For creating backups we have multiple ways:

**WAL (Write-Ahead Logging)** archiving is PostgreSQL’s native approach for incremental backups. By archiving WAL files, you create a log of all changes since the last full backup, which can be used to recover the database to a specific point in time. which means that after taking a full backup, you can capture only incremental changes using archived WAL files.

The frequency of these incremental backups does not depend on time instead it depends upon the rate of changes in the database if there is a high rate of changes in the database WAL files will fill up and be archived more frequently. The space of wal file is upto 16mb and can be updated as well.

**Step-by-Step Guide to WAL Archiving for Incremental Backups**

#### **1. Enable WAL Archiving in postgresql.conf**

To start using WAL archiving, configure PostgreSQL to archive WAL files. This setup will ensure that each change in the database is recorded in the WAL files, which are archived for incremental backup.

* Open the PostgreSQL configuration file, typically located at /etc/postgresql/17/main/postgresql.conf or similar path.

Edit the following parameters in the file:  
conf  
  
wal\_level = replica # Enables detailed WAL logging necessary for backups

archive\_mode = on # Enables WAL archiving

archive\_command = 'cp %p /path/to/archive/%f' # Command to copy each WAL file to an archive directory

max\_wal\_senders = 3 # Increases WAL senders (optional but useful for larger setups)

* + Explanation of Parameters:
    - wal\_level = replica ensures the WAL files contain sufficient information for restoring backups.
    - archive\_mode = on enables the archiving process.
    - archive\_command specifies where and how to save WAL files. Here, each WAL file is copied to the /path/to/archive directory.
    - max\_wal\_senders = 3 is optional but can improve performance by allowing more WAL files to be processed simultaneously.
* Archive Command Note: Replace /path/to/archive with the directory where you want to store WAL files. Ensure the specified directory exists and has the necessary permissions.

#### 

#### **2. Create the WAL Archive Directory**

Create the directory specified in archive\_command:  
bash  
  
mkdir -p /path/to/archive

Ensure PostgreSQL has the correct permissions to write to this directory:  
bash  
  
chown postgres:postgres /path/to/archive

chmod 700 /path/to/archive

#### **3. Restart PostgreSQL**

Apply the changes by restarting PostgreSQL:  
bash  
  
sudo systemctl restart postgresql

#### **4. Take an Initial Full Backup**

To use WAL files for incremental backups, you first need a full backup as the baseline. This is because WAL files only contain changes made after the full backup.

Use pg\_basebackup to take a full backup:  
bash  
  
pg\_basebackup -D /path/to/backups -Ft -z -P -X stream

* + -D /path/to/backups: Specifies the destination directory for the full backup.
  + -Ft: Saves the backup in tar format (single compressed file).
  + -z: Compresses the backup to save storage space.
  + -P: Displays progress during the backup.
  + -X stream: Includes WAL files needed for consistency within the full backup.
* Scheduling Full Backups: It’s recommended to take a full backup periodically (e.g., weekly) to keep the recovery process manageable.

#### 

#### **5. Automate WAL Archiving for Incremental Changes**

After configuring WAL archiving, PostgreSQL will automatically store incremental changes in the WAL files and save them to the archive directory (/path/to/archive). You can configure this directory to store all WAL files created between full backups, allowing for point-in-time recovery.

* Automate Full Backups with Cron:
  + Schedule a full backup weekly with pg\_basebackup and let WAL archiving capture changes in between.

Example cron job to run a full backup every Sunday at 2 AM:  
bash  
  
0 2 \* \* 0 pg\_basebackup -D /path/to/backups/$(date +\%Y-\%m-\%d) -Ft -z -P -X stream

Edit cron jobs with:  
bash  
  
crontab -e

#### **6. Restoring from WAL Archiving (Incremental Recovery)**

When you need to restore the database to a specific point in time using the full backup and archived WAL files:

1. Restore the Full Backup:

Extract the latest full backup:  
bash  
  
tar -xvf /path/to/backups/full-backup-YYYY-MM-DD.tar -C /var/lib/postgresql/17/main

1. Copy WAL Files to pg\_wal Directory:
   * Copy the WAL files from the archive to the PostgreSQL pg\_wal directory for replay during recovery.

Example command:  
bash  
  
cp /path/to/archive/\* /var/lib/postgresql/17/main/pg\_wal/

1. Start PostgreSQL in Recovery Mode:

Create a recovery.signal file in the PostgreSQL data directory to trigger recovery mode.  
bash  
  
touch /var/lib/postgresql/17/main/recovery.signal

1. Start PostgreSQL:
   * When PostgreSQL starts, it will replay the WAL files to restore the database to the point of failure or a specified time.
2. Stopping Recovery at a Specific Point in Time (Optional):
   * You can stop recovery at a particular point by adding the recovery\_target\_time parameter in postgresql.conf:  
     conf  
       
     recovery\_target\_time = 'YYYY-MM-DD HH:MM:SS'

### **Barman (Backup and Recovery Manager)**

**Barman** (Backup and Recovery Manager) is a tool that manages PostgreSQL backups and provides incremental backup capabilities. Barman automates WAL management but does not require manual WAL archiving.

#### **How Barman Works for Incremental Backups**

1. **Differential and Incremental Backup Support**:
   * **Differential Backups**: Only backups data changed since the last full backup.
   * **Incremental Backups**: Uses WAL files generated during regular operations without requiring traditional WAL archiving.
2. **Setup**:
   * Take a full backup and schedule incremental backups with Barman.

Example full backup:  
bash  
  
barman backup mydb

* + Schedule incremental backups in Barman’s configuration or with cron jobs.

1. **Restore Process**:
   * Barman can restore the latest state by applying the latest full and incremental backups.

Example restore command:  
bash  
  
barman recover mydb latest /path/to/restore

1. **Point-in-Time Recovery**:
   * Barman enables point-in-time recovery using its own management of WAL logs, without requiring WAL archiving.
   * Example for point-in-time recovery:  
     bash  
       
     barman recover mydb "YYYY-MM-DD HH:MM:SS" /path/to/restore