

---

# **Rabbit Hole Documentation**

***Release 0.3.0***

**Javier Collado**

**May 04, 2017**



---

## Contents

---

<b>1</b>	<b>Rabbit Hole</b>	<b>3</b>
1.1	Features . . . . .	3
1.2	Credits . . . . .	3
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Stable release . . . . .	5
2.2	From sources . . . . .	5
<b>3</b>	<b>Usage</b>	<b>7</b>
3.1	CLI . . . . .	7
3.2	Blocks . . . . .	8
3.3	Flow . . . . .	8
3.4	Available blocks . . . . .	9
<b>4</b>	<b>Contributing</b>	<b>11</b>
4.1	Types of Contributions . . . . .	11
4.2	Get Started! . . . . .	12
4.3	Pull Request Guidelines . . . . .	13
4.4	Tips . . . . .	13
<b>5</b>	<b>Credits</b>	<b>15</b>
5.1	Development Lead . . . . .	15
5.2	Contributors . . . . .	15
<b>6</b>	<b>History</b>	<b>17</b>
6.1	0.1.0 (2016-11-25) . . . . .	17
6.2	0.2.0 (2016-11-28) . . . . .	17
6.3	0.3.0 (2017-05-04) . . . . .	17
<b>7</b>	<b>Indices and tables</b>	<b>19</b>



Contents:



# CHAPTER 1

---

## Rabbit Hole

---

Store messages from an AMQP server into a SQL database

### Features

- Get messages from multiple AMQP exchanges
- Group messages in batches
- Write message batches to a SQL database

### Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.



### Stable release

To install Rabbit Hole, run this command in your terminal:

```
$ pip install rabbithole
```

This is the preferred method to install Rabbit Hole, 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.

### From sources

The sources for Rabbit Hole can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/jcollado/rabbithole
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/jcollado/rabbithole/tarball/master
```

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

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



### CLI

Rabbit Hole is a command line tool that has been written as a lightweight alternative to [logstash](#) for the specific use case in which the *input* is an *amqp* server and the *output* is a SQL database.

It can be executed from the command line like this:

```
$ rabbithole config.yml
```

where *config.yml* is a YAML configuration file. For example:

```
size_limit: 5
time_limit: 15
blocks:
  - name: input
    type: amqp
    kwargs:
      url: 'amqp://username:password@localhost:5672'
  - name: output
    type: sql
    kwargs:
      url: 'postgres://username:password@localhost:5432/db_name'
flows:
  - - name: input
      kwargs:
        exchange: logs
        exchange_type: fanout
        durable: true
    - name: output
      kwargs:
        query:
          INSERT INTO logs (timestamp, message)
          VALUES (CAST (:timestamp AS TIMESTAMP), :message)
        parameters:
```

```
        timestamp: timestamp
        message: message.text
- - name: input
    kwargs:
        exchange: events
        exchange_type: fanout
        durable: true
- name: output
    kwargs:
        query:
            INSERT INTO events (timestamp, message)
            VALUES (CAST (:timestamp AS TIMESTAMP), :message)
        parameters:
            timestamp: timestamp
            message: message.text
```

where:

- *size\_limit*: batcher size limit
- *time\_limit*: batcher size limit
- *blocks*: list of building blocks to use in the flows
- *flows*: list of blocks connected to transfer information information

## Blocks

A block rabbithole is the name of the little piece that can be added to a flow to receive/send messages as needed to build the desired flow of information. There are currently three different kinds of blocks:

### input

an input block is a block that receives a messages from an external source, such as an amqp server, and transfers them as they are received to the next block in the flow.

### batchers

rabbithole uses the concept of batchers that is also used in [logstash](#). A batcher is just an in-memory queue whose goal is to output data more efficiently by writing multiple messages at once. It keeps messages in memory until its capacity has been filled up or until a time limit is exceeded. Both parameters can be set in the configuration file.

Batchers are automatically added between blocks in a flow, so there's no need to include them explicitly in the configuration file.

### output

an output block is a block that receives messages from the previous block and sends them to an external output such as a database.

## Flow

A flow is a sequence of blocks that are connected to transfer information from the initial input block to the final output one.

## Available blocks

The following blocks are available in rabbithole.

### amqp

amqp is an input flow that can receive data from amqp servers.

```
blocks:
  - name: input
    type: amqp
    kwargs:
      url: 'amqp://username:password@localhost:5672'

flows:
  - - name: input
    kwargs:
      exchange: logs
      exchange_type: fanout
      durable: true
```

where:

- *url*: is the [AMQP connection string](#).
- *exchange* is the name of the exchange for which messages will be transferred in a given flow.
- additional parameters are optional and passed directly to `pika.channel.Channel.exchange_declare`.

### sql

sql is an output flow that can write data to SQL databases.

```
blocks:
  - name: output
    type: sql
    kwargs:
      url: 'postgres://username:password@localhost:5432/db_name'

flows:
  - - name: output
    kwargs:
      query:
        INSERT INTO logs (timestamp, message)
        VALUES (CAST (:timestamp AS TIMESTAMP), :message)
      parameters:
        timestamp: timestamp
        message: message.text
```

where:

- *url* is the [database connection string](#).
- *query* is the [query](#) to execute when a message is received in a given flow.
- *parameters* is an optional mapping from the message received to the object passed to the query (useful when the message contains nested data since nesting is not supported in query parameters).



Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

### Types of Contributions

#### Report Bugs

Report bugs at <https://github.com/jcollado/rabbithole/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.

#### Fix Bugs

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

#### Implement Features

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

## Write Documentation

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

## Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/jcollado/rabbithole/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 :)

## Get Started!

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

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

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

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv rabbithole
$ cd rabbithole/
$ python setup.py develop
```

4. Create a branch for local development:

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

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 rabbithole tests
$ python setup.py test or py.test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. 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.

## 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 and 3.5. Check [https://travis-ci.org/jcollado/rabbithole/pull\\_requests](https://travis-ci.org/jcollado/rabbithole/pull_requests) and make sure that the tests pass for all supported Python versions.

## Tips

To run a subset of tests:

```
$ python -m unittest tests.test_rabbithole
```



## CHAPTER 5

---

### Credits

---

#### Development Lead

- Javier Collado <[javier@gigaspace.com](mailto:javier@gigaspace.com)>

#### Contributors

None yet. Why not be the first?



#### **0.1.0 (2016-11-25)**

- First release on PyPI.

#### **0.2.0 (2016-11-28)**

- Make batcher size/time limit configurable.
- Added test cases and documentation.

#### **0.3.0 (2017-05-04)**

- Added new flexible configuration format based on the blocks and flows concept.
- Added query parameters support to SQL block.
- Update AMQP block to use a connection URL instead of just the server address.



## CHAPTER 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`