KIP-734: Improve AdminClient.listOffsets to return timestamp and offset for the record with the largest timestamp

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: *KAFKA-12541*

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

Motivation

Confirming topic/partition "liveness" (i.e. whether a topic is being actively written to) can currently be difficult in Kafka. Approaches include:

- consuming from the partition - this is heavyweight for an indicator.
- using metrics - this can be hard to access remotely.
- monitoring changes in offsets - this can be difficult with partitions that are not often written to.

In a lot of cases the highest timestamp on the partition provides a good indication of "liveness" and the AdminClient provides an existing good interface to collect this information.

Note: this proposal returns the highest message timestamp but this may not correspond to the time at which the latest message was produced. Replication tooling such a MM2 may preserve the source cluster timestamp in the message. However, it's understood that this will provide a good indication in most cases.

In Kafka 2.7 the following method was added to AdminClient that provides offset and timestamp information for records at various positions in a partition:

```
public ListOffsetsResult listOffsets(Map<TopicPartition,OffsetSpec> topicPartitionOffsets,
              ListOffsetsOptions options)
```

https://kafka.apache.org/27/javadoc/org/apache/kafka/clients/admin/KafkaAdminClient.html#listOffsets-java.util.Map-org.apache.kafka.clients.admin.ListOffsetsOptions-

This returns offsets and timestamps of records in the partitions that match the OffsetSpec.

OffsetSpec can be:

- OffsetSpec.EarliestSpec - The first offset on the partition and a timestamp of -1
- OffsetSpec.LatestSpec - The offset of the next message that will be appended to the log and a timestamp of -1
- OffsetSpec.TimestampSpec - The earliest offset whose timestamp is greater than or equal to the given timestamp and the timestamp of that record.

This proposal adds an additional offset spec:

```
OffsetSpec.MaxTimestampSpec
```

This returns the offset and timestamp corresponding to the record with the highest timestamp on the partition.

Public Interfaces

We will add the new offset spec in org.apache.kafka.clients.admin.OffsetSpec:

```
public class OffsetSpec {

    public static class EarliestSpec extends OffsetSpec { }

    public static class LatestSpec extends OffsetSpec { }

    public static class MaxTimestampSpec extends OffsetSpec { } // this is new

....

    /**
    * Used to retrieve the offset with the largest timestamp of a partition
    * as message timestamps can be specified client side this may not match
    * the log end offset returned by LatestSpec
    */
    public static OffsetSpec maxTimestamp() { // this is new
        return new MaxTimestampSpec();
    }

....
```

We will need to bump the ListOffsets API version to do this to ensure that requests made to earlier brokers that do not understand this specification are failed. This would be enforced in ListOffsetsRequest.Builder with something similar to the below (where 3 is the new version):

```
public static Builder forMaxTimestamp(IsolationLevel isolationLevel) {
  return new Builder(3, ApiKeys.LIST_OFFSETS.latestVersion(), CONSUMER_REPLICA_ID, isolationLevel);
}
```

Proposed Changes

OffsetSpecs are mapped to Long values for timestamp fetches in org.apache.kafka.clients.admin.KafkaAdminClient:

```
       long offsetQuery = (offsetSpec instanceof TimestampSpec)

           ? ((TimestampSpec) offsetSpec).timestamp()

           : (offsetSpec instanceof OffsetSpec.EarliestSpec)

               ? ListOffsetsRequest.EARLIEST_TIMESTAMP

               : ListOffsetsRequest.LATEST_TIMESTAMP;
```

this will be refactored to:

```
        long offsetQuery = getOffsetFromOffsetSpec(offsetSpec);

...

    private long getOffsetFromOffsetSpec(OffsetSpec offsetSpec) {

        if (offsetSpec instanceof TimestampSpec) {

            return ((TimestampSpec) offsetSpec).timestamp();

        } else if (offsetSpec instanceof OffsetSpec.EarliestSpec) {

            return ListOffsetsRequest.EARLIEST_TIMESTAMP;

        } else if (offsetSpec instanceof OffsetSpec.MaxTimestampSpec) {

            return ListOffsetsRequest.MAX_TIMESTAMP;

        }

        return ListOffsetsRequest.LATEST_TIMESTAMP;

    }
```

This corresponds to a new constant in org.apache.kafka.common.requests.ListOffsetRequest:

```
public static final long MAX_TIMESTAMP = -3L;
```

This follows the existing pattern of using negative timestamps to indicate the ends of the log

This new timestamp will be handled as normal all the way through the following calls

```
KafkaApis.handleListOffsetRequest

KafkaApis.handleListOffsetRequestV1AndAbove

ReplicaManager.fetchOffsetsForTimestamps

ReplicaManager.fetchOffsetForTimestamp

Partition.fetchOffsetForTimestamp

Log.fetchOffsetByTimestamp
```

LogSegments track the highest timestamp and associated offset so we don't have to go to disk to fetch this. An extra path in this block: https://github.com/apache/kafka/blob/trunk/core/src/main/scala/kafka/log/Log.scala#L1737 is added to handle the MAX_TIMESTAMP behaviour:

```
    } else if (targetTimestamp == ListOffsetsRequest.MAX_TIMESTAMP) {

        val latestTimestampSegment = logSegments.maxBy(_.maxTimestampSoFar)

        val latestEpochOpt = leaderEpochCache.flatMap(_.latestEpoch).map(_.asInstanceOf[Integer])

        val epochOptional = Optional.ofNullable(latestEpochOpt.orNull)

        Some(new TimestampAndOffset(latestTimestampSegment.maxTimestampSoFar,

            latestTimestampSegment.offsetOfMaxTimestampSoFar,

            epochOptional))

    }
```

Note: an OffsetSpec of Latest actually returns the next offset for the partition (i.e. the one beyond the last committed one) whereas Max Timestamp is expected not to move beyond the end of the log. This means that even when messages are committed in timestamp order MAX_TIMESTAMP and LATEST will return different offsets, e.g. given 30 offsets on a partition in timestamp order:

OffsetSpec	Offset	Timestamp
MAX_TIMESTAMP	30	> 0
LATEST	31	-1

Compatibility, Deprecation, and Migration Plan

The only behaviour changes created by this KIP are for requests to AdminClient.listOffsets that are looking for a timestamp of -3L. As this is not a valid timestamp there should be no migration, compatibility or deprecation concerns. As mentioned in public interfaces, this functionality is not backwards compatible and attempt to fetch by max timestamp on earlier brokers will be failed.

Rejected Alternatives

- Create a new API to retrieve the maximum timestamp record information, e.g. AdminClient.getMaxTimestampAndOffset(TopicPartition topicpartition). Rejected as AdminClient.listOffsets is already a good match for this functionality.
- Extend FetchedTimestampAndOffset to include a timestamp type - timestamp type (log append time or creation time) is a useful piece of information to return in any offset fetching method. It was considered to enrich the listOffsets API with this information as part of this KIP. However, this would involve significant changes to the existing API and for all but some corner cases (where the timestamp type used for the topic changes over time) the timestamp type can be fetched using the AdminClient (describeConfigs) at the topic level.