KIP-122: Add Reset Consumer Group Offsets tooling

Status

Current state: Accepted

Discussion thread: here

JIRA: Unable to render Jira issues macro, execution error.

Released: 0.11.0.0

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

Motivation

This KIP wants to introduce a command-line tool to enable users to reset Consumer Group offsets.

Today, if users want to reprocess *old* records (i.e. records where offset is smaller than current consumer offset), they will need to modify client-side code (eg. using "KafkaConsumer#seek()" operations).

This process will involve write their own program which starts a consumer and commits offset or change client-side code, redeploy the application, and eventually rollback changes in application to consume records as usual. This is a cumbersome process given that users have to be aware of how many partitions are by topic, in which offset they are, and to which offset they want to move to.

Bringing the option to reset offsets from outside of the application will enable a cleaner and consistent way to achieve reprocessing and move along the topic.

Currently, there are 2 indexes that allow users to reprocess records: **offset** index and **timestamp** index.

Offset index represent a pointer over the sequential id that each record has by partition, available from Kafka inception.

Timestamp index (added in release 0.10.1.0 by KIP-33) represents a pointer over the epoch milliseconds from record creation, that could be defined by the Producer or the Broker, depending in the Timestamp Type.

From KIP-33:

"Searching by timestamp will have better accuracy. The guarantees provided are:

* The messages whose timestamp are after the searched timestamp will be consumed.

* Some messages with earlier timestamp might also be consumed."

Proposed Changes

We are considering to add the options in ConsumerGroupCommand to reset offsets.

There will be 3 **execution options** available:

1. **Plan:** Print out the result of the operation (i.e. show a list of topic, partition, current offset, new offset to reset if the operation is executed). This will be the default option.
2. **Execute:** Users will have to explicitly ask to execute the *reset* operation depending on the scenario and the scope specified.
3. **Export:** Export plan to a CSV file to execute it later.

All these options are considered to be executed affecting only one Consumer Group.

The following scenarios will be supported:

ID	Scenario	Description
1.	**Reset to Datetime**	When we want to reset offsets to an specific point in time. (e.g. to 1/1/2017 at 00:00 to reprocess all records from these year)
2.	**Reset from Duration**	When we want to go back some period ago (e.g. P7D to reprocess all records from one week ago)
3.	**Reset to Earliest**	When we want to reprocess all the records available by partition.
4.	**Reset to Latest**	When a Consumer Groups has an offset lag and we don't want to process missing records, only move to the latest.
5.	**Reset to Current Time**	When we want to only print out and/or backup the current offset by partition.
6.	**Reset to Offset**	When we want to move to an specific offset.
7.	**Shift Offset by 'n'**	When we want to move forward or backward from the current position. 'n' can be a positive or negative value that will be add it to the current offset to move to a new position.
8.	**Reset from file**	When we have a file with the required offsets by topic/partition to reset to.

And the following scopes:

1. **All Topics consumed by Consumer Group:** This scope will consider all the topics that has been consumed by a Consumer Group.
2. **Specific List of Topics:** This scope will consider all the topics defined by user.
3. **One Topic, All Partitions:** This scope will consider only one topic and all partitions.
4. **One Topic, Specific Partition:** This scope will consider only one topic and partition specified by user.

You can check the concept-proof implementation of this feature on this pull request.

Public Interfaces

Consumer Group Reset Offset options

Main option

--reset-offsets

This option should be executed independently from other Consumer Group options (list, describe, delete, etc.) and it will only support **new consumers**.

Required Arguments

ID	Argument	Type	Descriptio
1.	--group	Required	Consumer Group ID.
2.	--bootstrap-server	Required	Server to connect to.

Scenarios

At least one of the scenarios should be defined to proceed with the execution

ID	Scenario	Arguments	Considerations	Example

1.	Reset to Datetime	**--to-datetime** YYYY-MM-DDTHH:mm:SS.sss±hh:mm **--to-datetime** YYYY-MM-DDTHH:mm:SS.sssZ **--to-datetime** YYYY-MM-DDTHH:mm:SS.sss	Datetime must be specified in ISO8601 format. This option will translate the datetime to Epoch milliseconds, find the offsets by timestamp, and reset to those offsets. If the Timezone is not specified, it will use UTC.	Reset to first offset since 01 January 2017, 00:00:00 hrs *--reset-offsets –group test.group --topic foo --to-datetime 2017-01-01T00:00:00Z*
2.	Reset by Duration	**--by-duration** PnDTnHnMnS	Duration must be specified in ISO8601 format. This option will subtract the duration to the current timestamp in the server, and find the offsets using that subtracted timestamp, and reset to those offsets. The duration specified won't consider daylight saving effects.	Reset to first offset since one week ago (from current timestamp): *--reset-offsets --group test.group --topic foo --by-duration P7D*
3.	Reset to Earliest	**--to-earliest**	This option will reset offsets to the earliest using Kafka Consumer's `#seekToBeginning`	Reset to earliest offset available: --reset-offsets --group test.group --topic foo --to-earliest
4.	Reset to Latest	**--to-latest**	This option will reset offsets to the earliest using Kafka Consumer's `#seekToEnd`	Reset to latest offset available: --reset-offsets --group test.group --topic foo --to-latest
5.	Reset to Current Position	**(no scenario arguments)**	This option won't reset the offset. It will be used to print and export current offset.	Reset to current position: --reset-offsets --group test.group --topic foo
6.	Reset to Offset	**--to-offset**	This option will reset offsets to an specific value.	Reset to offset 1 in all partitions: --reset-offsets --group test.group --topic foo --to-offset 1
7.	Shift Offset by 'n'	**--shift-by** *n*	This option will add the `n` value to the current offset, and reset to the result. `n` can be a positive or negative value, so offset will be move backward if it is negative, and forward if it is positive. If current offset + n is higher than the *latest* offset, new offset will be set to *latest*. If current offset + n is lower than the *earliest* offset, new offset will be set to *earliest*.	Reset to current offset plus 5 positions: --reset-offsets --group test.group –topic foo --shift-by 5
8	Reset from File	**--from-file** PATH_TO_FILE	This option will take a Reset Plan CSV file with the offsets to reset by topics/partitions. It does not require scope, because topics and partitions are defined in the file.	Reset using a file with reset plan: --reset-offsets --group test.group --from-file reset-plan.csv

Scopes

The scopes will be defined extending the existing `topic` argument, using the following format:

```
--topic <topic-name>:<partition numbers>
ex: --topic topic1 --topic topic2:0,1,2
```

ID	Scope	Arguments	Description	Example
1.	All topics	**--all-topics**	In this case the tool will run the scenario considering all topics that has been consumed by Consumer Group. It will consider all partitions by topic assigned.	Reset offsets in all topics and partitions: *--reset-offsets --group test.group --all-topics –to-earliest*
2.	List of Topics	**--topic** *topic1* **--topic** *topic2*	In this case, the tool first will validate that input topics are been consumed by Consumer Group, and then run the scenario. It will consider all partitions by topic defined.	Reset offsets on all partitions of a list of topics: *--reset-offsets --group test.group --topic foo --topic bar –to-earliest*

3.	Topics and List of Partitions	**--topic** *topic1:0,1,2* **--topic** *topic2:0*	In this case, the tool first will validate that input topic is been consumed by Consumer Group, and then run the scenario. It will consider only the partitions specified by topic.	Reset offsets on specific partitions by topics: *--reset-offsets --group test.group --topic foo:1,2 --topic bar:0,1,2,3 –to-earliest*
4.	Topic	**--topic** *topic1*	In this case, the tool first will validate that input topic is been consumed by Consumer Group, and then run the scenario. It will consider all partitions.	Reset offsets on a specifc topic, all partitions; *--reset-offsets --group test.group --topic foo --to-earliest*

Execution Options

ID	Option	Arguments	Description	Examples

1.	Plan	**(no execution arguments)**	This execution option will only print out the result of the scenario by scope.
			
			The output will look like this:
			
			`TOPIC PARTITION NEW-OFFSET NEW-LAG LOG-END-OFFSET CONSUMER-ID HOST CLIENT-ID`
			`foo 0 90 10 100 - - -`
				Prints result of resetting all topics and partitions to earliest:
				
				--reset-offsets --group test.group --all-topics –to-earliest

2.	Execute	**--execute**	This execution option will run the reset offset process based on scenario and scope.	Prints and execute resetting all topics and partitions to earliest: *--reset-offsets --group test.group --all-topics –to-earliest --execute*
3.	Export	**--export**	This execution option will print out the reset plan in CSV format, that later could be used in the scenario 8. (i.e. as backup)	Prints plan for resetting all topics and partitions to earliest in CSV format: *--reset-offsets --group test.group --all-topics –to-earliest --export* Execute and Prints plan for resetting all topics and partitions to earliest in CSV format: *--reset-offsets --group test.group --all-topics –to-earliest --export --execute*

Reset Plan File: CSV Format

The CSV schema will consist on:

```
<topic>,<partition>,<offset>
```

Sample:

```
topic-1,0,100
topic-1,1,200
topic-1,2,0
topic-2,0,0
```

Implementation Details

These options will be implemented inside the `ConsumerGroupCommand`. The `resetOffsets` operation will looks like this:

```
def resetOffsets(): Map[PartitionAssignmentState, Long] = {
  val groupId = opts.options.valueOf(opts.groupOpt)
  val (state, assignments) = describeGroup() //(1)
  assignments match {
    case None =>
      // applies to both old and new consumer
      printError(s"The consumer group '$groupId' does not exist.")
      Map.empty
    case Some(assignments) =>
      state match {
        case Some("Dead") =>
          printError(s"Consumer group '$groupId' does not exist.")
          Map.empty
        case Some("Empty") => //(2)
          val assignmentsToReset = getAssignmentsToReset(assignments) //(3)
          val assignmentsPrepared = prepareAssignmentsToReset(assignmentsToReset) //(4)
          val execute = opts.options.has(opts.executeOpt)
          if(execute) //(5)
            resetAssignments(assignmentsPrepared)
          assignmentsPrepared
        case Some("PreparingRebalance") | Some("AwaitingSync") =>
          printError(s"Consumer group '$groupId' offsets cannot be reset if it is rebalancing.")
          Map.empty
        case Some("Stable") =>
          printError(s"Consumer group '$groupId' offsets cannot be reset if it has members active.")
          Map.empty
        case other =>
          // the control should never reach here
          throw new KafkaException(s"Expected a valid consumer group state, but found '${other.getOrElse
("NONE")}'.")
      }
  }
}
```

(1) It will get assignments from `describeGroup` operation

(2) Then use `getAssignmentsToReset` to filter the assignments with the values defined by `–topic` of or `--all-topics`.

(3) Then get new offset by assignments using `prepareAssignmentsToReset`.

(4) It will be only executed when the ConsumerGroup selected **is inactive** o avoid race conditions.

(5) It will be executed only if it is asked explictly. This will consist in create a Consumer using the same `group.id` and use:

```
consumer.assign(List(topicPartition).asJava)
consumer.seek(topicPartition, offset)
consumer.commitSync()
```

To change the offsets.

Compatibility, Deprecation, and Migration Plan

This KIP won't require a Migration Plan.

'reset-to-datetime' and 'reset-to-period' will require Timestamp Index, which will make them only available from version 0.10.1.0.

Nevertheless, it is possible to achieve the same result since release 0.10.0 using timestamp metadata by record, doing a linear search starting from latest. However, we are not considering to support this option because it will be an expensive/slow option.

All the other options will be available for releases from 0.10.0.0 given that KIP-97 is implemented.

This KIP will not support old consumers that store offsets in Zookeeper.

Test Plan

- A unit test to validate that --reset-offset is executed independently from other Consumer Group options (list, delete, describe, etc.)
- A unit test to validate that only one scenario is specified
- A unit test to validate that only one scope is specified
- A unit test to validate that only one execution option is specified
- A unit test by combination of scenario, scope and execution option.
- A unit test to validate that when calculated offset by partition is older that the earliest offset, tool resets offset to earliest (e.g. when we specified --reset-to 0, when earliest offset is 10)
- A unit test to validate that when calculated offset by partition is bigger that the latest offset, tool resets offset to latest (e.g. when we specified --reset-to 0, when earliest offset is 10)

Rejected Alternatives

None

(no scenario arguments)