amazon-kinesis-utils

Oct 16, 2020

Contents

A part of Baikonur OSS project

CHAPTER 1

About Baikonur Kinesis/Lambda logging

1.1 Baikonur Kinesis/Lambda logging requirements

Baikonur Kinesis/Lambda logging can be defined as anything using Kinesis Data Streams with one or more of the following Baikonur OSS Lambda Modules:

- terraform-aws-lambda-kinesis-forward
- terraform-aws-lambda-kinesis-to-es
- terraform-aws-lambda-kinesis-to-s3

These modules have the following common schema requirements:

- All data must be JSON objects
- All data must include the following keys (key names are customizable for each Lambda module):
  - `log_type`: Log type identifier
  - `log_id`: Any unique identifier (e.g. `uuid.uuid4()`)
  - `time`: Any timestamp supported by dateutil.parser.parse

Common schema requirements are derived from following needs:

1. Easier parsing
2. Interoperability between different Lambda modules
   - Different modules can be attached to a single Kinesis Data Stream and work on same data as long as data are JSON-objects and common schema requirements are met.
3. Ability to create behaviour based on keys in common schema
   - One of the most important features is ability to apply whitelist on `log_type` field to, for instance, ignore logs other than those specified.

`log_id` and `time` keys are required by terraform-aws-lambda-kinesis-to-s3 (to ensure unique filenames) and terraform-aws-lambda-kinesis-to-es (for creating daily index names). Additionally, these fields are useful when troubleshooting.

Nevertheless `amazon-kinesis-utils` module name and default field names in description above, usage of this module and Lambda modules listed above are not limited to logging. As log as common schema requirements are met,

1.2 `amazon-kinesis-utils` submodules with specifics

This module was originally created for Baikonur OSS Lambda modules for logging with Kinesis. However, *baikonur_logging module* is the only submodule of `amazon-kinesis-utils` module that includes implications specific to Lambda modules above and their prerequisites. Other submodules are made to be universal.

1.2.1 `baikonur_logging` submodule specifics

Functions in *baikonur_logging module* submodule are provided to work with data structure called `log_dict`.

`log_dict` structure is as following:

```
>>> import datetime
>>> import uuid
>>> from kinesis_logging_utils import baikonur_logging
>>> log_dict = dict()
>>> baikonur_logging.append_to_log_dict(log_dict, log_type='test', log_data={'a':'b'},
↪ log_timestamp=datetime.datetime.now().isoformat(), log_id=str(uuid.uuid4()))
>>> log_dict
{'test': {'records': [{'a': 'b'}], 'first_timestamp': datetime.datetime(2020, 3, 1, 4,
↪ 55, 24, 966601), 'first_id': '4604dea6-5439-427a-bb41-c2f4807f3b72'}
```

This data structure allows us to iterate on `log_type` and retrieve all logs for a `log_type`.

`first_timestamp` and `first_id` are mainly used to generate timestamped, unique filenames when saving logs to S3.

CHAPTER 2

API Reference

2.1 API Reference

2.1.1 amazon_kinesis_utils package

Submodules

baikonur_logging module

Utilities specific to Baikonur Kinesis/Lambda logging modules.

amazon_kinesis_utils.baikonur_logging.**append_to_log_dict**(*dictionary: dict*, *log_type: str*, *log_data: object*, *log_timestamp=None*, *log_id=None*)

amazon_kinesis_utils.baikonur_logging.**parse_payload_to_log_dict**(*payload*, *log_dict*, *failed_dict*, *log_id_key*, *log_timestamp_key*, *log_type_key*, *log_type_unknown_prefix*, *log_type_whitelist=None*, *timestamp_required=False*)

amazon_kinesis_utils.baikonur_logging.**save_json_logs_to_s3**(*client*, *log_dict: dict*, *reason: str = 'not specified'*, *gzip_compress: bool = True*, *key_prefix: str = ''*)

kinesis module

Utilities to work with Kinesis Aggregated records, JSON events coming from CloudWatch Logs with subscription filters, gzipped JSON data and more.

exception amazon_kinesis_utils.kinesis.**KinesisException**
Bases: Exception

A custom exception returned on put_records_batch failures. Intentionally not catching this exception in Lambda Functions (source mapped to a Kinesis Data Stream) will make Lambda rerun until all record are successfully sent.

amazon_kinesis_utils.kinesis.**create_record**(*data: str*) → dict
Create a single Kinesis Record for use with PutRecords API

Parameters **data** – A string to convert to record

Returns Kinesis Record for PutRecords API

amazon_kinesis_utils.kinesis.**create_records**(*data: List[str]*) → List[dict]
Create Kinesis Records from multiple str data for use with PutRecords API

Parameters **data** – List of strings to convert to records

Returns List of Kinesis Records for PutRecords API

amazon_kinesis_utils.kinesis.**extract_data_from_json_cwl_message**(*message: dict*) → List[str]
Extract log events from CloudWatch Logs subscription filters JSON messages (parsed to dict). For details, see: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/SubscriptionFilters.html

Parameters **message** – Dictionary representing CloudWatch Logs subscription filters JSON messages

Returns List of raw log event messages

amazon_kinesis_utils.kinesis.**normalize_cloudwatch_messages**(*payload: str*) → List[str]
Normalize messages from CloudWatch Logs subscription filters and pass through other data

Parameters **payload** – A string containing JSON data (decoded payload inside Kinesis records)

Returns List of normalized raw data (CloudWatch Logs subscription filters may send multiple log events in one payload)

amazon_kinesis_utils.kinesis.**parse_records**(*raw_records: list*) → Generator[str, None, None]
Generator that de-aggregates, decodes, gzip decompresses Kinesis Records

Parameters **raw_records** – Raw Kinesis records (usually event['Records'] in Lambda handler function)

Returns

amazon_kinesis_utils.kinesis.**put_records_batch**(*client*, *stream_name: str*, *records: list*, *max_retries: int*, *max_batch_size: int = 500*) → List[dict]
Put multiple records to Kinesis Data Streams using PutRecords API in batches.

Parameters

- **client** – Kinesis API client (e.g. boto3.client('kinesis'))
- **stream_name** – Kinesis Data Streams stream name

- **records** – list of records to send. Records will be dumped with json.dumps
- **max_retries** – Maximum retries for resending failed records
- **max_batch_size** – Maximum number of records sent in a single PutRecords API call.

Returns Records failed to put in Kinesis Data Stream after all retries. Each PutRecords API call can receive up to 500 records: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/kinesis.html#Kinesis.Client.put_records

misc module

Various utilities useful when working with Kinesis Data Streams.

amazon_kinesis_utils.misc.**dict_get_default**(*dictionary: dict*, *key: str*, *default: any*, *verbose: bool = False*) → Any

Get key from dictionary if key is in dictionary, default value otherwise

Parameters

- **dictionary** – dictionary to retrieve key from
- **key** – key name in dictionary
- **default** – value to return if key is not in dictionary
- **verbose** – output detailed warning message when returning default value

Returns value for key if key is in dictionary, default value otherwise

amazon_kinesis_utils.misc.**split_list**(*lst: list*, *n: int*) → List[list]

Split a list of object in chunks of size n

Parameters

- **lst** – List to split in chunks
- **n** – Size of chunk (last chunk may be less than n)

s3 module

Utilities to save string data to S3 easily.

amazon_kinesis_utils.s3.**put_str_data**(*client*, *bucket: str*, *key: str*, *data: str*, *gzip_compress: bool = False*)

Put str data to S3 bucket with optional gzip compression

Parameters

- **client** – S3 API client (e.g. boto3.client('s3'))
- **bucket** – S3 bucket name
- **key** – S3 object key
- **data** – Data to save
- **gzip_compress** – Boolean switch to control gzip compression (default = False)

Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index