PostgreSQL Backup, Restore and Point-in-Time Recovery

Using pgBackRest

A comprehensive guide to setting up, configuring, and operating pgBackRest for PostgreSQL databases

Reliable backup and restore solution that seamlessly scales to the largest databases and workloads

Table of Contents

Introduction and Overview

What is pgBackRest?

pgBackRest is a reliable backup and restore solution specifically designed for PostgreSQL databases that offers:

- Seamless scaling for large databases and workloads
- Full, differential, and incremental backup types
- Point-in-Time Recovery (PITR) capabilities
- Compression and deduplication for efficient storage

Protect Your Data

Reliable, scalable backup solution

Installation Steps

1 Install Dependencies

Required libraries for pgBackRest functionality:

```
sudo apt-get install postgresql-client libxml2 libssh2-1
```

2 Install pgBackRest

Install the main backup tool:

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

3 Verify Installation

Check version and available options:

```
sudo -u postgres pgbackrest
```

This displays the version and command-line options available in pgBackRest

Repository Setup

Creating a Backup Repository

Create a dedicated directory for pgBackRest backup files:

```
# Create backup directory structure
mkdir -p /db_backup/base_backup

# Set proper ownership permissions
chown postgres:postgres /db_backup/
```

Why a Dedicated Repository?

- Centralized storage for all backup files
- Properly secured with PostgreSQL user permissions
- Easily managed backup storage and retention

Storage Structure

```
/db_backup/
└── base_backup/
```

This directory will store all backup files and WAL archives

Configuration

Key Configuration Options

repo1-path	Path where backups and WAL archives are stored
repo1-block	Enables block-level incremental backup
repo1-bundle	Bundles small files to improve performance
repo1-retention-diff	Number of differential backups to retain
repo1-retention-full	Number of full backups to retain
compress-level	Compression level (0-9) for backups
start-fast	Enables quick backup start without validation

Sample Configuration

Edit **/etc/pgbackrest.conf** to set required configuration:

```
[global]
repo1-block=y
repo1-bundle=y
repo1-path=/db_backup/base_backup
repo1-retention-diff=1
repo1-retention-full=2
start-fast=y
compress-level=6

[global:archive-push]
compress-level=6

[cluster]
pg1-path=/db_data/data
pg1-socket-path=/tmp
```

Configuration Tips:

- Create separate stanzas for each PostgreSQL cluster
- Configure retention policies to manage backup storage
- Set appropriate compression level based on CPU resources

PostgreSQL Archive Settings

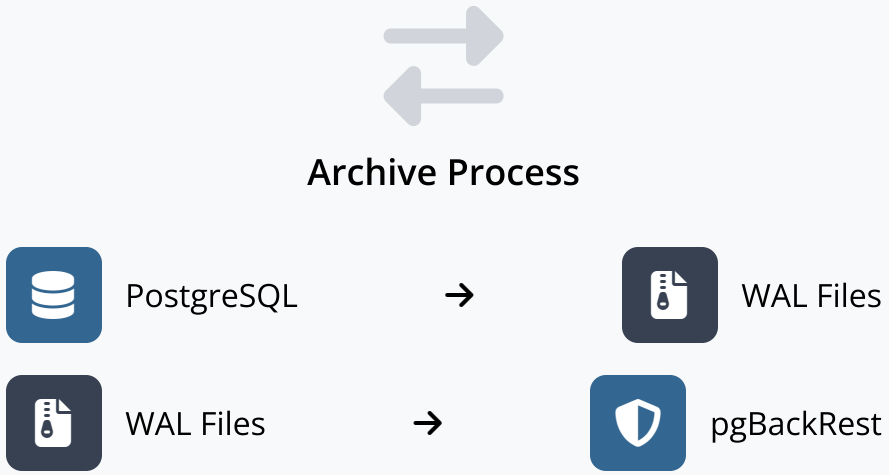
Update $PGDATA/postgresql.conf

These settings enable Write-Ahead Logging (WAL) archiving required for pgBackRest to function:

```
listen_addresses = '*'
wal_level = replica
archive_mode = on
archive_command = 'pgbackrest --stanza=master archive-push %p'
max_wal_senders = 3
```

Important: Restart PostgreSQL service after configuration changes

```
systemctl restart postgresql
```



Stanza Management

What is a Stanza?

A stanza defines a PostgreSQL database cluster for backup in pgBackRest configuration.

Create a stanza:

```
sudo -u postgres pgbackrest --stanza=cluster --log-level-console=info
stanza-create
```

Validate configuration:

```
sudo -u postgres pgbackrest --stanza=cluster --log-level-console=info
check
```

Must be created before performing any backup operations

Stanzas Organize Your Backups

Logical organization of PostgreSQL clusters

Tip: Always verify stanza configuration with the check command before attempting backups to ensure proper operation.

Full Backup Procedures

Performing Full Backups

Full backups capture the entire database cluster and create a complete backup set:

```
sudo -u postgres pgbackrest --stanza=cluster
--log-level-console=info --type=full backup
```

Key Command Parameters

- **--type=full**: Creates a full backup regardless of previous backups
- **--log-level-console=info**: Shows detailed progress during backup

Check Backup Information

```
sudo -u postgres pgbackrest info
```

Full Backup Benefits

- Complete point-in-time recovery
- Foundation for differential/incremental
- Disaster recovery preparation
- Database migration support

Differential & Incremental Backups

Full Backup

Complete backup of the entire database cluster.

- Base for other backup types

```
sudo -u postgres pgbackrest
--stanza=cluster
--log-level-console=info
--type=full backup
```

Differential Backup

Backs up changes since the last full backup.

- Smaller than full backups

```
sudo -u postgres pgbackrest
--stanza=cluster
--log-level-console=info
--type=diff backup
```

Incremental Backup

Backs up changes since the last backup of any type.

- Smallest, fastest backup type

```
sudo -u postgres pgbackrest
--stanza=cluster
--log-level-console=info
--type=incr backup
```

When to Use Each Backup Type

Full Backup
Weekly or during low-traffic periods

Incremental Backup
Daily or hourly for critical systems

Check backup info with: `sudo -u postgres pgbackrest info`

Restore Procedures

Follow these steps to restore a PostgreSQL database from a pgBackRest backup:

1 Stop PostgreSQL Service

```
systemctl stop postgresql
```

Ensure PostgreSQL is completely stopped before restoring data

2 Remove Existing Data Files

```
cd /db_data/data
rm -rf *
```

Warning: This will permanently delete all existing data files. Be certain you have a valid backup before proceeding.

3 Restore the Database Cluster

```
sudo -u postgres pgbackrest --stanza=cluster --log-level-console=info restore
```

This command restores the most recent backup by default

4 Start PostgreSQL Service

```
systemctl start postgresql
```

Verify successful restore by checking logs and database connectivity

Pro Tip

Always verify backup integrity with `pgbackrest info` command before attempting a restore operation.

Point-in-Time Recovery (PITR)

Recovering to a Specific Moment in Time

PITR allows recovery of a database to any point in time within your backup retention window, ideal for:

- Recovering from accidental data deletion or corruption
- Creating database copies at specific points in time

Step-by-Step PITR Process:

1. **Perform a full backup**
2. **Wait for data changes to occur**
3. **Stop PostgreSQL cluster**
   systemctl stop postgresql
4. **Remove data directory files**
   cd /db_data/data; rm -rf *

PITR Command Example

Restore to a specific timestamp:

```
sudo -u postgres pgbackrest \
--stanza=cluster \
--log-level-console=info \
--type=time \
"--target=2024-07-26 17:11:35" \
--target-action=promote \
restore
```

Key Parameters:

- **--type=time**: Specifies time-based recovery
- **--target**: The timestamp to recover to
- **--target-action**: Action after recovery completes (promote)

Rewind your database to any moment before data loss occurred

Advanced Features and Best Practices

Advanced Features

Block Incremental Backup
Reduce backup time by only storing changed blocks

Configurable Compression
Optimize storage vs. performance with compression levels

File Bundling
Bundle small files for better performance

```
# Stanza deletion:
sudo -u postgres pgbackrest --stanza=cluster \
--repo=1 --log-level-console=info stanza-delete
```

Best Practices & Troubleshooting

Regular Maintenance
Monitor backup logs and review retention policies

Test Restores
Regularly test restore procedures on test environments

Automation
Automate backup operations with scheduling tools

Troubleshooting Tips
Use --log-level-console=debug for detailed logging
Check disk space regularly to prevent backup failures