Spark with Python

Singh Rounak.

History of Growing Data

As data kept on increasing, there were no softwares to handle and process huge data loads. Companies only used to invest and buy more hardware (scale vertically) instead of creating softwares to Tackle the issue.

→Data Warehouses were Expensive and only vertically scalable.

→Problem with data warehouses - Ingest, Store, Process and Access data

→Horizontal scaling allows to start small and add more resource capability as requirements grow.

DATA LAKE - horizontally scalable solution to modern data problems, uses Cloud Computing.

Solution -

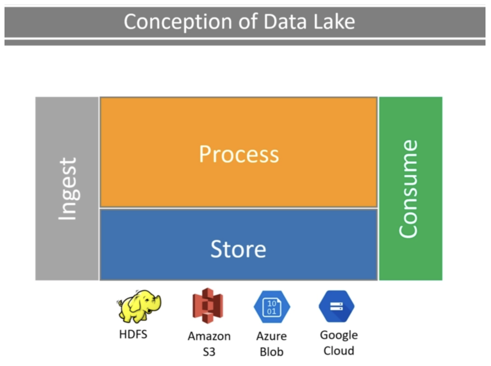
→ Google introduced the White paper - Google File System(GFS) which was the first thing to introduce Horizontal scaling to work towards developing new software.

→ Cloud Infra started becoming more economical and matured as a platform.

→ Concept of Data lake -

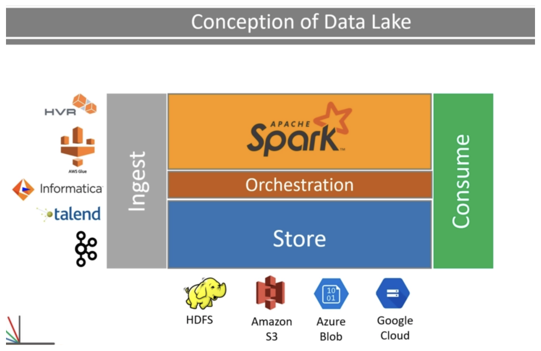
Ingest, Store, Process and Consume

1. Core of Data Lake is 'Storage'. Top vendors providing best storage service are (bottom)



→ The notion of the data lake recommends that one should bring data in raw format. Which means data must be ingested into data lake. And preserve an unmodified immutable copy of the data

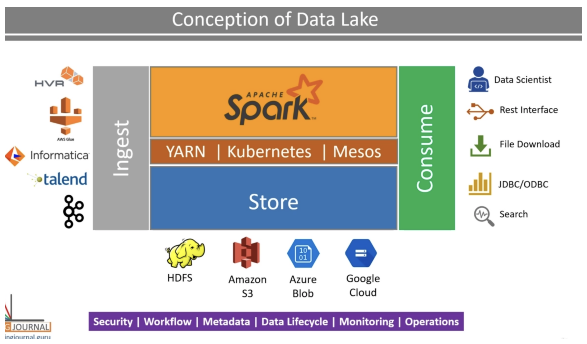
Ingest - Identifying, Implementing and managing the right tools tools to bring data from the source to the data lake. Vendors competing for Ingestion are (Left side of figure).



The next Layer is the processing layer (where all computation happens) - Quality checks, Correlating, Aggregating, Analyzing and extracting business insights.

Broken into 2 parts - Processing and Orchestration

1. **Processing** - Core Development and design of Distributed computing applications - Spark falls here(Above orchestration)
2. **Orchestration -** Responsible for formation of cluster, scaling up/down and managing similar things (The middle Orange line below Spark).



Consuming Data from Data Like (Right Hand Side).

<u>**Spark - Heart of Data Lake**</u>

Apache Spark - Distributed Data Processing Framework.

1.Core APIs - Scala, Python, Java and R - Used to write Analysis programs. Offer high level of flexibillity.

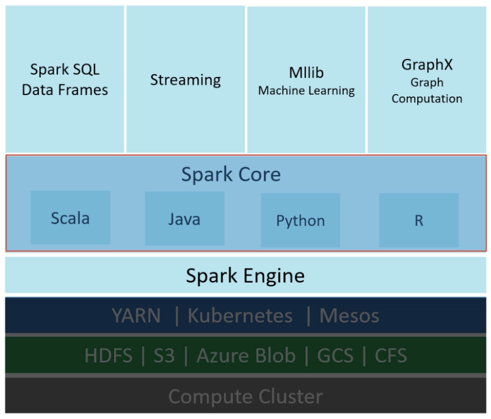
2.Set of DSLs, Libraries and APIs - **Area of Particular Interest of Data Scientists, Analysts and Spark Developers.**

These together are run on a cluster of computers.

A cluster manager is needed - Container Orchestrator

Inbuilt storage system - HDFS, S3, Azure BLOB, GCS, CFS

SPARK ECOSYSTEM -



The topmost layer is a set of libraries to solve Data Crunching problems.

Benefits - **Abstraction** (You don't have to code anything, Its like using a Database),

Unified Platform(Combines the capability of SQL queries, batch and stream processing, ML and DL all in one),

Ease of Use(Ready to use libraries and tools, No need of hardcore logic and programming).

These 3 things make Spark the best choice for Distributed Data processing.

HDFS - Provides fault tolerant Data Storage and access on a horizontally scalable distributed cluster.

Acces to Get Spark - Download SPARK

1. Download Java openjdk@11 on your machine
2. Set JAVA Home environment variable using command- *export JAVA_HOME* = `*directory...*` in Terminal. It should currently point to your JAVA installation.
3. Download SPARK from website
4. Set SPARK Home environment variable using command - export SPARK_HOME = `*directory*` in Terminal. It should currently point to your SPARK installation.
5. Path Environment variable - echo $PATH (copy what path gets displayed)

1. JDK 8 or 11
2. JAVA_HOME
3. SPARK_HOME
4. PATH
5. PYSPARK_PYTHON

Place commands in your start-up script, to access start-up script, type-

.nano -zshrc in terminal. The scripts are as follows:

```
export PATH="/opt/homebrew/opt/openjdk@11/bin:$PATH"
export JAVA_HOME=/opt/homebrew/Cellar/openjdk@11/11.0.14.1
export SPARK_HOME=/Users/rounaksingh/spark3/spark-3.2.1-bin-hadoop3.2
export PATH=$PATH:$SPARK_HOME/bin
export PYSPARK_PYTHON=python3
```

DATA BRICKS- allows you to do the following:

1. Store your data in cloud based storage called Databricks File System(DBFS)
2. Run your Spark Code on Databricks cluster
3. Process your data which is stored in DBFS using Spark application.

What is Spark? - Distributed Data Processing Engine that uses SQL and Python and executes programs in a cluster.

What do we do with Spark? - Create programs and execute them on the Spark cluster.

Q. **How do we create Spark Programs?**

Q. **How de we execute Spark programs?**

Execution Methods:

1. Stream Processing 2. Batch Processing

Ways of executing Spark Programs-

1. **Interactive clients** - Spark Shell, Notebooks: (Run your spark code line-by-line) best for learning and development.
2. **Submit Job** - Spark-submit, Data Bricks Notebook, Rest APIs(Vendor Specific) : For production use cases

To run Spark from a Jupyter Notebook, you need the following:

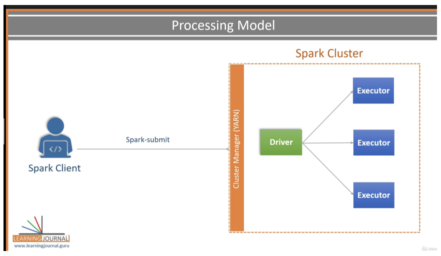
Spark Binaries, SPARK_HOME, PYSPARK_PYTHON, findspark package, pyspark package

SPARK Distributed Processing Model:

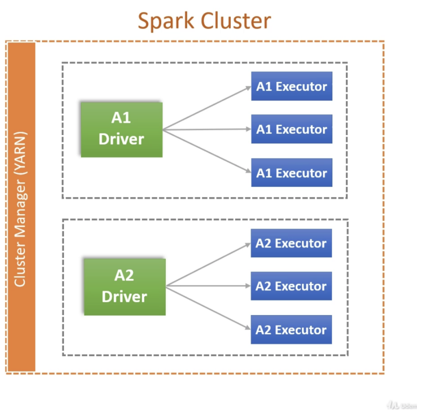
Uses a Master-Slave architecture to every application. The Master assigns the job, the Workers act as. runtime containers with dedicated CPUs and Memory.

Master(*Driver*) -Slave(*Executor*) configuration for your application.

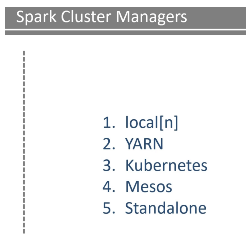
The cluster itself has a Master-Slave configuration too which is managed by the cluster manager.



The spark engine ask for a container to the cluster manager to start executing your driver process and more containers to execute the executor processes.



independent applications have their own drivers and executors. Every Spark applications applies a M-S configuration and runs independently on the cluster. This is how Spark is a distributed Computing Platform.



Spark Cluster Managers

1. local[n]
2. YARN
3. Kubernetes
4. Mesos
5. Standalone

→ *How does spark on your local machine on IDE/Notebook without a cluster or cluster manager?*

Local Machine - You can execute without even having a cluster. The Spark on your machine runs as a multi-threaded application. Each thread acts as Master and Executor respectively.

→ *How does Spark work with interactive Clients?*

Execution Modes:

1. **Client mode** - Spark driver runs locally on client machine. However, if you are quitting your client or log off. The application dies.
   a. Not suitable for long-running jobs
   b. Suitable for interactive small jobs.
2. **Cluster Mode** - The application is submitted to the cluster to let it run (driver and executor) and you can log off.
   a. Suitable for long running jobs.

EXECUTION MODEL:

1. SELECT CLUSTER MANAGER
   a. Local
   b. YARN→i)On Premise(Cloudera) ii)On-Cloud(DataBricks)
   c. Mesos
   d. Kuberbetes
   e. Standalone
2. SELECT EXECUTION MODE & TOOLS:
   a. Client Mode - IDE, Notebook,Interactive-Notebook, Spark-Shell
   b. Cluster Mode - Spark Submit.

→Spark-submit command is used to run your spark application on a YARN cluster :**- -master yarn**

→Command used to start spark-shell in client mode: **- -deploy-mode client**

→Set the driver's memory using the following: **- -driver-memory**

→Spark-submit command used to set the executor memory requirement: **- -executor-memory**

→ Assigning 'n' CPU cores to each executor: **- -executor-cores 2**

→Assigning 'n' GB memory to driver **- -driver-memory nG**

→ Assigning 'n' executors: **- -num-executors n**

SPARK Application UI:

→ Track your application performance

→Collects and shows information/metrics about the application.

→Can be used to investigate application’s runtime behavior.

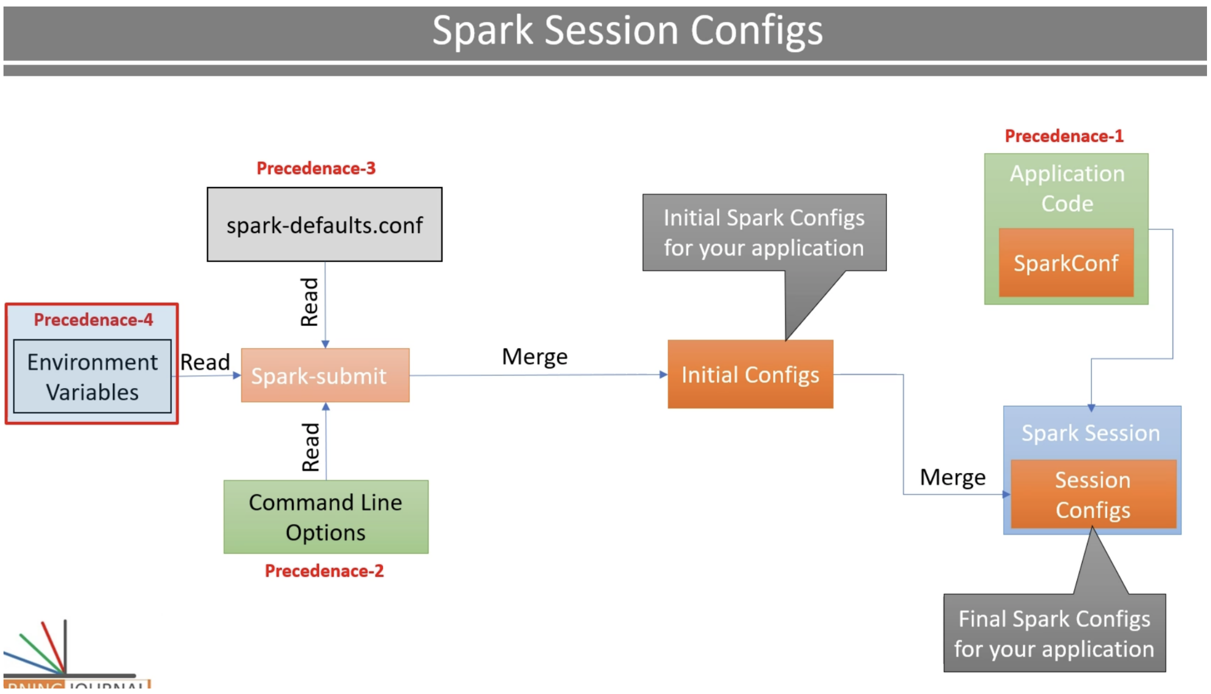
YARN Resource Manager → Your Application ID → Application Master in Tracking UI Column

Zeppelin Notebooks are used for the following:

→ Running Spark Applications in interactive Client Mode.

→ Running SQL code

→ Developing your Spark Application


Spark Session Configs
Precedenace-3
spark-defaults.conf
Read
Precedenace-4
Environment Variables
Read
Spark-submit
Read
Command Line Options
Precedenace-2
Merge
Initial Spark Configs for your application
Initial Configs
Precedenace-1
Application Code
SparkConf
Merge
Spark Session
Session Configs
Final Spark Configs for your application