

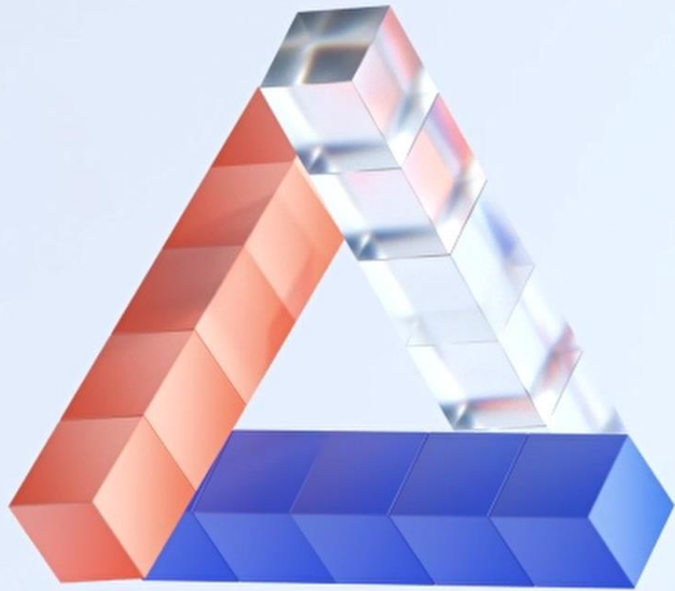


KafkaAdminclient API를 이용한 Kafka Managing 서비스 개발

이상헌 / 이진환



Contents



1. KafkaAdminClient?
2. 프로젝트 소개
3. 프로젝트 예시(CRUD)

<IceBreak> 커리어 입문 계기와 관심사



Data Lake



1. KafkaAdminClient



*Kafka Object : topics, brokers, acls, consumers 등

Q : KafkaAdminClient가 뭐죠?

A : Kafka Object의 **정보를 조회**하고 매니징하는데 사용할 수 있는 class이고, 이를 통해 Kafka Cluster의 **내부 옵션을 설정**할 수 있습니다.

공식적으로는 Admin API라고 불리며, org.apache.kafka.clients 경로에 producer, consumer, admin package가 구현되어 있습니다.

- Admin API

<https://kafka.apache.org/documentation/#api>

1. GETTING STARTED

[1.1 Introduction](#)

[1.2 Use Cases](#)

[1.3 Quick Start](#)

[1.4 Ecosystem](#)

[1.5 Upgrading](#)

2. APIS

[2.1 Producer API](#)

[2.2 Consumer API](#)

[2.3 Streams API](#)

[2.4 Connect API](#)

[2.5 Admin API](#)



- 1.The [Producer](#) API allows applications **to send streams of data to topics** in the Kafka cluster.
- 2.The [Consumer](#) API allows applications **to read streams of data from topics** in the Kafka cluster.
- 3.The [Streams](#) API allows **transforming streams of data** from input topics to output topics.
- 4.The [Connect](#) API allows implementing connectors that continually **pull from some source system** or application into Kafka or **push from Kafka into some sink system** or application.
- 5.The [Admin](#) API allows **managing and inspecting topics, brokers, and other Kafka objects**.

- 왜 사용할까요?

→ Kafka Cluster의 내부 옵션 설정/변경 관련된 부분을 자동화할 수 있다

Case 1.

Kafka Consumer를 Multi-thread로 생성하고 싶다.

토픽의 파티션 개수만큼 스레드를 생성하고 싶을 때, 스레드 생성 전에 해당 토픽의 파티션 개수를 Admin API를 통해 가져올 수 있다.

Case 2.

AdminClient Class로 구현한 웹 대시보드를 통해 ACL이 적용된 Cluster의 리소스 접근 권한 규칙을 추가할 수 있다.

Case 3.

특정 토픽의 데이터양이 늘어남을 감지하고 AdminClient Class로 해당 토픽의 파티션을 늘릴 수 있다.

- 주요 메소드

→ AdminClient class에서 제공하는 주요 메소드입니다.

<Topic 관련>

- 1.createTopics() : 토픽 생성
- 2.deleteTopics() : 토픽 삭제
- 3.listTopics() : 토픽 목록조회
- 4.describeTopics() : 토픽 상세조회

<Cluster, ConsumerGroup 관련>

- 1.describeClusters() : Node 정보 조회
- 2.listTransactions() : transactions 정보 조회
- 3.describeConsumerGroups() : Group ID 조회
- 4.listConsumerGroupOffsets() : consumer group offset 목록 조회

<Acl 관련>

- 1.createAcls() : Acl 생성
- 2.deleteAcls() : Acl 삭제
- 3.describeAcls() : Acl 상세조회

- 사용 방법 - dependency 설정

→ “kafka-clients”를 project에 dependency로 추가해야 합니다.

- With Maven

```
<dependency>  
    <groupId> org.apache.kafka </groupId>  
    <artifactId> kafka-clients </artifactId>  
    <version> 3.4.0 </version>  
</dependency>
```

- With Gradle

```
implementation 'org.apache.kafka:kafka-clients:3.4.0'
```


- 사용 방법 - Admin 객체 생성

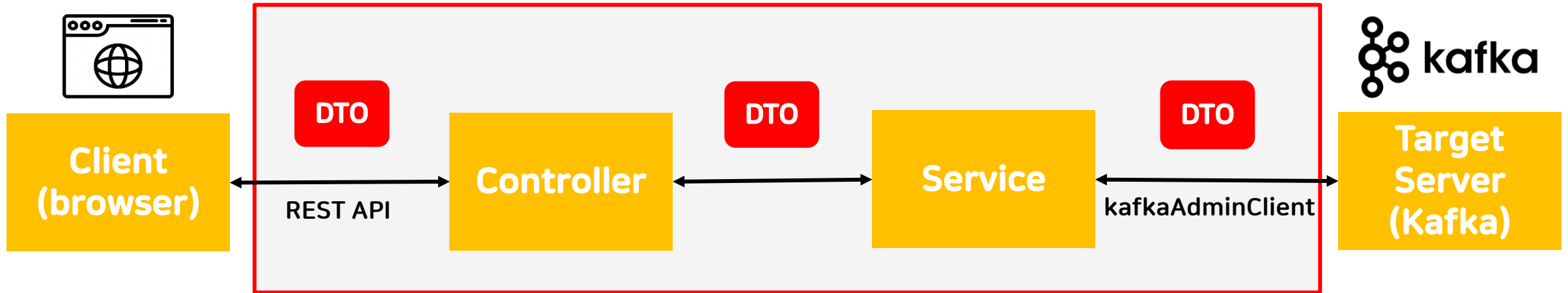
```

1 Properties props = new Properties();
2 props.put(AdminClientConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
3
4 try (Admin admin = Admin.create(props)) {
5     String topicName = "my-topic";
6     int partitions = 12;
7     short replicationFactor = 3;
8     // Create a compacted topic
9     CreateTopicsResult result = admin.createTopics(Collections.singleton(
10         new NewTopic(topicName, partitions, replicationFactor)
11             .configs(Collections.singletonMap(TopicConfig.CLEANUP_POLICY_CONFIG, TopicConfig.CLEANUP_POLICY_COMPACT))));
12
13     // Call values() to get the result for a specific topic
14     KafkaFuture<Void> future = result.values().get(topicName);
15
16     // Call get() to block until the topic creation is complete or has failed
17     // if creation failed the ExecutionException wraps the underlying cause.
18     future.get();
19 }

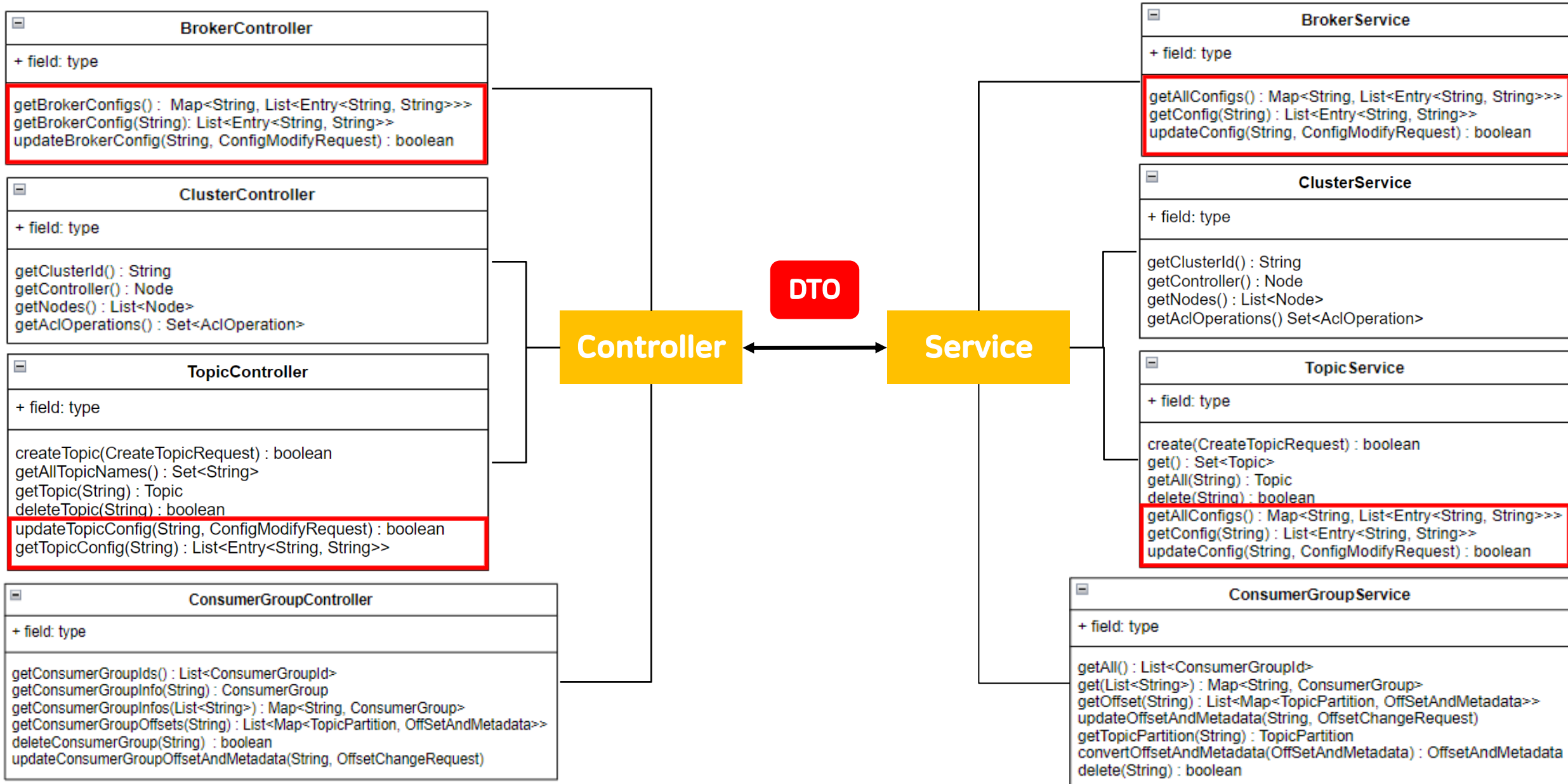
```



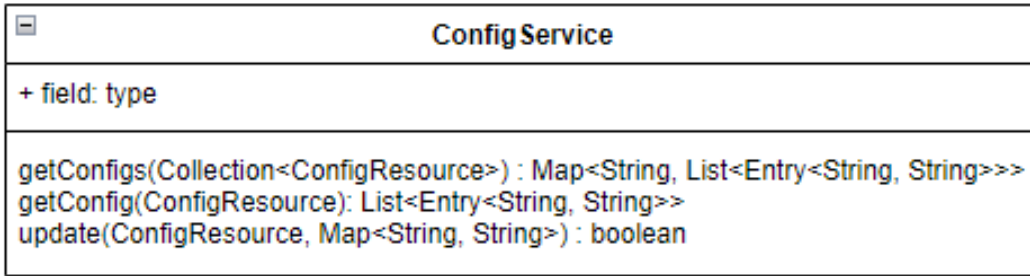
- 주제 : kafkaAdminClient API를 이용한 kafka Managing 서비스



- Controller : Client의 요청을 받았을 때, 그 요청에 대해 실제 동작을 수행하는 Service를 호출합니다.
- Service : Controller의 요청을 받아, 알맞은 정보를 가공하여 Controller에게 재전달합니다.
- DTO : 계층 간의 데이터 교환을 위한 객체입니다.
Kafka에서 사용하는 Node, Topic, ConsumerGroup과 같은 자료구조를 담고 있습니다.



- 서비스 내에 config 관련 메소드들은 ConfigService라는 class를 사용.



- broker-controller, topic-controller 모듈에서 describeConfigs와 incrementalAlterConfigs 메소드를 사용.

<describeConfigs 메소드>

DescribeConfigsResult describeConfigs(Collection<ConfigResource> resources, DescribeConfigsOptions options)

Parameters:

resources - The resources (topic and broker resource types are currently supported)
options - The options to use when describing configs

Returns: The DescribeConfigsResult

- 매개변수 Collection<>의 데이터 타입인 ConfigResource class의 내부에는 ConfigResource.Type 이라는 Enum 이 정의되어 있음.
- Enum ConfigResource.Type에는 BROKER, BROKER_LOGGER, TOPIC, UNKNOWN이 정의되어 있기 때문에 Broker, Topic 과 관련된 config 정보를 얻기 위해서는 ConfigResource.Type을 사용해야함.

3. 프로젝트 예시(CRUD)



<https://github.com/jinhwan2/kafka-admin-server>

- KafkaAdminClient를 이용한 서비스는 Kafka CLI에서 command를 입력하는 수고로움을 덜 수 있습니다.
- Spring과 Swagger 프레임워크를 이용하여 API 문서 자동화를 진행했습니다.

kafka admin server ^{1.0.0}

[Base URL: localhost:3000/]
<http://localhost:3000/v2/api-docs>

카프카 REST 어드민 서버

broker-controller Broker Controller

GET

/v1/brokers getBrokerConfigs

GET

/v1/brokers/{nodeId} getBrokerConfig

PUT

/v1/brokers/{nodeId} updateBrokerConfig

cluster-controller Cluster Controller

consumer-group-controller Consumer Group Controller

home-controller Home Controller

topic-controller Topic Controller

Models

GET

/v1/brokers/{nodeId} getBrokerConfig

Try it out

Parameters

Name	Description
nodeId * required string (path)	nodeId

Responses

Response content type */*

Code	Description
200	<div>OK</div> <div>Example Value Model</div> <div><pre>[{ "key": "string", "value": "string" }]</pre></div>
401	<div>Unauthorized</div>
403	<div>Forbidden</div>
404	<div>Not Found</div>

- KafkaAdminClient를 이용하여 **Topic Create(생성)** Example

<Kafka CLI> - Topic 생성

```
$ bin/kafka-topics.sh --create ₩
                        --bootstrap-server <hostname>:<port> ₩
                        --topic <topic-name> ₩
                        --partitions <number-of-partitions> ₩
                        --replication-factor <number-of-replicating-servers>
```

Option	Description
--create	the create action
--bootstrap-server <host name>: <port>	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster
--topic <topic-name>	the name of the topic your wish to create
--partitions <number-of-partitions>	the number of partitions you want the topic to have
--replication-factor <number-of-replicating-servers>	the replication factor for the topic. may not be greater than the number of brokers

- KafkaAdminClient를 이용하여 **Topic Create(생성)** Example

<REST API> - Topic 생성

POST

/v1/topics createTopic

Parameters

Cancel

Name	Description
numPartitions integer(\$int32) (query)	<input type="text" value="1"/>
replicationFactor integer(\$int32) (query)	<input type="text" value="3"/>
topicName string (query)	<input type="text" value="test-00"/>

Execute

Clear

Responses

Response content type

/

Curl

```
curl -X POST "http://localhost:3000/v1/topics?numPartitions=1&replicationFactor=3&topicName=test-00" -H "accept: */*"

```

- KafkaAdminClient를 이용하여 **Topic config Read(읽기)** Example

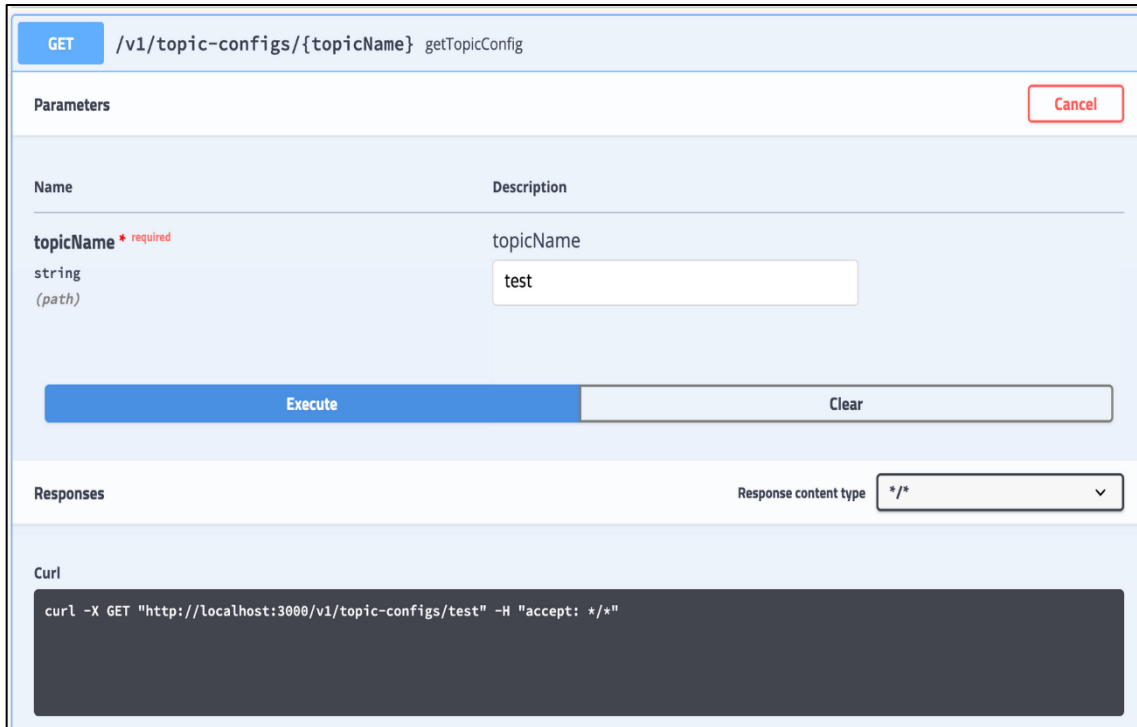
<Kafka CLI> - Topic config 조회

```
$ bin/kafka-configs.sh --describe \  
    --bootstrap-server <hostname>:<port> \  
    --entity-type <entity-type> \  
    --entity-name <entity-name>
```

Option	Description
--describe	provide details of one or more topics
--bootstrap-server <host name>: <port>	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster
--entity-type <entity-type>	the type of entity (topics/brokers/users/clients)
--entity-name <entity-name>	the name of entity

- KafkaAdminClient를 이용하여 **Topic config Read(읽기)** Example

<REST API> - Topic config 조회



The image shows a REST client interface for the endpoint `GET /v1/topic-configs/{topicName} getTopicConfig`. The interface includes a "Parameters" section with a table for the request parameters. The table has two columns: "Name" and "Description". The "Name" column contains `topicName` with a red asterisk indicating it is required, and its type is `string (path)`. The "Description" column contains `topicName`. Below the table, there is an input field for the `topicName` parameter, which contains the value `test`. There are "Execute" and "Clear" buttons. Below the parameters section, there is a "Responses" section with a dropdown for "Response content type" set to `*/*`. At the bottom, there is a "Curl" section with a text area containing the following curl command: `curl -X GET "http://localhost:3000/v1/topic-configs/test" -H "accept: */*"`.

Name	Description
<code>topicName</code> * required string (path)	<code>topicName</code>

Execute Clear

Responses Response content type `*/*`

Curl

```
curl -X GET "http://localhost:3000/v1/topic-configs/test" -H "accept: */*"
```

```
{
  "data": [
    { "key": "compression.type", "value": "producer" },
    { "key": "cleanup.policy", "value": "delete" },
    ...
    { "key": "flush.ms", "value": "922337203685477" },
    ...
    { "key": "retention.ms", "value": "604800000" }
  ],
  "status": 200,
  "failMessage": null
}
```

→ retention 기간, cleanup.policy 등을 간편히 조회할 수 있습니다.

- KafkaAdminClient를 이용하여 **Topic config Update(갱신)** Example

<Kafka CLI> - Topic config 갱신

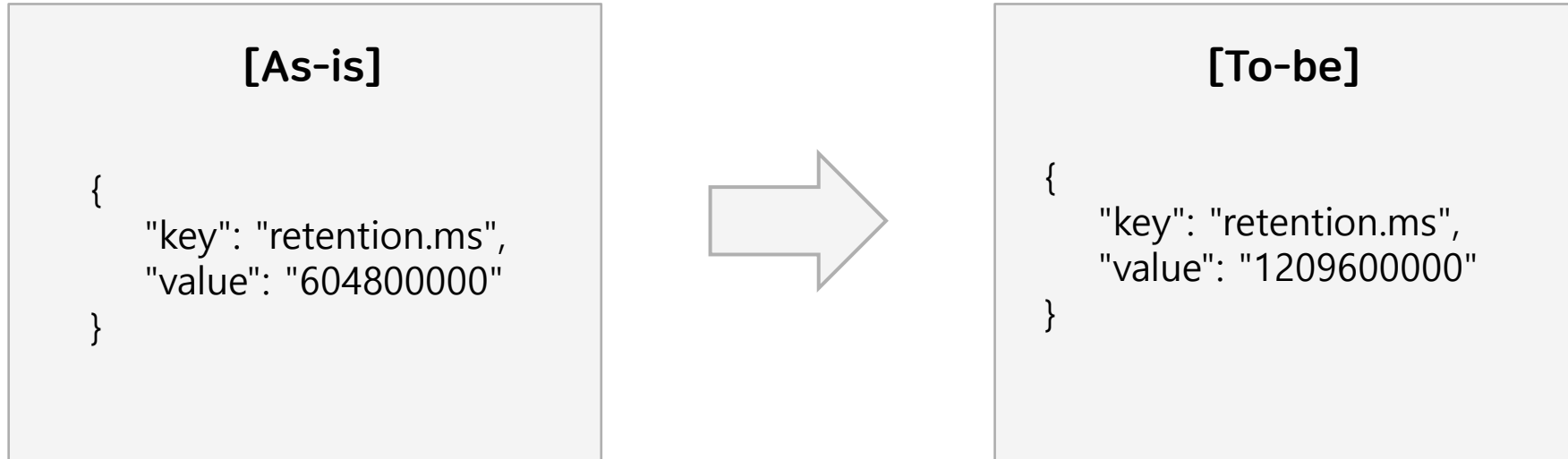
```
$ bin/kafka-topics.sh --alter ₩
    --bootstrap-server <hostname>:<port> ₩
    --topic <topic-name> ₩
    --config <key>=<value>
```

Option	Description
--alter	Alter the configuration for the topic.
--bootstrap-server <host name>: <port>	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster
--topic <topic-name>	the name of the topic your wish to create
--config <key>=<value>	a config value to change

- KafkaAdminClient를 이용하여 **Topic config Update(갱신)** Example

<REST API> - Topic config 갱신

만약, 토픽의 데이터를 유지하는 시간을 더 길게 조정하고 싶다면, "retention.ms" 파라미터 값을 바꿔줘야 합니다.



→ 현재 값은 1주(604800000 ms)인데, 이를 2배 늘려서 2주(1209600000 ms)로 변경하고자 합니다.

- KafkaAdminClient를 이용하여 **Topic config Update(갱신)** Example

<REST API> - Topic config 갱신

PUT

/v1/topics/{topicName} updateTopicConfig

Parameters

Name

Description

configModifyRequest * required

configModifyRequest

(body)

Example Value | Model

```
{
  "config": {
    "retention.ms": "1209600000"
  }
}
```

Cancel

Parameter content type

application/json

topicName * required

string

(path)

topicName

test-topic

Execute

Clear

Responses

Response content type */*

Curl

```
curl -X PUT "http://localhost:3000/v1/topics/test-topic" -H "accept: */*" -H "Content-Type: application/json" -d '{"config": { "retention.ms": "1209600000" }}'
```

Request URL

http://localhost:3000/v1/topics/test-topic

Server response

Code

Details

200

Response body

```
{
  "data": true,
  "status": 200
}
```

Download

- KafkaAdminClient를 이용하여 **Topic config Delete(삭제)** Example

<Kafka CLI> - Topic 삭제

→ server.properties에서 "delete.topic.enable = true" 로 설정 값을 사전에 바꿔주어야 합니다.

```
$ bin/kafka-topics.sh -delete ₩
                        --bootstrap-server <hostname>:<port> ₩
                        --topic <topic-name> ₩
```

Option	Description
--delete	delete one or more topics
--bootstrap-server <host name>: <port>	A list of host/port pairs to use for establishing the initial connection to the Kafka cluster
--topic <topic-name>	the name of the topic to be deleted.

[Topic lists]

```
{
  "data":["123", "test3", "test", "exam-topic", "test-00"],
  "status":200
}
```

- KafkaAdminClient를 이용하여 **Topic config Delete(삭제)** Example

< REST API> - Topic 삭제

DELETE /v1/topics/{topicName} deleteTopic

Parameters
Cancel

Name	Description
topicName * required string (path)	topicName <input type="text" value="123"/>

Execute
Clear

Responses
Response content type */*

Curl

```
curl -X DELETE "http://localhost:3000/v1/topics/123" -H "accept: */*"

```

[Topic lists]

```
{
  "data":["test3", "test", "exam-topic", "test-00"],
  "status":200
}
```



Thank you

