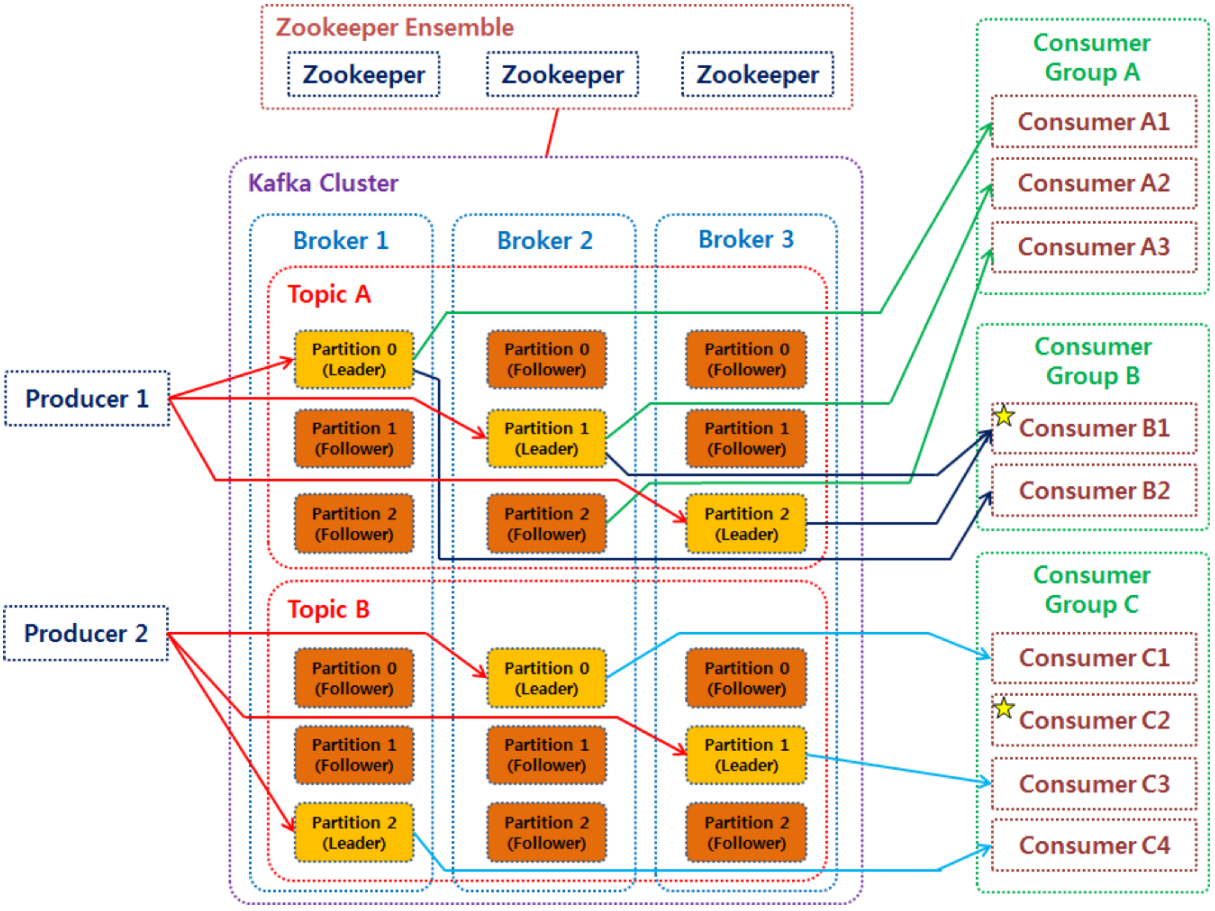
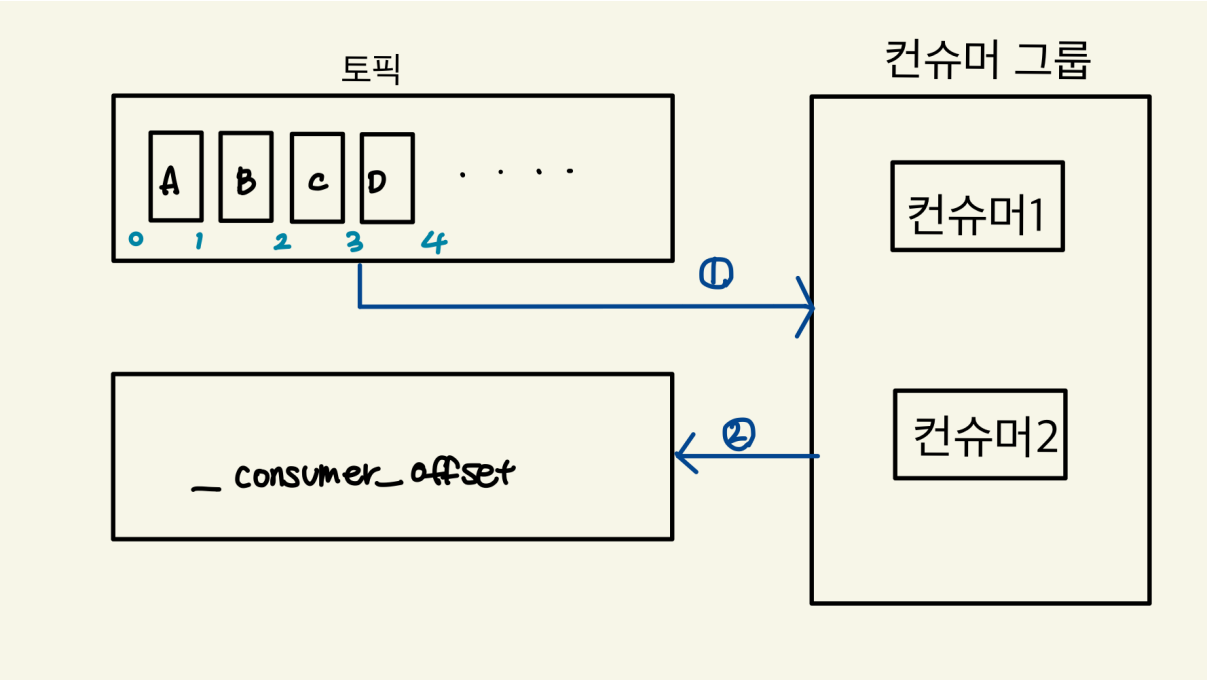
06장 컨슈머의 내부 동작 원리와 구현



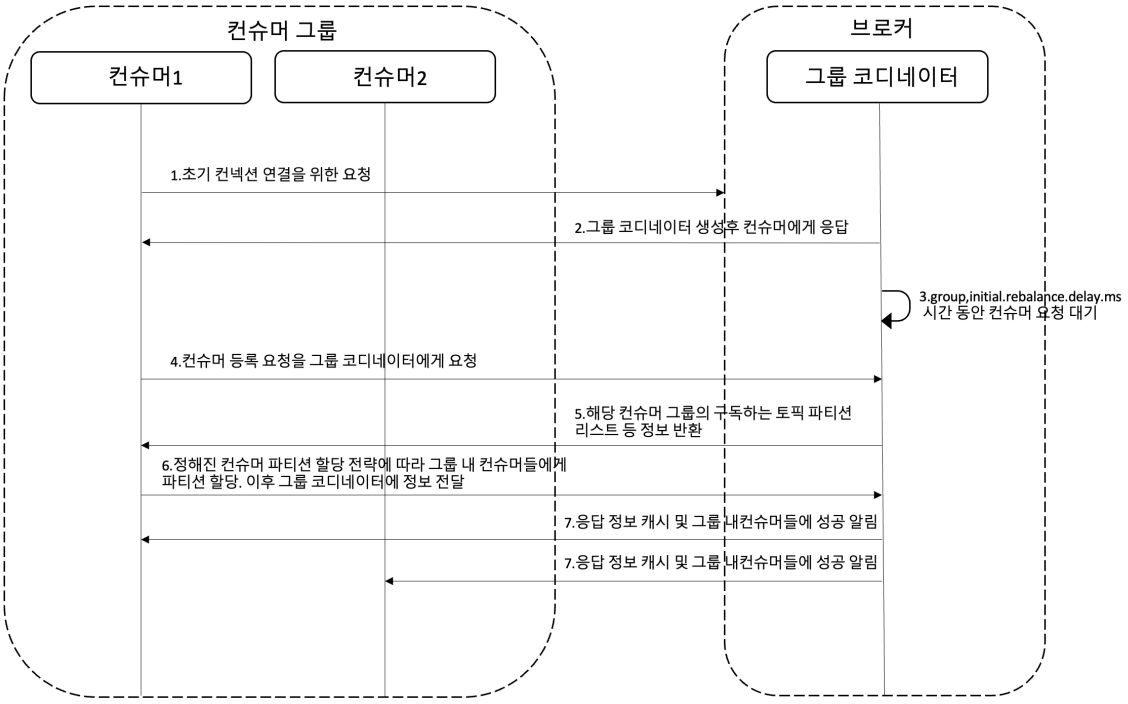
6.1 컨슈머 오프셋 관리

- 컨슈머는 오프셋으로 어디까지 메시지를 가져왔는지 알 수 있습니다.
- 카프카에서는 읽어야 할 메시지의 위치를 오프셋이라고 부르고, 컨슈머 그룹은 *_cosumer_offsets 토픽(내부토픽)에* 각 컨슈머 그룹별 오프셋 위치 정보를 기록합니다.



6.2 그룹 코디네이터

- 컨슈머들은 하나의 컨슈머 그룹에 소속됩니다.
- 컨슈머 그룹은 하나의 공동체로 동작합니다.
- 안정적인 컨슈머 그룹 관리를 위해 브로커중 하나에 그룹 코디네이터가 존재합니다.
- 컨슈머 그룹 등록 과정



- 컨슈머들은 현재 자신들이 속한 컨슈머 그룹에서 빠져나갈 수도, 새롭게 합류할 수도 있습니다.
- 그룹 코디네이터와 컨슈머들은 하트비트를 주고 받으면서 컨슈머의 상태를 체크합니다.
- 컨슈머 그룹은 각 컨슈머들에게 작업을 균등하게 분배합니다. 컨슈머 또는 파티션에 변화가 생기는 경우 컨슈머 리밸랜싱이 일어납니다.
    - 컨슈머 하트비트 옵션

컨슈머 옵션	값	설명
heartbeat.interval.ms	3000	그룹 코디네이터와 하트비트 인터벌 시간을 의미함. 기본값은 3000이며 session.timeout.ms보다 낮게 설정해야한다. 평균적으로 session.timeout.ms의 1/3이 적당함
session.timeout.ms	10000	어떤 컨슈머가 특정 시간 안에 하트비트를 받지 못하면 문제가 발생했다고 판단하는 기준. 만약 하트 비트를 받지 못했다고 판단하는 경우 컨슈머 그룹에서 해당 컨슈머는 제거되고 리밸런싱 동작이 일어남. 기본값은 10000
max.poll.interval.ms	300000	컨슈머 poll()을 활용해 컨슈머가 문제가 있는지를 판단하는 기준. 컨슈머는 주기적으로 poll()을 호출하여 토픽으로부터 레코드를 가져오는데 poll() 호출 후 최대 5분간 응답값이 없으면 리밸런싱 동작이 일어남.

6.3 스태틱 멤버십

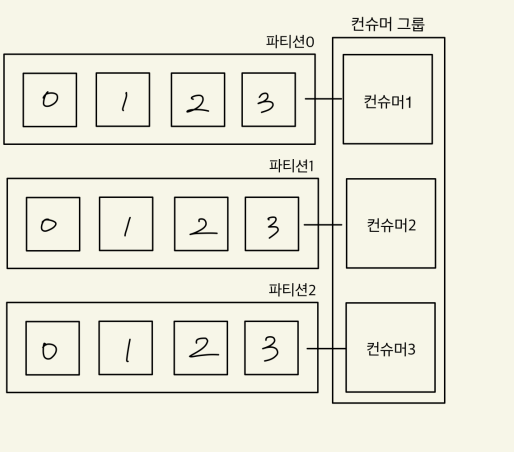
- 컨슈머 리밸랜싱 동작은 높은 비용이 지출됩니다.
- 리밸랜싱 작업동안 컨슈머는 메시지를 가져오지 않습니다.
- 컨슈머가 재시작될 때마다 리밸런싱 작업이 일어나는것은 비효율적입니다.
- 불필요한 리밸랜싱을 막기 위해 카프카는 스태틱 멤버십 도입도했습니다.(아파치 카프카 2.3 ~)
  - 컨슈머가 컨슈머 그룹에서 떠날 때 그룹 코디네이터에 알리지 않습니다. → 리밸랜싱 회피
  - 컨슈머가 다시 그룹으로 합류할때 컨슈머의 ID로 기존 구성원을 확인하고, 기존 구성원인 경우 리밸랜싱을 회피합니다.
- 옵션값 유의사항
  - 컨슈머 재시작 시간 보다 session.timeout.ms을 크게 설정해야 합니다.
  - 컨슈머 재시작전 리밸랜싱 작업이 이뤄지면 안되기 때문입니다.
- 실습

```
kafka-topics --create --topic peter-test06 --bootstrap-ser
Created topic peter-test06.

kafka-console-consumer \
    --bootstrap-server kafka1:9091 \
    --from-beginning \
    --group peter-consumer01 \
    --topic peter-test06

kafka-console-producer \
    --bootstrap-server kafka1:9091 \
    --topic peter-test06

kafka-consumer-groups --bootstrap-server kafka1:9091 --gro
```

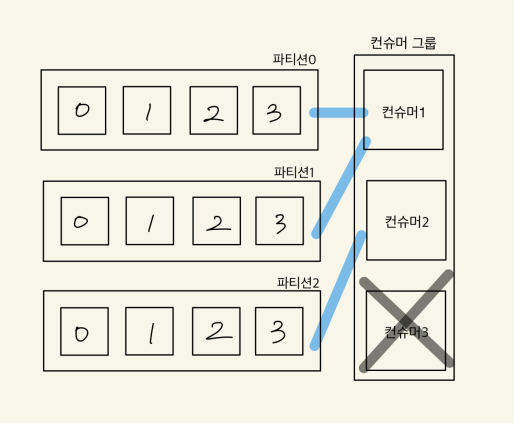
```
GROUP           TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID                                           HOST
peter-consumer01 peter-test06    0          2               2               0               consumer-peter-consumer01-1-9a5334be-6a80-4000-833c-6386b15b3385 /192.168.0.7
er-consumer01-1
peter-consumer01 peter-test06    1          2               2               0               consumer-peter-consumer01-1-9a5334be-6a80-4000-833c-6386b15b3385 /192.168.0.7
er-consumer01-1
peter-consumer01 peter-test06    2          1               1               0               consumer-peter-consumer01-1-9a5334be-6a80-4000-833c-6386b15b3385 /192.168.0.7
er-consumer01-1
```

6.4.1 일반 컨슈머 그룹의 리밸런싱

1. 컨슈머 그룹 현재 상태



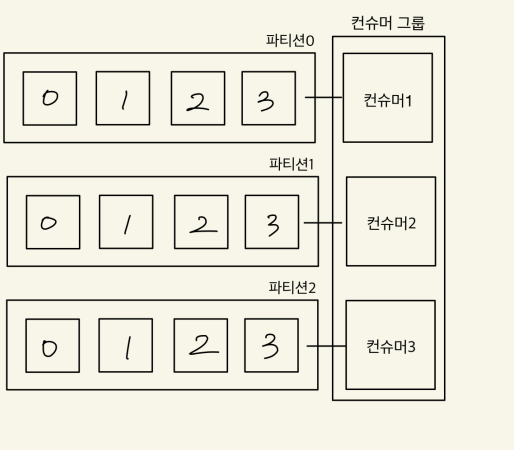
2. 컨슈머가 종료되는 경우



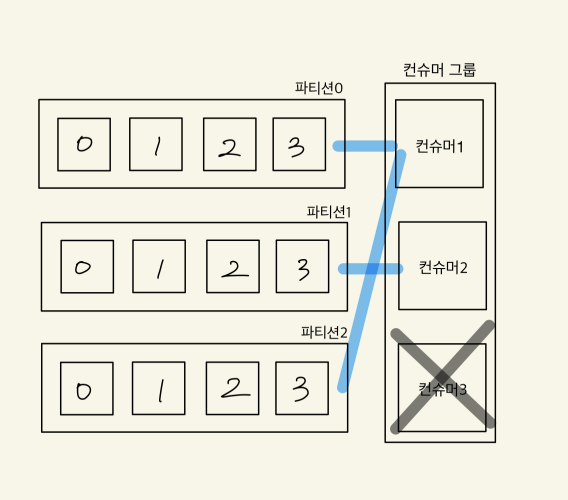
- 컨슈머 그룹에서 컨슈머 3이 빠지면서 리밸랜싱이 발생합니다.
- 기존의 매핑되있던 컨슈머-파티션 정보는 리셋되고, 다시 새롭게 매핑작업이 이뤄집니다.

6.4.2 스태틱 멤버십 적용 후 리밸랜싱

1. 컨슈머 그룹 현재 상태



2. 컨슈머가 종료되는 경우



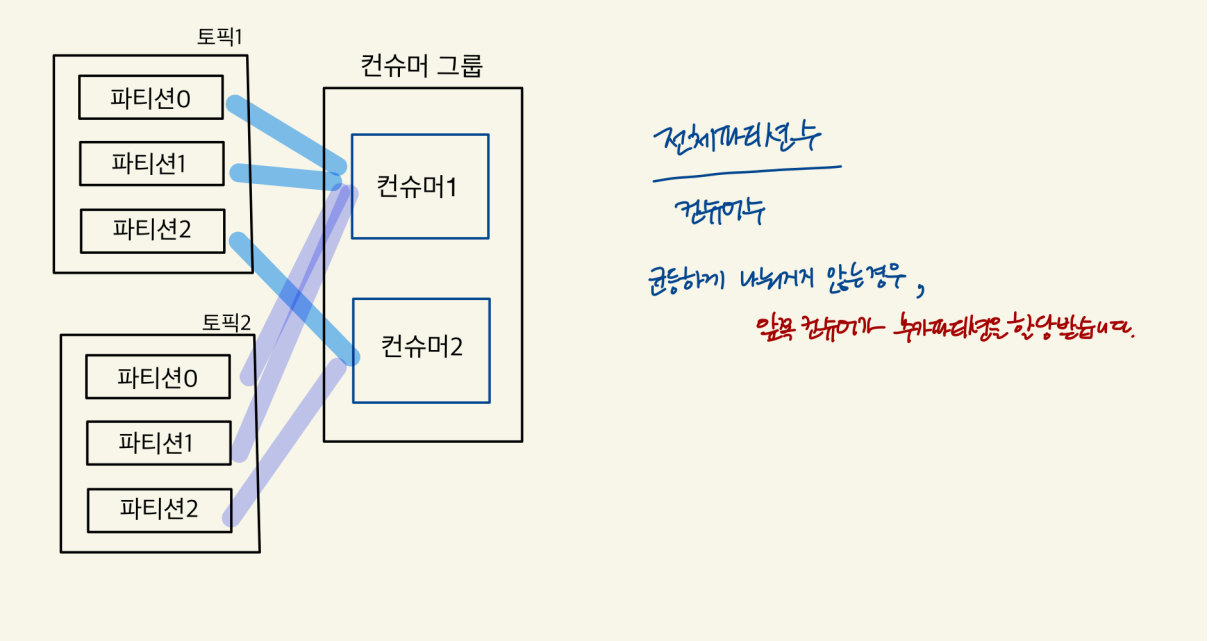
- session.time.out 전까지 리밸랜싱 작업은 이뤄지지 않습니다.

6.4 컨슈머 파티션 할당 전략

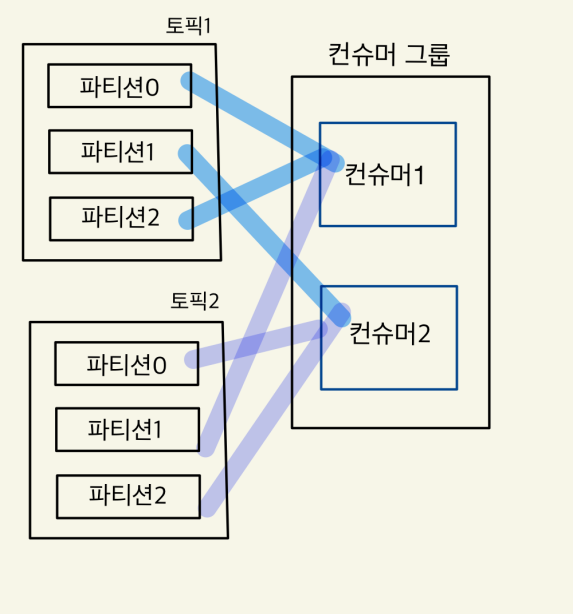
- 프로듀서가 어느 파티션으로 레코드를 전송할지 결정해야하는 것처럼, 컨슈머는 어느 파티션으로부터 레코드를 읽어올지 결정해야 합니다.
- 레인지 전략(RangeAssignor), 라운드 로빈전략(RoundRobinAssignor), 스티키 전략(stickyAssignor), 협력적 스티키 전략(CooperativeStickyAssignor)

6.4.1 레인지 파티션 전략

- 파티션 할당 전략의 기본값입니다.
- 불균형하게 파티션이 할당될 가능성이 있습니다.



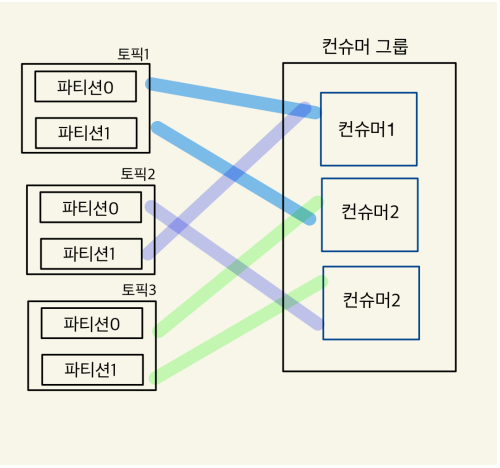
6.4.2 라운드 로빈 파티션 할당 전략

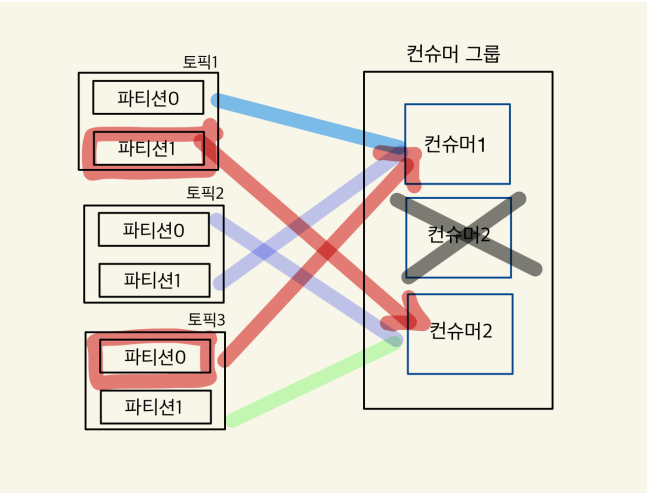


- 구독 대상의 토픽의 파티션을 나열합니다
- 컨슈머 그룹을 나열합니다.
- 나열된 파티션과 컨슈머들을 하나씩 라운드 로빈방식으로 매핑합니다.

6.4.3 스티키 할당 전략

- 기존에 매핑되던 파티션과 컨슈머를 최대한 유지하려는 전략입니다.
- 무조건 유지하는것이 아닌, 균형잡힌 파티션 할당을 지키면서 유지합니다.
  - 컨슈머들의 최대 할당된 파티션 수의 차이는 1을 넘어가면 안됩니다.
- 따라서, 일부 파티션은 기존의 컨슈머와 매핑을 유지하지 못하는 경우도 발생합니다.

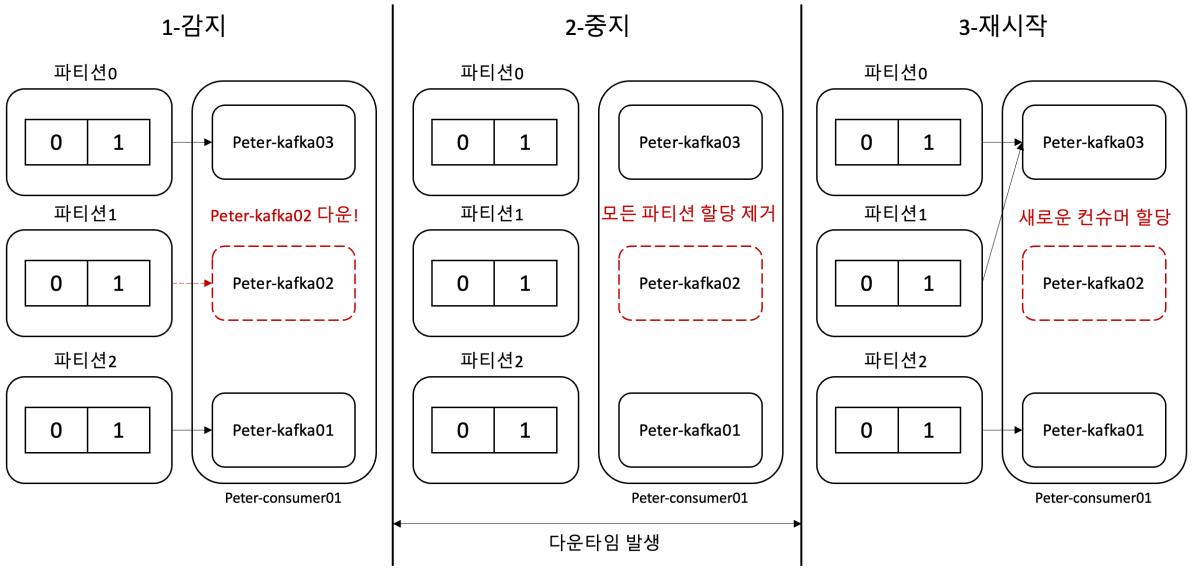




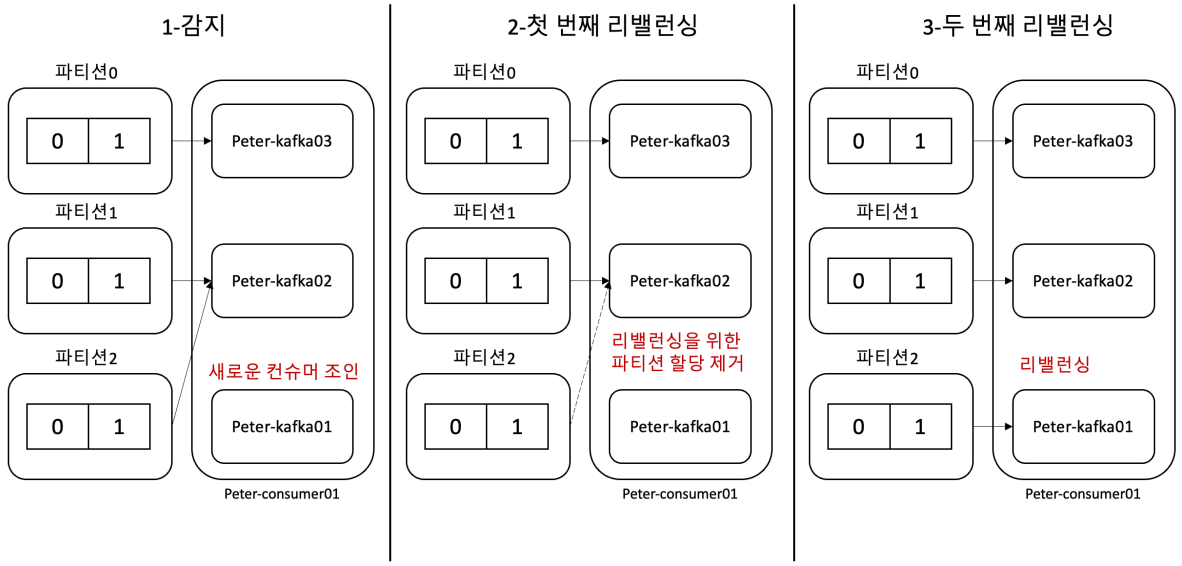
- 컨슈머 1과 3에 할당된 파티션들은 유지되고,
- 컨슈머 2에 할당된 토픽1-파티션1, 토픽3-파티션0이 재할당됩니다.

6.4.4 협력적 스티키 할당 전략

- 그동안의 리밸랭싱작업은 컨슈머에 할당된 모든 파티션을 취소하고 이뤄졌습니다
  - 파티션들의 소유권 문제 : 둘이상의 컨슈머가 동일한 파티션을 소유할 수 없습니다.
- 일반적인 리밸랜싱



- 협력적 스티키 리밸런싱 동작



1. 컨슈머 그룹의 peter-kafka01이 합류하면서 리밸런싱이 동작합니다.(1.감지)

2. 컨슈머 그룹 내 컨슈머들은 그룹 합류 요청과 자신들이 컨슘하는 토픽의 파티션정보를 그룹 코디네이터로 전송합니다.(1.감지)
3. 그룹 코디네이터는 해당 정보를 조합해 컨슈머 그룹의 리더에게 전송합니다.(1.감지)
4. 컨슈머 그룹의 리더는 현재 컨슈머들이 소유한 파티션 정보를 활용해 제외해야할 파티션 정보를 담은 새로운 파티션 할당 정보보를 컨슈머 그룹 맴버들에게 전달합니다.(2.첫 번째 리밸런싱)
5. 새로운 파티션 할당 정보를 받은 컨슈머 그룹 맴버들은 현재의 파티션 할당 전략과 차이를 비교해보고 필요 없는 파티션을 골라 제외합니다. 이전의 파티션 할당 정보와 새로운 파티션 할당 정보가 동일한 파티션들에 대해서는 어떤 작업도 수행할 필요가 없기 때문입니다.(2.첫 번째 리밸런싱)
6. 제외된 파티션 할당을 위해 컨슈머들은 다시 합류를 요청합니다. 여기서 두 번째리밸런싱이 트리거됩니다.

   (3.두 번째 리밸런싱)
7. 컨슈머 그룹의 리더는 제외된 파티션을 적절한 컨슈머에게 할당합니다.(3.두 번째 리밸런싱)

6.5 정확히 한 번 컨슈머 동작

- consumer.config에 `ISOLATION_LEVEL_CONFIG="read_committed"` 를 추가합니다.
- 해당 옵션은 컨슈머가 트랜잭션이 완료된 메시지만 읽도록하는 옵션값입니다.
- 하지만, 트랜잭션 컨슈머라고 해서 정확히 한번만 가지고 오는것은 아닙니다.
- 정확히 한번 처리를 위해서는, 컨슘-메시지처리-프로듀싱 동작이 모두 하나의 트랜잭션으로 처리해아합니다.
- `sendOffsetsToTransaction()` 메소드를 이용하여 컨슈머 그룹의 오프셋 커밋을 트랜잭션에 포함시킵니다.