Confluent Cloud Clients Python Library

The Confluent Cloud Clients Python Library provides a set of clients for interacting with Confluent Cloud REST APIs. The library includes clients for:

- **Flink**
- **Kafka**
- **Schema Registry**
- **Tableflow**
- **Metrics**
- **Environment**
- **IAM**

> **Note:** *This library is in active development and is subject to change. It covers only the methods I have needed so far. If you need a method that is not covered, please feel free to open an issue or submit a pull request.*

Table of Contents

1.0 Library Clients

1.1 Flink Client

The **Flink Client** provides the following methods:

- `delete_statement`
- `delete_statements_by_phase`
- `drop_table`

> ***Note:*** *"The `drop_table` method will drop the table and all associated statements, including the backing Kafka Topic and Schemas."*

- `get_compute_pool`
- `get_compute_pool_list`
- `get_statement_list`
- `stop_statement`

> ***Note:*** *"Confluent Cloud for Apache Flink enforces a **30-day** retention for statements in terminal states."*

- `submit_statement`
- `update_statement`

- `update_all_sink_statements`

1.2 Kafka Topic Client

The **Kafka Topic Client** provides the following methods:

- `delete_kafka_topic`
- `kafka_topic_exist`
- `kafka_get_topic`

1.3 Schema Registry Client

The **Schema Registry Client** provides the following methods:

- `convert_avro_schema_into_string`
- `delete_kafka_topic_key_schema_subject`
- `delete_kafka_topic_value_schema_subject`
- `get_global_topic_subject_compatibility_level`
- `get_topic_subject_compatibility_level`
- `get_topic_subject_latest_schema`
- `register_topic_subject_schema`
- `set_topic_subject_compatibility_level`

1.4 Tableflow Client

The **Tableflow Client** provides the following methods:

- `get_tableflow_topic`
- `get_tableflow_topic_table_path`

1.5 Metrics Client

1.5.1 Get Topic Totals

The **Metrics Client** provides the following methods:

- `get_topic_total`
- `get_topic_daily_aggregated_totals`

Metric Type	Description
`RECEIVED_BYTES`	The delta count of bytes of the customer's data received from the network. Each sample is the number of bytes received since the previous data sample. The count is sampled every 60 seconds.
`RECEIVED_RECORDS`	The delta count of records of the customer's data received from the network. Each sample is the number of records received since the previous data sample. The count is sampled every 60 seconds.
`SENT_BYTES`	The delta count of bytes of the customer's data sent to the network. Each sample is the number of bytes sent since the previous data sample. The count is sampled every 60 seconds.
`SENT_RECORDS`	The delta count of records of the customer's data sent to the network. Each sample is the number of records sent since the previous data sample. The count is sampled every 60 seconds.

1.5.2 Is Topic Partition Hot

The **Metrics Client** provides the following methods:

- `is_topic_partition_hot`

Metric Type	Description
`INGRESS`	An indicator of the presence of a hot partition caused by ingress throughput. The value is 1.0 when a hot partition is detected, and empty when there is no hot partition detected
`EGRESS`	An indicator of the presence of a hot partition caused by egress throughput. The value is 1.0 when a hot partition is detected, and empty when there is no hot partition detected

1.6 Environment Client

The **Environment Client** provides the following methods:

- `get_environment_list`
- `get_kafka_cluster_list`

1.7 IAM Client

The **IAM Client** provides the following methods:

- `create_api_key`
- `delete_api_key`

2.0 Unit Tests

The library includes unit tests for each client. The tests are located in the `tests` directory. To use them, you must clone the repo locally:

```
git clone https://github.com/j3-signalroom/cc-clients-python_lib.git
```

Since this project was built using **`uv`**, please install it, and then run the following command to install all the project dependencies:

```
 uv sync
```

Then within the `tests` directory, create the `.env` file and add the following environment variables, filling them with your Confluent Cloud credentials and other required values:

```
BOOTSTRAP_SERVER_CLOUD_PROVIDER=
BOOTSTRAP_SERVER_CLOUD_REGION=
BOOTSTRAP_SERVER_ID=
CLOUD_PROVIDER=
CLOUD_REGION=
COMPUTE_POOL_ID=
CONFLUENT_CLOUD_API_KEY=
CONFLUENT_CLOUD_API_SECRET=
ENVIRONMENT_ID=
FLINK_API_KEY=
FLINK_API_SECRET=
FLINK_CATALOG_NAME=
FLINK_DATABASE_NAME=
FLINK_STATEMENT_NAME=
FLINK_TABLE_NAME=
FLINK_URL=
KAFKA_API_KEY=
KAFKA_API_SECRET=
KAFKA_CLUSTER_ID=
KAFKA_TOPIC_NAME=
ORGANIZATION_ID=
PRINCIPAL_ID=
QUERY_START_TIME=
QUERY_END_TIME=
SCHEMA_REGISTRY_API_KEY=
SCHEMA_REGISTRY_API_SECRET=
SCHEMA_REGISTRY_URL=
TABLEFLOW_API_KEY=
TABLEFLOW_API_SECRET=
```

> **Note:** *The `QUERY_START_TIME` and `QUERY_END_TIME` environment variables should be in the format `YYYY-MM-DDTHH:MM:SS`, for example, `2025-09-01T00:00:00`.*

2.1 Flink Client

To run a specific test, use one of the following commands:

Unit Test	Command
Delete a Flink Statement	`uv run pytest -s tests/test_flink_client.py::test_delete_statement`

Unit Test	Command
Delete all Flink Statements by Phase	`uv run pytest -s tests/test_flink_client.py::test_delete_statements_by_phase`
Get list of the all the Statements	`uv run pytest -s tests/test_flink_client.py::test_get_statement_list`
Submit a Flink Statement	`uv run pytest -s tests/test_flink_client.py::test_submit_statement`
Get Compute Pool List	`uv run pytest -s tests/test_flink_client.py::test_get_compute_pool_list`
Get Compute Pool	`uv run pytest -s tests/test_flink_client.py::test_get_compute_pool`
Stop a Flink Statement	`uv run pytest -s tests/test_flink_client.py::test_stop_statement`
Update a Flink Statement	`uv run pytest -s tests/test_flink_client.py::test_update_statement`
Update all the Sink Statements	`uv run pytest -s tests/test_flink_client.py::test_update_all_sink_statements`
Drop a Flink Table along with any associated statements, including the backing Kafka Topic and Schemas	`uv run pytest -s tests/test_flink_client.py::test_drop_table`

Otherwise, to run all the tests, use the following command:

```
uv run pytest -s tests/test_flink_client.py
```

> **Note:** *The tests are designed to be run in a specific order. If you run them out of order, you may encounter errors. The tests are also designed to be run against a Confluent Cloud environment. If you run them against a local environment, you may encounter errors.*

2.2 Kafka Topic Client

To run a specific test, use one of the following commands:

Unit Test	Command
Delete a Kafka Topic	`uv run pytest -s tests/test_kafka_topic_client.py::test_delete_kafka_topic`
Checks if a Kafka Topic Exist	`uv run pytest -s tests/test_kafka_topic_client.py::test_kafka_topic_exist`
Get Kafka Topic Details	`uv run pytest -s tests/test_kafka_topic_client.py::test_kafka_get_topic`

Otherwise, to run all the tests, use the following command:

```
uv run pytest -s tests/test_kafka_topic_client.py
```

> **Note:** *The tests are designed to be run in a specific order. If you run them out of order, you may encounter errors. The tests are also designed to be run against a Confluent Cloud environment. If you run them against a local environment, you may encounter errors.*

2.3 Schema Registry Client

To run a specific test, use one of the following commands:

Unit Test	Command
Get the Subject Compatibility Level	`uv run pytest -s tests/test_schema_registry_client.py::test_get_subject_compatibility_level`
Delete the Kafka Topic Key Schema Subject	`uv run pytest -s tests/test_schema_registry_client.py::test_delete_kafka_topic_key_schema_subject`

Unit Test	Command
Delete the Kafka Topic Value Schema Subject	`uv run pytest -s tests/test_schema_registry_client.py::test_delete_kafka_topic_value_schema_subject`

Otherwise, to run all the tests, use the following command:

```
uv run pytest -s tests/test_schema_registry_client.py
```

> **Note:** *The tests are designed to be run in a specific order. If you run them out of order, you may encounter errors. The tests are also designed to be run against a Confluent Cloud environment. If you run them against a local environment, you may encounter errors.*

2.4 Tableflow Client

To run a specific test, use one of the following commands:

Unit Test	Command
Get the Tableflow Topic	`uv run pytest -s tests/test_tableflow_client.py::test_get_tableflow_topic`
Get the Tableflow Topic Table Path	`uv run pytest -s tests/test_tableflow_client.py::test_get_tableflow_topic_table_path`

Otherwise, to run all the tests, use the following command:

```
uv run pytest -s tests/test_tableflow_client.py
```

> **Note:** *The tests are designed to be run in a specific order. If you run them out of order, you may encounter errors. The tests are also designed to be run against a Confluent Cloud environment. If you run them against a local environment, you may encounter errors.*

2.5 Metrics Client

To run a specific test, use one of the following commands:

Unit Test	Command
Get the Topic Received Total Bytes	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_received_total_bytes`
Get the Topic Received Total Records	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_received_total_records`
Get the Topic Received Daily Aggregated Totals Bytes	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_received_daily_aggregated_totals_bytes`
Get the Topic Received Daily Aggregated Totals Records	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_received_daily_aggregated_totals_records`

Unit Test	Command
Compute the Topic Partition Count Based on Received Bytes and Record Count	`uv run pytest -s tests/test_metrics_client.py::test_compute_topic_partition_count_based_on_received_bytes_record_count`
Get the Topic Sent Total Bytes	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_sent_total_bytes`
Get the Topic Sent Total Records	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_sent_total_records`
Get the Topic Sent Daily Aggregated Totals Bytes	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_sent_daily_aggregated_totals_bytes`
Get the Topic Sent Daily Aggregated Totals Records	`uv run pytest -s tests/test_metrics_client.py::test_get_topic_sent_daily_aggregated_totals_records`
Compute the Topic Partition Count Based on Sent Bytes and Record Count	`uv run pytest -s tests/test_metrics_client.py::test_compute_topic_partition_count_based_on_sent_bytes_record_count`
Check if a Topic Partition is Hot Based on Ingress	`uv run pytest -s tests/test_metrics_client.py::test_is_topic_partition_hot_by_ingress_throughput`
Check if a Topic Partition is Hot Based on Egress	`uv run pytest -s tests/test_metrics_client.py::test_is_topic_partition_hot_by_egress_throughput`

Otherwise, to run all the tests, use the following command:

```
uv run pytest -s tests/test_metrics_client.py
```

> **Note:** *The tests are designed to be run in a specific order. If you run them out of order, you may encounter errors. The tests are also designed to be run against a Confluent Cloud environment. If you run them against a local environment, you may encounter errors.*

2.6 Environment Client

To run a specific test, use one of the following commands:

Unit Test	Command
Get list of all the Environments	`uv run pytest -s tests/test_environment_client.py::test_get_environment_list`
Get list of the all the Kafka clusters	`uv run pytest -s tests/test_environment_client.py::test_get_kafka_cluster_list`

Otherwise, to run all the tests, use the following command:

```
uv run pytest -s tests/test_environment_client.py
```

> **Note:** *The tests are designed to be run in a specific order. If you run them out of order, you may encounter errors. The tests are also designed to be run against a Confluent Cloud environment. If you run them against a local environment, you may encounter errors.*

2.7 IAM Client

To run a specific test, use one of the following commands:

Unit Test	Command
Create and Delete an API Key	`uv run pytest -s tests/test_iam_client.py::TestIamClient::test_create_and_delete_api_key`
Iterate through Environments Creating and Deleting API Keys	`uv run pytest -s tests/test_iam_client.py::TestIamClient::test_creating_and_deleting_kafka_api_keys`

Otherwise, to run all the tests, use the following command:

```
uv run pytest -s tests/test_iam_client.py
```

3.0 Installation

Install the Confluent Cloud Clients Python Library using `pip`:

```
pip install cc-clients-python-lib
```

Or, using `uv`:

```
uv add cc-clients-python-lib
```

4.0 Resources

4.1 Architecture Design Records (ADRs)

- 001 Architectural Design Record (ADR): Drop Table Plus

4.2 API Documentation

- Flink SQL REST API for Confluent Cloud for Apache Flink
- Kafka REST APIs for Confluent Cloud
- Confluent Cloud APIs - Topic (v3)
- Confluent Cloud Schema Registry REST API Usage

4.3 Flink Resources

- CCAF State management
- Monitor and Manage Flink SQL Statements in Confluent Cloud for Apache Flink
- DROP TABLE Statement in Confluent Cloud for Apache Flink

4.4 Tableflow Resources

- Tableflow Topics (tableflow/v1)

4.4 Tableflow Resources

- Tableflow Topics (tableflow/v1)

4.5 Metrics Resources

- Confluent Cloud Metrics API Version 2 Reference
- Confluent Cloud Metrics API: Metrics Reference
- Confluent Cloud Metrics

4.6 Other Resources

- How to programmatically pause and resume a Flink statement
- How to programmatically pause and resume a Flink statement REDUX