# Writing your own MapReduce applications



## Inverted Index

In this section, your task is to write a Hadoop MapReduce program that will build an inverted index of documents. An inverted index is a map associating words or phrases with individual documents. For example, assume we have two files, A and B. A contains the words **`blue`** and **`house`**, while B contains the words **`red`** and **`house`**. The inverted index for these two file would be:

```
blue: A
house: A,B
red: B
```

**Note**: For this checkpoint, assign the tag with Key: **`Project`** and Value: **`4.1`** for all resources

One approach to perform inverted indexing in MapReduce is to configure the mapper to take each word in the line (after the stop words are removed) and output the word followed by the name of the file that the word appears in. Since files are chunked by HDFS and each mapper gets an independent chunk, the filename corresponding to an input is not immediately available. However, the **`Context`** and **`FileSplit`** classes allow you to fetch the name of the current file being processed by a Map task. Adding the following code to your Map function will do the trick:

```
FileSplit fs = (FileSplit) context.getInputSplit();
String location = fs.getPath().getName();
```

Once the Mapper is configured to output words followed by their filename, you can configure the reducer to iterate over all of the values for a particular key. You can then emit a list of filenames associated with each word and store the result in HDFS. Sample output from your program should be as follows:

```
word1 : file1 file4 file8
word2 : file3 file7 file9
```

Once you have compiled and tested your code, please run it over a dataset comprising of a subset of books from Project Gutenberg. The dataset is stored in **`s3://15319_book_dataset/`**. You will want to download and extract the dataset before loading it into HDFS. Please mount and use ephemeral storage as the root partition (**`/`**) will not have enough space. The dataset contains about 6500 e-books in plain format, and is roughly 6.5 GB uncompressed. Please use ephemeral storage for input dataset decompression as the root partition (**`/`**) will not have enough space. By default, the ephemeral storage is mounted at **`/mnt`** on each instance in an EMR cluster once they are provisioned.

Notes and Assumptions:

1. Treat all words as case-insensitive. Words such as **`life`** and **`LIFE`** should be treated as the same, you can choose to output the case-insensitive version of the word.
2. All punctuation should be stripped and replaced with a space. The word **`don't`** will then become **`don t`**.
3. If you attempting the bonus below, remove the stop words after stripping punctuation.
4. You can visit the main page of your Hadoop cluster at **`http://<master-public-DNS-name>:9100`** to view your jobs and their progress. Port 9100 on your instance will have to be opened in security groups first for your browser to be able to access the page.

## Bonus: Stop Word Removal

In NLP, stop word removal is a common pre-processing step so that punctuation and frequently occurring words such as "and, the" etc. are removed. This allows the index to have terms that are more relevant to search queries. You can find a list of stop words here. For the bonus, your task is to filter out these stop words within your hadoop program, before they are indexed and output by your reducer. We recommend using a Distributed Cache in this application.

**checkpoint**

Hadoop MapReduce