Dynamic Topic Config

Currently we maintain per-topic configuration in our server.properties configuration file. This unfortunately requires bouncing the server every time you make a change. This wiki is a proposal for making some of these configurations dynamic.

Scope

The proposed scope is just for per-topic configurations. One could argue that we should do this globally for all configuration. I think that is not wise. The reason is because we currently assume configurations are immutable and pass them around in plain scala variables. This immutability is really nice. Many things cannot be changed dynamically (i.e. socket and other i/o buffer sizes) and for other things making them dynamic is just really hard.

I would argue that having server-level defaults be statically configured is not a major problem, as these change rarely. Furthermore configuration management systems that maintain versions, track changes, handle permissions and notifications only work with text files so moving away from this is not necessarily a good thing.

However maintaining topic-level settings in this way is a huge pain. These are set potentially every time you add a topic, and with hundreds of topics and users there are lots of changes. So having these all in a giant properties file on every server and bouncing each time is not a good solution. This proposal is just to move topic-level configuration out of the main server configuration.

Per-Topic Settings

The current pattern is that we maintain default settings and sometimes have per-topic overrides. Here are the relevant settings:

Default Setting	Per-Topic Setting	Notes
log.segment.bytes	log.segment.bytes.per.topic	
log.roll.hours	log.roll.hours.per.topic	
log.retention.hours	log.retention.hours.per.topic	
log.retention.bytes	log.retention.bytes.per.topic	
log.cleanup.policy	topic.log.cleanup.policy.per.topic	in KAFKA-631
log.cleaner.min.cleanable.ratio		in KAFKA-631
log.index.interval.bytes		
log.flush.interval.messages		
log.flush.interval.ms	log.flush.interval.ms.per.topic	
log.compression.type		proposed

Proposal

The proposed approach is that we no longer have the "per.topic" version of configurations. Instead the server will be configured with a default value for each of these settings. When a topic is created the user can either specify the value for each setting, or, if they don't they will inherit the default. In this proposal each topic will have a complete copy of the configuration with any missing values taking whatever was the default at the time the topic was created.

Already in KAFKA-631 Log.scala has been changed so that it takes a single LogConfig argument that holds all configurations. As part of this proposal we will add a new setter for updating this config. Log has been changed so that it no longer maintains a local copy of any of these values, so swapping in a new config object

The configuration itself will be maintained in zookeeper. Currently I believe zookeeper has a structure like the following:

```
get /brokers/topics/log-cleaner-test-1485952437-0  /brokers/
  topics/
    my-topic => {"0":["0", "1"]}
    your-topic => {...}
{ "0": ["0"] }
```

This structure would be changed to something like the following:

```
/brokers/
  topics/
    my-topic/
      replicas => {0:[0,1], ...}
      config => {"log.retention.bytes": 12345,...}
```

(not sure if this is the best layout...?)

All nodes would watch these config nodes and when one changed all brokers would update the appropriate in-memory log instances with the new config.

Creating, altering, and deleting topics

Currently we have a CreateTopicCommand object that can be used to create topics. There is no corresponding delete. If this change were made to config we would also need the ability to change configs too.

I propose adding a new APIs to the protocol:

```
ModifyTopicRequest => Operation TopicName [PropName PropValue]
Operation => "CREATE" | "DELETE" | "ALTER"
TopicName => String
PropName => String
PropValue => String


ModifyTopicResponse => ErrorCode
```

Note that there is no attempt to encode the properties as fields in the protocol since it is assumed these may evolve quickly, instead they are just key-value pairs.

The existing tool would just call this API. The existing logic would move into KafkaApis.

Open Questions

1. I am not so sure about the zookeeper layout, need help thinking that through...
2. One nuance of config is how defaults are handled. One approach would be to snapshot the full set of defaults into zookeeper at topic creation time for any setting not specified. The other approach would be to only save in zookeeper what the user gives and dynamically inherit the defaults. The difference is that in the later case it is easier to make a global config change and have it apply to all topics that haven't specified otherwise. This would also result in less data stored in zookeeper.
3. Should we maybe just move all config into zookeeper?