

---

# **tornadis Documentation**

*Release 0.8.0*

**Fabien MARTY**

**Apr 18, 2018**



---

## Contents

---

<b>1</b>	<b>Requirements</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>First try (coroutines)</b>	<b>5</b>
<b>4</b>	<b>Second try (callbacks)</b>	<b>7</b>
<b>5</b>	<b>Go further: Pipeline</b>	<b>9</b>
<b>6</b>	<b>Installation without pip</b>	<b>11</b>
<b>7</b>	<b>API</b>	<b>13</b>
7.1	Client API . . . . .	13
7.2	PubSubClient API . . . . .	15
7.3	Pipeline API . . . . .	17
7.4	Pool API . . . . .	17
7.5	Exceptions . . . . .	18
7.6	Connection API . . . . .	19
	<b>Python Module Index</b>	<b>21</b>



# CHAPTER 1

---

## Requirements

---

- Python 2.7 or Python  $\geq 3.2$
- unix operating system (linux, osx...)
- a running redis server ( $\geq 2.0$ )



## CHAPTER 2

---

### Installation

---

With `pip` (without `pip` see at then end of this document):

```
pip install tornadis
```





---

## First try (coroutines)

---

```
1 # Let's import tornado and tornadis
2 import tornado
3 import tornadis
4
5
6 @tornado.gen.coroutine
7 def talk_to_redis():
8     # let's (re)connect (autoconnect mode), call the ping redis command
9     # and wait the reply without blocking the tornado ioloop
10    # Note: call() method on Client instance returns a Future object (and
11    # should be used as a coroutine).
12    result = yield client.call("PING")
13    if isinstance(result, tornadis.TornadisException):
14        # For specific reasons, tornadis nearly never raises any exception
15        # they are returned as result
16        print "got exception: %s" % result
17    else:
18        # result is already a python object (a string in this simple example)
19        print "Result: %s" % result
20
21
22 # Build a tornadis.Client object with some options as kwargs
23 # host: redis host to connect
24 # port: redis port to connect
25 # autoconnect=True: put the Client object in auto(re)connect mode
26 client = tornadis.Client(host="localhost", port=6379, autoconnect=True)
27
28 # Start a tornado IOLoop, execute the coroutine and end the program
29 loop = tornado.ioloop.IOLoop.instance()
30 loop.run_sync(talk_to_redis)
```



---

## Second try (callbacks)

---

```
1  # Let's import tornado and tornadis
2  import tornado
3  import tornadis
4
5
6  def ping_callback(result):
7      if not isinstance(result, tornadis.TornadisException):
8          # For specific reasons, tornadis nearly never raises any exception
9          # they are returned as result
10         print "got exception: %s" % result
11     else:
12         # result is already a python object (a string in this simple example)
13         print "Result: %s" % result
14
15
16  @tornado.gen.coroutine
17  def main():
18      # let's (re)connect (autoconnect mode), call the ping redis command
19      # and wait the reply without blocking the tornado ioloop
20      # Note: async_call() method on Client instance does not return anything
21      # but the callback will be called later with the result.
22      client.async_call("PING", callback=ping_callback)
23      yield tornado.gen.sleep(1)
24
25
26  # Build a tornadis.Client object with some options as kwargs
27  # host: redis host to connect
28  # port: redis port to connect
29  # autoconnect=True: put the Client object in auto(re)connect mode
30  client = tornadis.Client(host="localhost", port=6379, autoconnect=True)
31
32  # Start a tornado IOloop, execute the coroutine and end the program
33  loop = tornado.ioloop.IOLoop.instance()
34  loop.run_sync(main)
```



---

## Go further: Pipeline

---

```
1 # Let's import tornado and tornadis
2 import tornado
3 import tornadis
4
5
6 @tornado.gen.coroutine
7 def pipeline_coroutine():
8     # Let's make a pipeline object to stack commands inside
9     pipeline = tornadis.Pipeline()
10    pipeline.stack_call("SET", "foo", "bar")
11    pipeline.stack_call("GET", "foo")
12
13    # At this point, nothing is sent to redis
14
15    # let's (re)connect (autoconnect mode), send the pipeline of requests
16    # (atomic mode) and wait all replies without blocking the tornado ioloop.
17    results = yield client.call(pipeline)
18
19    if isinstance(results, tornadis.TornadisException):
20        # For specific reasons, tornadis nearly never raises any exception
21        # they are returned as results
22        print "got exception: %s" % results
23    else:
24        # The two replies are in the results array
25        print results
26        # >>> ['OK', 'bar']
27
28
29 # Build a tornadis.Client object with some options as kwargs
30 # host: redis host to connect
31 # port: redis port to connect
32 # autoconnect=True: put the Client object in auto(re)connect mode
33 client = tornadis.Client(host="localhost", port=6379, autoconnect=True)
34
35 # Start a tornado IOLoop, execute the coroutine and end the program
```

```
36 loop = tornado.ioloop.IOLoop.instance()  
37 loop.run_sync(pipeline_coroutine)
```

---

## Installation without pip

---

- install `tornado >= 4.2`
- install `python wrapper for hiredis`
- install `six`
- download and uncompress a `tornadis` release
- run `python setup.py install` in the `tornadis` directory





## 7.1 Client API

**class** `tornadis.Client` (*autoconnect=True, password=None, db=0, \*\*connection\_kwargs*)

Bases: `object`

High level object to interact with redis.

### Variables

- **autoconnect** (*boolean*) – True if the client is in autoconnect mode (and in autoreconnection mode) (default True).
- **password** (*string*) – the password to authenticate with.
- **db** (*int*) – database number.
- **connection\_kwargs** (*dict*) – `Connection` object kwargs (note that `read_callback` and `close_callback` args are set automatically).

**\_\_init\_\_** (*autoconnect=True, password=None, db=0, \*\*connection\_kwargs*)

Constructor.

### Parameters

- **autoconnect** (*boolean*) – True if the client is in autoconnect mode (and in autoreconnection mode) (default True).
- **password** (*string*) – the password to authenticate with.
- **db** (*int*) – database number.
- **\*\*connection\_kwargs** – `Connection` object kwargs.

**async\_call** (*\*args, \*\*kwargs*)

Calls a redis command, waits for the reply and call a callback.

Following options are available (not part of the redis command itself):

- **callback** Function called (with the result as argument) when the result is available. If not set, the reply is silently discarded. In case of errors, the callback is called with a `TornadisException` object as argument.

#### Parameters

- **\*args** – full redis command as variable length argument list or a Pipeline object (as a single argument).
- **\*\*kwargs** – options as keyword parameters.

#### Examples

```
>>> def cb(result):  
    pass  
>>> client.async_call("HSET", "key", "field", "val", callback=cb)
```

**call** (\*args, \*\*kwargs)

Calls a redis command and returns a Future of the reply.

#### Parameters

- **\*args** – full redis command as variable length argument list or a Pipeline object (as a single argument).
- **\*\*kwargs** – internal private options (do not use).

#### Returns

a Future with the decoded redis reply as result (when available) or a `ConnectionError` object in case of connection error.

**Raises** `ClientError` – your Pipeline object is empty.

#### Examples

```
>>> @tornado.gen.coroutine  
    def foobar():  
        client = Client()  
        result = yield client.call("HSET", "key", "field", "val")
```

**connect** (\*args, \*\*kwargs)

Connects the client object to redis.

It's safe to use this method even if you are already connected. Note: this method is useless with `autoconnect` mode (default).

**Returns** a Future object with True as result if the connection was ok.

**disconnect** ()

Disconnects the client object from redis.

It's safe to use this method even if you are already disconnected.

**is\_connected** ()

Returns True if the client is connected to redis.

**Returns** True if the client is connected to redis.

## 7.2 PubSubClient API

**class** `tornadis.PubSubClient` (*autoconnect=True, password=None, db=0, \*\*connection\_kwargs*)

Bases: `tornadis.client.Client`

High level specific object to interact with pubsub redis.

The `call()` method is forbidden with this object.

More informations on the redis side: <http://redis.io/topics/pubsub>

**\_\_init\_\_** (*autoconnect=True, password=None, db=0, \*\*connection\_kwargs*)

Constructor.

### Parameters

- **autoconnect** (*boolean*) – True if the client is in autoconnect mode (and in autoreconnection mode) (default True).
- **password** (*string*) – the password to authenticate with.
- **db** (*int*) – database number.
- **\*\*connection\_kwargs** – *Connection* object kwargs.

**async\_call** (*\*args, \*\*kwargs*)

Not allowed method with PubSubClient object.

**call** (*\*args, \*\*kwargs*)

Not allowed method with PubSubClient object.

**connect** (*\*args, \*\*kwargs*)

Connects the client object to redis.

It's safe to use this method even if you are already connected. Note: this method is useless with autoconnect mode (default).

**Returns** a Future object with True as result if the connection was ok.

**disconnect** ()

Disconnects the client object from redis.

It's safe to use this method even if you are already disconnected.

**is\_connected** ()

Returns True is the client is connected to redis.

**Returns** True if the client if connected to redis.

**pubsub\_pop\_message** (*\*args, \*\*kwargs*)

Pops a message for a subscribed client.

**Parameters** **deadline** (*int*) – max number of seconds to wait (None => no timeout)

### Returns

**Future with the popped message as result (or None if timeout or ConnectionError object in case of connection errors or ClientError object if you are not subscribed)**

**pubsub\_psubscribe** (*\*args*)

Subscribes to a list of patterns.

<http://redis.io/topics/pubsub>

**Parameters** **\*args** – variable list of patterns to subscribe.

**Returns** Future with True as result if the subscribe is ok.

**Return type** Future

### Examples

```
>>> yield client.psubsub_psubscribe("channel*", "foo*")
```

**psubsub\_punsubscribe** (\*args)

Unsubscribes from a list of patterns.

<http://redis.io/topics/pubsub>

**Parameters** \*args – variable list of patterns to unsubscribe.

**Returns** Future with True as result if the unsubscribe is ok.

**Return type** Future

### Examples

```
>>> yield client.psubsub_punsubscribe("channel*", "foo*")
```

**psubsub\_subscribe** (\*args)

Subscribes to a list of channels.

<http://redis.io/topics/pubsub>

**Parameters** \*args – variable list of channels to subscribe.

**Returns** Future with True as result if the subscribe is ok.

**Return type** Future

### Examples

```
>>> yield client.psubsub_subscribe("channel1", "channel2")
```

**psubsub\_unsubscribe** (\*args)

Unsubscribes from a list of channels.

<http://redis.io/topics/pubsub>

**Parameters** \*args – variable list of channels to unsubscribe.

**Returns** Future with True as result if the unsubscribe is ok.

**Return type** Future

### Examples

```
>>> yield client.psubsub_unsubscribe("channel1", "channel2")
```

## 7.3 Pipeline API

**class** `tornadis.Pipeline`

Bases: `object`

Pipeline class to stack redis commands.

A pipeline object is just a kind of stack. You stack complete redis commands (with their corresponding arguments) inside it.

Then, you use the `call()` method of a `Client` object to process the pipeline (which must be the only argument of this `call()` call).

More informations on the redis side: <http://redis.io/topics/pipelining>

### Variables

- **`pipelined_args`** – A list of tuples, each tuple is a complete redis command.
- **`number_of_stacked_calls`** – the number of stacked redis commands (integer).

**`__init__`** ()

Constructor.

**`stack_call`** (`*args`)

Stacks a redis command inside the object.

The syntax is the same than the `call()` method a `Client` class.

**Parameters** `*args` – full redis command as variable length argument list.

### Examples

```
>>> pipeline = Pipeline()
>>> pipeline.stack_call("HSET", "key", "field", "value")
>>> pipeline.stack_call("PING")
>>> pipeline.stack_call("INCR", "key2")
```

## 7.4 Pool API

**class** `tornadis.ClientPool` (`max_size=-1`, `client_timeout=-1`, `autoclose=False`, `**client_kwargs`)

Bases: `object`

High level object to deal with a pool of redis clients.

**`__init__`** (`max_size=-1`, `client_timeout=-1`, `autoclose=False`, `**client_kwargs`)

Constructor.

### Parameters

- **`max_size`** (`int`) – max size of the pool (-1 means “no limit”).
- **`client_timeout`** (`int`) – timeout in seconds of a connection released to the pool (-1 means “no timeout”).
- **`autoclose`** (`boolean`) – automatically disconnect released connections with lifetime > `client_timeout` (test made every `client_timeout/10` seconds).
- **`client_kwargs`** (`dict`) – Client constructor arguments.

**connected\_client()**

Returns a ContextManagerFuture to be yielded in a with statement.

**Returns** A ContextManagerFuture object.

### Examples

```
>>> with (yield pool.connected_client()) as client:
    # client is a connected tornadis.Client instance
    # it will be automatically released to the pool thanks to
    # the "with" keyword
    reply = yield client.call("PING")
```

**destroy()**

Disconnects all pooled client objects.

**get\_client\_nowait()**

Gets a Client object (not necessary connected).

If max\_size is reached, this method will return None (and won't block).

**Returns** A Client instance (not necessary connected) as result (or None).

**get\_connected\_client(\*args, \*\*kwargs)**

Gets a connected Client object.

If max\_size is reached, this method will block until a new client object is available.

**Returns**

**A Future object with connected Client instance as a result** (or ClientError if there was a connection problem)

**preconnect(\*args, \*\*kwargs)**

(pre)Connects some or all redis clients inside the pool.

**Parameters** *size* (*int*) – number of redis clients to build and to connect (-1 means all clients if pool max\_size > -1)

**Raises** *ClientError* – when size == -1 and pool max\_size == -1

**release\_client(client)**

Releases a client object to the pool.

**Parameters** *client* – Client object.

## 7.5 Exceptions

**class** tornadis.**TornadisException**

Bases: exceptions.Exception

Base Exception class.

**class** tornadis.**ConnectionError**

Bases: tornadis.exceptions.TornadisException

Exception raised when there is a connection error.

```
class tornadis.ClientError
    Bases: tornadis.exceptions.TornadisException

    Exception raised when there is a client error.
```

## 7.6 Connection API

Warning: this class is not public, it appears here just to document some kwargs. Do not use directly.

```
class tornadis.Connection(read_callback, close_callback, host='127.0.0.1', port=6379,
                          unix_domain_socket=None, read_page_size=65536,
                          write_page_size=65536, connect_timeout=20, tcp_nodelay=False,
                          aggressive_write=False, read_timeout=0, ioloop=None)
```

Low level connection object.

### Variables

- **host** (*string*) – the host name to connect to.
- **port** (*int*) – the port to connect to.
- **unix\_domain\_socket** (*string*) – path to a unix socket to connect to (if set, overrides host/port parameters).
- **read\_page\_size** (*int*) – page size for reading.
- **write\_page\_size** (*int*) – page size for writing.
- **connect\_timeout** (*int*) – timeout (in seconds) for connecting.
- **tcp\_nodelay** (*boolean*) – set TCP\_NODELAY on socket.
- **aggressive\_write** (*boolean*) – try to minimize write latency over global throughput (default False).
- **read\_timeout** (*int*) – timeout (in seconds) to read something on the socket (if nothing is read during this time, the connection is closed) (default: 0 means no timeout)

```
__init__(read_callback, close_callback, host='127.0.0.1', port=6379, unix_domain_socket=None,
         read_page_size=65536, write_page_size=65536, connect_timeout=20, tcp_nodelay=False,
         aggressive_write=False, read_timeout=0, ioloop=None)
```

Constructor.

### Parameters

- **read\_callback** – callback called when there is something to read (private, do not use from Client constructor).
- **close\_callback** – callback called when the connection is closed (private, do not use from Client constructor).
- **host** (*string*) – the host name to connect to.
- **port** (*int*) – the port to connect to.
- **unix\_domain\_socket** (*string*) – path to a unix socket to connect to (if set, overrides host/port parameters).
- **read\_page\_size** (*int*) – page size for reading.
- **write\_page\_size** (*int*) – page size for writing.
- **connect\_timeout** (*int*) – timeout (in seconds) for connecting.

- **tcp\_nodelay** (*boolean*) – set TCP\_NODELAY on socket.
- **aggressive\_write** (*boolean*) – try to minimize write latency over global throughput (default False).
- **read\_timeout** (*int*) – timeout (in seconds) to read something on the socket (if nothing is read during this time, the connection is closed) (default: 0 means no timeout)
- **ioloop** (*IOLoop*) – the tornado ioloop to use.



**t**

tornadis, 15



## Symbols

`__init__()` (tornadis.Client method), 13  
`__init__()` (tornadis.ClientPool method), 17  
`__init__()` (tornadis.Connection method), 19  
`__init__()` (tornadis.Pipeline method), 17  
`__init__()` (tornadis.PubSubClient method), 15

## A

`async_call()` (tornadis.Client method), 13  
`async_call()` (tornadis.PubSubClient method), 15

## C

`call()` (tornadis.Client method), 14  
`call()` (tornadis.PubSubClient method), 15  
Client (class in tornadis), 13  
ClientError (class in tornadis), 18  
ClientPool (class in tornadis), 17  
`connect()` (tornadis.Client method), 14  
`connect()` (tornadis.PubSubClient method), 15  
`connected_client()` (tornadis.ClientPool method), 17  
Connection (class in tornadis), 19  
ConnectionError (class in tornadis), 18

## D

`destroy()` (tornadis.ClientPool method), 18  
`disconnect()` (tornadis.Client method), 14  
`disconnect()` (tornadis.PubSubClient method), 15

## G

`get_client_nowait()` (tornadis.ClientPool method), 18  
`get_connected_client()` (tornadis.ClientPool method), 18

## I

`is_connected()` (tornadis.Client method), 14  
`is_connected()` (tornadis.PubSubClient method), 15

## P

Pipeline (class in tornadis), 17  
`preconnect()` (tornadis.ClientPool method), 18

`pubsub_pop_message()` (tornadis.PubSubClient method), 15  
`pubsub_psubscribe()` (tornadis.PubSubClient method), 15  
`pubsub_punsubscribe()` (tornadis.PubSubClient method), 16  
`pubsub_subscribe()` (tornadis.PubSubClient method), 16  
`pubsub_unsubscribe()` (tornadis.PubSubClient method), 16  
PubSubClient (class in tornadis), 15

## R

`release_client()` (tornadis.ClientPool method), 18

## S

`stack_call()` (tornadis.Pipeline method), 17

## T

tornadis (module), 13, 15, 17–19  
TornadoException (class in tornadis), 18