KIP-574: CLI Dynamic Configuration with file input

Status

Current state: *Accepted*

Discussion thread: Discussion, Voting

JIRA: ⚠ Unable to render Jira issues macro, execution error.

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

Motivation

We'd like to support more complex configurations for some dynamically configurable options. The best way to describe these configurations is with a structured value like JSON or a nested list. It's possible to specify these configurations in `server.properties`, and to configure them in code with an AdminClient, but not using the CLI tools.

`kafka-configs.sh` currently makes it easy to set simple dynamic configs, but doesn't make it possible to set structured values more complex than a list. The current supported input for the --add-config option is:

```
Key Value pairs of configs to add. Square brackets can be used to group values which contain commas: 'k1=v1,k2=[v1,v2,v2],k3=v3'.
```

The implementation makes it impossible to set a value that contains commas (`,`) and square brackets (`[]`) together and rules out structured formats like JSON.

Proposed Changes

We will add an option to `kafka-configs.sh` (ConfigCommand.scala) to accept a properties file and add the properties from the file.

Example usage with file

```
bin/kafka-configs.sh --bootstrap-server localhost:9092 \
  --entity-type brokers --entity-default \
  --alter --add-config-file new.properties
```

Public Interfaces

We will add the `--add-config-file` option to `kafka-configs.sh`. It will be mutually exclusive with `--add-config`.

Compatibility, Deprecation, and Migration Plan

Since we're adding a new option to the command, existing use cases will be unaffected.

Rejected Alternatives

Allow specification of complex configs using `--add-config`

It would be possible to make the existing `--add-config` option do more sophisticated parsing of its input, for example to do brace matching. That would allow us to support things like JSON and nested lists:

```
bin/kafka-configs.sh --bootstrap-server localhost:9092 \
  --entity-type brokers --entity-default \
  --alter --add-config 'k1={"key1":"val1","key2":"val2"},k2=[[1,2],[3,4]]'
```

However, there are several downsides to this. It quickly becomes unwieldy as the values being set become more complex. Shell escaping of various special characters becomes an issue. This also presupposes that the formats we want to support will have balanced braces.

Use multiple `--add-config` options, each with a single argument

We could remove the ambiguity of commas and square brackets by removing the current logic about splitting the configurations into key/value pairs. Setting multiple configs could be achieved by multiple calls to `kafka-configs.sh` or by allowing `--add-config` to be specified multiple times.

```
bin/kafka-configs.sh --bootstrap-server localhost:9092 \
  --entity-type brokers --entity-default \
  --alter \
  --add-config k1=v1 \
  --add-config k2=v1,v2,v3 \
  --add-config k3=v3 \
  --add-config 'k4={"key1": "val1", "key2": "val2"}' \
  --add-config 'k5=[[1,2],[3,4]]'
```

However this would break existing multi-key uses of `kafka-configs.sh`.

Allow configuration from STDIN as well as from a file

It might be nice to enable input from standard input, to enable pipe/grep workflows. However, it's not clear that such workflows would be common. A script that would use such a workflow can send the output of the pipe to a temporary file and load that instead. Since the rest of the commands in the Kafka CLI don't support specifying STDIN, we won't support that here either, for consistency. STDIN support across the entire toolset can be added as a separate KIP if the need arises.

Input from STDIN can be used instead of a file on disk, to do this use a hyphen instead of a file path:

Example usage with STDIN

```
grep 'complex.property' server.properties | bin/kafka-configs.sh --bootstrap-server localhost:9092 \
  --entity-type brokers --entity-default \
  --alter --add-config-file -
```

Allow `--add-config` and `--add-config-file` at the same time

Instead of specifying that `--add-config` and `--add-config-file` are mutually exclusive, we could accept both. This creates ambiguity about what happens if a configuration is specified in both places. We could specify how this is resolved (eg. "values supplied in --add-config take precedence over values supplied in the file"), but this seems more complex than is warranted. Users can call `kafka-configs.sh` multiple times if they want to add configs from multiple sources.

There is no similar ambiguity with `--delete-config` and `--delete-config-file`, since there's no value to conflict.

Include a `--delete-config-file` option

Some systems like Kubernetes allow the cleanup of resources by deleting with the same config that was used to create the resources. This is not an exactly parallel situation, because deleting a configuration property doesn't necessarily put the system in the same state that it was in before the addition.

So that the add and delete usages are parallel, we will also add an option to accept a properties file and delete the properties from the file. Any property defined in the file will be deleted, regardless of what value is defined in the file.

Example usage with file

```
bin/kafka-configs.sh --bootstrap-server localhost:9092 \
  --entity-type brokers --entity-default \
  --alter --delete-config-file old.properties
```

Input from STDIN can be used instead of a file on disk, to do this use a hyphen instead of a file path:

Example usage with STDIN

```
grep 'complex.property' server.properties | bin/kafka-configs.sh --bootstrap-server localhost:9092 \
  --entity-type brokers --entity-default \
  --alter --delete-config-file -
```

...

We will add the `--delete-config-file` option to `kafka-configs.sh`. It will not be mutually exclusive with `--delete-config` because there is no ambiguity about precedence.

Use a format other than a properties file for the configs

Since dynamic configurations are stored in ZooKeeper in JSON format, it might make sense to accept a JSON file in that same format. This seems like it is tying the implementation details too closely to the interface. Users already use properties files to specify configurations. Introducing a different format would add complexity.