緯創IT先進技術實驗室

witlab

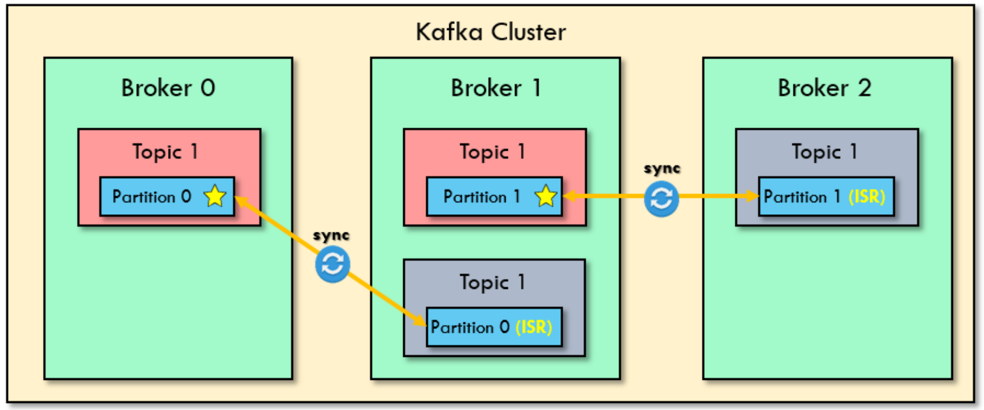
ak系列 – ak02

(Apache Kafka – 進階)

Critical Topic Configurations

Partitions Count, Replication Factor

- The two **most important** parameters when creating a topic.
- The impact performance and durability of the system overall



- It is best to get the parameters right the first time!
  - If the **Partitions Count** increases during a topic life-cycle, you will break your keys ordering guarantees
  - If the **Replication Factor** increases during a topic life-cycle, you put more pressure on your cluster, which can lead to unexpected performance decrease

Partitions Count

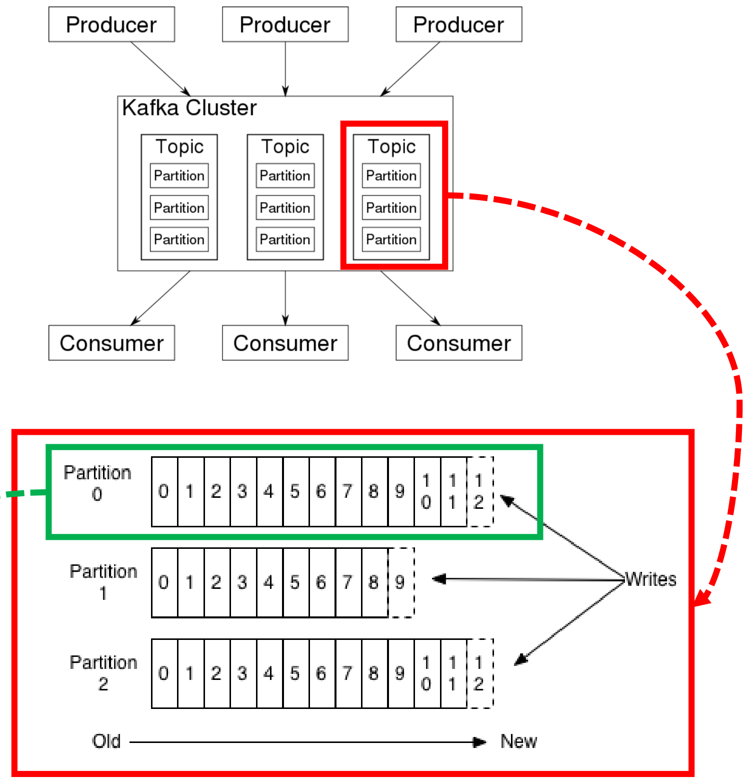
- Roughly, each partition can get a throughput of 10 MB / sec
- More partitions implies:
  - Better parallelism, better throughput
  - BUT more files opened on your system
  - BUT if a broker fails (unclean shutdown), lots of concurrent leader elections
  - BUT added latency to replicate (in the order of milliseconds)
- Guidelines:
  - Partitions per topic = (1 to 2) x (# of brokers), max 10 partitions
  - Example: in a 3 brokers setup, 3 or 6 partitions is a good number to start with

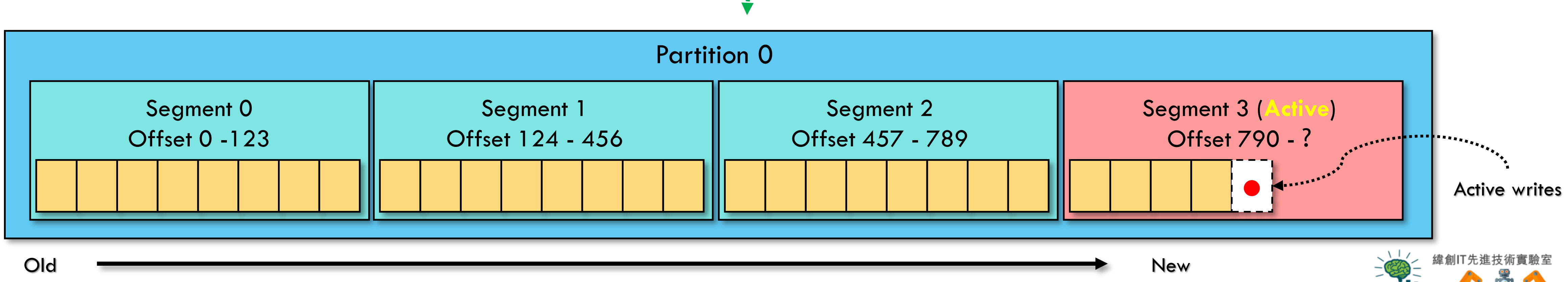
Replication Factor

- Should be at least **2**, maximum of **3**
- The higher the replication factor:
  - Better resilience of your system (N-1 brokers can fail)
  - BUT longer replication (higher latency is acks=all)
  - BUT more disk space on your system (50% more if RF is 3 instead of 2)
- Guidelines:
  - **Set it to 2** (if you have 3 brokers)
  - **Set it to 3** (if you have greater than 5 brokers)
  - If replication performance is an issue, get a better broker instead of less replication factor

Partitions and Segments

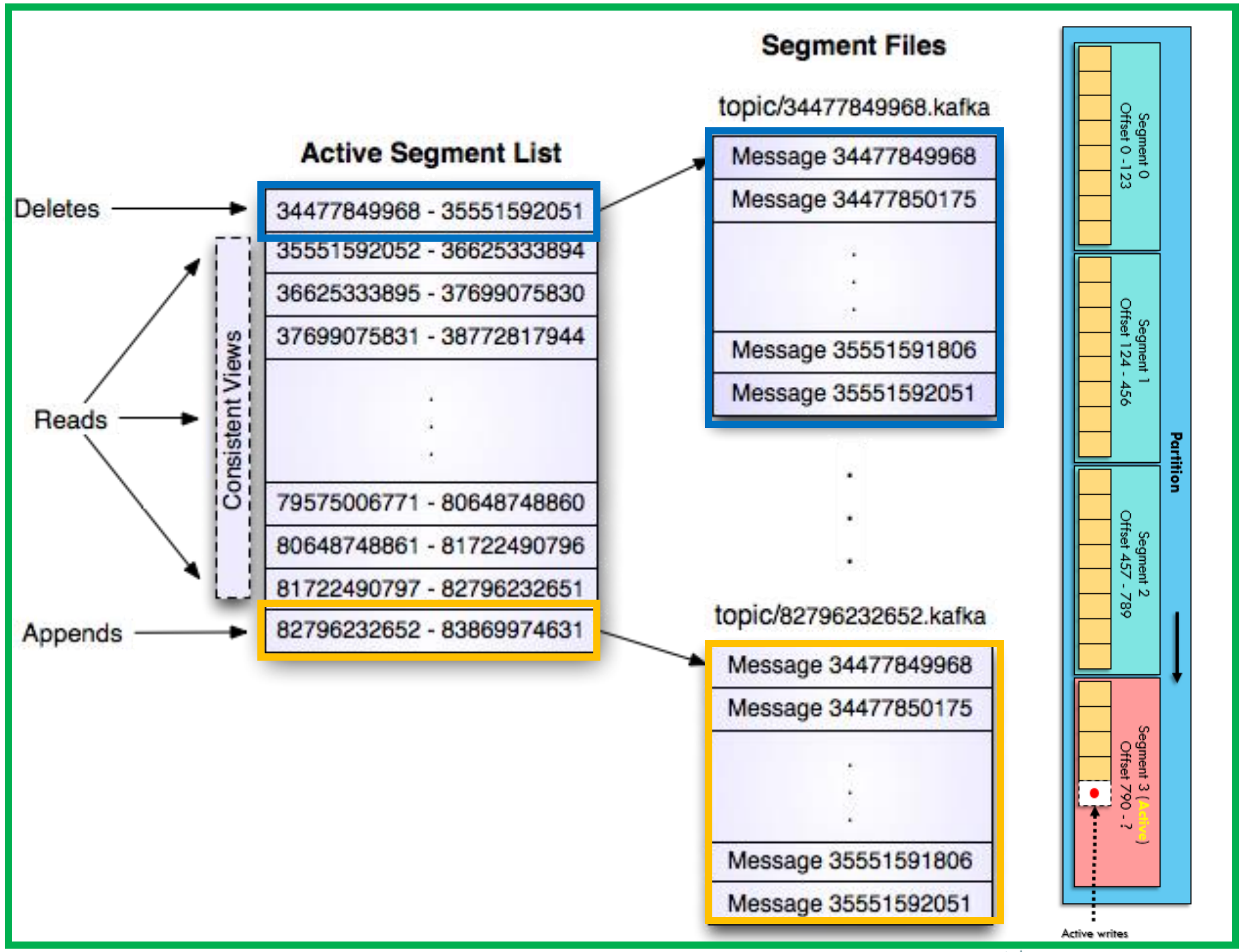
- **Topics** are made of **partitions** (we already know that)
- **Partitions** are made of … **segments** (files)!
- Only one **segment** is ACTIVE (the one data is being written to)





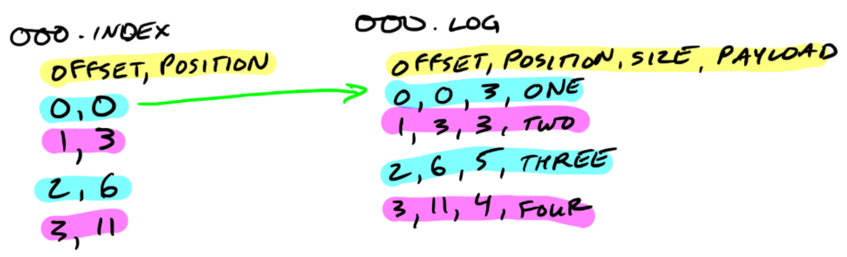
Partitions and Segments

- Two segment settings:
  - **log.segment.bytes**: the max size of a single segment in bytes
  - **log.segment.ms**: the time kafka will wait before committing the segment if not full

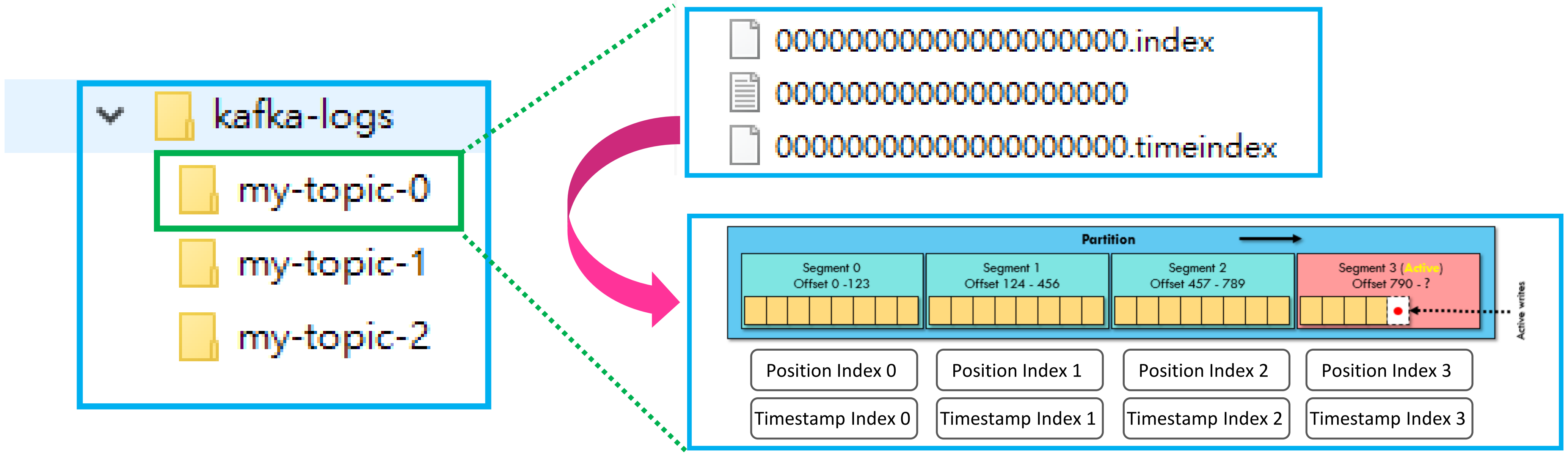


緯創IT先進技術實驗室
witlab

Segments and Indexes



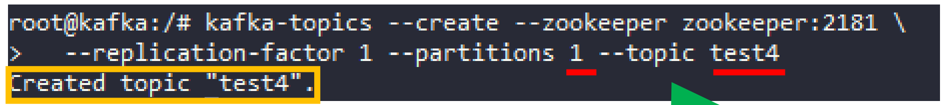
- Segments come with **two** indexes (files):
  - An offset to position index: allows Kafka where to read to find a message
  - A timestamp to offset index: allow Kafka to find messages with a timestamp
- Therefore, Kafka knows where to find data in a constant time!



Segments and Indexes

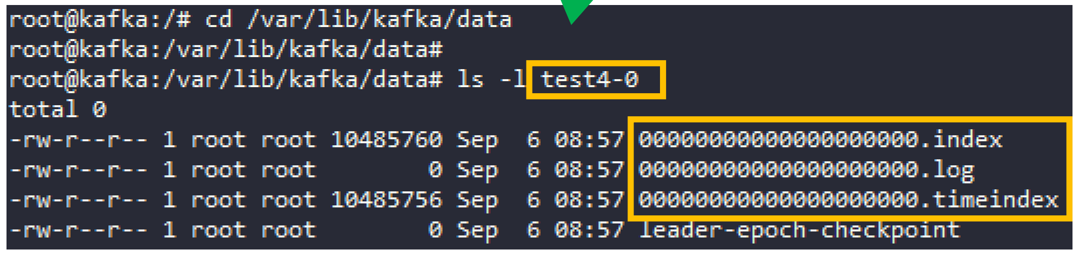
Create Topic (test4)

```
$ kafka-topics
    --create
    --zookeeper localhost:2181
    --replication-factor 1 --partitions 1
    --topic test4
```



```
root@kafka:/# kafka-topics --create --zookeeper zookeeper:2181 \
>   --replication-factor 1 --partitions 1 --topic test4
Created topic "test4".
```

請注意partitions的數量與topic名稱



```
root@kafka:/# cd /var/lib/kafka/data
root@kafka:/var/lib/kafka/data#
root@kafka:/var/lib/kafka/data# ls -l test4-0
total 0
-rw-r--r-- 1 root root 10485760 Sep  6 08:57 00000000000000000000.index
-rw-r--r-- 1 root root        0 Sep  6 08:57 00000000000000000000.log
-rw-r--r-- 1 root root 10485756 Sep  6 08:57 00000000000000000000.timeindex
-rw-r--r-- 1 root root        0 Sep  6 08:57 leader-epoch-checkpoint
```

在kafka的資料目錄下會找到對應的folder名稱

緯創IT先進技術實驗室 witlab

Segments: Why should I care?

- A small **log.segment.bytes** (size, default: 1GB) means:
  - More segments per partitions
  - Log Compaction happens more often
  - BUT Kafka has to keep more files opened (Error: Too many open files)
- Ask yourself: how fast will I have new segments based on throughput?
- A small **log.roll.hours** or **log.roll.ms**(time, default 1 week) means:
  - You set a max frequency for log compaction (more frequent triggers)
  - Maybe you want daily compaction instead of weekly?
- Ask yourself: how often do I need log compaction to happen?

Log Cleanup Policies

IMPORTANT NOTICE

- Many Kafka clusters make data expire, according to a policy
- That concept is called “**log cleanup**”.
  - Policy 1: **log.cleanup.policy**=delete (Kafka default for all user topics)
    - Delete based on age of data (default is a week)
    - Delete based on max size of log (default is -1 == infinite)
  - Policy 2: **log.cleanup.policy**=**compact** (Kafka default for topic **__consume_offsets**)
    - Delete based on keys of your messages
    - Will delete old duplicate keys **after** the active segment is committed
    - Infinite time and space retention

Log Cleanup: Why and When?

- Deleting data from Kafka allows you to:
  - Control the size of the data on the disk, delete obsolete data
  - Overall: Limit maintenance work on the Kafka Cluster

- How often does log cleanup happen?
  - Log cleanup happens on your **partition segments**!
  - Smaller / More segments means that log cleanup will happen more often!
  - Log cleanup shouldn't happen too often => takes CPU and RAM resources
  - The cleaner checks for work every 15 seconds (**log.cleaner.backoff.ms**)

Log Cleanup Policy:

log.cleanup.policy=delete

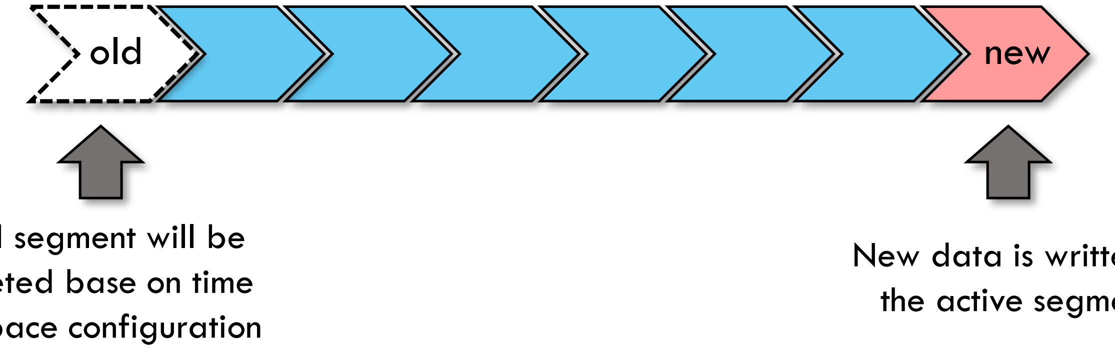
- **log.retention.hours**:
  - Number of hours to keep data for (default is 168 – one week)
  - Higher number means more disk space
  - Lower number means that less data is retained (your consumers may need to replay more data than less)

- **log.retention.bytes**:
  - Max size in Bytes for each partition (default is -1 == infinite)
  - Useful to keep the size of a log under a threshold

緯創IT先進技術實驗室 witlab

Log Cleanup Policy: Delete

log.cleanup.policy=delete


old
new
Old segment will be deleted base on time or space configuration
New data is written to the active segment

Use cases – two common pair of options:

- One week of retention:
  - log.retention.hours = 168 and log.retention.bytes = -1
- Infinite time retention bounded by 500 MB:
  - log.retention.hours = 17520 and log.retention.bytes = 524288000

Log Cleanup Policy: Compact

log.cleanup.policy=compact

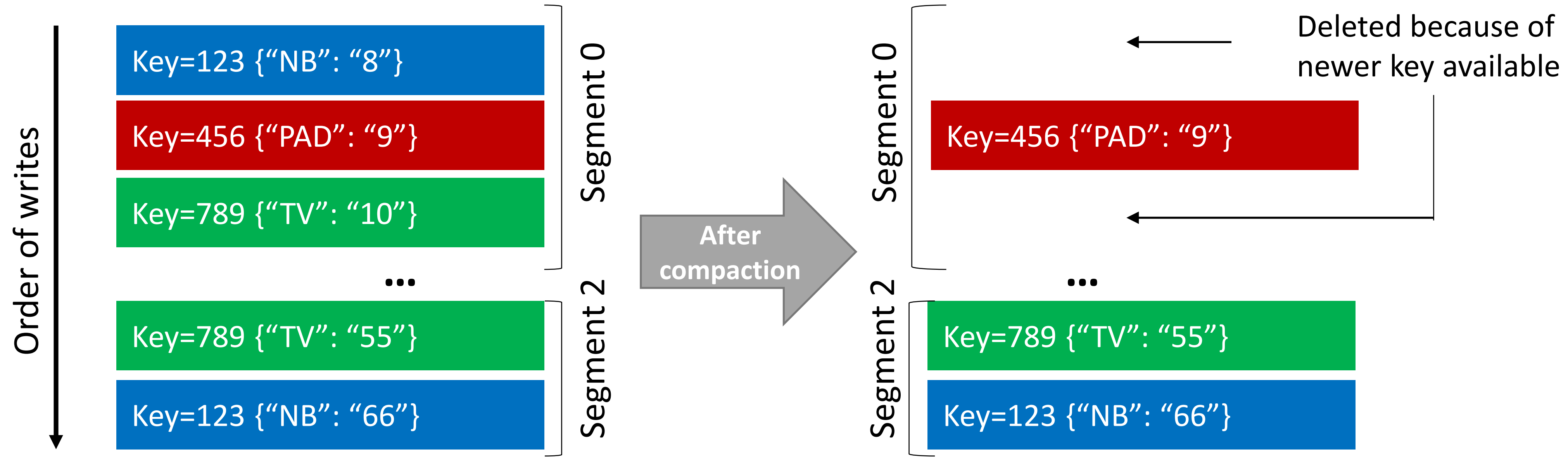
IMPORTANT NOTICE

- Log compaction ensures that your log contains *at least the last know value for a specific key within a partition*
- Very useful if we just require a SNAPSHOT instead of full history (such as for a data table in a database)
- The idea is that we only keep the latest “update” for a key in our log

Log Cleanup Policy: Example

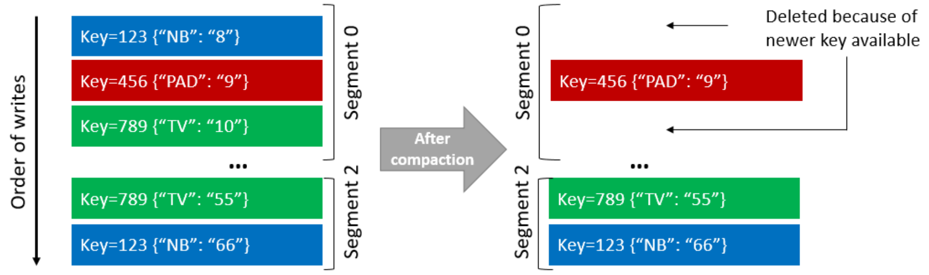
log.cleanup.policy=compact

- Our topic is: **product-inventory**
- We want to keep the most recent inventory for our products

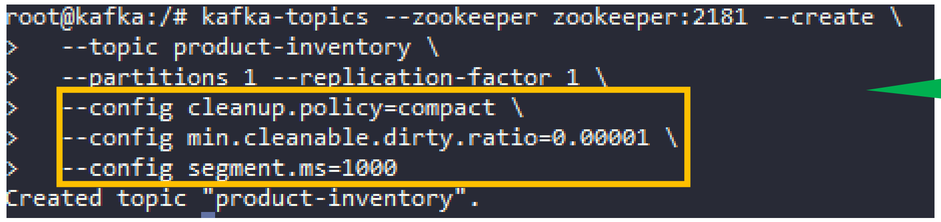


Log Cleanup Policy: Example

log.cleanup.policy=compact



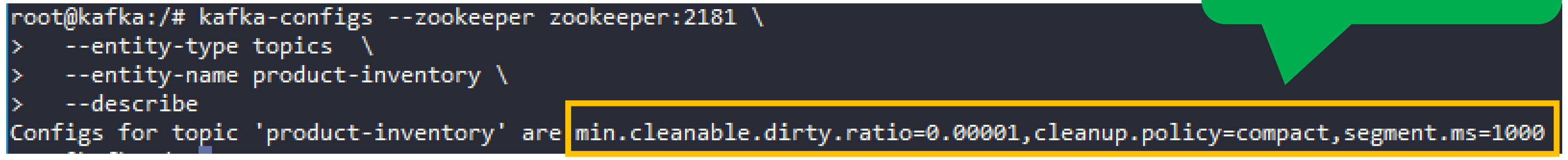
1



```
root@kafka:/# kafka-topics --zookeeper zookeeper:2181 --create \
>    --topic product-inventory \
>    --partitions 1 --replication-factor 1 \
>    --config cleanup.policy=compact \
>    --config min.cleanable.dirty.ratio=0.00001 \
>    --config segment.ms=1000
Created topic "product-inventory".
```

特別去設定這個topic的一些參數來展現log-compaction的效果

2

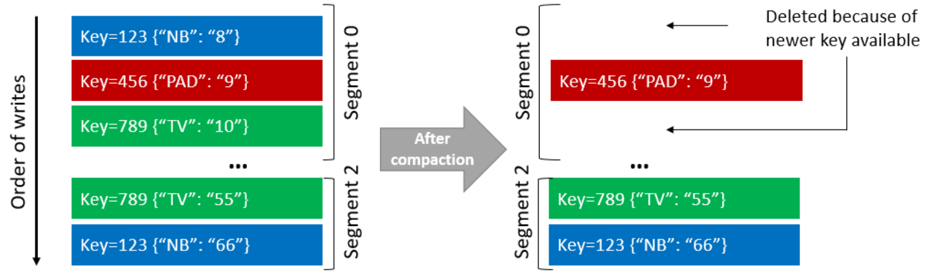


```
root@kafka:/# kafka-configs --zookeeper zookeeper:2181 \
>    --entity-type topics  \
>    --entity-name product-inventory \
>    --describe
Configs for topic 'product-inventory' are min.cleanable.dirty.ratio=0.00001,cleanup.policy=compact,segment.ms=1000
```

檢查topic的參數

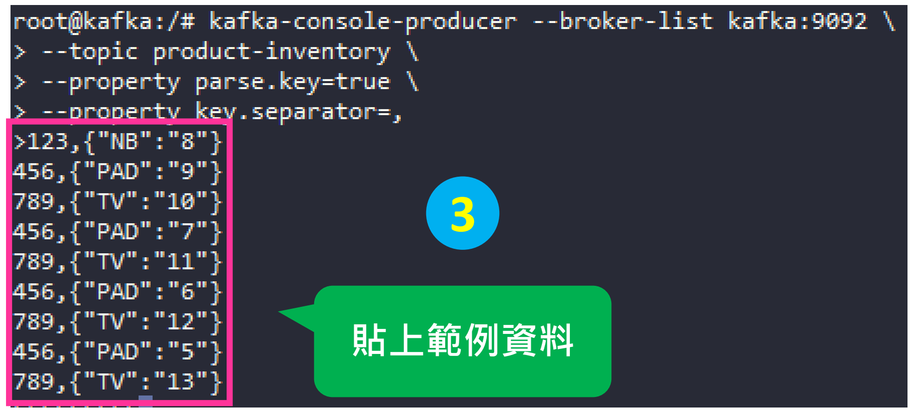
Log Cleanup Policy: Example

log.cleanup.policy=compact



2 Producer#1

再開一個Console-producer來發佈資料

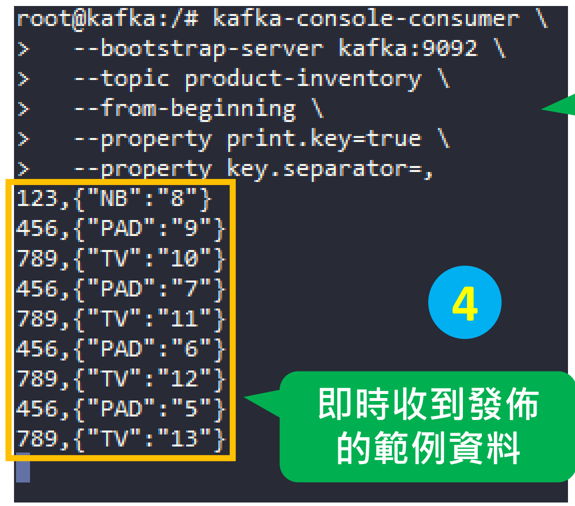


```
root@kafka:/# kafka-console-producer --broker-list kafka:9092 \
> --topic product-inventory \
> --property parse.key=true \
> --property key.separator=,
>123,{"NB":"8"}
456,{"PAD":"9"}
789,{"TV":"10"}
456,{"PAD":"7"}
789,{"TV":"11"}
456,{"PAD":"6"}
789,{"TV":"12"}
456,{"PAD":"5"}
789,{"TV":"13"}
```

3 貼上範例資料

1 Consumer#1

開一個Console-consumer訂閱這個"product-inventory"的topic

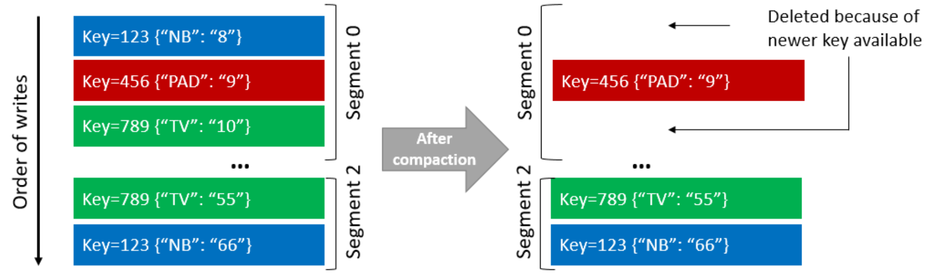


```
root@kafka:/# kafka-console-consumer \
>    --bootstrap-server kafka:9092 \
>    --topic product-inventory \
>    --from-beginning \
>    --property print.key=true \
>    --property key.separator=,
123,{"NB":"8"}
456,{"PAD":"9"}
789,{"TV":"10"}
456,{"PAD":"7"}
789,{"TV":"11"}
456,{"PAD":"6"}
789,{"TV":"12"}
456,{"PAD":"5"}
789,{"TV":"13"}
```

4 即時收到發佈的範例資料

Log Cleanup Policy: Example

log.cleanup.policy=compact



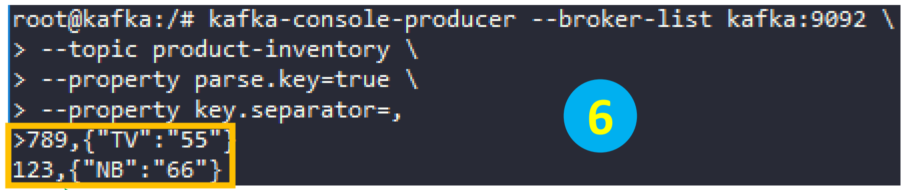
5

等待10秒後....

再開一個Console-consumer訂閱這個"product-inventory"的topic

8

Producer#1

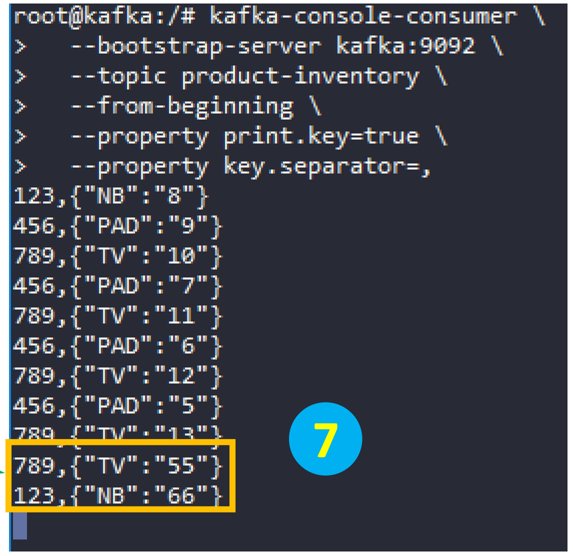


```
root@kafka:/# kafka-console-producer --broker-list kafka:9092 \
> --topic product-inventory \
> --property parse.key=true \
> --property key.separator=,
>789,{"TV":"55"}
123,{"NB":"66"}
```

6

貼上最後2筆範例資料

Consumer#1

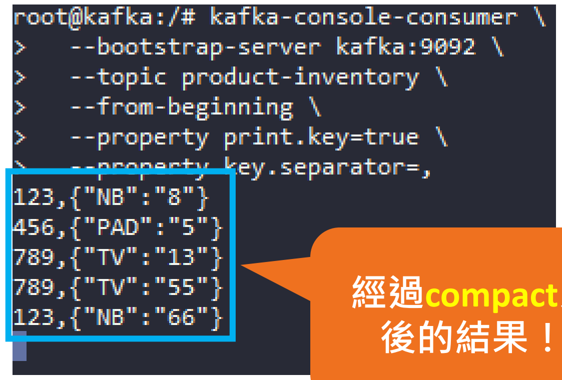


```
root@kafka:/# kafka-console-consumer \
>    --bootstrap-server kafka:9092 \
>    --topic product-inventory \
>    --from-beginning \
>    --property print.key=true \
>    --property key.separator=,
123,{"NB":"8"}
456,{"PAD":"9"}
789,{"TV":"10"}
456,{"PAD":"7"}
789,{"TV":"11"}
456,{"PAD":"6"}
789,{"TV":"12"}
456,{"PAD":"5"}
789,{"TV":"13"}
789,{"TV":"55"}
123,{"NB":"66"}
```

7

Console-consumer確實地收到了最後發佈的最後2筆範例資料

Consumer#2



```
root@kafka:/# kafka-console-consumer \
>    --bootstrap-server kafka:9092 \
>    --topic product-inventory \
>    --from-beginning \
>    --property print.key=true \
>    --property key.separator=,
123,{"NB":"8"}
456,{"PAD":"5"}
789,{"TV":"13"}
789,{"TV":"55"}
123,{"NB":"66"}
```

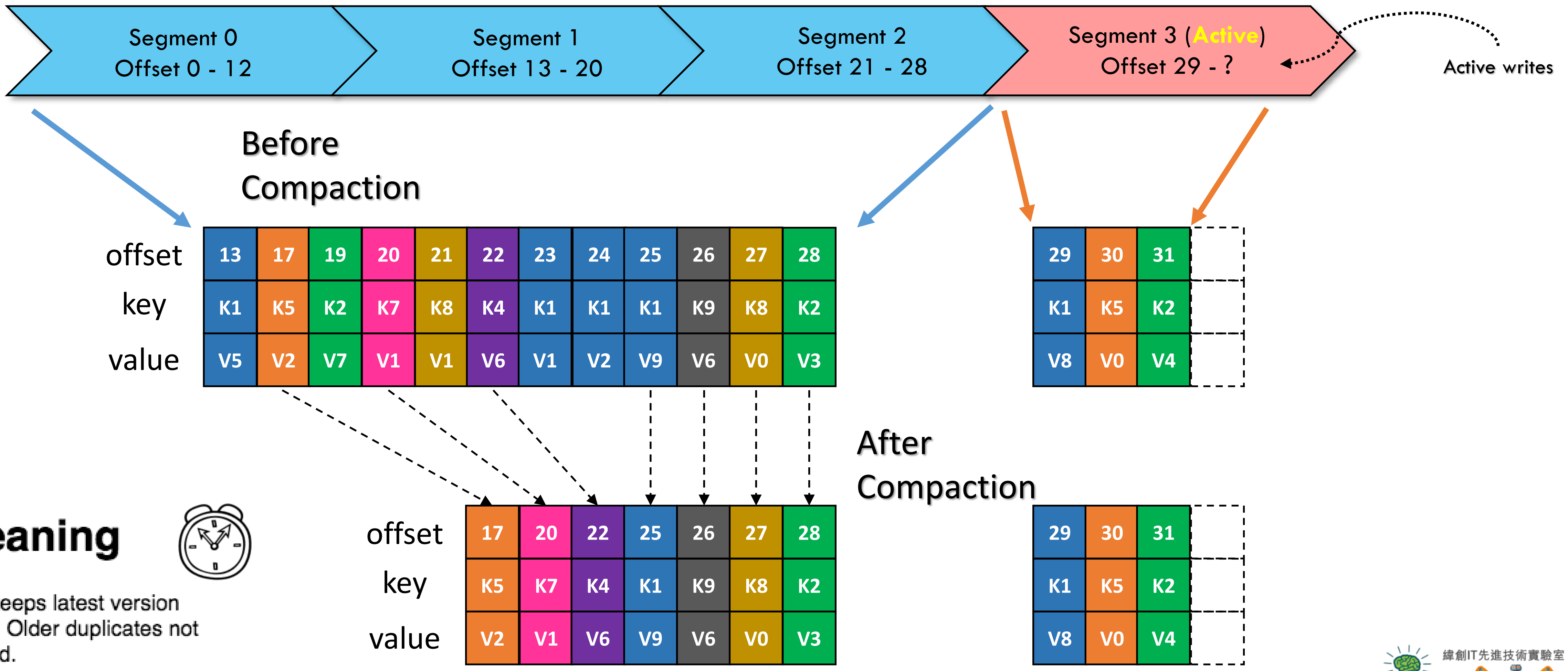
經過compact之後的結果！

Log Cleanup Policy: Example

log.cleanup.policy=compact

Segment 0

IMPORTANT NOTICE



Before Compaction

offset	13	17	19	20	21	22	23	24	25	26	27	28
key	K1	K5	K2	K7	K8	K4	K1	K1	K1	K9	K8	K2
value	V5	V2	V7	V1	V1	V6	V1	V2	V9	V6	V0	V3

29	30	31	
K1	K5	K2	
V8	V0	V4	

After Compaction

offset	17	20	22	25	26	27	28
key	K5	K7	K4	K1	K9	K8	K2
value	V2	V1	V6	V9	V6	V0	V3

29	30	31	
K1	K5	K2	
V8	V0	V4	

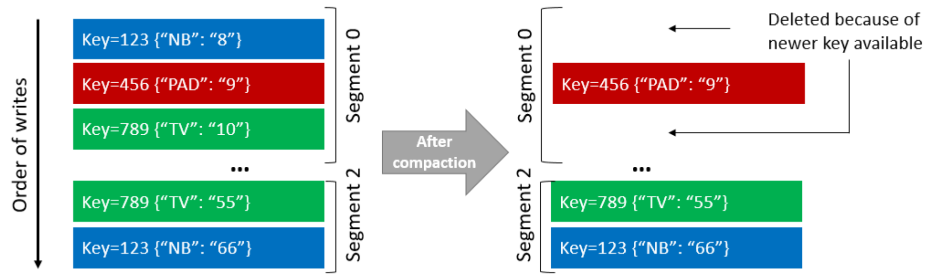
Cleaning

Only keeps latest version of key. Older duplicates not needed.

Log Compaction: Example

log.cleanup.policy=compact

- Any consumer that is reading *from the head of a log* will still see all the messages sent to the topic
- Ordering of message is kept, log compaction only removes some messages, but does not re-order them
- The offset of a message is immutable (it never changes). Offsets are just skipped if a message is missing



Log Compression

- Topics can be compressed using **compression.type** setting.
- Options are '**gzip**', '**snappy**', '**lz4**', '**uncompressed**', '**producer**'
- If you need compression, ideally you keep default as 'producer'.
    - The producer will perform the compression on its side
    - The broker will take the data as is => Saves CPU resources on the broker
- If compression is enabled, make sure you're sending batches of data
- Data will be uncompressed by the consumer!
- Compression only makes sense if you're sending non-binary data (json, xml, text…), don't enable compression for binary data (parquet, protobuf, avro..)

緯創IT先進技術實驗室 witlab

Other advanced configurations

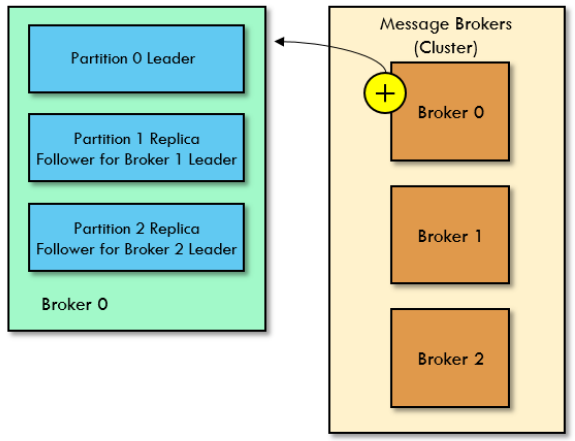
- **max.messages.bytes** (default is **1MB**): if your messages get bigger than 1MB, increase this parameter on the topic and your consumers buffer
- **min.isync.replicas** (default is **1**): if using acks=all, specify how many brokers need to acknowledge the write
- **unclean.leader.election** (danger zone! – default **false**): if set to true, it will allow replicas which are not in sync to become leader as a last resort if all ISRs are offline. This can lead to data loss. If set to false, the topic will go offline until the ISRs come back up

緯創IT先進技術實驗室
witlab

Advanced: Topic Configuration

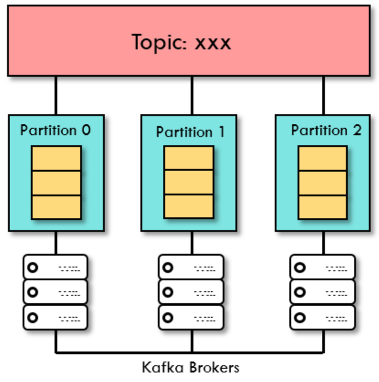
Why should I care about **topic** configuration?

- Brokers have defaults for all the topic configuration parameters
- These parameters impact **performance** and **topic behavior**
- Some topics may need different values than the defaults
  - Replication Factor
  - Number of Partitions
  - Message size
  - Compression level
  - Log Cleanup Policy
  - Other configurations
- A list of configuration can be found at:
  - https://kafka.apache.org/documentation/#topicconfigs


Partition 0 Leader
Partition 1 Replica Follower for Broker 1 Leader
Partition 2 Replica Follower for Broker 2 Leader
Broker 0
Message Brokers (Cluster)
Broker 0
Broker 1
Broker 2

Topic configuration example:

- Configurations pertinent to topics have both a **server default** as well an optional **per-topic override**.
- *If no **per-topic configuration** is given the **server default** is used.*
- The override can be set at topic creation time by giving one or more **--config** options.



Apache Kafka: Hands-on Practice

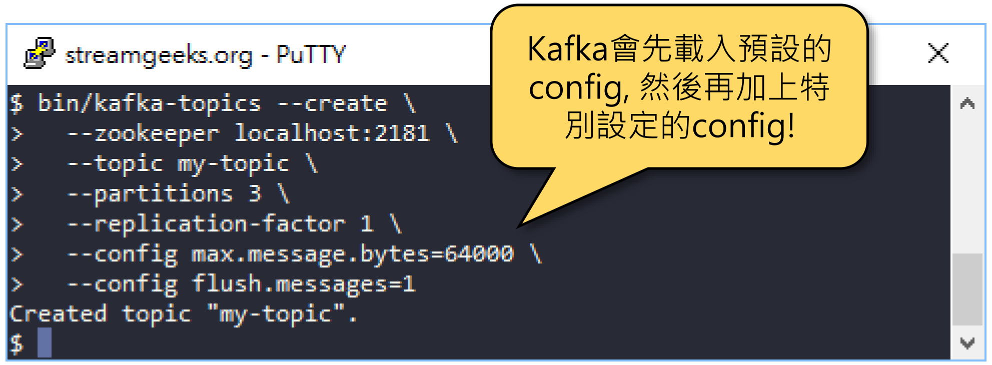
Change Topic Configurations

Topic configuration example: **(Create)**

CREATE READ UPDATE DELETE
C R U D

- Configurations pertinent to topics have both a server default as well an optional per-topic override.
- If no per-topic configuration is given the server default is used.

```
bin/kafka-topics
    --zookeeper localhost:2181
    --create
    --topic my-topic
    --partitions 3
    --replication-factor 1
    --config max.message.bytes=64000
    --config flush.messages=1
```



streamgeeks.org - PuTTY

```
$ bin/kafka-topics --create \
>   --zookeeper localhost:2181 \
>   --topic my-topic \
>   --partitions 3 \
>   --replication-factor 1 \
>   --config max.message.bytes=64000 \
>   --config flush.messages=1
Created topic "my-topic".
$
```

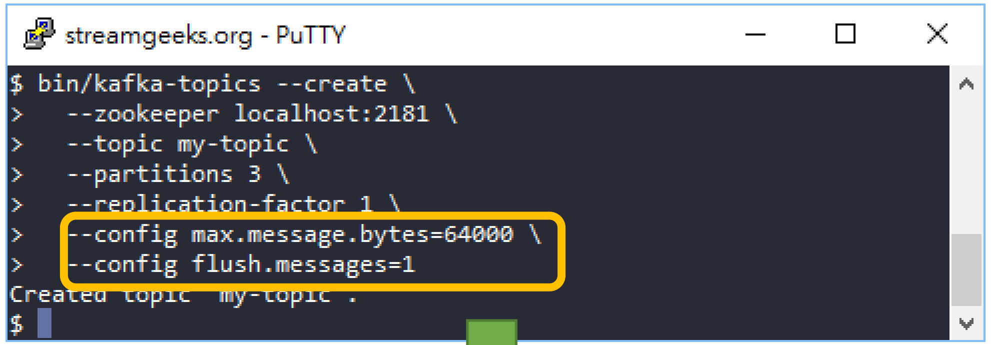
Kafka會先載入預設的config, 然後再加上特別設定的config!

緯創IT先進技術實驗室 witlab

Topic configuration example: (**Query**)

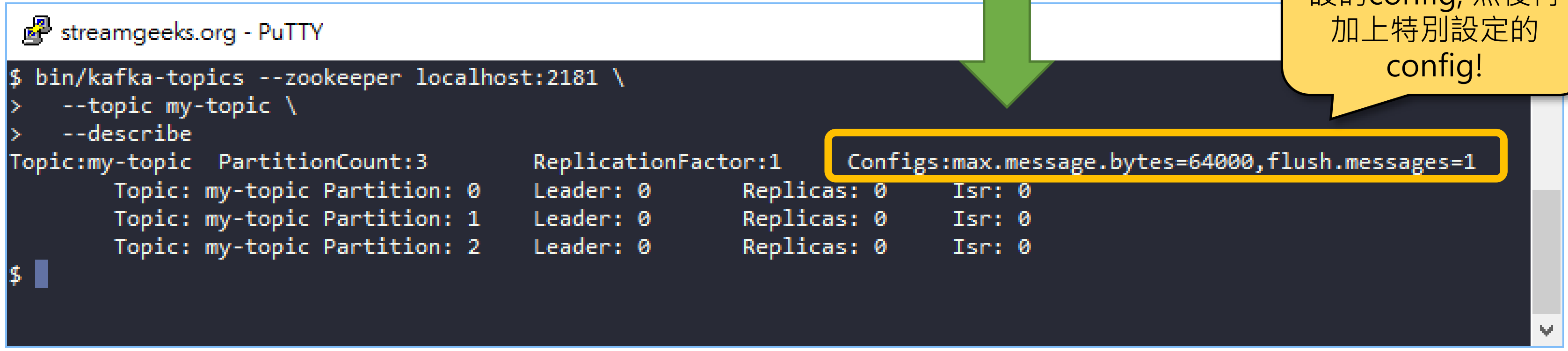
CREATE READ UPDATE DELETE
C R U D

bin/**kafka-topic**
 --zookeeper localhost:2181
 --topic my-topic
 --describe



streamgeeks.org - PuTTY

```
$ bin/kafka-topics --create \
>   --zookeeper localhost:2181 \
>   --topic my-topic \
>   --partitions 3 \
>   --replication-factor 1 \
>   --config max.message.bytes=64000 \
>   --config flush.messages=1
Created topic "my-topic".
$
```



streamgeeks.org - PuTTY

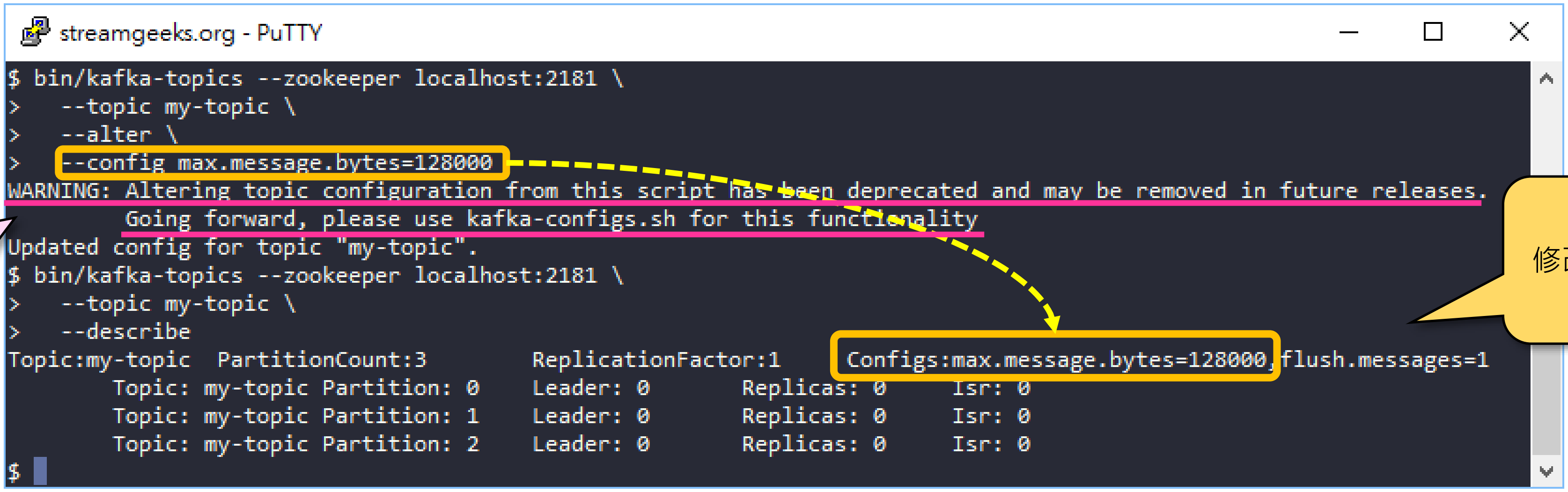
```
$ bin/kafka-topics --zookeeper localhost:2181 \
>   --topic my-topic \
>   --describe
Topic:my-topic  PartitionCount:3        ReplicationFactor:1     Configs:max.message.bytes=64000,flush.messages=1
        Topic: my-topic Partition: 0    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-topic Partition: 1    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-topic Partition: 2    Leader: 0       Replicas: 0     Isr: 0
$
```

Kafka會先載入預設的config, 然後再加上特別設定的config!

Topic configuration example: (**Update**)

bin/**kafka-topics**
  --zookeeper localhost:2181
  --topic my-topic
  --alter
  --config max.message.bytes=128000

為了讓修改Kafka的相關設定有一個共同的Utility工具,Kafka建議使用"bin/kafka-configs"來修改Topic設定



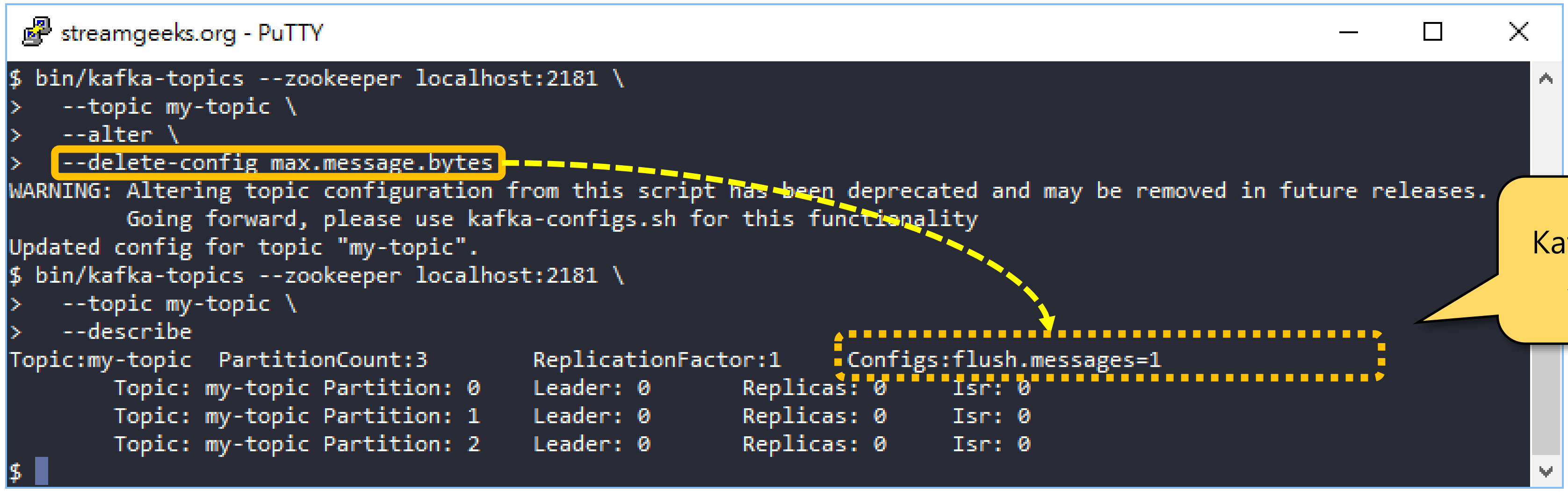
```
$ bin/kafka-topics --zookeeper localhost:2181 \
>   --topic my-topic \
>   --alter \
>   --config max.message.bytes=128000
WARNING: Altering topic configuration from this script has been deprecated and may be removed in future releases.
         Going forward, please use kafka-configs.sh for this functionality
Updated config for topic "my-topic".
$ bin/kafka-topics --zookeeper localhost:2181 \
>   --topic my-topic \
>   --describe
Topic:my-topic  PartitionCount:3        ReplicationFactor:1     Configs:max.message.bytes=128000,flush.messages=1
        Topic: my-topic Partition: 0    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-topic Partition: 1    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-topic Partition: 2    Leader: 0       Replicas: 0     Isr: 0
$
```

修改的設定生效了!

Topic configuration example: **(Remove)**

CREATE READ UPDATE DELETE
C R U D

bin/**kafka-topics**
  --zookeeper localhost:2181
  --topic my-topic
  --alter
  --delete-config max.message.bytes


streamgeeks.org - PuTTY

```
$ bin/kafka-topics --zookeeper localhost:2181 \
>   --topic my-topic \
>   --alter \
>   --delete-config max.message.bytes
WARNING: Altering topic configuration from this script has been deprecated and may be removed in future releases.
         Going forward, please use kafka-configs.sh for this functionality
Updated config for topic "my-topic".
$ bin/kafka-topics --zookeeper localhost:2181 \
>   --topic my-topic \
>   --describe
Topic:my-topic  PartitionCount:3        ReplicationFactor:1     Configs:flush.messages=1
        Topic: my-topic Partition: 0    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-topic Partition: 1    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-topic Partition: 2    Leader: 0       Replicas: 0     Isr: 0
$
```

Kafka移除了指定的設定!

緯創IT先進技術實驗室 witlab

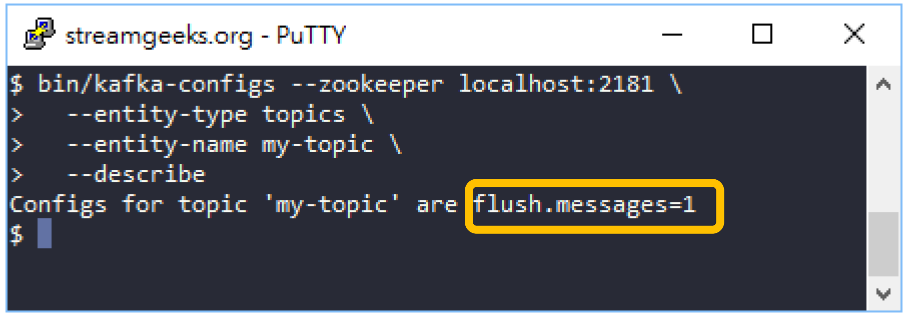
Topic configuration example: (Query)
** 2nd method - preferred**

C R U D

bin/kafka-configs
--zookeeper localhost:2181
--entity-type topics
--entity-name my-topic
--describe

使用bin/kafka-configs的工具可以讓我們對Kafka的不同的entity-type來進行設定。包括:
(topics/clients/users/brokers)

指定要修改的entity-name來進行設定。包括:
(topic name/client id/user principal name/broker id)


streamgeeks.org - PuTTY

```
$ bin/kafka-configs --zookeeper localhost:2181 \
>   --entity-type topics \
>   --entity-name my-topic \
>   --describe
Configs for topic 'my-topic' are flush.messages=1
$
```

緯創IT先進技術實驗室 witlab

Kafka configuration

CREATE READ UPDATE DELETE

C R U D

bin/kafka-configs

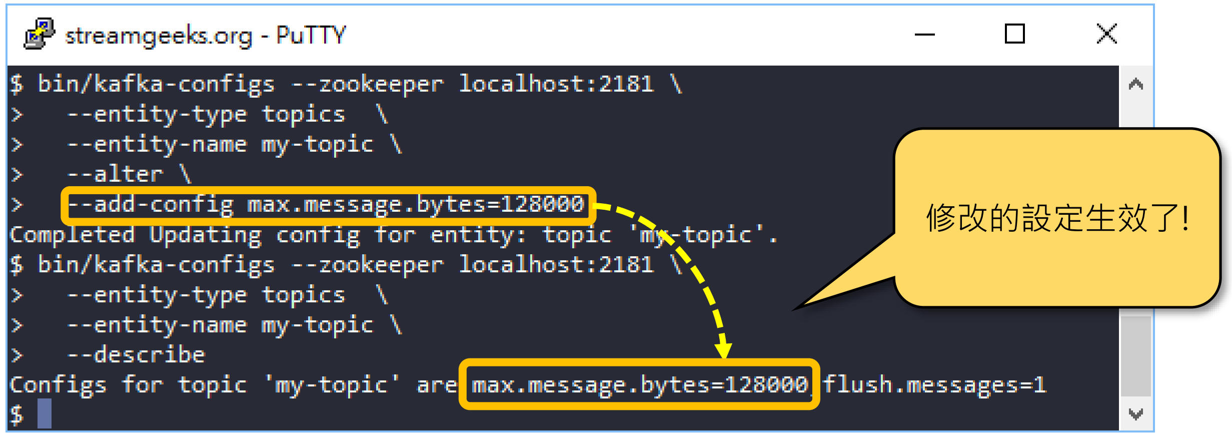
entity-type: **topics**

- cleanup.policy
- compression.type
- delete.retention.ms
- file.delete.delay.ms
- flush.messages
- flush.ms
- follower.replication.throttled.replicas
- index.interval.bytes
- leader.replication.throttled.replicas
- max.message.bytes
- message.format.version
- message.timestamp.difference.max.ms
- message.timestamp.type
- min.cleanable.dirty.ratio
- min.compaction.lag.ms
- min.insync.replicas
- preallocate
- retention.bytes
- retention.ms
- segment.bytes
- segment.index.bytes
- segment.jitter.ms
- segment.ms
- unclean.leader.election.enable

Topic configuration example: (Update)
** 2nd method - preferred**

C R U D

```
bin/kafka-configs
  --zookeeper localhost:2181
  --entity-type topics
  --entity-name my-topic
  --alter
  --add-config max.message.bytes=128000
```



streamgeeks.org - PuTTY

```
$ bin/kafka-configs --zookeeper localhost:2181 \
>    --entity-type topics   \
>    --entity-name my-topic \
>    --alter \
>    --add-config max.message.bytes=128000
Completed Updating config for entity: topic 'my-topic'.
$ bin/kafka-configs --zookeeper localhost:2181 \
>    --entity-type topics   \
>    --entity-name my-topic \
>    --describe
Configs for topic 'my-topic' are max.message.bytes=128000 flush.messages=1
$
```

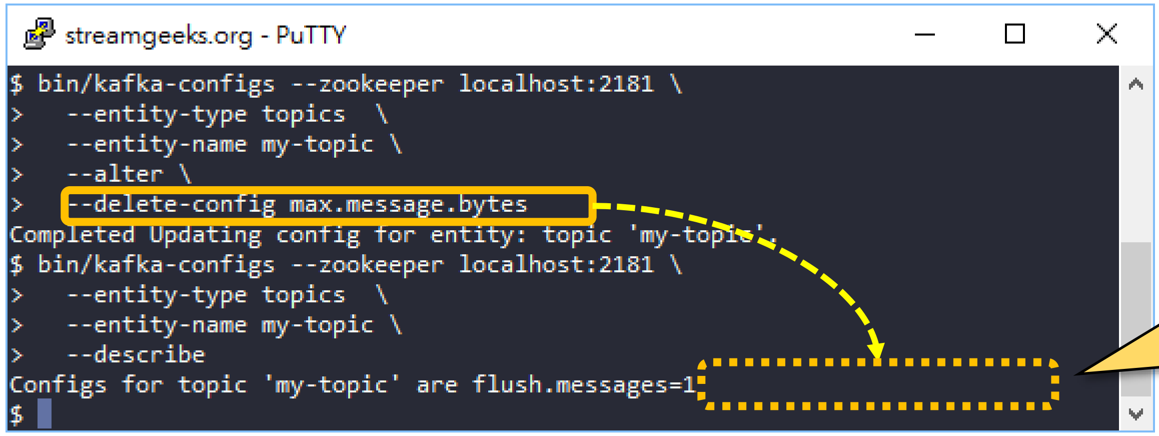
Topic configuration example: (Remove)

** 2nd method - preferred**

CREATE READ UPDATE DELETE

C R U D

```
bin/kafka-configs.sh
  --zookeeper localhost:2181
  --entity-type topics
  --entity-name my-topic
  --alter
  --delete-config max.message.bytes
```



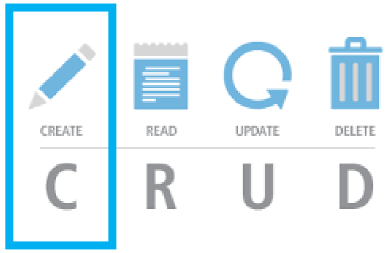
streamgeeks.org - PuTTY

```
$ bin/kafka-configs --zookeeper localhost:2181 \
>   --entity-type topics  \
>   --entity-name my-topic \
>   --alter \
>   --delete-config max.message.bytes
Completed Updating config for entity: topic 'my-topic'.
$ bin/kafka-configs --zookeeper localhost:2181 \
>   --entity-type topics  \
>   --entity-name my-topic \
>   --describe
Configs for topic 'my-topic' are flush.messages=1
$
```

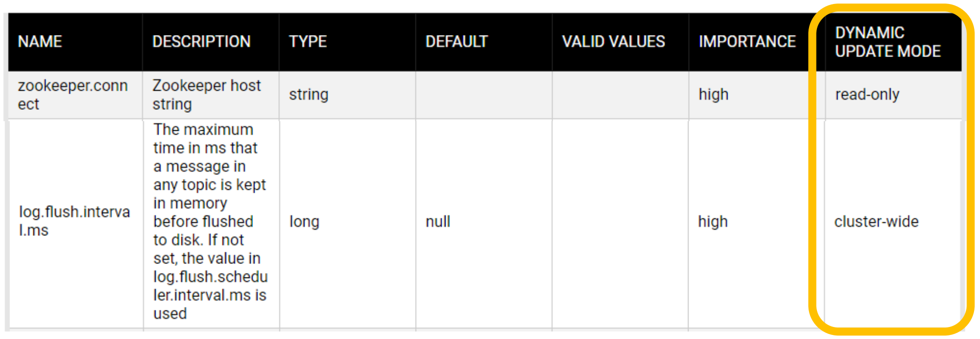
Kafka移除了指定的設定!

緯創IT先進技術實驗室 witlab

Broker configuration



- From Kafka version **1.1** onwards, some of the broker configs can be updated *without* restarting the broker.
- See the *Dynamic Update Mode* column in Broker Configs for the update mode of each broker config.
    - read-only: Requires a broker restart for update
    - per-broker: May be updated dynamically for each broker
    - cluster-wide: May be update dynamically as a cluster-wide default



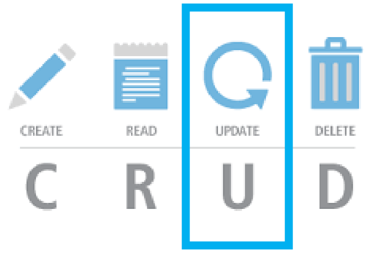
NAME	DESCRIPTION	TYPE	DEFAULT	VALID VALUES	IMPORTANCE	DYNAMIC UPDATE MODE
zookeeper.connect	Zookeeper host string	string			high	read-only
log.flush.interval.ms	The maximum time in ms that a message in any topic is kept in memory before flushed to disk. If not set, the value in log.flush.scheduler.interval.ms is used	long	null		high	cluster-wide

https://kafka.apache.org/documentation/#brokerconfigs

Kafka configuration

bin/kafka-configs

entity-type: **brokers**



- advertised.listeners
- background.threads
- compression.type
- follower.replication.throttled.rate
- leader.replication.throttled.rate
- listener.security.protocol.map
- listeners
- log.cleaner.backoff.ms
- log.cleaner.dedupe.buffer.size
- log.cleaner.delete.retention.ms
- log.cleaner.io.buffer.load.factor
- log.cleaner.io.buffer.size
- log.cleaner.io.max.bytes.per.second
- log.cleaner.min.cleanable.ratio
- log.cleaner.min.compaction.lag.ms
- log.cleaner.threads
- log.cleanup.policy
- log.flush.interval.messages
- log.flush.interval.ms
- log.index.interval.bytes
- log.index.size.max.bytes
- log.message.timestamp.difference.max.ms
- log.message.timestamp.type
- log.preallocate

Kafka configuration (Cont.)

bin/kafka-configs

entity-type: **brokers**

CREATE READ UPDATE DELETE

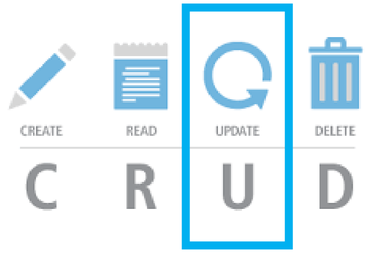
C R U D

- log.retention.bytes
- log.retention.ms
- log.roll.jitter.ms
- log.roll.ms
- log.segment.bytes
- log.segment.delete.delay.ms
- message.max.bytes
- metric.reporters
- min.insync.replicas
- num.io.threads
- num.network.threads
- num.recovery.threads.per.data.dir
- num.replica.fetchers
- principal.builder.class
- replica.alter.log.dirs.io.max.bytes.per.second
- sasl.enabled.mechanisms
- sasl.jaas.config
- sasl.kerberos.kinit.cmd
- sasl.kerberos.min.time.before.relogin
- sasl.kerberos.principal.to.local.rules
- sasl.kerberos.service.name
- sasl.kerberos.ticket.renew.jitter
- sasl.kerberos.ticket.renew.window.factor
- sasl.mechanism.inter.broker.protocol

Kafka configuration (Cont.)

bin/kafka-configs

entity-type: **brokers**

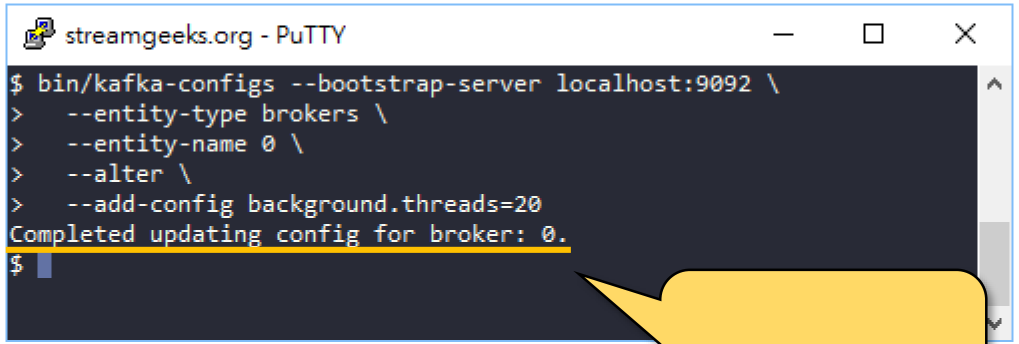


- ssl.cipher.suites
- ssl.client.auth
- ssl.enabled.protocols
- ssl.endpoint.identification.algorithm
- ssl.key.password
- ssl.keymanager.algorithm
- ssl.keystore.location
- ssl.keystore.password
- ssl.keystore.type
- ssl.protocol
- ssl.provider
- ssl.secure.random.implementation
- ssl.trustmanager.algorithm
- ssl.truststore.location
- ssl.truststore.password
- ssl.truststore.type
- unclean.leader.election.enable

Broker configuration example: **(Update)**

- To alter the current broker configs for broker id 0 (for example, the number of log cleaner threads):

```
bin/kafka-configs.sh
  --bootstrap-server localhost:9092
  --entity-type brokers
  --entity-name 0
  --alter
  --add-config background.threads=20
```



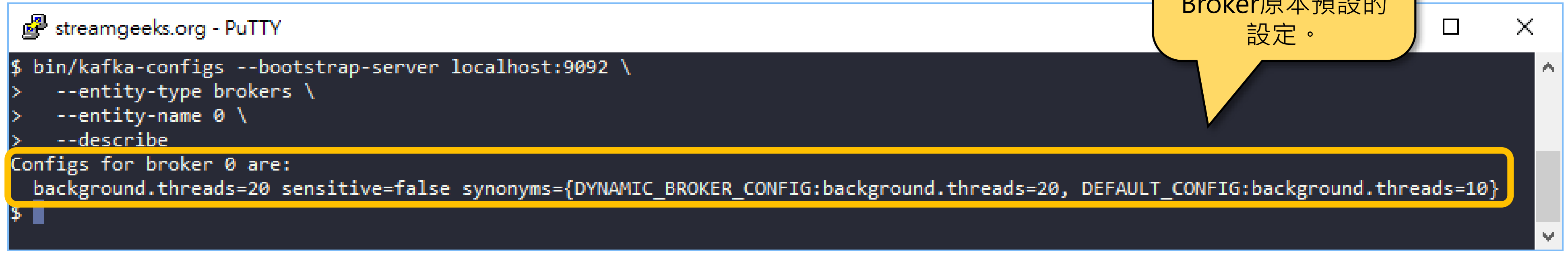
```
$ bin/kafka-configs --bootstrap-server localhost:9092 \
>   --entity-type brokers \
>   --entity-name 0 \
>   --alter \
>   --add-config background.threads=20
Completed updating config for broker: 0.
$
```

修改成功!

Broker configuration example: (**Query**)

C R U D

bin/**kafka-configs.sh**
 --bootstrap-server localhost:9092
--entity-type brokers
--entity-name 0
--describe



streamgeeks.org - PuTTY

```
$ bin/kafka-configs --bootstrap-server localhost:9092 \
>   --entity-type brokers \
>   --entity-name 0 \
>   --describe
Configs for broker 0 are:
  background.threads=20 sensitive=false synonyms={DYNAMIC_BROKER_CONFIG:background.threads=20, DEFAULT_CONFIG:background.threads=10}
$
```

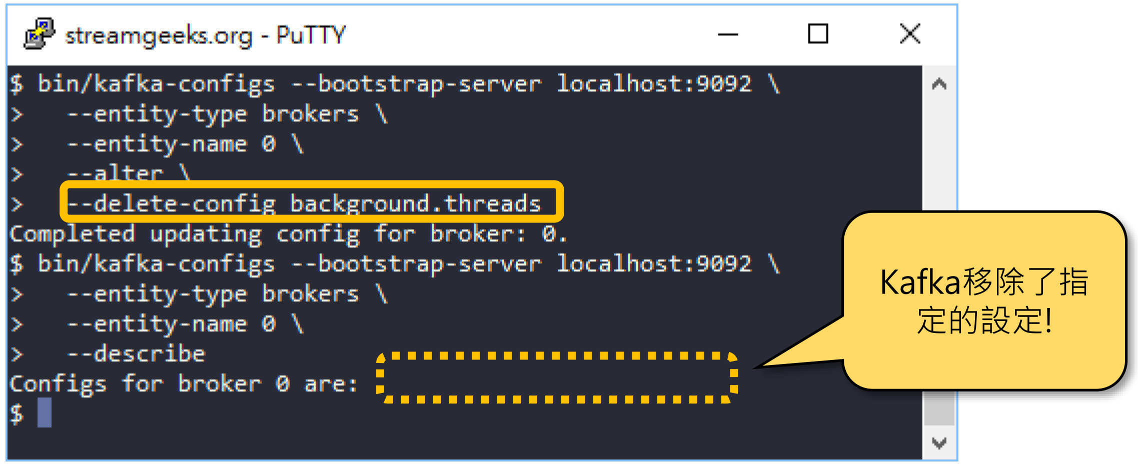
修改成功!
從UI上可以看到設定的結果以及Broker原本預設的設定。

Broker configuration example: (**Remove**)

CREATE READ UPDATE DELETE
C R U D

- To delete a config override and revert to the statically configured or default value for broker id 0 (for example, the number of log cleaner threads):

```
bin/kafka-configs.sh
  --bootstrap-server localhost:9092
  --entity-type topics
  --entity-name my-topic
  --alter
  --delete-config background.threads
```



```
$ bin/kafka-configs --bootstrap-server localhost:9092 \
>   --entity-type brokers \
>   --entity-name 0 \
>   --alter \
>   --delete-config background.threads
Completed updating config for broker: 0.
$ bin/kafka-configs --bootstrap-server localhost:9092 \
>   --entity-type brokers \
>   --entity-name 0 \
>   --describe
Configs for broker 0 are:
$
```

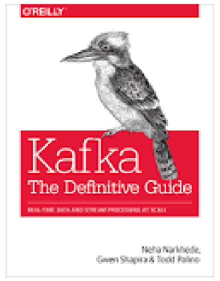
緯創IT先進技術實驗室 witlab

Where are these configuration stored?

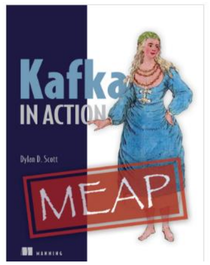
- Dynamic per-broker configurations stored in ZooKeeper
- Dynamic cluster-wide default configurations stored in ZooKeeper
- Static broker configurations from server.properties
- Kafka default, see broker configurations
  - https://kafka.apache.org/documentation/#brokerconfigs

Reference

1. Kafka: The Definitive Guide – O'REILLY



2. Kafka In Action - MANNING



3. Learn Apache Kafka for Beginners – Udemy (Stephane Maarek)

Apache Kafka Series - Learn Apache Kafka for Beginners v2
Stephane Maarek, Best Selling Instructor, Kafka Guru, AWS...

4. Confluent Document of Kafka – confluent.io

confluent

緯創IT先進技術實驗室 witlab

The End