KIP-686: API to ensure Records policy on the broker

Status

Current state: *Under Discussion*

Discussion thread: Dev-list

JIRA: KAFKA-10732

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

Motivation

During the adoption of Kafka in large enterprises, it's important to guarantee data in some topic conforms to the specific format.

When data are written and read by the different applications developed by the different teams it's hard to guarantee data format using only custom SerDe, because malicious applications can use different SerDe.

The data format can be enforced only on the broker side.

Public Interfaces

New public interfaces `RecordsPolicy, Record`

RecordsPolicy.java

```
package org.apache.kafka.server.policy;

import org.apache.kafka.common.Configurable;
import org.apache.kafka.common.errors.PolicyViolationException;
import org.apache.kafka.common.Record;

/**
 * <p>An interface for enforcing a policy on message in topics.
 *
 * <p>Common use case is requiring that the messages has desired format.
 *
 * <p>If <code>message.policy.class.name</code> is defined, Kafka will create an instance of the specified class
 * for each topic using the default constructor and will then pass the broker configs to its <code>configure()<
/code> method. During
 * broker shutdown, the <code>close()</code> method will be invoked so that resources can be released (if
necessary).
 */
public interface RecordsPolicy extends Configurable, AutoCloseable {
    /**
     * Validate the records and throw a <code>PolicyViolationException</code> with a suitable error
     * message if the records for the provided topic do not satisfy this policy.
     *
     * Clients will receive the POLICY_VIOLATION error code along with the exception's message. Note that
validation
     * failure only affects the relevant topic, other topics in the request will still be processed.
     *
     * @param records Records to validate.
     * @param partition Partition number.
     * @throws PolicyViolationException if the records do not satisfy this policy.
     */
    void validate(String topic, int partition, Iterable<? extends Record> records) throws
PolicyViolationException;
}
```

Record.java

```
package org.apache.kafka.common;

import java.nio.ByteBuffer;
import org.apache.kafka.common.header.Header;
import org.apache.kafka.server.policy.RecordsPolicy;

/**
 * A log record is a tuple consisting of a unique offset in the log, a sequence number assigned by
 * the producer, a timestamp, a key and a value.
 *
 * @see RecordsPolicy
 */
public interface Record {
    /**
     * Get the record's timestamp.
     * @return the record's timestamp
     */
    long timestamp();

    /**
     * Check whether this record has a key
     * @return true if there is a key, false otherwise
     */
    boolean hasKey();

    /**
     * Get the record's key.
     * @return the key or null if there is none
     */
    ByteBuffer key();

    /**
     * Check whether a value is present (i.e. if the value is not null)
     * @return true if so, false otherwise
     */
    boolean hasValue();

    /**
     * Get the record's value
     * @return the (nullable) value
     */
    ByteBuffer value();

    /**
     * Get the headers. For magic versions 1 and below, this always returns an empty array.
     *
     * @return the array of headers
     */
    Header[] headers();
}
```

Proposed Changes

I propose to introduce the

- **`org.apache.kafka.common.Record`** - new public interface. Expose information about Records to be added to the topic.
- **`org.apache.kafka.server.policy.RecordsPolicy`** - new public interface. Implementation of this interface contains specific logic to check records to be added to the topic.
- **`records.policy.class.name: String`** - Configuration option - sets class name of the implementation of `RecordsPolicy` for the specific topic.
- **`records.policy.enabled: Boolean`** - Configuration option - enable or disable records policy for the topic

If `records.policy.enabled=true` than an instance of the `RecordsPolicy` should check each `Record` batch before applying data to the log.

If `PolicyViolationException` thrown from the `validate` methods then no data added to the log and the client receives an error.

Compatibility, Deprecation, and Migration Plan

There are no compatibility issues or deprecation.

No migration required.

Rejected Alternatives

-