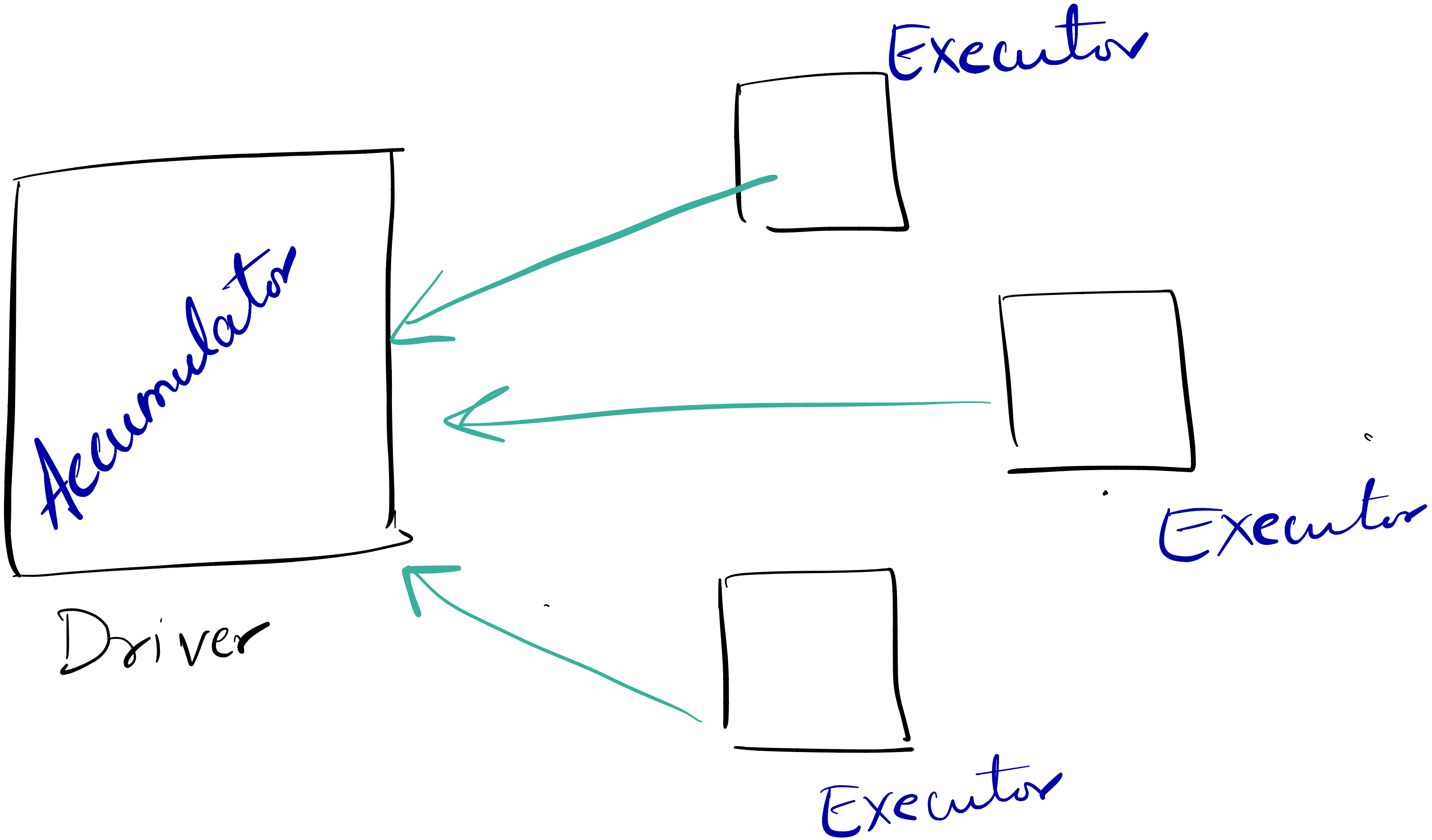
Executor
Accumulator
Executor
Driver
Executor

SparkAccumulatorBasicExample.scala

```scala
/* Demonstrate how accumulator works in Spark.
 *
 * @author Archit Soni
 * @LinkedIn https://www.linkedin.com/in/archit-soni-45b02797/
 */

import org.apache.log4j.Level
import org.apache.log4j.Logger
import org.apache.spark.SparkContext
import scala.tools.cmd.Spec.Accumulator

object SparkAccumulatorBasicExample extends App{

  Logger.getLogger("org").setLevel(Level.ERROR)

  val sc = new SparkContext("local[*]", "app")

  var counter = 0

  val sample_rdd = sc.parallelize(Array(1,2,3,4,5,6,7,8,9,10))

  sample_rdd.foreach(x=> counter+=x)

  println(counter)

  //output - 40 // which is obvisouly wrong.

  // Lets do the same thing via Accumulator
  val accum = sc.longAccumulator("Counter")

  sample_rdd.foreach(x=> accum.add(x))

  println(accum.value)
  // output is 55


  scala.io.StdIn.readLine() // to see the Accumulator variable in Spark UI

}

// Output
// 40
// 55
```

```scala
/* Demonstrate how an immutable variable val can reference mutable variable and show updated values.
 * The fundamentals of Scala says that variable val is immutable and we cant re-assign value of a val
 *
 * @author Archit Soni
 * @LinkedIn https://www.linkedin.com/in/archit-soni-45b02797/
 */

object ImmutableVariableMutableReference extends App {

  // Example of immutable variable with immutable reference

  case class Planets(val galaxy:String, val name: String){

    def getDetails():String = {

      s"Details- The Planet $name is in $galaxy galaxy"

    }

  }

val planetInfo = new Planets("Andromeda", "Kepler-452b")

println(planetInfo)
println(planetInfo.getDetails())
println(planetInfo.galaxy)
println(planetInfo.name)

//planetInfo.galaxy = "Exo"  // this gives me an error for reassignment of Val
println("\n lets see if this following statement works ? \n planetInfo.galaxy = \"Exo\" //this gives me an error for reassignment of Val")

// Example of immutable variable with mutable reference

println("\n \n Example of immutable variable with mutable reference
\n\n" )

  case class MatchInfo(var currentOver:Int, var currentBall: Int){

    def getDetails():String = {
      s"Current Over is $currentOver and the player is playing ball# $currentBall of the over"
    }
}

val scoreKeeper = new MatchInfo(0,1)

println(scoreKeeper.getDetails())

println("\n --- By the magic of time lapse, update the Info --- \n --- current over to 1 and current ball to 2 --- \n \n")
scoreKeeper.currentOver = 1  // current over is of mutable type(var)
scoreKeeper.currentBall = 2  // current ball is of mutable type(var)

println("--- Updated Info ---")
println(scoreKeeper.getDetails())

println("// print individual values also")
println(scoreKeeper.currentOver)
println(scoreKeeper.currentBall)


} //object

// -- Output --

/*
Planets(Andromeda,Kepler-452b)
Details- The Planet Kepler-452b is in Andromeda galaxy
Andromeda
Kepler-452b

 lets see if this following statement works ?
 planetInfo.galaxy = "Exo" //this gives me an error for reassignment
```

```
of Val


 Example of immutable variable with mutable reference


Current Over is 0 and the player is playing ball# 1 of the over

 --- By the magic of time lapse, update the Info ---
 --- current over to 1 and current ball to 2 ---


--- Updated Info ---
Current Over is 1 and the player is playing ball# 2 of the over
// print individual values also
1
2


 *
 */

```