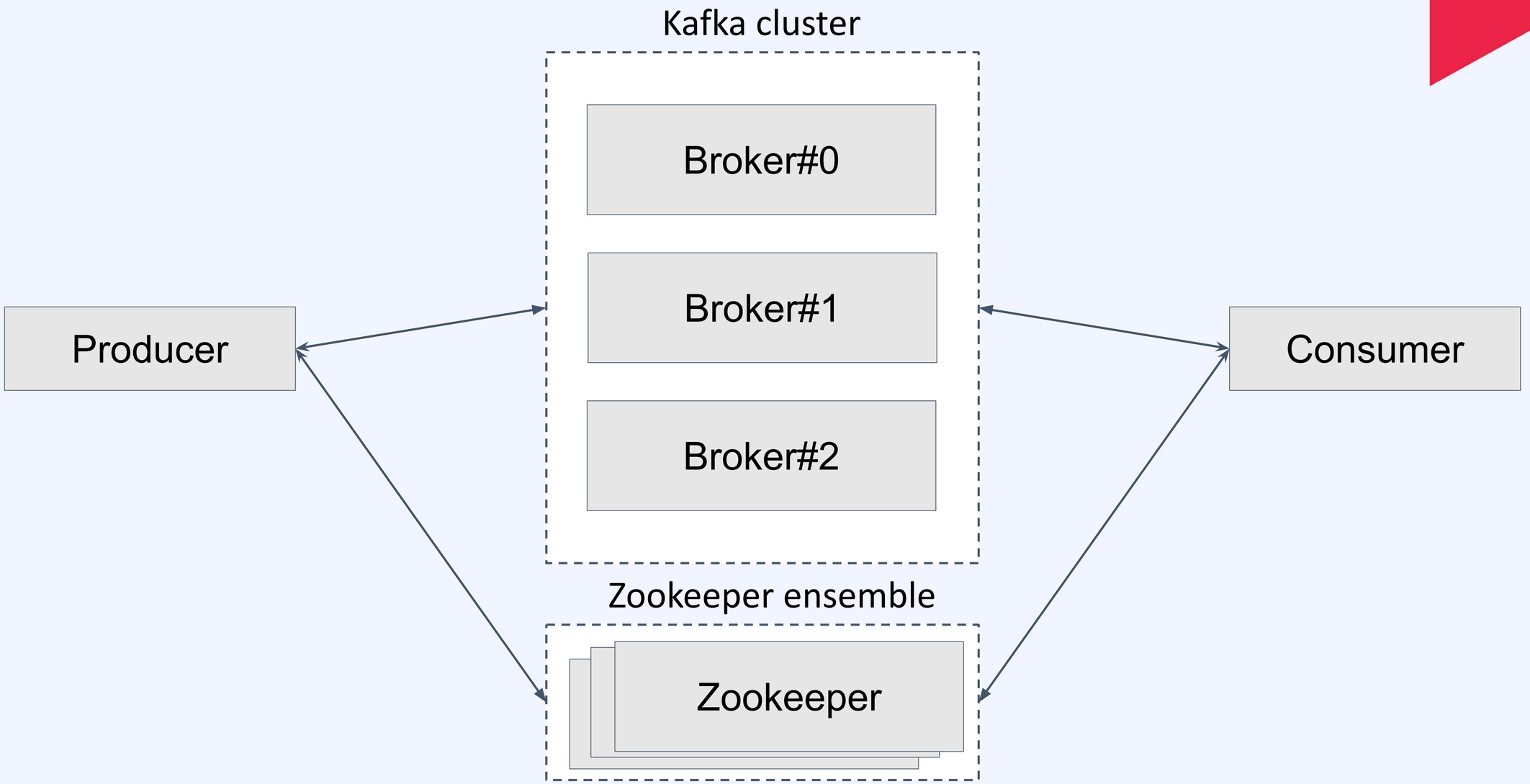
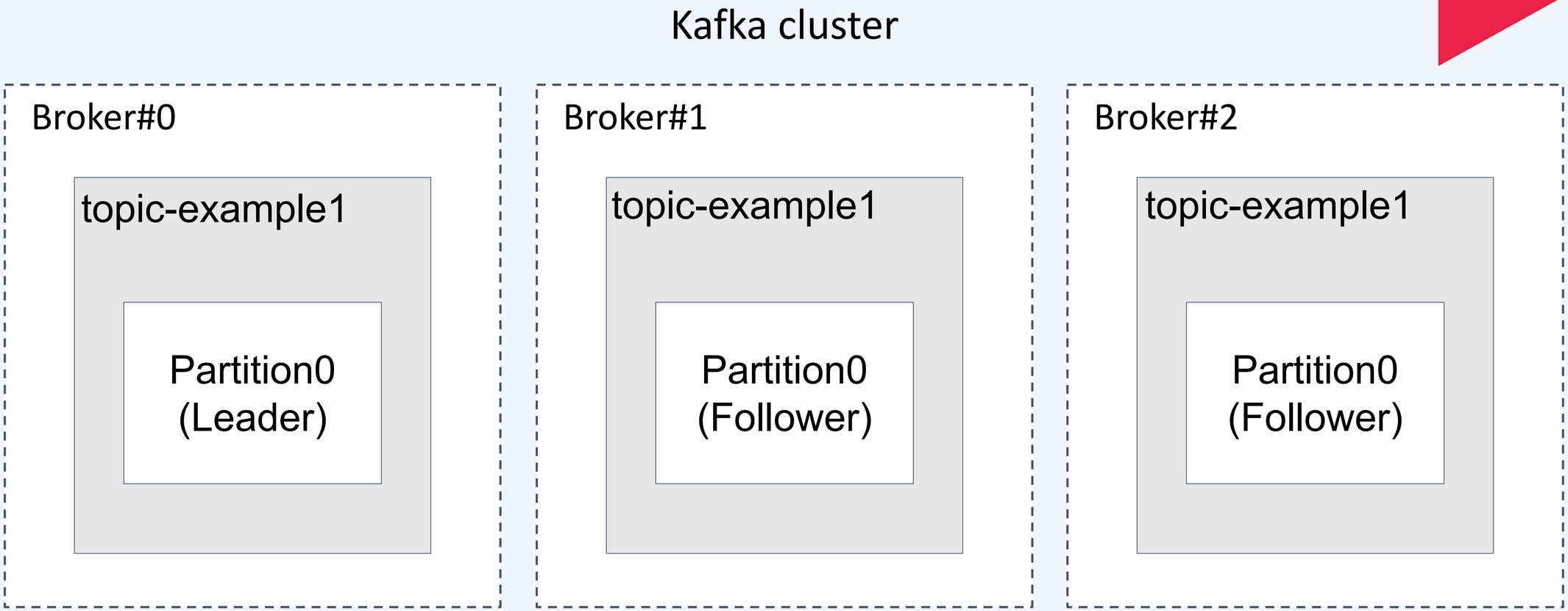
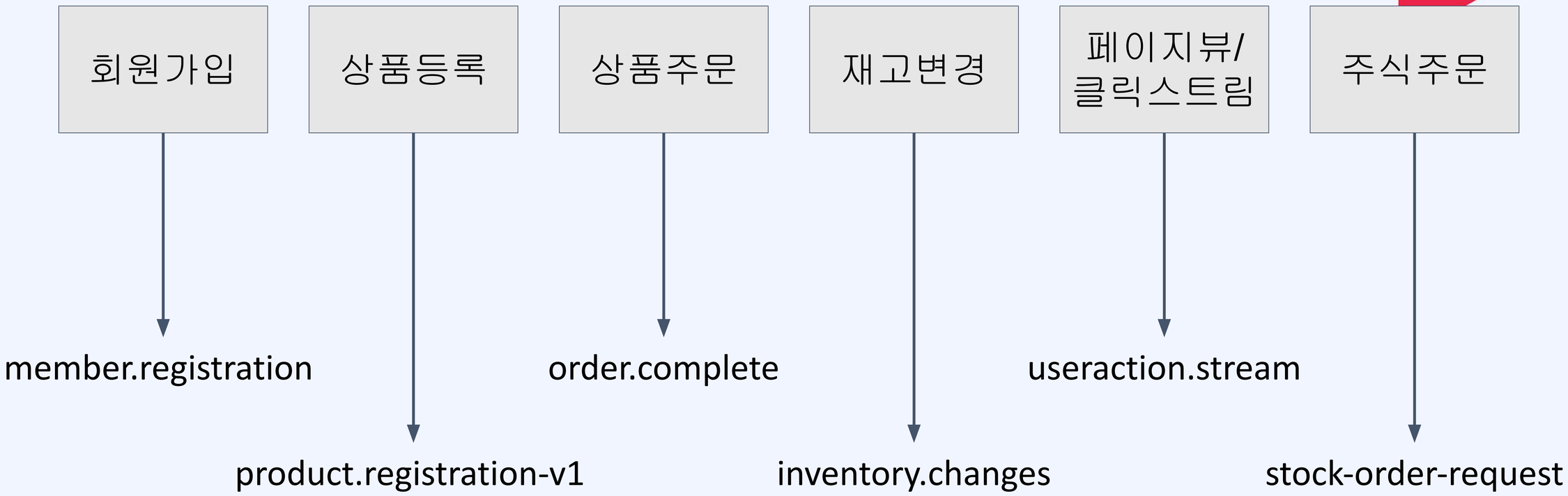
Kafka Cluster 및 상세 개념 이해

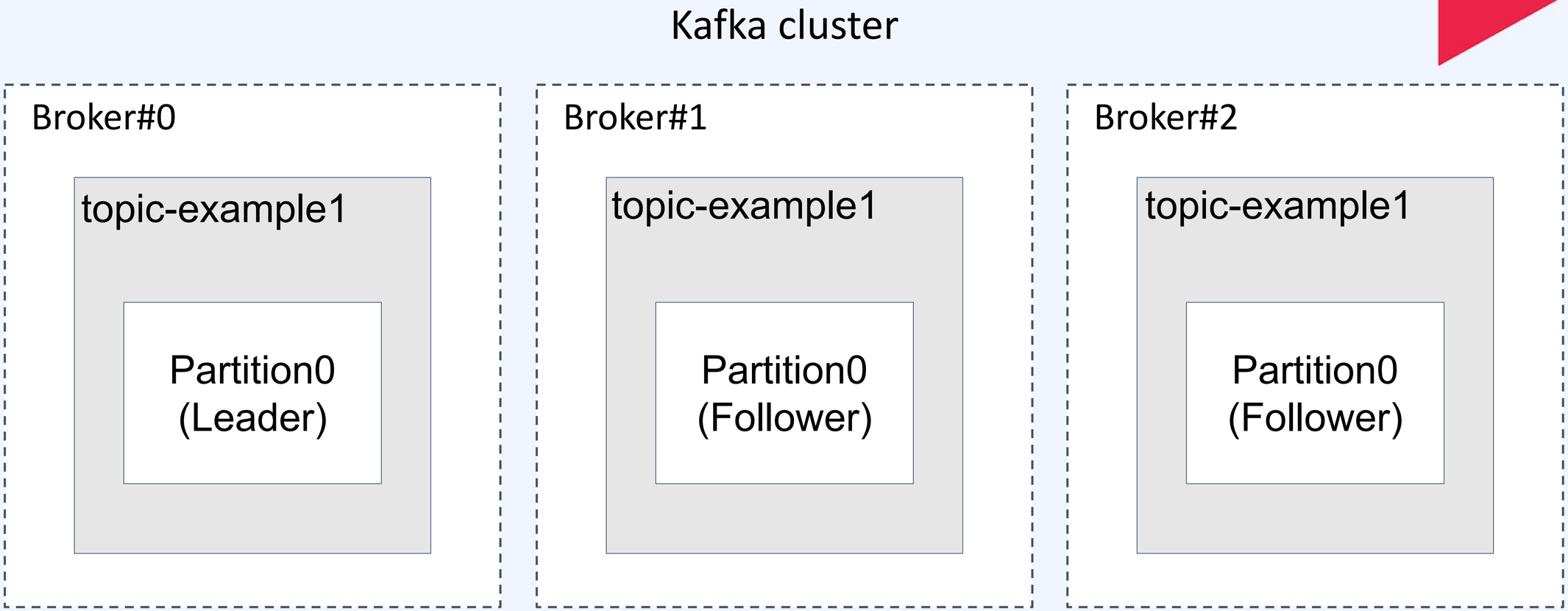
1 Kafka Broker/Topic 및 서버구성 상세설명

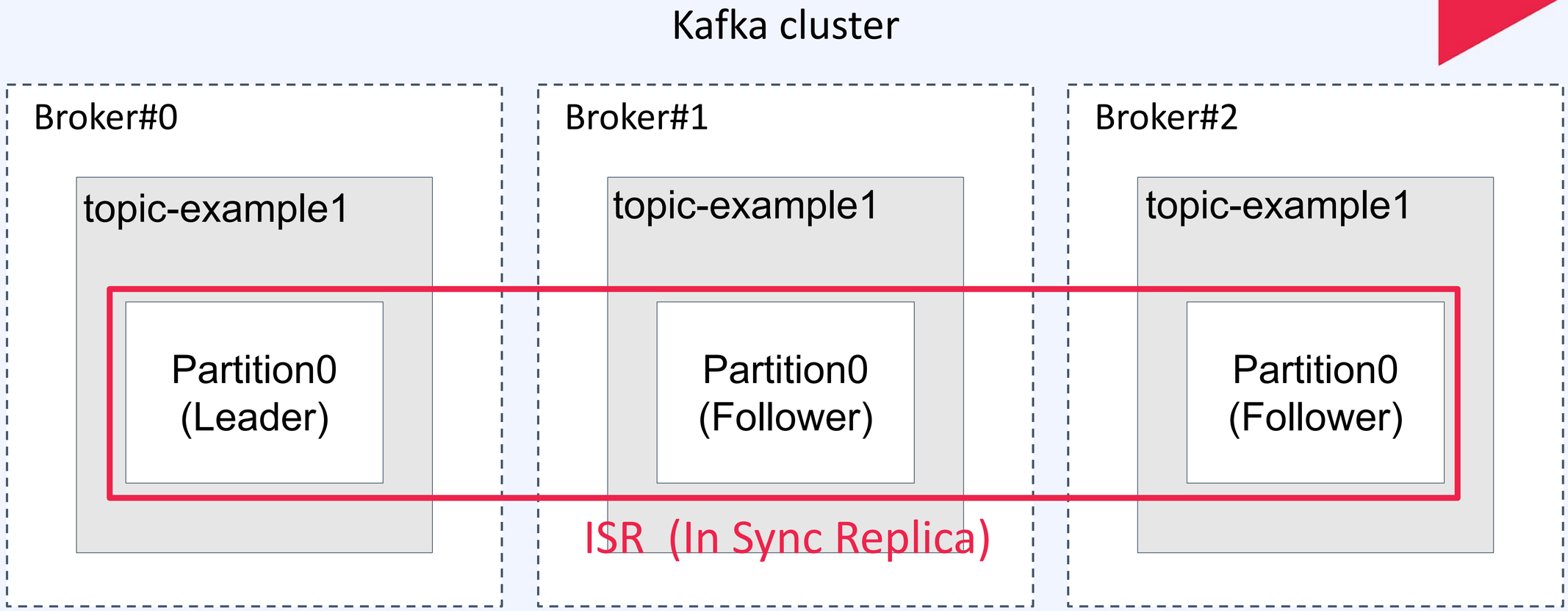


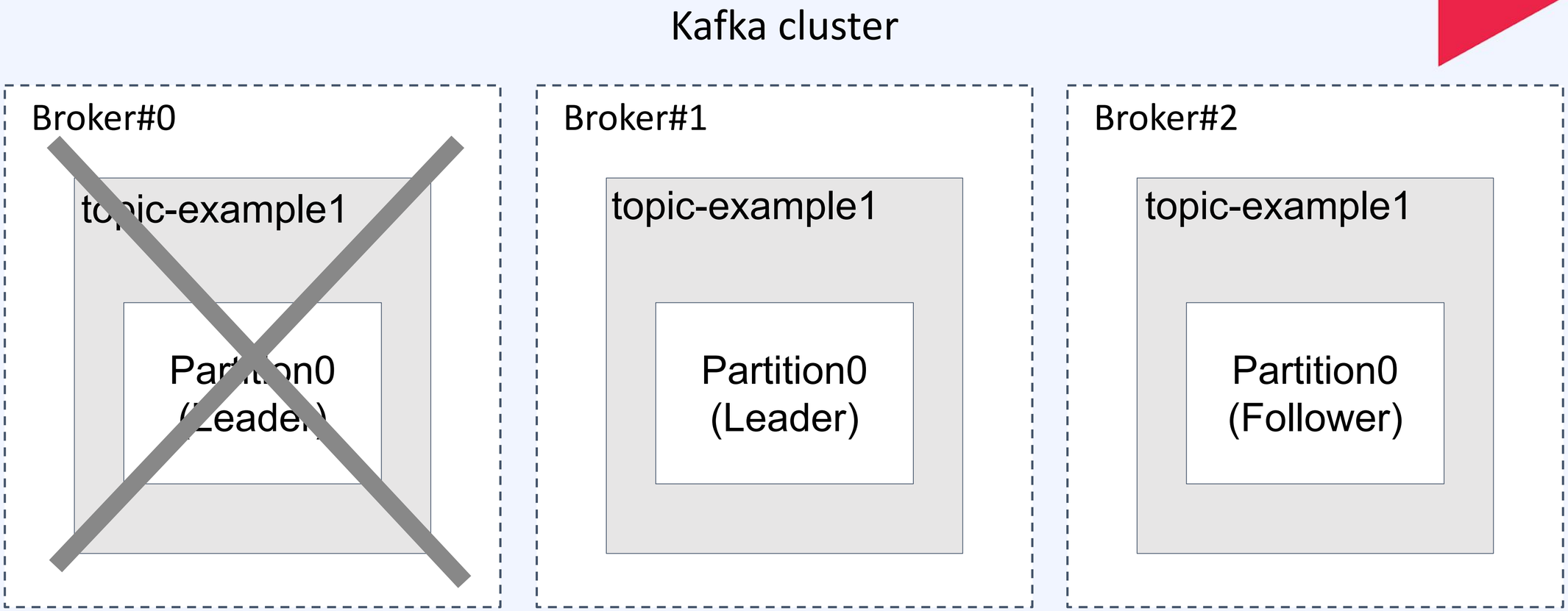




토픽이름 제약사항
- 249자 미만으로 생성.
- 영어대소문자, 0~9숫자, 마침표, 언더바, 하이픈 조합으로 생성가능 … 등

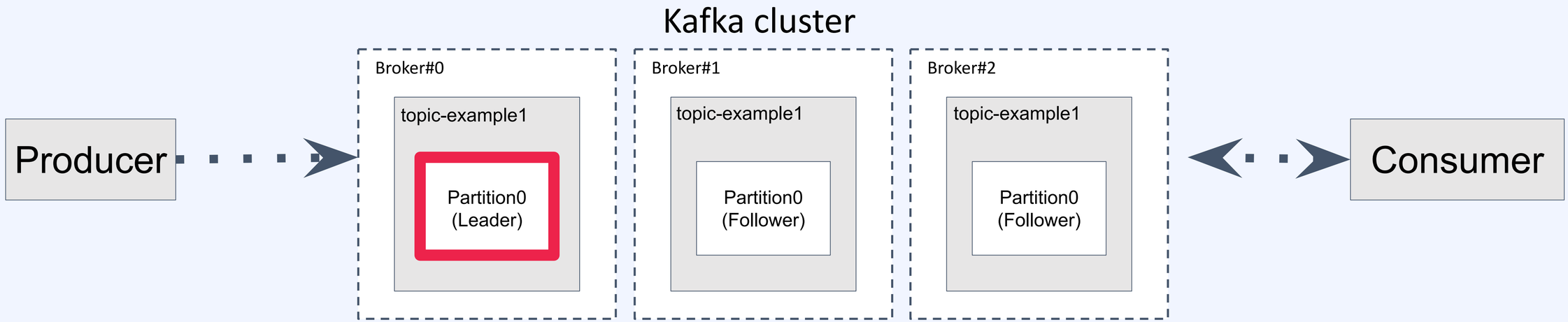


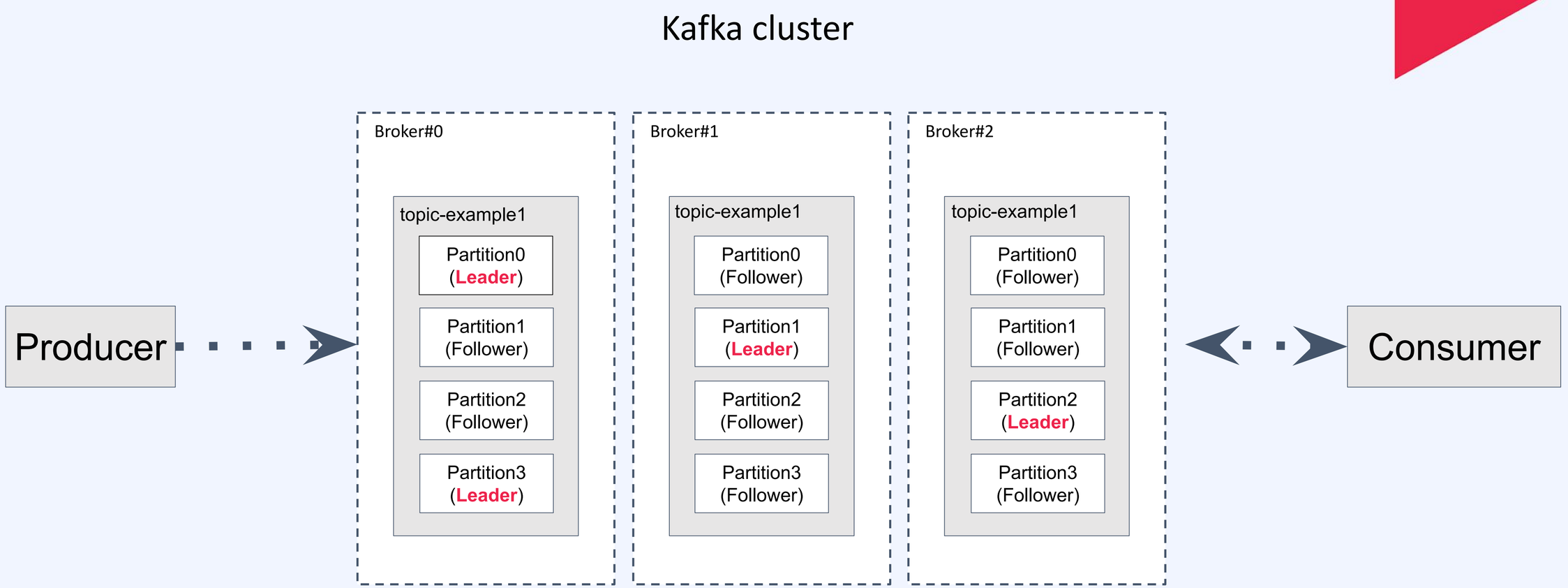


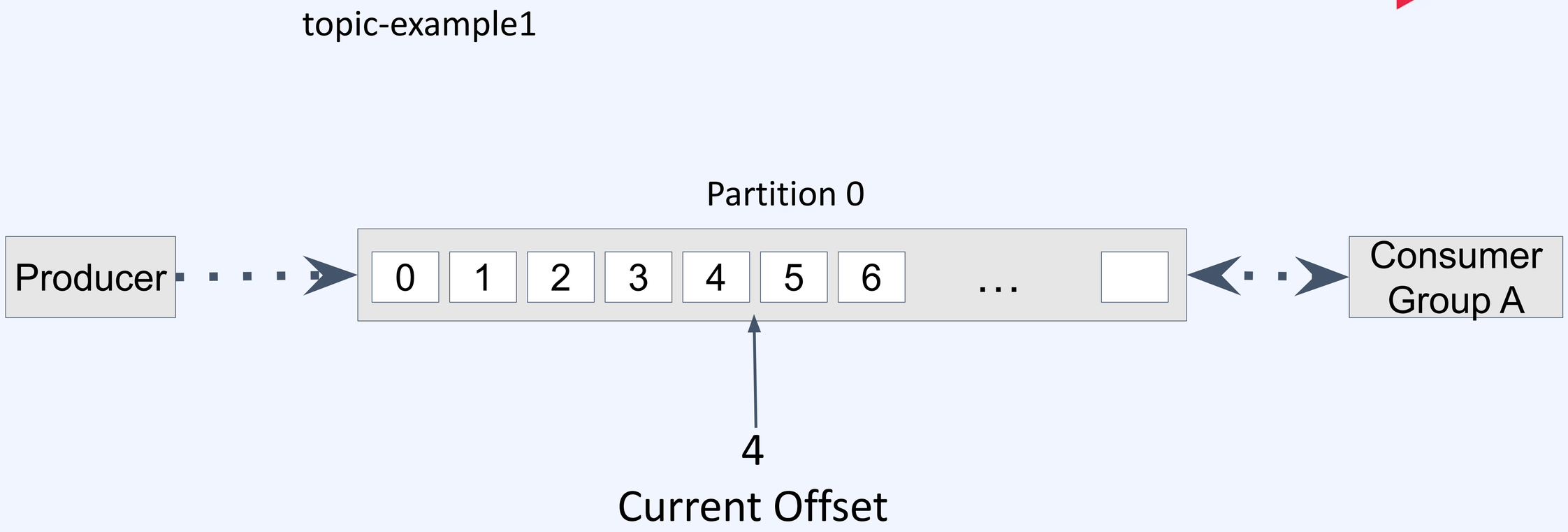


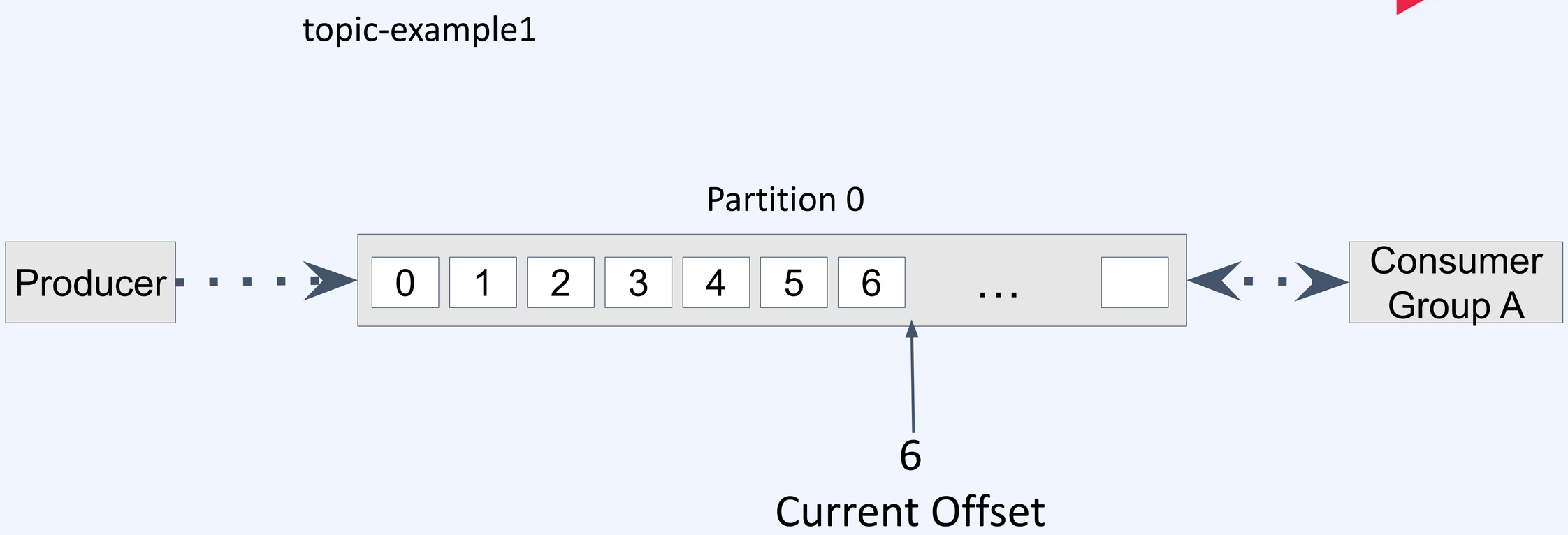
Kafka Cluster 및 상세 개념 이해

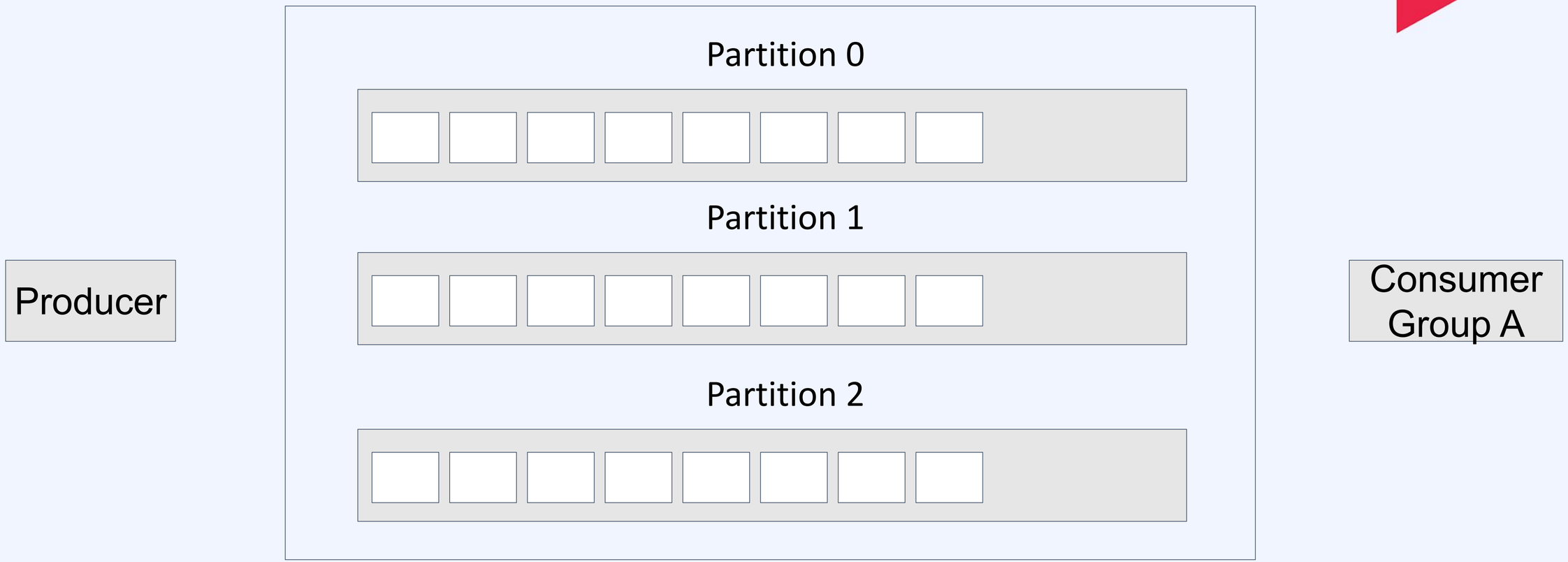
2 Kafka Partition 에 대해 이해하기





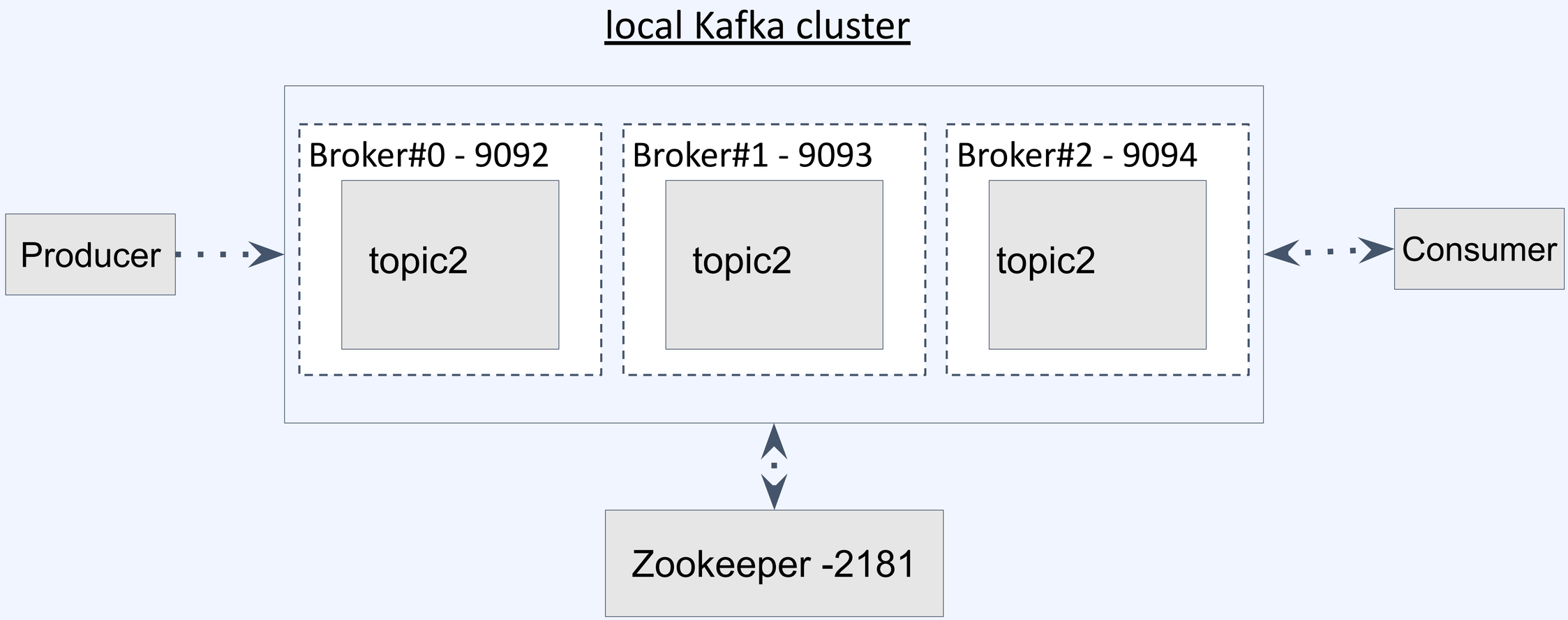






Kafka Cluster 및 상세 개념 이해

3 local 에 Kafka Cluster 구성하기(Mac)



Broker#0 - server.properties

```
broker.id=0
listeners=PLAINTEXT://localhost:9092
log.dirs=/tmp/kafka-logs
```

Broker#1 - server1.properties

```
broker.id=1
listeners=PLAINTEXT://localhost:9093
log.dirs=/tmp/kafka-logs1
```

Broker#2 - server2.properties

```
broker.id=2
listeners=PLAINTEXT://localhost:9094
log.dirs=/tmp/kafka-logs2
```

```
% bin/zookeeper-server-start.sh config/zookeeper.properties

새로운 터미널에서
% bin/kafka-server-start.sh config/server.properties &
% bin/kafka-server-start.sh config/server1.properties &
% bin/kafka-server-start.sh config/server2.properties &

새로운 터미널에서 Create a Topic
% bin/kafka-topics.sh --create --topic topic2 --bootstrap-server
localhost:9093 --partitions 3 --replication-factor 2

% bin/kafka-topics.sh --describe --topic topic2 --bootstrap-server
localhost:9092
```

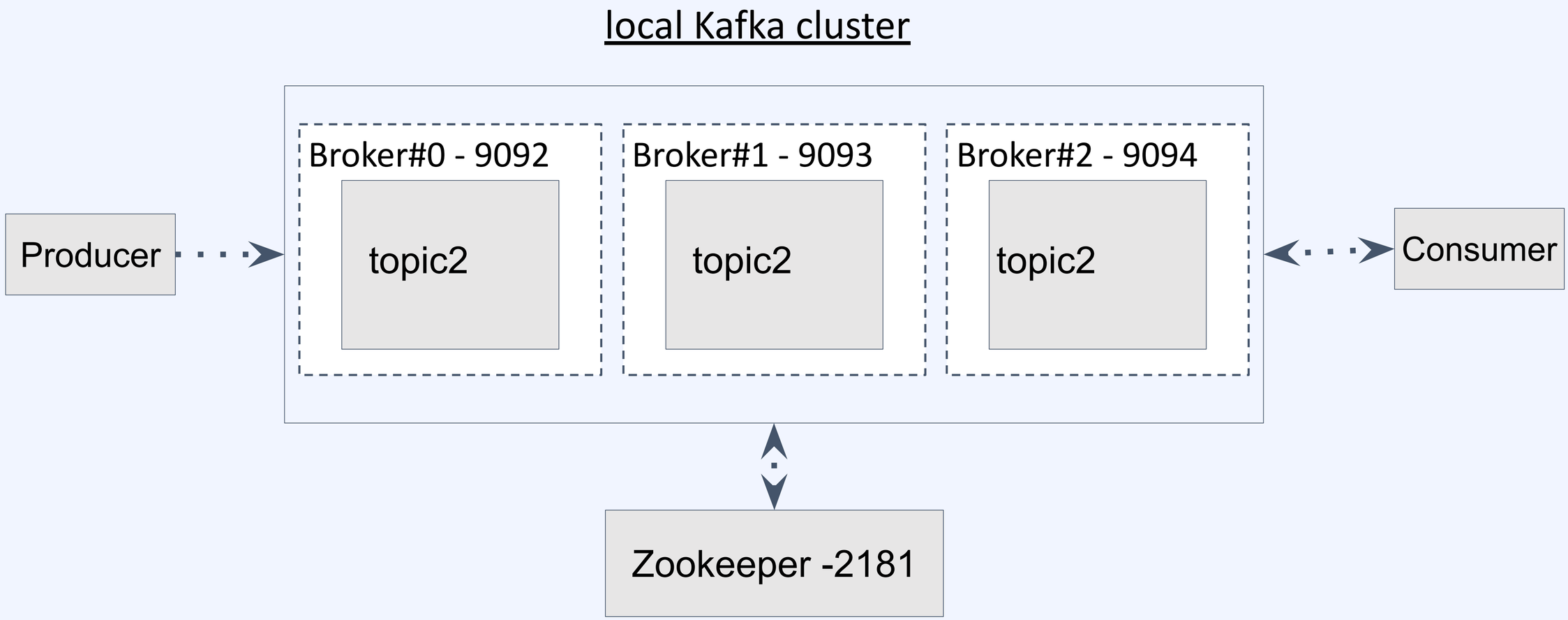
```
첫번째 Message 발행
% bin/kafka-console-producer.sh --topic topic2 --bootstrap-server
localhost:9092
>First Message
>Second Message

새로운 터미널에서 Consumer 실행하여 Message 읽기
% bin/kafka-console-consumer.sh --topic topic2 --bootstrap-server
localhost:9094 --from-beginning
```

Broker#2 을 Shutdown 시키고 서비스 상태 확인하기

```
#group 을 지정하여 Consumer 실행하여 Message 읽기
% bin/kafka-console-consumer.sh --topic topic2 --bootstrap-server
localhost:9092,localhost:9093,localhost:9094 --from-beginning --group
one

# 새로운 터미널에서 Broker#2 을 Kill 시킨다.
% kill -9 {PID}

# topic 의 상태를 확인한다.
% bin/kafka-topics.sh --describe --topic topic2 --bootstrap-server
localhost:9092

# 다시 producer 에서 메시지를 발생하여 Consumer 의 수신상황을 체크한다.
# Broker#1 도 추가로 Kill 된다면 어떻게 될까요?
```

Kafka Cluster 및 상세 개념 이해

4 local 에 Kafka Cluster 구성하기(Windows)



Broker#0 - server.properties

```
broker.id=0
listeners=PLAINTEXT://localhost:9092
log.dirs=/tmp/kafka-logs
```

Broker#1 - server1.properties

```
broker.id=1
listeners=PLAINTEXT://localhost:9093
log.dirs=/tmp/kafka-logs1
```

Broker#2 - server2.properties

```
broker.id=2
listeners=PLAINTEXT://localhost:9094
log.dirs=/tmp/kafka-logs2
```

```
> bin/windows/zookeeper-server-start.bat config/zookeeper.properties

새로운 터미널에서
> bin/windows/kafka-server-start.bat config/server.properties &
> bin/windows/kafka-server-start.bat config/server1.properties &
> bin/windows/kafka-server-start.bat config/server2.properties &

새로운 터미널에서 Create a Topic
> bin/windows/kafka-topics.bat --create --topic topic2
--bootstrap-server localhost:9093 --partitions 3 --replication-factor
2

> bin/kafka-topics.bat --describe --topic topic2 --bootstrap-server
localhost:9092
```

```
첫번째 Message 발행
> bin/windows/kafka-console-producer.bat --topic topic2
--bootstrap-server localhost:9092
>First Message
>Second Message

새로운 터미널에서 Consumer 실행하여 Message 읽기
> bin/windows/kafka-console-consumer.bat --topic topic2
--bootstrap-server localhost:9094 --from-beginning
```

Broker#2 을 Shutdown 시키고 서비스 상태 확인하기

```
#group 을 지정하여 Consumer 실행하여 Message 읽기
> bin/windows/kafka-console-consumer.bat --topic topic2
--bootstrap-server localhost:9092,localhost:9093,localhost:9094
--from-beginning --group one

# 새로운 터미널에서 Broker#2 을 종료시킨다.(Ctrl+C)

# topic 의 상태를 확인한다.
> bin/windows/kafka-topics.bat --describe --topic topic2
--bootstrap-server localhost:9092

# 다시 producer 에서 메시지를 발생하여 Consumer 의 수신상황을 체크한다.
# Broker#1 도 추가로 Kill 된다면 어떻게 될까요?
```

Kafka Cluster 및 상세 개념 이해

5 Kafka CLI 의 다양한 기능이해

topics

```
kafka-topics.sh --create --topic topic3 --partitions 1
kafka-topics.sh --describe --topic topic3
kafka-topics.sh --alter --topic topic3 --partitions 3
kafka-configs.sh --alter --entity-type topics --entity-name topic3 --add-config
retention.ms=86400000
kafka-topics.sh --list
```

producer

```
kafka-console-producer.sh --topic topic3 --request-required-acks 1
kafka-console-producer.sh --topic topic3 --message-send-max-retries 50
kafka-verifiable-producer.sh --topic topic3 --max-messages 100
```

consumer

```
kafka-console-consumer.sh --topic topic3 --from-beginning
kafka-console-consumer.sh --topic topic3 --from-beginning --group group1
kafka-console-consumer.sh --topic topic3 --from-beginning --group group2 --property
print.key=true --property key.separator="-"
```

consumer-groups

```
kafka-consumer-groups.sh --list
kafka-consumer-groups.sh --describe --group group1
```

단일모드 connect 설정파일 - config/connect-standalone.properties

```
key.converter.schemas.enable=false
value.converter.schemas.enable=false
```

File Sink connect 설정파일 - config/connect-file-sink.properties

```
topics=topic3
```

단일모드 connect 실행

```
bin/connect-standalone.sh config/connect-standalone.properties
config/connect-file-sink.properties
```

Kafka Cluster 및 상세 개념 이해

6 AWS Managed Kafka Service 소개

AWS 의 관리형 서비스

- RDS : Amazon Relational Database Service – Aurora , MySql, PostgreSQL, Oracle, MS Sql Server ..
- ElasticCache : Redis, Memcached
- Keyspaces : Apache Cassandra
- OpenSearch : ElasticSearch
- MSK : Apache Kafka
- …

AWS 의 관리형 서비스 사용의 장점

- 운영, 관리 리소스 비용을 획기적으로 낮출 수 있음.
- 학습 비용을 획기적으로 낮출 수 있음.
- 트래픽 당 과금 등을 활용해서 초기 구축비용을 절감하고 효율화할 여지가 있음.

AWS 의 관리형 서비스 사용의 단점

- EC2 에 직접 해당 Application 을 설치하는 것보다는 많은 비용이 발생함.(일반적으로 동일 Instance Type 대비 50%~ 이상 비쌈)
- EC2 에 직접 해당 Application 을 설치한 것과 기능과 설정에 제약이 있는 경우가 있음.

https://docs.aws.amazon.com/ko_kr/msk/index.html
https://aws.amazon.com/ko/msk/pricing/

Amazon Managed Streaming for Apache Kafka(MSK)

완전관리형의 고가용성 Apache Kafka 서비스를 통해 안전하게 데이터를 스트리밍

AWS 계정 생성 | 시작하기

고가용성 Apache Kafka 및 Kafka Connect 클러스터의 프로비저닝, 구성 및 유지 관리 등의 운영 오버헤드를 제거합니다.

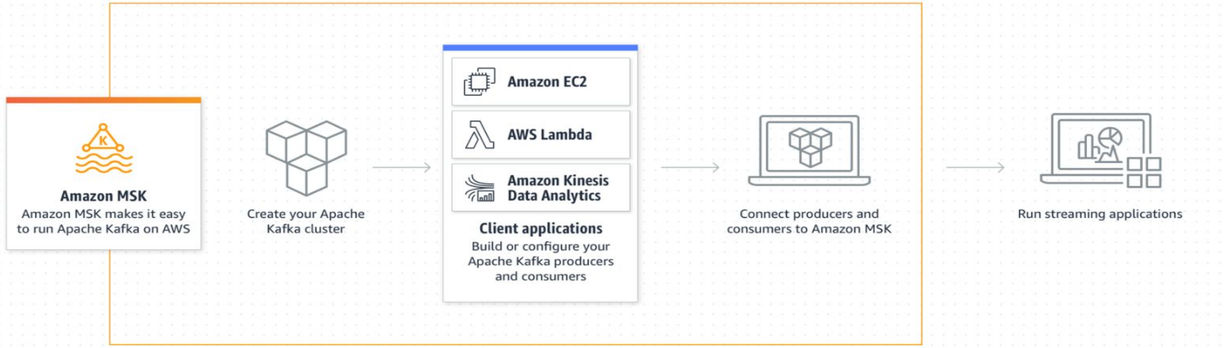
Apache Kafka용으로 구축된 애플리케이션 및 도구를 코드 변경 없이 바로 사용하고 클러스터 용량을 자동으로 확장합니다.

네이티브 AWS 통합을 사용하여 안전하게 규정을 준수하며 프로덕션용 애플리케이션을 쉽게 배포합니다.

Amazon MSK로 비용을 낮게 유지합니다. 종량제 가격을 통해 다른 공급자들에 비해 최저 1/13에 해당하는 저렴한 비용으로 서비스를 제공합니다.

작동 방식

Amazon MSK는 완전관리형 Apache Kafka를 통해 실시간으로 스트리밍 데이터를 손쉽게 수집하고 처리하게 해줍니다.



MSK 에 CLI 를 실행할 EC2 생성
- Amazon Linux. t2.small instance type

MSK 생성(사용자 지정 생성)
- Amazon EC2 의 보안그룹항목에서 위에서 생성한 EC2 의 보안그룹을 선택하여 추가해야함.
- 액세스 제어 방법 : 인증되지 않은 액세스 체크
- 클라이언트와 브로커 간 : 일반테스트 체크

```
$ sudo yum install -y java-1.8.0-openjdk-devel.x86_64

$ wget https://archive.apache.org/dist/kafka/2.8.2/kafka_2.13-2.8.2.tgz

$ tar -zxf kafka_2.13-2.8.2.tgz

$ cd kafka_2.13-2.8.2

$ bin/kafka-topics.sh --create --topic topic4 --bootstrap-server ${MSK브로커endpoint}

첫번째 Message 발행
$ bin/kafka-console-producer.sh --topic topic4 --bootstrap-server ${MSK브로커endpoint}
>First Message
>Second Message

새로운 터미널에서 Consumer 실행하여 Message 읽기
% bin/kafka-console-consumer.sh --topic topic4 --from-beginning --bootstrap-server
${MSK브로커endpoint}
```