Datadog Python Client Documentation

Datadog, Inc.

Jan 24, 2023

Contents

The *datadog* module provides

- `datadog.api`: A client for Datadog's HTTP API.
- `datadog.dogstatsd`: A UDP/UDS DogStatsd client.
- `datadog.threadstats`: A client for Datadog's HTTP API that submits metrics in a worker thread.

CHAPTER 1

Installation

Install from PyPI:

```
pip install datadog
```

CHAPTER 2

Initialization

datadog must be initialized with *datadog.initialize()*. An API key and an app key are required unless you intend to use only the *DogStatsd* client. The keys can be passed explicitly to *datadog.initialize()* or defined as environment variables DATADOG_API_KEY and DATADOG_APP_KEY respectively.

Here's an example where the statsd host and port are configured as well:

```
from datadog import initialize

initialize(
    api_key="<your api key>",
    app_key="<your app key>",
    statsd_host="127.0.0.1",
    statsd_port=8125
)
```

datadog.**initialize**(*api_key=None*, *app_key=None*, *host_name=None*, *api_host=None*, *statsd_host=None*, *statsd_port=None*, *statsd_disable_buffering=True*, *statsd_use_default_route=False*, *statsd_socket_path=None*, *statsd_namespace=None*, *statsd_constant_tags=None*, *return_raw_response=False*, *hostname_from_config=True*, ***kwargs*)

Initialize and configure Datadog.api and Datadog.statsd modules

Parameters

- **api_key** (*string*) – Datadog API key
- **app_key** (*string*) – Datadog application key
- **host_name** (*string*) – Set a specific hostname
- **proxies** (*dictionary mapping protocol to the URL of the proxy.*) – Proxy to use to connect to Datadog API; for example, 'proxies': {'http': "http:<user>:<pass>@<ip>:<port>/"}
- **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_disable_buffering** (*boolean*) – Enable/disable statsd client buffering support (default: True).
- **statsd_use_default_route** (*boolean*) – Dynamically set the statsd host to the default route (Useful when running the client in a container)
- **statsd_socket_path** – path to the DogStatsd UNIX socket. Supersedes statsd_host and stats_port if provided.
- **statsd_constant_tags** (*list of string*) – A list of tags to be applied to all metrics ("tag", "tag:value")
- **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).
- **return_raw_response** (*boolean*) – Whether or not to return the raw response object in addition to the decoded response content (default: False)
- **hostname_from_config** (*boolean*) – Set the hostname from the Datadog agent config (agent 5). Will be deprecated

CHAPTER 3

datadog.api

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

3.1 Usage

Be sure to initialize the client using *`datadog.initialize()`* and then use `datadog.api`:

```
from datadog import api

api.Event.create(
    title="Something big happened!",
    text="And let me tell you all about it here!",
    tags=["version:1", "application:web"],
)
```

class datadog.api.**Comment**

A wrapper around Comment HTTP API.

classmethod 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

classmethod 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.

classmethod cancel_downtime_by_scope(***body*)
Cancels all downtimes matching the scope.

Parameters **scope** (*string*) – scope to cancel downtimes by

Returns Dictionary representing the API's JSON response

classmethod 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

classmethod 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 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 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(*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(*attach_host_name=True*, ***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)
```

classmethod 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*, ***body*)
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.**Hosts**
A wrapper around Hosts HTTP API.

classmethod search(***params*)
Search among hosts live within the past 2 hours. Max 100 results at a time.

Parameters

- **filter** (*string*) – query to filter search results
- **sort_field** (*string*) – "status", "apps", "cpu", "iowait", or "load"
- **sort_dir** (*string*) – "asc" or "desc"

- **start** (*integer*) – host result to start at
- **count** (*integer*) – number of host results to return

Returns Dictionary representing the API's JSOn response

classmethod totals()
Get total number of hosts active and up.

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 search for host and metrics

Returns Dictionary representing the API's JSON response

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

classmethod 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 list(*from_epoch*)
Get a list of active metrics since a given time (Unix Epoc)

Parameters **from_epoch** – Start time in Unix Epoc (seconds)

Returns Dictionary containing a list of active metrics

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*, *attach_host_name=True*, *compress_payload=False*, ***single_metric*)
Submit a metric or a list of metrics to the metric API A metric dictionary should consist of 5 keys: metric, points, host, tags, type (some of which optional), see below:

Parameters

- **metric** (*string*) – the name of the time series
- **compress_payload** (*bool*) – compress the payload using zlib

- **metrics** (*list*) – a list of dictionaries, each item being a metric to send
- **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 'count' or 'rate' string*) – type of the metric

```
>>> api.Metric.send(metric='my.series', points=[(now, 15), (future_10s, 16)])
```

```
>>> metrics = [{'metric': 'my.series', 'type': 'gauge', 'points': [(now, 15),␣
↪(future_10s, 16)]},
        {'metric': 'my.series2', 'type': 'gauge', 'points': [(now, 15),␣
↪(future_10s, 16)]}]
>>> api.Metric.send(metrics=metrics)
```

Returns Dictionary representing the API's JSON response

class datadog.api.**Monitor**

A wrapper around Monitor HTTP API.

classmethod can_delete(***params*)

Checks if the monitors corresponding to the monitor ids can be deleted.

Returns Dictionary representing the API's JSON response

classmethod 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

classmethod 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*, ***body*)
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 search(***params*)
Search monitors.

Returns Dictionary representing the API's JSON response

classmethod search_groups(***params*)
Search monitor groups.

Returns Dictionary representing the API's JSON response

classmethod unmute(*id*, ***body*)
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

classmethod 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

classmethod validate(***body*)
Checks if the monitors definition is valid.

Returns Dictionary representing the API's JSON response

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

classmethod 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

classmethod 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 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 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

classmethod 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(***body*)

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

classmethod 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 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 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.

classmethod 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

classmethod 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 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 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(*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**

classmethod 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

classmethod 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 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 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(*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.**Dashboard**
A wrapper around Dashboard HTTP API.

classmethod 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

classmethod 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 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 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(*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.**DashboardList**
A wrapper around Dashboard List HTTP API.

classmethod add_items(*id*, *params=None*, ***body*)
Add new API sub-resource objects to a resource

Parameters

- **id** (*id*) – resource id to add sub-resource objects to
- **params** (*dictionary*) – request parameters
- **body** (*dictionary*) – new sub-resource objects attributes

Returns Dictionary representing the API's JSON response

classmethod 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

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

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

Returns Dictionary representing the API's JSON response

classmethod delete_items(*id*, *params=None*, ***body*)
Delete API sub-resource objects from a resource

Parameters

- **id** (*id*) – resource id to delete sub-resource objects from
- **params** (*dictionary*) – request parameters
- **body** (*dictionary*) – deleted sub-resource objects attributes

Returns Dictionary representing the API's JSON response

classmethod 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 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 get_items(*id*, ***params*)
List API sub-resource objects from a resource

Parameters

- **id** (*id*) – resource id to retrieve sub-resource objects from
- **params** (*dictionary*) – parameters to filter API sub-resource stream

Returns Dictionary representing the API's JSON response

classmethod 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

classmethod update_items(*id*, *params=None*, ***body*)
Update API sub-resource objects of a resource

Parameters

- **id** (*id*) – resource id to update sub-resource objects from
- **params** (*dictionary*) – request parameters
- **body** (*dictionary*) – updated sub-resource objects attributes

Returns Dictionary representing the API's JSON response

CHAPTER 4

datadog.threadstats

`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. Submission is done through the HTTP API.

4.1 Usage

Be sure to initialize the library with *datadog.initialize()*. Then create an instance of `datadog.threadstats.ThreadStats`:

```
from datadog.threadstats import ThreadStats

statsd = ThreadStats()
statsd.start()  # Creates a worker thread used to submit metrics.

# Use statsd just like any other DatadogStatsd client.
statsd.increment("home.page.hits")
```

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

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)
```

distribution(*metric_name*, *value*, *timestamp=None*, *tags=None*, *sample_rate=1*, *host=None*)
Sample a distribution value. Distributions will produce metrics that describe the distribution of the recorded values, namely the maximum, median, average, count and the 50/75/90/95/99 percentiles. Optionally, specify a list of `tags` to associate with the metric.

```
>>> stats.distribution("uploaded_file.size", uploaded_file.size())
```

event (*title*, *message*, *alert_type=None*, *aggregation_key=None*, *source_type_name=None*, *date_happened=None*, *priority=None*, *tags=None*, *hostname=None*)
Send an event. See http://docs.datadoghq.com/api/ for more info.

```
>>> 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, minimum, average, count and the 75/85/95/99 percentiles. 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())
```

set (*metric_name*, *value*, *timestamp=None*, *tags=None*, *sample_rate=1*, *host=None*)
Add `value` to the current set. The length of the set is flushed as a gauge to Datadog. Optionally, specify a set of tags to associate with the metric.

```
>>> stats.set("example_metric.set", "value_1", tags=["environment:dev"])
```

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

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()
```

(continues on next page)

(continued from previous page)

```
    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 5

datadog.dogstatsd

`datadog.dogstatsd` is a Python client for DogStatsd that submits metrics to the Agent.

5.1 Usage

```
from datadog.dogstatsd import DogStatsd

client = DogStatsd()
client.increment("home.page.hits")
```

class datadog.dogstatsd.base.**DogStatsd**(*host='localhost'*, *port=8125*, *max_buffer_size=None*, *flush_interval=0.3*, *disable_buffering=True*, *namespace=None*, *constant_tags=None*, *use_ms=False*, *use_default_route=False*, *socket_path=None*, *default_sample_rate=1*, *disable_telemetry=False*, *telemetry_min_flush_interval=10*, *telemetry_host=None*, *telemetry_port=None*, *telemetry_socket_path=None*, *max_buffer_len=0*, *container_id=None*, *origin_detection_enabled=True*)

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

Note: This method must be called after a matching open_buffer() invocation.

close_socket()
Closes connected socket if connected.

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

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

distributed(*metric=None*, *tags=None*, *sample_rate=None*, *use_ms=None*)
A decorator or context manager that will measure the distribution of a function's/context's run time using custom metric distribution. 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.distributed("user.query.time", sample_rate=0.5)
def get_user(user_id):
    # Do what you need to ...
    pass

# Is equivalent to ...
with statsd.distributed("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.distribution("user.query.time", time.time() - start)
```

distribution(*metric*, *value*, *tags=None*, *sample_rate=None*)
Send a global distribution value, optionally setting tags and a sample rate.

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

event(*title*, *message*, *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("Web server restart", "The web server is up", alert_type=
↪"success")  # NOQA
```

flush()
Flush the metrics buffer by sending the data to the server.

gauge(*metric*, *value*, *tags=None*, *sample_rate=None*)
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(*telemetry=False*)
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=None*)
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=None*)
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=None*)
Open a buffer to send a batch of metrics.

To take advantage of automatic flushing, you should use the context manager instead

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

Note: This method must be called before close_buffer() matching invocation.

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

Parameters

- **host** (*string*) – host
- **use_default_route** (*bool*) – Use the system default route as host (overrides *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=None*)
Sample a set value.

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

timed(*metric=None*, *tags=None*, *sample_rate=None*, *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:
```

(continues on next page)

(continued from previous page)

```
    get_user(user_id)
finally:
    statsd.timing("user.query.time", time.time() - start)
```

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

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

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

datadog.**statsd**

A global *DogStatsd* instance that can be used across an application:

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

CHAPTER 6

Get in Touch

If you'd like to suggest a feature or report a bug, please submit an issue here. If you have questions about Datadog in general, reach out to support@datadoghq.com.

Python Module Index

d

Index

H

I

L

M

O

Q

R

S

T

U

V