MDML Python Client

Feb 10, 2022

Contents

This client connects users to the Manufacturing Data and Machine Learning Platform at Argonne National Laboratory.

CHAPTER 1

Producer class

class mdml_client.**kafka_mdml_producer**(*topic*, *schema=None*, *config=None*, *add_time=True*, *kafka_host='merf.egs.anl.gov'*, *kafka_port=9092*, *schema_host='merf.egs.anl.gov'*, *schema_port=8081*)

Creates a producer instance for producing data to an MDML instance.

Parameters

- **topic** (*str*) – Topic to send under
- **schema** (*dict or str*) – JSON schema for the message value. If dict, value is used as the schema. If string, value is used as a file path to a json file.
- **config** (*dict*) – Confluent Kafka client config (only recommended for advanced usage - overwrites other parameters)
- **add_time** (*bool*) – If True, adds a value named 'mdml_time' to the data object that represents when the producer sent the message
- **kafka_host** (*str*) – Host name of the kafka broker
- **kafka_port** (*int*) – Port used for the kafka broker
- **schema_host** (*str*) – Host name of the kafka schema registry
- **schema_port** (*int*) – Port of the kafka schema registry

flush()

Flush (send) any messages currently waiting in the producer.

produce(*data*, *key=None*, *partition=None*)

Produce data to the supplied topic

Parameters

- **data** (*dict*) – Dictionary of the data
- **key** (*str*) – String for the Kafka assignor to use to calculate a partition
- **partition** (*int*) – Number of the partition to assign the message to

CHAPTER 2

Consumer class

class mdml_client.**kafka_mdml_consumer**(*topics*, *group*, *auto_offset_reset='earliest'*, *show_mdml_time=True*, *kafka_host='merf.egs.anl.gov'*, *kafka_port=9092*, *schema_host='merf.egs.anl.gov'*, *schema_port=8081*)

Creates a consumer to consume messages from an MDML instance.

Parameters

- **topics** (*list (str)*) – Topics to consume from
- **group** (*str*) – Consumer group ID. Messages are only consumed by a given group ID once.
- **auto_offset_reset** (*str*) – 'earliest' or 'latest'. 'earliest' is the default and will start consuming messages from where the consumer group left off. 'latest' will start consuming messages from the time that the consumer is started.
- **show_mdml_time** (*bool*) – Indicator if the value of 'mdml_time' should be shown or suppressed
- **kafka_host** (*str*) – Host name of the kafka broker
- **kafka_port** (*int*) – Port used for the kafka broker
- **schema_host** (*str*) – Host name of the kafka schema registry
- **schema_port** (*int*) – Port of the kafka schema registry

close()

Closes down the consumer. Ensures that received messages have been acknowledged by Kafka.

consume(*poll_timeout=1.0*, *overall_timeout=300.0*, *verbose=True*)

Start consuming from the specified topic

Parameters

- **poll_timeout** (*float*) – Timeout to wait when consuming one message

- **`overall_timeout`** (*`float`*) – Timeout to wait until the consume generator is closed down. This timeout is restarted every time a new message is received
- **`verbose`** (*`bool`*) – Print a message with notes when the consume loop starts

Yields *dict* – A dictionary containing the topic and value of a single message

`consume_chunks`(*poll_timeout=1.0*, *overall_timeout=300.0*, *save_file=True*, *save_dir='.'*, *passthrough=True*, *verbose=True*)

Consume messages from a topic that contains chunked messages. The original file is saved to disk by default.

Parameters

- **`poll_timeout`** (*`float`*) – Timeout for one message to reach the consumer
- **`overall_timeout`** (*`float`*) – Time until the consumer will be shutdown if no messages are received
- **`save_file`** (*`bool`*) – True if the chunked file should be saved. False will return the original data contained in the file
- **`save_dir`** (*`str`*) – Directory to save files
- **`passthrough`** (*`bool`*) – If multiple topics are subscribed to and one of them is not using chunking, passthrough=True will ensure those messages are still yielded by the generator
- **`verbose`** (*`bool`*) – Print details regarding the consumer on start

Yields

- *tuple* – A tuple containing (timestamp, data) where timestamp is the time the first chunk of the message was sent and where data is either a filepath (save_file=True) or the bytes of the file that was chunked and streamed (save_file=False).
- *dict* – If passthrough=True is used and a message from a topic without chunking is received, a dictionary containing the topic and value of the message will be yielded. Otherwise, a tuple is returned

CHAPTER 3

Schema-less Producer & Consumer classes

class mdml_client.**kafka_mdml_producer_schemaless**(*topic*, *config=None*, *kafka_host='merf.egs.anl.gov'*, *kafka_port=9092*)

Creates a schemaless Producer instance for interacting with the MDML.

Parameters

- **topic** (*str*) – Topic to send under
- **config** (*dict*) – Confluent Kafka client config
- **kafka_host** (*str*) – Host name of the kafka broker
- **kafka_port** (*int*) – Port used for the Kafka broker

flush()

Flush (send) any messages currently waiting in the producer.

produce(*data*, *key=None*, *partition=None*)

Produce data to the supplied topic

Parameters

- **data** (*dict*) – Dictionary of the data
- **key** (*string*) – Key of the message (used in determining a partition) - not required
- **partition** (*int*) – Partition used to save the message - not required

class mdml_client.**kafka_mdml_consumer_schemaless**(*topics*, *group*, *kafka_host='merf.egs.anl.gov'*, *kafka_port=9092*)

Creates a serializingProducer instance for interacting with the MDML.

Parameters

- **topics** (*list(str)*) – Topics to consume from
- **group** (*str*) – Consumer group ID. Messages are only consumed by a given group ID once.

- **`kafka_host`** (*`str`*) – Host name of the kafka broker
- **`kafka_port`** (*`int`*) – Port used for the kafka broker

`close`()

Closes down the consumer. Ensures that received messages have been acknowledged by Kafka.

`consume`(*poll_timeout=1.0*, *overall_timeout=300.0*, *verbose=True*)

Yields *dict* – A dictionary containing the topic and value of a single message

CHAPTER 4

Experiment & Replay Service Functions

`mdml_client.`**`start_experiment`**(*id*, *topics*, *producer_kwargs={}*)

Start an experiment with the MDML Experiment service. Messages produced on all of the specified topics will be saved to a file and upload to S3.

Parameters

- **`id`** (`str`) – Unique ID for the experiment
- **`topics`** (`list(str)`) – Topics to consume from that make up the experiment
- **`producer_kwargs`** (`dict`) – Dictionary that is passed as kwargs to the underlying producer in this function. Parameter names should be the same as those in a kafka_mdml_producer.

`mdml_client.`**`stop_experiment`**(*id*, *producer_kwargs={}*)

Stop a previously started experiment. Upon stopping, the experiment service will package all data streamed during an experiment, verify all data is present, and write a file to S3.

Parameters

- **`id`** (`str`) – Unique ID for the experiment
- **`producer_kwargs`** (`dict`) – Dictionary that is passed as kwargs to the underlying producer in this function. Parameter names should be the same as those in a kafka_mdml_producer.

`mdml_client.`**`replay_experiment`**(*experiment_id*, *speed=1*, *producer_kwargs={}*)

Replay an experiment - stream data back down their original topics

Parameters

- **`experiment_id`** (`str`) – Unique ID of the experiment to replay
- **`speed`** (`int`) – Speed multiplier used during the replay
- **`producer_kwargs`** (`dict`) – Dictionary of kwargs for this functions internal producer

CHAPTER 5

MDML S3 Client

This is used for "coat-checking" large files.

class mdml_client.**kafka_mdml_s3_client**(*topic*, *s3_endpoint=None*, *s3_access_key=None*, *s3_secret_key=None*, *kafka_host='merf.egs.anl.gov'*, *kafka_port=9092*, *schema_host='merf.egs.anl.gov'*, *schema_port=8081*, *schema=None*)

Creates an MDML producer for sending >1MB files to an s3 location. Simultaneously, the MDML sends upload information along a Kafka topic to be received by a client that can retrieve the file.

Parameters

- **topic** (*str*) – Topic to send under
- **s3_endpoint** (*str*) – Host of the S3 service
- **s3_access_key** (*str*) – S3 access key
- **s3_secret_key** (*str*) – S3 secret key
- **kafka_host** (*str*) – Host name of the kafka broker
- **kafka_port** (*int*) – Port used for the kafka broker
- **schema_host** (*str*) – Host name of the kafka schema registry
- **schema_port** (*int*) – Port of the kafka schema registry
- **schema** (*dict or str*) – Schema of the messages sent on the supplied topic. Default schema sends a dictionary containing the time of upload and the location for retrieval. If dict, value is used as the schema. If string, value is used as a file path to a json file.

consume(*bucket*, *object_name*, *save_filepath*)

Gets a file from an S3 bucket. Can return the bytes of the file or save the file to a specified path.

Parameters

- **bucket** (*str*) – Name of the bucket the object is saved in
- **object_name** (*str*) – Name/key of the object to retrieve from the bucket

- **`save_filepath`** (*`str`*) – Path in which to save the downloaded file. Using a value of None will return the bytes of the file instead of saving to a file

`produce` (*filepath*, *obj_name*, *payload=None*)
Produce data to supplied S3 endpoint and Kafka topic

Parameters

- **`filepath`** (*`str`*) – Path of the file to upload to the S3 bucket
- **`obj_name`** (*`str`*) – Name to store the file under
- **`payload`** (*`dict`*) – Payload for the message sent on the Kafka topic. Only used when the default schema has been overridden.

CHAPTER 6

Helper Functions

`mdml_client.`**`create_schema`**(*d*, *title*, *descr*, *required_keys=None*, *add_time=False*)

Create a schema for use in a kafka_mdml_producer object. An example of the data object that will be produced is needed to create the schema.

Parameters

- **`d`** (*`dict`*) – Data object to translate into a schema
- **`title`** (*`str`*) – Title of the schema
- **`descr`** (*`str`*) – Description of the schema
- **`required_keys`** (*`list(str)`*) – List of strings of the keys that are required in the schema

Returns

Return type Schema dictionary compatible with kafka_mdml_producer

`mdml_client.`**`chunk_file`**(*fn*, *chunk_size*, *use_b64=True*, *encoding='utf-8'*, *file_id=None*)

Chunks a file into parts. Yields dictionaries containing the file bytes encoded in base64. Base64 is used since the kafka Producer requires a string and some files must be opened in byte format.

Parameters

- **`fn`** (*`str`*) – Path to the file
- **`chunk_size`** (*`int`*) – Size of chunk to use
- **`use_b64`** (*`bool`*) – True to return the file bytes as a base64 encoded string
- **`encoding`** (*`string`*) – Encoding to use to open the file if use_b64 is False
- **`file_id`** (*`string`*) – File ID to use in the chunking process if the fn param is not suitable

Yields

- *Dictionary containing a chunk of data and metadata information*
- *required to piece all of the chunks back together.*

CHAPTER 7

Indices and tables

- genindex

Index