Dynamically Adjust Log Levels in Connect

Status

Current state: *Draft (WIP).*

Discussion thread: here

Vote thread: here

JIRA: KAFKA-7772

Released: AK x.y.z

Pull request: https://github.com/apache/kafka/pull/6069

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

Motivation

Kafka Connect does not provide an out-of-the-box facility to change log levels. When debugging connectors or the Connect framework, one has to update the `log4j.properties` file and restart the worker to see new logs. This is cumbersome in most cases, and restarting the worker sometimes hides bugs by resetting the internal state.

Proposed Changes

The Kafka broker currently has a utility to adjust log levels via MBeans. This is a scala utility, that we will rewrite in Java, move to kafka-clients package and use this utility to initialize JMX interface in the Kafka broker and Connect worker upon startup.

Public Interfaces

We propose adding the following log4j controller:

```
public interface Log4jControllerMBean {

    /**
     * @return a list of all registered loggers
     */
    List<String> getLoggers();

    /**
     * Get the effective log level for a given logger
     *
     * @param logger name of the logger
     * @return its log level ("INFO", for example)
     */
    String getLogLevel(String logger);

    /**
     * Set the log level for a logger
     *
     * @param logger name of the logger
     * @param level desired level ("INFO", for example)
     * @return true, if successfully set, false otherwise.
     */
    Boolean setLogLevel(String logger, String level);
}
```

Applications will register this mBean with the mBean server via a `LogLevelManager` utility class:

```
public class LogLevelManager {
    /**
     * Create and register a JMX mBean called Log4jController in the specified domain
     *
     * @param domain domain of the mBean
     */
    public static void registerLog4jController(String domain) {
        // implementation
    }
}
```

Example Usage

An application (such as the Connect worker) will use the `LogLevelManager` utility to initialize dynamic logging as follows:

```
LogLevelManager.registerLog4jController("kafka.connect");
```

This will provide a JMX bean `Log4jController` in the `kafka.connect` domain, that will include a `Loggers` attribute, along with two operations: `getLogLevel` and `setLogLevel` that we can use to get or set log levels for individual loggers in the process.

Breaking Changes

Although it was not formally introduced via a KIP, Kafka brokers already provide a similar feature. With this change, we bring about the following changes in it.

1. The name of the JMX bean changes from `kafka.Log4jController` to simply `Log4jController`. However, the bean will continue to be present in the `kafka` domain.
2. The `getLogLevel` operation now returns the effective log level. This means that previously, this operation could return null, if a log level was not explicitly set on the given logger. With the proposed changes, the log level of the parent logger will be queried until a non-null level is found (if none of the parents have their levels set, the root logger's level is returned).

Rejected Alternatives

- Changing log levels in a single node of an application will affect other nodes in the cluster (for example, changing log level of a class in one Connect worker will update levels in all workers in a Connect cluster) and this new level will be persisted across node restarts. This is beyond the scope of this proposal.
- Rest extensions were rejected, as Kafka doesn't offer a REST server and we are aiming for a consistent experience across Kafka and Connect in this proposal.