

---

**influxdb***client*

***Release 1.9.0***

**Jul 17, 2020**



---

## Contents:

---

<b>1</b>	<b>User Guide</b>	<b>1</b>
1.1	Query . . . . .	2
1.2	Pandas DataFrame . . . . .	2
1.3	Write . . . . .	3
1.3.1	The data could be written as . . . . .	3
1.3.2	Batching . . . . .	3
1.3.3	Default Tags . . . . .	5
1.3.4	Asynchronous client . . . . .	6
1.3.5	Synchronous client . . . . .	6
1.4	Queries . . . . .	7
1.4.1	Pandas DataFrame . . . . .	8
1.5	Examples . . . . .	9
1.5.1	How to efficiently import large dataset . . . . .	9
1.6	Gzip support . . . . .	11
1.7	Debugging . . . . .	11
<b>2</b>	<b>API Reference</b>	<b>13</b>
2.1	InfluxDBClient . . . . .	13
2.2	QueryApi . . . . .	15
2.3	WriteApi . . . . .	16
2.4	BucketsApi . . . . .	17
2.5	LabelsApi . . . . .	19
2.6	OrganizationsApi . . . . .	20
2.7	UsersApi . . . . .	21
2.8	TasksApi . . . . .	22
2.9	DeleteApi . . . . .	27
<b>3</b>	<b>InfluxDB 2.0 client features</b>	<b>29</b>
<b>4</b>	<b>Installation</b>	<b>31</b>
4.1	pip install . . . . .	31
4.2	Setuptools . . . . .	31
<b>5</b>	<b>Getting Started</b>	<b>33</b>
<b>6</b>	<b>Client configuration</b>	<b>35</b>
6.1	Via File . . . . .	35

6.2 Via Environment Properties . . . . .	35
<b>7 Indices and tables</b>	<b>37</b>
<b>Index</b>	<b>39</b>

- *Query*
- *Pandas DataFrame*
- *Write*
  - *The data could be written as*
  - *Batching*
  - *Default Tags*
    - \* *Via API*
    - \* *Via Configuration file*
    - \* *Via Environment Properties*
  - *Asynchronous client*
  - *Synchronous client*
- *Queries*
  - *Pandas DataFrame*
- *Examples*
  - *How to efficiently import large dataset*
- *Gzip support*
- *Debugging*

## 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 than one table then the client generates a DataFrame for each of them.

---

The client is able to retrieve data in Pandas DataFrame format through `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
<b>batch_size</b>	the number of data points to collect in a batch	1000
<b>flush_interval</b>	the number of milliseconds before the batch is written	1000
<b>jitter_interval</b>	the number of milliseconds to increase the batch flush interval by a random amount	0
<b>retry_interval</b>	the number of milliseconds to retry unsuccessful write. The retry interval is used when the InfluxDB server does not specify "Retry-After" header.	1000

```

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))

"""
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"},
                                           "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}
↪])

```

(continues on next page)



(continued from previous page)

```

"""
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")

```

(continues on next page)

(continued from previous page)

```
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])
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])

"""
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()

```

(continues on next page)

(continued from previous page)

```

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 than one table then the client generates a DataFrame for each of them.

The client is able to retrieve data in [Pandas DataFrame](#) format through `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

```

(continues on next page)

(continued from previous page)

```

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_
↪11981952000000000000

    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'])

    """
    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)

```

(continues on next page)

(continued from previous page)

```

"""
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

_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")
```



- *InfluxDBClient*
- *QueryApi*
- *WriteApi*
- *BucketsApi*
- *LabelsApi*
- *OrganizationsApi*
- *UsersApi*
- *TasksApi*
- *DeleteApi*

## 2.1 InfluxDBClient

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

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)

**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'.

**classmethod from\_env\_properties** (debug=None, enable\_gzip=False)  
Configure client via environment properties.

**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 than one table then 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 than one table then 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 precedence and is used for write.
- **record** – Points, line protocol, Pandas DataFrame, RxPY Observable to write
- **data\_frame\_measurement\_name** – name of measurement for writing Pandas DataFrame
- **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** cloned 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, last_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

**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.



---

### 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
```



### 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)

```
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)

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





## CHAPTER 7

---

### Indices and tables

---

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



## A

add\_label() (*influxdb\_client.TasksApi method*), 22  
 add\_member() (*influxdb\_client.TasksApi method*), 23  
 add\_owner() (*influxdb\_client.TasksApi method*), 23  
 authorization\_id (*influxdb\_client.domain.Task attribute*), 25  
 authorizations\_api() (*influxdb\_client.InfluxDBClient method*), 14

## B

Bucket (*class in influxdb\_client.domain*), 18  
 buckets\_api() (*influxdb\_client.InfluxDBClient method*), 14  
 BucketsApi (*class in influxdb\_client*), 17

## C

cancel\_run() (*influxdb\_client.TasksApi method*), 23  
 clone\_label() (*influxdb\_client.LabelsApi method*), 19  
 clone\_task() (*influxdb\_client.TasksApi method*), 23  
 close() (*influxdb\_client.InfluxDBClient method*), 14  
 create\_bucket() (*influxdb\_client.BucketsApi method*), 17  
 create\_label() (*influxdb\_client.LabelsApi method*), 19  
 create\_organization() (*influxdb\_client.OrganizationsApi method*), 20  
 create\_task() (*influxdb\_client.TasksApi method*), 23  
 create\_task\_cron() (*influxdb\_client.TasksApi method*), 23  
 create\_task\_every() (*influxdb\_client.TasksApi method*), 23  
 create\_user() (*influxdb\_client.UsersApi method*), 21  
 created\_at (*influxdb\_client.domain.Bucket attribute*), 18

created\_at (*influxdb\_client.domain.Organization attribute*), 21  
 created\_at (*influxdb\_client.domain.Task attribute*), 25  
 cron (*influxdb\_client.domain.Task attribute*), 25

## D

delete\_api() (*influxdb\_client.InfluxDBClient method*), 14  
 delete\_bucket() (*influxdb\_client.BucketsApi method*), 17  
 delete\_label() (*influxdb\_client.LabelsApi method*), 19  
 delete\_label() (*influxdb\_client.TasksApi method*), 23  
 delete\_member() (*influxdb\_client.TasksApi method*), 23  
 delete\_organization() (*influxdb\_client.OrganizationsApi method*), 20  
 delete\_owner() (*influxdb\_client.TasksApi method*), 23  
 delete\_task() (*influxdb\_client.TasksApi method*), 23  
 description (*influxdb\_client.domain.Bucket attribute*), 18  
 description (*influxdb\_client.domain.Organization attribute*), 21  
 description (*influxdb\_client.domain.Task attribute*), 25

## E

every (*influxdb\_client.domain.Task attribute*), 25

## F

find\_bucket\_by\_id() (*influxdb\_client.BucketsApi method*), 17  
 find\_bucket\_by\_name() (*influxdb\_client.BucketsApi method*), 17

[find\\_buckets\(\)](#) ([influxdb\\_client.BucketsApi](#) [method](#)), 17  
[find\\_label\\_by\\_id\(\)](#) ([influxdb\\_client.LabelsApi](#) [method](#)), 20  
[find\\_label\\_by\\_org\(\)](#) ([influxdb\\_client.LabelsApi](#) [method](#)), 20  
[find\\_labels\(\)](#) ([influxdb\\_client.LabelsApi](#) [method](#)), 20  
[find\\_organization\(\)](#) ([influxdb\\_client.OrganizationsApi](#) [method](#)), 20  
[find\\_organizations\(\)](#) ([influxdb\\_client.OrganizationsApi](#) [method](#)), 20  
[find\\_task\\_by\\_id\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 23  
[find\\_tasks\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 23  
[find\\_tasks\\_by\\_user\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 23  
[flush\(\)](#) ([influxdb\\_client.WriteApi](#) [method](#)), 16  
[flux](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 25  
[from\\_config\\_file\(\)](#) ([influxdb\\_client.InfluxDBClient](#) [class](#) [method](#)), 14  
[from\\_env\\_properties\(\)](#) ([influxdb\\_client.InfluxDBClient](#) [class](#) [method](#)), 14

## G

[get\\_labels\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 24  
[get\\_logs\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 24  
[get\\_members\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 24  
[get\\_owners\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 24  
[get\\_run\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 24  
[get\\_run\\_logs\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 24  
[get\\_runs\(\)](#) ([influxdb\\_client.TasksApi](#) [method](#)), 24

## H

[health\(\)](#) ([influxdb\\_client.InfluxDBClient](#) [method](#)), 14

## I

[id](#) ([influxdb\\_client.domain.Bucket](#) [attribute](#)), 18  
[id](#) ([influxdb\\_client.domain.Organization](#) [attribute](#)), 21  
[id](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[id](#) ([influxdb\\_client.domain.User](#) [attribute](#)), 22  
[InfluxDBClient](#) ([class](#) in [influxdb\\_client](#)), 13

## L

[labels](#) ([influxdb\\_client.domain.Bucket](#) [attribute](#)), 18  
[labels](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[labels\\_api\(\)](#) ([influxdb\\_client.InfluxDBClient](#) [method](#)), 14

[LabelsApi](#) ([class](#) in [influxdb\\_client](#)), 19  
[last\\_run\\_error](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[last\\_run\\_status](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[latest\\_completed](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[links](#) ([influxdb\\_client.domain.Bucket](#) [attribute](#)), 18  
[links](#) ([influxdb\\_client.domain.Organization](#) [attribute](#)), 21  
[links](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[links](#) ([influxdb\\_client.domain.User](#) [attribute](#)), 22

## M

[me\(\)](#) ([influxdb\\_client.OrganizationsApi](#) [method](#)), 20  
[me\(\)](#) ([influxdb\\_client.UsersApi](#) [method](#)), 22

## N

[name](#) ([influxdb\\_client.domain.Bucket](#) [attribute](#)), 18  
[name](#) ([influxdb\\_client.domain.Organization](#) [attribute](#)), 21  
[name](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[name](#) ([influxdb\\_client.domain.User](#) [attribute](#)), 22

## O

[oauth\\_id](#) ([influxdb\\_client.domain.User](#) [attribute](#)), 22  
[offset](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[org](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 26  
[org\\_id](#) ([influxdb\\_client.domain.Bucket](#) [attribute](#)), 18  
[org\\_id](#) ([influxdb\\_client.domain.Task](#) [attribute](#)), 27  
[Organization](#) ([class](#) in [influxdb\\_client.domain](#)), 20  
[organizations\\_api\(\)](#) ([influxdb\\_client.InfluxDBClient](#) [method](#)), 14  
[OrganizationsApi](#) ([class](#) in [influxdb\\_client](#)), 20

## Q

[query\(\)](#) ([influxdb\\_client.QueryApi](#) [method](#)), 15  
[query\\_api\(\)](#) ([influxdb\\_client.InfluxDBClient](#) [method](#)), 14  
[query\\_csv\(\)](#) ([influxdb\\_client.QueryApi](#) [method](#)), 15  
[query\\_data\\_frame\(\)](#) ([influxdb\\_client.QueryApi](#) [method](#)), 15  
[query\\_data\\_frame\\_stream\(\)](#) ([influxdb\\_client.QueryApi](#) [method](#)), 15  
[query\\_raw\(\)](#) ([influxdb\\_client.QueryApi](#) [method](#)), 16  
[query\\_stream\(\)](#) ([influxdb\\_client.QueryApi](#) [method](#)), 16  
[QueryApi](#) ([class](#) in [influxdb\\_client](#)), 15

## R

[ready\(\)](#) ([influxdb\\_client.InfluxDBClient](#) [method](#)), 14  
[retention\\_rules](#) ([influxdb\\_client.domain.Bucket](#) [attribute](#)), 18

`retry_run()` (*influxdb\_client.TasksApi method*), 24  
`rp` (*influxdb\_client.domain.Bucket attribute*), 19  
`run_manually()` (*influxdb\_client.TasksApi method*), 24

## S

`status` (*influxdb\_client.domain.Organization attribute*), 21  
`status` (*influxdb\_client.domain.Task attribute*), 27  
`status` (*influxdb\_client.domain.User attribute*), 22

## T

`Task` (*class in influxdb\_client.domain*), 25  
`tasks_api()` (*influxdb\_client.InfluxDBClient method*), 14  
`TasksApi` (*class in influxdb\_client*), 22  
`to_dict()` (*influxdb\_client.domain.Bucket method*), 19  
`to_dict()` (*influxdb\_client.domain.Organization method*), 21  
`to_dict()` (*influxdb\_client.domain.Task method*), 27  
`to_dict()` (*influxdb\_client.domain.User method*), 22  
`to_str()` (*influxdb\_client.domain.Bucket method*), 19  
`to_str()` (*influxdb\_client.domain.Organization method*), 21  
`to_str()` (*influxdb\_client.domain.Task method*), 27  
`to_str()` (*influxdb\_client.domain.User method*), 22  
`type` (*influxdb\_client.domain.Bucket attribute*), 19  
`type` (*influxdb\_client.domain.Task attribute*), 27

## U

`update_label()` (*influxdb\_client.LabelsApi method*), 20  
`update_task()` (*influxdb\_client.TasksApi method*), 24  
`update_task_request()` (*influxdb\_client.TasksApi method*), 24  
`updated_at` (*influxdb\_client.domain.Bucket attribute*), 19  
`updated_at` (*influxdb\_client.domain.Organization attribute*), 21  
`updated_at` (*influxdb\_client.domain.Task attribute*), 27  
`User` (*class in influxdb\_client.domain*), 22  
`users_api()` (*influxdb\_client.InfluxDBClient method*), 14  
`UsersApi` (*class in influxdb\_client*), 21

## W

`write()` (*influxdb\_client.WriteApi method*), 16  
`write_api()` (*influxdb\_client.InfluxDBClient method*), 14  
`WriteApi` (*class in influxdb\_client*), 16