

Evolution Logging Service

Version 1



August 12, 2024

Veda Taiwan

Contents

[Evolution Log Service 2](#_Toc185423493)

[Main Components 2](#_Toc185423494)

[Pipeline Architecture 2](#_Toc185423495)

[Components Explanation 3](#_Toc185423496)

[Setup and Installation 4](#_Toc185423497)

[Current Service 6](#_Toc185423498)

[Error Handling and Troubleshooting 7](#_Toc185423499)

[Node.js: Duplicated data Handling 10](#_Toc185423500)

[MongoDB 11](#_Toc185423501)

[Veda Service APIs 12](#_Toc185423502)

[Evolution Logs – by date range 12](#_Toc185423503)

[Evolution Logs – by playerGameId 12](#_Toc185423504)

[RedEnvelopeAPI – playerGameId list 12](#_Toc185423505)

[Evolution Log Service (KR) 14](#_Toc185423506)

[서비스 구성 요소 14](#_Toc185423507)

[데이터 흐름 설명 14](#_Toc185423508)

[구성 요소 설명 15](#_Toc185423509)

[설치 및 설정 16](#_Toc185423510)

[현재 서비스 18](#_Toc185423511)

[오류 처리 및 문제 해결 20](#_Toc185423512)

# Evolution Log Service

This service has used Node.js with Kafka and MongoDB to achieve the high-throughput, fast-speed processing data pipeline.

## Main Components

* Zookeeper
* Kafka
* Kafka Connect
* Node.js
* MongoDB

## Pipeline Architecture

The processing flow begins by requesting data from the Evolution-provided API for logging services. The data from the API is delivered in chunks, each containing large, nested JSON objects. Occasionally, the end of a chunk may contain incomplete JSON data, necessitating further processing. Node.js plays a crucial role in storing this data in a buffer and waiting for subsequent chunks to complete the JSON data. Once the data is complete, it is sent to the Kafka topic "*evolog-topic*" for queuing and distributed storage via Kafka Connect. Kafka Connect, equipped with a Sink Connector, automatically saves the JSON data into the database collection.

## Components Explanation

|  |  |
| --- | --- |
| Node.js | Purpose   * Request Evolution Logging API * Processing incomplete JSON in data chunk * Retry requesting API upon stream error, ended, timeout... * Batch Sending to Kafka Topic (evolog-topic) as producer |
| Kafka | Purpose   * Queuing large sets of messages * Distributed broker of processing logs * High throughput * Fault-tolerance |
| Kafka-Connect | Purpose   * Automated sinking data into MongoDB |
| MongoDB | Purpose   * TTL automatic removing expired data |

## Setup and Installation

**Prerequisites**

Before setting up the service, ensure that the following dependencies are installed on the system:

* **Node.js** (Version 16.x or above)
* **Kafka** (Version 2.5.1 or above)
* **MongoDB** (Version 5.x or above)
* **Kafka Connect** with the necessary connectors
  + (MongoDB Sink Connector)
  + Timestamp Converter

#### Zookeeper Configuration

1. Download zookeeper from official site.
2. Check the conf folder
3. Rename the sample\_zoo.cfg to zoo.cfg
4. Update the log\_dir (necessary)
5. Change other config if needed such as port. By default, 2128.

\*\* For this service is using port 2128.

#### Kafka Configuration

1. Download Kafka from the official site.
2. Redirect to ‘*config’* folder search for server.properties
3. Update the broker configuration as wanted.
4. Run the service through

bin/kafka-server-start.sh config/server.properties

#### MongoDB

As the Sink connector has issue with MongoDB from CosmosDB, a genuine DB is needed in this project.

#### Kafka Connect Configuration

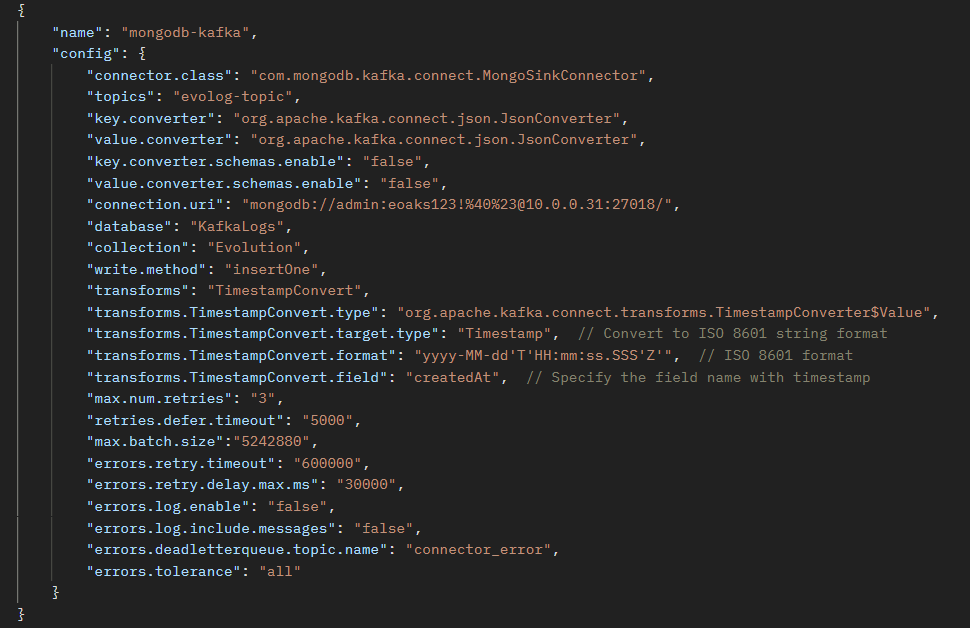
For evolution logs, it is using Kafka connect in distributed mode.

1. Search file for ‘*config/connect-distributed.properties’*
2. Update the connect configuration as wanted.
3. Run the Kafka-connect service through

bin/console-distributed.sh config/console-distributed.properties

[ Kafka Connector Configuration]

1. Start the Kafka Connect service
2. Send POST request to ‘http://{Domain or IP address}:8083/connectors’
3. With below configuration



**Important points:**

Must run zookeeper before Kafka brokers, run brokers before Kafka connect.

### Current Service

In Production server, most of service has been created as a service file, run the command as

Step 1:

* systemctl start zookeeper.service
* systemctl start kafka-broker1
* systemctl start kafka-broker2
* systemctl start kafka-broker3
* systemctl start kafka-connect

Step 2:

* create connector as [here](#_Kafka_Connect_Configuration).

Step 3:

* Test a running script to send message to the topic ‘evolog-topic’, it should be successfully saved into the MongoDB.

### Error Handling and Troubleshooting

#### Connector Error Handling

1. Wrong Connector class name

Error Code: 400

"Connector configuration is invalid and contains the following 1 error(s):\nInvalid value org.apache.kafka.connect.json.asf for configuration key.converter: Class org.apache.kafka.connect.json.asf could not be found.\nYou can also find the above list of errors at the endpoint `/connector-plugins/{connectorType}/config/validate`"

Cause:

This error is due ‘connector.class’ has been wrong,

Solution:

check back the correct connector.class name.

1. Wrong target type for timestamp converter

"Connector configuration is invalid and contains the following 2 error(s):\nInvalid value org.apache.kafka.connect.json.asf for configuration key.converter: Class org.apache.kafka.connect.json.asf could not be found.\nInvalid value time for configuration target.type: String must be one of: string, unix, Date, Time, Timestamp\nYou can also find the above list of errors at the endpoint `/connector-plugins/{connectorType}/config/validate`"

}

Cause:

This error is due ‘transforms.TimestampConvert.target.type’ has been wrong, provided only 4 types of conversion type available.

Solution:

Correct the typo or naming of that of conversion type.

#### Node.js Script Error

1. API Error

Error in streaming process {error: 'Request failed with status code 400', stack: 'AxiosError: Request failed with status code 4…ects\\Evolution logs\\evoStreaming.js:185:13)'}

evoStreaming.js:128

No debugger available, can not send 'variables'

Streaming error {error: 'startFromTime is not defined', attempt: 1, nextD

Explanation:

While this type of error were thrown, normally because of the startTime being used in the API to retrieve data was too long ago. Tested only the time within 24 hours of data would be available to successfully call the streaming API.

Important:

The API’s startTime has few requirements:

* ‘startTime’ in ISO String
* Single connection is allowed
  + Means calling the same API from two different scripts will stop either one
* ‘startTime’ parameter cannot be more than 24 hours ago

1. Connected to API but Unable to receive data

Possible causes:

* Wrong API parameter ‘startTime’ format
* Brokers Down/ Zookeeper Down ( Probably disk storage or memory not enough)

#### Server Storage Issue and Error Handling

* + 1. While the Node.js API is correct, but brokers and zookeeper service has been failed by running command:

sudo systemctl status zookeeper.service

sudo systemctl status kafka-broker1 kafka-broker2 kafka-broker3

Result

status = failed

Cause:

* Disk out of space issue
  + Check with command: du -h
* Memory usage over limitation

Solution:

* Extends larger disk storage spaces
* Add memory or add function to handle memory leaks in node.js

### Node.js - POPS

Current Flow (18-12-2024)

|  |  |
| --- | --- |
| Script | Explanation |
| 1. evoStreaming.js | Flow   1. Call mongoDBManager.js to get last record’s time 2. Use the last record time from step one as parameter to call streaming API. 3. Receive data chunk |
| * 1. mongoDBmanager.js | Return last record’s time ( *getLatestRecordTime*(‘source’) ) |
| 1. processJsonChuk.js | **Flow**   1. filter duplicated playerGameId 2. filter data.gameProvider != ‘evolution’ 3. filter redEvelopePayout != 0 && betPayout != 0 4. batch sending to kafka topic (‘evolog-topic’ / ‘redEnvelopePayouts-topic’) |

## MongoDB

Database Name: KafkaLogs

Collections:

|  |  |
| --- | --- |
| Collection Name | Explanations |
| EvolutionLogs | What is stored?  All evolution logs are stored here including both pops and swix service.  How to know the log record from which service?  Check every record’s *source\_website* as below.    Current Logic (18-12-2024)  *data.gameProvider* == ‘evolution’ |
| RedEnvelopeLogs | What is stored?  All evolution logs with redenvelope payouts are stored here including both pops and swix service.  Current Logic (18-12-2024)   1. Check *redEnvelopePayout* != 0 2. save *payout* != 0 |

## Veda Service APIs

### Evolution Logs – by date range

|  |  |
| --- | --- |
| **Method** | GET |
| **Endpoint** | http://40.83.252.157:3307api/game-records-by-date/?service={serviceName}&startDate={startDate}&endDate={endDate}& pageSize={pageSize} |
| **Params** | {startDate}/{endDate} format: 2024-11-14T00:00:00  {serviceName}: swix/pops |
| **Response** | {      "result": 1,      "data": [          {},  {}  ]  } |

### Evolution Logs – by playerGameId

|  |  |
| --- | --- |
| **Method** | GET |
| **Endpoint** | http://40.83.252.157:3307api/game-records-by-date/? service={serviceName}&startDate={startDate}&endDate={endDate} |
| **Params** | {startDate}/{endDate} format: 2024-11-14T00:00:00  {serviceName}: swix/pops |
| **Response** | {      "result": 1,      "data": [          {},  {}  ]  } |

### RedEnvelopeAPI – playerGameId list

|  |  |
| --- | --- |
| **Method** | POST |
| **Endpoint** | http://40.83.252.157:3307/api/red-envelopes-by-id/?service={serviceName} |
| **Params** | {serviceName}: swix/pops  req.body:  {  "playerGameIdList": [ "playerGameId1",  " playerGameId2",  " playerGameId3" ]} |
| **Response** | {  "result": 1,  "data": {  "count": 2,  "list": [  {  "roundId": "playerGameId1",  "betCode": "XXX",  "betPayout": 10000,  "datetime": "2024-12-12T00:00:00.000z"  },  // when this playerGameId2 does not have red envelope payout record  // the playerGameId2 will not appear in the result.  ]  }  } |

# Evolution Log Service (KR)

이 서비스는 Node.js와 Kafka, MongoDB를 사용하여 높은 처리량과 빠른 속도의 데이터 파이프라인을 구현했습니다.

## 서비스 구성 요소

* Kafka
* Kafka Connect
* Node.js
* MongoDB

## 데이터 흐름 설명

처리 흐름은 Evolution에서 제공하는 API를 통해 로깅 서비스를 요청하는 것으로 시작됩니다. API에서 제공되는 데이터는 각 청크에 대용량의 중첩된 JSON 객체를 포함한 청크 형태로 전달됩니다. 때때로 청크의 끝부분에 불완전한 JSON 데이터가 포함될 수 있어 추가적인 처리가 필요합니다. Node.js는 이 데이터를 버퍼에 저장하고, 이후 청크가 도착하여 JSON 데이터를 완성할 때까지 대기하는 중요한 역할을 합니다. 데이터가 완성되면, "evolog-topic"이라는 Kafka 토픽으로 전송되어 큐잉되고 Kafka Connect를 통해 분산 저장됩니다. Kafka Connect는 ‘Sink Connector’가 설치되어 있어 JSON 데이터를 데이터베이스 컬렉션에 자동으로 저장합니다.

## 구성 요소 설명

|  |  |
| --- | --- |
| Node.js | 목적:  Evolution 로깅 API 요청  세부사항:   * 데이터 청크에서 불완전한 JSON 처리 * 스트림 오류, 종료, 시간 초과 시 API 재요청 * Kafka 토픽(evolog-topic)으로 배치 전송 |
| Kafka | 목적:  대량 메시지 큐잉  세부사항:   * 로그 처리의 분산 브로커 * 고처리량 * 내결함성 |
| Kafka-Connect | 목적:  MongoDB로 데이터 자동 싱크  적용 도구:   * 싱크 커넥터 * 타임스탬프 변환기 |
| MongoDB | 목적:  TTL을 통한 만료 데이터 자동 제거 |

## 설치 및 설정

사전 준비 사항

서비스를 설정하기 전에 시스템에 다음 종속성이 설치되어 있는지 확인하십시오:

* Node.js: 버전 16.x 이상
* Kafka: 버전 2.5.1 이상
* **Zookeeper**: Kafka와 함께 사용되는 분산 코디네이터
* MongoDB: 버전 5.x 이상
* Kafka Connect: 필요한 커넥터 포함
  + MongoDB 싱크 커넥터
  + 타임스탬프 변환기

이러한 사전 준비 사항은 데이터 처리 파이프라인의 설치 및 구성을 위한 환경이 준비되었음을 보장합니다. 각 구성 요소가 원활하게 작동하도록 적절히 설치 및 구성되었는지 확인하십시오.

#### Zookeeper 설정

1. 공식 사이트에서 Zookeeper를 다운로드합니다.
2. conf 폴더를 확인합니다.
3. sample\_zoo.cfg 파일의 이름을 zoo.cfg로 변경합니다.
4. log\_dir을 업데이트합니다 (필수).
5. 포트와 같은 다른 설정을 필요에 맞게 변경합니다. 기본 포트는 2128입니다.

**이 서비스에서는 포트 2128을 사용합니다.**

#### Kafka 설정

1. 공식 사이트에서 Kafka를 다운로드합니다.
2. config 폴더로 이동하여 server.properties 파일을 검색합니다.
3. 원하는 대로 브로커 설정을 업데이트합니다.
4. 아래 명령어로 서비스를 실행합니다:

bin/kafka-server-start.sh config/server.properties

#### MongoDB

CosmosDB의 MongoDB Sink 커넥터에는 문제가 있기 때문에, 이 프로젝트에서는 실제 MongoDB 데이터베이스를 사용해야 합니다.

#### Kafka Connect 설정

이 서비스는 **evolution logs**를 위해 Kafka Connect를 분산 모드로 사용합니다.

1. config/connect-distributed.properties 파일을 검색합니다.
2. 원하는 대로 연결 설정을 업데이트합니다.
3. Kafka-connect 서비스를 아래 명령어로 실행합니다:

bin/connect-distributed.sh config/connect-distributed.properties

#### Kafka Connector 설정

1. Kafka Connect 서비스를 시작합니다.
2. 아래 구성으로 POST 요청을 http://{도메인 또는 IP 주소}:8083/connectors로 전송합니다.

A computer screen shot of text

Description automatically generated

중요한 사항:

Zookeeper는 Kafka 브로커보다 먼저 실행해야 하며, Kafka 브로커는 Kafka Connect보다 먼저 실행해야 합니다.

### 현재 서비스

프로덕션 서버에서 대부분의 서비스는 서비스 파일로 생성되어 있으며, 다음 명령어를 실행하여 서비스를 시작합니다.

1단계:

* systemctl start zookeeper.service
* systemctl start kafka-broker1
* systemctl start kafka-broker2
* systemctl start kafka-broker3
* systemctl start kafka-connect

2단계:

* [위에 설명된](#_Kafka_Connector_설정) 대로 커넥터를 생성합니다.

3단계:

* 주제 evolog-topic에 메시지를 보내는 실행 스크립트를 테스트합니다. 메시지가 MongoDB에 성공적으로 저장되어야 합니다.

### 오류 처리 및 문제 해결

#### 커넥터 오류 처리

1. 잘못된 커넥터 클래스 이름

"Connector configuration is invalid and contains the following 1 error(s):\nInvalid value org.apache.kafka.connect.json.asf for configuration key.converter: Class org.apache.kafka.connect.json.asf could not be found.\nYou can also find the above list of errors at the endpoint `/connector-plugins/{connectorType}/config/validate`"

설명:

이 오류는 connector.class가 잘못 지정된 경우 발생합니다. 올바른 커넥터 클래스 이름을 다시 확인하세요.

1. 타임스탬프 변환기의 잘못된 대상 유형

"Connector configuration is invalid and contains the following 2 error(s):\nInvalid value org.apache.kafka.connect.json.asf for configuration key.converter: Class org.apache.kafka.connect.json.asf could not be found.\nInvalid value time for configuration target.type: String must be one of: string, unix, Date, Time, Timestamp\nYou can also find the above list of errors at the endpoint `/connector-plugins/{connectorType}/config/validate`"

}

설명:

이 오류는 transforms.TimestampConvert.target.type이 잘못 지정된 경우 발생합니다. 변환 가능한 네 가지 유형만 제공됩니다.

#### Node.js Script 오류

* + 1. API Error

Error in streaming process {error: 'Request failed with status code 400', stack: 'AxiosError: Request failed with status code 4…ects\\Evolution logs\\evoStreaming.js:185:13)'}

evoStreaming.js:128

No debugger available, can not send 'variables'

Streaming error {error: 'startFromTime is not defined', attempt: 1, nextD

설명:

이러한 유형의 오류가 발생하는 것은 일반적으로 API에서 데이터를 검색하는 데 사용되는 startTime이 너무 오래되었기 때문입니다. 스트리밍 API를 성공적으로 호출하려면 24시간 이내의 데이터만 사용할 수 있다는 것이 테스트되었습니다.

중요사항:

API의 startTime에는 몇 가지 요구사항이 있습니다:

* 'startTime'은 ISO 문자열 형식이어야 함
* 단일 연결만 허용됨
  + 두 개의 다른 스크립트에서 동일한 API를 호출하면 둘 중 하나가 중지됨을 의미
* 'startTime' 매개변수는 24시간 이전의 시간을 사용할 수 없음