horizontal line

**Decathlon**

Lan Song Road NO. 555

Barman work manual

**22th October 2018**

# OVERVIEW

Barman (Backup and Recovery Manager) is an open-source administration tool for disaster recovery of PostgreSQL servers written in Python. It allows your organisation to perform remote backups of multiple servers in business critical environments to reduce risk and help DBAs during the recovery phase.

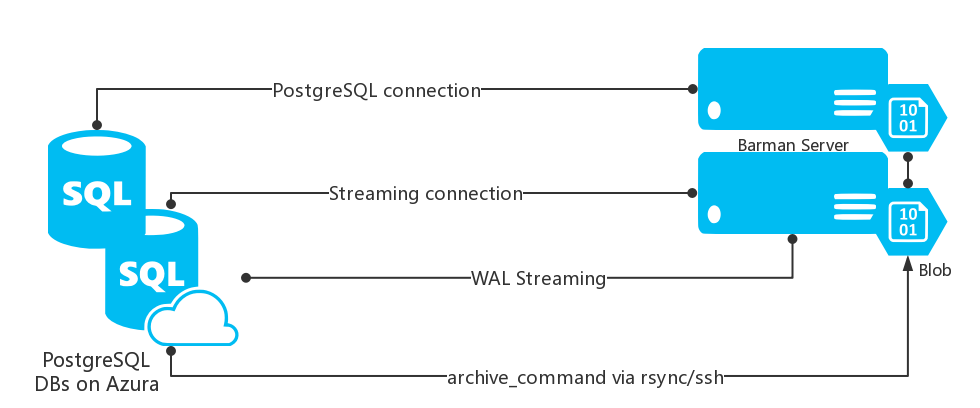
# MISSION WITH BARMAN

1. focuses on backup procedures.
2. focuses even more on recovery procedures.
3. relies on education and training on strong theoretical and practical concepts of PostgreSQL’s crash recovery, backup, Point-In-Time-Recovery, and replication for your team members.
4. promotes testing your backups (only a backup that is tested can be considered to be valid), either manually or automatically (be creative with Barman’s hook scripts!).
5. fosters regular practice of recovery procedures, by all members of your devops team (yes, developers too, not just system administrators and DBAs).
6. solicites to regularly scheduled drills and disaster recovery simulations with the team every 3-6 months.
7. relies on continuous monitoring of PostgreSQL and Barman, and that is able to promptly identify any anomalies.

**GOAL**

1. Provide quick and reliable Backup Policy to keep instances more integrated.
2. Provide centralized and manageable Backup-sets for PostgreSQL’s instances.
3. Provide flexible Recovery methods to deal with different kinds of disasters.

**ARCHITECTURE**



In this scenario, we will need to configure:

1. A standard connection to PostgreSQL for management, coordination, and monitoring purposes.
2. An SSH connection for base backup operations to be used by rsync that allows the barman user on the Barman server to connect as postgres user on the PostgreSQL server
3. An SSH connection for WAL archiving to be used by the archive\_command

**SYSTEM REQUIREMENTS**  
 Python 2.7+ (Linux/Unix)  
 Python modules:  
 argcomplete  
 argh >= 0.21.2 <= 0.26.2  
 argparse (Python 2.6 only)  
 psycopg2 >= 2.4.2  
 python-dateutil <> 2.0  
 setuptools  
 PostgreSQL >= 8.3  
 rsync >= 3.0.4 (optional for PostgreSQL >= 9.2)

Install pip tools first:

Pip Projects: https://pypi.org/

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

pip install argh

pip install python-dateutil

pip install psycopg2-binary

**★ Installation Barman tools on RedHat/CentOS ★**

Install from sources:

<https://sourceforge.net/projects/pgbarman/files/>

$ tar zxvf barman-2.4.tar.gz

$ cd barman-2.4

$ ./setup.py build

$ ./setup.py install

Install from RPM package：

rpm -ivh barman-2.4-1.el7.noarch.rpm

or

yum install barman

Note: see more options <https://docs.python.org/3/install/index.html#alternate-installation-the-user-scheme>

**Configuration**

There are two types of configurations:

* global/general configuration
* server configuration

The main configuration file (set to /etc/barman.conf by default)，also we can put each server configuration into /etc/barman.d/ and configuration file must have “.conf” suffix.

Examples of configuration:

Global configuration

[barman]

barman\_home = /barman

barman\_user = barman

log\_file = /var/log/barman/barman.log

log\_level = INFO

minimum\_redundancy = 1

retention\_policy = REDUNDANCY 1

Server configuration

[pg\_streaming\_backup]

description = "Example of PostgreSQL Database (Streaming-Only)"

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

streaming\_conninfo = host=pg user=streaming\_barman dbname=postgres port=5432

backup\_method = postgres

streaming\_archiver = on

slot\_name = barman

[**pg\_ssh\_backup**]  
description = "Example of PostgreSQL Database (via Ssh)"  
ssh\_command = ssh postgres@pg  
conninfo = host=pg user=barman dbname=postgres port=5432  
backup\_method = rsync  
parallel\_jobs = 1  
reuse\_backup = link  
archiver = on

\* conninfo (see 34.1.2. Parameter Key Words)

\* configuration\_files\_directory(Can also put separated servers configuration under this path)

https://www.postgresql.org/docs/current/static/libpq-connect.html#LIBPQ-PARAMKEYWORDS

Note: see more configuration documentation.

http://docs.pgbarman.org/release/2.5/barman.5.html

How to configure on PostgreSQL side

1. Provide a PG user for backup server who can access DB as a superuser.

$ createuser -s -P barman (it is for **conninfo** connection user)

1. Configure pg\_hba.conf to allow the connection user access from remote.

Type Database User ADDRESS METHOD

host all all 192.0.0.0/32 md5

1. Provide a replication PG user for streaming connection to transfer the WAL.

$ createuser -s

Validate the replication user

$ psql -U streaming\_barman -c"IDENTIFY\_SYSTEM" replication=1

1. Set up WAL archive, sender processes and Replication management processes by wal\_level, max\_wal\_senders and max\_replication\_slots parameters in postgresql.conf.

wal\_level = 'replica' (for version older than 9.6, plz using “hot\_standby”)

max\_wal\_senders = 2

max\_replication\_slots = 2

SSH access between barman and postgres users

PostgreSQL archive parameter “archive\_command” need to be configured to transfer the WAL to barman

$ su - postgres

$ ssh-keygen

$ ssh-copy-id -i ~/.ssh/id\_rsa.pub barman@hostname

Barman also need to connect to PostgreSQL to synchronize datafiles by “ssh\_command”

$ su - barman

$ ssh-keygen

$ ssh-copy-id -i ~/.ssh/id\_rsa.pub postgres@hostname

Verification of WAL archiving configuration

$ barman switch-wal --force --archive **pg\_ssh\_backup**

**★ Barman command line ★**

1. General commands： observe the all backup instances status.
2. Server commands：observe specific details about the instance status(such as WAL, backupset)
3. Backup commands：works directly on backups

General：

List scheduler barman task

$ barman cron

If the barman is installed with RPM, we can find a configuration “barman” inside /etc/cront.d/.

◆ List all configured instances’ reports with JSON format

$ barman diagnose



◆ List all activity instances

$ barman list-server

$ barman list-server --minimal

Server:

◆ Show the configuration parameters for a given server

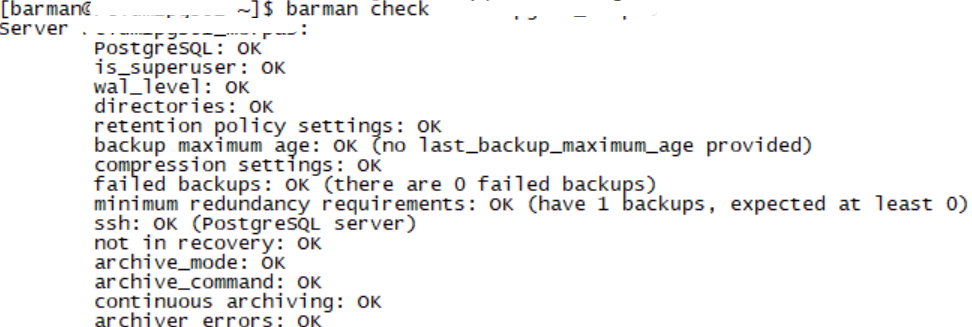
$ barman show-server **pg\_ssh\_backup**

◆ Takes a full backup (*base backup*) of a given server.

$ barman bakcup **pg\_ssh\_backup**

◆ Check a instance status

$ barman check **pg\_ssh\_backup**



Provide a “WAL hub” for the server who wanna recovery from barman

$ barman get-wal **pg\_ssh\_backup wal-name**

The recovered PostgreSQL will also configure **recovery.conf** with following option

restore\_command = 'sudo -u barman barman get-wal **pg\_ssh\_backup** %f > %p'

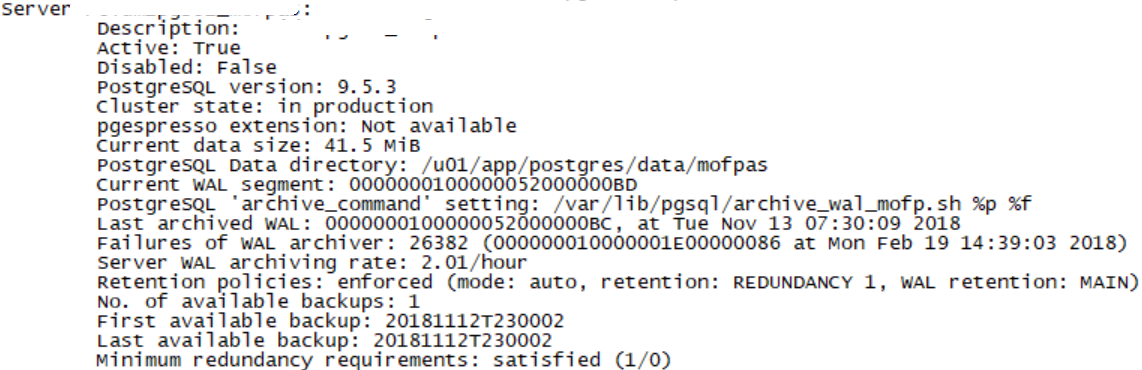
◆ List all available backup sets

$ barman list-backup **pg\_ssh\_backup**

****

◆ Shows live information and status of a PostgreSQL server

$ barman status **pg\_ssh\_backup**

****

Rebuild xlog database which keeps all received WALs.

$ barman rebuild-xlogdb **pg\_ssh\_backup**

$ ls -l /barman/**pg\_ssh\_backup**/wals/xlog.db

Reports the status of any streaming client currently attached to the PostgreSQL server

$ barman replication-status **pg\_ssh\_backup**

This command makes the PostgreSQL server switch to another transaction log file (WAL)

$ barman switch-wal **pg\_ssh\_backup**

Handle the archived wal log from a given server

$ barman archive-wal **pg\_ssh\_backup**.

Backup:

Check that all required WAL files for the consistency of a full backup have been correctly archived by barman

$ barman check-backup **pg\_ssh\_backup backupiece\_id**

Delete a given backup

$ barman delete **pg\_ssh\_backup backupiece\_id**

◆ Recovery whole server

$ barman recover **pg\_ssh\_backup backupiece\_id** /path/to/recover/dir

Recovery to a remote server

$ barman recover --remote-ssh-command "ssh postgres@pg" **pg\_ssh\_backup backupiece\_id** /path/to/recover/dir

◆ Point in time recovery

* --target-time TARGET\_TIME: to specify a timestamp
* --target-xid TARGET\_XID: to specify a transaction ID
* --target-name TARGET\_NAME: to specify a named restore point previously created with the pg\_create\_restore\_point(name) function
* --target-immediate: recovery ends when a consistent state is reached (that is the end of the base backup process)

$ barman recover --remote-ssh-command "ssh postgres@pg" --target-time "2018-11-07 15:39:57" **pg\_ssh\_backup backupiece\_id** /path/to/recover/dir

$ barman recover --remote-ssh-command "ssh postgres@pg" --target-xid "1692" **pg\_ssh\_backup backupiece\_id** /path/to/recover/dir

## ◆ List all the available information for a particular backup of a given server

$ barman show-backup pg\_ssh\_backup backupiece\_id