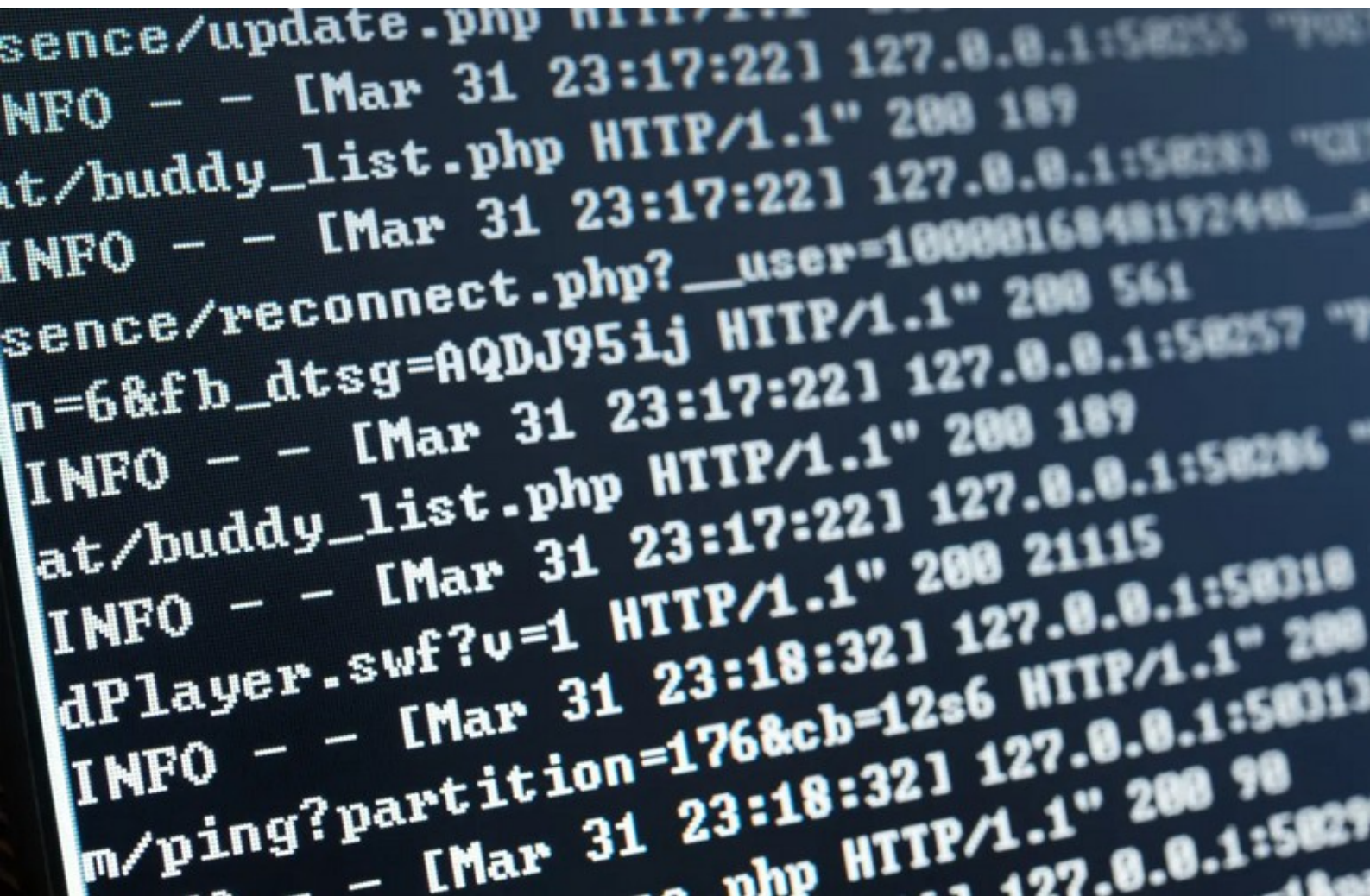


Rapport Mini-projet : Analyse des fichiers logs avec Akka & Scala.

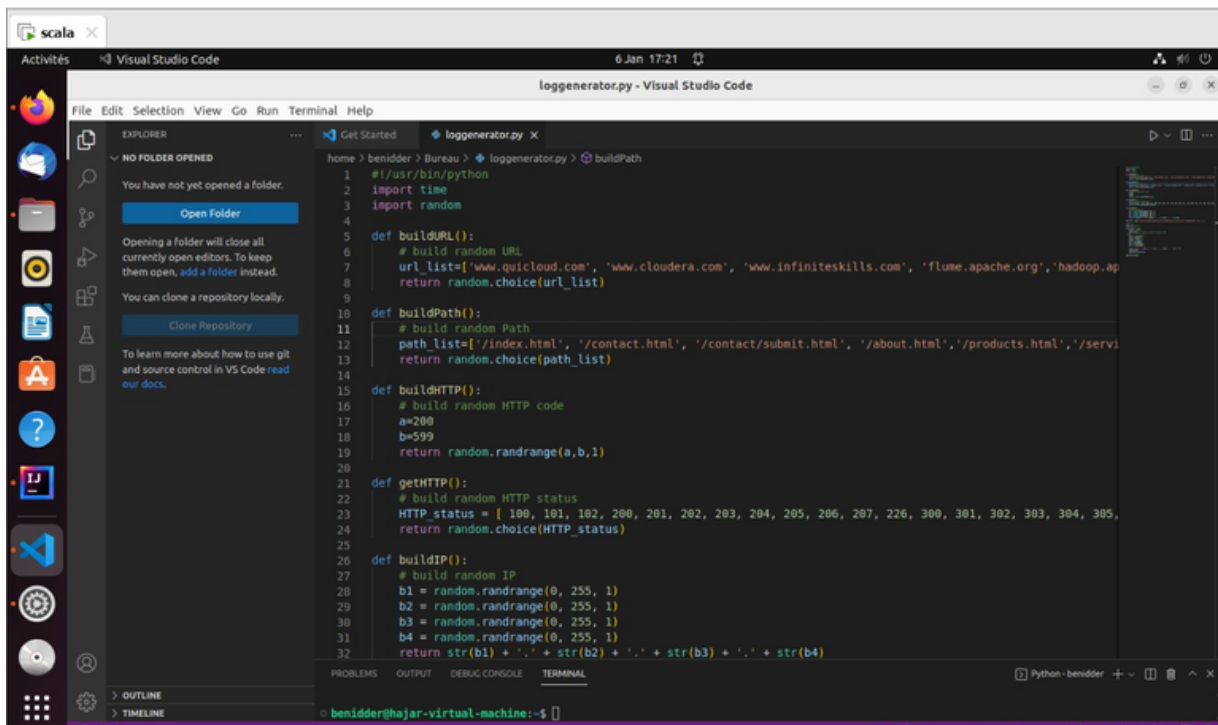


Fait par :

Hajar Benjarnij & Latifa Benidder

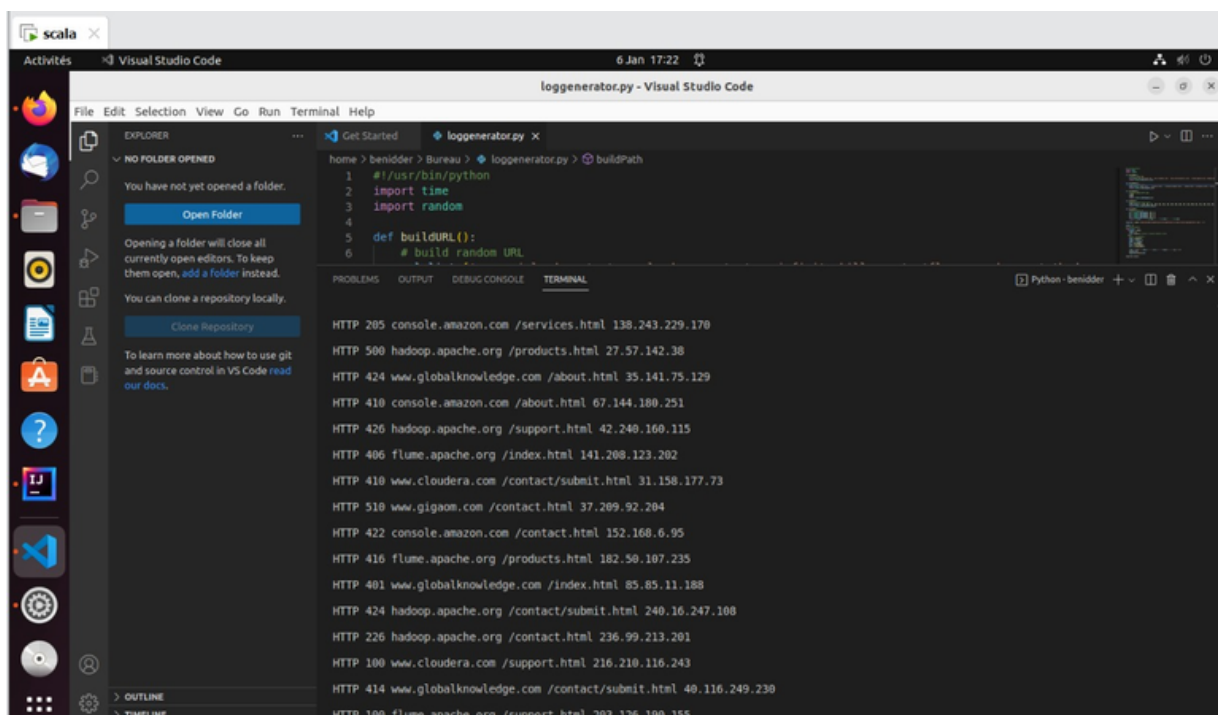
I- Génération du fichier log

D'abord, on lance le fichier python pour générer le fichier log.



The screenshot shows the Visual Studio Code editor with the file `loggenerator.py` open. The script is a Python program that generates random HTTP log entries. The code is as follows:

```
1 #!/usr/bin/python
2 import time
3 import random
4
5 def buildURL():
6     # build random URL
7     url_list=['www.quicloud.com', 'www.cloudera.com', 'www.infinetkills.com', 'flume.apache.org', 'hadoop.ap
8     return random.choice(url_list)
9
10 def buildPath():
11     # build random Path
12     path_list=['/index.html', '/contact.html', '/contact/submit.html', '/about.html', '/products.html', '/servi
13     return random.choice(path_list)
14
15 def buildHTTP():
16     # build random HTTP code
17     a=200
18     b=599
19     return random.randrange(a,b,1)
20
21 def getHTTP():
22     # build random HTTP status
23     HTTP_status = [ 100, 101, 102, 200, 201, 202, 203, 204, 205, 206, 207, 226, 300, 301, 302, 303, 304, 305,
24     return random.choice(HTTP_status)
25
26 def buildIP():
27     # build random IP
28     b1 = random.randrange(0, 255, 1)
29     b2 = random.randrange(0, 255, 1)
30     b3 = random.randrange(0, 255, 1)
31     b4 = random.randrange(0, 255, 1)
32     return str(b1) + '.' + str(b2) + '.' + str(b3) + '.' + str(b4)
```

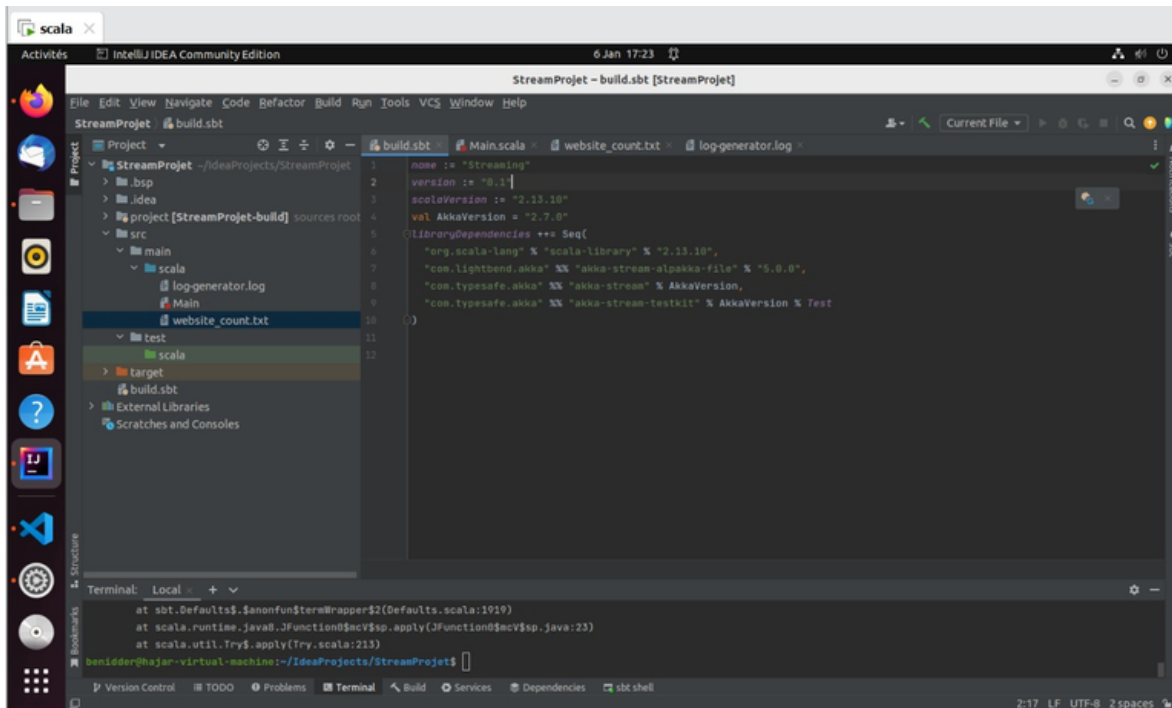


The screenshot shows the Visual Studio Code editor with the file `loggenerator.py` open. The terminal window at the bottom displays the output of the script, which is a list of random HTTP log entries. The output is as follows:

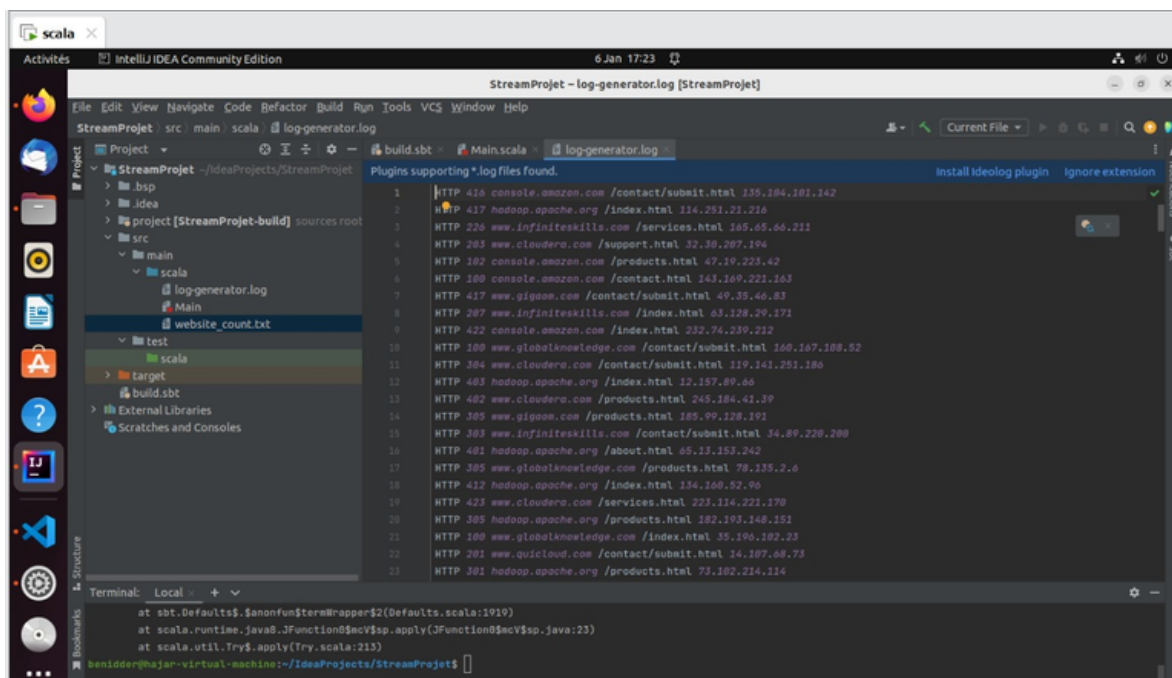
```
HTTP 205 console.amazon.com /services.html 138.243.229.170
HTTP 500 hadoop.apache.org /products.html 27.57.142.38
HTTP 424 www.globalknowledge.com /about.html 35.141.75.129
HTTP 410 console.amazon.com /about.html 67.144.180.251
HTTP 426 hadoop.apache.org /support.html 42.240.160.115
HTTP 406 flume.apache.org /index.html 141.208.123.202
HTTP 410 www.cloudera.com /contact/submit.html 31.158.177.73
HTTP 510 www.gigaom.com /contact.html 37.209.92.204
HTTP 422 console.amazon.com /contact.html 152.168.6.95
HTTP 416 flume.apache.org /products.html 182.50.107.235
HTTP 401 www.globalknowledge.com /index.html 85.85.11.188
HTTP 424 hadoop.apache.org /contact/submit.html 240.16.247.108
HTTP 226 hadoop.apache.org /contact.html 236.99.213.201
HTTP 100 www.cloudera.com /support.html 216.210.116.243
HTTP 414 www.globalknowledge.com /contact/submit.html 40.116.249.230
HTTP 100 flume.apache.org /support.html 203.126.190.155
```

II- Création du projet sbt

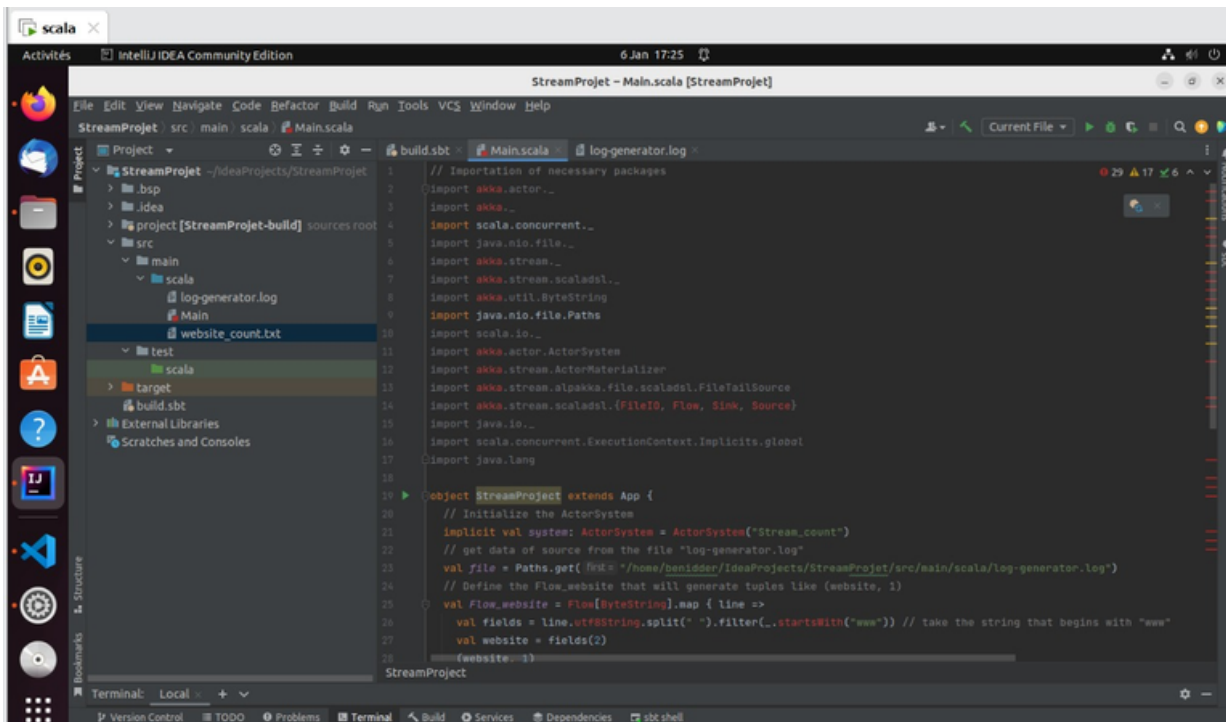
Avec l'IDE intelli, on crée un projet sbt.
fichier build.sbt



log-generator.log



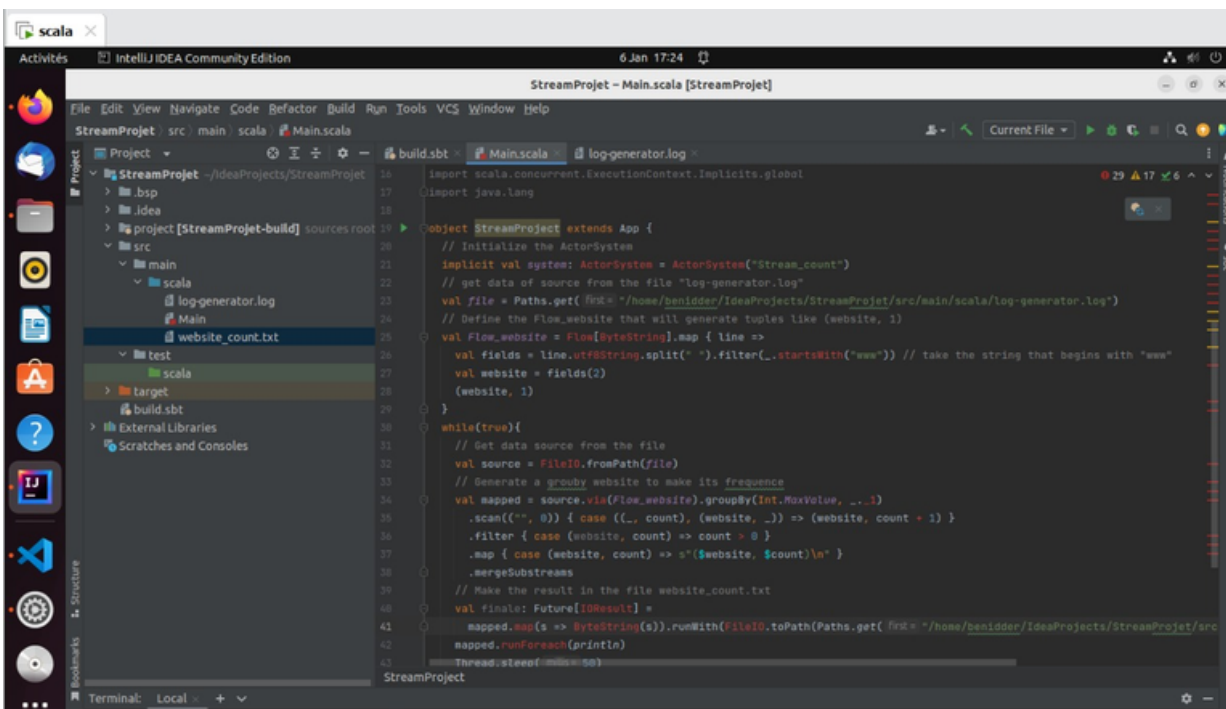
Main.scala



```
// Importation of necessary packages
import akka.actor._
import akka._
import scala.concurrent._
import java.nio.file._
import akka.stream._
import akka.stream.scaladsl._
import akka.util.ByteString
import java.nio.file.Paths
import scala.io._

import akka.actor.ActorSystem
import akka.stream.ActorMaterializer
import akka.stream.alpakka.file.scaladsl.FileTailSource
import akka.stream.scaladsl.{FileIO, Flow, Sink, Source}
import java.io._
import scala.concurrent.ExecutionContext.Implicits.global
import java.lang

object StreamProject extends App {
  // Initialize the ActorSystem
  implicit val system: ActorSystem = ActorSystem("Stream_count")
  // get data of source from the file "log-generator.log"
  val file = Paths.get("/home/benidder/IdeaProjects/StreamProject/src/main/scala/log-generator.log")
  // Define the Flow_website that will generate tuples like (website, 1)
  val Flow_website = Flow[ByteString].map { line =>
    val fields = line.utf8String.split(" ").filter(_.startsWith("www")) // take the string that begins with "www"
    val website = fields(2)
    (website, 1)
  }
}
```



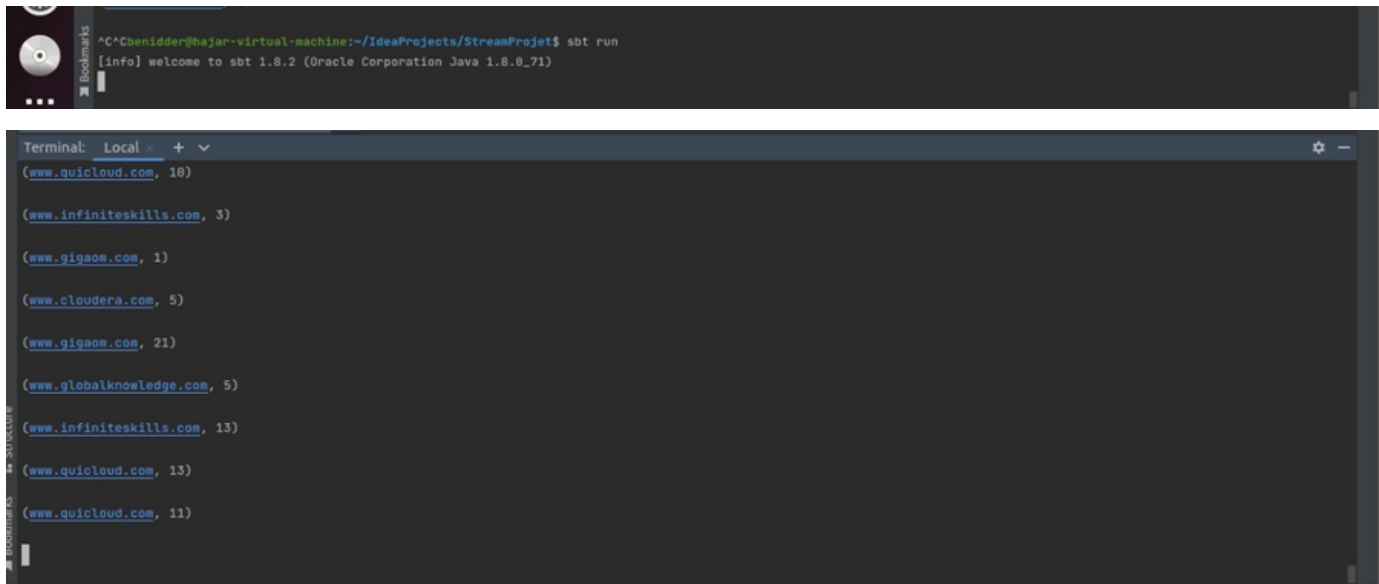
```
import scala.concurrent.ExecutionContext.Implicits.global
import java.lang

object StreamProject extends App {
  // Initialize the ActorSystem
  implicit val system: ActorSystem = ActorSystem("Stream_count")
  // get data of source from the file "log-generator.log"
  val file = Paths.get("/home/benidder/IdeaProjects/StreamProject/src/main/scala/log-generator.log")
  // Define the Flow_website that will generate tuples like (website, 1)
  val Flow_website = Flow[ByteString].map { line =>
    val fields = line.utf8String.split(" ").filter(_.startsWith("www")) // take the string that begins with "www"
    val website = fields(2)
    (website, 1)
  }

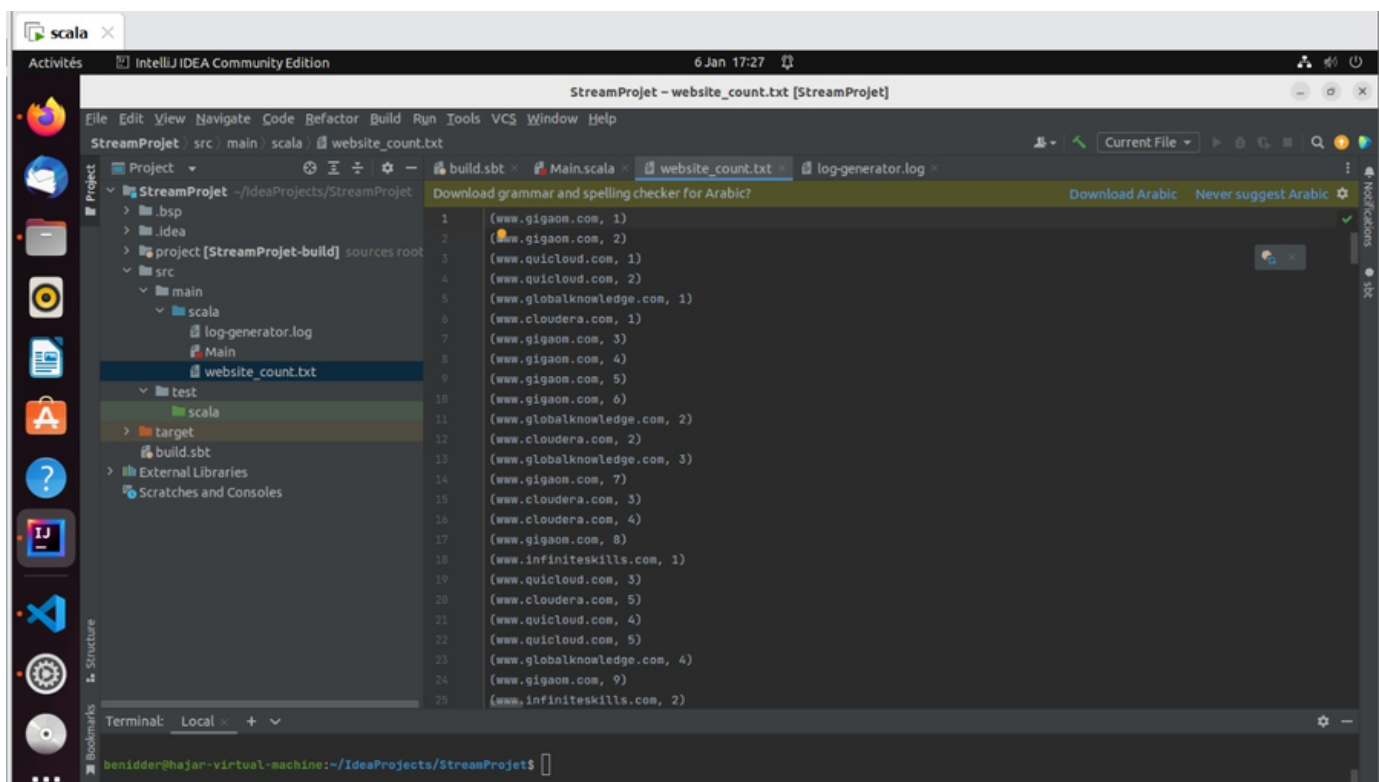
  while(true){
    // Get data source from the file
    val source = FileIO.fromPath(file)
    // Generate a groupby website to make its frequency
    val mapped = source.via(Flow_website).groupBy(int.MaxValue, 1)
    .scan("", 0) { case (_, count), (website, _) => (website, count + 1) }
    .filter { case (website, count) => count > 0 }
    .map { case (website, count) => s"($website, $count)\n" }
    .mergeSubstreams
    // Make the result in the file website_count.txt
    val final: Future[IOResult] =
      mapped.map(s => ByteString(s)).runWith(FileIO.toPath(Paths.get("/home/benidder/IdeaProjects/StreamProject/src/main/scala/website_count.txt")))
    mapped.runForeach(println)
    Thread.sleep(500)
  }
}
```

A l'aide du Akka Stream. on fait le processing du fichier log générer et on stocke le résultat dans un autre fichier.

Après, l'exécution avec la commande `sbt run`. L'affichage de chaque site avec sa fréquence.



Après, on a stockés le résultat dans le fichier `website_count`.



A l'aide du Data Mining et le langage Python, on représente le résultat sous forme de visualisation.

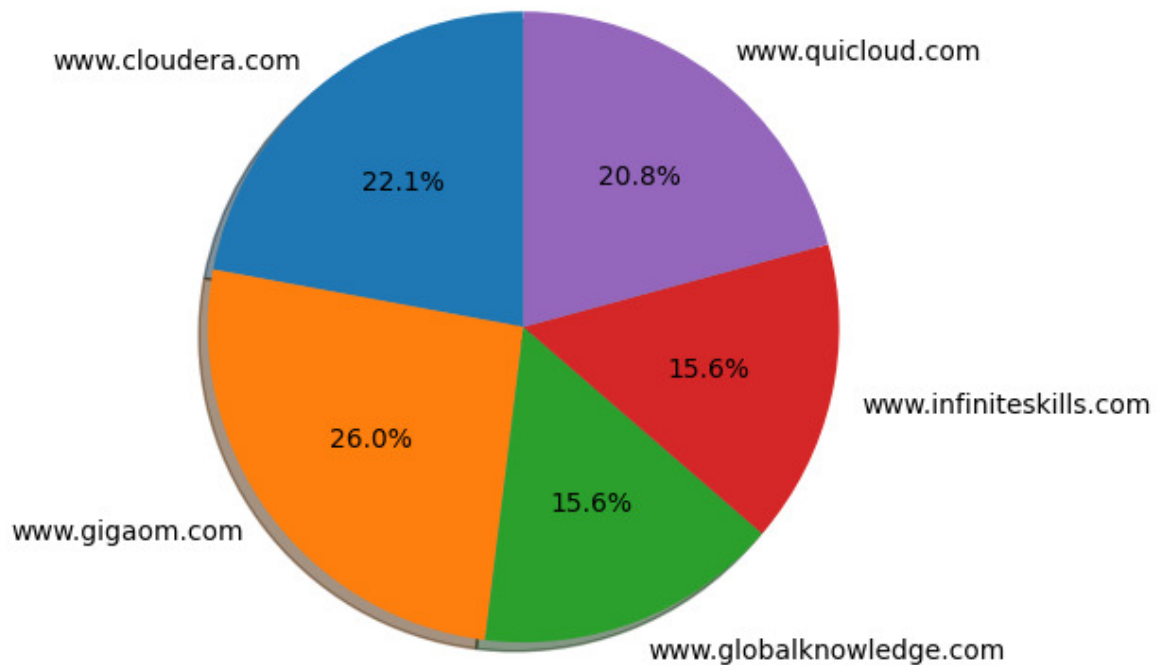


Figure 1 : Fréquence de visite pour chaque site web