PgBackMan - PostgreSQL Backup Manager

Version-1.2.0

Author: Rafael Martinez Guerrero (University of Oslo)
E-mail: rafael@postgresql.org.es
Source: https://github.com/rafaelma/pgbackman
Web: http://www.pgbackman.org/
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Main features</td>
<td>4</td>
</tr>
<tr>
<td>Architecture and components</td>
<td>5</td>
</tr>
<tr>
<td>Installation</td>
<td>6</td>
</tr>
<tr>
<td>System requirements</td>
<td>6</td>
</tr>
<tr>
<td>Installing from source</td>
<td>7</td>
</tr>
<tr>
<td>Installing via RPM packages</td>
<td>7</td>
</tr>
<tr>
<td>Installing via Deb packages</td>
<td>7</td>
</tr>
<tr>
<td>Installing the pgbackman Database</td>
<td>8</td>
</tr>
<tr>
<td>Upgrading PgBackMan</td>
<td>8</td>
</tr>
<tr>
<td>Configuration</td>
<td>10</td>
</tr>
<tr>
<td>Backup servers</td>
<td>10</td>
</tr>
<tr>
<td>PgSQL nodes</td>
<td>12</td>
</tr>
<tr>
<td>Configuration file</td>
<td>12</td>
</tr>
<tr>
<td>System administration and maintenance</td>
<td>13</td>
</tr>
<tr>
<td>pgbackman_crontrol</td>
<td>13</td>
</tr>
<tr>
<td>pgbackman_maintenance</td>
<td>14</td>
</tr>
<tr>
<td>pgbackman_alerts</td>
<td>14</td>
</tr>
<tr>
<td>PgBackMan shell</td>
<td>14</td>
</tr>
<tr>
<td>clear</td>
<td>16</td>
</tr>
<tr>
<td>delete_backup_definition dbname</td>
<td>16</td>
</tr>
<tr>
<td>delete_backup_definition_id</td>
<td>17</td>
</tr>
<tr>
<td>delete_backup_server</td>
<td>18</td>
</tr>
<tr>
<td>delete_pgsq1_node</td>
<td>19</td>
</tr>
<tr>
<td>move_backup_definition</td>
<td>20</td>
</tr>
<tr>
<td>quit</td>
<td>21</td>
</tr>
<tr>
<td>register_backup_definition</td>
<td>21</td>
</tr>
<tr>
<td>register_backup_server</td>
<td>23</td>
</tr>
<tr>
<td>register_pgsq1_node</td>
<td>24</td>
</tr>
<tr>
<td>register_restore_definition</td>
<td>25</td>
</tr>
<tr>
<td>register_snapshot_definition</td>
<td>26</td>
</tr>
<tr>
<td>set</td>
<td>28</td>
</tr>
<tr>
<td>shell</td>
<td>28</td>
</tr>
<tr>
<td>show_backup_catalog</td>
<td>29</td>
</tr>
<tr>
<td>show_backup_definitions</td>
<td>30</td>
</tr>
<tr>
<td>show_backup_details</td>
<td>31</td>
</tr>
</tbody>
</table>
Introduction

PgBackMan is a tool for managing PostgreSQL logical backups created with `pg_dump` and `pg_dumpall`.

It is designed to manage backups from thousands of databases running in multiple PostgreSQL nodes, and it supports a multiple backup server topology.

It also manages role and database configuration information when creating a backup of a database. This information is necessary to ensure a 100% restoration of a logical backup of a database and the elements associated to it.

Even though a backup created with `pg_dump` or `pg_dumpall` can never guarantee a full disaster recovery of all data changed between the moment when the backup was taken and the moment of a
future crash, they are still necessary if you need to archive versions of a database, move databases between PgSQL nodes and clone databases between production, pre-production and/or development servers.

 Logical backups are also an easy way of taken backups of databases not requiring PITR backups.

 PgBackMan is not a tool for managing PITR (Point in time recovery) backups. There are other solutions that can be used for managing PITR backups, such as PITRTools, OmniPITR, and Barman.

 The PgBackMan code is distributed under the GNU General Public License 3 and it is written in Python and PL/PgSQL. It has been developed and tested by members of the Database Operations Group at the Center for Information Technology at the University of Oslo.

 An example of how a system using PgBackMan may look like can be seen in the next figure:

![Diagram of PgBackMan system]

**Main features**

The main features of PgBackMan are:

- Central database with metadata information.
- PgBackMan shell for interaction with the system.
- Management of multiple backup servers.
- Management of multiple PostgreSQL servers.
- Management of thousands of backups dumps through a catalogue.
- Full backup of role information for a database.
- Full backup of database configuration for a database.
- Manual and scheduled backups.
- Management of retention policies for backups dumps.
- Fully detailed backup reports.
- Multiple predefined database backup types, CLUSTER, FULL, SCHEMA, DATA.
• Automatic definitions of backups for all databases running in a PgSQL node.
• Automatic definitions of backups for all databases without definitions in a PgSQL node.
• Automatic deletion after a quarantine period of backup definitions and associated files for databases than have been deleted in a PgSQL node.
• Automatic restore procedures.
• Possibility of pausing / resuming replication on slaves/standby nodes when taking large backups.
• Autonomous pgbackman_dump program that functions even if the central database with metadata is not available.
• Possibility of sending alerts via SMTP when an error happens.
• Possibility of moving backup definitions between backup servers in a bulk operation.
• Handling of error situations.
• Written in Python and PL/PgSQL.
• Distributed under the GNU General Public License 3.

Architecture and components

The components forming part of PgBackMan could be listed as follows:

• **Backup servers**: One or several backup servers running PgBackMan. All SQL dumps and logfiles are saved in these servers. They need access via *libpq* to the postgresQL nodes where the backup server will be allowed to run backups and restores.

• **PGnodes**: PostgreSQL servers running postgresQL databases.

• **PgBackMan DB**: Central postgresQL metadata database used by PgBackMan. All backup servers need access to this database.

• **PgBackMan shell**: This is a program that must be run in a text terminal. It can be run in any of the backup servers registered in the system. It is the console used to manage PgBackMan.

• **pgbackman_control**: This program runs in every backup server and takes care of updating crontab files and creating AT jobs when backup, snapshots or restore definitions are created, when PgSQL nodes are stopped or deleted, or when backup definitions are stopped or deleted.

• **pgbackman_maintenance**: This program runs in every backup server and runs some maintenance jobs needed by PgBackMan. It enforces retentions for backup and snapshot definitions. It deletes backup and log files from catalog entries associated to a backup definition after this definition has been deleted with the force parameter. It stops automatically all backup definitions for databases that have been deleted with DROP DATABASE or renamed in the PgSQL nodes running them. And it processes all pending backup/restore catalog log files created in the server if the pgbackman database has been down when pgbackman_dump and pgbackman_restore have been running.

• **pgbackman_dump**: This program runs in the backup servers when a backup or snapshot has to be taken.

• **pgbackman_restore**: This program runs in the backup servers when a restore has to be run.

• **pgbackman_alerts**: This programs sends alerts via SMTP when a backups fails. This feature is activated in the configuration file.

The next figure shows all the components forming part of PgBackMan and how they interact with each other:
Installation

You will have to install the PgBackMan software in all the servers that are going to be used as backup servers by PgBackMan.

System requirements

- Linux/Unix
- Python 2.6 or 2.7
- Python modules:
  - psycopg2 >= 2.4.0
  - argparse >= 1.2.1
- PostgreSQL >= 9.2 for the pgbackman database
- PostgreSQL >= 9.0 and <=9.6 in all PostgreSQL nodes that are going to use PgBackMan to manage logical backups.
• AT and CRON installed and running.

Before you install PgBackMan you have to install the software needed by this tool
In systems using `yum`, e.g. Centos, RHEL, ...

```
yum install python-psycopg2 python-argparse at cronie
```

In system using `apt-get`, e.g. Debian, Ubuntu, ...

```
apt-get install python-psycopg2 python-argparse at cron
```

If you are going to install from source, you need to install also these packages: python-dev(el), python-setuptools, git, make, rst2pdf
In systems using `yum`:

```
yum install python-devel python-setuptools git make rst2pdf
```

In system using `apt-get`:

```
apt-get install python-dev python-setuptools git make rst2pdf
```

## Installing from source

The easiest way to install PgBackMan from source is to get the last version from the master branch at the GitHub repository.

```
[root@server]# cd
[root@server]# git clone https://github.com/rafaelma/pgbackman.git
[root@server]# cd pgbackman
[root@server]# ./setup2.py install --install-scripts=/usr/bin
```

This will install all users, groups, programs, configuration files, logfiles and the pgbackman module in your system.

## Installing via RPM packages

RPM packages for CentOS 6/7 and RHEL6/7 are available at [http://www.pgbackman.org/download.html](http://www.pgbackman.org/download.html)

Install the RPM package with:

```
[root@server]# rpm -Uvh pgbackman-<version>.rpm
```

We are working to get RPM packages for PgBackMan in the official PostgreSQL Yum repository.

## Installing via Deb packages

Deb packages for Debian7 are available at [http://www.pgbackman.org/download.html](http://www.pgbackman.org/download.html)

Install the Deb package with:

```
[root@server]# dpkg -i pgbackman_<version>.deb
```
We are working to get DEB packages for PgBackMan in the official PostgreSQL apt repository.

**Installing the pgbackman Database**

After the requirements and the PgBackMan software are installed, you have to install the pgbackman database in a server running PostgreSQL. This database is the core of the PgBackMan tool and it is used to save all the metadata needed to manage the system.

You can get this database from the directory sql/ in the source code or under the directory /usr/share/pgbackman if you have installed PgBackMan via source, rpm or deb packages.

You can install the pgbackman database for the first time with this command:

```
psql -h <dbhost.domain> -f /usr/share/pgbackman/pgbackman.sql
```

One should update some default parameters in the pgbackman database before one starts using the system. These parameters will be copied to the default configuration of the servers registered in PgBackMan.

We recommend to update these three parameters with the values you want to use in your PgBackMan installation:

```
UPDATE pgsql_node_default_config SET value = 'address@your.domain' WHERE parameter = 'logs_email';
UPDATE pgsql_node_default_config SET value = 'your.domain' WHERE parameter = 'domain';
UPDATE backup_server_default_config SET value = 'your.domain' WHERE parameter = 'domain';
```

These values are only the default suggestion one will get when a new backup server or PgSQL node is registered in the system. They can be changed or updated via the PgBackMan shell at any time.

**Upgrading PgBackMan**

This section has information about how to upgrade to a newer version of PgBackMan when you already are using PgBackMan.

Two things has to be done to run an upgrade of PgBackMan:

- Upgrade the PgBackMan software to the new version
- Upgrade the pgbackman database to the new version

There are a few things we have to take care of when these two steps are done to avoid problems:

- All backup servers have to run the same version of PgBackMan.
- No new backups should be started during the upgrade.
- No backups should be running during the upgrade

The recommended procedure to upgrade to a new version will be as follow:

1. Be sure no backups will be started during the upgrade.

   We recommend to have e.g. a 30 min. maintenance time window everyday or week where you do not have any backup definitions running backup jobs. This way you can run your upgrades in this maintenance time window without having to think that a backup will be startet when you are upgrading PgBackMan.

   To be on the safe side, stop crond, atd, pgbackman_control and pgbackman_maintenance with these commands:
This has to be done in all backup servers running PgBackMan.

2. Check that no backups or restores are running:

```
[pgbackman@pg-backup01]# ps ax | egrep "pgbackman_dump|pgbackman_restore"
```

If you have PgBackMan backup or restore jobs running, wait until they finish or kill them if you do not want to wait for them to finish.

3. Upgrade the PgBackMan software via your favorite method, source, rpm packages or deb packages. Check the "Installation section" for more information.

4. Check that you have the new PgBackMan configuration file saved as /etc/pgbackman/pgbackman.conf and that it has the information about where to find the pgbackman database.

5. Start the pgbackman shell in one of the backup servers and follow the instructions to upgrade the pgbackman database:

```
[pgbackman@pg-backup01]# pgbackman

############################
ATTENTION
############################

The PgBackMan software version [2:v_1_1_0] is different from the PgBackMan database version [1:v_1_0_0].

# Do you want to upgrade the PgBackMan database to version: [2:v_1_1_0] (yes/no): yes

############################
Upgrading PgBackMan database
############################

[OK]: File: /usr/share/pgbackman/pgbackman_2.sql exists.
[OK]: File /usr/share/pgbackman/pgbackman_2.sql installed.

Welcome to the PostgreSQL Backup Manager shell ver.1.1.0
Type help or \? to list commands.

[pgbackman]$ show_pgbackman_config
+-------------------------------------+--------------------------+
| Software version:                   | [2]:1_1_0               |
| Configuration file used:            | /etc/pgbackman/pgbackman.conf |
| PGBACKMAN DATABASE                  |                         |
|       DBhost:                        | pgbackmandb.example.net |
|       DBhostaddr:                    |                         |
|       DBport:                        | 5432                    |
|       DBname:                        | pgbackman               |
```
6. After the pgbackman database has been upgraded, start crond, atd, pgbackman_control and pgbackman_maintenance:

```
[root@pg-backup01]# /etc/init.d/pgbackman start
[root@pg-backup01]# /etc/init.d/crond stop
[root@pg-backup01]# /etc/init.d/atd stop
```

7. Use PgBackMan as usual.

## Configuration

### Backup servers

A backup server needs to have access to the pgbackman database and to all PgSQL nodes in which we need to take backups or restore data. This can be done as follows:

1. Update `/etc/pgbackman/pgbackman.conf` with the database parameters needed by PgBackMan to access the central metadata database. You need to define `host` or `hostaddr`, `port`, `dbname`, `user` under the section `[pgbackman_database]`. You can also define a password in this section but we discourage to do this and recommend to define a `.pgpass` file in the home directory of the users `root` and `pgbackman` with this information, e.g.:

   ```
   <dbhost.domain>:5432:pgbackman:pgbackman_role_rw:PASSWORD
   ```

   and set the privileges of this file with `chmod 400 ~/.pgpass`.

   An even better solution will be to use cert autentication for the pgbackman database user, so we do not need to save passwords values.

2. Update and reload the `pg_hba.conf` file in the postgresQL server running the pgbackman database, with a line that gives access to the pgbackman database from the new backup server. We recommend to use a SSL connection to encrypt all the traffic between the database server and the backup server, e.g.:

   ```
   hostssl pgbackman pgbackman_role_rw <backup_server_IP>/32 md5
   ```
3. Install the PostgreSQL clients for all the versions you want to support. PgBackMan can take backups of PostgreSQL servers running a version >= 9.0. We recommend using http://yum.postgresql.org/ or http://apt.postgresql.org/ to install the client packages for the different versions.

4. Define the backup server in PgBackMan via the PgBackMan shell:

```
[pgbackman@pg-backup01 ~]# pgbackman

#################################################################
Welcome to the PostgreSQL Backup Manager shell (v.1.1.0)
#################################################################
Type help or \? to list commands.

[pgbackman]$ register_backup_server

# Hostname []: pg-backup01
# Domain [uio.no]:
# Remarks []: Main backup server

# Are all values correct (yes/no): yes

[Done]
```

```
[pgbackman]$ show_backup_servers

+-------+------------------+----------------------+
| SrvID | FQDN              | Remarks              |
+-------+--------------------+--------------------+
  00001 | pg-backup01.uio.no | Main backup server  |
+-------+--------------------+--------------------+
```

5. Check that the configuration parameters for the backup server are correct. e.g. One will have to update the directories with the PostgreSQL client binaries if you are using Debian:

```
[pgbackman]$ update_backup_server_config

# SrvID / FQDN []: 1

# PgSQL bindir 9.0 [/usr/pgsql-9.0/bin]: /usr/lib/postgresql/9.0/bin
# PgSQL bindir 9.1 [/usr/pgsql-9.1/bin]: /usr/lib/postgresql/9.1/bin
# PgSQL bindir 9.2 [/usr/pgsql-9.2/bin]: /usr/lib/postgresql/9.2/bin
# PgSQL bindir 9.3 [/usr/pgsql-9.3/bin]: /usr/lib/postgresql/9.3/bin
# PgSQL bindir 9.4 [/usr/pgsql-9.4/bin]: /usr/lib/postgresql/9.4/bin
# PgSQL bindir 9.5 [/usr/pgsql-9.5/bin]: /usr/lib/postgresql/9.5/bin
# PgSQL bindir 9.6 [/usr/pgsql-9.6/bin]: /usr/lib/postgresql/9.6/bin
# Main backup dir [/srv/pgbackman]:

# Are all values to update correct (yes/no): yes

[Done] Configuration parameters for SrvID: 1 updated.
```

```
[pgbackman]$ show_backup_server_config

# SrvID / FQDN: 1
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_user</td>
<td>postgres</td>
<td>postgreSQL admin user</td>
</tr>
<tr>
<td>backup_server_status</td>
<td>RUNNING</td>
<td>Default backup server status</td>
</tr>
<tr>
<td>domain</td>
<td>example.net</td>
<td>Default domain</td>
</tr>
<tr>
<td>pgbackman_dump</td>
<td>/usr/bin/pgbackman_dump</td>
<td>Program used to take backup dumps</td>
</tr>
<tr>
<td>pgbackman_restore</td>
<td>/usr/bin/pgbackman_restore</td>
<td>Program used to restore backup dumps</td>
</tr>
<tr>
<td>postgresql_bin_9_0</td>
<td>/usr/lib/postgresql/9.0/bin</td>
<td>postgreSQL 9.0 bin directory</td>
</tr>
<tr>
<td>postgresql_bin_9_1</td>
<td>/usr/lib/postgresql/9.1/bin</td>
<td>postgreSQL 9.1 bin directory</td>
</tr>
<tr>
<td>postgresql_bin_9_2</td>
<td>/usr/lib/postgresql/9.2/bin</td>
<td>postgreSQL 9.2 bin directory</td>
</tr>
<tr>
<td>postgresql_bin_9_3</td>
<td>/usr/lib/postgresql/9.3/bin</td>
<td>postgreSQL 9.3 bin directory</td>
</tr>
<tr>
<td>postgresql_bin_9_4</td>
<td>/usr/lib/postgresql/9.4/bin</td>
<td>postgreSQL 9.4 bin directory</td>
</tr>
<tr>
<td>postgresql_bin_9_5</td>
<td>/usr/lib/postgresql/9.5/bin</td>
<td>postgreSQL 9.5 bin directory</td>
</tr>
<tr>
<td>postgresql_bin_9_6</td>
<td>/usr/lib/postgresql/9.6/bin</td>
<td>postgreSQL 9.6 bin directory</td>
</tr>
<tr>
<td>root_backup_partition</td>
<td>/srv/pgbackman</td>
<td>Main partition used by pgbackman</td>
</tr>
<tr>
<td>root_cron_file</td>
<td>/etc/cron.d/pgbackman</td>
<td>Crontab file used by pgbackman</td>
</tr>
</tbody>
</table>

6. Create the directory or partition in the backup server that will be used to save all backups, logfiles, and system data needed by PgBackMan. By default the system will use /srv/pgbackman.

Set the privileges of this directory with:

```bash
chown -R pgbackman:pgbackman /srv/pgbackman
chmod -R 700 /srv/pgbackman
```

**PgSQL nodes**

Every PgSQL node defined in PgBackMan will need to update and reload its own `pg_hba.conf` file to give access to the admin user (postgres per default) from the backup servers defined in PgBackMan, e.g.:

```plaintext
hostssl   *   postgres    <backup_server_IP>/32     md5
```

Remember that the `.pgpass` file of the `pgbackman` user in the backup server has to be updated with the information needed to access every PgSQL node we are going to take backups for.

We recommend to use a SSL connection to encrypt all the traffic between the database server and the backup server.

One can also use cert autentication so we do not need to save passwords values.

**Configuration file**

By default PgBackMan will look for a configuration file in these two locations and in this order:

```
$HOME/.pgbackman/pgbackman.conf, /etc/pgbackman/pgbackman.conf.
```

Several parameters can be configured in this file. The most important ones are `host` or `hostaddr`, `port`, `dbname`, `user` under the section `[pgbackman_database]`.

Check `/etc/pgbackman/pgbackman.conf` in your system for a list of parameters, what they are used for and default values.
System administration and maintenance

PgBackMan has three components which are used to administrate and maintain the backups, snapshots, restores, alerts and information associated to PgSQL nodes registered in the system. They are started with the script `/etc/init.d/pgbackman` and must run in every Backup server running PgBackMan.

Run this command after installing and configuring PgBackMan:

```
[root@server]# /etc/init.d/pgbackman start
```

One can stop the PgBackMan components with the same script:

```
[root@server]# /etc/init.d/pgbackman stop
```

If you want the PgBackMan components to start automatically at the boot time, type this if you are using CentOS or RHEL:

```
[root@server]# chkconfig pgbackman on
```

Or if you are using debian:

```
[root@server]# update-rc.d pgbackman defaults
```

**pgbackman_crontrol**

This program runs in a loop waiting for NOTIFY messages from the `pgbackman` database before executing an action. It will get a notification when:

- A new PgSQL node has been defined in the system.
- A PgSQL node is deleted from the system.
- A PgSQL node changes its status from RUNNING to STOPPED or vice versa.
- A snapshot backup has been defined.
- A backup restore has been defined.
- A new backup definition has been defined.
- A backup definition has been deleted.
- A backup definition has been updated.

The actions this program can execute are:

- Create the directory used for cached information from backup servers and PgSQL nodes.
- Delete the associated cache information when a PgSQL node gets deleted.
- Create a directory for pending log information.
- Create directories for backups and logs per PgSQL node defined in the system.
- Delete directories for backups and logs when a PgSQL node gets deleted.
- Update crontab files when new backup definitions get defined or deleted.
- Update crontab files when nodes get updated.
- Delete crontab files when nodes get deleted.
- Create an `at` job when a snapshot backup gets defined.
• Create an at job when a backup restore gets defined.
Every PgSQL node in the system will have its own directory and crontab file in every backup server running PgBackMan.

pgbackman_maintenance
This program can be executed in a cron modus (one single interaction per execution) or in a loop (default).
It runs these maintenance tasks:

• Enforce retention policies for backup definitions. It deletes backup files, log files and catalog information for backups that have expired.

• Enforce retention policies for snapshots. It deletes backup files, log files and catalog information for snapshots that have expired.

• Delete backup and log files from catalog entries associated to a backup definition after this definition has been deleted with the force-deletion parameter.

• Update the status of backup definitions to DELETED for databases than have been deleted in a PgSQL node. The DELETED definitions and all files associated to them will be deleted after a quarantine period defined by the PgSQL node configuration parameter automatic_deletion_retention.

• Delete restore logs files when definitions/catalogs used by the restore are deleted.

• Process pending backup catalog log files in the backup server. These files are created when the pgbackman database is not available for updating the catalog information metadata after a backup.

• Process pending restore catalog log files in the backup server. These files are created when the pgbackman database is not available for updating the catalog information metadata after a restore.

pgbackman_alerts
This program runs in a loop waiting for alerts that have to be sent via SMTP.
When a backup, a snapshot or a restore job terminates with an error, an e-mail will be sent to the e-mail address defined in the configuration (logs_email) for the PgSQL node where the error happens.
Use the commands show_pgsq1_node_config and update_pgsq1_node_config if you need to check or adjust the value of the parameter logs_email.

pgbackman_alerts will not send any message if it is not activated in the PgBackMan configuration file /etc/pgbackman/pgbackman.conf. Check the section [pgbackman_alerts] to activate and configure SMTP.
The file /etc/pgbackman/pgbackman_alerts.template can be modified to define the body of the e-mail message that will be sent with the alert.

PgBackMan shell
The PgBackMan interactive shell can be started by running the program /usr/bin/pgbackman

[pbackman@pg-backup01]# pgbackman

#################################################################################
Welcome to the PostgreSQL Backup Manager shell ver.1.2.0
#################################################################################
Type help or \? to list commands.
[pgbackman]$ help

Documented commands (type help <topic>):
===============================================
EOF                          show_databases_without_backup_definitions
clear                        show_empty_backup_catalogs
delete_backup_definition_dbname  show_history
delete_backup_definition_id      show_jobs_queue
delete_backup_server          show_pgbackman_config
delete_pgsql_node             show_pgbackman_stats
move_backup_definition        show_pgsql_node_config
quit                          show_pgsql_node_stats
register_backup_definition    show_pgsql_nodes
register_backup_server        show_restore_catalog
register_pgsql_node           show_restore_definitions
register_restore_definition   show_restore_details
register_snapshot_definition  show_restores_in_progress
set                           show_snapshot_definitions
shell                         show_snapshots_in_progress
show_backup_catalog           update_backup_definition
show_backup_definitions       update_backup_server
show_backup_details           update_backup_server_config
show_backup_server_config     update_pgsql_node
show_backup_server_stats      update_pgsql_node_config
show_backup_servers

Miscellaneous help topics:
============================
shortcuts  support

Undocumented commands:
============================
help

NOTE: It is possible to use the PgBackMan shell in a non-interactive modus by running
/usr/bin/pgbackman with the parameter --command <pgbackman_command> or -C
<pgbackman_command> in the OS shell. This can be used to run PgBackMan commands from shell
scripts.e.g.:

```bash
[pgbackman@pg-backup01 ~]# pgbackman -C "show_backup_servers"
+-------------------+------------------------+-----------------------------+
| SrvID  | FQDN                  | Remarks                     |
+---------+------------------------+-----------------------------+
| 00001   | pg-backup01.example.net | Main backup server           |
+---------+------------------------+-----------------------------+

[pgbackman@pg-backup01 ~]# pgbackman -C "show_backup_definitions all all pgbackman"
+-----------------------------+-------------------+--------------------------+-------------------+--------------------------+--------+--------------------------+
| SrvID / FQDN: all           | NodeID / FQDN: all| DBname                   | DefID  | ID  | Backup server | ID | PgSQL node | DBname |
+-----------------------------+-------------------+--------------------------+--------+-------------------+--------+--------------------------+
# Srvid / FQDN: all
# NodeID / FQDN: all
# DBname: pgbackman
```
One can also use the parameters `--output/-o csv` or `--output/-o json` when running `pgbackman` in non-interactive mode to generate an output in CSV or JSON format:

```bash
[pgbackman@pg-backup01 ~]$ pgbackman -o json -C "show_backup_servers"
{
    "backup_servers": [
        {
            "srvid": "00001",
            "fqdn": "pg-backup01.example.net",
            "remarks": "testing"
        }
    ]
}
```

clear

This command clears the screen and shows the welcome banner

```
[pgbackman]$ clear
```

This command can be run only without parameters. e.g.

```
[pgbackman]$ clear
```

delete_backup_definition_dbname

**NOTE:** Use this command with precaution

This command deletes all backup definitions for a database:

```
delete_backup_definition_dbname [NodeID/FQDN] [DBname] [force-deletion]
```

Parameters:

- **[NodeID/FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node running the database.
- **[DBname]**: Database name to delete
- **[force-deletion]**: Use force deletion.
You have to use the parameter `force-deletion` if you want to force the deletion of backup definitions with active backups in the catalog. If you use `force-deletion`, all backups in the catalog for the backup definition deleted, will be deleted regardless of the retention period or retention redundancy used.

This command can be run with or without parameters. e.g.

```
[pgbbackman]$ delete_backup_definition_dbname 1 testdb force-deletion
[Done] Backup definition for DBname: testdb deleted with force.
```

```
[pgbbackman]$ delete_backup_definition_dbname
# NodeID / FQDN: 1
# DBname: testdb
# Force deletion (y/n): y
# Are you sure you want to delete this backup definition? (yes/no): yes

[Done] Backup definition for DBname: testdb deleted with force.
```

```
[pgbbackman]$ delete_backup_definition_dbname
# NodeID / FQDN: pg-node01.example.net
# DBname: testdb
# Force deletion (y/n): n
# Are you sure you want to delete this backup definition? (yes/no): yes

[ERROR]: Could not delete this backup job definition
MESSAGE: update or delete on table "backup_definition" violates foreign key constraint "backup_catalog_def_id_fkey" on table "backup_catalog"
DETAIL: Key (def_id)=(1) is still referenced from table "backup_catalog".
```

`delete_backup_definition_id`  
**NOTE:** Use this command with precaution

This command deletes a backup definition for a DefID:

```
delete_backup_definition_id [DefID] [force-deletion]
```

Parameters:

- **[DefID]:** ID of the backup definition to delete.
- **[force-deletion]:** Use force deletion.
You have to use the parameter `force-deletion` if you want to force the deletion of backup definitions with active backups in the catalog. If you use `force-deletion`, all backups in the catalog for the backup definition deleted will be deleted regardless of the retention period or retention redundancy used.

This command can be run with or without parameters. E.g.

```
[pgbman]$ delete_backup_definition_id 1 force-deletion
[Done] Backup definition for DefID: 1 deleted with force.
```

```
[pgbman]$ delete_backup_definition_id
--------------------------------------------------------
# DefID: 1
# Force deletion (y/n): y
# Are you sure you want to delete this backup definition? (yes/no): yes
--------------------------------------------------------
[Done] Backup definition for DefID: 1 deleted with force.
```

```
[pgbman]$ delete_backup_definition_id
--------------------------------------------------------
# DefID: 1
# Force deletion (y/n): n
# Are you sure you want to delete this backup definition? (yes/no): yes
--------------------------------------------------------
[ERROR]: Could not delete this backup job definition
--------------------------------------------------------
EXCEPTION:
MESSAGE: update or delete on table "backup_definition" violates foreign key constraint "backup_catalog_def_id_fkey" on table "backup_catalog"
DETAIL: Key (def_id)=(1) is still referenced from table "backup_catalog".
```

**delete_backup_server**

This command deletes a backup server defined in PgBackMan:

```
Command: delete_backup_server [SrvID | FQDN]
```

Parameters:

- **[SrvID | FQDN]**: SrvID in PgBackMan or FQDN of the backup server to delete.

You can use the backup server ID in PgBackMan or the FQDN of the server to choose the server to be deleted.

One have to delete all backup definitions associated to a backup server or move them to another backup server before one can delete a backup server from the system.
You will get an error if you try to delete a backup server that has active backup definitions associated. This is a safety measure to avoid operational errors with catastrophic consequences. This type of deletion cannot be forced.

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ delete_backup_server 2
[Done] Backup server deleted.
```

```
[pgbackman]$ delete_backup_server
# SrvID / FQDN: 2
# Are you sure you want to delete this server? (yes/no): yes
[Done] Backup server deleted.
```

```
[pgbackman]$ delete_backup_server
# SrvID / FQDN: 2
# Are you sure you want to delete this server? (yes/no): yes
[ERROR]: Could not delete this backup server
EXCEPTION:
MESSAGE: update or delete on table "backup_server" violates foreign key constraint "backup_definition_backup_server_id_fkey" on table "backup_definition"
DETAIL : Key (server_id)=(2) is still referenced from table "backup_definition".
```

delete_pgsql_node

This command deletes a PgSQL node registered in PgBackMan.

```
delete_pgsql_node [NodeID | FQDN]
```

Parameters:

- **[NodeID | FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node to delete.

One have to delete all backup definitions associated to a PgSQL node before one can delete a PgSQL node from the system.

You will get an error if you try to delete a PgSQL node that has active backup definitions associated. This is a safety measure to avoid operational errors with catastrophic consequences. This type of deletion cannot be forced.

This command can be run with or without parameters. e.g.:
move_backup_definition

This command moves backup definitions between backup servers for a particular combination of search values.

move_backup_definition  [From SrvID|FQDN]  [To SrvID|FQDN]  [NodeID|FQDN]  [DBname]  [DefID]

Parameters:

• [From SrvID | FQDN]: SrvID in PgBackMan or FQDN of the backup server running the backup jobs that will be moved to another backup server.

• [To SrvID | FQDN]: SrvID in PgBackMan or FQDN of the backup server where we will move the backup jobs.

• [NodeID | FQDN]: NodeID in PgBackMan or FQDN of the PgSQL node where we take the backup jobs we want to move.

One can use 'all' or '*' with this parameter.
• **[Dbname]**: Database name in the backup jobs we want to move.
  One can use 'all' or '*' with this parameter.
  
• **[DefID]**: Backup definition ID we want to move.
  The default value for a parameter is shown between brackets []. If the user does not define any value, the default value will be used.

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ move_backup_definition pg-backup01.example.net pg-backup02.example.net * *
[DONE] Moving backup definitions from backup server [pg-backup01.example.net] to backup
```

```
[pgbackman]$ move_backup_definition

# From backup server Srvid / FQDN [pg-backup01.example.net]:
# To Backup server Srvid / FQDN [pg-backup0.example.net]:
# PgSQL node NodeID / FQDN [all]:
# DBname [all]:
# DefID []:
# Are all values correct (yes/no): yes

[DONE] Moving backup definitions from backup server [pg-backup01.example.net] to backup
```

**quit**

This command quits/terminates the PgBackMan shell.

```
quit
A shortcut to this command is \q.
This command can be run only without parameters. e.g.:
```

```
[pgbackman]$ quit
Done, thank you for using PgBackMan

[pgbackman]$ \q
Done, thank you for using PgBackMan
```

**register_backup_definition**

This command registers a backup definition that will be run periodically by PgBackMan.:

```
register_backup_definition [Srvid ] FQDN
[NodeID ] FQDN
[Dbname]
[Dbname exceptions]
[min_cron]
[hour_cron]
[daymonth_cron]
[month_cron]
[weekday_cron]
[backup code]
```
Parameters:

- **[SrvID | FQDN]**: SrvID in PgBackMan or FQDN of the backup server that will run the backup job.
- **[NodeID | FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node running the database to backup.
- **[DBname]**: Database name. This parameter can be empty if defining a backup definition with code CLUSTER.

One can use two special values instead of a database name:

- `#all_databases#`: if you want to register the backup definition for all databases in the cluster (Except 'template0', 'template1' and 'postgres').
- `#databases_without_backups#`: if you want to register the backup definition for all databases in the cluster without a backup definition (Except 'template0', 'template1' and 'postgres').

- **[DBname exceptions]**: Databases that will not be considered when using the values `#all_databases#` or `#databases_without_backups#` in [DBname].

One can define several DBnames in a comma separated list.


- **[backup code]**:
  - CLUSTER: Backup of all databases in a PgSQL node using pg_dumpall. The backup file will be compressed with gzip if gzip is installed.
  - DATA: Data backup of the database.

- **[encryption]**: This parameter is not used at the moment. But it will be used in the future.
  - TRUE: GnuPG encryption activated.
  - FALSE: GnuPG encryption not activated.

- **[retention period]**: Time interval a backup will be available in the catalog, e.g. 2 hours, 3 days, 1 week, 1 month, 2 years

- **[retention redundancy]**: Minimum number of backups to keep in the catalog regardless of the retention period used. e.g. 1,2,3

- **[extra backup parameters]**: Extra parameters that can be used with pg_dump / pg_dumpall

- **[job status]**
  - ACTIVE: Backup job activated and in production.
  - STOPPED: Backup job stopped.

The default value for a parameter is shown between brackets [ ]. If the user does not define any value, the default value will be used. This command can be run with or without parameters. e.g.:
register_backup_server

This command registers a backup server in PgBackMan:

Command: register_backup_server [hostname] [domain] [remarks]

Parameters:

- [hostname]: Hostname of the backup server.
- [domain]: Domain name of the backup server.
- [remarks]: Remarks

The default value for a parameter is shown between brackets []. If the user does not define any value, the default value will be used. This command can be run with or without parameters. e.g

[pgbackman]$ register_backup_server backup01 "" "Test server"

[Done] Backup server backup01.example.org registered.
## Remarks 
Test server

## Are all values correct (yes/no): yes

---

**[Done]** Backup server backup01.example.org registered.

---

### register_pgsq1_node

This command registers a PostgreSQL node in PgBackMan:

```
register_pgsq1_node [hostname]
[domain]
[pgport]
[admin_user]
[status]
[remarks]
```

**Parameters:**

- **[hostname]:** Hostname of the PostgreSQL node
- **[domain]:** Domain name of the PostgreSQL node
- **[pgport]:** PostgreSQL port
- **[admin_user]:** PostgreSQL admin user
- **[status]:**
  - RUNNING: PostgreSQL node running and online
  - DOWN: PostgreSQL node not online.
- **[remarks]:** Remarks

All backup definitions from a PostgreSQL node will be started/stopped automatically if the PostgreSQL node gets the status changed to RUNNING/DOWN.

The default value for a parameter is shown between brackets []. If the user does not define any value, the default value will be used. This command can be run with or without parameters. e.g:

```
[pbackman]$ register_pgsq1_node pg-node01 "" "" "" running "Test node"
[Done] PostgreSQL node pg-node01.example.net registered.
```

```
[pbackman]$ register_pgsq1_node
Hostname []: pg-node01
Domain [example.org]:
Port [5432]:
Admin user [postgres]:
Status[STOPPED]: running
Remarks []: Test node

Are all values correct (yes/no): yes

[Done] PostgreSQL node pg-node01.example.org registered.
```
**register_restore_definition**

This command defines a restore job of a backup from the catalog. Nowadays it can only restore backups with code FULL (Schema + data).

It can be run only interactively.

Parameters:

- **[AT time]**: Timestamp to run the restore job.
- **[BckID]**: ID of the backup to restore.
- **[Target NodeID | FQDN]**: PgSQL node ID or FQDN where we want to restore the backup.
- **[Target DBname]**: Database name where we want to restore the backup. The default name is the DBname defined in BckID.
- **[Extra parameters]**: Extra parameters that can be used with pg_restore

This command can be run only without parameters. e.g:

```
[pgbckman]$ register_restore_definition

# AT timestamp [2014-05-30 09:44:04.503880]:
# BckID []: 35
# Target NodeID / FQDN []: 2
# Target DBname [pgbackman]:
# Extra parameters []:

# Are all values correct (yes/no): yes
[Processing restore data]
[OK]: Target DBname pgbackman does not exist on target PgSQL node.

[OK]: Role 'pgbackman_role_rw' does not exist on target PgSQL node.

[WARNING]: Role 'postgres' already exists on target PgSQL node.
# Use the existing role? (yes/no): yes

[Restore definition accepted]

AT time: 2014-05-30 09:44:04.503880
BckID to restore: 35
Roles to restore: pgbackman_role_rw
Backup server: [1] pg-backup01.example.net
Target PgSQL node: [2] pg-node01.example.net
Target DBname: pgbackman
Extra restore parameters:
Existing database will be renamed to : None

# Are all values correct (yes/no): yes

[Done] Restore definition registered.
```

There are some issues we have to take care of when running a restore of a backup. What happens if we want to restore a backup of a database or a role that already exists in the target server?
register_snapshot_definition

This command registers a one time snapshot backup of a database.
Parameters:

- **[SrvID | FQDN]**: SrvID in PgBackMan or FQDN of the backup server that will run the snapshot job.

- **[NodeID | FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node running the database to backup.

- **[DBname]**: Database name
  
  One can define several DBnames in a comma separated list.
  
  One can use the special value, '#all_databases#' if you want to register the snapshot backup for all databases in the cluster (except 'template0','template1' and 'postgres').
  
  This parameter will be ignored if backup-code=CLUSTER.

- **[DBname exceptions]**: Databases that will not be considered when using '#all_databases#' in [DBname].
  
  One can define several DBnames in a comma separated list.
  
  This parameter will be ignored if backup-code=CLUSTER.

- **[AT time]**: Timestamp to run the snapshot

- **[backup code]**:
  
  - CLUSTER: Backup of all databases in a PgSQL node using pg_dumpall
  
  
  
  - DATA: Data backup of the database.

- **[retention period]**: Time interval a backup will be available in the catalog, e.g. 2 hours, 3 days, 1 week, 1 month, 2 years

- **[extra backup parameters]**: Extra parameters that can be used with pg_dump / pg_dumpall

- **[tag]**: Define a tag for this snapshot registration. This value can be helpful when we register a snapshot for many databases at the same time. This tag could be used in the future when registering a backup recovery for all the databases from the same snapshot registration.

  If no value is defined, the system will generate a random alphanumeric tag.

- **[pg_dump/all release]**: Release of pg_dump / pg_dumpall to use when taking the snapshot, e.g. 9.0, 9.1, 9.2, 9.3, 9.4, 9.5 or 9.6. This parameter can be necessary if we are going to restore the snapshot in a postgresQL installation running a newer release than the source.

  This release version cannot be lower than the one used in the source installation running the database we are going to backup.

  The release of the source installation will be used per default if this parameter is not defined.

The default value for a parameter is shown between brackets [ ]. If the user does not define any value, the default value will be used. This command can be run with or without parameters. e.g.: ```
[pgbackman]$ register_snapshot_definition 1 1 test02 "" 2014-05-31 full "7 days" "Test snapshot" 

[Done] Snapshot for dbname: test02 defined.

---

# Backup server SrvID / FQDN []: pg-backup01.example.net
# PgSQL node NodeID / FQDN []: pg-node01.example.net
# DBname []: test02
# AT timestamp [2014-05-31 17:52:28.756359]:
# Backup code [FULL]:
# Retention period [7 days]:
# Extra parameters []:
# Tag [5D9012AA3]:
# pg_dump/all release [Same as pgSQL node running dbname]:
# Are all values correct (yes/no): yes

[Done] Snapshot for dbname: test02 defined.

---

set

This command can be used to change the value of some internal parameters used to configure the behavior of PgBackMan

set [parameter=value]

- **[parameter = value]**:
  - output_format: [TABLE | JSON | CSV]

shell

This command runs a command in the operative system.

shell [command]

Parameters:

- **[command]**: Any command that can be run in the operative system.

It exists a shortcut `[!]` for this command that can be used instead of `shell`. This command can be run only with parameters. e.g.:

[pgbackman]$ ! ls -l
total 88
-rw-rw-r--. 1 vagrant vagrant 135 May 30 10:04 AUTHORS
drwxrwxr-x. 2 vagrant vagrant 4096 May 30 10:03 bin
drwxrwxr-x. 4 vagrant vagrant 4096 May 30 10:03 docs
drwxrwxr-x. 2 vagrant vagrant 4096 May 30 10:03 etc
-rw-rw-r--. 1 vagrant vagrant 0 May 30 10:04 INSTALL
-rw-rw-r--. 1 vagrant vagrant 35121 May 30 10:04 LICENSE
drwxrwxr-x. 2 vagrant vagrant 4096 May 30 10:03 pgbackman
show_backup_catalog

This command shows all backup catalog entries for a particular combination of parameter values. These values are combined with AND.

```
show_backup_catalog [SrvID|FQDN] [NodeID|FQDN] [DBname] [DefID] [Status]
```

Parameters:

- **[SrvID|FQDN]**: SrvID in PgBackMan or FQDN of the backup server. One can use 'all' or '*' with this parameter.
- **[NodeID|FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node. One can use 'all' or '*' with this parameter.
- **[DBname]**: Database name. One can use 'all' or '*' with this parameter.
- **[DefID]**: Backup definition ID. One can use 'all' or '*' with this parameter.
- **[Status]**: Execution status of the backup. One can use 'all' or '*' with this parameter.
  - SUCCEEDED: Execution finished without error.
  - ERROR: Execution finished with errors.

The default value for a parameter is shown between brackets [ ]. If the user does not define any value, the default value will be used.

One can define multiple values for each parameter separated by a comma. These values are combined using OR.

This command can be run with or without parameters. e.g.:  

```
[pgbackman]$ show_backup_catalog 1 all dump_test,postgres all all
```

<table>
<thead>
<tr>
<th>BckID</th>
<th>DefID</th>
<th>SnapshotID</th>
<th>Finished</th>
<th>ID</th>
<th>Backup server</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000029</td>
<td>0000000006</td>
<td>0000000007</td>
<td>2014-05-28 09:08:20+00:00</td>
<td>1</td>
<td>pg-backup01.example</td>
</tr>
<tr>
<td>0000000027</td>
<td>0000000007</td>
<td>0000000006</td>
<td>2014-05-28 09:01:05+00:00</td>
<td>1</td>
<td>pg-backup01.example</td>
</tr>
<tr>
<td>0000000028</td>
<td>0000000006</td>
<td>0000000007</td>
<td>2014-05-28 09:01:05+00:00</td>
<td>1</td>
<td>pg-backup01.example</td>
</tr>
<tr>
<td>0000000026</td>
<td>0000000005</td>
<td>0000000005</td>
<td>2014-05-28 08:51:43+00:00</td>
<td>1</td>
<td>pg-backup01.example</td>
</tr>
<tr>
<td>0000000025</td>
<td>0000000002</td>
<td>0000000002</td>
<td>2014-05-28 08:47:03+00:00</td>
<td>1</td>
<td>pg-backup01.example</td>
</tr>
<tr>
<td>0000000024</td>
<td>0000000001</td>
<td>0000000001</td>
<td>2014-05-28 08:41:09+00:00</td>
<td>1</td>
<td>pg-backup01.example</td>
</tr>
</tbody>
</table>
show_backup_definitions
This command shows all backup definitions for a particular combination of parameter values. These values are combined with AND.

Parameters:

- **[SrvID|FQDN]**: SrvID in PgBackMan or FQDN of the backup server. One can use 'all' or '*' with this parameter.

- **[NodeID|FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node. One can use 'all' or '*' with this parameter.

- **[DBname]**: Database name. One can use 'all' or '*' with this parameter.

The default value for a parameter is shown between brackets [ ]. If the user does not define any value, the default value will be used.

One can define multiple values for each parameter separated by a comma. These values are combined using OR.

This command can be run with or without parameters. e.g.:
show_backup_details

This command shows all the details for one particular backup job.

Parameters:

- **[BckID]**: Backup ID

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_backup_details 25
```

```
BckID: 000000000025
ProcPID: 2067
Registered: 2014-05-28 08:47:03+00:00

Started: 2014-05-28 08:47:00+00:00
Finished: 2014-05-28 08:47:03+00:00
Duration: 0:00:02
Total size: 3468 bytes
Execution method: AT
Execution status: SUCCEEDED

DefID: 
SnapshotID: 00000002
DBname: dump_test
Backup server (ID/FQDN): [1] / pg-backup01.example.net
PgSQL node (ID/FQDN): [1] / pgbackmandb.example.net
PgSQL node release: 9.3
```
show_backup_server_config

This command shows the default configuration for a backup server.

```
show_backup_server_config [SrvID | FQDN]
```

Parameters:

- **[SrvID | FQDN]**: SrvID in PgBackMan or FQDN of the backup server

This command can be run with or without parameters. e.g.:

```
$ show_backup_server_config 1
```

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_user</td>
<td>postgres</td>
<td>PostgreSQL admin user</td>
</tr>
<tr>
<td>backup_server_status</td>
<td>RUNNING</td>
<td>Default backup server status - <em>Not used</em></td>
</tr>
<tr>
<td>domain</td>
<td>example.org</td>
<td>Default domain</td>
</tr>
<tr>
<td>pgbackman_dump</td>
<td>/usr/bin/pgbackman_dump</td>
<td>Program used to take backup dumps</td>
</tr>
<tr>
<td>pgbackman_restore</td>
<td>/usr/bin/pgbackman_restore</td>
<td>Program used to restore backup dumps</td>
</tr>
<tr>
<td>pgsql_bin_9_0</td>
<td>/usr/pgsql-9.0/bin</td>
<td>postgreSQL 9.0 bin directory</td>
</tr>
<tr>
<td>pgsql_bin_9_1</td>
<td>/usr/pgsql-9.1/bin</td>
<td>postgreSQL 9.1 bin directory</td>
</tr>
<tr>
<td>pgsql_bin_9_2</td>
<td>/usr/pgsql-9.2/bin</td>
<td>postgreSQL 9.2 bin directory</td>
</tr>
<tr>
<td>pgsql_bin_9_3</td>
<td>/usr/pgsql-9.3/bin</td>
<td>postgreSQL 9.3 bin directory</td>
</tr>
<tr>
<td>pgsql_bin_9_4</td>
<td>/usr/pgsql-9.4/bin</td>
<td>postgreSQL 9.4 bin directory</td>
</tr>
<tr>
<td>root_backup_partition</td>
<td>/srv/pgbackman</td>
<td>Main partition used by pgbackman</td>
</tr>
<tr>
<td>root_cron_file</td>
<td>/etc/cron.d/pgbackman</td>
<td>Crontab file used by pgbackman</td>
</tr>
</tbody>
</table>

---

On disk until: 2014-06-04 08:47:03+00:00
Error message:
show_backup_server_stats
This command shows global statistics for a backup server

```
show_backup_server_stats [SrvID | FQDN]
```

Parameters:

- **[SrvID | FQDN]**: SrvID in PgBackMan or FQDN of the backup server

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_backup_server_stats 1
```

```
# SrvID: 1
+-----------------------------------------------------+-----------------------------+
|                                      Backup server:       | [1] pg-backup01.example.net |
|                                                     |                             |
|               PgSQL nodes using this backup server:   | 1                           |
|                                                     |                             |
|                       Different databases:           | 1                           |
|                             Active Backup job defs:   | 3                           |
|                              Stopped Backup job defs: | 0                           |
|                       Backup job defs with CLUSTER code: | 0                           |
|                         Backup job defs with DATA code: | 0                           |
|                        Backup job defs with FULL code: | 2                           |
|                      Backup job defs with SCHEMA code: | 1                           |
|                                                     |                             |
|          Succeeded backups in catalog:         3890|
|                  Faulty backups in catalog:      2|
|     Total size of backups in catalog:         1106 MB|
| Total running time of backups in catalog: 5:03:08.108701 |
|                Oldest backup in catalog: 2014-05-28 08:40:06+00:00 |
|              Newest backup in catalog: 2014-06-01 19:44:07+00:00 |
|                                                     |                             |
|  Jobs waiting to be processed by pgbackman_control: | 1                           |
| Forced deletion of backups waiting to be processed: | 0                           |
+-----------------------------------------------------+-----------------------------+
```

show_backup_servers
This command shows all backup servers registered in PgBackMan.

```
show_backup_servers
```

This command can be run only without parameters. e.g.:

```
[pgbackman]$ show_backup_servers
```

```
<table>
<thead>
<tr>
<th>SrvID</th>
<th>FQDN</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>pg-backup01.example.net</td>
<td></td>
</tr>
<tr>
<td>00003</td>
<td>backup02.example.org</td>
<td>test server</td>
</tr>
</tbody>
</table>
```
show_databases_without_backup_definitions
This command shows all databases in a PgSQL node without a backup definition in PgBackMan.

```
show_databases_without_backup_definitions [Node ID | FQDN]
```

Parameters:

- **[Node ID | FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node. One can use 'all' or '*' with this parameter.

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_databases_without_backupDefinitions pg-node01.example.net
                          # NodeID / FQDN: pg-node01.example.net
+-----------------------+---------+
| PgSQL node            | DBname  |
+-----------------------+---------+
| pg-node01.example.net | example |
| pg-node01.example.net | test    |
| pg-node01.example.net | test02  |
```

show_empty_backup_catalogs
This command shows a list with all backup definitions with empty catalogs.

```
show_empty_backup_catalogs
```

This command can be run only without parameters. e.g.:

```
[pgbackman]$ show_empty_backup_catalogs
                                  +-------------+---------------------------+-----+-------------------------+----+-------------------------+-----------+-------------+------+------------+--------+------------+
                                  |    DefID    |         Registered        | ID. | Backup server           | ID | PgSQL node              |   DBname  | Schedule    | Code | Retention  | Status | Parameters |
                                  +-------------+---------------------------+-----+-------------------------+----+-------------------------+-----------+-------------+------+------------+--------+------------+
                                  | 000000000012 | 2014-05-30 07:29:28+00:00 |  1  | pg-backup01.example.net | 1  | pgbackmandb.example.net | pgbackman | 41 01 * * * | FULL | 7 days (1) | ACTIVE |            |
```

show_history
Show the list of commands that have been entered during the PgBackMan shell session.
show_history

A shortcut to this command is \s. One can also use the Emacs Line-Edit Mode Command History Searching to get previous commands containing a string. Hit [CTRL]+[r] in the PgBackMAN shell followed by the search string you are trying to find in the history.

This command can be run only without parameters. e.g.:

```
[pgbackman]$ show_history
[0]: help
[1]: help support
[2]: help show_history
[3]: shell df -h | grep /srv/pgbackman
[4]: show_history
[5]: help
[6]: show_history
[7]: show_backup_servers
[8]: show_pgsql_nodes
```

show_jobs_queue

This command shows the queue of jobs waiting to be processed by pgbackman_control.

show_jobs_queue

This queue changes when backup definitions get defined, updated or deleted. The queue has entries for the combination of backup server + PgSQL node affected by a change.

This command can be run only without parameters. e.g.:

```
[pgbackman]$ show_jobs_queue
| JobID | Registered                | SrvID | Backup server           | NodeID | PgSQL node              | Assigned |
+-------+---------------------------+-------+-------------------------+--------+-------------------------+----------+
| 10    | 2014-05-30 07:29:28+00:00 |   1   | pg-backup01.example.net |   1    | pgbackmandb.example.net |  False   |
```

show_pgbackman_config

This command shows the configuration parameters used by this PgBackMan shell session.

show_pgbackman_config

This command can be run only without parameters. e.g.:

```
[pgbackman]$ show_pgbackman_config
| Running modus: | interactive |
| Backup server: | pgbackup.example.org |
| Software version: | [3]:1_2_0 |
| Configuration file used: | /etc/pgbackman/pgbackman.conf |
| PGBACKMAN DATABASE: | |
```
<table>
<thead>
<tr>
<th>DBhost:</th>
<th>pgbackmandb.example.org</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBhostaddr:</td>
<td></td>
</tr>
<tr>
<td>DBport:</td>
<td>5432</td>
</tr>
<tr>
<td>DBname:</td>
<td>pgbackman</td>
</tr>
<tr>
<td>DBuser:</td>
<td>pgbackman_role_rw</td>
</tr>
<tr>
<td>Connection retry interval:</td>
<td>10 sec.</td>
</tr>
<tr>
<td>Database source dir:</td>
<td>/usr/share/pgbackman</td>
</tr>
<tr>
<td>DB version installed:</td>
<td>2017-05-24 17:48:43.308920+02:00</td>
</tr>
<tr>
<td>DB version:</td>
<td>[3]:1_2_0</td>
</tr>
<tr>
<td>PGBACKMAN_DUMP:</td>
<td></td>
</tr>
<tr>
<td>Temp directory:</td>
<td>/tmp</td>
</tr>
<tr>
<td>Pause recovery on slave node:</td>
<td>OFF</td>
</tr>
<tr>
<td>PGBACKMAN_MAINTENANCE:</td>
<td></td>
</tr>
<tr>
<td>Maintenance interval:</td>
<td>70 sec.</td>
</tr>
<tr>
<td>PGBACKMAN_ALERTS:</td>
<td></td>
</tr>
<tr>
<td>SMTP alerts activated:</td>
<td>OFF</td>
</tr>
<tr>
<td>Alerts check interval:</td>
<td>300 sec.</td>
</tr>
<tr>
<td>SMTP server:</td>
<td>localhost</td>
</tr>
<tr>
<td>SMTP port:</td>
<td>25</td>
</tr>
<tr>
<td>Use SMTP SSL:</td>
<td>ON</td>
</tr>
<tr>
<td>SMTP user:</td>
<td></td>
</tr>
<tr>
<td>Default From address:</td>
<td></td>
</tr>
<tr>
<td>Alerts e-mail template:</td>
<td>/etc/pgbackman/pgbackman_alerts.template</td>
</tr>
<tr>
<td>LOGGING:</td>
<td></td>
</tr>
<tr>
<td>Log level:</td>
<td>INFO</td>
</tr>
<tr>
<td>Log file:</td>
<td>/var/log/pgbackman/pgbackman.log</td>
</tr>
<tr>
<td>OUTPUT:</td>
<td></td>
</tr>
<tr>
<td>Default output format:</td>
<td>table</td>
</tr>
</tbody>
</table>

**show_pgbackman_stats**

This command shows global statistics for this PgBackMan installation.

```shell
show_pgbackman_stats
```

This command can be run only without parameters. e.g.:

```shell
[pgbackman]$ show_pgbackman_stats
```

<table>
<thead>
<tr>
<th>Running Backup servers:</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stopped Backup servers:</td>
<td>0</td>
</tr>
<tr>
<td>Running PgSQL nodes:</td>
<td>5</td>
</tr>
<tr>
<td>Stopped PgSQL nodes:</td>
<td>2</td>
</tr>
<tr>
<td>Different databases:</td>
<td>1</td>
</tr>
<tr>
<td>Active Backup job defs:</td>
<td>3</td>
</tr>
<tr>
<td>Stopped Backup job defs:</td>
<td>0</td>
</tr>
</tbody>
</table>
show_pgsql_node_config
This command shows the default configuration for a PgSQL node.

show_pgsql_node_config [NodeID | FQDN]

Parameters:
- **[NodeID/FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node.

This command can be run with or without parameters. e.g.:

```bash
[pgbackman]$ show_pgsql_node_config 5
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_user</td>
<td>postgres</td>
<td>PostgreSQL admin user</td>
</tr>
<tr>
<td>automatic_deletion_retention</td>
<td>14 days</td>
<td>Retention after automatic deletion of a backup job definition</td>
</tr>
<tr>
<td>backup_code</td>
<td>FULL</td>
<td>Backup job code</td>
</tr>
<tr>
<td>backup_day_month_cron</td>
<td>*</td>
<td>Backup day_month cron default</td>
</tr>
<tr>
<td>backup_hours_interval</td>
<td>01-06</td>
<td>Backup hours interval</td>
</tr>
<tr>
<td>backup_job_status</td>
<td>ACTIVE</td>
<td>Backup job status</td>
</tr>
<tr>
<td>backup_minutes_interval</td>
<td>01-59</td>
<td>Backup minutes interval</td>
</tr>
<tr>
<td>backup_month_cron</td>
<td>*</td>
<td>Backup month cron default</td>
</tr>
<tr>
<td>backup_weekday_cron</td>
<td>*</td>
<td>Backup weekday cron default</td>
</tr>
<tr>
<td>domain</td>
<td>example.org</td>
<td>Default domain</td>
</tr>
<tr>
<td>encryption</td>
<td>false</td>
<td>GnuPG encryption - <em>Not used</em></td>
</tr>
<tr>
<td>extra_backup_parameters</td>
<td><a href="mailto:example@example.org">example@example.org</a></td>
<td>Extra backup parameters</td>
</tr>
<tr>
<td>extra_restore_parameters</td>
<td></td>
<td>Extra restore parameters</td>
</tr>
<tr>
<td>logs_email</td>
<td></td>
<td>E-mail to send logs</td>
</tr>
<tr>
<td>pgnode_backup_partition</td>
<td>/srv/pgbackman/pgsql_node_5</td>
<td>Partition to save pgbackup information</td>
</tr>
<tr>
<td>pgnode_crontab_file</td>
<td>/etc/cron.d/pgsql_node_5</td>
<td>Crontab file for pgnode in the backup server</td>
</tr>
<tr>
<td>pgport</td>
<td>5432</td>
<td>PostgreSQL port</td>
</tr>
<tr>
<td>pgsql_node_status</td>
<td>STOPPED</td>
<td>pgsql node status</td>
</tr>
<tr>
<td>retention_period</td>
<td>7 days</td>
<td>Retention period for a backup job</td>
</tr>
<tr>
<td>retention_redundancy</td>
<td>1</td>
<td>Retention redundancy for a backup job</td>
</tr>
</tbody>
</table>
show_pgsql_node_stats
This command shows global statistics for a PgSQL node.

show_pgsql_node_stats [NodeID | FQDN]

Parameters:

• [NodeID|FQDN]: NodeID in PgBackMan or FQDN of the PgSQL node.

This command can be run with or without parameters. e.g.:

[pgbackman]$ show_pgsql_node_stats 1

--------------------------------------------------------
# NodeID: 1
--------------------------------------------------------
|                                        PgSQL node: | [1] pgbackmandb.example.net |
|                                                    |                             |
|      Backup servers running backups for this Node: | 1                           |
|                                                    |                             |
|                        Different databases: | 1                          |
|      Active Backup job defs: | 3                          |
|                        Stopped Backup job defs: | 0                          |
|      Backup job defs with CLUSTER code: | 0                          |
|                        Backup job defs with DATA code: | 0                          |
|      Backup job defs with FULL code: | 2                          |
|                        Backup job defs with SCHEMA code: | 1                          |
|                                                    |                             |
|                                    Succeeded backups in catalog: | 4527                      |
|        Faulty backups in catalog: | 2                          |
|                                    Total size of backups in catalog: | 1371 MB                    |
|        Total running time of backups in catalog: | 5:56:02.793539             |
|                                    Oldest backup in catalog: | 2014-05-28 08:40:06+00:00  |
|        Newest backup in catalog: | 2014-06-02 07:56:06+00:00  |
|                                                    |                             |
| Jobs waiting to be processed by pgbackman_control: | 1                          |

show_pgsql_nodes
This command shows all PgSQL nodes registered in PgBackMan.

show_pgsql_nodes

This command can be run only without parameters. e.g.:

[pgbackman]$ show_pgsql_nodes

+--------+-------------------------+--------+------------+---------+-------------+
| NodeID | FQDN                    | Pgport | Admin user | Status  | Remarks     |
|--------+-------------------------+--------+------------+---------+-------------+
| 000001 | pgbackmandb.example.net |  5432  | postgres   | RUNNING |             |
| 000002 | pg-node01.example.net   |  5432  | postgres   | RUNNING |             |
| 000008 | pg-node02.example.net   |  5432  | postgres   | STOPPED | test node   |
show_restore_catalog

This command shows all restore catalog entries for a particular combination of parameters values. These values are combined with AND.

```
show_restore_catalog [SrvID|FQDN]
[NodeID|FQDN]
[DBname]
```

Parameters:

- **[SrvID|FQDN]**: SrvID in PgBackMan or FQDN of the backup server. One can use 'all' or '*' with this parameter.
- **[NodeID|FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node. One can use 'all' or '*' with this parameter.
- **[DBname]**: Database name. One can use 'all' or '*' with this parameter.

The default value for a parameter is shown between brackets []. If the user does not define any value, the default value will be used.

One can define multiple values for each parameter separated by a comma. These values are combined using OR.

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_restore_catalog
```

show_restore_definitions

This command shows all restore definitions for a particular combination of parameter values. These values are combined with AND.

```
show_restore_definitions [SrvID|FQDN]
[NodeID|FQDN]
[DBname]
```

Parameters:

- **[SrvID|FQDN]**: SrvID in PgBackMan or FQDN of the backup server. One can use 'all' or '*' with this parameter.
- **[NodeID|FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node. One can use 'all' or '*' with this parameter.

- **[DBname]**: Database name. One can use 'all' or '*' with this parameter.

The default value for a parameter is shown between brackets `[]`. If the user does not define any value, the default value will be used.

One can define multiple values for each parameter separated by a comma. These values are combined using OR.

The status column in the output can have different values with these meanings:

- WAITING: Waiting to define an AT job to run this restore job
- DEFINED: AT job for this restore job has been defined
- ERROR: Could not define the AT job for this restore job.

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_restore_definitions
```

### show_restore_details

This command shows all the details for one particular restore job.

```
show_restore_details [RestoreID]
```

#### Parameters:

- **[RestoreID]**: Restore ID in the restore catalog.

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_restore_details
```

<table>
<thead>
<tr>
<th>RestoreDef</th>
<th>Registered</th>
<th>BckID</th>
<th>ID</th>
<th>Target PgSQL node</th>
<th>Target DBname</th>
<th>Renamed database</th>
<th>AT time</th>
<th>Extra parameters</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000005</td>
<td>2014-05-28 13:15:54+00:00</td>
<td>34</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
<td>pgbackman_1212</td>
<td>None</td>
<td>201405281316</td>
<td></td>
<td>DEFINED</td>
</tr>
<tr>
<td>00000006</td>
<td>2014-05-28 13:18:13+00:00</td>
<td>34</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
<td>pgbackman_1313</td>
<td>None</td>
<td>201405281318</td>
<td>-j 4</td>
<td>DEFINED</td>
</tr>
<tr>
<td>00000007</td>
<td>2014-05-30 09:43:31+00:00</td>
<td>35</td>
<td>2</td>
<td>pg-node01.example.net</td>
<td>pgbackman</td>
<td>None</td>
<td>201405300944</td>
<td></td>
<td>WAITING</td>
</tr>
</tbody>
</table>

show_restore_details
This command shows all the details for one particular restore job.

```
show_restore_details [RestoreID]
```

Parameters:

- **[RestoreID]**: Restore ID in the restore catalog.

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_restore_details
```

<table>
<thead>
<tr>
<th>RestoreID:</th>
<th>0000000006</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcPID:</td>
<td>6041</td>
</tr>
<tr>
<td>Registered:</td>
<td>2014-05-28 13:18:49.879066+00:00</td>
</tr>
</tbody>
</table>

| Started: | 2014-05-28 13:18:47+00:00 |
| Finished:| 2014-05-28 13:18:49+00:00 |
| Duration:| 0:00:01 |
| Execution status: | SUCCEEDED |
show_restores_in_progress

This command shows all restore jobs that are in progress and have not been completed.

```
show_restores_in_progress
```

This command can be run only without parameters. e.g.:

```
[pgbackman]$ show_restores_in_progress
```

show_snapshot_definitions

This command shows all snapshot definitions for a particular combination of parameter values. These values are combined with AND.

```
show_snapshot_definitions [SrvID|FQDN] [NodeID|FQDN] [DBname]
```

Parameters:

- **[SrvID|FQDN]**: SrvID in PgBackMan or FQDN of the backup server. One can use 'all' or '*' with this parameter.
- **[NodeID|FQDN]**: NodeID in PgBackMan or FQDN of the PgSQL node. One can use 'all' or '*' with this parameter.
- **[DBname]**: Database name. One can use 'all' or '*' with this parameter.
The default value for a parameter is shown between brackets \[\]. If the user does not define any value, the default value will be used.

One can define multiple values for each parameter separated by a comma. These values are combined using OR.

The status column in the output can have different values with these meanings:

- **WAITING**: Waiting to define an AT job to run this restore job
- **DEFINED**: AT job for this restore job has been defined
- **ERROR**: Could not define the AT job for this restore job

This command can be run with or without parameters. e.g.:

```
[pgbackman]$ show_snapshot_definitions
```

<table>
<thead>
<tr>
<th>SnapshotID</th>
<th>Registered</th>
<th>ID.</th>
<th>Backup server</th>
<th>ID</th>
<th>PgSQL node</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000000002</td>
<td>2014-05-28 08:45:19+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000005</td>
<td>2014-05-28 08:50:47+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000006</td>
<td>2014-05-28 08:59:47+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000007</td>
<td>2014-05-28 09:00:11+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000004</td>
<td>2014-05-28 08:48:50+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000003</td>
<td>2014-05-28 08:48:32+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000008</td>
<td>2014-05-28 10:06:08+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000010</td>
<td>2014-05-28 10:06:57+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000009</td>
<td>2014-05-28 10:06:31+00:00</td>
<td>1</td>
<td>pg-backup01.example.net</td>
<td>1</td>
<td>pgbackmandb.example.net</td>
</tr>
</tbody>
</table>

```
show_snapshots_in_progress
```

This command shows all snapshot jobs that are in progress and have not been completed.

```
[pgbackman]$ show_snapshots_in_progress
```

<table>
<thead>
<tr>
<th>SnapshotID</th>
<th>Registered</th>
<th>ID.</th>
<th>Backup server</th>
<th>ID</th>
<th>PgSQL node</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000000002</td>
<td>2014-09-22 21:09:25+00:00</td>
<td>2</td>
<td>pg-backup01.example.net</td>
<td>2</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000007</td>
<td>2014-09-22 22:17:07+00:00</td>
<td>2</td>
<td>pg-backup01.example.net</td>
<td>2</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000008</td>
<td>2014-09-22 22:17:25+00:00</td>
<td>2</td>
<td>pg-backup01.example.net</td>
<td>2</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000009</td>
<td>2014-09-24 06:45:43+00:00</td>
<td>2</td>
<td>pg-backup01.example.net</td>
<td>2</td>
<td>pgbackmandb.example.net</td>
</tr>
<tr>
<td>000000000010</td>
<td>2014-09-24 07:05:16+00:00</td>
<td>2</td>
<td>pg-backup01.example.net</td>
<td>2</td>
<td>pgbackmandb.example.net</td>
</tr>
</tbody>
</table>

```
update_backup_definition
```

This command updates the information of a backup definition.
Parameters:

- **[DefID]**: Backup definition ID to update.
- **[*_cron]**: Schedule definition using the cron expression.
- **[retention period]**: Time interval a backup will be available in the catalog, e.g. 2 hours, 3 days, 1 week, 1 month, 2 years.
- **[retention redundancy]**: Minimum number of backups to keep in the catalog regardless of the retention period used. e.g. 1,2,3
- **[extra backup parameters]**: Extra parameters that can be used with pg_dump / pg_dumpall
- **[job status]**
  - ACTIVE: Backup job activated and in production.
  - STOPPED: Backup job stopped.

The default value for a parameter is shown between brackets []). If the user does not define any value, the default value will be used. This command can be run with or without parameters. e.g.:

```
[pgbbackman]$ update_backup_definition
--------------------------------------------------------
# DefID []: 12
# Minutes cron [41]:
# Hours cron [01]:
# Day-month cron [*]:
# Month cron [*]:
# Weekday cron [*]:
# Retention period [7 days]: 5 days
# Retention redundancy [1]:
# Extra backup parameters []:
# Job status [STOPPED]: active
# Remarks []:
# Are all values to update correct (yes/no): yes
--------------------------------------------------------
[Done] Backup definition DefID: 12 updated.
```

**update_backup_server**

This command updates the information of a backup server.
update_backup_server [SrvID | FQDN] [remarks]

Parameters:

- **[SrvID|FQDN]**: SrvID in PgBackMan or FQDN of the backup server
- **[remarks]**: Remarks

The default value for a parameter is shown between brackets []). If the user does not define any value, the default value will be used.

This command can be run with or without parameters. e.g.:

```bash
[pgbackman]$ update_backup_server
------------------------------------------
# SrvID / FQDN []: 1
# Remarks []: Backup server - 01
# Are all values to update correct (yes/no): yes
------------------------------------------
[Done] Backup server with SrvID: 1 updated.
```

**update_backup_server_config**

This command updates the default configuration of a backup server.

update_backup_server_config [SrvID / FQDN] [PgSQL_bin_9.0] [PgSQL_bin_9.1] [PgSQL_bin_9.2] [PgSQL_bin_9.3] [PgSQL_bin_9.4] [root_backup_dir]

Parameters:

- **[SrvID|FQDN]**: SrvID in PgBackMan or FQDN of the backup server
- **[PgSQL_bin_9.0]**: Directory with postgresQL 9.0 bin software
- **[PgSQL_bin_9.1]**: Directory with postgresQL 9.1 bin software
- **[PgSQL_bin_9.2]**: Directory with postgresQL 9.2 bin software
- **[PgSQL_bin_9.3]**: Directory with postgresQL 9.3 bin software
- **[PgSQL_bin_9.4]**: Directory with postgresQL 9.4 bin software
- **[root_backup_dir]**: Backup directory used by PgBackMan.

The default value for a parameter is shown between brackets []). If the user does not define any value, the default value will be used.

This command can be run with or without parameters. e.g.:

```bash
[pgbackman]$ update_backup_server_config
------------------------------------------
# SrvID / FQDN []: 1
```
updatepgsql_node

This command updates the information of a PgSQL node.

updatepgsql_node [NodeID | FQDN]
 [pgport]
 [admin_user]
 [status]
 [remarks]

Parameters:

- [NodeID | FQDN]: NodeID in PgBackMan or FQDN of the PgSQL node to update.
- [pgport]: PostgreSQL port
- [admin_user]: PostgreSQL admin user
- [status]:
  - RUNNING: PostgreSQL node running and online
  - DOWN: PostgreSQL node not online.
- [remarks]: Remarks

All backup definitions from a PgSQL node will be started/stopped automatically if the PgSQL node gets
the status changed to RUNNING/DOWN.

The default value for a parameter is shown between brackets []. If the user does not define any value,
the default value will be used. This command can be run with or without parameters. e.g:

[pgbackman]$ updatepgsql_node

# NodeID / FQDN [ ]: 1
# Port [5432]:
# Admin user [postgres]:
# Status[RUNNING]: stopped
# Remarks [ ]: Testing update

# Are all values to update correct (yes/no): yes

[Done] PgSQL node with NodeID: 1 updated.
### update_pgsql_node_config

This command updates the default configuration parameters of a PgSQL node.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>update_pgsql_node_config [NodeID / FQDN] [min_cron interval] [hours_cron interval] [daymonth_cron] [month_cron] [weekday_cron] [backup code] [retention period] [retention redundancy] [automatic deletion retention] [extra backup parameters] [extra restore parameters] [backup job status] [domain] [logs email] [admin user] [pgport] [pgnode backup dir] [pgnode crontab file] [pgnode status]</td>
<td></td>
</tr>
</tbody>
</table>

#### Parameters:

- **[NodeID / FQDN]:** NodeID in PgBackMan or FQDN of the PgSQL node to update.
- **[min_cron interval]:** Backup minutes interval, e.g. 01-59
- **[hours_cron interval]:** Backup hours interval, e.g. 01-06
- **[daymonth_cron]:** Backup day-month cron
- **[month_cron]:** Backup month cron
- **[weekday_cron]:** Backup weekday cron
- **[backup code]:** Backup job code
- **[retention period]:** Retention period for a backup job
- **[retention redundancy]:** Retention redundancy for a backup job
- **[automatic deletion retention]:** Retention period that backups for a dbname will be kept in the catalog after the dbname has been deleted in the PgSQL node. This parameter overrides [retention period] and [retention redundancy] if the database has been deleted in the PgSQL node.
- **[extra backup parameters]:** Extra backup parameters
- **[extra restore parameters]:** Extra restore parameters
- **[backup job status]:** Backup job status
- **[domain]:** Default domain
- **[logs email]:** E-mail to send logs
- **[admin user]:** PostgreSQL admin user
- **[pgport]:** PostgreSQL port
- **[pgnode backup dir]:** Directory to save pgbackman information for a pgnode
- **[pgnode crontab file]:** Crontab file for PgSQL node in the backup server
• **[pgnode status]**: PostgreSQL node status

The default value for a parameter is shown between brackets []. If the user does not define any value, the default value will be used. This command can be run with or without parameters. e.g:

```
[pgbackman]$ update_pgsql_node_config

# NodeID / FQDN []: 1
# Minutes cron interval [01-59]:
# Hours cron interval [01-06]:
# Day-month cron [*]:
# Month cron [*]:
# Weekday cron [*]:
# Backup code [FULL]:
# Retention period [7 days]: 5 days
# Retention redundancy [1]:
# Automatic deletion retention [14 days]: 30 days
# Extra backup parameters []:
# Extra restore parameters []:
# Backup Job status [ACTIVE]:
# Domain [example.net]:
# Logs e-mail [example@example.net]:
# PostgreSQL admin user [postgres]:
# Port [5432]:
# Backup directory [/srv/pgbackman/pgsql_node_1]:
# Crontab file [/etc/cron.d/pgsql_node_1]:
# PgSQL node status [STOPPED]:
# Are all values to update correct (yes/no): yes

[Done] Default configuration parameters for NodeID: 1 updated.
```

## About backups in PostgreSQL

Taking backups is an important administrative task that can have some disastrous consequences if it is not done right. The use of RAID configurations in your storage system, replication between nodes, clustering and trusting 100% that your SAN will be up ARE NOT backup strategies. These measures are necessary for HA (High availability) but do not replace the necessity of taking backups of our databases.

There are two different types of backup that can be use with PostgreSQL to implement a good backup and restore strategy. They are:

- **Physical backups**
- **Logical backups**

Regardless of the type of backup used to backup your databases, one needs a good *backup and restore plan* that takes into account intervals, retention policies and performance issues for a backup and the time needed to get a full restoration of a database.
Physical backups
This type of backup takes copies of the files where the PostgreSQL saves the databases. There are several techniques that can be used to take physical backups and we are not going to explain them here. Check Chapter 24. Backup and Restore of the PostgreSQL documentation for more information.

The important thing with physical backups is that some of these techniques together with continuous archiving of write ahead log (WAL) files can be used to implement PITR (Point in time recovery) backups and achieve a full disaster recovery solution.

There are several solutions that can be used for managing PITR backups, such as PITRTools, OmniPITR, and Barman.

Logical backups
PostgreSQL has two utilities, pg_dump and pg_dumpall, for taking logical backups of databases. They take a snapshot of a database at a given moment.

These utilities take consistent backups of a database or the whole cluster even if the databases are being used concurrently. At the same time pg_dump and pg_dumpall do not block other users accessing the database when backups are being taken.

Even though a backup or snapshot created with pg_dump or pg_dumpall can never guarantee a full disaster recovery of all data changed between the moment when the backup was taken and the moment of a future crash, they are still necessary if you need to archive versions of a database, move databases between PostgreSQL nodes and clone databases between production / pre-production and/or development servers.

Nevertheless, logical backups give us a great flexibility in several situations and are also an easy way of taking backups of databases not requiring PITR backups.

When taking a backup of a database we need the following information to be sure we can make a restoration that includes 100% of the data and definitions from the target database:

1. Database schema
2. Database data
3. Roles owning objects in the database
4. Roles with privileges on objects in the database
5. Roles with privileges on the database or schemas
6. Creation of all the roles owning something or with privileges
7. Configuration parameters defined explicitly for a role
8. Configuration parameters defined explicitly for the database

Unfortunately all this information cannot be obtained in a single execution for only one database. 1, 2, 3 and 4 can be obtained with pg_dump. 5, 7 and 8 can be obtained with a full pg_dumpall and 6 either with a pg_dumpall -r or a full pg_dumpall.

At the same time, pg_dumpall will return all this information for all databases in a cluster, not only the database one wants to take a backup of.

This is something that PostgreSQL will have to improve in the future so it gets easier to take a backup/snapshot of a database in a single execution.

In the meantime, PgBackMan takes care of all this and it delivers all the information needed to run a 100% restoration of a database when we define a backup in the system.
Submitting a bug

PgBakMan has been extensively tested, and is currently being used in production at the University of Oslo. However, as any software, PgBakMan is not bug free.
If you discover a bug, please file a bug through the GitHub Issue page for the project at: https://github.com/rafaelma/pgbackman/issues

Authors

In alphabetical order:

Rafael Martinez Guerrero
E-mail: rafael@postgresql.org.es / rafael@usit.uio.no
PostgreSQL-es / University Center for Information Technology (USIT), University of Oslo, Norway

License and Contributions

PgBackMan is the property of Rafael Martinez Guerrero / PostgreSQL-es and USIT-University of Oslo, and its code is distributed under GNU General Public License 3.

Copyright © 2013-2014 Rafael Martinez Guerrero / PostgreSQL-es
Copyright © 2014 USIT-University of Oslo.