KIP-842: Add richer group offset reset mechanisms

Status

Current state: Under Discuss

JIRA: *here*

Pull Request: *here*

Discussing thread: *here*

Vote thread: *here*

Motivation

When the server expands partitions for a topic, the producer firstly perceives the expansion, and some data is written in the newly expanded partitions. But the consumer group perceives the expansion later, after the rebalance is completed, the newly expanded partitions will be consumed from the latest if "auto.offset.reset" is set to "latest". Within a period of time, the data of the newly expanded partitions is skipped and lost by the consumer. But for a group that is consuming, it does not want to skip the data. Therefore, we hope to provide some richer offset reset mechanisms to solve this problem, and secondly, to deal with the problem of out of range more flexibly.

For this case of data loss, I did a test. you can see the linked JIRA for details.

Public Interfaces

- Add an additional enum class named InitialOffsetResetStrategy to represent strategies for group startup with no initial offset.

InitialOffsetResetStrategy.java

```
package org.apache.kafka.clients.consumer;

import java.util.Locale;

public enum InitialOffsetResetStrategy {
    NONE, LATEST_ON_START, EARLIEST_ON_START;

    @Override
    public String toString() {
        return super.toString().toLowerCase(Locale.ROOT);
    }
}
```

- Add an additional enum class named InvalidOffsetResetStrategy to represent strategies for handling out-of-range.

InvalidOffsetResetStrategy.java

```
package org.apache.kafka.clients.consumer;

import java.util.Locale;

public enum InvalidOffsetResetStrategy {
    NONE, NEAREST;

    @Override
    public String toString() {
        return super.toString().toLowerCase(Locale.ROOT);
    }
}
```

- Then add two new consumer config named "auto.offset.reset.on.initial.offset" and "auto.offset.reset.on.invalid.offset"

ConsumerConfig

```
public static final String INVALID_OFFSET_RESET_CONFIG = "auto.offset.reset.on.invalid.offset";
private static final String INVALID_OFFSET_RESET_CONFIG_DOC = "If not NONE, when out of range errors
will reset the consumer's offset according to strategy. For example of NEAREST, to the earliest end of
the broker range if it was under the range, or to the latest end of the broker range if it was over the
range. If set NONE, fall back to auto.offset.reset";
public static final String INITIAL_OFFSET_RESET_CONFIG = "auto.offset.reset.on.initial.offset";
private static final String INITIAL_OFFSET_RESET_CONFIG_DOC = "If not NONE, when group startup for the
first time with no initial offset, it will reset to latest or earliest by LATEST_ON_START and
EARLIEST_ON_START, If set NONE, fall back to auto.offset.reset";

CONFIG = new ConfigDef().define(INVALID_OFFSET_RESET_CONFIG,
                                Type.STRING,
                                                    InvalidOffsetResetStrategy.NONE.
toString(),
                                in(Utils.enumOptions(InvalidOffsetResetStrategy.class)),
                                Importance.MEDIUM,
                                                    INVALID_OFFSET_RESET_CONFIG_DOC)
                                            .define(INITIAL_OFFSET_RESET_CONFIG,
                                Type.STRING,
                                                        InitialOffsetResetStrategy.NONE.toString(),
                                in(Utils.enumOptions(InitialOffsetResetStrategy.class)),
                                Importance.MEDIUM,
                                                        INITIAL_OFFSET_RESET_CONFIG_DOC)
```

Proposed Changes

1. In addition to the "earliest", "latest", and "none" provided by the existing "auto.offset.reset", it also provides more abundant reset semantics, such as "latest_on_start" (application startup is reset to latest, and an exception is thrown if out of range occurs), "earliest_on_start" (application startup is reset to earliest, and an exception is thrown if out of range occurs), "nearest"(determined by "auto.offset.reset" when the program starts, and choose earliest or latest according to the distance between the current offset and log start offset and log end offset when out of range occurs).
2. "auto.offset.reset.on.no.initial.offset": Indicates the strategy used to initialize the offset. The default value is the parameter configured by "auto.offset.reset". If so, the strategy for initializing the offset remains unchanged from the previous behavior, ensuring compatibility. If the parameter is configured with "latest_on_start" or "earliest_on_start", then the offset will be reset according to the configured semantics when initializing the offset. In this way, the problem of data loss during partition expansion can be solved: configure "auto.offset.reset.on.no.initial.offset" to "latest_on_start", and configure "auto.offset.reset" to earliest.
3. "auto.offset.reset.on.invalid.offset": Indicates that the offset is illegal or out of range occurs. The default value is the parameter configured by "auto.offset.reset". If so, the processing of out of range is the same as before to ensure compatibility. If "nearest" is configured, then the semantic logic corresponding to "nearest" is used only for the case of out of range.

The semantics of "auto.offset.reset" remain unchanged. In order to describe in more detail what these parameters mean, and how they behave in various situations. We decide two categories where need reset offset.

"auto.offset.reset.on.no.initial.offset" (When the group starts consumption for the first time, the offset needs to be initialized):

initial offset reset strategy	proposed reset behavior when set initial offset

none	fall back to *auto.offset.reset*: if none, throw exception if earliest, reset to earliest if latest, reset to latest
earliest_on_start	reset to earliest
latest_on_start	reset to latest

"auto.offset.reset.on.invalid.offset" (When out of range or other abnormal offset inconsistencies occur during consumption):

invalid offset reset strategy	proposed reset behavior when trigger out of range
none	fall back to *auto.offset.reset*: if none, throw exception if earliest, reset to earliest if latest, reset to latest
nearest	to the earliest if it was under the range, or to the latest if it was over the range.

Compatibility, Deprecation, and Migration Plan

Existing or old behaviors have no impact. It only provide some rich mechanisms to use, users can choose to use according to their own needs, the existing behaviors will be retained and will not be changed.

Rejected Alternatives

For the problem of losing data due to expand partitions, it is not necessarily set "auto.offset.reset=earliest" for a huge data flow topic when starts up, this will make the group consume historical data from the broker crazily, which will affect the performance of brokers to a certain extent. Therefore, it is necessary to consume these new partitions from the earliest separately, which is set:"auto.offset.reset.on.no.initial.offset"="latest_on_start", "auto.offset.reset"="earliest".

It has been implemented according to Proposed Changes, see pr: https://github.com/apache/kafka/pull/10726