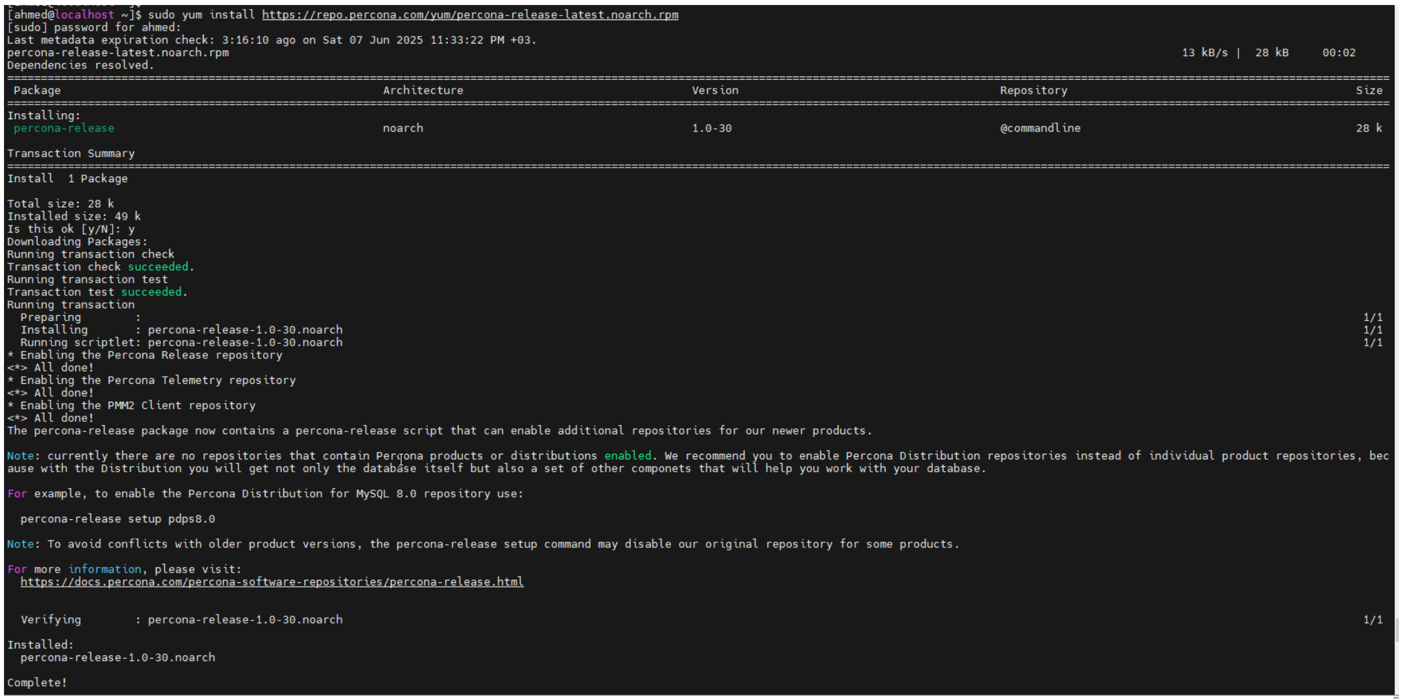
key features of Percona XtraBackup include:

- Non-blocking: Creates consistent backups without locking your database.
- Hot Backup: Backs up your database while it's running.
- Incremental Backup: Supports incremental backups, reducing backup time and storage requirements.
- Compression and Encryption: Supports data compression and encryption.
- Streaming: Can stream backups to another server or to cloud storage.
- Point-in-Time Recovery: Supports point-in-time recovery when used with binary logs.
- Partial Backups: Supports backing up specific databases or tables.

Installing Percona XtraBackup

to install Percona XtraBackup on redhat follow the below steps

```
# Add Percona repository
sudo yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
sudo percona-release setup ps80
# Install Percona XtraBackup
sudo yum install percona-xtrabackup-80
```



```
[ahmed@localhost ~]$ sudo yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
[sudo] password for ahmed:
Last metadata expiration check: 3:16:10 ago on Sat 07 Jun 2025 11:33:22 PM +03.
percona-release-latest.noarch.rpm                                                                                      13 kB/s |  28 kB     00:02
Dependencies resolved.
=========================================================================================================================================================
 Package                                   Architecture                        Version                              Repository                    Size
=========================================================================================================================================================
Installing:
 percona-release                           noarch                              1.0-30                               @commandline                  28 k

Transaction Summary
=========================================================================================================================================================
Install  1 Package

Total size: 28 k
Installed size: 49 k
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                                                 1/1
  Installing       : percona-release-1.0-30.noarch                                                                                                   1/1
  Running scriptlet: percona-release-1.0-30.noarch                                                                                                   1/1
* Enabling the Percona Release repository
<*> All done!
* Enabling the Percona Telemetry repository
<*> All done!
* Enabling the PMM2 Client repository
<*> All done!
The percona-release package now contains a percona-release script that can enable additional repositories for our newer products.

Note: currently there are no repositories that contain Percona products or distributions enabled. We recommend you to enable Percona Distribution repositories instead of individual product repositories, bec
ause with the Distribution you will get not only the database itself but also a set of other componets that will help you work with your database.

For example, to enable the Percona Distribution for MySQL 8.0 repository use:

  percona-release setup pdps8.0

Note: To avoid conflicts with older product versions, the percona-release setup command may disable our original repository for some products.

For more information, please visit:
  https://docs.percona.com/percona-software-repositories/percona-release.html


  Verifying        : percona-release-1.0-30.noarch                                                                                                   1/1

Installed:
  percona-release-1.0-30.noarch

Complete!
```

Creating a full backup

to create full backup use the following syntax , for this instant i will take full backup and store it in the dedicated backup directory with separate mount point

```
xtrabackup --backup --target-dir=/path/to/backup
```

Creating a full backup with compression:

```
xtrabackup --backup --compress --target-dir=/path/to/backup
```

Creating a full backup for specific databases:

```
xtrabackup --backup --databases="db1 db2" --target-dir=/path/to/ backup
```

Creating an incremental backup:

```
xtrabackup --backup --target-dir=/path/to/incremental/backup1 -- incremental-basedir=/path/to/full/backup
```

Preparing Backups for Restoration

Before a backup can be restored, it needs to be prepared. This applies the committed transactions and rolls back the uncommitted ones:

Preparing a full backup:

```
xtrabackup --prepare --target-dir=/path/to/backup
```

Preparing a full backup with incremental backups:

Prepare the full backup with --apply-log-only

```
xtrabackup --prepare --apply-log-only --target-dir=/path/to/ full/backup
```

Apply the first incremental backup with --apply-log-only

```
xtrabackup --prepare --apply-log-only --target-dir=/path/to/ full/backup --incremental-dir=/path/to/incremental/backup1
```

Apply the second incremental backup (last one without --apply-log-only)

```
xtrabackup --prepare --target-dir=/path/to/full/backup -- incremental-dir=/path/to/incremental/backup2
```

Stop MySQL:

```
sudo systemctl stop mysqld
```

Move or rename the current data directory:

```
sudo mv /var/lib/mysql /var/lib/mysql.bak
```

Create a new data directory:

```
sudo mkdir /var/lib/mysql
```

Copy the prepared backup to the data directory:

```
sudo xtrabackup --copy-back --target-dir=/path/to/backup
```

Set the correct ownership:

```
sudo chown -R mysql:mysql /var/lib/mysql
```

Start MySQL:

```
sudo systemctl start mysqld
```

https://www.linkedin.com/in/ahmed-mohamed-423583151