pgmoneta

Developer 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 32+
- RHEL 8 / RockyLinux 8
- 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 8 / RockyLinux 8

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest
    -8.noarch.rpm
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL
    -8-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

and do the install via

```
dnf install -y pgmoneta
```

2.3 RHEL 9 / RockyLinux 9

```
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
```

and do the install via

```
dnf install -y pgmoneta
```

2.4 Compiling the source

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

pgmoneta requires

- gcc 8+ (C17)
- cmake
- make
- libev
- OpenSSL
- zlib
- zstd
- lz4
- bzip2
- systemd
- rst2man
- libssh
- libcurl
- libarchive

```
dnf install git gcc 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 clang 8+ can be used.

2.4.1 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

```
# On RHEL 8 / Rocky 8
dnf config-manager --set-enabled codeready-builder-for-rhel-8-rhui-rpms
dnf config-manager --set-enabled powertools
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.
    noarch.rpm

# On RHEL 9 / Rocky 9, PowerTools is called crb (CodeReady Builder)
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.

2.4.2 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
```

2.4.3 Build

2.4.3.1 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**.

2.4.3.2 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_BUILD_TYPE=Debug ..
make
```

2.5 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/2.4.2/Eisvogel-2.4.2.tar.gz
tar -xzf Eisvogel-2.4.2.tar.gz
mkdir -p $HOME/.local/share/pandoc/templates
mv eisvogel.latex $HOME/.local/share/pandoc/templates
```

where $HOME is your home directory.

2.5.1 Build

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

2.6 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.

2.6.1 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.

2.6.2 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)
```

3 Git guide

Here are some links that will help you

- How to Squash Commits in Git
- ProGit book

3.1 Basic steps

3.1.1 Start by forking the repository

This is done by the “Fork” button on GitHub.

3.2 Clone your repository locally

This is done by

```
git clone git@github.com:<username>/pgmoneta.git
```

3.2.1 Add upstream

Do

```
cd pgmoneta
git remote add upstream https://github.com/pgmoneta/pgmoneta.git
```

3.2.2 Do a work branch

```
git checkout -b mywork main
```

3.2.3 Make the changes

Remember to verify the compile and execution of the code.

Use

```
[#xyz] Description
```

as the commit message where `[#xyz]` is the issue number for the work, and `Description` is a short description of the issue in the first line

3.2.4 Multiple commits

If you have multiple commits on your branch then squash them

```
git rebase -i HEAD~2
```

for example. It is `p` for the first one, then `s` for the rest

3.2.5 Rebase

Always rebase

```
git fetch upstream
git rebase -i upstream/main
```

3.2.6 Force push

When you are done with your changes force push your branch

```
git push -f origin mywork
```

and then create a pull request for it

3.2.7 Format source code

Use

```
./uncrustify.sh
```

to format the source code

3.2.8 Repeat

Based on feedback keep making changes, squashing, rebasing and force pushing

3.2.9 Undo

Normally you can reset to an earlier commit using `git reset <commit hash> --hard`.

But if you accidentally squashed two or more commits, and you want to undo that, you need to know where to reset to, and the commit seems to have lost after you rebased.

But they are not actually lost - using `git reflog`, you can find every commit the HEAD pointer has ever pointed to. Find the commit you want to reset to, and do `git reset --hard`.

4 Architecture

4.1 Overview

pgmoneta use a process model (`fork()`), where each process handles one Write-Ahead Log (WAL) receiver to PostgreSQL.

The main process is defined in main.c.

Backup is handled in backup.h (backup.c).

Restore is handled in restore.h (restore.c) with linking handled in link.h (link.c).

Archive is handled in achv.h (archive.c) backed by restore.

Write-Ahead Log is handled in wal.h (wal.c).

Backup information is handled in info.h (info.c).

Retention is handled in retention.h (retention.c).

Compression is handled in gzip.h (gzip.c) and zstandard.h (zstandard.c).

4.2 Shared memory

A memory segment (shmem.h) is shared among all processes which contains the **pgmoneta** state containing the configuration and the list of servers.

The configuration of **pgmoneta** (`struct configuration`) and the configuration of the servers (`struct server`) is initialized in this shared memory segment. These structs are all defined in pgmoneta.h.

The shared memory segment is created using the `mmap()` call.

4.3 Network and messages

All communication is abstracted using the `struct message` data type defined in messge.h.

Reading and writing messages are handled in the message.h (message.c) files.

Network operations are defined in network.h (network.c).

4.4 Memory

Each process uses a fixed memory block for its network communication, which is allocated upon startup of the process.

That way we don't have to allocate memory for each network message, and more importantly free it after end of use.

The memory interface is defined in memory.h (memory.c).

4.5 Management

pgmoneta has a management interface which defines the administrator abilities that can be performed when it is running. This include for example taking a backup. The `pgmoneta-cli` program is used for these operations (cli.c).

The management interface use Unix Domain Socket for communication.

The management interface is defined in management.h. The management interface uses its own protocol which always consist of a header

Field	Type	Description
`id`	Byte	The identifier of the message type

The rest of the message is depending on the message type.

4.5.1 Remote management

The remote management functionality uses the same protocol as the standard management method.

However, before the management packet is sent the client has to authenticate using SCRAM-SHA-256 using the same message format that PostgreSQL uses, e.g. StartupMessage, AuthenticationSASL, AuthenticationSASLContinue, AuthenticationSASLFinal and AuthenticationOk. The SSLRequest message is supported.

The remote management interface is defined in remote.h (remote.c).

4.6 libev usage

libev is used to handle network interactions, which is "activated" upon an `EV_READ` event.

Each process has its own event loop, such that the process only gets notified when data related only to that process is ready. The main loop handles the system wide “services” such as idle timeout checks and so on.

4.7 Signals

The main process of **pgmoneta** supports the following signals `SIGTERM`, `SIGINT` and `SIGALRM` as a mechanism for shutting down. The `SIGABRT` is used to request a core dump (`abort()`).

The `SIGHUP` signal will trigger a reload of the configuration.

It should not be needed to use `SIGKILL` for **pgmoneta**. Please, consider using `SIGABRT` instead, and share the core dump and debug logs with the **pgmoneta** community.

4.8 Reload

The `SIGHUP` signal will trigger a reload of the configuration.

However, some configuration settings requires a full restart of **pgmoneta** in order to take effect. These are

- `hugepage`
- `libev`
- `log_path`
- `log_type`
- `unix_socket_dir`
- `pidfile`

The configuration can also be reloaded using `pgmoneta-cli -c pgmoneta.conf conf reload`. The command is only supported over the local interface, and hence doesn’t work remotely.

4.9 Prometheus

pgmoneta has support for Prometheus when the `metrics` port is specified.

The module serves two endpoints

- / - Overview of the functionality (`text/html`)
- `/metrics` - The metrics (`text/plain`)

All other URLs will result in a 403 response.

The metrics endpoint supports `Transfer-Encoding: chunked` to account for a large amount of data.

The implementation is done in prometheus.h and prometheus.c.

4.10 Logging

Simple logging implementation based on a `atomic_schar` lock.

The implementation is done in logging.h and logging.c.

4.11 Protocol

The protocol interactions can be debugged using Wireshark or pgprtdbg.

5 Encryption

5.1 Overview

AES Cipher block chaining (CBC) mode and AES Counter (CTR) mode are supported in **pgmoneta**. The default setup is no encryption.

CBC is the most commonly used and considered save mode. Its main drawbacks are that encryption is sequential (decryption can be parallelized).

Along with CBC, CTR mode is one of two block cipher modes recommended by Niels Ferguson and Bruce Schneier. Both encryption and decryption are parallelizable.

Longer the key length, safer the encryption. However, with 20% (192 bit) and 40% (256 bit) extra workload compare to 128 bit.

5.2 Encryption Configuration

`none`: No encryption (default value)

`aes` | `aes-256` | `aes-256-cbc`: AES CBC (Cipher Block Chaining) mode with 256 bit key length

`aes-192` | `aes-192-cbc`: AES CBC mode with 192 bit key length

`aes-128` | `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

5.3 Encryption / Decryption CLI Commands

5.3.1 decrypt

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

Command

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

5.3.2 encrypt

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

Command

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

5.4 Benchmark

Check if your CPU have AES-NI

```
cat /proc/cpuinfo | grep aes
```

Query number of cores on your CPU

```
lscpu | grep '^CPU(s):'
```

By default openssl using AES-NI if the CPU have it.

```
openssl speed -elapsed -evp aes-128-cbc
```

Speed test with explicit disabled AES-NI feature

```
OPENSSL_ia32cap="~0x200000200000000" openssl speed -elapsed -evp aes-128-
    cbc
```

Test decrypt

```
openssl speed -elapsed -decrypt -evp aes-128-cbc
```

Speed test with 8 cores

```
openssl speed -multi 8 -elapsed -evp aes-128-cbc
```

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  12
  On-line CPU(s) list:   0-11
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
    CPU family:          6
    Model:               158
    Thread(s) per core:  2
    Core(s) per socket:  6
    Socket(s):           1
```

```
    Stepping:             10
    BogoMIPS:             5183.98
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
       pge mca cmov pat pse36 clflush mmx fxsr sse sse2 s
                          s ht syscall nx pdpe1gb rdtscp lm constant_tsc
                             rep_good nopl xtopology cpuid pni pclmulqdq
                             vmx ssse
                          3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes
                             xsave avx f16c rdrand hypervisor lahf_lm abm 3
                             dnowpr
                          efetch invpcid_single pti ssbd ibrs ibpb stibp
                             tpr_shadow vnmi ept vpid ept_ad fsgsbase bmi1
                             avx2 s
                          mep bmi2 erms invpcid rdseed adx smap clflushopt
                             xsaveopt xsavec xgetbv1 xsaves flush_l1d
                             arch_capa
                          bilities
Virtualization features:
  Virtualization:         VT-x
  Hypervisor vendor:      Microsoft
  Virtualization type:    full
Caches (sum of all):
  L1d:                    192 KiB (6 instances)
  L1i:                    192 KiB (6 instances)
  L2:                     1.5 MiB (6 instances)
  L3:                     12 MiB (1 instance)
Vulnerabilities:
  Itlb multihit:          KVM: Mitigation: VMX disabled
  L1tf:                   Mitigation; PTE Inversion; VMX conditional cache
     flushes, SMT vulnerable
  Mds:                    Vulnerable: Clear CPU buffers attempted, no
     microcode; SMT Host state unknown
  Meltdown:               Mitigation; PTI
  Spec store bypass:      Mitigation; Speculative Store Bypass disabled via
      prctl and seccomp
  Spectre v1:             Mitigation; usercopy/swapgs barriers and __user
     pointer sanitization
  Spectre v2:             Mitigation; Full generic retpoline, IBPB
     conditional, IBRS_FW, STIBP conditional, RSB filling
  Srbds:                  Unknown: Dependent on hypervisor status
  Tsx async abort:        Not affected

openssl version: 3.0.5
built on: Tue Jul  5 00:00:00 2022 UTC
options: bn(64,64)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -O2 -flto=auto -ffat-
    lto-objects -fexceptions -g -grecord-gcc-switches -pipe -Wall -Werror=
    format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-D_GLIBCXX_ASSERTIONS -
    specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -fstack-protector-strong
    -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1  -m64  -mtune=generic -
    fasynchronous-unwind-tables -fstack-clash-protection -fcf-protection -
```

```
    O2 -flto=auto -ffat-lto-objects -fexceptions -g -grecord-gcc-switches -
    pipe -Wall -Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-
    D_GLIBCXX_ASSERTIONS -specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -
    fstack-protector-strong -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -
    m64 -mtune=generic -fasynchronous-unwind-tables -fstack-clash-
    protection -fcf-protection -Wa,--noexecstack -Wa,--generate-missing-
    build-notes=yes -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -specs=/
    usr/lib/rpm/redhat/redhat-annobin-cc1 -DOPENSSL_USE_NODELETE -DL_ENDIAN
     -DOPENSSL_PIC -DOPENSSL_BUILDING_OPENSSL -DZLIB -DNDEBUG -DPURIFY -
    DDEVRANDOM="\"/dev/urandom\"" -DSYSTEM_CIPHERS_FILE="/etc/crypto-
    policies/back-ends/openssl.config"
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192
    bytes  16384 bytes
AES-128-CBC *   357381.06k   414960.06k   416301.23k   416687.10k
    416175.45k   416268.29k
AES-128-CBC     902160.83k  1496344.68k  1514778.62k  1555236.52k
    1542537.22k  1569259.52k
AES-128-CBC d   909710.79k  2941259.46k  5167110.31k  5927086.76k
    6365967.70k  6349198.68k
AES-128-CBC 8  3912786.36k  8042348.31k  9870507.86k 10254096.38k
    10653332.82k 10310331.05k
AES-128-CBC 8d 4157037.26k 12337480.36k 26613686.27k 29902703.27k
    32306793.13k 31440366.25k

AES-128-CTR *   146971.83k   165696.94k   574871.64k   634507.61k
    676448.94k   668139.52k
AES-128-CTR     887783.06k  2255074.22k  4800168.19k  5930596.01k
    6431110.49k  6376062.98k
AES-128-CTR d   793432.63k  2181439.06k  4541298.09k  5743022.42k
    6480090.45k  6271221.76k
AES-128-CTR 8  3833975.47k 10832239.55k 23757293.40k 28413146.79k
    30514317.99k 30092356.27k
AES-128-CTR 8d 3456838.44k  9749773.91k 22107652.18k 27229352.28k
    30703026.18k 29387025.07k

AES-192-CBC     853380.50k  1238507.90k  1299788.12k  1257189.03k
    1272591.70k  1271840.77k
AES-192-CBC d   876094.29k  2843770.82k  4523019.52k  5177496.92k
    5442652.84k  5372559.36k
AES-192-CTR     869039.84k  2285946.18k  4229439.91k  5049118.04k
    5422994.77k  5309748.57k
AES-192-CTR d   789470.51k  2177050.05k  4194812.76k  4935891.63k
    5257865.90k  5323046.91k

AES-256-CBC     834298.24k  1100648.64k  1117826.90k  1104301.40k
    1130657.11k  1097285.63k
AES-256-CBC d   843079.68k  2714917.67k  4084088.23k  4510005.59k
    4557821.27k  4594783.57k
AES-256-CTR     811325.74k  2222582.89k  3749333.08k  4412143.27k
    4640549.55k  4554828.46k
```

```
AES-256-CTR d   730844.97k  2081179.20k  3673258.15k  4346793.64k
    4515722.58k  4594335.74k
```

*: AES-NI disabled; 8: 8 cores; d: decryption

6 RPM

pgmoneta can be built into a RPM for Fedora systems.

6.1 Requirements

```
dnf install gcc rpm-build rpm-devel rpmlint make python bash coreutils
   diffutils patch rpmdevtools chrpath
```

6.2 Setup RPM development

```
rpmdev-setuptree
```

6.3 Create source package

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make package_source
```

6.4 Create RPM package

```
cp pgmoneta-$VERSION.tar.gz ~/rpmbuild/SOURCES
QA_RPATHS=0x0001 rpmbuild -bb pgmoneta.spec
```

The resulting RPM will be located in `~/rpmbuild/RPMS/x86_64/`, if your architecture is `x86_64`.

7 Test

7.1 Container Environment

7.1.1 Docker

First, ensure your system is up to date.

```
dnf update
```

Install the necessary packages for Docker.

```
dnf -y install dnf-plugins-core
```

Add the Docker repository to your system.

```
sudo dnf config-manager --add-repo https://download.docker.com/linux/
   fedora/docker-ce.repo
```

Install Docker Engine, Docker CLI, and Containerd.

```
sudo dnf install docker-ce docker-ce-cli containerd.io
```

Start the Docker service and enable it to start on boot.

```
sudo systemctl start docker
sudo systemctl enable docker
```

Verify that Docker is installed correctly.

```
docker --version
```

If you see the Docker version, then you have successfully installed Docker on Fedora.

7.1.2 Podman

Install Podman and the Docker alias package.

```
dnf install podman podman-docker.noarch
```

Verify that Podman is installed correctly.

```
podman --version
```

If you see the Podman version, then you have successfully installed Podman on Fedora.

The podman-docker.noarch package simplifies the use of Podman for users accustomed to Docker.

7.2 Test suite

You can simply use CTest to test all PostgreSQL versions from 13 to 16. It will automatically run testsuite.sh to test pgmoneta and pgmoneta_ext for each version. The script will automatically create the Docker container, run it, and then use the check framework to test their functions inside it. After that, it will automatically clean up everything for you.

Go to the directory /pgmoneta/test, and give permission to testsuite.sh using:

```
chmod +x testsuite.sh
```

After you follow the DEVELOPERS.md to install pgmoneta, go to the directory /pgmoneta/build and run the test.

```
make test
```

CTest will output logs into /pgmoneta/build/Testing/Temporary/LastTest.log. If you want to check the specific process, you can review that log file.

testsuite.sh accepts three variables. The first one is dir, which specifies the /test directory location, with a default value of ./. The second one is dockerfile, with a default value of Dockerfile.rocky8. The third one is the PostgreSQL version, with a default value of 13.

8 Troubleshooting

8.1 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.

9 Acknowledgement

9.1 Authors

pgmoneta was created by the following authors:

```
Jesper Pedersen <jesper.pedersen@comcast.net>
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>
```

9.2 Committers

```
Jesper Pedersen <jesper.pedersen@comcast.net>
Haoran Zhang <andrewzhr9911@gmail.com>
```

9.3 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.

10 License

```
Copyright (C) 2024 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

10.1 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.
```