Backup and Recovery using PITR

Mark Jones

Agenda

- Introduction
- Business Impact Vs Cost
- Downtime Scenarios
- Backup Methods
  - SQL Dump
  - Cold Backup (Offline Backups)
  - Hot Backup (Online Backup)
- Online Backups
  - Continuous Archiving
  - Online Backups
- Recovery Solutions
- Point-In Time Recovery Concepts
- Recovery Example

Introduction

- Backup and Recovery Strategies are absolutely key to the successful operation of any business critical data management.

- Database backup strategies must plan for catastrophic failure of a device, maintenance, disaster at a location, operator error, data corruption, and compliance requirements.

Business Impact vs Cost

- How long will recovery take?
- How long do we need to store backup data?
- How much will data storage cost?
- Will an outage effect our Brand?
- Can any part of the database remain operational whilst I recover elsewhere?
- Do I know when the problem I am recovering from started?

Downtime Scenarios

- Device Failure
- Maintenance
- Site Failure
- Operator Error
- Data Corruption
- Compliance

Device Failure

- Loss of Machine
- Loss of Disk
- Loss Of Power

Site Failure

- Datacenter Failure
- Network Failure
- Office Break-In

Data Corruption

- Application introduces poor code and in turn corrupts the data
- Disk level corruption

Operator Error

- Update error
- Dropped table
- Dropped schema
- Deletion of datafile

Maintenance

- Hardware upgrades
  - O/S upgrades

Compliance

- Data Retention Periods
  - Readable data
  - Writeable data
  - Storage of data

Backup Methods

- As with any database, Postgres database should be backed up regularly.
- There are three fundamentally different approaches to backing up Postgres data:
  - SQL dump
  - File system level backup
  - Continuous Archiving

Backup - SQL Dump

- Generate a text file with SQL commands
- Postgre 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]
```

Backup - Entire Cluster – SQL Dump

- pg_dumpall is used to dump an entire database cluster in plain-text SQL format
- Dumps global objects - user, groups, and associated permissions
- Use PSQL to restore

Syntax:

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

SQL Dump - Large Databases

- If operating systems have 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.
  - You can use your favorite 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
```

Backup – Cold Backup

- An alternative backup strategy is to directly copy the files that Postgres 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 in order to get a usable backup.
- File system backups only work for complete backup and restoration of an entire database cluster.

Backup – Hot Backup

- A Hot Backup allows the database to stay operational during backup
- Postgres maintains WAL files for all transactions in the **pg_xlog** directory
- Postgres 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 archive
  - archive_mode must be set to on
  - archive_command must be set in **postgresql.conf** which archives WAL logs and supports PITR

Continuous Archiving

- Edit the **postgresql.conf** file and set the archive parameters:

  wal_level=archive

  archive_mode=on

  Unix:

  ```
  archive_command= 'cp %p /mnt/server/archivedir/%f'
  ```

  %p is the absolute path of WAL otherwise you can define the path

  %f is a unique file name which will be created on above path

Online Base Backup

- Make a base backup

  Connect using psql and issue the command:

  ```
  SELECT pg_start_backup('any useful label');
  ```

  Use a standard file system backup utility to back up the /data subdirectory

  Connect using psql and issue the command:

  ```
  SELECT pg_stop_backup();
  ```

  Continuously archive the WAL segment files

Online Base Backup Using pg_basebackup tool

- **pg_basebackup** can take a base backup of a Postgres cluster.
- This backup can be used for PITR or Streaming Replication.
- **pg_basebackup** makes a binary copy of the database cluster files.
- The system is automatically put in and out of backup mode.

Options for pg_basebackup

- Command line options for **pg_basebackup**:
  -D <directory name> - Location of backup
  -F <p or t> - Backup files format. Plain(p) or tar(t)
  -X - include required transaction log files
  -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 locahost -D /usr/backup
```

pg_basebackup - Online Backup

- Steps require to take Base Backup:
  - Modify **pg_hba.conf**
    host replication enterprisedb [Ipv4 address of client]/32 trust
  - Modify **postgresql.conf**
    archive_command = 'cp -i %p /Users/postgres/archive/%f‘
    archive_mode = on # Require Restart
    max_wal_senders = 3 # Maximum 'wal_senders‘
    wal_keep_segments = 50 # How many WAL segments (=files should be kept on the server
    wal_level=archive
- Backup Command:

```
pg_basebackup -D /usr/backup/testbackup -v -h 127.0.0.1
-U postgres
```

Recovery Solutions

- PITR
- Pg_restore
- Replication
- Replication with Failover and Switchback
- Delayed Replication
- Snapshots

Backup and Recovery Paths

	PITR	Replication	Delayed Replication	Failover/ switchback	Pg_restore	Restore from Cold Backup	Snapshots
Device Failure							
Site Failure							
Data Corruption							
Operator error							
Maintenance							
Compliance							

Best Fit | Good Fit | Choose Better Alternatives

EDB ENTERPRISEDB

Point-in-Time Recovery Concepts

- 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_xlog` subdirectory.
- Must be configured before it is needed (write-ahead log archiving must be enabled).
- Must have information of when you want to recover to?

Performing Point-in-Time Recovery

- Stop the server, if it's running.
- If you have enough space keep a copy of the data directory and transaction logs.
- Remove all directories and files from the cluster data directory.
- Restore the database files from your file system backup.
- Verify the ownership of restored backup directories (must not be root).
- Remove any files present in `pg_xlog/`.
- If you have any unarchived WAL segment files recovered from crashed cluster, copy them into `pg_xlog/`.
- Create a recovery command file **recovery.conf** in the cluster data directory.
- Start the server.
- Upon completion of the recovery process, the server will rename **recovery.conf** to **recovery.done**.

Point-in-Time Recovery Options

- Settings in the **recovery.conf** file:
  - restore_command(string)
    - Unix:

```
restore_command = 'cp /mnt/server/archivedir/%f "%p"'
recovery_target_name (string)
 (select pg_create_restore_point('Name');
recovery_target_time(timestamp)
recovery_target_xid(string)
recovery_target_inclusive(boolean)
pause_at_recovery_target (boolean)
recovery_target 'immediate'
```

Point-in-Time Recovery help

- Hot_standby = on postgres.conf if using pause
- pause_at_recovery_target = true
- Select pg_is_xlog_replay_paused();
- Select pg_xlog_replay_resume();

Summary

- Introduction
- Business Impact Vs Cost
- Backup Methods
  - SQL Dump
  - Cold Backup (Offline Backups)
  - Hot Backup (Online Backup)
- Online Backups
  - Continuous Archiving
  - Online Backups
- Point-In Time Recovery Concepts
- Recovery Example

EDB ENTERPRISEDB