KIP-531: Drop support for Scala 2.11 in Kafka 2.5

Status

Current state: *Adopted*

Discussion thread: *here*

JIRA: ⚠ Unable to render Jira issues macro, execution error. , ⚠ 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 currently build Kafka for 3 Scala versions: 2.11, 2.12 and the recently released 2.13. Since we have to compile and run the tests on each supported version, there is a non-trivial cost from a development and testing perspective.

Scala 2.11.0 was released in April 2014 and support for 2.11.x ended in Nov 2017 (it will be more than 2 years by the time Kafka 2.5 is released). Scala 2.12.0 was released in Nov 2016 and Scala 2.13.0 was released in June 2019. Given this, it's time to drop support for Scala 2.11 so that we keep the testing matrix manageable (a recent kafka-trunk-jdk8 took nearly 10 hours, it builds and runs the unit and integration tests with the 3 Scala versions). In addition, Scala 2.12 and later have improved interoperability with Java 8 functional interfaces (first introduced in Scala 2.12). More specifically, lambdas in Scala 2.12 can be used with Java 8 functional interfaces in the same way as Java 8 code.

In our downloads page, we have recommended the Kafka binaries built with Scala 2.12 since Kafka 2.1.0. We switched to Scala 2.12 as the default Scala version for the source tarball, build and system tests in Kafka 2.2.0.

Public Interfaces

None.

Proposed Changes

- Deprecate Scala 2.11 build in Kafka 2.4.
- Update `build.gradle` to remove all tasks for Scala 2.11 and remove code in `build.gradle` and `dependencies.gradle` that is conditional on Scala 2.11.
- Deprecate `org.apache.kafka.streams.scala.kstream.Suppressed` in kafka-streams-scala as it is only needed for Scala 2.11 support.
- Update Readme not to mention Scala 2.11 as a supported version.
- Remove any other workarounds needed by Scala 2.11 (e.g. a number of methods in `CoreUtils` can be removed).
- Introduce a shim `core` module that targets the default scala version. This useful for applications that do not require a specific Scala version. Java applications that shade Scala dependencies or Java applications that have a single Scala dependency would fall under this category. We will target Scala 2.13 in the initial version of this module.

Compatibility, Deprecation, and Migration Plan

Users who have not and cannot upgrade to Scala 2.12/2.13 and are either using kafka-streams-scala or the kafka core module will have to stick with Kafka 2.4.x until they can upgrade to one of the more recent Scala versions. Note that the classes in the kafka core module are not public API although many projects use some of the classes for ZK access and integration tests.

Rejected Alternatives

1. Drop support for Scala 2.11 in Kafka 3.0 instead. There is no date set for Kafka 3.0 and the cost seems to outweigh the benefits.

Potential Future Work

1. Remove different scala versions as an option for the `core` jar in Kafka 3.0. This would work best if combined with a Java API for broker integration testing.