Here's a detailed guide on configuring Point-in-Time Recovery (PITR) in PostgreSQL 17 with a master and standby setup:

**Understanding PITR**

PostgreSQL's Point-in-Time Recovery (PITR) lets you restore your database to a specific point in time. It works by combining:

* **Base backups:** Full copies of your database's data files.
* **Write-Ahead Logs (WAL):** A record of every change made to the database.

By replaying the WAL files from a base backup, you can reconstruct the database's state at any point in time after that backup.

**Steps to Configure PITR**

Here's a breakdown of the configuration process:

**1. Configure WAL Archiving on the Master Server**

* **Edit postgresql.conf:**
  + Locate your PostgreSQL configuration file. (e.g., /etc/postgresql/17/main/postgresql.conf)
  + Modify these parameters:  
    wal\_level = replica # or logical  
    archive\_mode = on  
    archive\_command = 'test ! -f /path/to/wal\_archive/%f && cp %p /path/to/wal\_archive/%f'  
    max\_wal\_senders = 5  
    wal\_keep\_size = 1GB  
    - wal\_level: Set to replica for replication and PITR. If using logical replication, use logical.
    - archive\_mode: Enables WAL archiving.
    - archive\_command: Specifies how WAL files are archived.
      * Replace /path/to/wal\_archive/ with your desired archive location. Make sure this directory exists, is writable by the postgres user, and is on storage you trust.
      * %p is the path to the WAL file.
      * %f is the WAL filename.
      * The test ! -f ... && prevents overwriting existing files.
    - max\_wal\_senders: The maximum number of concurrent connections for replication and backups.
    - wal\_keep\_size: The minimum size of WAL files kept in pg\_wal.
* **Create the Archive Directory:**  
  sudo mkdir -p /path/to/wal\_archive  
  sudo chown postgres:postgres /path/to/wal\_archive  
  sudo chmod 700 /path/to/wal\_archive
* **Restart PostgreSQL:**  
  sudo systemctl restart postgresql@17-main.service # Adjust the service name if needed
* **Verify Archiving:** Check the PostgreSQL logs on the master to confirm WAL files are being archived.

**2. Take a Base Backup**

* Use pg\_basebackup from the standby server (or another client) to create a base backup:  
  pg\_basebackup -h <master\_host> -p 5432 -U <replication\_user> -D /var/lib/postgresql/17/main -R -Fp -Xs -P  
  + <master\_host>: Hostname or IP of the master server.
  + 5432: PostgreSQL port.
  + <replication\_user>: A user with the REPLICATION attribute. Create on the master if needed:  
    CREATE ROLE replication\_user WITH REPLICATION LOGIN PASSWORD 'your\_password';
  + /var/lib/postgresql/17/main: The (empty) data directory on the standby server.
  + -R: Creates standby.signal and postgresql.auto.conf for the standby.
  + -Fp: Plain format.
  + -Xs: Start streaming WAL during the backup.
  + -P: Show progress.

**3. Configure the Standby Server**

* The pg\_basebackup -R command usually sets up the standby correctly with a postgresql.auto.conf file. If you need to do this manually, here's what's needed.
* **Important:** The standby server needs to be able to access the WAL archive. This might be via shared storage (like NFS), or by some other method of copying the files.
* **Check or Create postgresql.auto.conf:** This file, in the standby's data directory, contains the connection settings.
* **Standby Configuration (usually in postgresql.auto.conf):**  
  standby\_mode = on  
  primary\_conninfo = 'host=<master\_host> port=5432 user=<replication\_user> password=your\_password application\_name=standby'  
  restore\_command = 'cp /path/to/wal\_archive/%f %p'  
  trigger\_file = '/tmp/promote'  
  + standby\_mode = on: Designates this server as a standby.
  + primary\_conninfo: Connection string to the master.
  + restore\_command: How the standby retrieves archived WAL files. Crucially, this path (/path/to/wal\_archive/) must be correct *on the standby server*.
  + trigger\_file: A file that, when created, promotes the standby to master (for failover).

**4. Start the Standby Server**

* Start the PostgreSQL service on the standby:  
  sudo systemctl start postgresql@17-main.service

**Testing PITR**

* **Verify Replication:**
  + Check PostgreSQL logs on both servers.
  + On the master, use SELECT \* FROM pg\_stat\_replication; to see the standby's connection.
* **Simulate Failure and Recover:**
  1. **Determine Recovery Point:** Find a time *before* the data loss you want to simulate.
  2. **Stop Standby:** sudo systemctl stop postgresql@17-main.service (on standby).
  3. **Create recovery.conf:** On the *standby* server, in the data directory. If postgresql.auto.conf exists, rename it.  
     standby\_mode = off  
     recovery\_target\_time = '2024-08-28 10:00:00 EDT' # Replace with your target time  
     restore\_command = 'cp /path/to/wal\_archive/%f %p' # Make sure this is correct
  4. **Start in Recovery Mode:** sudo systemctl start postgresql@17-main.service (on standby). The server will recover to the specified time.
  5. **Verify:** Connect to the standby and check the data.
  6. **Promote (if needed):** If the master is down, create the trigger\_file: touch /tmp/promote

**Important Notes**

* **WAL Archive Location:** This is critical. It must be reliable, have enough space, and be accessible during recovery.
* **Security:** Secure your WAL archive. It contains your database's history.
* **Base Backups:** Take regular base backups in addition to WAL archiving. This speeds up recovery.
* **Backup Retention:** Have a policy for how long to keep base backups and WAL files.
* **Testing:** *Test* your recovery process regularly!
* **PostgreSQL 17:** PostgreSQL 17 has improvements, including incremental backups, which can make backups more efficient. Investigate those for a long-term strategy.