influxdb$_{client}$

Release 1.10.0

Aug 14, 2020

Contents:

CHAPTER 1

User Guide

1.1 Query

```
from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import SYNCHRONOUS

bucket = "my-bucket"

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")

write_api = client.write_api(write_options=SYNCHRONOUS)
query_api = client.query_api()

p = Point("my_measurement").tag("location", "Prague").field("temperature", 25.3)

write_api.write(bucket=bucket, record=p)

## using Table structure
tables = query_api.query('from(bucket:"my-bucket") |> range(start: -10m)')

for table in tables:
    print(table)
    for row in table.records:
        print (row.values)


## using csv library
csv_result = query_api.query_csv('from(bucket:"my-bucket") |> range(start: -10m)')
val_count = 0
for row in csv_result:
    for cell in row:
        val_count += 1
```

1.2 Pandas DataFrame

Note: For DataFrame querying you should install Pandas dependency via `pip install influxdb-client[extra]`.

Note: Note that if a query returns more then one table then the client generates a `DataFrame` for each of them.

The `client` is able to retrieve data in Pandas DataFrame format thought `query_data_frame`:

```
from influxdb_client import InfluxDBClient, Point, Dialect
from influxdb_client.client.write_api import SYNCHRONOUS

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")

write_api = client.write_api(write_options=SYNCHRONOUS)
query_api = client.query_api()

"""
Prepare data
```

(continues on next page)

(continued from previous page)

```
"""

_point1 = Point("my_measurement").tag("location", "Prague").field("temperature", 25.3)
_point2 = Point("my_measurement").tag("location", "New York").field("temperature", 24.
↪3)

write_api.write(bucket="my-bucket", record=[_point1, _point2])

"""
Query: using Pandas DataFrame
"""
data_frame = query_api.query_data_frame('from(bucket:"my-bucket") '
                                        '|> range(start: -10m) '
                                        '|> pivot(rowKey:["_time"], columnKey: ["_
↪field"], valueColumn: "_value") '
                                        '|> keep(columns: ["location", "temperature"])
↪')
print(data_frame.to_string())

"""
Close client
"""
client.__del__()
```

Output:

1.3 Write

The WriteApi supports synchronous, asynchronous and batching writes into InfluxDB 2.0. The data should be passed as a InfluxDB Line Protocol, Data Point or Observable stream.

The default instance of WriteApi use batching.

1.3.1 The data could be written as

1. `string` or `bytes` that is formatted as a InfluxDB's line protocol
2. Data Point structure
3. Dictionary style mapping with keys: `measurement`, `tags`, `fields` and `time`
4. List of above items
5. A `batching` type of write also supports an `Observable` that produce one of an above item
6. Pandas DataFrame

1.3.2 Batching

The batching is configurable by `write_options`:

Property	Description	Default Value
batch_size	the number of data pointx to collect in a batch	`1000`
flush_interval	the number of milliseconds before the batch is written	`1000`
jitter_interval	the number of milliseconds to increase the batch flush interval by a random amount	`0`
retry_interval	the number of milliseconds to retry unsuccessful write. The retry interval is used when the InfluxDB server does not specify "Retry-After" header.	`5000`
max_retries	the number of max retries when write fails	`3`
max_retry_delay	the maximum delay between each retry attempt in milliseconds	`180_000`
exponential_base	the base for the exponential retry delay, the next delay is computed as `retry_interval * exponential_base^(attempts-1) + random(jitter_interval)`	`5`

```
import rx
from rx import operators as ops

from influxdb_client import InfluxDBClient, Point, WriteOptions
from influxdb_client.client.write_api import SYNCHRONOUS

_client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")
_write_client = _client.write_api(write_options=WriteOptions(batch_size=500,
                                                             flush_interval=10_000,
                                                             jitter_interval=2_000,
                                                             retry_interval=5_000,
                                                             max_retries=5,
                                                             max_retry_delay=30_000,
                                                             exponential_base=2))

"""
Write Line Protocol formatted as string
"""
_write_client.write("my-bucket", "my-org", "h2o_feet,location=coyote_creek water_
↪level=1.0 1")
_write_client.write("my-bucket", "my-org", ["h2o_feet,location=coyote_creek water_
↪level=2.0 2",
                                            "h2o_feet,location=coyote_creek water_
↪level=3.0 3"])

"""
Write Line Protocol formatted as byte array
"""
_write_client.write("my-bucket", "my-org", "h2o_feet,location=coyote_creek water_
↪level=1.0 1".encode())
_write_client.write("my-bucket", "my-org", ["h2o_feet,location=coyote_creek water_
↪level=2.0 2".encode(),
                                            "h2o_feet,location=coyote_creek water_
↪level=3.0 3".encode()])

"""
Write Dictionary-style object
"""
_write_client.write("my-bucket", "my-org", {"measurement": "h2o_feet", "tags": {
↪"location": "coyote_creek"},
```

(continues on next page)

(continued from previous page)

```
                                                "fields": {"water_level": 1.0}, "time": 1}
↪)
_write_client.write("my-bucket", "my-org", [{"measurement": "h2o_feet", "tags": {
↪"location": "coyote_creek"},
                                                "fields": {"water_level": 2.0}, "time": 2}
↪,
                                                {"measurement": "h2o_feet", "tags": {
↪"location": "coyote_creek"},
                                                "fields": {"water_level": 3.0}, "time": 3}
↪])

"""
Write Data Point
"""
_write_client.write("my-bucket", "my-org", Point("h2o_feet").tag("location", "coyote_
↪creek").field("water_level", 4.0).time(4))
_write_client.write("my-bucket", "my-org", [Point("h2o_feet").tag("location", "coyote_
↪creek").field("water_level", 5.0).time(5),
                                            Point("h2o_feet").tag("location", "coyote_
↪creek").field("water_level", 6.0).time(6)])

"""
Write Observable stream
"""
_data = rx \
    .range(7, 11) \
    .pipe(ops.map(lambda i: "h2o_feet,location=coyote_creek water_level={0}.0 {0}".
↪format(i)))

_write_client.write("my-bucket", "my-org", _data)

"""
Write Pandas DataFrame
"""
_now = pd.Timestamp().now('UTC')
_data_frame = pd.DataFrame(data=[["coyote_creek", 1.0], ["coyote_creek", 2.0]],
                           index=[now, now + timedelta(hours=1)],
                           columns=["location", "water_level"])

_write_client.write(bucket.name, record=data_frame, data_frame_measurement_name='h2o_
↪feet',
                    data_frame_tag_columns=['location'])

"""
Close client
"""
_write_client.__del__()
_client.__del__()
```

1.3.3 Default Tags

Sometimes is useful to store same information in every measurement e.g. `hostname`, `location`, `customer`. The client is able to use static value or env property as a tag value.

The expressions:

- `California Miner` - static value
- `${env.hostname}` - environment property

Via API

```
point_settings = PointSettings()
point_settings.add_default_tag("id", "132-987-655")
point_settings.add_default_tag("customer", "California Miner")
point_settings.add_default_tag("data_center", "${env.data_center}")

self.write_client = self.client.write_api(write_options=SYNCHRONOUS, point_
↪settings=point_settings)
```

```
self.write_client = self.client.write_api(write_options=SYNCHRONOUS,
                                          point_settings=PointSettings(**{"id":
↪"132-987-655",

↪"customer": "California Miner"}))
```

Via Configuration file

In a ini configuration file you are able to specify default tags by `tags` segment.

```
self.client = InfluxDBClient.from_config_file("config.ini")
```

Via Environment Properties

You are able to specify default tags by environment properties with prefix `INFLUXDB_V2_TAG_`.

Examples:

- `INFLUXDB_V2_TAG_ID`
- `INFLUXDB_V2_TAG_HOSTNAME`

```
self.client = InfluxDBClient.from_env_properties()
```

1.3.4 Asynchronous client

Data are writes in an asynchronous HTTP request.

```
from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import ASYNCHRONOUS

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")
write_api = client.write_api(write_options=ASYNCHRONOUS)

_point1 = Point("my_measurement").tag("location", "Prague").field("temperature", 25.3)
_point2 = Point("my_measurement").tag("location", "New York").field("temperature", 24.
↪3)

async_result = write_api.write(bucket="my-bucket", record=[_point1, _point2])
```

(continues on next page)

(continued from previous page)

```
async_result.get()

client.__del__()
```

1.3.5 Synchronous client

Data are writes in a synchronous HTTP request.

```
from influxdb_client import InfluxDBClient, Point
from influxdb_client .client.write_api import SYNCHRONOUS

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")
write_api = client.write_api(write_options=SYNCHRONOUS)

_point1 = Point("my_measurement").tag("location", "Prague").field("temperature", 25.3)
_point2 = Point("my_measurement").tag("location", "New York").field("temperature", 24.
↪3)

write_api.write(bucket="my-bucket", record=[_point1, _point2])

client.__del__()
```

1.4 Queries

The result retrieved by QueryApi could be formatted as a:

1. Flux data structure: FluxTable, FluxColumn and FluxRecord
2. csv.reader which will iterate over CSV lines
3. Raw unprocessed results as a `str` iterator
4. Pandas DataFrame

The API also support streaming `FluxRecord` via query_stream, see example below:

```
from influxdb_client import InfluxDBClient, Point, Dialect
from influxdb_client.client.write_api import SYNCHRONOUS

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")

write_api = client.write_api(write_options=SYNCHRONOUS)
query_api = client.query_api()

"""
Prepare data
"""

_point1 = Point("my_measurement").tag("location", "Prague").field("temperature", 25.3)
_point2 = Point("my_measurement").tag("location", "New York").field("temperature", 24.
↪3)

write_api.write(bucket="my-bucket", record=[_point1, _point2])
```

(continues on next page)

(continued from previous page)

```
"""
Query: using Table structure
"""
tables = query_api.query('from(bucket:"my-bucket") |> range(start: -10m)')

for table in tables:
    print(table)
    for record in table.records:
        print(record.values)

print()
print()

"""
Query: using Stream
"""
records = query_api.query_stream('from(bucket:"my-bucket") |> range(start: -10m)')

for record in records:
    print(f'Temperature in {record["location"]} is {record["_value"]}')

"""
Interrupt a stream after retrieve a required data
"""
large_stream = query_api.query_stream('from(bucket:"my-bucket") |> range(start: -100d)
↪')
for record in large_stream:
    if record["location"] == "New York":
        print(f'New York temperature: {record["_value"]}')
        break

large_stream.close()

print()
print()

"""
Query: using csv library
"""
csv_result = query_api.query_csv('from(bucket:"my-bucket") |> range(start: -10m)',
                                 dialect=Dialect(header=False, delimiter=",", comment_
↪prefix="#", annotations=[],
                                                 date_time_format="RFC3339"))
for csv_line in csv_result:
    if not len(csv_line) == 0:
        print(f'Temperature in {csv_line[9]} is {csv_line[6]}')

"""
Close client
"""
client.__del__()
```

1.4.1 Pandas DataFrame

Note: For DataFrame querying you should install Pandas dependency via `pip install influxdb-client[extra]`.

Note: Note that if a query returns more then one table then the client generates a `DataFrame` for each of them.

The `client` is able to retrieve data in Pandas DataFrame format thought `query_data_frame`:

```
from influxdb_client import InfluxDBClient, Point, Dialect
from influxdb_client.client.write_api import SYNCHRONOUS

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")

write_api = client.write_api(write_options=SYNCHRONOUS)
query_api = client.query_api()

"""
Prepare data
"""

_point1 = Point("my_measurement").tag("location", "Prague").field("temperature", 25.3)
_point2 = Point("my_measurement").tag("location", "New York").field("temperature", 24.
↪3)

write_api.write(bucket="my-bucket", record=[_point1, _point2])

"""
Query: using Pandas DataFrame
"""
data_frame = query_api.query_data_frame('from(bucket:"my-bucket") '
                                        '|> range(start: -10m) '
                                        '|> pivot(rowKey:["_time"], columnKey: ["_
↪field"], valueColumn: "_value") '
                                        '|> keep(columns: ["location", "temperature"])
↪')
print(data_frame.to_string())

"""
Close client
"""
client.__del__()
```

Output:

1.5 Examples

1.5.1 How to efficiently import large dataset

The following example shows how to import dataset with dozen megabytes. If you would like to import gigabytes of data then use our multiprocessing example: import_data_set_multiprocessing.py for use a full capability of your hardware.

- sources - import_data_set.py

```
"""
Import VIX - CBOE Volatility Index - from "vix-daily.csv" file into InfluxDB 2.0

https://datahub.io/core/finance-vix#data
"""

from collections import OrderedDict
from csv import DictReader

import rx
from rx import operators as ops

from influxdb_client import InfluxDBClient, Point, WriteOptions

def parse_row(row: OrderedDict):
    """Parse row of CSV file into Point with structure:

        financial-analysis,type=ily close=18.47,high=19.82,low=18.28,open=19.82␣
↪1198195200000000000

    CSV format:
        Date,VIX Open,VIX High,VIX Low,VIX Close\n
        2004-01-02,17.96,18.68,17.54,18.22\n
        2004-01-05,18.45,18.49,17.44,17.49\n
        2004-01-06,17.66,17.67,16.19,16.73\n
        2004-01-07,16.72,16.75,15.5,15.5\n
        2004-01-08,15.42,15.68,15.32,15.61\n
        2004-01-09,16.15,16.88,15.57,16.75\n
        ...

    :param row: the row of CSV file
    :return: Parsed csv row to [Point]
    """

    """
     For better performance is sometimes useful directly create a LineProtocol to␣
↪avoid unnecessary escaping overhead:
     """
     # from pytz import UTC
     # import ciso8601
     # from influxdb_client.client.write.point import EPOCH
     #
     # time = (UTC.localize(ciso8601.parse_datetime(row["Date"])) - EPOCH).total_
↪seconds() * 1e9
     # return f"financial-analysis,type=vix-daily" \
     #        f" close={float(row['VIX Close'])},high={float(row['VIX High'])},low=
↪{float(row['VIX Low'])},open={float(row['VIX Open'])} " \
     #        f" {int(time)}"

    return Point("financial-analysis") \
        .tag("type", "vix-daily") \
        .field("open", float(row['VIX Open'])) \
        .field("high", float(row['VIX High'])) \
        .field("low", float(row['VIX Low'])) \
        .field("close", float(row['VIX Close'])) \
        .time(row['Date'])
```

(continues on next page)

(continued from previous page)

```
"""
Converts vix-daily.csv into sequence of datad point
"""
data = rx \
    .from_iterable(DictReader(open('vix-daily.csv', 'r'))) \
    .pipe(ops.map(lambda row: parse_row(row)))

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org",␣
↪debug=True)

"""
Create client that writes data in batches with 50_000 items.
"""
write_api = client.write_api(write_options=WriteOptions(batch_size=50_000, flush_
↪interval=10_000))

"""
Write data into InfluxDB
"""
write_api.write(bucket="my-bucket", record=data)
write_api.__del__()

"""
Querying max value of CBOE Volatility Index
"""
query = 'from(bucket:"my-bucket")' \
        ' |> range(start: 0, stop: now())' \
        ' |> filter(fn: (r) => r._measurement == "financial-analysis")' \
        ' |> max()'
result = client.query_api().query(query=query)

"""
Processing results
"""
print()
print("=== results ===")
print()
for table in result:
    for record in table.records:
        print('max {0:5} = {1}'.format(record.get_field(), record.get_value()))

"""
Close client
"""
client.__del__()
```

1.6 Gzip support

`InfluxDBClient` does not enable gzip compression for http requests by default. If you want to enable gzip to reduce transfer data's size, you can call:

```
from influxdb_client import InfluxDBClient
```

(continues on next page)

(continued from previous page)

```
_db_client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org
↪", enable_gzip=True)
```

1.7 Debugging

For debug purpose you can enable verbose logging of http requests. Both request header and body will be logged to standard output.

```
_client = InfluxDBClient(url="http://localhost:9999", token="my-token", debug=True,␣
↪org="my-org")
```

CHAPTER 2

API Reference

2.1 InfluxDBClient

class influxdb_client.**InfluxDBClient**(*url*, *token*, *debug=None*, *timeout=10000*, *enable_gzip=False*, *org: str = None*, *default_tags: dict = None*, ***kwargs*)

InfluxDBClient is client for InfluxDB v2.

Initialize defaults.

Parameters

- **url** – InfluxDB server API url (ex. http://localhost:9999).
- **token** – auth token
- **debug** – enable verbose logging of http requests

- **timeout** – default http client timeout
- **enable_gzip** – Enable Gzip compression for http requests. Currently only the "Write" and "Query" endpoints supports the Gzip compression.
- **org** – organization name (used as a default in query and write API)

Key bool verify_ssl Set this to false to skip verifying SSL certificate when calling API from https server.

Key urllib3.util.retry.Retry retries Set the default retry strategy that is used for all HTTP requests except batching writes. As a default there is no one retry strategy.

authorizations_api() → influxdb_client.client.authorizations_api.AuthorizationsApi
Create the Authorizations API instance.

Returns authorizations api

buckets_api() → influxdb_client.client.bucket_api.BucketsApi
Create the Bucket API instance.

Returns buckets api

close()
Shutdown the client.

delete_api() → influxdb_client.client.delete_api.DeleteApi
Get the delete metrics API instance.

Returns delete api

classmethod from_config_file(*config_file: str = 'config.ini'*, *debug=None*, *enable_gzip=False*)
Configure client via '*.ini' file in segment 'influx2'.

Supported options:

- url
- org
- token
- timeout,
- verify_ssl

classmethod from_env_properties(*debug=None*, *enable_gzip=False*)
Configure client via environment properties.

Supported environment properties:

- INFLUXDB_V2_URL
- INFLUXDB_V2_ORG
- INFLUXDB_V2_TOKEN
- INFLUXDB_V2_TIMEOUT
- INFLUXDB_V2_VERIFY_SSL

health() → influxdb_client.domain.health_check.HealthCheck
Get the health of an instance.

Returns HealthCheck

labels_api() → influxdb_client.client.labels_api.LabelsApi
Create the Labels API instance.

Returns labels api

organizations_api() → influxdb_client.client.organizations_api.OrganizationsApi
Create the Organizations API instance.

Returns organizations api

query_api() → influxdb_client.client.query_api.QueryApi
Create a Query API instance.

Returns Query api instance

ready() → influxdb_client.domain.ready.Ready
Get The readiness of the InfluxDB 2.0.

Returns Ready

tasks_api() → influxdb_client.client.tasks_api.TasksApi
Create the Tasks API instance.

Returns tasks api

users_api() → influxdb_client.client.users_api.UsersApi
Create the Users API instance.

Returns users api

write_api(*write_options=<influxdb_client.client.write_api.WriteOptions object>*, *point_settings=<influxdb_client.client.write_api.PointSettings object>*) → influxdb_client.client.write_api.WriteApi
Create a Write API instance.

Parameters

- **point_settings** –
- **write_options** – write api configuration

Returns write api instance

2.2 QueryApi

class influxdb_client.**QueryApi**(*influxdb_client*)
Implementation for '/api/v2/query' endpoint.

Initialize query client.

Parameters **influxdb_client** – influxdb client

query(*query: str*, *org=None*) → List[influxdb_client.client.flux_table.FluxTable]
Execute synchronous Flux query and return result as a List['FluxTable'].

Parameters

- **query** – the Flux query
- **org** – organization name (optional if already specified in InfluxDBClient)

Returns

query_csv(*query: str, org=None, dialect: influxdb_client.domain.dialect.Dialect = {'annotations': ['datatype', 'group', 'default'], 'comment_prefix': '#', 'date_time_format': 'RFC3339', 'delimiter': ',', 'header': True}*)
Execute the Flux query and return results as a CSV iterator. Each iteration returns a row of the CSV file.

Parameters

- **query** – a Flux query
- **org** – organization name (optional if already specified in InfluxDBClient)
- **dialect** – csv dialect format

Returns The returned object is an iterator. Each iteration returns a row of the CSV file (which can span multiple input lines).

query_data_frame(*query: str*, *org=None*, *data_frame_index: List[str] = None*)
Execute synchronous Flux query and return Pandas DataFrame.

Note that if a query returns more then one table than the client generates a DataFrame for each of them.

Parameters

- **query** – the Flux query
- **org** – organization name (optional if already specified in InfluxDBClient)
- **data_frame_index** – the list of columns that are used as DataFrame index

Returns

query_data_frame_stream(*query: str*, *org=None*, *data_frame_index: List[str] = None*)
Execute synchronous Flux query and return stream of Pandas DataFrame as a Generator['pd.DataFrame'].

Note that if a query returns more then one table than the client generates a DataFrame for each of them.

Parameters

- **query** – the Flux query
- **org** – organization name (optional if already specified in InfluxDBClient)
- **data_frame_index** – the list of columns that are used as DataFrame index

Returns

query_raw(*query: str, org=None, dialect={'annotations': ['datatype', 'group', 'default'], 'comment_prefix': '#', 'date_time_format': 'RFC3339', 'delimiter': ',', 'header': True}*)
Execute synchronous Flux query and return result as raw unprocessed result as a str.

Parameters

- **query** – a Flux query
- **org** – organization name (optional if already specified in InfluxDBClient)
- **dialect** – csv dialect format

Returns str

query_stream(*query: str*, *org=None*) → Generator[[influxdb_client.client.flux_table.FluxRecord, Any], None]
Execute synchronous Flux query and return stream of FluxRecord as a Generator['FluxRecord'].

Parameters

- **query** – the Flux query
- **org** – organization name (optional if already specified in InfluxDBClient)

Returns

2.3 WriteApi

class influxdb_client.**WriteApi**(*influxdb_client*, *write_options: influxdb_client.client.write_api.WriteOptions = <influxdb_client.client.write_api.WriteOptions object>*, *point_settings: influxdb_client.client.write_api.PointSettings = <influxdb_client.client.write_api.PointSettings object>*)

Implementation for '/api/v2/write' endpoint.

Initialize defaults.

flush()

Flush data.

write(*bucket: str, org: str = None, record: Union[str, List[str], influxdb_client.client.write.point.Point, List[Point], dict, List[dict], bytes, List[bytes], rx.core.observable.observable.Observable] = None, write_precision: influxdb_client.domain.write_precision.WritePrecision = 'ns', **kwargs*) → Any

Write time-series data into InfluxDB.

Parameters

- **org** (*str*) – specifies the destination organization for writes; take either the ID or Name interchangeably; if both orgID and org are specified, org takes precedence. (required)
- **bucket** (*str*) – specifies the destination bucket for writes (required)
- **write_precision** (*WritePrecision*) – specifies the precision for the unix timestamps within the body line-protocol. The precision specified on a Point has precedes and is use for write.
- **record** – Points, line protocol, Pandas DataFrame, RxPY Observable to write

Key data_frame_measurement_name name of measurement for writing Pandas DataFrame

Key data_frame_tag_columns list of DataFrame columns which are tags, rest columns will be fields

2.4 BucketsApi

class influxdb_client.**BucketsApi**(*influxdb_client*)

Implementation for '/api/v2/buckets' endpoint.

Initialize defaults.

create_bucket(*bucket=None*, *bucket_name=None*, *org_id=None*, *retention_rules=None*, *description=None*) → influxdb_client.domain.bucket.Bucket

Create a bucket.

Parameters

- **bucket** (Bucket) – bucket to create (required)
- **bucket_name** – bucket name
- **description** – bucket description
- **org_id** – org_id

- **bucket_name** – bucket name
- **retention_rules** – retention rules array or single BucketRetentionRules

Returns Bucket If the method is called asynchronously, returns the request thread.

delete_bucket(*bucket*)
Delete a bucket.

Parameters **bucket** – bucket id or Bucket

Returns Bucket If the method is called asynchronously, returns the request thread.

find_bucket_by_id(*id*)
Find bucket by ID.

Parameters **id** –

Returns

find_bucket_by_name(*bucket_name*)
Find bucket by name.

Parameters **bucket_name** – bucket name

Returns Bucket

find_buckets()
Get all buckets.

class influxdb_client.domain.**Bucket**(*links=None*, *id=None*, *type='user'*, *name=None*, *description=None*, *org_id=None*, *rp=None*, *created_at=None*, *updated_at=None*, *retention_rules=None*, *labels=None*)
NOTE: This class is auto generated by OpenAPI Generator.

Ref: https://openapi-generator.tech

Do not edit the class manually.

Bucket - a model defined in OpenAPI.

created_at
Get the created_at of this Bucket.

Returns The created_at of this Bucket.

Return type datetime

description
Get the description of this Bucket.

Returns The description of this Bucket.

Return type str

id
Get the id of this Bucket.

Returns The id of this Bucket.

Return type str

labels
Get the labels of this Bucket.

Returns The labels of this Bucket.

Return type list[Label]

links
Get the links of this Bucket.

Returns The links of this Bucket.

Return type BucketLinks

name
Get the name of this Bucket.

Returns The name of this Bucket.

Return type str

org_id
Get the org_id of this Bucket.

Returns The org_id of this Bucket.

Return type str

retention_rules
Get the retention_rules of this Bucket.

Rules to expire or retain data. No rules means data never expires.

Returns The retention_rules of this Bucket.

Return type list[BucketRetentionRules]

rp
Get the rp of this Bucket.

Returns The rp of this Bucket.

Return type str

to_dict()
Return the model properties as a dict.

to_str()
Return the string representation of the model.

type
Get the type of this Bucket.

Returns The type of this Bucket.

Return type str

updated_at
Get the updated_at of this Bucket.

Returns The updated_at of this Bucket.

Return type datetime

2.5 LabelsApi

class influxdb_client.**LabelsApi**(*influxdb_client*)
Implementation for '/api/v2/labels' endpoint.

Initialize defaults.

clone_label(*cloned_name: str*, *label: influxdb_client.domain.label.Label*) → influxdb_client.domain.label.Label
Create the new instance of the label as a copy existing label.

Parameters

- **cloned_name** – new label name
- **label** – existing label

Returns clonned Label

create_label(*name: str*, *org_id: str*, *properties: Dict[str, str] = None*) → influxdb_client.domain.label.Label
Create a new label.

Parameters

- **name** – label name
- **org_id** – organization id
- **properties** – optional label properties

Returns created label

delete_label(*label: Union[str, influxdb_client.domain.label.Label]*)
Delete the label.

Parameters **label** – label id or Label

find_label_by_id(*label_id: str*)
Retrieve the label by id.

Parameters **label_id** –

Returns Label

find_label_by_org(*org_id*) → List[influxdb_client.domain.label.Label]
Get the list of all labels for given organization.

Parameters **org_id** – organization id

Returns list of labels

find_labels() → List[influxdb_client.domain.label.Label]
Get all available labels.

Returns labels

update_label(*label: influxdb_client.domain.label.Label*)
Update an existing label name and properties.

Parameters **label** – label

Returns the updated label

2.6 OrganizationsApi

class influxdb_client.**OrganizationsApi**(*influxdb_client*)
Implementation for '/api/v2/orgs' endpoint.

Initialize defaults.

create_organization(*name: str = None*, *organization: influxdb_client.domain.organization.Organization = None*) → influxdb_client.domain.organization.Organization
Create an organization.

delete_organization(*org_id: str*)
Delete an organization.

find_organization(*org_id*)
Retrieve an organization.

find_organizations()
List all organizations.

me()
Return the current authenticated user.

class influxdb_client.domain.**Organization**(*links=None*, *id=None*, *name=None*, *description=None*, *created_at=None*, *updated_at=None*, *status='active'*)
NOTE: This class is auto generated by OpenAPI Generator.

Ref: https://openapi-generator.tech

Do not edit the class manually.

Organization - a model defined in OpenAPI.

created_at
Get the created_at of this Organization.

Returns The created_at of this Organization.

Return type datetime

description
Get the description of this Organization.

Returns The description of this Organization.

Return type str

id
Get the id of this Organization.

Returns The id of this Organization.

Return type str

links
Get the links of this Organization.

Returns The links of this Organization.

Return type OrganizationLinks

name
Get the name of this Organization.

Returns The name of this Organization.

Return type str

status
Get the status of this Organization.

If inactive the organization is inactive.

Returns The status of this Organization.

Return type str

to_dict()
Return the model properties as a dict.

to_str()
Return the string representation of the model.

updated_at
Get the updated_at of this Organization.

Returns The updated_at of this Organization.

Return type datetime

2.7 UsersApi

class influxdb_client.**UsersApi**(*influxdb_client*)
Implementation for '/api/v2/users' endpoint.

Initialize defaults.

create_user(*name: str*) → influxdb_client.domain.user.User
Create a user.

me() → influxdb_client.domain.user.User
Return the current authenticated user.

class influxdb_client.domain.**User**(*id=None*, *oauth_id=None*, *name=None*, *status='active'*, *links=None*)
NOTE: This class is auto generated by OpenAPI Generator.

Ref: https://openapi-generator.tech

Do not edit the class manually.

User - a model defined in OpenAPI.

id
Get the id of this User.

Returns The id of this User.

Return type str

links
Get the links of this User.

Returns The links of this User.

Return type UserLinks

name
Get the name of this User.

Returns The name of this User.

Return type str

oauth_id
Get the oauth_id of this User.

Returns The oauth_id of this User.

Return type str

status
Get the status of this User.

If inactive the user is inactive.

Returns The status of this User.

Return type str

to_dict()
Return the model properties as a dict.

to_str()
Return the string representation of the model.

2.8 TasksApi

class influxdb_client.**TasksApi**(*influxdb_client*)
Implementation for '/api/v2/tasks' endpoint.

Initialize defaults.

add_label(*label_id: str*, *task_id: str*) → influxdb_client.domain.label_response.LabelResponse
Add a label to a task.

add_member(*member_id*, *task_id*)
Add a member to a task.

add_owner(*owner_id*, *task_id*)
Add an owner to a task.

cancel_run(*task_id: str*, *run_id: str*)
Cancel a currently running run.

Parameters

- **task_id** –
- **run_id** –

clone_task(*task: influxdb_client.domain.task.Task*) → influxdb_client.domain.task.Task
Clone a task.

create_task(*task: influxdb_client.domain.task.Task = None*, *task_create_request: influxdb_client.domain.task_create_request.TaskCreateRequest = None*) → influxdb_client.domain.task.Task
Create a new task.

create_task_cron(*name: str*, *flux: str*, *cron: str*, *org_id: str*) → influxdb_client.domain.task.Task
Create a new task with cron repetition schedule.

create_task_every(*name*, *flux*, *every*, *organization*) → influxdb_client.domain.task.Task
Create a new task with every repetition schedule.

delete_label(*label_id: str*, *task_id: str*)
Delete a label from a task.

delete_member(*member_id*, *task_id*)
Remove a member from a task.

delete_owner(*owner_id*, *task_id*)
Remove an owner from a task.

delete_task(*task_id: str*)
Delete a task.

find_task_by_id(*task_id*) → influxdb_client.domain.task.Task
Retrieve a task.

find_tasks(***kwargs*)
List all tasks.

Parameters

- **name** (*str*) – only returns tasks with the specified name
- **after** (*str*) – returns tasks after specified ID
- **user** (*str*) – filter tasks to a specific user ID
- **org** (*str*) – filter tasks to a specific organization name
- **org_id** (*str*) – filter tasks to a specific organization ID
- **limit** (*int*) – the number of tasks to return

Returns Tasks

find_tasks_by_user(*task_user_id*)
List all tasks by user.

get_labels(*task_id*)
List all labels for a task.

get_logs(*task_id: str*) → List[influxdb_client.domain.log_event.LogEvent]
Retrieve all logs for a task.

Parameters **task_id** – task id

get_members(*task_id: str*)
List all task members.

get_owners(*task_id*)
List all owners of a task.

get_run(*task_id: str*, *run_id: str*) → influxdb_client.domain.run.Run
Get run record for specific task and run id.

Parameters

- **task_id** – task id
- **run_id** – run id

Returns Run for specified task and run id

get_run_logs(*task_id: str*, *run_id: str*) → List[influxdb_client.domain.log_event.LogEvent]
Retrieve all logs for a run.

get_runs(*task_id*, ***kwargs*) → List[influxdb_client.domain.run.Run]
Retrieve list of run records for a task.

Parameters

- **task_id** – task id
- **after** (*str*) – returns runs after specified ID

- **limit** (*int*) – the number of runs to return
- **after_time** (*datetime*) – filter runs to those scheduled after this time, RFC3339
- **before_time** (*datetime*) – filter runs to those scheduled before this time, RFC3339

retry_run(*task_id: str*, *run_id: str*)
Retry a task run.

Parameters

- **task_id** – task id
- **run_id** – run id

run_manually(*task_id: str*, *scheduled_for: <module 'datetime' from '/home/docs/.pyenv/versions/3.6.8/lib/python3.6/datetime.py'> = None*)
Manually start a run of the task now overriding the current schedule.

Parameters

- **task_id** –
- **scheduled_for** – planned execution

update_task(*task: influxdb_client.domain.task.Task*) → influxdb_client.domain.task.Task
Update a task.

update_task_request(*task_id*, *task_update_request: influxdb_client.domain.task_update_request.TaskUpdateRequest*) → influxdb_client.domain.task.Task
Update a task.

class influxdb_client.domain.**Task**(*id=None*, *type=None*, *org_id=None*, *org=None*, *name=None*, *description=None*, *status=None*, *labels=None*, *authorization_id=None*, *flux=None*, *every=None*, *cron=None*, *offset=None*, *latest_completed=None*, *last_run_status=None*, *last_run_error=None*, *created_at=None*, *updated_at=None*, *links=None*)
NOTE: This class is auto generated by OpenAPI Generator.

Ref: https://openapi-generator.tech

Do not edit the class manually.

Task - a model defined in OpenAPI.

authorization_id
Get the authorization_id of this Task.

The ID of the authorization used when this task communicates with the query engine.

Returns The authorization_id of this Task.

Return type str

created_at
Get the created_at of this Task.

Returns The created_at of this Task.

Return type datetime

cron
Get the cron of this Task.

A task repetition schedule in the form '* * * * * *'; parsed from Flux.

Returns The cron of this Task.

Return type str

description
Get the description of this Task.

An optional description of the task.

Returns The description of this Task.

Return type str

every
Get the every of this Task.

A simple task repetition schedule; parsed from Flux.

Returns The every of this Task.

Return type str

flux
Get the flux of this Task.

The Flux script to run for this task.

Returns The flux of this Task.

Return type str

id
Get the id of this Task.

Returns The id of this Task.

Return type str

labels
Get the labels of this Task.

Returns The labels of this Task.

Return type list[Label]

last_run_error
Get the last_run_error of this Task.

Returns The last_run_error of this Task.

Return type str

last_run_status
Get the last_run_status of this Task.

Returns The last_run_status of this Task.

Return type str

latest_completed
Get the latest_completed of this Task.

Timestamp of latest scheduled, completed run, RFC3339.

Returns The latest_completed of this Task.

Return type datetime

links
Get the links of this Task.

Returns The links of this Task.

Return type TaskLinks

name
Get the name of this Task.

The name of the task.

Returns The name of this Task.

Return type str

offset
Get the offset of this Task.

Duration to delay after the schedule, before executing the task; parsed from flux, if set to zero it will remove this option and use 0 as the default.

Returns The offset of this Task.

Return type str

org
Get the org of this Task.

The name of the organization that owns this Task.

Returns The org of this Task.

Return type str

org_id
Get the org_id of this Task.

The ID of the organization that owns this Task.

Returns The org_id of this Task.

Return type str

status
Get the status of this Task.

Returns The status of this Task.

Return type TaskStatusType

to_dict()
Return the model properties as a dict.

to_str()
Return the string representation of the model.

type
Get the type of this Task.

The type of task, this can be used for filtering tasks on list actions.

Returns The type of this Task.

Return type str

updated_at
Get the updated_at of this Task.

Returns The updated_at of this Task.

Return type datetime

2.9 DeleteApi

InfluxDB 2.0 python client library.

Note: Use this client library with InfluxDB 2.x and InfluxDB 1.8+. For connecting to InfluxDB 1.7 or earlier instances, use the influxdb-python **client library.**

CHAPTER 3

InfluxDB 2.0 client features

- **Querying data**
  - using the Flux language
  - into csv, raw data, flux_table structure, Pandas DataFrame
  - *How to queries*
- **Writing data using**
  - Line Protocol
  - Data Point
  - RxPY Observable
  - Pandas DataFrame
  - *How to writes*
- **InfluxDB 2.0 API client for management**
  - the client is generated from the swagger by using the openapi-generator
  - organizations & users management
  - buckets management
  - tasks management
  - authorizations
  - health check
  - ...
- **'InfluxDB 1.8 API compatibility'_**
- **Examples**
  - **'Connect to InfluxDB Cloud'_**
  - **'How to efficiently import large dataset'_**

- **'Efficiency write data from IOT sensor'_**
- **'How to use Jupyter + Pandas + InfluxDB 2'_**

CHAPTER 4

Installation

InfluxDB python library uses RxPY - The Reactive Extensions for Python (RxPY).

Python 3.6 or later is required.

Note: It is recommended to use `ciso8601` with client for parsing dates. `ciso8601` is much faster than built-in Python datetime. Since it's written as a `C` module the best way is build it from sources:

Windows:

You have to install Visual C++ Build Tools 2015 to build `ciso8601` by `pip`.

conda:

Install from sources: `conda install -c conda-forge/label/cf202003 ciso8601`.

4.1 pip install

The python package is hosted on PyPI, you can install latest version directly:

```
pip install influxdb-client[ciso]
```

Then import the package:

```
import influxdb_client
```

4.2 Setuptools

Install via Setuptools.

```
python setup.py install --user
```

(or `sudo python setup.py install` to install the package for all users)

CHAPTER 5

Getting Started

Please follow the *Installation* and then run the following:

```
from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import SYNCHRONOUS

bucket = "my-bucket"

client = InfluxDBClient(url="http://localhost:9999", token="my-token", org="my-org")

write_api = client.write_api(write_options=SYNCHRONOUS)
query_api = client.query_api()

p = Point("my_measurement").tag("location", "Prague").field("temperature", 25.3)

write_api.write(bucket=bucket, record=p)

## using Table structure
tables = query_api.query('from(bucket:"my-bucket") |> range(start: -10m)')

for table in tables:
    print(table)
    for row in table.records:
        print (row.values)


## using csv library
csv_result = query_api.query_csv('from(bucket:"my-bucket") |> range(start: -10m)')
val_count = 0
for row in csv_result:
    for cell in row:
        val_count += 1
```

CHAPTER 6

Client configuration

6.1 Via File

A client can be configured via `*.ini` file in segment `influx2`.

The following options are supported:

- `url` - the url to connect to InfluxDB
- `org` - default destination organization for writes and queries
- `token` - the token to use for the authorization
- `timeout` - socket timeout in ms (default value is 10000)
- `verify_ssl` - set this to false to skip verifying SSL certificate when calling API from https server

```
self.client = InfluxDBClient.from_config_file("config.ini")
```

6.2 Via Environment Properties

A client can be configured via environment properties.

Supported properties are:

- `INFLUXDB_V2_URL` - the url to connect to InfluxDB
- `INFLUXDB_V2_ORG` - default destination organization for writes and queries
- `INFLUXDB_V2_TOKEN` - the token to use for the authorization
- `INFLUXDB_V2_TIMEOUT` - socket timeout in ms (default value is 10000)
- `INFLUXDB_V2_VERIFY_SSL` - set this to false to skip verifying SSL certificate when calling API from https server

```
self.client = InfluxDBClient.from_env_properties()
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Index

G

H

I

L

M

N

O

Q

R

S

T

U

W