Python client library

Use the InfluxDB Python client library to integrate InfluxDB into Python scripts and applications.

This guide presumes some familiarity with Python and InfluxDB. If just getting started, see Get started with InfluxDB.

Before you begin

1. Install the InfluxDB Python library:

   ```
   pip install influxdb-client
   ```

2. Ensure that InfluxDB is running. If running InfluxDB locally, visit http://localhost:8086. (If using InfluxDB Cloud, visit the URL of your InfluxDB Cloud UI. For example: https://us-west-2-1.aws.cloud2.influxdata.com.)

Write data to InfluxDB with Python

We are going to write some data in line protocol using the Python library.

1. In your Python program, import the InfluxDB client library and use it to write data to InfluxDB.

```python
import influxdb_client
from influxdb_client.client.write_api import SYNCHRONO
```

2. Define a few variables with the name of your bucket, organization, and token.

```python
bucket = "<my-bucket>"
org = "<my-org>"
token = "<my-token>"
# Store the URL of your InfluxDB instance
url="http://localhost:8086"
```

⚙ *Change InfluxDB URL*

3. Instantiate the client. The `InfluxDBClient` object takes three named parameters: `url`, `org`, and `token`. Pass in the named parameters.

```python
client = influxdb_client.InfluxDBClient(
   url=url,
   token=token,
   org=org
)
```

The `InfluxDBClient` object has a `write_api` method used for configuration.

4. Instantiate a **write client** using the `client` object and the `write_api` method. Use the `write_api` method to configure the writer object.

```
write_api = client.write_api(write_options=SYNCHRONOUS
```

5. Create a point object and write it to InfluxDB using the `write` method of the API writer object. The write method requires three parameters: `bucket` , `org` , and `record` .

```
p = influxdb_client.Point("my_measurement").tag("locat
write_api.write(bucket=bucket, org=org, record=p)
```

Complete example write script

```
import influxdb_client
from influxdb_client.client.write_api import SYNCHRONOUS

bucket = "<my-bucket>"
org = "<my-org>"
token = "<my-token>"
# Store the URL of your InfluxDB instance
url="http://localhost:8086"

client = influxdb_client.InfluxDBClient(
    url=url,
    token=token,
    org=org
)

write_api = client.write_api(write_options=SYNCHRONOUS)

p = influxdb_client.Point("my_measurement").tag("locatior
write_api.write(bucket=bucket, org=org, record=p)
```

Change InfluxDB URL

Query data from InfluxDB with Python

1. Instantiate the **query client**.

   ```
   query_api = client.query_api()
   ```

2. Create a Flux query, and then format it as a Python string.

   ```
   query = ' from(bucket:"my-bucket")\
   |> range(start: -10m)\
   |> filter(fn:(r) => r._measurement == "my_measurement"
   |> filter(fn: (r) => r.location == "Prague")\
   |> filter(fn:(r) => r._field == "temperature" ) '
   ```

   The query client sends the Flux query to InfluxDB and returns a Flux object with a table structure.

3. Pass the `query()` method two named parameters: `org` and `query` .

   ```
   result = query_api.query(org=org, query=query)
   ```

4. Iterate through the tables and records in the Flux object.

   - Use the `get_value()` method to return values.
   - Use the `get_field()` method to return fields.

```
results = []
for table in result:
  for record in table.records:
    results.append((record.get_field(), record.get_val

print(results)
[(temperature, 25.3)]
```

The Flux object provides the following methods for accessing your data:

- `get_measurement()` : Returns the measurement name of the record.
- `get_field()` : Returns the field name.
- `get_value()` : Returns the actual field value.
- `values` : Returns a map of column values.
- `values.get("<your tag>")` : Returns a value from the record for given column.
- `get_time()` : Returns the time of the record.
- `get_start()` : Returns the inclusive lower time bound of all records in the current table.
- `get_stop()` : Returns the exclusive upper time bound of all records in the current table.

Complete example query script

```
query_api = client.query_api()
query = ‘ from(bucket:"my-bucket")\
|> range(start: -10m)\
|> filter(fn:(r) => r._measurement == "my_measurement")\
|> filter(fn: (r) => r.location == "Prague")\
|> filter(fn:(r) => r._field == "temperature" )‘
result = query_api.query(org=org, query=query)
```

```
results = []
for table in result:
    for record in table.records:
        results.append((record.get_field(), record.get_va

print(results)
[(temperature, 25.3)]
```

For more information, see the Python client README on GitHub.

client libraries   *python*

Was this page helpful?   Yes   No

Support and feedback

Thank you for being part of our community! We welcome and encourage your feedback and bug reports for InfluxDB and this documentation. To find support, use the following resources:

InfluxData Community   InfluxDB Community Slack

InfluxDB Cloud and InfluxDB Enterprise customers can contact InfluxData Support.

Edit this page

Submit docs issue

Submit InfluxDB issue