# PostgreSQL Backup and Recovery Management using Barman

Barman is a production grade tool for managing the backup and recovery process of PostgreSQL databases. It not only handles the physical backups but also provides automatic management of retention policies, compression, near zero Recovery point objective(RPO) and enables recovery at any desired point(PITR) in time within the recovery window.

Barman's Stream Archiving feature stands out as a key component in achieving zero Recovery Point Objective (RPO). This is achieved by using **pg_recievewal** utility which continuously backs up Write-Ahead Logging (WAL) files in real-time to a designated Barman server. This capability is particularly important for applications where even minimal data loss is unacceptable.

In certain scenarios, configuring Barman involves setting up a standard connection to PostgreSQL for management, coordination, and monitoring purposes, along with a streaming replication connection utilized by both pg_basebackup and pg_receivewal for backup and WAL streaming, respectively. This configuration is known as streaming-only setup in Barman's terminology and there is no need for an SSH connection for backup and archiving operations.

## Question: What is pg_recievewal and why are we using it instead of native archiving?

If we talk about archiving(Archive_command) in PostgreSQL, it only archives the wal files when it is full(16MB). In case of disaster, there is a chance that we can lose up to 16MB worth of committed transactions.

**pg_receivewal** utility establishes a replication connection to PostgreSQL, allowing it to stream Write-Ahead Log data in real-time on disk. While pg_receivewal is actively writing a current wal file, it appends the **".partial"** extension to differentiate it from completed WAL archives. Once the segment is fully written, pg_receivewal proceeds to rename it accordingly.

## Question: What is the mode of replication with pg_receivewal?

**pg_receivewal** utilizes a streaming replication connection which is **asynchronous** by default, so there is a small possibility of losing committed transactions. In cases where such risk is unacceptable, synchronous replication can be used alongside pg_receivewal.

However, this approach has certain trade-offs.

- With synchronous replication, each commit requires a roundtrip to the server where pg_receivewal is running which can decrease the overall throughput of the system.
- If the backup server (where pg_receivewal is running in synchronous mode) experiences an outage, PostgreSQL will stop committing transactions until the backup server is online.

If you want to use synchronous mode with pg_receivewal or physical/logical replication, make sure to go through **synchronous_standby_names** and **synchronous_commit** parameters inside postgressql.conf file.

# Barman in Action

```
We are going to use 2 servers in this example



10.0.1.1 → PostgreSQL Server
```

```
10.0.1.2 → Remote Backup Server for BarmanCopy to Clipboard
```

We are using Ubuntu machines on AWS EC2.

**Note:** It is recommended to deploy Barman on a Remote server.

We are going to work on the following steps:

1. Configure Streaming backups
2. Configure wal archiving using pg_receivewal
3. Take full backup
4. Restore full backup from Barman
5. Recover using PITR from Barman

# PostgreSQL Server Configuration:

## 1: Create a database user for backups

```
Create user barman REPLICATION ENCRYPTED PASSWORD ‘barman’;


GRANT EXECUTE ON FUNCTION pg_start_backup(text, boolean, boolean) to barman;


GRANT EXECUTE ON FUNCTION pg_stop_backup() to barman ;


GRANT EXECUTE ON FUNCTION pg_stop_backup(boolean, boolean) to barman;


GRANT EXECUTE ON FUNCTION pg_switch_wal() to barman ;


GRANT EXECUTE ON FUNCTION pg_create_restore_point(text) to barman ;


GRANT pg_read_all_settings TO barman ;


GRANT pg_read_all_stats TO barman ;


GRANT CONNECT on DATABASE postgres to barman;Copy to Clipboard
```

**Note:** We can also provide superuser privilege to barman user but above privileges are enough for backup user.

## 2: Edit PostgreSQL Configuration Files

### Postgresql.conf

Configure the settings for barman

```
wal_level = replica or logical

max_wal_senders > 3

max_replication_Slots > 3

listen_addresses='*'Copy to Clipboard
```

We will need at least 2 connections(1 for backup and other for wal streaming)

Note that changing the above parameter requires database restart.

### pg_hba.conf

Add the following line inside pg_hba.conf to allow the barman user to fetch wal files and backup using a replication connection. Restrict this connection only to the IP address of the Barman backup server.

```
host    replication     barman  10.0.1.2/32 md5Copy to Clipboard
```

In order to persist above change, reload postgresql server using the following command:

```
select pg_reload_conf();Copy to Clipboard
```

# Barman Server Configuration

## 1: Install Barman Utility

Run the following commands on the Barman Backup server(10.0.1.2) from root user.

```
sudo su - (Switch to root user)



apt-get update



apt-get install -y curl ca-certificates gnupg



curl https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add -



sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'



apt-get update



apt-get -y install barmanCopy to Clipboard
```

**Note:** Here you need to make sure that **pg_basebackup** utility has the same major version as the source PostgreSQL server. Barman will install postgreSQL binaries with the latest stable version as seen below:

```
psql --version

psql (PostgreSQL) 16.2 (Debian 16.2-1.pgdg120+1)Copy to Clipboard
```

For our system, we need to backup PostgreSQL v13. Run the following command to identify the package.

```
sudo apt list postgres* --installed



Listing... Done

postgresql-client-16/bookworm-pgdg,now 16.2-1.pgdg120+1 arm64 [installed,automatic]

postgresql-client-common/bookworm-pgdg,now 257.pgdg120+1 all [installed,automatic]

postgresql-client/bookworm-pgdg,now 16+257.pgdg120+1 all [installed]Copy to Clipboard
```

Remove postgresql-client-16 package and install postgresql 13 one

```
sudo apt remove postgresql-client-16

sudo apt install postgresql-client-13Copy to Clipboard
```

After successful installation, now check psql

```
psql --version

psql (PostgreSQL) 13.14 (Debian 13.14-1.pgdg120+1)Copy to Clipboard
```

It also creates a barman user on the OS. We are going to run all backup operations using a barman user.

```
barman --version

3.9.0 Barman by EnterpriseDB (www.enterprisedb.com)Copy to Clipboard
```

# 2: Configure Barman

Barman employs two types of configuration files

**global/general** configuration is the primary configuration file, typically set to **/etc/barman.conf** by default, encompasses general options such as backup directory, logging, retention policies, and more.

**Server configuration** where Barman utilizes individual configuration files located in the **/etc/barman.d** directory, each suffixed with **.conf**. We can define connections and settings related to streaming backup and archive methodology.

**Note:** Grant ownership to barman user so it can edit files easily.(Optional)

```
sudo chown -R barman:barman /etc/barman*Copy to Clipboard
```

## Barman Server Configuration

Create a configuration file for your database server where you will provide connectivity details and backup modes. We are creating a **pg_server.conf** file.

```
sudo su - barman

vi /etc/barman.d/pg_server.confCopy to Clipboard
```

Add the following details

```
[pg_server]

description = "Streaming Backups using pg_basebackup and pg_recievewal for archiving wal
files"

conninfo = host=10.0.1.1 user=barman dbname=postgres port=5432

streaming_conninfo = host=10.0.1.1 user=barman dbname=postgres port=5432

backup_method = postgres

streaming_archiver = on

slot_name = barman_slot

create_slot = autoCopy to Clipboard
```

Where

**[pg_server]** is a unique identifier for the postgresql server which needs to be backed up.
**Conninfo** is provided in order to connect with the postgres database.
**Streaming_conninfo** is provided to create a replication connection for streaming backups and pg_receivewal.
**backup_method=postgres** shows that we are going to take streaming backups using pg_basebackup.
Another backup method is **rsync** which requires a passwordless SSH connection between barman and PostgreSQL server. This is the only option that supports the incremental backup feature.
**Streaming_archiver = on** shows that we are going to archive wal logs using pg_recievewal utility which will be invoked by the barman server.
**slot_name** shows the replication slot which needs to be created.
**create_slot=auto** means barman will handle the slot creation automatically.

**Note:** backup_method=postgres doesn't provide incremental/differential backup features.

## Barman Global Configuration

The file resides inside **/etc/barman.conf** where you specify the backup parameters such as logging, retention policies, timeouts etc.

We are going to set the following parameters under [barman] identifier.

```
vi /etc/barman.conf

barman_home = /mnt/backups

immediate_checkpoint = true

retention_policy = RECOVERY WINDOW OF 1 WEEK

log_level = INFO

compression = gzip
```

```
basebackup_retry_times = 2

basebackup_retry_sleep = 30Copy to Clipboard
```

Where **barman_home** is your backup directory

## 3: PostgreSQL Connection Password File

In order for Barman to connect via the barman user, we'll need to create a **pgpass** file on the barman server for passwordless authentication.

```
vi .pgpassCopy to Clipboard
```

The generic entry would be

```
ServerIP:database:port:user:passwordCopy to Clipboard
```

We are going to add the following entry

```
10.0.1.1:5432:*:barman:barmanCopy to Clipboard
```

Where * means for all(All databaase connection + Replication)

After setting up pgpass file, provide the following mode

```
chmod 0600 ~/.pgpassCopy to Clipboard
```

Test passwordless connection

```
psql -h 10.0.1.1 -U barman -d postgres -p 5432Copy to Clipboard
```

## 4: Verification

In order to verify the barman connectivity with PostgreSQL server, Run the following commands

To check if the server configuration is valid you can use the barman check command:

```
barman check pg_server



Server pg_server:

WAL archive: FAILED (please make sure WAL shipping is setup)

PostgreSQL: OK

superuser or standard user with backup privileges: OK

PostgreSQL streaming: OK

wal_level: OK

replication slot: OK

directories: OK

retention policy settings: OK

backup maximum age: OK (no last_backup_maximum_age provided)
```

```
backup minimum size: OK (0 B)

wal maximum age: OK (no last_wal_maximum_age provided)

wal size: OK (0 B)

compression settings: OK

failed backups: OK (there are 0 failed backups)

minimum redundancy requirements: FAILED (have 0 backups, expected at least 1)

pg_basebackup: OK

pg_basebackup compatible: OK

pg_basebackup supports tablespaces mapping: OK

systemid coherence: OK (no system Id stored on disk)

pg_receivexlog: OK

pg_receivexlog compatible: OK

receive-wal running: OK

archiver errors: OKCopy to Clipboard
```

For wal-archive error, we need to start pg_recievewal utility for streaming archives

```
barman cronCopy to Clipboard
```

We can ignore minimum redundancy error for now as as there are no backups taken.

## Question: what is the responsibility of barman cron command?

The barman cron performs various tasks such as initiating the WAL receiver when necessary, processing accumulated WALs, and archiving them. It also deletes backups and their associated archives according to retention policies. This process runs in the foreground, and it's advised to schedule the barman cron to execute every minute.

Keep in mind that starting a backup will not be possible if wals are not being archived correctly to Barman, either through the archiver or the streaming_archiver.

Make sure to monitor barman log file using the following command

```
tail -100f /var/log/barman/barman.logCopy to Clipboard
```

For more detail on these commands and their options, refer to Barman Manual.

## Question: How to set retention policies with Barman?

Barman provides support for backup retention policies, allowing users to establish guidelines on the duration of backups and associated archive logs (Write Ahead Log segments) for future recovery needs. Users have the flexibility to define retention policies based on either backup redundancy (the number of periodic backups) or a recovery window (a specific timeframe).

In a redundancy-based retention policy, users indicate the desired number of periodic backups to retain. For example if we need to retain 8 full backups and their wal files, we can define retention like this:

```
retention_policy: REDUNDANCY 8Copy to Clipboard
```

Another option is to define a recovery window(Mostly used) which designates a time period. Barman ensures the retention of backups and archived WAL files necessary for point-in-time recovery up to any moment within the specified window.

```
retention_policy: RECOVERY WINDOW OF 1 WEEKCopy to Clipboard
```

This configuration signifies that Barman will retain backups and archives for one week, enabling recovery to any point in time within the last week.

We can also define a minimum redundancy level for backups, ensuring they are not deleted even when the retention period expires. This can be achieved by configuring the following parameter.

```
minimum_redundancy = 2Copy to Clipboard
```

# Backup using Barman

Now that you have Barman ready, let's create a backup manually.

Run the following command as the barman user on the barman backup server to create your first backup:

```
sudo su - barman

barman backup pg_serverCopy to Clipboard
```

## Barman Directories

Barman creates a directory as per server unique name as shown below.

```
ls -ltr /mnt/backups/

total 24

drwxrwxr-x 7 barman barman  4096 Feb  8 22:20 pg_serverCopy to Clipboard
```

Let's take a look inside pg_server folder

```
ls -ltr /mnt/backups/pg_server

total 20

drwxr-xr-x 2 barman barman 4096 Feb 11 20:37 incoming

drwxr-xr-x 2 barman barman 4096 Feb 11 20:37 errors

drwxr-xr-x 2 barman barman 4096 Feb 11 20:37 base

drwxr-xr-x 3 barman barman 4096 Feb 11 20:41 wals

drwxr-xr-x 2 barman barman 4096 Feb 11 20:46 streaming

Copy to Clipboard
```

The **incoming** directory comes into play when using the rsync method for streaming.

All transaction WAL files will be directed first to the **streaming** folder.

When barman cron command is executed manually or through cron job, wal files from the streaming folder will be compressed and transferred/archived to the **wals** folder.

The **base** directory serves as the repository for physical backups, within which multiple backup timestamps can be found.

# Schedule Backups

Make sure to schedule cron and backups.

```
crontab -e

# m h  dom mon dow   command

* * * * * /usr/bin/barman cron

00 01 * * fri /usr/bin/barman backup pg_serverCopy to Clipboard
```

Barman cron command will run every minute.

We are scheduling backup to run 2 times a week as per 1 week retention policy.

# Barman Commands

## 1: check

To check if the server configuration is valid you can use the **barman check** command:

```
barman check pg_server

Server postgres_server:

PostgreSQL: OK

superuser or standard user with backup privileges: OK

PostgreSQL streaming: OK

wal_level: OK

replication slot: OK

directories: OK

retention policy settings: OK

backup maximum age: OK (no last_backup_maximum_age provided)

backup minimum size: OK (603.5 MiB)

wal maximum age: OK (no last_wal_maximum_age provided)

wal size: OK (210.9 MiB)

compression settings: OK

failed backups: OK (there are 0 failed backups)

minimum redundancy requirements: OK (have 1 backups, expected at least 1)
```

```
pg_basebackup: OK

pg_basebackup compatible: OK

pg_basebackup supports tablespaces mapping: OK

systemid coherence: OK

pg_receivexlog: OK

pg_receivexlog compatible: OK

receive-wal running: OK

archiver errors: OKCopy to Clipboard
```

## 2: List backup

You can list the catalog of available backups for a given server with:

```
barman list-backup pg_server



pg_server 20240215T182843 - Thu Feb 15 18:28:53 2024 - Size: 603.5 MiB - WAL Size: 210.9
MiBCopy to Clipboard
```

Where

**20240215T182843** is unique Backup-ID(Can be found inside base folder)

**603.5 MiB** is the total size of backup

**210.9MB** is the size of wal files created during/after backup. Wal file size will keep increasing as per activity on PostgreSQL server

## 3: show-backup

You can retrieve all the available information for a particular backup of a given server with:

```
barman show-backup servername Backup-ID

barman show-backup pg_server 20240215T182843



Backup 20240215T182843:

Server Name        : pg_server

System Id          : 7335180256203400928

Status             : DONE

PostgreSQL Version : 130014

PGDATA directory   : /mnt/data/database



Base backup information:
```

```
Disk usage       : 603.5 MiB (603.5 MiB with WALs)

Incremental size : 603.5 MiB (-0.00%)

Timeline         : 3

Begin WAL        : 000000030000000000000004D

End WAL          : 000000030000000000000004D

WAL number       : 1

WAL compression ratio: 99.90%

Begin time       : 2024-02-15 18:28:43.742794+00:00

End time         : 2024-02-15 18:28:53.111891+00:00

Copy time        : 9 seconds

Estimated throughput : 64.5 MiB/s

Begin Offset     : 40

End Offset       : 0

Begin LSN        : 0/4D000028

End LSN          : 0/4E000000


WAL information:

No of files      : 65

Disk usage       : 210.9 MiB

WAL rate         : 247.27/hour

Compression ratio : 79.72%

Last available   : 000000030000000000000008E

Reachable timelines  : 4


Catalog information:

Retention Policy : VALID

Previous Backup  : - (this is the oldest base backup)

Next Backup      : - (this is the latest base backup)Copy to Clipboard
```

## 4: show-servers

You can show the configuration parameters for a given server with:

```
barman show-servers <server_name>

barman show-servers pg_server
```

```
Server pg_server:

active: True

archive_timeout: 0

archiver: False

archiver_batch_size: 0

autogenerate_manifest: False

aws_profile: None

aws_region: None

azure_credential: None

azure_resource_group: None

azure_subscription_id: None

.

.

.

wal_level: replica

wal_retention_policy: MAIN

wals_directory: /backups/cluster/wals

xlog_segment_size: 16777216

xlogpos: 4/B1C09F8Copy to Clipboard
```

## 5: replication-status

The replication-status command reports the status of any streaming client currently attached to the PostgreSQL server, including the receive-wal process of your Barman server (if configured).

You can execute the command as follows:

```
barman replication-status <server_name>

barman replication-status pg_server

Status of streaming clients for server 'cluster':

Current LSN on master: 4/B1C09F8

Number of streaming clients: 2Copy to Clipboard


1: Async WAL streamerApplication name: barman_receive_wal

Sync stage  : 3/3 Remote write

Communication   : TCP/IP

IP Address  : 172.31.85.163 / Port: 60992 / Host: -
```

```
User name    : barman

Current state    : streaming (async)

Replication slot: barman_repslot

WAL sender PID  : 1241

Started at  : 2024-02-19 21:38:02.854754+00:00

Sent LSN    : 4/B1C09F8 (diff: 0 B)

Write LSN  : 4/B1C09F8 (diff: 0 B)

Flush LSN  : 4/B000000 (diff: -1.8 MiB)



2: Async standbyApplication name: pgreplica1

Sync stage  : 5/5 Hot standby (max)

Communication    : TCP/IP

IP Address  : 172.31.90.152 / Port: 58064 / Host: -

User name    : repmgr

Current state    : streaming (async)

Replication slot: slot_2

WAL sender PID  : 1243

Started at  : 2024-02-19 21:38:25.812454+00:00

Sent LSN    : 4/B1C09F8 (diff: 0 B)

Write LSN  : 4/B1C09F8 (diff: 0 B)

Flush LSN  : 4/B1C09F8 (diff: 0 B)

Replay LSN : 4/B1C09F8 (diff: 0 B)Copy to Clipboard
```

## 6: delete

You can delete a given backup manually with this command:

```
barman delete <server_name> <backup_id>Copy to Clipboard
```

# Recover using Barman

Remote recovery is definitely the most common way to restore a PostgreSQL server with Barman.

## Pre-requisites

1. Both systems should have the same hardware architecture.
2. PostgreSQL major versions on both systems should match.
3. Synchronize the time between the Barman and PostgreSQL servers.
4. Establish bi-directional passwordless SSH connectivity between the Barman and PostgreSQL servers intended for restoration.

5. It’s recommended to install the barman-cli package on all PostgreSQL servers, whether primary or standby. This can be achieved using the command:

```
apt-get install barman-cliCopy to Clipboard
```

6.

7. SSH connections between Barman and the remote host should utilize public key exchange authentication.
8. The remote user needs permissions to create the backup directory structure in the designated destination and ensure space should be available for storing both the base backup and necessary WAL files for recovery on the remote server.

## Question: Why do we need barman-cli?

barman-cli contains barman-wal-restore utility which can be used for

- Standby recovery if it is out of sync.
- Point in time recovery.

## Full recovery(Backups only)

First, select a backup to restore

```
barman list-backup pg_server

postgres_server 20240210T000102 - Sat Feb 10 00:07:59 2024 - Size: 23.9 GiB - WAL Size: 0 B

postgres_server 20240209T232959 - Fri Feb  9 23:33:57 2024 - Size: 23.0 GiB - WAL Size: 578.5
MiB

postgres_server 20240209T203252 - Fri Feb  9 20:34:24 2024 - Size: 7.2 GiB - WAL Size: 9.4 GiB

postgres_server 20240208T223050 - Thu Feb  8 22:31:27 2024 - Size: 3.6 GiB - WAL Size: 2.7
GiBCopy to Clipboard
```

We are going to restore the latest backup

**pg_server 20240210T000102**

### Recovery command:

You can run the following command to copy backup to remote server

```
barman recover --remote-ssh-command "ssh postgres@10.0.1.1" postgres_server 20240210T000102
/mnt/data/postgresql/13/main



Starting remote restore for server pg_server using backup 20240210T000102

Destination directory: /mnt/data/postgresql/13/main

Remote command: ssh postgres@10.0.1.1

Copying the base backup.Copy to Clipboard
```

Once you recover, make sure to check configuration files especially **postgresql.auto.conf** and restart the server. Note that this will only restore backup and wal files created during backup.

## Full recovery(Backups + end of wal files)

Incase you want to restore till the end point of wal files, you need to run the following command

### Recovery command:

```
barman recover --remote-ssh-command "ssh postgres@172.31.30.185" pg_server 20240208T223050
/mnt/data/postgresql/13/main --get-wal



WARNING: 'get-wal' is in the specified 'recovery_options'.

Before you start up the PostgreSQL server, please review the postgresql.auto.conf file

inside the target directory. Make sure that 'restore_command' can be executed by the
PostgreSQL user.

Recovery completed (start time: 2024-02-10 21:20:36.786116+00:00, elapsed time: 49 seconds)

Your PostgreSQL server has been successfully prepared for recovery!Copy to Clipboard
```

The **–get-wal** option enables users to request any WAL file from the Barman location. Barman configures the recovery by establishing a **restore_command** using the **barman-wal-restore** script.

**barman-wal-restore** offers various useful functionalities, including automatic compression and decompression of WAL files and the peek feature, which allows retrieval of subsequent WAL files while PostgreSQL is applying one. This optimizes bandwidth usage between PostgreSQL and Barman.

Check **restore_command** inside **postgresql.auto.conf**

```
restore_command = 'barman-wal-restore -P -U barman ip-172-31-85-163.ec2.internal
postgres_server %f %p'Copy to Clipboard
```

By utilizing the **-P** option, it is possible to retrieve the content of the current **.partial WAL file**.

Set the restore command properly:

```
restore_command = 'barman-wal-restore -P -U barman 172.31.85.163 pg_server %f %p'Copy to
Clipboard
```

Now start postgresql server

```
/usr/lib/postgresql/13/bin/pg_ctl -D /mnt/data/postgresql/13/main startCopy to Clipboard
```

It will recover to a consistent point and then apply all available files from barman.

## Point in time recovery

For PITR, all above steps will be same but we we need to change our recovery command and add **target time** flag

### Recovery command:

```
barman recover --remote-ssh-command "ssh postgres@10.0.1.1"  --target-time="2024-02-10
21:16:17" postgres_server 20240208T223050 /mnt/data/postgresql/14/main --get-wal
```

```
Starting remote restore for server postgres_server using backup 20240208T223050

Destination directory: /mnt/data/postgresql/14/main

Remote command: ssh postgres@172.31.30.185

Doing PITR. Recovery target time: '2024-02-10 21:16:17+00:00'

Using safe horizon time for smart rsync copy: 2024-02-08 22:30:50.190930+00:00

Copying the base backup.Copy to Clipboard
```

Check **restore_command** and **recovery_target_time** inside postgresql.auto.conf

```
restore_command = 'barman-wal-restore -P -U barman ip-172-31-85-163.ec2.internal
postgres_server %f %p

Recovery_target_time = ‘2024-02-10 21:16:17’Copy to Clipboard
```

Configure the restore command and start the server. It will proceed to apply all WAL files until 2024-02-10 21:16:17. Upon completion of the recovery process, it will pause recovery and we need to run the following command to enable writes:

```
Select pg_wal_replay_resume();Copy to Clipboard
```

**Note:** barman-wal-restore utility also provides **-p(parallel)** option to speedup wal fetch process. Check out other options using **barman-wal-restore –-help**

## Question: How to redirect barman to a new server incase of a failover?

In the event of a cluster setup and a failover occurrence, the procedure to redirect Barman to a new server involves updating the IP address within the /**etc/barman.d/pg_server.conf** file. Modify the **conninfo** and **streaming_conninfo** parameters with the new server’s details, then execute the command **barman cron** to initiate a pg_receivewal process from the new master. It’s advisable to perform a full base backup after a promotion or recovery.

Barman v3.10(Latest) provides **config-switch** option for cluster configurations. Replica information should be provided like this

```
[pg_server]

description =  "Streaming Backups with pg-recievewal for archiving"

conninfo = host=10.0.1.1 user=barman dbname=postgres port=5432

streaming_conninfo = host=10.0.1.1 user=barman dbname=postgres port=5432

backup_method = postgres

streaming_archiver = on

slot_name = barman_repslot

create_slot = auto



[pg_server:pgreplica1]

description =  "Streaming Backups with pg-recievewal for archiving"
```

```
conninfo = host=10.0.1.3 user=barman dbname=postgres port=5432

streaming_conninfo = host=10.0.1.3 user=barman dbname=postgres port=5432

model=true

backup_method = postgres

streaming_archiver = on

slot_name = barman_repslot

create_slot = autoCopy to Clipboard
```

You can see that the main server name is attached to the pgreplica1 server like this

```
[pg_server:pgreplica1]Copy to Clipboard
```

Now Incase of failover, you need to switch the server by running this command:

```
barman config-switch pg_server pg_server:pgreplica2Copy to Clipboard
```

That will override the cluster configuration options with the values defined in the selected model.

## Question: How to stop the barman server?

1: Remove all entries from crontab. Barman installation also put the crontab entry automatically here:

```
cat /etc/cron.d/barman



# /etc/cron.d/barman: crontab entries for the barman package

MAILTO=root

* * * * * barman [ -x /usr/bin/barman ] && /usr/bin/barman -q cron

Copy to Clipboard
```

2: Run pg_receivewal stop to stop archiving. This can be done using following command:

```
barman receive-wal --stop pg_serverCopy to Clipboard
```

3: Remove replication slot

```
barman receive-wal --drop-slot pg_serverCopy to Clipboard
```

## Conclusion

Setting up a regular backup schedule that matches the organization's needs for how quickly they can recover data(RTO) and how much data they're okay with losing(RPO) is key. Barman offers flexible options for how long backups are kept. It's also important to store backup data in different places and test them periodically to protect against disasters.