**PostgreSQL Backup**

**Introduction**

This document provides instructions for creating a reliable backup and recovery strategy for PostgreSQL databases. The strategy is designed to ensure data integrity, prevent downtime, and enable quick recovery in case of failure. It includes automation techniques and best practices tailored for critical tasks.

**Creating the Batch File for PostgreSQL Backup**

1. Open Notepad or any text editor.
2. Enter the script below, replacing placeholders with your actual values:

@echo off

set PGPASSWORD=root

set USER=user

set DBNAME=postgres

set BACKUP\_DIR=C:\Users\USER\Desktop\

set BACKUP\_FILE=C:\Users\USER\Desktop\postgresBackup.sql

echo Backing up database %DBNAME% to %BACKUP\_FILE%

"C:\Program Files\PostgreSQL\16\bin\pg\_dump.exe" -U %USER% %DBNAME% > %BACKUP\_FILE%

1. Replace the "USER" and "DBNAME" values with your actual credentials.
2. Verify the path to “pg\_dump.exe” corresponds to your PostgreSQL installation.
3. Save the file with a .bat extension, e.g., “pg\_backup.bat”.

**Scheduling Your Batch File with Windows Task Scheduler**

1. Open Windows Task Scheduler:

-right click on start button/computer management

A screenshot of a computer menu

Description automatically generated

1. Create a new task by selecting **Create Task...**.A screenshot of a computer

   Description automatically generated
2. Configure the task as follows:
   * **General Tab:**
     + Name the task (e.g., "PostgreSQL Backup").
   * **Triggers Tab:**
     + Define the schedule (e.g., daily or weekly).

A screenshot of a computer

Description automatically generated

* + **Actions Tab:**
    - Choose **Start a Program** and select your pg\_backup.bat file.
  + **Conditions and Settings Tabs:**
    - Adjust settings as necessary (e.g., run task only if the computer is idle).

1. Save the task.

Your batch file will now run on the defined schedule, automating PostgreSQL backups.

**Backup Strategy**

1. **Backup Types:**
   * Weekly full backups.
   * Daily incremental backups.
2. **Storage Locations:**
   * On-premise and cloud storage (e.g., AWS S3).
   * Encrypt backups for security.
3. **Testing Backups:**
   * Perform quarterly restoration tests to ensure reliability.
4. **Retention Policy:**
   * Maintain backups for at least six months.

**Preventing Database Downtime**

1. **High Availability:**
   * Configure streaming replication to maintain a standby database.
2. **Monitoring and Alerts:**
   * Use tools like pgAdmin or Nagios to monitor performance.
   * Set up alerts for issues like low disk space or backup failures.
3. **Maintenance:**
   * Schedule regular vacuuming and reindexing to optimize database performance.

**Recovery Strategy**

1. **Testing Backups:**
2. Restore backups in a sandbox environment to validate data integrity.
3. **Recovery Steps:**
   * For logical backups:

psql -U user -d postgres -f postgresBackup.sql

* + For physical backups:
    - Stop the PostgreSQL service.
    - Replace the data directory with the backup copy.
    - Restart the PostgreSQL service.

1. **Recovery Time Objective (RTO):**
   * Aim for a recovery time of under 2 hours.

**Conclusion**

This PostgreSQL backup and recovery strategy ensures data safety, minimizes downtime, and supports seamless recovery. By automating processes and conducting regular testing, this protocol delivers a robust database management solution.