# Backup and Recovery in Postgres

Presented by
Mansur Shaikh | Sr. S.E. EDB

EDB

# Agenda

- Backup Types
- Database SQL Dumps and restore dumps
- Offline Physical Backups
- Continuous Archiving
- Online Physical Backups Using pg_basebackup
- Point-in-time Recovery and recovery settings
- EDB tools

# Why do we needbackups?

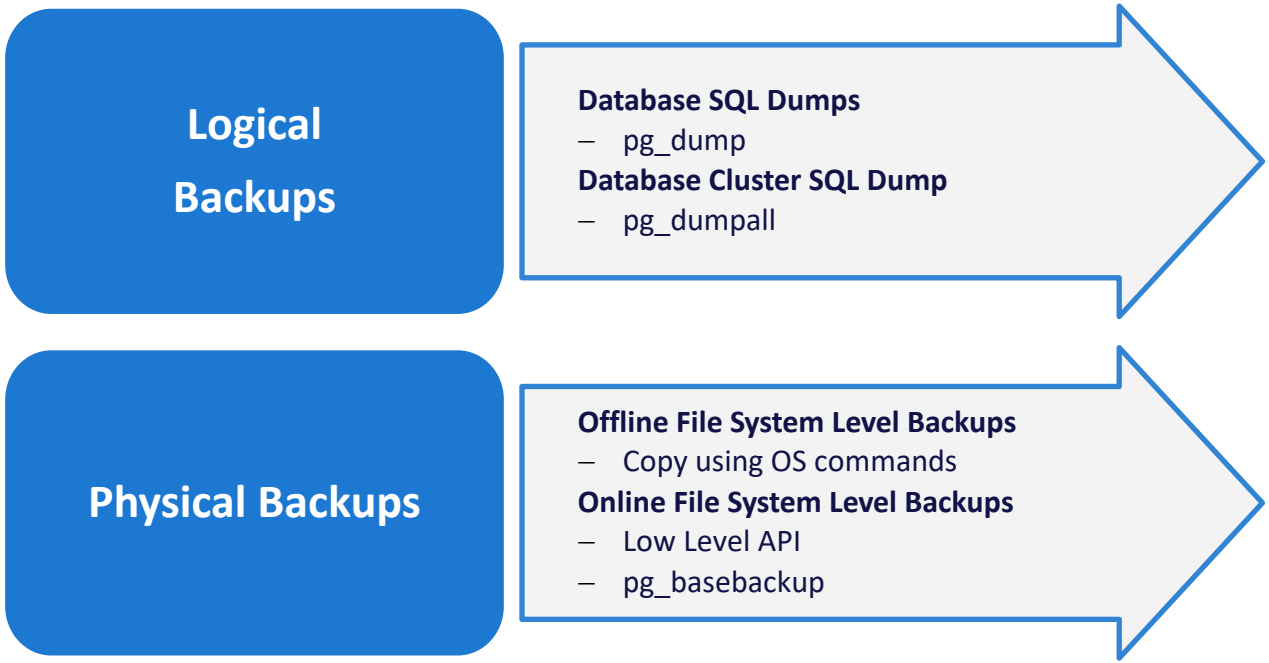
- A backup is a consistent copy of the data that can be used to recover the database.
- Databases need to be backed up to avoid data loss due to:
- User error
- Hardware failure
- Data corruption
- Need the ability to restore old data due to Compliance reasons
- Databases need to be quickly restored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



© Copyright EnterpriseDB Corporation, 2020-21. All rights reserved.

# Types of Backup

- As with any database, PostgreSQL databases should be backed up regularly



# Logical Backups

# Database SQL Dump

- Generate a text file with SQL commands
- Postgres provides the utility program **pg_dump** for this purpose
- **pg_dump** does not block readers or writers
- Dumps created by **pg_dump** are internally consistent, that is, the dump represents a snapshot of the database as of the time **pg_dump** begins running
- Syntax:

```
$ pg_dump [options] [dbname]
```

# pg_dump Options

| | |
|---|---|
| -a | - Data only. Do not dump the data definitions (schema) |
| -s | - Data definitions (schema) only. Do not dump the data |
| -n <schema> | - Dump from the specified schema only |
| -t <table> | - Dump specified table only |
| -f <file name> | - Send dump to specified file. Filename can be specified using absolute or relative location |
| -Fp | - Dump in plain-text SQL script (default) |
| -Ft | - Dump in tar format |
| -Fc | - Dump in compressed, custom format |
| -Fd | - Dump in directory format |
| -j njobs - dump in parallel by dumping n jobs tables simultaneously. Only supported with –Fd | |
| -B, --no-blobs - Excludes large objects in dump | |
| -v | - Verbose option |

# SQL Dump - Large Databases

- If the operating system has maximum file size limits, it can cause problems when creating large **pg_dump** output files
- Standard Unix tools can be used to work around this potential problem
  - Use a compression program, for example gzip:

    ```
    $ pg_dump dbname | gzip > filename.gz
    ```

  - Also the split command allows you to split the output into smaller files:

    ```
    $ pg_dump dbname | split -b 1m - filename
    ```

# Restore – SQL Dump

- Backups taken using pg_dump with plain text format(Fp)
- Backups taken using pg_dumpall

psql client

- Backup taken using pg_dump with custom(Fc), tar(Ft) or director(Fd) formats
- Supports parallel jobs for during restore
- Selected objects can be restored

pg_restore utility

EDB

# pg_restore Options

| | |
|---|---|
| -F [c\|d\|t] | - Backup file format |
| -d <database name> | - Connect to the specified database. Also restores to this database if -C option is omitted |
| -C | - Create the database named in the dump file and restore directly into it |
| -a | - Restore the data only, not the data definitions (schema) |
| -s | - Restore the data definitions (schema) only, not the data |
| -n <schema> | - Restore only objects from specified schema |
| -N <schema> | - do not restore objects in this schema |
| -t <table> | - Restore only specified table |
| -v | - Verbose option |

# Entire Cluster - SQL Dump

- **pg_dumpall** is used to dump an entire database cluster in plain-text SQL format
- Use psql to restore
- Syntax:

```
$ pg_dumpall [options…] > filename.backup
```

# pg_dumpall Options

| | |
|---|---|
| -a | - Data only. Do not dump schema |
| -s | - Data definitions (schema) only |
| -g | - Dump global objects only - not databases |
| -r | - Dump only roles |
| -c | - Clean (drop) databases before recreating |
| -O | - Skip restoration of object ownership |
| -x | - do not dump privileges (grant/revoke) |
| -v | - Verbose option |
| --exclude-database | -exclude database whose name match with given pattern |

# Physical Backups

# Backup - File system level backup

- An alternative backup strategy is to directly copy the files that PostgreSQL uses to store the data in the database
- You can use whatever method you prefer for doing usual file system backups, for example:

```
$ tar -cf backup.tar /usr/local/pgsql/data
```

- The database server must be shut down or in backup mode in order to get a usable backup
- File system backups only work for complete backup and restoration of an entire database cluster
- Two types of File system backup
  - Offline backups
  - Online backups

EDB

# File System Backups

**Offline Backups**

- Taken using OS Copy command
- Database Server must be shutdown
- Complete Backups
- Used to restore data

**Online Backups**

- Continuous archiving must be enabled
- Database server start/end backup mode
- Complete backups
- Used to recover data
- Two methods - Low Level API  & pg_basebackup

# Continuous Archiving

- PostgreSQL maintains WAL files for all transactions in **pg_wal** directory
- PostgreSQL automatically maintains the WAL logs which are full and switched
- Continuous archiving can be setup to keep a copy of switched WAL Logs which can be later used for recovery
- It also enables online file system backup of a database cluster
- Requirements:
  - `wal_level` must be set to `replica`
  - `archive_mode` must be set to `on` (can be set to always)
  - `archive_command` must be set in **postgresql.conf** which archives WAL logs and supports PITR

# Continuous Archiving Methods

## Archiver Process

- Parameters in postgresql.conf file
  - wal_level = replica
  - archive_mode = on
  - archive_command = 'cp -i %p /pgsql/archive/%f'
- Restart the database server
- Archive files are generated after every log switch

## Streaming WAL

- Parameters in postgresql.conf file
  - wal_level = replica
  - archive_mode = on
  - max_wal_senders = 3
- Restart the database server
- pg_receivewal –h localhost –D /pgsql/archive
- Transactions are streamed and written to archive files

EDB

# Base Backup Using pg_basebackup Tool

- `pg_basebackup` can take an online base backup of a database cluster
- This backup can be used for PITR or Streaming Replication
- `pg_basebackup` makes a binary copy of the database cluster files
- System is automatically put in and out of backup mode

# pg_basebackup - Online Backup

- Steps require to take Base Backup:
  - Modify **pg_hba.conf**

```
host replication  postgres [Ipv4 address of client]/32  md5
```

  - Modify **postgresql.conf**

```
wal_level = replica
archive_command = 'cp -i %p /users/postgres/archive/%f'
archive_mode = on
max_wal_senders = 3
wal_keep_size = 512
```

- Backup Command:

```
$ pg_basebackup [options] ..
```

# Options for pg_basebackup command

| | |
|---|---|
| -D <directory name> | - Location of backup |
| -F <p or t> | - Backup files format. Plain(p) or tar(t) |
| -R | - write standby.signal and append postgresql.auto.conf |
| -T OLDDIR=NEWDIR | - relocate tablespace in OLDDIR to NEWDIR |
| --waldir | - Write ahead logs location |
| -z | - enable gzip compression for files |
| -Z level | - Compression level |
| -P | - Progress Reporting |
| -h host | - host on which cluster is running |
| -p port | - cluster port |

- To create a base backup of the server at localhost and store it in the local directory /usr/local/pgsql/backup

```
$ pg_basebackup -h localhost -D /usr/local/pgsql/backup
```

# Point-in-time Recovery

- Point-in-time recovery (PITR) is the ability to restore a database cluster up to the present or to a specified point of time in the past
- Uses a full database cluster backup and the write-ahead logs found in the `/pg_wal` subdirectory
- Must be configured before it is needed (write-ahead log archiving must be enabled)

# Performing Point-in-Time Recovery

## Prepare

Stop the server

Take a file system level backup if possible

Clean the data directory

## Restore

Copy data cluster files and folders from backup location to the data directory

Use `cp -rp` to preserve privileges

## Configure

Configure recovery settings in **postgresql.conf** file

Create **recovery.signal** file in the data directory

## Recover

Start the server using service or `pg_ctl` utility

Check error log for any issue

**recovery.signal** file is removed automatically after recovery

# Point-in-Time Recovery Settings

- Restoring archived WAL using restore_command parameter:
  - Unix:

```
restore_command = 'cp /mnt/server/archivedir/%f "%p"'
```

  - Windows:

```
restore_command = 'copy c:\\mnt\\server\\archivedir\\"%f" "%p"'
```

- Recovery target settings:
  - `recovery_target_name`
  - `recovery_target_time`
  - `recovery_target_xid`
  - `recovery_target_action`

# Backup and Recovery

EDB supports multiple options for simple and reliable PostgreSQL Backup and Recovery

## Barman

- 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

## pgBackRest

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



25

# 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 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 is not suitable for doing PITR.
- While restoring the backup, restore it in a directory other than the source data directory.



© Copyright EnterpriseDB Corporation, 2021. All rights reserved.

# Summary

- Backup Types
- Database SQL Dumps
- Restoring SQL Dumps
- Offline Physical Backups
- Continuous Archiving
- Online Physical Backups Using pg_basebackup
- Point-in-time Recovery
- Recovery Settings

# Thank you