Ohara Configurator

Public Restful APIs

Only support JSON type. It means both **content type** and **accept type** should be JSON. Even if the configurator encounters an exception, it still returns a response with json body recording the error details.

<table>
<tr><th>API</th><th>request body</th><th>response body</th></tr>
<tr><td colspan="3">Topic
uuid: topic uuid
name: topic name
numberOfPartitions: the number of partition of the topic.
numberOfReplications: the number of replication of topic
lastModified: the last time to modify the topic</td></tr>
<tr><td>GET
/v0/topics

return a list of topic

NOTED: this topic name is NOT equal to the name used in kafka</td><td rowspan="4">N/A</td><td>[
{
"name": "topic_0",
"lastModified": 1540957175158,
"uuid": "d312871a-4a05-488d-aae0-c8b27c5312c2",
"numberOfReplications": 1,
"numberOfPartitions": 1
}
],</td></tr>
<tr><td>GET
/v0/topics/(string: uuid)

get information about a specific topic.</td><td rowspan="4">{
"name": "topic_0",
"lastModified": 1540957175158,
"uuid": "d312871a-4a05-488d-aae0-c8b27c5312c2",
"numberOfReplications": 1,
"numberOfPartitions": 1
}</td></tr>
<tr><td>POST
/v0/topics

create a new topic; the request body should be a JSON object</td><td rowspan="2">{
"name": "topic_0",
"numberOfPartitions": 1,
"numberOfReplications": 1
}</td></tr>
<tr><td>PUT
/v0/topics/(string: uuid)

update the topic. Currently, only name can be changed.

Following operations are invalid:

1) update a non-existed topic

2) decrease number of replications or partitions</td></tr>
<tr><td>DELETE
/v0/topics/(string: uuid)

delete a topic from kafka and ohara.
NOTED: Deleting a topic used by pipeline is disallowed</td><td>N/A</td></tr>
<tr><td colspan="3">HDFS information
uuid: schema uuid
name: schema name
uri: hdfs uri
lastModified: the last time to modify the topic</td></tr>
<tr><td>GET
/v0/hdfs

return a list of hdfs information</td><td rowspan="2">N/A</td><td>[
{
"uuid": "ea343452-0292-4a04-9c7b-2c8872443e93",
"name": "hdfs",
"uri": "file://",
"lastModified": 1540957274892
}
],</td></tr>
<tr><td>GET
/v0/hdfs/(string: uuid)

get information about a specific hdfs information</td><td>{
"uuid": "ea343452-0292-4a04-9c7b-2c8872443e93",
"name": "hdfs",
"uri": "file://",</td></tr>
</table>

POST
/v0/**hdfs**
 	
store a new **hdfs information**; the request body should be a JSON object.
	{
	"name": "hdfs",
	"uri": "file://"
	}
		"lastModified": 1540957274892
		}

PUT /v0/**hdfs**/(string: uuid) update the **hdfs information**. NOTED: It is invalid to update a un-existed schema		
DELETE /v0/**hdfs**/(string: uuid) delete a **hdfs information**	N/A	

FTP information

- **uuid**: ftp info uuid
- **name**: ftp info name
- **hostname:**: ftp hostname
- **port:** ftp port
- **user:** ftp username
- **password:** ftp password
- **lastModified**: the last time to modify

GET
/v0/**ftp**
 	
return a list of **ftp information**
	N/A	[
		{
		"name": "ftp_name",
		"hostname": "localhost",
		"lastModified": 1540969019274,
		"uuid": "1bb64620-6f2d-40a9-bb81-10345a504d63",
		"port": 12314,
		"user": "user",
		"password": "123"
		}
		],

GET
/v0/**ftp**/(string: uuid)
 	
get information about a specific **ftp information**
		{
		"name": "ftp_name",
		"hostname": "localhost",
		"lastModified": 1540969019274,
		"uuid": "1bb64620-6f2d-40a9-bb81-10345a504d63",
		"port": 12314,
		"user": "user",
		"password": "123"
		}

POST
/v0/**ftp**
 	
store a new **ftp information**; the request body should be a JSON object.
 	
NOTED: when port is empty ,using default no.
	{
	"name": "ftp_name",
	"hostname": "localhost",
	"port": 12314,
	"user": "user",
	"password": "123"
	}	

PUT /v0/**ftp**/(string: uuid) update the **ftp information**.		
DELETE /v0/**ftp**/(string: uuid) delete a **ftp information**	N/A	

JDBC information

- **uuid**: jdbc info uuid
- **name**: jdbc info name
- **url**: jdbc url
- **user:** ftp username
- **password:** ftp password
- **lastModified**: the last time to modify

GET /v0/**jdbc** return a list of **jdbc information**	N/A	[{ "name": "jdbc_name", "url": "jdbc:mysql", "lastModified": 1540967970407, "uuid": "9d128f43-8725-42b2-9377-0dad10863166",

		"user": "user", "password": "aaa" }],
GET /v0/**jdbc**/(string: uuid) get information about a specific **jdbc information**		{ "name": "jdbc_name", "url": "jdbc:mysql", "lastModified": 1540967970407, "uuid": "9d128f43-8725-42b2-9377-0dad10863166", "user": "user", "password": "aaa" }
POST /v0/**jdbc** store a new **jdbc information**; the request body should be a JSON object.	{ "name": "jdbc_name", "url": "jdbc:mysql", "user": "user", "password": "aaa" }	
PUT /v0/**jdbc**/(string: uuid) update the **jdbc information**.		
DELETE /v0/**jdbc**/(string: uuid) delete a **jdbc information**	N/A	

Pipeline

- **uuid:** pipeline uuid
- **name:** pipeline name
- **rules (optional):** from -> to
- **objects:** abstraction of objects in rules
  - **uuid**: object's uuid
  - **kind**: object's kind
  - **name**: object's name
  - **state (optional):** UNASSIGNED, RUNNING, PAUSED, FAILED, DESTROYED

GET
/v0/**pipelines**
 	
return a list of **pipelines information**
	N/A	[
		{
		"name": "pipeline",
		"lastModified": 1540974448834,
		"uuid": "d2f16ef1-a726-4ca4-86c2-be2d9cb36d17",
		"rules": {
		"d3e46fad-9bf3-4434-8d09-5791c10ecb0f": "4ef3664e-361c-47c0-87f2-79a3ee8ddafc"
		},
		"objects": [
		{
		"uuid": "4ef3664e-361c-47c0-87f2-79a3ee8ddafc",
		"name": "topic_0",
		"kind": "topic"
		},
		{
		"uuid": "d3e46fad-9bf3-4434-8d09-5791c10ecb0f",
		"name": "perf",
		"kind": "com.island.ohara.connector.perf.PerfSource"
		}
		],
		}
		],

GET /v0/**pipelines**/(string: uuid)		{ "name": "pipeline",

<table>
<tr>
<td>get pipeline information</td>
<td></td>
<td rowspan="3">"lastModified": 1540974448834,
"uuid": "d2f16ef1-a726-4ca4-86c2-be2d9cb36d17",
"rules": {
"d3e46fad-9bf3-4434-8d09-5791c10ecb0f" : "4ef3664e-361c-47c0-87f2-79a3ee8ddafc"
},
"objects": [
{
"uuid": "4ef3664e-361c-47c0-87f2-79a3ee8ddafc",
"name": "topic_0",
"kind": "topic"
},
{
"uuid": "d3e46fad-9bf3-4434-8d09-5791c10ecb0f",
"name": "perf",
"kind": "com.island.ohara.connector.perf.PerfSource"
}
],
}</td>
</tr>
<tr>
<td>POST
/v0/pipelines

store a new pipeline information; the request body should be a JSON object.
NOTED: if the (from, to) isn’t completed, it accepts to use “?” to replace the “to” in order to form the valid rules</td>
<td rowspan="2">{
"name": "pipeline",
"rules": {
"d3e46fad-9bf3-4434-8d09-5791c10ecb0f": "4ef3664e-361c-47c0-87f2-79a3ee8ddafc"
}
}</td>
</tr>
<tr>
<td>PUT
/v0/pipelines/(string: uuid)

update the hdfs information.

NOTED: if the (from, to) isn’t completed, it accepts to use “?” to replace the “to” in order to form the valid rules</td>
</tr>
<tr>
<td>DELETE
/v0/pipelines/(string: uuid)

delete a hdfs information
NOTED: Deleting a used schema is disallowed</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td colspan="3">Source
● name => source name
● className => source’s class name
The alias mapping of official source:
○ com.island.ohara.connector.jdbc.JDBCSourceConnector -> jdbc
○ com.island.ohara.connector.ftp.FtpSource -> ftp
● schema => row schema
○ name: original column name
○ newName (optional): final column name
○ dataType: column type
○ order: column index in row
● configs => source configuration
see FTP Source Arguments JDBC Source Arguments Perf Source Arguments
● state (optional): UNASSIGNED, RUNNING, PAUSED, FAILED, DESTROYED
● topics
● numberOfTasks</td>
</tr>
<tr>
<td>GET
/v0/sources

return a list of source</td>
<td>N/A</td>
<td>[
{
"topics": [
"a401f1d5-46a1-42e5-9999-022355693b54"
],
"name": "perf",
"lastModified": 1540969182646,
"uuid": "66b398cb-8991-4888-82b8-060356f61171",
"configs": {
"perf.batch": "1",
"perf.frequence": "2 seconds"
},
"numberOfTasks": 1,
"schema": [
{
"name": "cf0",
"newName": "cf0",
"dataType": "integer",</td>
</tr>
</table>

		"order": 1
		},
		{
		"name": "cf1",
		"newName": "cf1",
		"dataType": "boolean",
		"order": 2
		}
		],
		"className": "com.island.ohara.connector.perf.PerfSource"
		}
		],

GET
/v0/**sources**/(string: uuid)
 	
get information about a specific **source.**
		{
		"topics": [
		"a401f1d5-46a1-42e5-9999-022355693b54"
		],
		"name": "perf",
		"lastModified": 1540969182646,
		"uuid": "66b398cb-8991-4888-82b8-060356f61171",
		"configs": {
		"perf.batch": "1",
		"perf.frequence": "2 seconds"
		},
		"numberOfTasks": 1,
		"schema": [
		{
		"name": "cf0",
		"newName": "cf0",
		"dataType": "integer",
		"order": 1
		},
		{
		"name": "cf1",
		"newName": "cf1",
		"dataType": "boolean",
		"order": 2
		}
		],
		"className": "com.island.ohara.connector.perf.PerfSource"
		}

POST
/v0/**sources**
 	
create a new **source**; the request body should be a JSON object
	{
	"name": "perf",
	"className": "com.island.ohara.connector.perf.PerfSource",
	"topics": [
	"a401f1d5-46a1-42e5-9999-022355693b54"
	],
	"numberOfTasks": 1,
	"configs": {
	"perf.batch": "1",
	"perf.frequence": "2 seconds"
	},
	"schema": [
	{
	"name": "cf0",
	"newName": "cf0",
	"dataType": "integer",
	"order": 1
	},
	{
	"name": "cf1",
	"newName": "cf1",
	"dataType": "boolean",
	"order": 2
	}
	]
	}	

PUT /v0/**sources**/(string: uuid) update the source. NOTED: It is invalid to update a un-existed source NOTED: Updating a source is used by a running pipeline is disallowed		
DELETE /v0/**sources**/(string: uuid) delete a source NOTED: Deleting a source used by pipeline is disallowed	N/A	
PUT /v0/**sources**/(string: uuid)/**start** run the source by kafka worker.		OK
PUT /v0/**sources**/(string: uuid)/**pause** pause the source. NOTED: the connector is still running. this command only “puase” the data convertion.		
PUT /v0/**sources**/(string: uuid)/**resume** resume the paused source.		
PUT /v0/**sources**/(string: uuid)/**stop**		

stop and remove the connector from kafka worker		

Sink

- **name** => source name
- **className** => sink's class name
  The alias mapping of official source:
  - com.island.ohara.connector.hdfs.HDFSSinkConnector -> hdfs
  - com.island.ohara.connector.ftp.FtpSink -> ftp
- **schema =>** row schema
  - **name**: original column name
  - **newName (optional)**: final column name
  - **dataType**: column type
  - **order**: column index in row
- **configs** => sink configuration
  see FTP Sink Arguments HDFS Sink Arguments
- **state (optional):** UNASSIGNED, RUNNING, PAUSED, FAILED, DESTROYED
- **topics**
- **numberOfTasks**

GET
/v0/**sinks**
 	
return a list of **sink**
	N/A	[
		{
		"topics": [
		"a401f1d5-46a1-42e5-9999-022355693b54"
		],
		"name": "ftp",
		"lastModified": 1540969301809,
		"uuid": "147dd6f2-dc5e-4538-8cb7-fdcdb785a17d",
		"configs": {
		"ftp.output.folder": "/output",
		"ftp.user.name": "user-0",
		"ftp.port": "12353",
		"ftp.needHeader": "true",
		"ftp.user.password": "password-1",
		"ftp.hostname": "backend"
		},
		"numberOfTasks": 1,
		"schema": [
		{
		"name": "cf0",
		"newName": "cf0",
		"dataType": "integer",
		"order": 1
		},
		{
		"name": "cf1",
		"newName": "cf1",
		"dataType": "boolean",
		"order": 2
		}
		],
		"className": "com.island.ohara.connector.ftp.FtpSink"
		}
		],

GET /v0/**sinks**/(string: uuid) get information about a specific **sink.**		{ "topics": ["a401f1d5-46a1-42e5-9999-022355693b54"],
POST	{	

<table>
<tr>
<td>/v0/sinks

create a new sink; the request body should be a JSON object

PUT
/v0/sinks/(string: uuid)

update the sink.
NOTED: It is invalid to update a un-existed sink
NOTED: Updating a sink is used by a running pipeline is disallowed</td>
<td><pre>{
  "name": "ftp",
  "className":
"com.island.ohara.connector.ftp.FtpSink",
  "topics": [
    "a401f1d5-46a1-42e5-9999-022355693b54"
  ],
  "numberOfTasks": 1,
  "configs": {
    "ftp.output.folder": "/output",
    "ftp.hostname": "backend",
    "ftp.port": "12353",
    "ftp.needHeader": "true",
    "ftp.user.name": "user-0",
    "ftp.user.password": "password-1"
  },
  "schema": [
    {
      "name": "cf0",
      "newName": "cf0",
      "dataType": "integer",
      "order": 1
    },
    {
      "name": "cf1",
      "newName": "cf1",
      "dataType": "boolean",
      "order": 2
    }
  ]
}</pre></td>
<td><pre>{
  "name": "ftp",
  "lastModified": 1540969301809,
  "uuid":
"147dd6f2-dc5e-4538-8cb7-fdcdb785a17d",
  "configs": {
    "ftp.output.folder": "/output",
    "ftp.user.name": "user-0",
    "ftp.port": "12353",
    "ftp.needHeader": "true",
    "ftp.user.password": "password-1",
    "ftp.hostname": "backend"
  },
  "numberOfTasks": 1,
  "schema": [
    {
      "name": "cf0",
      "newName": "cf0",
      "dataType": "integer",
      "order": 1
    },
    {
      "name": "cf1",
      "newName": "cf1",
      "dataType": "boolean",
      "order": 2
    }
  ],
  "className":
"com.island.ohara.connector.ftp.FtpSink"
}</pre></td>
</tr>
<tr>
<td>DELETE
/v0/sinks/(string: uuid)

delete a sink
NOTED: Deleting a source used by pipeline is disallowed</td>
<td rowspan="5">N/A</td>
<td rowspan="5">N/A</td>
</tr>
<tr>
<td>PUT
/v0/sinks/(string: uuid)/start

run the sink by kafka worker.</td>
</tr>
<tr>
<td>PUT
/v0/sinks/(string: uuid)/pause

pause the sink.

NOTED: the connector is still running. this command only “puase” the data convertion.</td>
</tr>
<tr>
<td>PUT
/v0/sinks/(string: uuid)/resume

resume the paused sink.</td>
</tr>
<tr>
<td>PUT
/v0/sinks/(string: uuid)/stop

stop and remove the connector from kafka worker</td>
</tr>
<tr>
<td colspan="3">Validate

● hostname: name of worker node
● message: something hint
● pass: true if it works</td>
</tr>
<tr>
<td>PUT
/v0/validate/hdfs

test the hdfs connection

NOTED: configurator will send the validation request to all workers to run the connection test</td>
<td><pre>{
  "uri": "file://"
}</pre></td>
<td rowspan="2"><pre>[
  {
    "hostname": "chia7712-island",
    "message": "a fake report",
    "pass": true
  },
  {
    "hostname": "chia7712-island",
    "message": "a fake report",
    "pass": true</pre></td>
</tr>
<tr>
<td>PUT
/v0/validate/rdb

test the database connection

NOTED: configurator will send the validation request to all workers to</td>
<td><pre>{
  "url": "jdbc:mysql",
  "user": "user",
  "password": "password"
}</pre></td>
</tr>
</table>

run the connection test		}, { "hostname": "chia7712-island", "message": "a fake report", "pass": true }],
PUT /v0/**validate/ftp** test the ftp connection NOTED: configurator will send the validation request to all workers to run the connection test	{ "hostname": "localshot", "port": 12345, "user": "user", "password": "password" }	

Cluster

- **brokers:** brokers information
- **workers:** workers information
- **sources:** source plugin information
  - **className**: class's full name
  - **version**: release version
  - **revision**: last commit
- **sinks:** sink plugin information
  - **className**: class's full name
  - **version**: release version
  - **revision**: last commit
- **supportedDatabases**: supported database in ohara
- **supportedTypes**: supported data types in ohara
- **versionInfo:**
  - **version:** release version
  - **user:** the man who build this world
  - **revision:** last commit
  - **date:** the build date

GET /v0/cluster get the cluster information	N/A	{ "workers": "chia7712-island:12349,chia7712-island:12350,chia7712-island:12351", "sinks": [{ "className": "com.island.ohara.connector.ftp.FtpSink", "version": "0.1-SNAPSHOT", "revision": "8acc46fa59ad926dea11a4b63a940bbbc90ed2ad" }, { "className": "com.island.ohara.connector.hdfs.HDFSSinkConnector", "version": "0.1-SNAPSHOT", "revision": "8acc46fa59ad926dea11a4b63a940bbbc90ed2ad" }, { "className": "org.apache.kafka.connect.file.FileStreamSinkConnector", "version": "1.0.1", "revision": "unknown" }], "brokers": "chia7712-island:12346,chia7712-island:12347,chia7712-island:12348", "supportedDatabases": ["mysql"], "supportedDataTypes": ["byte array", "boolean", "byte", "short",

```
        "integer",
        "long",
        "float",
        "double",
        "string",
        "object",
        "row"
    ],
    "sources": [
        {
            "className":
            "com.island.ohara.configurator.endpoint
            .Validator",
            "version": "0.1-SNAPSHOT",
            "revision":
            "8acc46fa59ad926dea11a4b63a940bbb
            c90ed2ad"
        },
        {
            "className":
            "com.island.ohara.connector.ftp.FtpSou
            rce",
            "version": "0.1-SNAPSHOT",
            "revision":
            "8acc46fa59ad926dea11a4b63a940bbb
            c90ed2ad"
        },
        {
            "className":
            "com.island.ohara.connector.jdbc.JDBC
            SourceConnector",
            "version": "0.1-SNAPSHOT",
            "revision":
            "8acc46fa59ad926dea11a4b63a940bbb
            c90ed2ad"
        },
        {
            "className":
            "com.island.ohara.connector.perf.PerfS
            ource",
            "version": "0.1-SNAPSHOT",
            "revision":
            "8acc46fa59ad926dea11a4b63a940bbb
            c90ed2ad"
        },
        {
            "className":
            "org.apache.kafka.connect.file.FileStrea
            mSourceConnector",
            "version": "1.0.1",
            "revision": "unknown"
        }
    ],
    "versionInfo": {
        "version": "0.1-SNAPSHOT",
        "user": "Chia-Ping Tsai",
        "revision":
        "8acc46fa59ad926dea11a4b63a940bbbc90
        ed2ad",
        "date": "五 11月 2 16:50:18 CST 2018"
    }
}
```


Query

- **url**: jdbc url
- **user**
- **password**
- **catalogPattern**: a catalog name

- **schemaPattern**: a schema name pattern
- **name**: a table name pattern
- **name**: target name
- **tables**: table information
  - **catalogPattern:** table caclog
  - **schemaPattern**: table schema
  - **name**: table name
  - **schema** => table columns
    - **name** => column name
    - **type** => column type
    - **pk** => true if this column is primary key

POST
/v0/query/rdb

list tables of database

You can obtain the correct url, user and password from mini cluster backend.log. The url port is dynamic so whenever you restart the mini cluster the url will be different

```
{
  "url":
"jdbc:mysql://chia7712-island:38351/db-2",
  "user": "user-0",
  "password": "password-1"
}
```

```
{
    "name": "mysql",
    "tables": [
        {
            "catalogPattern": "db-2",
            "name": "table",
            "schema": [
                {
                    "name": "cf",
                    "dataType": "INT",
                    "pk": true
                },
                {
                    "name": "cf2",
                    "dataType": "BIT",
                    "pk": false
                }
            ],
        }
    ],
}
```

Error

- **message** => error description
- **code** => ohara's error code
- **stack** => error stack tree

NOTED: all requests may encounter the error (server's error, error format, and so on)

```
{
    "code": "java.lang.IllegalArgumentException",
    "message": "Unsupported restful api:vasdasd. Or the request is invalid to the vasdasd",
    "stack": "java.lang.IllegalArgumentException: Unsupported restful api:vasdasd. Or the request is invalid to the vasdasd at
    com.island.ohara.configurator.Configurator$$anonfun$4.apply(Configurator.scala:79) at
    com.island.ohara.configurator.Configurator$$anonfun$4.apply(Configurator.scala:78) at
    akka.http.scaladsl.server.util.ApplyConverterInstances$$anon$1$$anonfun$apply$1.apply(ApplyConverterInstances.scala:14) at
    akka.http.scaladsl.server.util.ApplyConverterInstances$$anon$1$$anonfun$apply$1.apply(ApplyConverterInstances.scala:13) at
}
```

Private Restful APIs

a way to access internal services...just for testing!!!

API	request body	response body

Creation
• **name:** table name
• **schema:**
  ○ **name:** column name
  ○ **dataType:** column type
  ○ **pk:** true if this column is primary key		

<table>
<tr>
<td>POST
/_private/creation/rdb

create a rdb table

Note that there’s no /v0/ before the API endpoint</td>
<td><pre>{
 "name": "table",
 "schema": [
  {
   "name": "cf",
   "dataType": "integer",
   "pk": true
  },
  {
   "name": "cf2",
   "dataType": "boolean",
   "pk": false
  }
 ]
}</pre></td>
<td>N/A</td>
</tr>
<tr>
<td colspan="3">Services

● database:
○ url: jdbc url
○ user: user name
○ password: user’s password
● ftpServer:
○ host: host address
○ port: listene port
○ user: user name
○ password: user’s password</td>
</tr>
<tr>
<td>GET
/_private/services

retrieve the information of embedded services

Note that there’s no /v0/ before the API endpoint</td>
<td>N/A</td>
<td><pre>{
    "ftpServer": {
        "host": "chia7712-island",
        "dataPort": 43667,
        "port": 34053,
        "user": "user-0",
        "password": "password-1"
    },
    "database": {
        "url":
        "jdbc:mysql://chia7712-island:35129/db-2
        ",
        "user": "user-0",
        "password": "password-1"
    },
    "workers":
    "chia7712-island:36469,chia7712-island:3301
    9,chia7712-island:34051",
    "brokers":
    "chia7712-island:33635,chia7712-island:3769
    9,chia7712-island:44351",
    "zookeeper": "chia7712-island:44611"
}</pre></td>
</tr>
</table>

FTP Connector

FTP Source

The FTP source is used to read csv file from FTP server and convert the csv data to ohara row to put into kafka topic. User can define the FTP folder(s) as input. FTP source will assign different “hash code” to each FTP task. The hash code is used to indicate which files should be handled by task. The hash code of a file is generated by file name.

Recovery

FTP task log the offset of file in kafka topic (powered by kafka connector). Hence, in recovery phase FTP task can retrieve all offsets. The offsets can help FTP task to avoid duplicate data. For example, there is a csv file (called file_a) having 100 lines. After FTP task have completed 50 lines, the connector crash. The FTP task is recovered by other connector node. The restarted FTP task will retrieve the offsets of file_a - offsets = 51 - so FTP task will skip 1~50 lines.

POST commit

Users have to define a FTP folder (called backup_folder) storing all “completed” files. When all lines of a file is copied to kafka topic, FTP task will move the file to the backed_up folded so as to prevent read the file again.

Input FIle

The input files must be csv file and the header is required. For example:

```
name,number,good
abc,1,false
ab,2,true
```

Above csv file has a header defining 3 columns - name, number and good. And there are two records - abc,1, false” and “ab,2,true”.

FTP Source Arguments

FTP connector extends ohara’s Row connector so there are some inner argument (starting with __). Instead of invoking FTP connector by kafka interface, using ConnectorClient is more graceful and friendly.

key	description	default	type
ftp.input.folder	files source. example, /a,/b,/c	NONE	string
ftp.completed.folder	this folder is used to store the completed files	NONE	string
ftp.error.folder	the folder is used to store the files can’t be handled by FTP source.	NONE	string
ftp.encode	used to parse the file	UTF-8	string
ftp.hostname	fpt host	NONE	string
ftp.port	ftp port	NONE	Int
ftp.user.name	user name	NONE	string
ftp.user.password	password	NONE	string

FTP Sink

FTP sink connector serves in converting row from topic to csv file in FTP server.

FTP Sink Arguments

FTP connector extend ohara’s Row connector so there are some inner argument (starting with __). Instead of invoking FTP connector by kafka interface, using ConnectorClient is more graceful and friendly.

key	description	default	type
ftp.completed.folder	this folder is used to store the completed files	NONE	string
ftp.encode	used to parse the file	UTF-8	string
ftp.hostname	fpt host	NONE	string
ftp.port	ftp port	NONE	Int

ftp.user.name	user name	NONE	string
ftp.user.password	password	NONE	string
ftp.needHeader	true if output csv file should have header	NONE	boolean

HDFS Connector

HDFS Sink

系統架構

Ohara HDFS Sink Connector 會將 Kafka Broker 上面的 Topic 資料 pull 下來，之後會把 pull 下來的資料寫入到 HDFS Storage 裡。

資料儲存的資料夾結構設計

Ohara HDFS Sink Connector 主要的設計是會先把 Kafka Topic 裡面每一筆 message 的資料寫入到 HDFS 上的 tmp 資料夾裡面的檔案裡，等到到達了 flush 的條件時，就會將這些檔案移動到 data 資料夾並且將 offset 的資訊寫入到 檔案名稱裡。紀錄 offset 的目的主要是，當 HDFS Sink Connector 執行 Task 的 rebalance 時或是重新啟動 HDFS Sink Connector 時，可以讓 Task 知道需要從 offset 的哪個位置上繼續執行取資料的動作。

```
<root>
  |----- tmp ------|<TopicName> |-------partition0--------------| ${timestamp}.csv
  |                             |-------partition1--------------| ${timestamp}.csv
  |                             |-------partition2--------------| ${timestamp}.csv
  |
  |
  |
  |
  |----- data ------|<TopicName> |-------partition0----------------| part-000000000-000000099.csv
  |                                                               | part-000000100-000000199.csv
  |                                                               | part-000000200-000000299.csv
  |
  |                             |-------partition1----------------| part-000000000-000000099.csv
  |                                                               | part-000000100-000000199.csv
  |                                                               | part-000000200-000000299.csv
  |
  |                             |-------partition2----------------| part-000000000-000000099.csv
  |                                                               | part-000000100-000000199.csv
  |                                                               | part-000000200-000000299.csv
```

資料夾和檔案說明

tmp：HDFS Sink Connect 在取資料時會先寫入到 tmp 資料夾下面的一個暫存檔裡面，檔案名稱使用 timestamp 的方式 建立，等到達了 flush 檔案的條件時，會將此檔案移動到 data 資料夾下面。另外檔案內容是每一個 message 為一條的 line，檔案格式為 csv 的格式。

data：當達到 flush 資料的條件時，會將 tmp 資料夾下面的檔案移動到此資料夾下面。offset 的資訊存放在檔案名稱裡，檔案名稱的格式如下：
part-000000000-000000099.csv
這裡的 part 是 file prefix name，000000000-000000099 代表了 pull 的 offset 位置是從 0 到 99

資料儲存邏輯

HDFS Sink Connector 的 Task，把資料寫入到 HDFS 的 Flow 為 **start -> open -> put -> close -> stop** 每一個階段要做的事說明如下：

1. **start：**取得設定檔的的內容，並且建立 Hadoop 的 Configuration 物件，用來存取 HDFS 時使用，並且會建立 HDFS 的 FileSystem 物件。

2. **open：**收集此 Task 需要處理 Topic 的 Partition 有哪些。並且將 offset 的位置 Recover 到正確的位置上，offset 的位置由 data dir 的檔案名稱來決定。

3. **put：**把取到的資料寫入到 HDFS 裡的 tmp 資料夾下面的檔案，等到達到 flush 資料的條件時會將此檔案移動到 data 資料夾裡面。

4. **close：**當 HDFS Sink Connect 執行 rebalance 或是 Delete Connector 時會呼叫此方法，主要會先把正在寫入到 HDFS 的 tmp 資料先 close 並且把 tmp 的檔案刪除。

5. **stop：**當要關閉 HDFS Sink Connect 時會呼叫到此方法，需要把 HDFS 的 FileSystem 物件關閉。FileSystem 的呼叫方法使用 newInstance(url, conf)，避免再close FileSystem 影響到其它的 Task 執行。

flsuh 資料的條件定義

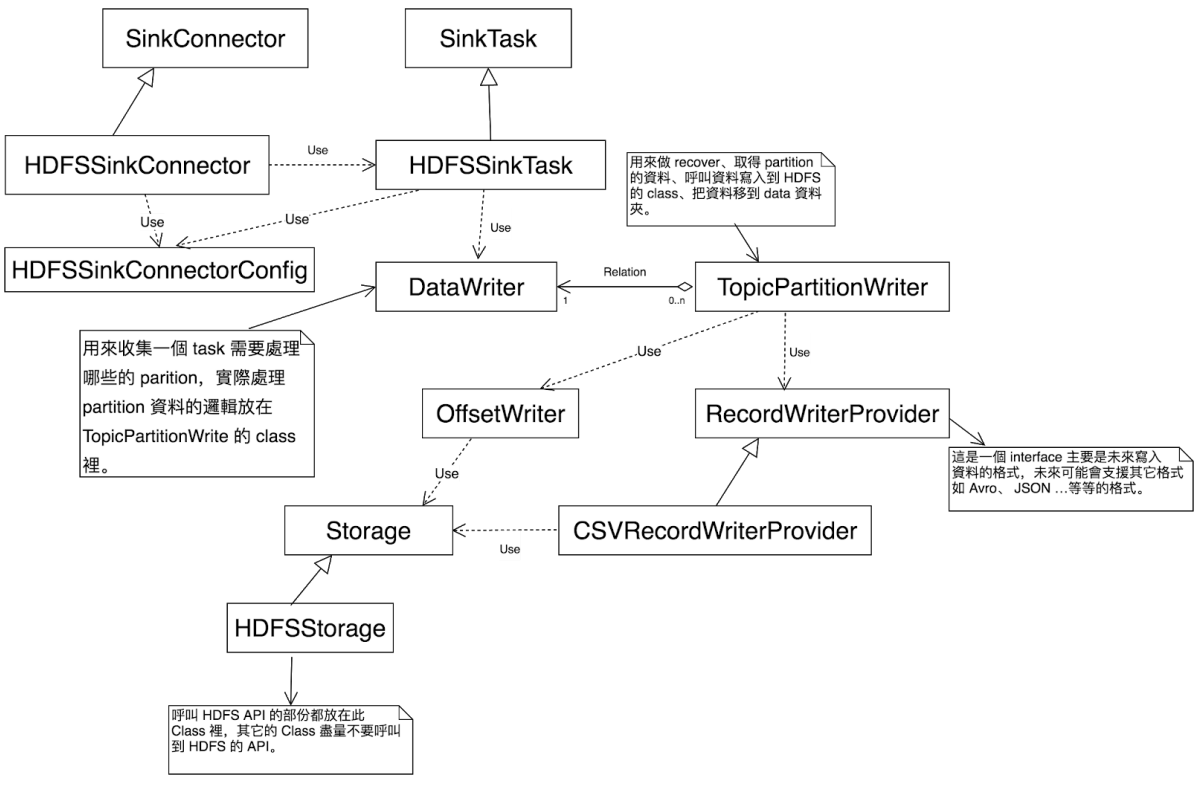
1. 寫入到 tmp 檔案的筆數已經到達了設定的 flush size 資料筆數
   Example：
   flush.line.count=100
   tmp file 已經寫入 100 筆資料，此時會執行 flush 的動作，把資料從 tmp 檔案移到 data 資料夾下面

2. 當寫入到 tmp 檔案的筆數還沒有到達設定的 flush size 資料筆數，但是已經到達了設定的 flush 資料的時間，也會執行 flsuh 的動作
   Example：
   flush.line.count=100
   rotate.interval.ms=120000 ms
   tmp file 已經寫入50 筆資料，此時 flush 資料筆數還沒到達 100 筆資料。但是執行 flush 的時間 2 分鐘已經到了，會執行 flush 的動作，把資料從 tmp file 移到 data 資料夾下面

HDFS Sink Arguments

Key	Description	Default	Type
hdfs.url	設定 HDFS 的 NameNode 的 URL	NONE	string
flush.line.count	設定當寫入到 HDFS tmp 檔案的資料筆數等於 flush.line.count，就會把此 tmp 檔案移動到 data 資料夾下面	1000	string
rotate.interval.ms	定期將資料 flush 到 data 資料夾裡面的時間設定，此設定是使用毫秒為單位	60000	string
tmp.dir	設定 HDFS Sink Connect 的 tmp資料夾要存放在 HDFS 上的哪個路徑裡	/tmp	string
data.dir	設定 HDFS Sink Connect 的 data 資料夾要存放在 HDFS 上的哪個路徑裡	/data	string
datafile.prefix.name	當把 tmp 的檔案 flush 到 data 資料夾時檔案名稱開頭的字串設定	part	string
datafile.needheader	寫入到 HDFS 的資料是否需要 header	true	string
data.encode	寫入到 HDFS 的檔案編碼格式	UTF-8	string

Class Diagram 設計



JDBC Connector

JDBC Source

關聯式資料庫的資料透過 JDBC Source Connector 把資料寫入到 Kafka Topic 裡。JDBC Source 是透過使用 JDBC Driver 連到 RDB 的 Server 上並且使用 SQL 語法查詢哪些的資料需要被寫入到 Kafka Topic 裡。目前 Ohara JDBC Source Connector 支援的關聯式資料庫主要有 **MySQL** 和 **Oracle DB**。

使用 Ohara JDBC Source Connector 前需要先確認 DataBase 裡面的 Table 是否有 Timestamp 的欄位，這個欄位主要的目的是用來查詢哪些資料是最新 Insert 進來的資料，目前還沒被寫入到 Kafka 的 Topic 內，當這個時間的資料寫入到 Kafka Topic 裡此時間的資訊也就會被當成是 offset，成為下一次 SQL 要查詢的條件。

Query Data 的 SQL 語法

目前只支援 Timestmap mode

1. Timestamp mode 的查詢 SQL 語法如下：

    ```
    SELECT *
    FROM TABLE
    WHERE TIMESTAMP_COLUMN_NAME > TS_OFFSET AND
            TIMESTAMP_COLUMN_NAME < CURRENT_TIME
    ```

    第一次執行此 Table 的資料 TS_OFFSET 的初始值為 1970-01-01 00:00:00，之後有查到資料就會將 TIMESTAMP_COLUMN_NAME 的時間資訊寫入到 TS_OFFSET 內

    TS_OFFSET 指的是 timestamp offset

時區設定

此程式使用的時區為抓系統的時間 System.getProperty(“user.timezone”)，如果有特殊需求需要修改時區設定的話，可以在啟動 Worker 修改時區的設定，如下指令：

```
$ java -Duser.timezone=UTC org.apache.kafka.connect.cli.ConnectDistributed
```

RDB Table 的資料轉換

1. JDBC Source Connector 使用 JDBC 取得資料會得到一個 ResultSet 的資料集，此資料集需要透過對應的 RDB Table MetaData 轉成正確的資料格式，舉例如下：
   如果有一個 Table 裡面有 COLUMN1 的欄位它的資料型態為 String，因此再使用 RestulSet 時就會去呼叫 result.getString(“column1”)，這樣才能把正確的值取出來

2. 我們要把資料塞入到 RowSourceRecord 之前需要將上一個步驟取出來的資料
   轉換成 Ohara Schema 對應的資料型態，如果資料型態無法轉換就會丟出
   RuntimeException

Offset 的寫入和讀取

Ohara JDBC Source Connector 會把 Offset 的紀錄直接寫入到 Kafka Topic 內，目前的 offset 的內容主要的資料為 Timestamp 時間資訊。

1. offset 的寫入方式，如下
   new RowSourceRecord(sourcePartition, offset.toMap(), topic, partition, row, timestamp)

2. Offset 的讀取方式，如下
   context.offsetStorageReader.offsets(partitions)

JDBC Source Arguments

Key	Description	Default	Type
source.db.url	設定連線 JDBC DataBase 的 URL 路徑	NONE	string
source.db.username	設定登入 DataBase 的使用者名稱	NONE	string
source.db.password	設定登入 DataBase 的密碼	NONE	string
source.table.name	設定要使用哪張資料表把資料寫入到 Kafka Topic 裡	NONE	string
source.schema.catalog	設定查詢 catalog 時所要使用的參數	(optional)	string

		NULL	
source.schema.pattern	設定查詢 Schema 時所要使用的參數	(optional) NULL	string
mode	設定查詢資料的模式，目前只支援 Timestamp	timestamp	string
source.timestamp.column.name	設定使用哪個 Timestamp 的欄位，當作 offset 的查詢條件	NONE	string

Perf Connector

Perf Source

Perf source connector is used to generate data automatically. The data is created based on scheam. For example, a schema with column_a(int) and column_b(double) is mapped to data (column_a + random int and column_b + random double).

Perf Source Arguments

key	description	default	type
perf.batch	batch size. If value of batch is 2, the number of rows sent to kafka is 2.	NONE	int
perf.frequence	how fast to generate data	NONE	string ex. 5 seconds