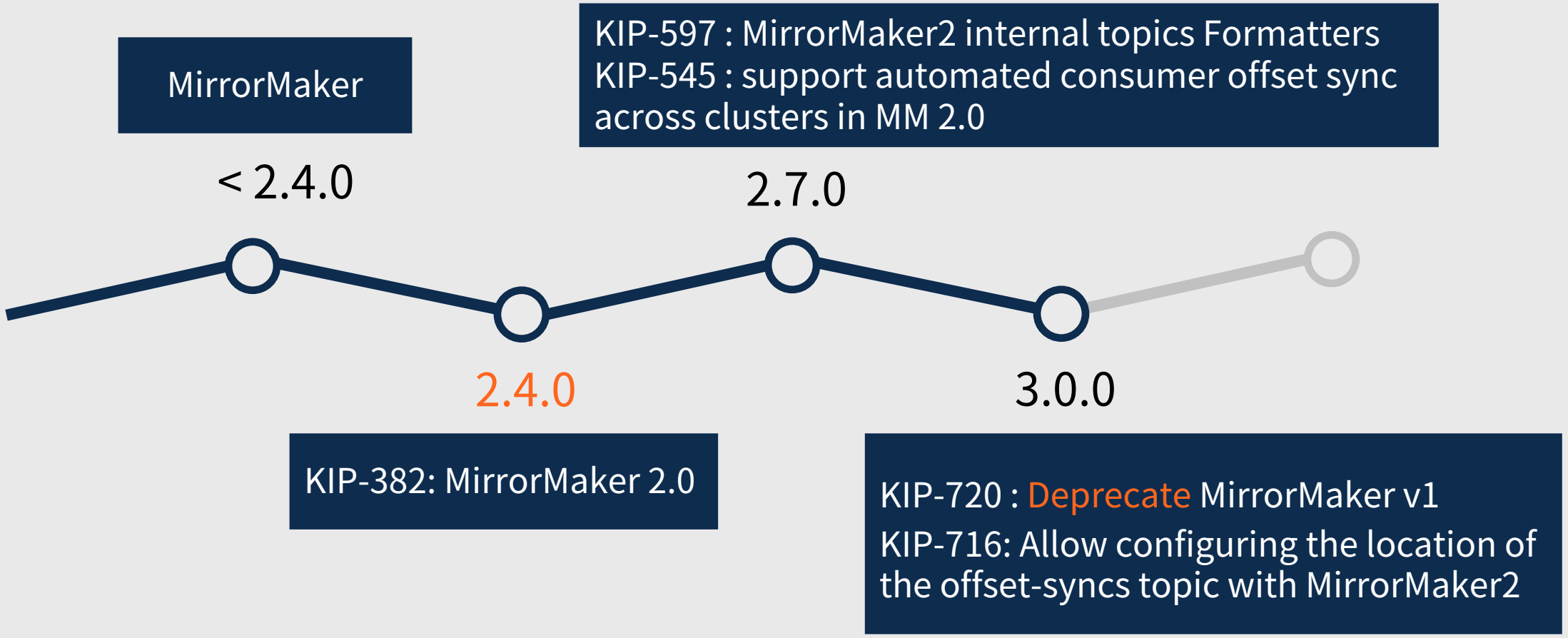
MirrorMaker2 소개

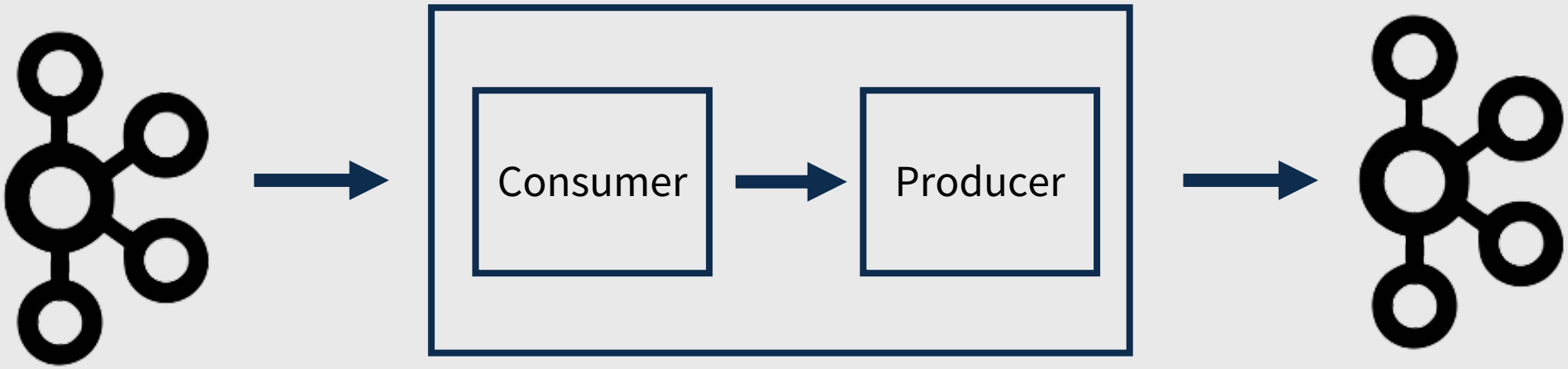
최중호

목 차

MirrorMaker2 등장



MirrorMaker

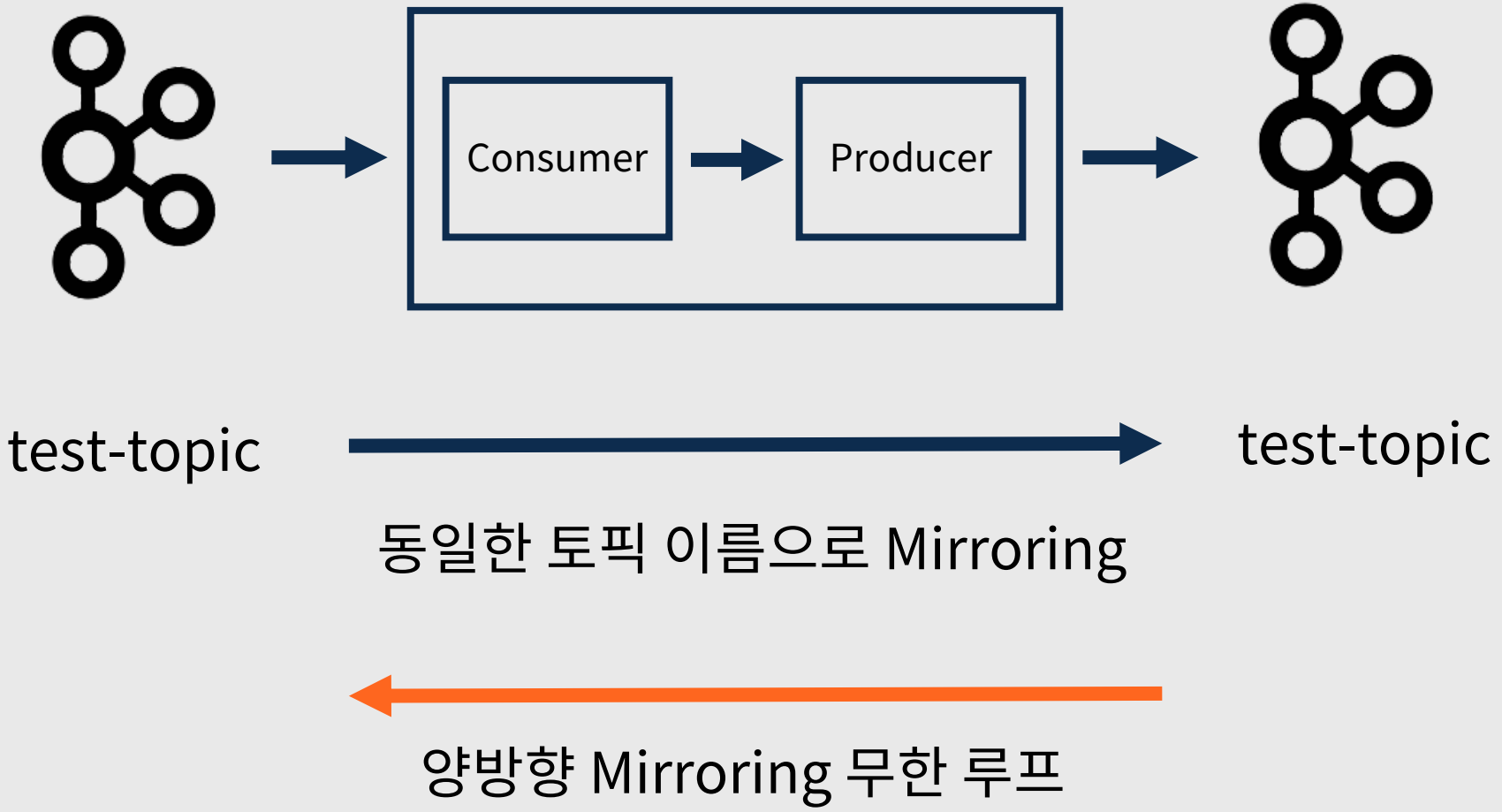



```
./bin/kafka-run-class.sh –daemon –name mirrormaker-v1 kafka.tools.MirrorMaker \
--consumer.config consumer.properties –producer.config producer.properties \
--whitelist= "test-topic"
```

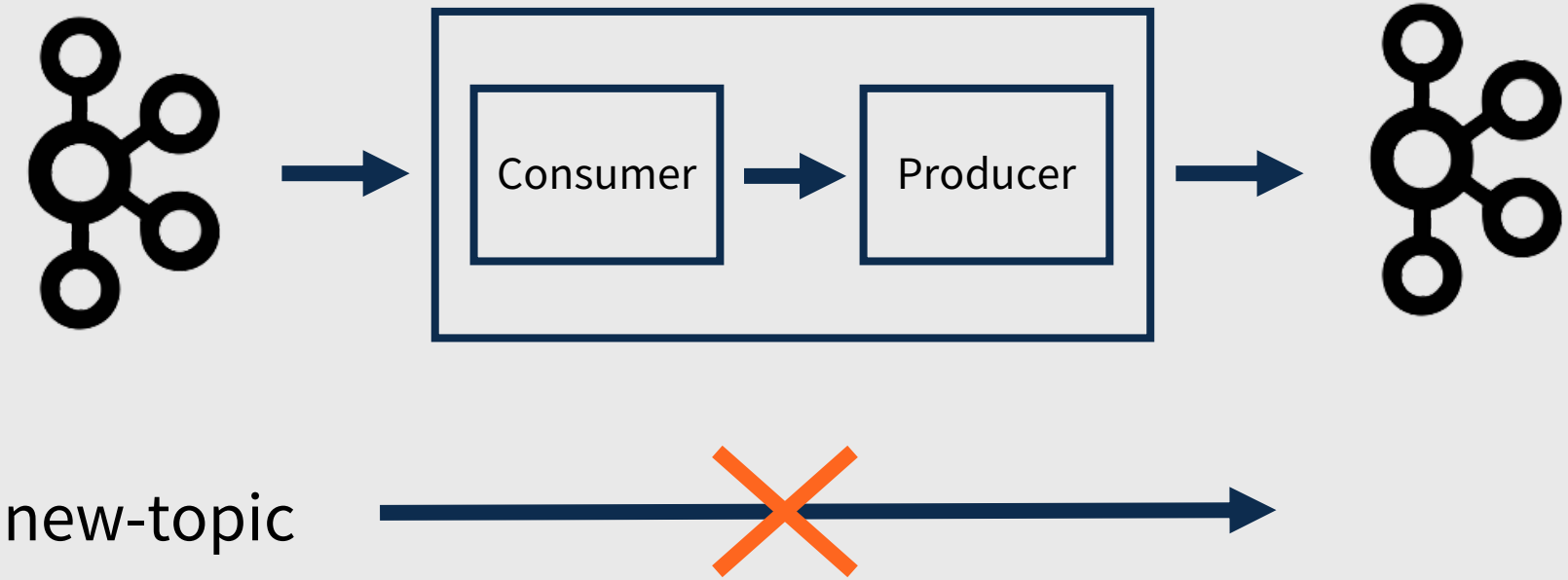
MirrorMaker 단점

- 양방향
- 신규 토픽
- 파티션
- 컨슈머 그룹

MirrorMaker - 양방향

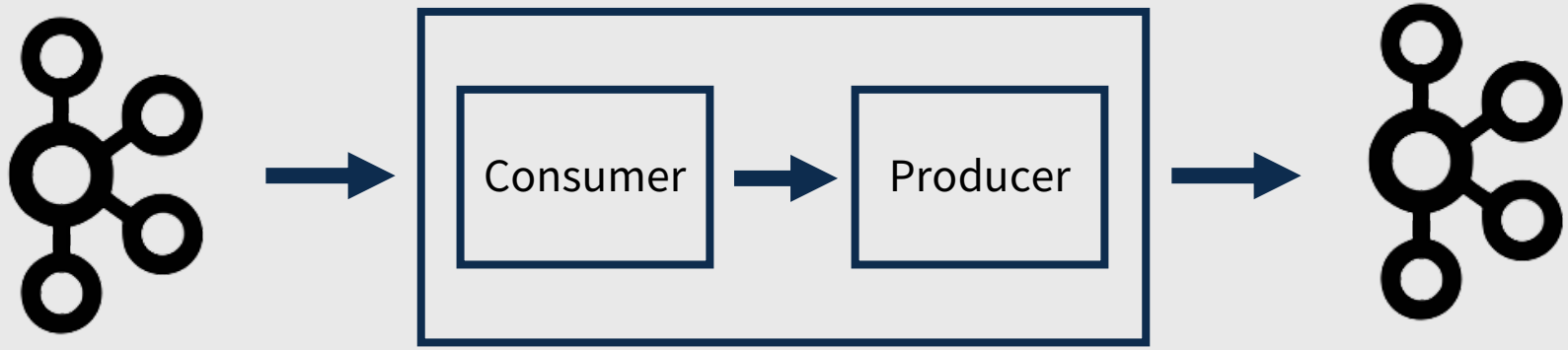


MirrorMaker - 신규 토픽




```
./bin/kafka-run-class.sh –daemon –name mirrormaker-v1 kafka.tools.MirrorMaker \
--consumer.config consumer.properties –producer.config producer.properties \
--whitelist= "test-topic"
```

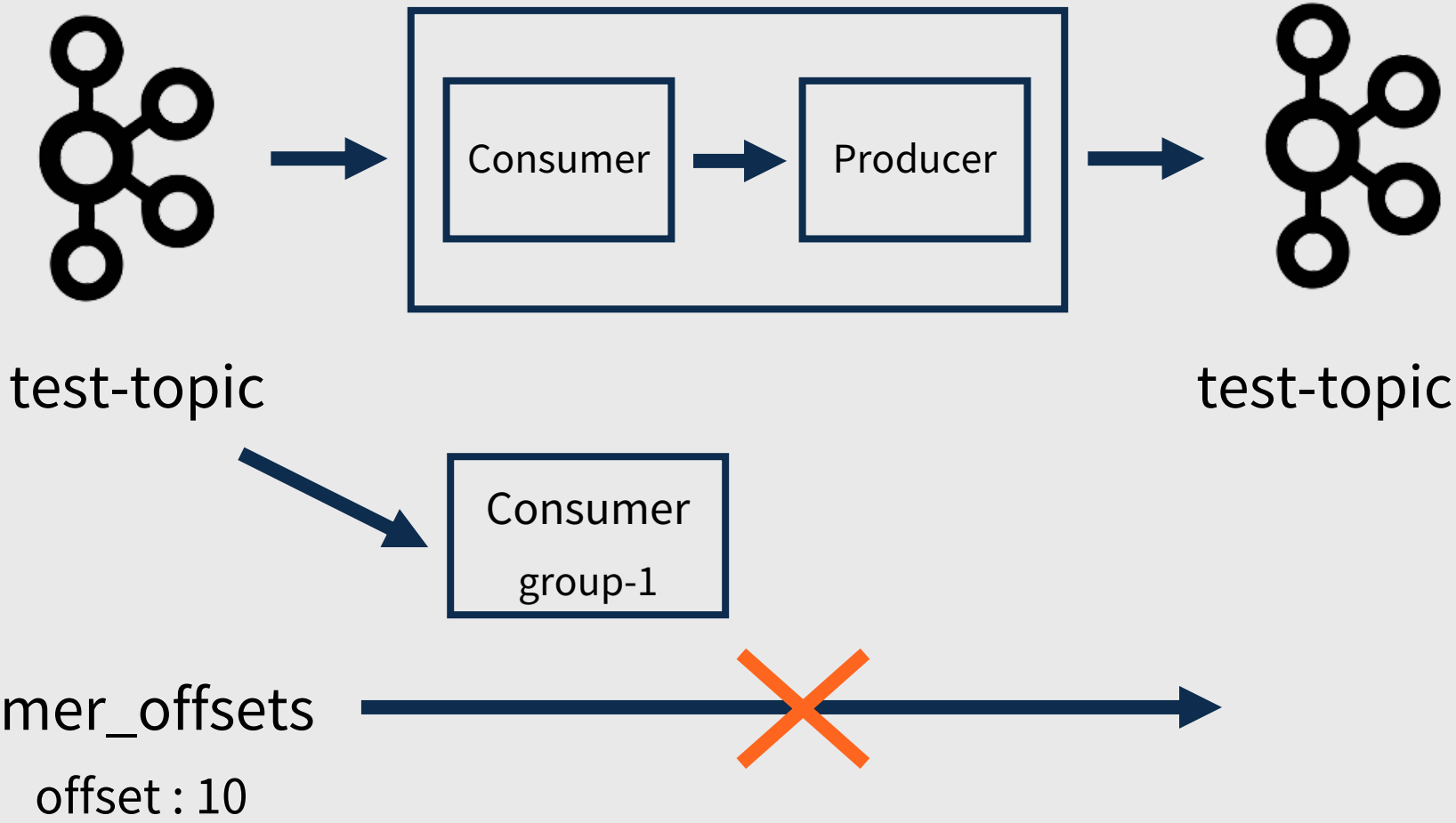
MirrorMaker - 파티션



test-topic
partition 1개

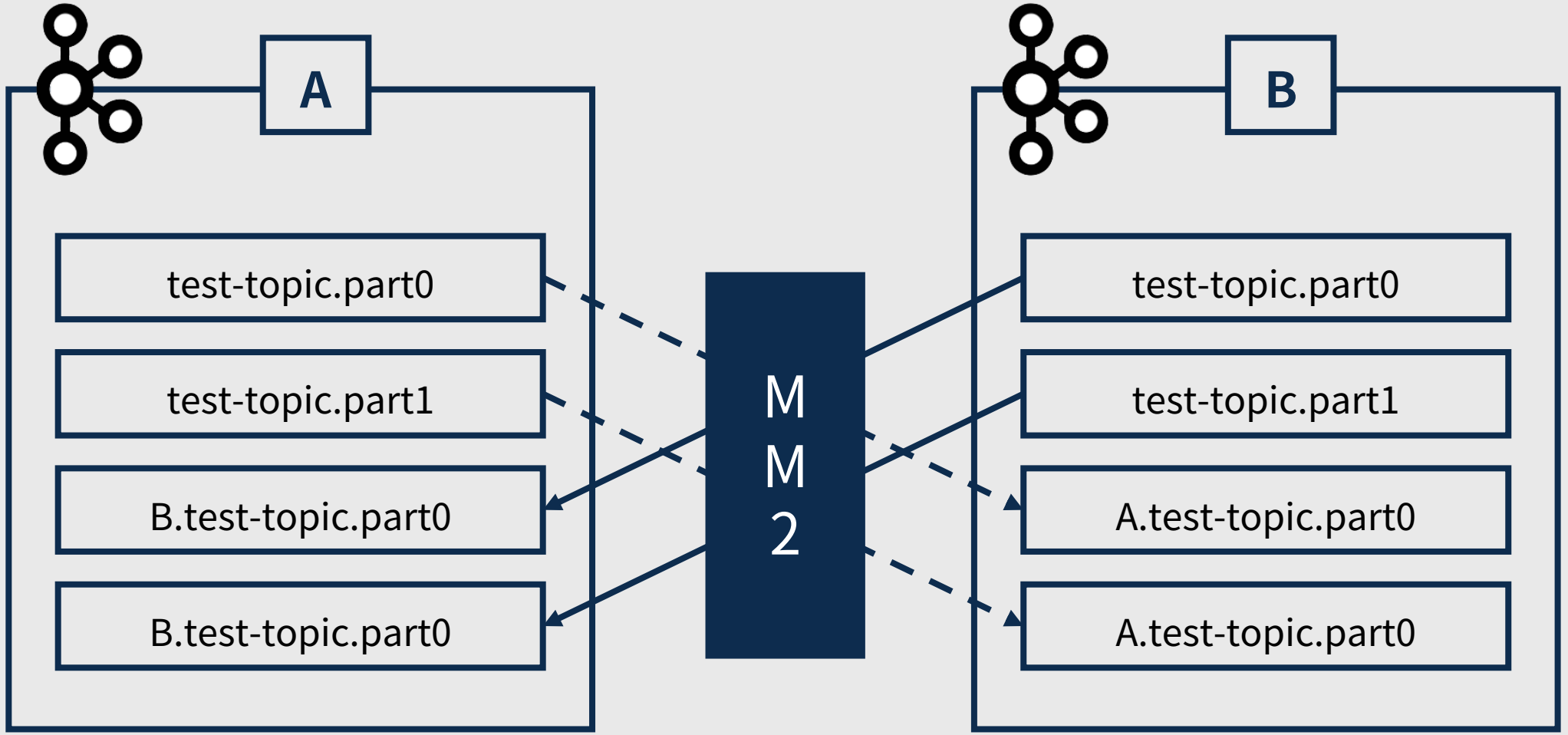
test-topic
partition 2개

MirrorMaker - 컨슈머 그룹



MirrorMaker2

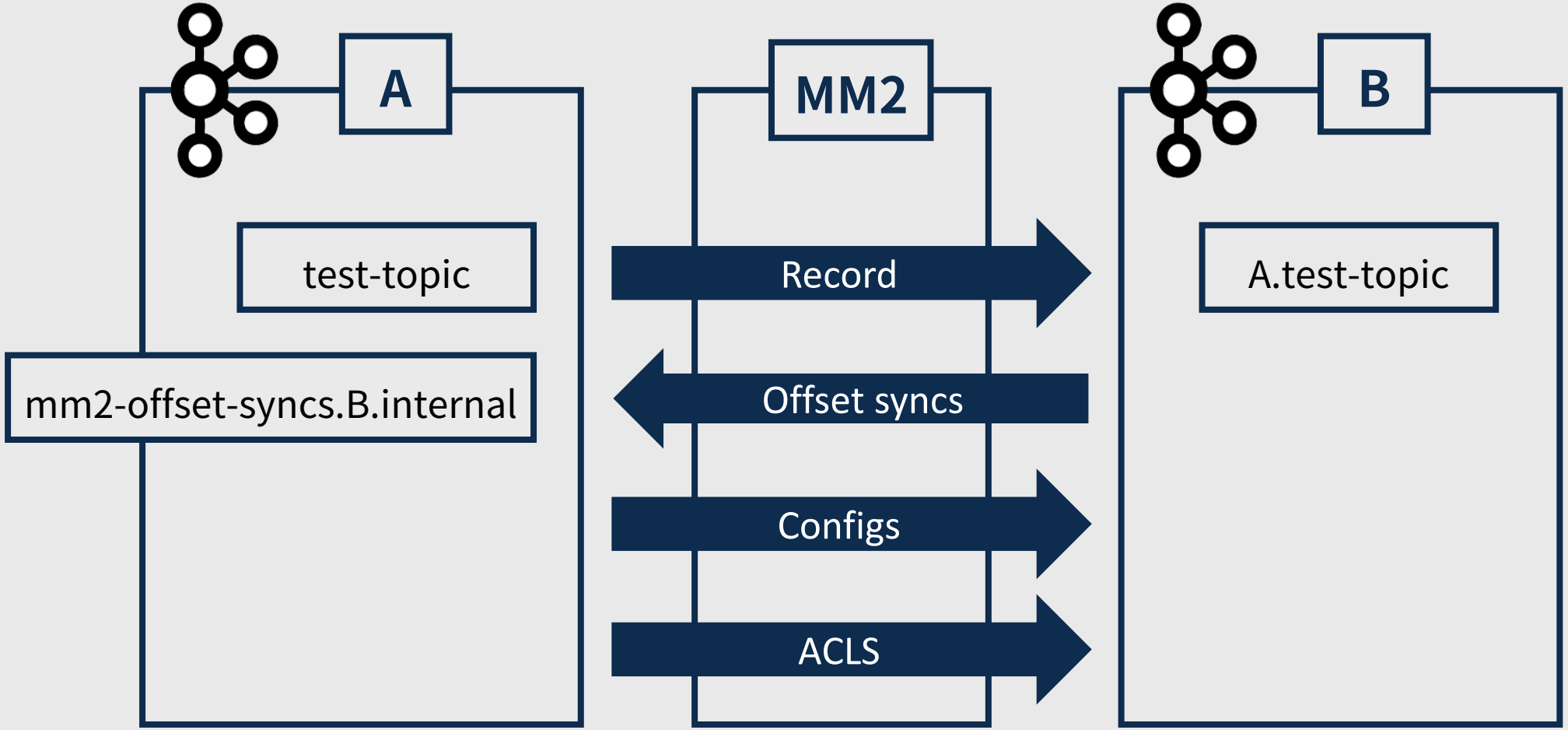
MirrorMaker2 - Active/Active



MirrorMaker2 - 구성

- Kafka **Connect** 기반
- 3개의 Connector로 구성
  - **MirrorSourceConnector**
    - topic의 데이터를 복제
    - topic의 파티션 정보 및 ACLs 동기화
  - **MirrorCheckpointConnector**
    - consumer group offset 동기화
  - **MirrorHeartbeatConnector**
    - 복제의 모니터링을 위한 Heartbeat 전송

MirrorMaker2 - MirrorSourceConnector



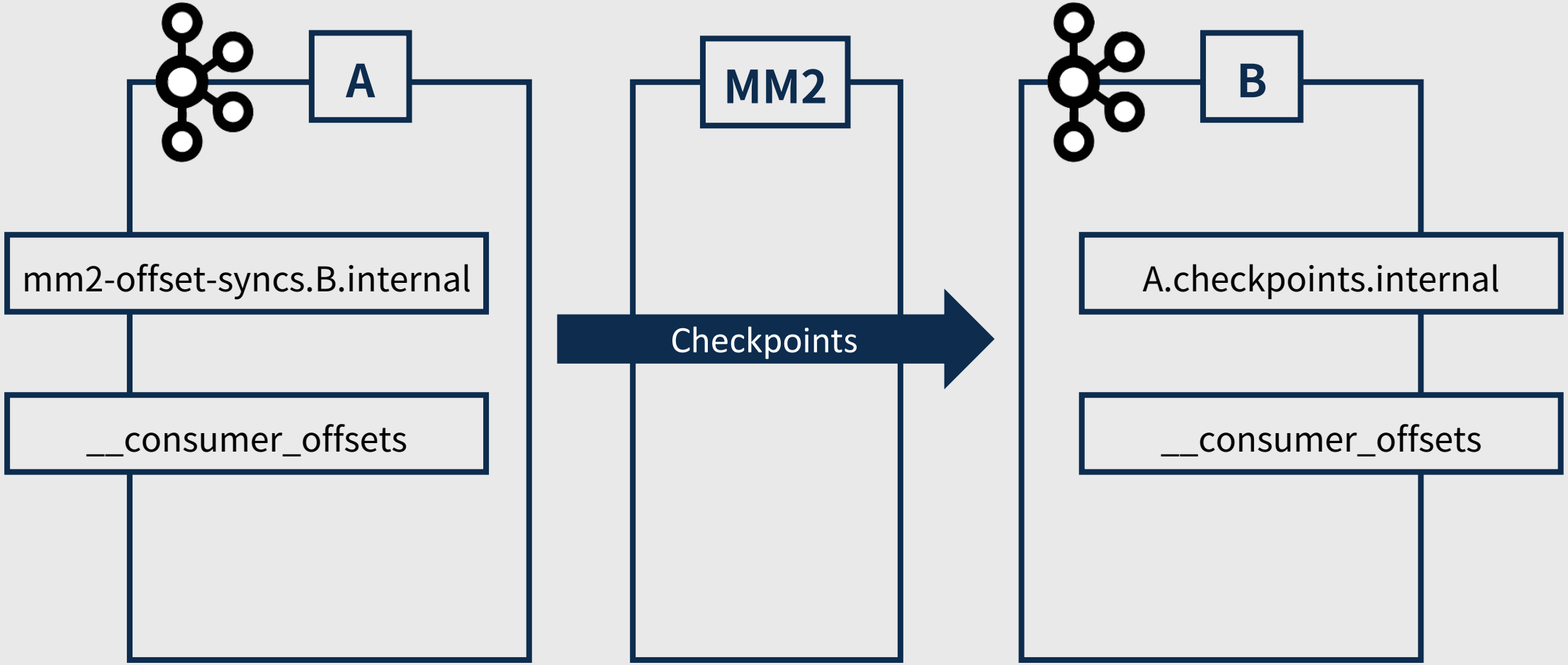
MirrorMaker2 – MirrorSourceConnector

- KIP-597: MirrorMaker2 internal topics Formatters

```
./bin/kafka-console-consumer.sh –bootstrap-server A:9092 \
  --topic mm2-offset-syncs.B.internal \
  --formatter org.apache.kafka.connect.mirror.formatters.OffsetSyncFormatter
```

```
OffsetSync{topicPartition=test-topic-0,
upstreamOffset=224, downstreamOffset=65}
```

MirrorMaker2 - MirrorCheckpointConnector



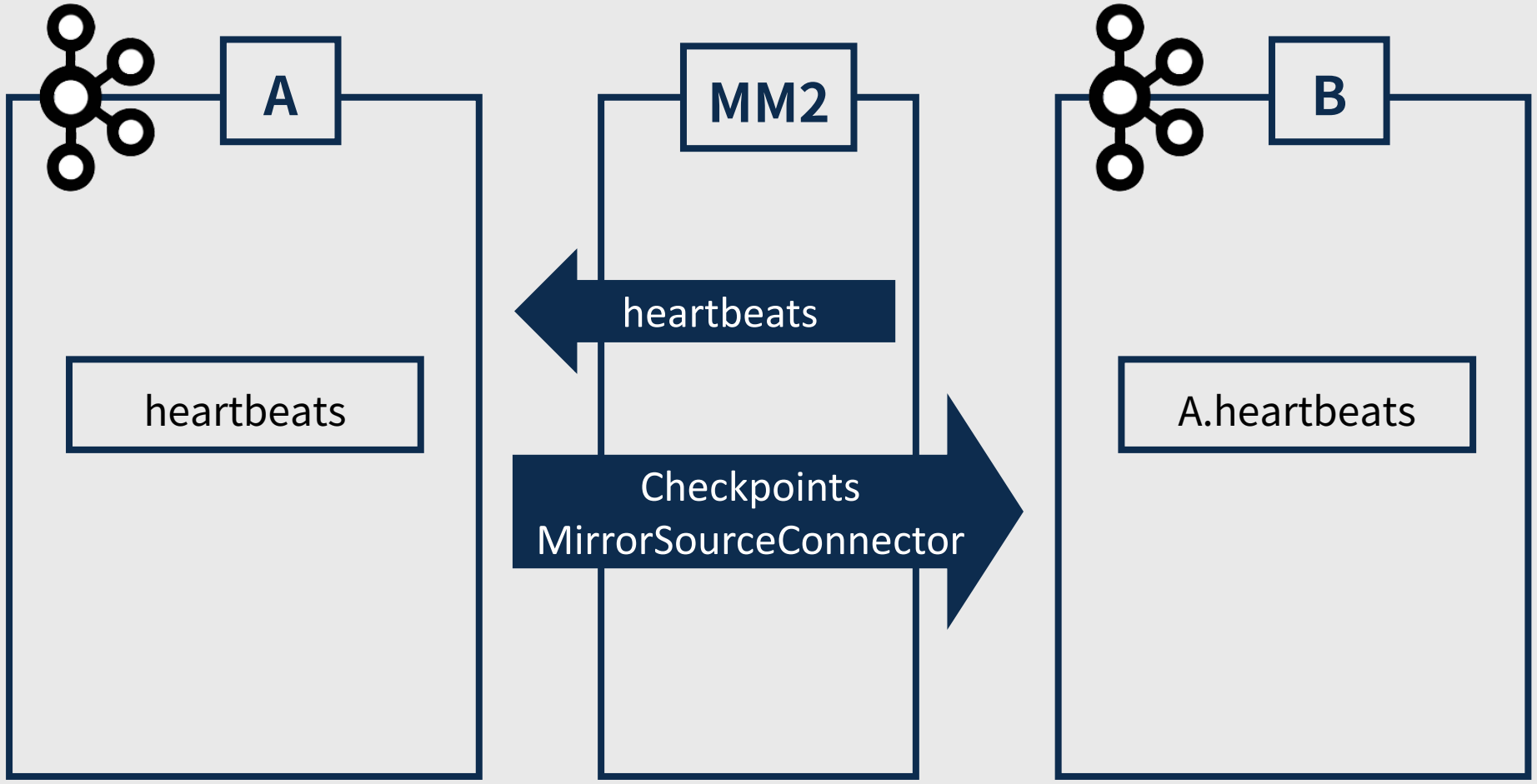
MirrorMaker2 - MirrorCheckpointConnector

- KIP-545: support automated consumer offset sync across clusters in MM 2.0

```
A->B.sync.group.offsets.enabled = true
```

- A 카프카의 컨슈머 오프셋을 B 카프카 오프셋으로 변환한다.
  - 변환할 때 A/B의 오프셋 차이는 mm2-offset-syncs.B.internal로 판별
- 주기적으로 __consumer_offsets에 동기화한다.

MirrorMaker2 - MirrorHeartbeatConnector



MirrorMaker2 - 실행 방법

- Dedicated MirrorMaker cluster
  - MirrorMaker2 전용 클러스터
  - 별도의 properties 구성 후 실행
- Running MirrorMaker in Connect cluster
  - Connect Cluster에 3개의 Connector를 각각 실행

MirrorMaker2 - dedicated cluster (Active/Standby)

mm2.properties

```
clusters = source, target
source.bootstrap.servers = source:9092
target.bootstrap.servers = target:9092

source->target.enabled = true
source->target.topics = .*
source->target.group = .*
source->target.sync.group.offsets.enabled = true

target->source.enabled = false
```

./bin/connect-mirror-maker.sh mm2.properties

MirrorMaker2 - Connect Cluster (Active/Standby)

MirrorSourceConnector

```
{
  "name": "mm2-msc",
  "config": {
    "name": "mm2-msc",
    "connector.class": "org.apache.kafka.connect.mirror.MirrorSourceConnector",
    "clusters": "source, target",
    "source.cluster.alias": "source",
    "source.cluster.bootstrap.servers": "source_kafka:9092",
    "target.cluster.alias": "target",
    "target.cluster.bootstrap.servers": "target_kafka:9092",
    "topics": ".*",
    "tasks.max": "5"
  }
}
```

POST http://connect_url/connectors

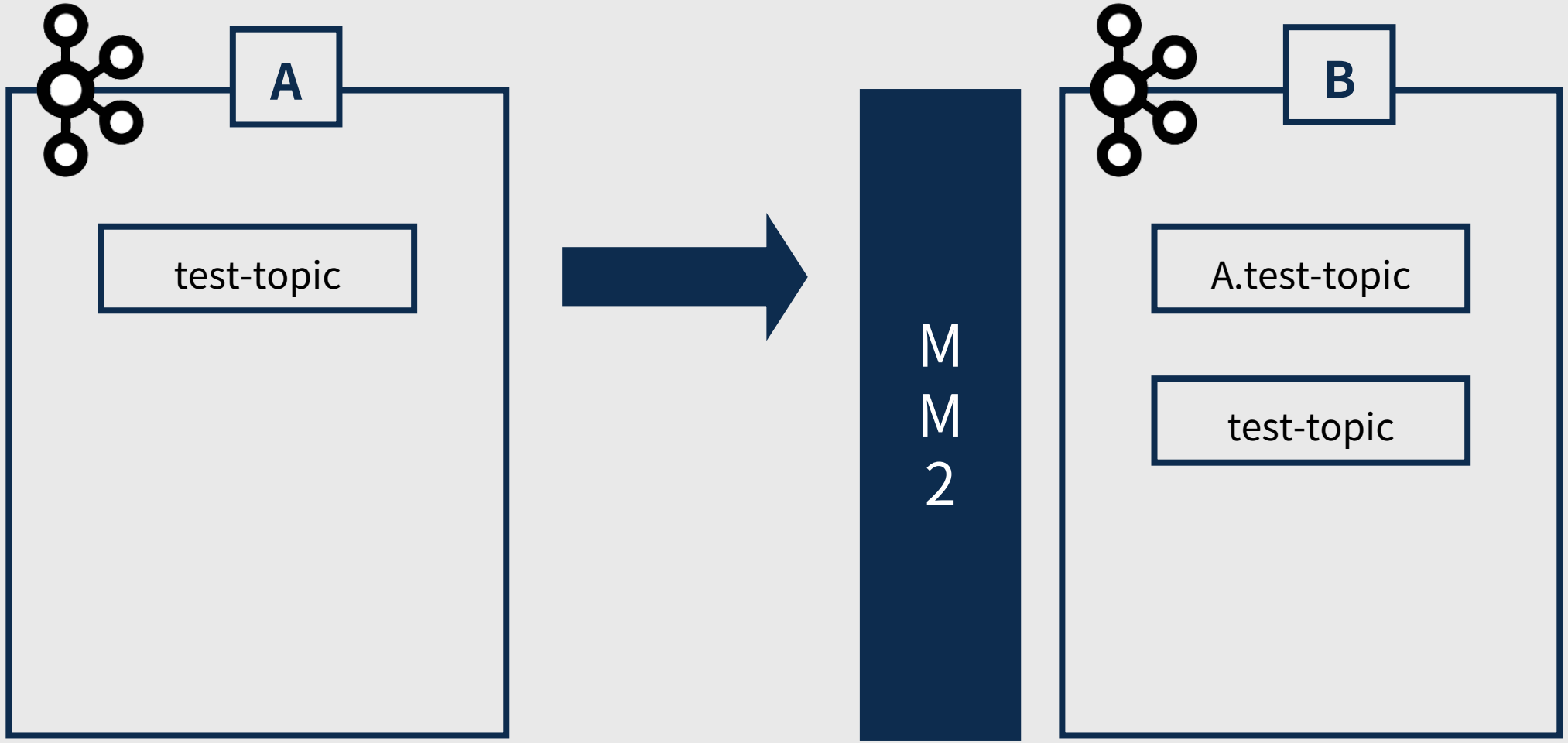
MirrorMaker2 - Connect Cluster (Active/Standby)

MirrorCheckpointConnector

```
{
  "name": "mm2-cpc",
  "config":
  {
    "name": "mm2-cpc",
    "connector.class": "org.apache.kafka.connect.mirror.MirrorCheckpointConnector",
    "clusters": "source, target",
    "source.cluster.alias": "source",
    "source.cluster.bootstrap.servers": "source_kafka:9094",
    "target.cluster.alias": "target",
    "target.cluster.bootstrap.servers": "target_kafka:9094",
    "groups": ".*",
    "tasks.max": "5"
  }
}
```

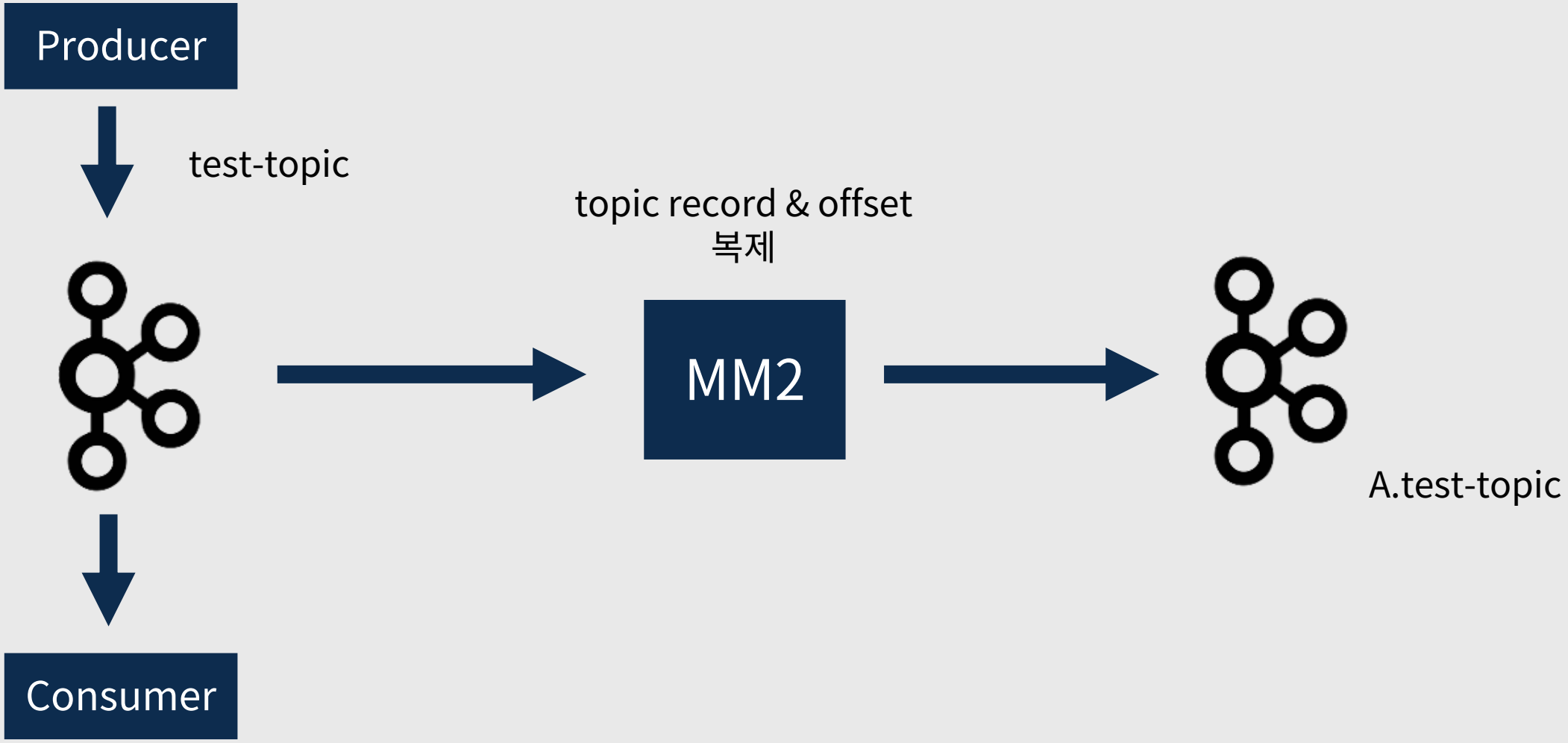
POST http://connect_url/connectors

MirrorMaker2 - Active/ Standby

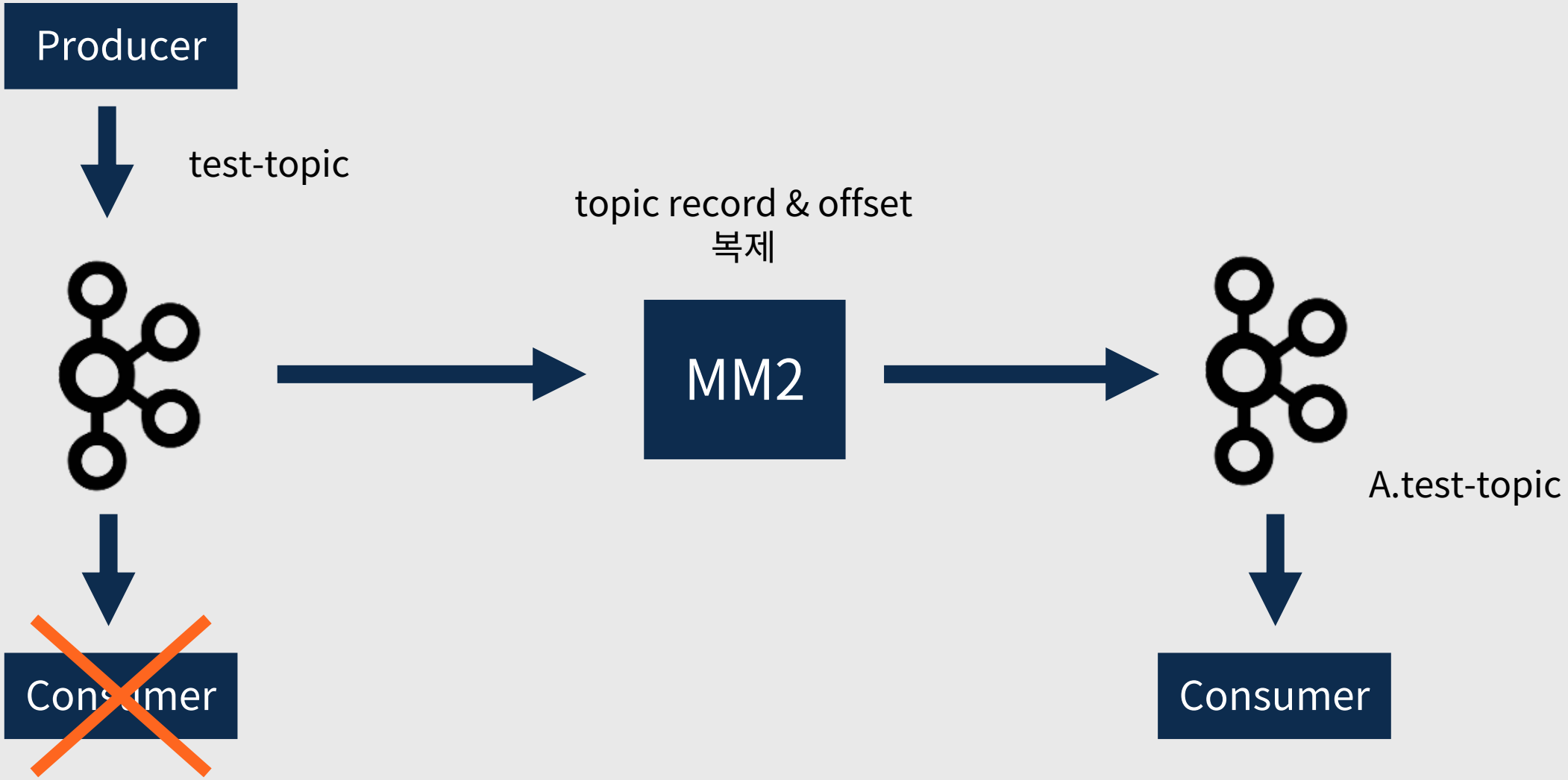


MirrorMaker2 활용

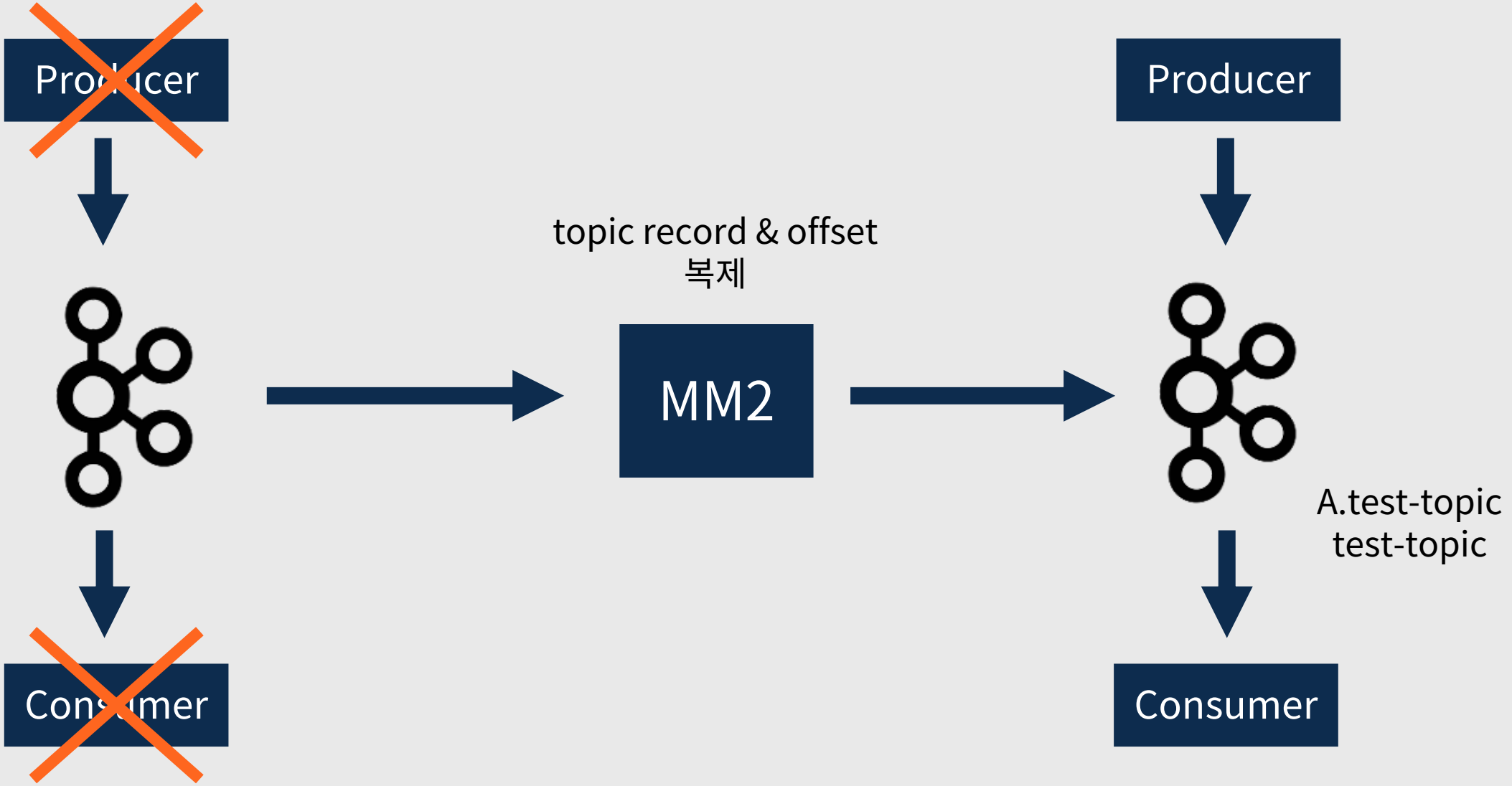
use case 1 - Live Migration



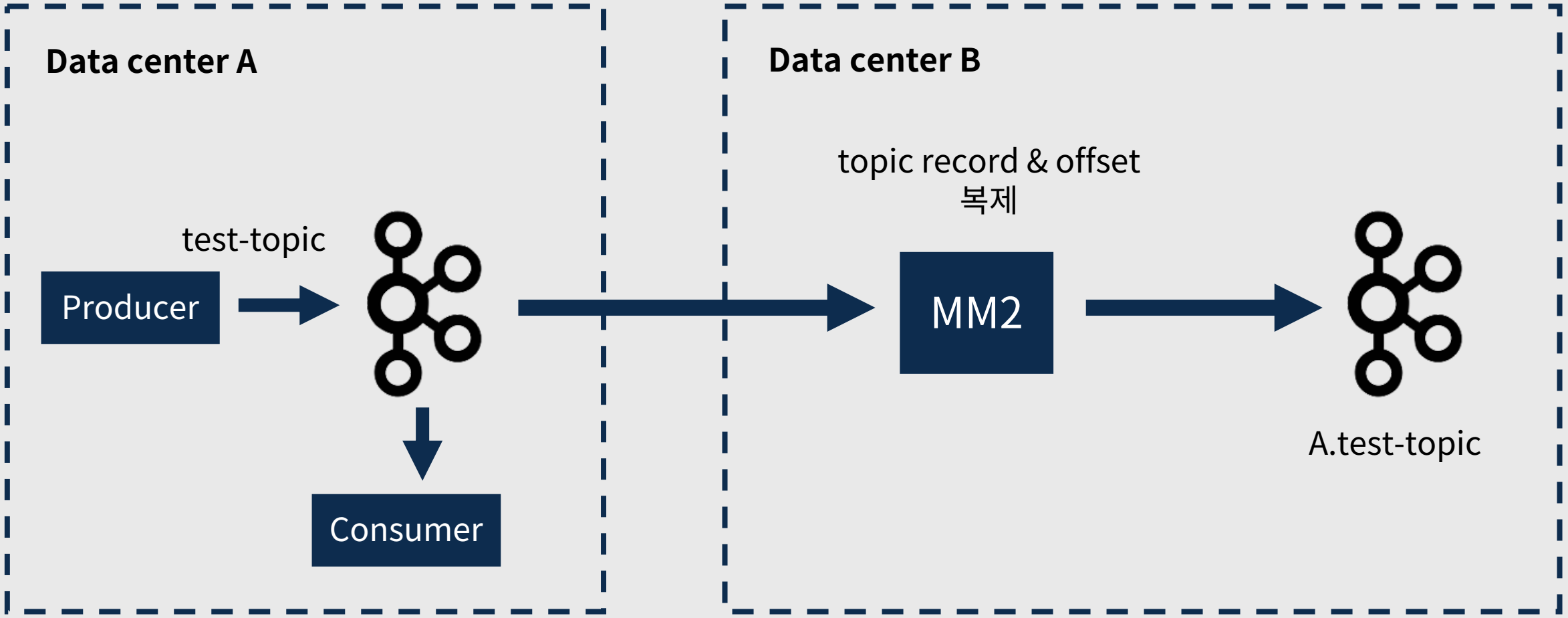
use case 1 - Live Migration



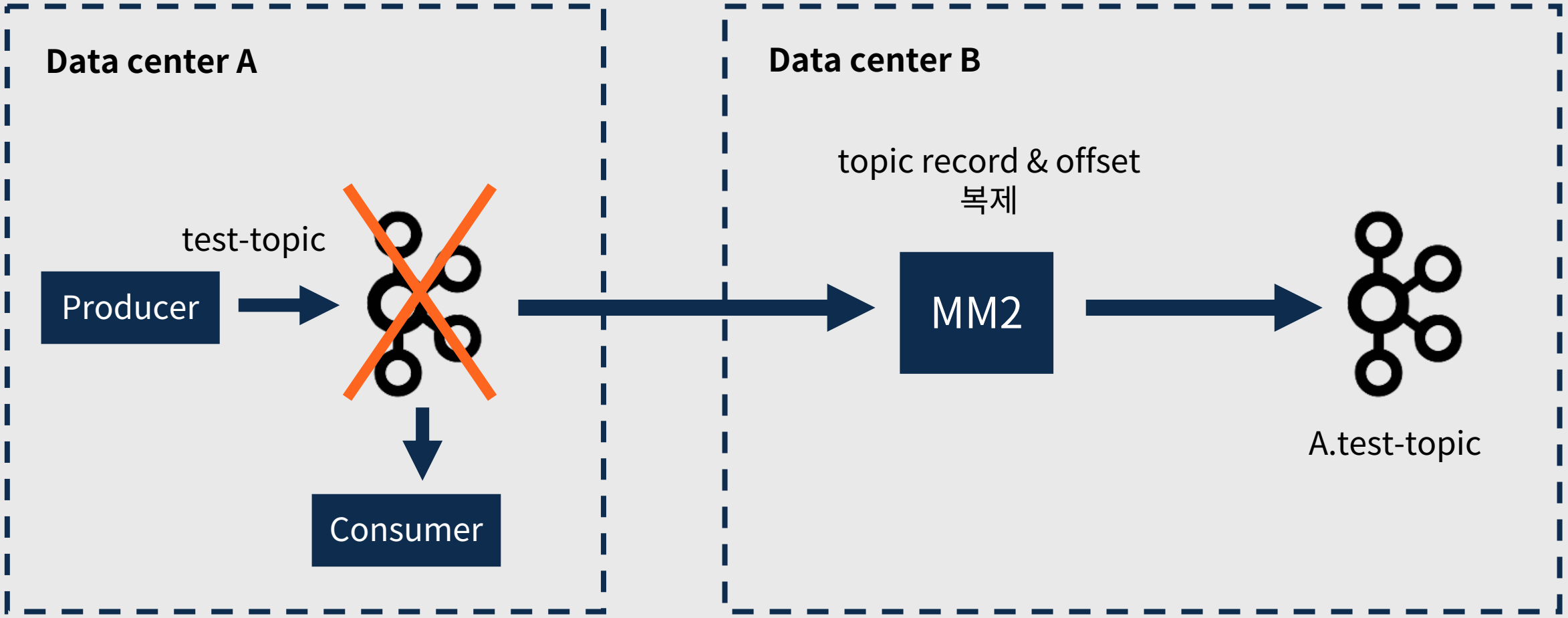
use case 1 - Live Migration



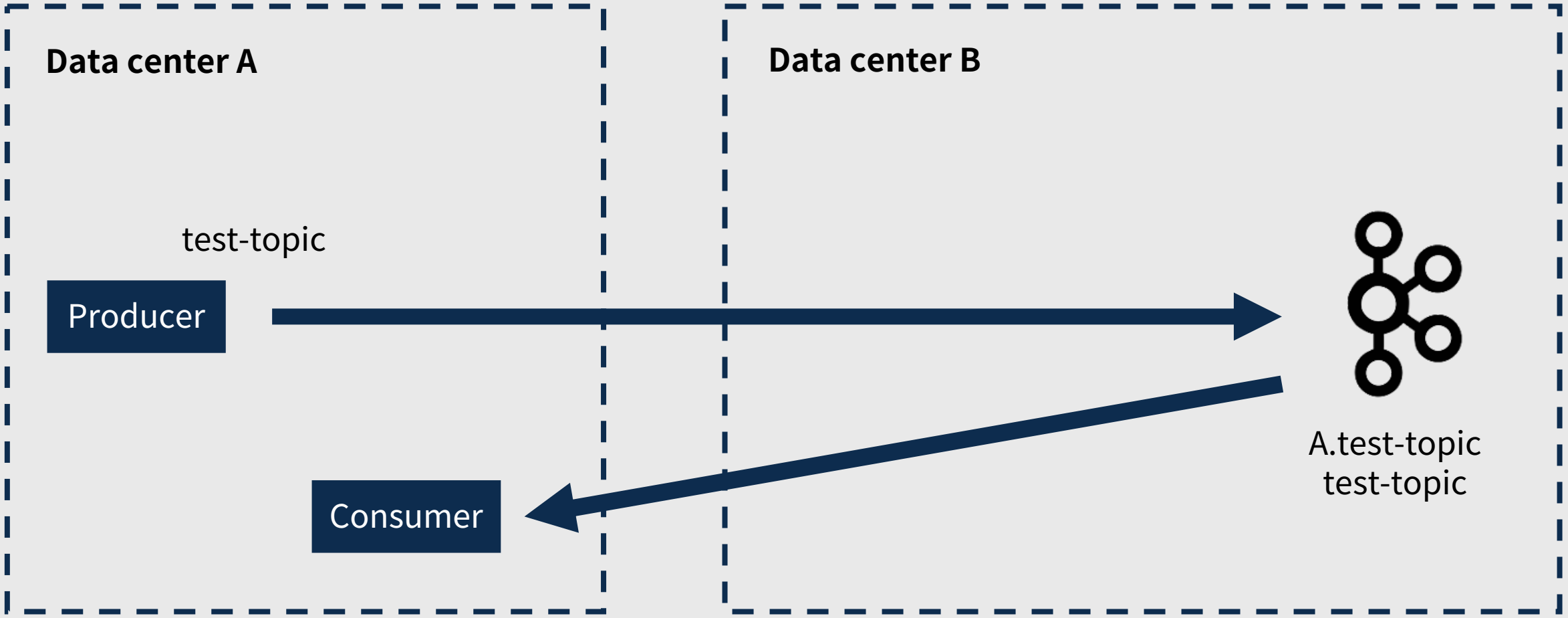
use case 2 - Active/Standby



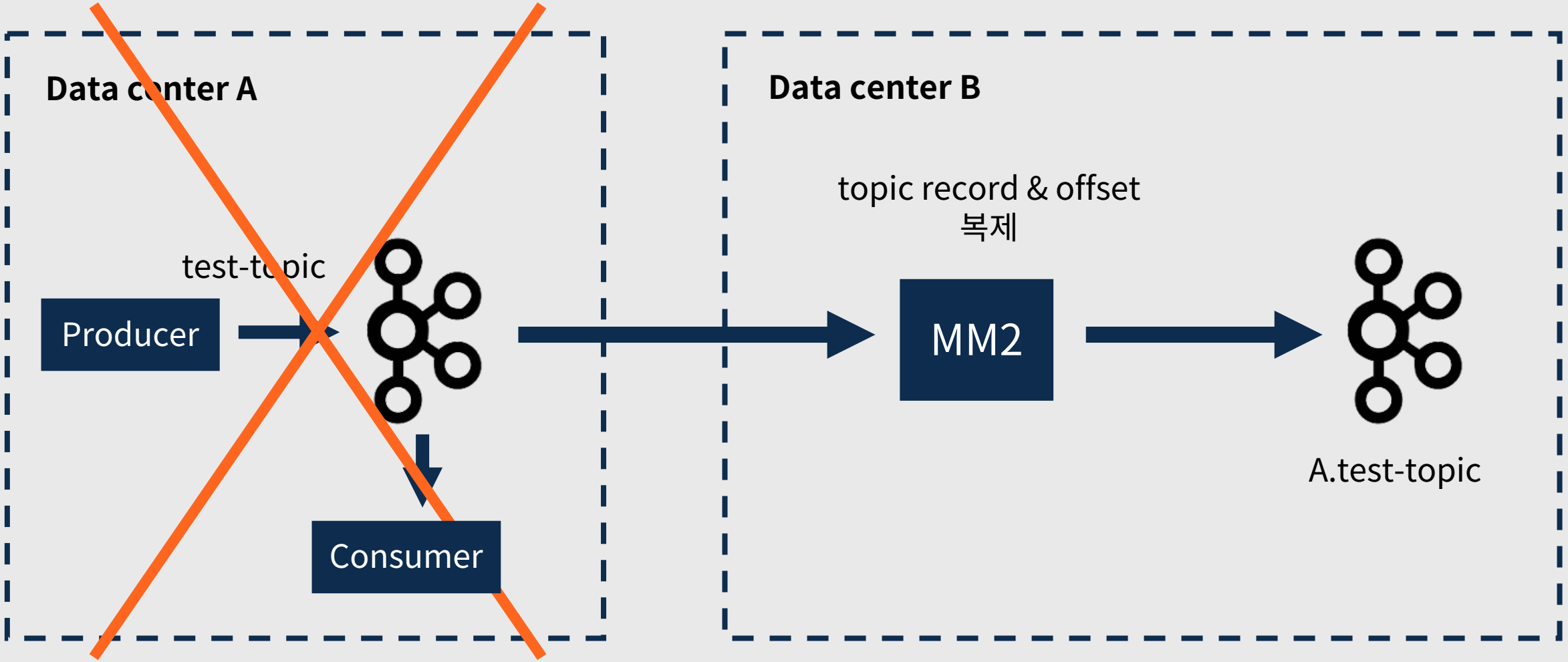
use case 2 - Active/Standby



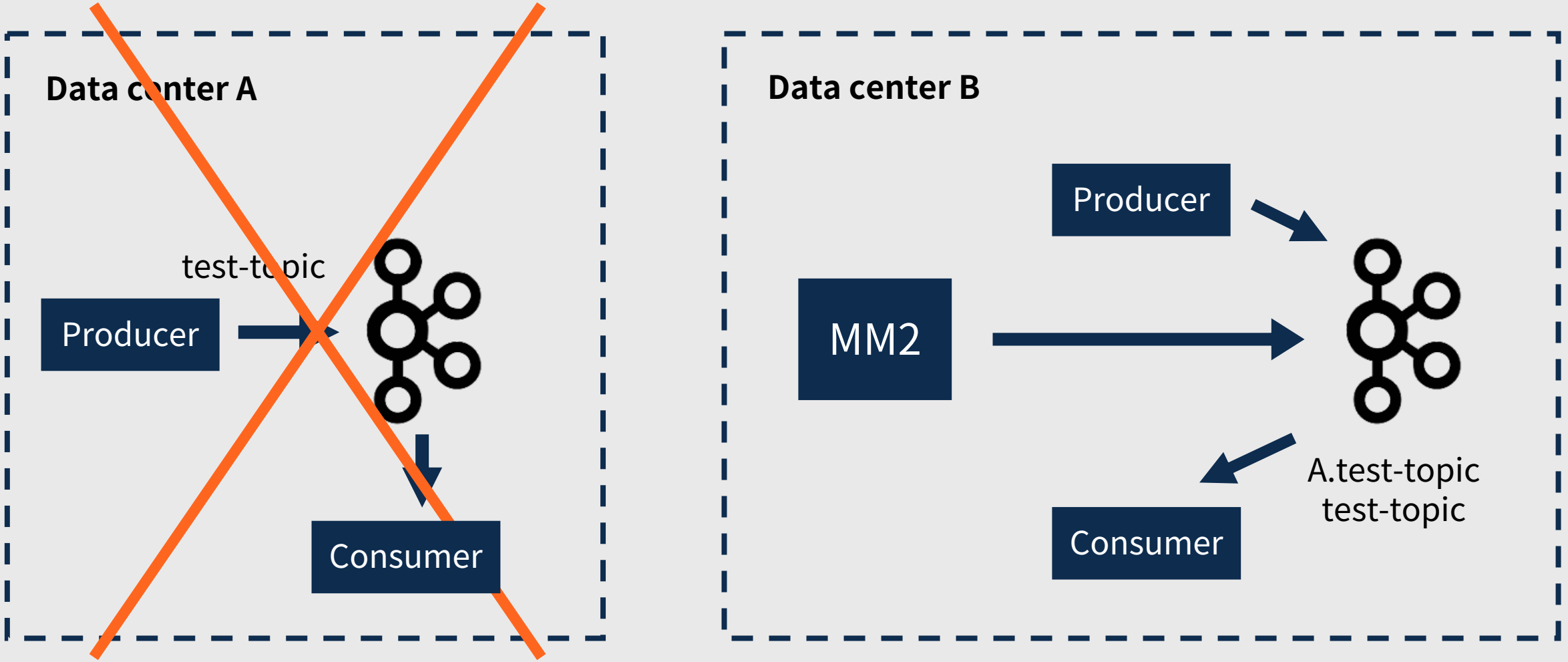
use case 2 - Active/Standby



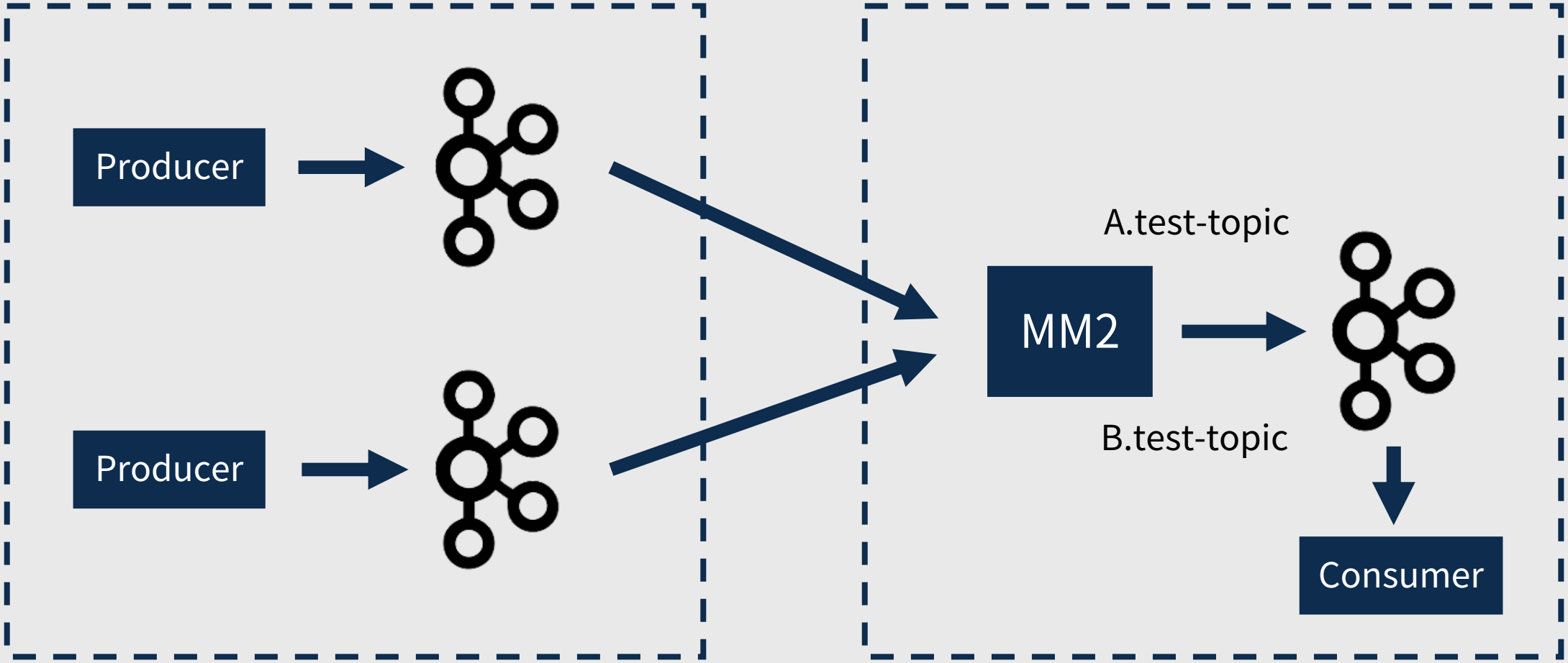
use case 2 - Active/Standby



use case 2 - Active/Standby



use case 3 - Aggregation



MirrorMaker2 모니터링

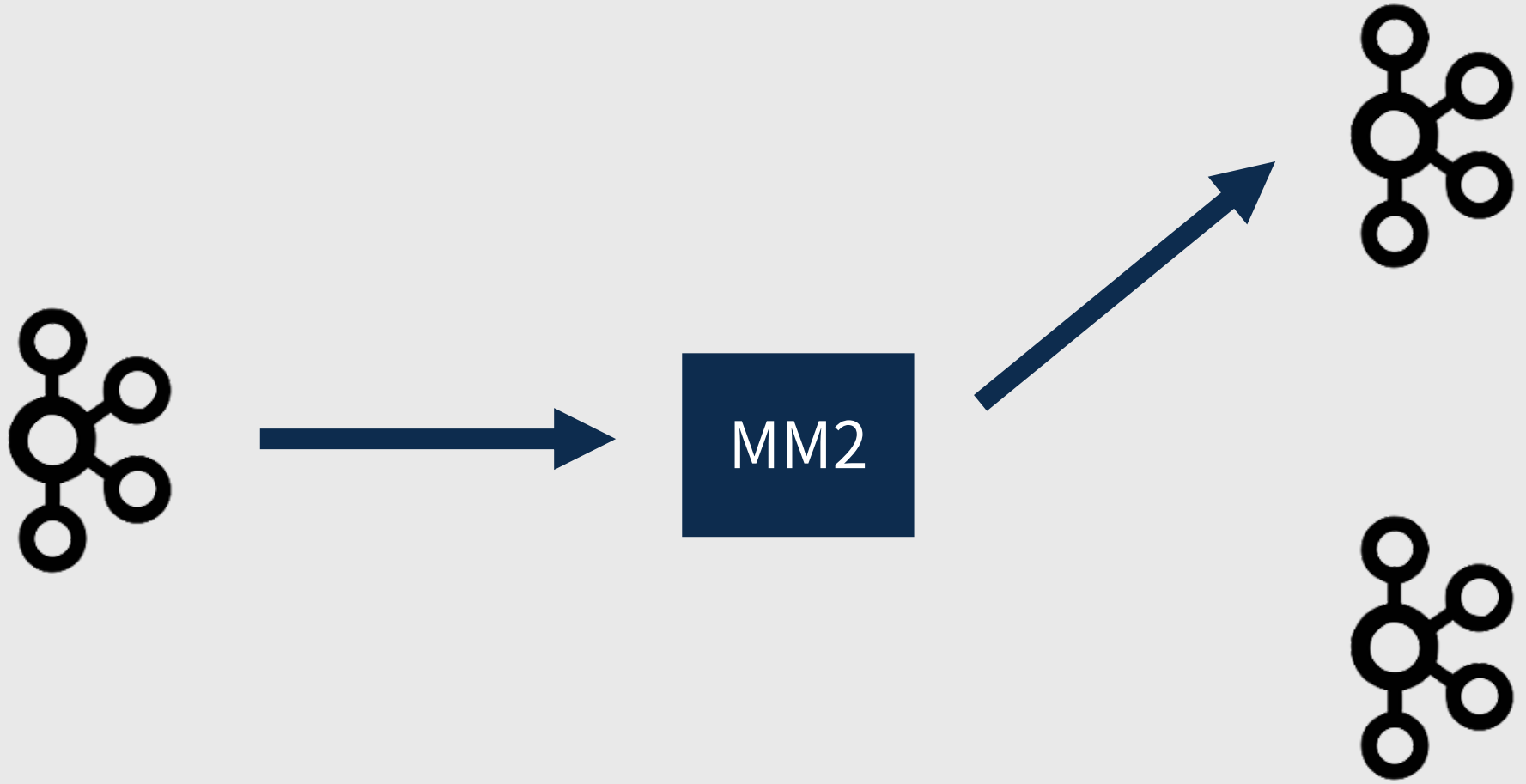
MirrorMaker2 - 모니터링

- Throughput/latency per partition
  - kafka.connect.mirror:type=MirrorSourceConnector
    - record-count
    - byte-rate
    - record-age-ms
    - replication-latency-ms
- offset checkpoint latency
  - kafka.connect.mirror:type=MirrorCheckpointConnector
    - checkpoint-latency-ms

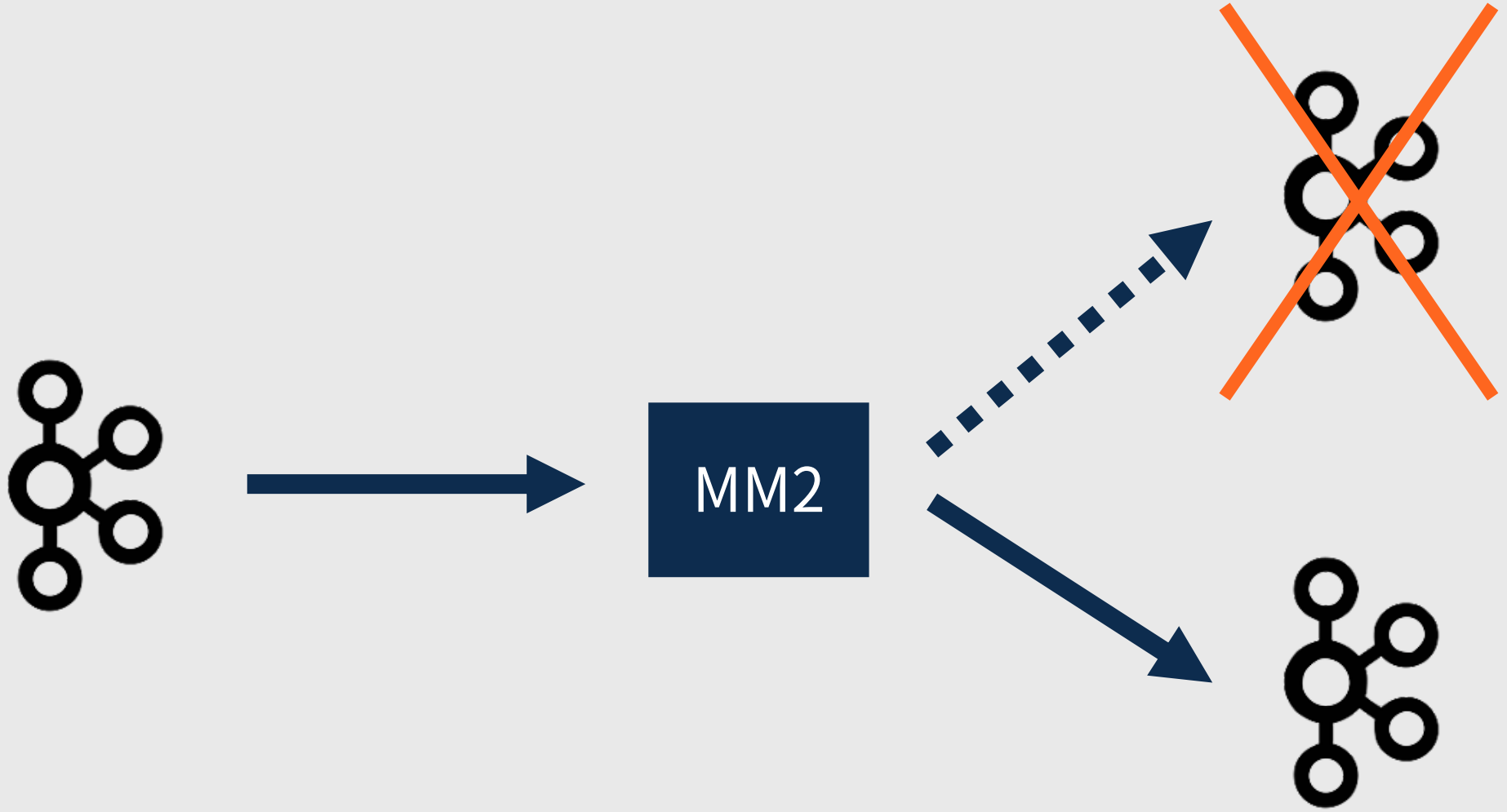
MirrorMaker2 - prometheus & grafana

MirrorMaker2 운영 팁

MirrorMaker2 - Migration



MirrorMaker2 - Migration



MirrorMaker2 - Migration

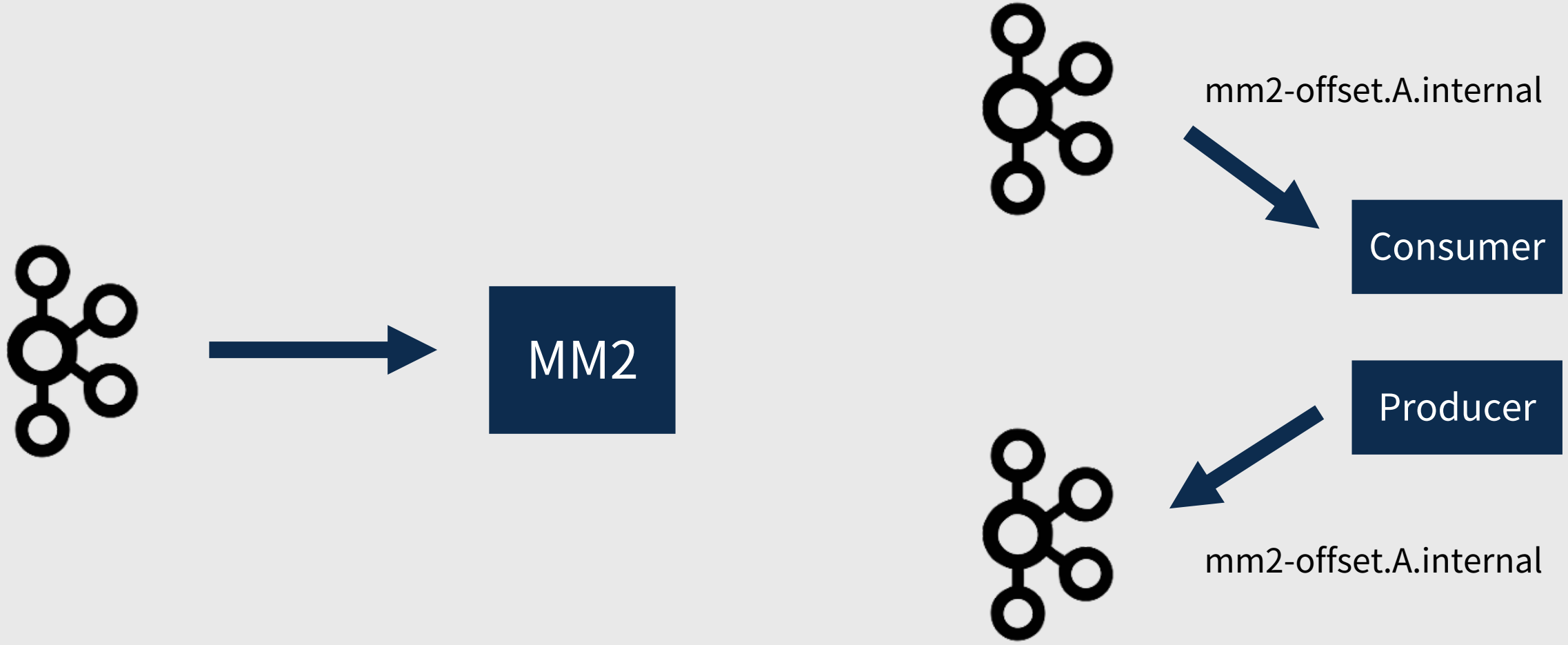
- source connector는 처리한 offset을 저장한다.
  - offset.storage.topic : MirrorMaker2의 설정 (mm2-offset.A.internal)
- MirrorSourceConnector는 재시작할 때 mm2-offset.A.internal 이후 오프셋부터 복제

```
./bin/kafka-console-consumer.sh –bootstrap-server B:9092 \
 --topic mm2-offset.A.internal \
 --property print.key=true --from-beginning
```

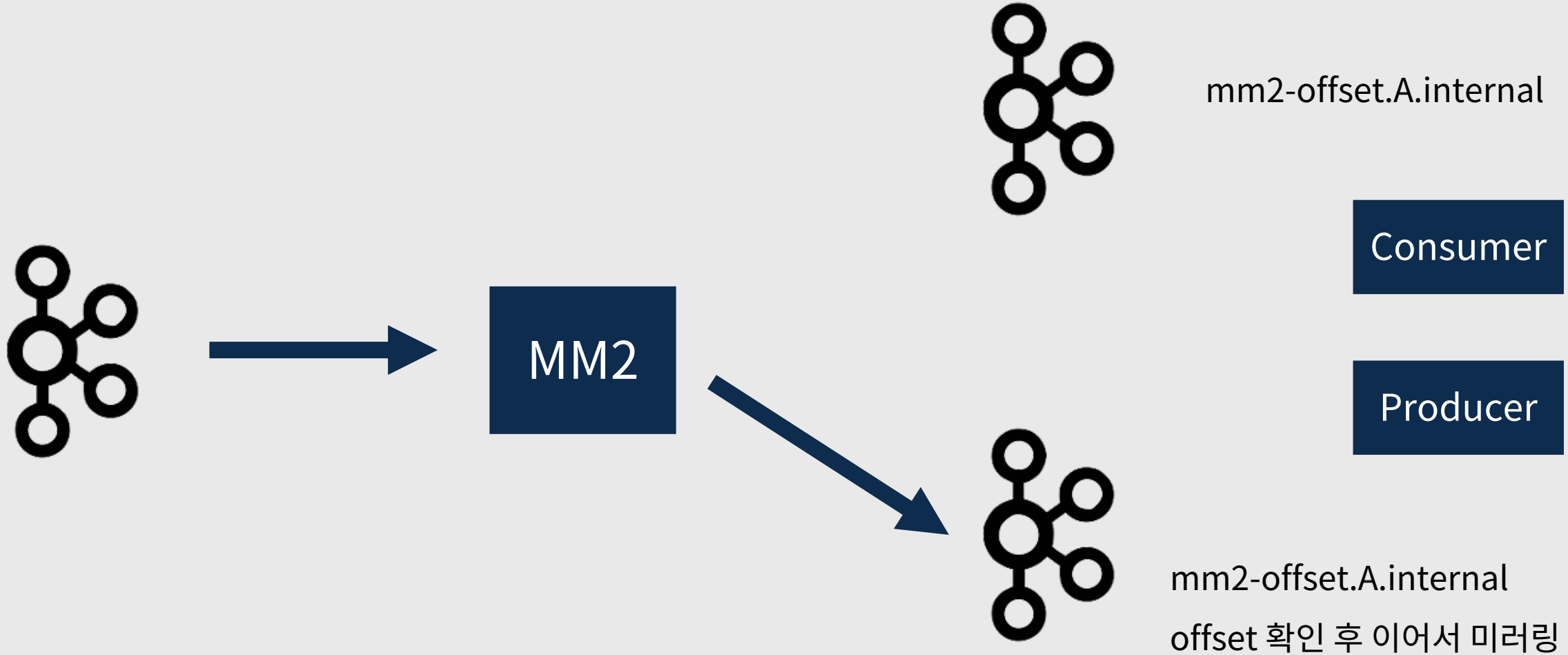
header : ["MirrorSourceConnector",{"cluster":"A","partition":0,"topic":"test-topic"}]

body : {"offset":225}

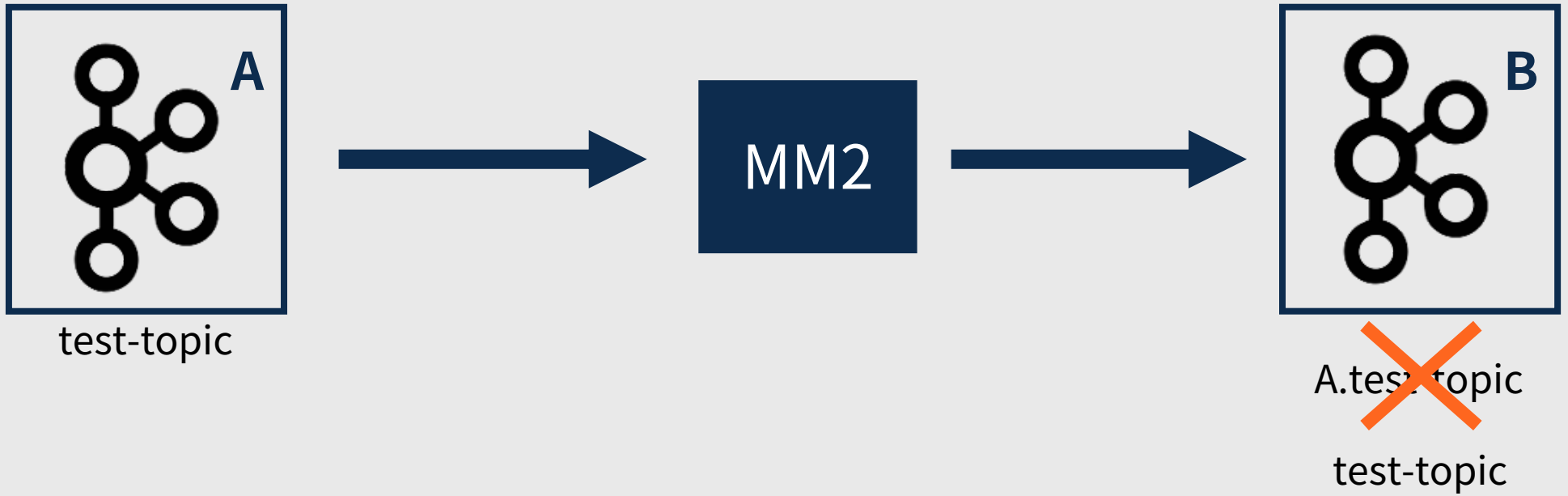
MirrorMaker2 - Migration



MirrorMaker2 - Migration



MirrorMaker2 - Prefix 제거



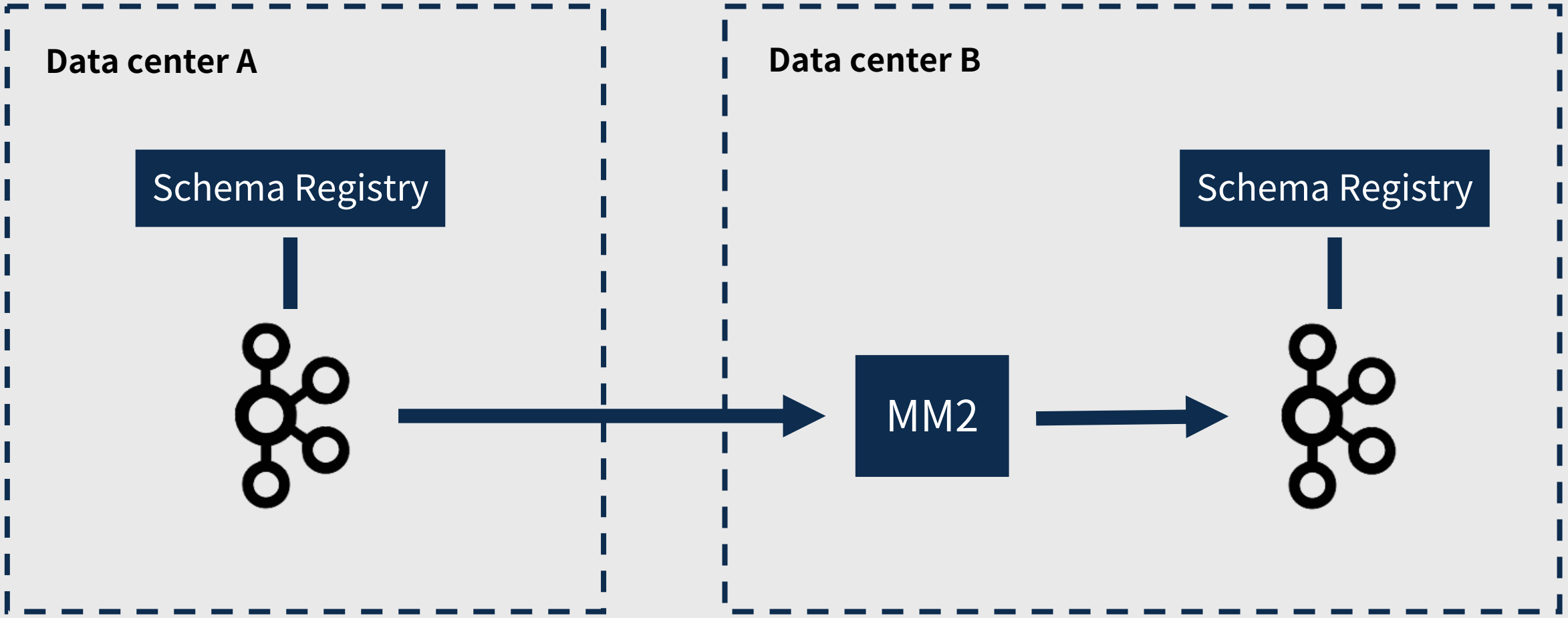
MirrorMaker2 - Prefix 제거

- DefaultReplicationPolicy
  - prefix를 붙여서 토픽 복제하는 클래스
- IdentityReplicationPolicy (KAFKA-9726)
  - prefix 없이 동일 이름으로 토픽 복제

```
replication.policy.class = org.apache.kafka.connect.mirror.IdentityReplicationPolicy
```

- replication.policy.class
  - DefaultReplicationPolicy 클래스 상속 및 ReplicationPolicy, Configurable 구현
  - 다른 복제 정책을 구현하여 적용 가능

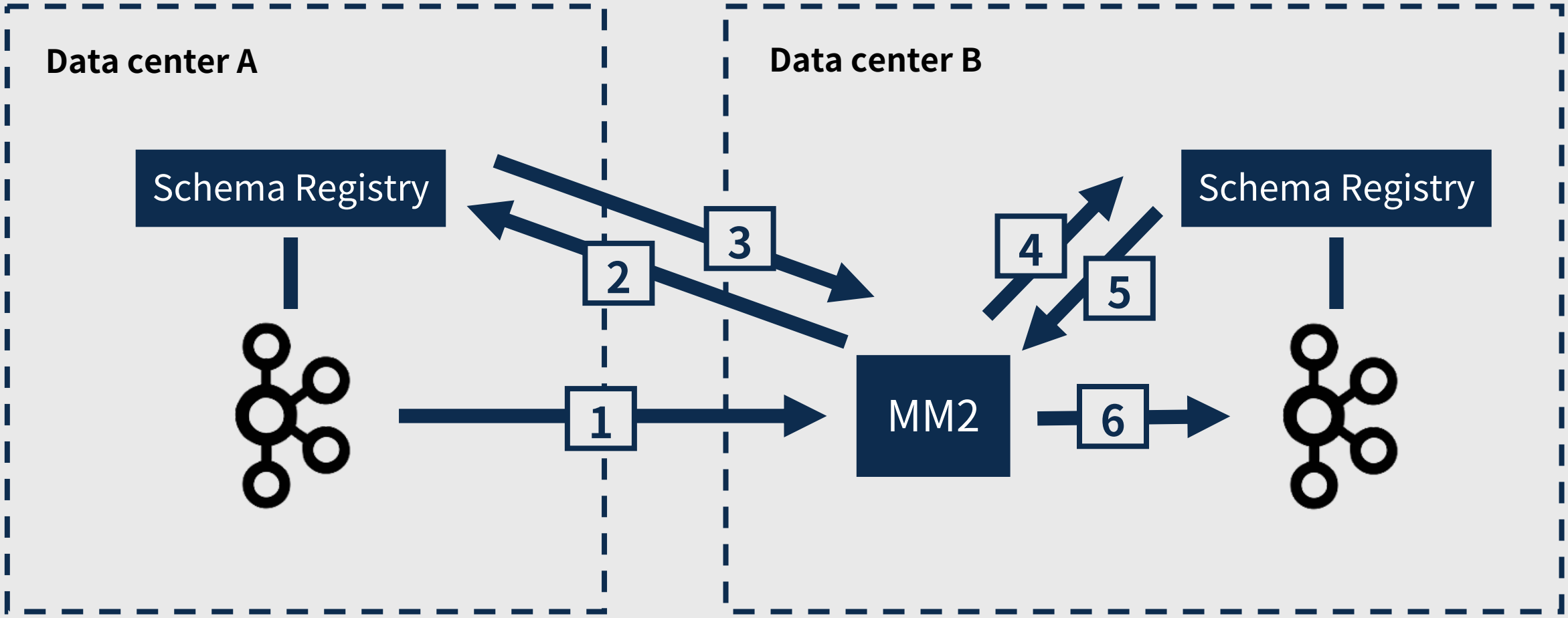
MirrorMaker2 - schema registry



MirrorMaker2 - schema registry

- _schemas 복제가 아닌 schema 신규 등록
- Single Message Transforms (SMT)
  - SMT로 카프카에 데이터 전송 전 변환 가능 (source connector)
- Schema Registry Transfer SMT
  1. source 카프카 레코드의 schema ID 확인
  2. source schema registry에서 schema 확인
  3. target schema registry에 schema 등록
  4. target 카프카에 레코드 전송

MirrorMaker2 - schema registry



감사합니다