KIP-88: OffsetFetch Protocol Update

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: *KAFKA-3853*

Released: 0.10.2.0

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

This KIP was prepared thanks to valuable feedback by Jason Gustafson.

Motivation

KAFKA-3853 asks for an improvement to the `describe` option of the consumer group command for new (Java API based) consumers. This command, when passed a consumer group that has no consumer (i.e., when the group state is `Empty`), currently reports an error indicating that there is no active member:

```
$ bin/kafka-consumer-groups.sh --new-consumer --bootstrap-server localhost:9092 --describe --group group1
Note: This will only show information about consumers that use the Java consumer API (non-ZooKeeper-based consumers).
Error: Consumer group 'group1' has no active members.
```

The requested improvement is returning offsets within the group (and leaving the consumer column empty) instead of returning the error message above. The error message can still be printed to `stderr` as a warning.

When the group is `Stable` (i.e. when there are active consumers in the group), the above command returns the associated topic partition assignment for each member of the group, and that assignment can be used to extract the corresponding committed offset(s). However, if the group state is `Empty` (i.e. when there are no active consumers in the group) there is no associated topic partition info in `DescribeGroups` response. Therefore, `DescribeGroups` response in its current protocol would not help.

The `OffsetFetch` protocol can be used to extract offsets associated with given topic partitions in a consumer group. The problem is, when consumer group is in `Empty` state or even when it is `Stable` but not all its topics are being consumed, currently there is no way to extract all its topic partitions that it has consumed from (i.e. has offset for). We can modify the behavior of `OffsetFetch` protocol so it returns all topic partitions associated with the group if it is passed a `null` value (as the list of topic partitions).

Public Interfaces

This is the current schema for `OffsetFetch` (version `1`, that applies to fetching from Kafka, and not ZooKeeper).

```
OffsetFetch Request (Version: 1) => group_id [topics]
  group_id => STRING
  topics => topic [partitions]
    topic => STRING
    partitions => partition
      partition => INT32

OffsetFetch Response (Version: 1) => [responses]
  responses => topic [partition_responses]
    topic => STRING
    partition_responses => partition offset metadata error_code
      partition => INT32
      offset => INT64
      metadata => NULLABLE_STRING
      error_code => INT16
```

The first suggestion is to use a similar `Request` / `Response` protocol and bump up the version to 2. There will be two main changes to version 1 of the protocol in version 2:

1. The first change to the protocol is that if no topics (`null` input for list of topics) are provided, the offset information of all topics (or topic partitions) associated with the group is returned. So the protocol will be slightly modified to define the input `topics` array as a nullable array.
2. With version 0 and 1 of the protocol, a list of topic partitions is passed as part of `OffsetFetch` request, and, in return, the same topic partitions are returned. Along with each topic partition in the response there is an `error_code` field for reporting any error that could occur in extracting offset for that topic partition (e.g. if the user is not authorized to access that topic partition, if the topic partition does not exist, ...). It is possible to pass an empty array of topic partitions in `OffsetFetch` request but in that scenario an error is never reported (irrespective of group or coordinator status) and an empty list is always returned in the response. In other words, there is no existing scenario, given the current protocol, in which an empty list is returned due to some error.
   But with the proposed change above (passing a null array in `OffsetFetch` request) it is now possible to have an empty list of topic partitions in the response because of an error situation (e.g. if there is a coordinator related error, or if user if not authorized to access the group). Therefore, it would not be possible to detect, for example, when the coordinator has moved to another broker or when it is still in the process of loading the offsets. This means it would be impossible to tell if there was an error or if there were just no offsets stored for the group. To handle these scenarios the `OffsetFetch` response should be able to report an error at the top level (not associated with individual topic partitions). In fact, reporting coordinator or group related errors at the top level of the response rather than along with each topic partition is a more reasonable solution. Topic partition specific errors can still be returned in the internal `error_code` field associated with the individual topic partition.

```
OffsetFetch Request (Version: 2) => group_id [topics]
  group_id => STRING
  topics => topic [partitions]
    topic => STRING
    partitions => partition
      partition => INT32

OffsetFetch Response (Version: 2) => [responses] error_code
  responses => topic [partition_responses]
    topic => STRING
    partition_responses => partition offset metadata error_code
      partition => INT32
      offset => INT64
      metadata => NULLABLE_STRING
      error_code => INT16
  error_code => INT16
```

```
public static final Schema OFFSET_FETCH_REQUEST_V2 = new Schema(new Field("group_id",
                                                                          STRING,
                                                                          "The consumer group id."),
                                                                          new Field("topics",
                                                                                    ArrayOf.nullable
(OFFSET_FETCH_REQUEST_TOPIC_V0),
                                                                                    "Topics to fetch
offsets."));

public static final Schema OFFSET_FETCH_RESPONSE_V2 = new Schema(new Field("responses",
                                                                           new ArrayOf
(OFFSET_FETCH_RESPONSE_TOPIC_V0)),
                                                                 new Field("error_code",
                                                                           INT16));
```

The second suggestion has to do with how the above API is accessed and called. Currently, the way the offset information for each topic partition in a `Stable` group is returned is through creating a "dummy" consumer in the group and use its `committed` interface to extract those offset information:

```
val consumer = getConsumer()
...
... consumer.committed(new TopicPartition(topicPartition.topic, topicPartition.partition)) ...
...
})
```

This `committed` call makes use of the `OffsetFetch` API to extract the offset of the given partition. The suggestion here is to add a method to `AdminClient` that extracts offset information of a consumer group by making a call to `OffsetFetch` API, and passing a `null` input as list of topic partitions. That will return all offsets of topic partitions associated with the consumer's group:

```
def listGroupOffsets(groupId: String): Map[TopicPartition, PartitionData] = {
    val coordinator = findCoordinator(groupId)
    val responseBody = send(coordinator, ApiKeys.OFFSET_FETCH, OffsetFetchRequest.
allPartitionsOffsetFetchRequest(groupId))
    val response = responseBody.asInstanceOf[OffsetFetchResponse]
    response.responseData().asScala.toMap
}
```

One benefit of using this method instead of using the `KafkaConsumer`'s `committed` method is that we no longer need to create the dummy consumer to retrieve offsets. The other benefit here is that with one API call all offsets within the group are returned. Whereas, in the existing describe group implementation, for each topic partition in the group one API call is made.

Proposed Changes

The proposal is to

1. Create version 2 of the `OffsetFetch` API in which,
    a. in the request a `null` value can also be passed as the list of topic partitions, and in response, offsets of all topic partitions associated with the group are returned.
    b. an `error_code` is returned at the top level to report coordinator or group related errors (scenarios in which an empty list is returned in the response), and to distinguish between when there is an error and when there are no stored offsets for the group.
2. Add a `listGroupOffsets` interface to `AdminClient` that makes use of the updated `OffsetFetch` API above and returns offsets of all topic partitions associated with the consumer's group.

Compatibility, Deprecation, and Migration Plan

With respect to the first proposed change above there should be no concern when it comes to compatibility because the older APIs (versions 0 and 1) did not accept a `null` value for the input array. In the case where current users somehow rely on the exception raised when a `null` value is passed to the current `OffsetFetch` API as the list of topic partitions, they can stick to the current implementation of the API by using version 1 of the API. The behavioral change suggested in this KIP would apply to version 2 of the API only. The addition of the `error_code` field in the response is also limited to version 2 of the API and the existing clients would have to either stick to version 1 of the API, or modify their implementation to look for errors in two places (the top level `error_code` and the `error_code` associated with each topic partition).

With respect to the added interface to `AdminClient` there would be no issue as that interface does not exist in current implementation.

Rejected Alternatives

1. Changing the `DescribeGroups` protocol so it also returns the offset information for all topic partitions from which the group has consumed from since its creation. More detailed can be found here.
2. Exposing the added `OffsetFetch` behavior through a new interface in `KafkaConsumer`, which would still imply that the dummy consumer has to be created in the group in order to retrieve offsets. More details can be found here.
3. Passing an empty list, instead of `null`, to the API to get all offsets. The `null` value was chosen to remain consistent with how some other APIs handle a similar situation (e.g. `TopicMetadata` API returns metadata of all topics if it is provided with a `null` value as list of topics). More details can be found here.
4. On the solution for reporting errors when there is no topic partition in the response, the following options were also considered, but set aside in favor of the cleaner solution provided above:
    a. inserting a "dummy" partition into the response so that we have somewhere to return an error code.
    b. including no error code, but using a null array in the response to indicate that there was some error. If there was no error, and the group simply had no partitions, then we return an empty array. I guess in this case, if the client receives a null array in the response, it should assume the worst
    and rediscover the coordinator and try again (because there is no way to indicate what the error is).