**Managing/Monitoring Tolls for Apache Kafka**

**Cruise Control**

The Cruise Control project is an open source tool to help monitor and manage large-scale Kafka clusters. Out of the box it enables you to track resource utilization for brokers, topics, and partitions, query cluster state, to view the status of partitions, to monitor server capacity (i.e. CPU, network IO, etc.), message traffic distribution, add and remove brokers, rebalance your cluster, and so on. Cruise Control is used within LinkedIn to manage almost 3000 Kafka brokers.

Cruise Control has a separate front-end component project to help visualize Kafka cluster state as monitored by Cruise Control itself. Cruise Control Front End (CCFE) is implemented as a single-page web application, either deployed with Cruise Control or an existing web server installation. Details about the clusters to be managed are made available via a configuration file.

One of the views provided gives an overview of the configured Kafka cluster status, including broker count, leader partitions, replicas, throughput data, and status, such as out-of-sync replicas.

**Burrow**

Burrow is an open source monitoring tool to track consumer lag in Apache Kafka clusters. It’s designed to monitor every consumer group that is committing offsets to either Kafka or Zookeeper, and to monitor every topic and partition consumed by those groups. Burrow specifically does not monitor MaxLag, and has its reasons for it (as do other tools such as Sematext Kafka monitoring agent) due to limitations around when its value is useful.

Burrow is written in Go, so you’ll need to download, install and set up Go separately. After that, you can download and install Burrow using the Go commands.

To visualize Kafka cluster data as gathered by Burrow, there are open source projects available, such as the browser-based BurrowUI and burrow-dashboard, the command-line UI tool burrow-client, and various plug-ins to other tools.

**Filebeat**

Filebeat is a tool from Elastic that eases the pain in collecting scores of files distributed across a multitude of servers (and their respective VMs and containers). It is one of several good Logstash alternatives. The Kafka module for Filebeat collects and parses logs created by running Kafka instances, and provides a dashboard to visualize the log data. To use it, begin by downloading and installing Filebeat.

Configuring Filebeat requires you to add options to the output section of the tool’s filebeat.yml file.

You define the brokers and topics, for example, in your Kafka cluster, along with many other options you can find in the documentation, such as partition strategy, authentication, and the topic used to produce log events.

**Kafka Monitor**

The goal of the Kafka Monitor framework is to make it as easy as possible to develop and execute long-running Kafka-specific system tests in real clusters and monitor application performance. It helps you execute long-running tests in a Kafka cluster, and works with Kafka’s existing system tests by capturing issues that can occur after running for an extended period of time. Kafka Monitor allows you to monitor a cluster using end-to-end pipelines to obtain vital statistics such as end-to-end latency, service availability and message loss rate. For example, to start Kafka Monitor and begin monitoring a cluster, use the script where you add the parameters specific to your cluster.

Each class must implement the com.linkedin.kmf.services.Test Java interface, and each service class implements the com.linkedin.kmf.services.Service interface. The key for each test and service in the JSON map identifies it in the log or JMX metrics.