ProxySQL Tools Documentation

Release 0.3.12

TwinDB Development Team

Dec 29, 2017

Contents

Contents:

CHAPTER 1

ProxySQL Tools

ProxySQL Tools

- Free software: Apache Software License 2.0
- Documentation: https://proxysql-tools.readthedocs.io.

1.1 Features

- ProxySQL on AWS support
- ProxySQL and Percona XtraDB Cluster integration
- ProxySQL scheduler

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install ProxySQL Tools, run this command in your terminal:

```
$ pip install proxysql_tools
```

This is the preferred method to install ProxySQL Tools, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for ProxySQL Tools can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/twindb/proxysql_tools
```

Or download the tarball:

```
$ curl  -OL https://github.com/twindb/proxysql_tools/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

ProxySQL Tool is a command line tool. Check help message for syntax and options:

```
$ proxysql-tool
Usage: proxysql-tool [OPTIONS] COMMAND [ARGS]...

Options:
  --debug        Print debug messages
  --config TEXT  ProxySQL Tools configuration file.  [default: /etc/twindb
                 /proxysql-tools.cfg]
  --version      Show tool version and exit.
  --help         Show this message and exit.

Commands:
  aws     Commands to interact with ProxySQL on AWS.
  galera  Commands for ProxySQL and Galera integration.
  ping    Checks the health of ProxySQL.
```

3.1 Configuration file

By default `proxysql-tool` looks for a config in `/etc/twindb/proxysql-tools.cfg`.

Example:

```
[proxysql]
host=172.25.3.100
admin_port=6032
admin_username=admin
admin_password=admin

monitor_username=monitor
monitor_password=monitor

[galera]
```

```
cluster_host=172.25.3.10
cluster_port=3306
cluster_username=root
cluster_password=r00t

load_balancing_mode=singlewriter

writer_hostgroup_id=10
reader_hostgroup_id=11
```

CHAPTER 4

proxysql_tools

4.1 proxysql_tools package

4.1.1 Subpackages

proxysql_tools.aws package

Submodules

proxysql_tools.aws.aws module

proxysql_tools.aws.aws.**attach_network_interface**(*network_interface*, *instance_id*)

proxysql_tools.aws.aws.**aws_notify_master**(*cfg*)

The function moves network interface to local instance and brings it up. Steps:

- Detach network interface if attached to anywhere.
- Attach the network interface to the local instance.
- Configure IP address on this instance

Parameters **cfg** – config object

proxysql_tools.aws.aws.**configure_local_interface**(*local_interface*, *ip*, *netmask*)

proxysql_tools.aws.aws.**detach_network_interface**(*network_interface*)

proxysql_tools.aws.aws.**ensure_local_interface_is_gone**(*local_interface*)

proxysql_tools.aws.aws.**ensure_network_interface_is_detached**(*network_interface*)

proxysql_tools.aws.aws.**get_my_instance_id**()

proxysql_tools.aws.aws.**get_network_interface**(*ip*)

proxysql_tools.aws.aws.**get_network_interface_state**(*network_interface*)

proxysql_tools.aws.aws.**network_interface_attached**(*network_interface*)

Check whether network interface is attached

Parameters **network_interface** – network interface id

Returns True or False

Module contents

proxysql_tools.galera package

Submodules

proxysql_tools.galera.galera_cluster module

Module describes GaleraCluster class

class proxysql_tools.galera.galera_cluster.**GaleraCluster**(*cluster_hosts*, *user='root'*, *password=None*)

Bases: object

GaleraCluster describes Galera cluster.

Parameters

- **cluster_hosts** (*str*) – .
- **user** (*str*) – MySQL user to connect to a cluster node.
- **password** (*str*) – MySQL password.

find_node(*host*, *port*)

BY given host and port find a node in the cluster.

Parameters

- **host** – IP or address of node.
- **port** – node port.

Returns GaleraNode instance.

Return type *GaleraNode*

Raise GaleraClusterNodeNotFound

find_synced_nodes()

Find a node in the cluster in SYNCED state. :return: List of Galera node in SYNCED state. :rtype: list(GaleraNode) :raise: GaleraClusterSyncedNodeNotFound

nodes

Get list of Galera nodes

Returns Return list of Galera nodes

Return type list(*GaleraNode*)

proxysql_tools.galera.galera_node module

Module describes GaleraNode class

class proxysql_tools.galera.galera_node.**GaleraNode**(*host*, *port=3306*, *user='root'*, *password=None*)

Bases: object

GaleraNode class describes a single node in Galera Cluster.

Parameters

- **host** – hostname of the node.
- **port** – port to connect to.
- **user** – MySQL username to connect to the node.
- **password** – MySQL password.

execute(*query*, **args*)

Execute query in Galera Node.

Parameters **query** (*str*) – Query to execute.

Returns Query result or None if the query is not supposed to return result.

Return type dict

wsrep_cluster_name

The logical cluster name for the node.

wsrep_cluster_state_uuid

Provides the current State UUID. This is a unique identifier for the current state of the cluster and the sequence of changes it undergoes.

wsrep_cluster_status

Status of this cluster component. That is, whether the node is part of a PRIMARY or NON_PRIMARY component.

wsrep_local_state

Internal Galera Cluster FSM state number.

class proxysql_tools.galera.galera_node.**GaleraNodeState**

Bases: object

State of Galera node http://bit.ly/2r1tUGB

DONOR = 5

JOINED = 3

JOINER = 2

PRIMARY = 1

SYNCED = 4

Module contents

proxysql_tools.proxysql package

Submodules

proxysql_tools.proxysql.proxysql module

ProxySQL classes

class proxysql_tools.proxysql.proxysql.**BackendStatus**
Bases: object

Status of ProxySQL backend

offline_hard = 'OFFLINE_HARD'

offline_soft = 'OFFLINE_SOFT'

online = 'ONLINE'

shunned = 'SHUNNED'

class proxysql_tools.proxysql.proxysql.**ProxySQL**(*host='localhost'*, *port=3306*, *user='root'*, *password=None*, *socket=None*)
Bases: object

ProxySQL describes a single ProxySQL instance.

Parameters

- **host** – ProxySQL hostname.
- **port** – Port on which ProxySQL listens to admin connections.
- **user** – ProxySQL admin user.
- **password** – Password for ProxySQL admin.
- **socket** – Socket to connect to ProxySQL admin interface.

add_user(*user*)
Add MySQL user

Parameters **user** (ProxySQLMySQLUser) – user for add

backend_registered(*backend*)
Check if backend is registered.

Parameters **backend** – ProxySQLMySQLBackend instance

Returns True if registered, False otherwise

Return type bool

delete_user(*username*)
Delete MySQL user

Parameters **username** (*str*) – username of user

deregister_backend(*backend*)
Deregister a Galera node from ProxySQL

Parameters **backend** (ProxySQLMySQLBackend) – Galera node.

execute(*query*, **args*)
Execute query in ProxySQL.

Parameters **query** (*str*) – Query to execute.

Returns Query result or None if the query is not supposed to return result

Return type dict

find_backends(*hostgroup_id*, *status=None*)
Get writer from mysql_servers

Parameters

- **hostgroup_id** (*int*) – writer hostgroup_id
- **status** (BackendStatus) – Look only for backends in this status

Returns Writer MySQL backend or None if doesn't exist

Return type list(*ProxySQLMySQLBackend*)

Raise ProxySQLBackendNotFound

get_user(*username*)
Get user by username

Parameters **username** – Username

Returns User information

Return type *ProxySQLMySQLUser*

Raise ProxySQLUserNotFound

get_users()
Get mysql users

Returns List of users or empty list

Return type list(*ProxySQLMySQLUser*)

ping()
Check health of ProxySQL.

Returns True if ProxySQL healthy and False otherwise.

Return type bool

register_backend(*backend*)
Register Galera node in ProxySQL

Parameters **backend** (ProxySQLMySQLBackend) – Galera node.

reload_runtime()
Reload the ProxySQL runtime configuration.

save_user()
Save user to on-disk database

set_status(*backend*, *status*)
Update status of a backend in ProxySQL

class proxysql_tools.proxysql.proxysql.**ProxySQLMySQLBackend**(*hostname*, *hostgroup_id=0*, *port=3306*, *status='ONLINE'*, *weight=1*, *compression=0*, *max_connections=10000*, *max_replication_lag=0*, *use_ssl=False*, *max_latency_ms=0*, *comment=None*)

Bases: object

ProxySQLMySQLBackend describes record in ProxySQL table mysql_servers.

```
CREATE TABLE mysql_servers (
    hostgroup_id INT NOT NULL DEFAULT 0,
    hostname VARCHAR NOT NULL,
    port INT NOT NULL DEFAULT 3306,
    status VARCHAR CHECK (UPPER(status) IN
        ('ONLINE','SHUNNED','OFFLINE_SOFT', 'OFFLINE_HARD'))
        NOT NULL DEFAULT 'ONLINE',
    weight INT CHECK (weight >= 0) NOT NULL DEFAULT 1,
    compression INT CHECK (compression >=0 AND compression <= 102400)
        NOT NULL DEFAULT 0,
    max_connections INT CHECK (max_connections >=0) NOT NULL DEFAULT 1000,
    max_replication_lag INT CHECK (max_replication_lag >= 0
        AND max_replication_lag <= 126144000) NOT NULL DEFAULT 0,
    use_ssl INT CHECK (use_ssl IN(0,1)) NOT NULL DEFAULT 0,
    max_latency_ms INT UNSIGNED CHECK (max_latency_ms>=0)
        NOT NULL DEFAULT 0,
    comment VARCHAR NOT NULL DEFAULT '',
    PRIMARY KEY (hostgroup_id, hostname, port) )
```

connect(*username*, *password*)

Make a MySQL connection to the backend.

Parameters

- **username** – MySQL user.
- **password** – MySQL password.

execute(*query*, **args*)

Execute query in MySQL Backend.

Parameters **query** (*str*) – Query to execute.

Returns Query result or None if the query is not supposed to return result

Return type dict

class proxysql_tools.proxysql.proxysql.**ProxySQLMySQLUser**(*username='root'*, *password=None*, *active=True*, *use_ssl=False*, *default_hostgroup=0*, *default_schema='information_schema'*, *schema_locked=False*, *transaction_persistent=False*, *fast_forward=False*, *backend=True*, *frontend=True*, *max_connections=10000*)

Bases: `object`

ProxySQLMySQLUser describes record in ProxySQL table `mysql_users`.

```
CREATE TABLE mysql_users (
    username VARCHAR NOT NULL,
    password VARCHAR,
    active INT CHECK (active IN (0,1)) NOT NULL DEFAULT 1,
    use_ssl INT CHECK (use_ssl IN (0,1)) NOT NULL DEFAULT 0,
    default_hostgroup INT NOT NULL DEFAULT 0,
    default_schema VARCHAR,
    schema_locked INT CHECK (schema_locked IN (0,1)) NOT NULL DEFAULT 0,
    transaction_persistent INT CHECK (transaction_persistent IN (0,1))
        NOT NULL DEFAULT 0,
    fast_forward INT CHECK (fast_forward IN (0,1)) NOT NULL DEFAULT 0,
    backend INT CHECK (backend IN (0,1)) NOT NULL DEFAULT 1,
    frontend INT CHECK (frontend IN (0,1)) NOT NULL DEFAULT 1,
    max_connections INT CHECK (max_connections >=0) NOT NULL DEFAULT 10000,
    PRIMARY KEY (username, backend),
    UNIQUE (username, frontend))
```

Parameters

- **username** – MySQL username to connect to ProxySQL or Galera node.
- **password** – MySQL password.
- **active** (*bool*) – Users with active = 0 will be tracked in the database, but will be never loaded in the in-memory data structures.
- **use_ssl** (*bool*) – Use SSL to connect to MySQL or not
- **default_hostgroup** – If there is no matching rule for the queries sent by the users, the traffic it generates is sent to the specified hostgroup.
- **default_schema** – The schema to which the connection should change by default.
- **schema_locked** (*bool*) – not supported yet.
- **transaction_persistent** (*bool*) – if this is set for the user with which the MySQL client is connecting to ProxySQL (thus a "frontend" user - see below), transactions started within a hostgroup will remain within that hostgroup regardless of any other rules.
- **fast_forward** (*bool*) – If set, it bypasses the query processing layer (rewriting, caching) and passes the query directly to the backend server.
- **frontend** (*bool*) – If True, this (username, password) pair is used for authenticating to the ProxySQL instance.

- **backend** – If True, this (username, password) pair is used for authenticating to the mysqld servers against any hostgroup.
- **max_connections** – Maximum number of connection this user can create to MySQL node.

Module contents

4.1.2 Submodules

4.1.3 proxysql_tools.cli module

Entry points for proxysql-tools

proxysql_tools.cli.**validate_password**(*ctx*, *param*, *value*)
CHeck password value and confirm again if it's empty.

4.1.4 Module contents

proxysql_tools module

proxysql_tools.**execute**(*conn*, *query*, **args*)
Execute query in connection

proxysql_tools.**setup_logging**(*logger*, *debug=False*)
Configure logging

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/twindb/proxysql-tools/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

ProxySQL Tools could always use more documentation, whether as part of the official ProxySQL Tools docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/twindb/proxysql-tools/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *proxysql_tools* for local development.

1. Fork the *proxysql_tools* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/proxysql_tools.git
```

3. Install `virtualenvwrapper` and create virtual environment:

```
$ sudo pip install virtualenvwrapper
```

   Check virtualenvwrapper documentation for details.

   Create environment for `proxysql-tools`:

```
$ mkvirtualenv proxysql-tools
```

4. Install your local copy into a virtualenv:

```
$ workon proxysql-tools
$ make bootstrap
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

6. When you're done making changes, check that your changes pass tests:

```
$ make lint
$ make test
```

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7. Check https://travis-ci.org/twindb/proxysql-tools/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 6

Credits

6.1 Development Lead

- TwinDB Development Team <dev@twindb.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

Development

7.1 Relase new version

1. Checkout to develop branch and get clean code:

```
> git checkout develop
> git reset --hard
> git pull
```

2. Get current version and choose new one:

```
less setup.cfg
[bumpversion]
current_version = <version>
```

or:

```
proxysql-tool --version
```

3. Start release branch using git-flow:

```
git flow release start <version>
bumpversion patch | minor | major
git flow release finish <version>
Tag commit message "Release <version>"
```

4. Push:

```
git push --all
git push --tags
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

N

O

P

R

S

V

W