KIP-603: Change `ConfigEntry` value to use optional fields

Status

Current state: Draft

Discussion thread: *here [Change the link from the KIP proposal email archive to your own email thread]*

JIRA: *here [Change the link from KAFKA-1 to your own ticket]*

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

Motivation

In the `ConfigEntry` class, the `value` field can be `null`. However, the users can assume it can't be and encounter NullPointerException since the type is not explicit. Thus we want to change its type to `Optional<String>` to make it explicit that it might not be set.

Furthermore, in our implementation here the `AlterConfigsRequest.ConfigEntry` value is requires not be null, however there is nothing preventing `configEntry.value()` to not returning null. Similar cases can also be found in here.

Considering the following configurations:

1. **follower.replication.throttled.replicas**
2. **leader.replication.throttled.replicas**
3. **client.id**

All of the above contain default empty string values and there is nothing prevent them to be set to null. Having read the documentation the caller might easily think that the value of the above configuration can't be null given it's initial value is empty string. Consequently, they forget the null check and hit NullPointerException when these configuration are indeed `null`. The users can write code like `leaderConfig.value().length() == 0` which throws a NullPointerException in execution.

Public Interfaces

We plan to add a new method `valueOptional()` and mark the `value()` method as deprecated.

```
public Optional<String> valueOptional() {
        return value;
}
```

Proposed Changes

We plan to change the type of the value field to be an `Optional<String>` which requires us to also change the getter, `equals` and `hashCode` methods.

```
private final String value; => private final Optional<String> value;



@Override
public boolean equals(Object o) {
    if (this == o)
        return true;
    if (o == null || getClass() != o.getClass())
        return false;

    ConfigEntry that = (ConfigEntry) o;

    return this.name.equals(that.name) &&
            this.value.equals(that.value) &&               // CHANGED
            this.isSensitive == that.isSensitive &&
            this.isReadOnly == that.isReadOnly &&
            this.source == that.source &&
            Objects.equals(this.synonyms, that.synonyms);
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + name.hashCode();
    result = prime * result + value.hashCode();            // CHANGED
    result = prime * result + (isSensitive ? 1 : 0);
    result = prime * result + (isReadOnly ? 1 : 0);
    result = prime * result + source.hashCode();
    result = prime * result + synonyms.hashCode();
    return result;
}
```

Compatibility, Deprecation, and Migration Plan

Various callers that uses the constructor and getter will need to be changed to account for the change in type.

Rejected Alternatives

We have also consider using a default sentinel value to represent value not set, e.g. "". However semantically, null and empty string is not always equivalent. For example we can have a configuration of bootstrap.servers which has the default value of "" and also with another configuration ssl.Key.Password with a default value of null.