To set up an environment for Scala asynchronous programming (using `scala-async` with Futures), follow these steps:

1. Install Scala and sbt (Scala Build Tool)

- Download and install Scala from [scala-lang.org]
- Download and install sbt from [sbt official site]

Verify installation by running in terminal/cmd:

```
scala -version
sbt sbtVersion
```

2. Create a new sbt project

In a new folder, create the following structure:

```
project-root/
 ├── build.sbt
 └── src/
     └── main/
         └── scala/
             └── AsyncExample.scala
```

3. Configure `build.sbt`

Add dependencies and compiler options to enable `scala-async` support:

```
name := "ScalaAsyncExample"

version := "0.1"

scalaVersion := "2.13.8"
```

```
// Add scala-async and scala-reflect (needed for async macro)
libraryDependencies ++= Seq(
  "org.scala-lang.modules" %% "scala-async" % "1.0.1",
  "org.scala-lang" % "scala-reflect" % scalaVersion.value %
Provided
)

// Add async compiler plugin flag
scalacOptions += "-Xasync"
```

4. Write Scala asynchronous code using async/await

Example src/main/scala/AsyncExample.scala:

```
import scala.concurrent.Future
import scala.concurrent.ExecutionContext.Implicits.global
import scala.async.Async.{async, await}

object AsyncExample {
  def slowComputation(): Future[Int] = Future {
    Thread.sleep(1000)
    42
  }

  def main(args: Array[String]): Unit = {
    val result: Future[Int] = async {
      val value = await(slowComputation())
      value * 2
    }

    result.foreach { r =>
      println(s"The result of async computation: $r")
    }

    Thread.sleep(2000) // keep JVM alive to see output
  }
}
```

5. Compile and run

From project root:

```
sbt run
```

You should see:

```
The result of async computation: 84
```

Summary

- Scala async requires adding `"org.scala-lang.modules" %% "scala-async"` and `"scala-reflect"` libraries.
- Enable async support by adding `-Xasync` compiler option.
- Use `async { ... }` blocks and `await(future)` to write asynchronous code in a sequential style.
- Keep the JVM alive while waiting for asynchronous computations to complete, for example, by adding `Thread.sleep` in small test apps.

If you want, I can provide a ready-to-use `build.sbt` file or help you configure popular IDEs like IntelliJ or Eclipse for Scala async programming!