Datadog Python Client Documentation

Release

Datadog, Inc.

Aug 04, 2017

Contents

The *datadog* module provides `datadog.api` - a simple wrapper around Datadog's HTTP API - `datadog.threadstats` - a tool for collecting metrics in high performance applications - and `datadog.dogstatsd` a DogStatsd Python client.

CHAPTER 1

Installation

To install from source, download a distribution and run:

```
>>> sudo python setup.py install
```

If you use virtualenv you do not need to use sudo.

CHAPTER 2

Datadog.api module

Datadog.api is a Python client library for Datadog's HTTP API.

Datadog.api client requires to run `datadog` *initialize* method first.

`datadog.`**`initialize`**(*api_key=None*, *app_key=None*, *host_name=None*, *api_host=None*, *statsd_host=None*, *statsd_port=None*, *statsd_use_default_route=False*, ***kwargs*)

Initialize and configure Datadog.api and Datadog.statsd modules

Parameters

- **`api_key`** (*`string`*) – Datadog API key
- **`app_key`** (*`string`*) – Datadog application key
- **`proxies`** (*`dictionary mapping protocol to the URL of the proxy.`*) – Proxy to use to connect to Datadog API
- **`api_host`** (*`url`*) – Datadog API endpoint
- **`statsd_host`** (*`address`*) – Host of DogStatsd server or statsd daemon
- **`statsd_port`** (*`port`*) – Port of DogStatsd server or statsd daemon
- **`statsd_use_default_route`** – Dynamically set the statsd host to the default route

(Useful when running the client in a container) :type statsd_use_default_route: boolean

Parameters

- **`cacert`** (*`path or boolean`*) – Path to local certificate file used to verify SSL certificates. Can also be set to True (default) to use the systems certificate store, or False to skip SSL verification
- **`mute`** (*`boolean`*) – Mute any ApiError or ClientError before they escape from datadog.api.HTTPClient (default: True).

class `datadog.api.`**`Comment`**

A wrapper around Comment HTTP API.

create(*attach_host_name=False*, *method='POST'*, *id=None*, *params=None*, ***body*)
Create a new API resource object

Parameters

- **attach_host_name** (*bool*) – link the new resource object to the host name
- **method** (*HTTP method string*) – HTTP method to use to contact API endpoint
- **id** (*id*) – create a new resource object as a child of the given object
- **params** (*dictionary*) – new resource object source
- **body** (*dictionary*) – new resource object attributes

Returns Dictionary representing the API's JSON response

delete(*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

update(*id*, *params=None*, ***body*)
Update an API resource object

Parameters

- **params** (*dictionary*) – updated resource object source
- **body** (*dictionary*) – updated resource object attributes

Returns Dictionary representing the API's JSON response

class datadog.api.**Downtime**
A wrapper around Monitor Downtiming HTTP API.

create(*attach_host_name=False*, *method='POST'*, *id=None*, *params=None*, ***body*)
Create a new API resource object

Parameters

- **attach_host_name** (*bool*) – link the new resource object to the host name
- **method** (*HTTP method string*) – HTTP method to use to contact API endpoint
- **id** (*id*) – create a new resource object as a child of the given object
- **params** (*dictionary*) – new resource object source
- **body** (*dictionary*) – new resource object attributes

Returns Dictionary representing the API's JSON response

delete(*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

get(*id*, ***params*)
Get information about an API resource object

Parameters

- **id** (*id*) – resource object id to retrieve

- **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

get_all (***params*)
List API resource objects

Parameters **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

update (*id*, *params=None*, ***body*)
Update an API resource object

Parameters

- **params** (*dictionary*) – updated resource object source
- **body** (*dictionary*) – updated resource object attributes

Returns Dictionary representing the API's JSON response

class datadog.api.**Event**
A wrapper around Event HTTP API.

classmethod create (***params*)
Post an event.

Parameters

- **title** (*string*) – title for the new event
- **text** (*string*) – event message
- **aggregation_key** (*string*) – key by which to group events in event stream
- **alert_type** (*string*) – "error", "warning", "info" or "success".
- **date_happened** (*integer*) – when the event occurred. if unset defaults to the current time. (POSIX timestamp)
- **handle** (*string*) – user to post the event as. defaults to owner of the application key used to submit.
- **priority** (*string*) – priority to post the event as. ("normal" or "low", defaults to "normal")
- **related_event_id** (*id*) – post event as a child of the given event
- **tags** (*list of strings*) – tags to post the event with
- **host** (*string*) – host to post the event with
- **device_name** (*list of strings*) – device_name to post the event with

Returns Dictionary representing the API's JSON response

```
>>> title = "Something big happened!"
>>> text = 'And let me tell you all about it here!'
>>> tags = ['version:1', 'application:web']
```

```
>>> api.Event.create(title=title, text=text, tags=tags)
```

delete (*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

get (*id*, ***params*)
Get information about an API resource object

Parameters

- **id** (*id*) – resource object id to retrieve
- **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

classmethod query (***params*)
Get the events that occurred between the *start* and *end* POSIX timestamps, optional filtered by *priority* ("low" or "normal"), *sources* and *tags*.

See the event API documentation for the event data format.

Returns Dictionary representing the API's JSON response

```
>>> api.Event.query(start=1313769783, end=1419436870, priority="normal",
↪       tags=["application:web"])
```

class datadog.api.**Graph**
A wrapper around Graph HTTP API.

classmethod create (***params*)
Take a snapshot of a graph, returning the full url to the snapshot.

Parameters

- **metric_query** (*string query*) – metric query
- **start** (*POSIX timestamp*) – query start timestamp
- **end** (*POSIX timestamp*) – query end timestamp
- **event_query** (*string query*) – a query that will add event bands to the graph

Returns Dictionary representing the API's JSON response

classmethod status (*snapshot_url*)
Returns the status code of snapshot. Can be used to know when the snapshot is ready for download.

Parameters **snapshot_url** (*string url*) – snapshot URL to check

Returns Dictionary representing the API's JSON response

class datadog.api.**Host**
A wrapper around Host HTTP API.

classmethod mute (*host_name*, ***params*)
Mute a host.

Parameters

- **host_name** (*string*) – hostname
- **end** (*POSIX timestamp*) – timestamp to end muting
- **override** (*bool*) – if true and the host is already muted, will override existing end on the host
- **message** (*string*) – message to associate with the muting of this host

Returns Dictionary representing the API's JSON response

classmethod **unmute**(*host_name*)
Unmute a host.

Parameters **host_name** (*string*) – hostname

Returns Dictionary representing the API's JSON response

class datadog.api.**Infrastructure**
A wrapper around Infrastructure HTTP API.

classmethod **search**(***params*)
Search for entities in Datadog.

Parameters **q** (*string query*) – a query to serch for host and metrics

Returns Dictionary representing the API's JSON response

class datadog.api.**Metric**
A wrapper around Metric HTTP API

classmethod **query**(***params*)
Query metrics from Datadog

Parameters

- **start** (*POSIX timestamp*) – query start timestamp
- **end** (*POSIX timestamp*) – query end timestamp
- **query** (*string query*) – metric query

Returns Dictionary representing the API's JSON response

start and *end* should be less than 24 hours apart. It is *not* meant to retrieve metric data in bulk.

```
>>> api.Metric.query(start=int(time.time()) - 3600, end=int(time.time()),
                     query='avg:system.cpu.idle{*}')
```

classmethod **send**(*metrics=None*, ***single_metric*)
Submit a metric or a list of metrics to the metric API

Parameters

- **metric** (*string*) – the name of the time series
- **points** (*list*) – a (timestamp, value) pair or list of (timestamp, value) pairs
- **host** (*string*) – host name that produced the metric
- **tags** (*string list*) – list of tags associated with the metric.
- **type** (*'gauge' or 'counter' string*) – type of the metric

Returns Dictionary representing the API's JSON response

class datadog.api.**Monitor**
A wrapper around Monitor HTTP API.

create(*attach_host_name=False*, *method='POST'*, *id=None*, *params=None*, ***body*)
Create a new API resource object

Parameters

- **attach_host_name** (*bool*) – link the new resource object to the host name
- **method** (*HTTP method string*) – HTTP method to use to contact API endpoint
- **id** (*id*) – create a new resource object as a child of the given object

- **params** (*dictionary*) – new resource object source
- **body** (*dictionary*) – new resource object attributes

Returns Dictionary representing the API's JSON response

delete(*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

classmethod get(*id*, ***params*)
Get monitor's details.

Parameters

- **id** (*id*) – monitor to retrieve
- **group_states** (*string list, strings are chosen from one or more from 'all', 'alert', 'warn', or 'no data'*) – string list indicating what, if any, group states to include

Returns Dictionary representing the API's JSON response

classmethod get_all(***params*)
Get all monitor details.

Parameters

- **group_states** (*string list, strings are chosen from one or more from 'all', 'alert', 'warn', or 'no data'*) – string list indicating what, if any, group states to include
- **name** (*string*) – name to filter the list of monitors by
- **tags** (*string list*) – tags to filter the list of monitors by scope
- **monitor_tags** (*string list*) – list indicating what service and/or custom tags, if any, should be used to filter the list of monitors

Returns Dictionary representing the API's JSON response

classmethod mute(*id*, ***params*)
Mute a monitor.

Parameters

- **scope** (*string*) – scope to apply the mute
- **end** (*POSIX timestamp*) – timestamp for when the mute should end

Returns Dictionary representing the API's JSON response

classmethod mute_all()
Globally mute monitors.

Returns Dictionary representing the API's JSON response

classmethod unmute(*id*, ***params*)
Unmute a monitor.

Parameters

- **scope** (*string*) – scope to apply the unmute
- **all_scopes** (*boolean*) – if True, clears mute settings for all scopes

Returns Dictionary representing the API's JSON response

classmethod **unmute_all**()
Cancel global monitor mute setting (does not remove mute settings for individual monitors).

Returns Dictionary representing the API's JSON response

update(*id*, *params=None*, ***body*)
Update an API resource object

Parameters

- **params** (*dictionary*) – updated resource object source
- **body** (*dictionary*) – updated resource object attributes

Returns Dictionary representing the API's JSON response

class datadog.api.**Screenboard**
A wrapper around Screenboard HTTP API.

create(*attach_host_name=False*, *method='POST'*, *id=None*, *params=None*, ***body*)
Create a new API resource object

Parameters

- **attach_host_name** (*bool*) – link the new resource object to the host name
- **method** (*HTTP method string*) – HTTP method to use to contact API endpoint
- **id** (*id*) – create a new resource object as a child of the given object
- **params** (*dictionary*) – new resource object source
- **body** (*dictionary*) – new resource object attributes

Returns Dictionary representing the API's JSON response

delete(*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

get(*id*, ***params*)
Get information about an API resource object

Parameters

- **id** (*id*) – resource object id to retrieve
- **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

get_all(***params*)
List API resource objects

Parameters **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

classmethod **revoke**(*board_id*)
Revoke a shared screenboard with given id

Parameters **board_id** (*id*) – screenboard to revoke

Returns Dictionary representing the API's JSON response

classmethod **share**(*board_id*)
Share the screenboard with given id

Parameters **board_id** (*id*) – screenboard to share

Returns Dictionary representing the API's JSON response

update(*id*, *params=None*, ***body*)
Update an API resource object

Parameters

- **params** (*dictionary*) – updated resource object source
- **body** (*dictionary*) – updated resource object attributes

Returns Dictionary representing the API's JSON response

class datadog.api.**ServiceCheck**
A wrapper around ServiceCheck HTTP API.

classmethod **check**(***params*)
Post check statuses for use with monitors

Parameters

- **check** (*string*) – text for the message
- **host_name** (*string*) – name of the host submitting the check
- **status** (*Options: '0': OK, '1': WARNING, '2': CRITICAL, '3': UNKNOWN*) – integer for the status of the check
- **timestamp** (*POSIX timestamp*) – timestamp of the event
- **message** (*string*) – description of why this status occurred
- **tags** (*string list*) – list of tags for this check

Returns Dictionary representing the API's JSON response

class datadog.api.**Tag**
A wrapper around Tag HTTP API.

classmethod **create**(*host*, ***body*)
Add tags to a host

Parameters

- **tags** (*string list*) – list of tags to apply to the host
- **source** (*string*) – source of the tags

Returns Dictionary representing the API's JSON response

delete(*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

get(*id*, ***params*)
Get information about an API resource object

Parameters

- **id** (*id*) – resource object id to retrieve

- **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

get_all(***params*)
List API resource objects

Parameters **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

classmethod update(*host*, ***body*)
Update all tags for a given host

Parameters

- **tags** (*string list*) – list of tags to apply to the host
- **source** (*string*) – source of the tags

Returns Dictionary representing the API's JSON response

class datadog.api.**Timeboard**
A wrapper around Timeboard HTTP API.

create(*attach_host_name=False*, *method='POST'*, *id=None*, *params=None*, ***body*)
Create a new API resource object

Parameters

- **attach_host_name** (*bool*) – link the new resource object to the host name
- **method** (*HTTP method string*) – HTTP method to use to contact API endpoint
- **id** (*id*) – create a new resource object as a child of the given object
- **params** (*dictionary*) – new resource object source
- **body** (*dictionary*) – new resource object attributes

Returns Dictionary representing the API's JSON response

delete(*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

get(*id*, ***params*)
Get information about an API resource object

Parameters

- **id** (*id*) – resource object id to retrieve
- **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

get_all(***params*)
List API resource objects

Parameters **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

update(*id*, *params=None*, ***body*)
Update an API resource object

Parameters

- **params** (*dictionary*) – updated resource object source
- **body** (*dictionary*) – updated resource object attributes

Returns Dictionary representing the API's JSON response

class datadog.api.**User**

create(*attach_host_name=False*, *method='POST'*, *id=None*, *params=None*, ***body*)
Create a new API resource object

Parameters

- **attach_host_name** (*bool*) – link the new resource object to the host name
- **method** (*HTTP method string*) – HTTP method to use to contact API endpoint
- **id** (*id*) – create a new resource object as a child of the given object
- **params** (*dictionary*) – new resource object source
- **body** (*dictionary*) – new resource object attributes

Returns Dictionary representing the API's JSON response

delete(*id*, ***params*)
Delete an API resource object

Parameters **id** (*id*) – resource object to delete

Returns Dictionary representing the API's JSON response

get(*id*, ***params*)
Get information about an API resource object

Parameters

- **id** (*id*) – resource object id to retrieve
- **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

get_all(***params*)
List API resource objects

Parameters **params** (*dictionary*) – parameters to filter API resource stream

Returns Dictionary representing the API's JSON response

update(*id*, *params=None*, ***body*)
Update an API resource object

Parameters

- **params** (*dictionary*) – updated resource object source
- **body** (*dictionary*) – updated resource object attributes

Returns Dictionary representing the API's JSON response

CHAPTER 3

Datadog.threadstats module

Datadog.threadstats is a tool for collecting application metrics without hindering performance. It collects metrics in the application thread with very little overhead and allows flushing metrics in process, in a thread or in a greenlet, depending on your application's needs.

To run properly Datadog.threadstats requires to run *datadog initialize* method first.

datadog.**initialize**(*api_key=None*, *app_key=None*, *host_name=None*, *api_host=None*, *statsd_host=None*, *statsd_port=None*, *statsd_use_default_route=False*, ***kwargs*)

Initialize and configure Datadog.api and Datadog.statsd modules

Parameters

- **api_key** (*string*) – Datadog API key
- **app_key** (*string*) – Datadog application key
- **proxies** (*dictionary mapping protocol to the URL of the proxy.*) – Proxy to use to connect to Datadog API
- **api_host** (*url*) – Datadog API endpoint
- **statsd_host** (*address*) – Host of DogStatsd server or statsd daemon
- **statsd_port** (*port*) – Port of DogStatsd server or statsd daemon
- **statsd_use_default_route** – Dynamically set the statsd host to the default route

(Useful when running the client in a container) :type statsd_use_default_route: boolean

Parameters

- **cacert** (*path or boolean*) – Path to local certificate file used to verify SSL certificates. Can also be set to True (default) to use the systems certificate store, or False to skip SSL verification
- **mute** (*boolean*) – Mute any ApiError or ClientError before they escape from datadog.api.HTTPClient (default: True).

class datadog.threadstats.base.**ThreadStats**(*namespace=''*, *constant_tags=None*)

decrement(*metric_name*, *value=1*, *timestamp=None*, *tags=None*, *sample_rate=1*, *host=None*)
Decrement a counter, optionally setting a value, tags and a sample rate.

```
>>> stats.decrement('files.remaining')
>>> stats.decrement('active.connections', 2)
```

event(*title*, *text*, *alert_type=None*, *aggregation_key=None*, *source_type_name=None*, *date_happened=None*, *priority=None*, *tags=None*, *hostname=None*)
Send an event. Attributes are the same as the Event API. (http://docs.datadoghq.com/api/)

```
>>> stats.event('Man down!', 'This server needs assistance.')
>>> stats.event('The web server restarted', 'The web server is up␣
↪again', alert_type='success')
```

flush(*timestamp=None*)
Flush and post all metrics to the server. Note that this is a blocking call, so it is likely not suitable for user facing processes. In those cases, it's probably best to flush in a thread or greenlet.

gauge(*metric_name*, *value*, *timestamp=None*, *tags=None*, *sample_rate=1*, *host=None*)
Record the current `value` of a metric. The most recent value in a given flush interval will be recorded. Optionally, specify a set of tags to associate with the metric. This should be used for sum values such as total hard disk space, process uptime, total number of active users, or number of rows in a database table.

```
>>> stats.gauge('process.uptime', time.time() - process_start_time)
>>> stats.gauge('cache.bytes.free', cache.get_free_bytes(), tags=['version:1.0
↪'])
```

histogram(*metric_name*, *value*, *timestamp=None*, *tags=None*, *sample_rate=1*, *host=None*)
Sample a histogram value. Histograms will produce metrics that describe the distribution of the recorded values, namely the maximum, median, average, count and the 95th percentile. Optionally, specify a list of `tags` to associate with the metric.

```
>>> stats.histogram('uploaded_file.size', uploaded_file.size())
```

increment(*metric_name*, *value=1*, *timestamp=None*, *tags=None*, *sample_rate=1*, *host=None*)
Increment the counter by the given `value`. Optionally, specify a list of `tags` to associate with the metric. This is useful for counting things such as incrementing a counter each time a page is requested.

```
>>> stats.increment('home.page.hits')
>>> stats.increment('bytes.processed', file.size())
```

start(*flush_interval=10*, *roll_up_interval=10*, *device=None*, *flush_in_thread=True*, *flush_in_greenlet=False*, *disabled=False*)
Start the ThreadStats instance with the specified metric flushing method and preferences.

By default, metrics will be flushed in a thread.

```
>>> stats.start()
```

If you're running a gevent server and want to flush metrics in a greenlet, set *flush_in_greenlet* to True. Be sure to import and monkey patch gevent before starting ThreadStats.

```
>>> from gevent import monkey; monkey.patch_all()
>>> stats.start(flush_in_greenlet=True)
```

If you'd like to flush metrics in process, set *flush_in_thread* to False, though you'll have to call `flush` manually to post metrics to the server.

```
>>> stats.start(flush_in_thread=False)
```

If for whatever reason, you need to disable metrics collection in a hurry, set `disabled` to True and metrics won't be collected or flushed.

```
>>> stats.start(disabled=True)
```

Note: **Please remember to set your API key before,** using datadog module `initialize` method.

```
>>> from datadog import initialize, ThreadStats
>>> initialize(api_key='my_api_key')
>>> stats = ThreadStats()
>>> stats.start()
>>> stats.increment('home.page.hits')
```

Parameters

- **flush_interval** (*int*) – The number of seconds to wait between flushes.
- **flush_in_thread** (*bool*) – True if you'd like to spawn a thread to flush metrics. It will run every *flush_interval* seconds.
- **flush_in_greenlet** (*bool*) – Set to true if you'd like to flush in a gevent greenlet.
- **disabled** (*bool*) – Disable metrics collection

timed(*metric_name*, *sample_rate=1*, *tags=None*, *host=None*)

A decorator that will track the distribution of a function's run time. Optionally specify a list of tags to associate with the metric.

```
@stats.timed('user.query.time')
def get_user(user_id):
    # Do what you need to ...
    pass

# Is equivalent to ...
start = time.time()
try:
    get_user(user_id)
finally:
    stats.histogram('user.query.time', time.time() - start)
```

timer(**args*, ***kwds*)

A context manager that will track the distribution of the contained code's run time. Optionally specify a list of tags to associate with the metric.

```
def get_user(user_id):
    with stats.timer('user.query.time'):
        # Do what you need to ...
        pass

# Is equivalent to ...
def get_user(user_id):
    start = time.time()
```

```
    try:
        # Do what you need to ...
        pass
    finally:
        stats.histogram('user.query.time', time.time() - start)
```

timing(*metric_name*, *value*, *timestamp=None*, *tags=None*, *sample_rate=1*, *host=None*)

Record a timing, optionally setting tags and a sample rate.

```
>>> stats.timing("query.response.time", 1234)
```

CHAPTER 4

Datadog.dogstatsd module

class datadog.dogstatsd.base.**DogStatsd**(*host='localhost'*, *port=8125*, *max_buffer_size=50*, *namespace=None*, *constant_tags=None*, *use_ms=False*, *use_default_route=False*, *socket_path=None*)

close_buffer()
Flush the buffer and switch back to single metric packets.

close_socket()
Closes connected socket if connected.

decrement(*metric*, *value=1*, *tags=None*, *sample_rate=1*)
Decrement a counter, optionally setting a value, tags and a sample rate.

```
>>> statsd.decrement('files.remaining')
>>> statsd.decrement('active.connections', 2)
```

event(*title*, *text*, *alert_type=None*, *aggregation_key=None*, *source_type_name=None*, *date_happened=None*, *priority=None*, *tags=None*, *hostname=None*)

Send an event. Attributes are the same as the Event API. http://docs.datadoghq.com/api/

```
>>> statsd.event('Man down!', 'This server needs assistance.')
>>> statsd.event('The web server restarted', 'The web server is up again',␣
↪alert_type='success')  # NOQA
```

gauge(*metric*, *value*, *tags=None*, *sample_rate=1*)
Record the value of a gauge, optionally setting a list of tags and a sample rate.

```
>>> statsd.gauge('users.online', 123)
>>> statsd.gauge('active.connections', 1001, tags=["protocol:http"])
```

get_socket()
Return a connected socket.

Note: connect the socket before assigning it to the class instance to avoid bad thread race conditions.

histogram(*metric*, *value*, *tags=None*, *sample_rate=1*)
Sample a histogram value, optionally setting tags and a sample rate.

```
>>> statsd.histogram('uploaded.file.size', 1445)
>>> statsd.histogram('album.photo.count', 26, tags=["gender:female"])
```

increment(*metric*, *value=1*, *tags=None*, *sample_rate=1*)
Increment a counter, optionally setting a value, tags and a sample rate.

```
>>> statsd.increment('page.views')
>>> statsd.increment('files.transferred', 124)
```

open_buffer(*max_buffer_size=50*)
Open a buffer to send a batch of metrics in one packet.

You can also use this as a context manager.

```
>>> with DogStatsd() as batch:
>>>     batch.gauge('users.online', 123)
>>>     batch.gauge('active.connections', 1001)
```

static **resolve_host**(*host*, *use_default_route*)
Resolve the DogStatsd host.

Args: host (string): host use_default_route (bool): use the system default route as host

(overrides the *host* parameter)

service_check(*check_name*, *status*, *tags=None*, *timestamp=None*, *hostname=None*, *message=None*)
Send a service check run.

```
>>> statsd.service_check('my_service.check_name', DogStatsd.WARNING)
```

set(*metric*, *value*, *tags=None*, *sample_rate=1*)
Sample a set value.

```
>>> statsd.set('visitors.uniques', 999)
```

timed(*metric=None*, *tags=None*, *sample_rate=1*, *use_ms=None*)
A decorator or context manager that will measure the distribution of a function's/context's run time. Optionally specify a list of tags or a sample rate. If the metric is not defined as a decorator, the module name and function name will be used. The metric is required as a context manager.

```
@statsd.timed('user.query.time', sample_rate=0.5)
def get_user(user_id):
    # Do what you need to ...
    pass

# Is equivalent to ...
with statsd.timed('user.query.time', sample_rate=0.5):
    # Do what you need to ...
    pass

# Is equivalent to ...
start = time.time()
try:
    get_user(user_id)
```

```
finally:
    statsd.timing('user.query.time', time.time() - start)
```

timing(*metric*, *value*, *tags=None*, *sample_rate=1*)

Record a timing, optionally setting tags and a sample rate.

```
>>> statsd.timing("query.response.time", 1234)
```

datadog.**statsd**

A global *DogStatsd* instance that is easily shared across an application's modules. Initialize this once in your application's set-up code and then other modules can import and use it without further configuration.

```
>>> from datadog import initialize, statsd
>>> initialize(statsd_host='localhost', statsd_port=8125)
>>> statsd.increment('home.page.hits')
```

CHAPTER 5

Source

The Datadog's Python library source is freely available on Github. Check it out here.

CHAPTER 6

Get in Touch

If you'd like to suggest a feature or report a bug, please add an issue here. If you want to talk about Datadog in general, reach out at datadoghq.com.

Python Module Index

d

Index

M

O

Q

R

S

T

U