pgmoneta

User Guide

Contents

1 Introduction

pgmoneta is a backup / restore solution for PostgreSQL.

Ideally, you would not need to do backups and disaster recovery, but that isn't how the real World works.

Possible scenarios that could happen

- Data corruption
- System failure
- Human error
- Natural disaster

and then it is up to the database administrator to get the database system back on-line, and to the correct recovery point.

Two key factors are

- Recovery Point Objective (RPO): Maximum targeted period in which data might be lost from an IT service due to a major incident
- Recovery Time Objective (RTO): The targeted duration of time and a service level within which a business process must be restored after a disaster (or disruption) in order to avoid unacceptable consequences associated with a break in business continuity

You would like to have both of these as close to zero as possible, since RPO of 0 means that you won't lose data, and RTO of 0 means that your system recovers at once. However, that is easier said than done.

pgmoneta is focused on having features that will allow database systems to get as close to these goals as possible such that high availability of 99.99% or more can be implemented, and monitored through standard tools.

pgmoneta is named after the Roman Goddess of Memory.

1.1 Features

- Full backup
- Restore
- Compression (gzip, zstd, lz4, bzip2)
- AES encryption support
- Symlink support
- WAL shipping support

- Hot standby
- Prometheus support
- Remote management
- Offline mode
- Transport Layer Security (TLS) v1.2+ support
- Daemon mode
- User vault

1.2 Platforms

The supported platforms are

- Fedora 38+
- RHEL 9
- RockyLinux 9
- FreeBSD
- OpenBSD

2 Installation

2.1 Fedora

You need to add the PostgreSQL YUM repository, for example for Fedora 40

```
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/F
   -40-x86_64/pgdg-fedora-repo-latest.noarch.rpm
```

and do the install via

```
dnf install -y pgmoneta
```

Additional information

- PostgreSQL YUM
- Linux downloads

2.2 RHEL 9 / RockyLinux 9

x86_64

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest
   -9.noarch.rpm
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL
   -9-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

aarch64

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest
   -9.noarch.rpm
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL
   -9-aarch64/pgdg-redhat-repo-latest.noarch.rpm
```

and do the install via

```
dnf install -y pgmoneta
```

2.3 Compiling the source

We recommend using Fedora to test and run **pgmoneta**, but other Linux systems, FreeBSD and MacOS are also supported.

pgmoneta requires

- clang
- cmake
- make
- libev
- OpenSSL
- zlib
- zstd
- lz4
- bzip2
- systemd
- rst2man
- libssh
- libcurl
- libarchive

```
dnf install git gcc clang clang-analyzer cmake make libev libev-devel \
            openssl openssl-devel \
            systemd systemd-devel zlib zlib-devel \
            libzstd libzstd-devel \
            lz4 lz4-devel libssh libssh-devel \
            libcurl libcurl-devel \
            python3-docutils libatomic \
            bzip2 bzip2-devel \
            libarchive libarchive-devel
```

Alternative gcc can be used.

2.3.1 RHEL / RockyLinux

On RHEL / Rocky, before you install the required packages some additional repositories need to be enabled or installed first.

First you need to install the subscription-manager

```
dnf install subscription-manager
```

It is ok to disregard the registration and subscription warning.

Otherwise, if you have a Red Hat corporate account (you need to specify the company/organization name in your account), you can register using

```
subscription-manager register --username <your-account-email-or-login> --
    password <your-password> --auto-attach
```

Then install the EPEL repository,

```
dnf install epel-release
```

Then to enable powertools

```
dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
dnf config-manager --set-enabled crb
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.
   noarch.rpm
```

Then use the `dnf` command for **pgmoneta** to install the required packages.

2.3.2 FreeBSD

On FreeBSD, `pkg` is used instead of `dnf` or `yum`.

Use `pkg install <package name>` to install the following packages

```
git gcc cmake libev openssl libssh zlib-ng zstd liblz4 bzip2 curl \
    py39-docutils libarchive
```

2.3.3 Build

2.3.3.1 Release build The following commands will install **pgmoneta** in the `/usr/local` hierarchy.

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ..
make
sudo make install
```

See RPM for how to build a RPM of **pgmoneta**.

2.3.3.2 Debug build The following commands will create a `DEBUG` version of **pgmoneta**.

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
mkdir build
cd build
cmake -DCMAKE_C_COMPILER=clang -DCMAKE_BUILD_TYPE=Debug ..
make
```

You can do

```
cmake -DCMAKE_C_COMPILER=clang -DCMAKE_BUILD_TYPE=Debug -DCMAKE_C_FLAGS="-
   DCORE_DEBUG" ..
```

in order to get information from the core libraries too.

2.4 Compiling the documentation

pgmoneta's documentation requires

- pandoc
- texlive

```
dnf install pandoc texlive-scheme-basic \
            'tex(footnote.sty)' 'tex(footnotebackref.sty)' \
            'tex(pagecolor.sty)' 'tex(hardwrap.sty)' \
            'tex(mdframed.sty)' 'tex(sourcesanspro.sty)' \
            'tex(ly1enc.def)' 'tex(sourcecodepro.sty)' \
            'tex(titling.sty)' 'tex(csquotes.sty)' \
            'tex(zref-abspage.sty)' 'tex(needspace.sty)'
```

You will need the `Eisvogel` template as well which you can install through

```
wget https://github.com/Wandmalfarbe/pandoc-latex-template/releases/
   download/v3.1.0/Eisvogel-3.1.0.tar.gz
tar -xzf Eisvogel-3.1.0.tar.gz
mkdir -p $HOME/.local/share/pandoc/templates
mv Eisvogel-3.1.0/eisvogel.latex $HOME/.local/share/pandoc/templates
```

where `$HOME` is your home directory.

2.4.0.1 Generate API guide

This process is optional. If you choose not to generate the API HTML files, you can opt out of downloading these dependencies, and the process will automatically skip the generation.

Download dependencies

```
dnf install graphviz doxygen
```

2.4.1 Build

These packages will be detected during `cmake` and built as part of the main build.

2.5 Extension installation

When you configure the `extra` parameter in the server section of `pgmoneta.conf`, it requires the server side to have the pgmoneta_ext extension installed to make it work.

The following instructions can help you easily install `pgmoneta_ext`. If you encounter any problems, please refer to the more detailed instructions in the DEVELOPERS documentation.

2.5.1 Install pgmoneta_ext

After you have successfully installed `pgmoneta`, the following commands will help you install `pgmoneta_ext`:

```
dnf install -y pgmoneta_ext
```

You need to add the `pgmoneta_ext` library for PostgreSQL in `postgrersql.conf` as well:

```
shared_preload_libraries = 'pgmoneta_ext'
```

And remember to restart PostgreSQL to make it work.

2.5.2 Verify success

You can use the `postgres` role to test.

1. Log into PostgreSQL

   ```
   psql
   ```

2. Create a new test database

   ```
   CREATE DATABASE testdb;
   ```

3. Enter the database

   ```
   \c testdb
   ```

4. Follow the SQL commands below to check the function

   ```
   DROP EXTENSION IF EXISTS pgmoneta_ext;
   CREATE EXTENSION pgmoneta_ext;
   SELECT pgmoneta_ext_version();
   ```

You should see

```
 pgmoneta_ext_version
----------------------
 0.1.0
(1 row)
```

2.5.3 Granting SUPERUSER Privileges

Some functions in `pgmoneta_ext` require `SUPERUSER` privileges. To enable these, grant the `repl` role superuser privileges using the command below. **Please proceed with caution**: granting superuser privileges bypasses all permission checks, allowing unrestricted access to the database, which can pose security risks. We are committed to enhancing privilege security in future updates.

```
ALTER ROLE repl WITH SUPERUSER;
```

To revoke superuser privileges from the `repl` role, use the following command:

```
ALTER ROLE repl WITH NOSUPERUSER;
```

3 Quick start

Make sure that **pgmoneta** is installed and in your path by using `pgmoneta -?`. You should see

```
pgmoneta 0.16.0
  Backup / restore solution for PostgreSQL

Usage:
  pgmoneta [ -c CONFIG_FILE ] [ -u USERS_FILE ] [ -d ]

Options:
  -c, --config CONFIG_FILE                          Set the path to the
     pgmoneta.conf file
  -h, --host HOST                                   Set the host name
  -p, --port PORT                                   Set the port number
  -U, --user USERNAME                               Set the user name
  -P, --password PASSWORD                           Set the password
  -L, --logfile FILE                                Set the log file
  -v, --verbose                                     Output text string of
     result
  -V, --version                                     Display version
     information
  -F, --format text|json|raw                        Set the output format
  -C, --compress none|gz|zstd|lz4|bz2               Compress the wire
     protocol
  -E, --encrypt none|aes|aes256|aes192|aes128       Encrypt the wire
     protocol
  -?, --help                                        Display help

Commands:
  annotate                 Annotate a backup with comments
  archive                  Archive a backup from a server
  backup                   Backup a server
  clear <what>             Clear data, with:
                           - 'prometheus' to reset the Prometheus
                              statistics
  compress                 Compress a file using configured method
  conf <action>            Manage the configuration, with one of
     subcommands:
                           - 'get' to obtain information about a runtime
                              configuration value
                             conf get <parameter_name>
                           - 'ls' to print the configurations used
                           - 'reload' to reload the configuration
                           - 'set' to modify a configuration value;
                             conf set <parameter_name> <parameter_value>;
  decompress               Decompress a file using configured method
  decrypt                  Decrypt a file using master-key
  delete                   Delete a backup from a server
  encrypt                  Encrypt a file using master-key
```

```
  expunge                   Expunge a backup from a server
  info                      Information about a backup
  list-backup               List the backups for a server
  ping                      Check if pgmoneta is alive
  restore                   Restore a backup from a server
  retain                    Retain a backup from a server
  shutdown                  Shutdown pgmoneta
  status [details]          Status of pgmoneta, with optional details
  verify                    Verify a backup from a server

pgmoneta: https://pgmoneta.github.io/
Report bugs: https://github.com/pgmoneta/pgmoneta/issues
```

If you encounter any issues following the above steps, you can refer to the **Installation** chapter to see how to install or compile pgmoneta on your system.

3.1 Configuration

Lets create a simple configuration file called `pgmoneta.conf` with the content

```
[pgmoneta]
host = *
metrics = 5001

base_dir = /home/pgmoneta

compression = zstd

retention = 7

log_type = file
log_level = info
log_path = /tmp/pgmoneta.log

unix_socket_dir = /tmp/

[primary]
host = localhost
port = 5432
user = repl
wal_slot = repl
```

In our main section called `[pgmoneta]` we setup **pgmoneta** to listen on all network addresses. We will enable Prometheus metrics on port 5001 and have the backups live in the `/home/pgmoneta` directory. All backups are being compressed with zstd and kept for 7 days. Logging will be performed at `info` level and put in a file called `/tmp/pgmoneta.log`. Last we specify the location of the `unix_socket_dir` used for management operations and the path for the PostgreSQL command

line tools.

Next we create a section called `[primary]` which has the information about our PostgreSQL instance. In this case it is running on `localhost` on port 5432 and we will use the `repl` user account to connect, and the Write+Ahead slot will be named `repl` as well.

The `repl` user must have the `REPLICATION` role and have access to the `postgres` database, so for example

```
CREATE ROLE repl WITH LOGIN REPLICATION PASSWORD 'secretpassword';
```

and in `pg_hba.conf`

```
local   postgres        repl                    scram-sha-256
host    postgres        repl    127.0.0.1/32    scram-sha-256
host    postgres        repl    ::1/128         scram-sha-256
host    replication     repl    127.0.0.1/32    scram-sha-256
host    replication     repl    ::1/128         scram-sha-256
```

The authentication type should be based on `postgresql.conf`'s `password_encryption` value.

Then, create a physical replication slot that will be used for Write-Ahead Log streaming, like

```
SELECT pg_create_physical_replication_slot('repl', true, false);
```

Alternatively, configure automatically slot creation by adding `create_slot = yes` to `[pgmoneta]` or corresponding server section.

We will need a user vault for the `repl` account, so the following commands will add a master key, and the `repl` password. The master key should be longer than 8 characters.

```
pgmoneta-admin master-key
pgmoneta-admin -f pgmoneta_users.conf user add
```

For scripted use, the master key and user password can be provided using the `PGMONETA_PASSWORD` environment variable.

We are now ready to run **pgmoneta**.

See the **Configuration** charpter for all configuration options.

3.2 Running

We will run **pgmoneta** using the command

```
pgmoneta -c pgmoneta.conf -u pgmoneta_users.conf
```

If this doesn't give an error, then we are ready to do backups.

pgmoneta is stopped by pressing Ctrl-C (^C) in the console where you started it, or by sending the SIGTERM signal to the process using kill <pid>.

3.3 Run-time administration

pgmoneta has a run-time administration tool called pgmoneta-cli.

You can see the commands it supports by using pgmoneta-cli -? which will give

```
pgmoneta-cli 0.12.0
  Command line utility for pgmoneta

Usage:
  pgmoneta-cli [ -c CONFIG_FILE ] [ COMMAND ]

Options:
  -c, --config CONFIG_FILE                          Set the path to the
     pgmoneta.conf file
  -h, --host HOST                                   Set the host name
  -p, --port PORT                                   Set the port number
  -U, --user USERNAME                               Set the user name
  -P, --password PASSWORD                           Set the password
  -L, --logfile FILE                                Set the log file
  -v, --verbose                                     Output text string of
     result
  -V, --version                                     Display version
     information
  -F, --format text|json|raw                        Set the output format
  -C, --compress none|gz|zstd|lz4|bz2               Compress the wire
     protocol
  -E, --encrypt none|aes|aes256|aes192|aes128       Encrypt the wire
     protocol
  -?, --help                                        Display help

Commands:
  backup                  Backup a server
  list-backup             List the backups for a server
  restore                 Restore a backup from a server
  verify                  Verify a backup from a server
  archive                 Archive a backup from a server
  delete                  Delete a backup from a server
  retain                  Retain a backup from a server
  expunge                 Expunge a backup from a server
  encrypt                 Encrypt a file using master-key
  decrypt                 Decrypt a file using master-key
  ping                    Check if pgmoneta is alive
  shutdown                Shutdown pgmoneta
```

```
  status [details]          Status of pgmoneta, with optional details
  conf <action>             Manage the configuration, with one of
     subcommands:
                            - 'reload' to reload the configuration
  clear <what>              Clear data, with:
                            - 'prometheus' to reset the Prometheus
                               statistics
```

This tool can be used on the machine running **pgmoneta** to do a backup like

```
pgmoneta-cli -c pgmoneta.conf backup primary
```

A restore would be

```
pgmoneta-cli -c pgmoneta.conf restore primary <timestamp> /path/to/restore
```

To shutdown pgmoneta you would use

```
pgmoneta-cli -c pgmoneta.conf shutdown
```

Check the outcome of the operations by verifying the exit code, like

```
echo $?
```

or by using the `-v` flag.

If pgmoneta has both Transport Layer Security (TLS) and `management` enabled then `pgmoneta-cli` can connect with TLS using the files `~/.pgmoneta/pgmoneta.key` (must be 0600 permission), `~/.pgmoneta/pgmoneta.crt` and `~/.pgmoneta/root.crt`.

3.4 Administration

pgmoneta has an administration tool called `pgmoneta-admin`, which is used to control user registration with **pgmoneta**.

You can see the commands it supports by using `pgmoneta-admin -?` which will give

```
pgmoneta-admin 0.12.0
  Administration utility for pgmoneta

Usage:
  pgmoneta-admin [ -f FILE ] [ COMMAND ]

Options:
  -f, --file FILE         Set the path to a user file
  -U, --user USER         Set the user name
  -P, --password PASSWORD Set the password for the user
  -g, --generate          Generate a password
```

```
  -l, --length            Password length
  -V, --version           Display version information
  -?, --help              Display help

Commands:
  master-key              Create or update the master key
  user <subcommand>       Manage a specific user, where <subcommand> can
     be
                          - add  to add a new user
                          - del  to remove an existing user
                          - edit to change the password for an existing
                             user
                          - ls   to list all available users
```

In order to set the master key for all users you can use

```
pgmoneta-admin -g master-key
```

The master key must be at least 8 characters.

Then use the other commands to add, update, remove or list the current user names, f.ex.

```
pgmoneta-admin -f pgmoneta_users.conf user add
```

3.5 Next Steps

Next steps in improving pgmoneta's configuration could be

- Update `pgmoneta.conf` with the required settings for your system
- Enable Transport Layer Security v1.2+ (TLS) for administrator access

See Configuration for more information on these subjects.

4 Configuration

The configuration is loaded from either the path specified by the `-c` flag or `/etc/pgmoneta/pgmoneta.conf`.

The configuration of **pgmoneta** is split into sections using the `[` and `]` characters.

The main section, called `[pgmoneta]`, is where you configure the overall properties of **pgmoneta**.

Other sections doesn't have any requirements to their naming so you can give them meaningful names like `[primary]` for the primary PostgreSQL instance.

All properties are in the format `key = value`.

The characters `#` and `;` can be used for comments; must be the first character on the line.

The `Bool` data type supports the following values: `on`, `yes`, `1`, **`true`**, `off`, `no`, `0` and **`false`**.

See a sample configuration for running **pgmoneta** on `localhost`.

4.1 pgmoneta

Property	Default	Unit	Required	Description
host		String	Yes	The bind address for pgmoneta
unix_socket_dir		String	Yes	The Unix Domain Socket location
base_dir		String	Yes	The base directory for the backup
metrics	0	Int	No	The metrics port (disable = 0)

Property	Default	Unit	Required	Description
metrics_cache_max_age	0	String	No	The time to keep a Prometheus (metrics) response in cache. If this value is specified without units, it is taken as seconds. Setting this parameter to 0 disables caching. It supports the following units as suffixes: ‘S’ for seconds (default), ‘M’ for minutes, ‘H’ for hours, ‘D’ for days, and ‘W’ for weeks.

Property	Default	Unit	Required	Description
metrics_cache_max_size	256k	String	No	The maximum amount of data to keep in cache when serving Prometheus responses. Changes require restart. This parameter determines the size of memory allocated for the cache even if `metrics_cache_max_age` or `metrics` are disabled. Its value, however, is taken into account only if `metrics_cache_max_age` is set to a non-zero value. Supports suffixes: 'B' (bytes), the default if omitted, 'K' or 'KB' (kilobytes), 'M' or 'MB' (megabytes), 'G' or 'GB' (gigabytes).
management	0	Int	No	The remote management port (disable = 0)
compression	zstd	String	No	The compression type (none, gzip, client-gzip, server-gzip, zstd, client-zstd, server-zstd, lz4, client-lz4, server-lz4, bzip2, client-bzip2)

Property	Default	Unit	Required	Description
compression_level	3	Int	No	The compression level
workers	0	Int	No	The number of workers that each process can use for its work. Use 0 to disable. Maximum is CPU count
workspace	/tmp/pgmoneta-workspace/	String	No	The directory for the workspace that incremental backup can use for its work
storage_engine	local	String	No	The storage engine type (local, ssh, s3, azure)

Property	Default	Unit	Required	Description
encryption	none	String	No	The encryption mode for encrypt wal and data `none`: No encryption `aes` or `aes-256` or `aes-256-cbc`: AES CBC (Cipher Block Chaining) mode with 256 bit key length `aes-192` or `aes-192-cbc`: AES CBC mode with 192 bit key length `aes-128` or `aes-128-cbc`: AES CBC mode with 128 bit key length `aes-256-ctr`: AES CTR (Counter) mode with 256 bit key length `aes-192-ctr`: AES CTR mode with 192 bit key length `aes-128-ctr`: AES CTR mode with 128 bit key length
create_slot	no	Bool	No	Create a replication slot for all server. Valid values are: yes, no
ssh_hostname		String	Yes	Defines the hostname of the remote system for connection

Property	Default	Unit	Required	Description
ssh_username		String	Yes	Defines the username of the remote system for connection
ssh_base_dir		String	Yes	The base directory for the remote backup
ssh_ciphers	aes-256-ctr, aes-192-ctr, aes-128-ctr	String	No	The supported ciphers for communication. `aes` or `aes-256` or `aes-256-cbc`: AES CBC (Cipher Block Chaining) mode with 256 bit key length `aes-192` or `aes-192-cbc`: AES CBC mode with 192 bit key length `aes-128` or `aes-128-cbc`: AES CBC mode with 128 bit key length `aes-256-ctr`: AES CTR (Counter) mode with 256 bit key length `aes-192-ctr`: AES CTR mode with 192 bit key length `aes-128-ctr`: AES CTR mode with 128 bit key length. Otherwise verbatim
s3_aws_region		String	Yes	The AWS region

Property	Default	Unit	Required	Description
s3_access_key_id		String	Yes	The IAM access key ID
s3_secret_access_key		String	Yes	The IAM secret access key
s3_bucket		String	Yes	The AWS S3 bucket name
s3_base_dir		String	Yes	The base directory for the S3 bucket
azure_storage_account		String	Yes	The Azure storage account name
azure_container		String	Yes	The Azure container name
azure_shared_key		String	Yes	The Azure storage account key
azure_base_dir		String	Yes	The base directory for the Azure container
retention	7, - , - , -	Array	No	The retention time in days, weeks, months, years
retention_interval	300	Int	No	The retention check interval
log_type	console	String	No	The logging type (console, file, syslog)

Property	Default	Unit	Required	Description
log_level	info	String	No	The logging level, any of the (case insensitive) strings `FATAL`, `ERROR`, `WARN`, `INFO` and `DEBUG` (that can be more specific as `DEBUG1` thru `DEBUG5`). Debug level greater than 5 will be set to `DEBUG5`. Not recognized values will make the log_level be `INFO`
log_path	pgmoneta.	String	No	The log file location. Can be a strftime(3) compatible string.
log_rotation_age	0	String	No	The time after which log file rotation is triggered. If this value is specified without units, it is taken as seconds. Setting this parameter to 0 disables log rotation based on time. It supports the following units as suffixes: ‘S’ for seconds (default), ‘M’ for minutes, ‘H’ for hours, ‘D’ for days, and ‘W’ for weeks.

Property	Default	Unit	Required	Description
log_rotation_size	0	String	No	The size of the log file that will trigger a log rotation. Supports suffixes: 'B' (bytes), the default if omitted, 'K' or 'KB' (kilobytes), 'M' or 'MB' (megabytes), 'G' or 'GB' (gigabytes). A value of `0` (with or without suffix) disables.
log_line_prefix	%Y-%m-%d %H:%M:%S	String	No	A strftime(3) compatible string to use as prefix for every log line. Must be quoted if contains spaces.
log_mode	append	String	No	Append to or create the log file (append, create)

Property	Default	Unit	Required	Description
blocking_timeout	30	String	No	The number of seconds the process will be blocking for a connection. If this value is specified without units, it is taken as seconds. Setting this parameter to 0 disables it. It supports the following units as suffixes: 'S' for seconds (default), 'M' for minutes, 'H' for hours, 'D' for days, and 'W' for weeks.
tls	`off`	Bool	No	Enable Transport Layer Security (TLS)
tls_cert_file		String	No	Certificate file for TLS. This file must be owned by either the user running pgmoneta or root.
tls_key_file		String	No	Private key file for TLS. This file must be owned by either the user running pgmoneta or root. Additionally permissions must be at least `0640` when owned by root or `0600` otherwise.

Property	Default	Unit	Required	Description
tls_ca_file		String	No	Certificate Authority (CA) file for TLS. This file must be owned by either the user running pgmoneta or root.
metrics_cert_file		String	No	Certificate file for TLS for Prometheus metrics. This file must be owned by either the user running pgmoneta or root.
metrics_key_file		String	No	Private key file for TLS for Prometheus metrics. This file must be owned by either the user running pgmoneta or root. Additionally permissions must be at least `0640` when owned by root or `0600` otherwise.
metrics_ca_file		String	No	Certificate Authority (CA) file for TLS for Prometheus metrics. This file must be owned by either the user running pgmoneta or root.

Property	Default	Unit	Required	Description
libev	`auto`	String	No	Select the libev backend to use. Valid options: `auto`, `select`, `poll`, `epoll`, `iouring`, `devpoll` and `port`
backup_max_rate	0	Int	No	The number of bytes of tokens added every one second to limit the backup rate
network_max_rate	0	Int	No	The number of bytes of tokens added every one second to limit the netowrk backup rate
manifest	sha256	String	No	The hash algoritm for the manifest. Valid options: `crc32c`, `sha224`, `sha256`, `sha384` and `sha512`
keep_alive	on	Bool	No	Have `SO_KEEPALIVE` on sockets
nodelay	on	Bool	No	Have `TCP_NODELAY` on sockets
non_blocking	on	Bool	No	Have `O_NONBLOCK` on sockets
backlog	16	Int	No	The backlog for `listen()`. Minimum 16

Property	Default	Unit	Required	Description
hugepage	**try**	String	No	Huge page support (off, **try**, on)
pidfile		String	No	Path to the PID file. If not specified, it will be automatically set to unix_socket_dir/pgmoneta.<host>.pid where <host> is the value of the host parameter or all if host = *.

Property	Default	Unit	Required	Description
update_process_title	`verbose`	String	No	The behavior for updating the operating system process title. Allowed settings are: `never` (or `off`), does not update the process title; `strict` to set the process title without overriding the existing initial process title length; `minimal` to set the process title to the base description; `verbose` (or `full`) to set the process title to the full description. Please note that `strict` and `minimal` are honored only on those systems that do not provide a native way to set the process title (e.g., Linux). On other systems, there is no difference between `strict` and `minimal` and the assumed behaviour is `minimal` even if `strict` is used. `never` and `verbose` are always honored, on every system. On Linux systems the process title is always trimmed to 255 characters, while on

4.2 Server section

Property	Default	Unit	Required	Description
host		String	Yes	The address of the PostgreSQL instance
port		Int	Yes	The port of the PostgreSQL instance
user		String	Yes	The replication user name
wal_slot		String	Yes	The replication slot for WAL
create_slot	no	Bool	No	Create a replication slot for this server. Valid values are: yes, no
follow		String	No	Failover to this server if follow server fails
retention		Array	No	The retention for the server in days, weeks, months, years
wal_shipping		String	No	The WAL shipping directory
workspace	/tmp/pgmoneta-workspace/	String	No	The directory for the workspace that incremental backup can use for its work
hot_standby		String	No	Hot standby directory
hot_standby_overrides		String	No	Files to override in the hot standby directory
hot_standby_tablespaces		String	No	Tablespace mappings for the hot standby. Syntax is [from -> to,?]+

Property	Default	Unit	Required	Description
workers	-1	Int	No	The number of workers that each process can use for its work. Use 0 to disable, -1 means use the global settting. Maximum is CPU count
backup_max_rate	-1	Int	No	The number of bytes of tokens added every one second to limit the backup rate. Use 0 to disable, -1 means use the global settting
network_max_rate	-1	Int	No	The number of bytes of tokens added every one second to limit the netowrk backup rate. Use 0 to disable, -1 means use the global settting
manifest	sha256	String	No	The hash algoritm for the manifest. Valid options: `crc32c`, `sha224`, `sha256`, `sha384` and `sha512`
tls_cert_file		String	No	Certificate file for TLS. This file must be owned by either the user running pgmoneta or root.

Property	Default	Unit	Required	Description
tls_key_file		String	No	Private key file for TLS. This file must be owned by either the user running pgmoneta or root. Additionally permissions must be at least `0640` when owned by root or `0600` otherwise.
tls_ca_file		String	No	Certificate Authority (CA) file for TLS. This file must be owned by either the user running pgmoneta or root.
extra		String	No	The source directory for retrieval on the server side (details are in the extra section)

The `user` specified must have the `REPLICATION` option in order to stream the Write-Ahead Log (WAL), and must have access to the `postgres` database in order to get the necessary configuration parameters.

Note, that PostgreSQL 13+ is required, as well as having `wal_level` at `replica` or `logical` level.

Note, that if `host` starts with a `/` it represents a path and **pgmoneta** will connect using a Unix Domain Socket.

4.2.1 extra parameter

The `extra` configuration is set in the server section. It is not required, but if you configure this parameter, when you perform a backup using the CLI `pgmoneta-cli -c pgmoneta.conf backup primary`, it will also copy all specified files on the server side and send them back to the client side.

This extra feature requires the server side to install the pgmoneta_ext extension and also make the role repl a SUPERUSER (this will be improved in the future). Currently, this feature is only available to the SUPERUSER role.

You can set up pgmoneta_ext by following the README to easily install the extension. There are also more detailed instructions available in the DEVELOPERS documentation.

The format for the extra parameter is a path to a file or directory. You can list more than one file or directory separated by commas. The format is as follows:

```
extra = /tmp/myfile1, /tmp/myfile2, /tmp/mydir1, /tmp/mydir2
```

4.3 pgmoneta_users configuration

The pgmoneta_users configuration defines the users known to the system. This file is created and managed through the pgmoneta-admin tool.

The configuration is loaded from either the path specified by the -u flag or /etc/pgmoneta/pgmoneta_users.conf.

4.4 pgmoneta_admins configuration

The pgmoneta_admins configuration defines the administrators known to the system. This file is created and managed through the pgmoneta-admin tool.

The configuration is loaded from either the path specified by the -A flag or /etc/pgmoneta/pgmoneta_admins.conf.

If pgmoneta has both Transport Layer Security (TLS) and management enabled then pgmoneta-cli can connect with TLS using the files ~/.pgmoneta/pgmoneta.key (must be 0600 permission), ~/.pgmoneta/pgmoneta.crt and ~/.pgmoneta/root.crt.

5 pgmoneta_walinfo configuration

The pgmoneta_walinfo configuration defines the info needed for walinfo to work.

The configuration is loaded from either the path specified by the -c flag or /etc/pgmoneta/pgmoneta_walinfo.conf if -c wasn't provided.

5.1 [pgmoneta_walinfo]

Property	Default	Unit	Required	Description
log_type	console	String	No	The logging type (console, file, syslog)
log_level	info	String	No	The logging level, any of the (case insensitive) strings `FATAL`, `ERROR`, `WARN`, `INFO` and `DEBUG` (that can be more specific as `DEBUG1` thru `DEBUG5`). Debug level greater than 5 will be set to `DEBUG5`. Not recognized values will make the log_level be `INFO`
log_path	pgmoneta.log	String	No	The log file location. Can be a strftime(3) compatible string.

5.2 Server section

Property	Default	Unit	Required	Description
host		String	Yes	The address of the PostgreSQL instance
port		Int	Yes	The port of the PostgreSQL instance
user		String	Yes	The replication user name

6 Tutorials

6.1 Install pgmoneta

This tutorial will show you how to do a simple installation of **pgmoneta**.

At the end of this tutorial you will have a backup of a PostgreSQL cluster, and will be streaming Write-Ahead Log (WAL) to **pgmoneta**.

Please note that inside the brackets at the end of each step it's the user account you should be using, switch the account when needed.

6.1.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

For RPM based distributions such as Fedora and RHEL you can add the PostgreSQL YUM repository and do the install via

```
dnf -qy module disable postgresql
dnf install -y postgresql13 postgresql13-server pgmoneta
```

6.1.2 Initialize cluster

```
export PATH=/usr/pgsql-13/bin:$PATH
initdb -k /tmp/pgsql
```

(`postgres` user)

6.1.3 Remove default access

Remove

```
host    all             all             127.0.0.1/32            trust
host    all             all             ::1/128                 trust
host    replication     all             127.0.0.1/32            trust
host    replication     all             ::1/128                 trust
```

from `/tmp/pgsql/pg_hba.conf`

(`postgres` user)

6.1.4 Add access for users and a database

Add

```
host    mydb            myuser          127.0.0.1/32            md5
host    mydb            myuser          ::1/128                 md5
host    postgres        repl            127.0.0.1/32            md5
host    postgres        repl            ::1/128                 md5
host    replication     repl            127.0.0.1/32            md5
host    replication     repl            ::1/128                 md5
```

to `/tmp/pgsql/pg_hba.conf`

Remember to check the value of `password_encryption` in `/tmp/pgsql/postgresql.conf` to setup the correct authentication type.

(`postgres` user)

6.1.5 Make sure that replication level is set

Check that

```
wal_level = replica
```

is set in `/tmp/pgsql/postgresql.conf` - or `logical`

(`postgres` user)

6.1.6 Start PostgreSQL

```
pg_ctl  -D /tmp/pgsql/ start
```

(`postgres` user)

6.1.7 Add users and a database

```
createuser -P myuser
createdb -E UTF8 -O myuser mydb
```

with `mypass` as the password.

Then

```
psql postgres
CREATE ROLE repl WITH LOGIN REPLICATION PASSWORD 'secretpassword';
\q
```

(`postgres` user)

6.1.8 Add Write-Ahead Log (WAL) replication slot

```
psql postgres
SELECT pg_create_physical_replication_slot('repl', true, false);
\q
```

(`postgres` user)

6.1.9 Verify access

For the user `myuser` using `mypass` as the password

```
psql -h localhost -p 5432 -U myuser mydb
\q
```

For the user `repl` using `secretpassword` as the password

```
psql -h localhost -p 5432 -U repl postgres
\q
```

(`postgres` user)

6.1.10 Add pgmoneta user

```
sudo su -
useradd -ms /bin/bash pgmoneta
passwd pgmoneta
exit
```

(`postgres` user)

6.1.11 Create pgmoneta configuration

Switch to the pgmoneta user

```
sudo su -
su - pgmoneta
```

Add the master key and create vault

```
pgmoneta-admin master-key
pgmoneta-admin -f pgmoneta_users.conf -U repl user add
```

You have to choose a password for the master key - remember it !

For scripted use, the master key and user password can be provided using the `PGMONETA_PASSWORD` environment variable.

If you see an error saying `error while loading shared libraries: libpgmoneta.so.0: cannot open shared object` running the above command. you may need to locate where your `libpgmoneta.so.0` is. It could be in `/usr/local/lib` or `/usr/local/lib64` depending on your environment. Add the corresponding directory into `/etc/ld.so.conf`, or alternatively, create a file called `pgmoneta_shared_library.conf` under `/etc/ld.so.conf.d/`, and add your directory into it. Remember to run `ldconfig` to make the change effective

Create the `pgmoneta.conf` configuration

```
cat > pgmoneta.conf
[pgmoneta]
host = *
metrics = 5001
create_slot = yes

base_dir = /home/pgmoneta/backup

compression = zstd

storage_engine = local

retention = 7

log_type = file
log_level = info
log_path = /tmp/pgmoneta.log

unix_socket_dir = /tmp/

[primary]
host = localhost
port = 5432
user = repl
wal_slot = repl
```

and press `Ctrl-d`

(`pgmoneta` user)

6.1.12 Create base directory

```
mkdir backup
```

(pgmoneta user)

6.1.13 Start pgmoneta

```
pgmoneta -c pgmoneta.conf -u pgmoneta_users.conf
```

(pgmoneta user)

6.1.14 Create a backup

In another terminal

```
pgmoneta-cli -c pgmoneta.conf backup primary
```

(pgmoneta user)

6.1.15 View backup

In another terminal

```
pgmoneta-cli -c pgmoneta.conf status details
```

(pgmoneta user)

6.1.16 Shell completion

There is a minimal shell completion support for `pgmoneta-cli` and `pgmoneta-admin`. If you are running such commands from a Bash or Zsh, you can take some advantage of command completion.

6.1.16.1 Installing command completions in Bash There is a completion script into `contrib/shell_comp/pgmoneta_comp.bash` that can be used to help you complete the command line while you are typing.

It is required to source the script into your current shell, for instance by doing:

```
source contrib/shell_comp/pgmoneta_comp.bash
```

At this point, the completions should be active, so you can type the name of one the commands between `pgmoneta-cli` and `pgmoneta-admin` and hit <TAB> to help the command line completion.

6.1.16.2 Installing the command completions on Zsh

In order to enable completion into `zsh` you first need to have `compinit` loaded; ensure your `.zshrc` file contains the following lines:

```
autoload -U compinit
compinit
```

and add the sourcing of the `contrib/shell_comp/pgmoneta_comp.zsh` file into your `~/.zshrc` also associating the `_pgmoneta_cli` and `_pgmoneta_admin` functions to completion by means of `compdef`:

```
source contrib/shell_comp/pgmoneta_comp.zsh
compdef _pgmoneta_cli    pgmoneta-cli
compdef _pgmoneta_admin  pgmoneta-admin
```

If you want completions only for one command, e.g., `pgmoneta-admin`, remove the `compdef` line that references the command you don't want to have automatic completion. At this point, digit the name of a `pgmoneta-cli` or `pgmoneta-admin` command and hit <TAB> to trigger the completion system.

6.2 Remote administration for pgmoneta

This tutorial will show you how to do setup remote management for **pgmoneta**.

6.2.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.2.2 Change the pgmoneta configuration

Change `pgmoneta.conf` to add

```
management = 5002
```

under the `[pgmoneta]` setting, like

```
[pgmoneta]
...
management = 5002
```

(pgmoneta user)

6.2.3 Add pgmoneta admin

```
pgmoneta-admin -f pgmoneta_admins.conf -U admin -P admin1234 user add
```

(pgmoneta user)

6.2.4 Restart pgmoneta

Shutdown pgmoneta and start it again with

```
pgmoneta-cli -c pgmoneta.conf shutdown
pgmoneta -c pgmoneta.conf -u pgmoneta_users.conf -A pgmoneta_admins.conf
```

(pgmoneta user)

6.2.5 Connect via remote administration interface

```
pgmoneta-cli -h localhost -p 5002 -U admin status details
```

and use admin1234 as the password

(pgmoneta user)

6.2.6 Using Transport Level Security for access

You can security the administration level interface by using Transport Level Security (TLS).

It is done by setting the following options,

```
[pgmoneta]
tls_cert_file=/path/to/server.crt
tls_key_file=/path/to/server.key
tls_ca_file=/path/to/root.crt

...
```

in pgmoneta.conf.

The client side setup must go into ~/.pgmoneta/ with the following files

```
~/.pgmoneta/pgmoneta.key
~/.pgmoneta/pgmoneta.crt
~/.pgmoneta/root.crt
```

They must have 0600 permission.

6.3 Prometheus metrics for pgmoneta

This tutorial will show you how to do setup Prometheus metrics for **pgmoneta**.

6.3.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.3.2 Change the pgmoneta configuration

Change pgmoneta.conf to add

```
metrics = 5001
```

under the [pgmoneta] setting, like

```
[pgmoneta]
...
metrics = 5001
```

(pgmoneta user)

6.3.3 Restart pgmoneta

Shutdown pgmoneta and start it again with

```
pgmoneta-cli -c pgmoneta.conf shutdown
pgmoneta -c pgmoneta.conf -u pgmoneta_users.conf
```

(pgmoneta user)

6.3.4 Get Prometheus metrics

You can now access the metrics via

```
http://localhost:5001/metrics
```

(pgmoneta user)

6.3.5 TLS support

To add TLS support for Prometheus metrics, first we need a self-signed certificate. 1. Generate CA key and certificate

```
openssl genrsa -out ca.key 2048
openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out ca.crt -
    subj "/CN=My Local CA"
```

2. Generate server key and CSR

```
openssl genrsa -out server.key 2048
openssl req -new -key server.key -out server.csr -subj "/CN=localhost"
```

3. Create a config file for Subject Alternative Name

```
cat > server.ext << EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment,
    dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = localhost
IP.1 = 127.0.0.1
EOF
```

4. Sign the server certificate with our CA

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial
    -out server.crt -days 3650 -sha256 -extfile server.ext
```

5. Generate client key and certificate

```
openssl genrsa -out client.key 2048
openssl req -new -key client.key -out client.csr -subj "/CN=Client
    Certificate"
```

```
openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key -CAcreateserial
    -out client.crt -days 3650 -sha256
```

6. Create PKCS#12 file (Optional, needed for browser import)

```
openssl pkcs12 -export -out client.p12 -inkey client.key -in client.crt -
    certfile ca.crt -passout pass:<your_password>
```

Edit `pgmoneta.conf` to add the following keys under pgmoneta section:

```
[pgmoneta]
.
.
.
metrics_cert_file=<path_to_server_cert_file>
metrics_key_file=<path_to_server_key_file>
metrics_ca_file=<path_to_ca_file>
```

You can now access the metrics at `https://localhost:5001` using curl as follows:

```
curl -v -L "https://localhost:5001" --cacert <path_to_ca_file> --cert <
    path_to_client_cert_file> --key <path_to_client_key_file>
```

(Optional) If you want to access the page through the browser: - First install the certificates on your system - For Fedora: “‘ # Create directory if it doesn’t exist sudo mkdir -p /etc/pki/ca-trust/source/anchors/

```
# Copy CA cert to the trust store
sudo cp ca.crt /etc/pki/ca-trust/source/anchors/

# Update the CA trust store
sudo update-ca-trust extract
‘‘‘

- For Ubuntu:
‘‘‘
# Copy the CA certificate to the system certificate store
sudo cp ca.crt /usr/local/share/ca-certificates/

# Update the CA certificate store
sudo update-ca-certificates
‘‘‘

- For MacOS:
    - Open Keychain Access and import the certificate file
    - Set the certificate to "Always Trust"
```

- For browsers like Firefox
  - Go to Menu → Preferences → Privacy & Security

  - Scroll down to “Certificates” section and click “View Certificates”
  - Go to “Authorities” tab and click “Import”
  - Select your `ca.crt` file
  - Check “Trust this CA to identify websites” and click OK
  - Go to “Your Certificates” tab
  - Click “Import” and select the `client.p12` file
  - Enter the password you set when creating the PKCS#12 file

- For browsers like Chrome/Chromium

  - For client certificates, go to Settings → Privacy and security → Security → Manage certificates
  - Click on “Import” and select your `client.p12` file
  - Enter the password you set when creating it

You can now access metrics at `https://localhost:5001`

6.4 Backup and restore

This tutorial will show you how to do a backup and a restore using **pgmoneta**.

6.4.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.4.2 Backup

```
pgmoneta-cli -c pgmoneta.conf backup primary
```

will take a backup of the `[primary]` host.

(`pgmoneta` user)

6.4.3 List backups

```
pgmoneta-cli -c pgmoneta.conf list-backup primary
```

(`pgmoneta` user)

6.4.4 Restore

```
pgmoneta-cli -c pgmoneta.conf restore primary newest current /tmp/
```

will take the latest backup and all Write-Ahead Log (WAL) segments and restore it into the /tmp/primary-<timestamp> directory for an up-to-date copy.

The 2nd to last parameter allows

- current means copy the Write-Ahead Log (WAL), and restore to first stable checkpoint
- name=X means copy the Write-Ahead Log (WAL), and restore to the label specified
- xid=X means copy the Write-Ahead Log (WAL), and restore to the XID specified
- time=X means copy the Write-Ahead Log (WAL), and restore to the timestamp specified
- lsn=X means copy the Write-Ahead Log (WAL), and restore to the Log Sequence Number (LSN) specified
- inclusive=X means that the restore is inclusive of the specified information
- timeline=X means that the restore is done to the specified information timeline
- action=X means which action should be executed after the restore (pause, shutdown)
- primary means that the cluster is setup as a primary
- replica means that the cluster is setup as a replica

More information

(pgmoneta user)

6.5 Verify

This tutorial will show you how to verify a backup using **pgmoneta**.

6.5.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.5.2 Verify

```
pgmoneta-cli -c pgmoneta.conf verify primary oldest /tmp
```

will verify the oldest backup of the [primary] host.

(pgmoneta user)

6.6 Archive

This tutorial will show you how to do an archive using **pgmoneta**.

6.6.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.6.2 Creating an archive

```
pgmoneta-cli -c pgmoneta.conf archive primary newest current /tmp/
```

will take the latest backup and all Write-Ahead Log (WAL) segments and create an archive named `/tmp/primary-<timestamp>.tar.zstd`. This archive will contain an up-to-date copy.

(`pgmoneta` user)

6.7 Delete a backup

This tutorial will show you how to delete a backup from **pgmoneta**.

6.7.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.7.2 Delete the oldest backup

```
pgmoneta-cli -c pgmoneta.conf delete primary oldest
```

will delete the oldest backup on `[primary]`.

Note that if the backup has an incremental backup child that depends on it, its data will be rolled up to its child before getting deleted.

(`pgmoneta` user)

6.8 Encryption and Decryption

This tutorial will show you how to use encryption and decryption features in **pgmoneta**.

6.8.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.8.2 Enable Encryption and Decryption in pgmoneta workflow

By default, the encryption is disabled. To enable this feature, modify `pgmoneta.conf`:

```
encryption = aes-256-cbc
```

(`pgmoneta` user)

Many encryption modes are supported, see Configuration for details.

6.8.3 Encryption and Decryption Commands

pgmoneta use the same key created by `pgmoneta-admin master-key` to encrypt and decrypt files.

Encrypt a file with `pgmoneta-cli encrypt`, the file will be encrypted in place and remove unencrypted file on success.

```
pgmoneta-cli -c pgmoneta.conf encrypt '<path-to-your-file>/file.tar.zstd'
```

Decrypt a file with `pgmoneta-cli decrypt`, the file will be decrypted in place and remove encrypted file on success.

```
pgmoneta-cli -c pgmoneta.conf decrypt '<path-to-your-file>/file.tar.zstd.
   aes'
```

`pgmoneta-cli encrypt` and `pgmoneta-cli decrypt` are built to deal with files created by `pgmoneta-cli archive`. It can be used on other files though.

6.9 Retention Policy

This tutorial will show you how to configure retention to retain backups.

6.9.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.9.2 Retention Setup

In `pgmoneta.conf`, you can use `retention = 7, 4, 12, 5` to configure **pgmoneta** to retain backups within the nearest 7 days, 4 weeks, 12 months and 5 years. Specifically, **pgmoneta** will retain all the backups within the nearest 7 days, the latest backup on each Monday within the nearest 4 weeks, the latest backup on the first day of each month in the last 12 months and the latest backup on the first day of each year in the last 5 years. If you input more than 4 values, **pgmoneta** will only read the first 4.

Note that if a backup has an incremental backup child that depends on it, its data will be rolled up to its child before getting deleted.

There are a lot of ways to leave a parameter unspecified. For trailing parameters, you can simply omit them. And for parameters in between, you can use placeholders. Currently, placeholders we allow are: `-`, `X`, `x`, `0` or whitespaces (spaces or tabs).

Please note that you should always configure `days` to retain the nearest backups. If you don't configure retention, by default **pgmoneta** keeps backups within the nearest 7 days and other parameters (weeks, months, years) are unspecified. Additionally, if you are using prometheus, unspecified values will be shown as `0`.

6.9.3 Retention Validation Rule

Current validation rule is:

1. Retention days >= 1
2. If retention months is specified, then 1 <= weeks <= 4, otherwise weeks >= 1
3. If retention years is specified, then 1 <= months <= 12, otherwise months >= 1
4. Retention years >= 1 Please note that the rule above only checks specified parameters, except for days, which should always be specified

6.10 Retention check

The retention check runs every 5 minutes, and will delete one backup per run.

You can change this to every 30 minutes by

```
retention_interval = 1800
```

under the [pgmoneta] configuration.

6.11 Grafana Dashboard

This tutorial will show you how to encapsulate Prometheus standard API and use them to monitor state of **pgmoneta** with the help of Grafana dashboard.

6.11.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

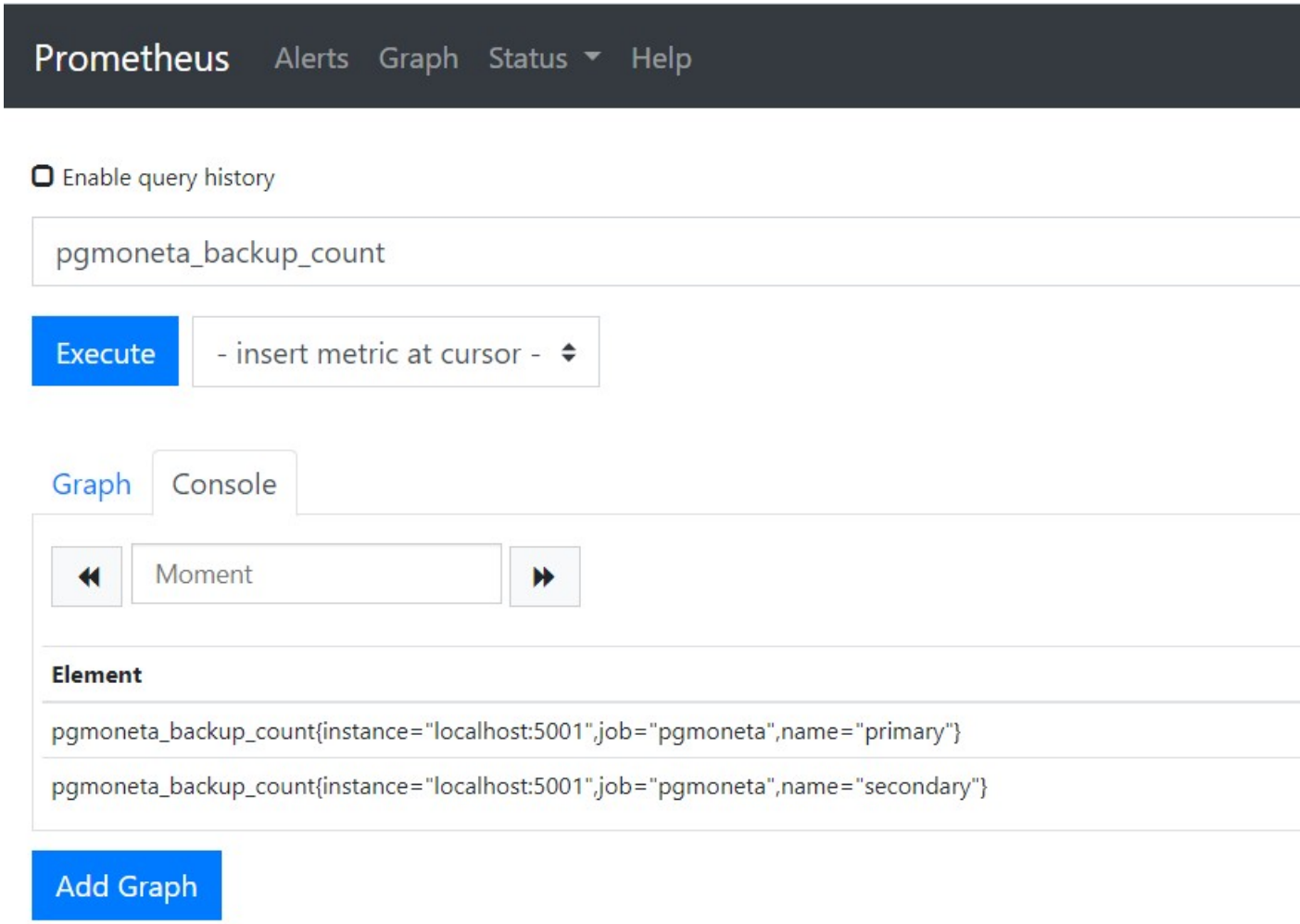
6.11.2 Prometheus Configuration

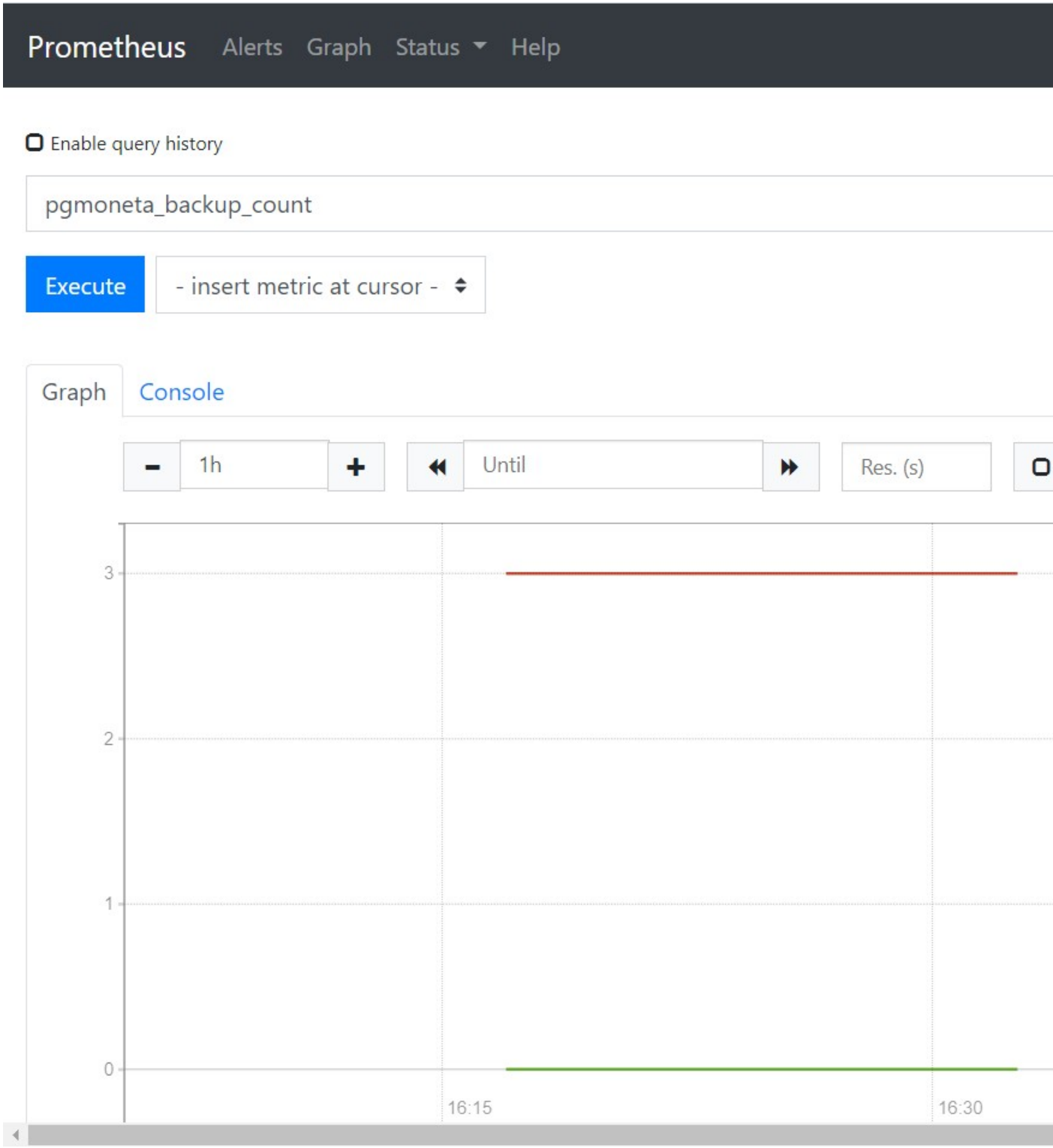
First of all, you should install Prometheus in your **pgmoneta** server.

After you successfully installed Prometheus, you should replace `prometheus.yml` with the content below to configure how to query your **pgmoneta** metrics.

```
scrape_configs:
  - job_name: 'pgmoneta'
    metrics_path: '/metrics'
    static_configs:
      - targets: ['localhost:5001']
```

Then the Prometheus service will query your **pgmoneta** metrics every 15 seconds and package them as time-series data. You can query your **pgmoneta** metrics and watch their changes as time passed in Prometheus web page (default port is `9090`).

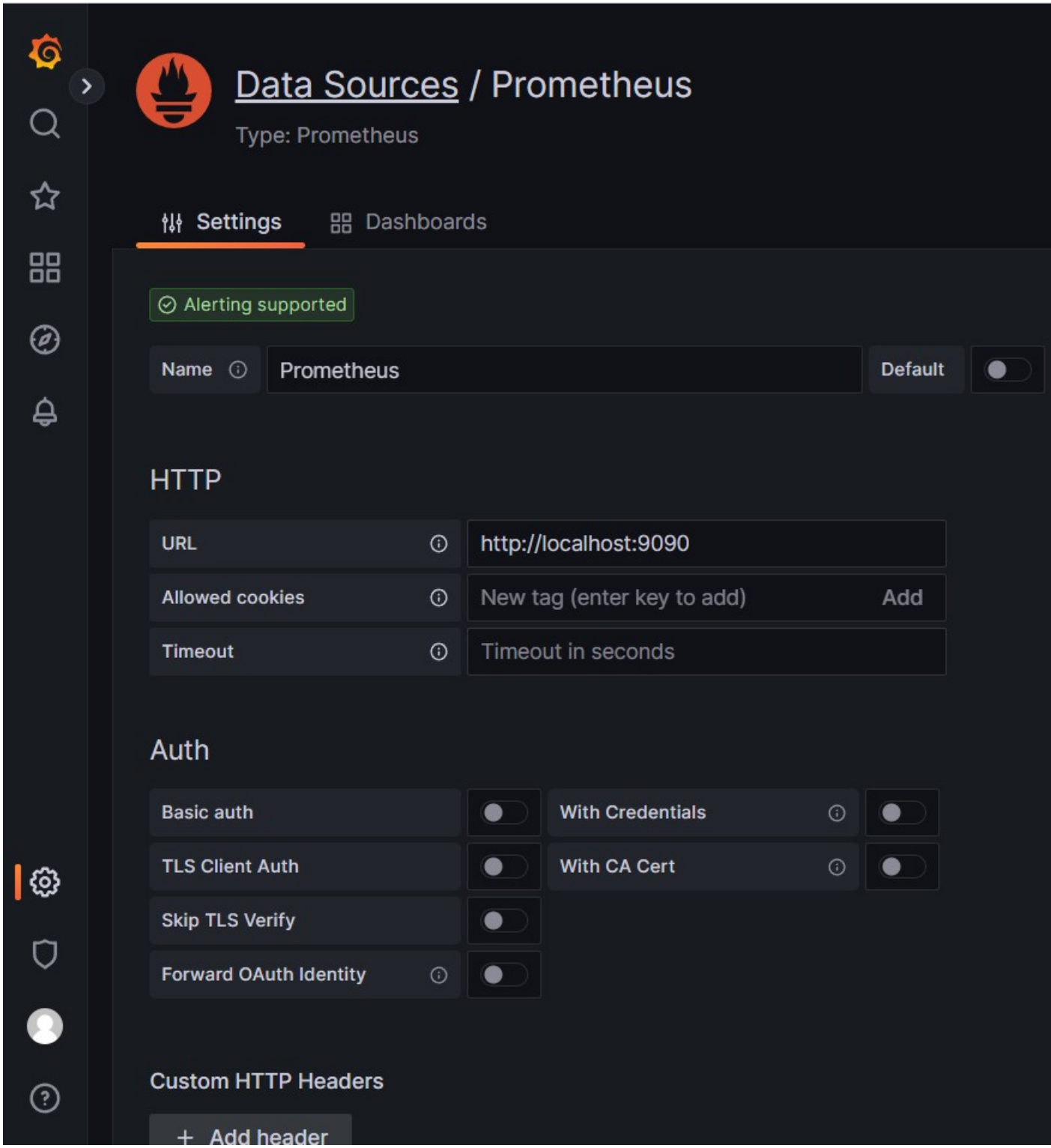

Prometheus
Alerts
Graph
Status
Help
Enable query history
pgmoneta_backup_count
Execute
- insert metric at cursor -
Graph
Console
Moment
Element
pgmoneta_backup_count{instance="localhost:5001",job="pgmoneta",name="primary"}
pgmoneta_backup_count{instance="localhost:5001",job="pgmoneta",name="secondary"}
Add Graph


Prometheus
Alerts
Graph
Status
Help
Enable query history
pgmoneta_backup_count
Execute
- insert metric at cursor -
Graph
Console
1h
Until
Res. (s)
3
2
1
0
16:15
16:30

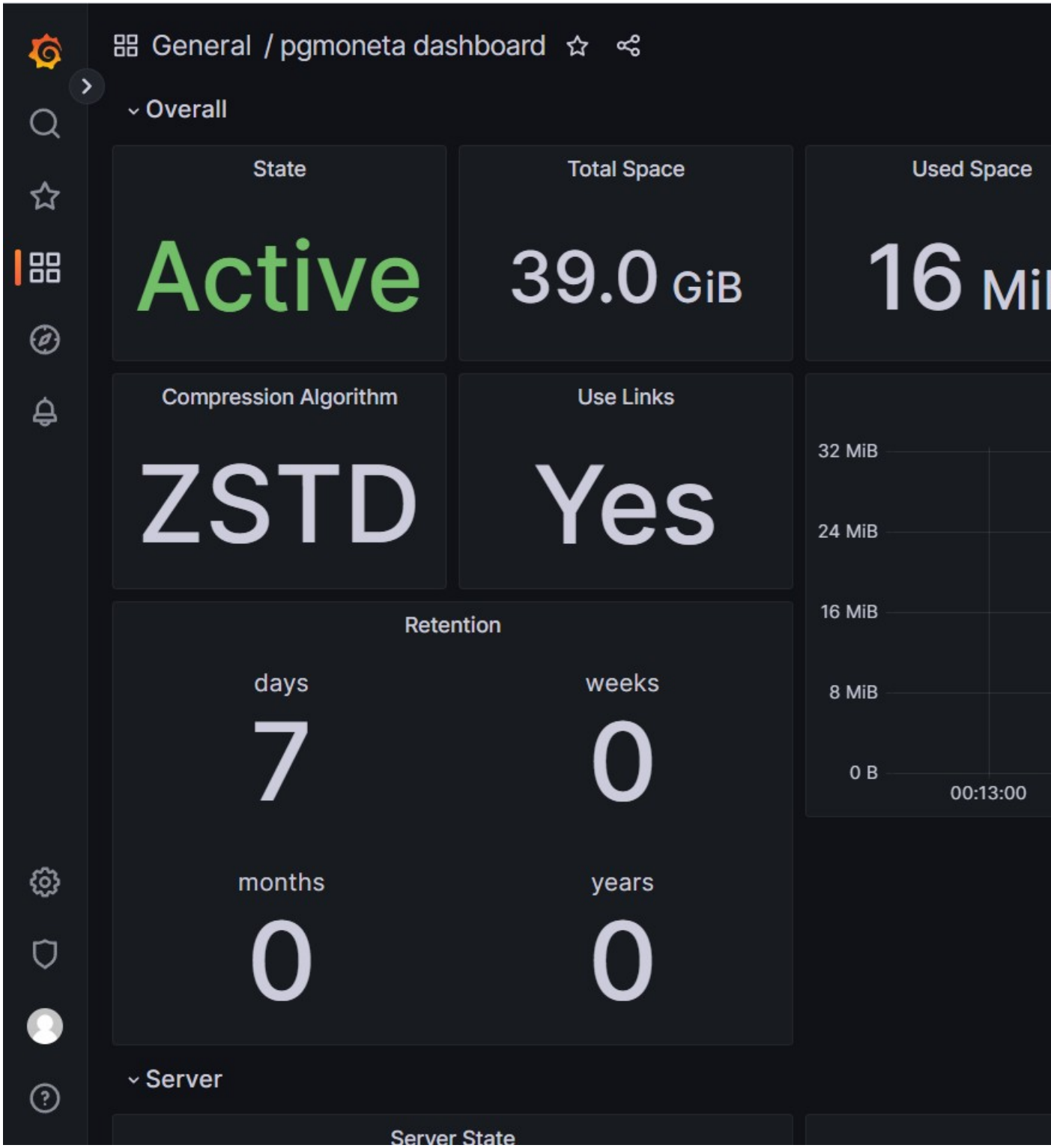
6.11.3 Grafana Dashboard Import

Although Prometheus provides capacity of querying and monitoring metrics, we can not customize graphs for each metric and provide a unified view.

As a result, we use Grafana to help us manage all graphs together. First of all, we should install Grafana in the computer you need to monitor **pgmoneta** metrics. You can browse Grafana web page with default port `3000`, default user `admin` and default password `admin`. Then you can create Prometheus data source of **pgmoneta**.



Finally you can create dashboard by importing `contrib/grafana/dashboard.json` and monitor metrics about **pgmoneta**.


General / pgmoneta dashboard
Overall
State
Active
Total Space
39.0 GiB
Used Space
16 Mi
Compression Algorithm
ZSTD
Use Links
Yes
32 MiB
24 MiB
16 MiB
8 MiB
0 B
00:13:00
Retention
days
7
weeks
0
months
0
years
0
Server
Server State

6.12 WAL shipping

This tutorial will show you how to configure WAL shipping, so that if the backup server crashes, you will be able to use an older archive to do Point-in-Time recovery with WAL segments shipped to another server/local directory.

Note that this feature is still at its early stages, that it currently only ships WAL segments to another LOCAL directory. You do not need to make sure the directory is unique for each server, the WAL copy will be saved under the subdirectory `server_name/wal/`

6.12.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.12.2 Configuration

In order to use WAL shipping, simply add

```
wal_shipping = your/local/wal/shipping/directory
```

to the corresponding server section of `pgmoneta.conf`, **pgmoneta** will create the directory if it doesn't exist, and ship a copy of WAL segments under the subdirectory `your/local/wal/shipping/directory/server_name/wal`.

6.12.3 Prometheus

You can monitor the disk usage regarding WAL shipping using prometheus. Here are some metrics.

```
pgmoneta_wal_shipping(name)  -- size of the WAL shipping for the server (
    wal/shipping/directory/server_name/wal)
pgmoneta_wal_shipping_used_space -- size of everything under the WAL
    shipping directory; this could include archives (wal/shipping/directory
    /server_name/)
pgmoneta_wal_shipping_free_space -- free size of the WAL shipping
    directory for the server (wal/shipping/directory/server_name/)
pgmoneta_wal_shipping_total_space -- total size of the WAL shipping
    directory for the server (wal/shipping/directory/server_name/)
```

6.13 Use of Transport Level Security (TLS)

This tutorial is about using Transport Level Security (TLS) in PostgreSQL, and how it affects **pgmoneta**.

Note, that this tutorial is an example on how to setup a PostgreSQL TLS environment for development use only !

6.13.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+, OpenSSL and **pgmoneta**.

See Install pgmoneta for more detail.

6.13.2 PostgreSQL

Generate the server key

```
openssl genrsa -aes256 8192 > server.key
```

Remove the passphase

```
openssl rsa -in server.key -out server.key
```

Set the server key permission

```
chmod 400 server.key
```

Generate the server certificate

```
openssl req -new -key server.key -days 3650 -out server.crt -x509
```

Use the server certificate as the root certificate (self-signed)

```
cp server.crt root.crt
```

In `postgresql.conf` change the following settings

```
listen_addresses = '*'
ssl = on
ssl_ca_file = '/path/to/root.crt'
ssl_cert_file = '/path/to/server.crt'
ssl_key_file = '/path/to/server.key'
ssl_prefer_server_ciphers = on
```

In `pg_hba.conf` change

```
host        all             all             0.0.0.0/0           scram-sha-256
```

to

```
hostssl     all             all             0.0.0.0/0           scram-sha-256
```

In this scenario there are no changes to the `pgmoneta.conf` configuration file.

6.13.3 Using client certificate

Create the client key

```
openssl ecparam -name prime256v1 -genkey -noout -out client.key
```

Create the client request - remember that the `CN` has to have the name of the replication user

```
openssl req -new -sha256 -key client.key -out client.csr -subj "/CN=repl"
```

Generate the client certificate

```
openssl x509 -req -in client.csr -CA root.crt -CAkey server.key -
   CAcreateserial -out client.crt -days 3650 -sha256
```

You can test your setup by copying the files into the default PostgreSQL client directory, like

```
mkdir ~/.postgresql
cp client.crt ~/.postgresql/postgresql.crt
cp client.key ~/.postgresql/postgresql.key
cp root.crt ~/.postgresql/ca.crt
chmod 0600 ~/.postgresql/postgresql.crt ~/.postgresql/postgresql.key ~/.
   postgresql/ca.crt
```

and then test with the `psql` command.

In `pg_hba.conf` change

```
hostssl     all             all             0.0.0.0/0           scram-sha-256
```

to

```
hostssl     all             all             0.0.0.0/0           scram-sha-256
   clientcert=verify-ca
```

In `pgmoneta.conf` add the paths to the server in question, like

```
[pgmoneta]
...
```

```
[primary]
host=...
port=...
user=repl
tls_cert_file=/path/to/home/.postgresql/postgresql.crt
tls_key_file=/path/to/home/.postgresql/postgresql.key
tls_ca_file=/path/to/home/.postgresql/ca.crt
```

6.13.4 More information

- Secure TCP/IP Connections with SSL
- The pg_hba.conf File

6.14 Hot standby

This tutorial will show you how to configure a hot standby instance.

6.14.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.14.2 Configuration

In order to use hot standby, simply add

```
hot_standby = /your/local/hot/standby/directory
```

to the corresponding server section of `pgmoneta.conf`. **pgmoneta** will create the directory if it doesn't exist, and keep the latest backup in the defined directory.

You can use

```
hot_standby_overrides = /your/local/hot/standby/overrides/
```

to override files in the `hot_standby` directory.

6.14.3 Tablespaces

By default tablespaces will be mapped to a similar path than the original one, for example `/tmp/mytblspc` becomes `/tmp/mytblspchs`.

However, you can use the directory name to map it to another directory, like

```
hot_standby_tablespaces = /tmp/mytblspc->/tmp/mybcktblspc
```

You can also use the `OID` for the key part, like

```
hot_standby_tablespaces = 16392->/tmp/mybcktblspc
```

Multiple tablespaces can be specified using a `,` between them.

6.15 Annotate a backup with comments

This tutorial is about adding commnents to a backup.

6.15.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+, OpenSSL and **pgmoneta**.

See Install pgmoneta for more detail.

6.15.2 Add a comment

You can add a comment by

```
pgmoneta-cli -c pgmoneta.conf annotate primary newest add mykey mycomment
```

6.15.3 Update a comment

You can update a comment by

```
pgmoneta-cli -c pgmoneta.conf annotate primary newest update mykey
   mynewcomment
```

6.15.4 Remove a comment

You can remove a comment by

```
pgmoneta-cli -c pgmoneta.conf annotate primary newest remove mykey
```

6.15.5 View comments

You can view the comments by

```
pgmoneta-cli -c pgmoneta.conf info primary newest
```

6.16 Feature extra

This tutorial will show you how to configure the `extra` feature to retrieve extra files and directories (recursively) from the PostgreSQL server instance.

6.16.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 13+, **pgmoneta_ext**, and **pgmoneta**.

See Install pgmoneta_ext and Install pgmoneta for more detail.

6.16.2 Configuration

In order to use `exta`, simply add

```
extra = /tmp/myfile, /tmp/mydir, /tmp/mydirs
```

to the corresponding server section of `pgmoneta.conf`. **pgmoneta** will retrieve `extra` files and directories (recursively) from the PostgreSQL server instance when performing a backup.

You can use

```
pgmoneta-cli -c pgmoneta.conf backup primary
```

to trigger the client to get `extra` files from the server side.

All the files will be stored in the `/extra` directory under the backup directory. For our configuration, they will be stored in the `/tmp/primary/<timestamp>/extra` directory.

6.16.3 Info

You can use

```
pgmoneta-cli -c pgmoneta.conf info primary newest
```

to show all copied `extra` files, something like this (only showing successfully copied files):

```
Extra                : myfile, ...
```

6.17 Incremental backup and restore

This tutorial will show you how to do an incremental backup and a restore using **pgmoneta**.

6.17.1 Preface

This tutorial assumes that you have an installation of PostgreSQL 17+ and **pgmoneta**.

See Install pgmoneta for more detail.

6.17.2 Full backup

```
pgmoneta-cli -c pgmoneta.conf backup primary
```

will take a full backup of the `[primary]` host.

(`pgmoneta` user)

6.17.3 List backups

```
pgmoneta-cli -c pgmoneta.conf list-backup primary
```

(`pgmoneta` user)

6.17.4 Incremental backup

```
pgmoneta-cli -c pgmoneta.conf backup primary newest
```

will take an incremental backup of the [primary] host.

Note that currently branching is not allowed for incremental backup – a backup can have at most 1 incremental backup child.

(pgmoneta user)

6.17.5 List backups

```
pgmoneta-cli -c pgmoneta.conf list-backup primary
```

(pgmoneta user)

6.17.6 Restore

```
pgmoneta-cli -c pgmoneta.conf restore primary newest current /tmp/
```

will take the latest backup and all Write-Ahead Log (WAL) segments and restore it into the /tmp/primary-<timestamp> directory for an up-to-date copy.

The 2nd to last parameter allows

- current means copy the Write-Ahead Log (WAL), and restore to first stable checkpoint
- name=X means copy the Write-Ahead Log (WAL), and restore to the label specified
- xid=X means copy the Write-Ahead Log (WAL), and restore to the XID specified
- time=X means copy the Write-Ahead Log (WAL), and restore to the timestamp specified
- lsn=X means copy the Write-Ahead Log (WAL), and restore to the Log Sequence Number (LSN) specified
- inclusive=X means that the restore is inclusive of the specified information
- timeline=X means that the restore is done to the specified information timeline
- action=X means which action should be executed after the restore (pause, shutdown)
- primary means that the cluster is setup as a primary
- replica means that the cluster is setup as a replica

More information

(pgmoneta user)

7 Running pgmoneta with Docker

You can run pgmoneta using Docker instead of compiling it manually.

7.1 Prerequisites

- **Docker** or **Podman** must be installed on the server where PostgreSQL is running.
- Ensure PostgreSQL is configured to allow external connections.

7.2 Step 1: Enable External PostgreSQL Access

Modify the local PostgreSQL server's `postgresql.conf` file to allow connections from outside:

```
listen_addresses = '*'
```

Update `pg_hba.conf` to allow remote connections:

```
host    all    all    0.0.0.0/0    scram-sha-256
```

Then, restart PostgreSQL for the changes to take effect:

```
sudo systemctl restart postgresql
```

7.3 Step 2: Clone the Repository

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
```

7.4 Step 3: Build the Docker Image‘

There are two Dockerfiles available: 1. **Alpine-based image**
Using Docker `sh docker build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.alpine .` **Using Podman** `sh podman build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.alpine .`

2. **Rocky Linux 9-based image**
   Using Docker

```
docker build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.rocky9
   .
```

Using Podman

```
podman build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.rocky9
   .
```

7.5 Step 4: Run pgmoneta as a Docker Container

Once the image is built, run the container using: - **Using Docker** `sh docker run -d --name pgmoneta --network host pgmoneta:latest` - **Using Podman** `sh podman run -d --name pgmoneta --network host pgmoneta:latest`

7.6 Step 5: Verify the Container

Check if the container is running:

- **Using Docker** `sh docker ps | grep pgmoneta`
- **Using Podman** `sh podman ps | grep pgmoneta`

Check logs for any errors: - **Using Docker** `sh docker logs pgmoneta` - **Using Podman** `sh podman logs pgmoneta`

You can also inspect the exposed server at:

```
http://localhost:5001/
```

8 Test

8.1 Local Environment

To ensure the test suite works well, please make sure you have installed PostgreSQL 17.x version installed

For RPM based distributions such as Fedora and RHEL you can add the PostgreSQL YUM repository and do the install via

```
dnf -qy module disable postgresql
dnf install -y postgresql17 postgresql17-server pgmoneta
```

also make sure that the `initdb`, `pg_ctl` and `psql` are in PATH variable.

8.1.1 Add Path variable

Add the `initdb`, `pg_ctl` and `psql` binaries into the environment path.

```
export PATH=$PATH:$(dirname $(which initdb))
export PATH=$PATH:$(dirname $(which psql))
```

Note: `initdb` and `pg_ctl` belongs to same binary directory

8.1.2 Install check library

Before you test, you need to install the `check` library. If there is no package for `check`, the `CMakeLists.txt` will not compile the test suite. Only after you have installed `check` will it compile the test suite.

```
dnf install -y check check-devel check-static
```

8.1.3 Build the project

Make sure to execute the test script inside the project build. Run the following commands if project is not already built.

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
mkdir build
cd build
cmake -DCMAKE_C_COMPILER=clang -DCMAKE_BUILD_TYPE=Debug ..
make
```

8.1.4 Run test suites

To run the testsuites get inside your build and just execute -

```
./testsuite.sh
```

The script creates the PostgreSQL and pgmoneta environment inside the build itself for example - - the PostgreSQL related files like the data directory and PostgreSQL configuration will be stored in `pgmoneta-postgres` - the pgmoneta related files like pgmoneta configuration and users file will be stored in `pgmoneta-testsiute`

It will be the responsibility of the script to clean up the setup environment.

Note: You can however view the PostgreSQL and pgmoneta server logs in a separate `log` directory inside build.

In case you see those setup directories like `pgmoneta-postgres` and `pgmoneta-testsiute` in build after successfully executing the script, you should probably run

```
./testsuite clean
```

before running the script again to avoid any inconsistency or errors. The clean subcommand will however clean the logs as well.

8.1.5 Add testcases

To add an additional testcase go to testcases directory inside the `pgmoneta` project.

Create a `.c` file that contains the test suite and its corresponding `.h` file (see pgmoneta_test_1.c or pgmoneta_test_2.c for reference). Add the above created suite to the test runner in runner.c

Also remember to link the new test suite in CMakeLists file inside test directory

```
30:  set(SOURCES
31:    testcases/common.c
32:    testcases/pgmoneta_test_1.c
33:    testcases/pgmoneta_test_2.c
34:    testcases/runner.c
35:  )
```

9 Command line interface

The **pgmoneta-cli** command line interface controls your interaction with **pgmoneta**.

It is important that you only use the pgmoneta-cli command line interface to operate on your backup directory

Using other commands on the backup directory could cause problems.

```
pgmoneta-cli 0.16.0
  Command line utility for pgmoneta

Usage:
  pgmoneta-cli [ -c CONFIG_FILE ] [ COMMAND ]

Options:
  -c, --config CONFIG_FILE                          Set the path to the
     pgmoneta.conf file
  -h, --host HOST                                   Set the host name
  -p, --port PORT                                   Set the port number
  -U, --user USERNAME                               Set the user name
  -P, --password PASSWORD                           Set the password
  -L, --logfile FILE                                Set the log file
  -v, --verbose                                     Output text string of
     result
  -V, --version                                     Display version
     information
  -F, --format text|json|raw                        Set the output format
  -C, --compress none|gz|zstd|lz4|bz2               Compress the wire
     protocol
  -E, --encrypt none|aes|aes256|aes192|aes128       Encrypt the wire
     protocol
  -?, --help                                        Display help

Commands:
  annotate                 Annotate a backup with comments
  archive                  Archive a backup from a server
  backup                   Backup a server
  clear <what>             Clear data, with:
                           - 'prometheus' to reset the Prometheus
                              statistics
  compress                 Compress a file using configured method
  conf <action>            Manage the configuration, with one of
     subcommands:
                           - 'get' to obtain information about a runtime
                              configuration value
                             conf get <parameter_name>
                           - 'ls' to print the configurations used
                           - 'reload' to reload the configuration
                           - 'set' to modify a configuration value;
```

```
                              conf set <parameter_name> <parameter_value>;
  decompress                Decompress a file using configured method
  decrypt                   Decrypt a file using master-key
  delete                    Delete a backup from a server
  encrypt                   Encrypt a file using master-key
  expunge                   Expunge a backup from a server
  info                      Information about a backup
  list-backup               List the backups for a server
  ping                      Check if pgmoneta is alive
  restore                   Restore a backup from a server
  retain                    Retain a backup from a server
  shutdown                  Shutdown pgmoneta
  status [details]          Status of pgmoneta, with optional details
  verify                    Verify a backup from a server

pgmoneta: https://pgmoneta.github.io/
Report bugs: https://github.com/pgmoneta/pgmoneta/issues
```

9.1 backup

Backup a server

The command for a full backup is

```
pgmoneta-cli backup <server>
```

Example

```
pgmoneta-cli backup primary
```

The command for an incremental backup is

```
pgmoneta-cli backup <server> <identifier>
```

where the `identifier` is the identifier for a backup.

Example

```
pgmoneta-cli backup primary 20250101120000
```

9.2 list-backup

List the backups for a server

Command

```
pgmoneta-cli list-backup <server> [--sort asc|desc]
```

The `--sort` option allows sorting backups by timestamp: - `asc` for ascending order (oldest first) - `desc` for descending order (newest first)

Example

```
pgmoneta-cli list-backup primary
```

Example with sorting

```
pgmoneta-cli list-backup primary --sort desc
```

9.3 restore

Restore a backup from a server

Command

```
pgmoneta-cli restore <server> [<timestamp>|oldest|newest] [[current|name=X
   |xid=X|lsn=X|time=X|inclusive=X|timeline=X|action=X|primary|replica],*]
    <directory>
```

where

- `current` means copy the Write-Ahead Log (WAL), and restore to first stable checkpoint
- `name=X` means copy the Write-Ahead Log (WAL), and restore to the label specified
- `xid=X` means copy the Write-Ahead Log (WAL), and restore to the XID specified
- `time=X` means copy the Write-Ahead Log (WAL), and restore to the timestamp specified
- `lsn=X` means copy the Write-Ahead Log (WAL), and restore to the Log Sequence Number (LSN) specified
- `inclusive=X` means that the restore is inclusive of the specified information
- `timeline=X` means that the restore is done to the specified information timeline
- `action=X` means which action should be executed after the restore (pause, shutdown)

More information

Example

```
pgmoneta-cli restore primary newest name=MyLabel,primary /tmp
```

9.4 verify

Verify a backup from a server

Command

```
pgmoneta-cli verify <server> <directory> [failed|all]
```

Example

```
pgmoneta-cli verify primary oldest /tmp
```

9.5 archive

Archive a backup from a server

Command

```
pgmoneta-cli archive <server> [<timestamp>|oldest|newest] [[current|name=X
   |xid=X|lsn=X|time=X|inclusive=X|timeline=X|action=X|primary|replica],*]
    <directory>
```

Example

```
pgmoneta-cli archive primary newest current /tmp
```

9.6 delete

Delete a backup from a server

Command

```
pgmoneta-cli delete <server> [<timestamp>|oldest|newest]
```

Example

```
pgmoneta-cli delete primary oldest
```

9.7 retain

Retain a backup from a server. The backup will not be deleted by the retention policy

Command

```
pgmoneta-cli retain <server> [<timestamp>|oldest|newest]
```

Example

```
pgmoneta-cli retain primary oldest
```

9.8 expunge

Expunge a backup from a server. The backup will be deleted by the retention policy

Command

```
pgmoneta-cli expunge <server> [<timestamp>|oldest|newest]
```

Example

```
pgmoneta-cli expunge primary oldest
```

9.9 encrypt

Encrypt the file in place, remove unencrypted file after successful encryption.

Command

```
pgmoneta-cli encrypt <file>
```

9.10 decrypt

Decrypt the file in place, remove encrypted file after successful decryption.

Command

```
pgmoneta-cli decrypt <file>
```

9.11 compress

Compress the file in place, remove uncompressed file after successful compression.

Command

```
pgmoneta-cli compress <file>
```

9.12 decompress

Decompress the file in place, remove compressed file after successful decompression.

Command

```
pgmoneta-cli decompress <file>
```

9.13 info

Information about a backup.

Command

```
pgmoneta-cli info <server> <timestamp|oldest|newest>
```

9.14 ping

Verify if **pgmoneta** is alive

Command

```
pgmoneta-cli ping
```

Example

```
pgmoneta-cli ping
```

9.15 shutdown

Shutdown **pgmoneta**

Command

```
pgmoneta-cli shutdown
```

Example

```
pgmoneta-cli shutdown
```

9.16 status

Status of **pgmoneta**, with a `details` option

Command

```
pgmoneta-cli status [details]
```

Example

```
pgmoneta-cli status details
```

9.17 conf

Manage the configuration

Command

```
pgmoneta-cli conf [reload]
```

Subcommand

- `reload`: Reload configuration

Example

```
pgmoneta-cli conf reload
```

9.18 clear

Clear data/statistics

Command

```
pgmoneta-cli clear [prometheus]
```

Subcommand

- `prometheus`: Reset the Prometheus statistics

Example

```
pgmoneta-cli clear prometheus
```

9.19 Shell completions

There is a minimal shell completion support for `pgmoneta-cli`.

Please refer to the Install pgmoneta tutorial for detailed information about how to enable and use shell completions.

10 Prometheus metrics

pgmoneta has the following Prometheus metrics.

10.1 pgmoneta_state

The state of pgmoneta

1 = Running

10.2 pgmoneta_version

The version of pgmoneta

10.3 pgmoneta_logging_info

The number of INFO statements

10.4 pgmoneta_logging_warn

The number of WARN statements

10.5 pgmoneta_logging_error

The number of ERROR statements

10.6 pgmoneta_logging_fatal

The number of FATAL statements

10.7 pgmoneta_retention_days

The retention of pgmoneta in days

10.8 pgmoneta_retention_weeks

The retention of pgmoneta in weeks

10.9 pgmoneta_retention_months

The retention of pgmoneta in months

10.10 pgmoneta_retention_years

The retention of pgmoneta in years

10.11 pgmoneta_retention_server

The retention of a server

Attribute	Description
name	The identifier for the server
parameter	days weeks months years

10.12 pgmoneta_extension

The version of pgmoneta extension

10.13 pgmoneta_compression

The compression used

0 = None

1 = GZip

2 = ZSTD

3 = LZ4

4 = BZIP2

10.14 pgmoneta_used_space

The disk space used for pgmoneta

10.15 pgmoneta_free_space

The free disk space for pgmoneta

10.16 pgmoneta_total_space

The total disk space for pgmoneta

10.17 pgmoneta_server_valid

Is the server in a valid state

10.18 pgmoneta_wal_streaming

The WAL streaming status of a server

10.19 pgmoneta_server_operation_count

The count of client operations of a server

10.20 pgmoneta_server_failed_operation_count

The count of failed client operations of a server

10.21 pgmoneta_server_last_operation_time

The time of the latest client operation of a server

10.22 pgmoneta_server_last_failed_operation_time

The time of the latest failed client operation of a server

10.23 pgmoneta_wal_shipping

The disk space used for WAL shipping for a server

10.24 pgmoneta_wal_shipping_used_space

The disk space used for everything under the WAL shipping directory of a server

10.25 pgmoneta_wal_shipping_free_space

The free disk space for the WAL shipping directory of a server

10.26 pgmoneta_wal_shipping_total_space

The total disk space for the WAL shipping directory of a server

10.27 pgmoneta_workspace

The disk space used for workspace for a server

10.28 pgmoneta_workspace_free_space

The free disk space for the workspace directory of a server

10.29 pgmoneta_workspace_total_space

The total disk space for the workspace directory of a server

10.30 pgmoneta_hot_standby

The disk space used for hot standby for a server

10.31 pgmoneta_hot_standby_free_space

The free disk space for the hot standby directory of a server

10.32 pgmoneta_hot_standby_total_space

The total disk space for the hot standby directory of a server

10.33 pgmoneta_server_timeline

The current timeline a server is on

Attribute	Description
name	The identifier for the server

10.34 pgmoneta_server_parent_tli

The parent timeline of a timeline on a server

Attribute	Description
name	The identifier for the server
tli	The current/previous timeline ID in the server history

10.35 pgmoneta_server_timeline_switchpos

The WAL switch position of a timeline on a server (showed in hex as a parameter)

Attribute	Description
name	The identifier for the server
tli	The current/previous timeline ID in the server history
walpos	The WAL switch position of this timeline

10.36 pgmoneta_server_workers

The number of workers for a server

Attribute	Description
name	The identifier for the server

10.37 pgmoneta_backup_oldest

The oldest backup for a server

Attribute	Description
name	The identifier for the server

10.38 pgmoneta_backup_newest

The newest backup for a server

Attribute	Description
name	The identifier for the server

10.39 pgmoneta_backup_count

The number of valid backups for a server

Attribute	Description
name	The identifier for the server

10.40 pgmoneta_backup

Is the backup valid for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.41 pgmoneta_backup_version

The version of PostgreSQL for a backup

Attribute	Description
name	The identifier for the server
label	The backup label
major	The backup PostgreSQL major version
minor	The backup PostgreSQL minor version

10.42 pgmoneta_backup_throughput

The throughput of the backup for a server (bytes/s)

name The identifier for the server label The backup label

10.43 pgmoneta_backup_elapsed_time

The backup in seconds for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.44 pgmoneta_backup_start_timeline

The starting timeline of a backup for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.45 pgmoneta_backup_end_timeline

The ending timeline of a backup for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.46 pgmoneta_backup_start_walpos

The starting WAL position of a backup for a server

Attribute	Description
name	The identifier for the server
label	The backup label
walpos	The backup starting WAL position

10.47 pgmoneta_backup_checkpoint_walpos

The checkpoint WAL pos of a backup for a server

Attribute	Description
name	The identifier for the server
label	The backup label
walpos	The backup checkpoint WAL position

10.48 pgmoneta_backup_end_walpos

The ending WAL pos of a backup for a server

Attribute	Description
name	The identifier for the server
label	The backup label
walpos	The backup ending WAL position

10.49 pgmoneta_restore_newest_size

The size of the newest restore for a server

Attribute	Description
name	The identifier for the server

10.50 pgmoneta_backup_newest_size

The size of the newest backup for a server

Attribute	Description
name	The identifier for the server

10.51 pgmoneta_restore_size

The size of a restore for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.52 pgmoneta_restore_size_increment

The increment size of a restore for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.53 pgmoneta_backup_size

The size of a backup for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.54 pgmoneta_backup_compression_ratio

The ratio of backup size to restore size for each backup

Attribute	Description
name	The identifier for the server
label	The backup label

10.55 pgmoneta_backup_retain

Retain a backup for a server

Attribute	Description
name	The identifier for the server
label	The backup label

10.56 pgmoneta_backup_total_size

The total size of the backups for a server

Attribute	Description
name	The identifier for the server

10.57 pgmoneta_wal_total_size

The total size of the WAL for a server

Attribute	Description
name	The identifier for the server

10.58 pgmoneta_total_size

The total size for a server

Attribute	Description
name	The identifier for the server

10.59 pgmoneta_active_backup

Is there an active backup for a server

Attribute	Description
name	The identifier for the server

10.60 pgmoneta_current_wal_file

The current streaming WAL filename of a server

Attribute	Description
name	The identifier for the server
file	The current WAL filename for this server

10.61 pgmoneta_current_wal_lsn

The current WAL log sequence number

Attribute	Description
name	The identifier for the server
lsn	The current WAL log sequence number

11 SSH

11.1 Prerequisites

First of all, you need to have a remote server where you can store your backups on.

Lets take an EC2 instance as an example, after launching an EC2 instance you need to add new user account with SSH access to the EC2 instance:

1. Connect to your Linux instance using SSH.
2. Use the adduser command to add a new user account to an EC2 instance (replace new_user with the new account name).

```
sudo adduser new_user --disabled-password
```

3. Change the security context to the new_user account so that folders and files you create have the correct permissions:

```
sudo su - new_user
```

4. Create a .ssh directory in the new_user home directory and use the chmod command to change the .ssh directory's permissions to 700:

```
mkdir .ssh && chmod 700 .ssh
```

5. Use the touch command to create the authorized_keys file in the .ssh directory and use the chmod command to change the .ssh/authorized_keys file permissions to 600:

```
touch .ssh/authorized_keys && chmod 600 .ssh/authorized_keys
```

6. Retrieve the public key for the key pair in your local computer:

```
cat ~/.ssh/id_rsa.pub
```

7. In the EC2 instance, run the cat command in append mode:

```
cat >> .ssh/authorized_keys
```

8. Paste the public key into the .ssh/authorized_keys file and then press Enter.
9. Press and hold Ctrl+d to exit cat and return to the command line session prompt.

To verify that the new user can use SSH to connect to the EC2 instance, run the following command from a command line prompt on your local computer:

```
ssh new_user@public_dns_name_of_EC2_instance
```

11.2 Modify the pgmoneta configuration

You need to create a directory on your remote server where backups can be stored in.

In addition, your local computer needs to have a storage space for 1 backup.

Change `pgmoneta.conf` to add

```
storage_engine = ssh
ssh_hostname =  your-public_dns_name_of_EC2_instance
ssh_username = new_user
ssh_base_dir = the-path-of-the-directory-where-backups-stored-in
```

under the `[pgmoneta]` section.

12 Azure

12.1 Prerequisites

First of all, you need to have an Azure account, an Azure storage account and a blob container.

A container organizes a set of blobs, similar to a directory in a file system. A storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs.

To create an Azure storage account with the Azure portal:

1. Sign in to the Azure portal.
2. From the left portal menu, select Storage accounts to display a list of your storage accounts. If the portal menu isn't visible, click the menu button to toggle it on.
3. On the Storage accounts page, select Create.
4. On the Basics tab, provide a resource group name and storage account name. You can go for the default settings of the other fields.
5. Choose Next: Advanced.
6. On the Advanced tab, you can configure additional options and modify default settings for your new storage account. You can go for the default settings.
7. Choose Next: Networking.
8. On the Networking tab, you can go for the default settings.
9. Choose Next: Data protection.
10. On the Data protection tab, you can for the default settings.
11. Choose Next: Encryption.
12. On the Encryption tab, you can for the default settings.
13. Choose Next: Tags.
14. Choose Next: Review to see all of the choices you made up to this point. When you are ready to proceed, choose Create.

To create a blob container with the Azure portal:

1. In the navigation pane for the storage account, scroll to the `Data storage` section and select Containers.
2. Within the Containers pane, select the + `Container` button to open the New container pane.

3. Within the New Container pane, provide a Name for your new container.
4. Select Create to create the container.

To get the Azure storage account shared key which is required for pgmoneta configuration:

1. In the navigation pane for the storage account, scroll to the `Security + networking` section and select Access Keys.
2. Under key1, find the Key value. Select the Copy button to copy the account key.

You can use either of the two keys to access Azure Storage, but in general it's a good practice to use the first key, and reserve the use of the second key for when you are rotating keys.

12.2 Modify the pgmoneta configuration

You need to have a storage space for 1 backup on your local computer.

Change `pgmoneta.conf` to add

```
storage_engine = azure
azure_storage_account = the-storage-account-name
azure_container = the-container-name
azure_shared_key = the-storage-account-shared-key
azure_base_dir = directory-where-backups-will-be-stored-in
```

under the `[pgmoneta]` section.

13 S3

13.1 Prerequisites

First of all, you need to have an AWS account, an IAM user and S3 bucket.

To create an IAM user:

1. Sign in to the AWS Management Console and open the IAM console.
2. In the navigation pane, choose Users and then choose Add users.
3. Type the user name for the new user.
4. Select the type of access to be both programmatic access and access to the AWS Management Console.
5. Choose Next: Permissions.
6. On the Set permissions page, select attach existing policies directly, search for AmazonS3FullAccess and choose it, then choose Next: Review, then choose Add permissions.
7. Choose Next: Tags.
8. Choose Next: Review to see all of the choices you made up to this point. When you are ready to proceed, choose Create user.
9. To view the users' access keys (access key IDs and secret access keys), choose Show next to each password and access key that you want to see. To save the access keys, choose Download .csv and then save the file to a safe location.

You are now ready to create a S3 bucket, To create a S3 bucket:

1. Sign in to the AWS Management Console using your IAM user credentials and open the Amazon S3 console.
2. Choose Create bucket.
3. In Bucket name, enter a name for your bucket.
4. In Region, choose the AWS Region where you want the bucket to reside.
5. Keep the default values as it is and Choose Create bucket.

13.2 Modify the pgmoneta configuration

You need to have a storage space for 1 backup on your local computer.

Change `pgmoneta.conf` to add

```
storage_engine = s3
s3_aws_region = the-aws-region
s3_access_key_id = your-access-key-id-from-the-downloaded-file
s3_secret_access_key = your-secret-access-key-from-the-downloaded-file
s3_bucket = your-s3-bucket-name
s3_base_dir = directory-where-backups-will-be-stored-in
```

under the `[pgmoneta]` section.

14 Running pgmoneta with Docker

You can run `pgmoneta` using Docker instead of compiling it manually.

14.1 Prerequisites

- **Docker** or **Podman** must be installed on the server where PostgreSQL is running.
- Ensure PostgreSQL is configured to allow external connections.

14.2 Step 1: Enable External PostgreSQL Access

Modify the local PostgreSQL server's `postgresql.conf` file to allow connections from outside:

```
listen_addresses = '*'
```

Update `pg_hba.conf` to allow remote connections:

```
host    all    all    0.0.0.0/0    scram-sha-256
```

Then, restart PostgreSQL for the changes to take effect:

```
sudo systemctl restart postgresql
```

14.3 Step 2: Clone the Repository

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
```

14.4 Step 3: Build the Docker Image

There are two Dockerfiles available: 1. **Alpine-based image**
Using Docker `sh docker build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.alpine .` **Using Podman** `sh podman build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.alpine .`

2. **Rocky Linux 9-based image**

```
docker build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.rocky9
   .
```

Using Podman

```
podman build -t pgmoneta:latest -f ./contrib/docker/Dockerfile.rocky9
   .
```

14.5 Step 4: Run pgmoneta as a Docker Container

Once the image is built, run the container using: - **Using Docker** `sh docker run -d --name pgmoneta --network host pgmoneta:latest` - **Using Podman** `sh podman run -d --name pgmoneta --network host pgmoneta:latest`

14.6 Step 5: Verify the Container

Check if the container is running:

- **Using Docker** `sh docker ps | grep pgmoneta`
- **Using Podman** `sh podman ps | grep pgmoneta`

Check logs for any errors: - **Using Docker** `sh docker logs pgmoneta` - **Using Podman** `sh podman logs pgmoneta`

You can also inspect the exposed server at:

```
http://localhost:5001/
```

15 Troubleshooting

15.1 Could not get version for server

If you get this `FATAL` during startup check your PostgreSQL logins

```
psql postgres
```

and

```
psql -U repl postgres
```

And, check the PostgreSQL logs for any error.

Setting `log_level` to `DEBUG5` in `pgmoneta.conf` could provide more information about the error.

16 Acknowledgement

16.1 Authors

pgmoneta was created by the following authors:

```
Jesper Pedersen <jesperpedersen.db@gmail.com>
David Fetter <david@fetter.org>
Will Leinweber <will@bitfission.com>
Luca Ferrari <fluca1978@gmail.com>
Nikita Bugrovsky <nbugrovs@redhat.com>
Mariam Fahmy <mariamfahmy66@gmail.com>
Jichen Xu <kyokitisin@gmail.com>
Saurav Pal <resyfer.dev@gmail.com>
Bokket <bokkett@gmail.com>
Haoran Zhang <andrewzhr9911@gmail.com>
Hazem Alrawi <hazemalrawi7@gmail.com>
Shahryar Soltanpour <shahryar.soltanpour@gmail.com>
Shikhar Soni <shikharish05@gmail.com>
Nguyen Cong Nhat Le <lenguyencongnhat2001@gmail.com>
Chao Gu <chadraven369@gmail.com>
Luchen Zhao <lucian.zlc@gmail.com>
Joan Jeremiah J <joanjeremiah04@gmail.com>
Iury Santos <iuryroberto@gmail.com>
Palak Chaturvedi <palakchaturvedi2843@gmail.com>
Jakub Jirutka <jakub@jirutka.cz>
Mario Rodas
Annupamaa <annu242005@gmail.com>
Ashutosh Sharma <ash2003sharma@gmail.com>
Mohab Yaser <mohabyaserofficial2003@gmail.com>
Georg Pfuetzenreuter <mail@georg-pfuetzenreuter.net>
Ahmed Ashour <a8087027@gmail.com>
Sangkeun J.C. Kim <jchrys@me.com>
Tejas Tyagi <tejastyagi.tt@gmail.com>
Aryan Arora <aryanarora.w1@gmail.com>
Arshdeep Singh <balarsh535@gmail.com>
Din Xin Chen <s990346@gmail.com>
Mingzhuo Yin <yinmingzhuo@gmail.com>
Vanes Angelo <k124k3n@gmail.com>
Bassam Adnan <mailbassam@gmail.com>
```

16.2 Committers

```
Jesper Pedersen <jesperpedersen.db@gmail.com>
Haoran Zhang <andrewzhr9911@gmail.com>
```

16.3 Contributing

Contributions to **pgmoneta** are managed on GitHub

- Ask a question
- Raise an issue
- Feature request
- Code submission

Contributions are most welcome!

Please, consult our Code of Conduct policies for interacting in our community.

Consider giving the project a star on GitHub if you find it useful. And, feel free to follow the project on Twitter as well.

17 License

```
Copyright (C) 2025 The pgmoneta community

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following
disclaimer in the documentation and/or other materials provided
with the distribution.

3. Neither the name of the copyright holder nor the names of
its contributors may be used to endorse or promote products
derived from this software without specific prior written
permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF SUCH DAMAGE.
```

BSD-3-Clause

17.1 libart

Our adaptive radix tree (ART) implementation is based on The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases and libart which has a 3-BSD license as

```
Copyright (c) 2012, Armon Dadgar
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

```
* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in the
  documentation and/or other materials provided with the distribution.

* Neither the name of the organization nor the
  names of its contributors may be used to endorse or promote products
  derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ARMON DADGAR
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```