\*\*\*\*\*\*\*\*\*\*\*\*\* Spark \*\*\*\*\*\*\*\*\*\*

--Why

1. Readability

2. Expressiveness

3. Fast

4. Testability

5. Interactive

6. Fault Tolerant

7. Unify Big Data

----Course Overview

1. Basics

2. Core API

3. Cluster Managers

4. Spark Maintanance

--Libraries

1. SQL

2. Streaming

3. MLlib/GraphX

4. Troubleshooting/Optimization

5. Future of Spark

--Section Overview

1. Hadoop

2. History of Spark

3. Installation

4. Big Data's Hello World.

5. Course Prep

Defination => Unified platform of BIG Data

site spark.apache.org

\*\*\*\*Spark Core\*\*\*\*

=> Core API

--Appify

--RDD

--Transforming Data

--Action

---Spark Context

Task Creator

Scheduler

data locality

Fault tolerance

Resilient Distributed Dataset

DAG -> Directed Acyclic Graph

Transformations

- map

- filter

...

Actions

- collect

- count

- Reduce

-...

Lambda Functions

Spark Submit

Spark Cluster Manager => Distributed Kernal => Single Managing other machines

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Spark \*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. RDD

2. Unified API for datasets.

3. Higher level ML API's

4. Unified Batch and Streaming Queries

To understand Spark

1. Basic Understanding of Distributed Computing

2. Representing Structured data as data frames, spark and SQL Context

Spark Libraries ->

* Spark SQL
* Spark Streaming
* MLlib
* GraphX

Exploring and Analyzing data with DataFrames

* Tranformations, Built in Aggregations, Sampling, Grouping and Sorting Data
* Accumulators and Broadcast Variables.

Apache Spark

REPL Environment

Hadoop

1. HDFS

2. MAP REDUCE -> A Framework to define the data processing Task

3. YARN -> A Framework to Run the Data Processing task.

RDD -> Defination

All operations on Spark is performed as In Memory Objects

RDD is a collection of Rows and Entities.

RDD Characteristics

* Partitioned -> Split across data nodes in a cluster
* 2. Immutable -> RDD's once created Cannot be changed
* 3. Resilient -> Can be reconstructed even if a Node Crashes

IN RDD Processing occurrs in Parallel,

Data is stored "in Memory" for each node in the Cluster.

Operations =>

* Transformation => Transform into another RDD
* Action => Any action that request a result.

Individual transformation is done only when you request and action.

Lazy Evaluation =>

Project Tungsten -> Used in Spark 2.0

RDD can be created in 2 Ways

* Read a file and contents can be stored as an RDD.
* Apply the transformation on already existing RDD and generate a new RDD from it.

Every RDD keep track of where it came from.

Lineage of RDD -> Allows RDD to be reconstructed when the Nodes Crash.

Data Frames and Data Sets Difference.

1. Python Program

2. Jetbrains

* Keywords

jupitar

REPL

Broadcast

Cubernatives

Container

AWS EC-2

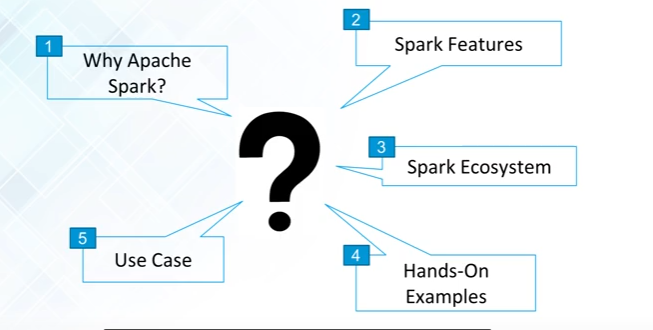
Docker

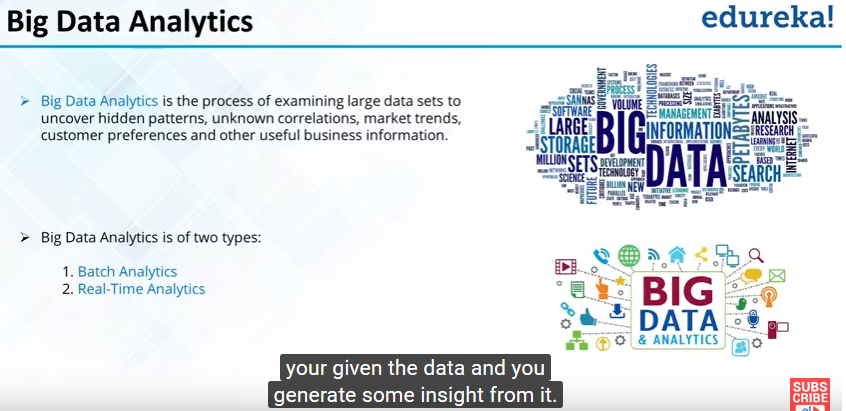
EKS

amazon EMR

LSH

**\*\*\*\*\*\*\*\*\* Spark In Detail \*\*\*\*\*\*\*\*\*\*\*\*\***

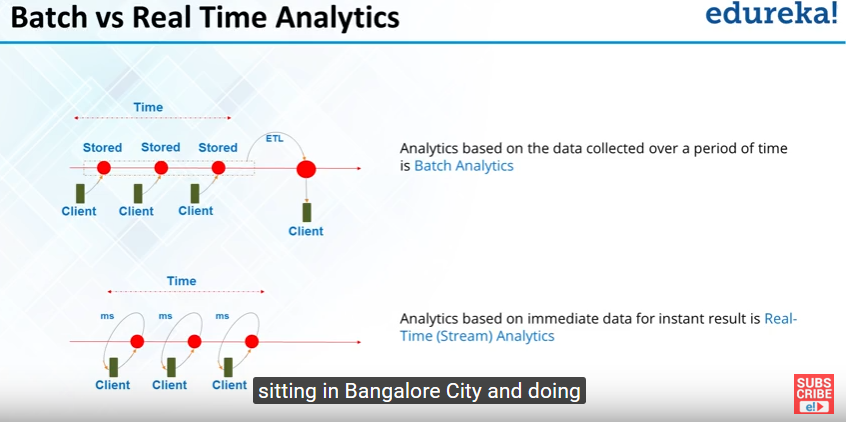




1. Batch Analytics -> Historical Data -> Hadoop
2. Real time analytics -> Immediate data ->Spark

Batch Analytics -> Analytics based on the data collected over time.

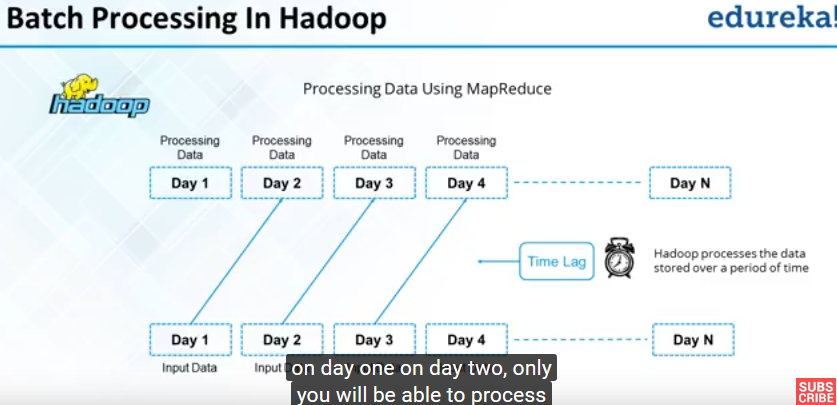
Real time analytics -> Analytics based on immediate data for instant results is Real-Time(Stream) Analytics.



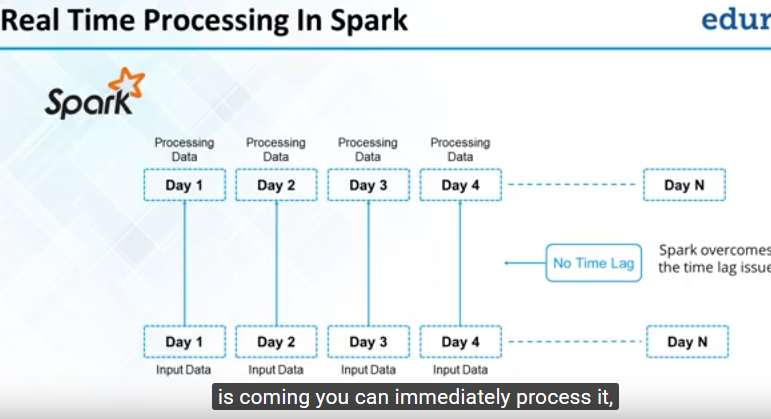
.



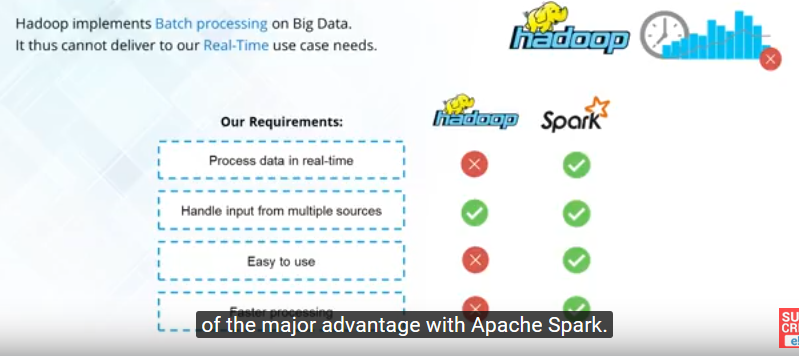
* **Time LAG in Batch Processing:-**



**No Time LAG in apache Spark -: Immediate processing Batch+Real time processing**



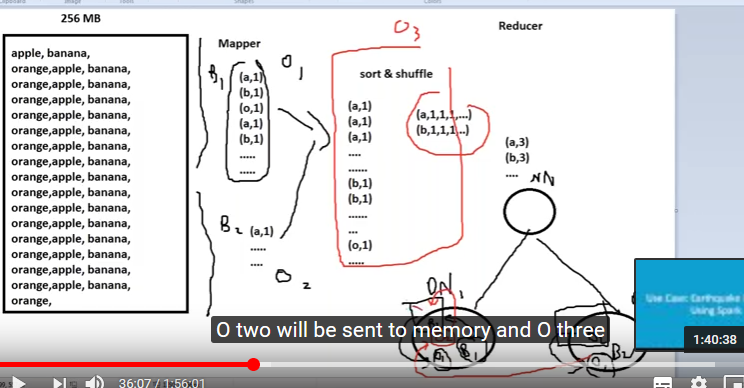
* **HADOOP VS APACHE SPARK**



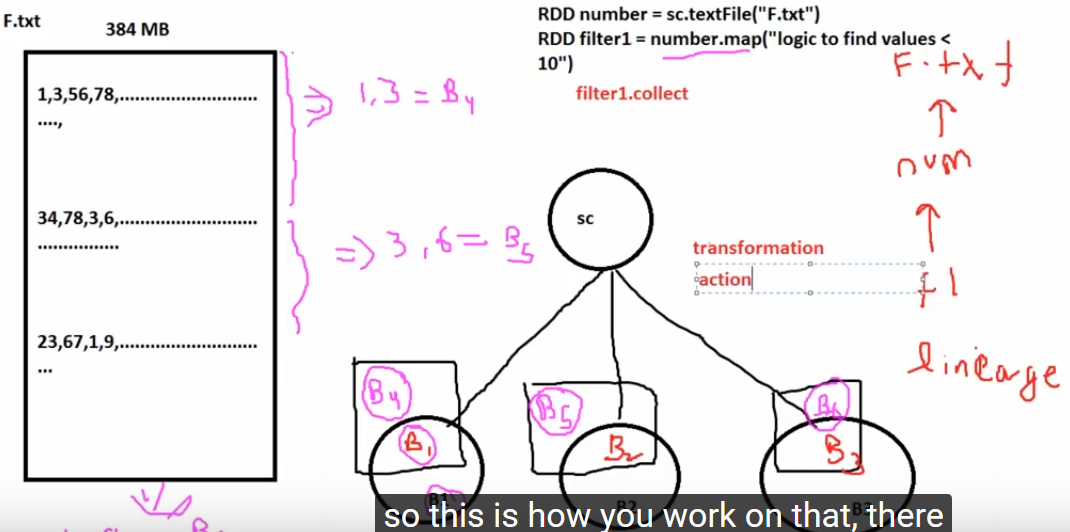
**MapReduce ->**

1. Map-Sort-Reducer

1. Input Output at every Stage

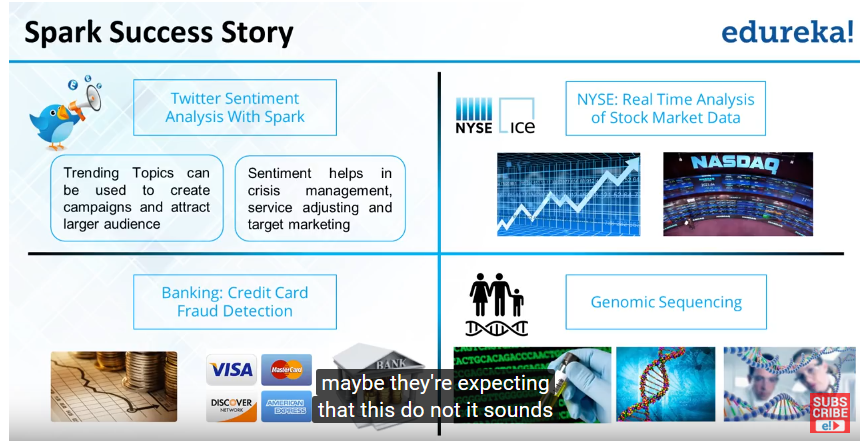


* DAG-LINEAGE => Dependency of Steps
* Transformation => F1-Number-> F2
* Action => filter1.collect

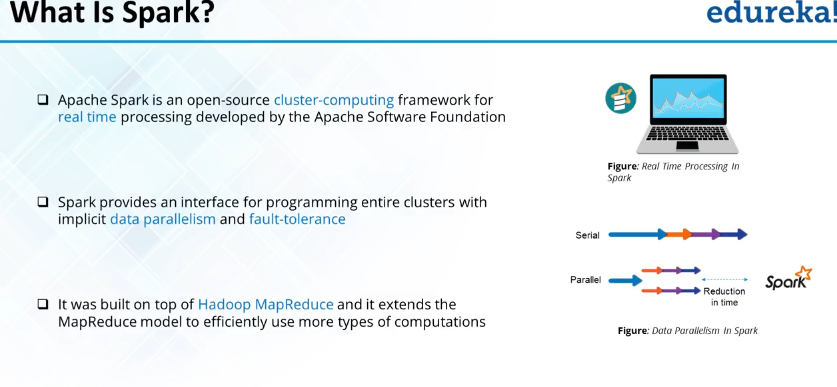


* **Spark Features in Details**

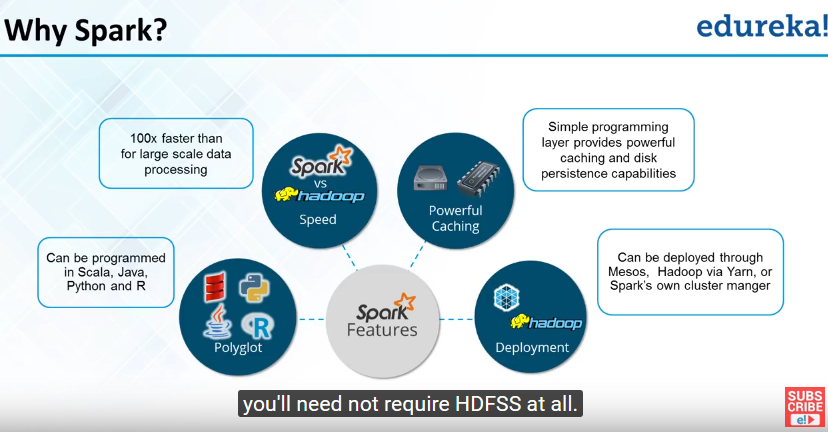
1.



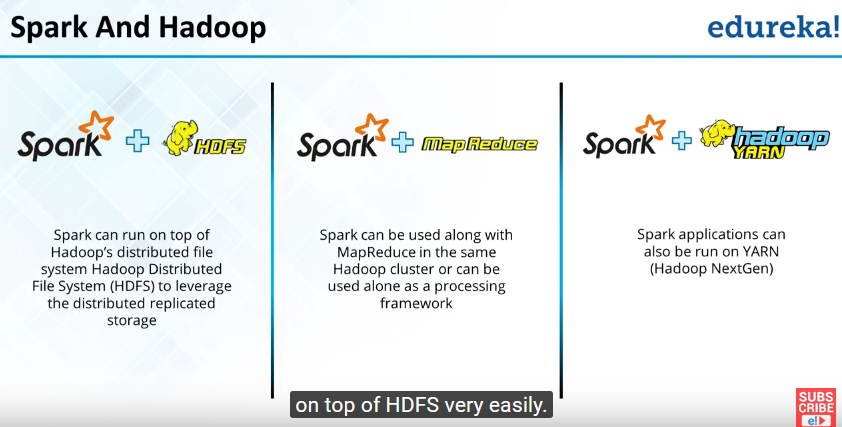
2.

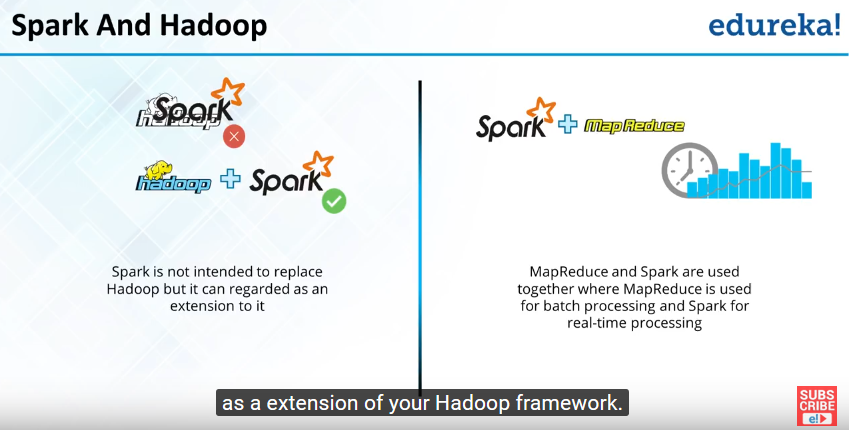


3.

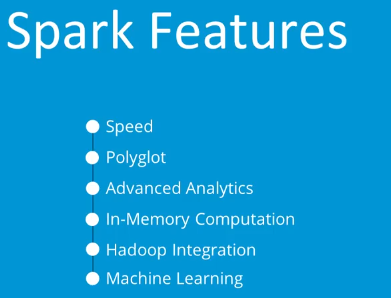


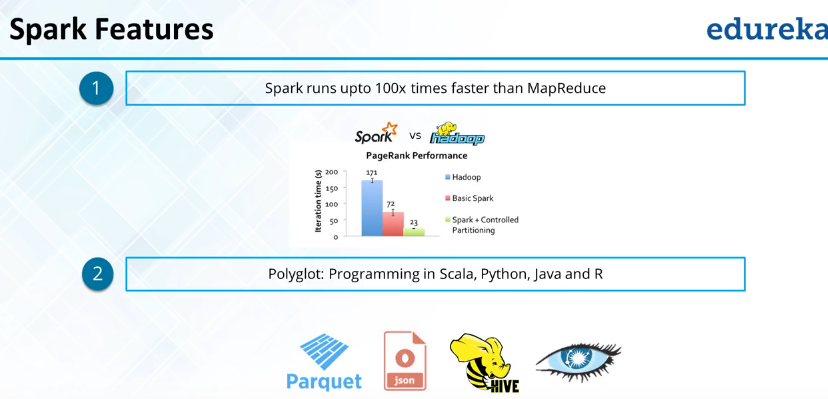
4.

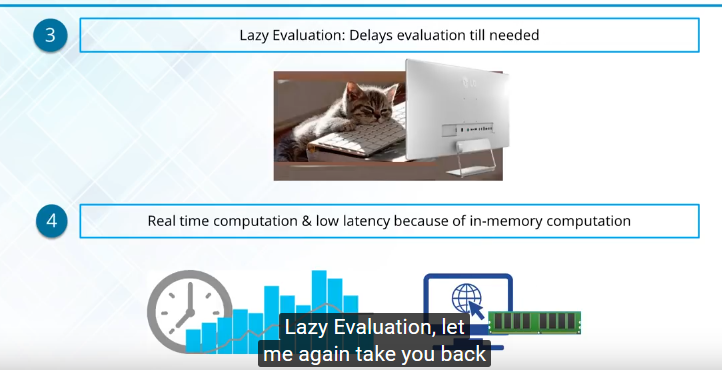




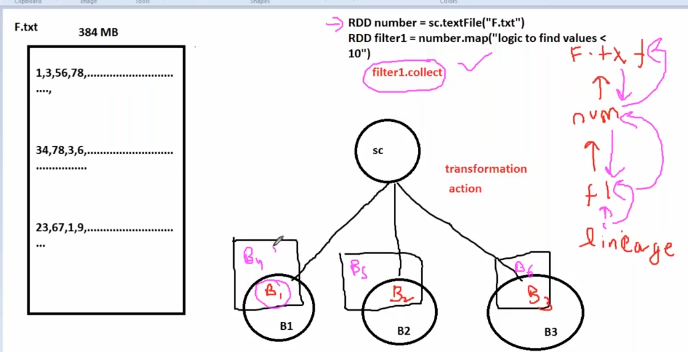
5.

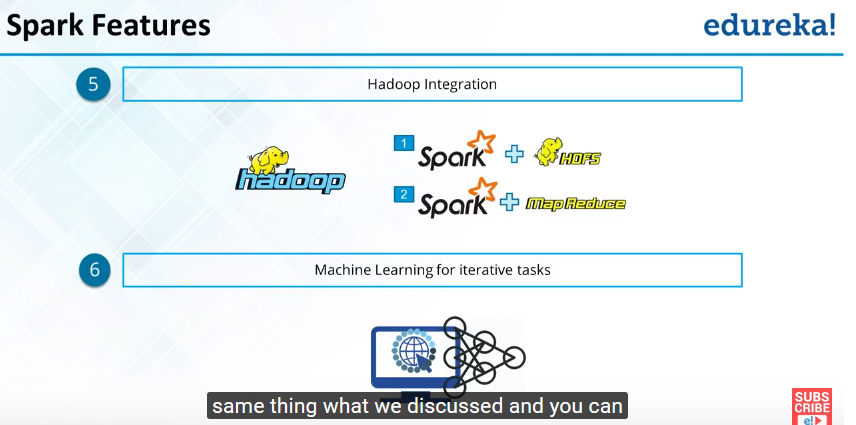


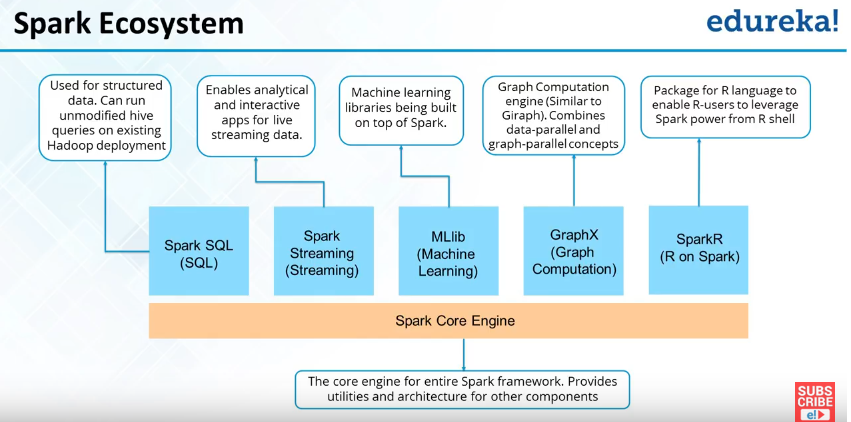


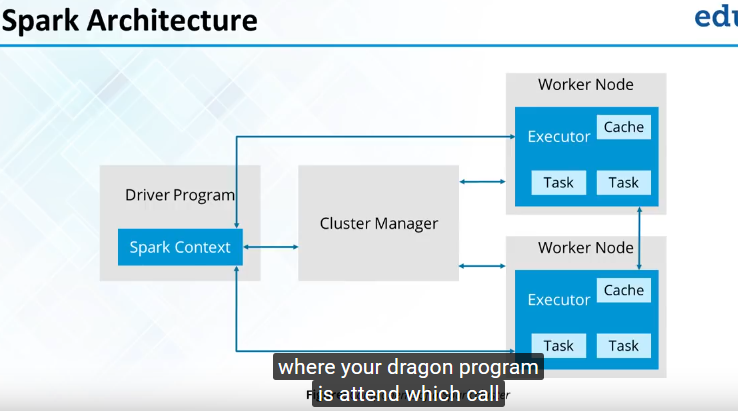


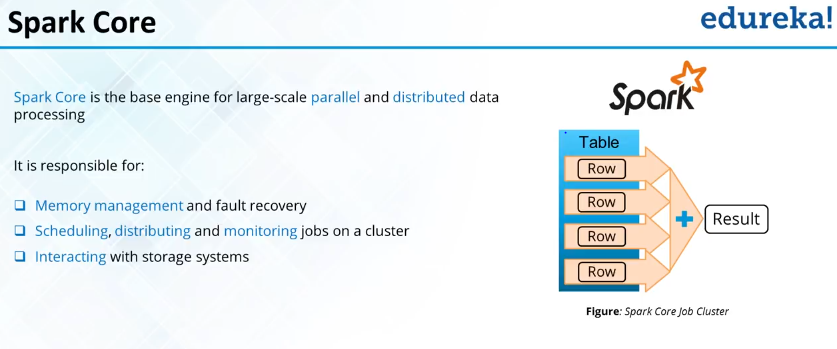
Lazy Evaluation:-

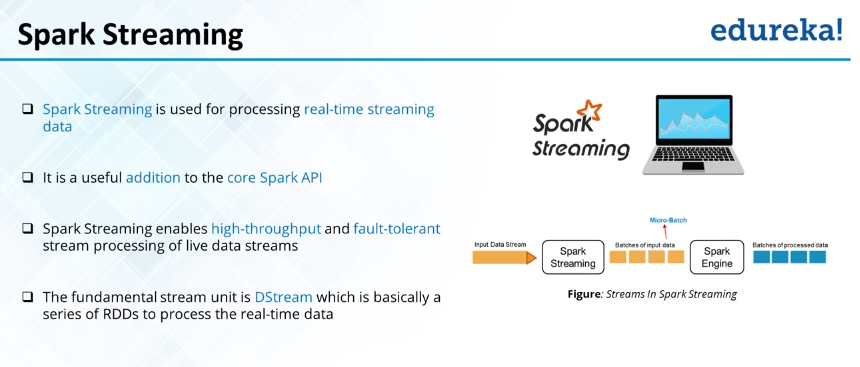


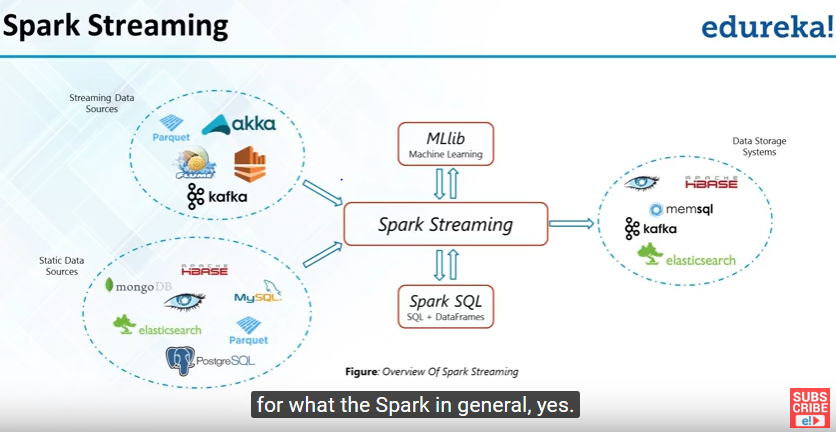


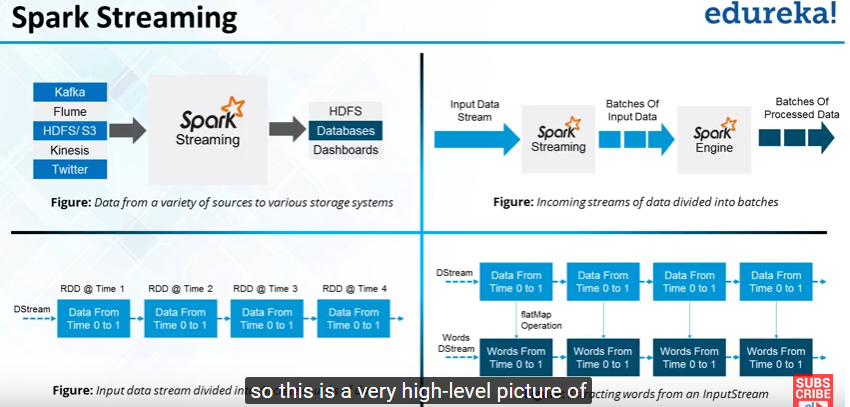


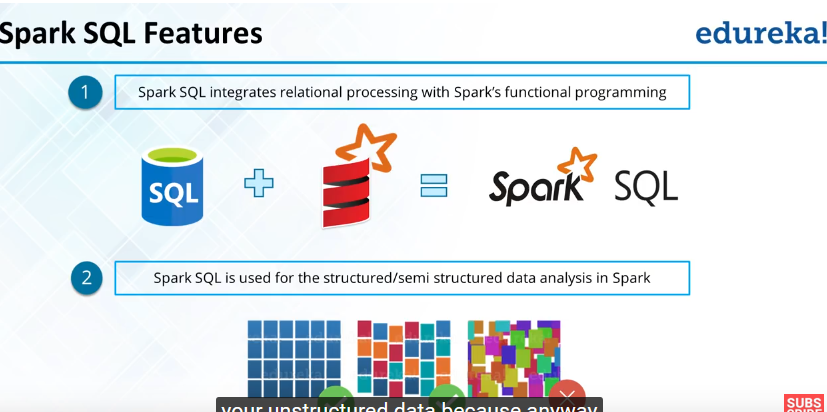


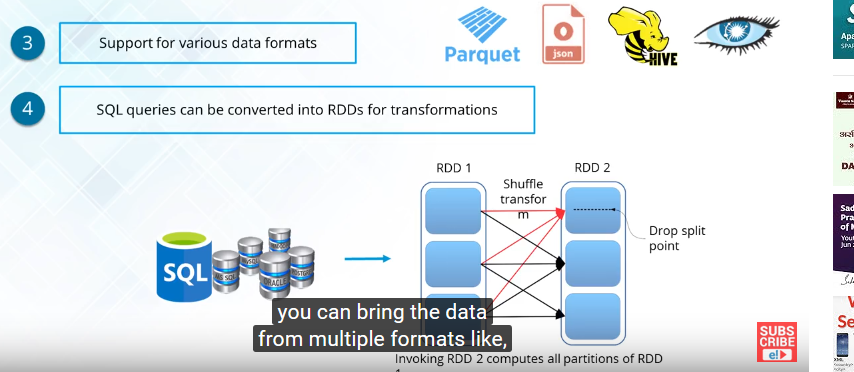




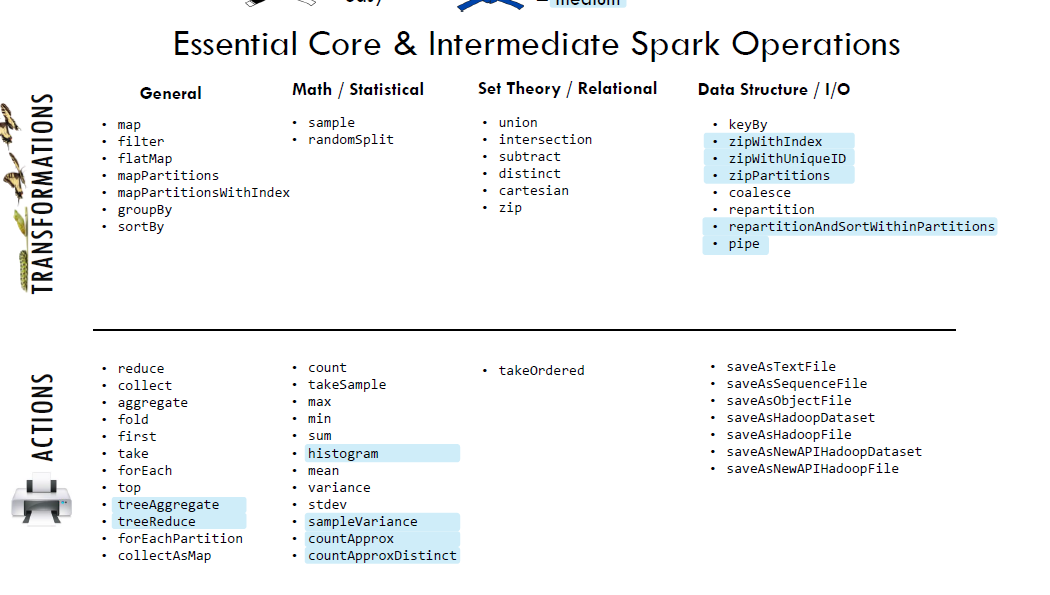


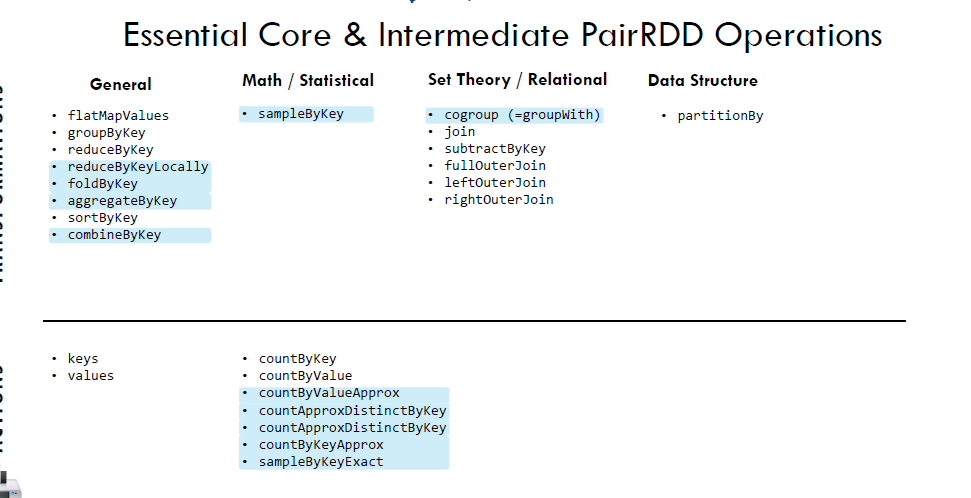






**Spark Transformation and Actions**





****

**Transformation and Action Books**

**API’s**

**Rest(REpresentational State Transfer) APIs =>**

REST API is a way of accessing the web services in a simple and flexible way without having any processing. Preferred over SOAP **(**Simple Object Access Protocol**).** All communication done via REST API used only HTTP request.

**Methods :-**

|  |  |  |
| --- | --- | --- |
| **Action** | **HTTP Verb/Methods** | **Description** |
| Create | POST | Create a new, unique thing |
| Read | GET | Read the information about a thing or collection of things |
| Update | PUT | Update the information about an existing thing |
| Delete | DELETE | Delete a thing |

Idempotence: An idempotent HTTP method is a HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. Again, this only applies to the result, not the resource itself. Example,

filter\_none

edit

play\_arrow

brightness\_4

1. a = 4 // It is Idempotence, as final value(a = 4)

// would not change after executing it multiple

// times.

2. a++ // It is not Idempotence because the final value

// will depend upon the number of times the

// statement is executed.

**Note-:** RESTful web services are very popular because they are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.

**Status codes**

The request we just made had a *status code* of 200. Status codes are returned with every request that is made to a web server. Status codes indicate information about what happened with a request. Here are some codes that are relevant to *GET* requests:

* 200 — everything went okay, and the result has been returned (if any)
* 301 — the server is redirecting you to a different endpoint. This can happen when a company switches domain names, or an endpoint name is changed.
* 401 — the server thinks you’re not authenticated. This happens when you don’t send the right credentials to access an API (we’ll talk about authentication in a later post).
* 400 — the server thinks you made a bad request. This can happen when you don’t send along the right data, among other things.
* 403 — the resource you’re trying to access is forbidden — you don’t have the right permissions to see it.
* 404 — the resource you tried to access wasn’t found on the server.

Query Parameters: = {“late”: 40.30, “long”:-10}

Response = response.get(<http://api.open-notify.org/iss-pass.json>”, params = parameters)

Work with JSON Data

**HTTP Libraries**

Now, to make HTTP requests in python, we can use several HTTP libraries like:

* [httplib](https://docs.python.org/2/library/httplib.html)
* [urllib](https://docs.python.org/2/library/urllib.html)
* [requests](http://docs.python-requests.org/en/master/)