JSON Transporter Documentation

Release 0.2.1

Jason Haas

September 02, 2015

Contents

`tport` is a simple command line tool written in Python for all your JSON transporting needs. Tools currently supported:

- Amazon S3
- Elastic Search
- Kafka
- MongoDB
- HBase

CHAPTER 1

Requirements

- Unix based machine (Linux or OS X)
- Python 2.7.x

This tool has been tested on CentOS 6.5, Ubuntu 14.04, and MacOSX 10.10.5. If you wish to install in a different environment I *highly* recommend installing inside a Python `virtualenv`. Setting up a virtual environment is outside the scope of this document.

CHAPTER 2

Quickstart

1. `pip install json-transporter`
2. Add your specific connections to a **.tport** file in your home directory. For example,

```
[elasticsearch]
host = localhost:9200
ssl = False

[kafka]
broker = localhost:9092

[s3]
access_key = my_access_key
secret_key = my_secret_key

[mongo]
host = localhost
db = test

[hbase]
host = localhost
```

3. To view command line usage just type `tport --help`

CHAPTER 3

Configuration

If `tport` does not find the relevant settings on the command line or the **.tport** file, it will resort to the default settings for each tools. For example,

- `localhost:9200` –> Elastic search
- `localhost:9092` –> Kafka

Connection settings such as the **host** and **db** can also be specified on the command line. Anything specified on the command line will have presedence over settings in files. Order of precedence:

1. command line
2. .tport
3. defaults

3.1 JSON Transporter CLI

`tport` is a one stop shop for shipping JSON around to some commonly used big data tools.

3.2 JSON Transporter API

class `transporter.tools.`**`ElasticPort`**(*host*, *ssl*, *logger=None*)
Bases: `object`

Class to handle Elastic Search actions.

Parameters

- **`host`** (*str*) – elasticsearch host
- **`ssl`** (*bool*) – ssl enable
- **`logger`** (*str*) – name of logger

`create`(*iname*)
Create a new Elastic Search index.

Parameters **`iname`** (*str*) – Index name

`index`(*jsonit*, *iname*, *dtype*, *chunksize=500*)
Data input is a JSON generator. If using the command-line tool, this is handled via the JsonPort method which creates a JSON generator from lines read in from files.

Parameters

- **jsonit** (list) – JSON list or iterator
- **iname** (*str*) – elasticsearch index name
- **dtype** (*str*) – document type
- **chunksize** (*int*) – number of docs to send at one time

map(*iname*, *dtype*, *mapping*)
After creating a new index, specify a mapping.

Parameters

- **iname** (*str*) – Index name
- **dtype** (*str*) – Document type
- **mapping** (*dict*) – Elastic Search mapping

query()
Query Elastic Search

class transporter.tools.**HbasePort**(*hostname*)
Bases: object

Class to interface with HBase.

Parameters **hostname** (*str*) – Hbase hostname

scan(*tablename*)
Print HBase rows one row at a time. Ctrl+d to stop.

Parameters **tablename** (*str*) – Hbase table name

class transporter.tools.**JsonPort**(*jsonlist*, *ignore_errors=False*)
Bases: object

Parses out a JSON iterator object.

Parameters **jsonlist** (list) – a list or iterator of JSON objects.

inspect()
Output the serialized JSON object one line at a time. To continue, press any key. To end, Ctrl+d.

parse()
Returns an JSON iterator object if input is valid JSON, else it returns an empty dictionary.

class transporter.tools.**KafkaPort**(*kafkabroker*, *logger=None*)
Bases: object

Class to interface with Kafka.

Parameters

- **kafkabroker** (*str*) – Kafka broker and port, eg localhost:9092
- **logger** (*str*) – Logger to use for Kafka.

consume(*topic_name*)
Receive data from a Kafka topic.

Parameters **topic_name** (*str*) – Kafka topic name

produce(*topic_name*, *jsonit*)
Send data to a Kafka topic.

Parameters

- **topic_name** (*str*) – Kafka topic name
- **jsonlist** (list) – List or iterator of JSON objects

topics()
List Kafka topics.

class transporter.tools.**MongoPort**(*host*, *db*)
Bases: object

Class to handle interfacing with MongoDB.

Parameters

- **host** (*str*) – Mongo hostname
- **db** (*str*) – Mongo database

add(*collection*)
Add data to a MongoDB collection.

Parameters **collection** (*str*) – Name of Mongo collection or table

export(*collection*, *f='-'*)
Export serailized JSON data from a MongoDB colllection. Not working yet.

Parameters

- **collection** (*str*) – Name of Mongo collection or table
- **f** (*str*) – File to write to

list()
List all collections in the Mongo database.

preview(*collection*)
View a collection line by line. Ctrl+d to stop.

Parameters **collection** (*str*) – Name of Mongo collection or table

class transporter.tools.**S3Port**(*access_key*, *secret_key*)
Bases: object

Class to handle uploading and donwloading data to and from S3. There is also an option to compress the files with gzip before uploading.

Parameters

- **access_key** (*str*) – S3 access key
- **secret_key** (*str*) – S3 secret key

destroy(*bucket_name*)
Destroy an S3 bucket and all data inside it.

Parameters **bucket_name** (*str*) – S3 bucket to destroy

download(*bucket_name*, *folder*)
Download all data in an S3 bucket.

Parameters

- **bucket_name** (*str*) – Name of S3 bucket
- **folder** (*str*) – Folder to store S3 data

list()
List all S3 buckets.

upload(*bucket_name*, *filelist*, *compress=False*, *replace_key=False*)
Given a S3 bucket and a fileglob, upload data to S3. Using the –compress option is recommended to save on S3 costs.

Parameters

- **bucket_name** (*str*) – S3 bucket name
- **filelist** (list) – List of files to upload to S3
- **compress** (*bool*) – Compress the data before uploading
- **replace_key** (*bool*) – Replace existing data

CHAPTER 4

JSON Transporter CLI

Once installed, `tport` takes serialized JSON text files from the command line and transporter it to a supported tool.

CHAPTER 5

JSON Transporter API

If you wish to create your own CLI scripts you can *from transporter import tools* in your code to leverage the JSON transporter API.

t