BIG DATA

TOO BIG TO IGNORE

SÜMEYYE KAYNAK

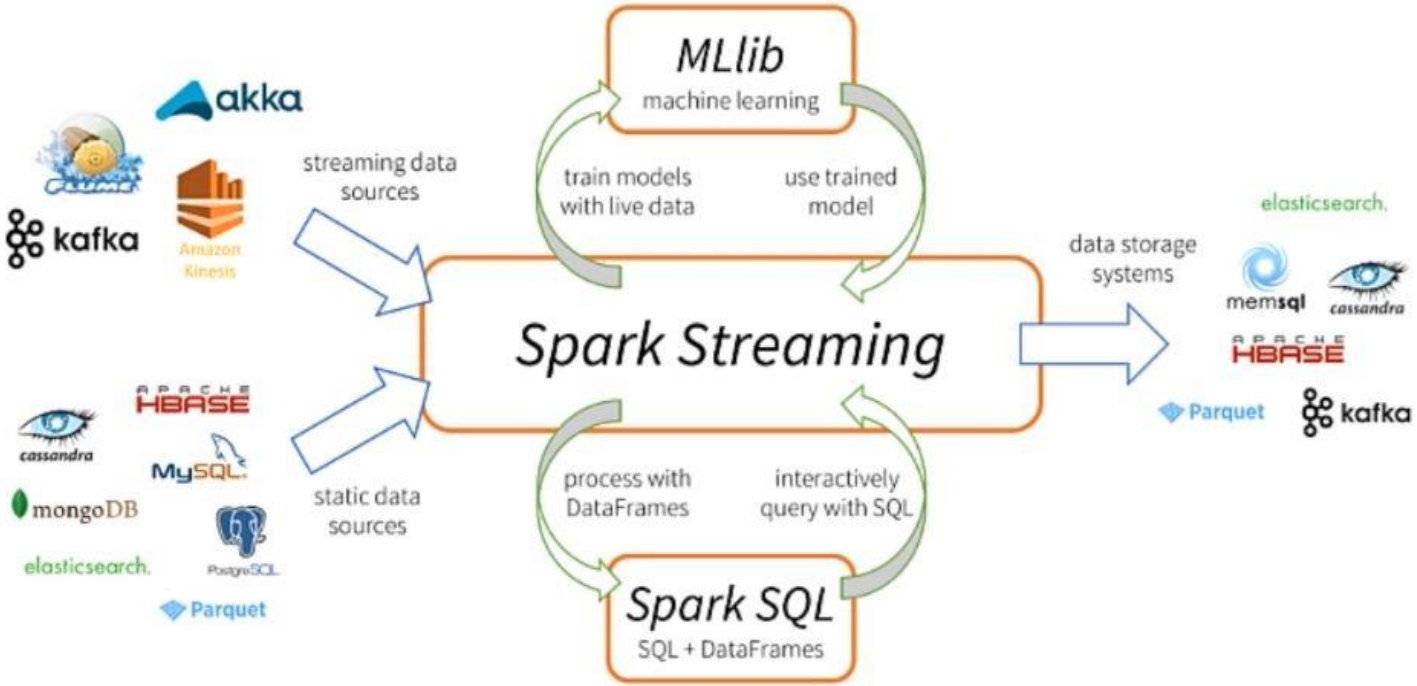
OUTLINE

Spark

SPARK

- It is a big data library developed with Scala to analyze big data with distributed processing. Spark works in-memory.
- Therefore, there is no storage unit. It performs data analysis on RAM.

SPARK


MLlib
machine learning
train models with live data
use trained model
akka
kafka
Amazon Kinesis
streaming data sources
Spark Streaming
data storage systems
elasticsearch.
memsql
cassandra
HBASE
Parquet
kafka
HBASE
cassandra
MySQL
mongoDB
elasticsearch.
PostgreSQL
Parquet
static data sources
process with DataFrames
interactively query with SQL
Spark SQL
SQL + DataFrames

SPARK AND HADOOP

Storage:

Hadoop=> HDFS

Spark=> There is no internal storage. But thanks to integration, data can be stored in different technologies.

Speed:

Spark is 100 times faster than Hadoop

Difficult:

Spark => It is easy to program thanks to Spark RDD and SQL.

Hadoop => Developing Map-reduce is difficult.

SPARK AND HADOOP

Management:

Hadoop=> Yarn

Spark=> provides its own management

Real-time analysis:

Hadoop=> There is no real-time analysis tool.

Spark => With Spark streaming, millions of data per second can be analyzed instantly.

SPARK MLLIB

- has an extra library (Spark MLlib) where you can apply machine learning techniques in Spark.
- Estimation and classification can be made with techniques such as logistic regression, K Means, K NN, Naive Bayes.

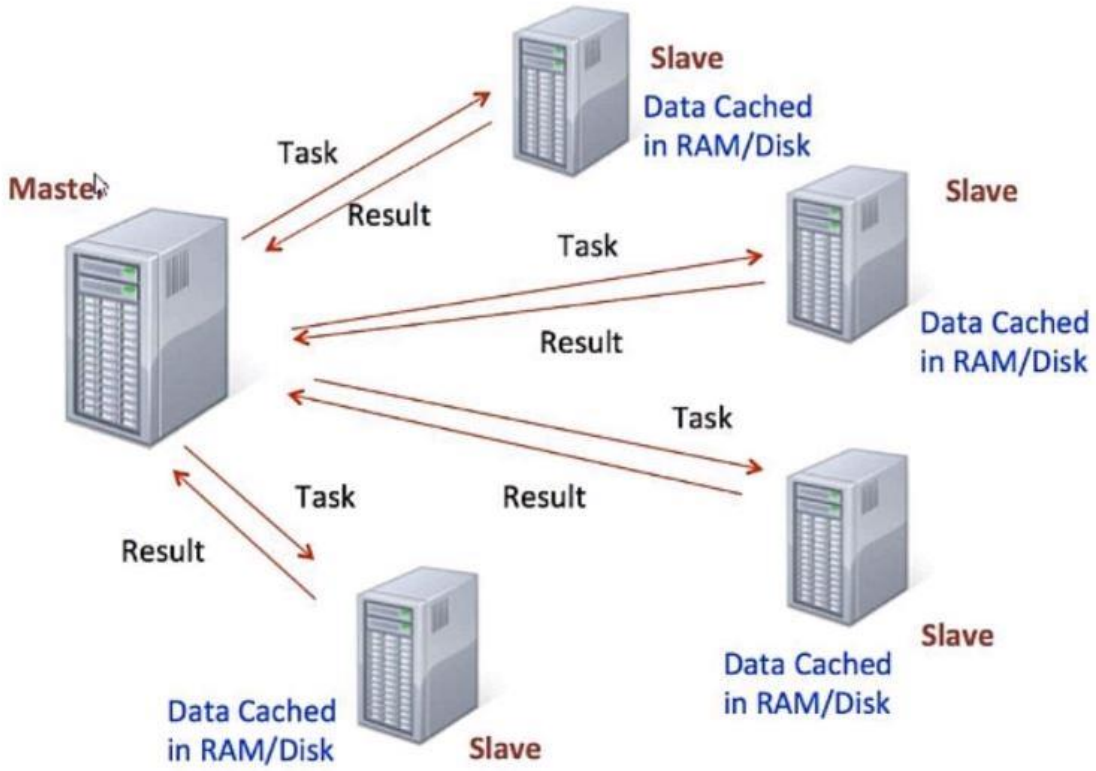
SPARK STREAMING

- Instant, real-time data analysis is performed with Spark's streaming library.

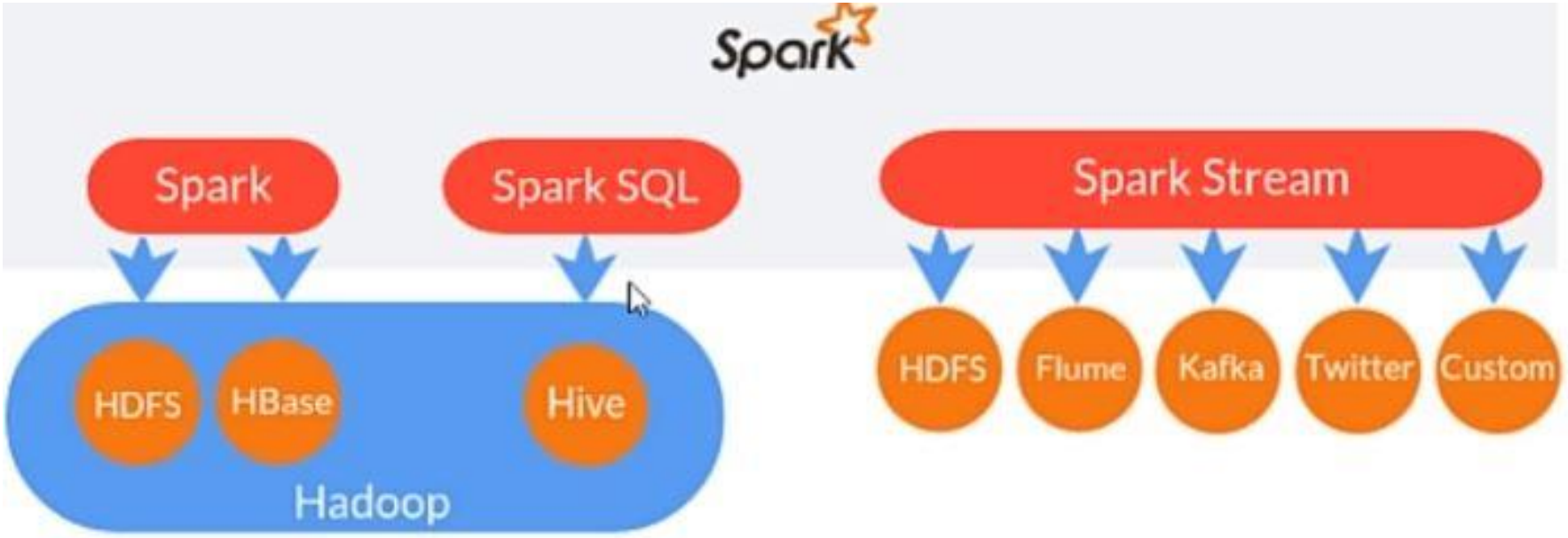
SPARK SQL

- Spark has a library for performing big data analysis with SQL-based queries.

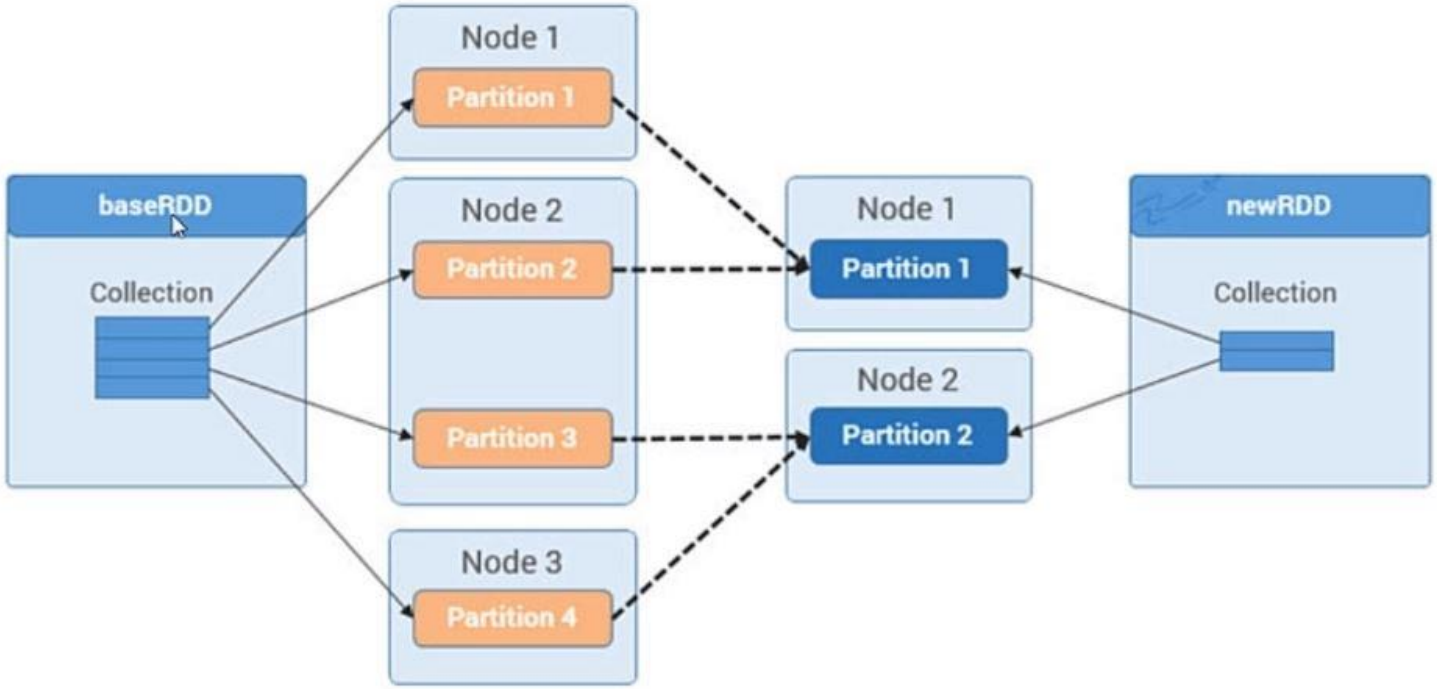
SPARK ARCHITECTURE



SPARK

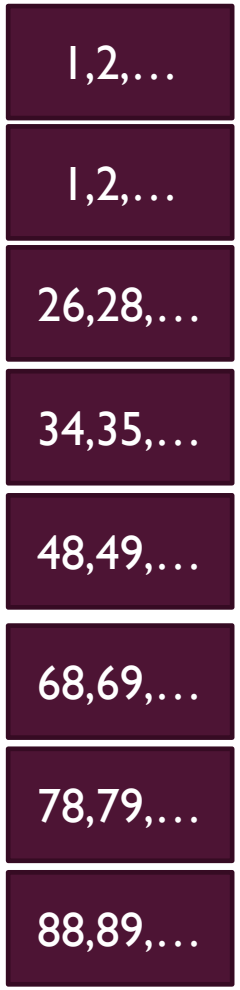

Spark
Spark
Spark SQL
Spark Stream
HDFS
HBase
Hive
Hadoop
HDFS
Flume
Kafka
Twitter
Custom

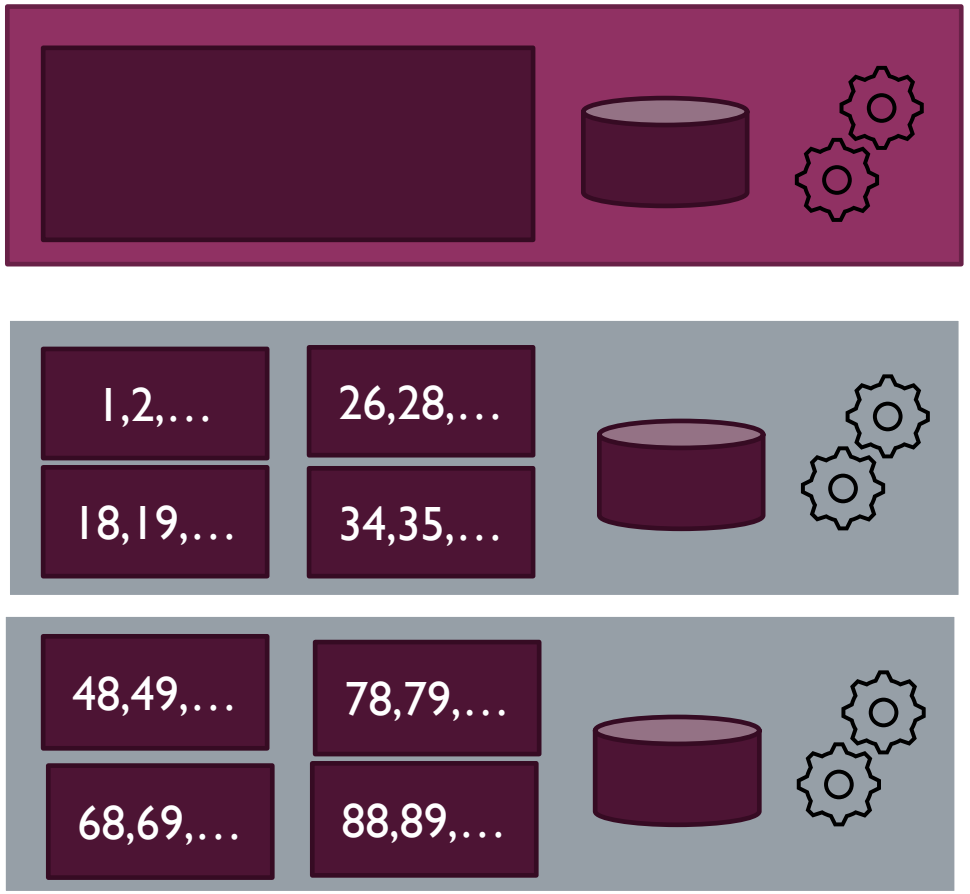
SPARK RDD


Node 1
Partition 1
baseRDD
Collection
Node 2
Partition 2
Partition 3
Node 3
Partition 4
Node 1
Partition 1
Node 2
Partition 2
newRDD
Collection

SPARK RDD

[1,2,……,99]



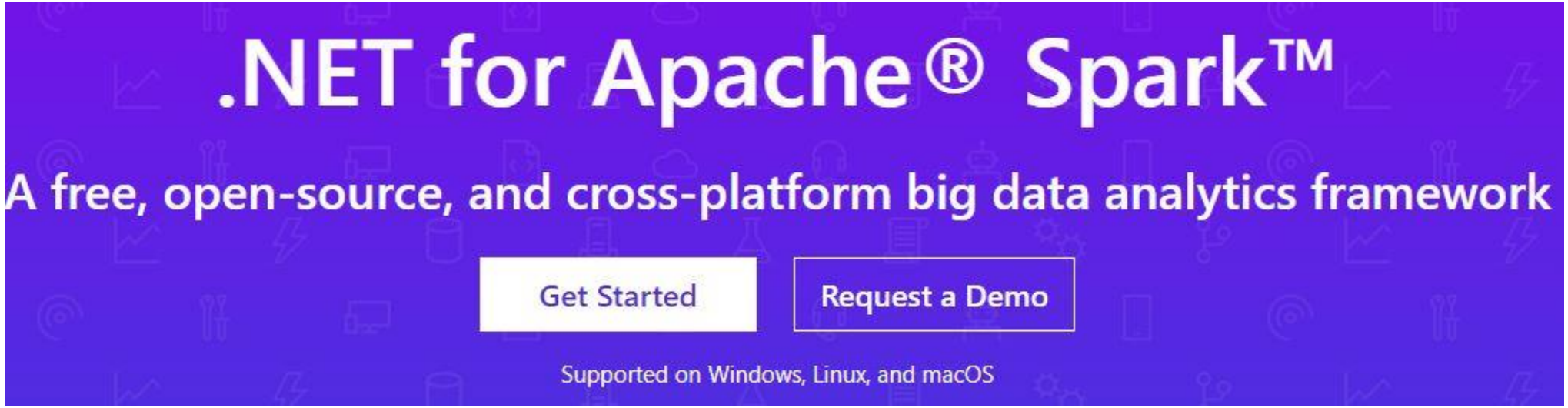


List

RDD

Physical

.NET FOR APACHE SPARK APP



.NET FOR APACHE SPARK APP

```
C:\Users\Sumeyye>dotnet

Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help         Display help.
  --info            Display .NET information.
  --list-sdks       Display the installed SDKs.
  --list-runtimes   Display the installed runtimes.

path-to-application:
  The path to an application .dll file to execute.
```

.NET FOR APACHE SPARK APP-INSTALL JAVA

1. Visit **Java SE Development Kit 8 Downloads**.
2. Click the download for **Windows x64**.
3. Review the License agreement and accept it if you agree.
4. Oracle requires an account to download the JDK. So, sign in to an existing account or complete the account registration process to start downloading.
5. Once the download completes, run the installer using the default settings.

Once you've installed, open a **new** command prompt and run the following command:

```
C:\Users\Sumeyye>java -version
java version "11.0.13" 2021-10-19 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.13+10-LTS-370)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.13+10-LTS-370, mixed mode)
```

.NET FOR APACHE SPARK APP-INSTALL JAVA

To download Java 8, click Dowload Java

1. Click the download for **Windows x64**.
2. Review the License agreement and accept it if you agree.
3. Oracle requires an account to download the JDK. So, sign in to an existing account or complete the account registration process to start downloading.
4. Once the download completes, run the installer using the default settings.

.NET FOR APACHE SPARK APP-INSTALL JAVA JDK8

JDK 8 software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

JDK 8u311 checksum

Linux macOS Solaris W

Product/file description

x86 Installer

x64 Installer

Documentation Download

You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.

☑ I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

Required

You will be redirected to the login screen in order to download the file.

Download jdk-8u311-windows-x64.exe

.NET FOR APACHE SPARK APP-DOWNLOAD APACHE SPARK

1. Apache Spark is downloaded as a .tgz file. https://archive.apache.org/dist/spark/spark-3.0.1/spark-3.0.1-bin-hadoop2.7.tgz
2. Extract Apache Spark files in C directories
3. Open a **new** command prompt, run the following commands to set the environment variables used to locate Apache Spark:

```
C:\Users\Sumeyye>setx HADOOP_HOME C:\bin\spark-3.0.1-bin-hadoop2.7\

SUCCESS: Specified value was saved.

C:\Users\Sumeyye>setx SPARK_HOME C:\bin\spark-3.0.1-bin-hadoop2.7\

SUCCESS: Specified value was saved.
```

CHECK SPARK VERSION

```
C:\bin\spark-3.0.1-bin-hadoop2.7\bin>spark-submit --version
```

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.1
      /_/

Using Scala version 2.12.10, Java HotSpot(TM) 64-Bit Server VM, 11.0.13
Branch HEAD
Compiled by user ubuntu on 2020-08-28T07:36:48Z
Revision 2b147c4cd50da32fe2b4167f97c8142102a0510d
Url https://gitbox.apache.org/repos/asf/spark.git
Type --help for more information.
```

MICROSOFT.SPARK.WORKER

1. Download the Microsoft.Spark.Worker release from the .NET for Apache Spark GitHub repository: (https://github.com/dotnet/spark/releases/download/v1.0.0/Microsoft.Spark.Worker.netcoreapp3.1.win-x64-1.0.0.zip?WT.mc_id=dotnet-35129-website)
2. Enter C:\bin in the Extract to field.

INSTALL WINUTILS

1. .NET for Apache Spark requires WinUtils to be installed alongside Apache Spark.

https://github.com/steveloughran/winutils/raw/master/hadoop-2.7.1/bin/winutils.exe?WT.mc_id=dotnet-35129-website

Once **winutils.exe** downloads, copy it into **C:\bin\spark-3.0.1-bin-hadoop2.7\bin**.

SET DOTNET_WORKER_DIR

- This is used by .NET apps to locate .NET for Apache Spark.

```
C:\bin\spark-3.0.1-bin-hadoop2.7\bin>setx DOTNET_WORKER_DIR "C:\bin\Microsoft.Spark.Worker-1.0.0"

SUCCESS: Specified value was saved.
```

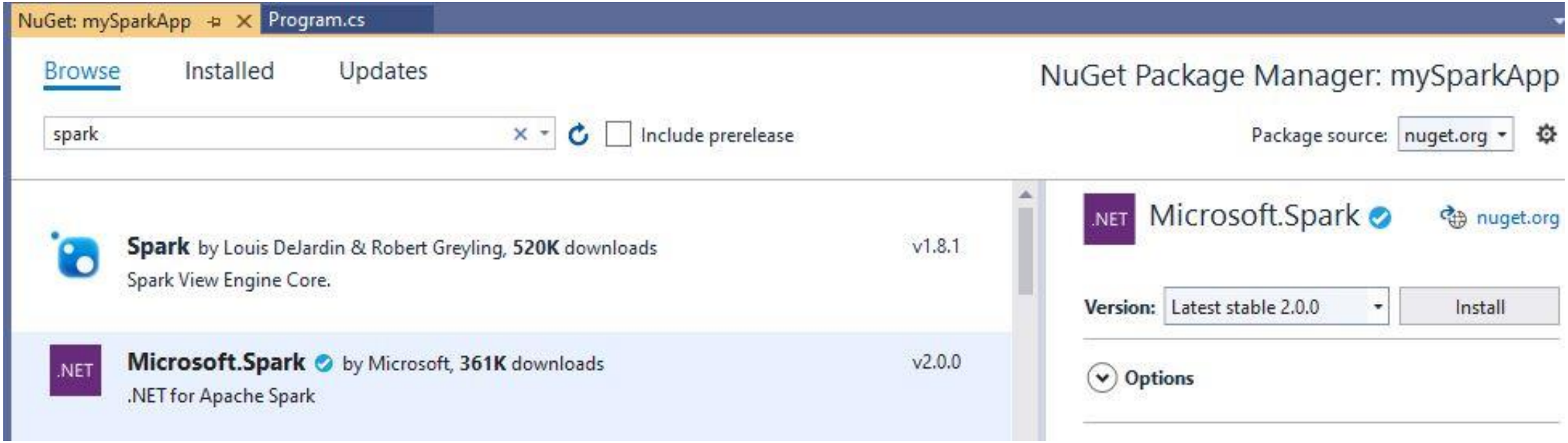
CREATE YOUR APP

```
> dotnet new console -f netcoreapp3.1 -o mySparkApp
```

```
> cd mySparkApp
```

Open visual studio and create a console application.

INSTALL THE NUGET PACKAGE



RUNNING SPARK

Select Windows PowerShell

```
PS C:\Users\Sumeyye\source\repos\mySparkApp\mySparkApp> spark-submit --class org.apache.spark.deploy.dotnet.DotnetRunner
 --master local  .\bin\Debug\netcoreapp3.1\microsoft-spark-3-0_2.12-2.0.0.jar  debug
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/bin/spark-3.0.1-bin-hadoop2.7/jars/spar
k-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
21/12/12 15:34:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
21/12/12 15:34:40 INFO DotnetRunner: Starting DotnetBackend with .
21/12/12 15:34:40 INFO DotnetBackend: The number of DotnetBackend threads is set to 10.
21/12/12 15:34:42 INFO DotnetRunner: Port number used by DotnetBackend is 5567
*****************************************************************************
* .NET Backend running debug mode. Press enter to exit *
*****************************************************************************
```

ADD DATA FILE

Create an `input.txt` file in your `mySparkApp` directory, containing the following text.

```
input.txt

Hello World
This .NET app uses .NET for Apache Spark
This .NET app counts words with Apache Spark
```

CODE YOUR APP

```
// Create a Spark session
SparkSession spark = SparkSession
    .Builder()
    .AppName("word_count_sample")
    .GetOrCreate();

// Create initial DataFrame
DataFrame dataFrame = spark.Read().Text("input.txt");

// Count words
DataFrame words = dataFrame
    .Select(Functions.Split(Functions.Col("value"), " ").Alias("words"))
    .Select(Functions.Explode(Functions.Col("words"))
    .Alias("word"))
    .GroupBy("word")
    .Count()
    .OrderBy(Functions.Col("count").Desc());

// Show results
words.Show();

// Stop Spark session
spark.Stop();
```

RUNNING APP

Microsoft Visual Studio Debug Console

```
[2021-12-12T12:34:55.8163166Z] [DESKTOP-GK3VE45] [Info] [ConfigurationService] 'DOTNETBACKEND_PORT' environment variable
 is not set.
[2021-12-12T12:34:55.8415011Z] [DESKTOP-GK3VE45] [Info] [ConfigurationService] Using port 5567 for connection.
[2021-12-12T12:34:55.8639466Z] [DESKTOP-GK3VE45] [Info] [JvmBridge] JvMBridge port is 5567
[2021-12-12T12:34:55.8806514Z] [DESKTOP-GK3VE45] [Info] [JvmBridge] The number of JVM backend thread is set to 10. The m
ax number of concurrent sockets in JvmBridge is set to 7.
+------+-----+
|  word|count|
+------+-----+
|  .NET|    3|
|Apache|    2|
|   app|    2|
|  This|    2|
| Spark|    2|
| World|    1|
|counts|    1|
|   for|    1|
| words|    1|
|  with|    1|
| Hello|    1|
|  uses|    1|
+------+-----+


C:\Users\Sumeyye\source\repos\mySparkApp\mySparkApp\bin\Debug\netcoreapp3.1\mySparkApp.exe (process 19780) exited with c
ode 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

DATA LOADING STAGES

Installing from local computer

```
DataFrame wordRDD = spark.Read().Text("input.txt");
```

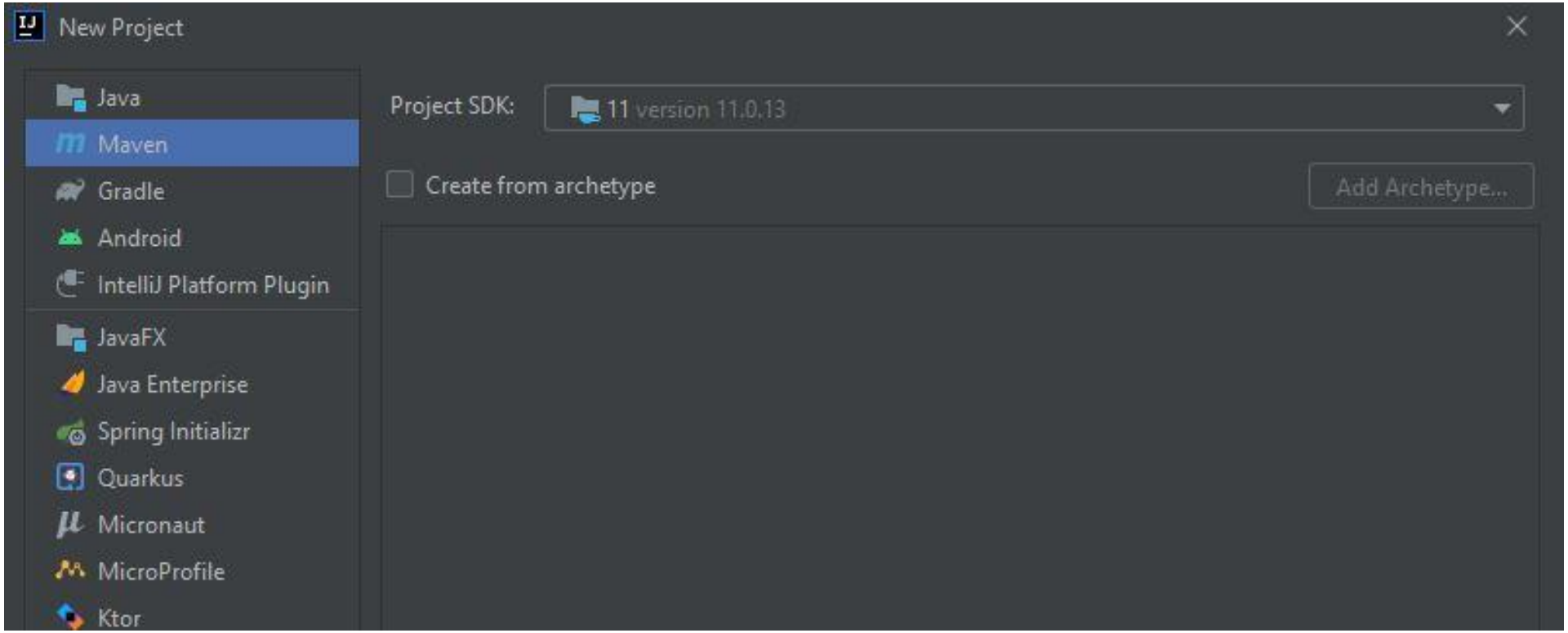
Installing from HDFS

```
DataFrame wordRDD = spark.Read().Text(“hdfs://user/file");
```

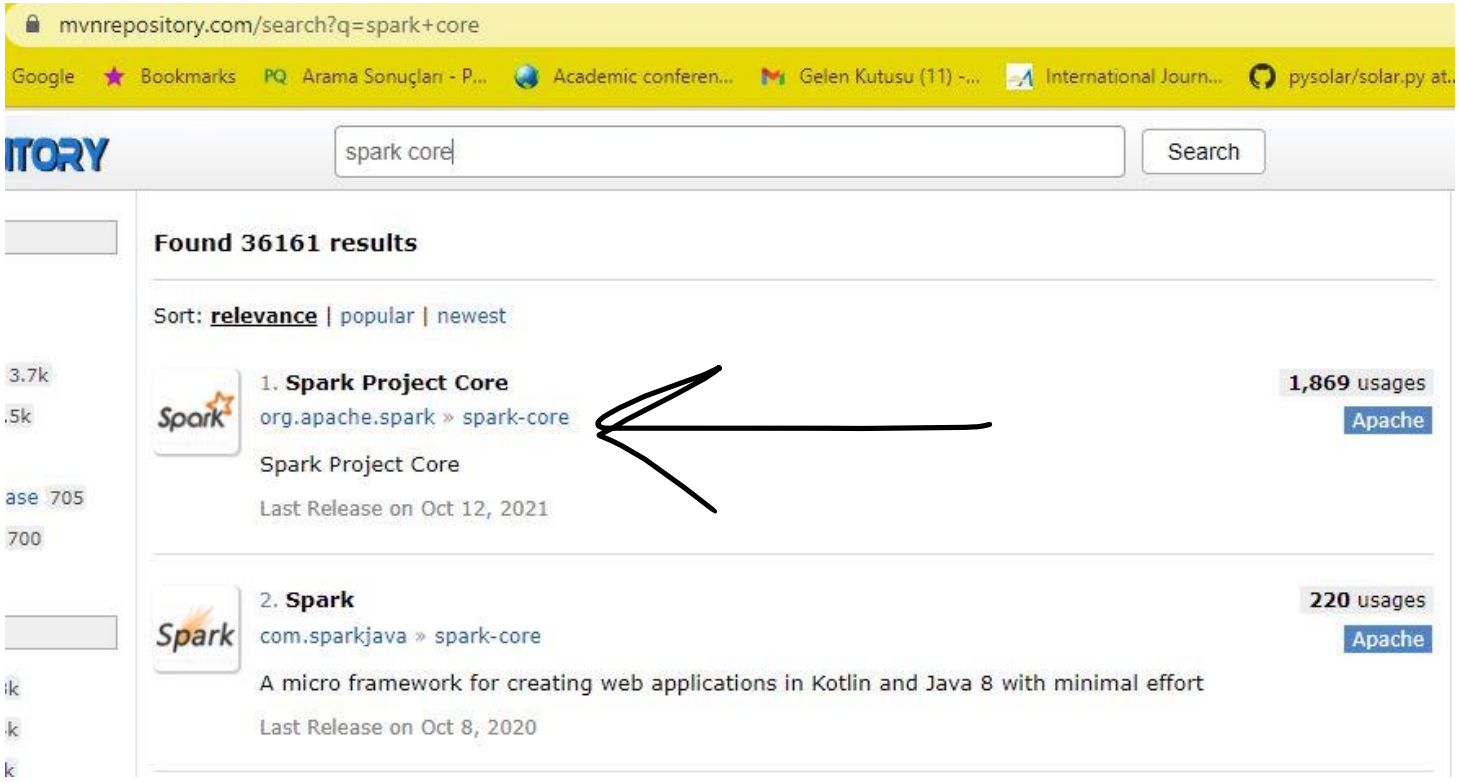
Download from zipped file

```
 DataFrame wordRDD = spark.Read().Text(“*.gz");
```

SPARK USING JAVA



SPARK USING JAVA



DEPENDENCE IN JAVA

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.apache.spark/spark-core -->
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-core_2.12</artifactId>
        <version>3.0.1</version>
    </dependency>

</dependencies>
```

CREATE FIRST RDD

```
public static void main(String[] args) {
    SparkConf conf= new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");;
    JavaSparkContext sc = new JavaSparkContext (conf);
    String path = "C:\\Users\\Sumeyye\\spark\\input.txt";
    // read text file to RDD
    JavaRDD<String> lines = sc.textFile(path);
    // collect RDD for printing
    for(String line:lines.collect()){
        System.out.println(line);
    }
}
```

CREATE FIRST RDD

pom.xml (FirstRDD) | App.java

```
13  public class App {
14
15      public static void main(String[] args) {
16          SparkConf conf= new SparkConf().setAppName("ABC").setMaster("local[*]").se
17          JavaSparkContext cont = new JavaSparkContext (conf);
18          String path = "C:\\Users\\Sumeyye\\spark\\input.txt";
19          // read text file to RDD
20          JavaRDD<String> lines = cont.textFile(path);
21          lines.
22          //
23          for
24
25          }
26      }
```

cache()	JavaRDD<String>
classTag()	ClassTag<String>
coalesce(int numPartitions)	JavaRDD<String>
collect()	List<String>
coalesce(int numPartitions, boolean shu…	JavaRDD<String>
distinct()	JavaRDD<String>
distinct(int numPartitions)	JavaRDD<String>
filter(Function<String, Boolean> f)	JavaRDD<String>
intersection(JavaRDD<String> other)	JavaRDD<String>
persist(StorageLevel newLevel)	JavaRDD<String>
randomSplit(double[] weights)	JavaRDD<String>[]

Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip

```
ackerMasterEndpoint
: MemoryStore clear
r: BlockManager st
rMaster: BlockManag
tCoordinator$Outpu
t: Successfully stopped SparkContext
```

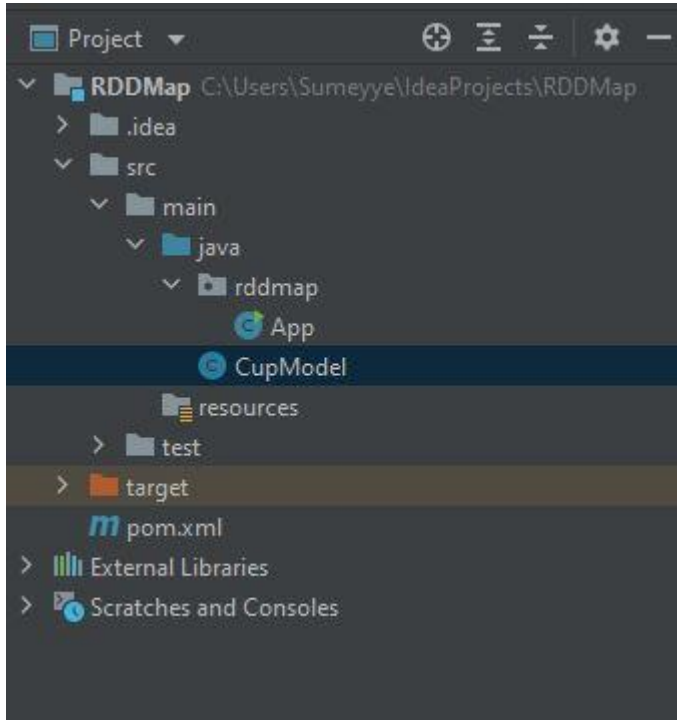
DATA LOAD TYPES

```
pom.xml (FirstRDD)   App.java

14  public class App {
15
16      public static void main(String[] args) {
17          SparkConf conf= new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");
18          JavaSparkContext cont = new JavaSparkContext (conf);
19          String path = "C:\\Users\\Sumeyye\\spark\\input.txt";
20          // read text file to RDD
21          JavaRDD<String> lines = cont.textFile(path);
22
23          List<String> scientist= Arrays.asList("Albert Einstein", "Isaac Newton", "Stephen Hawking" );
24          JavaRDD<String> sci=cont.parallelize(scientist);
25          sci.
26
27

ackerMasterEndpoi
: MemoryStore cle
: BlockManager
Master: BlockMan
tCoordinator$Outp
t: Successfully
kManager: Shutdow
kManager: Deleti

cache()                                          JavaRDD<String>
classTag()                                       ClassTag<String>
coalesce(int numPartitions)                      JavaRDD<String>
coalesce(int numPartitions, boolean shu…         JavaRDD<String>
distinct()                                       JavaRDD<String>
distinct(int numPartitions)                      JavaRDD<String>
filter(Function<String, Boolean> f)              JavaRDD<String>
intersection(JavaRDD<String> other)              JavaRDD<String>
persist(StorageLevel newLevel)                   JavaRDD<String>
randomSplit(double[] weights)                    JavaRDD<String>[]
randomSplit(double[] weights, long se…           JavaRDD<String>[]

Ctrl+Down and Ctrl+Up will move caret down and up in the editor  Next Tip

-f343-4a75-9d84-dd547b98b17d
```

RDD OPERATIONS

- Transformation: The result of operations on an rdd creates a new rdd. (map, filter)
- Actions: Calculation and saving operations on an rdd are done. (count, first)

CREATE JAVA OBJECT




```java
import scala.Int;

public class CupModel {
    String year;
    String host;
    String first;
    String second;
    String third;
    String fourth;
    Int totalgoals;
    Int totalcountry;
    Int totalmatches;
    Int totalpartic;

    public CupModel(String year, String host, String first, String second, String third, String fourth,
        this.year = year;
        this.host = host;
        this.first = first;
        this.second = second;
        this.third = third;
        this.fourth = fourth;
        this.totalgoals = totalgoals;
        this.totalcountry = totalcountry;
        this.totalmatches = totalmatches;
        this.totalpartic = totalpartic;
    }

    public String getYear() {
        return year;
    }

    public void setYear(String year) {
```

```java
public class App {
    public static void main(String[] args) {
        SparkConf conf= new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");;
        JavaSparkContext cont = new JavaSparkContext (conf);
        String path = "C:\\bin\\WorldCup\\WorldCups.csv";
        // read text file to RDD
        JavaRDD<String> Raw_Data = cont.textFile(path);
        System.out.println(Raw_Data.count());

        Raw_Data.map(new Function<String,CupModel>() {
            @Override
            public CupModel call(String line) throws Exception {
                return null;
            }
        });
```

Raw_Data, String türündendir. Bu nedenle; new function ilk parametresi string olmalıdır. Bu String yapıyı object çevireceğiz. "CupModel" isminde object oluşturmuştuk. Fonksiyonu yazdıktan sonra implemente edilmelidir.

```java
JavaRDD<CupModel> map = Raw_Data.map(new Function<String, CupModel>() {
    @Override
    public CupModel call(String line) throws Exception {
        String[] split = line.split(",");
        var cupModel = new CupModel(split[0], split[1], split[2], split[3], split[4], split[5], Integer.par
        return cupModel;
    }
});
```

```java
JavaRDD<CupModel> map = Raw_Data.map(new Function<String, CupModel>() {
    @Override
    public CupModel call(String line) throws Exception {
        String[] split = line.split( regex: ",");
        var cupModel = new CupModel(split[0], split[1], split[2], split[3], split
        return cupModel;
    }
});
map.
```

```
map(Function<CupModel, R> f)                    JavaRDD<R>
cache()                                         JavaRDD<CupModel>
classTag()                                      ClassTag<CupModel>
coalesce(int numPartitions)                     JavaRDD<CupModel>
coalesce(int numPartitions, boolean s…          JavaRDD<CupModel>
distinct()                                      JavaRDD<CupModel>
distinct(int numPartitions)                     JavaRDD<CupModel>
filter(Function<CupModel, Boolean> f)           JavaRDD<CupModel>
intersection(JavaRDD<CupModel> other)           JavaRDD<CupModel>
persist(StorageLevel newLevel)                  JavaRDD<CupModel>
randomSplit(double[] weights)                   JavaRDD<CupModel>[]
randomSplit(double[] weights, long              JavaRDD<CupModel>[]
```

Press Enter to insert, Tab to replace Next Tip

```java
JavaRDD<CupModel> map = Raw_Data.map(new Function<String, CupModel>() {
    @Override
    public CupModel call(String line) throws Exception {
        String[] split = line.split( regex: ",");
        var cupModel = new CupModel(split[0], split[1], split[2], split[3], split[4], split[5], Integer.
        return cupModel;
    }
});
map.foreach(new VoidFunction<CupModel>() {
    @Override
    public void call(CupModel cupModel) throws Exception {
        System.out.println(cupModel.getHost());
    }
});
```

```java
public static void main(String[] args) {
    SparkConf conf= new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");;
    JavaSparkContext cont = new JavaSparkContext (conf);
    String path = "C:\\bin\\WorldCup\\WorldCups.csv";
    // read text file to RDD
    JavaRDD<String> Raw_Data = cont.textFile(path);
    System.out.println(Raw_Data.count());

    JavaRDD<CupModel> map = Raw_Data.map(new Function<String, CupModel>() {
        @Override
        public CupModel call(String line) throws Exception {
            String[] split = line.split(",");
            var cupModel = new CupModel(
                    split[0],
                    split[1],
                    split[2],
                    split[3],
                    split[4],
                    split[5]);
            return cupModel;
        }
    });
    map.foreach(new VoidFunction<CupModel>() {
        @Override
        public void call(CupModel cupModel) throws Exception {
            System.out.println(cupModel.getHost());
        }
    });
}
```

```java
JavaRDD<CupModel> italy = map.filter(new Function<CupModel, Boolean>() {
    @Override
    public Boolean call(CupModel cupModel) throws Exception {
        boolean italy = cupModel.getFirst().equals("Italy");
        return italy;
    }
});
italy.foreach(new VoidFunction<CupModel>() {
    @Override
    public void call(CupModel cupModel) throws Exception {
        System.out.println(cupModel.getYear()+" "+cupModel.getFirst());

    }
});
```

```
21/12/14 09:44:25
1982 Italy
2006 Italy
1934 Italy
1938 Italy
21/12/14 09:44:25
```

```java
JavaRDD<String> flatmapRDD = Raw_Data.flatMap(new FlatMapFunction<String, String>() {
    @Override
    public Iterator<String> call(String s) throws Exception {
        return Arrays.asList(s.split( regex: ",")).iterator();

    }
});

flatmapRDD.foreach(new VoidFunction<String>() {
    @Override
    public void call(String s) throws Exception {
        System.out.println(s);
    }
});
```

```
21/12/14 10:58:09 INFO HadoopRDD: Input s
1982
Spain
Italy
Germany FR
Poland
France
146
24
52
2.109.723
1986
Mexico
```

RDD OPERATIONS

- Transformation: The result of operations on an rdd creates a new rdd.
- (flatmap, mappartitions, distinct, sample, substract…)
- Actions: Calculation and saving operations on an rdd are done. (count, first)

```java
public class Distinct {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");
        JavaSparkContext cont = new JavaSparkContext(conf);
        String path = "C:\\bin\\WorldCup\\WorldCups.csv";
        JavaRDD<String> Raw_Data = cont.textFile(path);

        JavaRDD<String> distinct = Raw_Data.distinct();
    }
}
```

```java
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");
    JavaSparkContext cont = new JavaSparkContext(conf);
    String path = "C:\\bin\\WorldCup\\WorldCups.csv";
    JavaRDD<String> Raw_Data = cont.textFile(path);

    JavaRDD<String> sample = Raw_Data.sample( withReplacement: false,  fraction: 0.5);
    sample.foreach(new VoidFunction<String>() {
        @Override
        public void call(String s) throws Exception {
            System.out.println(s);
        }
    });
}
```

```java
public class Union {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");
        JavaSparkContext cont = new JavaSparkContext(conf);
        String first_path = "C:\\bin\\WorldCup\\WorldCups.csv";

        JavaRDD<String> Raw_Data1 = cont.textFile(first_path);
        System.out.println(Raw_Data1.count());

        String second_path = "C:\\bin\\WorldCup\\WorldCupPlayers.csv";
        JavaRDD<String> Raw_Data2 = cont.textFile(second_path);
        System.out.println(Raw_Data2.count());

        JavaRDD<String> unionsample = Raw_Data1.union(Raw_Data2);
        System.out.println(unionsample.count());
    }
}
```

PAIRRDD

```java
JavaRDD<CupModel> map = Raw_Data.map(new Function<String, CupModel>() {
    @Override
    public CupModel call(String line) throws Exception {
        String[] split = line.split( regex: ",");
        var cupModel = new CupModel(
                split[0],
                split[1],
                split[2],
                split[3],
                split[4],
                split[5]);
        return cupMo
    }
});

JavaPairRDD<String, String> JavaPairRDD = map.mapToPair(new PairFunction<CupModel, String, String>() {
    @Override
    public Tuple2<String, String> call(CupModel cupModel) throws Exception {
        return new Tuple2<String, String>(cupModel.getFirst(), cupModel.getSecond());
    }
});
```

Declare final

Declare var type

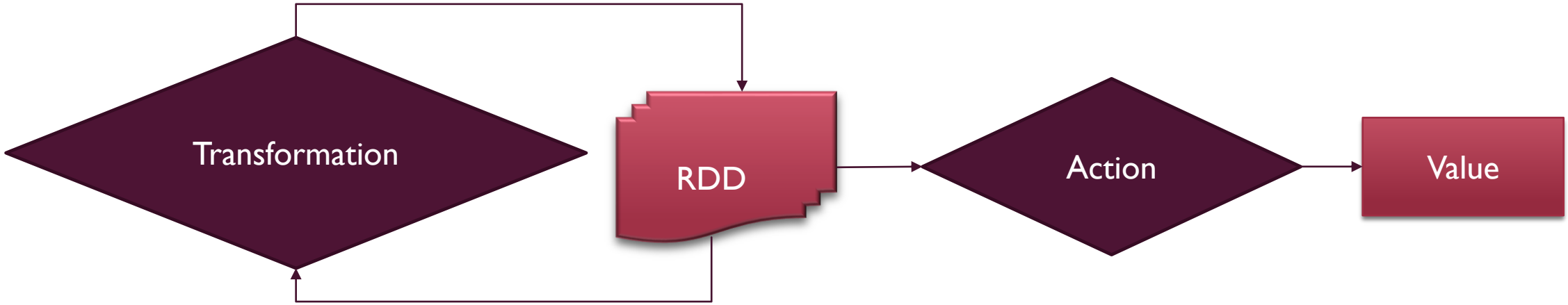
```java
JavaPairRDD.foreach(new VoidFunction<Tuple2<String, String>>() {
    @Override
    public void call(Tuple2<String, String> line) throws Exception {
        System.out.println(line._1 + "  "+ line._2);
    }
});
```

PAIRRDD

```
org.apache.spark.api.java.JavaPairRDD<String, Iterable<String>> result = JavaPairRDD.groupByKey();

result.foreach(new VoidFunction<Tuple2<String, Iterable<String>>>() {
    @Override
    public void call(Tuple2<String, Iterable<String>> line) throws Exception {
        System.out.println(line._1+ "" + line._2);
    }
});
```

LAZY EVALUATION

- Transformation işlemlerinde herhangi bir işlem yapılmaz. Spark, action metodunu görünce işlem başlatır. Spark, action yöntemini gördüğünde değere dönüştürür.



LAZY EVALUATION

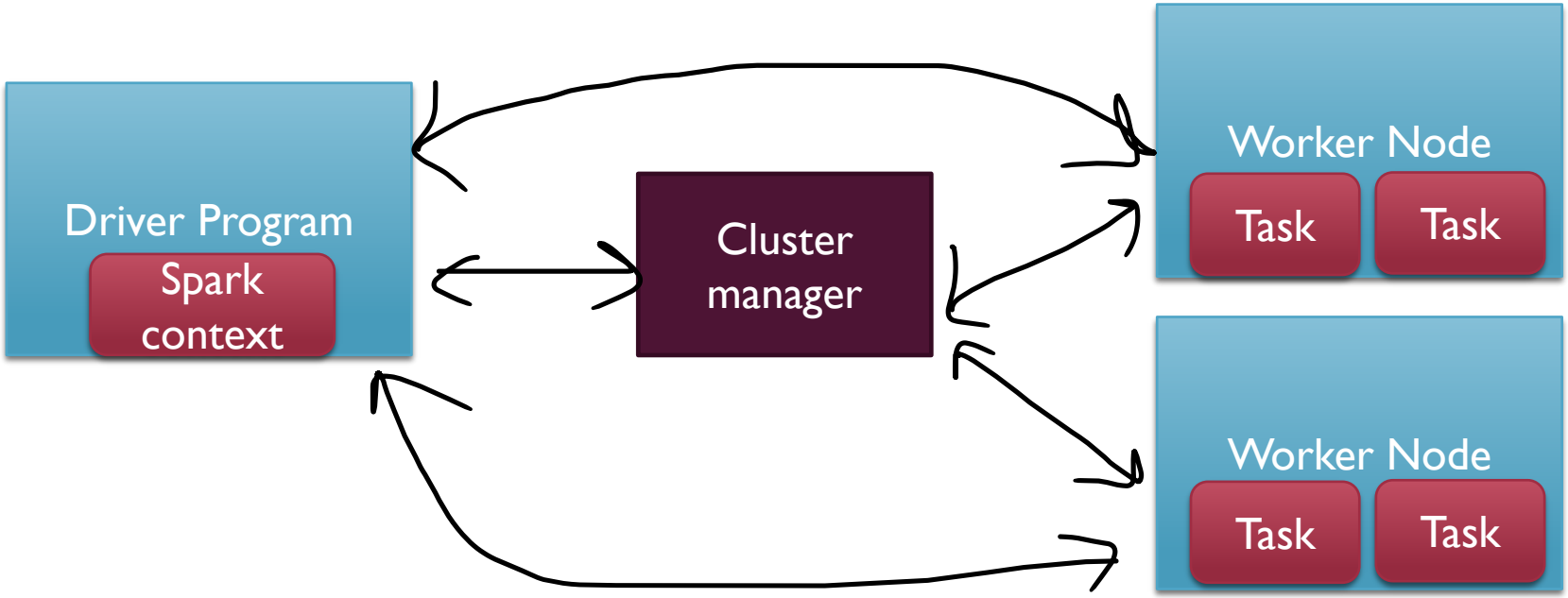
```
JavaRDD<CupModel> map = Raw_Data.map(new Function<String, CupModel>() {
    @Override
    public CupModel call(String line) throws Exception {
        System.out.println(line);
        String[] split = line.split( regex: ",");
        var cupModel = new CupModel(
                split[0],
                split[1],
                split[2],
                split[3],
                split[4],
                split[5]);
        return cupModel;
    }
});

System.out.println(map.count());
```

ACTION METHOD

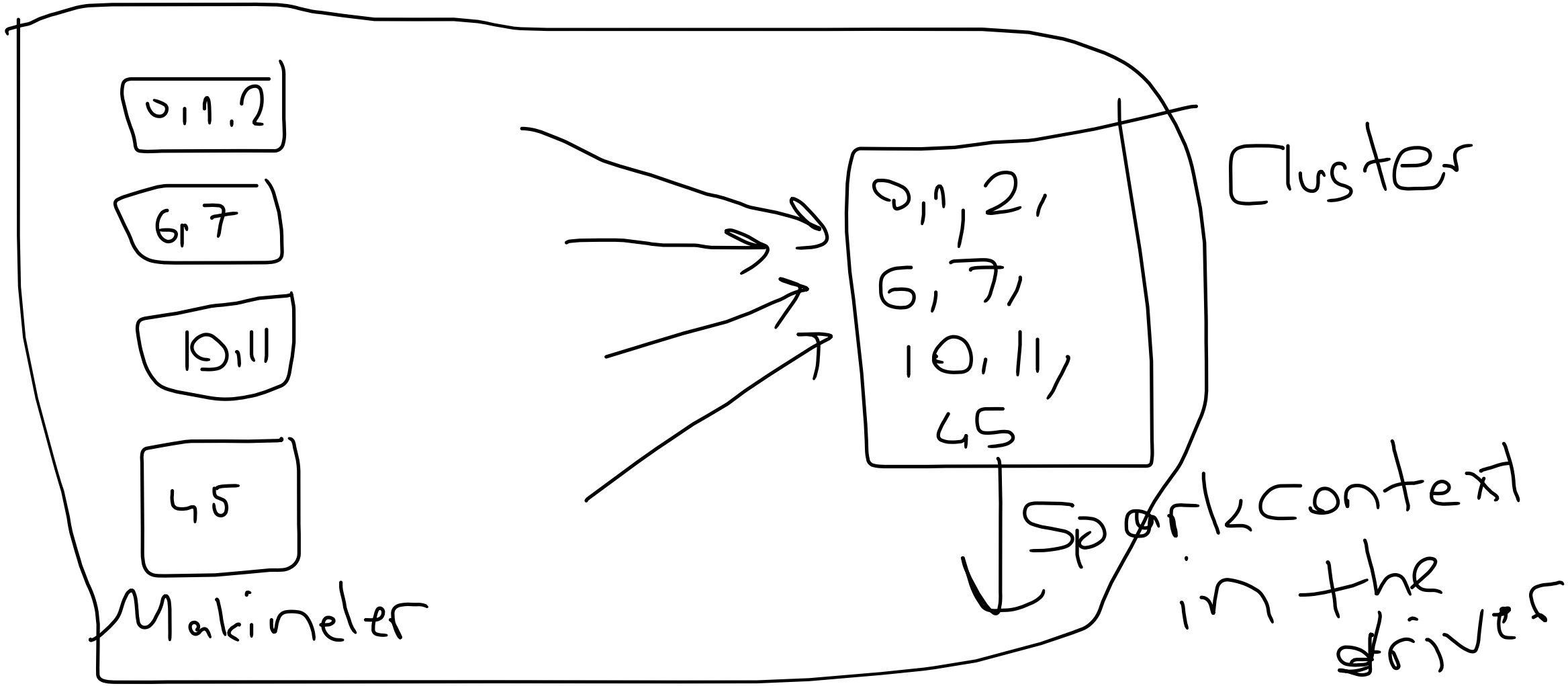
- Count
- First
- Collect
- Foreach
- Filter
- Take vb.

COLLECT

- It sends the data on the machines in the cluster to the master machine.



COLLECT


0,1,2
6,7
10,11
45
Makineler
0,1,2,
6,7,
10,11,
45
Cluster
SparkContext
in the
driver

COLLECT

```java
public class Collect {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");
        JavaSparkContext cont = new JavaSparkContext(conf);
        String path = "C:\\bin\\WorldCup\\WorldCups.csv";
        // read text file to RDD
        JavaRDD<String> Raw_Data = cont.textFile(path);

        for (String s : Raw_Data.collect()) {
            System.out.println(s);
        }


    }
```

TAKE

```
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");
    JavaSparkContext cont = new JavaSparkContext(conf);
    String path = "C:\\bin\\WorldCup\\WorldCups.csv";
    // read text file to RDD
    JavaRDD<String> Raw_Data = cont.textFile(path);

    for (                                  ata.collect()) {
        S                                  n(s);
    }
    List<String> take = Raw_Data.take(4);

}
```

Declare final

Declare var type

TAKESAMPLE-SAVEASTEXTFILE

```
List<String> take = Raw_Data.take( num: 4);
List<String> takesample = Raw_Data.takeSample( withReplacement: false,  num: 5);
```

```
Raw_Data.saveAsTextFile( path: "C:\\bin\\Raw_Data");
```