2ndQuadrant
Professional PostgreSQL

2ndQuadrant
BaRMan

15 September 2011

Table of Contents

Backup and Recovery Manager for PostgreSQL

Introduction

In a perfect world, there would not be the need for a backup. However, the unexpected is always upon us. And it is important, especially in business environments, to be prepared for when the "unexpected" happens. In a database scenario, the "unexpected" could be any of the following:

- data corruption
- system failure, including hardware failures
- human errors

In these cases, any ICT manager or DBA should be able to repair from the incident and recover in the shortest time possible. We normally refer to this discipline as **Disaster recovery**.

This guide assumes you are familiar with theoretical disaster recovery concepts and you have a grasp of PostgreSQL fundamentals in terms of physical backup and disaster recovery. If not, we encourage you to read the PostgreSQL documentation or any of the recommended books on PostgreSQL.

Professional training on this topic is another effective way of learning these concepts. There are many courses available all year round all over the world, delivered by many PostgreSQL companies, including our company, 2ndQuadrant.

For now, it is important to know that any PostgreSQL physical backup will be made up of:

- a base backup
- one or more WAL files (usually collected through continuous archiving)

PostgreSQL offers the core primitives in order to allow DBAs to setup a really robust Disaster Recovery environment. However, it is not that easy to manage multiple backups, from one or more PostgreSQL servers. Restore is another topic that any PostgreSQL DBA would love to see more automated and user friendly. Other commercial vendor have this kind of applications.

With these goals in mind, 2ndQuadrant started the development of BaRMan, which stands for "Backup and Recovery Manager" for PostgreSQL. Currently BaRMan works on Linux and Unix systems only.

Before you start

The first step is to decide the architecture of your backup. In a simple scenario, you have one **PostgreSQL instance** (server) running on a host. You want your data continuously backed up to another server, called the **backup server**.

BaRMan allows you to launch PostgreSQL backups directly from the backup server, using SSH connections. Another key feature of BaRMan is that it allows you to centralise your backups in case you have more than one PostgreSQL servers to manage.

During this guide, we will assume that:

- you have one PostgreSQL instance on an host (called `pg` for the sake of simplicity)
- one backup server on another host (called `backup`)
- communication between the two servers via SSH is enabled
- the PostgreSQL server can be reached from the backup server as `postgres` user (or another *superuser*)

It is important to note that, for disaster recovery, these two servers should not share any physical resource but the network. You can use BaRMan in geographical redundancy scenarios for better disaster recovery outcomes.

TODO: Plan your backup policy and workflow (with version 0.3).

System requirements

- Linux/Unix (what about Windows?)
- Python 2.6 or higher (recommended: with distribute module)
- The Python development package (the name varies between distributions)
- PsycoPG 2
- python-dateutil < 2.0 (as 2.0 version only supports python3)
- PostgreSQL >= 8.4
- rsync >= 3.0.4

> **Important**
>
> PostgreSQL versions on both servers should be exactly the same.

Installation

Create a system user called `barman` on the `backup` server. As `barman` user, download the sources and uncompress them.

For system wide installation you can type:

```
barman@backup$ ./setup.py build
barman@backup# ./setup.py install # run this command with root privileges
```

For local installation, type:

```
barman@backup$ ./setup.py install --user
```

> **Important**
>
> The `--user` option works only with `python-distribute`

This will install `barman` in your user directory (make sure to properly set your PATH environment variable).

Getting started

Pre-Requisites

SSH connection

You need SSH communication between your `barman` user and the `postgres` user on the `pg` server. Generate an SSH key with an empty password and append your public key in the `authorized_keys` file of the `postgres` user on the `pg` server.

You should now be able to perform this operation as `barman` from the `backup` server:

```
barman@backup$ ssh postgres@pg
```

Now perform the same operation in order to allow the `postgres` user to connect to `backup` as `barman` user.

```
postgres@pg$ ssh barman@backup
```

For further information, refer to SSH documentation.

PostgreSQL connection

You then need to make sure that connection to PostgreSQL as superuser (`postgres`) is granted from the backup server. You can setup your favourite client authentication method between the ones PostgreSQL offers you. More information can be found here: http://www.postgresql.org/docs/current/static/client-authentication.html

```
barman@backup$ psql -c 'SELECT version()' -U postgres -h pg
```

Backup directory

You need to have a main backup directory for storing all your backups done with `barman`. Even though `barman` allows you to define different folders for every server you want to back up and for every type of resource (backup or WALs for instance), we suggest that you use the default rules and stick with the conventions that BaRMan chooses for you.

You will see the configuration file (as explained below) allows you to define a `barman_home` variable, which is the directory where BaRMan will store all your backups by default. The home directory for BaRMan is `/srv/barman`.

```
barman@backup$ sudo mkdir /srv/barman
barman@backup$ sudo chown barman:barman /srv/barman
```

Important

We assume you have enough space and already thought about redundancy and safety of your disks.

Basic configuration

In the `docs` directory you will find a minimal configuration file. Use that as a base and copy it as `/etc/barman.conf` in your system (for a local installation you can save it as `~/.barman.conf`).

The configuration file uses a standard INI format and it is split in:

- a section for general configuration (identified by the `barman` label)
- a section for any PostgreSQL server to be backed up (identified by the server label, e.g. `main` or `pg`)

```
[barman]
; Main directory
barman_home = /srv/barman

; Log location
log_file = %(barman_home)s/log/barman.log

; Default compression level: none, bzip2, gz
compression_filter = None
decompression_filter = None

; 'main' PostgreSQL Server configuration
[main]
; Human readble description
description =  "Main PostgreSQL Database"

; SSH options
ssh_command = ssh postgres@pg

; PostgreSQL connection information (DSN)
conninfo = host=pg user=postgres
```

You can now test the configuration of BaRMan by executing:

```
barman@backup$ barman server show main
```

```
barman@backup$ barman server check main
```

Write down the `incoming_wals_directory` (printed by the `barman server show main` command) as you will need it to setup continuous WAL archiving.

Continuous WAL archiving

Edit the `postgresql.conf` file of the PostgreSQL instance on the `pg` database and activate the archive mode:

```
archive_mode = on
archive_command = 'rsync %p barman@backup:${incoming_wals_directory}/%f'
```

`barman server show main` command above.

Restart the PostgreSQL server.

In order to test that continuous archiving is on and properly working, you need to check both the PostgreSQL server [1] and the `backup` server (in particular that the WAL files are collected in the destination directory).

Listing the servers

```
barman@backup$ barman list
```

Executing a full backup

```
barman@backup$ barman server backup main
```

Viewing the list of backups for a server

```
barman@backup$ barman server list main
```

which returns something similar to:

```
master - 20110919T172439 - Mon Oct 17 12:53:19 2011 - Size: 21.0 MiB - WAL Size: 0 B
```

Where `20110919T172439` is the ID of the backup and `Mon Oct 17 12:53:19 2011` is the start time of the operation, `Size` is the size of the base backup and `WAL Size` is the size of WAL files archived.

Restoring a whole server

The command to recover a whole server is:

```
barman@backup$ barman server recover main 20110920T185953 /path/to/recover/directory
```

Where `20110920T185953` is the ID of the backup to restore. When this command finish succesfully, `/path/to/recover/directory` contains a complete data directory ready to be started as a PostgreSQL database cluster.

An example command to start the server is:

```
barman@backup$ pg_ctl -D /path/to/recover/directory start
```

> **Important**
>
> If you run this command as user `barman`, it will become the database superuser.

You can retrieve a list of backup IDs for a specific server with:

[1] For more information, refer to the PostgreSQL guide

```
barman server list srvpgsql
```

Restoring to a point in time

TODO

Available commands

Barman allows you to specify commands at three different stages:

- global: commands on the local backup catalog
- server: commands for a specific server (list available backups, execute a backup, etc.)
- specific backup: commands for a specific backup in the catalog (display information or issue a recovery, delete the backup, etc.)

The following sections will thoroughly describe the available commands, section per section.

General commands

- Display a list of server configured for backup:

```
barman list
```

- Performs maintenance operations, like compressing WAL files and moving them from the `Incoming` directory to the right one

```
barman cron
```

Note

This command should be executed in a *cron script*. In next version of BaRMan, it will manage the retetion policy feature.

Server commands

- Show all configuration parameters for the specified server

```
barman server show <server_name>
```

- Perform a full backup for the given server

```
barman server backup <server_name>
```

- Display available backups for the given server

```
barman server list <server_name>
```

- Check if connection settings work properly for the specified server

```
barman server check <server_name>
```

Backup commands

Note

Remember: a backup id can be retrieved with `server list main`

- Show information for a specific backup

```
barman backup show <server_name> <backup_id>
```

- Delete a backup

```
barman backup delete <server_name> <backup_id>
```

Advanced configuration

TODO

Support and sponsor opportunities

Barman is free software and it is written and maintained by 2ndQuadrant. If you need support on Barman or need new features, please get in touch with 2ndQuadrant. You can sponsor the development of new features of Barman and PostgreSQL which will be made publicly available as open source.

Authors

2ndQuadrant website: http://www.2ndquadrant.it/

- Marco Nenciarini <marco.nenciarini@2ndquadrant.it>
- Gabriele Bartolini <gabriele.bartolini@2ndquadrant.it>

License

Barman is the property of 2ndQuadrant and its code is distributed under GNU General Public License 3.