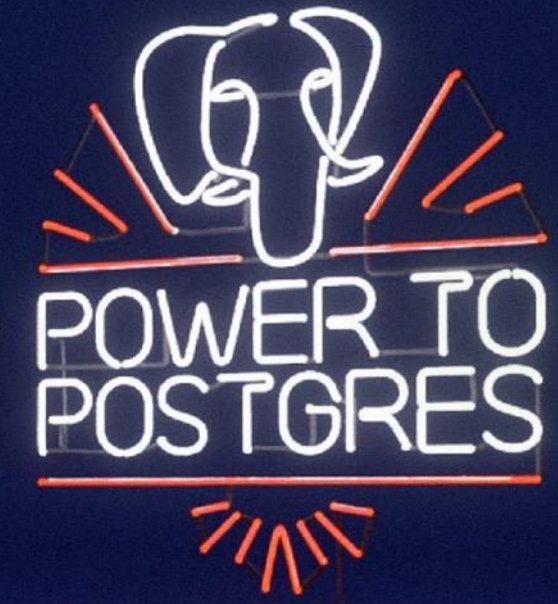
Backup and Recovery Best Practices Office Hour



Presented by:

Sunil Narain, Technical Director

Customer Success Team

EDB

Agenda

1. Why do we need backups?
2. PostgreSQL database Backup options
3. Backup strategies
4. Backup and recovery best practices
5. Demo
6. Q &A

Why do we need backups?

- A backup is a consistent copy of the data that can be used to recover the database.
- Databases need to be backed up to avoid data loss due to:
  - User error
  - Hardware failure
  - Data corruption
- Need the ability to restore old data due to Compliance reasons
- Databases need to be quickly restored to meet the RPO and RTO requirements.
- To protect company's business and reputation

PostgreSQL database backup options

- Logical backup using pg_dump / pg_dumpall
- Physical backup using pg_basebackup, file system snapshots, or a cold backup
- EDB supported and other third party tools:
  - Barman (EDB Supported)
  - pgBackRest (EDB Supported)
  - BART (EDB Supported, but reaching EOL)
  - Veritas Netbackup (EDB partnership)
  - Other third party tools

Logical backup

pg_dump/pg_dumpall

- The pg_dump can be used to take a consistent dump of the current database.
- The pg_dumpall can be used to backup the entire cluster and all global objects.
- Dumps can be in plain-text file containing the SQL commands or in archive format.
- Command examples:
  - pg_dump –h <db host> -p <db port> -U <dbuser> -W –F <format> -f <dumpfile> [-a|-s] –d <dbname>

    (Where format can be p (plain), c (custom), d (directory), or t (tar))
  - pg_dumpall –h <db host> -p <db port> -U <dbuser> -W –f <filename> [-a|-s] [-g] –l <dbname>

Restoring logical backup

- The plain format dumps created by pg_dump can be restored in a new database, as below:

  psql –d newdb –f <dump>

- The custom, directory, or tar format dump can be restored using pg_restore utility.
- Use –clean and –create options with pg_restore to restore the backup in same database.

Physical backup

pg_basebackup utility

- It's a utility to take a base backup of running PostgreSQL cluster
- The backup can be used to do a point-in-time recovery using the WAL files
- Makes an exact copy of cluster's files
- One can view the progress of the backup using pg_stat_progress_basebackup view.
- Examples:
  - pg_basebackup –h <remote_db_server> -p <port> -D <local data dir>
  - pg_basebackup -D - -Ft –X fetch | bzip2 > backup.tar.bz2

Physical backup

File system snapshot

- If your filesystem supports it, make a frozen snapshot of the database volume
- Copy the whole directory to a backup device
- Release the frozen snapshot
- Make sure to include all WAL files in your backup to perform a crash recovery when Postgres is started using the backed up data

Physical backup

Cold backup

- Shutdown Postgres
- Copy the entire data directory:
  - e.g. tar –cf backup.tar <PGDATA>

Barman utility

Features

- Remote backup with rsync OR PostgreSQL protocol
- Management of multiple PostgreSQL/EPAS servers
- Support for file level incremental backups with rsync method
- WAL archiving and streaming
- WAL archive compression with gzip, bzip2
- Point-In-Time-Recovery (PITR)
- Support for Local and remote recovery (via SSH)
- Management of retention policies of backups
- Integrated with EDB PEM (8.4 onwards)
- Supports backup to S3, Azure Blob Storage, and GCS

pgBackRest utility

Features

- Parallel backup & restore
- Local or remote operation
- Full, incremental, and differential backups
- Retention policies
- Backup integrity
- Backup encryption
- S3, Azure, and GCS support

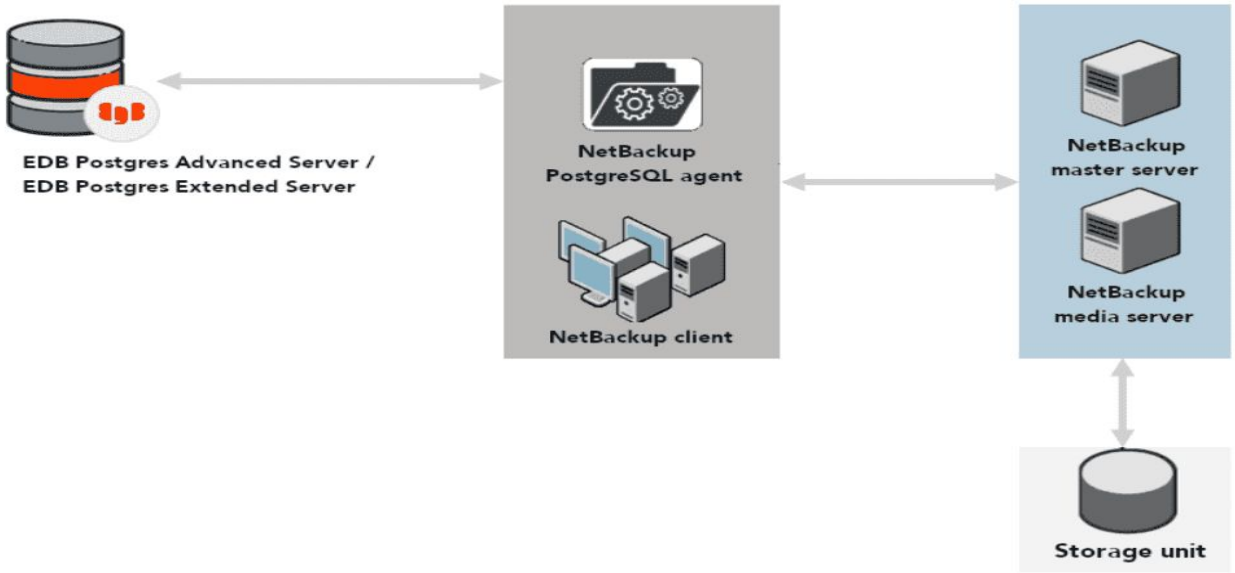
BART utility

(Reaching EOL)

- Backup and recovery management of local and remote database servers.
- Integrates with EDB Postgres Enterprise Manager
- Configurable retention policies
- Full and block-level incremental backup
- Parallel copy and restore
- Point-in-time recovery to specified transaction ID or timestamp

Veritas NetBackup

(EDB Partnership)



Feature comparison

Feature	Barman	pgBackRest	pg_basebackup
SSH Protocol	✓	✓	
PostgreSQL protocol	✓		✓
Incremental backups	✓	✓	
RPO=0	✓		
Rate limiting	✓		✓
Custom WAL sizes	✓	✓	✓
WAL archive compression	✓	✓	✓
Backup compression		✓	
Symmetric encryption		✓	
Parallel backup and restore	✓	✓	
Partial restore (slected databases)		✓	
Centralize repository	✓	✓	
Retention policy	✓	✓	
List backup	✓	✓	
S3 support	✓	✓	
Nagios integration	✓	✓	
PEM integration	✓		
No custom scripts required	✓	✓	

Backup strategies

- Depending on the backup option and database size, decide the frequency of full and incremental/differential backup.
- Setup wal archiving to keep the wals for point-in-time recovery.
- Backup strategy should meet the RTO and RPO requirements
- Adjust your backup retention policies to meet your legal/compliance requirements
- Use 3-2-1 rule and keep 3 copies of backup: 2 local copies and 1 offsite.
- Encrypt your backup

Best practices

- Make sure to have your backup and recovery policies and procedures documented
- Keeping a copy of the backup offsite or in cloud can prevent data loss during a disaster when you lose an on-prem data center.
- Perform regular tests of your backup by doing a recovery
- Monitor your backup process and get alerted when backup fails.
- When you use logical backup method, keep in mind that it’s just a snapshot of the data and the backup cannot be used for doing PITR.
- While restoring the backup, restore it in a directory other than the source data directory.

Demo

EDB

Thank you

Resources

Barman documentation:
https://www.enterprisedb.com/docs/supported-open-source/barman/

pgBackRest documentation
https://www.enterprisedb.com/docs/supported-open-source/pgbackrest/

BAR documentation:
https://www.enterprisedb.com/docs/bart/latest/

Veritas Implementation Guide:
https://www.enterprisedb.com/docs/partner_docs/VeritasGuide/

Blog on Backup and Recovery :

https://www.enterprisedb.com/postgresql-database-backup-recovery-what-works-wal-pitr

Feature comparison of various backup tools:

https://www.enterprisedb.com/products/backup-recovery-postgresql-database-auto-restore-script-tools