CS522 – Big Data

Project report

Ganijon Rahimov [985565]

Contents:

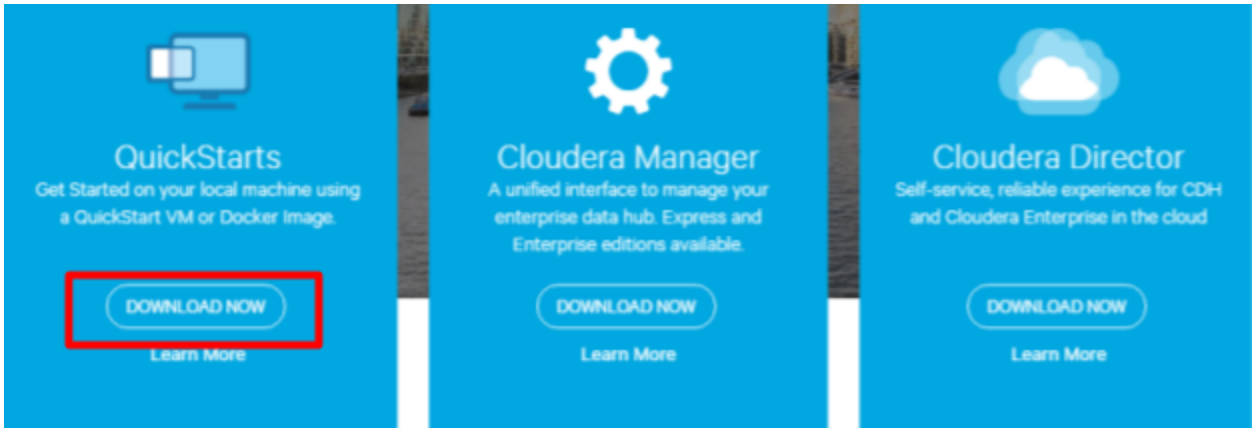
Part1. Setup Hadoop Single Node Cluster

Download VMWare Workstation Player

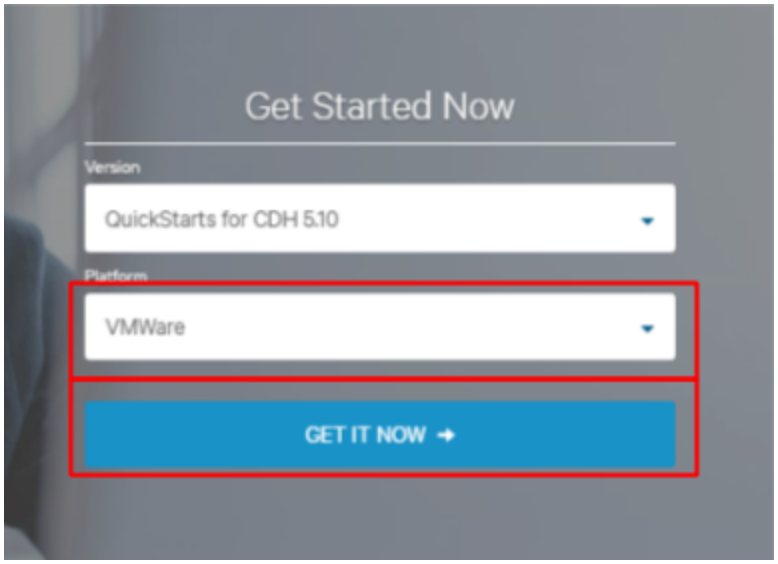
- Go to url www.vmware.com/go/downloadplayer
- Download the version for Windows 64-bit OS

Download Cloudera QuickStart VM

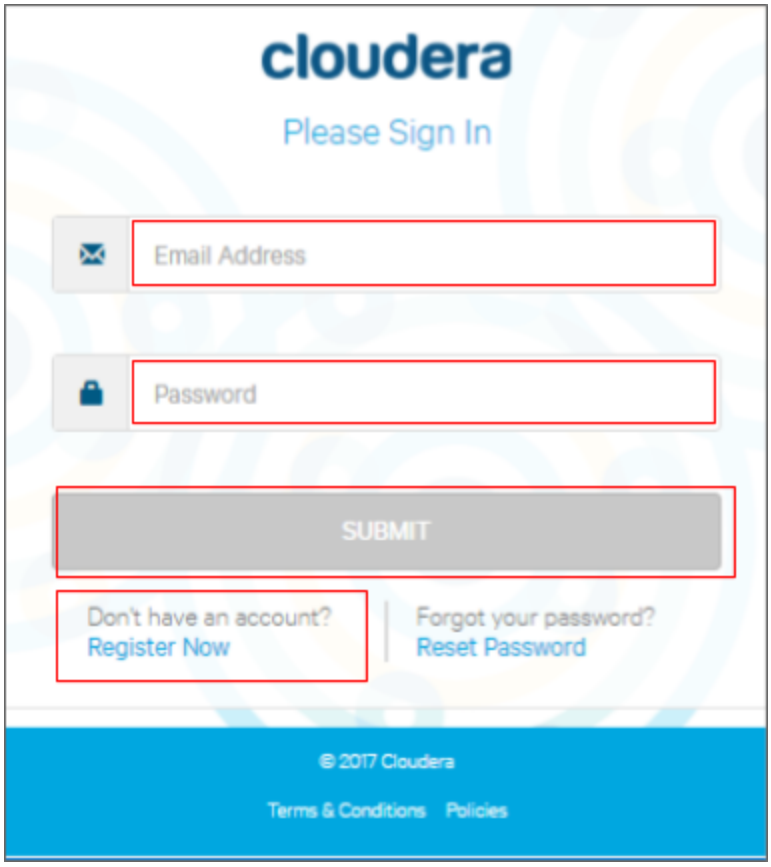
- Go to url https://www.cloudera.com/downloads.html
- Click on **Download Now** in **QuickStarts** section



- Select **VmWare** in Platforms list the click **Get it Now**



- Sign-in to to your account with clouder. Or, get an account through **Register Now**

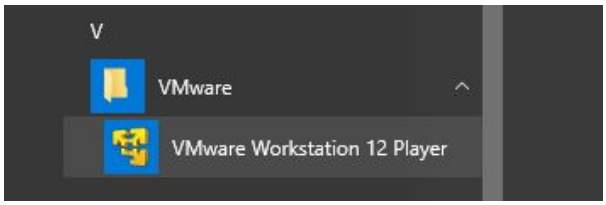


- Downloading starts automatically when sign-in is successful. The file is 4GB+.
- Unzip contents of downloaded file into a separate folder.

Setup VMWare Player

- Run installation wizard as Administrator

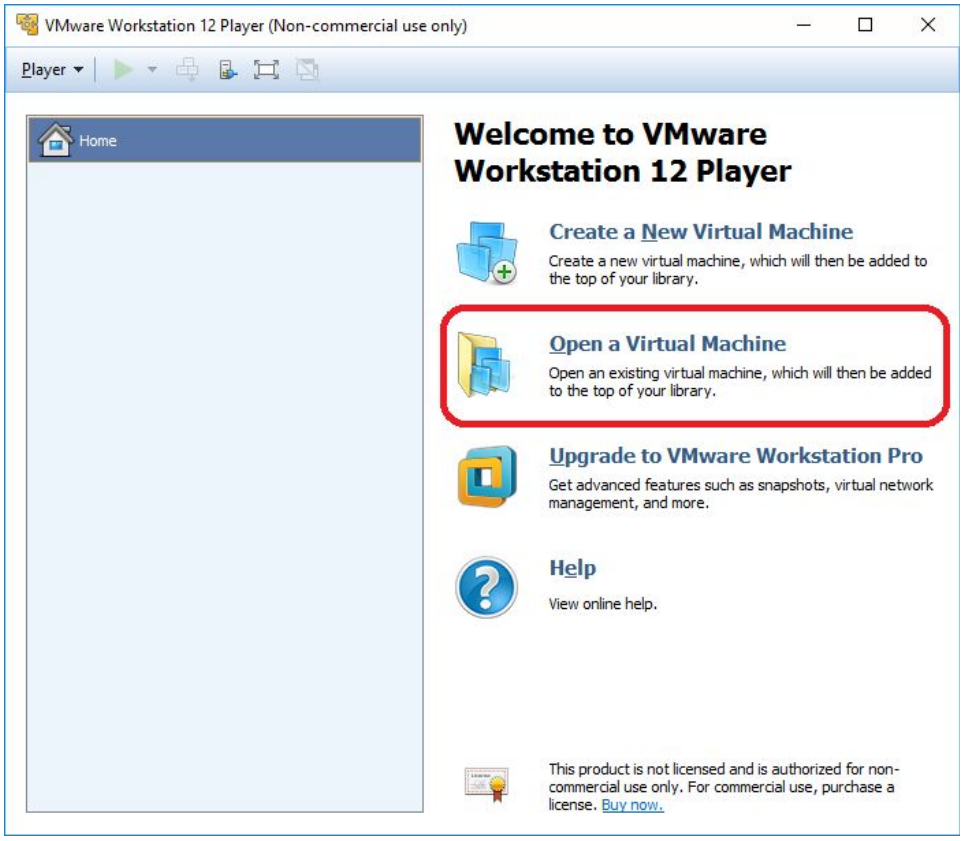
- Launch VmWare Player from Start menu



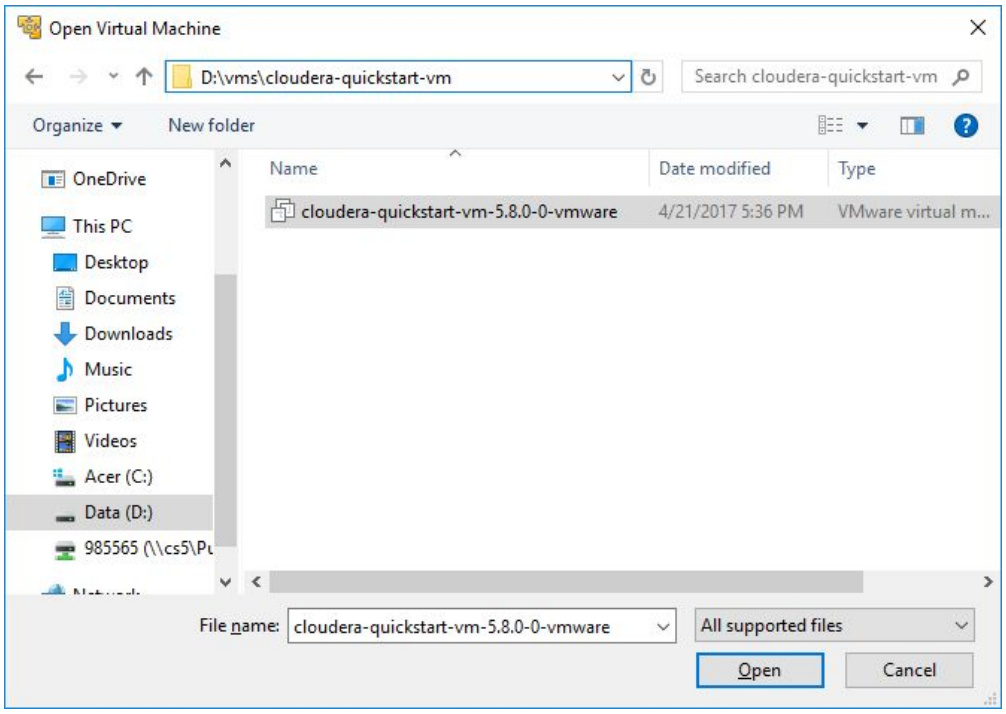
-
- Configuration of VmWare Player is not required

Setup Cloudera QuickStart VM

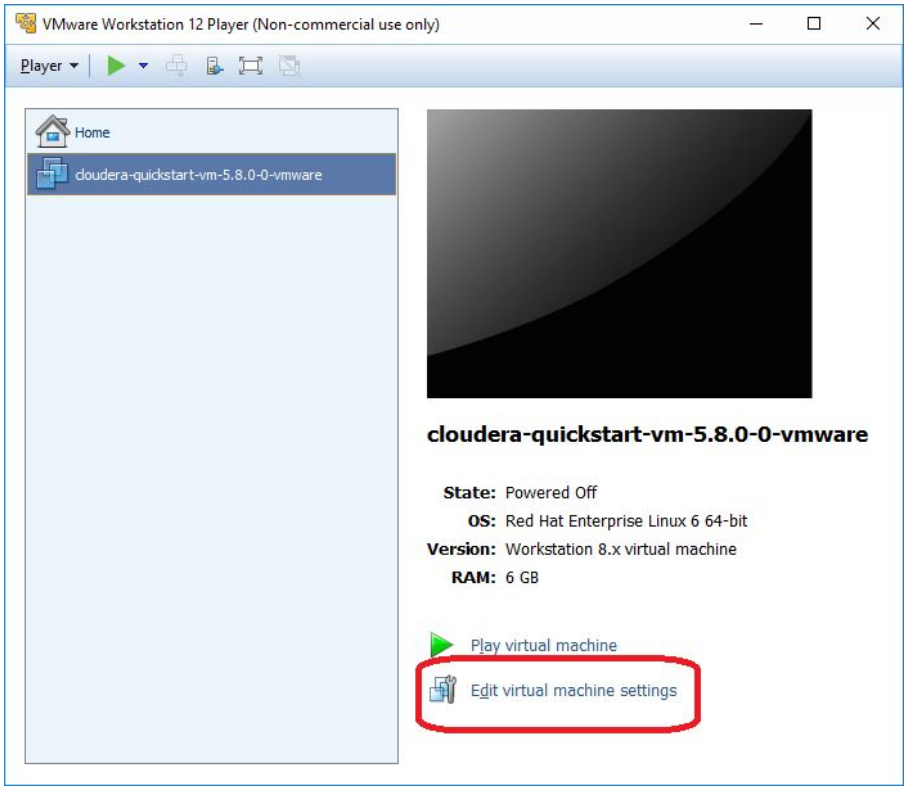
- Launch vmWare Workstation Player and click on **Open a Virtual Machine**



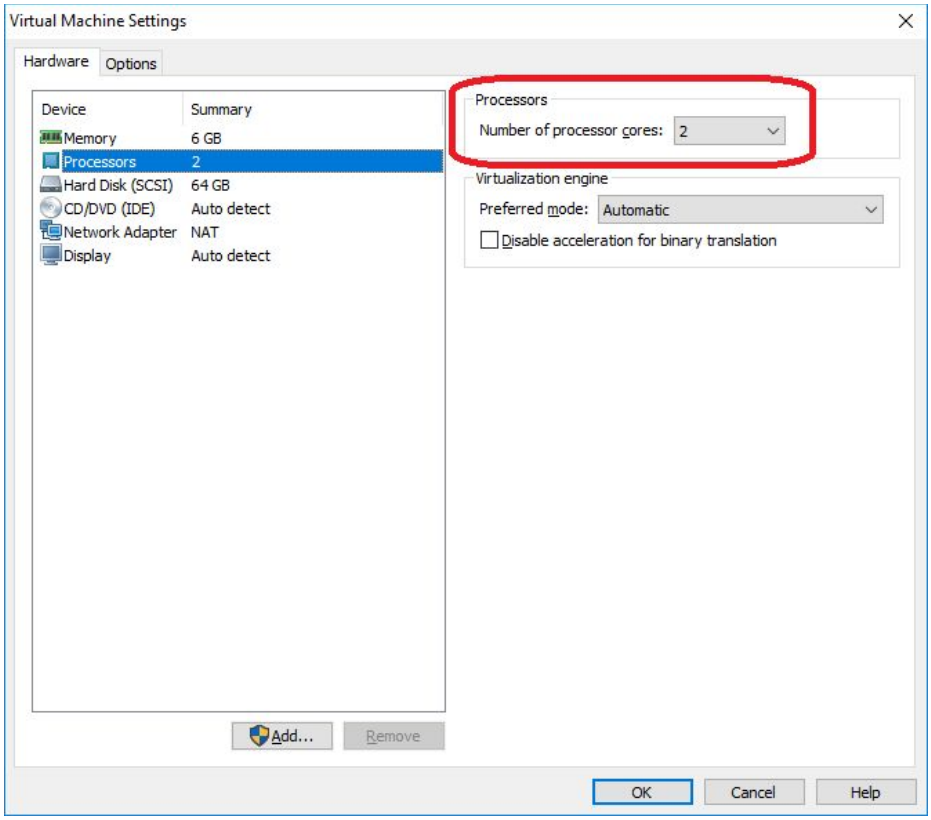
- Navigate to the folder and select QuickStart VM configuration file



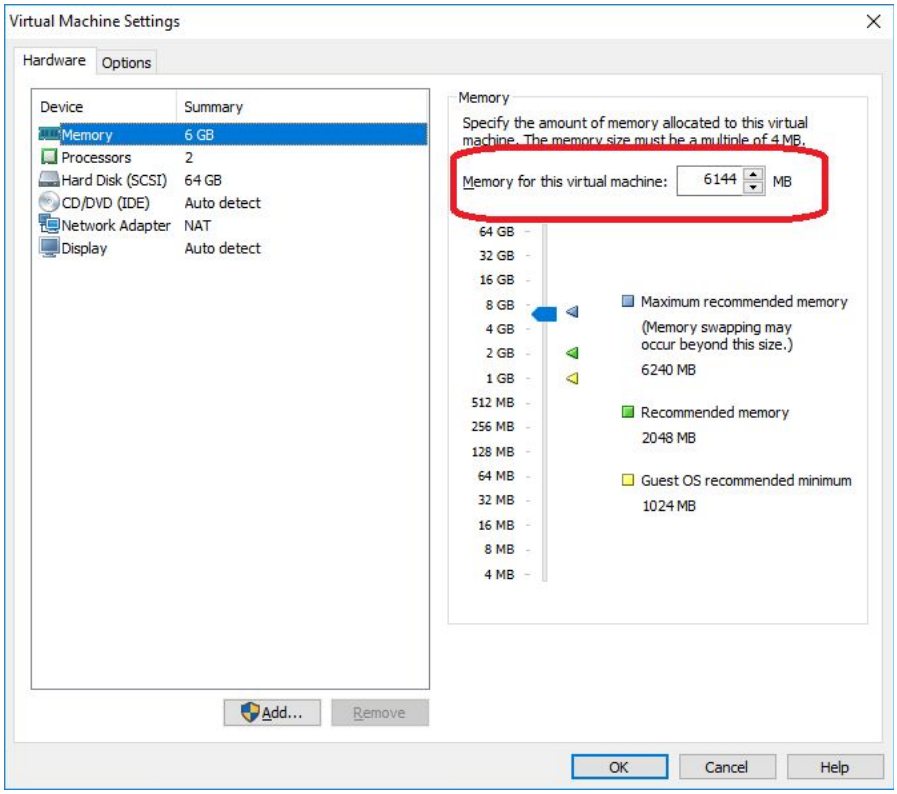
- To configure settings of VM select **Edit virtual machine settings**



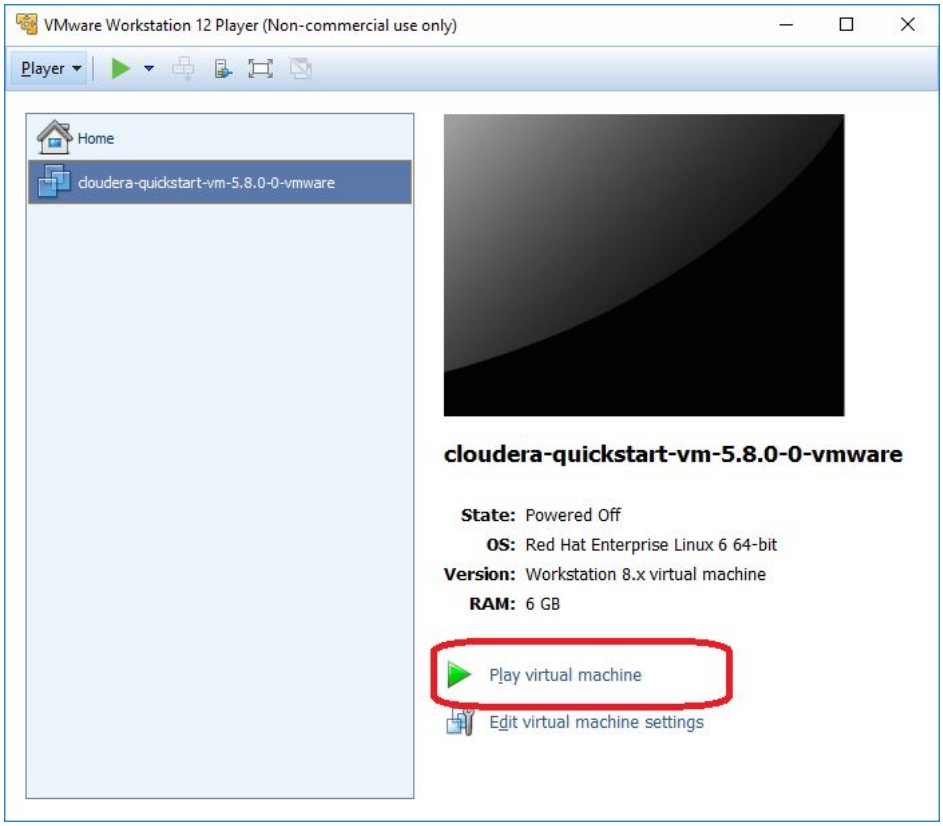
- Increase number of CPU cores for Quickstart VM

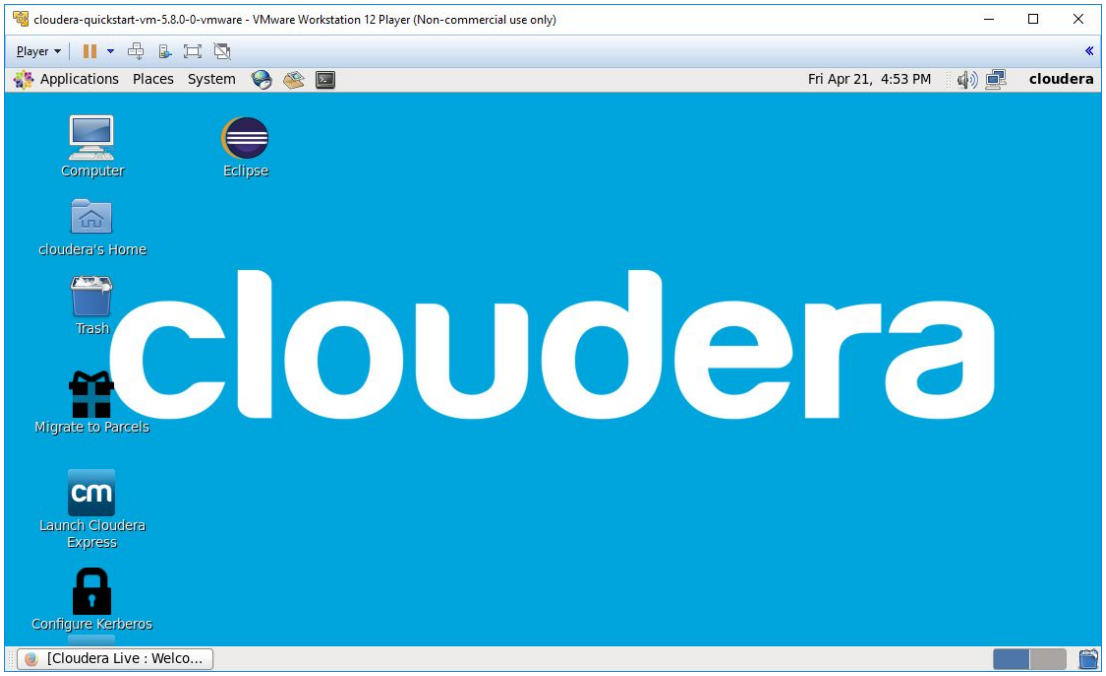


- Increase size of RAM for this VM



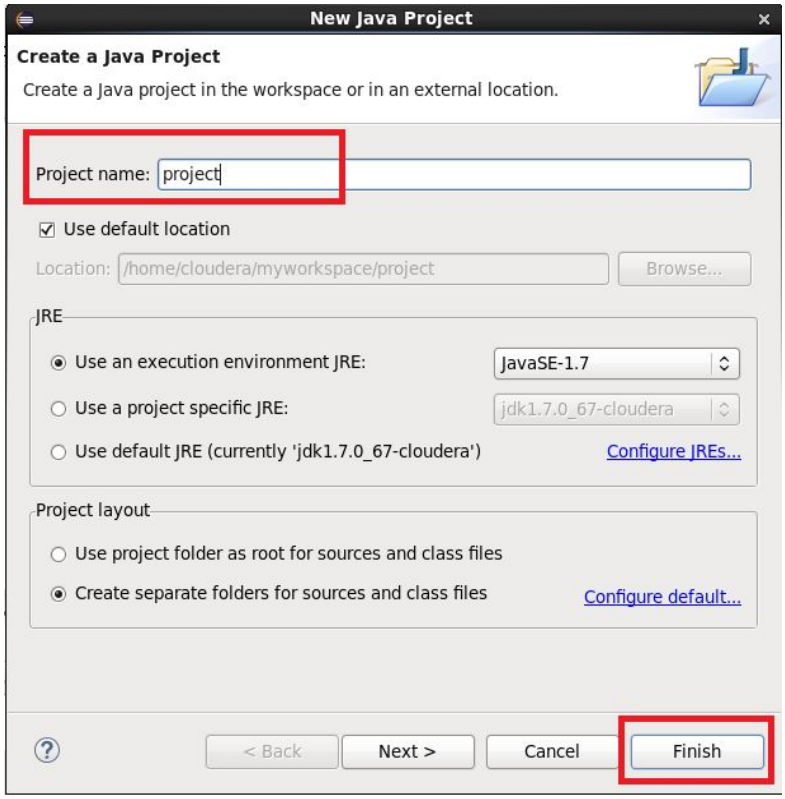
- Click on the **Play virtual machine** to boot the QuickStart VM



- QuickStart VM boots few minutes, press ESC for more details
- Finally, when system is ready you should see Cloudera QuickStart VM desktop
- 

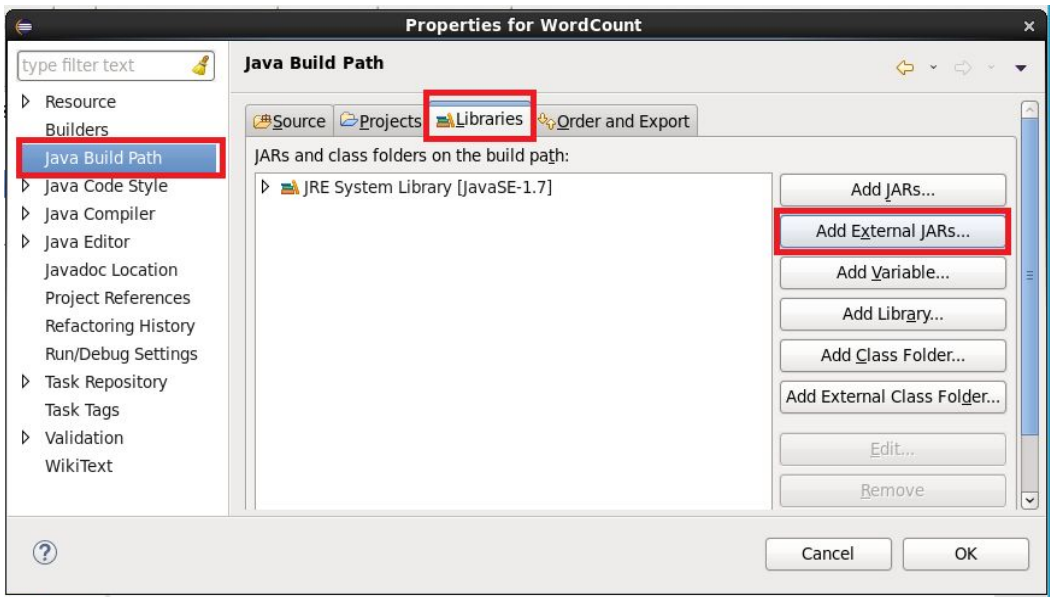
Test WordCount algorithm on Hadoop

- Open Eclipse and select **File** -> **New** -> **Java Project**. Give a name to the new project, i.e. WordCount and select **JavaSE-1.7** for JRE

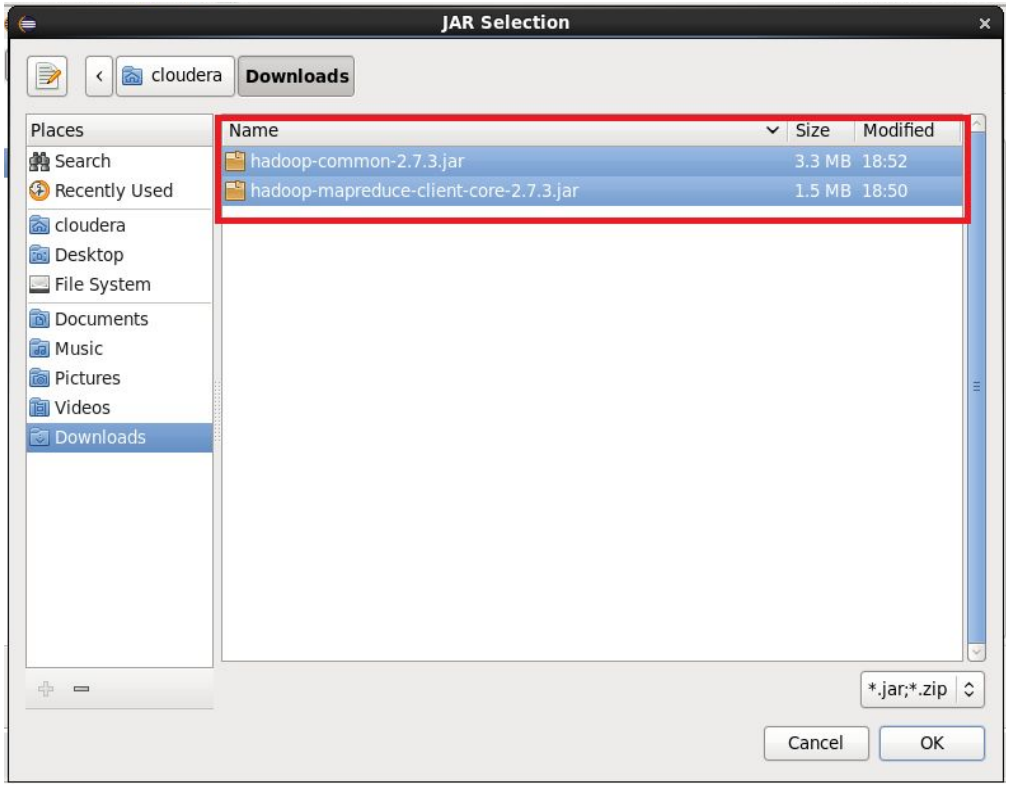


- **Download JAR** files (dependencies) from following urls:
  http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-common/2.7.3
  http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-client-core/2.7.3

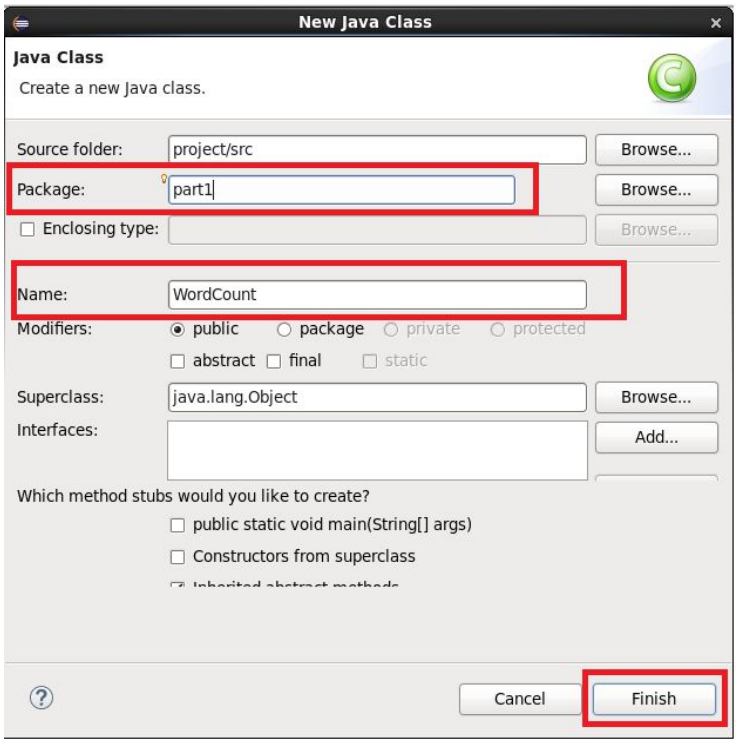
- To add dependencies to the project open properties window from menu **Project** -> **Properties.** Navigate through tabs **Java Build Path** and **Libraries** and click on **Add External JARs** button.



- From Jar Selection window select required JAR files

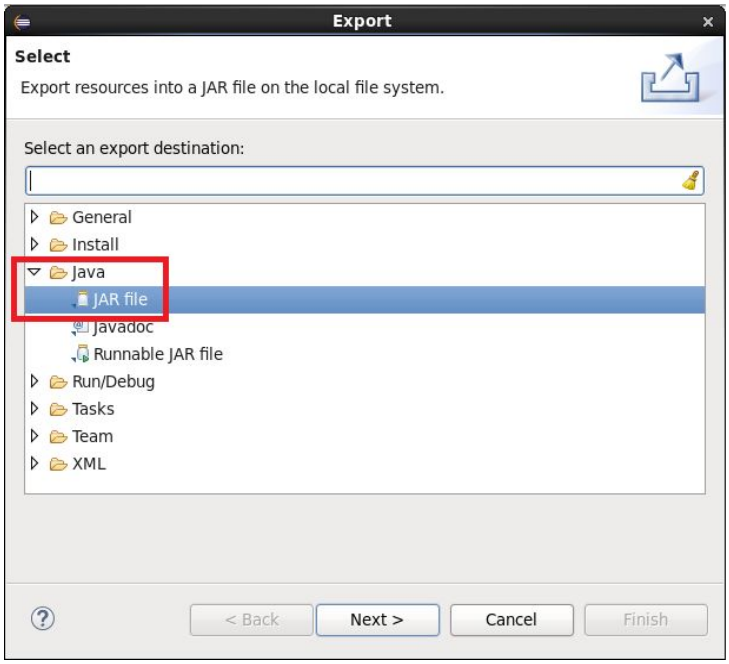


- Add new class from menu **File** -> **New** -> **Class**. In New Java Class window add **Package** name and class **Name** and press Finish button.

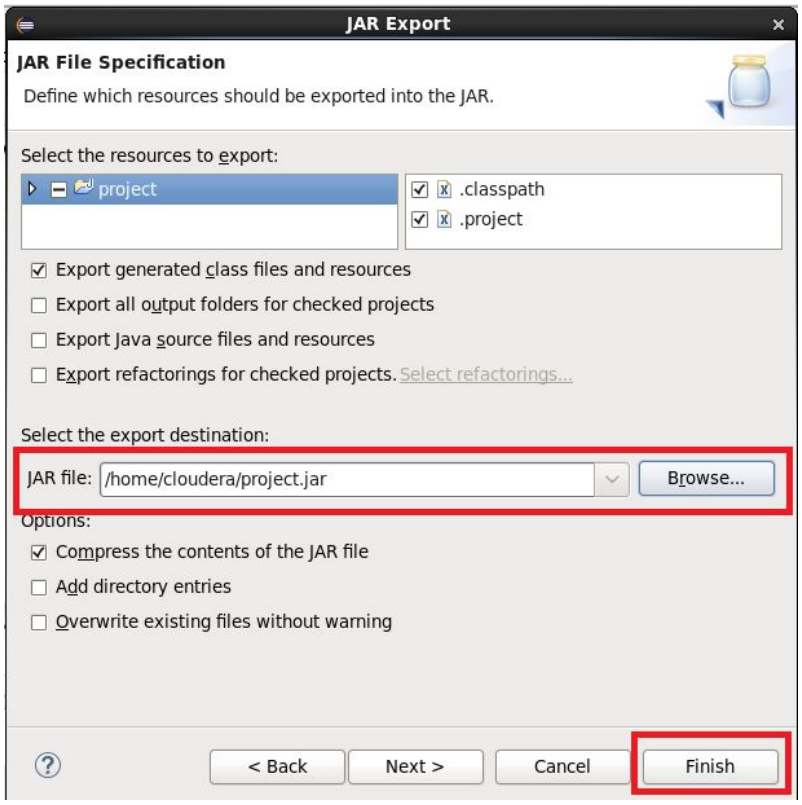


- Copy and paste the source code of WordCount example from https://wiki.apache.org/hadoop/WordCount.

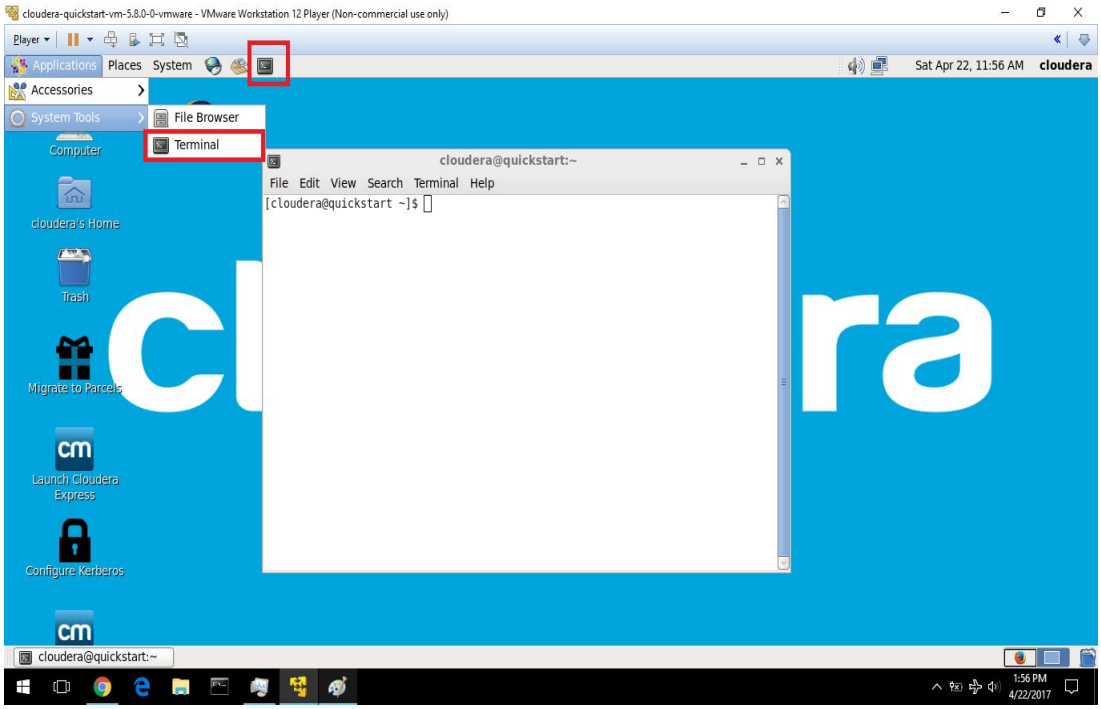
- Export WordCount.jar to test. Go to **File** -> **Export.** In the Export window will appear, choose **Java** -> **JAR file**. Click Next.



- Enter the JAR file full path, i.e. **/home/cloudera/wordcount.jar** and click **Finish**.



- Launch **Terminal (linux command line)** from menu bar or go to menu **Applications** -> **System** -> **Terminal**.



- Use following terminal commands to make input directory.

```
$ sudo su hdfs
$ hadoop fs -mkdir /user/cloudera
$ hadoop fs -chown cloudera /user/cloudera
$ exit
$ sudo su cloudera
$ hadoop fs -mkdir /user/cloudera/wordcount
$ hadoop fs -mkdir /user/cloudera/wordcount/input
```

- Create new file for input data using **cat**. Enter each input line. To finish press CTRL+C

```
$ cat > input.txt
B12 C31 D76 A12 B76 B12 D76 C31 A10 B12 D76
C31 D76 B12 A12 C31 D76 B12  A12 D76 A12 D76
^C
```

- Copy input file to input directory in HDFS

```
$ hadoop fs -put input.txt /user/cloudera/wordcount/input
```

- Launch the **wordcount.jar** by providing HDFS directories for input and output.

```
$ hadoop jar project.jar part1.WordCount user/cloudera/wordcount/input
/user/cloudera/wordcount/output
```

- Execution of the job will be reported on the screen. Make sure that mapreduce job completed successfully.

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
92877085950_0001
17/04/22 09:47:57 INFO impl.YarnClientImpl: Submitted application application_14
92877085950_0001
17/04/22 09:47:57 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1492877085950_0001/
17/04/22 09:47:57 INFO mapreduce.Job: Running job: job_1492877085950_0001
17/04/22 09:48:59 INFO mapreduce.Job: Job job_1492877085950_0001 running in uber
 mode : false
17/04/22 09:49:00 INFO mapreduce.Job:  map 0% reduce 0%
17/04/22 09:49:54 INFO mapreduce.Job:  map 100% reduce 0%
17/04/22 09:50:09 INFO mapreduce.Job:  map 100% reduce 100%
17/04/22 09:50:10 INFO mapreduce.Job: Job job_1492877085950_0001 completed succe
ssfully
17/04/22 09:50:11 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=226
                FILE: Number of bytes written=233023
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=220
                HDFS: Number of bytes written=36
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
```

- Show output results in terminal window using following command

```
$ hadoop fs -cat /user/cloudera/wordcount/output/*
```

- The Part 1 is finished if output results match the highlighted output

```
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/wordcount/output/*
A10     1
A12     4
B12     5
B76     1
C31     4
D76     7
[cloudera@quickstart ~]$
```

Part 2. Implement Pairs algorithm to compute relative frequencies

Pseudo-code

```
class Mapper

      method map(docid id, doc d)
            for all term w in doc d do
                  for all term u in N(w) do
                        Emit(pair(w,u), 1)
                        Emit(pair(w,*), 1)
```

```
class Reducer

      method initialize
            total = 0;

      method reduce(pair p, count [c1, c2, ...])
            sum = sum([c1, c2, ...])
            if(p.right == '*')
                  total = sum
            else
                  Emit(p, sum/total)
```

Java classes

- PairsMapper.java

```
package part2;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class PairsMapper extends Mapper<LongWritable, Text, WordPair, IntWritable> {
        private final IntWritable one = new IntWritable(1);
        private final WordPair pair = new WordPair();

        @Override
        public void map(LongWritable key, Text values, Context context)
                        throws IOException, InterruptedException {
                List<String> words = Arrays.asList(values.toString().split("\\s"));
                int wordIndex = 0;
                for (String word : words) {
                        pair.setWord(word);
```

```
                        for (String neighbor : getNeighbors(words, wordIndex)) {
                                pair.setNeighbor(neighbor);
                                context.write(pair, one);
                                pair.setNeighbor("*");
                                context.write(pair, one);
                        }
                        wordIndex++;
                }
        }
        private List<String> getNeighbors(List<String> words, int wordIndex) {
                List<String> neigbors = new ArrayList<>();
                for (int i = wordIndex + 1; i < words.size(); i++) {
                        if (words.get(wordIndex).equals(words.get(i))) {
                                break;
                        }
                        neigbors.add(words.get(i));
                }
                return neigbors;
        }
}
```

- PairReducer.java

```
package part2;

import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class PairsReducer extends Reducer<WordPair, IntWritable, WordPair, Text> {
        private int total;

        @Override
        protected void setup(Reducer<WordPair, IntWritable, WordPair, Text>.Context context)
                        throws IOException, InterruptedException {
                super.setup(context);
                total = 0;
        }

        @Override
        public void reduce(WordPair pair, Iterable<IntWritable> counts,
                        Context context) throws IOException, InterruptedException {
                int sum = sum(counts);
                if (pair.getNeighbor().equals(new Text("*"))) {
                        total = sum;
                } else {
                        String relativeFrequency = sum + "/" + total;
                        context.write(pair, new Text(relativeFrequency));
                }
        }
        private int sum(Iterable<IntWritable> values) {
                int sum = 0;
                for (IntWritable val : values) {
                        sum += val.get();
                }
                return sum;
        }
}
```

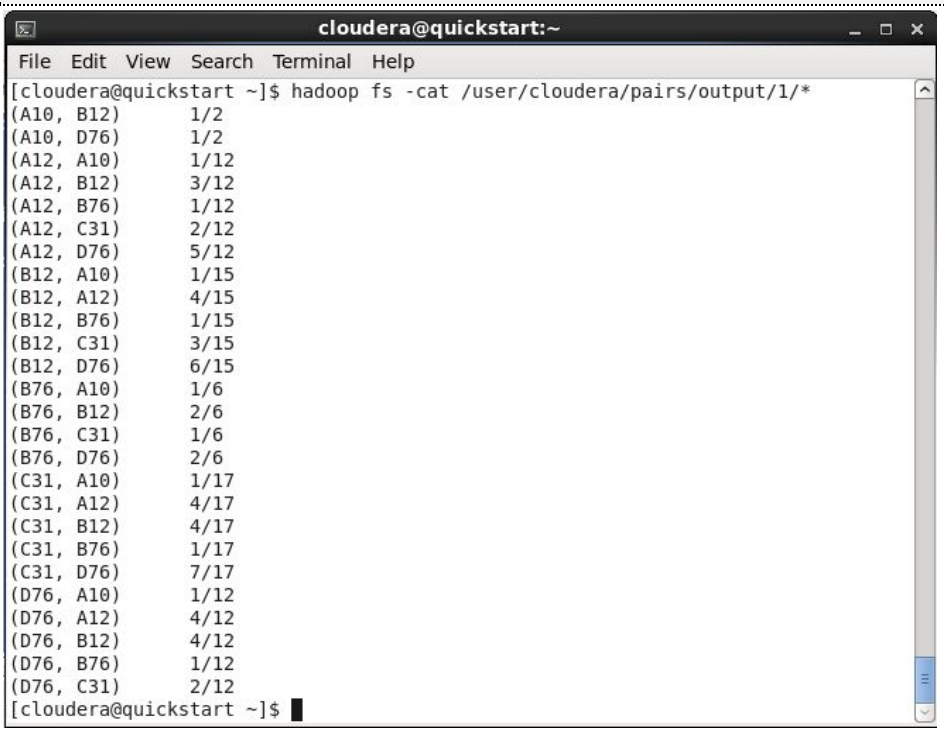
- WordPair.java

```
package part2;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;

public class WordPair implements WritableComparable<WordPair> {
	private Text word;
	private Text neighbor;
	public WordPair() {
		word = new Text();
		neighbor = new Text();
	}
	public Text getWord() {
		return word;
	}
	public void setWord(String word) {
		this.word.set(word);;
	}
	public Text getNeighbor() {
		return neighbor;
	}
	public void setNeighbor(String neighbor) {
		this.neighbor.set(neighbor);
	}
	@Override
	public void readFields(DataInput in) throws IOException {
		word.readFields(in);
		neighbor.readFields(in);
	}
	@Override
	public void write(DataOutput out) throws IOException {
		word.write(out);
		neighbor.write(out);
	}
	@Override
	public int compareTo(WordPair other) {
		int i = this.word.compareTo(other.getWord());
		if (0 == i)
			i = this.neighbor.compareTo(other.getNeighbor());
		return i;
	}
	@Override
	public int hashCode() {
		return 31 * word.hashCode() + 67 * neighbor.hashCode();
	}
	@Override
	public boolean equals(Object o) {
		boolean isEqual = false;
		if (o instanceof WordPair) {
			WordPair p = (WordPair) o;
			isEqual = word.equals(p.word) && neighbor.equals(p.neighbor);
		}
		return isEqual;
	}
	@Override
	public String toString() {
		return "(" + word + ", " + neighbor + ")";
	}
}
```

Output

- Once job completed successfully, display contents of output file:

```
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/pairs/output/1/*

(A10, B12)    1/2
(A10, D76)    1/2
(A12, A10)    1/12
(A12, B12)    3/12
(A12, B76)    1/12
(A12, C31)    2/12
(A12, D76)    5/12
(B12, A10)    1/15
(B12, A12)    4/15
(B12, B76)    1/15
(B12, C31)    3/15
(B12, D76)    6/15
(B76, A10)    1/6
(B76, B12)    2/6
(B76, C31)    1/6
(B76, D76)    2/6
(C31, A10)    1/17
(C31, A12)    4/17
(C31, B12)    4/17
(C31, B76)    1/17
(C31, D76)    7/17
(D76, A10)    1/12
(D76, A12)    4/12
(D76, B12)    4/12
(D76, B76)    1/12
(D76, C31)    2/12
```



Part 3. Implement Stripes algorithm to compute relative frequencies.

Pseudo-code

```
class Mapper

      method map(docid id, doc d)
            for all term w in doc d do
                  H = new AssociativeArray
                  for all term u in N(w) do
                        H{u} = H{u} + 1
                  Emit(term w, stripe H)
```

```
class Reducer

      method reduce(term w, stripes [H1,H2, ...])
            Hf = new AssociativeArray
            for all stripe H in stripes[H1, H2, ...] do
                  sum(Hf, H)              // element-wise sum
                  total = total(Hf)      // element-wise add
            Emit(term w, stripe Hf/total)
```

Java classes

- StripesMapper.java

```
package part3;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.MapWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class StripesMapper extends
               Mapper<LongWritable, Text, Text, Stripe> {
        private Stripe occurenceMap = new Stripe();
        @Override
        public void map(LongWritable key, Text line, Context context)
                       throws IOException, InterruptedException {
               List<String> words = Arrays.asList(line.toString().split("\\s"));
               int wordIndex = 0;
               for (String word : words) {
                       occurenceMap.clear();
                       for (String neighbor : getNeighbors(words, wordIndex)) {
```

```
                                IntWritable neighborCount = new IntWritable(1);
                                if (occurenceMap.containsKey(neighbor)) {
                                        neighborCount = (IntWritable) occurenceMap.get(neighbor);
                                        neighborCount.set(1 + neighborCount.get());
                                }
                                occurenceMap.put(new Text(neighbor), neighborCount);
                        }
                        wordIndex++;
                        context.write(new Text(word), occurenceMap);
                }
        }
        private List<String> getNeighbors(List<String> words, int wordIndex) {
                List<String> neighbors = new ArrayList<>();
                for (int i = wordIndex + 1; i < words.size(); i++) {
                        if (words.get(wordIndex).equals(words.get(i))) {
                                break;
                        }
                        neighbors.add(words.get(i));
                }
                return neighbors;
        }
}
```

- StripesReducer.java

```
package part3;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.mapreduce.Reducer;

public class StripesReducer extends Reducer<Text, Stripe, Text, Stripe> {

        @Override
        public void reduce(Text word, Iterable<Stripe> stripes, Context context)
                        throws IOException, InterruptedException {
                Stripe stripe = new Stripe();
                for (Stripe s : stripes) {
                        mergeStripes(stripe, s);
                }
                divideByTotal(stripe, getTotal(stripe));
                context.write(word, stripe);
        }

        private void mergeStripes(Stripe stripe, Stripe s) {
                for (Writable neighbor : s.keySet()) {
                        IntWritable neighborCount = (IntWritable) s.get(neighbor);
                        if (stripe.containsKey(neighbor)) {
                                IntWritable count = (IntWritable) stripe.get(neighbor);
                                neighborCount.set(neighborCount.get() + count.get());
                        }
                        stripe.put(neighbor, neighborCount);
                }
        }

        private void divideByTotal(Stripe stripe, int total) {
                Text newValue = new Text();
                IntWritable oldValue;
                for (Writable key : stripe.keySet()) {
                        oldValue = (IntWritable) stripe.get(key);
```

```
                        newValue.set(oldValue.toString() + "/" + String.valueOf(total));
                        stripe.put(key, newValue);
                }
        }

        private int getTotal(Stripe stripe) {
                int total = 0;
                for (Writable key : stripe.keySet())
                        total += ((IntWritable) stripe.get(key)).get();
                return total;
        }
}
```

- Stripe.java

```
package parts;

import org.apache.hadoop.io.MapWritable;
import org.apache.hadoop.io.Writable;

public class Stripe extends MapWritable {

        @Override
        public String toString() {
                if(isEmpty()) return "{ }";
                StringBuilder sb = new StringBuilder("{ ");
                for(Writable key: keySet()) {
                        sb.append(key.toString() + ": " + this.get(key));
                        sb.append(", ");
                }
                sb.replace(sb.length() - 2, sb.length(), " }");
                return sb.toString();
        }
}
```

Output

- Once job completed successfully, display contents of output file

```
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/stripes/output/1/*
A10    { B12: 1/2, D76: 1/2 }
A12    { B76: 2/10, A10: 2/10, B12: 2/10, D76: 2/10, C31: 2/10 }
B12    { B76: 3/13, A10: 3/13, A12: 3/13, D76: 3/13, C31: 3/13 }
B76    { A10: 1/4, B12: 1/4, D76: 1/4, C31: 1/4 }
C31    { B76: 4/13, A10: 4/13, A12: 4/13, B12: 4/13, D76: 4/13 }
D76    { B76: 2/12, A10: 2/12, A12: 2/12, B12: 2/12, C31: 2/12 }
```

cloudera@quickstart:~

File Edit View Search Terminal Help

```
                Spilled Records=44
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=173
                CPU time spent (ms)=1970
                Physical memory (bytes) snapshot=382926848
                Virtual memory (bytes) snapshot=3122540544
                Total committed heap usage (bytes)=268435456
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=88
        File Output Format Counters
                Bytes Written=322
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/stripes/output/1/*
A10     { B12: 1/2, D76: 1/2 }
A12     { B76: 2/10, A10: 2/10, B12: 2/10, D76: 2/10, C31: 2/10 }
B12     { B76: 3/13, A10: 3/13, A12: 3/13, D76: 3/13, C31: 3/13 }
B76     { A10: 1/4, B12: 1/4, D76: 1/4, C31: 1/4 }
C31     { B76: 4/13, A10: 4/13, A12: 4/13, B12: 4/13, D76: 4/13 }
D76     { B76: 2/12, A10: 2/12, A12: 2/12, B12: 2/12, C31: 2/12 }
[cloudera@quickstart ~]$
```

Part 4. Implement Pairs in Mapper and Stripes in Reducer to compute relative frequencies.

Pseudo-code

```
class Mapper

      method initialize
            H = new AssociativeArray

      method map(docid id, doc d)
            for all term w in doc d do
                  for all term u in N(w) do
                        H{u} = H{u} + 1
                  Emit(pair (w;u), H{u})
```

```
class Reducer

     method initialize
            H = new AssociativeArray
            prev = null
            total = 0

      method reduce(pair (w;u), counts [c1,c2, ...])
            if(w ≠ prev && prev ≠ null)
                  Emit(prev, H / total)
                  H = new AssociativeArray
                  total = 0

            sum = sum ([c1, c2, ...])
            total = total + sum;
            H{u} = sum
            prev = w

      method close
            Emit(prev, H / total)
```

Java classes

- HybridMapper.java

```
package part4;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map.Entry;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import parts.*;

public class HybridMapper extends Mapper<LongWritable, Text, WordPair, IntWritable> {

        private HashMap<WordPair, Integer> outputMap = new HashMap<>();

        @Override
        public void map(LongWritable key, Text values, Context context)
                        throws IOException, InterruptedException {

                String input = values.toString();
                String[] readLines = input.split("//.*\n");

                for (String line : readLines) {
                        String[] words = line.split("\\s");
                        for (int i = 0; i < words.length - 1; i++) {
                                for (int j = i + 1; j < words.length; j++) {
                                        if (words[i].equals(words[j]))
                                                break;

                                        WordPair pair = new WordPair(words[i], words[j]);

                                        if (outputMap.get(pair) != null)
                                                outputMap.put(pair, outputMap.get(pair) + 1);
                                        else
                                                outputMap.put(pair, new Integer(1));
                                }
                        }
                }

                for (Entry<WordPair, Integer> mapEntry : outputMap.entrySet()) {
                        context.write(mapEntry.getKey(), new IntWritable(mapEntry.getValue()));
                }
        }
}
```

- HybridReducer.java

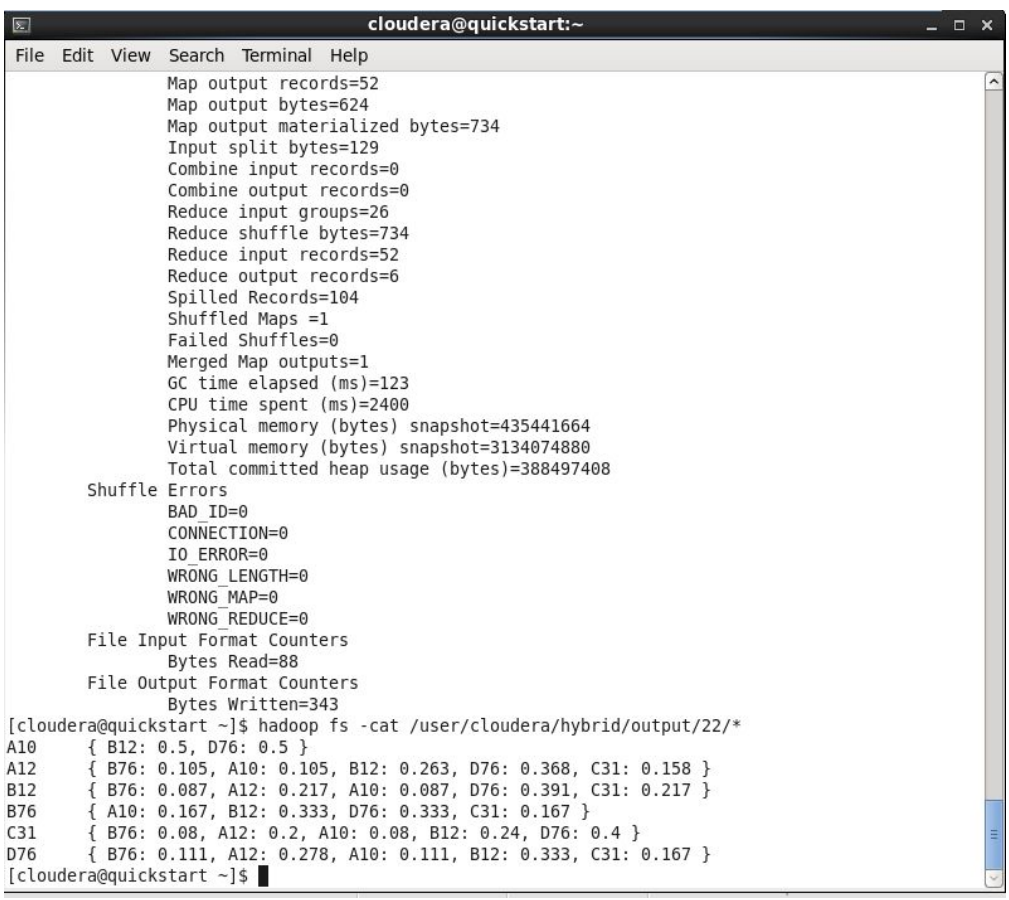
```
package part4;

import java.io.IOException;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.Map.Entry;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
```

```java
import org.apache.hadoop.mapreduce.Reducer;
import parts.*;

public class HybridReducer extends Reducer<WordPair, IntWritable, Text, Stripe> {

	private HashMap<String, Integer> H;
	private double total;
	private String prev;

	@Override
	protected void setup(Reducer<WordPair, IntWritable, Text, Stripe>.Context context)
			throws IOException, InterruptedException {
		super.setup(context);
		H = new HashMap<String, Integer>();
		prev = null;
		total = 0;
	}

	@Override
	protected void reduce(WordPair pair, Iterable<IntWritable> counts,
			Reducer<WordPair, IntWritable, Text, Stripe>.Context context)
			throws IOException, InterruptedException {

		String w = pair.getWord().toString();
		String u = pair.getNeighbor().toString();

		if (prev != null && !prev.equals(w)) {
			context.write(new Text(prev), getUpdatedStripe());
			H = new HashMap<String, Integer>();
			total = 0;
		}

		int sum = sum(counts);
		total += sum;
		H.put(u, sum);
		prev = w;
	}

	@Override
	protected void cleanup(Context context) throws IOException,
			InterruptedException {
		super.cleanup(context);
		context.write(new Text(prev), getUpdatedStripe());
	}

	private Stripe getUpdatedStripe() {
		Stripe stripe = new Stripe();
		DecimalFormat df = new DecimalFormat("#.###");
		double frequency = 0.0;
		for (Entry<String, Integer> entry : H.entrySet()) {
			frequency = entry.getValue() / total;
			stripe.put(new Text(entry.getKey()),
			new DoubleWritable(Double.valueOf(df.format(frequency))));
		}
		return stripe;
	}

	private int sum(Iterable<IntWritable> values) {
		int sum = 0;
		for (IntWritable intWritable : values) {
			sum += intWritable.get();
		}
		return sum;
	}
}
```

Output

- Once job completed successfully, display contents of output file Map-Reduce Framework

```
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/hybrid/output/1/*

A10    { B12: 0.5, D76: 0.5 }

A12    { B76: 0.105, A10: 0.105, B12: 0.263, D76: 0.368, C31: 0.158 }

B12    { B76: 0.087, A12: 0.217, A10: 0.087, D76: 0.391, C31: 0.217 }

B76    { A10: 0.167, B12: 0.333, D76: 0.333, C31: 0.167 }

C31    { B76: 0.08, A12: 0.2, A10: 0.08, B12: 0.24, D76: 0.4 }

D76    { B76: 0.111, A12: 0.278, A10: 0.111, B12: 0.333, C31: 0.167 }
```



Comparison and Analysis

Comparison Table

Metrics	Methods of computing Relative Frequencies		
	Pairs approach	Stripes Approach	Hybrid Approach
Map input records	2	2	2
Map output records	128	22	64
Reduce input groups	32	6	26
Reduce input records	128	22	52
Reduce output records	26	6	6
CPU time spent (ms)	2480	2360	2400
Physical memory (bytes) snapshot	368 214 016	433 135 616	435 441 664

Color code:

Best	Medium	Worst

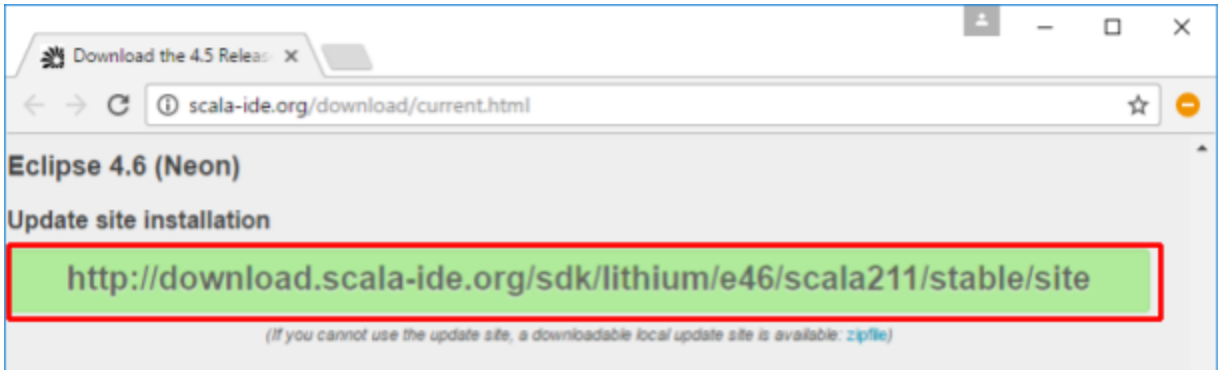
Analysis

- Best CPU time: **Stripes approach**
- Best Network Usage: **Stripes approach**
- Best Memory Usage: **Pairs approach**

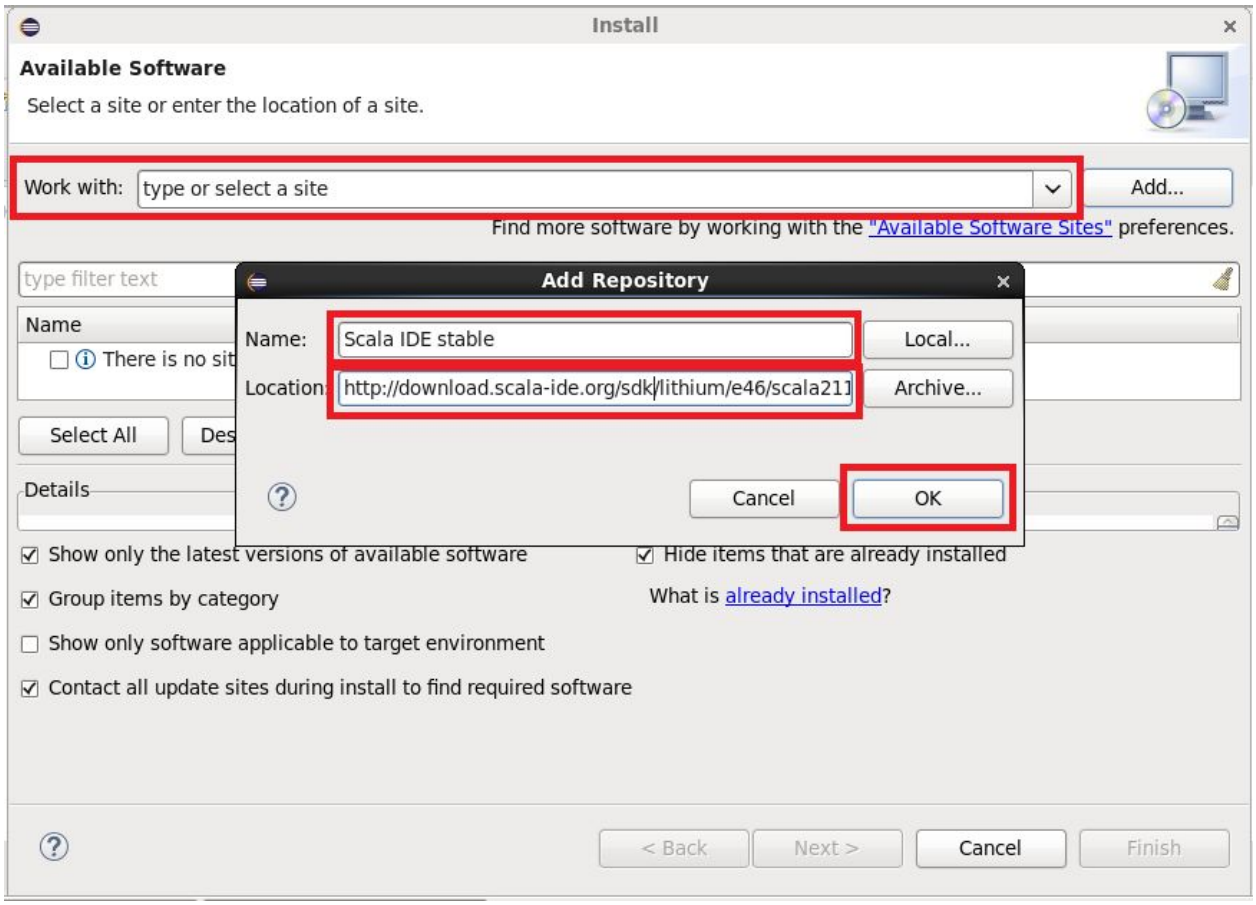
Part 5. Setup Spark and Scala

Setup Scala IDE extension in Eclipse

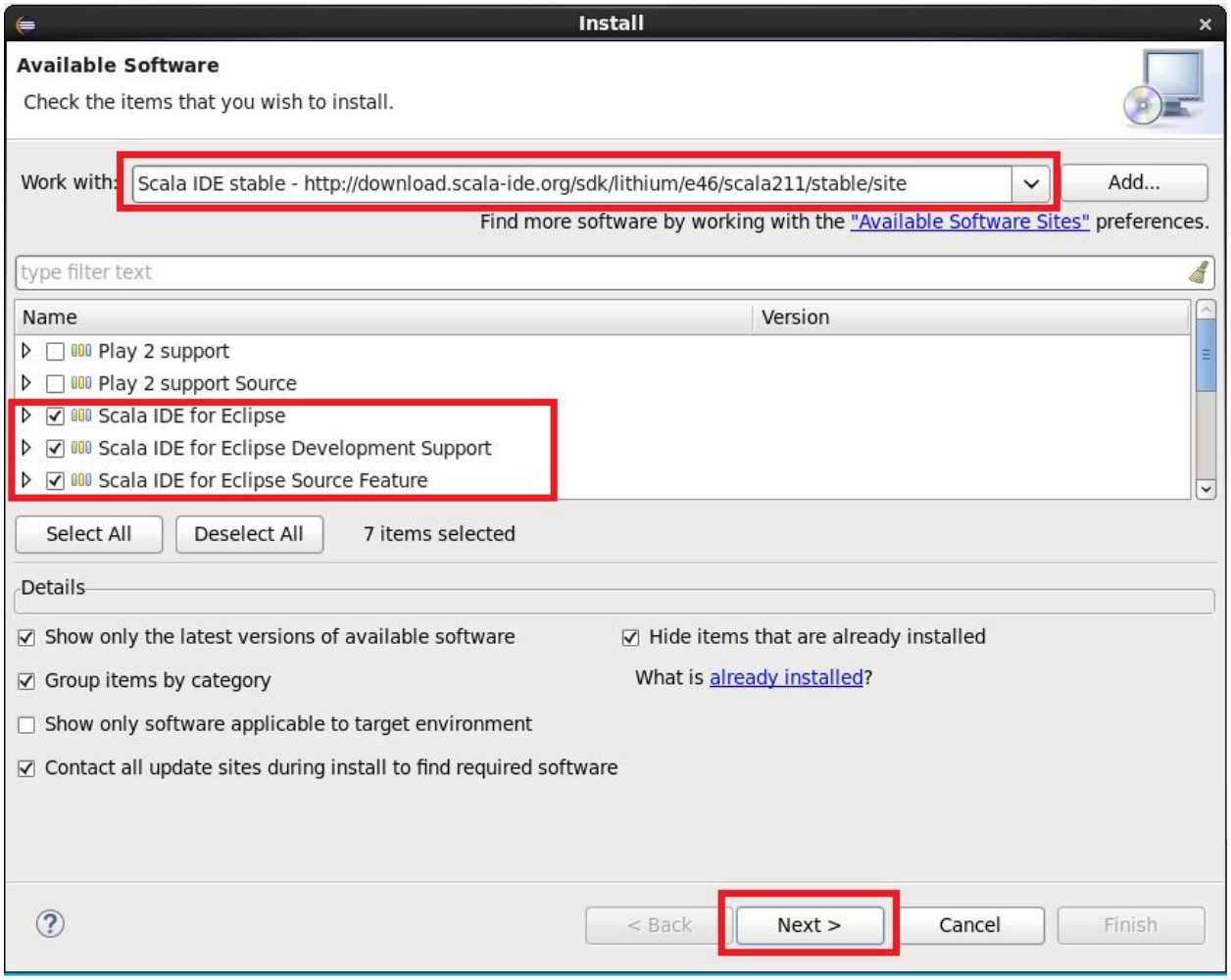
- Go to http://scala-ide.org/download/current.htm and copy the stable repository link



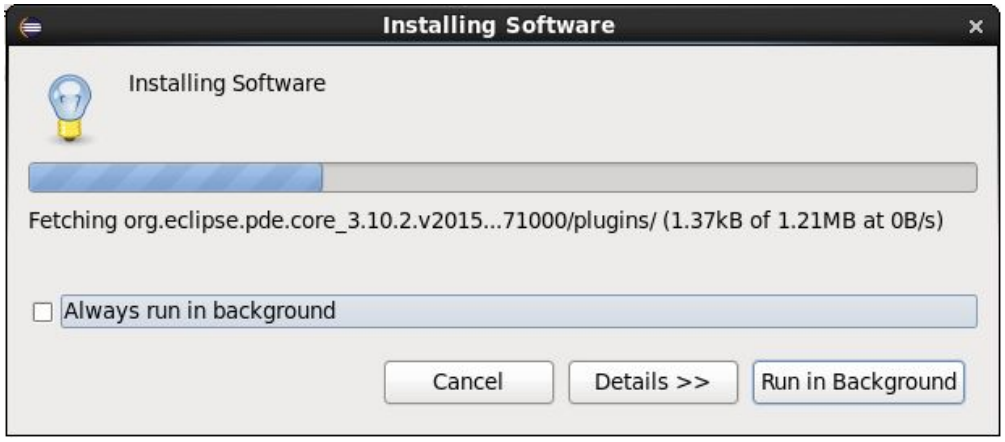
- Open Eclipse menu **Help** -> **Install New Software.** Press **Add...** button to open **Add repository** window. Type name, i.e 'Scala IDE stable', then paste the repository link obtained in previous step and press OK.



- Select for **Work with: Scala IDE stable -...** Check **Scala IDE for Eclipse** and press Next.



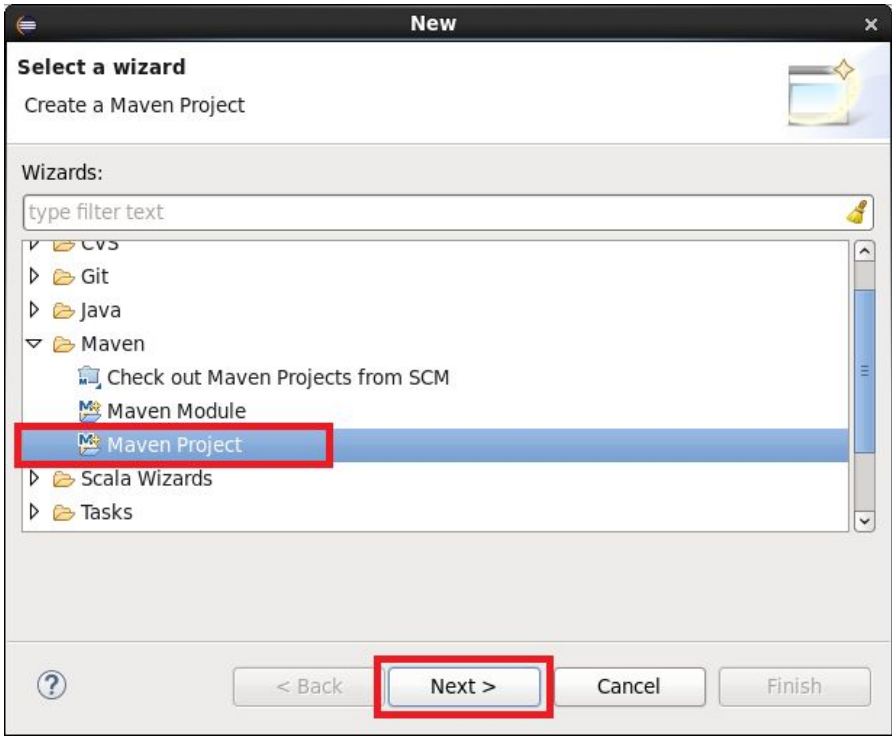
- In next windows you will be asked to confirm selection and agree with terms of usage. Finally, the installation process of Scala IDE extension will take few minutes.



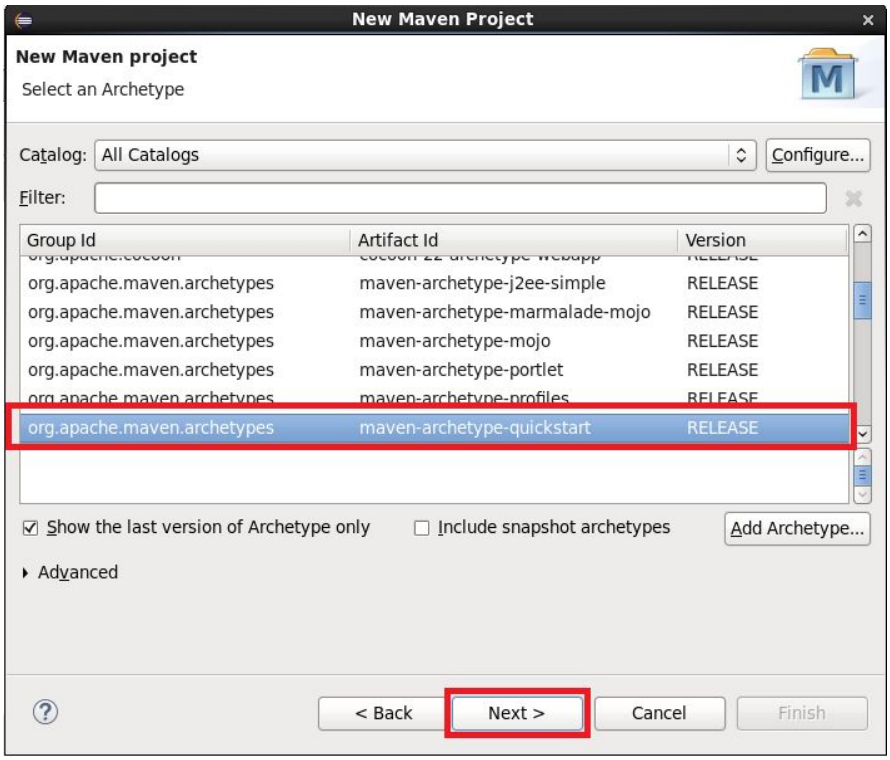
Wordcount Scala Project

Create Maven Project in Eclipse

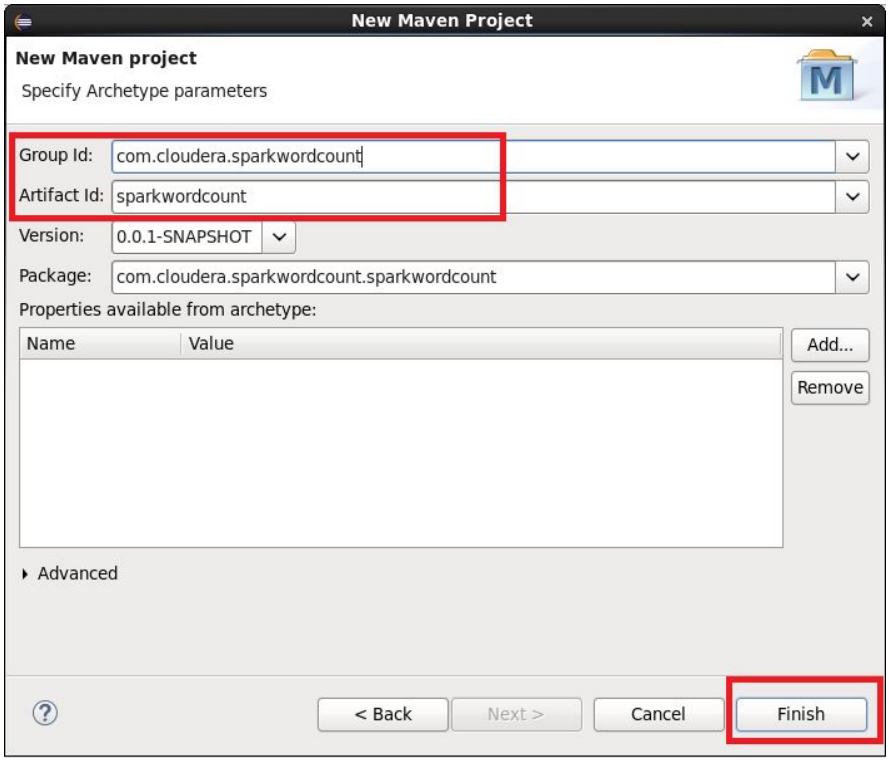
- Open Eclipse menu **File**-> **New** -> **Project** and **Maven Project** and press Next.



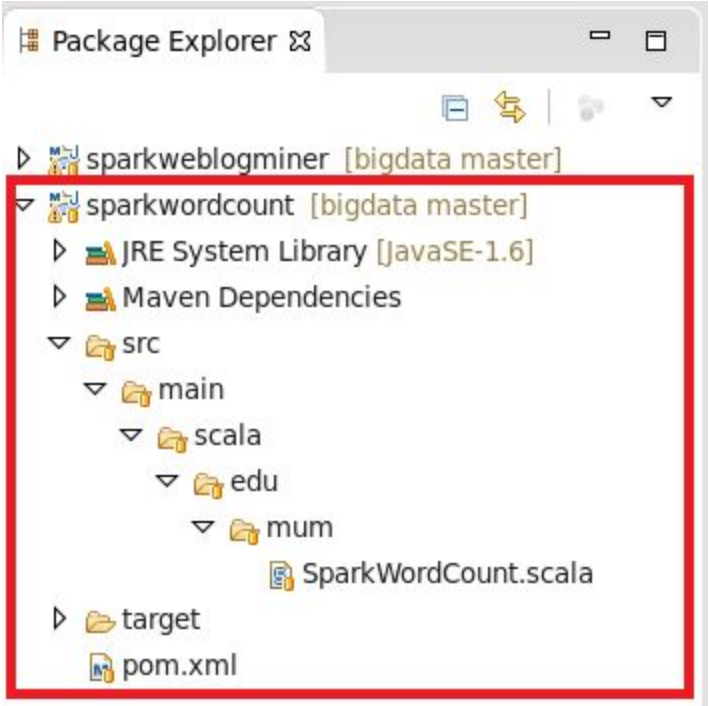
- Select for Artifact Id : **maven-archetype-quickstart** and press Next



- Type **Group Id** and **ArtifactId** for this project and press **Finish.** Example:



- In Eclipse Package Explorer modify folder structure to match with **Group Id** and **Artifact Id.** Then add new scala file. Example:



- Edit newly added **scala file** and paste following code:

```
package edu.mum

import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SparkWordCount {
  def main(args: Array[String]) {

    val sc = new SparkContext(new SparkConf().setAppName("Spark Count"))

    val threshold = args(1).toInt

    val tokenized = sc.textFile(args(0)).flatMap(_.split(" "))

    val wordCounts = tokenized.map((_, 1)).reduceByKey(_ + _)

    val filtered = wordCounts.filter(_._2 >= threshold)

    val charCounts = filtered.flatMap(_._1.toCharArray).map((_, 1)).reduceByKey(_ + _)

    System.out.println(charCounts.collect().mkString(", "))
  }
}
```

- Edit **pom.xml** file and change `repositories` and `pluginRepositories` sections:

```
  <repositories>
    <repository>
      <id>scala-tools.org</id>
      <name>Scala-tools Maven2 Repository</name>
      <url>http://scala-tools.org/repo-releases</url>
    </repository>
    <repository>
      <id>maven-hadoop</id>
      <name>Hadoop Releases</name>
      <url>https://repository.cloudera.com/content/repositories/releases/</url>
    </repository>
    <repository>
      <id>cloudera-repos</id>
      <name>Cloudera Repos</name>
      <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>scala-tools.org</id>
      <name>Scala-tools Maven2 Repository</name>
      <url>http://scala-tools.org/repo-releases</url>
    </pluginRepository>
  </pluginRepositories>
```

Edit **pom.xml** file and change `plugins` section:

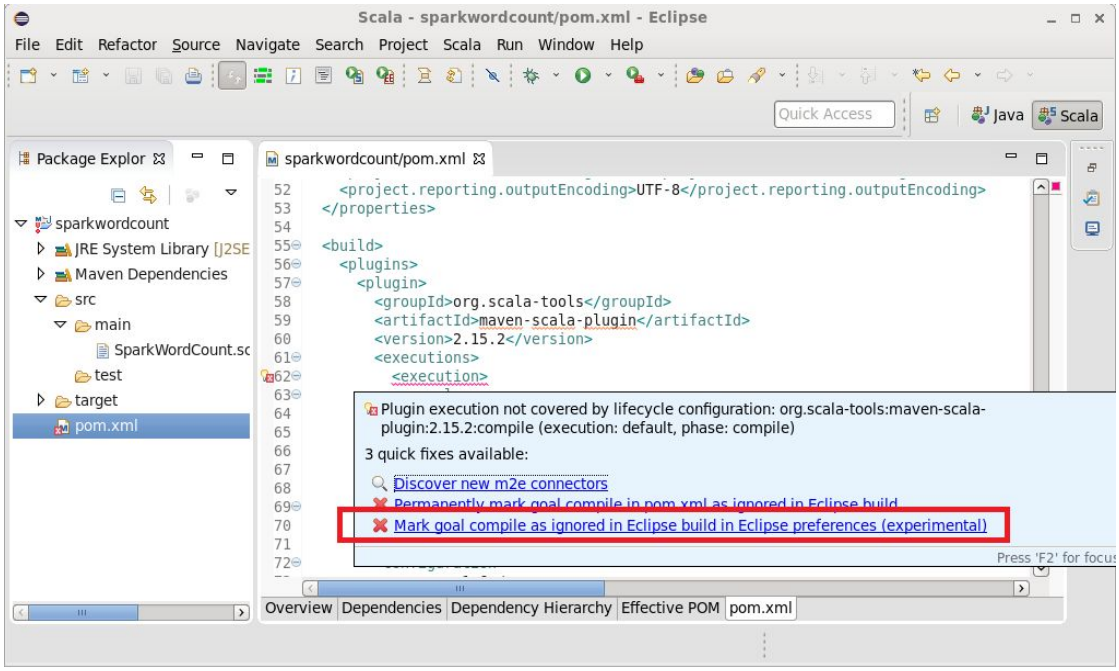
```
    <plugins>
      <plugin>
        <groupId>org.scala-tools</groupId>
        <artifactId>maven-scala-plugin</artifactId>
        <version>2.15.2</version>
        <executions>
          <execution>
            <goals>
```

```
            <goal>compile</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.1</version>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
      </configuration>
    </plugin>
  </plugins>
```

- Edit **pom.xml** file and change `dependencies` section. Make sure you get the correct versions of for both dependencies

```
<dependencies>
  <dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-library</artifactId>
    <version>2.10.4</version>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.10</artifactId>
    <version>1.0.0-cdh5.1.0</version>
  </dependency>
</dependencies>
```

- If **pom.xml** file has errors in `executions` section, press F2 and select option “Mark goal compile as ignored in Eclipse build”.



- Package the project into JAR file

```
$ mvn clean package
```

- Create new input file and put it into HDFS directory

```
$ cat > sample.txt
apple banana orange apple orange
banana orange apple orange apple
orange orange banana apple
banana apple apple
orange banana
apple
^C
$ hdfs dfs -put /user/cloudera/sample.txt
```

- Launch JAR file

```
$ spark-submit --class edu.mum.SparkWordCount --master local
sparkwordcount-0.0.1-SNAPSHOT.jar
hdfs://localhost:8020/user/cloudera/sample.in 2
```

- To verify output is correct, scroll up the terminal window and look for the ERROR logs. If you can't find any ERROR logs, then you should find the following correct output:

```
(e,2), (p,2), (a,5), (b,1), (o,1), (n,3), (r,1), (g,1), (l,1)
```

Part 6. Analyze Apache log files.

Input file

- Install **unrar** utility

```
$ cd /tmp
$ wget http://www.rarlab.com/rar/rarlinux-x64-5.3.b4.tar.gz
$ tar -zxvf rarlinux-*.tar.gz
$ cd rar
$ cp rar unrar /usr/local/bin
```

- Download apache log samples archive and extract it

```
$ wget --no-check-certificate
http://www.monitorware.com/en/logsamples/download/apache-samples.rar
$ unrar e apache-samples.rar
$ unrar e apache-access_log.rar
```

- Create a directory in HDFS and put extracted file to that directory

```
$ hdfs dfs -mkdir /user/cloudera/sparkweblog
$ hdfs dfs -put access_log /user/cloudera/sparkweblog
```

Log Analyzer

- Launch JAR file by spark-submit command

```
$ spark-submit --class edu.mum.LogAnalyzer --master local[*]
sparkweblogminer-0.0.1-SNAPSHOT.jar
hdfs://localhost:8020/user/cloudera/sparkweblog/access_log
```

Output

- Correct output follows:

```
# Content Size Avg: 7078, Min: 143, Max: 138789

# Response code counts: [(404,5),(401,123),(200,1273),(302,6)]

# IPAddresses > 10 times:
[lhr003a.dhl.com,207.195.59.160,10.0.0.153,prxint-sxb3.e-i.net,cr020r01-3.sac.
overture.com,64.242.88.10,ip68-228-43-49.tc.ph.cox.net,ogw.netinfo.bg,200-55-1
04-193.dsl.prima.net.ar,pc3-registry-stockholm.telia.net,ts04-ip92.hevanet.com
,market-mail.panduit.com,216-160-111-121.tukw.qwest.net,195.246.13.119,proxy0.
haifa.ac.il,ts05-ip44.hevanet.com,mail.geovariances.fr,p213.54.168.132.tisdip.
tiscali.de,128.227.88.79,ns.wtbts.org,208-38-57-205.ip.cal.radiant.net,,212.92
.37.62,203.147.138.233,h24-71-236-129.ca.shawcable.net,h24-70-69-74.ca.shawcab
le.net]
```

```
# Top 10 Endpoints:
[(/twiki/bin/view/Main/WebHome,41),(/twiki/pub/TWiki/TWikiLogos/twikiRobot46x5
0.gif,32),(/,31),(/favicon.ico,28),(/robots.txt,27),(/razor.html,23),(/twiki/b
in/view/Main/SpamAssassinTaggingOnly,18),(/twiki/bin/view/Main/SpamAssassinAnd
PostFix,17),(/cgi-bin/mailgraph.cgi/mailgraph_2.png,16)]
```

Analysis details

- Find all IP addresses that accessed server more than 10 times.

```
val ipAddresses: Array[String] = accessLogs
    .map(_.ipAddress -> 1L)
    .reduceByKey(_ + _)
    .filter(_._2 > 10)
    .map(_._1)
```

- Calculate statistics based on the content size.

```
val contentSizes: RDD[Long] = accessLogs
    .map(_.contentSize).cache()

println("# Content Size Avg: %s, Min: %s, Max: %s".format(
    contentSizes.reduce(_ + _) / contentSizes.count,
    contentSizes.min,
    contentSizes.max))
```

- Count Response Codes.

```
val responseCodeToCount: Array[(Int, Long)] = accessLogs
    .map(_.responseCode -> 1L)
    .reduceByKey(_ + _)
```

- Top 10 Endpoints

```
val top10Endpoints: Array[(String, Long)] = accessLogs
    .map(_.endpoint -> 1L)
    .reduceByKey(_ + _)
    .top(10)(Ordering.by[(String, Long), Long](_._2))
```

- Full code of LogAnalyzer scala file

```
package edu.mum

import org.apache.spark.rdd.RDD
import org.apache.spark.{ SparkConf, SparkContext }

object LogAnalyzer extends App {
```

```
  val sparkConf = new SparkConf().setAppName("Log Analyzer in Scala")
  val sc = new SparkContext(sparkConf)

  val logFile = args(0)

  val accessLogs: RDD[ApacheAccessLog] = sc.textFile(logFile).
    map(ApacheAccessLog.parseLogLine).cache()

  // Calculate statistics based on the content size.
  val contentSizes: RDD[Long] = accessLogs.map(_.contentSize).cache()
  println("# Content Size Avg: %s, Min: %s, Max: %s".format(
    contentSizes.reduce(_ + _) / contentSizes.count,
    contentSizes.min,
    contentSizes.max))

  // Compute Response Code to Count.
  val responseCodeToCount: Array[(Int, Long)] = accessLogs
    .map(_.responseCode -> 1L)
    .reduceByKey(_ + _)
    .take(100)
  println(s"""# Response code counts: ${responseCodeToCount.mkString("[", ",", "]")}""")

  // Any IPAddress that has accessed the server more than 10 times.
  val ipAddresses: Array[String] = accessLogs
    .map(_.ipAddress -> 1L)
    .reduceByKey(_ + _)
    .filter(_._2 > 10)
    .map(_._1)
    .take(100)
  println(s"""# IPAddresses > 10 times: ${ipAddresses.mkString("[", ",", "]")}""")

  // Top 10 Endpoints.
  val top10Endpoints: Array[(String, Long)] = accessLogs
    .map(_.endpoint -> 1L)
    .reduceByKey(_ + _)
    .top(10)(Ordering.by[(String, Long), Long](_._2))
  println(s"""# Top 10 Endpoints: ${topEndpoints.mkString("[", ",", "]")}""")

  sc.stop()
}
```

- Full code of ApacheAccessLog scala file

```
package edu.mum

case class ApacheAccessLog(
    ipAddress: String,
    clientIdentd: String,
    userId: String,
    dateTime: String,
    method: String,
    endpoint: String,
    protocol: String,
    responseCode: Int,
```

```
    contentSize: Long) {
}

object ApacheAccessLog {
  val PATTERN = """^(\S+) (\S+) (\S+) \[([\w:/]+\s[+\-]\d{4})\] "(\S+)" (\d{3}) (\d+)""".r

  /**
   * Parse log entry from a string.
    * @param log A string, typically a line from a log file
   * @return An entry of Apache access log
   */
  def parseLogLine(log: String): ApacheAccessLog = {
    log match {
      case PATTERN(ipAddress, clientIdentd, userId, dateTime, method, endpoint,
                   protocol, responseCode, contentSize) =>
        ApacheAccessLog(ipAddress, clientIdentd, userId, dateTime, method, endpoint,
                        protocol, responseCode.toInt,contentSize.toLong)
      case _ => throw new RuntimeException(s"""Cannot parse log line: $log""")
    }
  }
}
```

Resources and Links used in the Project

Cloudera blogs:

- https://www.cloudera.com/documentation/enterprise/5-5-x/topics/spark_develop_run.html
- https://blog.cloudera.com/blog/2014/04/how-to-run-a-simple-apache-spark-app-in-cdh-5/

Databricks references:

- https://www.gitbook.com/book/databricks/databricks-spark-reference-applications/details
- https://github.com/databricks/reference-apps

Tutorialspoint:

- www.tutorialspoint.com//apache_spark/index.htm

Project report:

- https://docs.google.com/document/d/1bhjLaTpJhC6Q-3NNuJZ8MrjQr_mLwCNyd8LkmJfrVjo/edit?usp=sharing

Project Presentation:

- https://docs.google.com/presentation/d/1O8VVBe4iA2vug4drh4uCDjkgqzQuyeHPzDDIO_EpBLE/edit?usp=sharing

Source code:

- https://github.com/ganijon/bigdata/tree/master/projects