KIP-754: Make Scala case class's final

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: Under Discussion

Discussion thread: here

JIRA: KAFKA-12913

Motivation

In Scala its considered best practice that when you use case class's you create them as final rather than non final (if possible). There are many reasons behind this

- case class's are are representation of ADT's which is a functional structure designed to only hold data. case class's do not typically define any behavior, they may define methods but this is typically only calculations based on existing fields inside a case class
    - If you want classic style OOP where you want inheritance then you use a standard class instead
- case class's already define a hashcode/equals/unapply who's definition becomes misleading if its overridden in a subclass. Critically code that uses a defined case class relies on the fact that the hashcode/equals/unapply definition for that case class is not arbitrarily override by a user
- Using final guarantees optimal performance for the case class
    - It is true that the JDK classes are "effectively" final if no one subclasses the class during runtime however using final guarantees this performance contract.

See https://nrinaudo.github.io/scala-best-practices/tricky_behaviours/final_case_classes.html for a more in detail explanation of what can go wrong if you don't mark case class as final

Public Interfaces

- There is a source breaking change for Kafka's Scala core if the user happened to subclass a case class. There are no binary incompatible changes

Proposed Changes

Here are the following changes

- Any public top level case class is made final
- For case classes that are defined within a class
    - If the case class is private then nothing is done
    - If the case class is public but should have been private (i.e. no public methods/fields reference that case class) then it is made private. If said case class was also defined within another class it is moved to top level
    - If the case class is public then it is made top level and made final
- Style guide updated to mention that case class should always be defined as final

Compatibility, Deprecation, and Migration Plan

- The only impact for users is if they would have made a subclass of a specific case class within Kafka core and used it within Kafka code.
    - As mentioned before, if a user actually did this this then they were likely already doing something wrong especially if they override equals /hashcode
    - There is one valid case of subclassing which could be adding additional methods to an already existing case class (without any other behavioral changes). In this case a user can simply use implicit final class (aka monkey patching/extension methods) to just add extra methods to an already existing class.
        - This technique is well documented and supported within Scala
        - You cannot override any existing methods (i.e. hashcode/equals/unapply) using this technique, you can only add methods. This makes it much safer
- Internally in Kafka no real changes were needed apart from removing a subclassing of a case class in a single test (this was only done to adjust the pretty print of some case class if the test failed so its not critical)

- Since 3.0.0 is a major release, doing this within 3.0.0 is ideal

Rejected Alternatives

None that I am aware of