



VRIJE  
UNIVERSITEIT  
BRUSSEL



# ASSIGNMENT 1: AKKA STREAMS

Software Architectures

Anisha Sachdeva

August 21, 2022

Prof. Coen De Roover

# 1 Introduction

Given a list of NPM packages from the NPM software registry, using the Pipe and Filter architecture we retrieve back the number of dependencies, devdependencies and the list of keywords (if any) along with the modifications in keywords list for each version of a package.

We are provided with the list of packages to be processed in Akka streams in a form of GZIP compressed file. As mentioned in project description we first extract the list of packages from the GZIP compressed file followed by retrieving the list of dependencies, devdependencies and keywords by implementing the two main components of Akka streams, i.e. Source and Flow. Once, we get the dependencies lists, using another custom FlowShape, we analyse the list of keywords for different versions of a package. Further using Sink, we display the required result consisting of package name being analysed having different versions with different number of dependencies, devdependencies and the difference in keywords.

Following section talk in detail about the implementation which is briefly mentioned above.

# 2 Implementation

We are provided with the GZIP file containing the list of packages that we need to process. We first feed the GZIP file into a Source using the FileIO method. Following the output from the Source, we then decompress the GZIP file using the Flow operator, `Compression.gunzip()`. This flow basically decompress the GZIP into a stream of ByteStrings. As after unzipping the file, we get one long ByteString representing the whole file, we make another Flow, to split the ByteString into 1 ByteString per newline. We call the flow as `flowSplitPerLine`. Further, we convert the ByteStrings into Strings using a flow named as `flowStringCase`. Now we get the package names in form of strings which are ready to be processed further.

As mentioned in the project description, the system should only be buffered with a maximum of 10 packages and in case of an overflow, the strategy adopted should be, backpressure strategy. Also to keep the API requests controlled, we should stream only one package each 3 seconds. Hence, we made a flow named as `bufferedSource`, which basically receives a string (package name) and emits one strings (package name) per 3 seconds using throttle along with buffer size as 10 and overflow strategy as backpressure. We used throttle as a flow operator to limit the throughput to 1 element (package) per 3 seconds.

Next, we made another flow which would take a package name as an input and would emit out the list of `DependenciesPerVersion` object for that package name. `DependenciesPerVersion` is a case class that we defined with the following parameters:

1. `packageName` of type String
2. `versionNumber` of type String
3. `dependenciesList` as a List of strings containing the name of dependencies for a version of a package
4. `devDependenciesList` as a List of strings containing the name of devdependencies for a version of a package
5. `keywordsList` as a List of strings containing the keywords for a version of a package

This flow would use the `get.json` method which would take the name of the package and request the NPM registry API to get the information regarding versions, dependencies lists and keywords lists of that particular package. The output from this `get.json` method is a list of `DependenciesPerVersion` objects for a particular package. This was we get a stream of one package to further work on.

Further, from this list of all objects for a package, we want to flatten the list to analyse the dependencies and devdependencies for each version. Hence, we made another flow named as `flatteningFlow`, which would take the List of `DependenciesPerVersion` objects and will give one object for `DependencyPerVersion`. That is, now we a `DependencyPerVersion` object for a particular package and a particular version number.

Again, as specified to control all versions we buffer 12 versions as maximum with backpressure strategy in our versionBuffer flow.

Next comes the interesting part wherein we make our custom FlowShape to process the number of dependencies in a parallel way. In this flow graph, we get one object of a DependencyPerVersion class and emit an object of TotalCountPerVersion. TotalCountPerVersion is case class consisting of following parameters:

1. packageName of type String
2. versionNumber of type String
3. dependenciesTotalCount of type integer
4. devDependenciesTotalCount of type integer
5. keywordList as a List of strings containing the keywords for a version of a package

As this project is an extension of session 1's project description, we already had a case class TotalCountPerVersion counting the required number of dependencies, but as now needed we simply added the list of keywords in the class but we don't analyse them here. In this custom flowShape, we use the balanced approach to send groups of versions of a package to two pipelines. In each pipeline, for a version, we map the DependencyPerVersion object to TotalCountPerVersion object and fill in the required fields to be displayed. From DependencyPerVersion we take the version number, count of dependencies in list and count of devdependencies in list and fill in the values in TotalCountPerVersion object for final display.

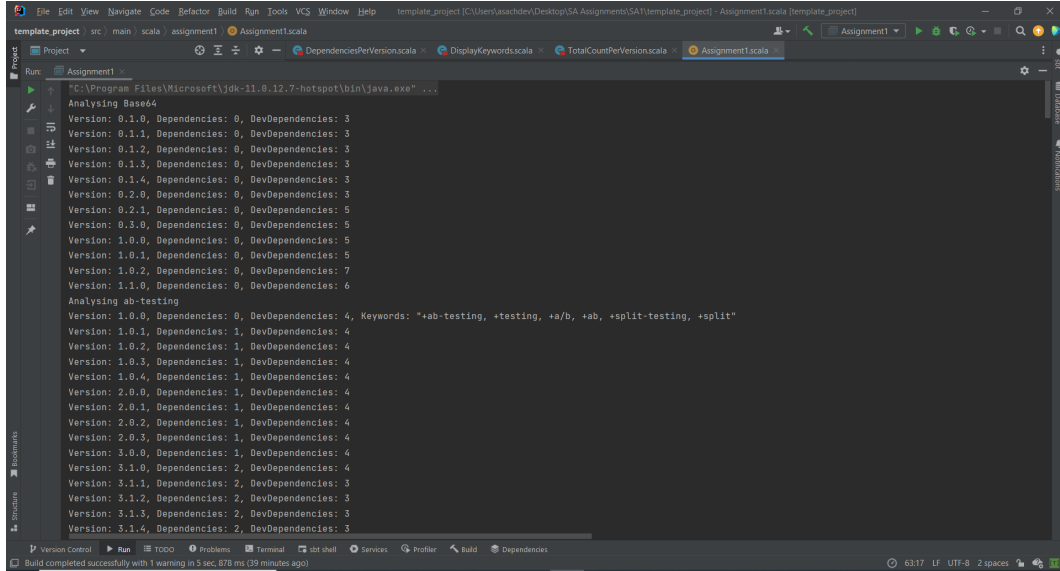
Further, as an extension of session 1's project, now we need to analyse the list of keywords for various versions of a package and we would only display the modifications in a list. For this, again we make another custom flowShape which takes input as TotalCountPerVersion object and gives back DisplayKeywords. DisplayKeywords is another case class defined with following parameters:

1. packageName of type String
2. versionNumber of type String
3. dependenciesTotalCount of type integer
4. devDependenciesTotalCount of type integer
5. additionalKeywords as a list of keywords that would not be present in previous version but are available in the current version. The list of (+) keywords.
6. subtractedKeywords as a list of keywords that were present in previous version but are not available in the current version. The list of (-) keywords.

In this keywordGraph, custom flowShape we do a number of things. As we need to compare the lists of keywords of two concurrent versions of a package, we used the flow operator, prepend. Using prepend, we get a dummy TotalCountPerVersion with empty list of keywords and other values. We used this object as our first object. This is done as we needed to compare something with the very first version of a package in order to display the keyword list. Also, we used broadcast to broadcast the TotalCountPerVersion to two ports and then used ZipWith2 graphshape to compare the value of keywords list of two successive versions of a package and display only the modifications using the DisplayKeywords object.

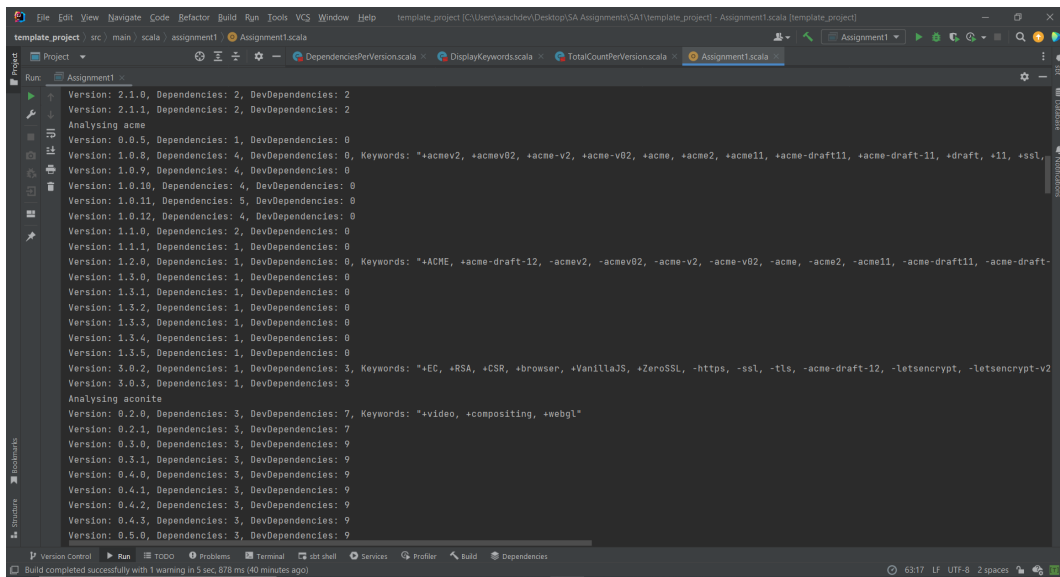
### 3 Result

At last using the Sink and Runnable graph, we display the required output as show below:



```
template_project [C:\Users\asachdev\Desktop\SA Assignments\SA1\template_project] - Assignment1.scala [template_project]
template_project src \ main \ scala \ assignment1 Assignment1.scala
Run: Assignment1
"C:\Program Files\Microsoft\jdk-11.0.12-hotspot\bin\java.exe" ...
Analysing Base64
Version: 0.1.0, Dependencies: 0, DevDependencies: 3
Version: 0.1.1, Dependencies: 0, DevDependencies: 3
Version: 0.1.2, Dependencies: 0, DevDependencies: 3
Version: 0.1.3, Dependencies: 0, DevDependencies: 3
Version: 0.1.4, Dependencies: 0, DevDependencies: 3
Version: 0.2.0, Dependencies: 0, DevDependencies: 3
Version: 0.2.1, Dependencies: 0, DevDependencies: 5
Version: 0.3.0, Dependencies: 0, DevDependencies: 5
Version: 1.0.0, Dependencies: 0, DevDependencies: 5
Version: 1.0.1, Dependencies: 0, DevDependencies: 5
Version: 1.0.2, Dependencies: 0, DevDependencies: 7
Version: 1.1.0, Dependencies: 0, DevDependencies: 6
Analysing ab-testing
Version: 1.0.0, Dependencies: 0, DevDependencies: 4, Keywords: "+ab-testing, +testing, +a/b, +ab, +split-testing, +split"
Version: 1.0.1, Dependencies: 1, DevDependencies: 4
Version: 1.0.2, Dependencies: 1, DevDependencies: 4
Version: 1.0.3, Dependencies: 1, DevDependencies: 4
Version: 1.0.4, Dependencies: 1, DevDependencies: 4
Version: 2.0.0, Dependencies: 1, DevDependencies: 4
Version: 2.0.1, Dependencies: 1, DevDependencies: 4
Version: 2.0.2, Dependencies: 1, DevDependencies: 4
Version: 2.0.3, Dependencies: 1, DevDependencies: 4
Version: 3.0.0, Dependencies: 1, DevDependencies: 4
Version: 3.1.0, Dependencies: 2, DevDependencies: 4
Version: 3.1.1, Dependencies: 2, DevDependencies: 3
Version: 3.1.2, Dependencies: 2, DevDependencies: 3
Version: 3.1.3, Dependencies: 2, DevDependencies: 3
Version: 3.1.4, Dependencies: 2, DevDependencies: 3
Build completed successfully with 1 warning in 5 sec, 878 ms (39 minutes ago)
```

Figure 1: Output



```
template_project [C:\Users\asachdev\Desktop\SA Assignments\SA1\template_project] - Assignment1.scala [template_project]
template_project src \ main \ scala \ assignment1 Assignment1.scala
Run: Assignment1
Version: 2.1.0, Dependencies: 2, DevDependencies: 2
Version: 2.1.1, Dependencies: 2, DevDependencies: 2
Analysing acme
Version: 0.0.5, Dependencies: 1, DevDependencies: 0
Version: 1.0.0, Dependencies: 4, DevDependencies: 0
Version: 1.0.9, Dependencies: 4, DevDependencies: 0
Version: 1.0.10, Dependencies: 4, DevDependencies: 0
Version: 1.0.11, Dependencies: 5, DevDependencies: 0
Version: 1.0.12, Dependencies: 4, DevDependencies: 0
Version: 1.1.0, Dependencies: 2, DevDependencies: 0
Version: 1.1.1, Dependencies: 1, DevDependencies: 0
Version: 1.2.0, Dependencies: 1, DevDependencies: 0, Keywords: "+ACME, +acme-draft-12, -acnev2, -acnev02, -acme-v2, -acme-v02, -acme, -acme2, -acme11, -acme-draft11, -acme-draft-"
Version: 1.3.0, Dependencies: 1, DevDependencies: 0
Version: 1.3.1, Dependencies: 1, DevDependencies: 0
Version: 1.3.2, Dependencies: 1, DevDependencies: 0
Version: 1.3.3, Dependencies: 1, DevDependencies: 0
Version: 1.3.4, Dependencies: 1, DevDependencies: 0
Version: 1.3.5, Dependencies: 1, DevDependencies: 0
Version: 3.0.2, Dependencies: 1, DevDependencies: 3, Keywords: "+EC, +RSA, +CSR, +browser, +VanillaJS, +ZeroSSL, -https, -ssl, -tls, -acme-draft-12, -letsencrypt, -letsencrypt-v2"
Version: 3.0.3, Dependencies: 1, DevDependencies: 3
Analysing aconite
Version: 0.2.0, Dependencies: 3, DevDependencies: 7, Keywords: "+video, +compositing, +webgl"
Version: 0.2.1, Dependencies: 3, DevDependencies: 7
Version: 0.3.0, Dependencies: 3, DevDependencies: 9
Version: 0.3.1, Dependencies: 3, DevDependencies: 9
Version: 0.4.0, Dependencies: 3, DevDependencies: 9
Version: 0.4.1, Dependencies: 3, DevDependencies: 9
Version: 0.4.2, Dependencies: 3, DevDependencies: 9
Version: 0.4.3, Dependencies: 3, DevDependencies: 9
Version: 0.5.0, Dependencies: 3, DevDependencies: 9
Build completed successfully with 1 warning in 5 sec, 878 ms (40 minutes ago)
```

Figure 2: Output showing difference in keywords list