KAFKA ADMINISTRATION

Đơn vị: Công ty CP Giáo dục và Công nghệ QNET

QNET JOINT STOCK COMPANY

Address: 14th Floor, VTC Online Tower
18 Tam Trinh Street. Hoang Mai District
Hanoi, Vietnam

Kafka Administration

Administering Kafka

Managing a Kafka cluster requires additional tooling to perform administrative changes to topics, configurations, and more. Kafka provides several command-line interface (CLI) utilities that are useful for making administrative changes to your clusters. The tools are implemented in Java classes, and a set of scripts are provided natively to call those classes properly. While these tools provide basic functions, you may find they are lacking for more complex operations or are unwieldy to use at larger scales. This chapter will describe only the basic tools that are available as part of the Apache Kafka open source project. More information about advanced tools that have been developed in the community, outside of the core project, can be found on the Apache Kafka website.

Kafka Administration

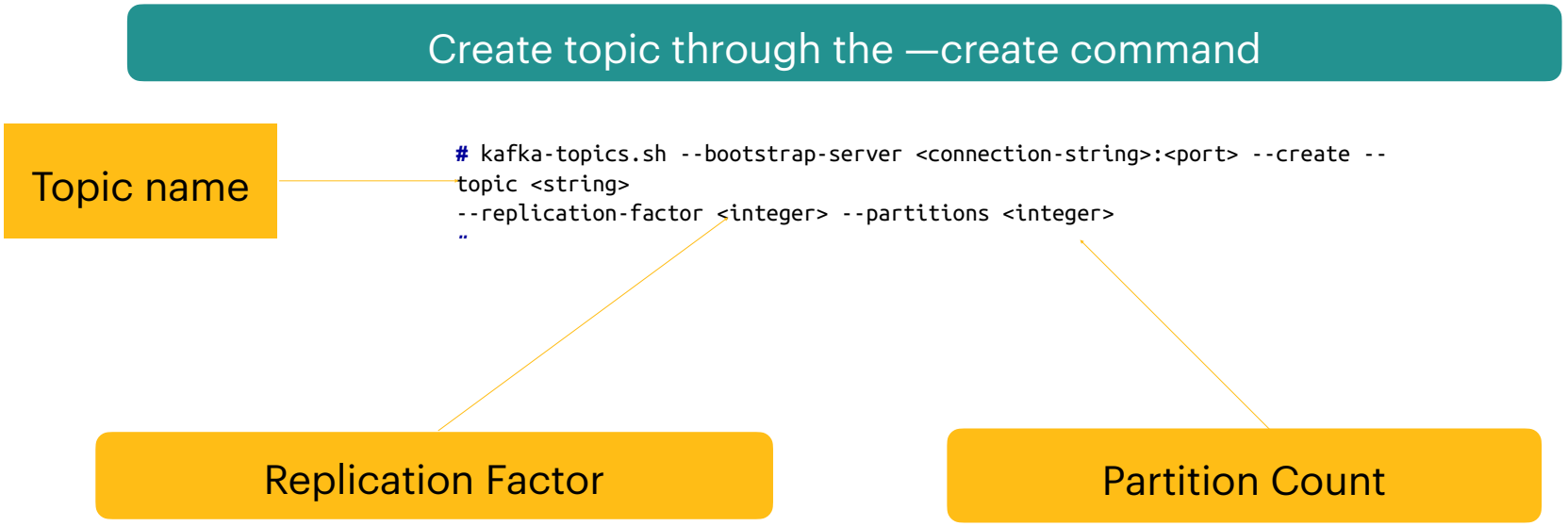
Topic Operation

Kafka-topics tool enables to perform following topic operations

1. Create
2. Modify
3. Listing
4. Deleting
5. Describe

Kafka Administration

Topic Operation - Create topic



Kafka Administration

Topic Operation - List all topic in a Cluster

Using —list command

```
# kafka-topics.sh --bootstrap-server localhost:9092 --list
__consumer_offsets
my-topic
other-topic
```

Cluster address

You'll notice the internal `__consumer_offsets` topic is listed here. Running the command with `--exclude-internal` will remove all topics from the list that begin with the double underscore mentioned earlier, which can be beneficial.

Kafka Administration

Topic Operation -Describing Topic Details

Using —describe command to get detail information on one or more

```
# kafka-topics.sh --boostrap-server localhost:9092 --describe --topic my-topic
Topic: my-topic PartitionCount: 8       ReplicationFactor: 2    Configs: seg
ment.bytes=1073741824
        Topic: my-topic Partition: 0    Leader: 1       Replicas: 1,0   Isr: 1,0
        Topic: my-topic Partition: 1    Leader: 0       Replicas: 0,1   Isr: 0,1
        Topic: my-topic Partition: 2    Leader: 1       Replicas: 1,0   Isr: 1,0
        Topic: my-topic Partition: 3    Leader: 0       Replicas: 0,1   Isr: 0,1
        Topic: my-topic Partition: 4    Leader: 1       Replicas: 1,0   Isr: 1,0
        Topic: my-topic Partition: 5    Leader: 0       Replicas: 0,1   Isr: 0,1
        Topic: my-topic Partition: 6    Leader: 1       Replicas: 1,0   Isr: 1,0
        Topic: my-topic Partition: 7    Leader: 0       Replicas: 0,1   Isr: 0,1
#
```

Kafka Administration

Topic Operation - Adding and Deleting partitions

Add Partition

```
# kafka-topics.sh --bootstrap-server localhost:9092
--alter --topic my-topic --partitions 16
```

Total number of

Reduce

It is not possible to reduce the number of partitions for a topic

We could workaround by is to delete the topic and create new topic with same name and less number of partitions

Kafka Administration

Topic Operation - Deleting topics

Delete a topics to free up disk space, open file handles, and memory

Setting delete.topic.enable = true on broker in the cluster

Topic deletion is an asynchronous operation

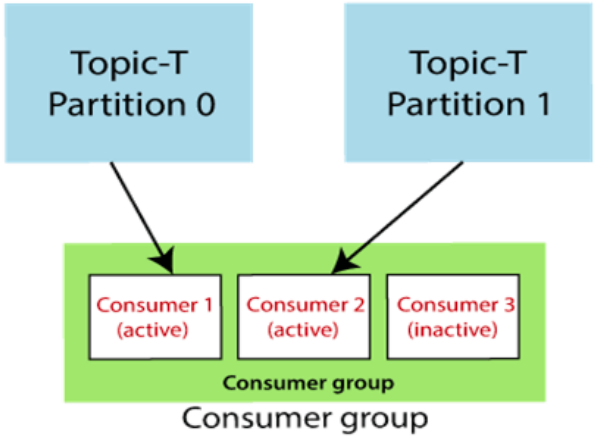
```
# kafka-topics.sh --bootstrap-server localhost:9092
--delete --topic my-topic

Note: This will have no impact if delete.topic.enable is not set
to true.
#
```

Deleting a topic will delete all its messages which is not reversible

Should not delete more than one or two topics at a time

Kafka Administration

Consumer Group



- List and describe groups
- Delete Group
- Offset Management

Kafka Administration

Consumer Group: List and describe groups

Use kafka-consumer-groups.sh tool to list and describe consumer groups

List

```
# kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
console-consumer-95554
console-consumer-9581
my-consumer
#
```

Describe

```
# kafka-consumer-groups.sh --bootstrap-server localhost:9092
--describe --group my-consumer
GROUP          TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET
LAG        CONSUMER-ID
HOST                           CLIENT-ID
my-consumer     my-topic       0          2               4
2          consumer-1-029af89c-873c-4751-a720-cefd41a669d6   /
127.0.0.1                      consumer-1
my-consumer     my-topic       1          2               3
1          consumer-1-029af89c-873c-4751-a720-cefd41a669d6   /
127.0.0.1                      consumer-1
my-consumer     my-topic       2          2               3
1          consumer-2-42c1abd4-e3b2-425d-a8bb-e1ea49b29bb2   /
127.0.0.1                      consumer-2
#
```

Kafka Administration

Consumer Group: Delete group

Delete of consumer group can be performed with the —delete argument

Broker address

Group to delete

```
# kafka-consumer-groups.sh --bootstrap-server localhost:9092 --delete --group
my-consumer
Deletion of requested consumer groups ('my-consumer') was successful.
#
```

Kafka Administration

Offset Management

Export

```
# kafka-consumer-groups.sh --bootstrap-server localhost:9092
--export --group my-consumer --topic my-topic
--reset-offsets --to-current --dry-run > offsets.csv
```

Import

```
# kafka-consumer-groups.sh --bootstrap-server localhost:9092
--reset-offsets --group my-consumer
--from-file offsets.csv --execute
  TOPIC                          PARTITION  NEW-OFFSET
  my-topic                        0          8905
  my-topic                        1          8915
  my-topic                        2          9845
```

Stop Consumers First

Before performing this step, it is important that all consumers in the group are stopped. They will not read the new offsets if they are written while the consumer group is active. The consumers will just overwrite the imported offsets.

Kafka Administration

Dynamic Configuration Changes

There is a plethora of configurations for topics and clients, brokers and more that can be updated dynamically during runtime

Topics

Brokers

Users

Clients

Configuration key	Description
`cleanup.policy`	If set to `compact`, the messages in this topic will be discarded and only the most recent message with a given key is retained (log compacted).
`compression.type`	The compression type used by the broker when writing message batches for this topic to disk.
`delete.retention.ms`	How long, in milliseconds, deleted tombstones will be retained for this topic. Only valid for log compacted topics.
`file.delete.delay.ms`	How long, in milliseconds, to wait before deleting log segments and indices for this topic from disk.
`flush.messages`	How many messages are received before forcing a flush of this topic's messages to disk.
`flush.ms`	How long, in milliseconds, before forcing a flush of this topic's messages to disk.
`follower.replication.throttled.replicas`	A list of replicas for which log replication should be throttled by the follower.
`index.interval.bytes`	How many bytes of messages can be produced between entries in the log segment's index.
`leader.replication.throttled.replica`	A list of replicas for which log replication should be throttled by the leader.
`max.compaction.lag.ms`	Maximum time limit a message won't be eligible for compaction in the log.
`max.message.bytes`	The maximum size of a single message for this topic, in bytes.

Kafka Administration

Console Produce Tool

Tool: *kafka-console-producer.sh*
Usage: To manually write messages into a kafka topic in a Kafka Cluster

1. By default, messages are read one per line, with a tab character separating key and value

2. Minimum two arguments are required by console producer:
   — broker-list
   — topic

```
# kafka-console-producer.sh --bootstrap-server localhost:9092 --topic my-topic
>Message 1
>Test Message 2
>Test Message 3
>Message 4
>^D
#
```

Kafka Administration

Configure console producer tool

We can pass producer configuration options to the console producer in two ways

Provide options in configuration file:

Pass configuration file by specifying —producer.config CONFIGFILE

CONFIGFILE: full path to file that contains the configuration options

Provide options using CLI:

- —producer-properties KEY=VALUE
  - KEY: configuration option name
  - VALUE: Value to be set
- —key-serializer CLASSNAME
- —value-serializer CLASSNAME
- —compression-codec STRING
- —sync

Kafka Administration

Console Consumer Tool

Tool: *kafka-console-consumer.sh*
Usage: To manually consume messages out of one or more topics in a Kafka Cluster

1. The messages are printed in standard output, delimited by a new line

2. By default, it outputs the raw bytes in the message with no formatting

3. When using older consumer:
   Only argument required is —zookeeper option followed by connection string for the cluster

4. When using new consumer:
   Arguments:
   —new-consumer flag(optional)
   —broker-list followed by comma-separated broker list
   —black-list : Blacklist will consume all topics except those matched the regular expression

Kafka Administration

Console Consumer Tool

Example:

```
# kafka-console-consumer.sh --bootstrap-server localhost:9092
--whitelist 'my.*' --from-beginning
Message 1
Test Message 2
Test Message 3
Message 4
^C
#
```

Kafka Administration

Console Consumer Options

`--formatter <classname>`
: Specifies a message formatter class to be used to decode the messages. This defaults to `kafka.tools.DefaultMessageFormatter`.

`--from-beginning`
: Consume messages in the topic(s) specified from the oldest offset. Otherwise, consumption starts from the latest offset.

`--max-messages <int>`
: The maximum number of messages to consume before exiting.

`--partition <int>`
: Consume only from the partition with the ID given.

`--offset`
: The offset ID to consume from, if provided (`<int>`). Other valid options are `earliest`, which will consume from the beginning, and `latest`, which will start consuming from the most recent offset.

`--skip-message-on-error`
: Skip a message if there is an error when processing instead of halting. Useful for debugging.

Kafka Administration

Console Consumer : Message Formater Options

There are three message formatters available to use besides the default:

`kafka.tools.LoggingMessageFormatter`
: Outputs messages using the logger, rather than standard out. Messages are printed at the INFO level and include the timestamp, key, and value.

`kafka.tools.ChecksumMessageFormatter`
: Prints only message checksums.

`kafka.tools.NoOpMessageFormatter`
: Consumes messages but does not output them at all.

Kafka Administration

Console Consumer : Message Formater Options

Example:

```
# kafka-console-consumer.sh --bootstrap-server localhost:9092
--whitelist 'my.*' --from-beginning
--formatter kafka.tools.ChecksumMessageFormatter
checksum:0
checksum:0
checksum:0
checksum:0
#
```

Discussion

Quality Network for Education and Technology

XIN CHÂN THÀNH CẢM ƠN!