KIP-10 - Running Producer, Consumers and Brokers on Mesos

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *here*

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

Motivation

To support launching, configuring and managing: producers, consumers and brokers via Mesos.

Public Interfaces

- *The network protocol and api behavior*

Proposed Changes

Goal

The goal of this project is to provide mesos framework that allows to deploy and manage cluster(s) of Kafka brokers. User is provided with a CLI to control broker configuration of running scheduler.

Overview

Following components are involved in the implementation:

blocked URL

Master daemon and slave daemon are part of mesos distribution. Brokers - are kafka servers.

Scheduler and executor are components that should be developed.

Scheduler

Scheduler is a process that registers itself on master daemon and manages brokers by using executors.

Main activity of the scheduler is to make the “required brokers run on required slaves”. Scheduler maintains internally a cluster configuration. Cluster configuration specifies following:

- number of broker to run;
- broker configuration;
- constraints to clarify which broker to run on which slave;

Sample cluster configuration in json:

```
{
  "brokers": [{
      "id": "0..1",
      "constraints": {
          "cpus": 0.5,
          "mem": 128,
          "nodeAttr": "value"
      }
  }, {
      "id": "2",
      "constraints": {
          "host": "slave0",
          "cpus": 1,
          "mem": 256
      },
      "configuration": {
          "num.io.threads": 10
      }
  }]
}
```

Scheduler is started for the first time with no cluster. Cluster is configured later using CLI. Scheduler persists cluster state in file to be able to recover after restarting.

Executor

Executor is a process spawned by the slave node as requested by scheduler. Executor starts and manages Kafka broker.

Executor is packaged into a jar and download from scheduler by http.

Admin Server

The Framework must handle incoming administrative requests (create topic, add broker etc) as an ordinary broker. Thus the separate component has to manage io communications reusing Kafka Wire Protocol (including SocketServer, NetworkClient etc).

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
- *If we are changing behavior how will we phase out the older behavior?*
- *If we need special migration tools, describe them here.*
- *When will we remove the existing behavior?*

Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.