



EDB

Moving to New Backup Tool: Barman

Release 8.1.0

EnterpriseDB

Aug 31, 2021

CONTENTS

1	Choosing your Backup Tool	3
1.1	Barman	3
1.2	pgBackRest	3
1.3	Tool Comparison	3
2	Configuring the New Tool	5
2.1	How Features are Configured	5
2.1.1	Postgres connection details	5
2.1.2	Use of direct file copy mode (ssh or rsync) instead of pg_basebackup	6
2.1.3	Incremental backups	6
2.1.4	WAL archive compression**	6
2.1.5	Backup compression	6
2.1.6	Parallel backup and restore	7
2.1.7	Retention Policy	7
2.1.8	Configure repository location	7
3	Planning the Migration	9
3.1	Testing and Verification	9
3.2	Sizing Considerations	9
3.3	Resource Availability and Timeline Communication	10
4	Executing the Migration	11
5	Maintaining and Cleaning Up	13
6	Scheduling	15
6.1	Retention management	16

Barman and pgBackRest are now EDB's recommended backup and data recovery tools. If you are currently using BART as your backup and recovery tool, EDB recommends that you move to one of these tools. Typically Barman is the right choice unless you have performance requirements for backing up large, multi-terabyte databases. EDB continues to support BART at present, but won't be adding any new features or enhancements. EDB will announce when BART support ends with enough lead time for you to move off of BART as your backup tool. BART will not be supported for PostgreSQL or EDB Postgres Advanced Server versions v14 and above.

For more information about EDB's backup and recovery tool options see, the [Backup and Recovery](#) on the EDB web site.

This documentation describes how to choose which tool is right for your environment, how to migrate from BART to your tool of choice, and how to configure your tool of choice based on the configuration options you used in BART.

Which tool is the right tool for your environment very much depends on your specific needs. Should you still have questions after reading this documentation, or require further assistance, please reach out to us through the following channels

- 'Customer Support Email: techsupport@enterprisedb.com'
- 'Trial Use Assistance: trial-help@enterprisedb.com'
- 'Phone:'
 - US +1-732-331-1320 / 1-800-235-5891
 - UK +44-2033719820
 - Brazil +55-2139581371
 - India +91-20-66449612

CHOOSING YOUR BACKUP TOOL

This section provides information on the two alternative back up tools.

1.1 Barman

EDB recommends BART users move to barman and plans to add the following capabilities soon

- Integration with PEM
- Integration with other backup solutions
- An API for implementing other integrations

Besides the benefits of choosing the tool that EDB is investing in, Barman is the right choice if you are doing remote backups using the PostgreSQL port and you are not using passwordless ssh. See the [Barman documentation](#) for more information.

1.2 pgBackRest

pgBackRest's advantages include

- Performance benefit for backing up large (multi-terabyte) databases. The internal algorithm for detecting file changes in delta backups adds less overhead, and partial restore allows to restore only one database from a large multi database cluster.
- Supplies backup compression, which is a planned feature for Barman.

See the [pgBackRest documentation](#) for more information.

1.3 Tool Comparison

The following selection matrix differentiates between the tools on a generic level

Capability	Barman	pgBackRest
PostgreSQL protocol	Yes	•
RPO=0	Yes	•
Rate limiting	Yes	•
Custom WAL sizes	Yes	Yes (v11+)
Backup compression	Coming soon	Yes
Partial restore (only selected databases)	•	Yes
PEM integration	Coming soon	•
License	GPLv3	MIT

CONFIGURING THE NEW TOOL

This section draws a comparison between the three tools and shows how they differ in configuration (config file), and in scheduling. During a migration it is important to be aware of all of these differences, and apply changes as necessary.

2.1 How Features are Configured

The following sections describes how to enable features in the different tools so you can easily compare the implementation details .

- *Postgres connection details*
- *Use of direct file copy mode (ssh or rsync) instead of pg_basebackup*
- *Incremental backups*
- *WAL archive compression*
- *Backup compression*
- *Parallel backup and restore*
- *Retention Policy*
- *Configure repository location*

2.1.1 Postgres connection details

Tool	Configuration
BART	In the [SERVER] section of the config file: [example]...host = localhostuser = postgresport = 5432
Barman	In the config file: [example]...con ninfo = host=localhost user=postgres dbname=postgres
pg Back- Rest[^*]	In the config file: [example]...p gl-host = localhostpg1-path = /var/lib/postgre s/datapg1-user = postgresport = 5432

Specify pg1-path when pgBackRest is running locally, and specify pg1-host when pgBackRest is running remote

2.1.2 Use of direct file copy mode (ssh or rsync) instead of pg_basebackup

Tool	Configuration
BART	Increase threads >1 in the [SERVER] or [BART] section of the config file: <code>[{}example]...thread_count = 2</code> Or, as a commandline option <code>--no-pg_basebackup</code>
Barman	In the config file: <code>[example]...backup_method = rsync</code>
pgBackRest	pgBackRest does not support pg_basebackup option

2.1.3 Incremental backups

Tool	Configuration
BART	Retrieve the backup_id or backup_name for the parent: <code>bart SHOW-BACKUPS example</code> Now supply the backup_id or backup_name as parent: <code>bart BACKUP -s example --parent { backup_id \ backup_name }</code>
Barman	In the config file: <code>[{}example]...reuse_backup = link</code> Or, as a command line option: <code>--reuse-backup=link</code>
pgBackRest	<code>pgbackrest --stanza=example --type=incr backup</code>

2.1.4 WAL archive compression**

Tool	Configuration
BART	In the [SERVER] or [BART] section of the config file: <code>[example]...wal_compression = enabled</code>
Barman	In the config file: <code>[{}example]...compression = gzip</code>
pgBackRest	Global compress settings. Can be overloaded for [global:archive-push]

2.1.5 Backup compression

Tool	Configuration
BART	Command line option: Enable: <code>--gzipLevel: --compress-level</code>
Barman	N/A
pgBackRest	<code>compress=ycompress-level=9 compress-type=gzcompress-level-network=3</code>

2.1.6 Parallel backup and restore

Tool	Configuration
BART	Set <code>thread_count</code> in the [SERVER] or [BART] section of the config file: <code>[example]...thread_count = 4</code> Or, as a Command line option <code>--thread-count=4</code>
Barman	Set <code>parallel_jobs</code> in the configfile: <code>[example]...parallel_jobs = 4</code> Or, as a command line option <code>--jobs 4</code>
pgBackRest	Set <code>process-max</code> in the configfile: <code>[example]...process-max = 4</code> Or, as a command line option <code>--process-max=4</code>

2.1.7 Retention Policy

Tool	Configuration
BART	Set <code>retention_policy</code> in the [SERVER] or [BART] section of the config file: <code>{[example]}...retention_policy = 2 BACKUPS # Or DAYS, WEEKS, or MONTHS</code>
Barman	Set <code>retention_policy</code> , and/or <code>wal_retention_policy</code> options in the config file: <code>{[example]}...retention_policy = REDUNDANCY 2 ``\ or\ ``retention_policy = RECOVERY WINDOW OF 2 DAYS # WEEKS/MONTHS.</code>
pgBackRest	Can be set in the config file: <code>{[example]}...repo-retention-full-type = count #can also be time ``\ \ ``repo-retention-full = 2repo-retention-diff = 6# Retain WAL archives for only 1 full back up:repo-retention-archive-type = full #can also be diff or incrrepo-retention-archive = 1See Retention Policy in edb documentation for more information.</code>

2.1.8 Configure repository location

Tool	Configuration
BART	In the config file in the [BART] section: <code>[BART]...backup_path = /tmp/bart</code>
Barman	In the config file in the main chapter: <code>backup_directory = /tmp/barman</code>
pgBackRest	<code>[global]repo1-path=/var/lib/pgbackrest</code>

PLANNING THE MIGRATION

Review this section for items EDB recommends you include in your migration plan.

3.1 Testing and Verification

Every organization will have different policies and capabilities which define what should be done in which environment. Some organizations restore a production database after migration to a temporary system in the test environment, where others rather test the process in a test environment, and expect their production environment to behave the same as their test environment. Whatever best suits your organization, make sure all of the following are updated and verified

- Deployment and management automation
- Processes for restoring a database
- Integrations such as integrations with the backup infrastructure

EDB advises to set up a temporary test system, specifically for onboarding the new tool, so that all of these actions can take place without impacting the existing environments. Make sure your test plan includes standing up the test system, planning for all actions that should take place on this test system, and cleaning up the system after migration.

Some organizations also require verifying a successful restore for every database that has been migrated to the new tool.

3.2 Sizing Considerations

Depending on the actual migration, different sizing considerations apply. Two distinct migration paths exist

1. New storage will be attached for the new backups.

In this case, a good starting point would be to size the storage equally to the current sizing requirements. Note this might be a good opportunity to scale down oversized storage locations. Optionally, consider running both backup tools simultaneously. If the impact is acceptable, it can be an option for rollback and extra security for successful backups. EDB recommends this approach as it is the most straightforward and least error prone.

2. The mount point for the existing backups is reused for the backups.

An important note to make is that the repositories for the different tools have different layouts, and are not interchangeable. That means that the exact location needs to be distinct. But they can exist on the same mount point. The upside is that the extra required storage can be expected to be less than when attaching new storage. That being said, during migration extra storage is still required, and downscaling might not be an option. Furthermore, this option leaves no room to run both backup tools simultaneously. Extra sizing requirements would depend on your exact backup scheme, maintenance schema, and the size difference for differential backups. A good starting point would be to prepare for an

extra set of backups (one full and all differentials), and an extra full backup. Make sure that the monitoring thresholds are also properly adjusted as required.

Note that it is crucial that you make sure that compression options are configured similarly between the old and new tool. Alternatively, extra storage space is required to compensate for the backup and/or WAL size.

During the planning phase the following needs to be taken care of:

- Select the approach.
- Size the expected extra storage and check for availability.
- Plan to attach or increase the extra storage as part of the migration plan.
- Next to the expected extra storage, also make sure that during the migration extra storage space is directly available should it be required due to unforeseen circumstances.
- When using the first approach (extra mount point), plan to clean out the old storage after migration. Make sure that the old repositories are preserved long enough to meet your organization's rollback policies.

3.3 Resource Availability and Timeline Communication

Depending on the size of your environment, the number of available DBAs to run the migrations, and the amount of automation, the migration can take a considerable amount of time. During the migration your organization may be stressed with some parts of the environment already running with the new tool, extra storage requirements being identified, and DBA resources focused on the migration and less available for other tasks.

It is therefore crucial to plan the migration per environment and derive the duration of the migration from that plan. Communicate these timelines to the rest of the organization so that they are aware that the DBA team is extra occupied during this period of time. Furthermore, make sure that the storage team is available to prepare for and attach the extra storage as required.

EXECUTING THE MIGRATION

Execution of the migration can be very straight forward. The steps include

1. Adding extra storage capacity to the backup server
 - As a new mount point when migrating to a new mount point (or Cloud Storage endpoints).
 - As extra storage on the existing mount point when reusing the same mount point.
2. Installing the new tool on the backup server.
3. Configuring the configuration file for the new tool. See *How features are configured* for more information on changing a configuration file for one tool to another tool.
4. Reconfiguring the scheduling. See *Scheduling* for more information on changing the scheduling from one tool to another tool.
5. Checking the configuration
 - `barman check <example>`
 - `pgbackrest --stanza=<example> --log-level-console=info check` Fix any issues.
6. Running an initial backup manually, before relying on the scheduling (recommended). The check commands will recover the most common issues, but there could always still be unforeseen issues, like storage space issues, IO or network latency issues, etc. It would be unfortunate if the scheduled backups fail since that would increase risk for unsuccessful restores. As your team runs more migrations, they will gain more confidence to skip this step as required.
7. Some organizations might prefer to actually restore and recover the backup (on a separate system), and check for the expected data. This is only an optional extra step, which greatly depends on your organization's policies.

MAINTAINING AND CLEANING UP

A transition period is required after the migration. This is because

- With an empty repository there will be more situations where a full backup is created, where in a normal situation a differential backup would be taken.
- The maintenance process of cleaning old backups kicks in only after exceeding the retention period.

During this transition period, extra care needs to be taken on the backups to make sure that any backup issue is identified and fixed. Options include

- Increasing monitoring notification levels and decreasing alert thresholds for free space.
- Planning manual checks of the backups during the migration.

EDB recommends keeping the original backup solution around during this transition period. You may want to keep the systems around for a longer period of time as an extra safety measure or extra rollback option. When the new backup system is fully operational, and the extra rollback time has exceeded, the old systems can be deprovisioned by

- Removing the original tool from the system.
- Deprovisioning the old mount point and storage after the new mount point is attached for the new storage repository.
- Deprovisioning the old servers If the new tool runs on new servers.

SCHEDULING

Since BART, Barman, and pgBackRest all schedule the backups using cron, changing the scheduling simply requires changing the scheduled commands. Example commands for BART, Barman, and pgBackRest

1. Run a full backup for cluster 'example'

Tool	Command
BART	<code>bart BACKUP -s example</code>
Barman	<code>barman backup example</code>
pgBackRest	<code>pgbackrest backup --type=full --stanza=example</code>

2. Run a full backup for all configured servers

Tool	Command
BART	<code>bart BACKUP -s all</code>
Barman	<code>barman backup all</code>
pgBackRest	pgbackrest does not have an option to run for all stanzas with one command

3. Run an incremental backup

Tool	Command
BART	Retrieve the backup_id/backup_name for the parent (select a full for an incremental, select another inc/diff for a differential backup plan): <code>bart SHOW-BACKUPS example</code> Now supply the backup_id/backup_name as parent: <code>bart BACKUP -s example --parent { backup_id backup_name }</code>
Barman	<code>barman backup example --reuse-backup=link</code> This can also be set in the configuration file at the global/server level with <code>reuse_backup = link</code>
pgBackRest	Incremental: <code>pgbackrest backup --type=incr --stanza=example</code> Differential: <code>pgbackrest backup --type=diff --stanza=example</code>

6.1 Retention management

- With BART usually a cron job is set up to run maintenance on the BART repository (bart MANAGE). By supplying the -d option, the obsoleted backups are automatically cleaned (together with their WAL archives).
- Similarly, Barman uses a cron command to run maintenance on the Barman repository ``barman cron``. The ``barman cron`` command takes care of more things (like copying streamed WAL files to the WAL archive directory), and needs to be scheduled to run every minute. The Barman rpm or debian package automatically creates a cron entry running every minute as the barman user.
- pgBackRest runs maintenance with the expire command (pgbackrest expire), but the expire command is run automatically after each successful backup, and is not required to be separately scheduled.

While migrating to a new tool, make sure that cron is reconfigured to run the proper retention management commands.