aiven

Preparing Apache Kafka for Scala 3

A Story in 3 acts

A Story in 3 acts

1. Origin

A Story in 3 acts

1. Origin
2. Challenges during the PoC

A Story in 3 acts

1. Origin
2. Challenges during the PoC
3. What should we improve?

Orig

Who?

Josep Prat

Might have seen me at the Akka corner some years ago.

Working at Aiven.

Director of the Open Source Program Office.

Why?

Aiven offers Managed Open Source Data Infrastructure as a Service.

Why?

Aiven offers Managed Open Source Data Infrastructure as a Service.

kafka

Why?

Aiven offers Managed Open Source Data Infrastructure as a Service.

kafka

Among many others…

Why?

Aiven offers Managed Open Source Data Infrastructure as a Service.

kafka

Among many others…

Apache Kafka,is a trademark of their respective owners.

Why?

The Open Source Program Office employs people to work full time on Open Source projects.

What?

Apache Kafka is:

What?

Apache Kafka is:

- Distributed event streaming platform

What?

Apache Kafka is:

- Distributed event streaming platform
- Extremely scalable

What?

Apache Kafka is:

- Distributed event streaming platform
- Extremely scalable
- High throughput

What?

Apache Kafka is:

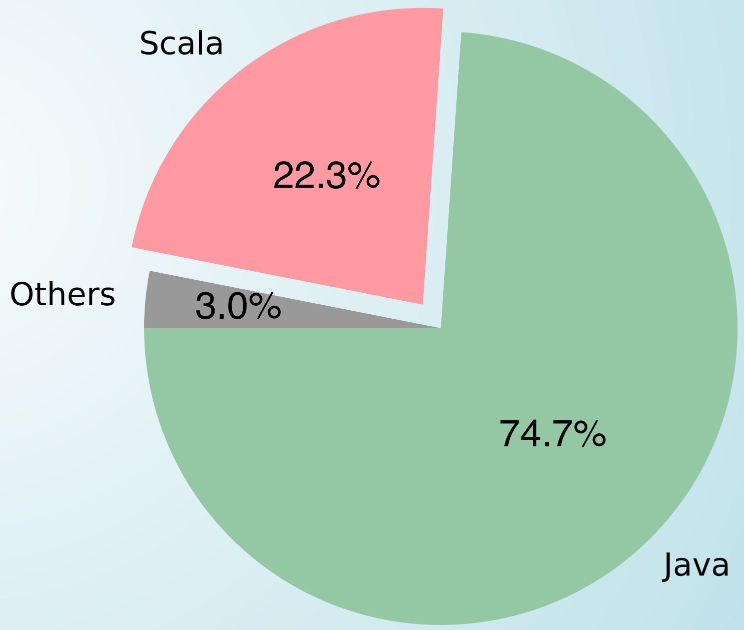
- Distributed event streaming platform
- Extremely scalable
- High throughput
- High availability

What?

Some internal details on Apache Kafka:

What?

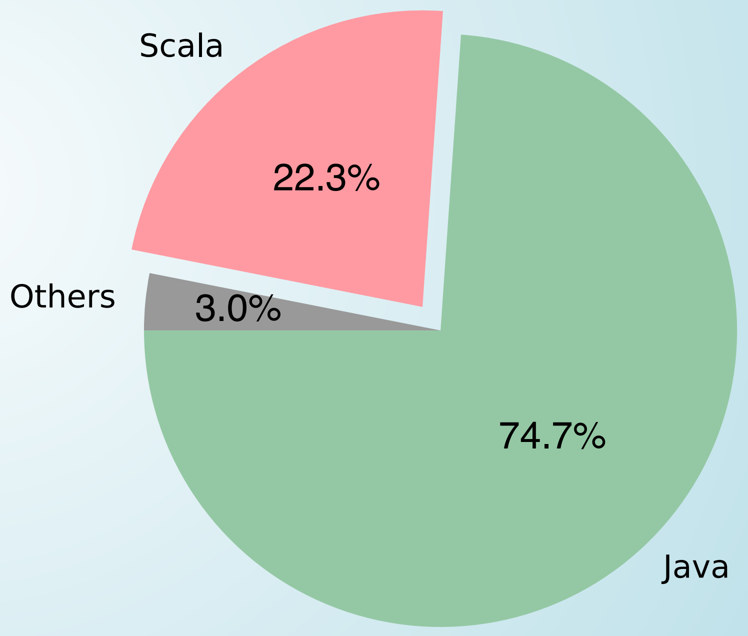
Some internal details on Apache Kafka:



What?

Some internal details on Apache Kafka:

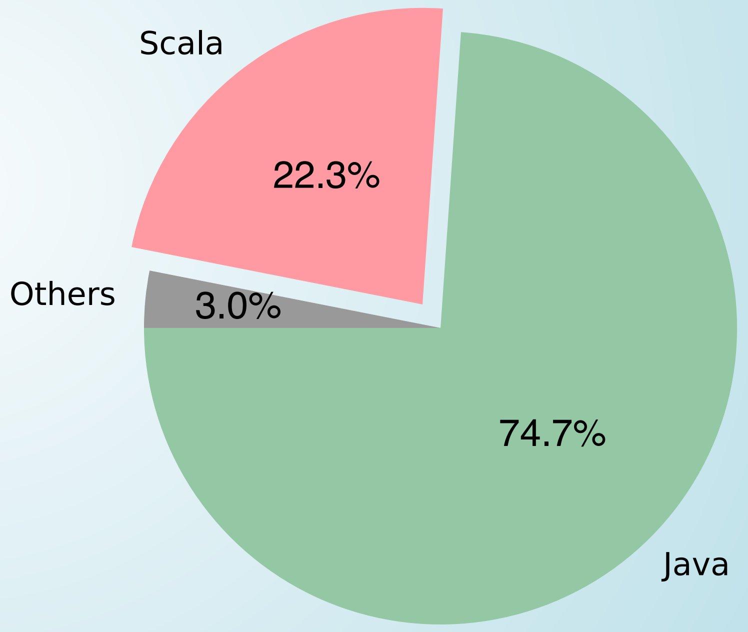
- Compiles against Scala 2.12 and 2.13



What?

Some internal details on Apache Kafka:

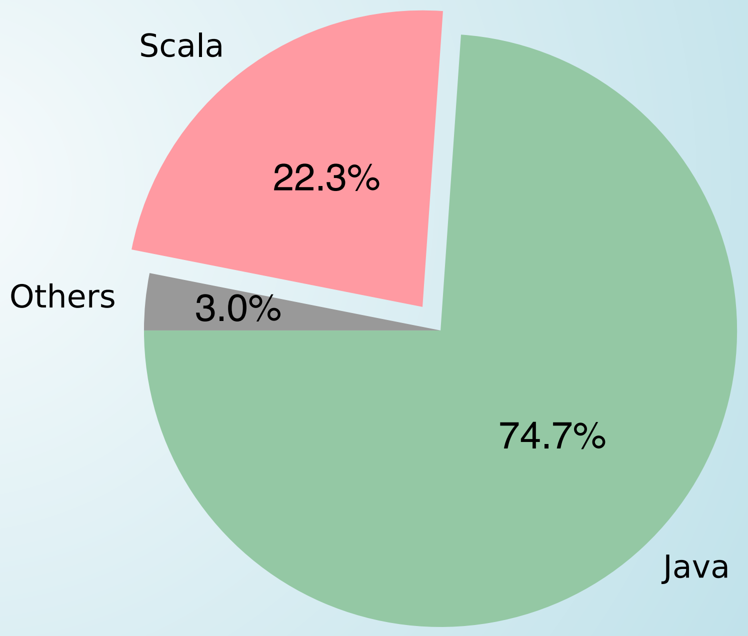
- Compiles against Scala 2.12 and 2.13
- Uses Gradle as a build tool



What?

Some internal details on Apache Kafka:

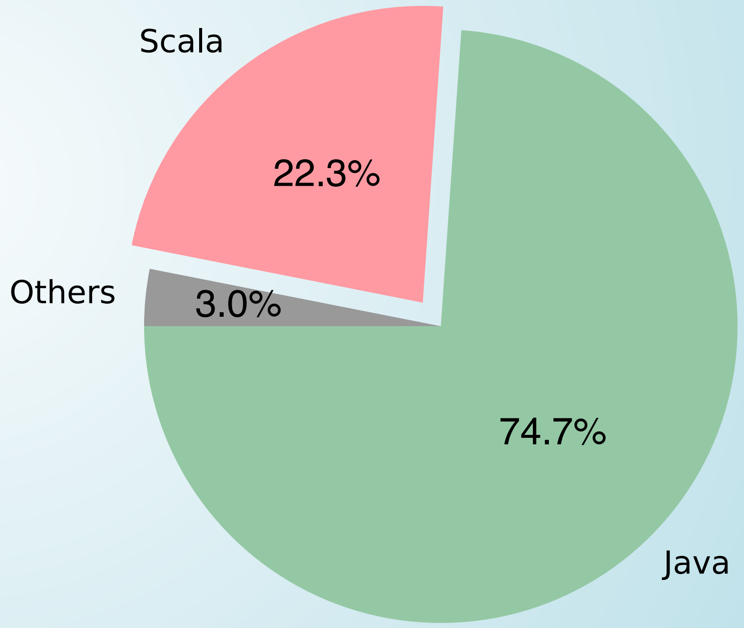
- Compiles against Scala 2.12 and 2.13
- Uses Gradle as a build tool
- Kafka Core is mostly written in Scala



What?

Some internal details on Apache Kafka:

- Compiles against Scala 2.12 and 2.13
- Uses Gradle as a build tool
- Kafka Core is mostly written in Scala
- No macros nor typeclasses



What?

Seemed like a walk in the park…

What?

Seemed like a walk in the park…

…or not!

Challenges durin

Several moons ago...

Decided to upgrade Apache Kafka to Scala 3, and started a PR...

1. Gradle

How complicated could it be to compile using the Scala 3 compiler instead?

Gradle

One might think:

Gradle

One might think:

- Change `dependencies.gradle`

Gradle

One might think:

- Change **`dependencies.gradle`**
- Change **`build.gradle`**

Gradle

One might think:

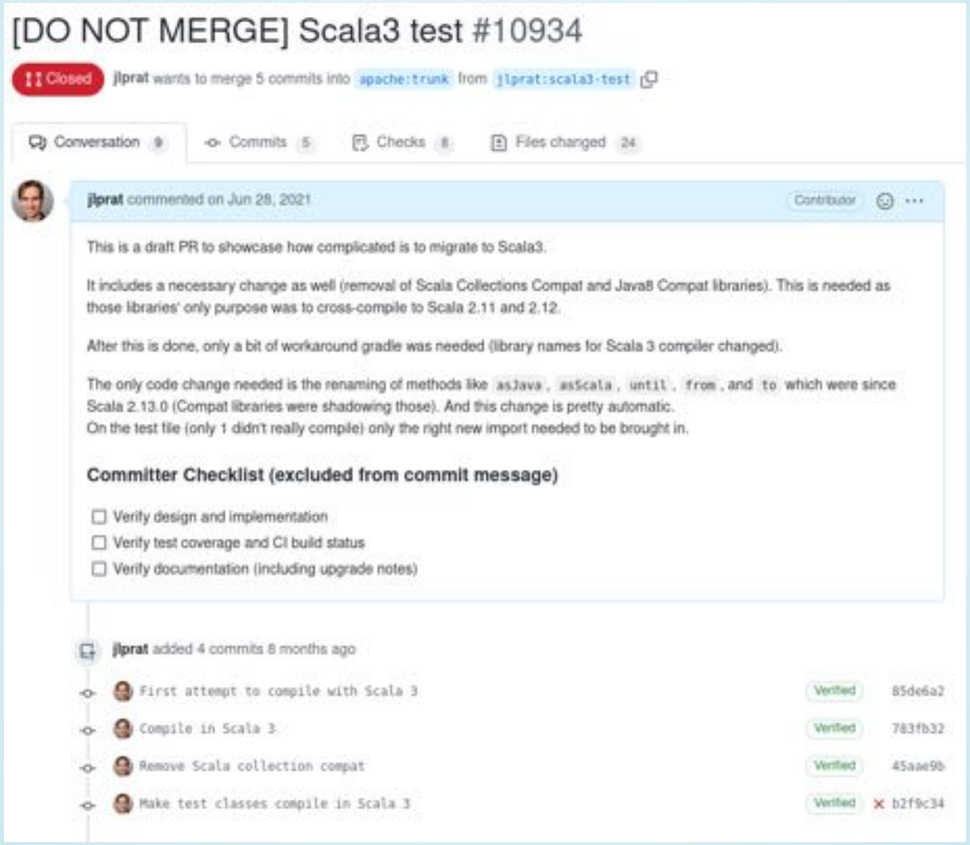
- Change `dependencies.gradle`
- Change `build.gradle`
- Do some magic with version names

Gradle

One might think:

- Change **`dependencies.gradle`**
- Change **`build.gradle`**
- Do some magic with version names
- Profit

Easy, right?



[DO NOT MERGE] Scala3 test #10934

Closed jlprat wants to merge 5 commits into apache:trunk from jlprat:scala3-test

Conversation 9 · Commits 5 · Checks 8 · Files changed 24

jlprat commented on Jun 28, 2021 · Contributor

This is a draft PR to showcase how complicated is to migrate to Scala3.

It includes a necessary change as well (removal of Scala Collections Compat and Java8 Compat libraries). This is needed as those libraries' only purpose was to cross-compile to Scala 2.11 and 2.12.

After this is done, only a bit of workaround gradle was needed (library names for Scala 3 compiler changed).

The only code change needed is the renaming of methods like `asJava`, `asScala`, `until`, `from`, and `to` which were since Scala 2.13.0 (Compat libraries were shadowing those). And this change is pretty automatic.
On the test file (only 1 didn't really compile) only the right new import needed to be brought in.

Committer Checklist (excluded from commit message)

- [ ] Verify design and implementation
- [ ] Verify test coverage and CI build status
- [ ] Verify documentation (including upgrade notes)

jlprat added 4 commits 8 months ago

```
First attempt to compile with Scala 3      Verified    85de6a2
Compile in Scala 3                         Verified    783fb32
Remove Scala collection compat             Verified    45aae9b
Make test classes compile in Scala 3       Verified  × b2f9c34
```

I was wrong…

Big kudos to Tomasz Godzik!

For singlehandedly introducing support for Scala 3!
gradle/gradle/pull/18001

Status: ✓

Issue under gradle/gradle/16527

From version: 7.3.0

2. Unit SAM and overloads

```scala
object Reproducer {
  // assertThrows has 3 overloads, 2 with 3 parameters and
  // 1 with only 2 parameters.

  // This overload is taking a subclass of `Throwable`,
  // and an `Executable` which is a parameterless SAM
  // returning void
  assertThrows(classOf[IllegalArgumentException],
               () => 3)
  // Compiles

  // This overload is taking a subclass of `Throwable`,
  // an `Executable` which is a parameterless SAM
  // returning void, and a `String`
  assertThrows(classOf[IllegalArgumentException],
               () => 3
```

2. Unit SAM and overloads

```scala
                                   ):Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   message: String
                                   ): Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   supplier: Supplier[String]
                                   ): Unit=???

  @FunctionalInterface
  trait Executable {
    @throws[Throwable]
    def execute(): Unit
  }
}
```

2. Unit SAM and overloads

```scala
                                   ):Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   message: String
                                   ): Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   supplier: Supplier[String]
                                   ): Unit=???

  @FunctionalInterface
  trait Executable {
    @throws[Throwable]
    def execute(): Unit
  }
}
```

2. Unit SAM and overloads

```scala
object Reproducer {
  // assertThrows has 3 overloads, 2 with 3 parameters and
  // 1 with only 2 parameters.

  // This overload is taking a subclass of `Throwable`,
  // and an `Executable` which is a parameterless SAM
  // returning void
  assertThrows(classOf[IllegalArgumentException],
               () => 3)
  // Compiles

  // This overload is taking a subclass of `Throwable`,
  // an `Executable` which is a parameterless SAM
  // returning void, and a `String`
  assertThrows(classOf[IllegalArgumentException],
               () => 3
```

2. Unit SAM and overloads

```
// This overload is taking a subclass of `Throwable`,
// and an `Executable` which is a parameterless SAM
// returning void
assertThrows(classOf[IllegalArgumentException],
             () => 3)
// Compiles

// This overload is taking a subclass of `Throwable`,
// an `Executable` which is a parameterless SAM
// returning void, and a `String`
assertThrows(classOf[IllegalArgumentException],
             () => 3,
             "This is a message")
// Doesn't compile in 3.0

```

2. Unit SAM and overloads

```scala
    // This overload is taking a subclass of `Throwable`,
    // an `Executable` which is a parameterless SAM
    // returning void, and a `String`
    assertThrows(classOf[IllegalArgumentException],
                 () => 3,
                 "This is a message")
    // Doesn't compile in 3.0

    //This overload is taking a subclass of `Throwable`,
    // an `Executable` which is a parameterless SAM
    // returning void, and a `Supplier` returning `String`
    assertThrows(classOf[IllegalArgumentException],
                 () => 3,
                 () => "This is a message")
    // Doesn't compile in 3.0

```

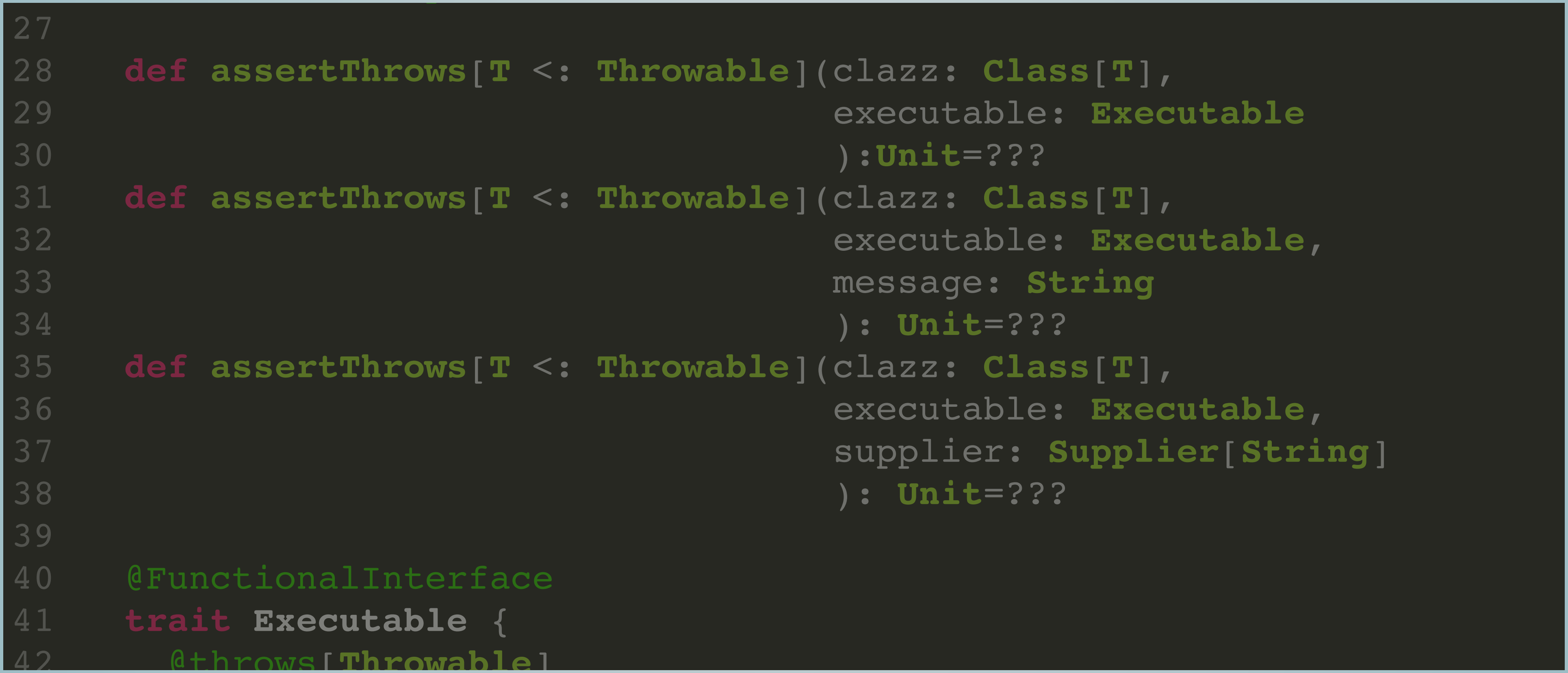
2. Unit SAM and overloads

```scala

    //This overload is taking a subclass of `Throwable`,
    // an `Executable` which is a parameterless SAM
    // returning void, and a `Supplier` returning `String`
    assertThrows(classOf[IllegalArgumentException],
                 () => 3,
                 () => "This is a message")
    // Doesn't compile in 3.0

    def assertThrows[T <: Throwable](clazz: Class[T],
                                     executable: Executable
                                     ):Unit=???
    def assertThrows[T <: Throwable](clazz: Class[T],
                                     executable: Executable,
                                     message: String
                                     ): Unit=???
```

2. Unit SAM and overloads



```

  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable
                                   ):Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   message: String
                                   ): Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   supplier: Supplier[String]
                                   ): Unit=???

  @FunctionalInterface
  trait Executable {
    @throws[Throwable]
```

2. Unit SAM and overloads

```
                                   ):Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   message: String
                                   ): Unit=???
  def assertThrows[T <: Throwable](clazz: Class[T],
                                   executable: Executable,
                                   supplier: Supplier[String]
                                   ): Unit=???

  @FunctionalInterface
  trait Executable {
    @throws[Throwable]
    def execute(): Unit
  }
}
```

Workaround

```scala
object Reproducer {
  assertThrows(classOf[IllegalArgumentException],
               () => 3)

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               "This is a message")

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               () => "This is a message")

}
```

Workaround

```scala
object Reproducer {
  assertThrows(classOf[IllegalArgumentException],
               () => 3)

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               "This is a message")

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               () => "This is a message")

}
```

Workaround

```
object Reproducer {
  assertThrows(classOf[IllegalArgumentException],
               () => 3)

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               "This is a message")

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               () => "This is a message")

}
```

Workaround

```scala
object Reproducer {
  assertThrows(classOf[IllegalArgumentException],
               () => 3)

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               "This is a message")

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               () => "This is a message")

}
```

Workaround

```scala
object Reproducer {
  assertThrows(classOf[IllegalArgumentException],
               () => 3)

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               "This is a message")

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               () => "This is a message")

}
```

Workaround

```scala
object Reproducer {
  assertThrows(classOf[IllegalArgumentException],
               () => 3)

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               "This is a message")

  assertThrows(classOf[IllegalArgumentException],
               () => {3; ()},
               () => "This is a message")

}
```

Status: Not solved ✗

Issue under lampepfl/dotty/issue/13549

It conflicts with existing Scala 3 code making functional code fail.

3. No static forwarder methods in trait companion

```scala
object ObjectTraitPair {
  val Constant: String = "Some Text"
}

// In Scala 2.13 this class bytecode will carry over
// any val and def defined in the object with the same name
// but not in Scala 3.0
trait ObjectTraitPair {
  val method: String = "bye"
}
```

3. No static forwarder methods in trait companion

```scala
object ObjectTraitPair {
  val Constant: String = "Some Text"
}

// In Scala 2.13 this class bytecode will carry over
// any val and def defined in the object with the same name
// but not in Scala 3.0
trait ObjectTraitPair {
  val method: String = "bye"
}
```

3. No static forwarder methods in trait companion

```scala
object ObjectTraitPair {
  val Constant: String = "Some Text"
}

// In Scala 2.13 this class bytecode will carry over
// any val and def defined in the object with the same name
// but not in Scala 3.0
trait ObjectTraitPair {
  val method: String = "bye"
}
```

3. No static forwarder methods in trait companion

```scala
object ObjectTraitPair {
  val Constant: String = "Some Text"
}

// In Scala 2.13 this class bytecode will carry over
// any val and def defined in the object with the same name
// but not in Scala 3.0
trait ObjectTraitPair {
  val method: String = "bye"
}
```

Bytecode discrepancy

Output of **`javap ObjectTraitPair.class`**:

```
public interface ObjectTraitPair {
  public static void $init$(example.ObjectTraitPair);
  public abstract java.lang.String method();
  public abstract void example$ObjectTraitPair$_setter_$method_$eq(java.
}
```

But it should be:

```
public interface ObjectTraitPair {
  public static java.lang.String Constant();
  public abstract void example$ObjectTraitPair$_setter_$method_$eq(jav
  public abstract java.lang.String method();
  public static void $init$(example.ObjectTraitPair);
}
```

Bytecode discrepancy

Output of `javap ObjectTraitPair.class`:

```
public interface ObjectTraitPair {
  public static void $init$(example.ObjectTraitPair);
  public abstract java.lang.String method();
  public abstract void example$ObjectTraitPair$_setter_$method_$eq(java.l
}
```

But it should be:

```
public interface ObjectTraitPair {
  public static java.lang.String Constant();
  public abstract void example$ObjectTraitPair$_setter_$method_$eq(jav
  public abstract java.lang.String method();
  public static void $init$(example.ObjectTraitPair);
}
```

Status: ✓

Issue under lampepfl/dotty/13572

From Scala version: 3.1.0

4. Variable handling in super calls

```scala
class ClassWithLambda(sup: () => Long)
class ClassWithVar(var msg: String)
       extends ClassWithLambda(() => 1)
val _ = new ClassWithVar("foo")
// Throws at runtime!
// java.lang.VerifyError: Bad type on operand stack
```

4. Variable handling in super calls

```scala
class ClassWithLambda(sup: () => Long)
class ClassWithVar(var msg: String)
      extends ClassWithLambda(() => 1)
val _ = new ClassWithVar("foo")
// Throws at runtime!
// java.lang.VerifyError: Bad type on operand stack
```

4. Variable handling in super calls

```scala
class ClassWithLambda(sup: () => Long)
class ClassWithVar(var msg: String)
      extends ClassWithLambda(() => 1)
val _ = new ClassWithVar("foo")
// Throws at runtime!
// java.lang.VerifyError: Bad type on operand stack
```

4. Variable handling in super calls

```
class ClassWithLambda(sup: () => Long)
class ClassWithVar(var msg: String)
      extends ClassWithLambda(() => 1)
val _ = new ClassWithVar("foo")
// Throws at runtime!
// java.lang.VerifyError: Bad type on operand stack
```

But this works!

```scala
class ClassWithLambda(sup: () => Long)
class ClassWithVar(_msg: String)
      extends ClassWithLambda(() => 1) {
  var msg: String = _msg
}
val _ = new ClassWithVar("foo")
```

But this works!

```scala
class ClassWithLambda(sup: () => Long)
class ClassWithVar(_msg: String)
      extends ClassWithLambda(() => 1) {
  var msg: String = _msg
}
val _ = new ClassWithVar("foo")
```

But this works!

```scala
class ClassWithLambda(sup: () => Long)
class ClassWithVar(_msg: String)
      extends ClassWithLambda(() => 1) {
  var msg: String = _msg
}
val _ = new ClassWithVar("foo")
```

Status: ✓

Issue under lampepfl/dotty/13630

From Scala version: 3.1.1

5. Handle Java varargs with parametrized T...

Given this Java class:

```java
public class TypedVarargs<V> {
  public TypedVarargs<V> varArgs(V thing, V... things) {
    return this;
    }
  }
```

And this Scala one:

```scala
val x = new TypedVarargs[Long]()
val y = x.varArgs(1L)
// This throws at runtime:
// java.lang.ClassCastException: [J cannot be cast to [Ljava.lang.Obje
```

5. Handle Java varargs with parametrized T...

Given this Java class:

```java
public class TypedVarargs<V> {
  public TypedVarargs<V> varArgs(V thing, V... things) {
    return this;
    }
  }
```

And this Scala one:

```scala
val x = new TypedVarargs[Long]()
val y = x.varArgs(1L)
// This throws at runtime:
// java.lang.ClassCastException: [J cannot be cast to [Ljava.lang.Obje
```

5. Handle Java varargs with parametrized T...

Given this Java class:

```java
public class TypedVarargs<V> {
  public TypedVarargs<V> varArgs(V thing, V... things) {
    return this;
    }
  }
```

And this Scala one:

```scala
val x = new TypedVarargs[Long]()
val y = x.varArgs(1L)
// This throws at runtime:
// java.lang.ClassCastException: [J cannot be cast to [Ljava.lang.Obje
```

5. Handle Java varargs with parametrized T...

Given this Java class:

```java
public class TypedVarargs<V> {
  public TypedVarargs<V> varArgs(V thing, V... things) {
    return this;
    }
  }
```

And this Scala one:

```scala
val x = new TypedVarargs[Long]()
val y = x.varArgs(1L)
// This throws at runtime:
// java.lang.ClassCastException: [J cannot be cast to [Ljava.lang.Obje
```

Workaround

```scala
val x = new TypedVarargs[java.lang.Long]()
val y = x.varArgs(1L)
```

Workaround

```scala
val x = new TypedVarargs[java.lang.Long]()
val y = x.varArgs(1L)
```

Status: ✓

Issue under lampepfl/dotty/13645

From Scala version: 3.1.1

6. Type erased for by-name parameters

Given this code:

```scala
object ByNameParam {
  def byNameParam(str: => String): Unit = {}
}
```

6. Type erased for by-name parameters

Given this code:

```scala
object ByNameParam {
  def byNameParam(str: => String): Unit = {}
}
```

Output of `javap ByNameParam.class`:

```
public final class ByNameParam {
  public static void byNameParam(scala.Function0);
}
```

But should be:

```
public final class ByNameParam {
  public static void byNameParam(scala.Function0<java.lang.String>);
}
```

Output of `javap ByNameParam.class`:

```
public final class ByNameParam {
  public static void byNameParam(scala.Function0);
}
```

But should be:

```
public final class ByNameParam {
  public static void byNameParam(scala.Function0<java.lang.String>);
}
```

Output of `javap ByNameParam.class`:

```java
public final class ByNameParam {
  public static void byNameParam(scala.Function0);
}
```

But should be:

```java
public final class ByNameParam {
  public static void byNameParam(scala.Function0<java.lang.String>);
}
```

Status: ✓

Issue under lampepfl/dotty/13638

From Scala version: 3.1.2

Summary:

Issue	Status	Since
gradle/gradle/16527	✓	Gradle 7.3.0
lampepfl/dotty/13549	✗	N/A
lampepfl/dotty/13572	✓	Scala 3.1.0
lampepfl/dotty/13630	✓	Scala 3.1.1
lampepfl/dotty/13645	✓	Scala 3.1.1
lampepfl/dotty/13638	✓	Scala 3.1.2

```
<irony>
```

Easy, huh?

```
</irony>
```

But sure this is now all done

But sure this is now all done right?

It looks we need to sit tight

Migration will happen for Apache Kafka 4.0.0 release.

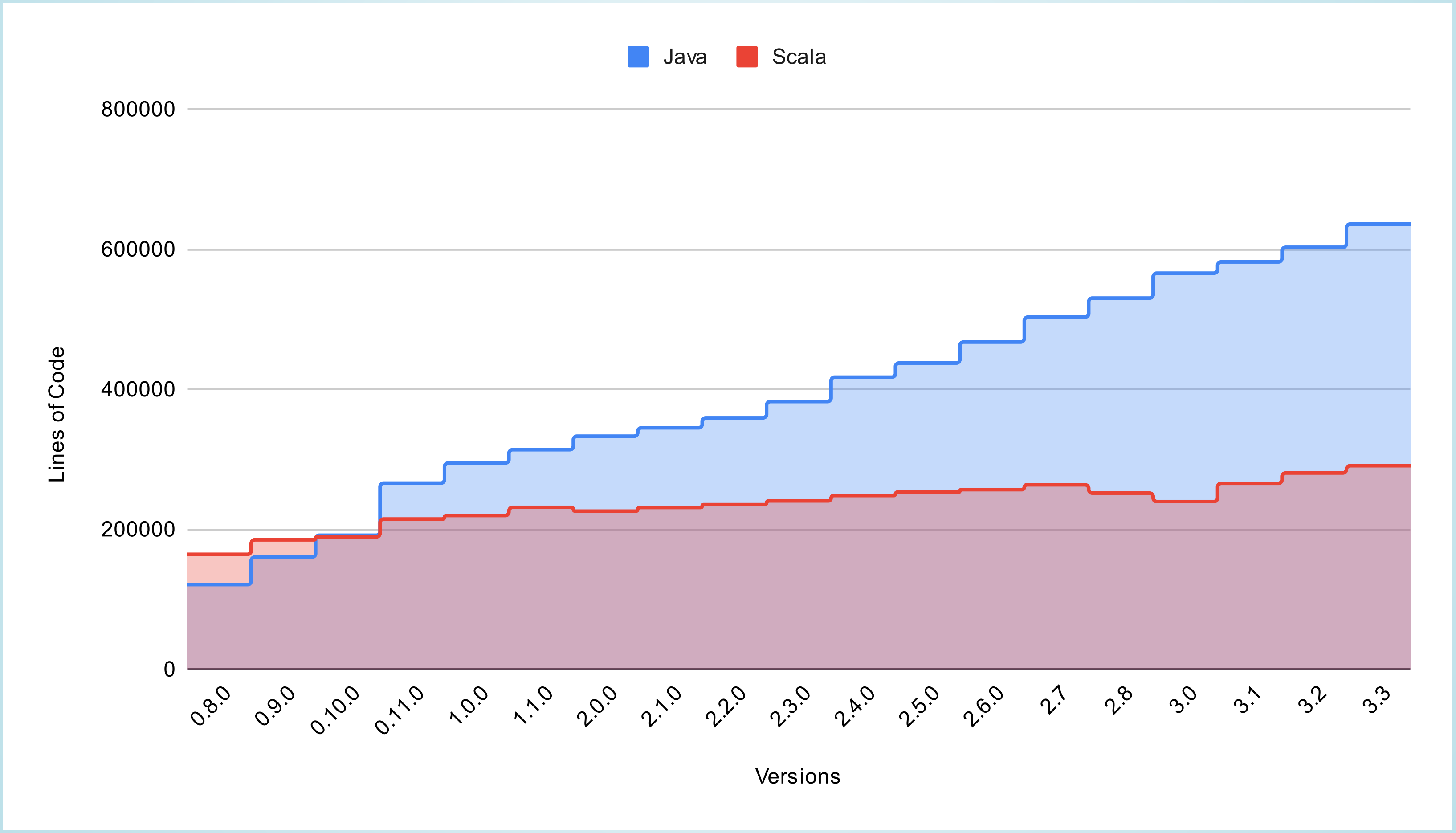
It looks we need to sit tight

Migration will happen for Apache Kafka 4.0.0 release.

And this looks to be still 1 year in the future.

What should we

Fading away?



🟦 Java
🟥 Scala
800000
600000
400000
200000
0
Lines of Code
0.8.0
0.9.0
0.10.0
0.11.0
1.0.0
1.1.0
2.0.0
2.1.0
2.2.0
2.3.0
2.4.0
2.5.0
2.6.0
2.7
2.8
3.0
3.1
3.2
3.3
Versions

Why wasn't it more straightforward?

Java → Scala interoperability improved substantially in 2.12 and 2.13.

Why wasn't it more straightforward?

Java → Scala interoperability improved substantially in 2.12 and 2.13.

Dotty followed kind of a parallel line branching out in 2.11.

Community build

We need more non-fully Scala friendly environment.

Community build

We need more non-fully Scala friendly environment.

To include projects on the fringe of the core of the Scala community.

Mixed Java/Scala projects

We, the Scala community, need to get closer to these projects.

Lack of Scala Understanding

Some projects don't have "in house" Scala experts.

Lack of Scala Understanding

Some projects don't have "in house" Scala experts.

And historically, Scala migrations have been tedious.

How can I help?

Here you have a couple of places where you can help!

How can I help?

Here you have a couple of places where you can help!

- FLIP-265: Deprecate and remove Scala API support: Call for Scala developers!

How can I help?

Here you have a couple of places where you can help!

- FLIP-265: Deprecate and remove Scala API support: Call for Scala developers!
- Bring Apache Kafka closer to Scala 3

Join some Java/Scala OSS Projects!

Further Info:

- Mailing list thread
- [Pull Request] Big proof of concept
- [Pull Request] In between step
- Blog post that originated this talk

Thanks!