WIZELINE ACADEMY

Grow your career:
Free courses in Artificial Intelligence, Software Development, User Experience and More

Wi-Fi: Academy
Password: academyGDL
Slack Channel: http://bit.ly/slackacademy

@WizelineAcademy

academy.wizeline.com

/WizelineAcademy

Get notified about courses:
tinyurl.com/WL-academy

Week 1 - Session 1/2

Big Data Engineering with Spark

Spark Basics

WIZELINE

Academy Code of Conduct

- **Be friendly and patient.**
- **Be welcoming:** We strive to be a community that welcomes and supports people of all backgrounds and identities.
- **Be respectful:** Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners.
- **Be careful in the words that you choose:** We are a community of professionals, and we conduct ourselves professionally.
- **Be honest**: We encourage you to share and discuss ideas, but we expect you to write your own code.
- **Be mindful of the resources:** Cloud resources are expensive, don’t create new resources or exploit the existing ones.

Academy Communications

All communications will be done via Slack.

To setup your account:

- Enter the following link for the Academy slack : http://bit.ly/slackacademy
- Join the **Data-Eng-Academy** channel
- Join the **de-mentee-<mentor-name>** channel

Big Data Engineering with Spark

By the end of this course you will be able to:

- Design and create your own Spark applications.
- Navigate the Google cloud environment
- Submit and troubleshoot Spark jobs in Google cloud.
- Optimize your Spark applications based on your knowledge of distributed processing.

Big Data Engineering with Spark - Class descriptions

Week 1: Spark Basics

- Class 1 Spark Hello World and Architecture
- Class 2 MapReduce theory and Fault tolerance

Week 2: Operations

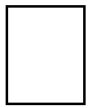
- Class 3 Actions, Transformations and Caching
- Class 4 Aggregation Operations, and Key-Values in: Dataframes and Datasets.

Week 3: Partitioning and Shuffling

- Class 5 Intro to shuffling and Partitioning
- Class 6 Optimizations Shuffling

Week 4: Structured data

- Class 7 Joins and broadcast variables.
- Class 8 SQL and more



How to get ITESM Certification

Attend!

- Attend to all of the 8 sessions we will be hosting.

Survey

- Every module finishes with a survey
- Answer it to **confirm your assistance**
- Your feedback is important!

Work and code

- Though there is no final project, we do require that you complete 80% of your weekly assignments with a score of at least 80/100 points each.

Get certification

- Show off your certification on your profile!

Homeworks

Submission means: dataproc job submission interface (not from zeppelin).
Submission attempts: Not restricted.
Grading: Your last attempt.
Deadline: Sunday 11:59pm for both homeworks.

Refer to your survival kit for the detailed instructions.

Working environment

You will be connecting to a handled spark environment in google cloud dataproc and google cloud storage.

The dataproc clusters will be available **daily** from **12pm until 12am**:

- Before midnight you must **save all your code** in your **local computer** as the **clusters** will be **destroyed**.

The storage buckets will be available for the whole month.

Exercises

During the sessions we will be doing exercises on Zeppelin:

1. Start the cloud shell
2. Input:

```
gcloud compute ssh --project data-castle-bravo --zone us-central1-a --ssh-flag="-L
8080:localhost:8080" de-training-<user-name>-m
```

Refer to your survival kit for the detailed instructions.

Teachers

Carlos Zubieta
Data Engineer
Guadalajara, JAL, Mexico

Ricardo Magana
Data Engineer
Guadalajara, JAL, Mexico

Abraham Alcantara
Data Engineer
Guadalajara, JAL, Mexico

Teachers and mentors introductions

Rodrigo Chaparro Plata
Guadalajara, JAL, Mexico

Willebaldo Gomez
Guadalajara, JAL, Mexico

Edgar Arenas
Guadalajara, JAL, Mexico

Said Montiel
Guadalajara, JAL, Mexico

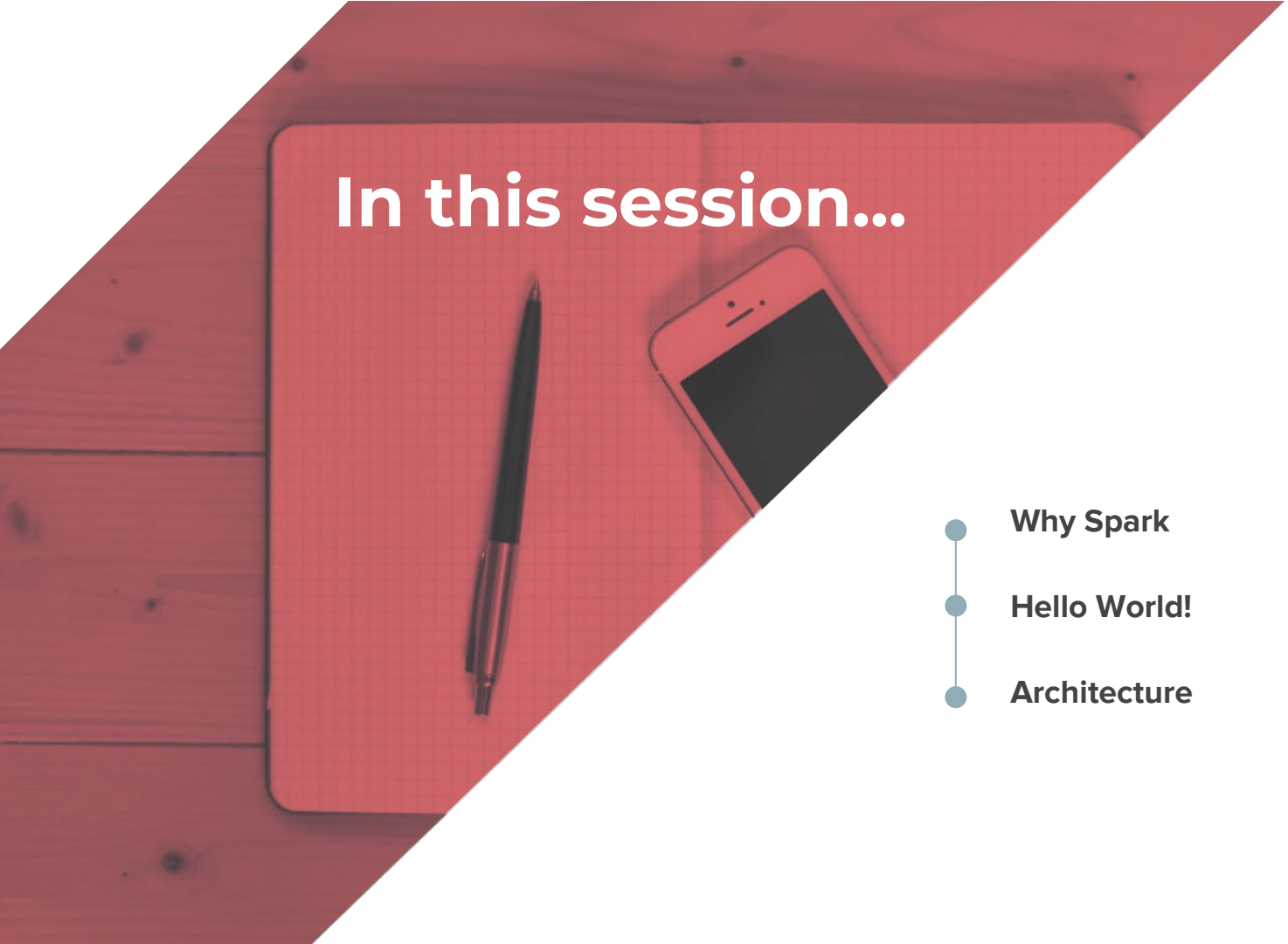
Ana Costilla
Guadalajara, JAL, Mexico

Mentor and mentees ice breaker

Introduce yourself with your mentor group and state:

1) Your name
2) Your job and responsibilities
3) What you expect from the academy
4) Your favourite childhood game.

In this session...



- Why Spark
- Hello World!
- Architecture

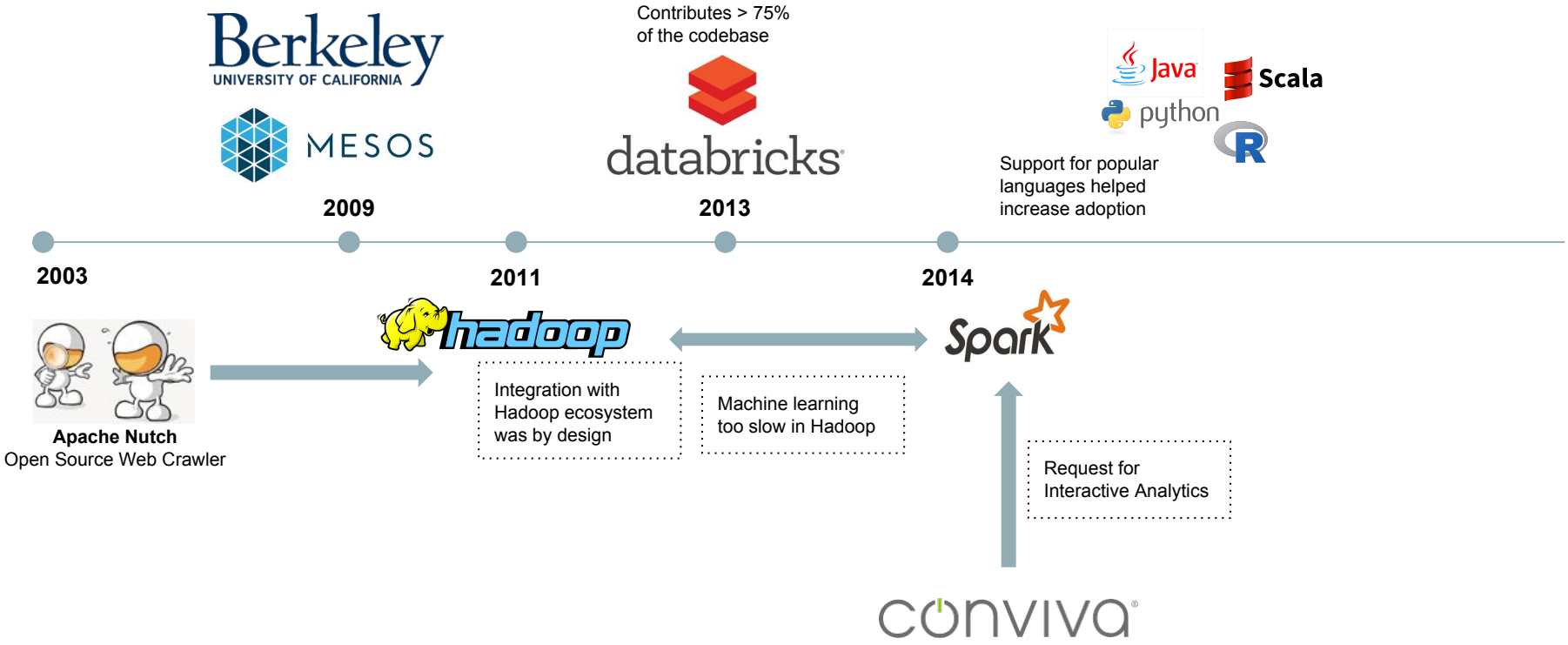
By the end of this session, you'll be able to explain:

- **Why Spark exists**
- **When to (and when not to) use Spark**
- **How "Hello World" looks like in Spark**
- **Spark's Ecosystem and Architecture**

WHY SPARK

Why Spark

A brief Story



Why Spark

What is Spark

Apache Spark™ is a unified analytics engine for large-scale data processing.

Speed

Run workloads 100x faster.

Ease of Use

Write applications quickly in Java, Scala, Python, R, and SQL.

Runs Everywhere

Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud. It can access diverse data sources.

Why Spark

Use Cases

Large Data

If your data fits into a single machine, why bother with Spark?

Iterative Algorithms (for example, Large Scale Machine Learning)

Do you need to make several passes over your data? Spark's in-memory caching can be a great advantage in this case.

Interactive Analytics

If you need to do interactive data queries, Spark's in-memory model can be orders of magnitude faster than Hadoop.

Spark Recap

What We've Learned So Far

- What Spark is
- The history and motivation behind Spark
- Some of the most common use cases of Spark

QA


HELLO WORLD!

Hello World

Word Count Problem

Problem

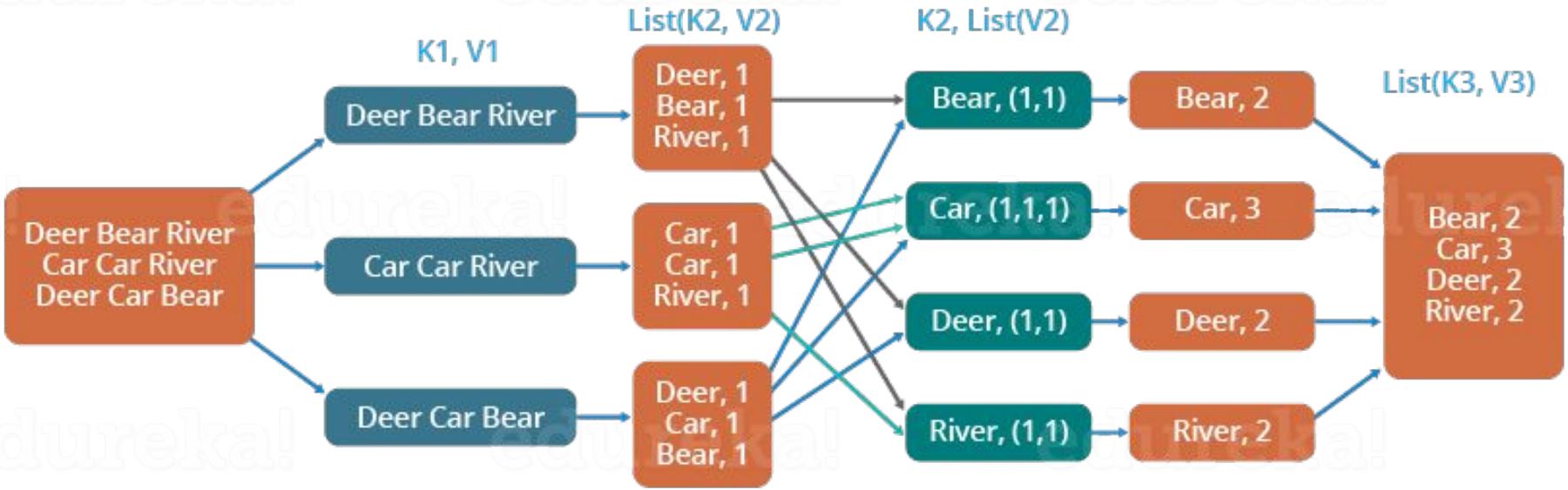
Find the frequency of appearance of each word in a set of documents.

Constraints

- The documents don’t fit in a single commodity machine (e.g. 10 TB)
- The execution time must be quite short (i.e. more than a few minutes is unacceptable)

Hello World

Distributed Solution Sketch



Hello World

Word Count in Hadoop

```java
import org.apache.hadoop.*;

public class WordCount {
    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
    public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
}
```

Everything outside of the dashed boxes is mostly boilerplate code

To the rescue!

This is the core of the program

Low-Level Computing Model

Hello World

Word Count in Spark

```
package com.wizeline.wordcount

import org.apache.spark.sql._

object WordCount {
  def wordCount(
      documents: Dataset[String],
      separatorsRegexp: String = """\s+""") : Dataset[(String, Long)] =
    {
      val words = documents.flatMap(doc => doc.split(separatorsRegexp))
      val lcWords = words.map(word => word.toLowerCase)
      val groups = lcWords.groupByKey(identity)
      val counts = groups.count()
      counts
    }
}
```

This is boilerplate code

This is the core of the program

To the rescue!

High-Level, Descriptive Computing Model

Hello World

Word Count Dissected: **map**

```
1  val words = documents.flatMap(doc => doc.split(separatorsregexp))
2  val lcwords = words.map(word => word.tolowercase)
3  val groups = lcwords.groupByKey(identity)
...
```

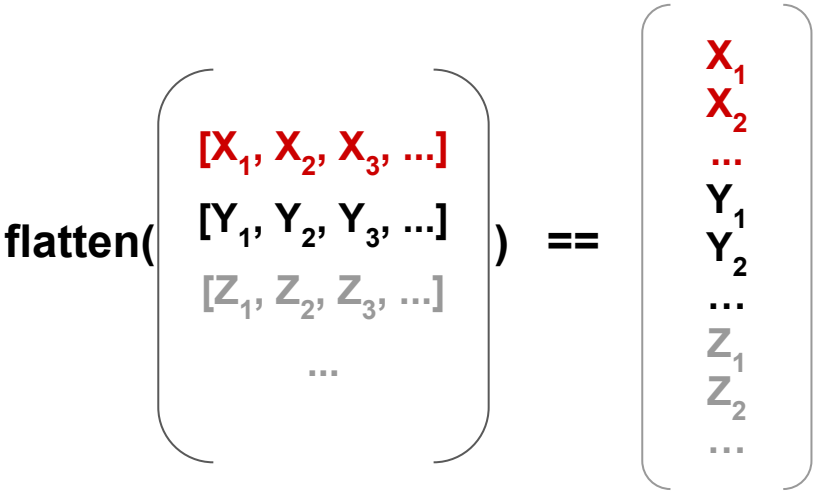
$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ \dots \end{pmatrix} .map(fn) == \begin{pmatrix} fn\ (X_1) \\ fn\ (X_2) \\ fn\ (X_3) \\ fn\ (X_4) \\ fn\ (X_5) \\ \dots \end{pmatrix}$$

We skipped line 1 because to understand the **flatMap** function we first need to understand the **map** function, but we'll get back to it in a couple of slides...

```
["HELLO", "World!", "BYE", "WORLD!"].map(word => word.toLowerCase)
        → [ "hello", "world!", "bye", "world!" ]
```

Hello World

Word Count Dissected: `flatten`




```
flatten([["HELLO", "World!"], ["BYE", "WORLD!"]])
    → [ "HELLO", "World!", "BYE", "WORLD!" ]
```

Hello World

Word Count Dissected: `flatmap`

`"Wizeline rocks!".split("\s+") => ["Wizeline", "rocks!"]`

```
1 val words = documents.flatMap(doc => doc.split(separatorsRegexp))
2 val lcWords = words.map(word => word.toLowerCase)
```

...

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \dots \end{bmatrix} .\text{flatMap}(fn) == \text{flatten}\left(\begin{bmatrix} fn\ (X_1) \\ fn\ (X_2) \\ fn\ (X_3) \\ \dots \end{bmatrix}\right) == \text{flatten}\left(\begin{bmatrix} [X_{11}, X_{12}, X_{13}, \dots] \\ [X_{21}, X_{22}, X_{23}, \dots] \\ [X_{31}, Z_{32}, Z_{33}, \dots] \\ \dots \end{bmatrix}\right) == \begin{bmatrix} X_{11} \\ X_{12} \\ \dots \\ X_{21} \\ X_{22} \\ \dots \end{bmatrix}$$

fn must return a list

`["HELLO World!", "BYE WORLD!"].flatMap(doc => doc.split("\s+"))`

→ `["HELLO", "World!", "BYE", "WORLD!"]`

Hello World

Word Count Dissected: **groupBy**

```
2 val lcwords = words.map(word => word.tolowercase)
3 val groups = lcwords.groupByKey(identity)
4 val counts = groups.count()
```

…

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \dots \end{pmatrix} \textbf{.groupByKey(}\textit{keyExtractor}\textbf{)} == \begin{pmatrix} (K_1, [X_{1,1}, X_{1,2}, \dots]) \\ (K_2, [X_{2,1}, X_{2,2}, \dots]) \\ \dots \end{pmatrix}$$

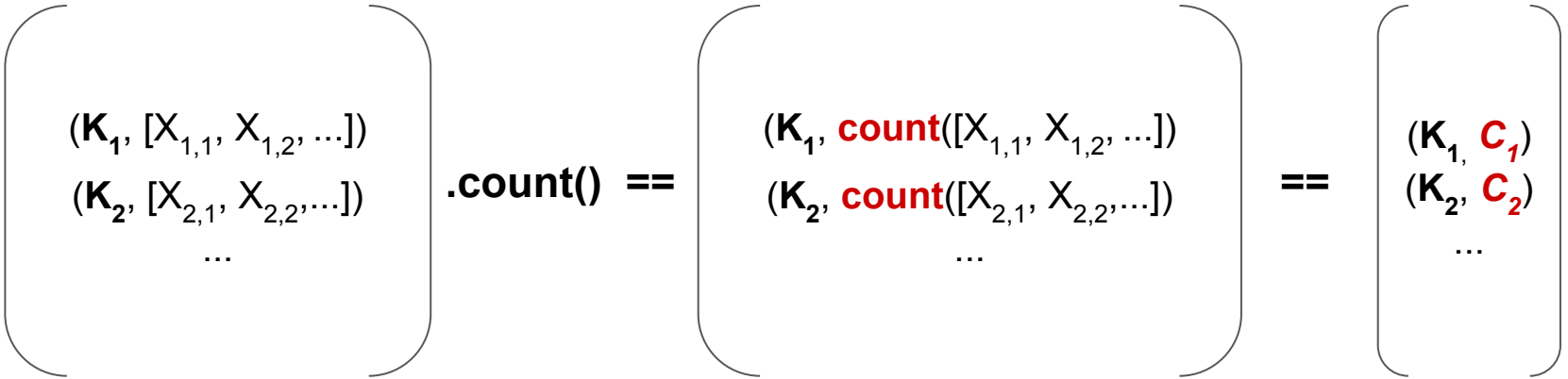
$K_i \leftarrow keyExtractor(X_{i,1})$
$K_i \leftarrow keyExtractor(X_{i,2})$
…

```
[“hello”, “world!”, “bye”, “world!”].groupByKey(x => x)
   → {“hello”: [“hello”], “world!”: [“world!”, “world!”], “bye”: [“bye”]}
```

Hello World

Word Count Dissected: `count`

```
3 val groups = lcwords.groupByKey(identity)
4 val counts = groups.count()
5 counts
```



{“**hello**”: [“hello”], “**world!**”: [“world!”, “world!”], “**bye**”: [“bye”]}.**count()**
→ [(“**hello**”: 1), (“**world!**”, 2), (“**bye**”, 1)]

Hello World

Word Count in Spark

```
package com.wizeline.wordcount

import org.apache.spark.sql._

object WordCount {
  def wordCount(
      documents: Dataset[String],
      separatorsRegexp: String = """\s+""") : Dataset[(String, Long)] =
    {
      val words = documents.flatMap(doc => doc.split(separatorsRegexp))
      val lcWords = words.map(word => word.toLowerCase)
      val groups = lcWords.groupByKey(identity)
      val counts = groups.count()
      counts
    }
}
```

This is boilerplate code

This is the core of the program

To the rescue!

Do you understand it all now?

Hello World

What We've Learned So Far

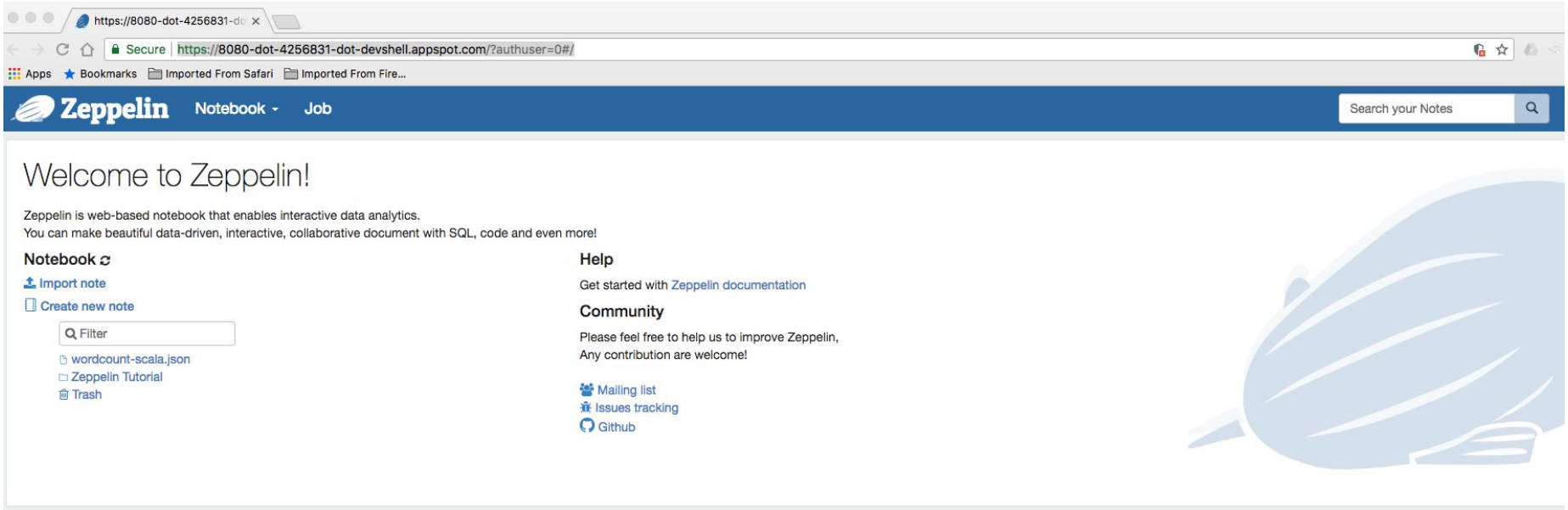
- What "Hello World" looks like in Spark
- Some commonly used functions in data processing
  - Map
  - flatMap
  - groupByKey
  - count
- Some of the most common use cases of Spark

QA

Hello World

Zeppelin in the Cloud

First, follow the instructions provided in your “Survival Kit” to set up the connection to your cluster and access the Zeppelin application

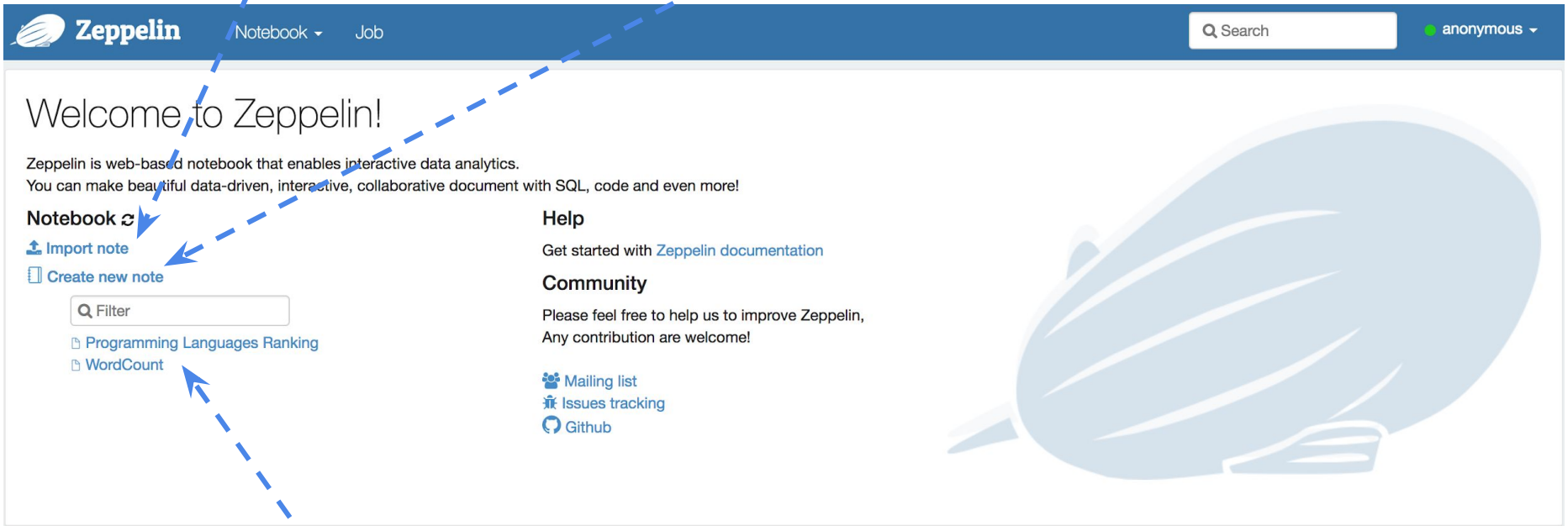


Hello World

Zeppelin in the Cloud

To run the example, you need to import a notebook here.

You can also create new notebooks here.



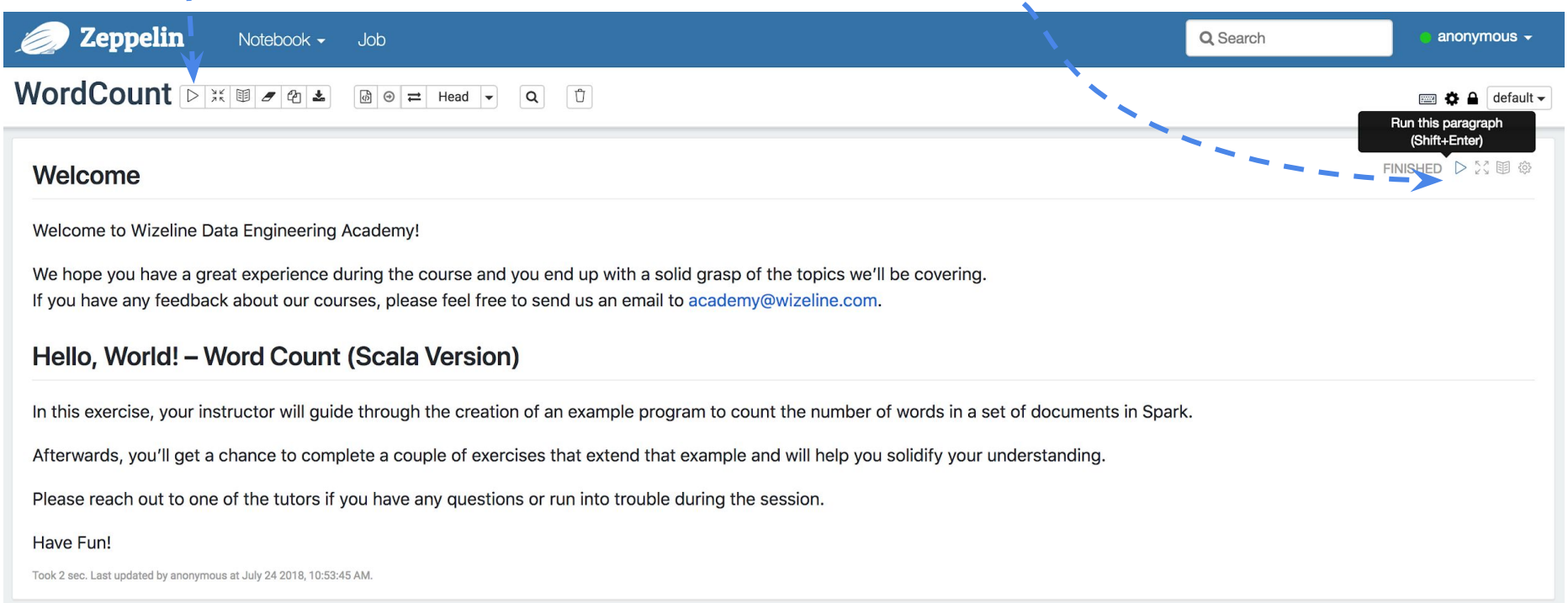
Afterwards, you should see your notebook in the list of recently opened notebooks here.

Hello World

Zeppelin in the Cloud!

Notebooks are divided into cells. You can run all cells in the notebook by clicking here.

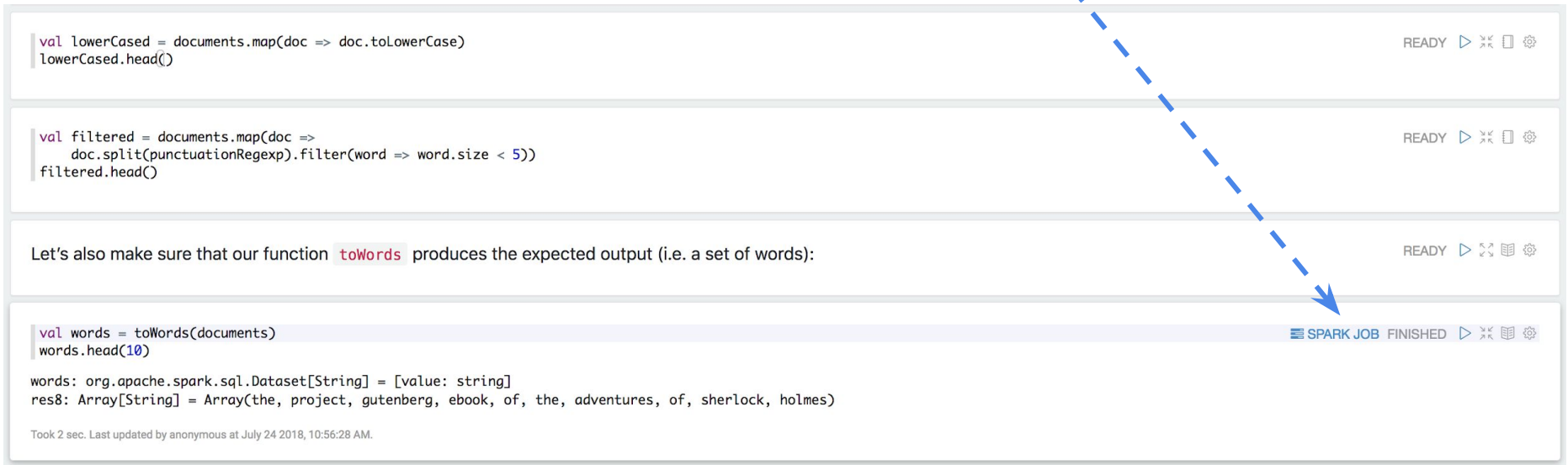
Or you can “execute” the contents of each cell by clicking here, (shortcut: **Shift** + **Enter**).

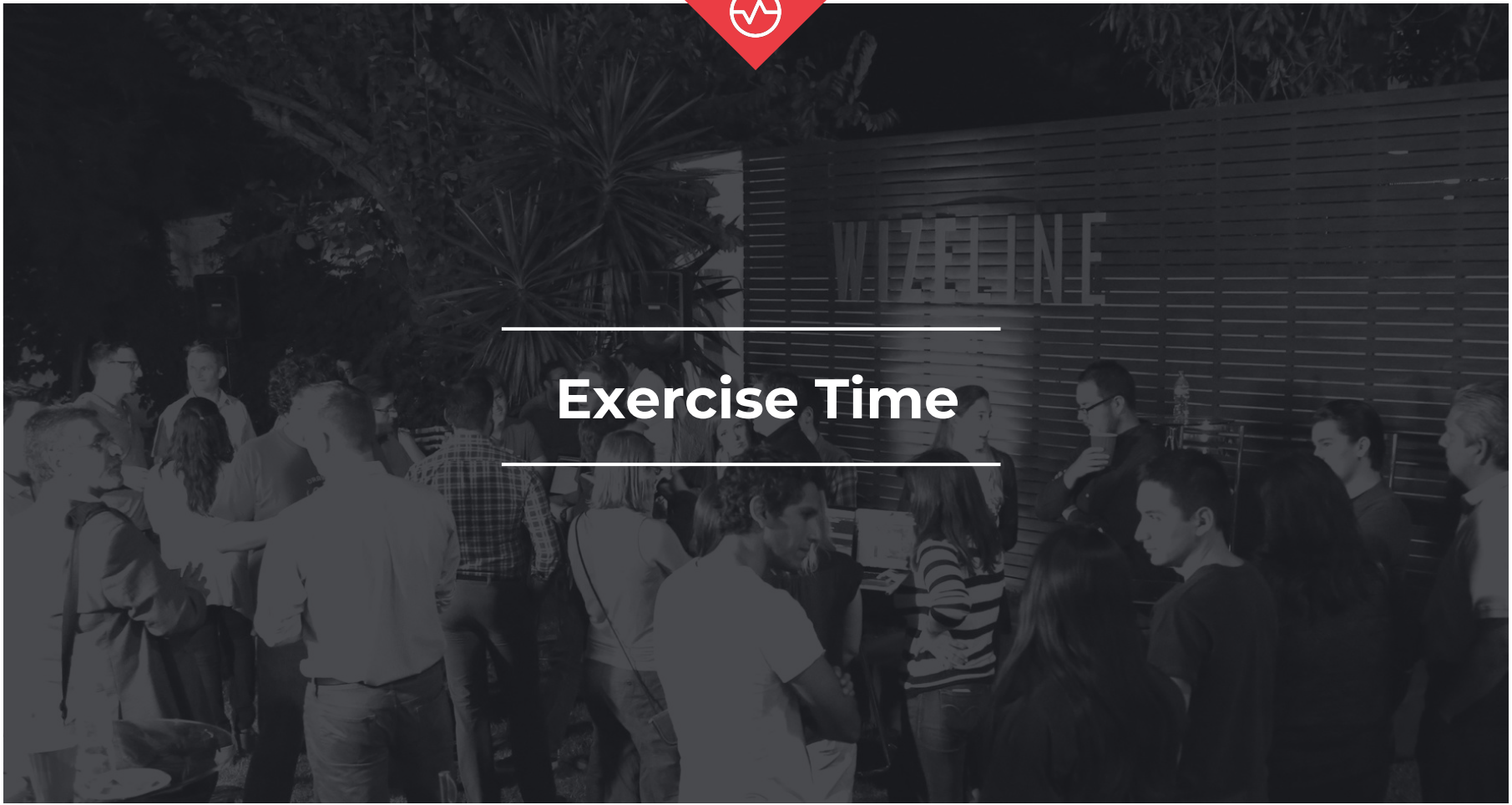


Hello World

Zeppelin in the Cloud!

While a cell is running (but also once it's done), you can see further information about the underlying Spark job by clicking here.





Exercise Time

Exercises

Count Characters

Using the same dataset of the example, **can you count the number of characters in the set of documents?**

For example, if the set of documents were:

```
documents = [ “my friend”, “told me”, “to”, “go”, “home” ]
```

then the expected answer would be:

```
counts = [
  (‘e’, 3), (‘o’, 3), (‘y’, 1), (‘f’, 1), (‘r’, 1), (‘i’, 1),
  (‘m’, 3), (‘n’, 1), (‘d’, 1), (‘t’, 1), (‘ ’, 2), (‘g’, 1),
  (‘h’, 1)
]
```

Exercises

Longest Words

Using the same dataset of the example, **can you find the top 10 longest words and how many of each there are in the dataset?**

Exercises

Longest Words - Hints

Using the same dataset of the example, **can you find the top 10 longest words and how many of each there are in the dataset?**

***Hint 1**: How can you extend the Word Count example to solve part of this problem?*

***Hint 2**: Can you find a method to order a dataset by a given key in the Dataset API documentation?*

Exercises

Anagram Sets

Using the same dataset of the example**, can you find all anagram sets with at least two words?**

Remember that ***two words are anagrams of each other if they contain the same number of occurrences of each letter***.

For example:

- **areAnagrams(**“mar”**,** “ram”**) == true**,
- **areAnagrams(**“line”**,** “nilee”**) == false**

Exercises

Anagram Sets

Consider the following set of documents:

documents = [“car art”, “rat arc”]

In this case, there are two such sets: [“rat”, “art”] and [“car”, “arc”] because they contain at least two elements.

Now consider the following set of documents:

documents = [“wizeline rocks”, “chuck norris”]

In this case, there are no such sets.

Exercises

Anagram Sets - Hints

Using the same dataset of the example**, can you find all anagram sets with at least two words?**

***Hint 1**: Can you find a method to filter out elements in the Dataset API documentation?*

***Hint 2:** Can sorting the characters in each word help you group them into anagrams of each other?*

***Hint 3**: Are you getting groups with repeated words? Check out the Dataset API for a way to obtain unique words.*

Exercises Recap

What We've Learned So Far

- How to use Zeppelin notebooks
- What "Hello World" ("Word Count") looks like in Spark
- How to modify it to solve a slightly different version of the same problem
- How to solve some more difficult problems using the functions we learned in "Hello World"

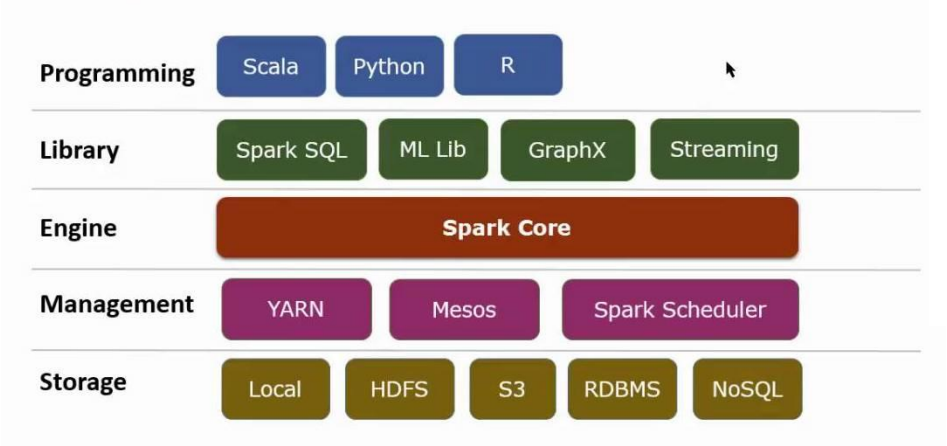
QA

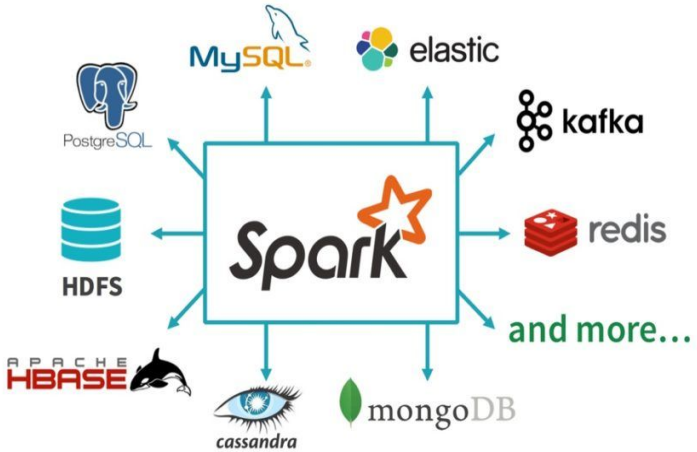

ARCHITECTURE

Architecture

Framework and Ecosystem

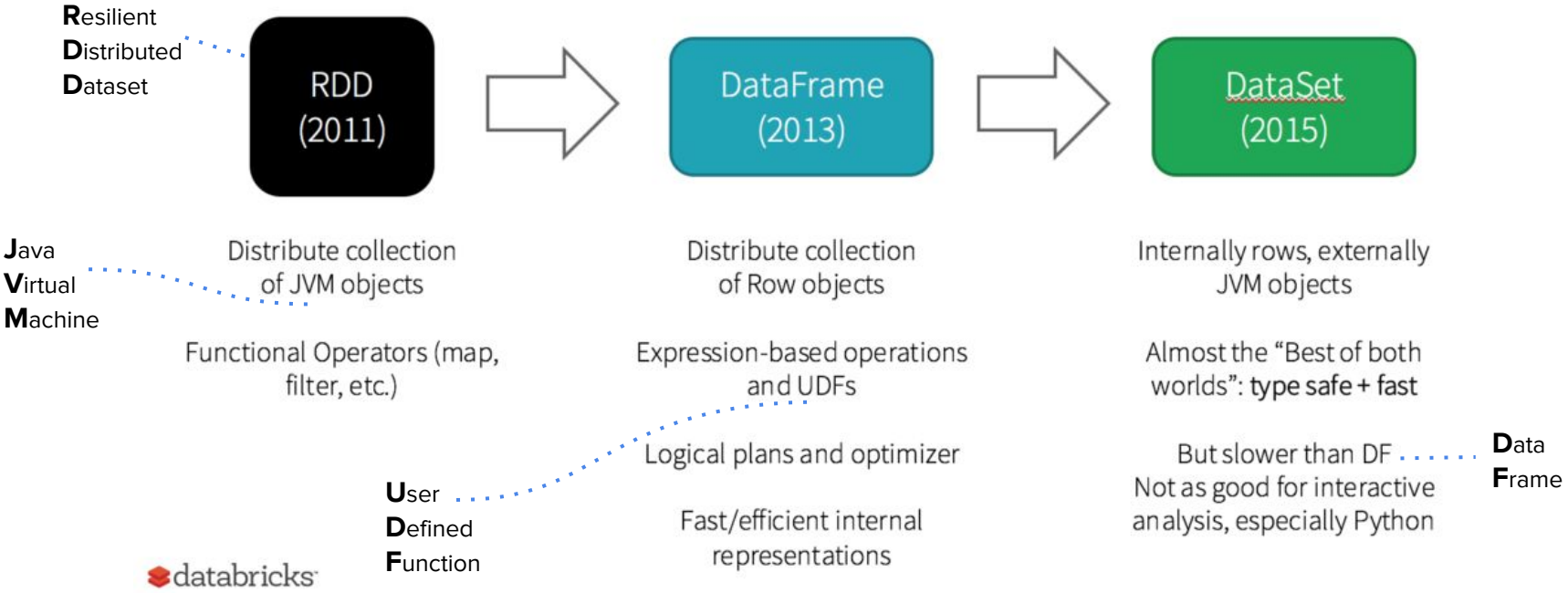
Apache Spark™ is a unified analytics engine for large-scale data processing.





Architecture

Evolution of Spark APIs



Architecture

Datasets

Dataset [T]

Datasets are a **type-safe** version of Spark's structured API for Java and Scala.

The **Dataset** class is **parameterized** with the type of object contained inside.

As of Spark 2.0, the types **T** supported are all classes following the **JavaBean pattern** in Java, and **case classes** in Scala.

Architecture

Datasets

When should I use Datasets?

1. When your data is structured and has a schema.
2. When you want to benefit from compile-time type safety on your data.
3. When you want to take advantage of optimization and performance benefits available in **Datasets** for structured and semi-structured data.

Architecture

Dataframes

DataFrame

A DataFrame is a **distributed** collection of data organized into **named columns**.

It is **conceptually equivalent to a table** in a relational database, but with **richer optimizations** under the hood.

Architecture

Dataframes

When should I use DataFrames?

1. When your data is structured and you care about having a schema.
2. When you want to take advantage of your existing knowledge of SQL or Hive (Using Spark SQL, you can run unmodified Hive queries on your existing Hive warehouses.)
3. When you want to take advantage of the query optimizer.

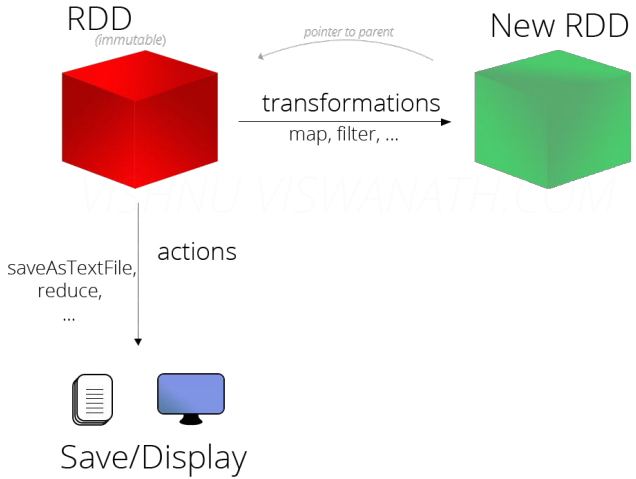
Architecture

Resilient Distributed Datasets

RDDs (**R**esilient **D**istributed **D**atasets) were the primary user-facing API in Spark since its inception.

An RDD is an **immutable distributed** collection of data that is **partitioned** across nodes in a cluster. An RDD can be operated in **parallel** with a **low-level API** that offers **transformations** and **actions**.

RDDs are "**Resilient**" because they can recover from failure in the underlying nodes, preventing data loss.



Architecture

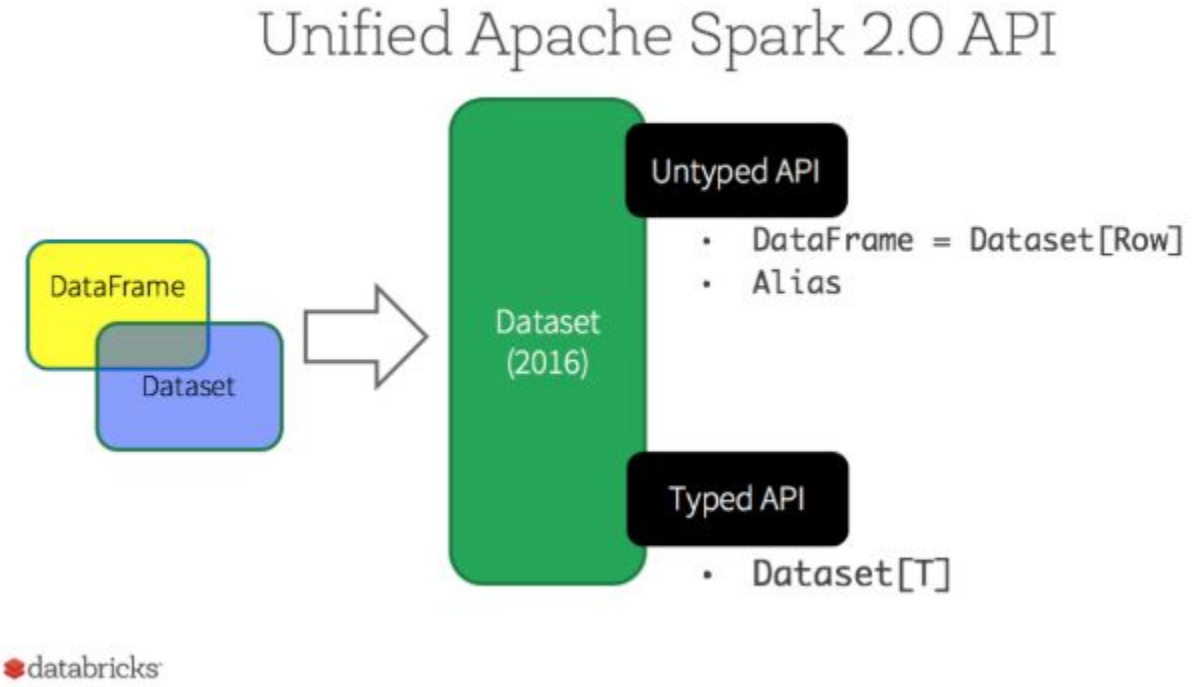
RDDs (Resilient Distributed Datasets)

When should I use RDDs?

1. When you want low-level transformations and control on your dataset.
2. When your data is unstructured, such as media streams or streams of text.
3. When you don’t care about imposing a schema.

Architecture

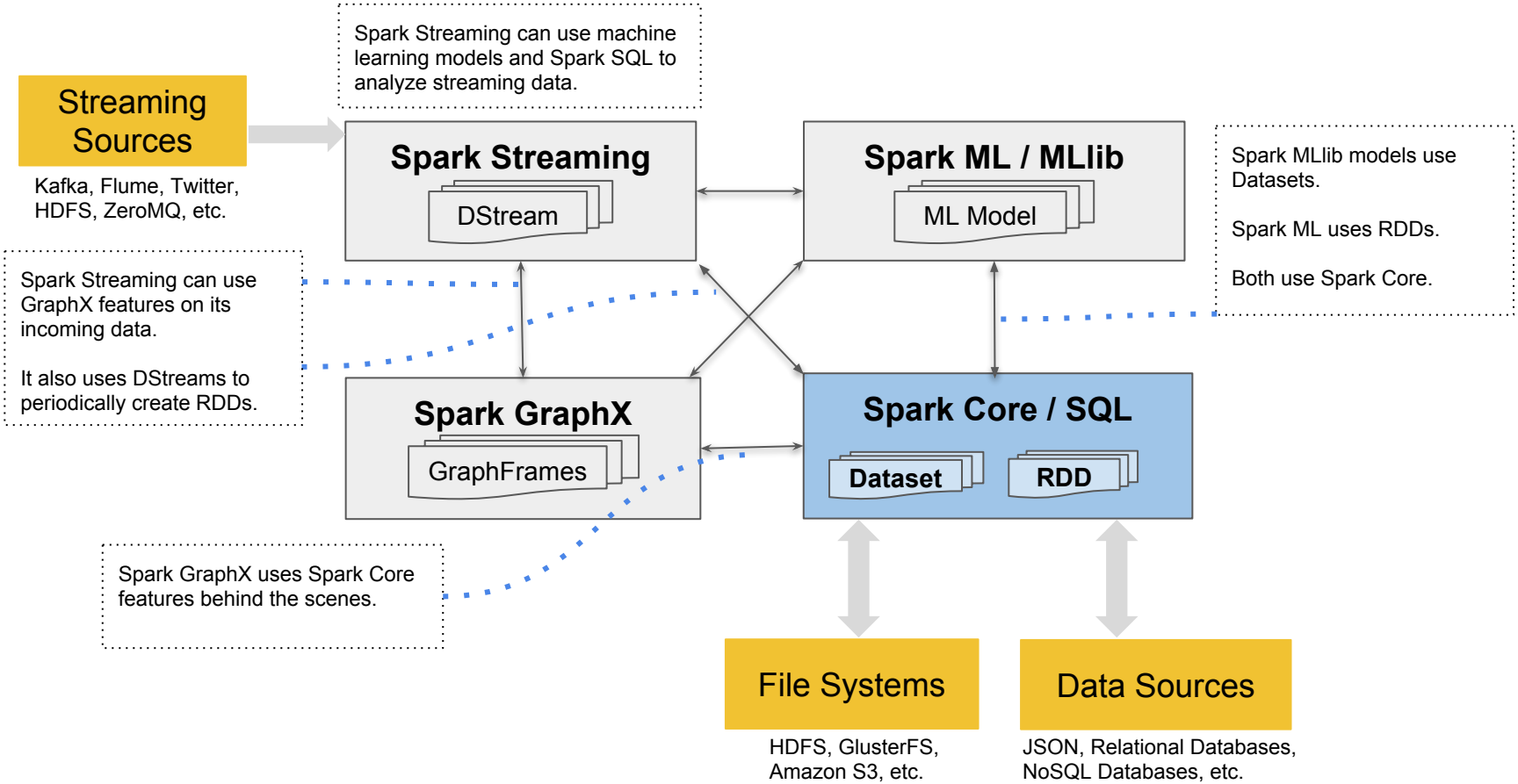
Unification if Datasets and Dataframes



For this course, we'll be using **Datasets** (in Scala) and **Dataframes** (in Python) exclusively

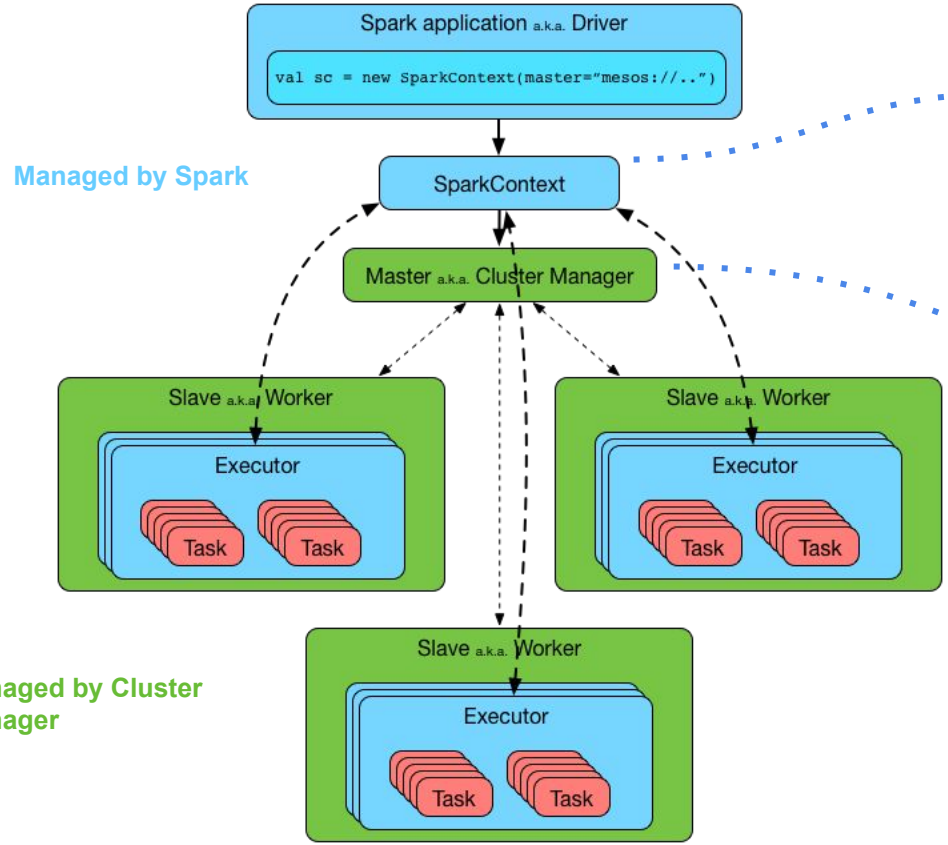
Architecture

Library Components



Architecture

Execution Model



Managed by Spark

Managed by Cluster Manager

`SparkContext` sets up internal services and establishes a connection to a Spark execution environment. It acts as the *master of your Spark application.*

MESOS

hadoop YARN

kubernetes

Cluster managers provide abstractions for CPU, memory, storage, job scheduling, and other compute resources in a distributed cluster.

Architecture

What We've Learned So Far

- That Spark is structured in layers and has a rich ecosystem of technologies around it.
- About Spark execution model based on a master/slaves type of architecture.
- About Spark's APIs, including RDDs, Datasets and Dataframes, and discussed when to use each.

QA

FURTHER READING

Spark
What is Apache Spark
(Databricks)

Big Data Analysis with Scala and Spark
(Coursera MOOC – Week 1)

Architecture
(Mastering Apache Spark ebook)

History of Apache Spark
Based on an interview to Ion Stoica (Databricks co-founder.)

Word Count
Hadoop Implementation
Spark Implementation

For the Knowledge Hungry...

Spark Internals
Introduction to Spark Internals

Spark Data Structures
RDDs vs Dataframes vs Datasets

Word Count
Word Count Implementations with RDDs, Datasets and Dataframes

Spark
Original Paper by Matei Zaharia

RDD (Resilient Distributed Datasets)
Original Paper by Matei Zaharia

Assignment

To Work on Your Own at Home

Introducing “Alimazon”

Client Purchase Orders Schema

“Alimazon” is a fictitious online retail store. We have at our disposal a dataset of client purchase orders (*.jsonl.gz, i.e. gzip-compressed JSON lines) with the following schema:

Data model

name	json type	type	required	notes
id	string	string	yes	uuidv4
timestamp	string	timestamp	yes	iso 8601
client_id	string	string	yes	uuidv4
product_id	string	string	yes	uuidv4
quantity	integer	integer	yes	larger than 1
total	float	float	yes	larger than or equal to 0

Introducing “Alimazon”

Client Purchase Orders Sample

“Alimazon” Client Purchase Orders Sample File:


```
{"id": "238075b7-dc16-40bb-a9a8-85ca5076822d", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-02-19T21:48:08.958189", "product_id": "0892048484", "quantity": 1, "total": 35.39}
{"id": "5a7a6f94-dc1e-4370-b47e-98b45c660d24", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-03-04T21:48:08.958189", "product_id": "0062026852", "quantity": 79, "total": 2670.2}
{"id": "28fb78a4-bd25-47fd-bcaa-6469d93182ec", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-03-17T21:48:08.958189", "product_id": "B0055V48C0", "quantity": 57, "total": 957.6}
{"id": "aba310be-4959-4bcd-afb3-103b3a5c7395", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-03-30T21:48:08.958189", "product_id": "B00DHSROQA", "quantity": 3, "total": 46.65}
{"id": "15dc6b6d-af94-49cf-a5e9-1b1aff9086c7", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-04-12T21:48:08.958189", "product_id": "B0091KJ18M", "quantity": 2, "total": 15.34}
{"id": "f6211323-67e7-4428-b6c3-57602acaf4e7", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-05-08T21:48:08.958189", "product_id": "B00FFL90E6", "quantity": 72, "total": 3121.92}
{"id": "c913f04e-abe8-4143-bfb6-64864b19b4e6", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-05-21T21:48:08.958189", "product_id": "B000FPVUAO", "quantity": 28, "total": 1886.64}
{"id": "2f9d9dab-7325-4055-b0dd-2f34fe0c8115", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-06-03T21:48:08.958189", "product_id": "B0002D0MHG", "quantity": 37, "total": 456.58}
{"id": "a2aa54b2-87a9-4293-914e-bf490ae7c1be", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-06-29T21:48:08.958189", "product_id": "B00IEOGLNE", "quantity": 97, "total": 1265.85}
{"id": "5b09dd98-7b25-4e31-8551-5d8eef56e301", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-07-12T21:48:08.958189", "product_id": "B0060JJLXC", "quantity": 34, "total": 3294.6}
{"id": "985d9a44-cf5a-4c48-a69b-0c03a5768b35", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-07-25T21:48:08.958189", "product_id": "B000YABFA0", "quantity": 73, "total": 2197.3}
{"id": "9ae1e54f-fd8d-4fa1-b2fc-c023e78dd97d", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-08-07T21:48:08.958189", "product_id": "B007DJ3KIU", "quantity": 53, "total": 5759.51}
{"id": "3fa09a8c-435f-4cfe-bd5f-002c66069953", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-08-20T21:48:08.958189", "product_id": "B005A2XLVI", "quantity": 32, "total": 475.84}
{"id": "cd6ebf8e-039c-47f6-a24a-5d19f250e7a1", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-09-02T21:48:08.958189", "product_id": "0394712587", "quantity": 7, "total": 328.02}
{"id": "2edc0716-6a94-42a8-a97b-bc6e24973da9", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-09-15T21:48:08.958189", "product_id": "B00297LW68", "quantity": 28, "total": 578.48}
{"id": "3f6fe102-ceb1-4662-a63a-0c4d60b053ad", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-09-28T21:48:08.958189", "product_id": "B00S01G0LO", "quantity": 45, "total": 3021.75}
{"id": "f3ecbdfe-bc05-4756-b8a8-65b016b473fd", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-10-11T21:48:08.958189", "product_id": "B000A0CAWQ", "quantity": 4, "total": 180.08}
{"id": "69a8c9d1-14fe-469a-a5c7-555d1d491f75", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-10-24T21:48:08.958189", "product_id": "B00E5AYKTI", "quantity": 83, "total": 5525.31}
{"id": "9fd3a0f2-fabf-4d46-b71b-2a74a16f51bc", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-11-06T21:48:08.958189", "product_id": "B0007T2OCG", "quantity": 60, "total": 3488.4}
{"id": "00297ff1-ed51-4a4e-9071-f1cee6a925b5", "client_id": "bc7c7455-6913-4973-8a96-6f36e5bda4b3", "timestamp": "2017-11-19T21:48:08.958189", "product_id": "B006W6XFX0", "quantity": 31, "total": 2601.83}
```

Introducing “Alimazon”

Client Purchase Orders Sample

“Alimazon” Client Purchase Orders Row Sample:


```
{
    "id": "c2f01f80-f1c3-419f-8471-dbd602ceedc4",
    "timestamp": "2018-04-02T14:30:11.754633",
    "client_id": "ad5fa320-2b3a-42a7-b8b8-e53b151aa37f",
    "product_id": "1c6df850-57ed-4312-bc06-e7f1bc2c5568",
    "quantity": 1,
    "total": 100.0
}
```

Assignment

Bucket Location and Files Structure

The dataset you'll need to read is located in the Google bucket:

`gs://de-training-input/alimazon/client-orders/50000`

Within that bucket, you'll find many files of the form:

`part_timestamp_number.jsonl.gz`

```
gs://de-training-input/alimazon/client-orders/
    part_20180709T170251_00000.jsonl.gz
    part_20180709T170252_00001.jsonl.gz
                    ...
```

Assignment

Products in “Alimazon”

Using the Client Purchase Orders Dataset:

1. **Can you count the number of orders grouped by product?**

For this assignment you’ll need to do several things:

- Figure out how to locally create a JAR (i.e. self-contained application) file containing your Spark program (check this guide; use the sample **build.sbt** file in the next slide.) You will need to install **sbt** (the Scala interactive build tool) on your machine to complete this.
- Figure out how to submit a Spark job to your assigned cluster (check this guide for Scala and this one for Python). If you haven’t installed the gcloud sdk, follow this guide to do it.
- Figure out how to read the multi-part dataset from the read-only bucket **gs://de-training-input**. You can refer to the “Word Count” example notebook to see how to do this.

Assignment

Sample `build.sbt` File

```
name := "Spark Project"

version := "1.0"

scalaVersion := "2.11.0"

libraryDependencies += "org.apache.spark" %% "spark-core" % "2.2.0" % "provided"
```

C1

Where Can I Get this Presentation?

goo.gl/g5ijp9

C1

Your feedback is very valuable!

http://bit.ly/Dataeng1