Udemy - 2 - Exploratory Data Analysis - Lab

Lab - preparing data for TF-IDF on Spark and EMR

TF-IDF

- Stands for *Term Frequency* and *Inverse Document Frequency*
- Important data for search – figures out what terms are most relevant for a document

TF-IDF Explained

- *Term Frequency* just measures how often a word occurs in a document
  - A word that occurs frequently is probably important to that document's meaning
- *Document Frequency* is how often a word occurs in an entire set of documents, i.e., all of Wikipedia or every web page
  - This tells us about common words that just appear everywhere no matter what the topic, like "a", "the", "and", etc.

TF-IDF Explained

- So a measure of the relevancy of a word to a document might be:

$$\frac{Term\ Frequency}{Document\ Frequency}$$

Or: Term Frequency * Inverse Document Frequency

That is, take how often the word appears in a document, over how often it just appears everywhere. That gives you a measure of how important and unique this word is for this document

TF-IDF In Practice

- We actually use the log of the IDF, since word frequencies are distributed exponentially. That gives us a better weighting of a words overall popularity
- TF-IDF assumes a document is just a "bag of words"
  - Parsing documents into a bag of words can be most of the work
  - Words can be represented as a hash value (number) for efficiency
  - What about synonyms? Various tenses? Abbreviations? Capitalizations? Misspellings?
- Doing this at scale is the hard part
  - That's where Spark comes in!

Unigrams, bigrams, etc.

- An extension of TF-IDF is to not only compute relevancy for individual words (terms) but also for *bi-grams* or, more generally, *n-grams*.
- "I love certification exams"
  - Unigrams: "I", "love", "certification", "exams"
  - Bi-grams: "I love", "love certification", "certification exams"
  - Tri-grams: "I love certification", "love certification exams"

Dimensions of Matrix: {2, 9}
9 unique terms (uni and bi grams) across 2 documents

Sample TF-IDF matrix with unigrams and bigrams

	I	Love	Certification	Exams	Puppies	I love	Love certification	Love puppies	Certification exams
"I love certification exams"									
"I love puppies"									

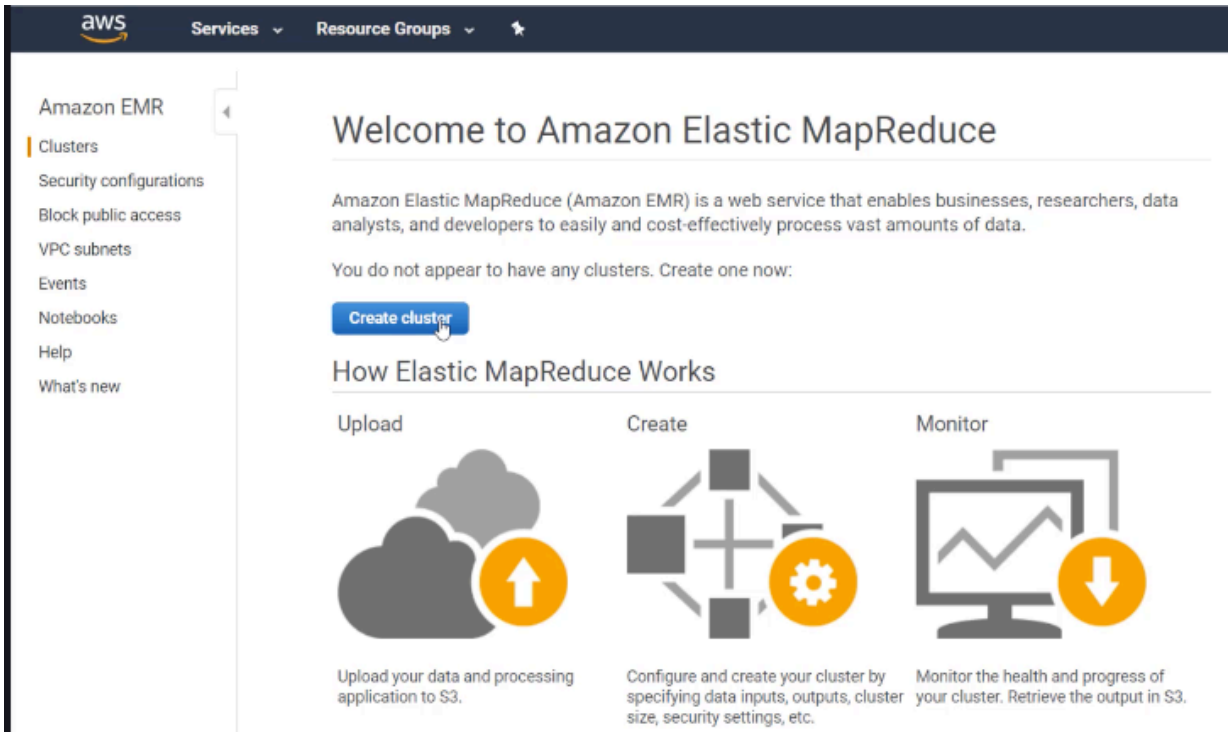
Using TF-IDF

- A very simple search algorithm could be:
  - Compute TF-IDF for every word in a corpus
  - For a given search word, sort the documents by their TF-IDF score for that word
  - Display the results

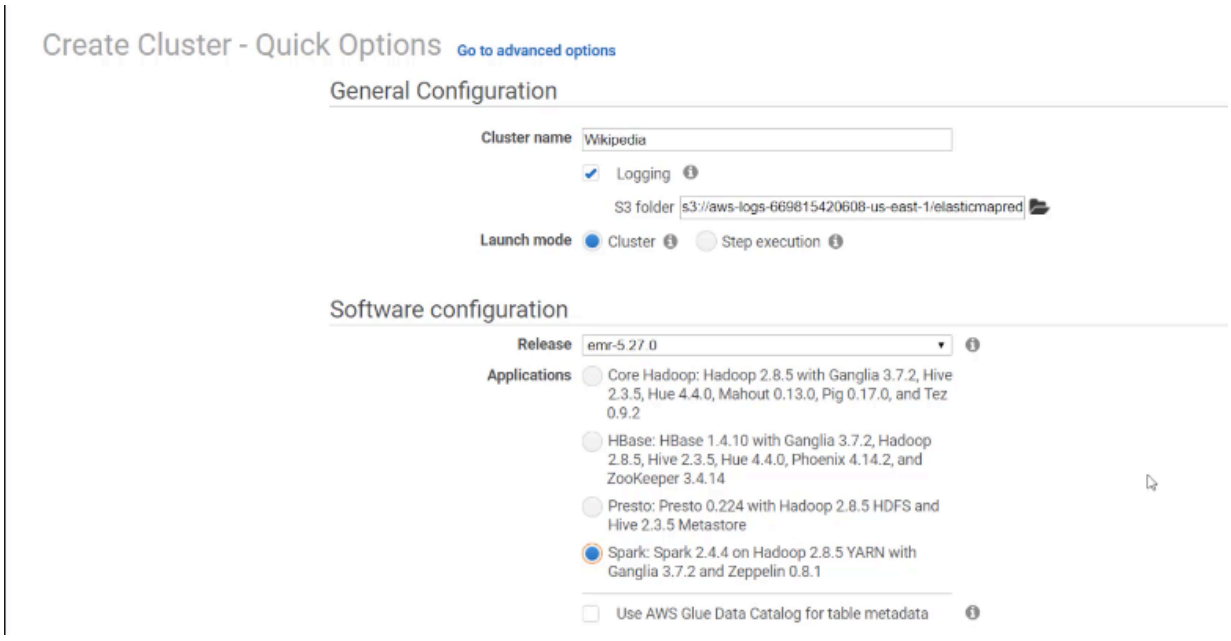
Let's use TF-IDF on Wikipedia

Now let’s implement that in AWS

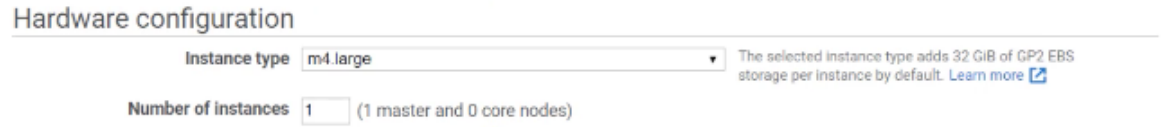
Create an EMR cluster



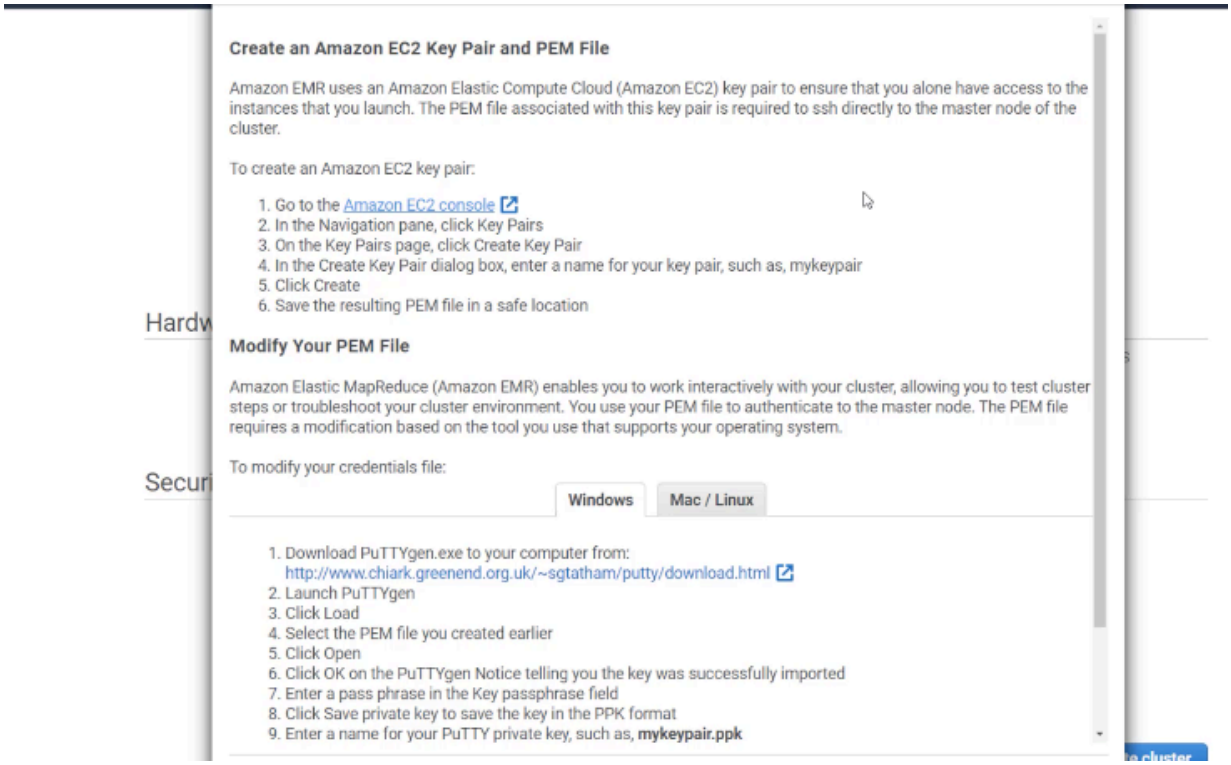
Spark cluster

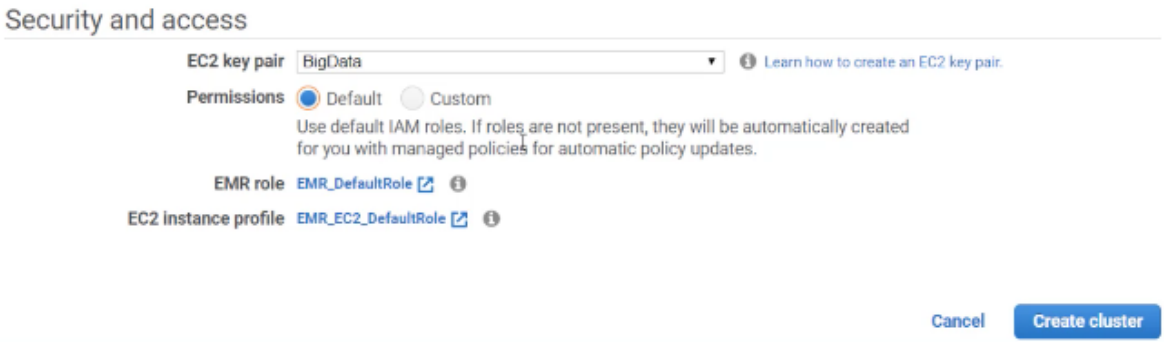


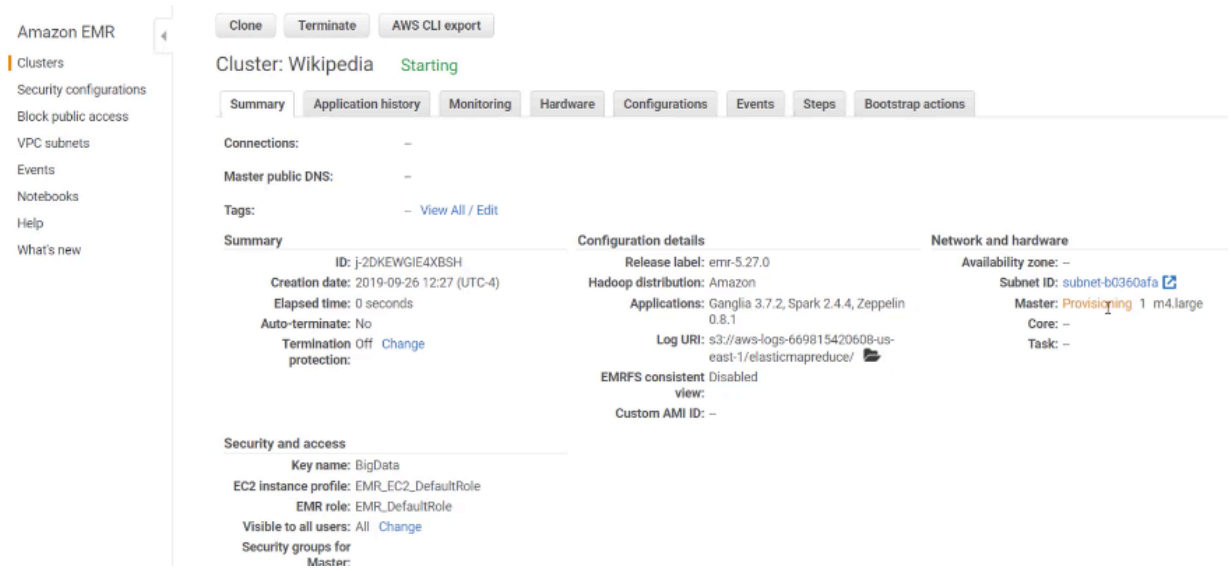
Keep it simple/cheap - just 1 instance m4.large



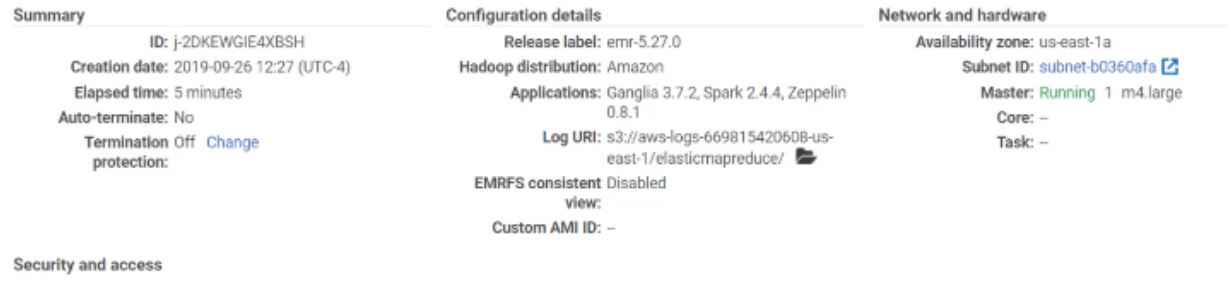
Create an EC2key pair if we don’t have one already - this is to connect to the master node of EMR cluster



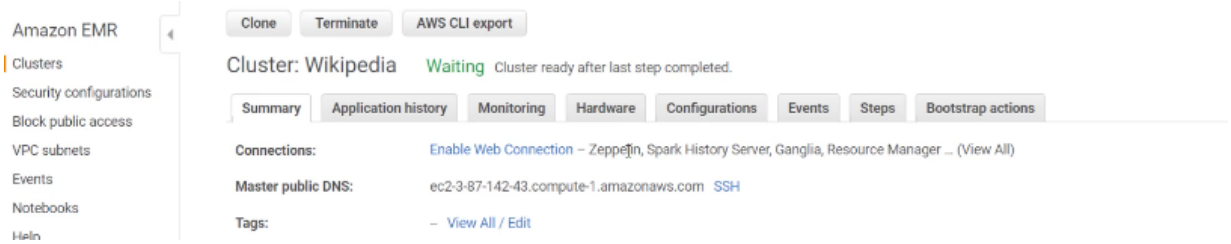


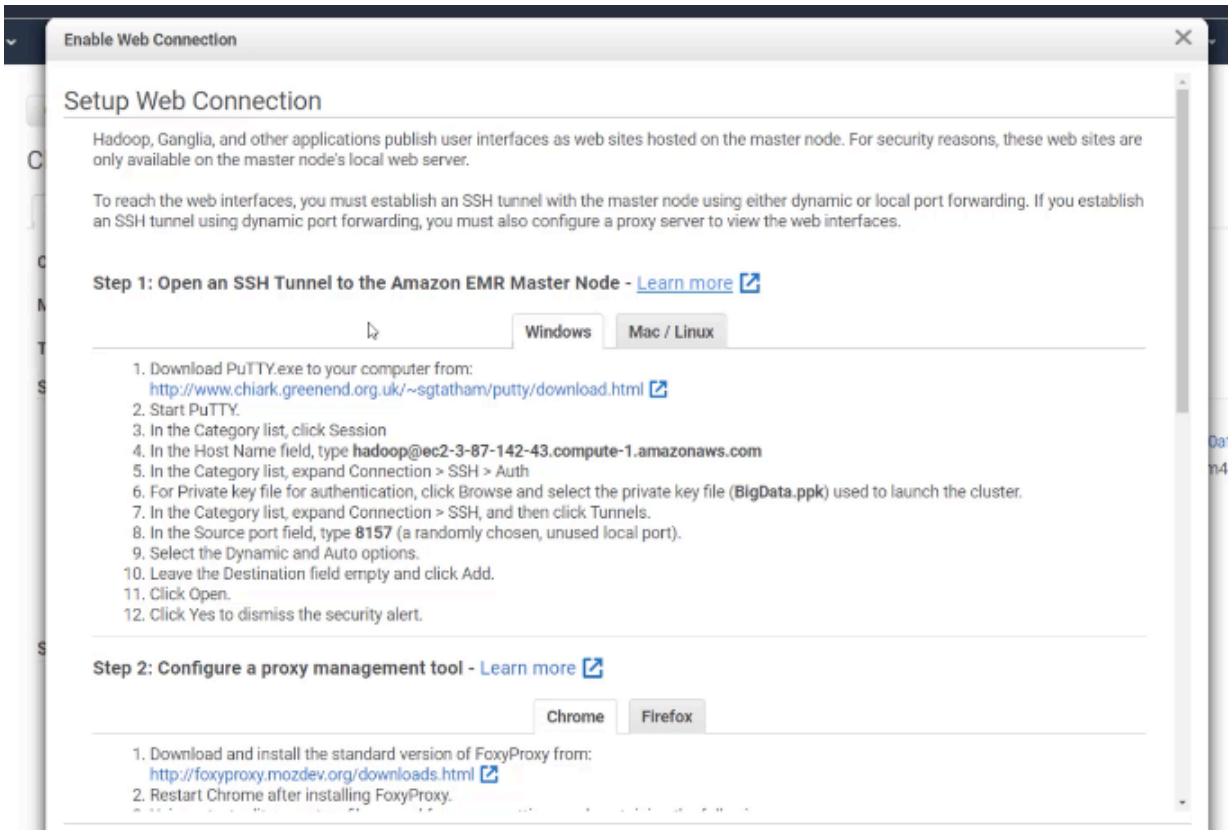


Cluster is now running - it comes with Zeppelin so we can interact with the cluster via notebook

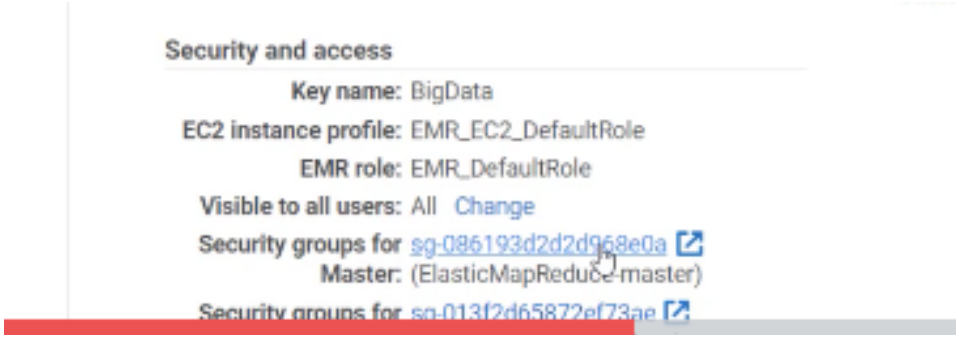


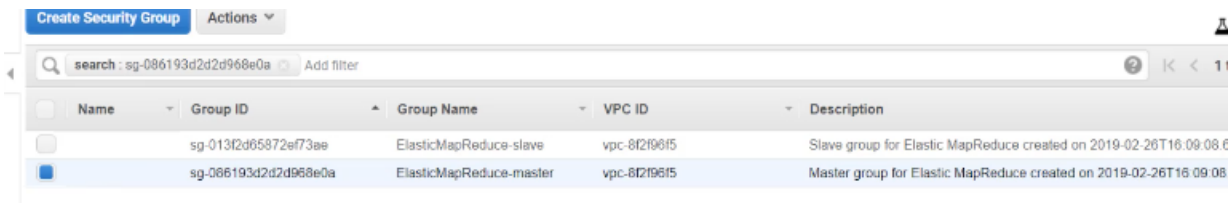
To connect to it: need to enable a web connection



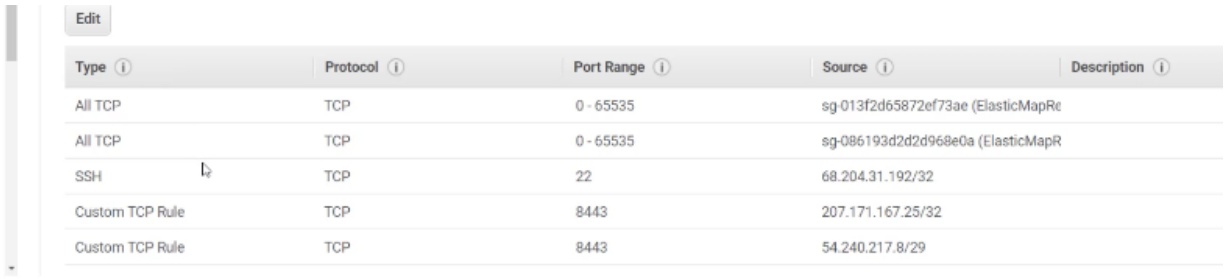


First, need to open a port to the machine
Click on security group

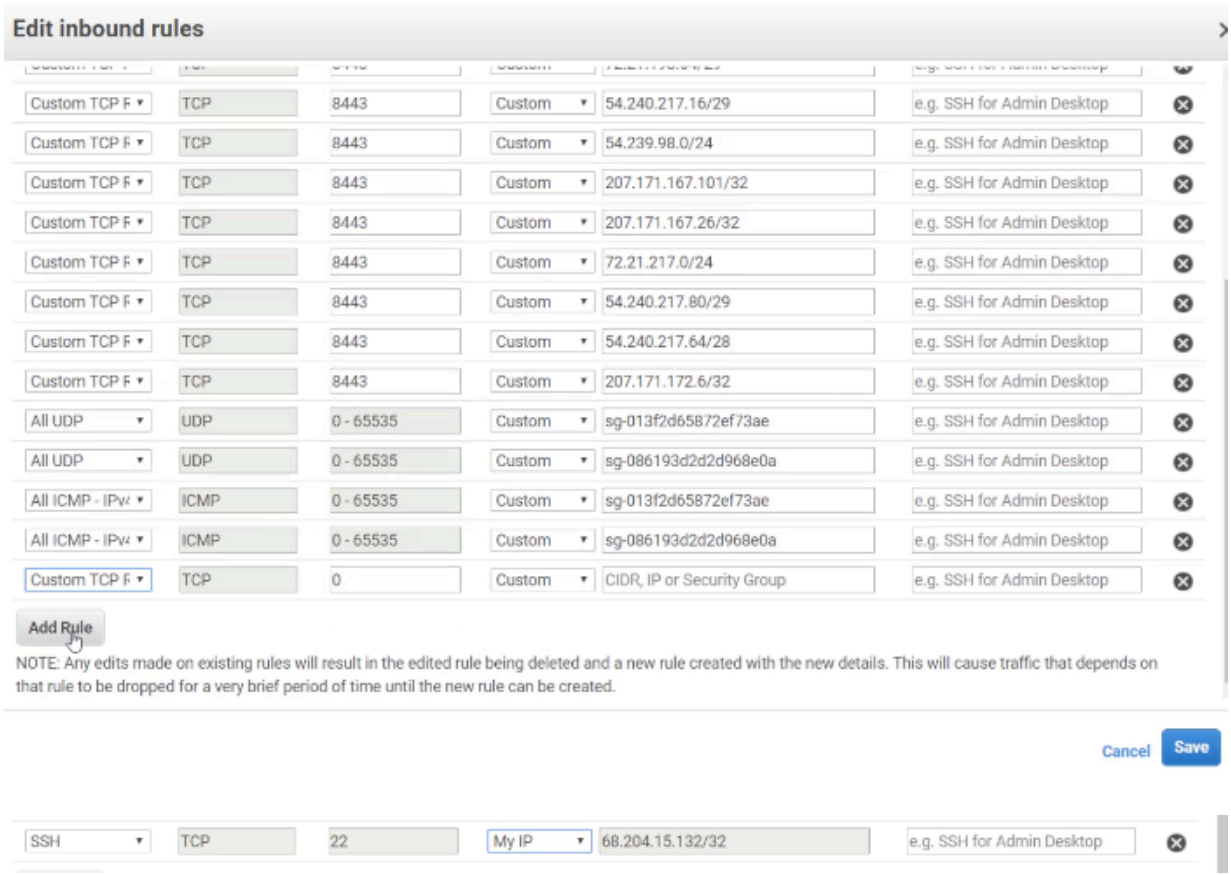




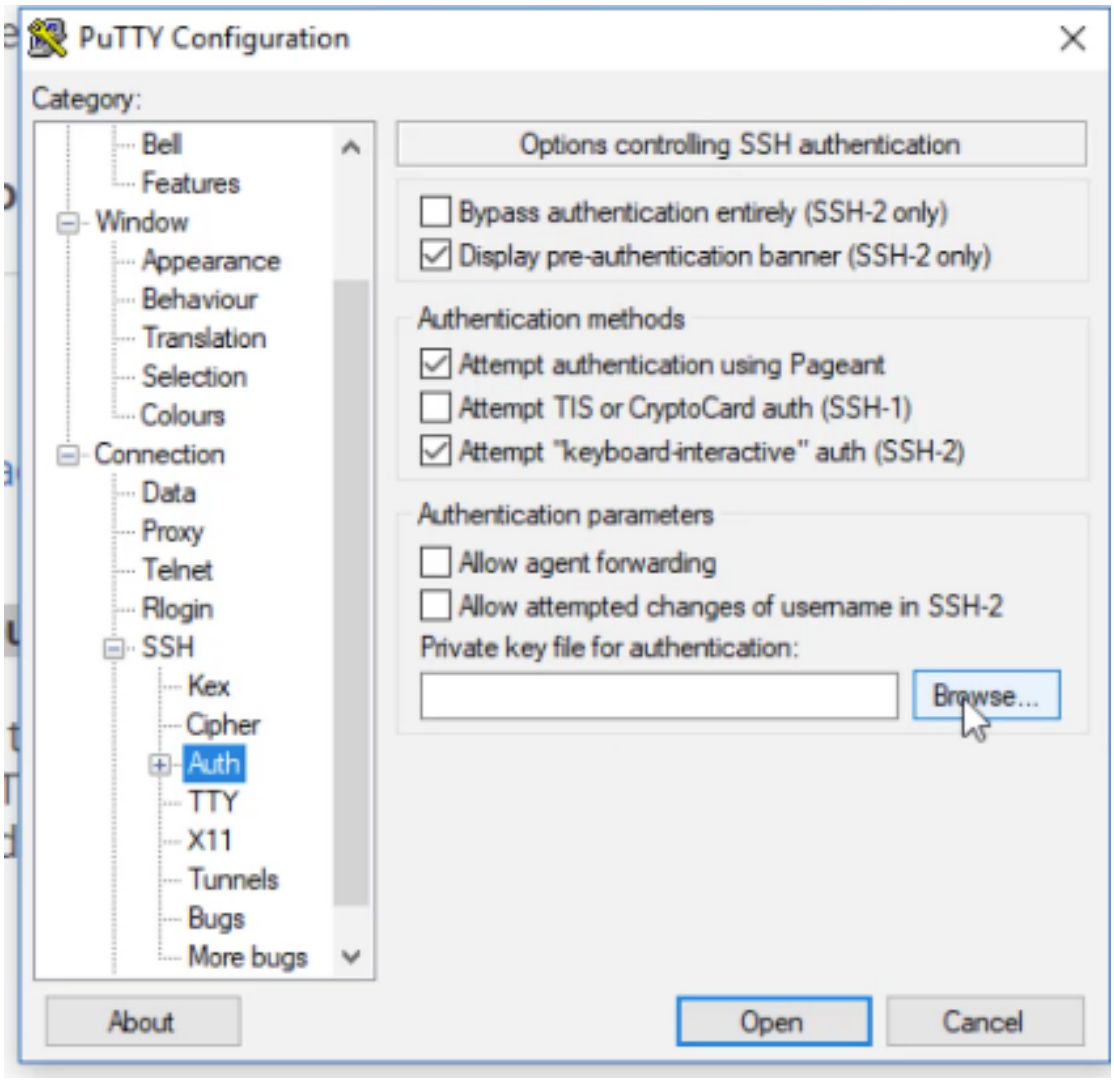
Click on inbound - we need to open port 22



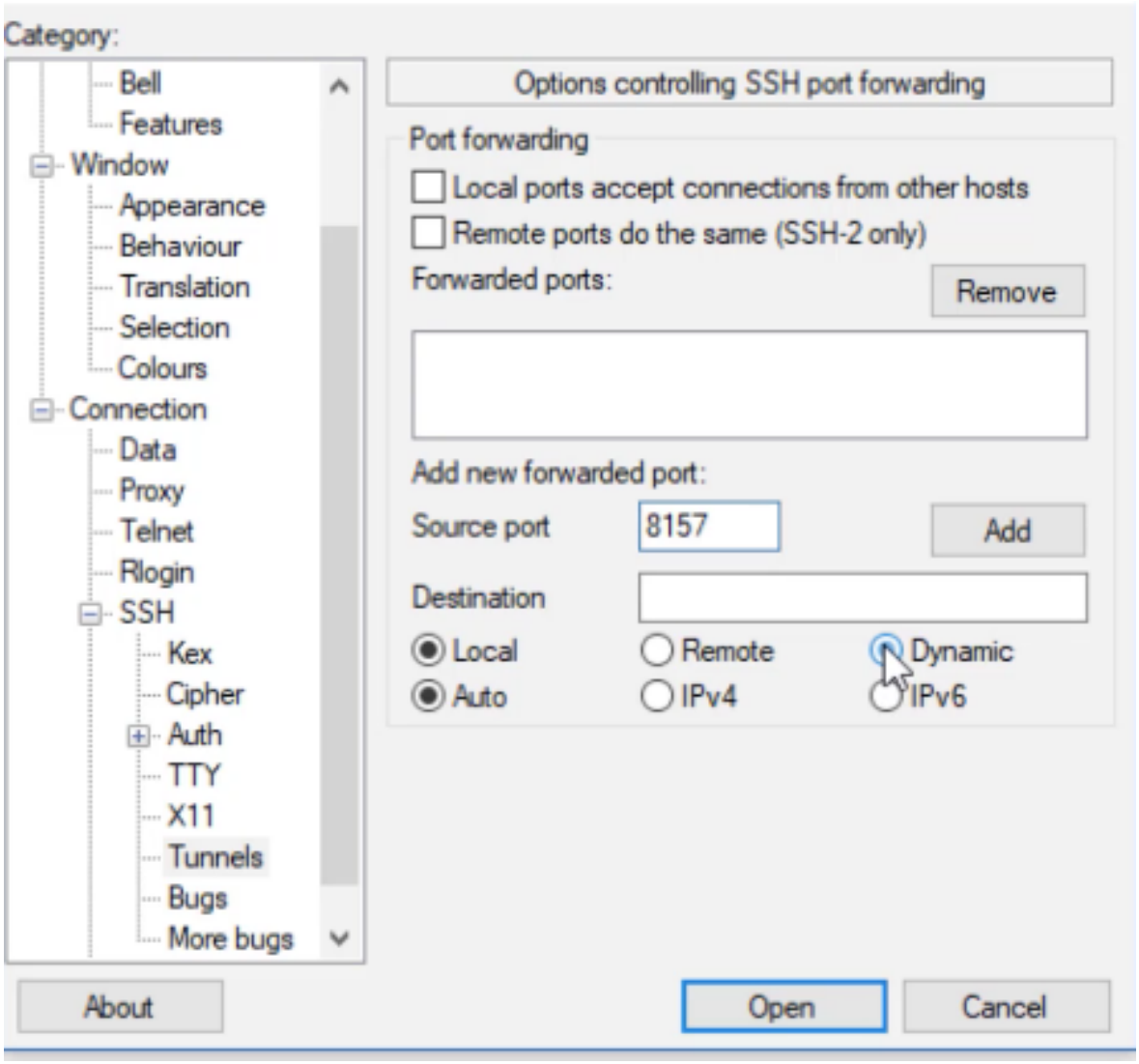
Click Edit > Add rules

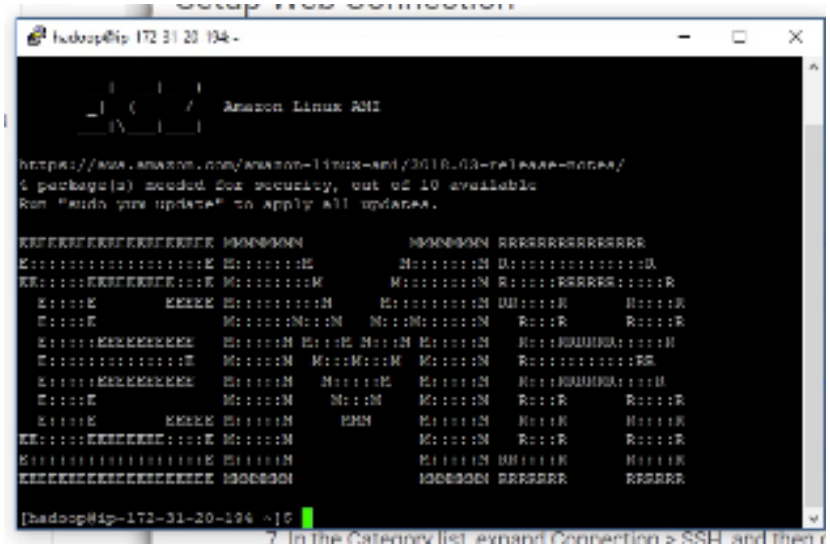


Open an SSH tunnel
If in windows, use Putty, and pass the private key



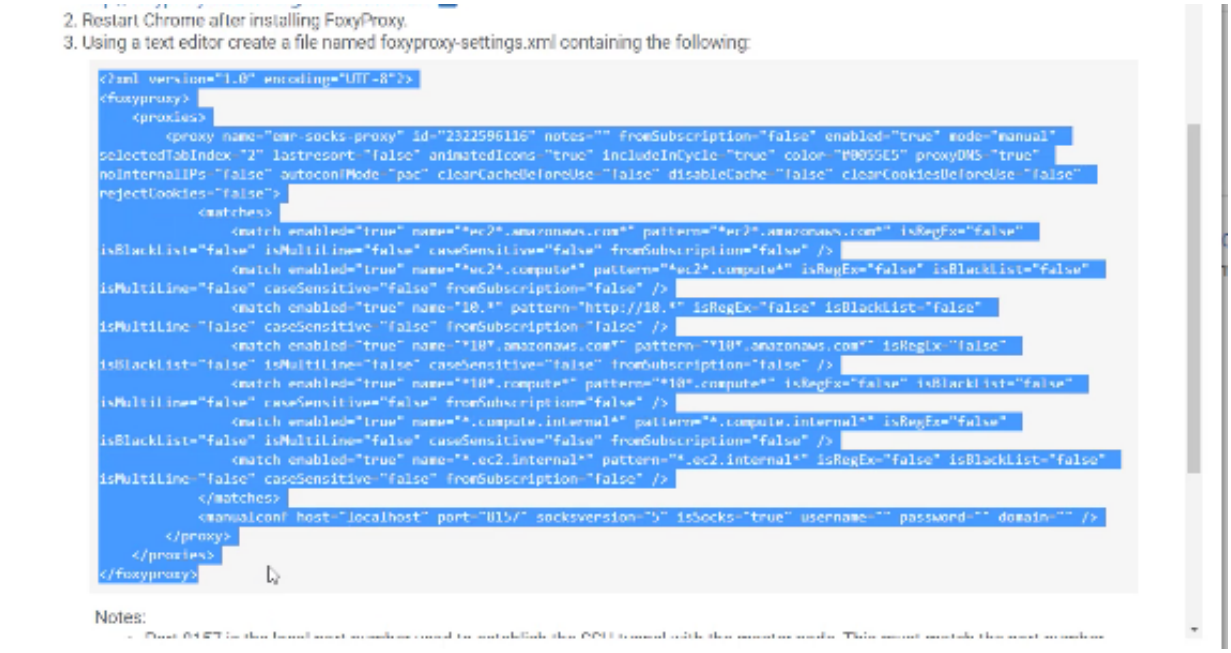
And set source port


Category:
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Cipher
Auth
TTY
X11
Tunnels
Bugs
More bugs
Options controlling SSH port forwarding
Port forwarding
Local ports accept connections from other hosts
Remote ports do the same (SSH-2 only)
Forwarded ports:
Remove
Add new forwarded port:
Source port
8157
Add
Destination
Local
Remote
Dynamic
Auto
IPv4
IPv6
About
Open
Cancel


Amazon Linux AMI

We still need a proxy management tool for the browser => foxy proxy
Chrome > more tools > extensions > foxy proxy basics

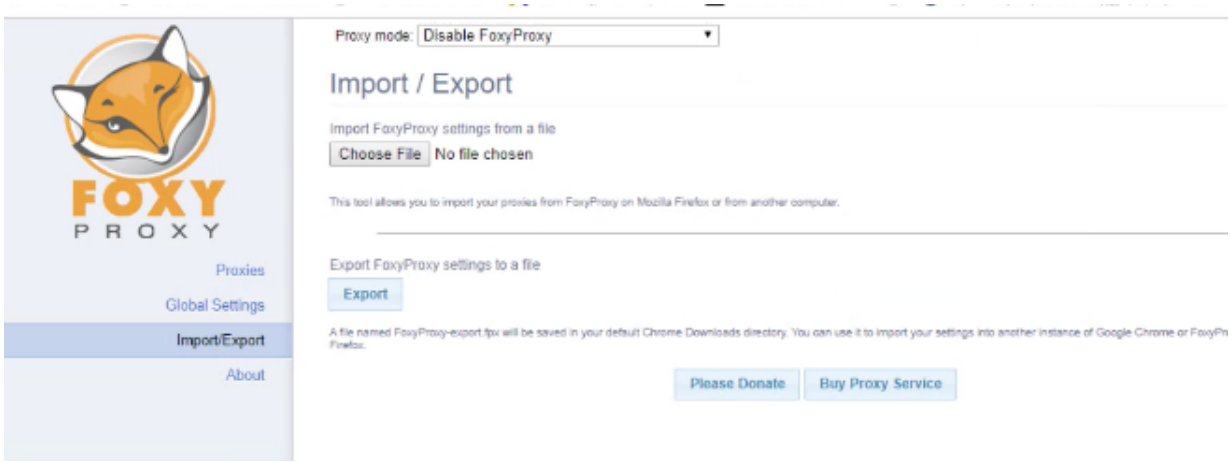
Copy this into a file for foxy proxy


2. Restart Chrome after installing FoxyProxy.
3. Using a text editor create a file named foxyproxy-settings.xml containing the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<foxyproxy>
    <proxies>
        <proxy name="emr-socks-proxy" id="2322596116" notes="" fromSubscription="false" enabled="true" mode="manual"
selectedTabIndex="2" lastresort="false" animatedIcons="true" includeInCycle="true" color="#0055E5" proxyDNS="true"
noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse="false" disableCache="false" clearCookiesBeforeUse="false"
rejectCookies="false">
            <matches>
                <match enabled="true" name="*ec2*.amazonaws.com*" pattern="*ec2*.amazonaws.com*" isRegEx="false"
isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
                <match enabled="true" name="*ec2*.compute*" pattern="*ec2*.compute*" isRegEx="false" isBlackList="false"
isMultiLine="false" caseSensitive="false" fromSubscription="false" />
                <match enabled="true" name="10.*" pattern="http://10.*" isRegEx="false" isBlackList="false"
isMultiLine="false" caseSensitive="false" fromSubscription="false" />
                <match enabled="true" name="*10*.amazonaws.com*" pattern="*10*.amazonaws.com*" isRegEx="false"
isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
                <match enabled="true" name="*10*.compute*" pattern="*10*.compute*" isRegEx="false" isBlackList="false"
isMultiLine="false" caseSensitive="false" fromSubscription="false" />
                <match enabled="true" name="*.compute.internal*" pattern="*.compute.internal*" isRegEx="false"
isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
                <match enabled="true" name="*.ec2.internal*" pattern="*.ec2.internal*" isRegEx="false" isBlackList="false"
isMultiLine="false" caseSensitive="false" fromSubscription="false" />
            </matches>
            <manualconf host="localhost" port="8157" socksversion="5" isSocks="true" username="" password="" domain="" />
        </proxy>
    </proxies>
</foxyproxy>
```

Notes:

Now need foxyproy to use it

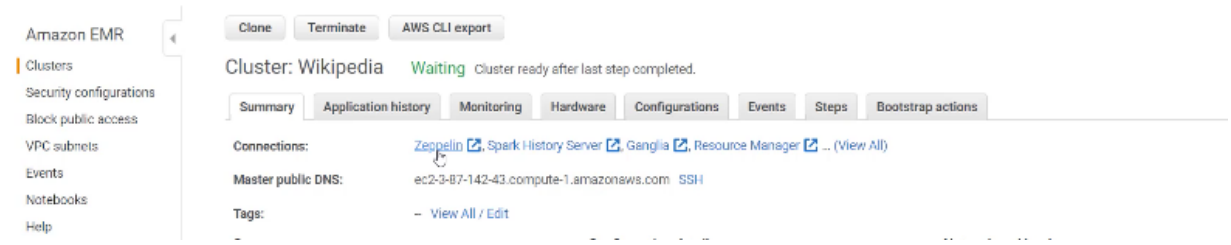


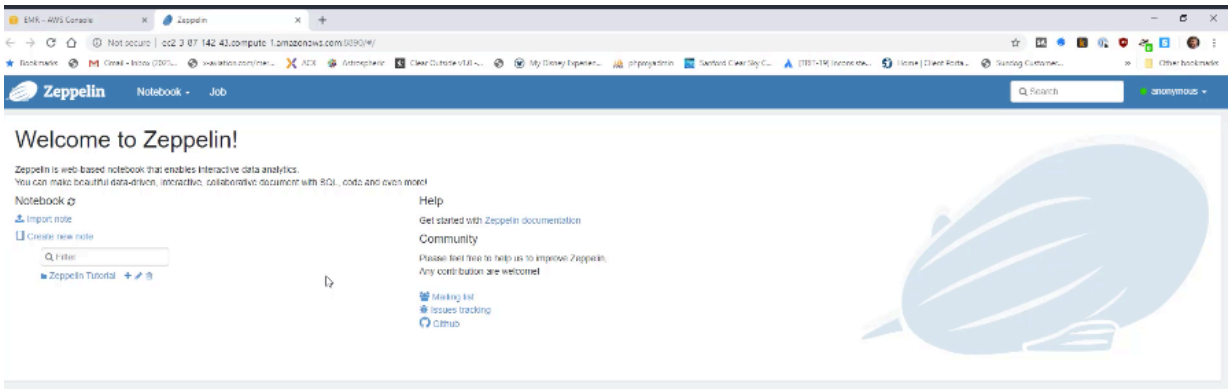
Notes:

- Port 8157 is the local port number used to establish the SSH tunnel with the master node. This must match the port number you used in PuTTY or terminal.
- The *ec2*.amazonaws.com* pattern matches the public DNS name of clusters in the us-east-1 region.
- The *ec2*.compute* pattern matches the public DNS name of clusters in all other regions.
- The 10.*pattern provides access to the JobTracker log files in Hadoop 1.x. Alter this filter if it conflicts with your network access plan.

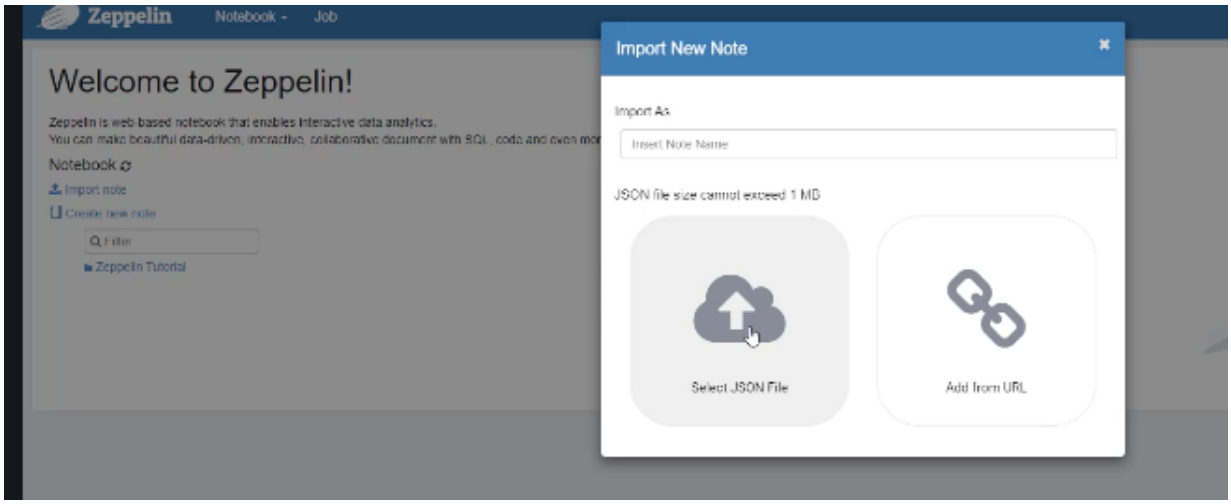
4. Click on the FoxyProxy icon in the toolbar and select Options.
5. Click Import/Export.
6. Click Choose File, select foxyproxy-settings.xml, and click Open.
7. In the Import FoxyProxy Settings dialog, click Add.
8. At the top of the page, for Proxy mode, choose Use proxies based on their pre-defined patterns and priorities
9. To open the web interfaces, in your browser's address bar, type `master-public-dns` followed by the port number or URL.

Now we have the Zepelin link that is activated:

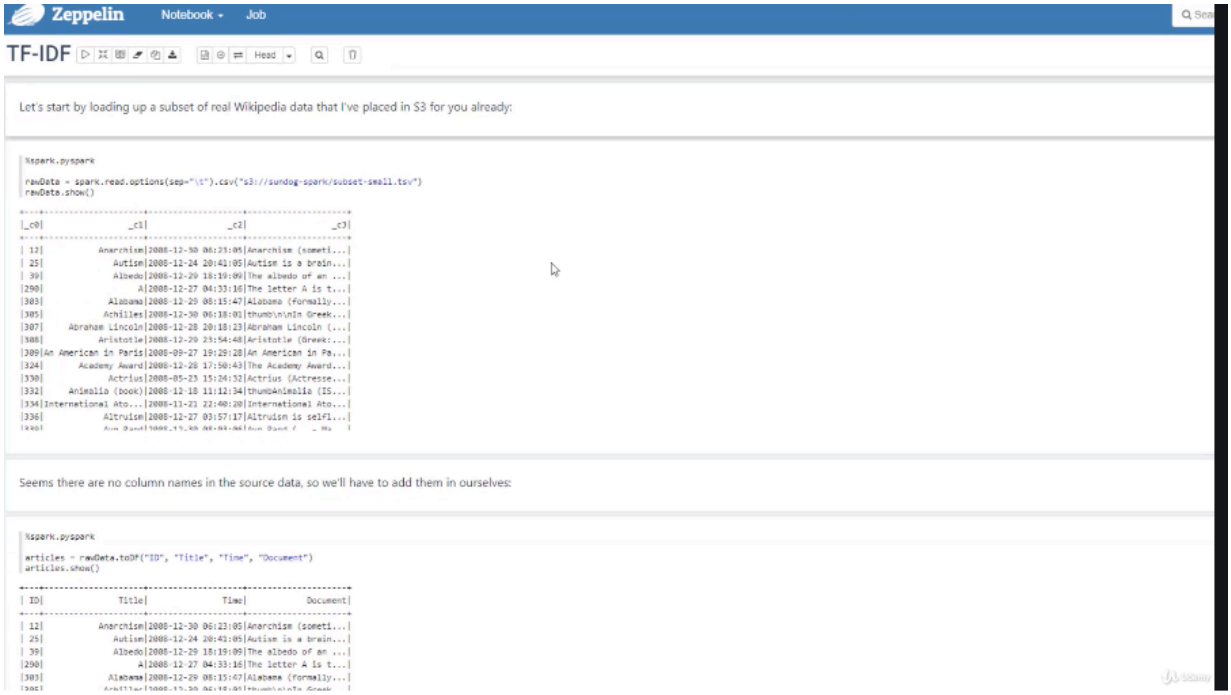




Import note/json from course materials (TF-IDF.json)



We now have a zepelin notebook that can be run on apache spark



TF-IDF

Let's start by loading up a subset of real Wikipedia data that I've placed in S3 for you already:

```
%spark.pyspark

rawData = spark.read.options(sep="\t").csv("s3://sundog-spark/subset-small.tsv")
rawData.show()
```

```
+---+--------------------+------------------+--------------------+
|_c0|                 _c1|               _c2|                 _c3|
+---+--------------------+------------------+--------------------+
| 12|           Anarchism|2008-12-30 06:23:05|Anarchism (someti...|
| 25|              Autism|2008-12-24 20:41:05|Autism is a brain...|
| 39|              Albedo|2008-12-29 18:19:09|The albedo of an ...|
|290|                   A|2008-12-27 04:33:16|The letter A is t...|
|303|             Alabama|2008-12-29 08:15:47|Alabama (formally...|
|305|            Achilles|2008-12-30 06:18:01|thumb\n\nIn Greek...|
|307|      Abraham Lincoln|2008-12-28 20:18:23|Abraham Lincoln (...|
|308|           Aristotle|2008-12-29 23:54:48|Aristotle (Greek:...|
|309|An American in Paris|2008-09-27 19:29:28|An American in Pa...|
|324|       Academy Award|2008-12-28 17:50:43|The Academy Award...|
|330|             Actrius|2008-05-23 15:24:32|Actrius (Actresse...|
|332|     Animalia (book)|2008-12-18 11:12:34|thumbAnimalia (IS...|
|334|International Ato...|2008-11-21 22:40:20|International Ato...|
|336|             Altruism|2008-12-27 03:57:17|Altruism is selfl...|
|339|           Ayn Rand|2008-12-30 08:03:06|Ayn Rand (  Ma  |
```

Add column names

Seems there are no column names in the source data, so we'll have to add them in ourselves:

```
%spark.pyspark

articles = rawData.toDF("ID", "Title", "Time", "Document")
articles.show()
```

```
+---+--------------------+------------------+--------------------+
| ID|               Title|              Time|            Document|
+---+--------------------+------------------+--------------------+
| 12|           Anarchism|2008-12-30 06:23:05|Anarchism (someti...|
| 25|              Autism|2008-12-24 20:41:05|Autism is a brain...|
| 39|              Albedo|2008-12-29 18:19:09|The albedo of an ...|
|290|                   A|2008-12-27 04:33:16|The letter A is t...|
|303|             Alabama|2008-12-29 08:15:47|Alabama (formally...|
|305|            Achilles|2008-12-30 06:18:01|thumb\n\nIn Greek...|
|307|      Abraham Lincoln|2008-12-28 20:18:23|Abraham Lincoln (...|
|308|           Aristotle|2008-12-29 23:54:48|Aristotle (Greek:...|
|309|An American in Paris|2008-09-27 19:29:28|An American in Pa...|
|324|       Academy Award|2008-12-28 17:50:43|The Academy Award...|
```

Remove null documents

Next we need to "clean" our data. We know TF/IDF can't handle null documents, so let's start by at least checking for that:

```
%spark.pyspark

articles.filter(articles.Document.isNull()).count()
```

```
1
```

Looks like there is one article that contains a null document. As it's only one and it's pretty clearly corrupt, we can just drop it and call it a day:

+ Add Paragraph

```
%spark.pyspark

cleanedArticles = articles.filter(articles.Document.isNotNull())

cleanedArticles.filter(cleanedArticles.Document.isNull()).count()
```

```
0
```

Split every documents into arrays of worlds, and map them to hash values

TF/IDF wants numbers, not words. So we need to pre-process our data before we can run any fun algorithms on it. We'll first tokenize the articles to split up them up in
words will be stored in a sparse vector that is now a numeric representation of the words in each article.

```
%spark.pyspark

from pyspark.ml.feature import HashingTF, IDF, Tokenizer

tokenizer = Tokenizer(inputCol="Document", outputCol="words")
wordsData = tokenizer.transform(cleanedArticles)

hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures")
featurizedData = hashingTF.transform(wordsData)
featurizedData.show()
```

```
+---+--------------------+--------------------+--------------------+--------------------+--------------------+
| ID|               Title|                Time|            Document|               words|         rawFeatures|
+---+--------------------+--------------------+--------------------+--------------------+--------------------+
| 12|          Anarchism|2008-12-30 06:23:05|Anarchism (someti...|[anarchism, (some...|(262144,[170,619,...|
| 25|             Autism|2008-12-24 20:41:05|Autism is a brain...|[autism, is, a, b...|(262144,[29,405,5...|
| 39|             Albedo|2008-12-29 18:19:09|The albedo of an ...|[the, albedo, of,...|(262144,[170,373,...|
|290|                   A|2008-12-27 04:33:16|The letter A is t...|[the, letter, a, ...|(262144,[4155,478...|
|303|            Alabama|2008-12-29 08:15:47|Alabama (formally...|[alabama, (formal...|(262144,[115,322,...|
|305|           Achilles|2008-12-30 06:18:01|thumb\n\nIn Greek...|[thumb\n\nin, gre...|(262144,[112,170,...|
|307|    Abraham Lincoln|2008-12-28 20:18:23|Abraham Lincoln (...|[abraham, lincoln...|(262144,[115,440,...|
|308|          Aristotle|2008-12-29 23:54:48|Aristotle (Greek:...|[aristotle, (gree...|(262144,[170,232,...|
```

We now have 2 more features:
- words: array of words that are normalized
- rawFeatures: hash the words into numerical values => sparse vector

Now using TF-IDF => see how relevant is a word in a document

That hashing operation basically computed term frequencies for us by storing how often each hashed word occurred in each article. So we have TF, but we want TF/IDF for every term in every document for us. We'll store these final scores in a new column called "features", which is a sparse vector containing TF/IDF scores for each featu

```
%spark.pyspark

idf = IDF(inputCol="rawFeatures", outputCol="features")
idfModel = idf.fit(featurizedData)
rescaledData = idfModel.transform(featurizedData)
```

```
%spark.pyspark

rescaledData.show()
```

```
+---+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| ID|               Title|                Time|            Document|               words|         rawFeatures|            features|
+---+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| 12|          Anarchism|2008-12-30 06:23:05|Anarchism (someti...|[anarchism, (some...|(262144,[170,619,...|(262144,[170,619,...|
| 25|             Autism|2008-12-24 20:41:05|Autism is a brain...|[autism, is, a, b...|(262144,[29,405,5...|(262144,[29,405,5...|
| 39|             Albedo|2008-12-29 18:19:09|The albedo of an ...|[the, albedo, of,...|(262144,[170,373,...|(262144,[170,373,...|
|290|                   A|2008-12-27 04:33:16|The letter A is t...|[the, letter, a, ...|(262144,[4155,478...|(262144,[4155,478...|
|303|             Alabama|2008-12-29 08:15:47|Alabama (formally...|[alabama, (formal...|(262144,[115,322,...|(262144,[115,322,...|
|305|            Achilles|2008-12-30 06:18:01|thumb\n\nIn Greek...|[thumb\n\nin, gre...|(262144,[112,170,...|(262144,[112,170,...|
|307|    Abraham Lincoln|2008-12-28 20:18:23|Abraham Lincoln (...|[abraham, lincoln...|(262144,[115,440,...|(262144,[115,440,...|
|308|           Aristotle|2008-12-29 23:54:48|Aristotle (Greek:...|[aristotle, (gree...|(262144,[170,232,...|(262144,[170,232,...|
```

6 people have written a note here.

So, let's use this to do a "search" for the term "Gettsburg." Again, we need numbers, not words - so the first thing we need to do is figure out what hash value "gettysbu

```
%spark.pyspark

from pyspark.sql.types import *

schema = StructType([StructField("words", ArrayType(StringType()))])

df = spark.createDataFrame([[["gettysburg"]]], schema).toDF("words")
df.show()

gettysburg = hashingTF.transform(df)
gettysburg.show()

featureVec = gettysburg.select('rawFeatures').collect()
print(featureVec)

gettysburgID = int(featureVec[0].rawFeatures.indices[0])
print(gettysburgID)
```

```
+------------+
|       words|
+------------+
|[gettysburg]|
+------------+
```

OK, we have the magic number that represents "Gettysburg." Now we can add another column - we'll call it "score" - that just extracts the TF/IDF score for Gettysburg f

```
%spark.pyspark


from pyspark.sql.types import FloatType
from pyspark.sql.functions import udf

termExtractor = udf(lambda x: float(x[gettysburgID]), FloatType())

gettysburgDF = rescaledData.withColumn('score', termExtractor(rescaledData.features))
gettysburgDF.show()
```

```
+---+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+---------+
| ID|               Title|                Time|            Document|               words|         rawFeatures|            features|    score|
+---+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+---------+
| 12|          Anarchism|2008-12-30 06:23:05|Anarchism (someti...|[anarchism, (some...|(262144,[170,619,...|(262144,[170,619,...|      0.0|
| 25|             Autism|2008-12-24 20:41:05|Autism is a brain...|[autism, is, a, b...|(262144,[29,405,5...|(262144,[29,405,5...|      0.0|
| 39|             Albedo|2008-12-29 18:19:09|The albedo of an ...|[the, albedo, of,...|(262144,[170,373,...|(262144,[170,373,...|      0.0|
|290|                   A|2008-12-27 04:33:16|The letter A is t...|[the, letter, a, ...|(262144,[4155,478...|(262144,[4155,478...|      0.0|
|303|             Alabama|2008-12-29 08:15:47|Alabama (formally...|[alabama, (formal...|(262144,[115,322,...|(262144,[115,322,...|      0.0|
|305|            Achilles|2008-12-30 06:18:01|thumb\n\nIn Greek...|[thumb\n\nin, gre...|(262144,[112,170,...|(262144,[112,170,...|      0.0|
|307|    Abraham Lincoln|2008-12-28 20:18:23|Abraham Lincoln (...|[abraham, lincoln...|(262144,[115,440,...|(262144,[115,440,...|33.128765|
|308|           Aristotle|2008-12-29 23:54:48|Aristotle (Greek:...|[aristotle, (gree...|(262144,[170,232,...|(262144,[170,232,...|      0.0|
|309|An American in Paris|2008-09-27 19:29:28|An American in Pa...|[an, american, in...|(262144,[1441,236...|(262144,[1441,236...|      0.0|
|324|      Academy Award|2008-12-28 17:50:43|The Academy Award...|[the, academy, aw...|(262144,[68,403,5...|(262144,[68,403,5...|      0.0|
|330|                    |                    |                    |                    |(262144,[4200,963...|(262144,[4200,963...|      0.0|
|332|                    |                    |                    |                    |(262144,[5125,524...|(262144,[5125,524...|      0.0|
|334|Inte                |                    |                    |                    |(262144,[925,1846...|(262144,[925,1846...|      0.0|
|336|           Altruism|2008-12-27 03:57:17|Altruism is selfl...|[altruism, is, se...|(262144,[813,914,...|(262144,[813,914,...|      0.0|
|339|                Ayn|2008-12-30 08:03:06|Ayn Rand (  - Ma...|[ayn, rand, (, , ...|(262144,[149,170,...|(262144,[149,170,...|      0.0|
```

6 people have written a note here.

Now, all we have to do is sort our articles by score and we'll see the most relevant articles for Gettysburg!

```
%spark.pyspark

sortedResults = gettysburgDF.filter("score > 0").orderBy('score', ascending=False).select('ID', 'Title', 'Document', 'score')
sortedResults.show(truncate=100)
```

```
+----+--------------------+----------------------------------------------------------------------------------------------------+---------+
|  ID|               Title|                                                                                            Document|    score|
+----+--------------------+----------------------------------------------------------------------------------------------------+---------+
| 307|    Abraham Lincoln|Abraham Lincoln (February 12, 1809 – April 15, 1865) was the sixteenth President of the United St...|33.128765|
|1135|    Abner Doubleday|Abner Doubleday (June 26, 1819 – January 26, 1893) was a career United States Army officer and Un...|27.607306|
| 863|American Civil War|The American Civil War (1861–1865), also known as the War Between the States and several other na...|16.564383|
+----+--------------------+----------------------------------------------------------------------------------------------------+---------+
```