



# Big Data

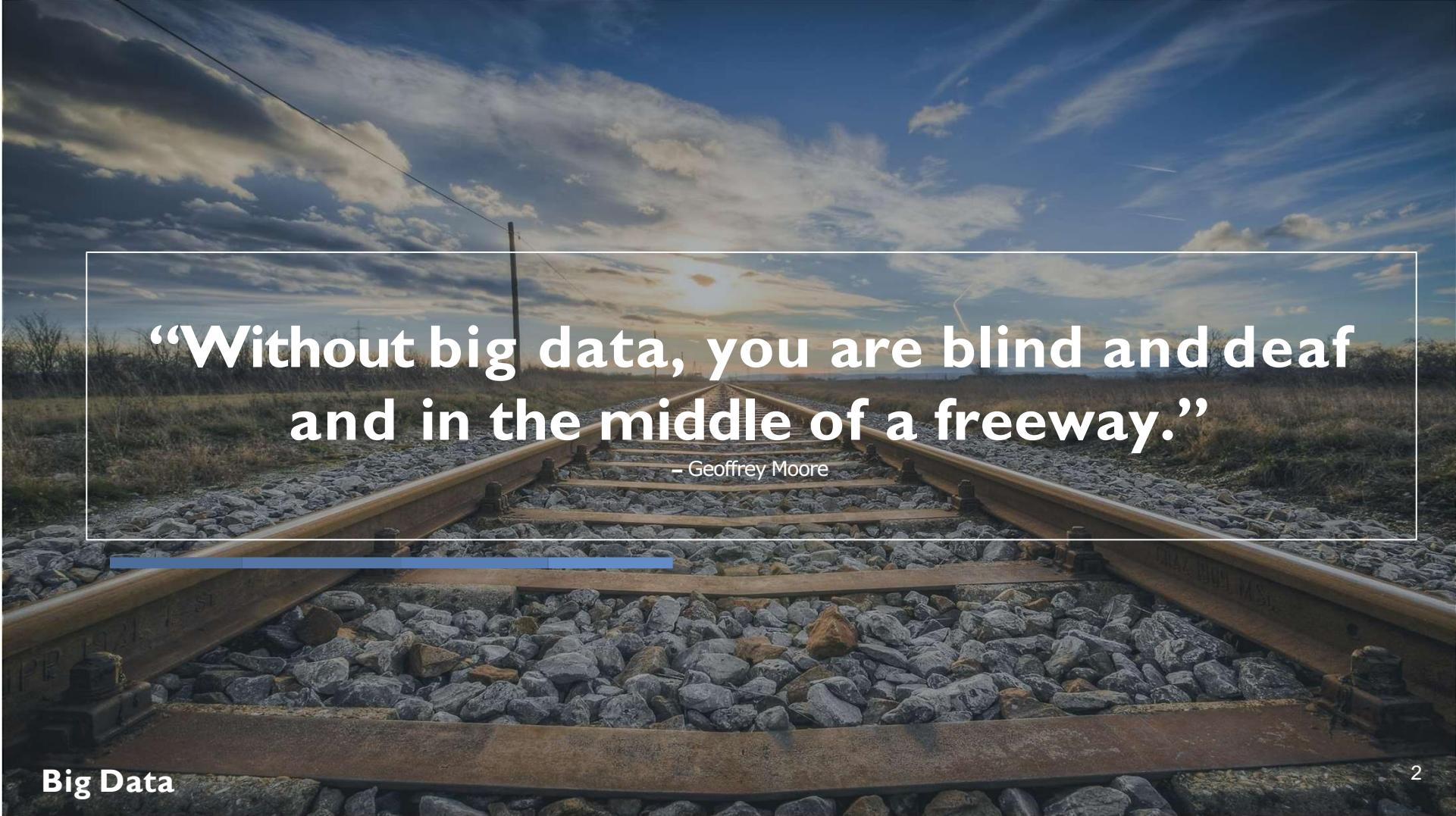
(Understanding about Big data)

Trong-Hop Do

September 8<sup>th</sup>, 2021

**S<sup>3</sup>Lab**

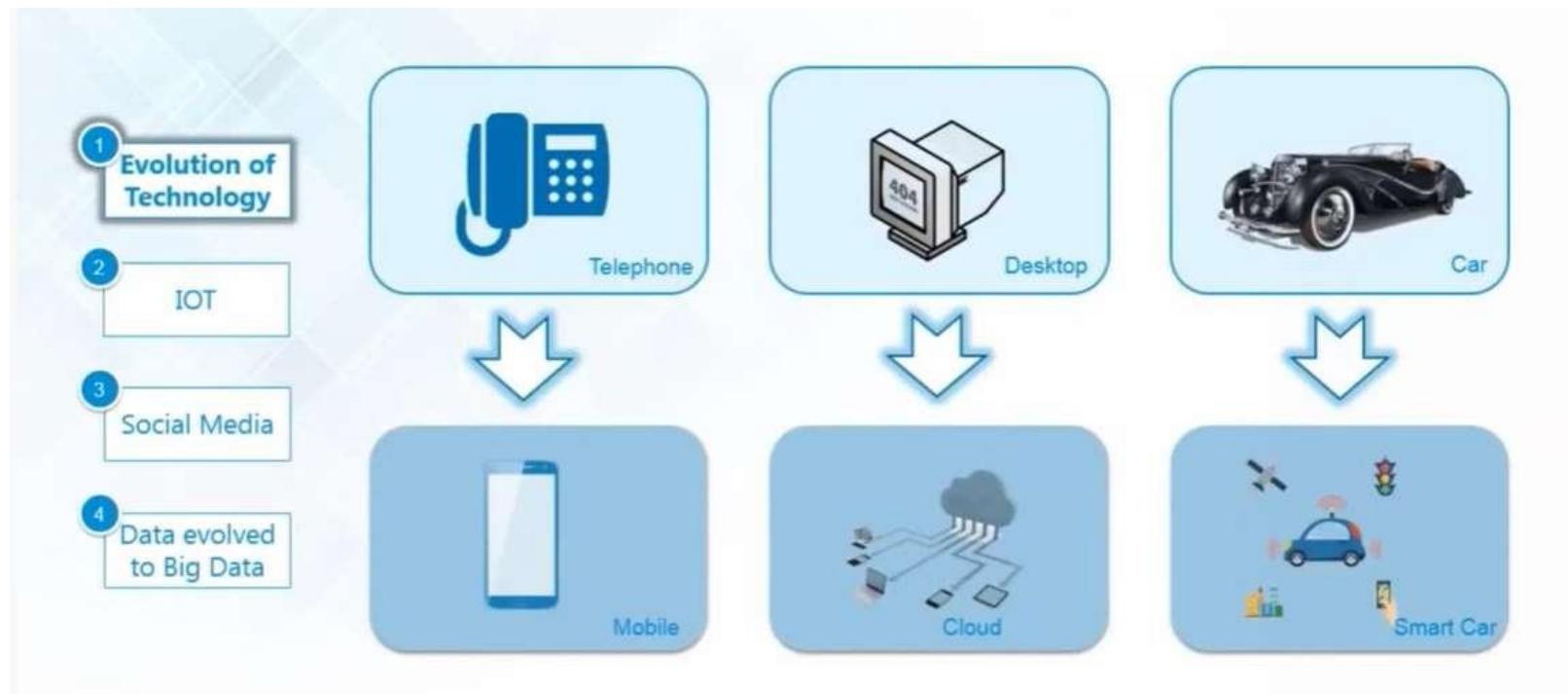
*Smart Software System Laboratory*



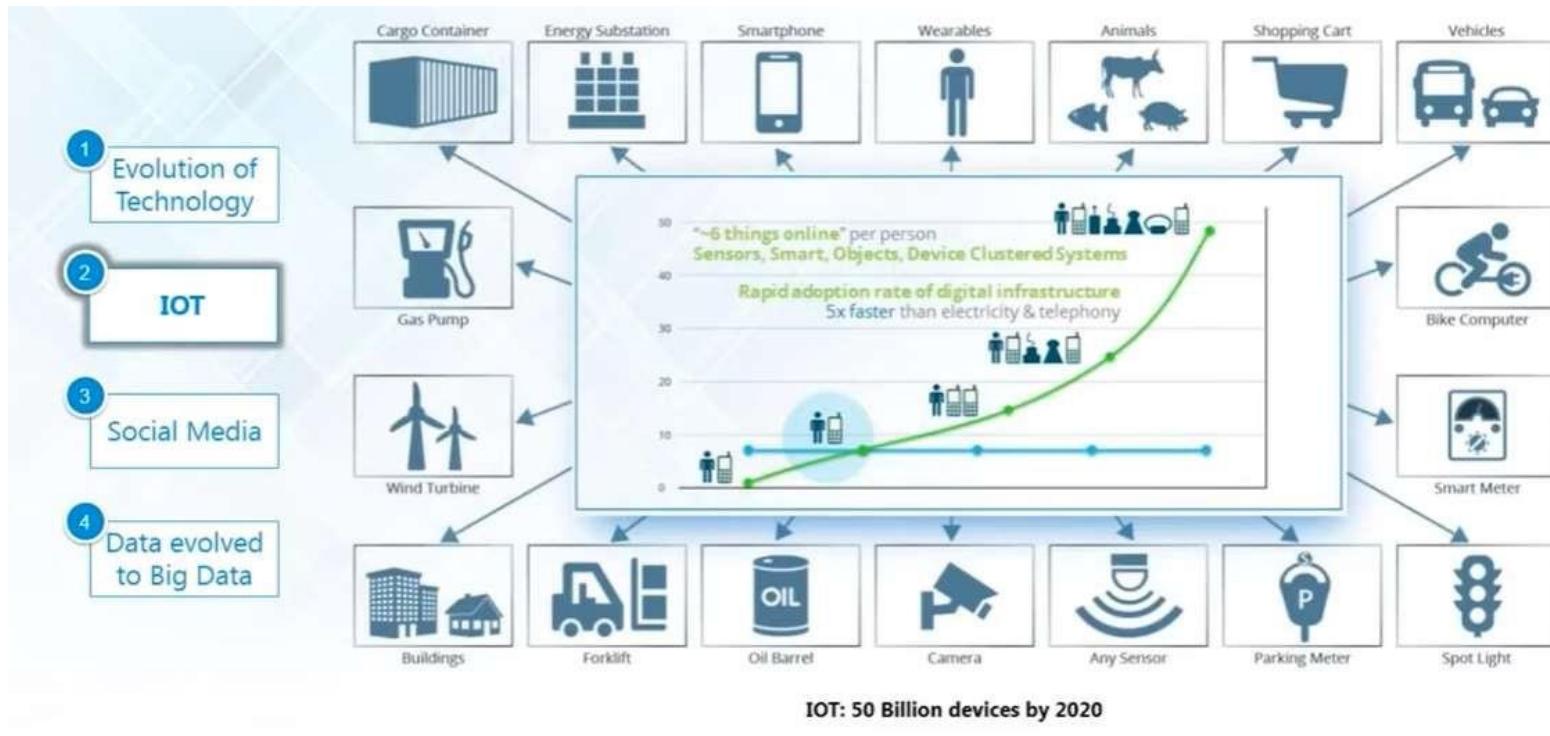
**“Without big data, you are blind and deaf  
and in the middle of a freeway.”**

- Geoffrey Moore

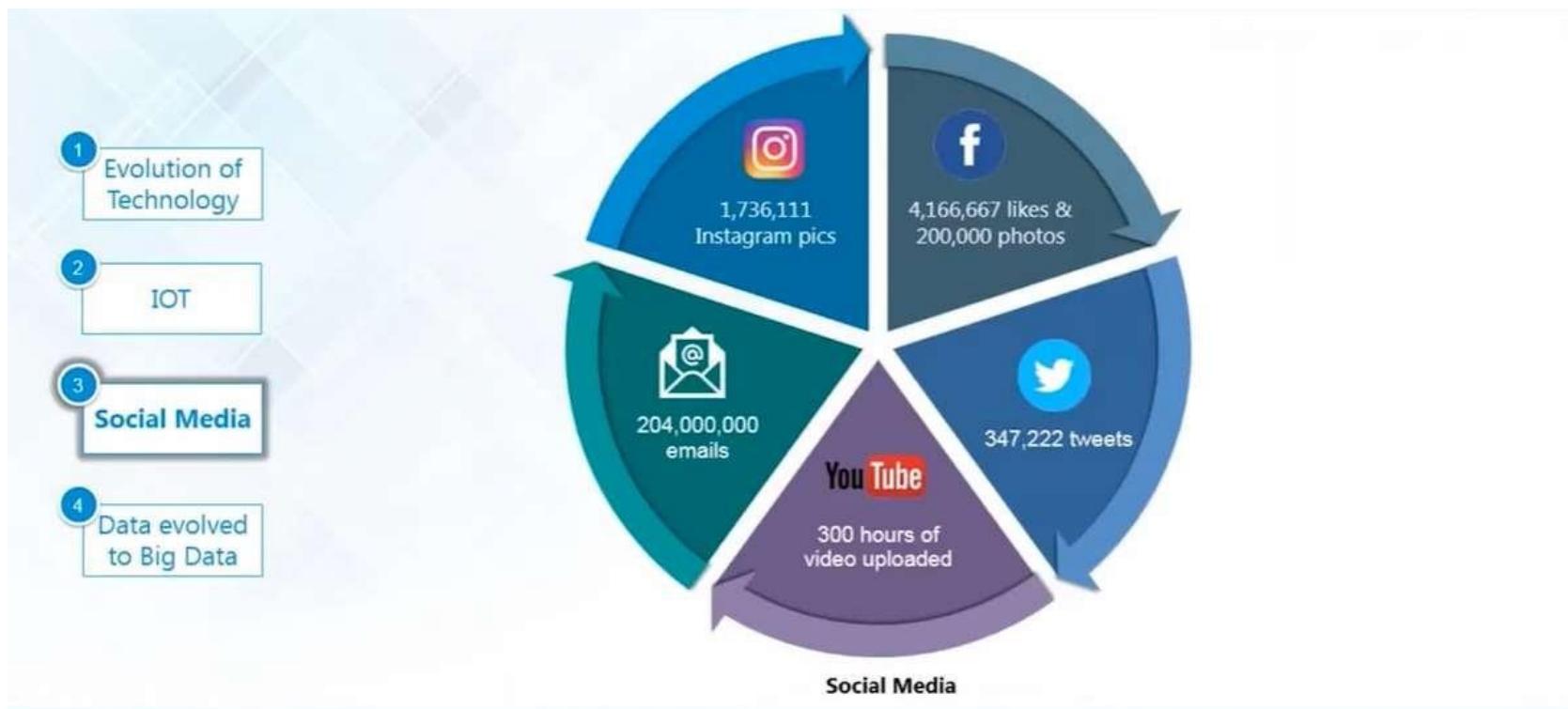
# Evolution of Technology



# IOT



# Social media



# Other factors



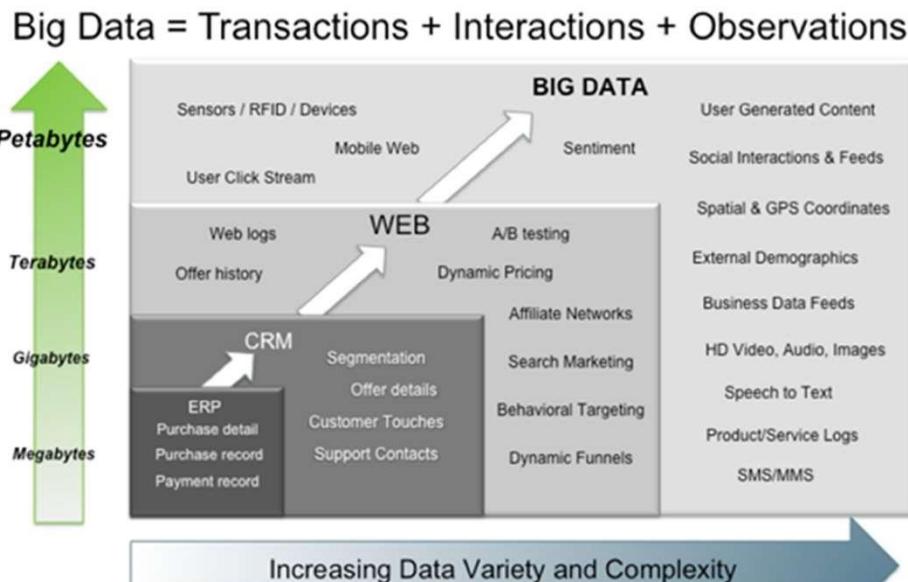
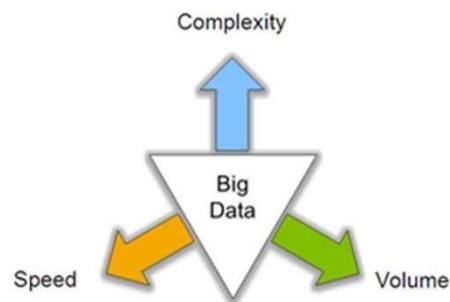
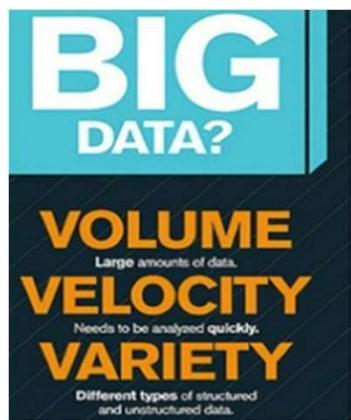


# What is Big Data

- Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.
- Challenges: Capture, Curation, Storage, Search, Sharing, Transfer, Analysis, and Visualization.



# Big Data: 3V's

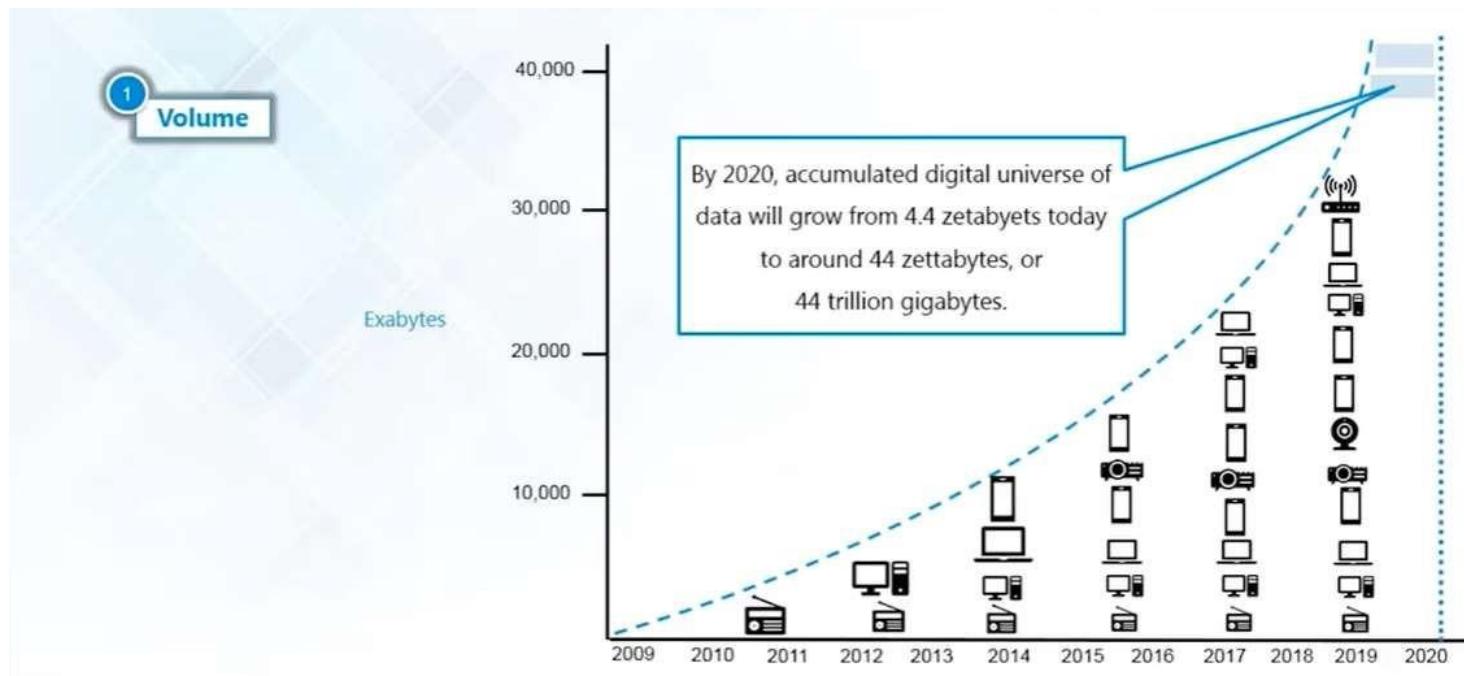


Source: Contents of above graphic created in partnership with Teradata, Inc.



# Big Data: 3V's

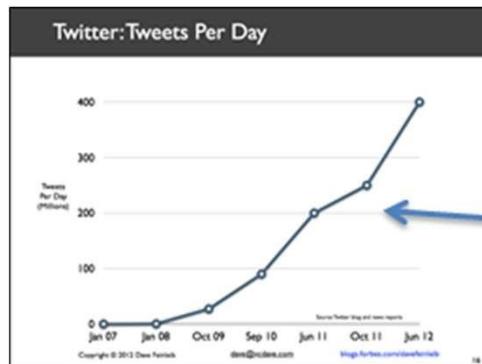
*Volume (scale)*



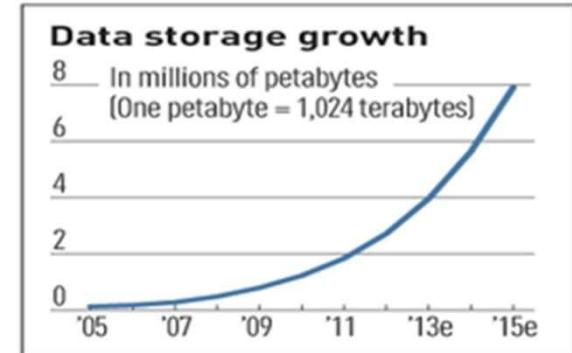


# Big Data: 3V's

*Volume (scale)*



*Exponential increase in collected/generated data*





# Big Data: 3V's

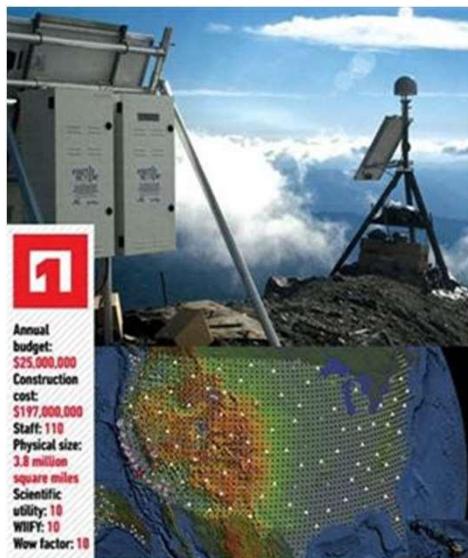
## *Volume (scale)*



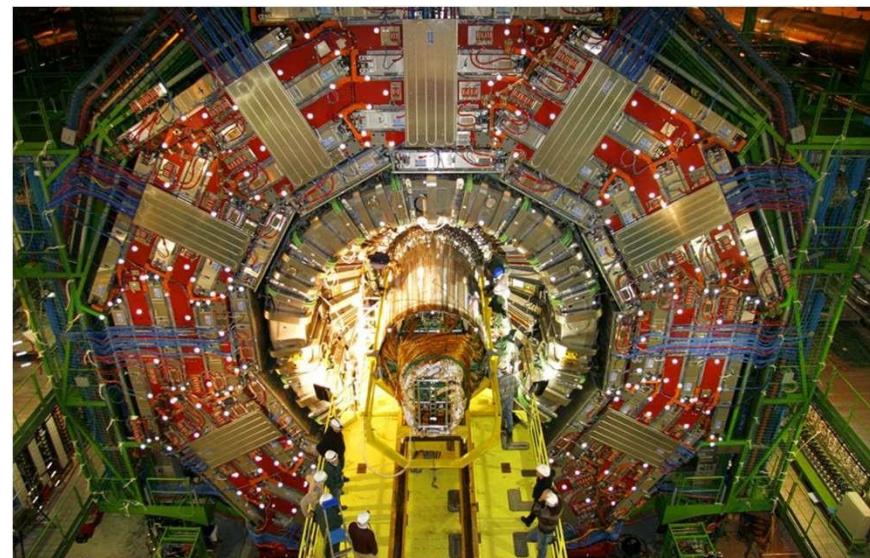


# Big Data: 3V's

*Volume (scale)*



EarthScope - 67 terabytes of data

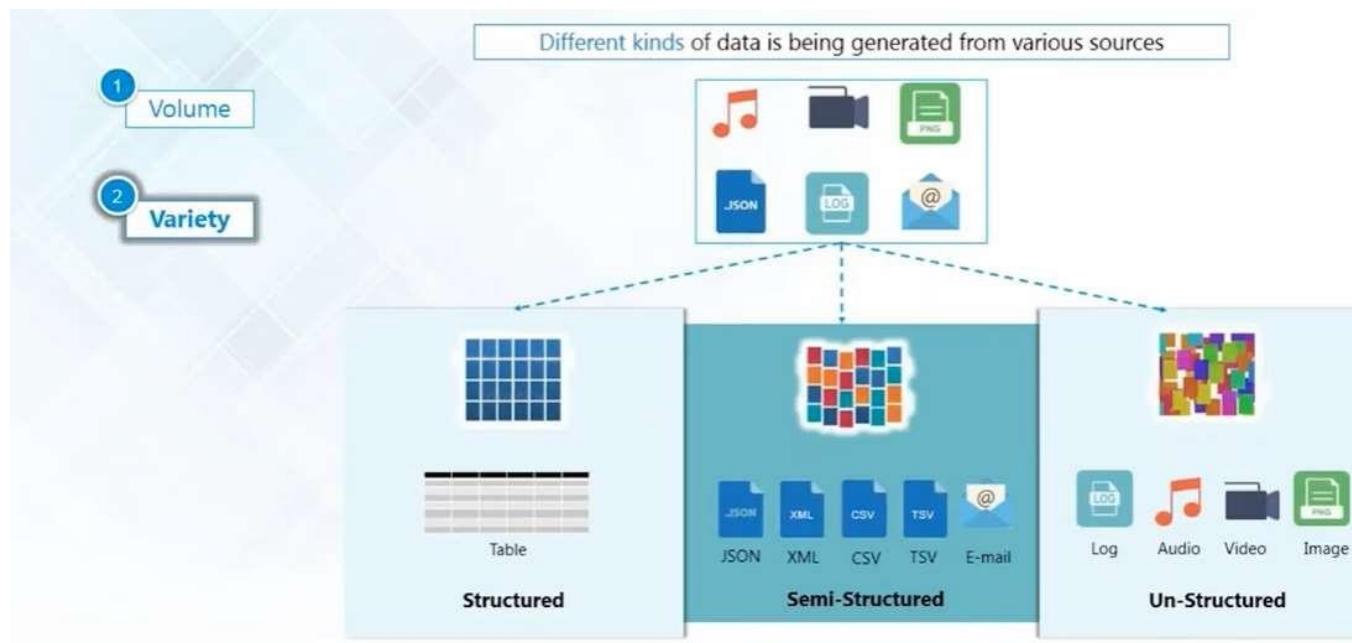


CERN's Large Hadron Collider (LHC) generates 15 PB a year



# Big Data: 3V's

*Variety (Complexity)*





# Big Data: 3V's

## *Variety (Complexity)*

- Big data could be of three types
  - **Structured:** The data that can be stored and processed in a fixed format (fixed schema) is called as Structured Data. Ex. RDBMS
  - **Semi-Structured:** not have a formal structure of a data model, but nevertheless it has some organizational properties like tags and other markers to separate semantic elements that makes it easier to analyze. Ex. XML files or JSON documents.
  - **Unstructured:** Text Files and multimedia contents like images, audios, videos are example of unstructured data. The unstructured data is growing quicker than others, experts say that 80 percent of the data in an organization are unstructured.



# Big Data: 3V's

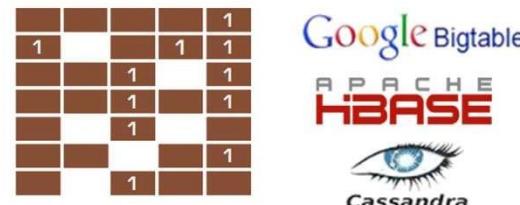
*Variety (Complexity)*

- Semi-Structured, NoSQL

## Key-Value Stores



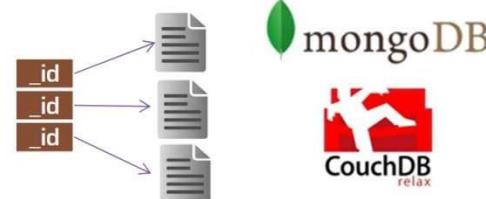
## Column Stores



## Graph Databases



## Document Stores



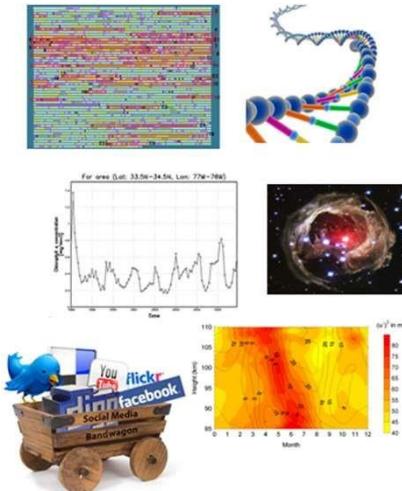


# Big Data: 3V's

*Variety (Complexity)*

- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web, log)
- Semi-structured Data (XML)
- Graph Data: Social network, Semantic web (RDF)...
- Streaming Data: You can only scan the data once
- A single application can be generating / collecting many types of data
- Big Public Data (online, weather, finance, etc.)

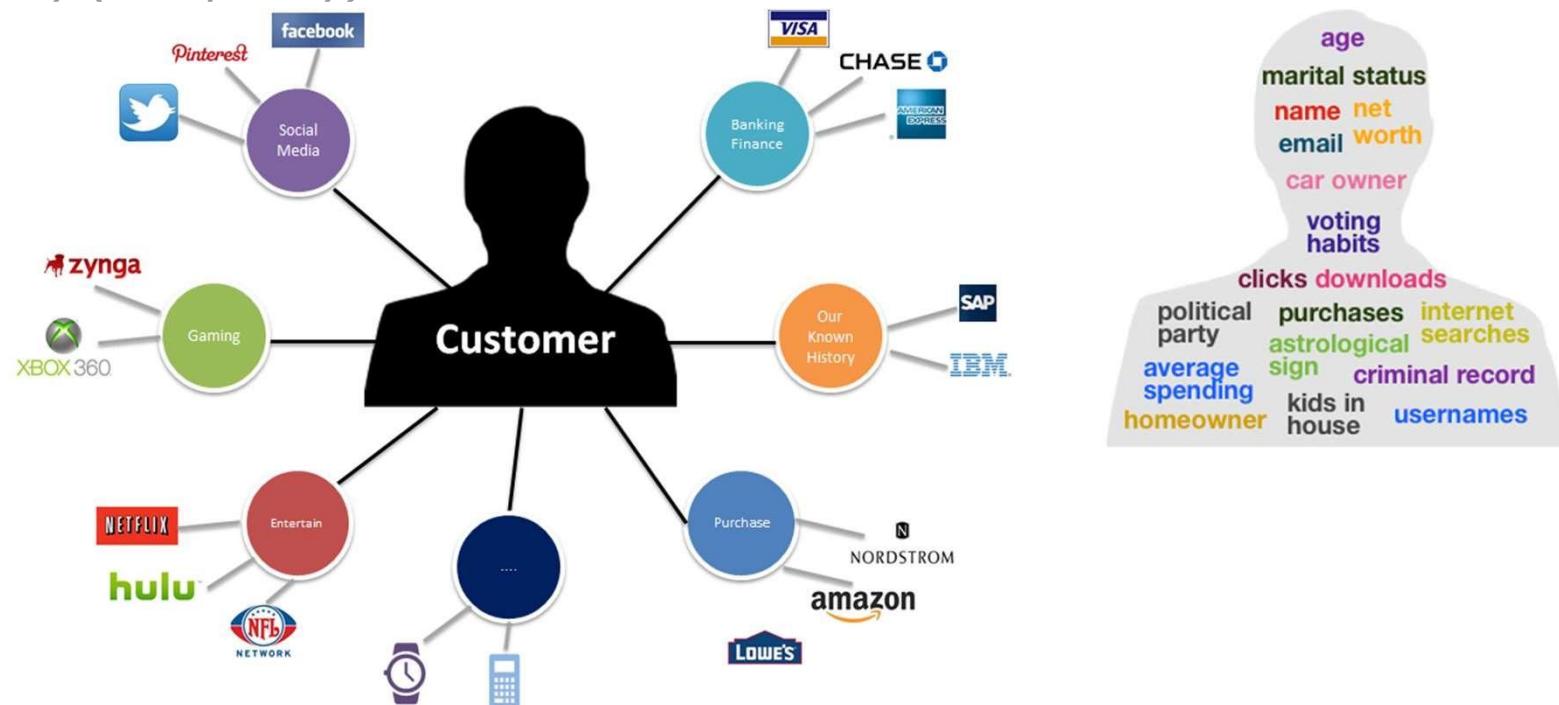
*To extract knowledge → all these types of data need to linked together*





# Big Data: 3V's

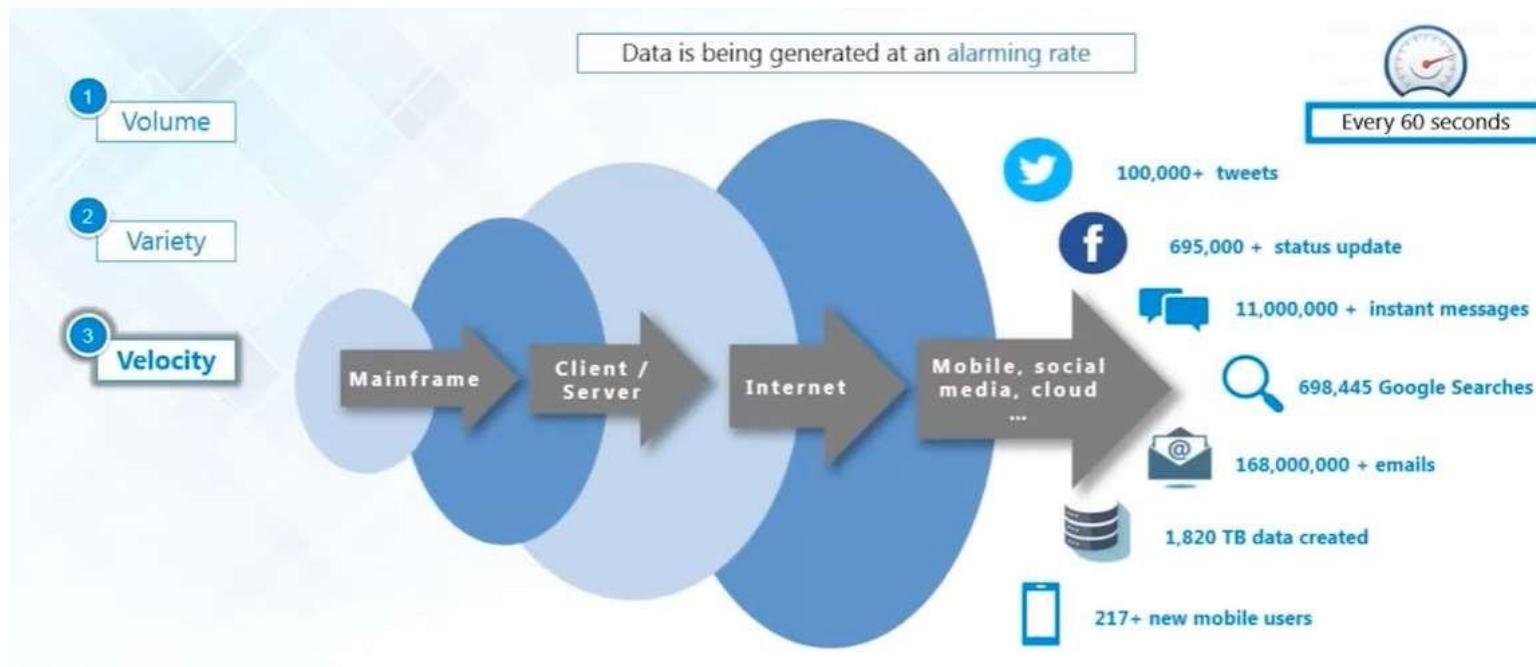
Variety (Complexity)





# Big Data: 3V's

## Velocity (Speed)





# Big Data: 3V's

## *Velocity (Speed)*

- Data is begin generated fast & need to be processed fast
- Online Data Analytics
- Late decisions → missing opportunities
- Examples
  - **E-Promotions:** Base on your current location, your purchase history, what you like → send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body → any abnormal measurements require immediate reaction





# Big Data: 3V's

## *Velocity (Speed)*



**Social media and networks**  
(all of us are generating data)



**Scientific instruments**  
(collecting all sorts of data)



**Mobile devices**  
(tracking all objects all the time)



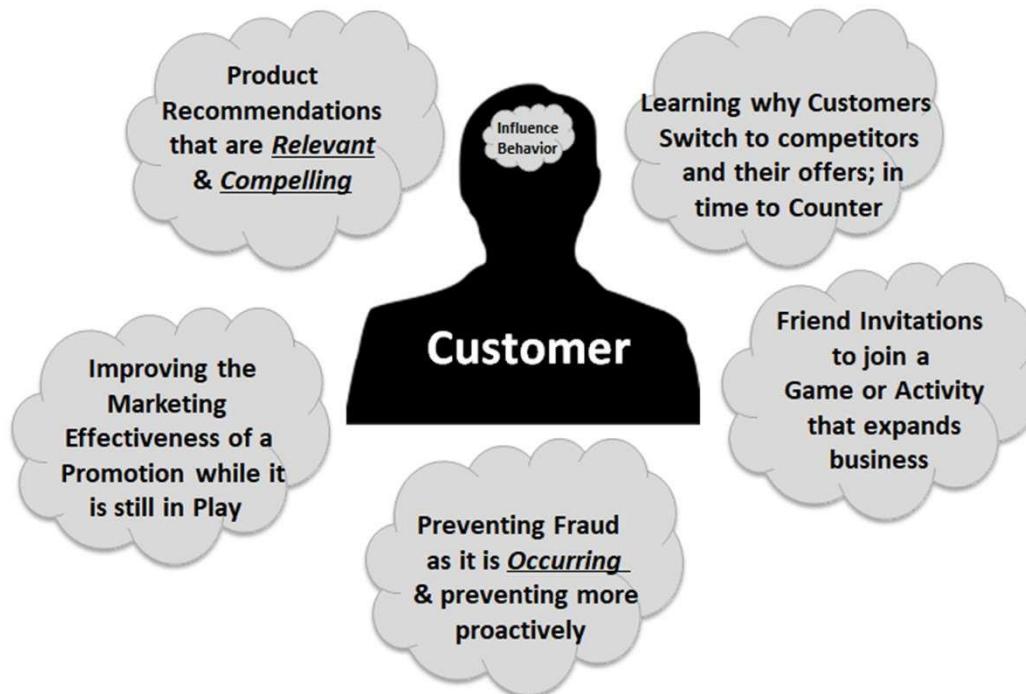
**Sensor technology and networks**  
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data. But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion



# Big Data: 3V's

*Velocity (Speed)*





# Big Data: 5V's

*Value*

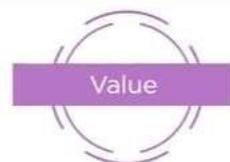




# Big Data: 5V's

## *Value*

How much data is useful and meaningful



Value refers to the ability to turn your data useful for business





# Big Data: 5V's

*Veracity*





# Big Data: 5V's

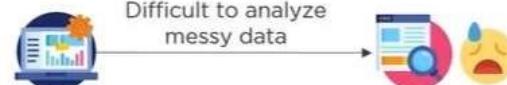
## *Veracity*

Trustworthiness of data in terms of quality and accuracy



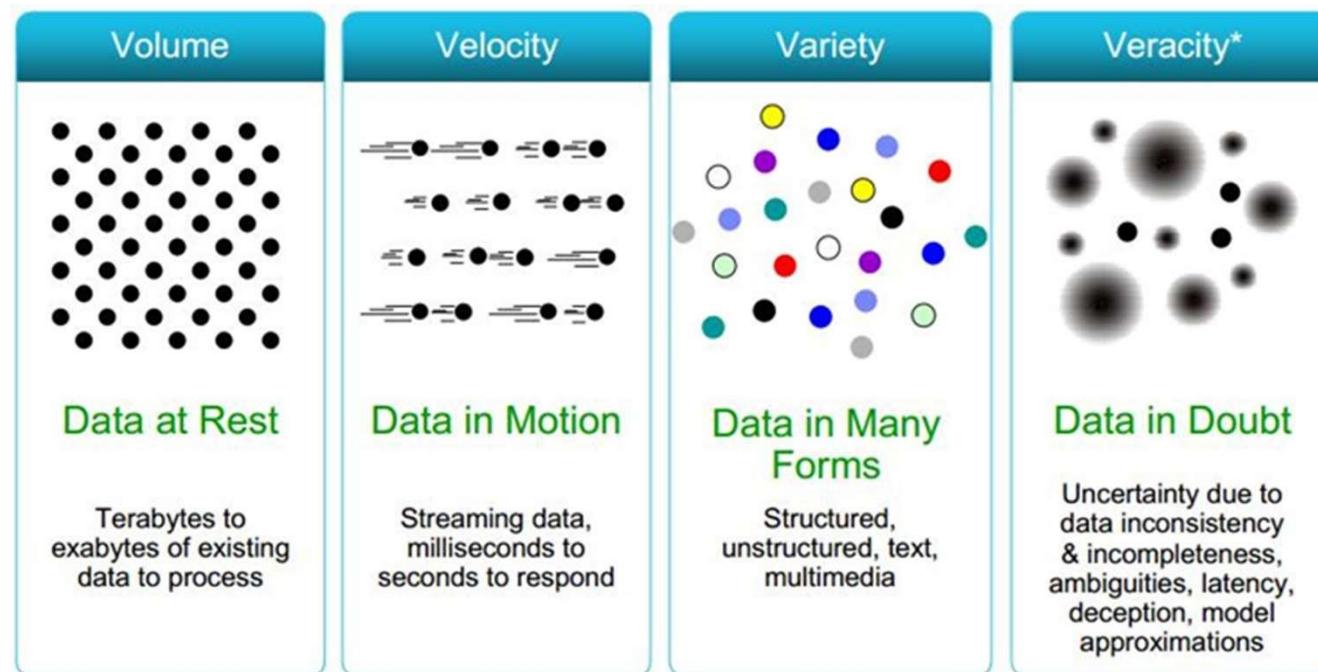
Extracting loads of data is not useful if the data is messy and poor in quality

Twitter posts with abbreviations, spelling mistakes, etc.



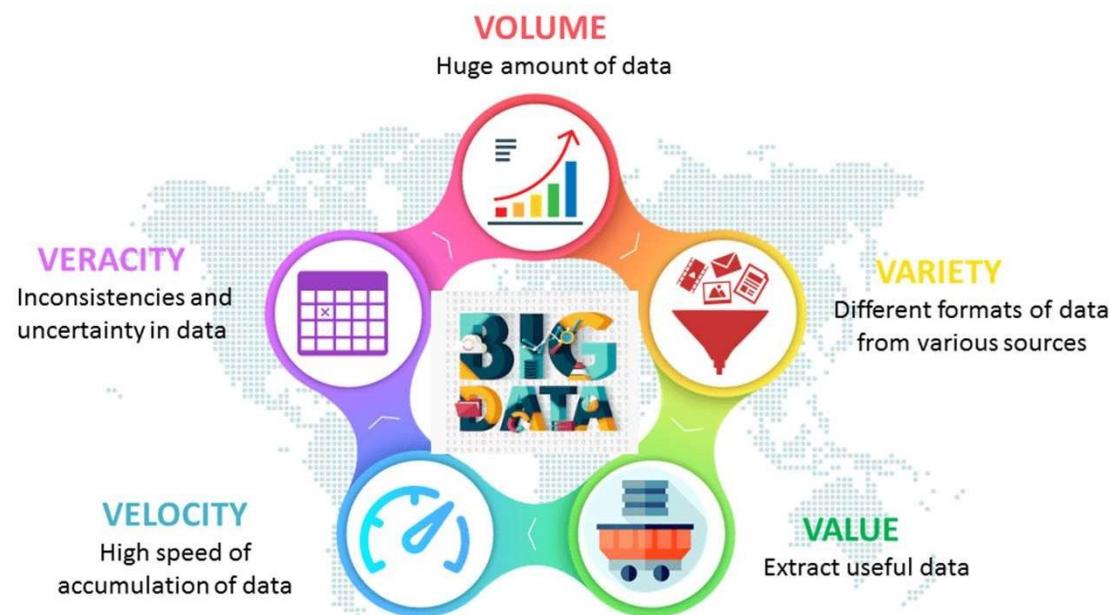


# Big Data: 4V's





# Big Data: 5V's



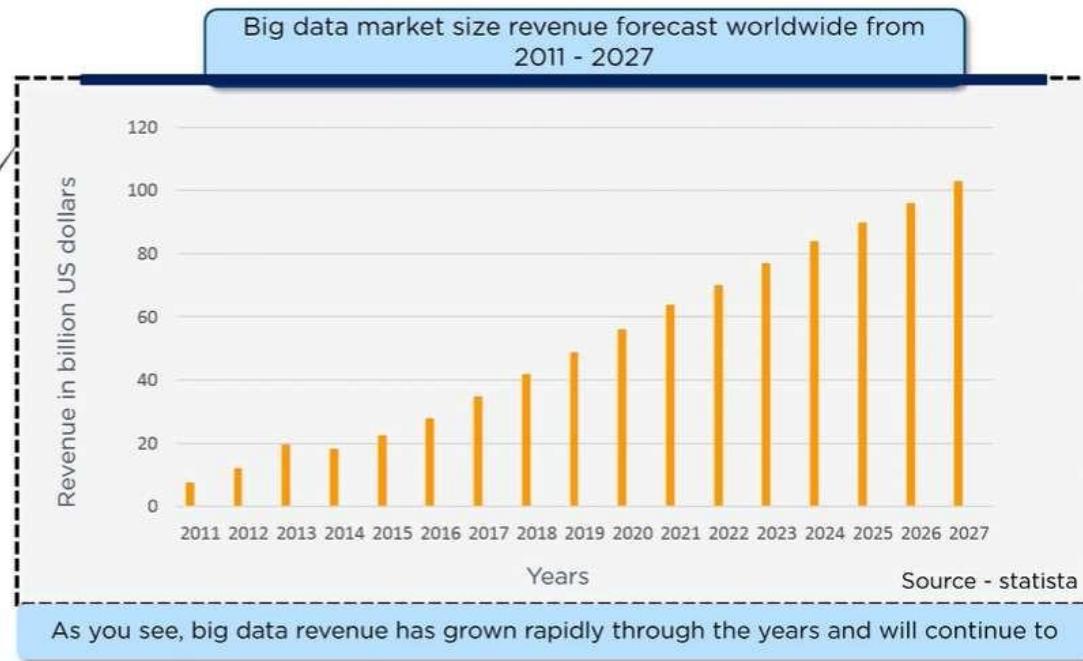


## Big Data: NV's

- The above image depicts the five V's of Big Data but as and when the data keeps evolving so will the V's. I am listing five more V's which have developed gradually overtime:
  - Validity: correctness of data
  - Variability: dynamic behaviour
  - Volatility: tendency to change in time
  - Vulnerability: vulnerable to breach or attacks
  - Visualization: visualizing meaningful usage of data



# Big Data: Applications





# Big Data: Applications





# Big Data: Applications

**BANKING AND SECURITIES**

**Challenges:**

- Early warning for Securities fraud and Trade visibility.
- Card fraud detection and audit trails.
- Enterprise credit risk reporting.
- Customer data transformation and analytics.

The Securities Exchange Commission (SEC) is using big data to monitor financial market activity by using network analytics and natural language processors. This helps to catch illegal trading activity in the financial markets.

**COMMUNICATIONS, MEDIA & ENTERTAINMENT**

**Challenges:**

- Collecting, analyzing and utilizing consumer insights.
- Leveraging mobile and social media content.
- Understanding patterns of real-time, media content usage.

Wimbledon Championships leverages big data to deliver detailed sentiment analysis on the tennis matches to TV, mobile and web users in real-time.



# Big Data: Applications



**3**

## HEALTHCARE PROVIDERS

**Challenges:**

- Rising Medical costs.
- Unavailability/inadequate/unusable Data.

Free public health data and Google Maps have been used by the University of Florida to create visual data that allows for faster identification and efficient analysis of healthcare information, used in tracking the spread of chronic disease.



**4**

## EDUCATION

**Challenges:**

• Incorporating data from varied sources.	• Untrained Staff and Institutions about Big Data	• Issues of privacy and data protection.
---	---	--

The University of Tasmania, Australia with over 26000 students has deployed a Learning and Management System that tracks, log time, time spent on different pages and the overall progress of a student over time.



# Big Data: Applications

 5

## MANUFACTURING & NATURAL RESOURCES

**Challenges:**

- Increase in the volume, complexity and velocity of data due to rising demands of Natural resources.
- Large volumes of untapped data from the manufacturing industry.
- Underutilization of data prevents improved quality, energy efficiency, reliability and better profit margins.

Enhancement in Supply chain capabilities from big data being used to increase productivity

 6

## GOVERNMENT

**Challenges:**

- Integration and
- Interoperability of big data.

The Food and Drug Administration (FDA) is using big data to detect and study patterns of food-related illnesses and diseases, allowing for faster response to treatments.



# Big Data: Applications

**INSURANCE**

**Challenges:**

- Lack of personalized services, pricing, targeted services to new market segments.
- Underutilization of data gathered by loss adjusters.
- Hunger for better insight.

**Customer insights for transparent and simpler products.**

**Predicting customer behavior through data derived from social media, GPS-enabled devices and CCTV footage.**

**Claims management, predictive analytics from big data has been used to offer faster service**

**RETAIL & WHOLESALE TRADE**

Unutilized Data derived from customer loyalty cards, POS scanners, RFID etc.

**Optimized staffing through data from shopping patterns, local events etc.**

**Reduced fraud and**

**Timely analysis of inventory**



# Big Data: Applications

 9 **TRANSPORTATION**

**Challenges:**

- Data from location-based social networks and high speed data from telecoms have affected travel behavior.
- Transport demand models are still based on poorly understood new social media structures.

**Some applications of big data by governments, private organizations and individuals include:**

<b>Governments use of big data:</b> traffic control, route planning, intelligent transport systems, congestion management (by predicting traffic conditions)	<b>Private sector use of big data in transport:</b> revenue management, technological enhancements, logistics and for competitive advantage (by consolidating shipments and optimizing freight movement)	<b>Individual use of big data includes:</b> route planning to save on fuel and time, for travel arrangements in tourism etc.
---	---	---

 10 **ENERGY & UTILITIES**

**Challenges:**

- 60% of electric grid assets will need replacement in this decade.
- Global installed wind capacity increased by 12.4%.
- Smart meters become mainstream, while consumers want more control & insights into energy consumption.

Smart meter readers allow data to be collected almost every 15 minutes. This granular data is being used to analyze consumption of utilities better which allows for improved customer feedback and better control of utilities use.



# Big Data: Applications

## Weather forecast



Challenge



It is very inconvenient when the weather changes suddenly. Imagine having storms, hurricanes, floods without a warning.

Solution

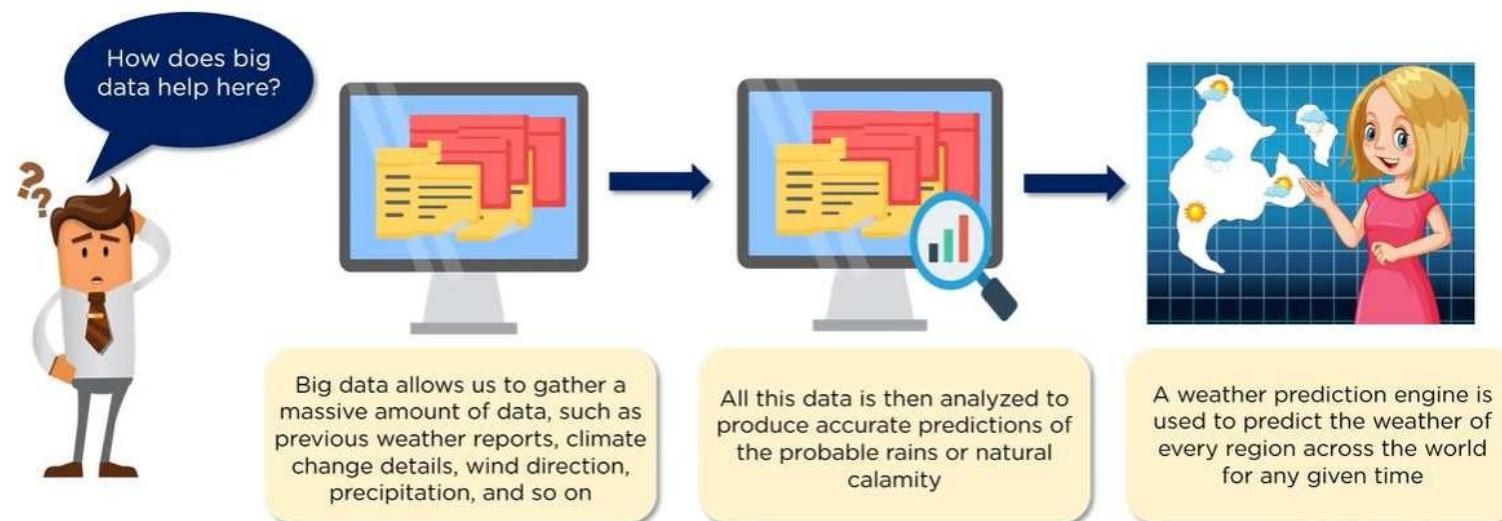


The solution is to design a system which predicts the weather conditions accurately. Big data is used in designing such a system



# Big Data: Applications

## *Weather forecast*





# Big Data: Applications

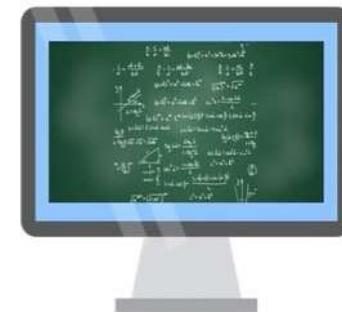
## *Weather forecast*



Predicting a landslide is very difficult with just the basic warning signs



The University of Melbourne developed an advanced tool which predicts the boundary where a landslide is likely to occur two weeks before



The predictions are made two weeks before the landslide occurs. The tool works on big data and applied mathematics



# Big Data: Applications

## *Media and entertainment*



Publishers are now able to gather more data about their visitors, which in turn enables them to serve relevant advertising

Used for targeted advertisement

Customer sentiment analysis

Recommendation engines

An insight into customer churn analysis



# Big Data: Applications

## *Media and entertainment*



The media and entertainment companies know how their customers feel by analyzing their emotions in posts, messages, and conversations

Used for targeted advertisement

Customer sentiment analysis

Recommendation engines

An insight into customer churn analysis



# Big Data: Applications

## *Media and entertainment*



Recommendation engines can accurately predict what a customer would like to watch next depending on their past watch and the browse history

Used for targeted advertisement

Customer sentiment analysis

Recommendation engines

An insight into customer churn analysis



# Big Data: Applications

## *Media and entertainment*



Analyzing the behavioral patterns of previously-churned customers, a marketer can identify the warning signs from current customers and prevent losing customers

Used for targeted advertisement

Customer sentiment analysis

Recommendation engines

An insight into customer churn analysis



# Big Data: Applications

*Media and entertainment*



Did you know that Starbucks app uses big data?





# Big Data: Applications

*Media and entertainment*



Did you know that Starbucks app uses big data?



Based on the customer's ordering history, the app suggests new products, sends personalized offers such as a birthday discount and so on



# Big Data: Applications

## Health care



With the available big data, medical researches are done very efficiently, and new treatments and medicines are discovered

Medical research

Personalized treatment

Cost reduction

Health of the population



# Big Data: Applications

## Health care



Hyper-personal care is provided to each depending on their past medical history

Medical research

Personalized treatment

Cost reduction

Health of the population



# Big Data: Applications

## Health care

Big data insights help in saving lives!

Readmissions account for a significant reason for increased healthcare costs. If all the available data is analyzed well, accurate insights regarding the patient's health are obtained

Medical research

Personalized treatment

Cost reduction

Health of the population



# Big Data: Applications

## Health care

Big data insights help in saving lives!

Analyzing big data helps in identifying disease trends based on geography and demographics

- Medical research
- Personalized treatment
- Cost reduction
- Health of the population



# Big Data: Applications

## *Health care*



United health care uses big data to detect medical fraud and identity threat



It predicts the likelihood of disease management programs to succeed, depending on how patients respond to it





# Big Data: Applications

## *Logistic*



The sensors within the vehicles analyze the fastest route to reach the destination. This dynamic routing plan will take into consideration the weather, traffic, and orders

Flexible routing

Capacity planning

Smart warehousing

Customer satisfaction



# Big Data: Applications

## *Logistic*



Here, predictive analytics is used to look into the availability of the workforce. This prevents more than 1 truck heading in the same direction and ensures more efficient operation

Flexible routing

Capacity planning

Smart warehousing

Customer satisfaction



# Big Data: Applications

## *Logistic*



Using big data analytics and tracking sensors, warehouse robotics is improved resulting in efficient resource allocation and reduced cost

Flexible routing

Capacity planning

Smart warehousing

Customer satisfaction



# Big Data: Applications

## *Logistic*



Semantic analysis and text processing helps to analyze customer reactions which will eventually create an instant feedback loop

Flexible routing

Capacity planning

Smart warehousing

Customer satisfaction



# Big Data: Applications

## *Logistic*



UPS is one of the biggest package shipping company in the world. They use a wide variety of data every moment

They use big data to optimize routes dynamically. The system can automatically change routes in real time. By doing so, UPS can anticipate traffic jams, bad weather, and so on



# Big Data: Applications

## Travel and tourism

We have the best packages!

Looking into past occupancy rates, room tariffs, school holidays, peak seasons, the tourism industry can anticipate demand and maximize the revenue

A bar chart on the tablet screen shows revenue increasing from January to April.

Month	Revenue (\$)
JAN	Low
FEB	Moderate
MAR	High
APR	Very High

Revenue management

Market research

Personalized offers

Investment opportunities



# Big Data: Applications

## *Travel and tourism*



Big data is used by companies to analyze information about their competitors which gives an understanding of what other hotels are offering their customers

Revenue management

Market research

Personalized offers

Investment opportunities



# Big Data: Applications

## Travel and tourism



Based on a tourists' past travel history and likes, they can receive personalized experiences that are focused on their needs

Revenue management

Market research

Personalized offers

Investment opportunities



# Big Data: Applications

## Travel and tourism



Few countries use Big Data to examine tourism flows and discover investment opportunities in their country.  
Investment in such areas is profitable

Revenue management

Market research

Personalized offers

Investment opportunities



# Big Data: Applications

*Travel and tourism*



Airbnb is one of the best online homestay networks. To help travelers around the world find the best properties, Airbnb analyzes all the big data to produce this result

It lists properties through customers preferences, keywords, and pricing. By doing so, it delivers its customers with the best stay



# Big Data: Applications

## *Government and law enforcement*

Lesser crime rates and more development!

POLICE

**Predictive policing**

Tackling unemployment

Poverty

Improving social and economic policies

Predictive policing uses big data to forecast criminal activity before it happens based on the information given to them through big data analysis



# Big Data: Applications

## *Government and law enforcement*



By analyzing the number of students graduating every year and the number of job openings, governments can have an idea of the unemployment in the country

Predictive policing

Tackling unemployment

Poverty

Improving social and economic policies.



# Big Data: Applications

## *Government and law enforcement*



Governments use big data tools to discover areas which fall under the poverty line and does the needful

Predictive policing

Tackling unemployment

Poverty

Improving social and economic policies



# Big Data: Applications

## *Government and law enforcement*



By analyzing the data collected from public surveys, governments can design efficient services and policies which will benefit the citizens

Predictive policing

Tackling unemployment

Poverty

Improving social and economic policies



# Big Data: Applications

*Government and law enforcement*



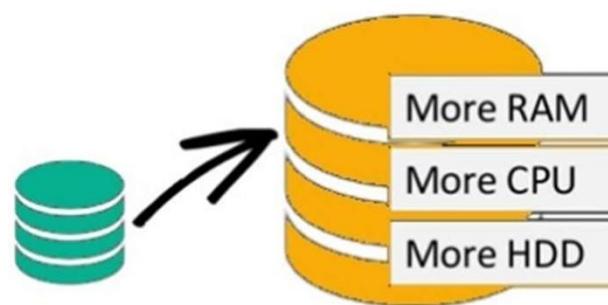
The New York Police Department uses big data analytics to protect its citizens

The department identifies crime trends, threats, and prevent crimes by analyzing data such as certain emails, fingerprints, and records from police investigations, and other public databases

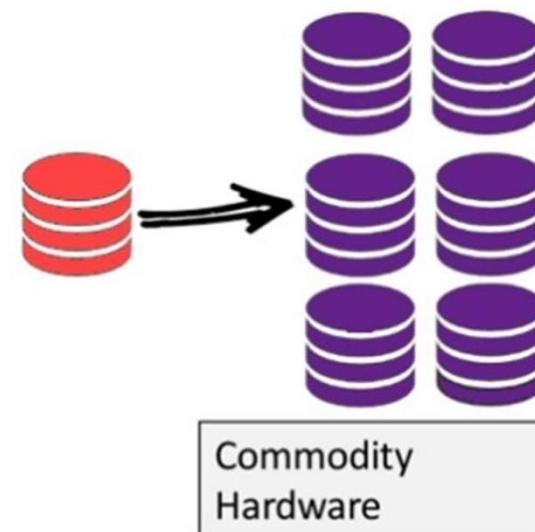


# Big Data: Scale

**Scale-Up (*vertical* scaling):**



**Scale-Out (*horizontal* scaling):**





# Big Data: Evolution

- The Model of Generating / Consuming Data has changed
  - Old Model: a few companies are generation data, all others are consuming data

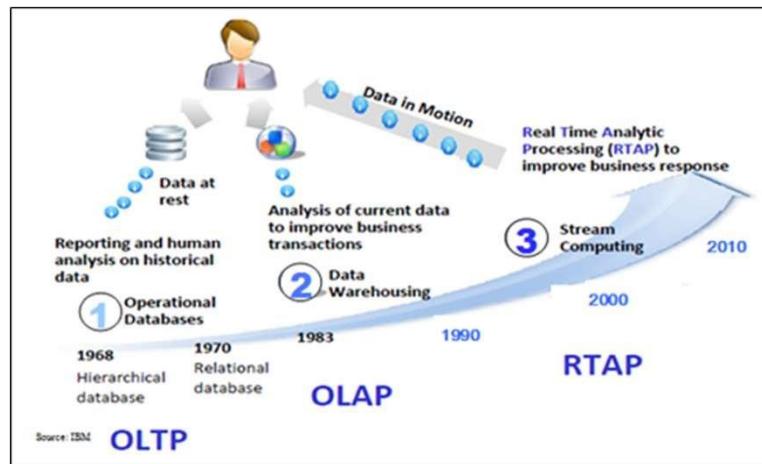


- New Model: All of us are generating data, and all of us are consuming data





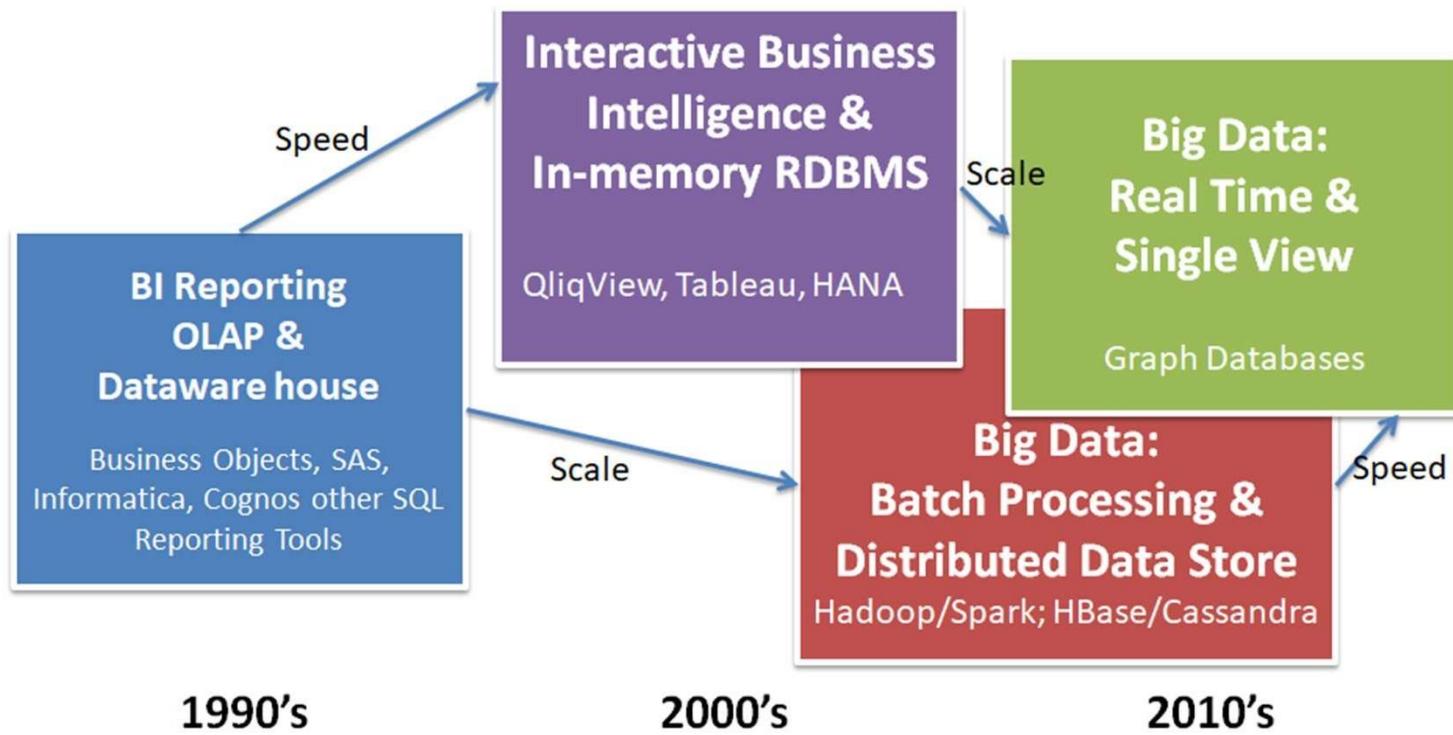
# Big Data: Evolution



- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-time Analytics Processing (Big Data Architecture & Technology)



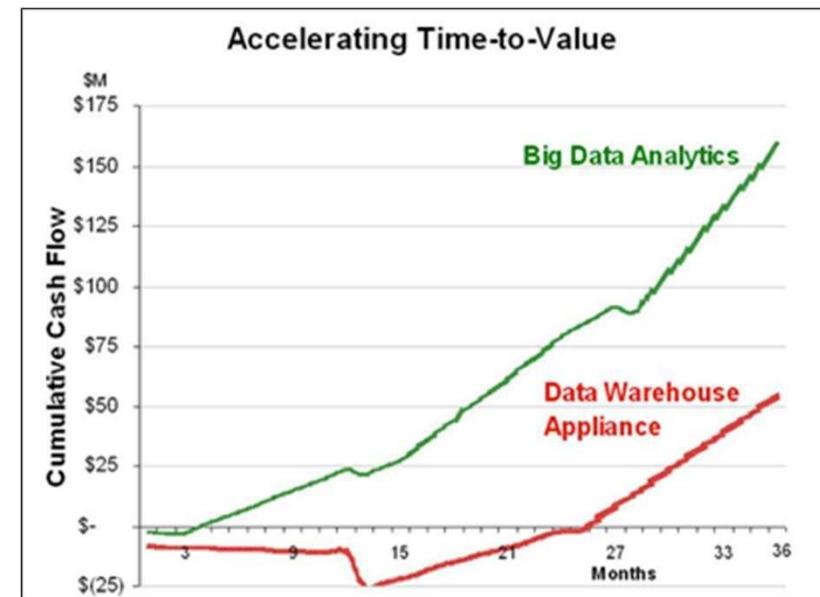
# Big Data: Evolution





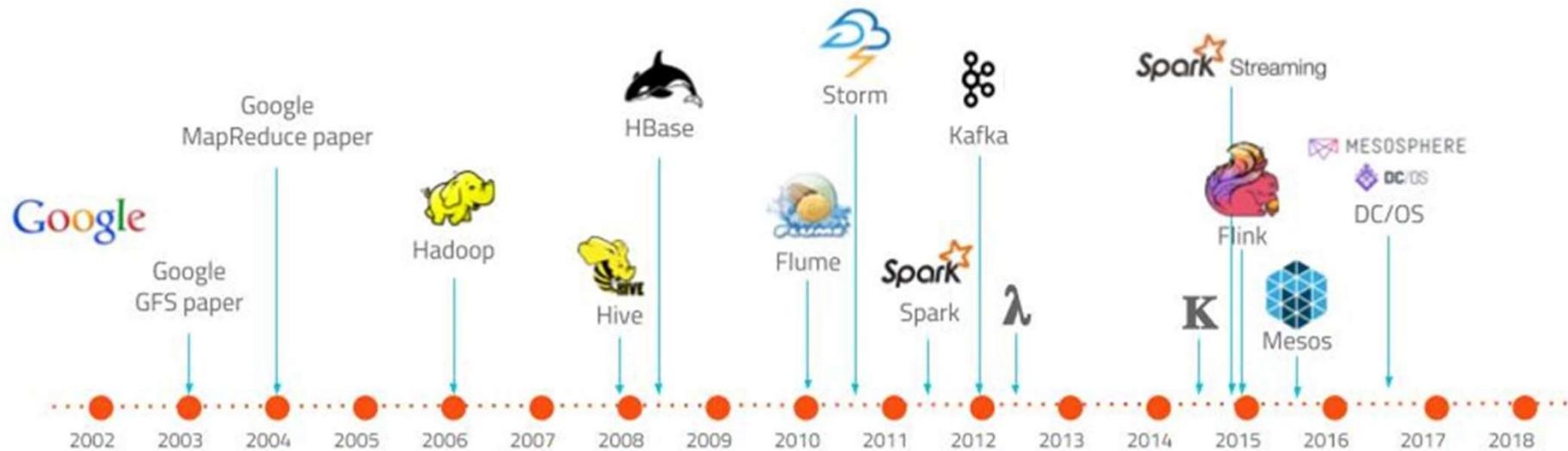
# Big Data: Evolution

- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps



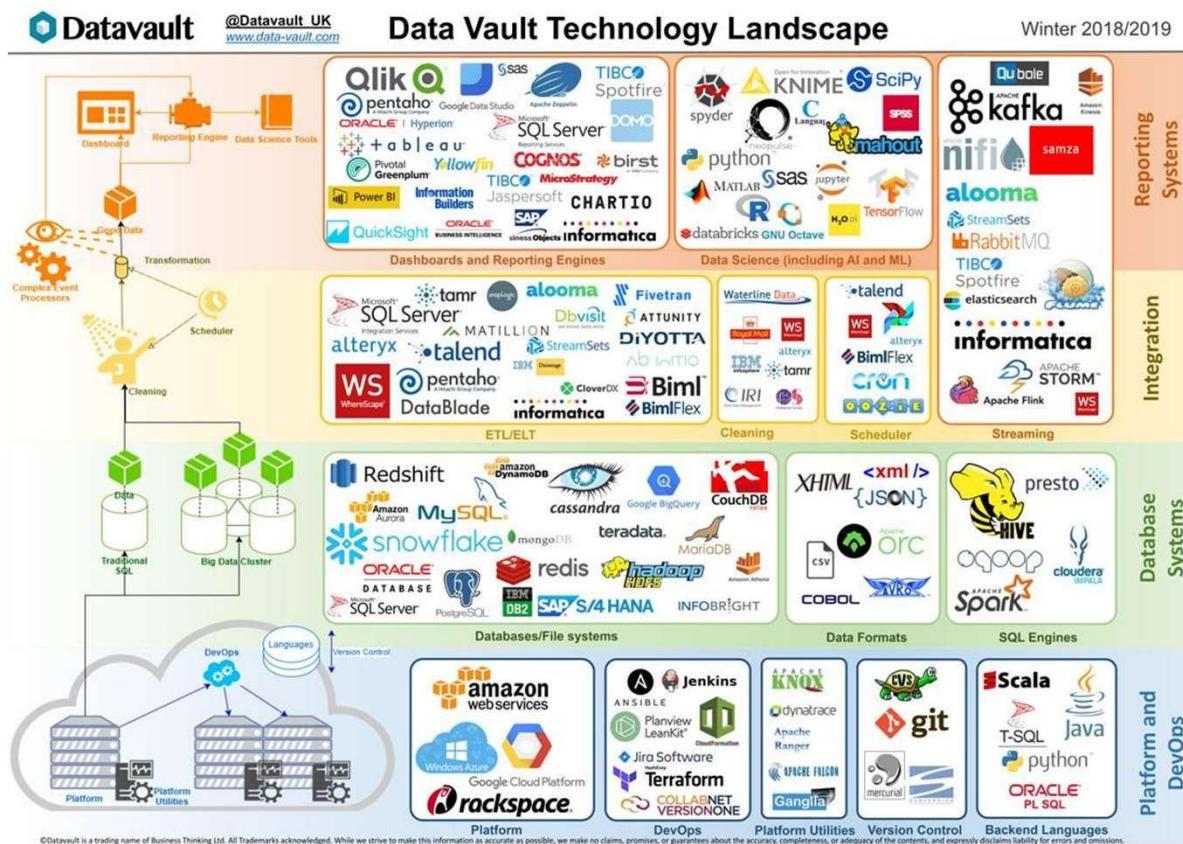


# Big Data: Evolution





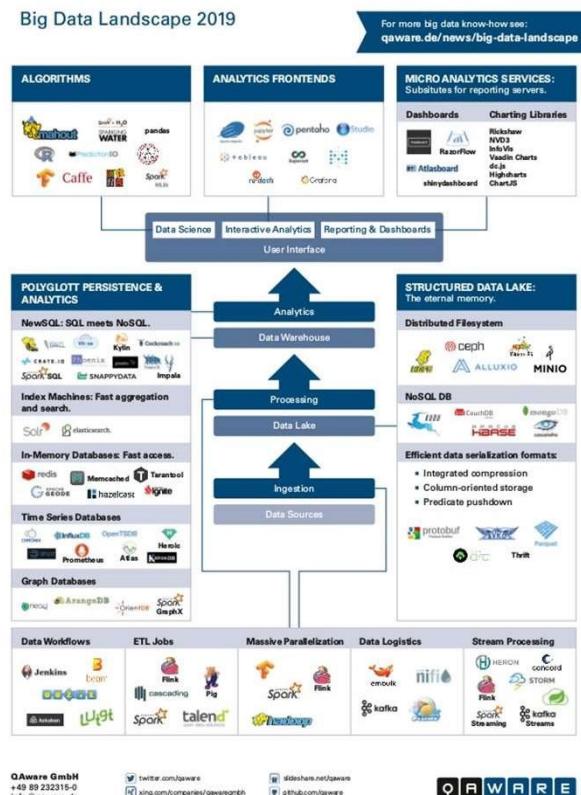
# Big Data: Landscape



Big Data



# Big Data: Landscape



# Big Data: Landscape (Open sources)

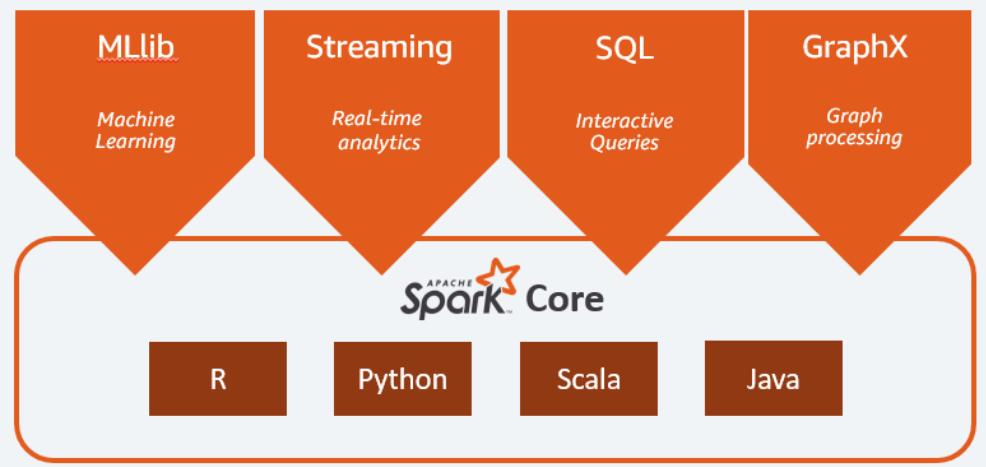
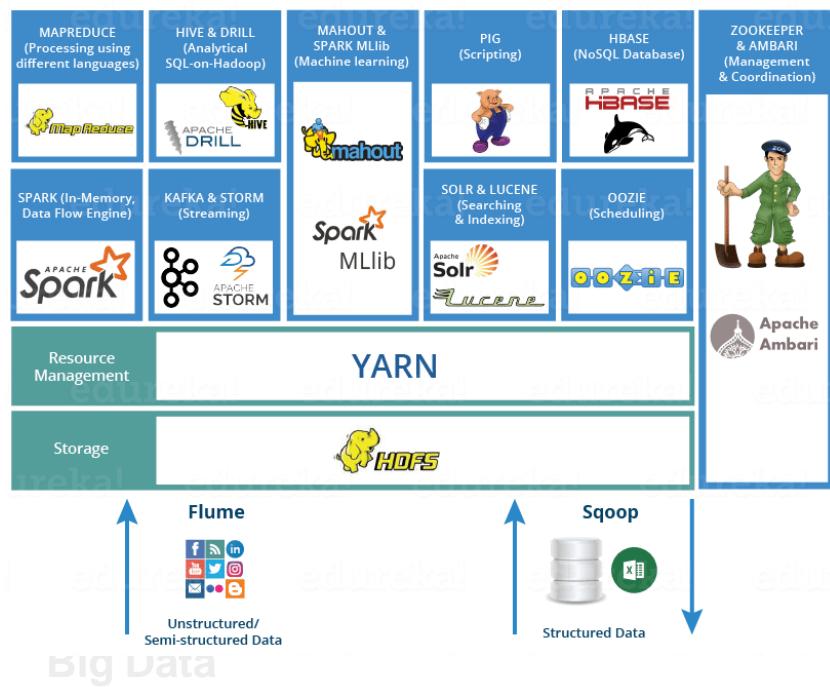


The Datafloq Open Source Landscape 2.0

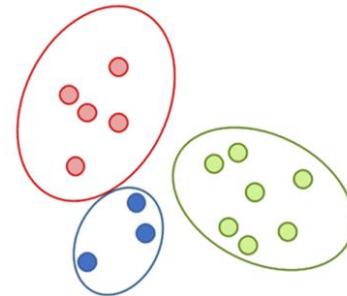
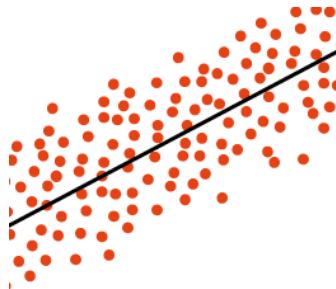
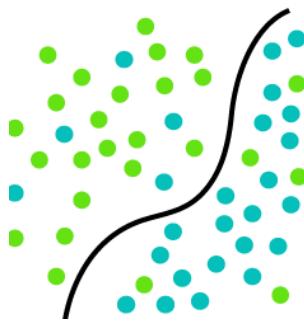




# In this course:



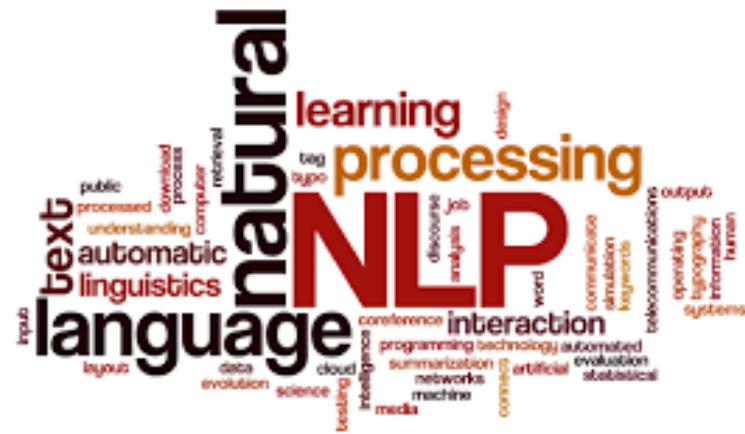
## Projects



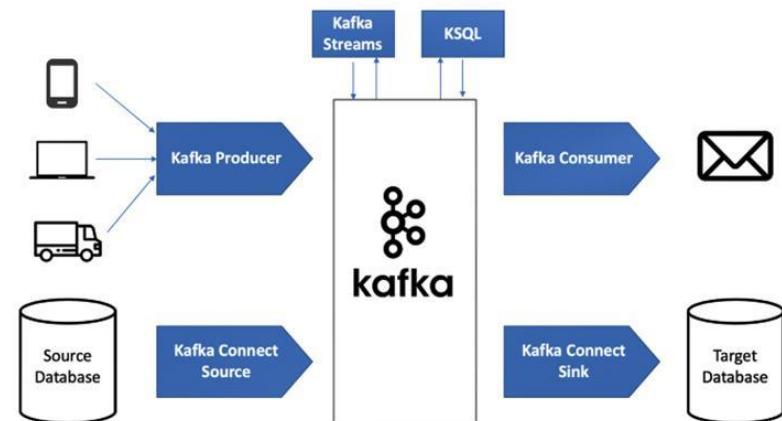
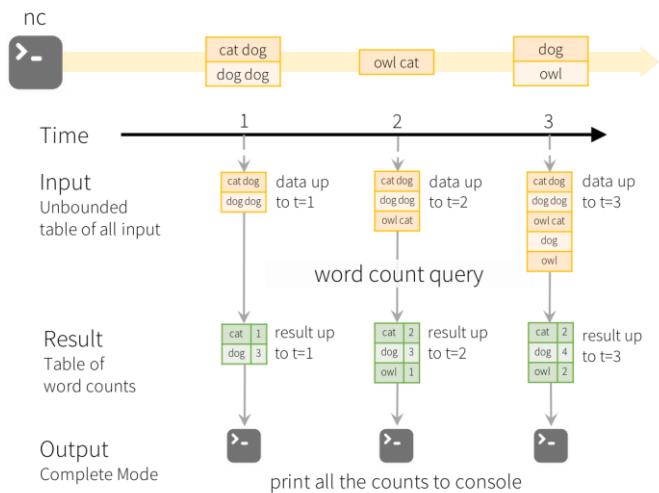
## Projects



## Projects



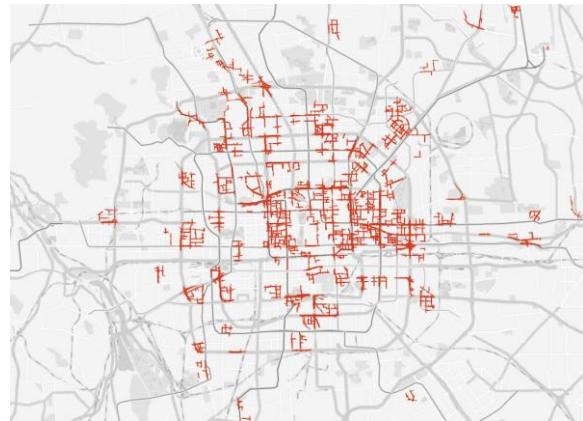
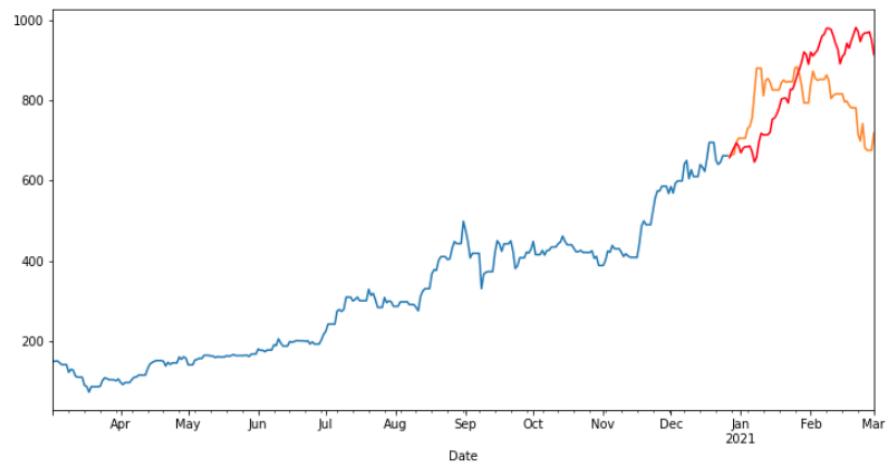
# Projects



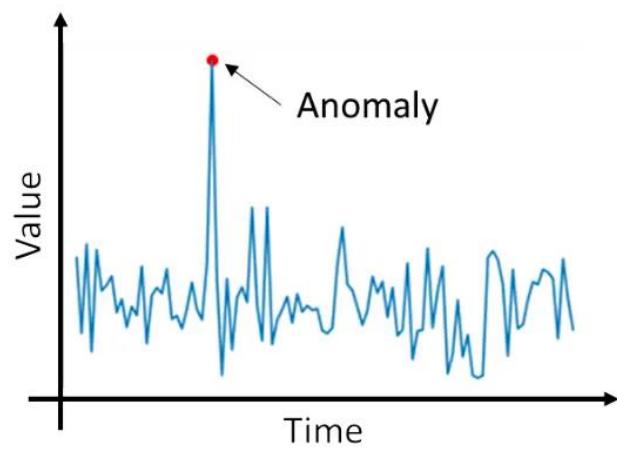
## Projects



## Projects



## Projects





# In this course:



## UIT Spring 2021 DS200 Assignment 9

Sentiment analysis of tweets

InClass · 6 days ago



## UIT Spring 2021 DS200 Assignment 8

Movie Recommendation System

InClass · 13 days ago



## MSIS405.L21.CTTT Spring 2021 Assignment 8

Predict acceptability of cars

InClass · 6 days ago



## UIT Spring 2021 DS200.L11 Assignment 8

Movie recommendation system

InClass · 18 days ago



## UIT Spring 2021 DS200.L11 Assignment 7

Predict the acceptability of cars

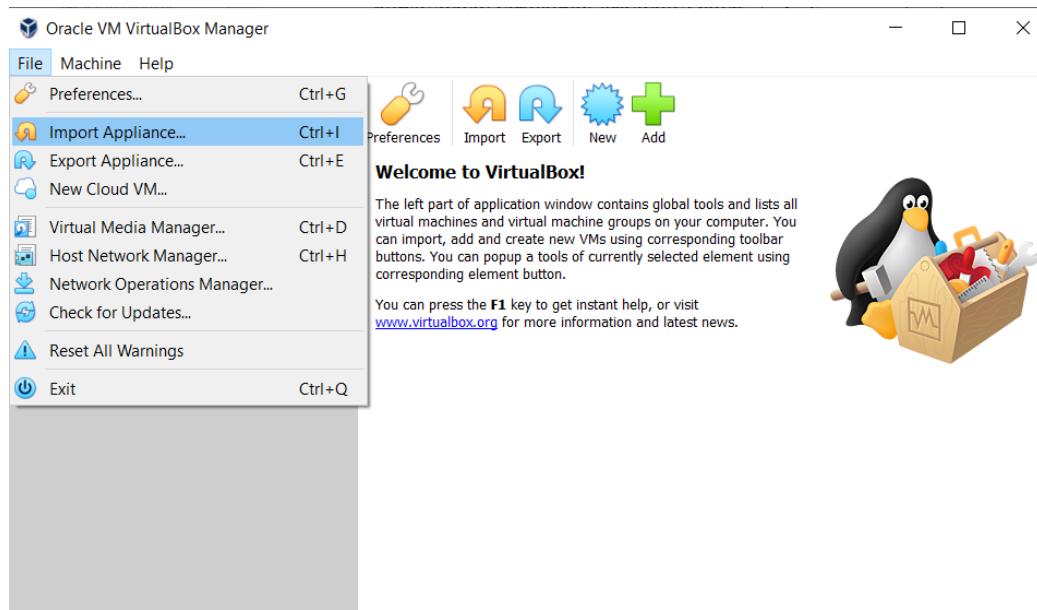
InClass · 21 days ago

# Install Cloudera Quickstart VM

- Download Cloudera Quickstart VM for VirtualBox

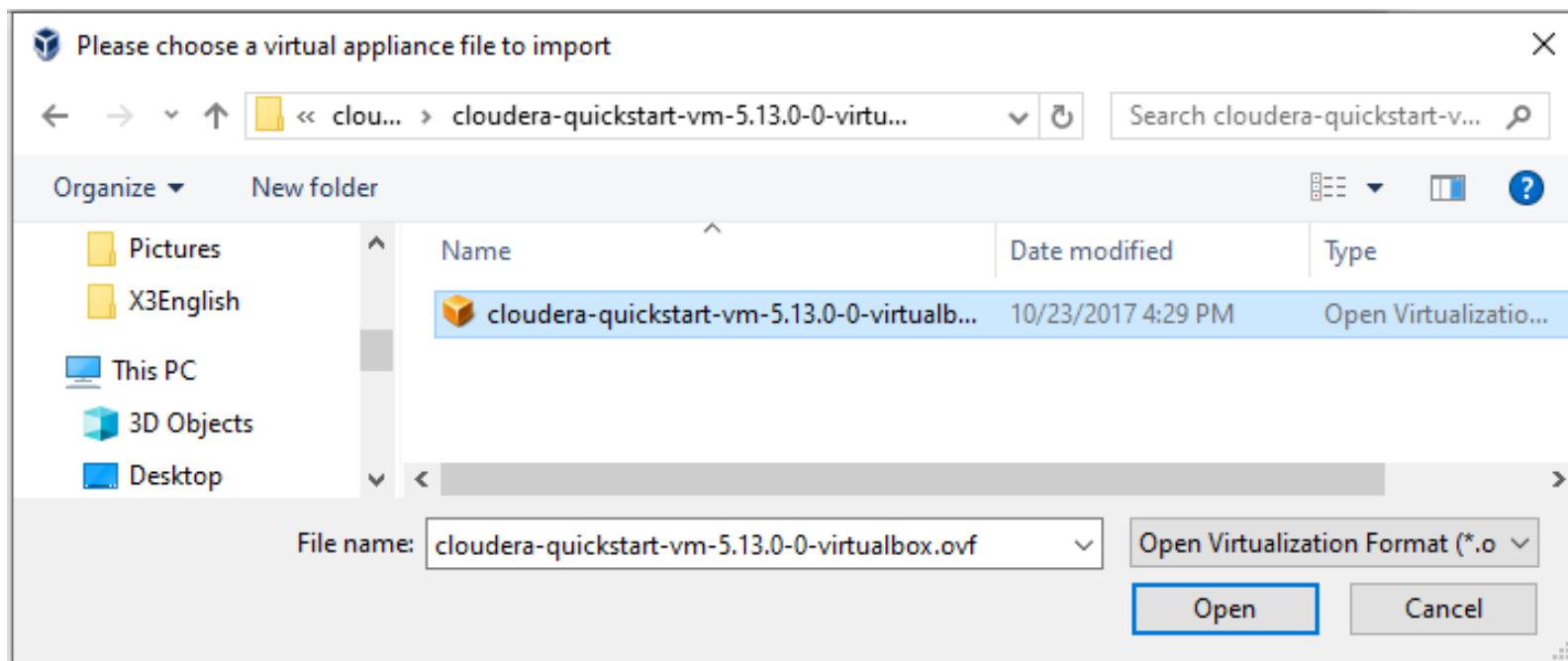
[https://downloads.cloudera.com/demo\\_vm/virtualbox/cloudera-quickstart-vm-5.13.0-0-virtualbox.zip](https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.13.0-0-virtualbox.zip)

- Open Oracle VM VirtualBox. Click File → Import Appliance

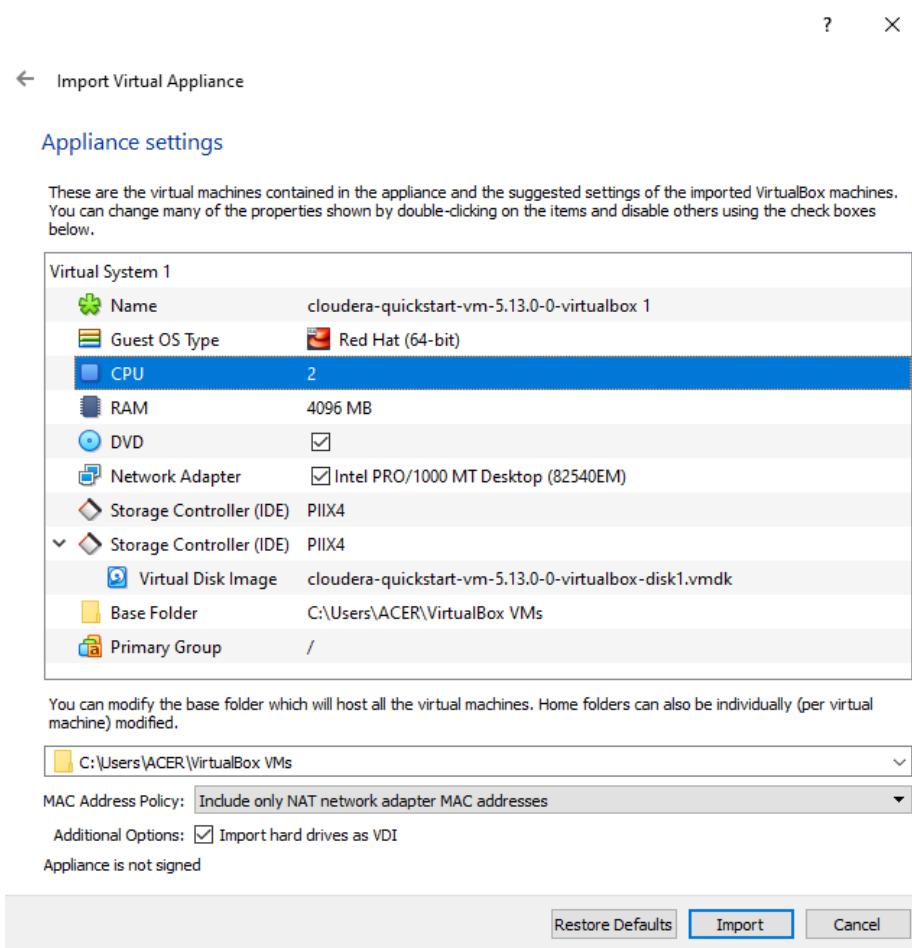


# Install Cloudera Quickstart VM

Choose “*cloudera-quickstart-vm-5.13.0-0-virtualbox.ovf*”



# Install Cloudera Quickstart VM

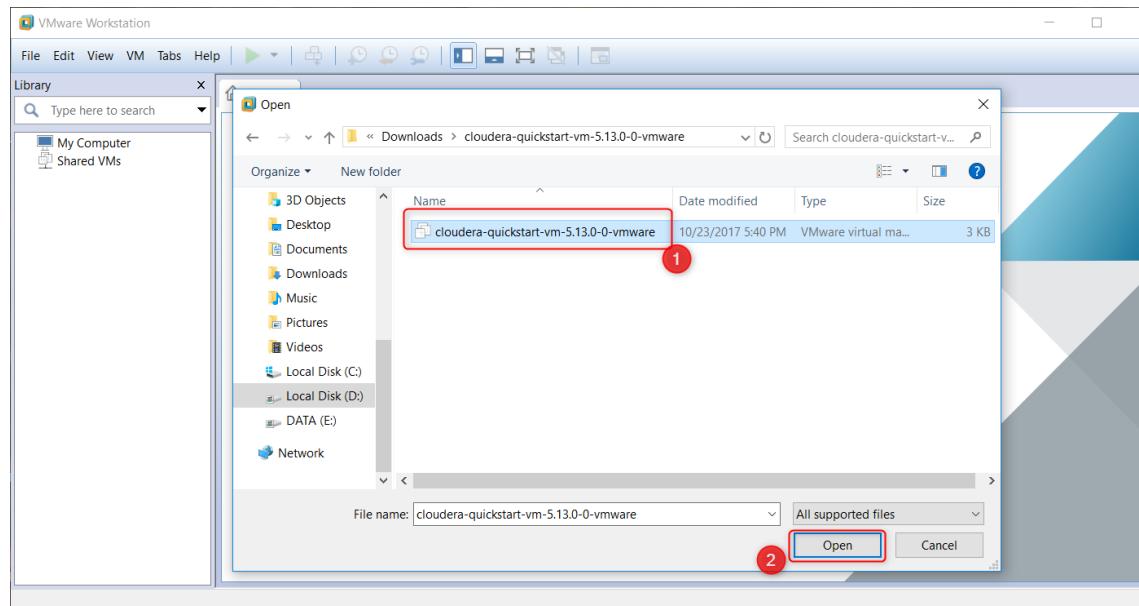


# Install Cloudera Quickstart VM

VMware

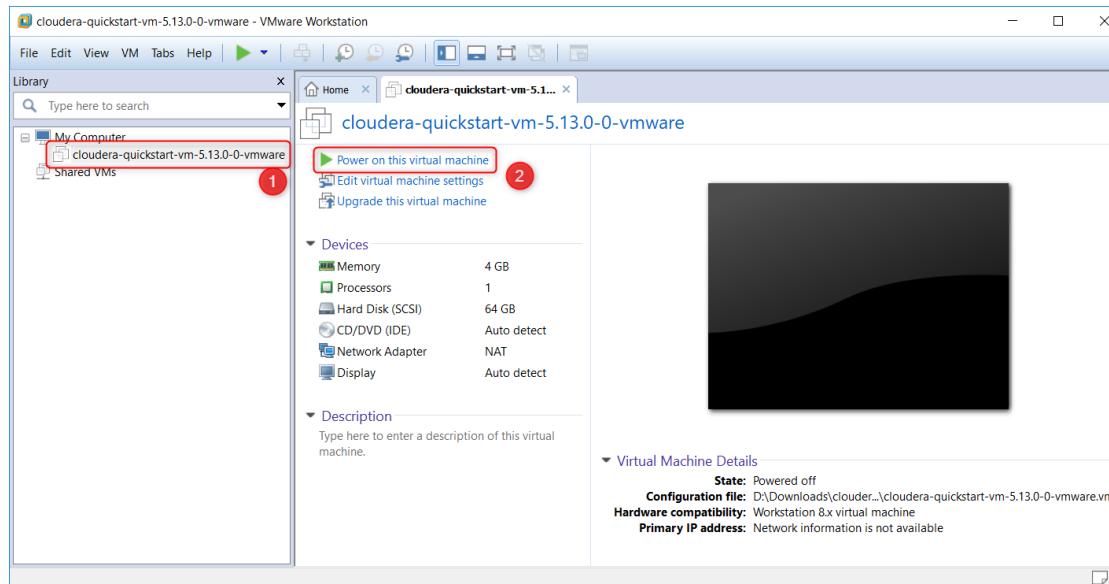
- Download Cloudera Quickstart VM for Vmware

[https://downloads.cloudera.com/demo\\_vm/vmware/cloudera-quickstart-vm-5.13.0-0-vmware.zip](https://downloads.cloudera.com/demo_vm/vmware/cloudera-quickstart-vm-5.13.0-0-vmware.zip)



# Install Cloudera Quickstart VM

VMware



# Q & A



## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.



# Big Data

## (Hadoop)

Instructor: Trong-Hop Do

July 4<sup>th</sup>, 2021

A photograph of a railway track curving into the distance under a dramatic, cloudy sky.

**“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”**

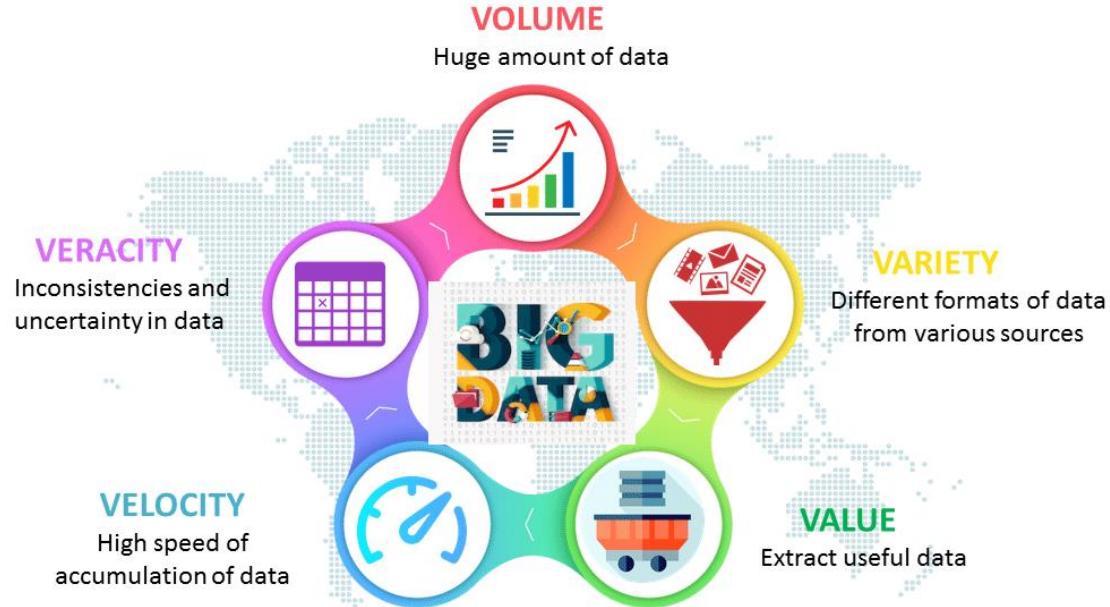
– Chris Lynch, Vertica Systems

# Distributed File System

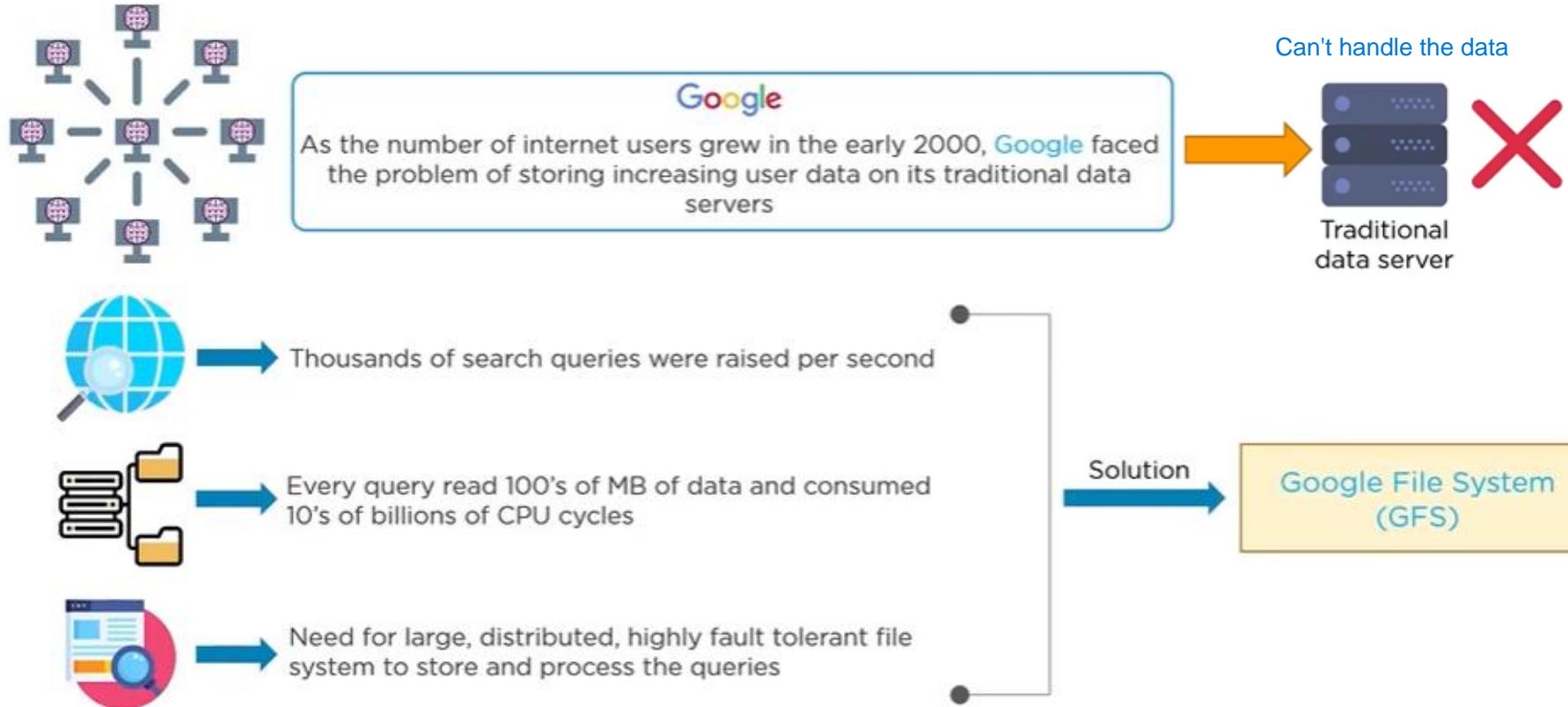
---

Google File System

# Recap – Big data 5V's



# Big data case study – Google File System



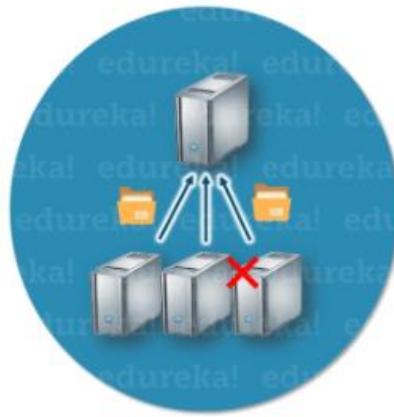
# Big data case study – Google File System



Storing huge and exponentially growing datasets

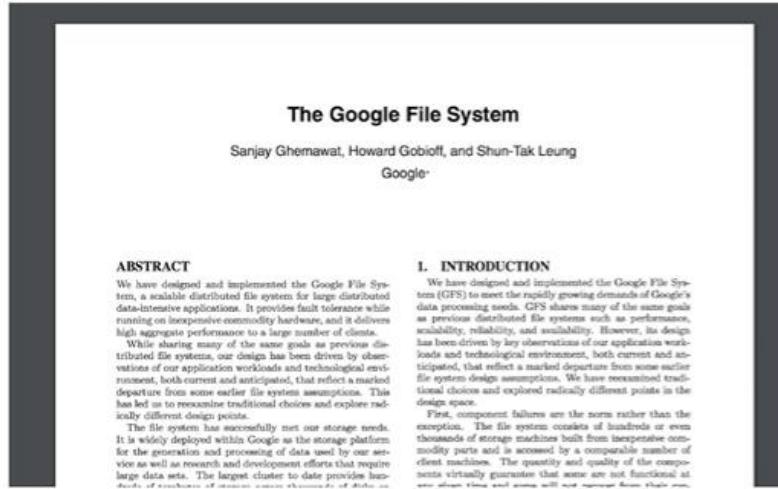


Processing data having complex structure  
(structured, un-structured, semi-structured)



Bringing huge amount of data to computation unit becomes a bottleneck

# The 2003 Google File System paper

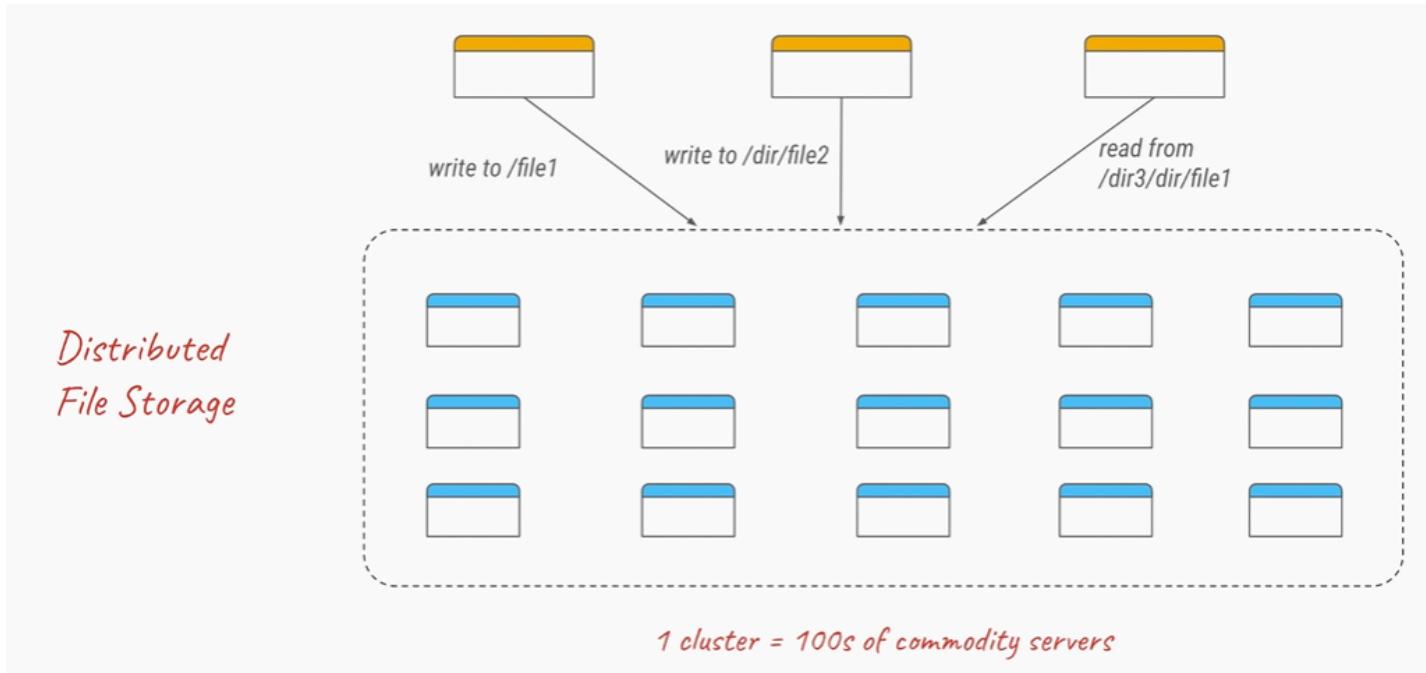


Paper describing  
Google File System



Used as a basis to  
design Hadoop

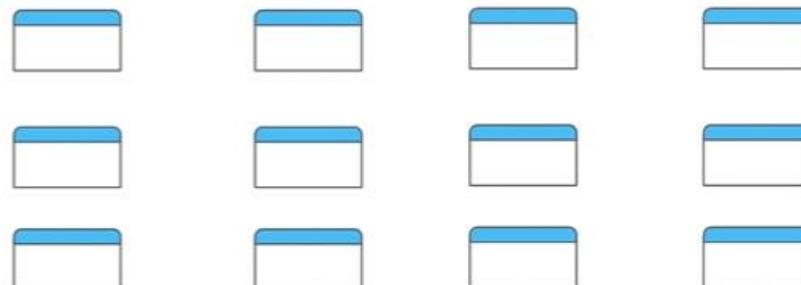
# What is Google File System?



# GFS design consideration: Commodity hardware

Failures are common

- disk / network / server
- OS bugs
- human errors



*Commodity servers are cheap & can be made to scale horizontally with right software*

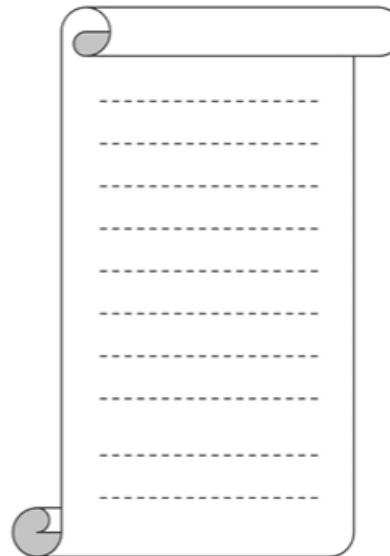
# GFS design consideration: Large files

100MB to multi-GB files

- Crawled web documents
- Batch processing

we collect the data in 7/month/year in one time

Streaming processing : Data need for IO

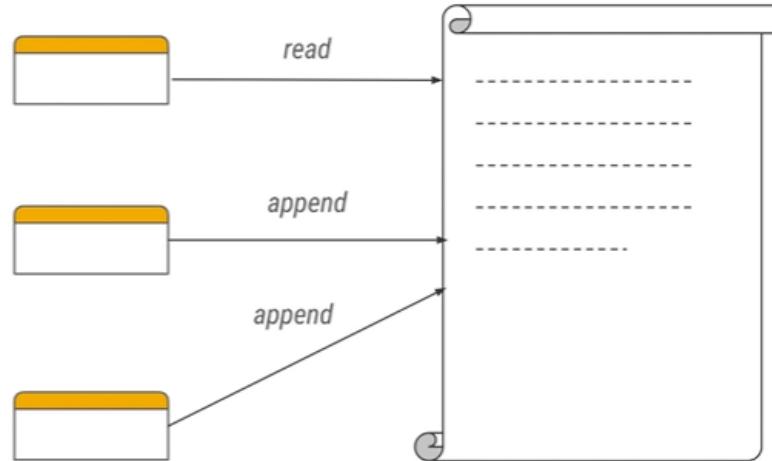


*Commodity servers are cheap & can be made to scale horizontally with right software*

# GFS design consideration: File operations

Read + Append only

- No random writes
- Mostly sequential reads

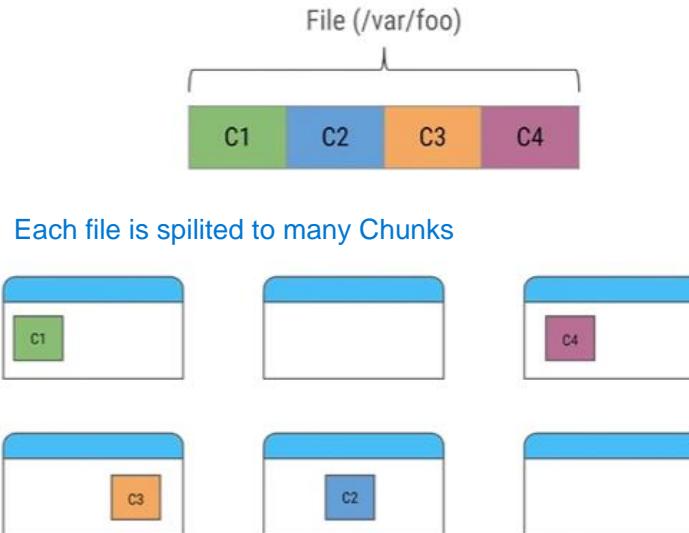


*Think of web crawling - Keep appending crawled content & Use batch processing (reads) to create index*

# GFS design consideration: Chunks

Files split into chunks

- Each chunk of 64MB
- Identified by 64 bit ID
- Stored in Chunkservers



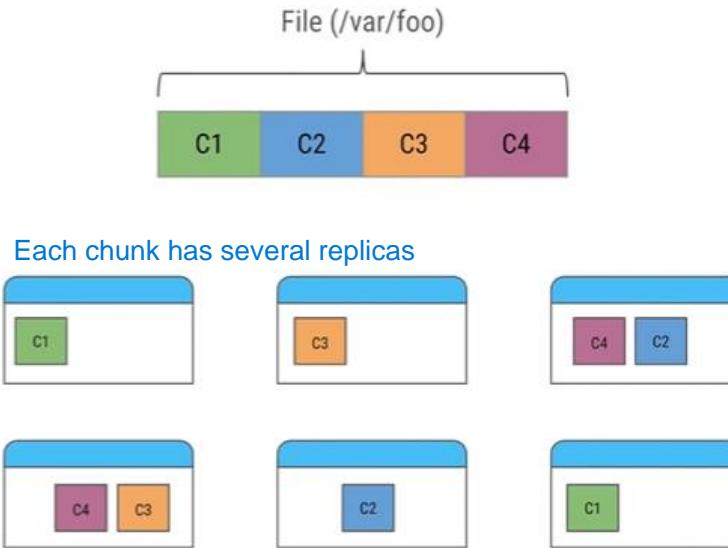
Each file is spilted to many Chunks

*Chunkservers - Chunks of single file are distributed on multiple machines*

# GFS design consideration: Replicas

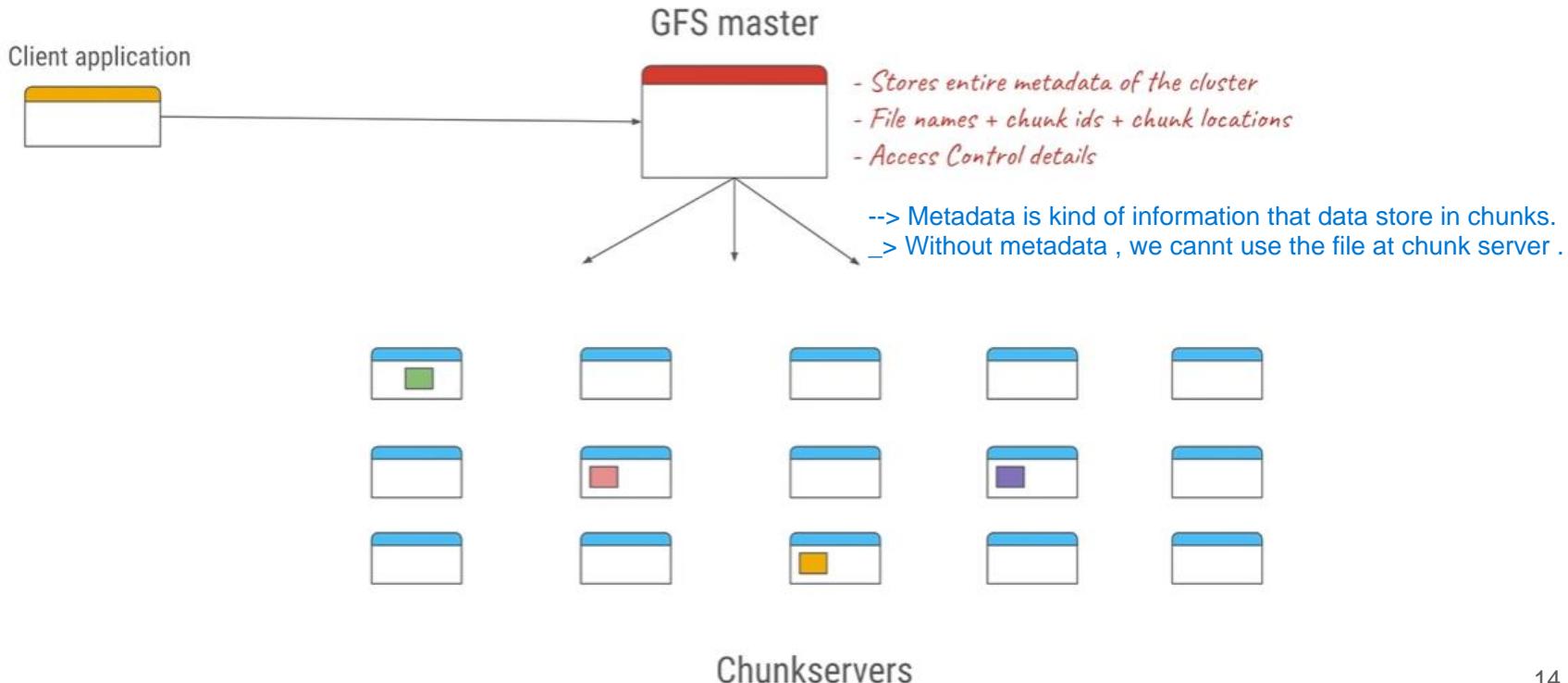
Files split into chunks

- Replica count by client
- commodity server failures

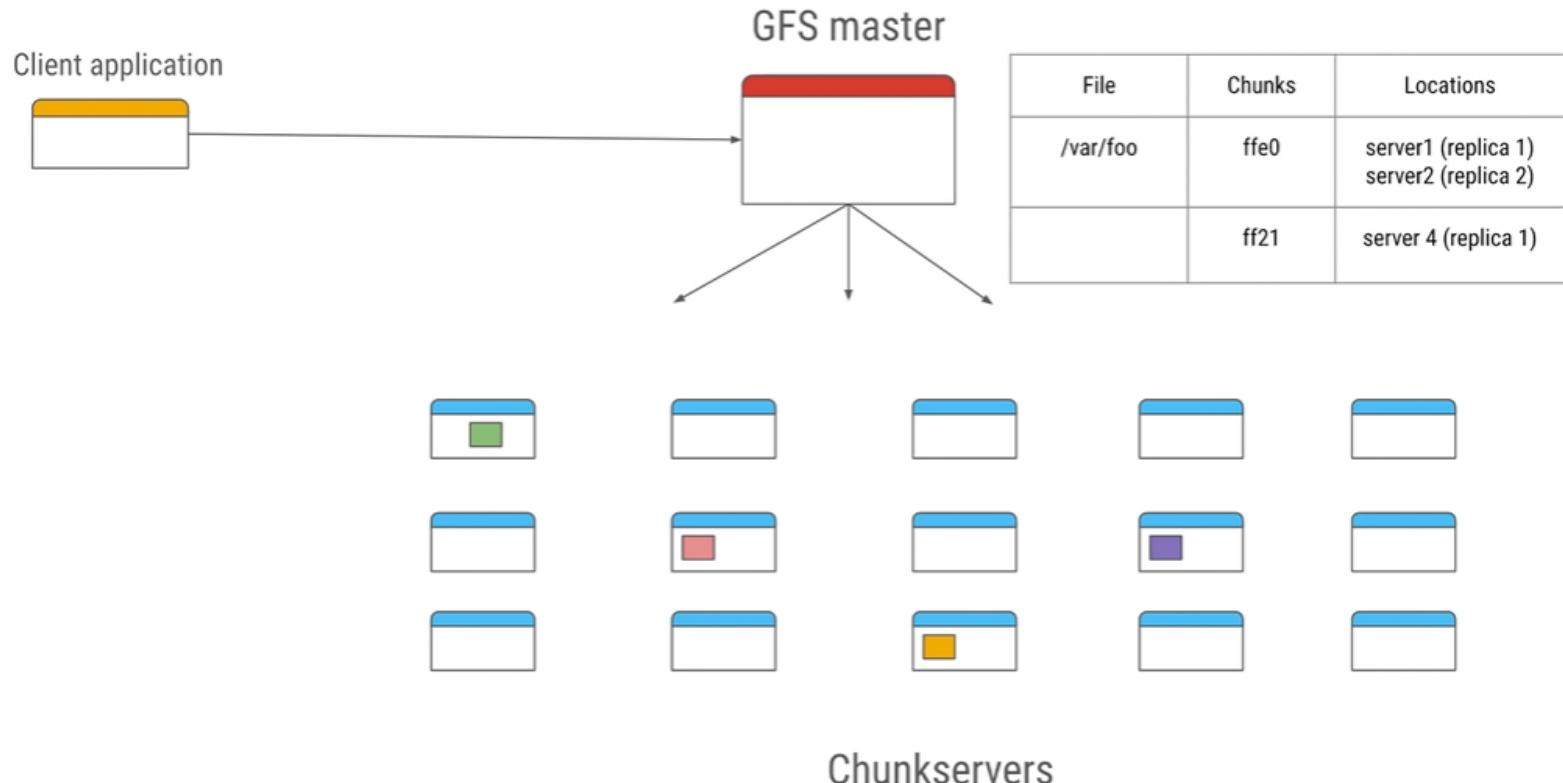


*Replicas ensure durability of data if chunkserver goes down*

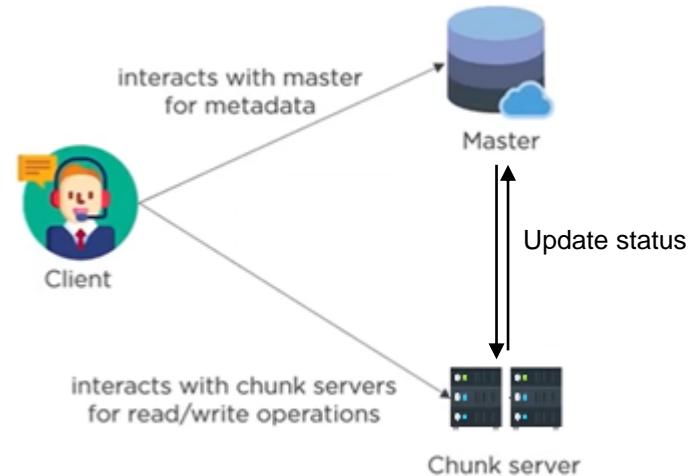
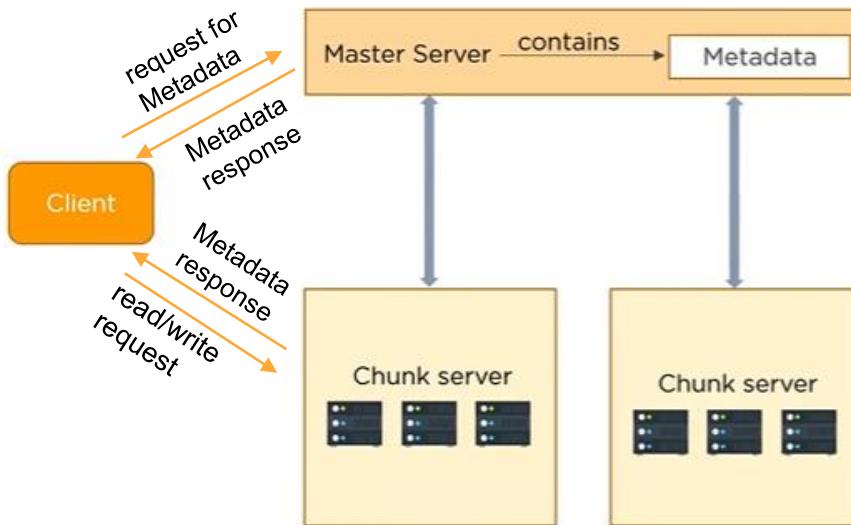
# GFS design consideration: Large files



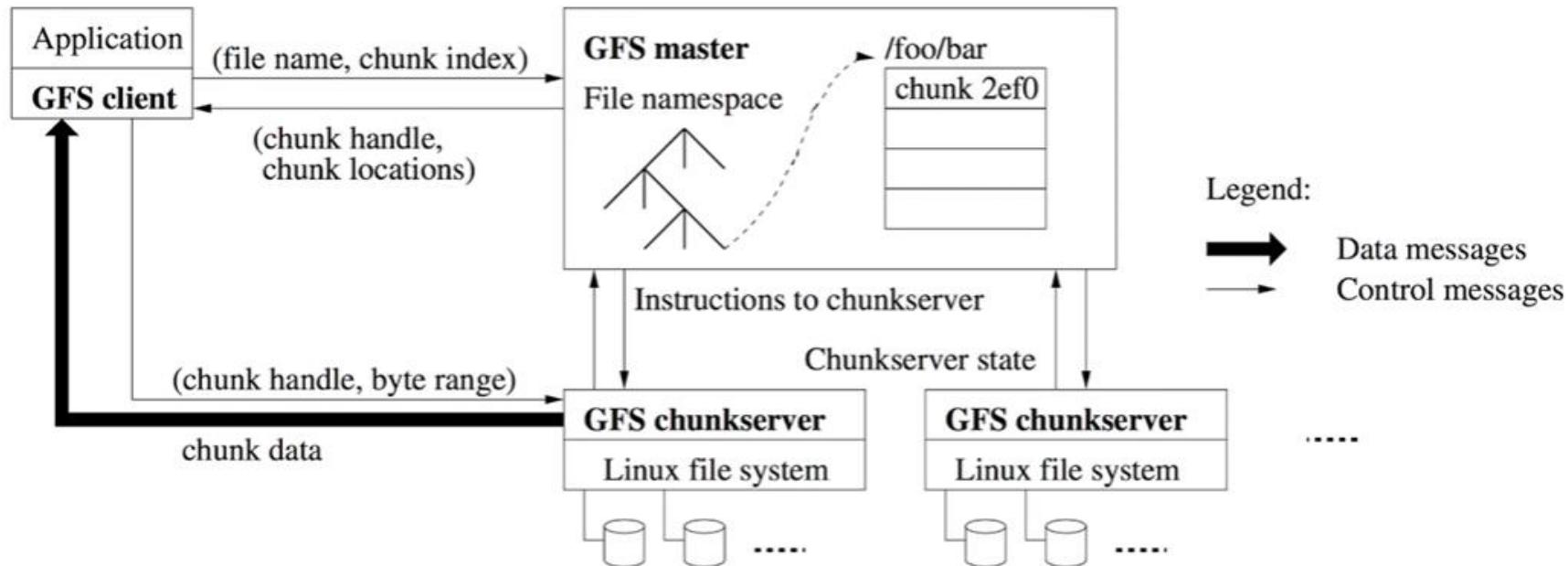
# GFS architecture



# GFS architecture

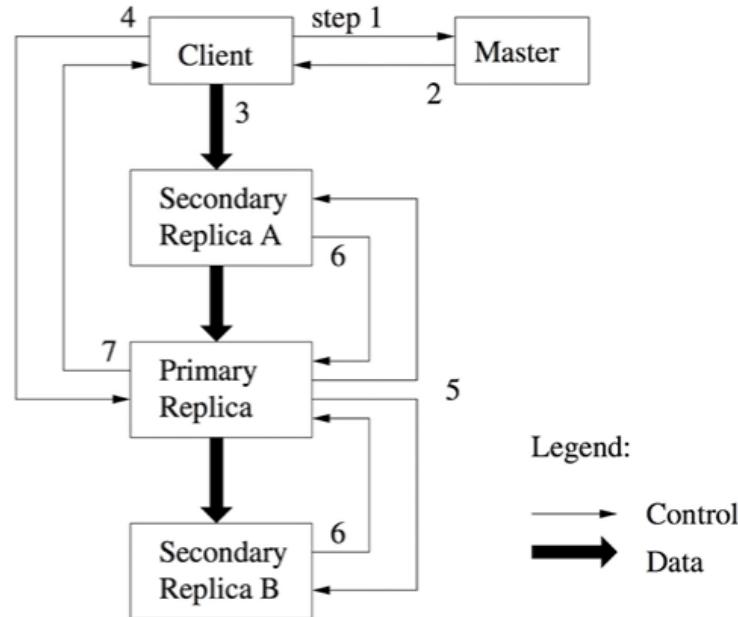


# GFS: Read



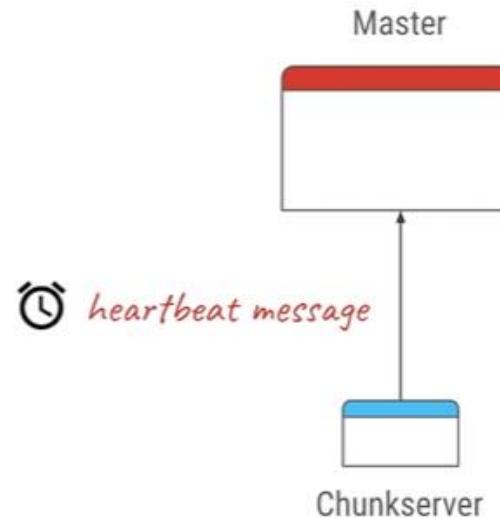
# GFS: Write

1. Ask for locations to write
2. Get replicate locations
3. Write data to closest replica.
4. Request commit to primary
5. Primary instructs order of writes to secondaries
6. Secondaries acknowledge
7. Primary ack to client



# GFS: Heartbeat

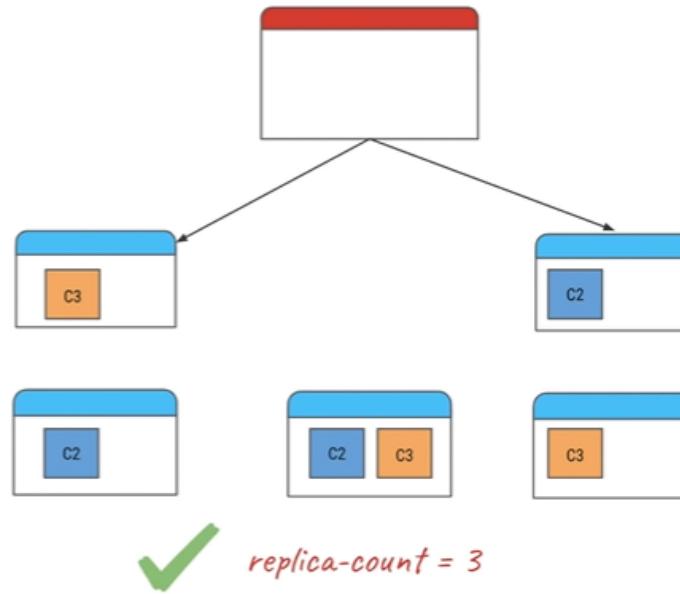
Regular heartbeats to ensure chunkservers are alive



# GFS: Ensure chunk replica count

If chunkserver is down,  
master ensures all chunks  
that were on it are copied  
on other servers.

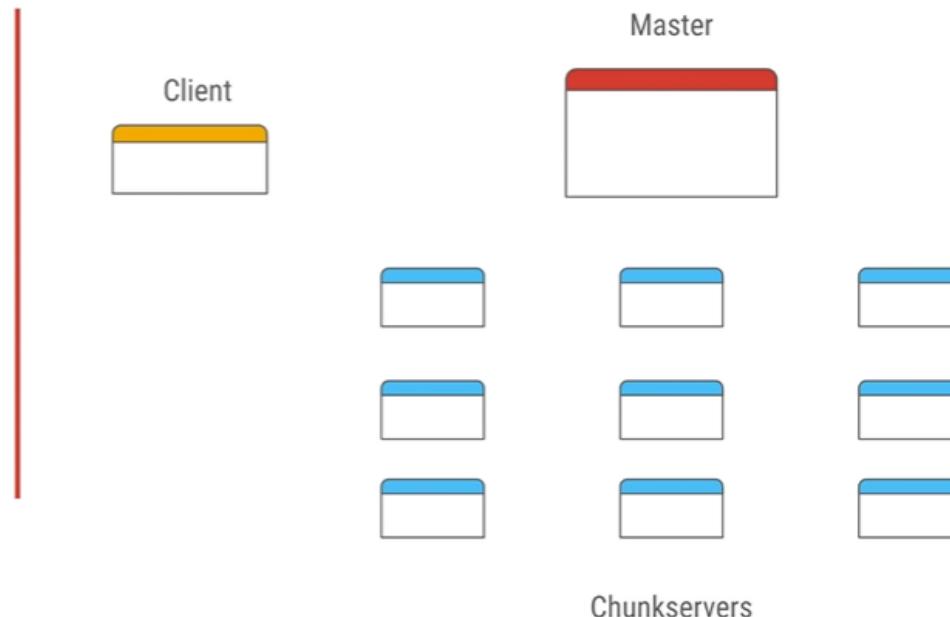
Ensures replica counts  
remains same.



# GFS: Single Master for multi-TB cluster

Large chunk size

- 64MB chunk
- Reduced meta-data
- Reduces client interactions
- Client caches location data

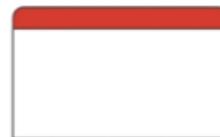


# GFS: Operation logs

## Record of all ops

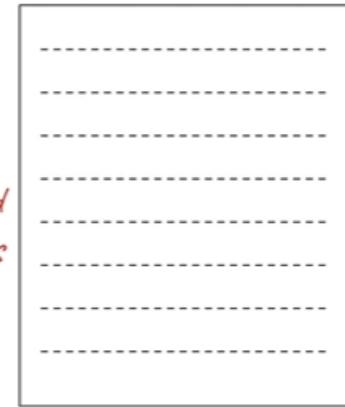
- Checkpointed regularly
- Happens in background thread
- Used if master crashes
- Rebooted master replays log

Master



*File operations and  
their timestamps*

Operations Log

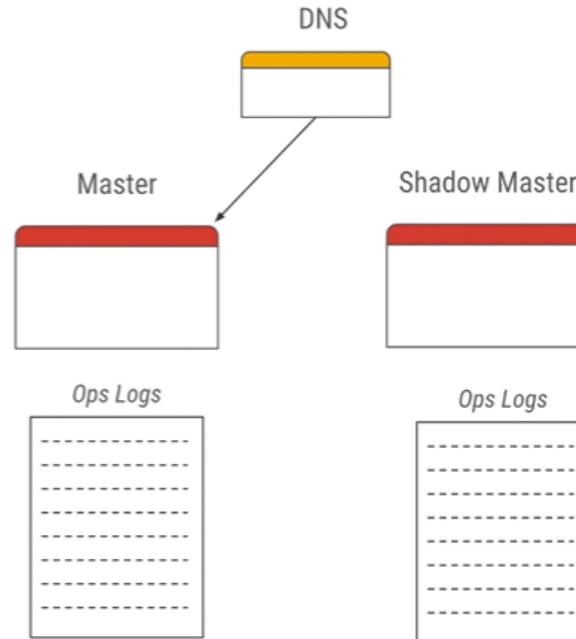


*Written to disk and  
remotely*

# GFS: Operation logs

## Single Point of Failure

- Ops log is replicated remotely
- Shadow master uses the logs
- DNS change can change master
- Shadow master may lag slightly

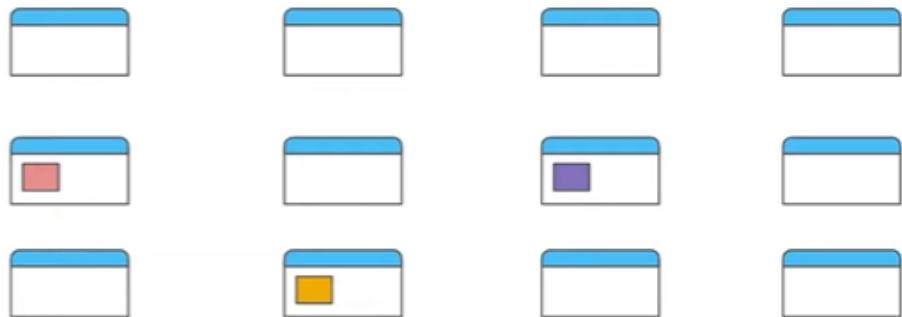


# Use case of data processing in GFS

Google Music analytics stored in GFS.

Write program to find while song was played how many times.

Client application



Analytics files stored on GFS

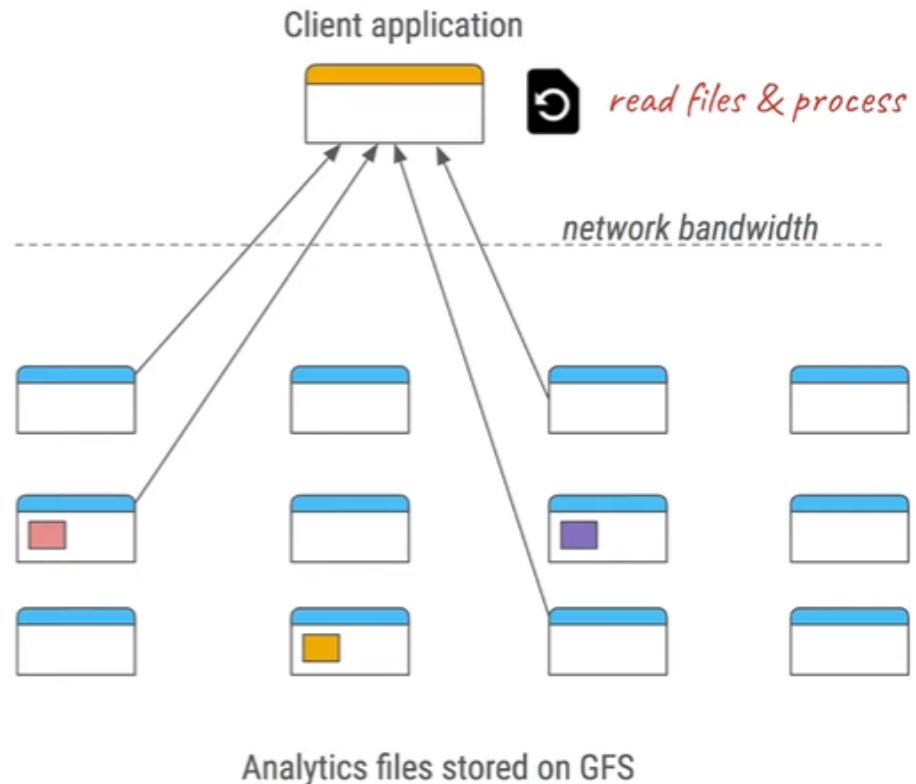
# Use case of data processing in GFS

Read files from GFS. Process the files.



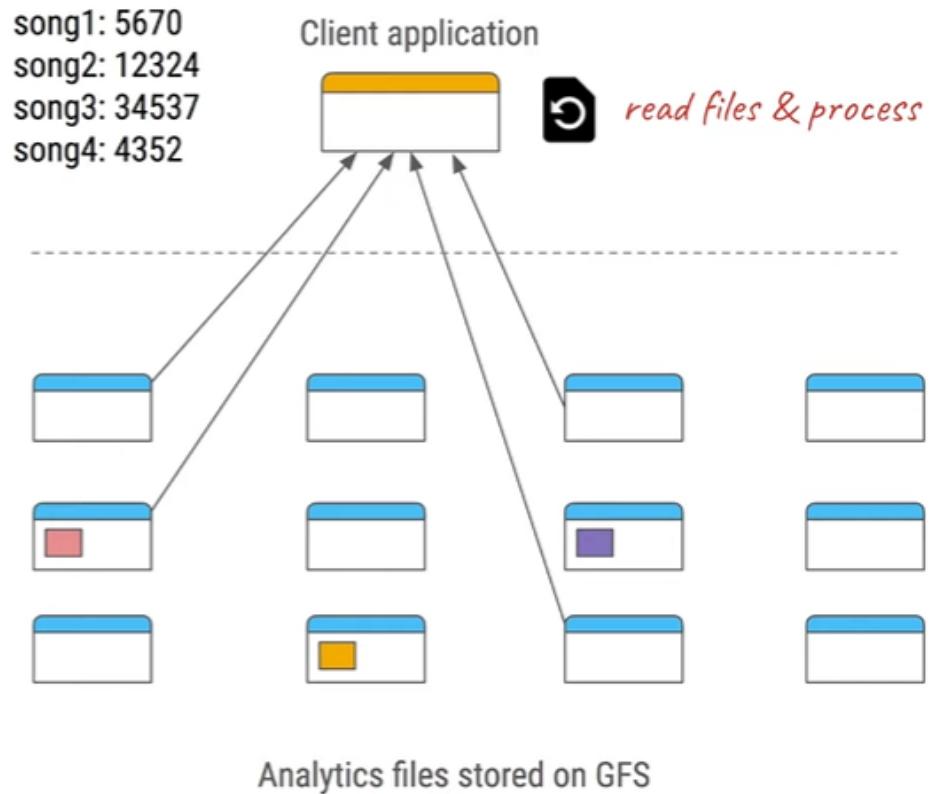
# Use case of data processing in GFS

- High network bandwidth
- Multi-TB file(s)
- Slow to process



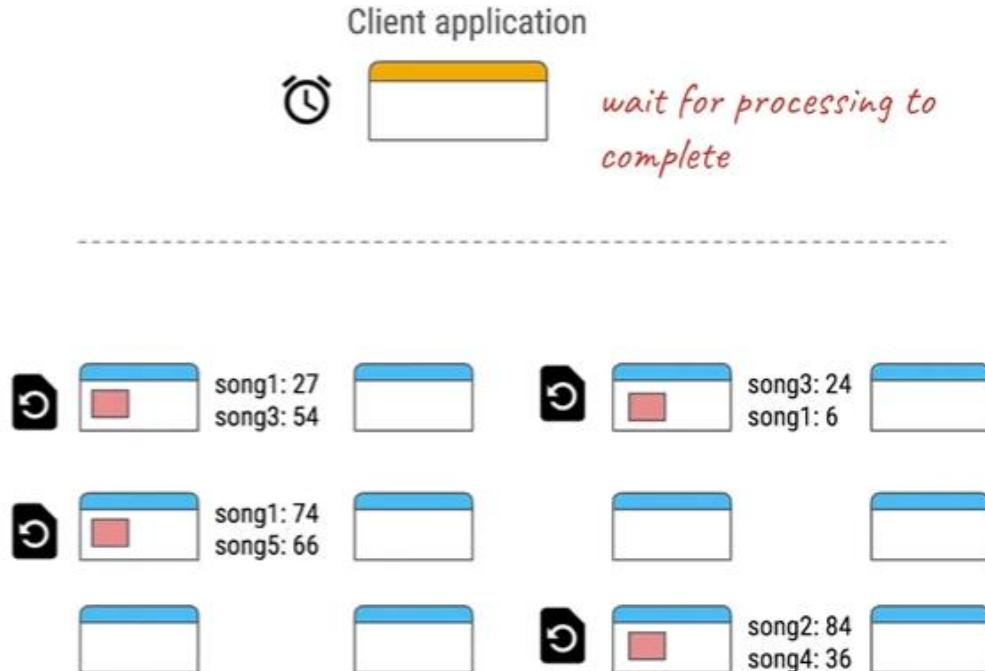
# Use case of data processing in GFS

Instead of bringing data to client, bring the processing function to the data.



# Use case of data processing in GFS

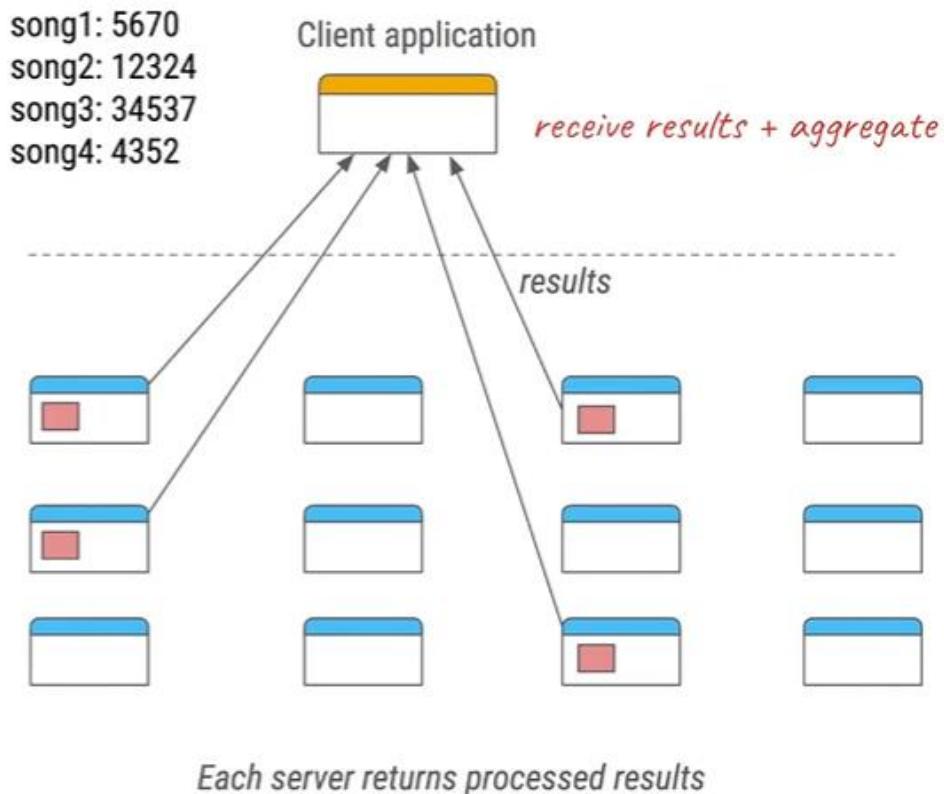
Each server applies function to the chunk it has.



*Each server processes part of the data*

# Use case of data processing in GFS

Once each server completes processing it can send result back to the application.



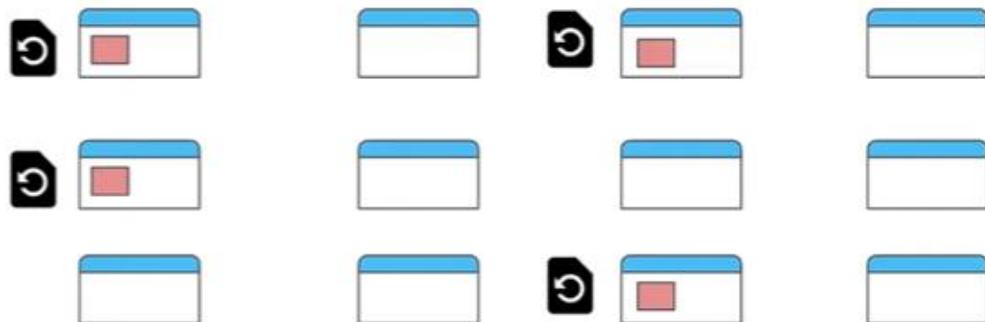
# Map Reduce in GFS

Programming model for distributed big data processing.

Client application



*only sends the functions to apply*



*Storage servers do processing on part of the data.*

# Map + Reduce

Map = Processing part of data

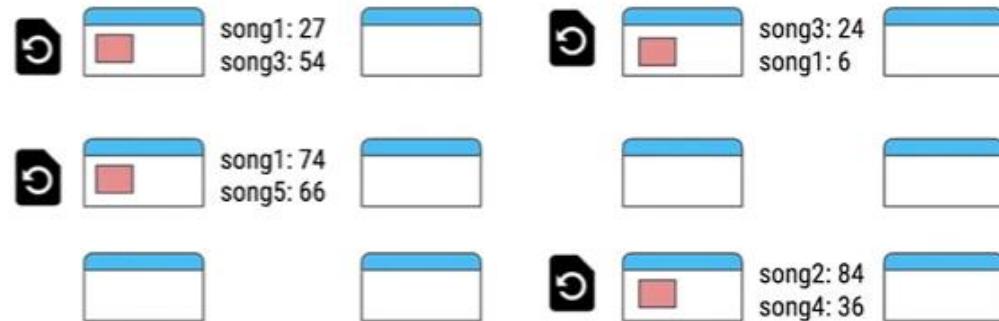
Reduce = Aggregation

Client application



*Reduce*

song1: 5670  
song2: 12324  
song3: 34537  
song4: 4352



*Map*

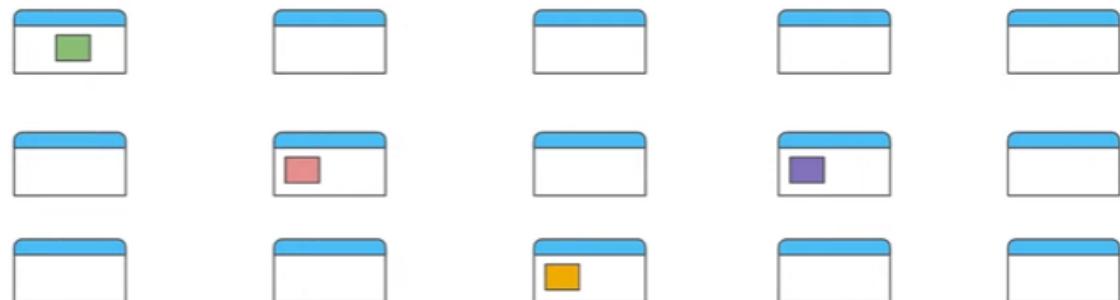
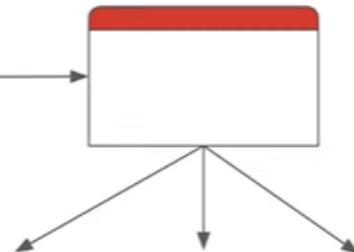
# Map + Reduce

Client application



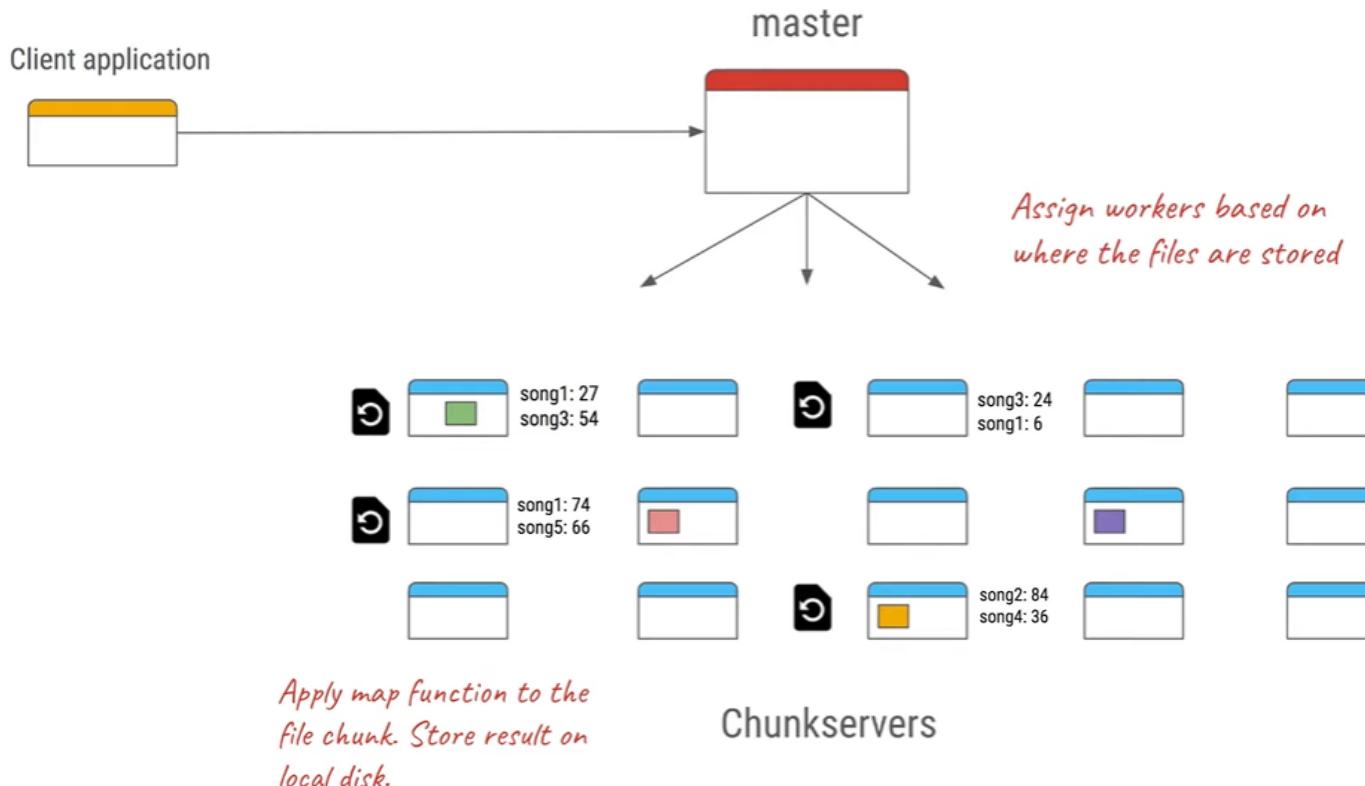
*tell master it wants to run  
a map reduce job*

master

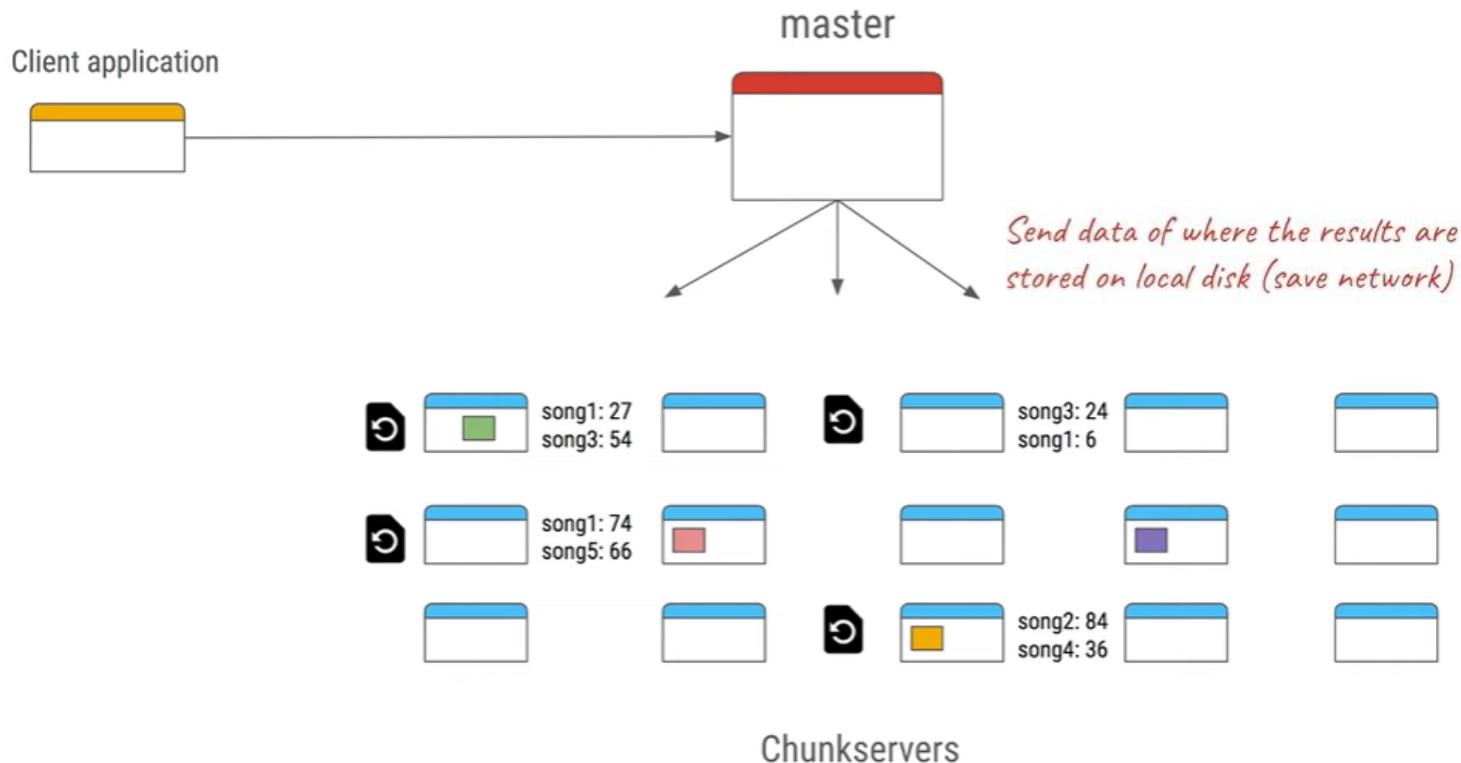


Chuckservers

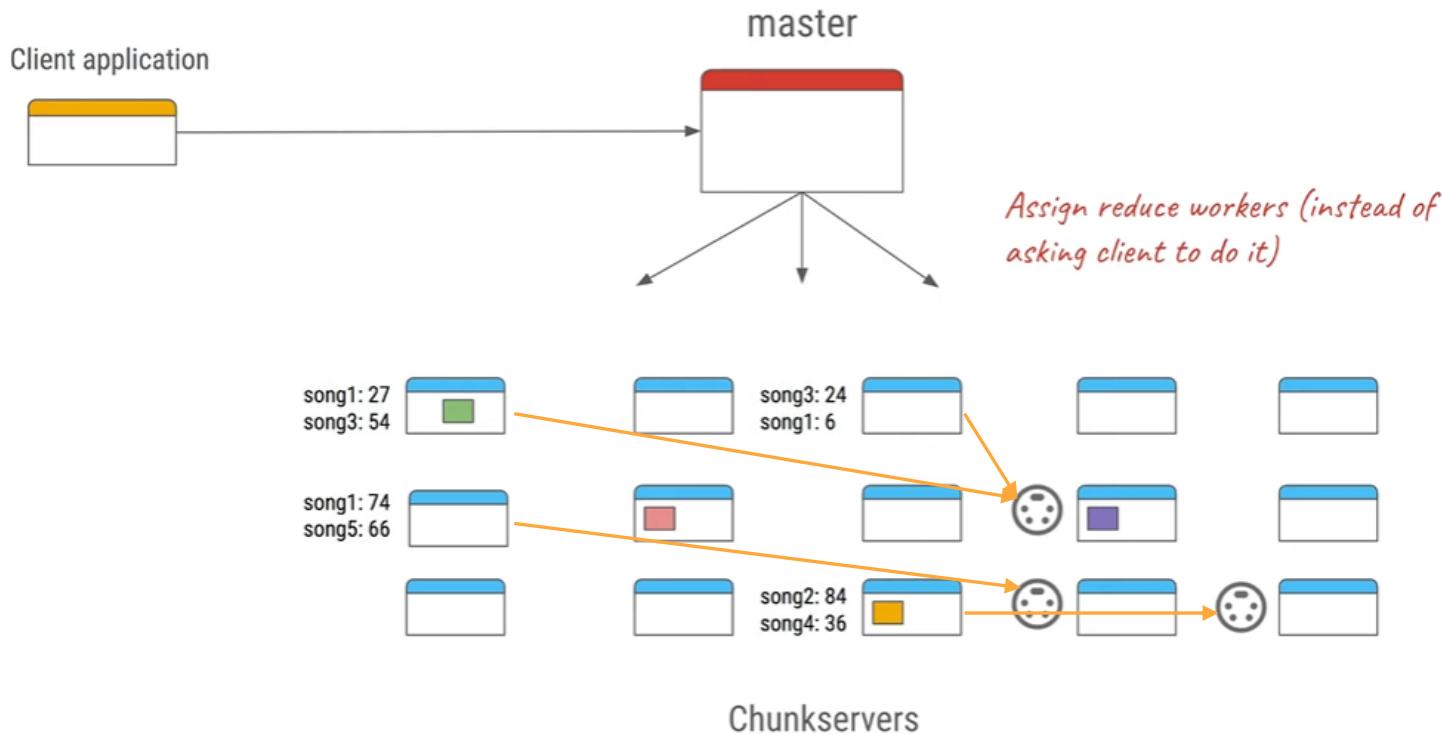
# Map + Reduce



# Map + Reduce



# Map + Reduce



# Hadoop

---

Hadoop architecture



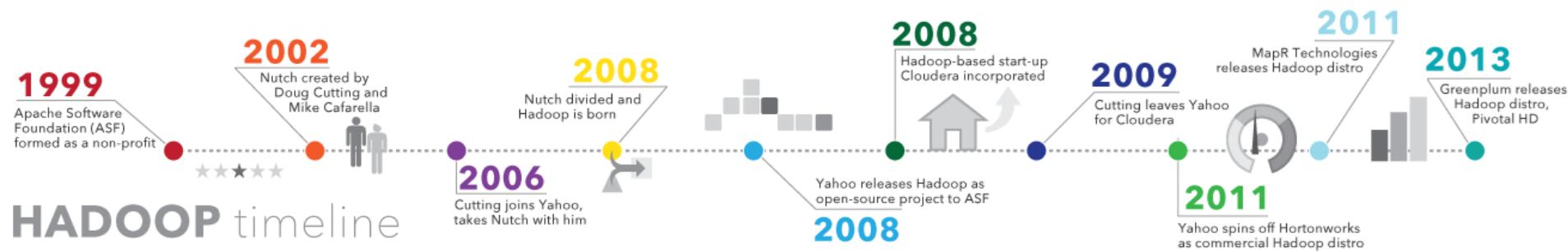
# Introduction

- Hadoop is a software framework written in **Java** for distributed processing of **large datasets** (terabytes or petabytes of data) across **large clusters** (thousands of nodes) of computers. Included some key components as below:
  - **Hadoop Common**: common utilities
  - **Hadoop Distributed File System (HDFS)** (Storage Component): A distributed file system that provides high-throughput access
  - **Hadoop YARN** (Scheduling): a framework for job scheduling & cluster resource management (available from **Hadoop 2.x**)
  - **Hadoop MapReduce** (Processing): A yarn-based system for parallel processing of large data sets



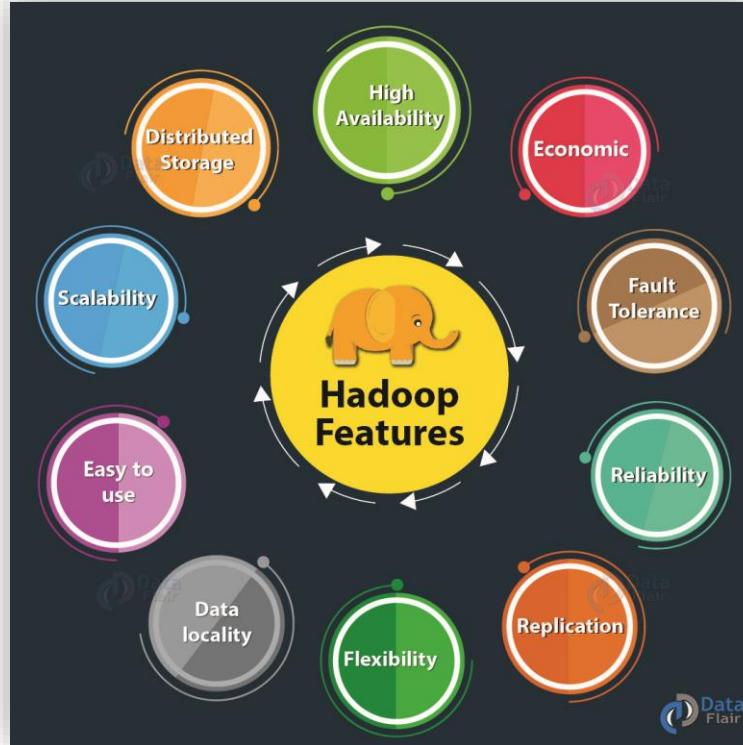
# Introduction

- Hadoop is a large and active ecosystem.
- Hadoop emerged as a solution for big data problems.
- Open source under the friendly Apache License
- Originally built as a Infrastructure for the “Nutch” project.
- Based on Google’s mapreduce and google File System.





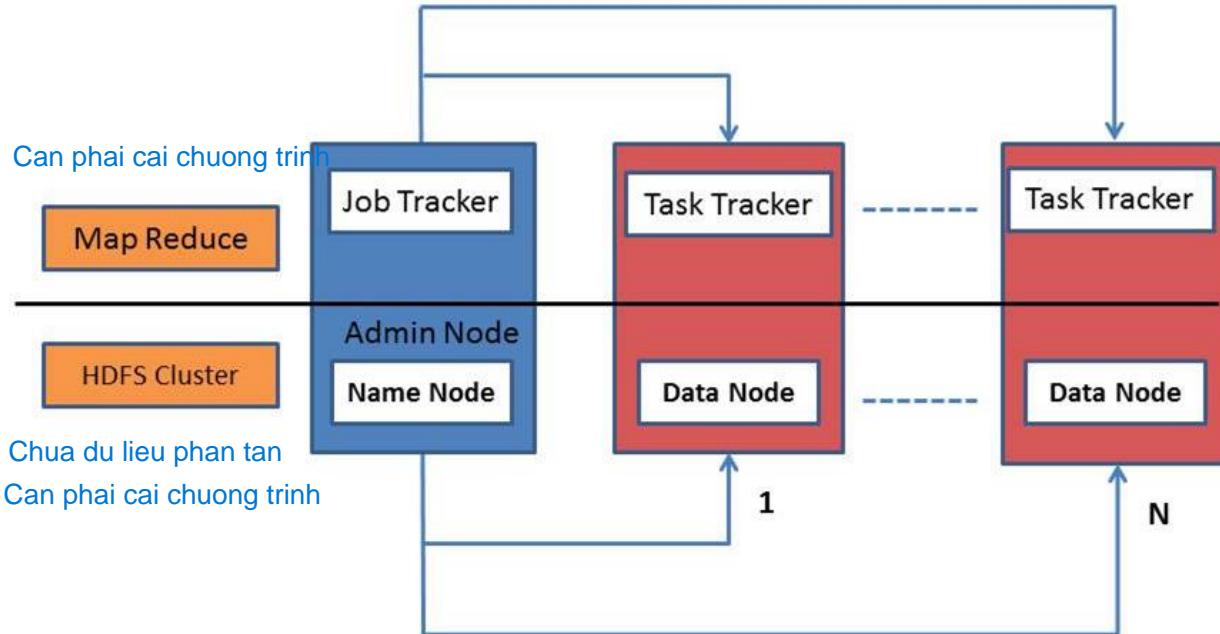
# Features





# Hadoop 1.x architecture

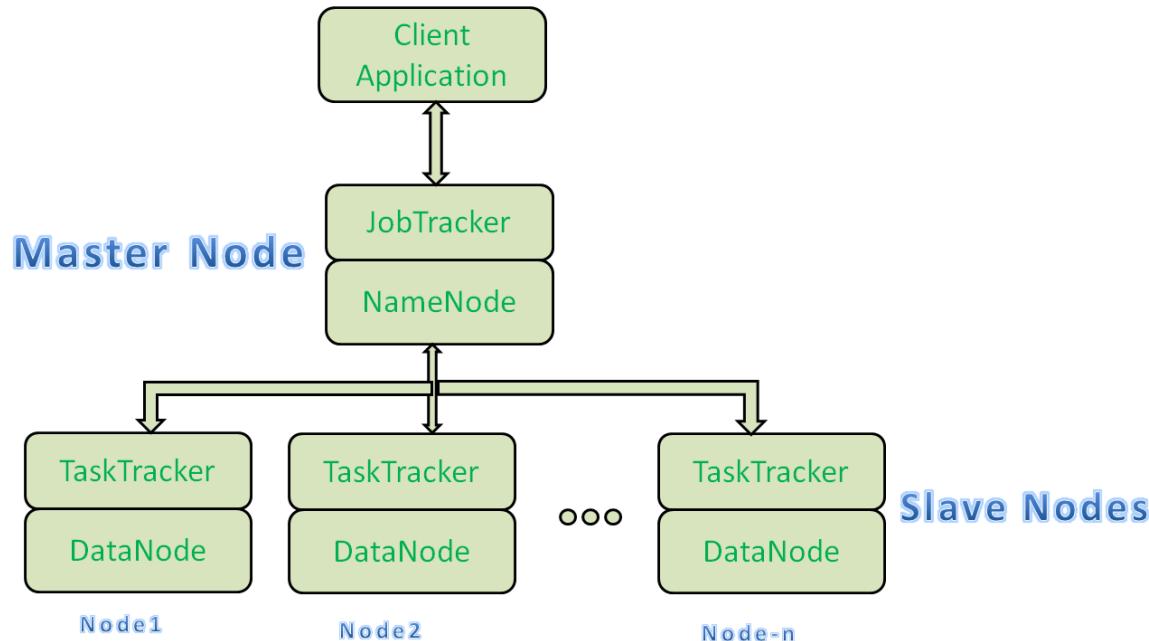
## Core components





# Hadoop 1.x architecture

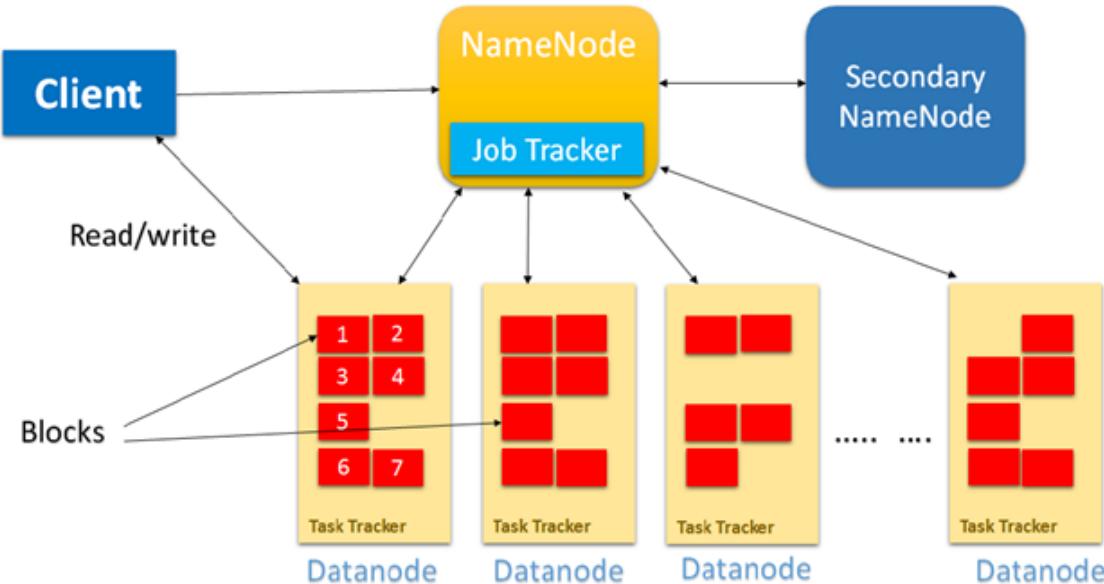
Core components





# Hadoop 1.x architecture

## Core components



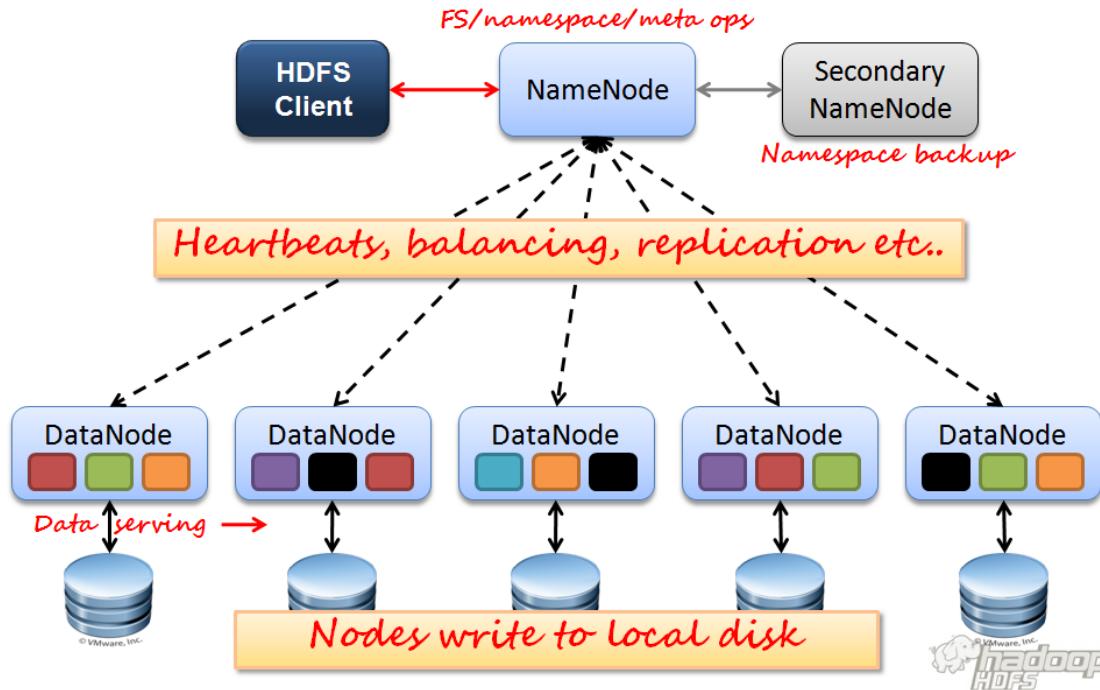
5 Hadoop daemons:

- ▶ Namenode
- ▶ Secondary namenode
- ▶ Jobtracker
- ▶ Datanode
- ▶ Tasktracker



# Hadoop 1.x architecture

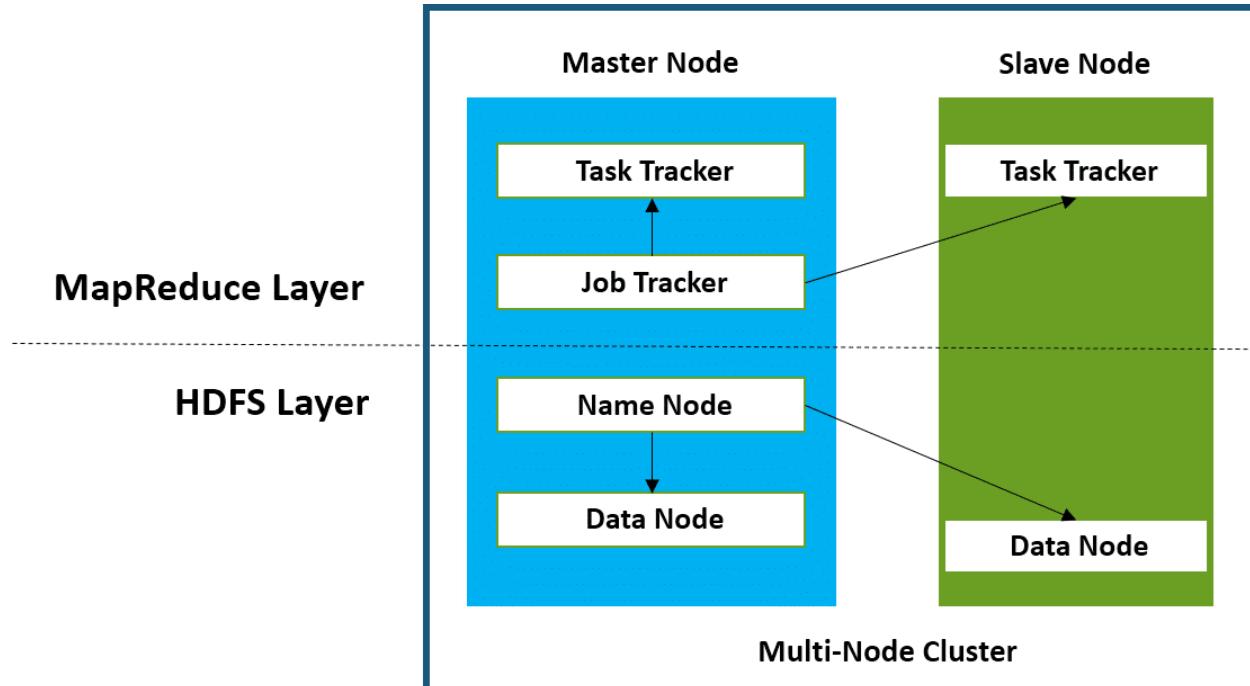
## Core components





# Architecture

## Multi-Node Cluster





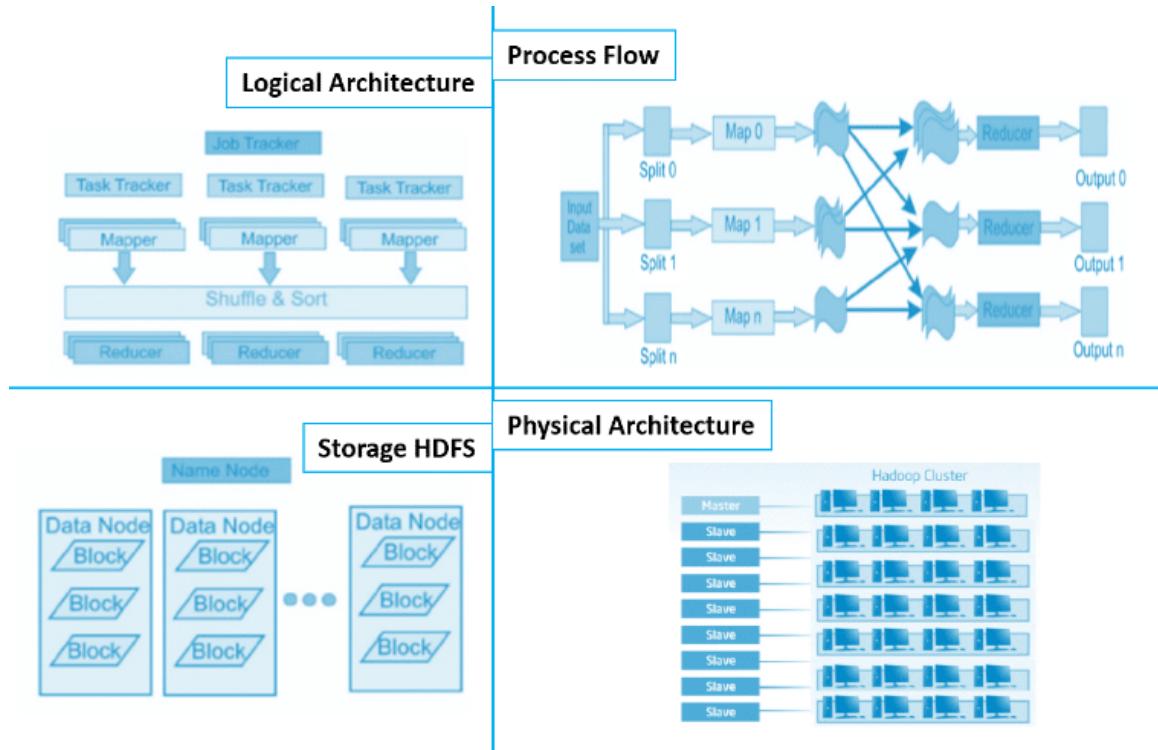
# What makes Hadoop unique

- **Data locality and Shared Nothing:** Moving computation to data, instead of moving data to computation. Each node can independently process a much smaller subset of the entire dataset without needing to communicate with one another.
- **Simplified programming model:** allows user to quickly write and test
- **Schema-on-read system** (same as NoSQL platforms) # **Schema-on-write system**
- Automatic distribution of data and work across machines



# Architecture

Architecture in different perspective





# HDFS

- **Hadoop Distributed File System (HDFS)** is designed to reliably store very large files across machines in a large cluster. It is inspired by the Google File System.
- Designed to reliably store data on **commodity hardware** (crash all the time)
- Intended for Large files and Batch inserts
- Distribute large data file into **blocks**
- Each block is replicated on multiple (slave) nodes
- HDFS component is divided into two sub-components: Name node and Data node



# HDFS

- **NameNode:**

- Master of the system, daemon runs on the **master machine**
- Maintains, monitoring and manages the **blocks** which are present on the **DataNodes**
- records the metadata of the files like the location of blocks, file size, permission, hierarchy etc.
- captures all the changes to the metadata like deletion, creation and renaming of the file in edit logs.
- It regularly receives **heartbeat** and block reports from the DataNodes.



# HDFS

- All of the Hadoop server processes (daemons) serve a web UI. For NameNode, it was on port 50070.

The screenshot shows a web browser window displaying the HDFS NameNode information at `localhost:50070/dfshealth.html`. The title bar says "All Applications" and "Namenode information". The main content area is titled "Overview 'localhost:9000' (active)". It contains a table with the following data:

<b>Started:</b>	Sun Apr 06 15:52:11 IST 2014
<b>Version:</b>	2.3.0, r1567123
<b>Compiled:</b>	2014-02-11T13:40Z by jenkins from branch-2.3.0
<b>Cluster ID:</b>	CID-5edbd0da-c69f-425b-bbc7-a662ac5d45dc
<b>Block Pool ID:</b>	BP-1127675761-127.0.1.1-1396692597591

Below the table is a "Summary" section with the following status information:

- Security is off.
- Safemode is off.
- 35 files and directories, 17 blocks = 52 total filesystem object(s).
- Heap Memory used 34.01 MB of 88.5 MB Heap Memory. Max Heap Memory is 889 MB.
- Non Heap Memory used 40.17 MB of 40.69 MB Committed Non Heap Memory. Max Non Heap Memory is -1 B.

At the bottom, there is a table showing capacity information:

<b>Configured Capacity:</b>	91.54 GB
-----------------------------	----------

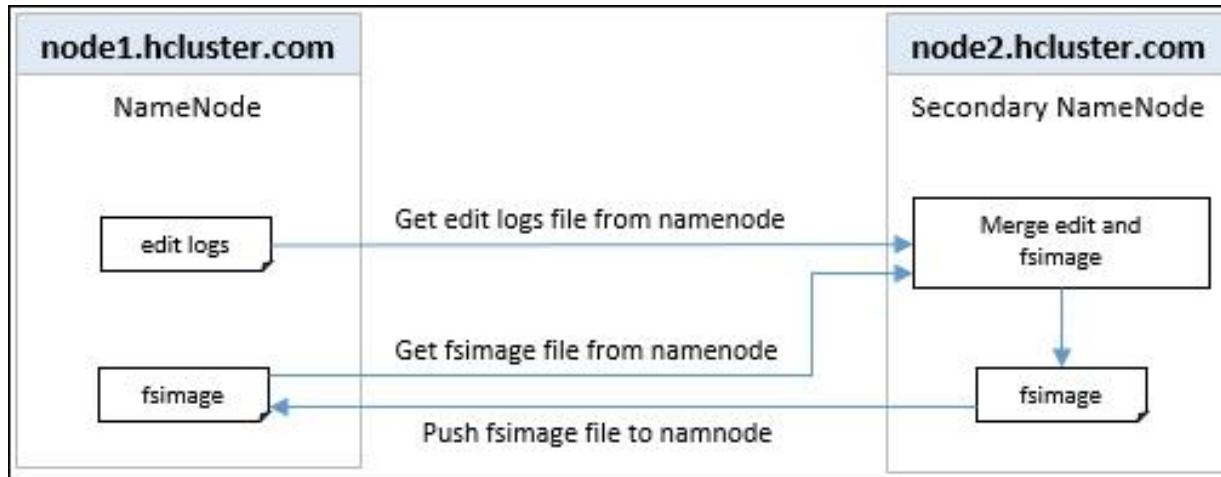
# HDFS

## Secondary namenode

- The secondary namenode daemon is responsible for performing periodic housekeeping functions for namenode.
- It creates checkpoints of the filesystem metadata (`fsimage`) present in namenode by merging the edits logfile and the `fsimage` file from the namenode daemon.
- In case the namenode daemon fails, this checkpoint could be used to rebuild the filesystem metadata.
- Checkpoints are done in intervals, thus checkpoint **data could be slightly outdated**. Rebuilding the `fsimage` file using such a checkpoint **could lead to data loss**.
- It is recommended that the secondary namenode daemon be hosted on a separate machine for large clusters.
- The checkpoints are created by merging the edits logfiles and the `fsimage` file from the namenode daemon.

# HDFS

## Secondary namenode



neu co gi do vua thay doi trong file ma ko luu tru trong fsimage -->Mat noi dung do



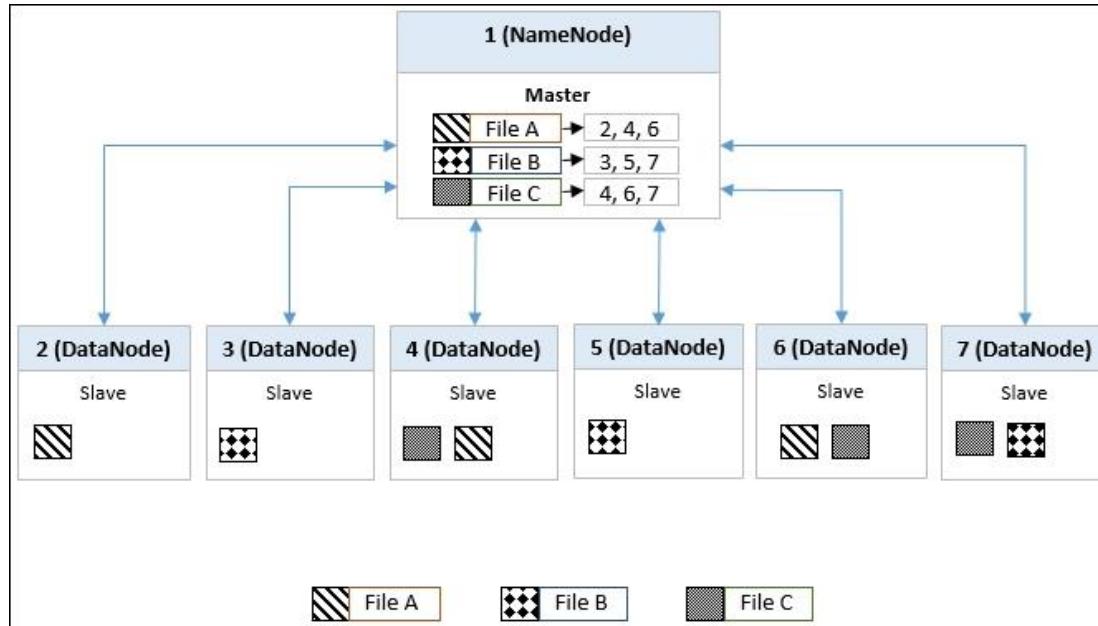
# HDFS

- **DataNode:**

- DataNode runs on the **slave machine**.
- It stores the actual business data.
- It serves the read-write request from the user.
- DataNode does the ground work of creating, replicating and deleting the blocks on the command of NameNode.
- After every **3 seconds**, by default, it sends **heartbeat** to NameNode reporting the health of HDFS.

# HDFS

## Data node



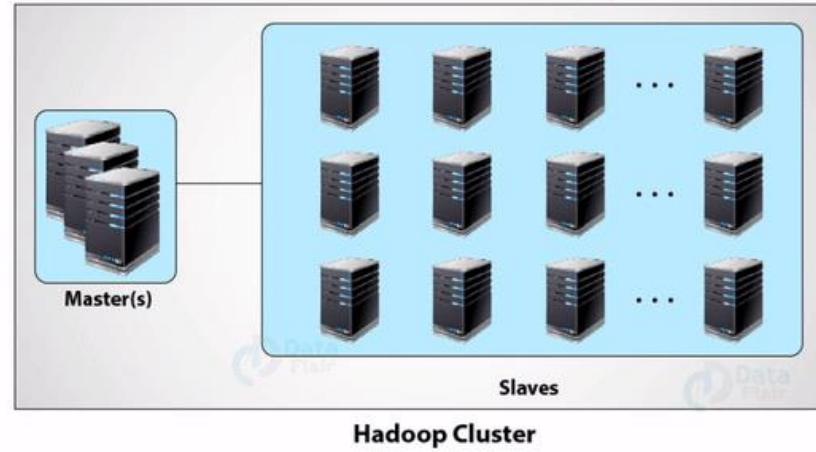


# HDFS

## Architecture



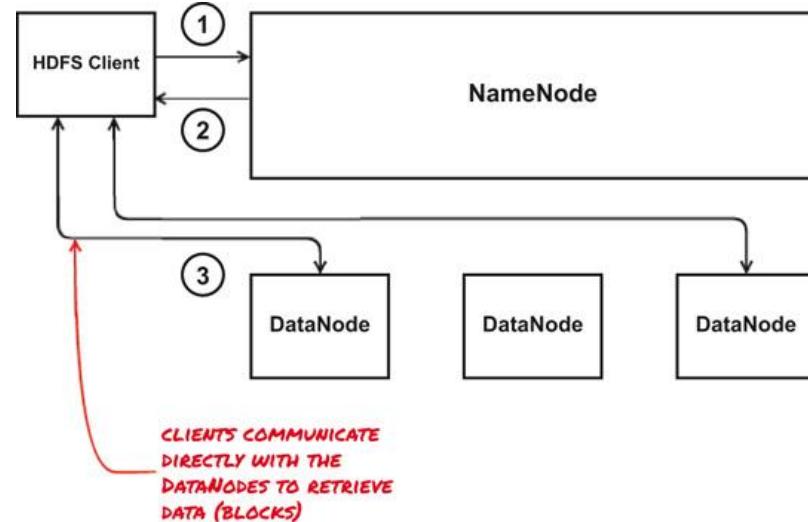
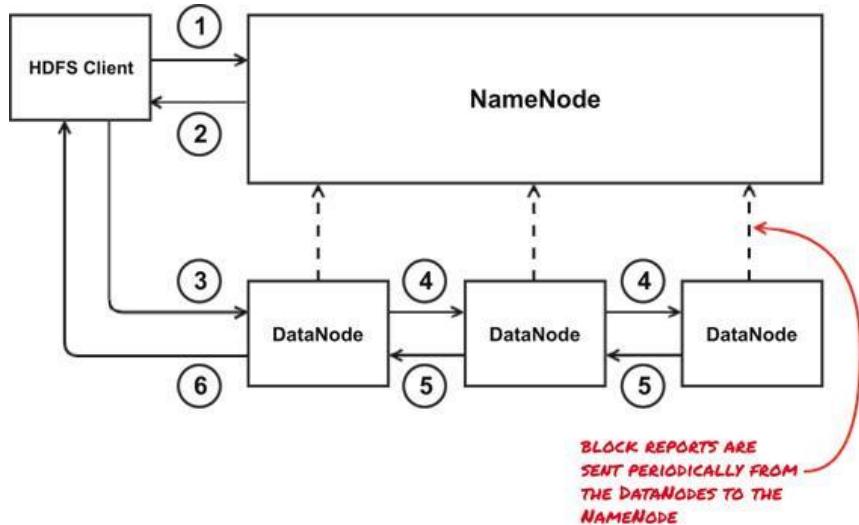
### Data Storage in HDFS





# HDFS

## Write & Read files





# Map Reduce

- **Programming model for distributed computations** at a massive scale
- Execution framework for organizing and performing such computations
- **Data locality** is king
- MapReduce component is again divided into two sub-components:  
JobTracker and TaskTracker
- **JobTracker**: takes care of all the job scheduling and assign tasks to Task Trackers.
- **TaskTracker**: a node in the cluster that accepts tasks - **Map, Reduce & Shuffle** operations - from jobtracker

# Map Reduce

## JobTracker

- The **jobtracker** daemon is responsible for accepting job requests from a client and scheduling/assigning tasktrackers with tasks to be performed.
- The jobtracker daemon tries to assign tasks to the tasktracker daemon on the datanode daemon where the data to be processed is stored. This feature is called **data locality**.
- If that is not possible, it will at least try to assign tasks to tasktrackers within the same physical server rack.
- If for some reason the node hosting the datanode and tasktracker daemons fails, the jobtracker daemon assigns the task to another tasktracker daemon where the replica of the data exists. This is possible because of the replication factor configuration for HDFS where the data blocks are replicated across multiple datanodes. This ensures that the job does not fail even if a node fails within the cluster.

# Map Reduce

## TaskTracker

- The tasktracker daemon is a daemon that accepts tasks (map, reduce, and shuffle) from the jobtracker daemon.
- The tasktracker daemon is the daemon that performs the actual tasks during a MapReduce operation.
- The tasktracker daemon sends a heartbeat message to jobtracker, periodically, to notify the jobtracker daemon that it is alive.
- Along with the heartbeat, it also sends the free slots available within it, to process tasks.
- The tasktracker daemon starts and monitors the map, and reduces tasks and sends progress/status information back to the jobtracker daemon.



# Map Reduce

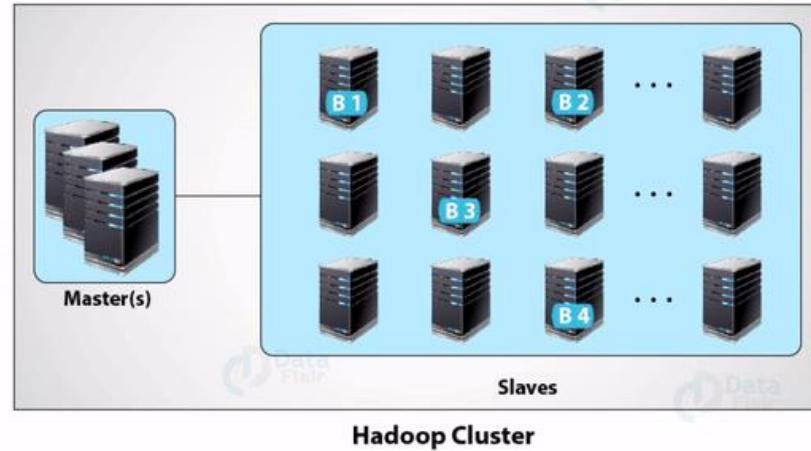
flow



## How MapReduce works



User





# Map Reduce

- **The Mapper:**

- Each block is processed in isolation by a map task called mapper
- Map task runs on the node where the block is stored
- Iterate over a large number of records
- Extract something of interest from each

- **Shuffle and sort intermediate results**

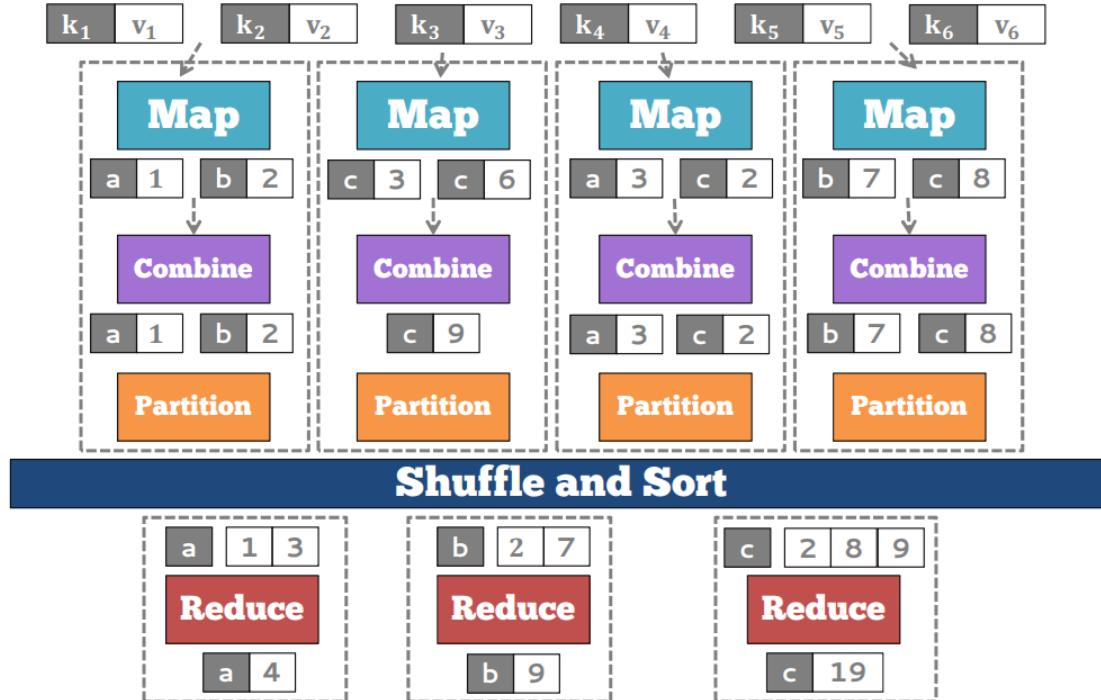
- **The Reducer:**

- Consolidate result from different mappers
- Aggregate intermediate results
- Produce final output



# Map Reduce

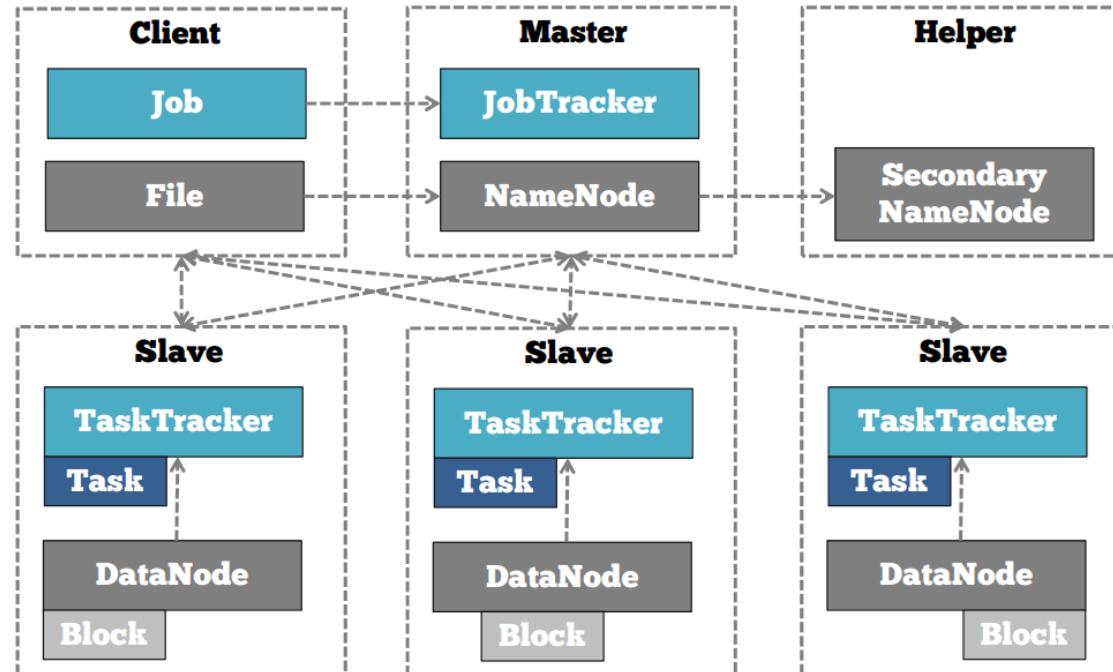
flow





# Map Reduce

Combined Hadoop architecture





# Map Reduce

Good for ...

- Embarrassingly parallel algorithms
- Summing, grouping, filtering, joining
- Off-line batch jobs on massive data sets
- Analyzing an entire large dataset



# Map Reduce

Not good for ...

- Jobs that need shared state/coordination
  - Tasks are shared-nothing
  - Shared-state requires scalable state store
- Iterative jobs (i.e., graph algorithms)
  - Each iteration must read/write data to disk
  - I/O and latency cost of an iteration is high
- Low-latency jobs
- Jobs on small datasets
- Finding individual records



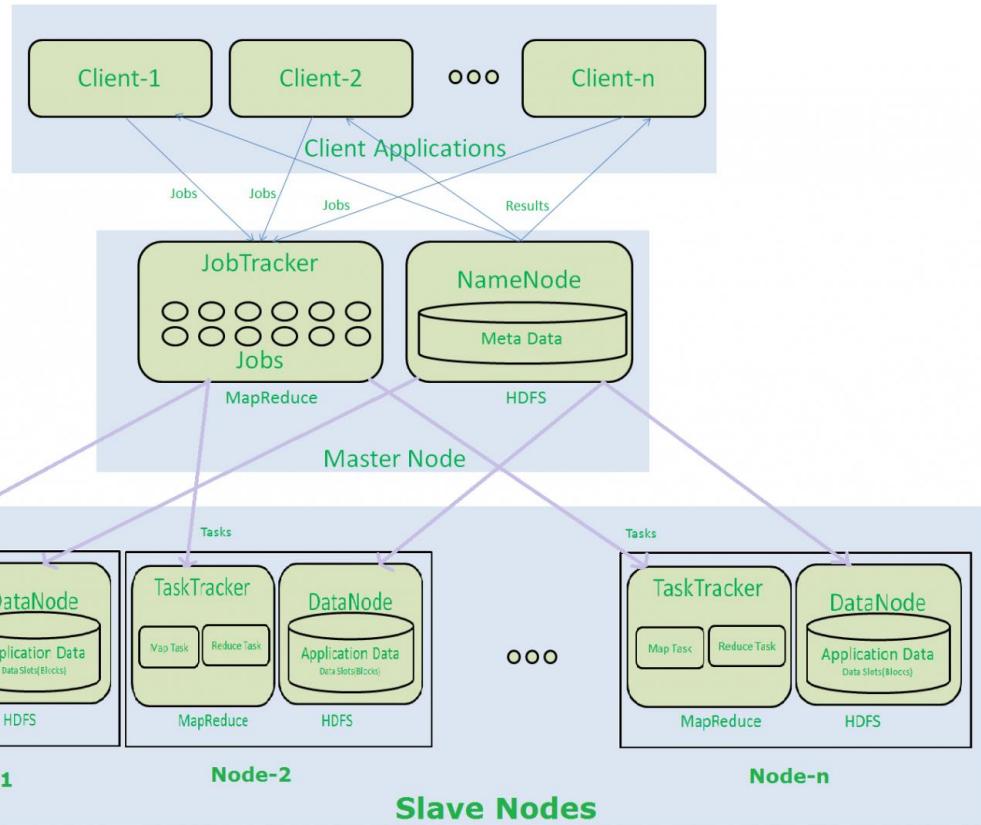
# Map Reduce

## limitations

- Scalability
  - Maximum cluster size ~ 4,500 nodes, concurrent tasks – 40,000
  - Coarse synchronization in JobTracker
- Availability
  - Failure kills all queued and running jobs
- Hard partition of resources into map & reduce slots
  - Low resource utilization
- Lacks support for alternate paradigms and services
  - Iterative applications implemented using MapReduce are 10x slower

# Hadoop 1.x

In small clusters, the namenode and jobtracker daemons reside on the same node. However, in larger clusters, there are dedicated nodes for the namenode and jobtracker daemons.

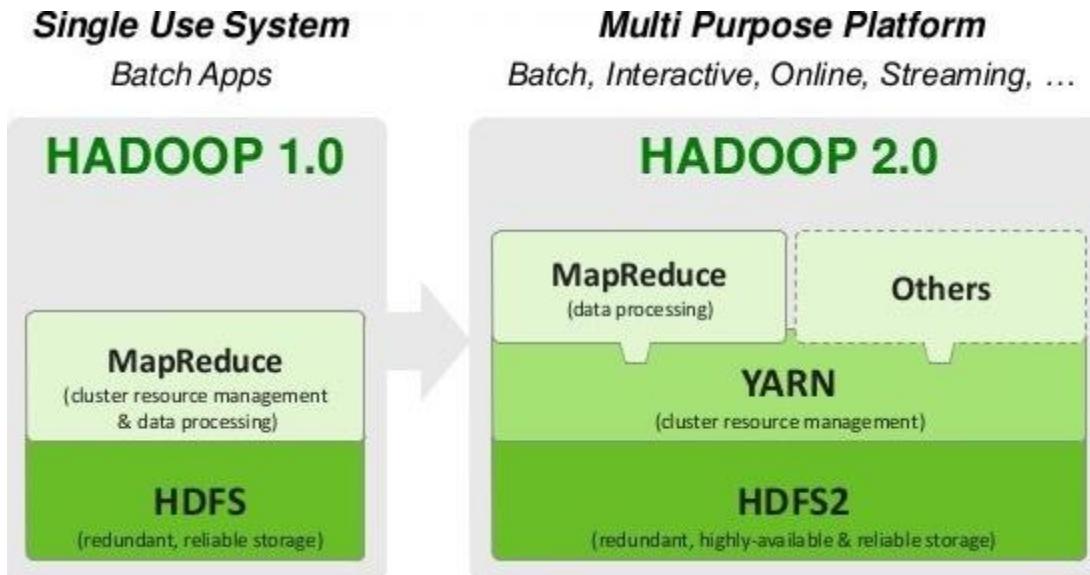


# Hadoop 1.x

By default, the ports for the Hadoop daemons are:

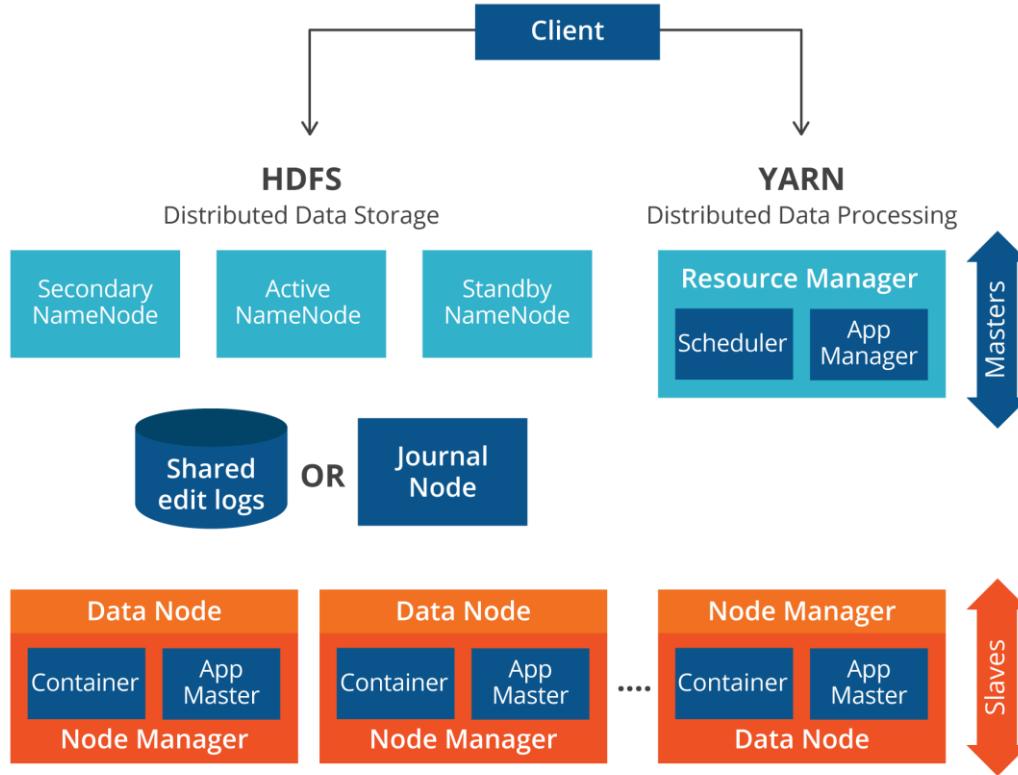
The Hadoop daemon	Port
Namenode	50070
Secondary namenode	50090
Jobtracker	50030
Datanode	50075
Tasktracker	50060

# Hadoop 2.x



HDFS2, YARN, MapReduce: Three Pillars of Hadoop 2

# Apache Hadoop 2.0 and YARN



# Hadoop 2.x

Following are the four main improvements in Hadoop 2.0 over Hadoop 1.x:

- **HDFS Federation** – horizontal scalability of NameNode
- **NameNode High Availability** – NameNode is no longer a Single Point of Failure
- **YARN** – ability to process Terabytes and Petabytes of data available in HDFS using Non-MapReduce applications such as MPI, GIRAPH
- **Resource Manager** – splits up the two major functionalities of overburdened JobTracker (resource management and job scheduling/monitoring) into two separate daemons: a global Resource Manager and per-application ApplicationMaster

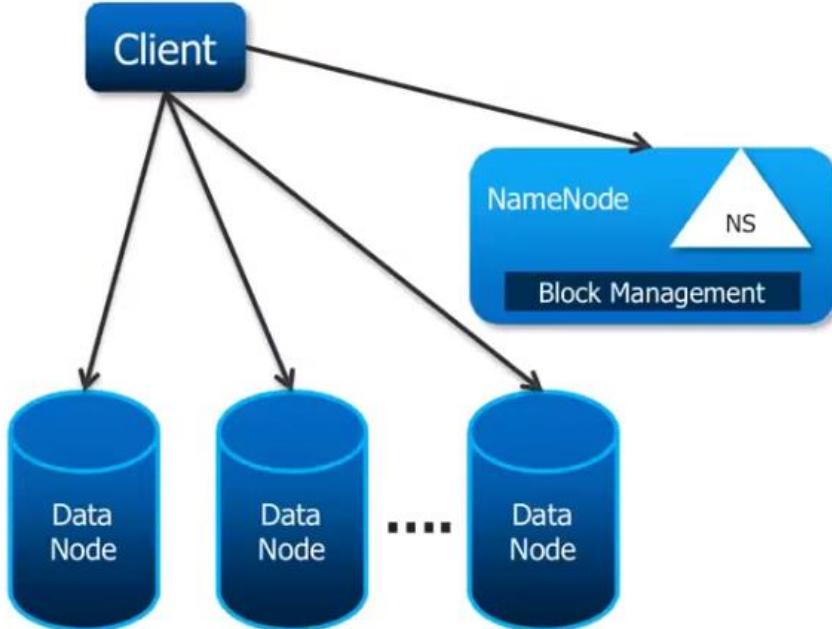
There are additional features such as **Capacity Scheduler** (Enable Multi-tenancy support in Hadoop), **Data Snapshot**, **Support for Windows**, **NFS access**, enabling increased Hadoop adoption in the Industry to solve Big Data problems.

# Limitation of Hadoop 1.x

- No horizontal scalability of NameNode
- Does not support NameNode High Availability
- Overburdened JobTracker
- Not possible to run Non-MapReduce Big Data Applications on HDFS
- Does not support Multi-tenancy

# Limitation of Hadoop 1.x

No Horizontal Scalability of NameNode



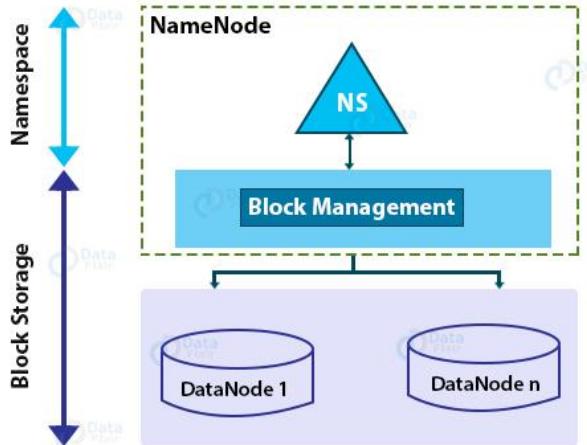
## Challenges:

- Meta is stored in NameNode memory
- Bottleneck after ~4000 nodes
- Results in cascading failures

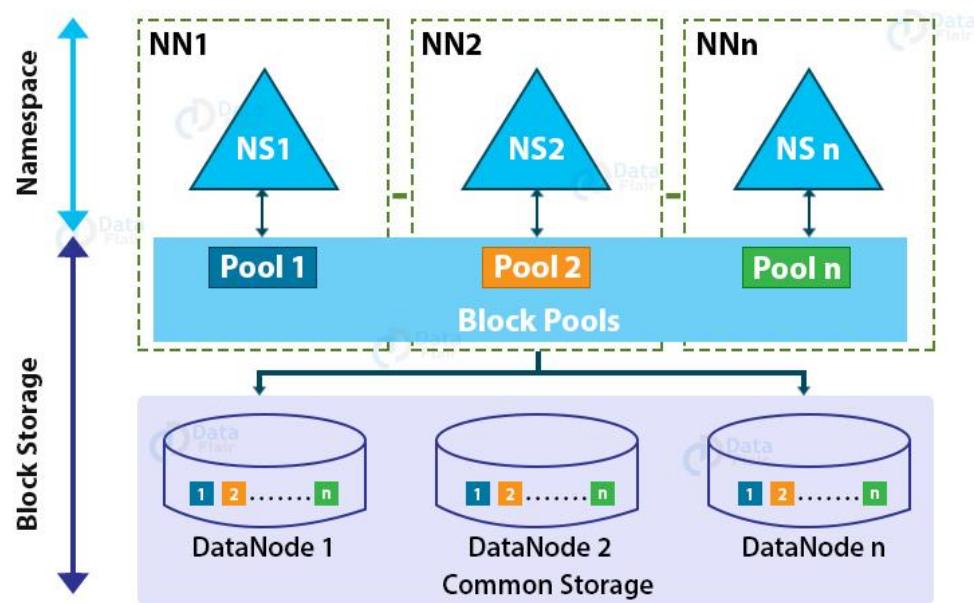
# Hadoop 2.x

## HDFS Federation

### HDFS Layers

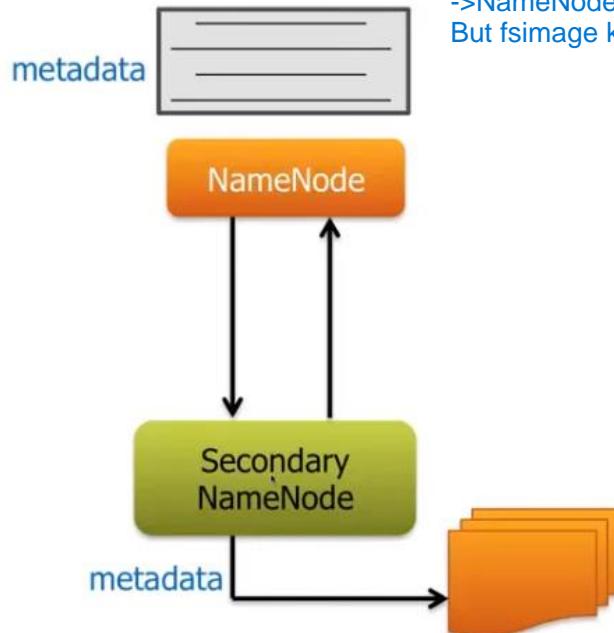


### HDFS Federation Architecture



# Limitation of Hadoop 1.x

## NameNode – Single point of failure



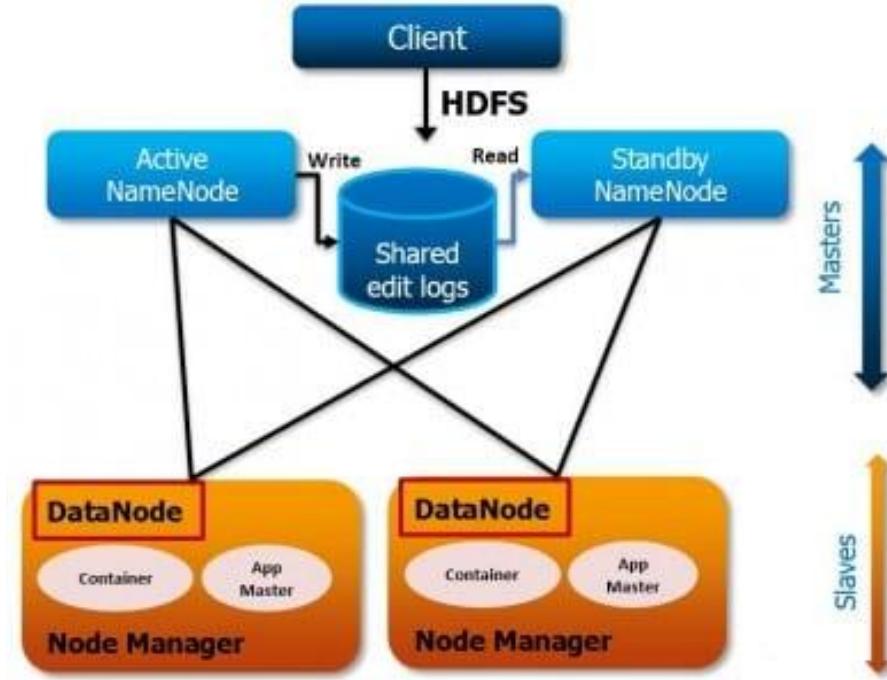
Toan bo meta data luu tru trong NameNode  
->NameNode chet thi toan bo metadata mat  
But fsimage ko cap nhat

### Secondary NameNode:

- “Not a hot standby” for the NameNode
- Connects to NameNode regularly
- Housekeeping, backup of NameNode metadata
- Saved metadata can build a failed NameNode

# Hadoop 2.x

## NameNode High Availability



# Limitation of Hadoop 1.x

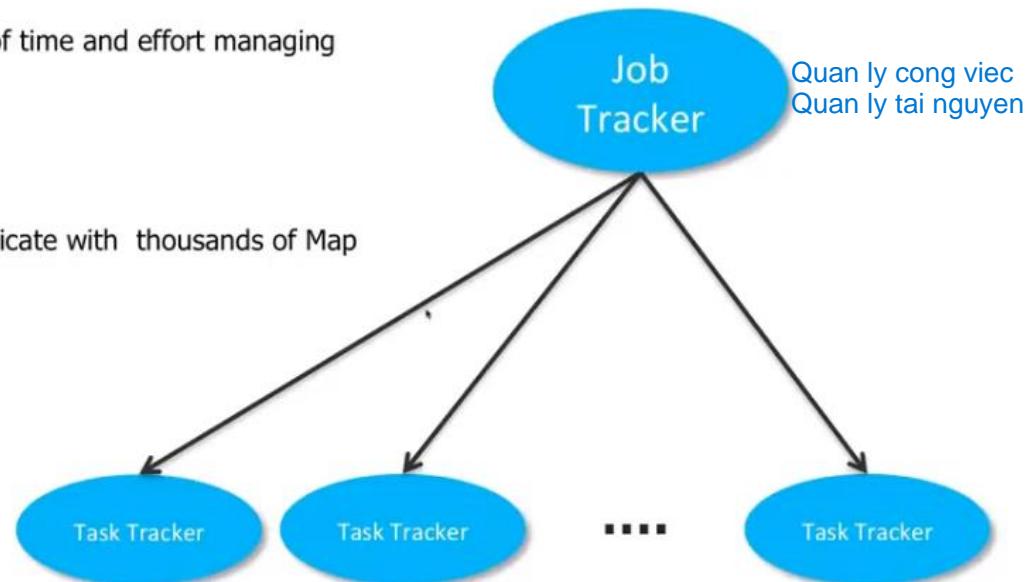
## Overburdened JobTracker

### CPU

- ✓ Spends a very significant portion of time and effort managing the life cycle of applications

### Network

- ✓ Single Listener Thread to communicate with thousands of Map and Reduce Jobs



# Hadoop 2.x

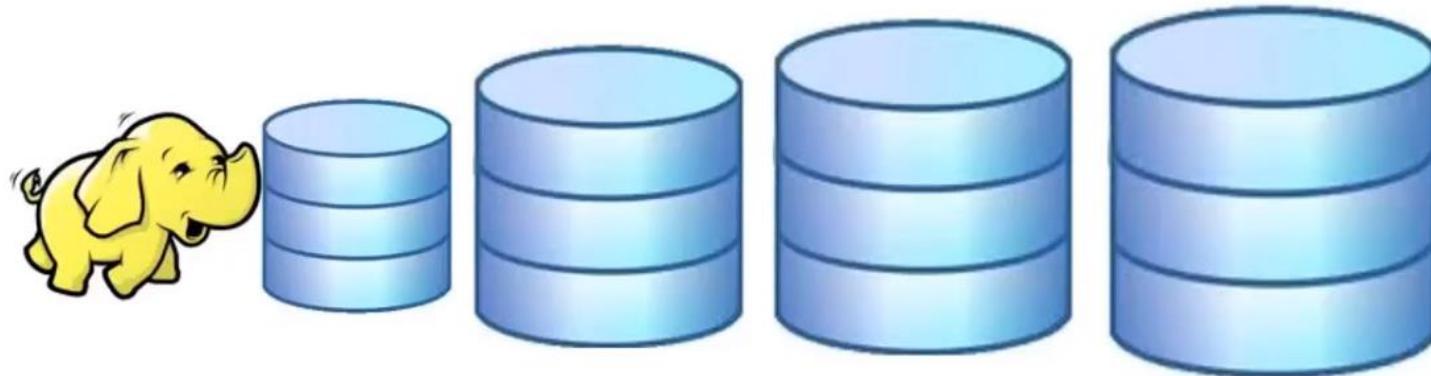
## YARN (Yet Another Resource Negotiator)

Mot tap hop tu nhieu program con



# Limitation of Hadoop 1.x

Unutilized data in HDFS

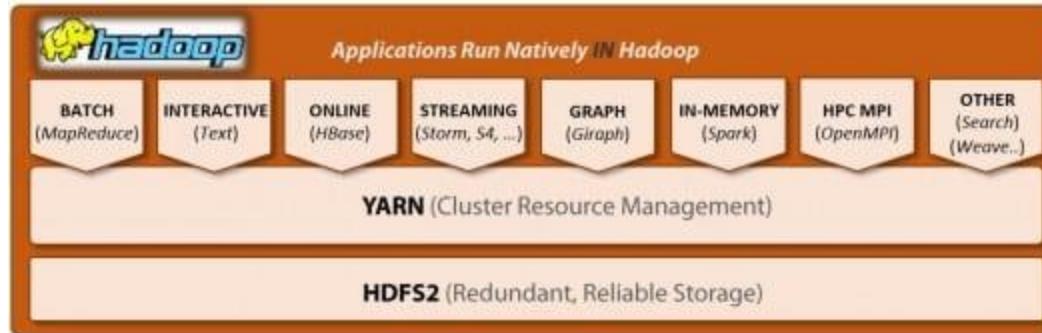


## Challenges:

- ✓ Only MapReduce processing can be achieved
- ✓ Alternate Data Storage is needed for other processing such as Real-time or Graph analysis
- ✓ Doesn't support Multi-Tenancy

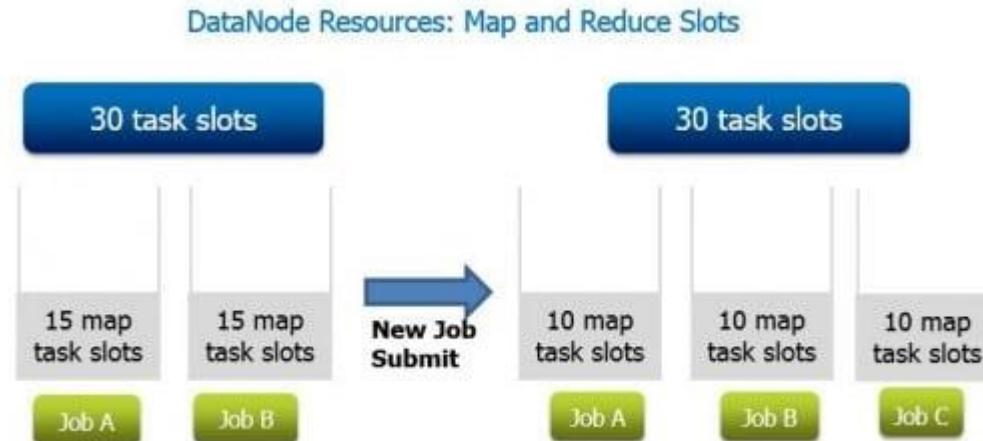
# Hadoop 2.x

## Non-MapReduce Big Data Application



# Limitation of Hadoop 1.x

No Multi-tenancy Support

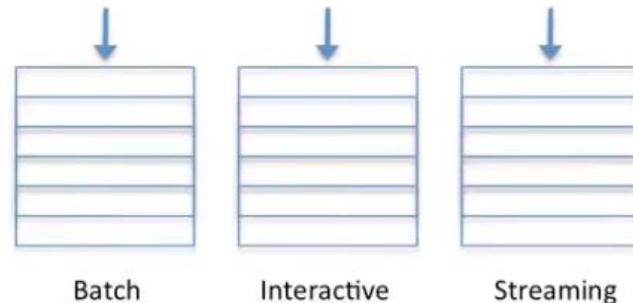


# Hadoop 2.x

## Capacity Scheduler – Multi-tenancy Support

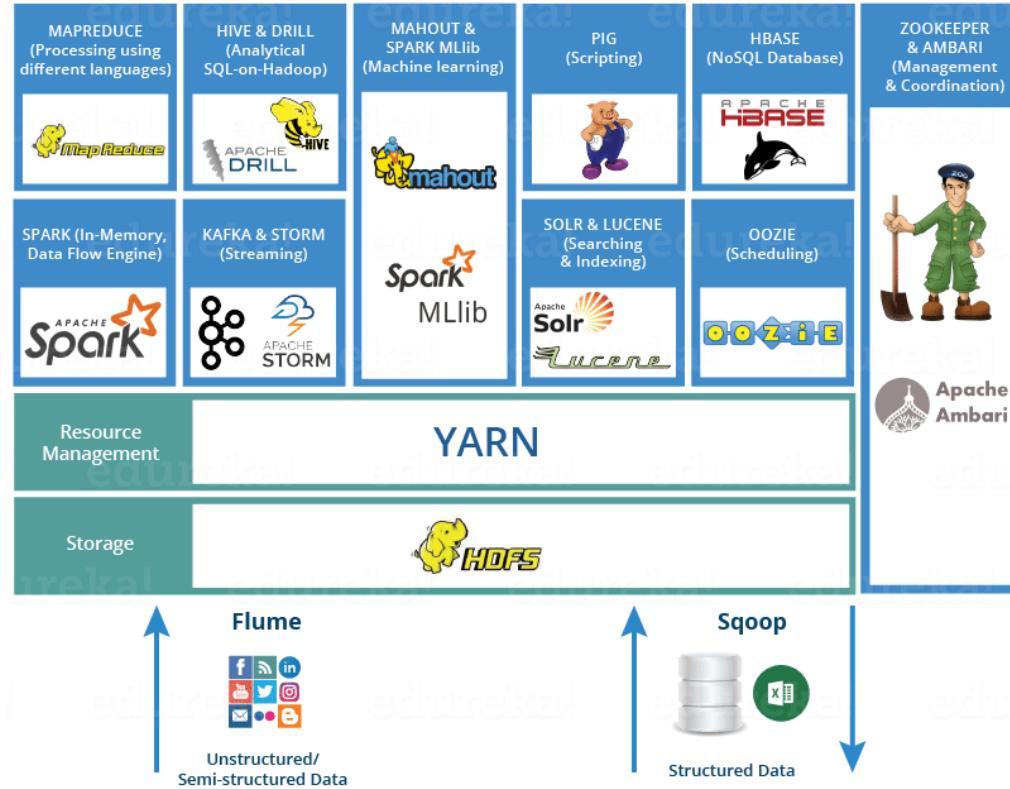
### Features:

- ✓ Different types of jobs are organized in different queues
- ✓ Queue shares as %'s of cluster
- ✓ Each queue has an associated priority
- ✓ FIFO scheduling within each queue
- ✓ Security ensured between applications





# Ecosystem

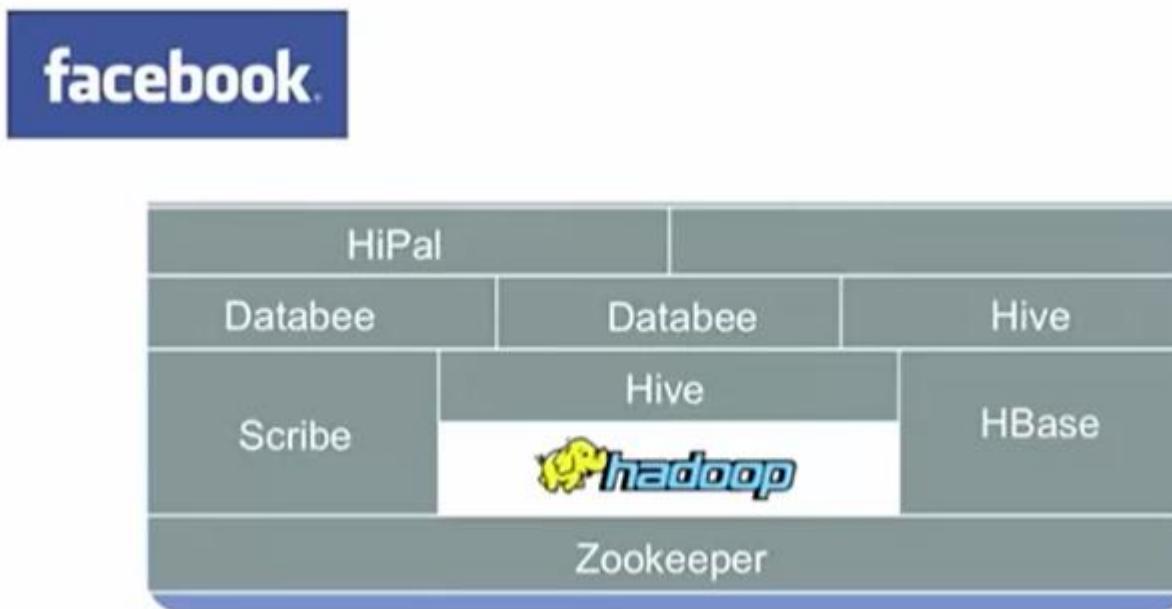


# Hadoop Ecosystem History

## The Google Stack



# Hadoop Ecosystem History



# Hadoop Ecosystem history

**YAHOO!**



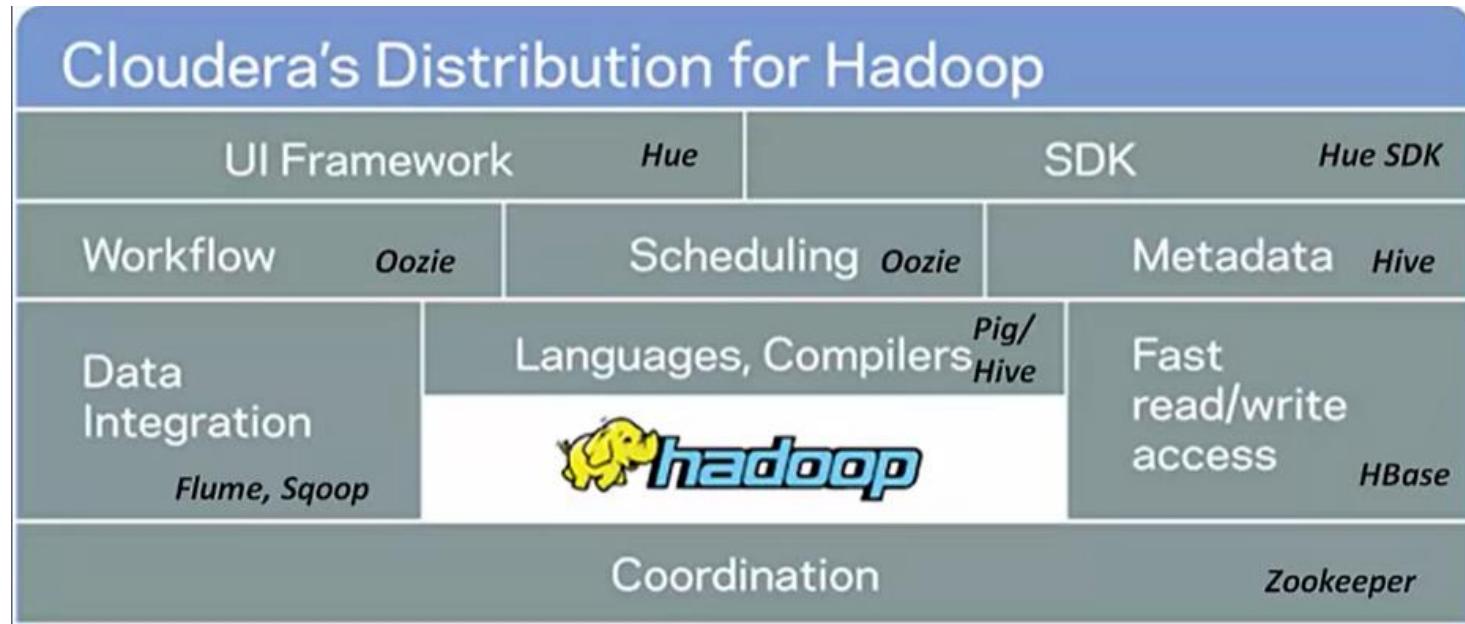
# Hadoop Ecosystem History

Linked



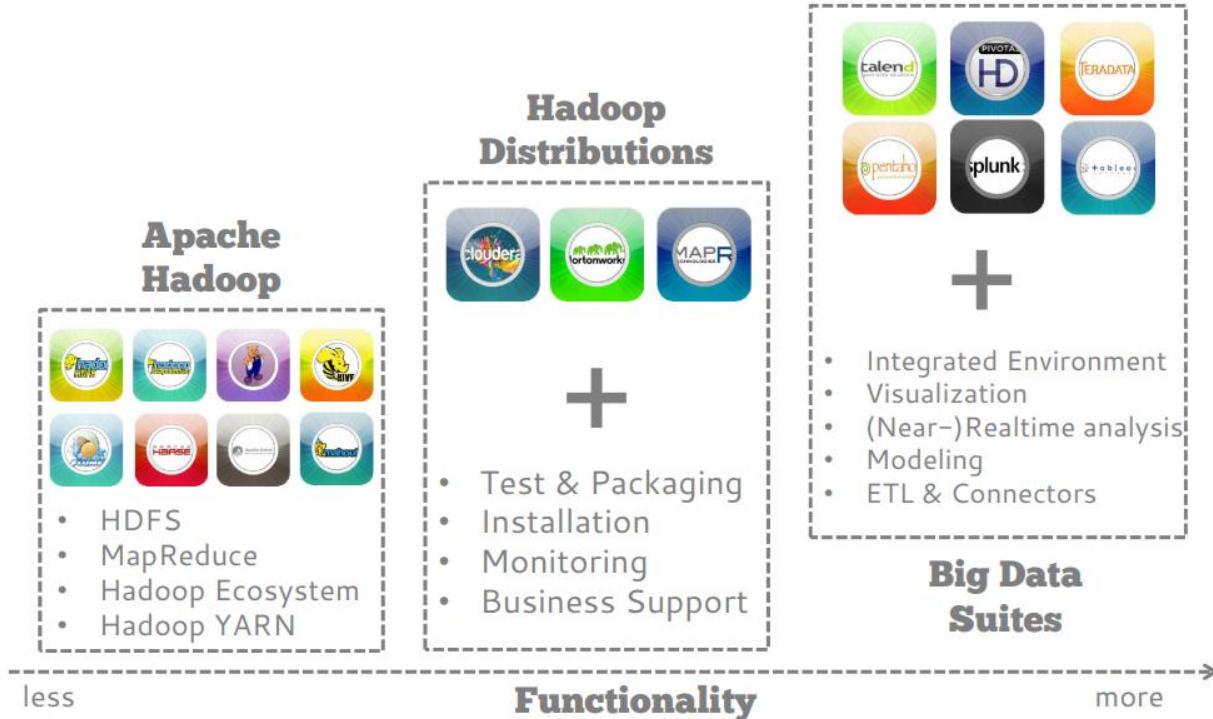
# Hadoop Ecosystem History

Bang phan phoi Hadoop





# Ecosystem

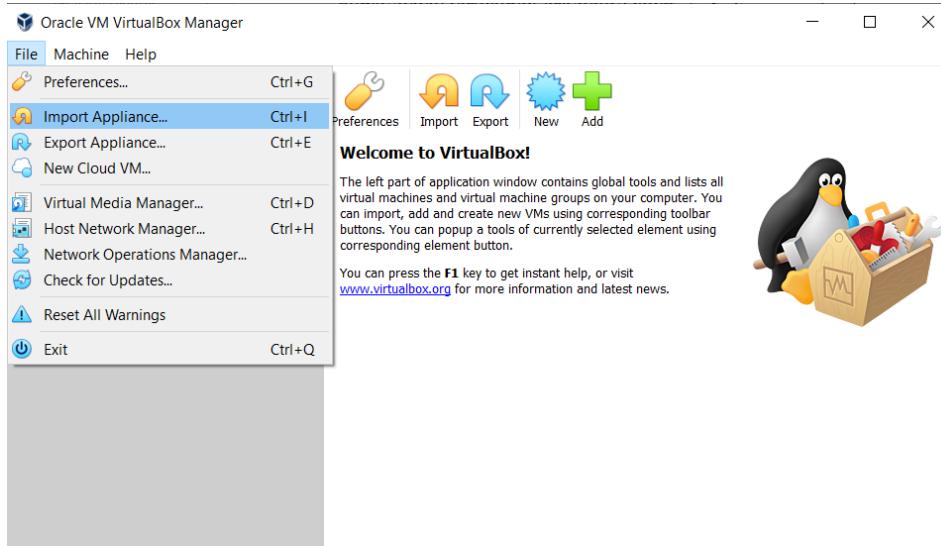


# Install Cloudera Quickstart VM

- Download Cloudera Quickstart VM for VirtualBox

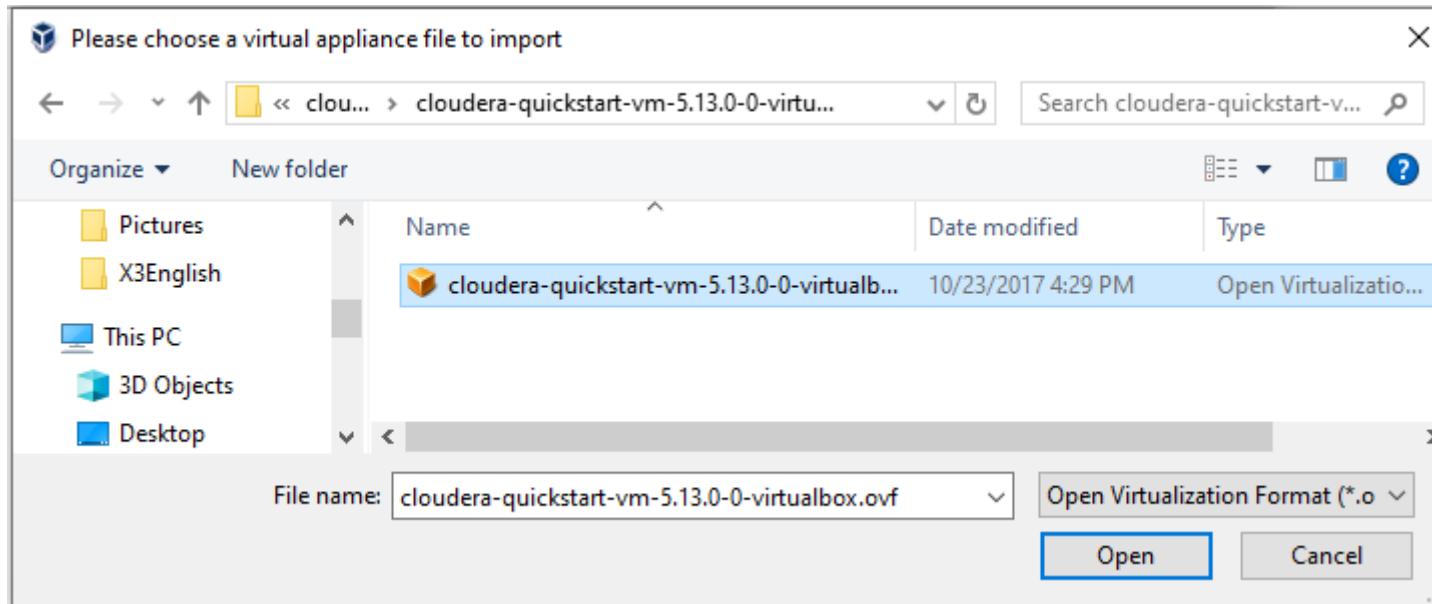
[https://downloads.cloudera.com/demo\\_vm/virtualbox/cloudera-quickstart-vm-5.13.0-0-virtualbox.zip](https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.13.0-0-virtualbox.zip)

- Open Oracle VM VirtualBox. Click File → Import Appliance



# Install Cloudera Quickstart VM

Choose “cloudera-quickstart-vm-5.13.0-0-virtualbox.ovf”



# Install Cloudera Quickstart VM

?   X

← Import Virtual Appliance

### Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1	
Name	cloudera-quickstart-vm-5.13.0-0-virtualbox 1
Guest OS Type	Red Hat (64-bit)
CPU	2
RAM	4096 MB
DVD	<input checked="" type="checkbox"/>
Network Adapter	<input checked="" type="checkbox"/> Intel PRO/1000 MT Desktop (82540EM)
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Virtual Disk Image	cloudera-quickstart-vm-5.13.0-0-virtualbox-disk1.vmdk
Base Folder	C:\Users\ACER\VirtualBox VMs
Primary Group	/

You can modify the base folder which will host all the virtual machines. Home folders can also be individually (per virtual machine) modified.

C:\Users\ACER\VirtualBox VMs

MAC Address Policy:  Include only NAT network adapter MAC addresses

Additional Options:  Import hard drives as VDI

Appliance is not signed

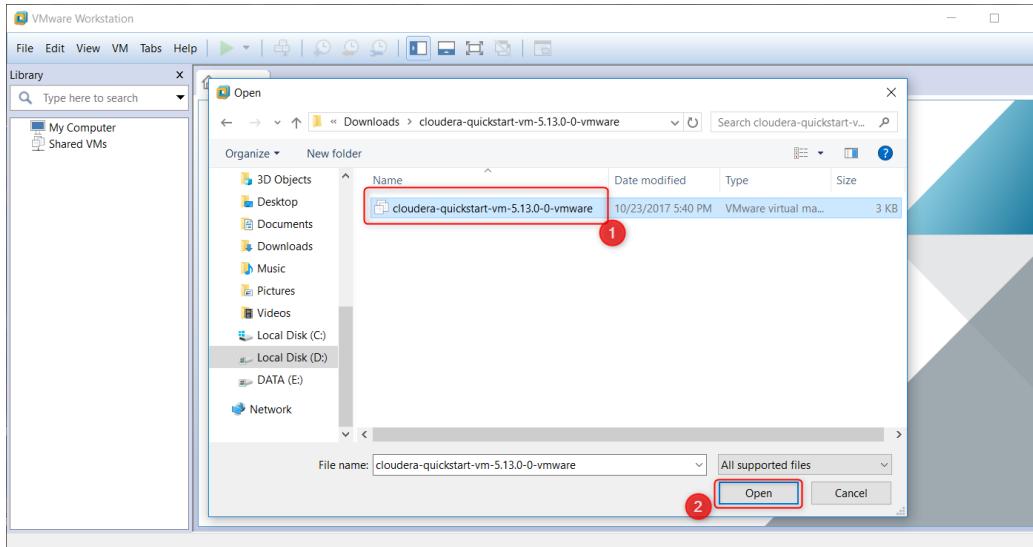
Restore Defaults Import Cancel

# Install Cloudera Quickstart VM

## VMware

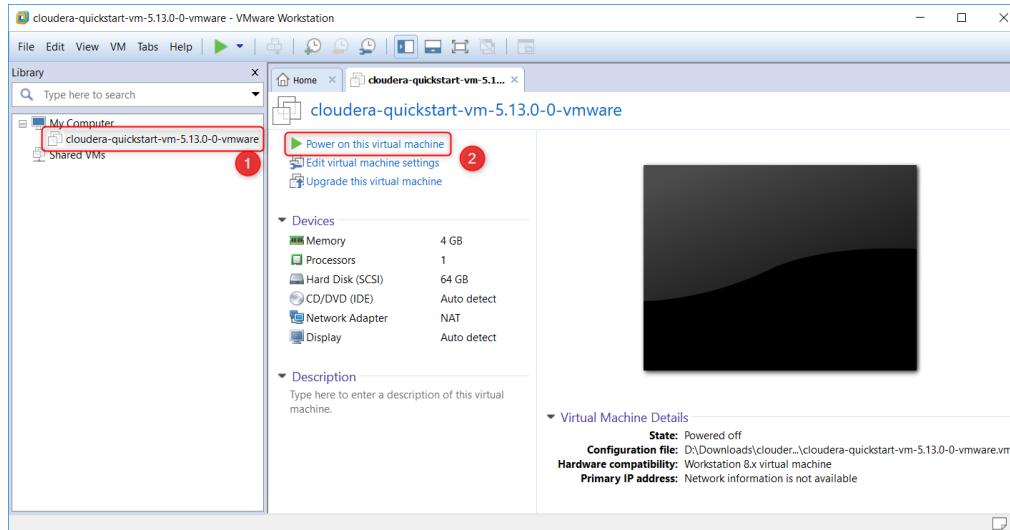
- Download Cloudera Quickstart VM for Vmware

[https://downloads.cloudera.com/demo\\_vm/vmware/cloudera-quickstart-vm-5.13.0-0-vmware.zip](https://downloads.cloudera.com/demo_vm/vmware/cloudera-quickstart-vm-5.13.0-0-vmware.zip)



# Install Cloudera Quickstart VM

VMware



# Projects

- Dasta Classifications
- Data anomaly detection
- Tourist behaviour analysis
- Credit scoring
- Price forecasting
- Streaming data
- Traffic data analysis
- Time series data



# Q & A



## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.

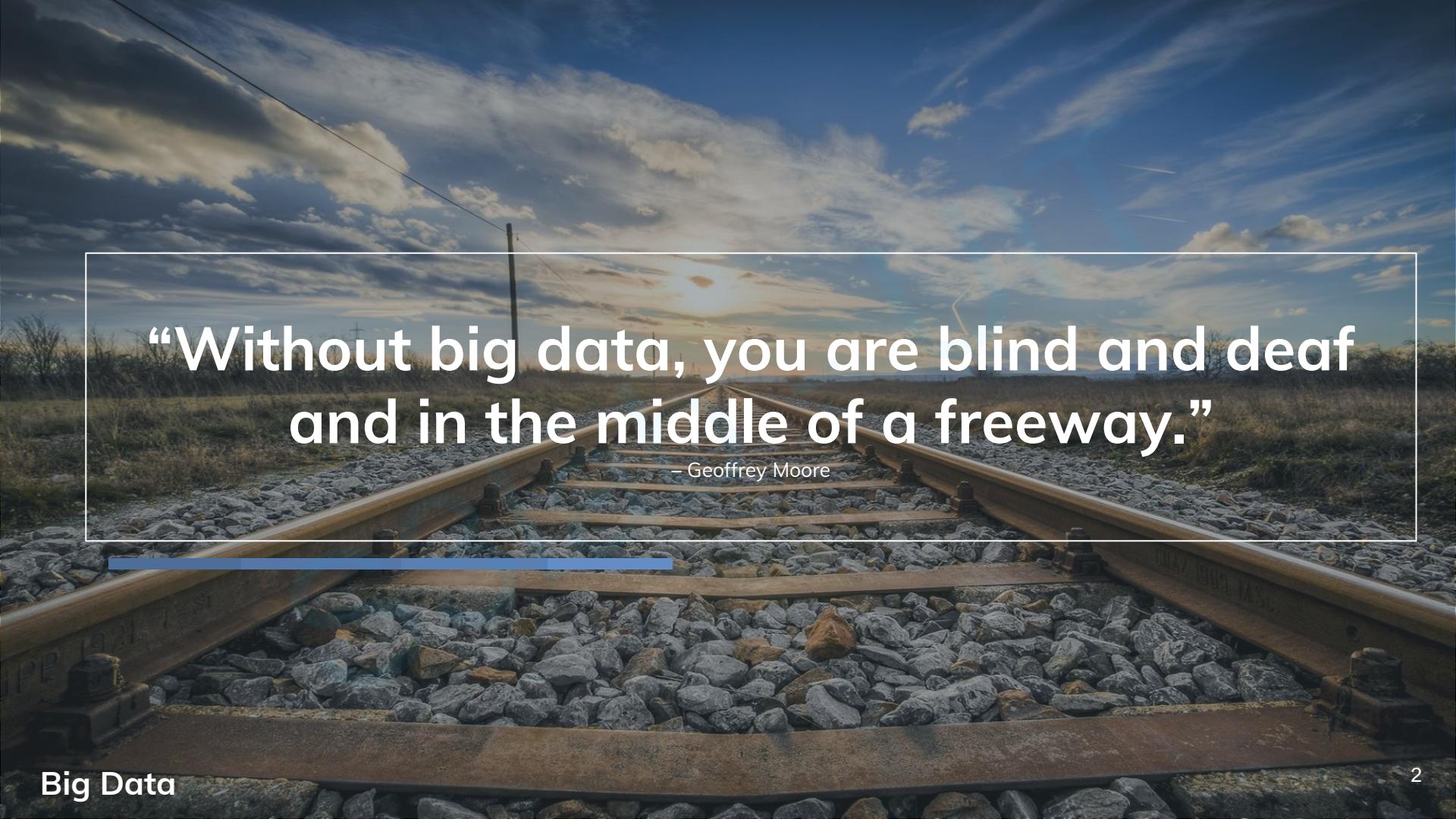
# Big Data

Hadoop tutorial

Trong-Hop Do

**S<sup>3</sup>Lab**

*Smart Software System Laboratory*



“Without big data, you are blind and deaf  
and in the middle of a freeway.”

– Geoffrey Moore

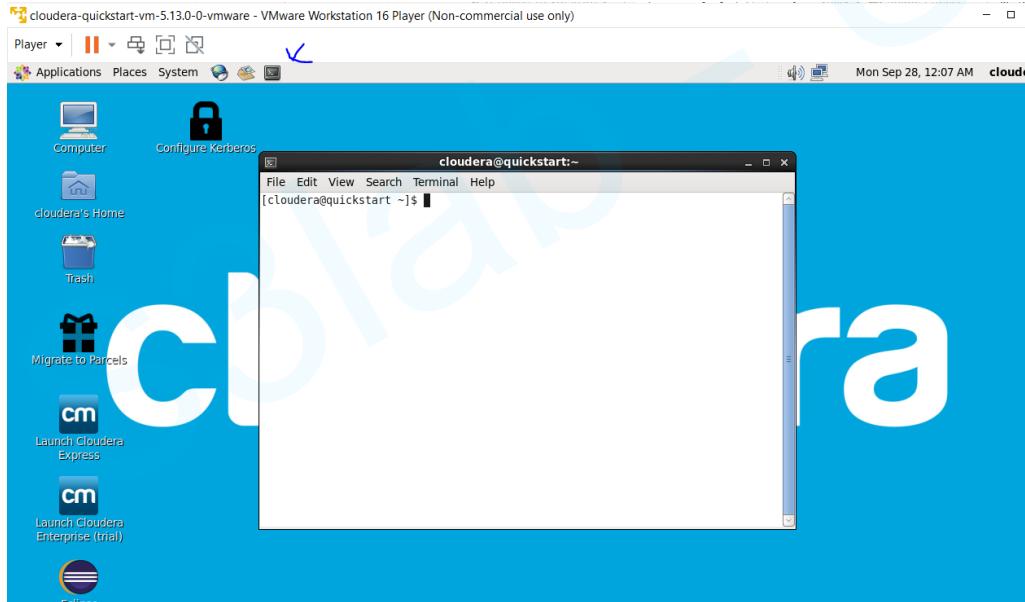
# Hadoop Shell Commands

---

# Hadoop Shell Commands to Manage HDFS

## Open terminal

- In this exercise, we will practise generic HDFS commands and get ourselves familiar with Hadoop command line interface.
- Open Terminal on your VM, navigate to Application > System Tools > Terminal Or use the shortcut on desktop.



# Hadoop Shell Commands to Manage HDFS

## hdfs dfs

- View description of all the commands associated with FsShell subsystem
- Command: **hdfs dfs**

```
[cloudera@quickstart ~]$ hdfs dfs
Usage: hadoop fs [generic options]
      [-appendToFile <localsrc> ... <dst>]
      [-cat [-ignoreCrc] <src> ...]
      [-checksum <src> ...]
      [-chgrp [-R] GROUP PATH...]
      [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
      [-chown [-R] [OWNER][:[GROUP]] PATH...]
      [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
      [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-count [-q] [-h] [-v] [-x] <path> ...]
      [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
      [-createSnapshot <snapshotDir> [<snapshotName>]]
      [-deleteSnapshot <snapshotDir> <snapshotName>]
      [-df [-h] [<path> ...]]
      [-du [-s] [-h] [-x] <path> ...]
      [-expunge]
      [-find <path> ... <expression> ...]
      [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-getfacl [-R] <path>]
      [-getfattr [-R] {-n name | -d} [-e en] <path>]
      [-getmerge [-nl] <src> <localdst>]
      [-help [cmd ...]]
      [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]
      [-mkdir [-p] <path> ...]
```

# Hadoop Shell Commands to Manage HDFS

## ls

- HDFS Command to display the list of Files and Directories in HDFS.
- Command: **hdfs dfs –ls**

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 1 items
drwxr-xr-x - cloudera cloudera
[cloudera@quickstart ~]$ 0 2020-09-16 07:29 student
```

# Hadoop Shell Commands to Manage HDFS

## ls

- View the content of **root** directory.
- Command: **hdfs dfs –ls /**

```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 7 items
drwxrwxrwx  - hdfs    supergroup          0 2017-10-23 10:29 /benchmarks
drwxr-xr-x  - cloudera supergroup          0 2020-09-26 05:36 /dataset
drwxr-xr-x  - hbase    supergroup          0 2020-09-27 23:42 /hbase
drwxr-xr-x  - solr    solr                0 2017-10-23 10:32 /solr
drwxrwxrwt  - hdfs    supergroup          0 2020-09-16 08:12 /tmp
drwxr-xr-x  - hdfs    supergroup          0 2017-10-23 10:31 /user
drwxr-xr-x  - hdfs    supergroup          0 2017-10-23 10:31 /var
```

- Note: the directory structure in HDFS has nothing to do with the directory structure of the local filesystem, they are separate namespaces.

# Hadoop Shell Commands to Manage HDFS

## ls

- The **home** directory is **/user/cloudera/**
- Command: **hdfs dfs –ls** or **hdfs dfs –ls /user/cloudera**

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 4 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-26 06:02 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:07 inputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-16 07:29 student
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera
Found 4 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-26 06:02 /user/cloudera/dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:07 /user/cloudera/inputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:29 /user/cloudera/outputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-16 07:29 /user/cloudera/student
```

# Hadoop Shell Commands to Manage HDFS

## mkdir

- HDFS Command to create the directory in HDFS.
- Usage: `hdfs dfs –mkdir directory_name`
- Command: **`hdfs dfs –mkdir dataset`**

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir dataset
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 2 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-26 04:21 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-16 07:29 student
[cloudera@quickstart ~]$
```

# Hadoop Shell Commands to Manage HDFS

## touchz

- HDFS Command to create a file in HDFS with file size 0 bytes.
- Usage: `hdfs dfs –touchz directory/filename`
- Command: **`hdfs dfs –touchz dataset/sample`**

```
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -touchz dataset/sample
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -ls dataset
Found 1 items
-rw-r--r-- 1 cloudera cloudera      0 2020-09-29 03:13 dataset/sample
[cloudera@quickstart HadoopStreaming]$ █
```

# Hadoop Shell Commands to Manage HDFS

du

- HDFS Command to check the file size.
- Usage: `hdfs dfs –du –s directory/filename`
- Command: **`hdfs dfs –du –s dataset/sample`**

```
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -du -s dataset/sample
0 0 dataset/sample
[cloudera@quickstart HadoopStreaming]$ █
```

# Hadoop Shell Commands to Manage HDFS

## put

- HDFS Command to copy single source or multiple sources from local file system to the destination file system.
- Usage: `hdfs dfs -put <localsrc> <destination>`
- Command: **`hdfs dfs -put data.txt dataset`**

```
[cloudera@quickstart HadoopStreaming]$ echo 'This is a test file' > data.txt
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -put data.txt dataset
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -ls dataset
Found 2 items
-rw-r--r--  1 cloudera cloudera      20 2020-09-29 03:17 dataset/data.txt
-rw-r--r--  1 cloudera cloudera       0 2020-09-29 03:13 dataset/sample
[cloudera@quickstart HadoopStreaming]$
```

# Hadoop Shell Commands to Manage HDFS

## cat

- HDFS Command that reads a file on HDFS and prints the content of that file to the standard output.
- Usage: `hdfs dfs –cat /path/to/file_in_hdfs`
- Command: **`hdfs dfs –cat dataset/data.txt`**

```
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -cat dataset/data.txt
This is a test file
[cloudera@quickstart HadoopStreaming]$ █
```

# Hadoop Shell Commands to Manage HDFS

## get

- HDFS Command to copy files from hdfs to the local file system.
- Usage: hdfs dfs -get <src> <localdst>
- Command: **hdfs dfs –get dataset/sample /home/cloudera**

```
[cloudera@quickstart ~]$ hdfs dfs -get /dataset/sample /home/cloudera
[cloudera@quickstart ~]$ ls /home/cloudera/
cloudera-manager      Documents          lib        student.java
cm_api.py              Downloads         Music      Templates
codegen_categories.java eclipse          enterprise-deployment.json    Videos
course.java            express-deployment.json  Pictures   workspace
data.txt               kerberos          Public
Desktop                [REDACTED]          sample
[cloudera@quickstart ~]$ █
```

# Hadoop Shell Commands to Manage HDFS

## count

- HDFS Command to count the number of directories, files, and bytes under the paths that match the specified file pattern.
- Usage: `hdfs dfs -count <path>`
- Command: **`hdfs dfs –count dataset`**

```
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -count dataset
          1          2          20 dataset
[cloudera@quickstart HadoopStreaming]$ █
```

# Hadoop Shell Commands to Manage HDFS

rm

- HDFS Command to remove the file from HDFS.
- Usage: `hdfs dfs –rm <path>`
- Command: **`hdfs dfs –rm dataset/sample`**

```
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -rm dataset/sample
Deleted dataset/sample
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -ls dataset
Found 1 items
-rw-r--r--    1 cloudera cloudera      20 2020-09-29 03:17 dataset/data.txt
[cloudera@quickstart HadoopStreaming]$ █
```

# Hadoop Shell Commands to Manage HDFS

rm -r

- HDFS Command to remove the entire directory and all of its content from HDFS.
- Usage: hdfs dfs -rm -r <path>
- Command: **hdfs dfs -rm -r dataset**

```
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -ls
Found 7 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 00:30 HSOutput
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 23:13 HadoopStreaming
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 05:17 ReduceJoin
drwxr-xr-x  - cloudera cloudera      0 2020-09-26 06:02 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:07 inputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-16 07:29 student
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -rm -r dataset
Deleted dataset
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -ls dataset
ls: `dataset': No such file or directory
[cloudera@quickstart HadoopStreaming]$
```

# Hadoop Shell Commands to Manage HDFS

## cp

- HDFS Command to copy files from source to destination. This command allows multiple sources as well, in which case the destination must be a directory.
- Usage: `hdfs dfs -cp <src> <dest>`
- Command: `hdfs dfs -cp /user/cloudera/dataset/data.txt /user/cloudera/dataset/datacopy.txt`

```
[cloudera@quickstart ~]$ hdfs dfs -cp /user/cloudera/dataset/data.txt /user/cloudera/dataset/datacopy.txt
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/dataset
Found 2 items
-rw-r--r-- 1 cloudera cloudera          20 2020-09-26 05:00 /user/cloudera/dataset/data.txt
-rw-r--r-- 1 cloudera cloudera          20 2020-09-26 06:02 /user/cloudera/dataset/datacopy.txt
[cloudera@quickstart ~]$ █
```

# Hadoop Shell Commands to Manage HDFS

## rmdir

- HDFS Command to remove a empty directory.
- Usage: `hdfs dfs -rmdir <path>`
- Command: `hdfs dfs –rmdir /user/cloudera/dataset`

# Hadoop Shell Commands to Manage HDFS

## usage

- HDFS Command that returns the help for an individual command.
- Usage: `hdfs dfs -usage <command>`
- Command: **`hdfs dfs -usage mkdir`**
- By using usage command you can get information about any command

# Hadoop Shell Commands to Manage HDFS

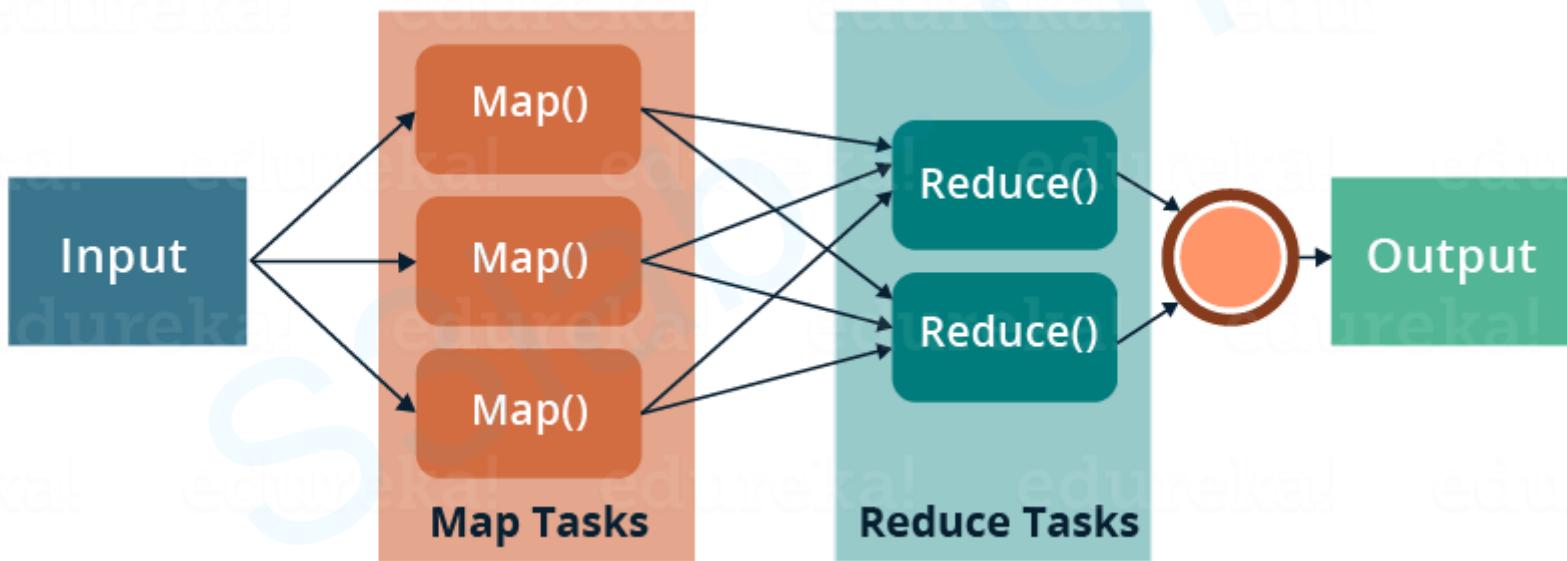
## help

- HDFS Command that displays help for given command or all commands if none is specified.
- Command: **hdfs dfs -help**

# Word Count MapReduce

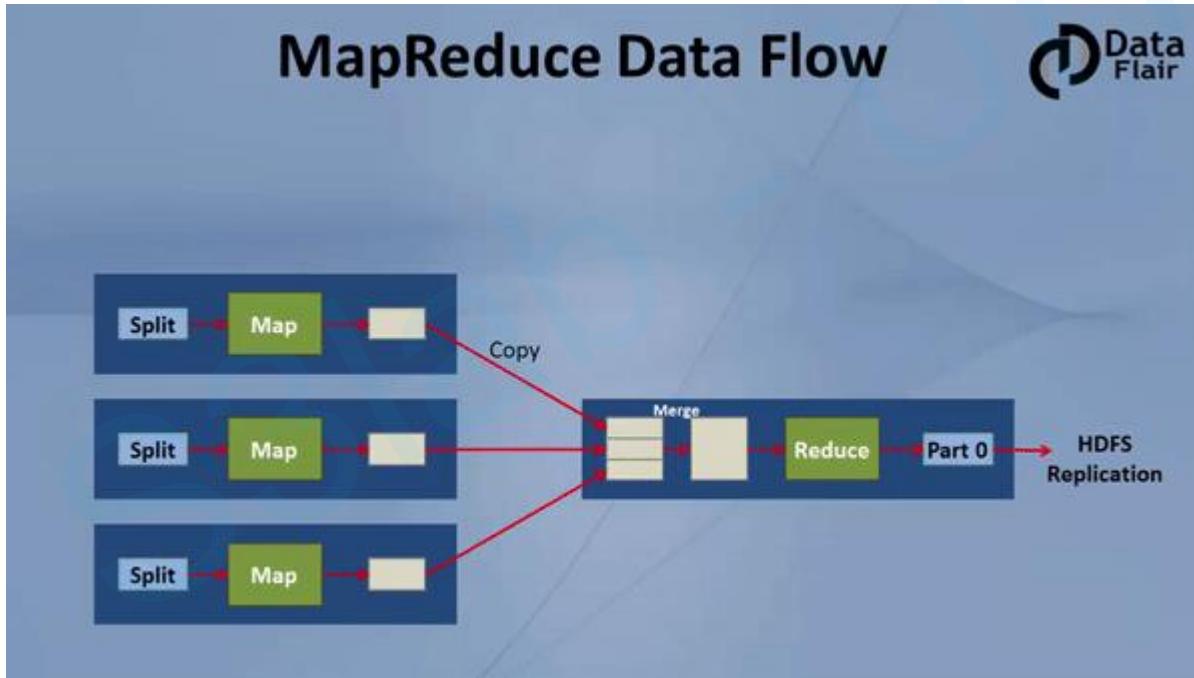
---

# Hadoop MapReduce



# Hadoop MapReduce

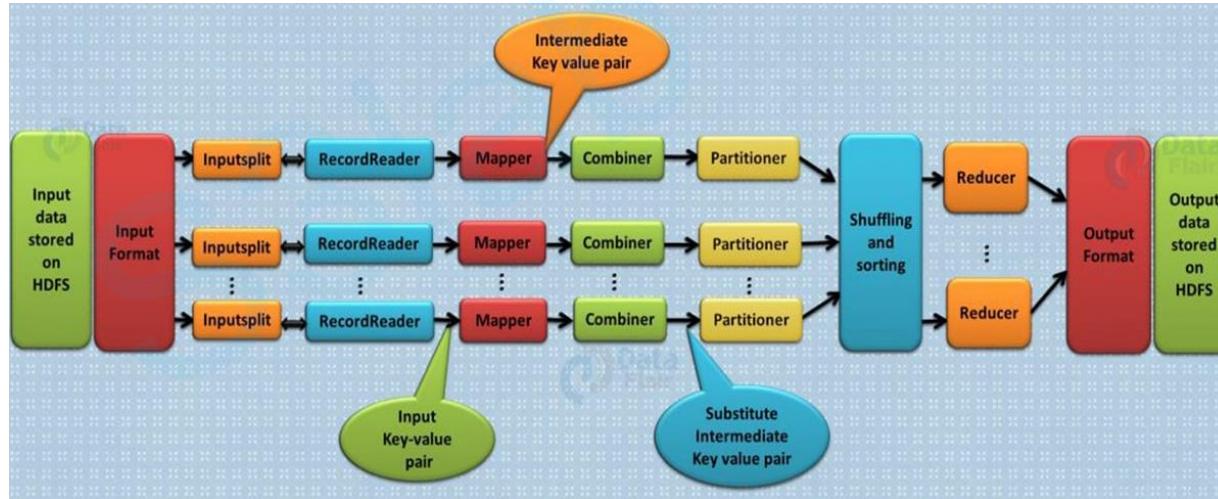
How Hadoop MapReduce work?



# Hadoop MapReduce

## Input Files

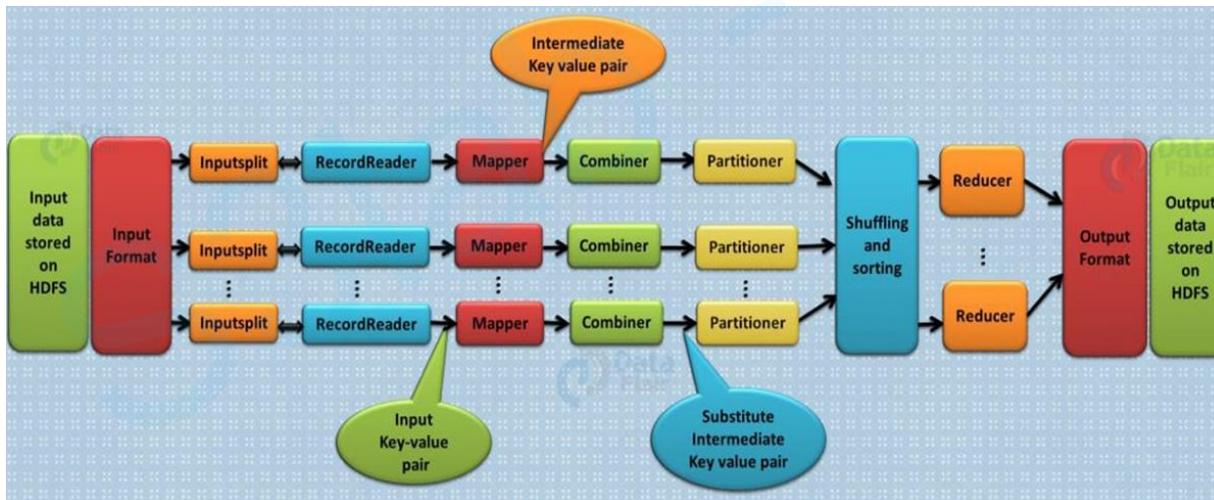
- The data for a MapReduce task is stored in input files, and input files typically lives in HDFS. The format of these files is arbitrary, while line-based log files and binary format can also be used.



# Hadoop MapReduce

## InputFormat

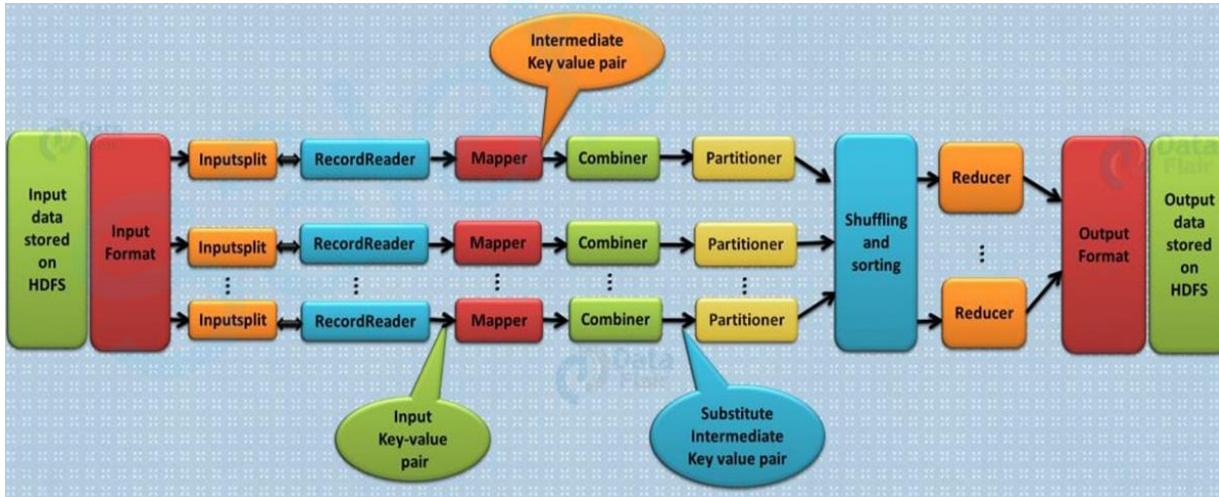
- Now, InputFormat defines how these input files are split and read. It selects the files or other objects that are used for input. InputFormat creates InputSplit.



# Hadoop MapReduce

## InputSplits

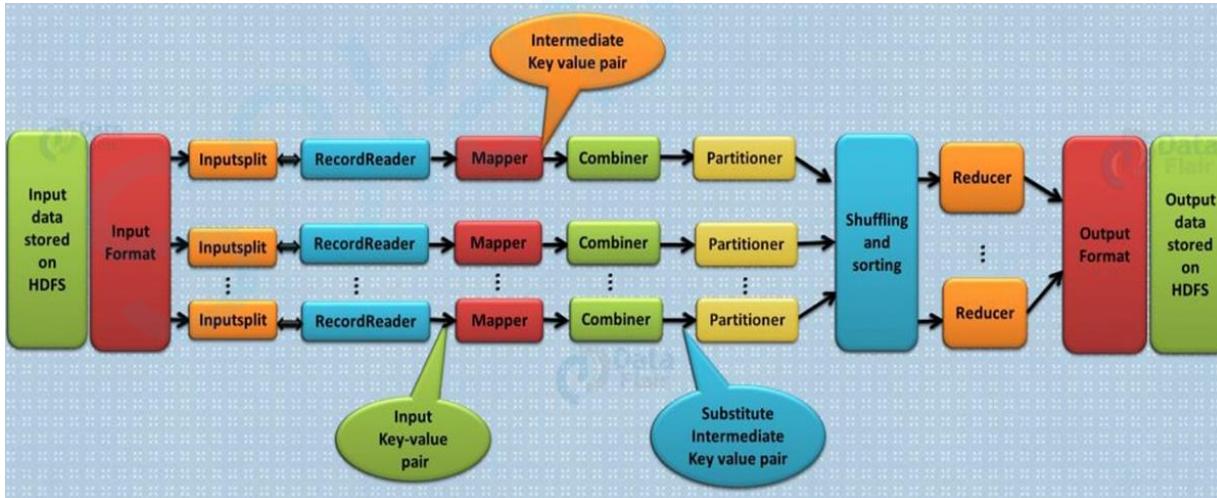
- It is created by InputFormat, logically represent the data which will be processed by an individual Mapper (We will understand mapper below). One map task is created for each split; thus the number of map tasks will be equal to the number of InputSplits. The split is divided into records and each record will be processed by the mapper.



# Hadoop MapReduce

## RecordReader

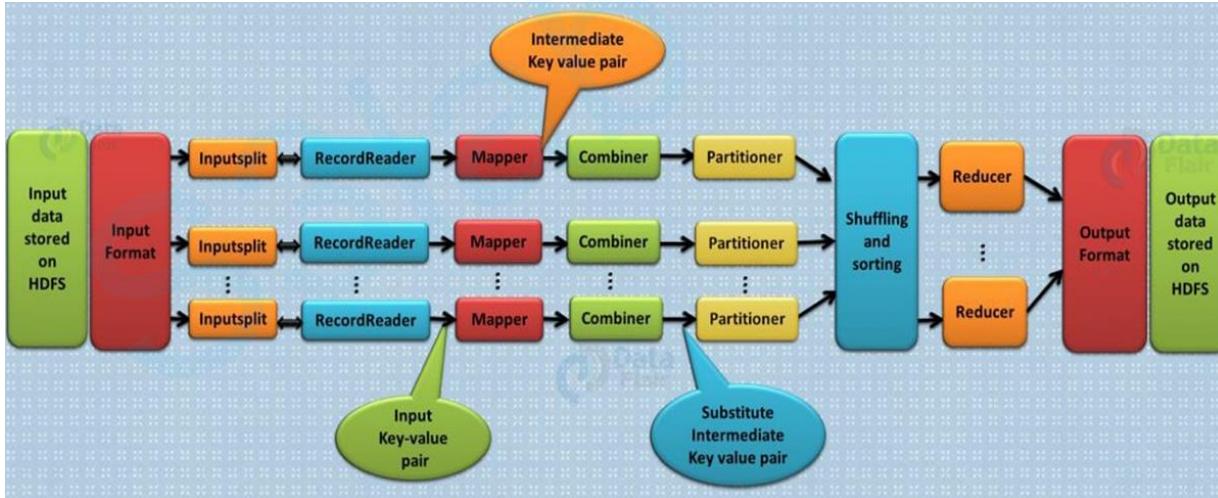
- It communicates with the **InputSplit** in Hadoop MapReduce and converts the data into key-value pairs suitable for reading by the mapper. By default, it uses TextInputFormat for converting data into a key-value pair. RecordReader communicates with the InputSplit until the file reading is not completed. It assigns byte offset (unique number) to each line present in the file. Further, these key-value pairs are sent to the mapper for further processing.



# Hadoop MapReduce

## Mapper

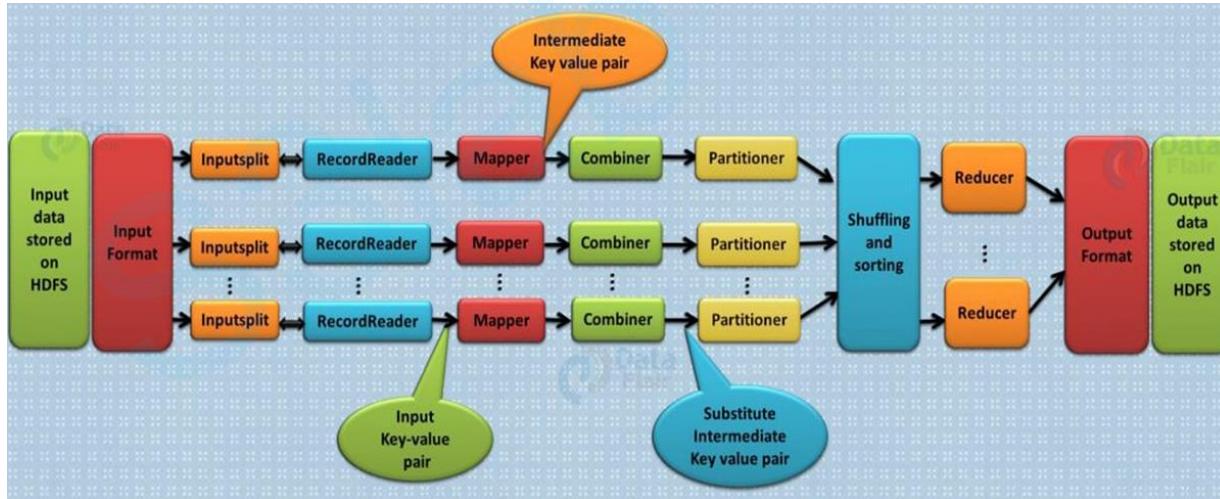
- It processes each input record (from RecordReader) and generates new key-value pair, and this key-value pair generated by Mapper is completely different from the input pair. The output of Mapper is also known as intermediate output which is written to the local disk. The output of the Mapper is not stored on HDFS as this is temporary data and writing on HDFS will create unnecessary copies (also HDFS is a high latency system).  
Mappers output is passed to the combiner for further process



# Hadoop MapReduce

## Combiner

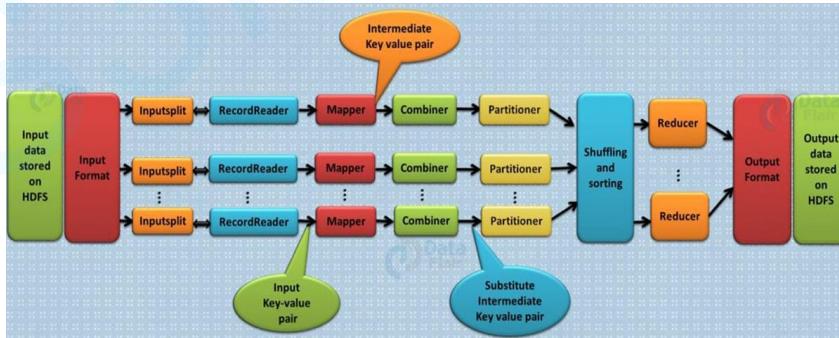
- The combiner is also known as ‘Mini-reducer’. Hadoop MapReduce Combiner performs local aggregation on the mappers’ output, which helps to minimize the data transfer between mapper and reducer (we will see reducer below). Once the combiner functionality is executed, the output is then passed to the partitioner for further work.



# Hadoop MapReduce

## Partitioner

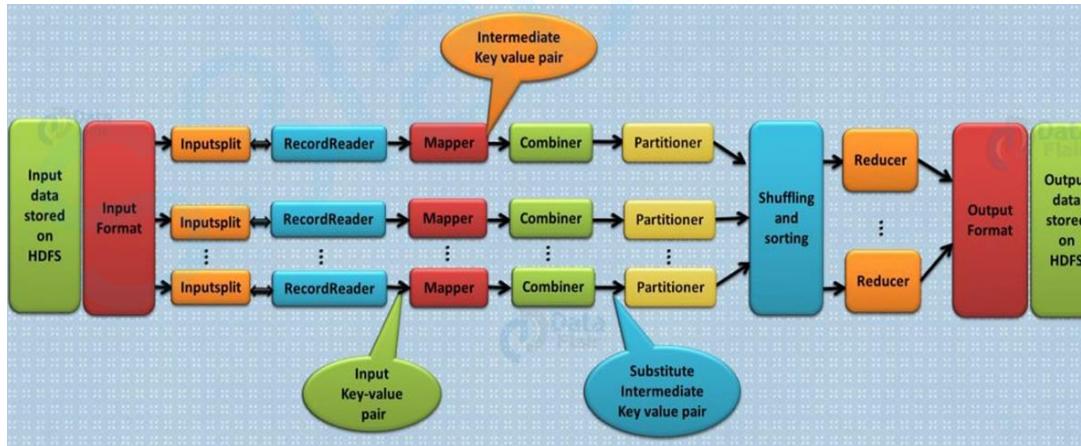
- Hadoop MapReduce, Partitioner comes into the picture if we are working on more than one reducer (for one reducer partitioner is not used).
- Partitioner takes the output from combiners and performs partitioning. Partitioning of output takes place on the basis of the key and then sorted. By hash function, key (or a subset of the key) is used to derive the partition.
- According to the key value in MapReduce, each combiner output is partitioned, and a record having the same key value goes into the same partition, and then each partition is sent to a reducer. Partitioning allows even distribution of the map output over the reducer.



# Hadoop MapReduce

## Shuffling and Sorting

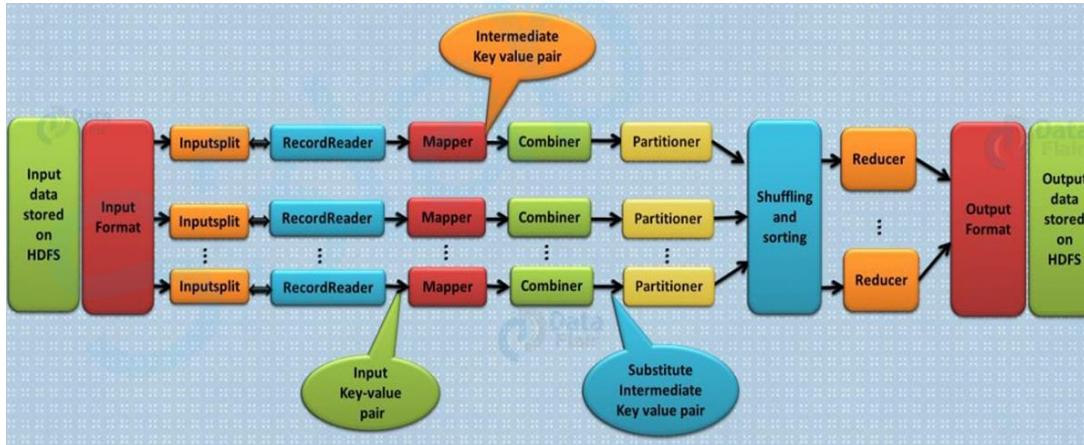
- Now, the output is Shuffled to the reduce node (which is a normal slave node but reduce phase will run here hence called as reducer node). The shuffling is the physical movement of the data which is done over the network. Once all the mappers are finished and their output is shuffled on the reducer nodes, then this intermediate output is merged and sorted, which is then provided as input to reduce phase.



# Hadoop MapReduce

## Reducer

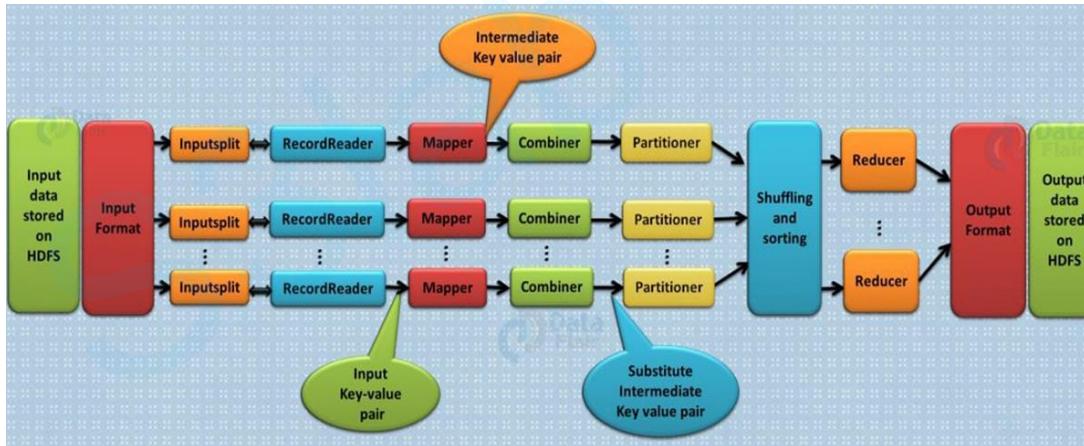
- It takes the set of intermediate key-value pairs produced by the mappers as the input and then runs a reducer function on each of them to generate the output. The output of the reducer is the final output, which is stored in HDFS.



# Hadoop MapReduce

## RecordWriter

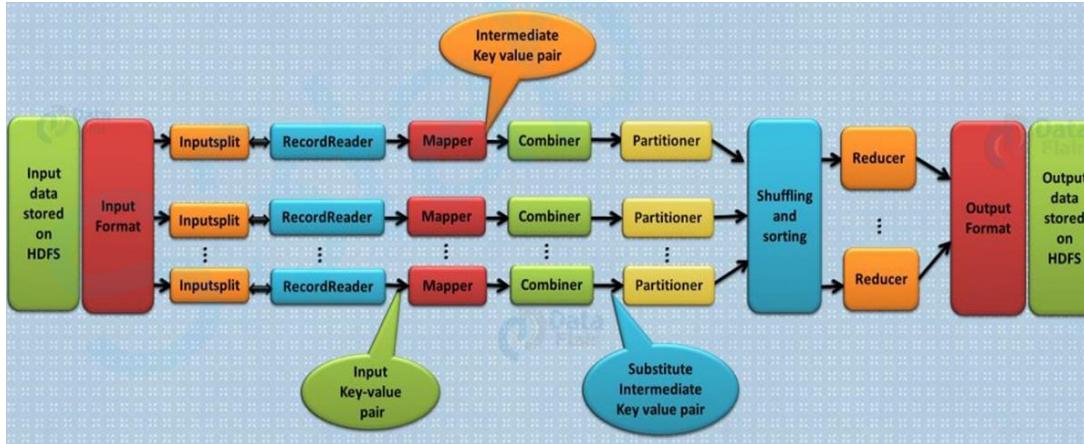
- It writes these output key-value pair from the Reducer phase to the output files.



# Hadoop MapReduce

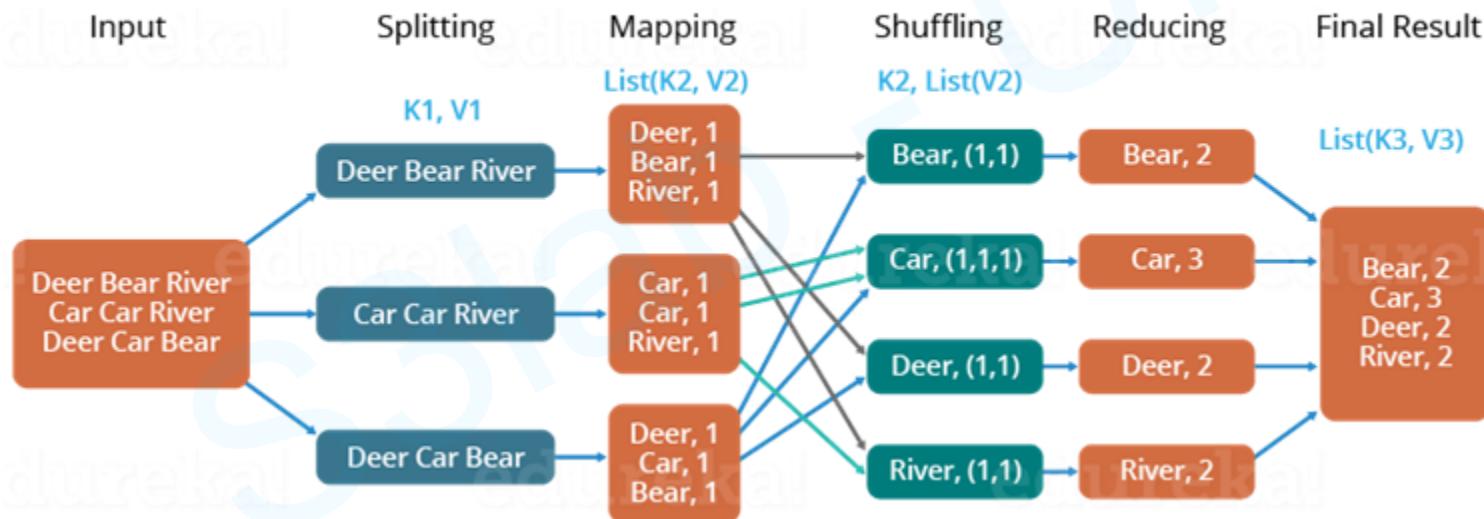
## OutputFormat

- The way these output key-value pairs are written in output files by RecordWriter is determined by the OutputFormat. OutputFormat instances provided by the Hadoop are used to write files in HDFS or on the local disk. Thus the final output of reducer is written on HDFS by OutputFormat instances.
- Hence, in this manner, a Hadoop MapReduce works over the cluster.



# Hadoop MapReduce

The Overall MapReduce Word Count Process



# Hadoop MapReduce

The entire MapReduce program can be fundamentally divided into three parts:

- Mapper Phase Code
- Reducer Phase Code
- Driver Code

# Hadoop MapReduce

## Mapper code

```
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);  
        }  
    }  
}
```

# Hadoop MapReduce

## Mapper code

```
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);
    }
}
```

# Hadoop MapReduce

## Driver code

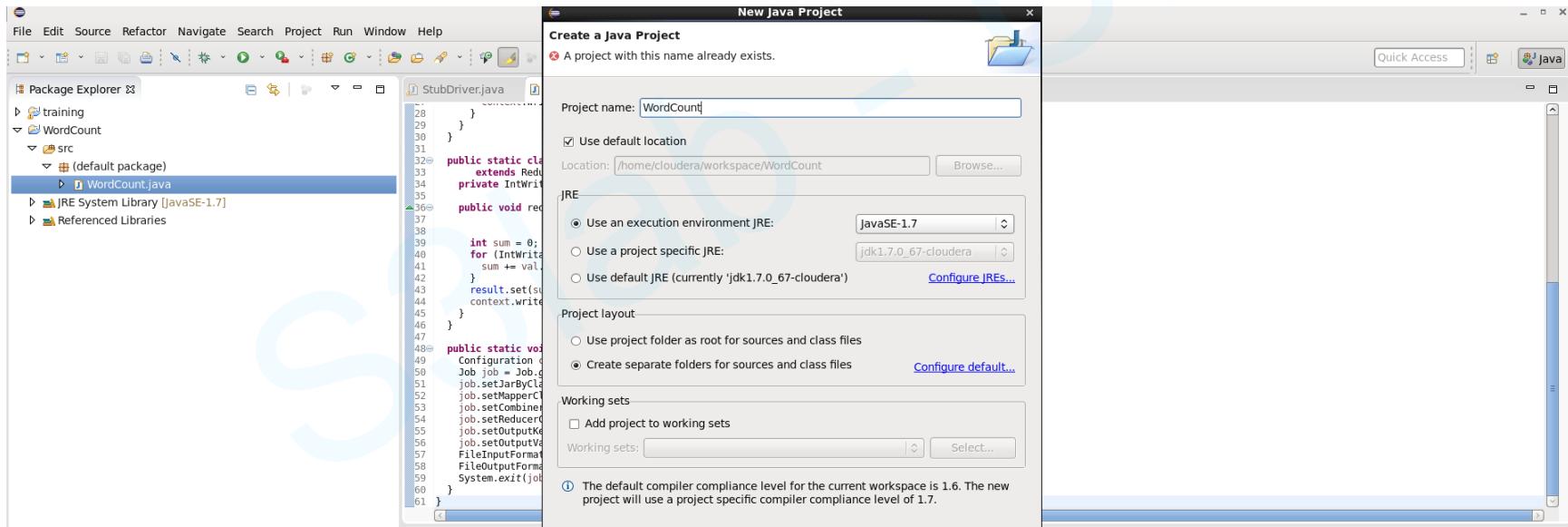
```
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
```

# Word Count MapReduce

Word Count MapReduce Tutorial starts from here

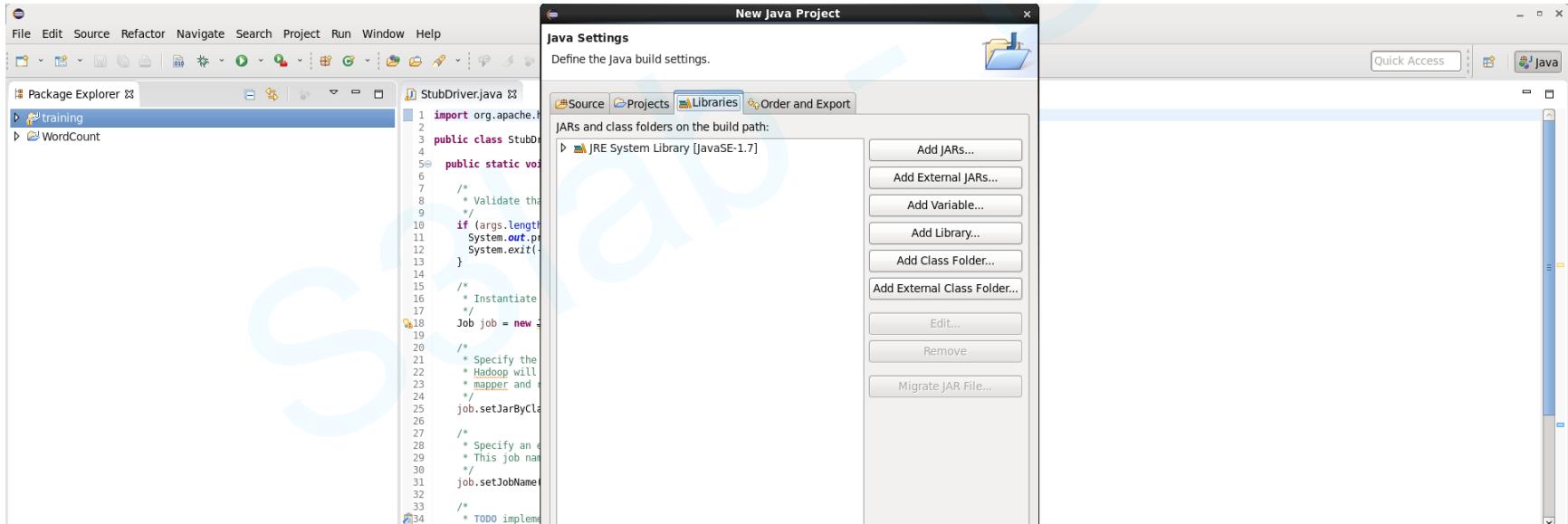
# Word Count MapReduce

- Open Eclipse on Cloudera Quickstart VM and create a new Java Project



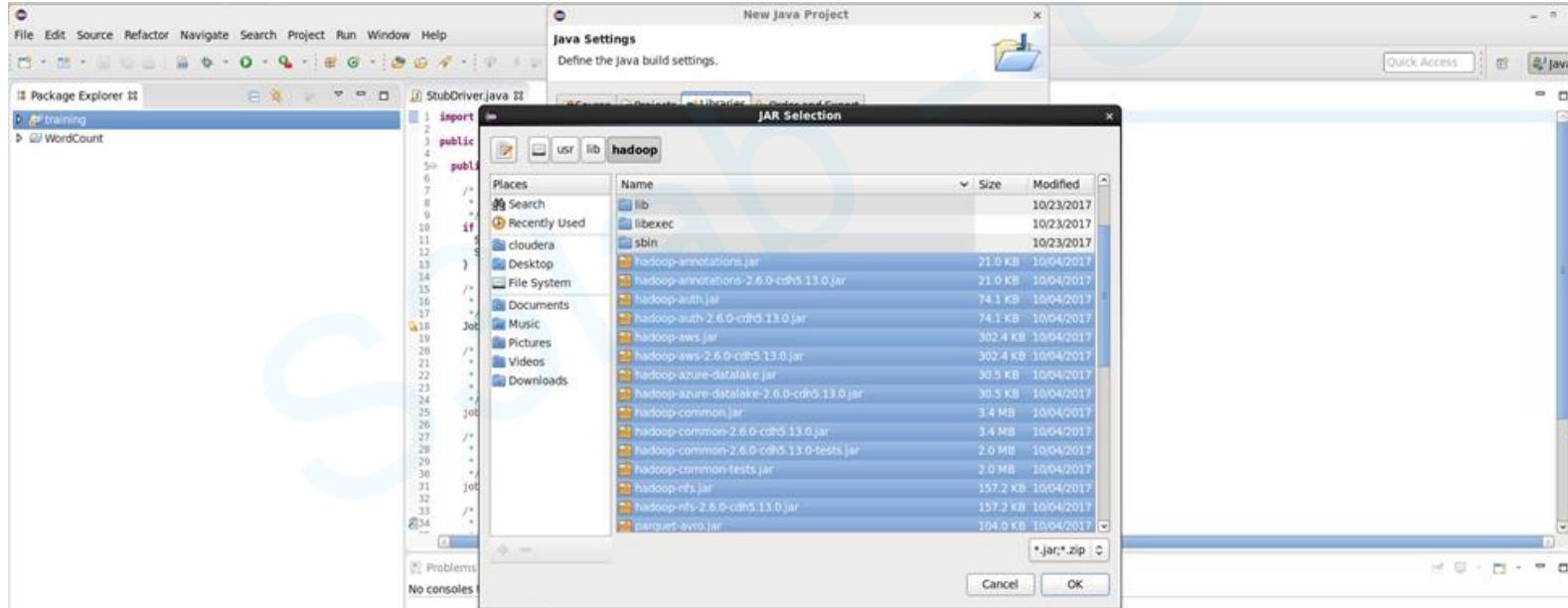
# Word Count MapReduce

- Add Hadoop Libraries to project through **Add External JARS**



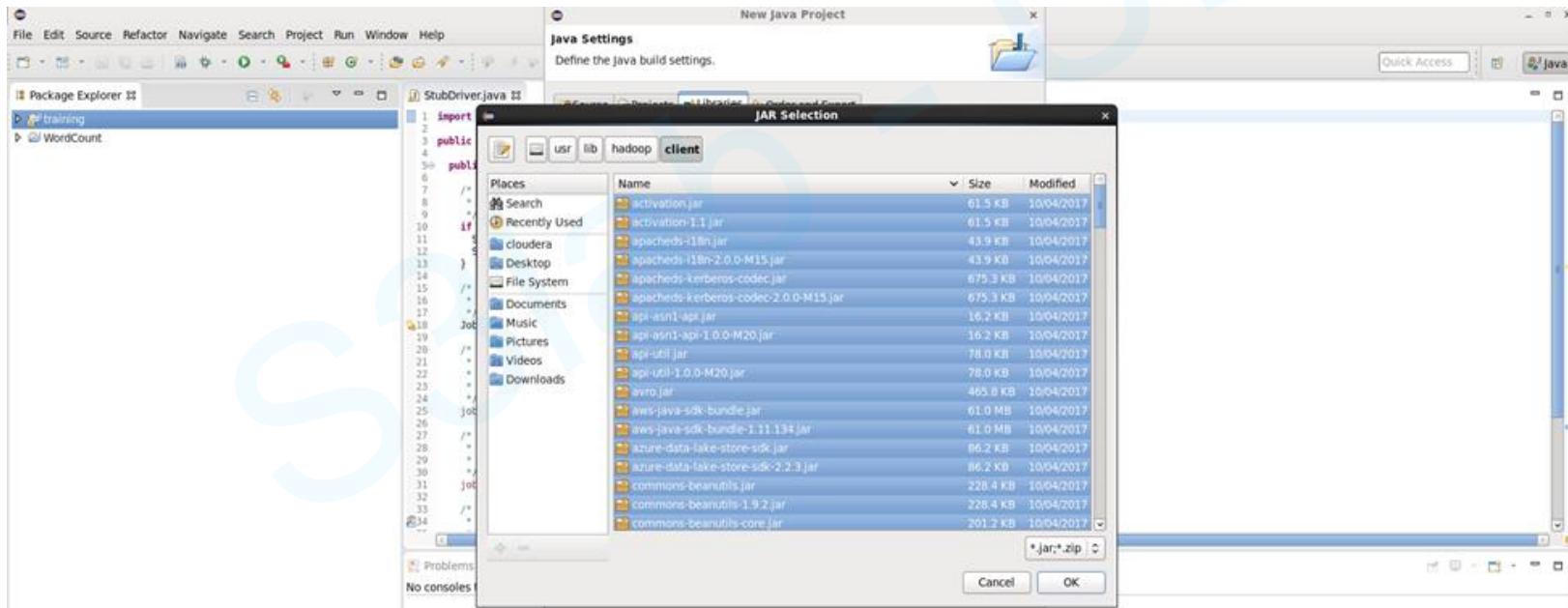
# Word Count MapReduce

- Add all .jar files in the folder **usr/lib/hadoop**



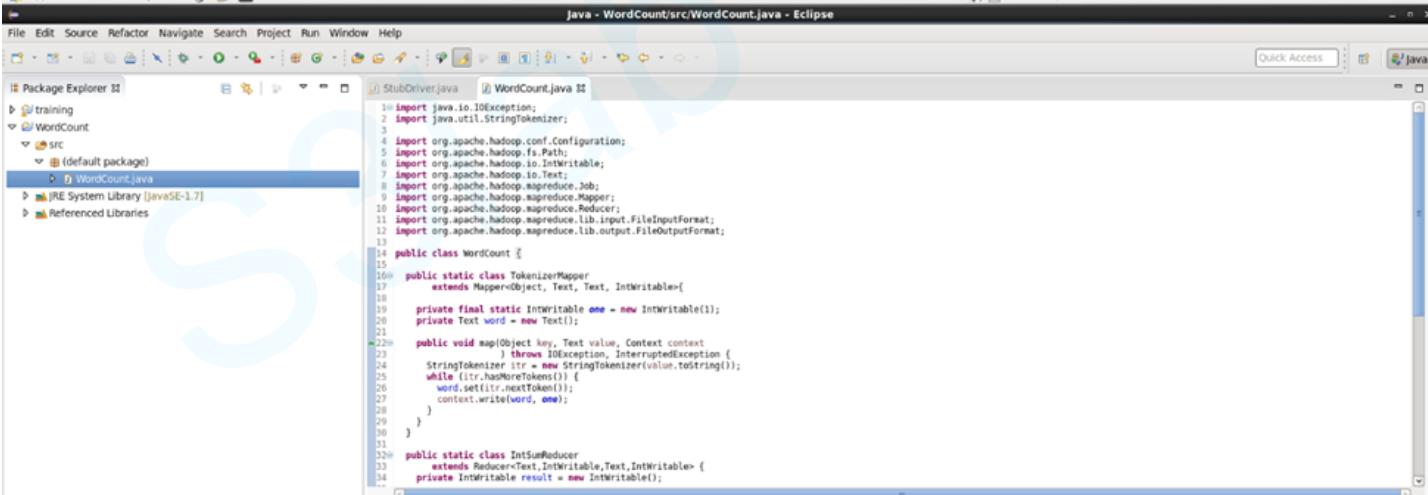
# Word Count MapReduce

- Add all .jar files in the folder **usr/lib/hadoop/client**



# Word Count MapReduce

- Copy the source code of WordCount programme from  
<https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

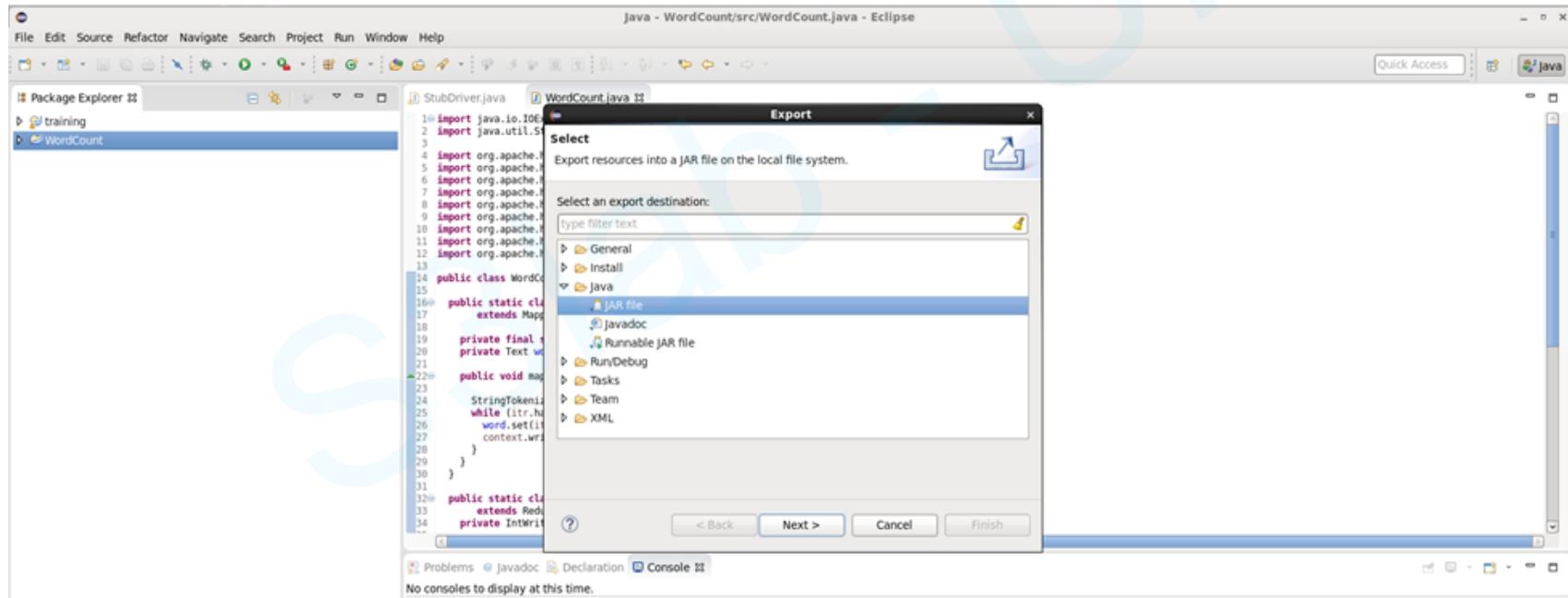


The screenshot shows the Eclipse IDE interface with the title "Java - WordCount/src/WordCount.java - Eclipse". The left side features the "Package Explorer" view, which displays a project structure with a "src" folder containing a "WordCount" package and a "WordCount.java" file. The right side is the code editor window showing the Java source code for the WordCount program.

```
Java - WordCount/src/WordCount.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java
Package Explorer
StubDriver.java WordCount.java
1 import java.io.IOException;
2 import java.util.StringTokenizer;
3
4 import org.apache.hadoop.conf.Configuration;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Job;
9 import org.apache.hadoop.mapreduce.Mapper;
10 import org.apache.hadoop.mapreduce.Reducer;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13
14 public class WordCount {
15
16     public static class TokenizerMapper
17         extends Mapper<Object, Text, Text, IntWritable> {
18
19         private final static IntWritable one = new IntWritable(1);
20         private Text word = new Text();
21
22         public void map(Object key, Text value, Context context
23                         ) throws IOException, InterruptedException {
24             StringTokenizer itr = new StringTokenizer(value.toString());
25             while (itr.hasMoreTokens()) {
26                 word.set(itr.nextToken());
27                 context.write(word, one);
28             }
29         }
30
31
32     public static class IntSumReducer
33         extends Reducer<Text,IntWritable,Text,IntWritable> {
34         private IntWritable result = new IntWritable();
35
36         protected void reduce(Text key, Iterable<IntWritable> values,
37                             Context context) throws IOException,
38                                     InterruptedException {
39             int sum = 0;
40             for (IntWritable val : values) {
41                 sum += val.get();
42             }
43             result.set(sum);
44             context.write(key, result);
45         }
46     }
47 }
48 }
```

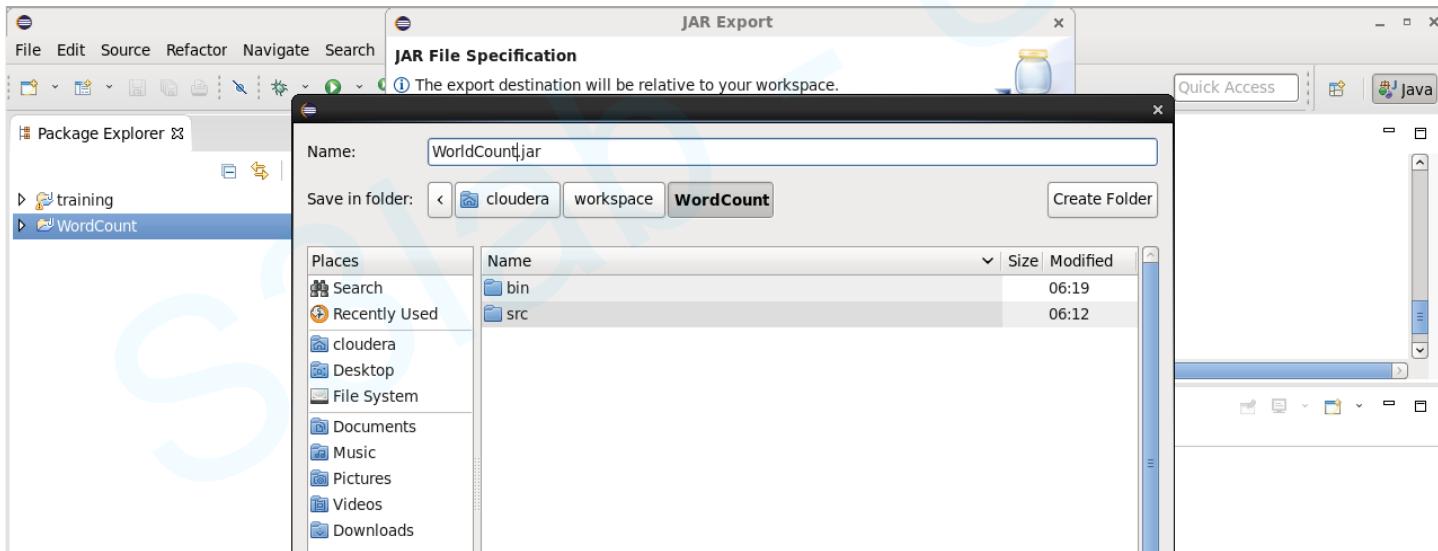
# Word Count MapReduce

- Export to jar file



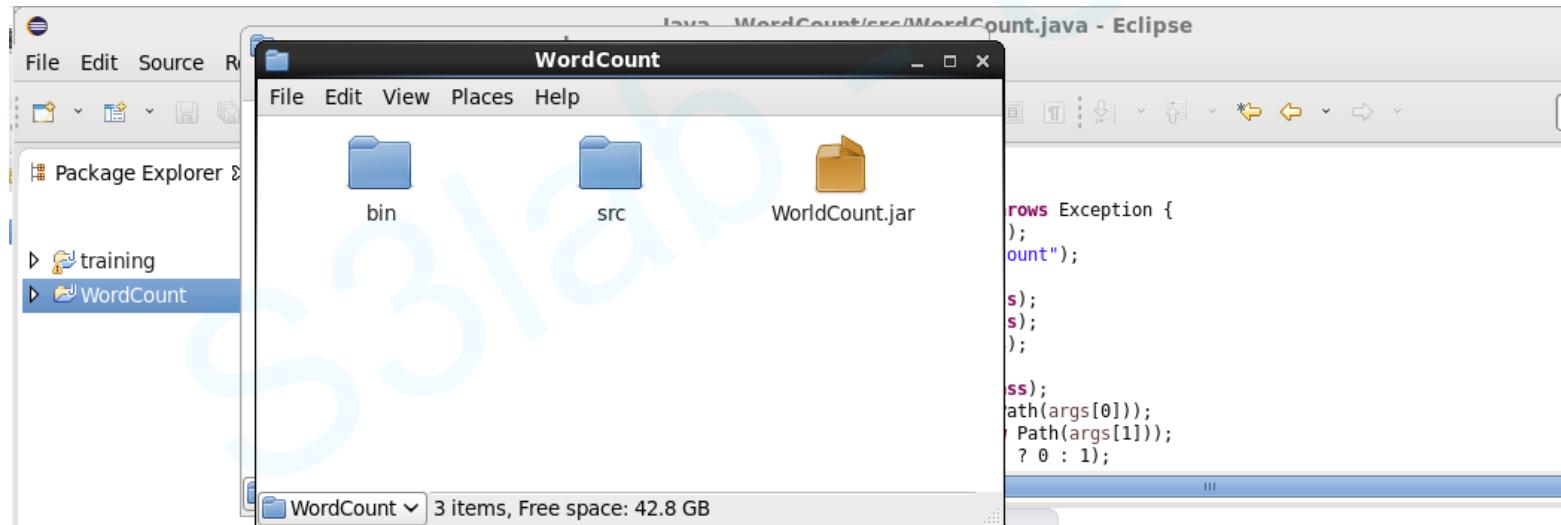
# Hadoop Shell Commands to Manage HDFS

- Name the file **WordCount.jar** and place it inside **/home/cloudera/workspace/WordCount**



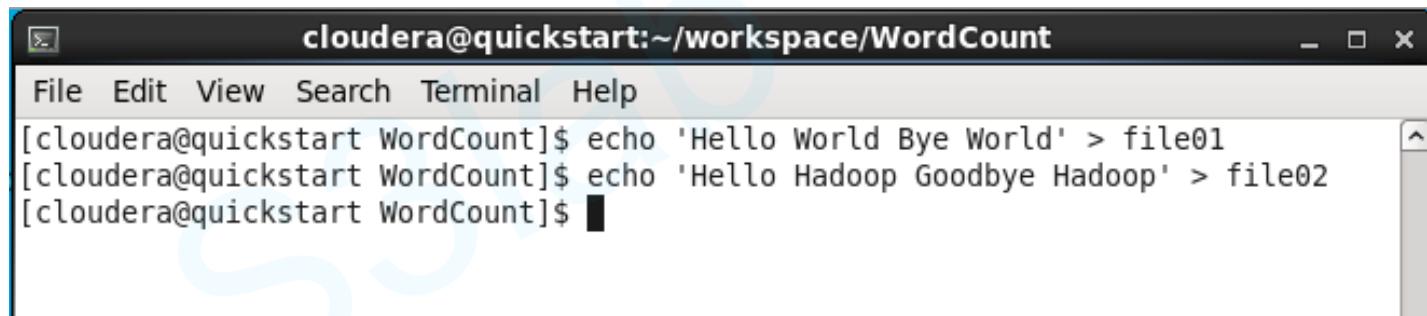
# Hadoop Shell Commands to Manage HDFS

- Check if the **WordCount.jar** file has been created



# Hadoop Shell Commands to Manage HDFS

- Create two text files inside `/home/cloudera/workspace/WordCount` folder



```
cloudera@quickstart:~/workspace/WordCount
File Edit View Search Terminal Help
[cloudera@quickstart WordCount]$ echo 'Hello World Bye World' > file01
[cloudera@quickstart WordCount]$ echo 'Hello Hadoop Goodbye Hadoop' > file02
[cloudera@quickstart WordCount]$
```

# Hadoop Shell Commands to Manage HDFS

- Create input directory in Hadoop

```
[cloudera@quickstart WordCount]$ hdfs dfs -mkdir inputWC
[cloudera@quickstart WordCount]$ hdfs dfs -ls
Found 3 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-26 06:02 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:00 inputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-16 07:29 student
[cloudera@quickstart WordCount]$ █
```

# Hadoop Shell Commands to Manage HDFS

- Move two text files to the input directory in Hadoop

```
[cloudera@quickstart WordCount]$ hdfs dfs -put file0* inputWC
[cloudera@quickstart WordCount]$ hdfs dfs -ls inputWC
Found 2 items
-rw-r--r--  1 cloudera cloudera      22 2020-09-27 07:07 inputWC/file01
-rw-r--r--  1 cloudera cloudera      28 2020-09-27 07:07 inputWC/file02
[cloudera@quickstart WordCount]$ █
```

# Hadoop Shell Commands to Manage HDFS

- Check the content of the two files in Hadoop

```
[cloudera@quickstart WordCount]$ hdfs dfs -cat inputWC/file01  
Hello World Bye World  
[cloudera@quickstart WordCount]$ hdfs dfs -cat inputWC/file02  
Hello Hadoop Goodbye Hadoop  
[cloudera@quickstart WordCount]$ █
```

# Hadoop Shell Commands to Manage HDFS

- Execute the JAR file on Hadoop

```
[cloudera@quickstart WordCount]$ hadoop jar WordCount.jar WordCount inputWC outputWC
```

- WordCount.jar: file to run
- WordCount: name of the class
- inputWC: input directory
- outputWC: output directory

- This will create a directory `/user/cloudera/outputWC` and two files within it

# Hadoop Shell Commands to Manage HDFS

- Check the output directory and the output files inside it

```
[cloudera@quickstart WordCount]$ hdfs dfs -ls
Found 4 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-26 06:02 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:07 inputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-16 07:29 student
[cloudera@quickstart WordCount]$ hdfs dfs -ls outputWC
Found 2 items
-rw-r--r--  1 cloudera cloudera      0 2020-09-27 07:29 outputWC/_SUCCESS
-rw-r--r--  1 cloudera cloudera    41 2020-09-27 07:29 outputWC/part-r-00000
```

- Note: Job ran with only one Reducer, so there should be only one part-file

# Hadoop Shell Commands to Manage HDFS

- Check the content of the output file

```
[cloudera@quickstart WordCount]$ hdfs dfs -cat outputWC/part*
Bye      1
Goodbye 1
Hadoop   2
Hello    2
World    2
[cloudera@quickstart WordCount]$ █
```

# Hadoop Shell Commands to Manage HDFS

- If ssh localhost is not working and it shows, ssh not installed. Then please enter the following: > sudo apt-get install ssh Enter password if prompted.
- If there is an error while executing the command. Please check the variables JAVA\_HOME and HADOOP\_CLASSPATH. Later reset the values and proceed.
- If the is ClassNotFoundException, then please find the details in the link here: LINK. Or one could compile and save the .jar file within the same source directory.
- If an error like the following appears on trying to make directories in HDFS viz. then please run start-dfs.sh.

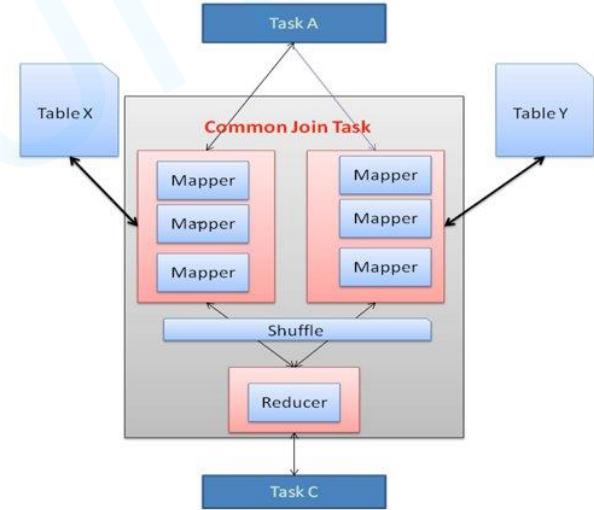
```
hadoop@hadoop:~/Desktop/hadoop-2.7.2$ bin/hdfs dfs -mkdir /user  
mkdir: Call From hadoop/127.0.1.1 to localhost:9000 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
```

# Join in MapReduce

---

# What is Join in MapReduce

- **Advantage** – Optimize Solution in terms of processing speed of the data.
- **Disadvantage** – Time Consuming for programmer, Non-ease mode of development due to 100's of lines of code and Availability of higher level frameworks like HIVE/PIG

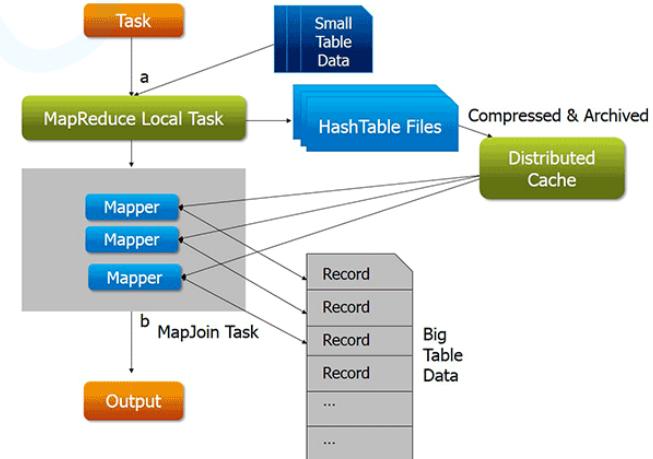


# Type of Join in MapReduce

## Map-Side Join

Read the data streams into the mappers and uses login within the mapper function to perform the join.

- **Where to use:** – When you have One large dataset and you need to join this with a small dataset. Also for Optimize performance.
- **Why:** – Smaller table will be loaded into memory and the join operation will happen during the mapper execution of large data set
- **Advantage:** Better performance.
- **Disadvantage:** Not Flexible i.e. can not be used if both data sets are large in size.
- **Note:** Reduce Side Join can also be used here but the performance will decrease.

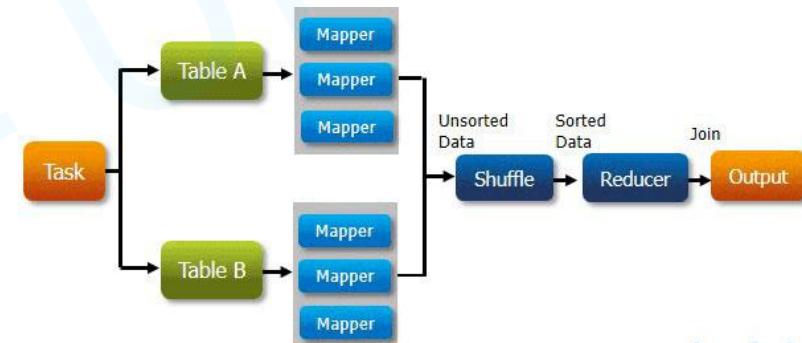


# Type of Join in MapReduce

## Reduce-Side Join

Process the multiple data streams through multiple map stages and perform the join at Reducer stage.

- **Where to use:** – When both the data sets are large.  
**Why:** – None of the dataset can be loaded in memory completely. Will have to process both tables separately and JOIN them at reducer side
- **Advantage:** Flexible and can be applied anywhere.  
**Disadvantage:** Poor performance in comparison with Map-side Joins
- **Note:** – Map Side Join can't be used here.



# Reduce side Join

- Suppose that I have two separate datasets of a sports complex:
  - **cust\_details:** It contains the details of the customer.
  - **transaction\_details:** It contains the transaction record of the customer.
- Using these two datasets, I want to know the lifetime value of each customer. In doing so, I will be needing the following things:
  - The person's name along with the frequency of the visits by that person.
  - The total amount spent by him/her for purchasing the equipment.

# Reduce side Join

Cust ID	First Name	Last Name	Age	Profession
4000001	Kristina	Chung	55	Pilot
4000002	Paige	Chen	74	Teacher
4000003	Sherri	Melton	34	Firefighter
4000004	Gretchen	Hill	66	Engineer
.....	.....	.....	.....	.....

Fig: cust\_details

Trans ID	Date	Cust ID	Amount	Game Type	Equipment	City	State	Mode
0000000	06-26-2011	4000001	40.33	Exercise & Fitness	Cardio Machine Accessories	Clarksville	Tennessee	credit
0000001	05-05-2011	4000002	198.44	Exercise & Fitness	Weightlifting Gloves	Long Beach	California	credit
0000002	06-17-2011	4000002	5.58	Exercise & Fitness	Weightlifting Machine Accessories	Anaheim	California	credit
0000003	06-14-2011	4000003	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin	credit
0000004	12-28-2011	4000002	98.81	Team Sports	Field Hockey	Nashville	Tennessee	credit
0000005	02-14-2011	4000004	193.63	Outdoor Recreation	Camping & Backpacking & Hiking	Chicago	Illinois	credit
0000006	10-17-2011	4000005	27.89	Puzzles	Jigsaw Puzzles	Charleston	South Carolina	credit
.....	.....	.....	.....	.....	.....	.....	.....	.....

Fig: transaction\_details

# Reduce side Join

## Map phase: Mapper for customer

- Read the input taking one tuple at a time.
- Tokenize each word in that tuple and fetch the cust ID along with the name of the person.
- The cust ID will be the key of the key-value pair that the mapper will generate eventually.
- Add a tag “cust” to indicate that this input tuple is of cust\_details type.
- Therefore, mapper for cust\_details will produce following intermediate key-value pair:

**Key – Value pair: [cust ID, cust      name]**

- Example: [4000001, cust    Kristina], [4000002, cust    Paige], etc.

# Reduce side Join

## Map phase: Mapper for transaction

- Fetch the amount value instead of name of the person.
- In this case, “tnxn” is used as a tag.
- Therefore, the cust ID will be the key of the key-value pair that the mapper will generate eventually.
- Finally, the output of mapper for transaction\_details will be of the following format:

**Key, Value Pair: [cust ID, tnxn amount]**

- Example: [4000001, tnxn 40.33], [4000002, tnxn 198.44], etc.

# Reduce side Join

## Sorting and Shuffling Phase

- The sorting and shuffling phase will generate an array list of values corresponding to each key. In other words, it will put together all the values corresponding to each unique key in the intermediate key-value pair. The output of sorting and shuffling phase will be of the following format:

**Key – list of Values:**

{cust ID1 – [(cust name1), (tnxn amount1), (tnxn amount2), (tnxn amount3),.....]}

{cust ID2 – [(cust name2), (tnxn amount1), (tnxn amount2), (tnxn amount3),.....]}

.....

- Example:

{4000001 – [(cust kristina), (tnxn 40.33), (tnxn 47.05),...]};

{4000002 – [(cust paige), (tnxn 198.44), (tnxn 5.58),...]};

.....

# Reduce side Join

## Reduce Phase

- The primary goal to perform this reduce-side join operation was to find out that how many times a particular customer has visited sports complex and the total amount spent by that very customer on different sports. Therefore, the final output should be of the following format:

**Key – Value pair: [Name of the customer] (Key) – [total amount, frequency of the visit] (Value)**

- Hence, the final output that my reducer will generate is given below:

**Kristina, 651.05 8**

**Paige, 706.97 6**

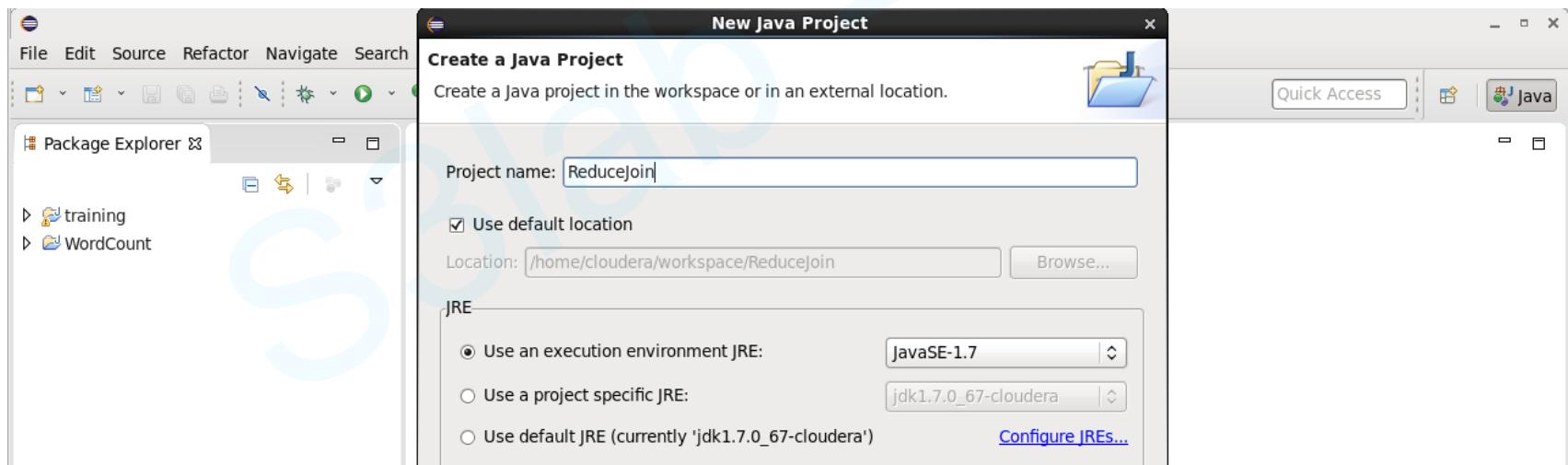
.....

# Reduce side Join

Reduce side Join tutorial starts from here

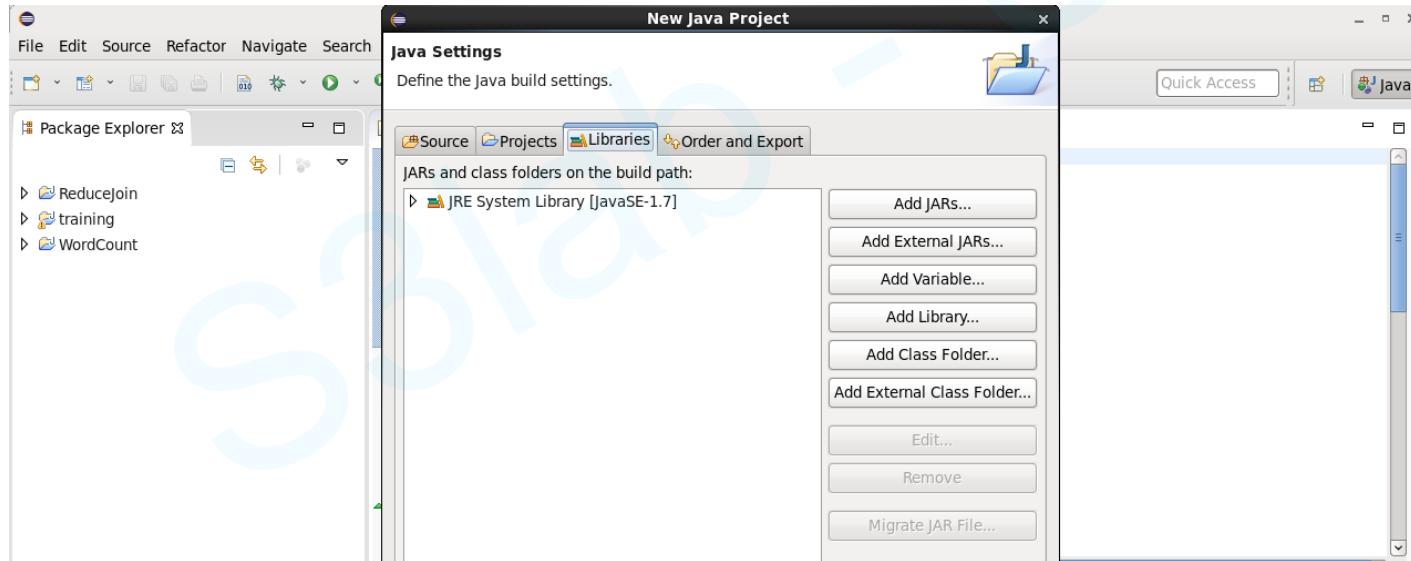
# Reduce side Join

- Create Java Project in Eclipse then click Next.



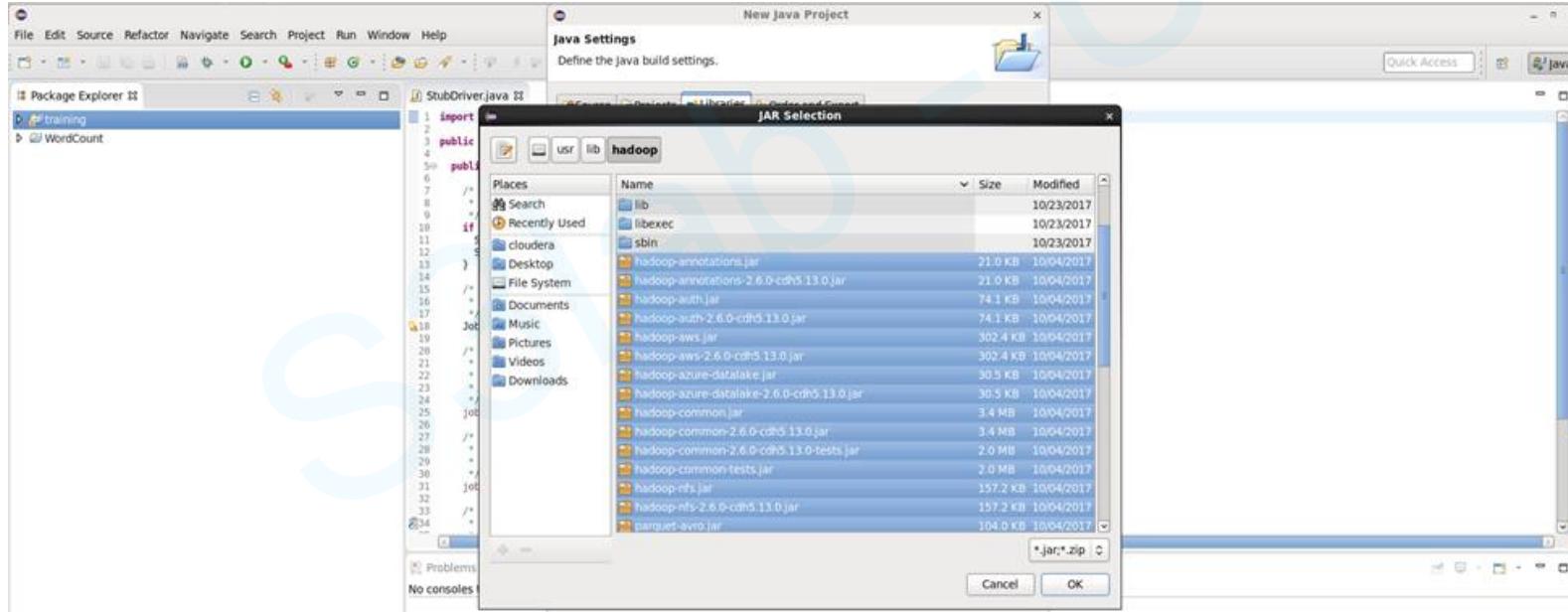
# Reduce side Join

- Add External JARs



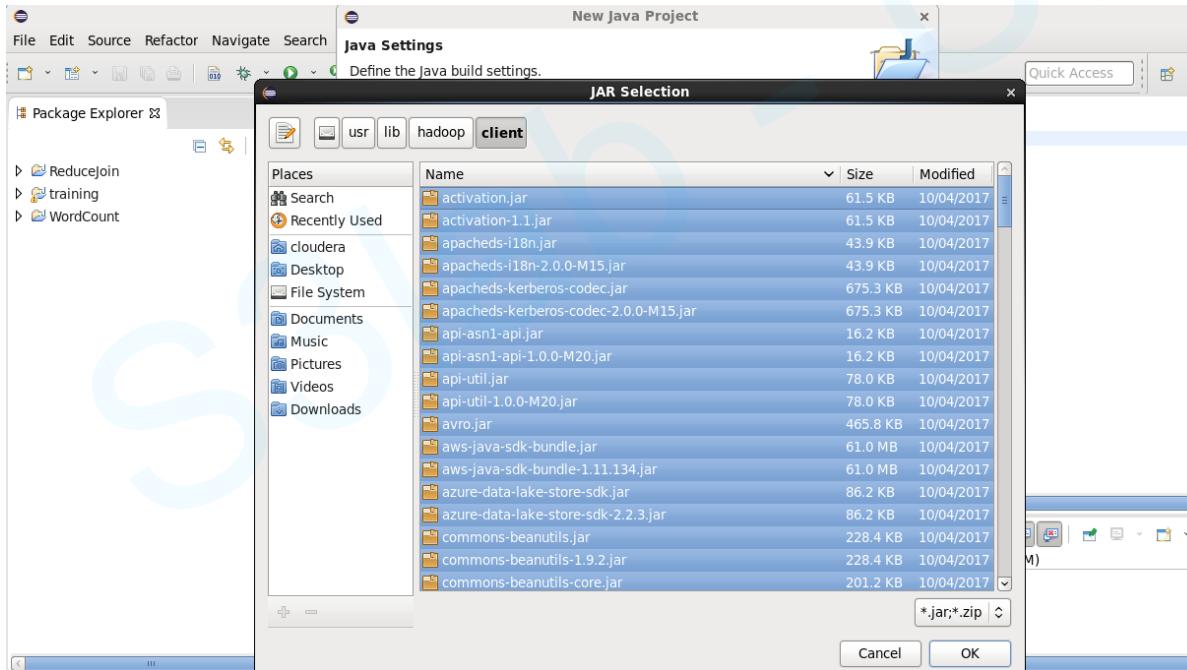
# Reduce side Join

- Add all .jar files in the folder **usr/lib/hadoop**



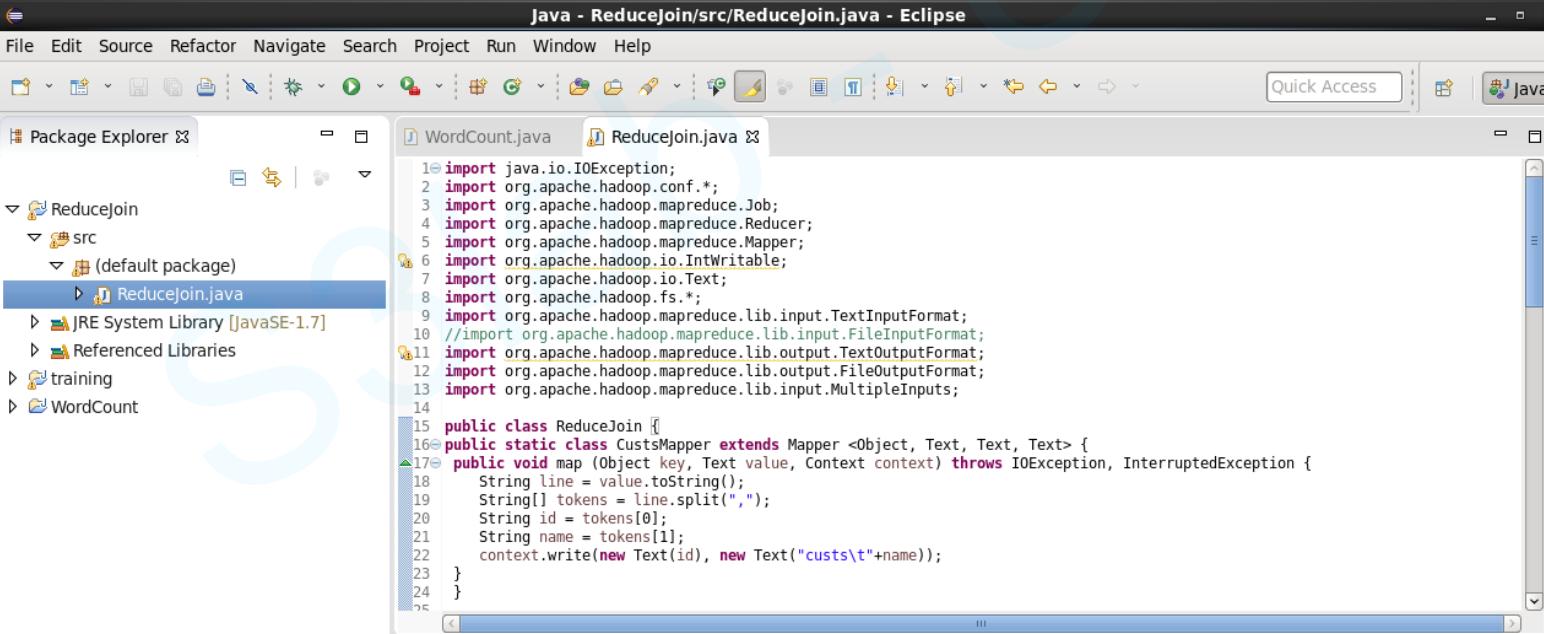
# Reduce side Join

- Add all .jar files in the folder **usr/lib/hadoop/client**



# Reduce side Join

- Create new class ReduceJoin

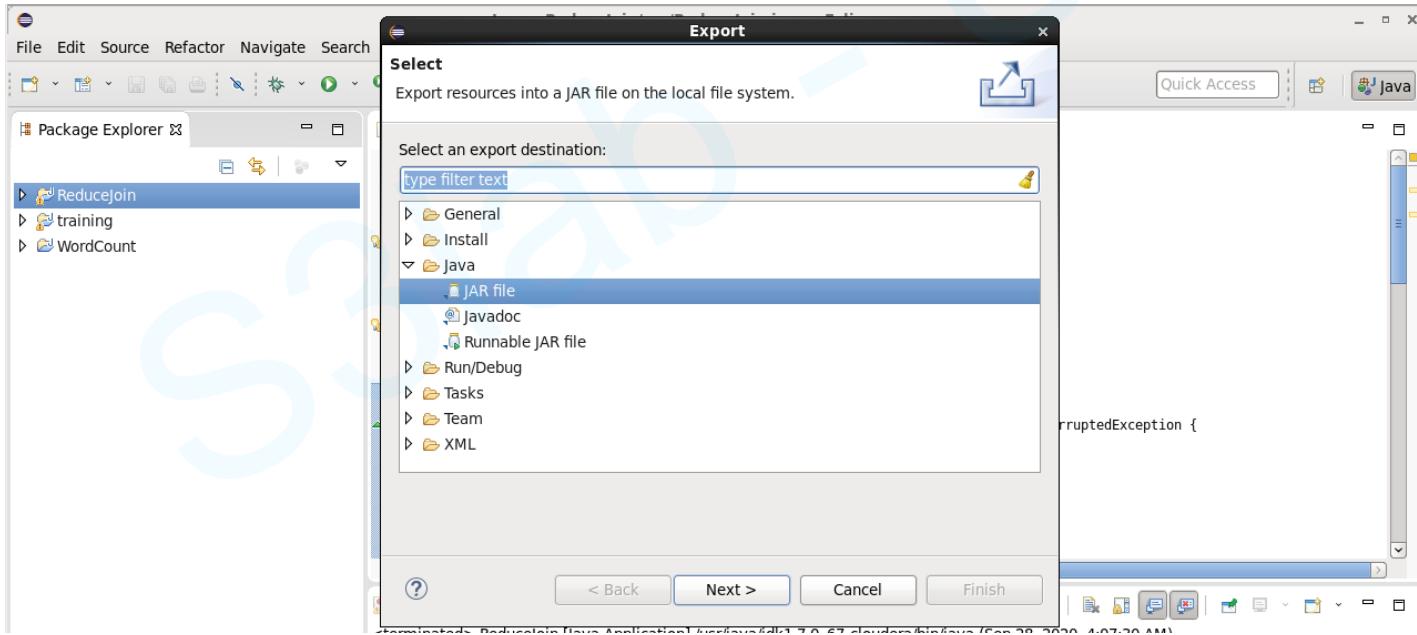


The screenshot shows the Eclipse IDE interface. The title bar reads "java - Reducejoin/src/ReduceJoin.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The Package Explorer view on the left shows a project structure with a package named "ReduceJoin" containing a source folder "src" with a file "ReduceJoin.java" selected. Other files in the package include "WordCount.java" and "WordCount.java". The Referenced Libraries and JRE System Library are also listed. The Java Editor view on the right displays the code for "ReduceJoin.java". The code imports several Hadoop classes and defines a custom mapper class "CustsMapper" that extends "Mapper". The "map" method takes an object key, a text value, and a context, and writes a new text output.

```
1 import java.io.IOException;
2 import org.apache.hadoop.conf.*;
3 import org.apache.hadoop.mapreduce.Job;
4 import org.apache.hadoop.mapreduce.Reducer;
5 import org.apache.hadoop.mapreduce.Mapper;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.fs.*;
9 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
10 //import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13 import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
14
15 public class ReduceJoin {
16     public static class CustsMapper extends Mapper <Object, Text, Text, Text> {
17         public void map (Object key, Text value, Context context) throws IOException, InterruptedException {
18             String line = value.toString();
19             String[] tokens = line.split(",");
20             String id = tokens[0];
21             String name = tokens[1];
22             context.write(new Text(id), new Text("custs\t"+name));
23         }
24     }
25 }
```

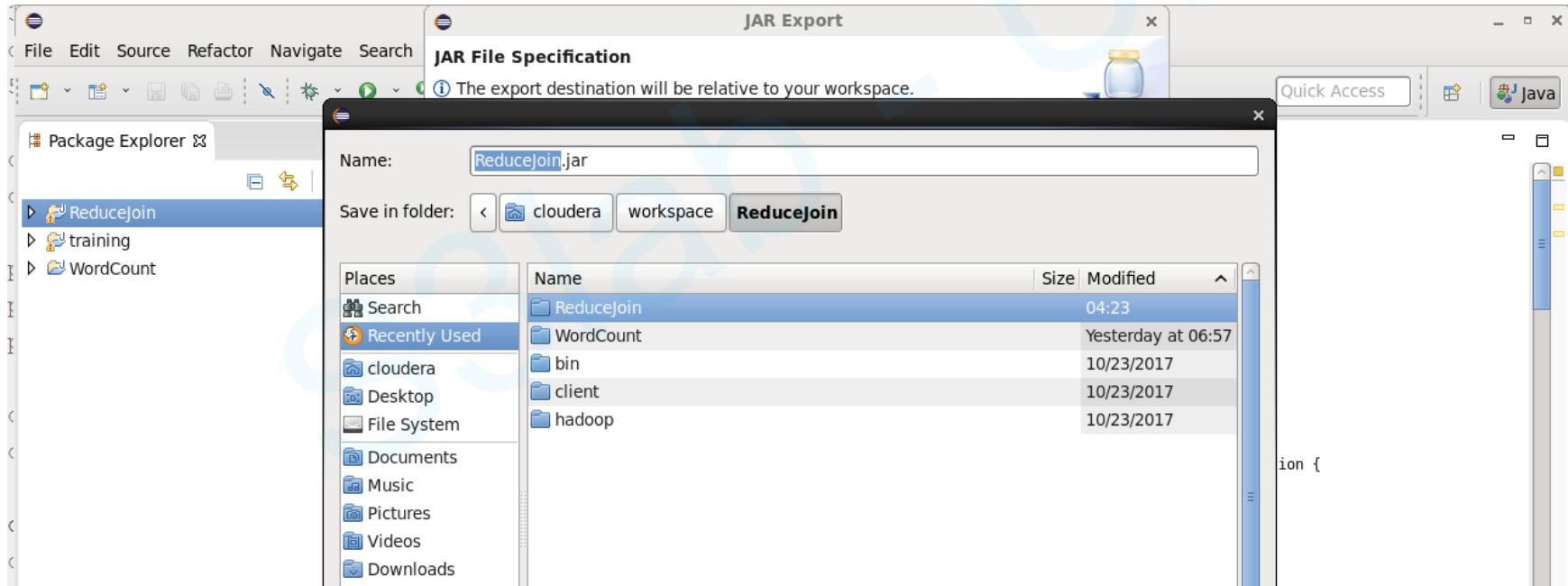
# Reduce side Join

- Export to JAR file



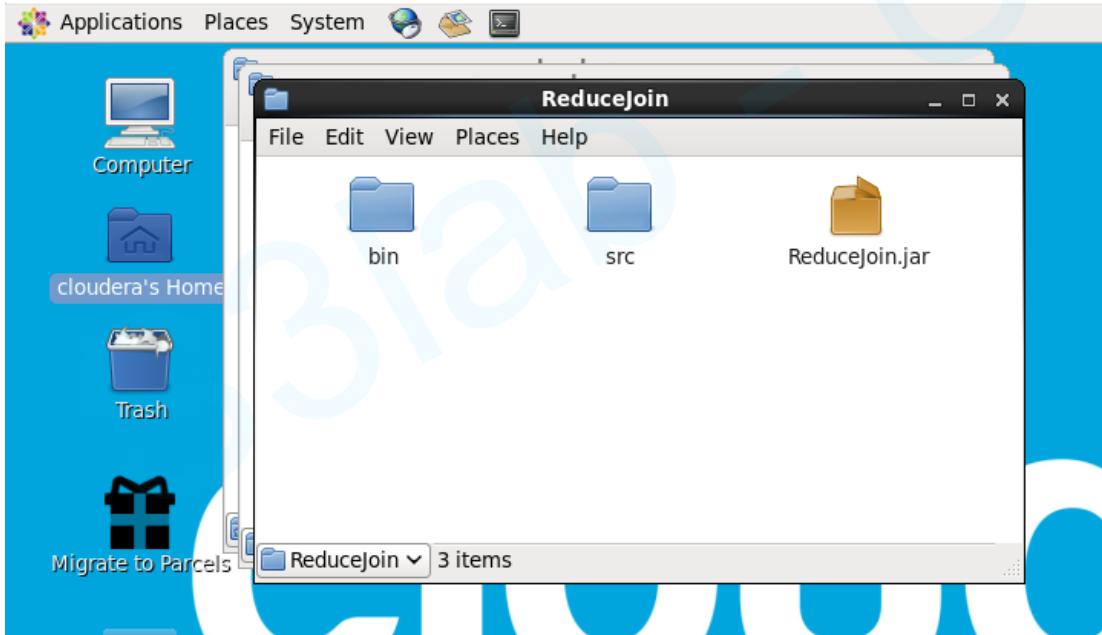
# Reduce side Join

- Name the file **ReduceJoin.jar** and place it inside **/home/cloudera/workspace/ReduceJoin**



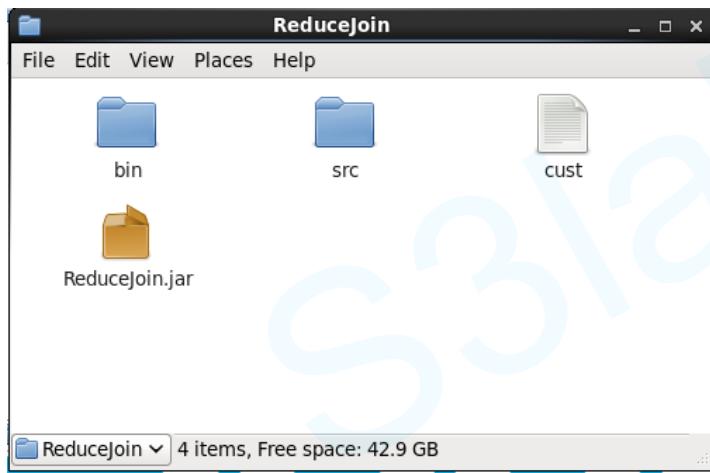
# Reduce side Join

- Check if the JAR file has been created



# Reduce side Join

- Create input file for customer inside `/home/cloudera/workspace/ReduceJoin`



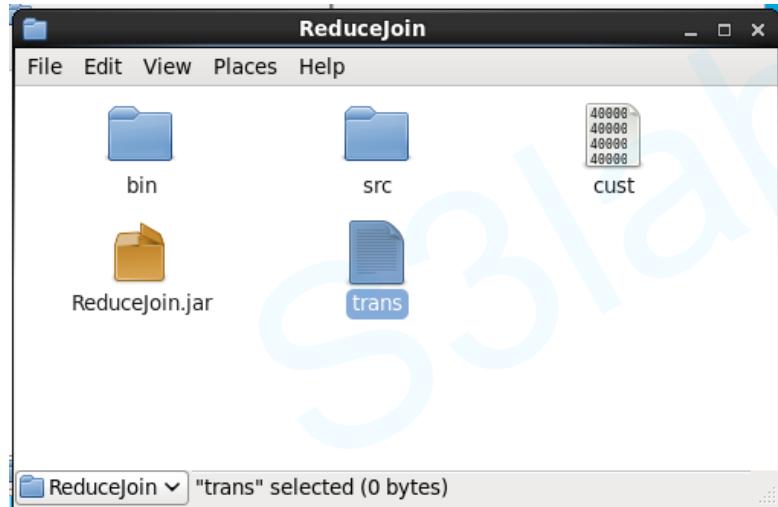
```
4000001,Kristina,Chung,55,Pilot
4000002,Paige,Chen,74,Teacher
4000003,Sherri,Melton,34,Firefighter
4000004,Gretchen,Hill,66,Computer hardware engineer
4000005,Karen,Puckett,74,Lawyer
4000006,Patrick,Song,42,Veterinarian
4000007,Elsie,Hamilton,43,Pilot
4000008,Hazel,Bender,63,Carpenter
4000009,Malcolm,Wagner,39,Artist
4000010,Dolores,McLaughlin,60,Writer|
```

A screenshot of a gedit text editor window titled "cust (~/workspace/ReduceJoin) - gedit". The window displays a list of customer data records in plain text format. The data consists of ten records, each containing a unique ID, name, last name, age, and profession. The status bar at the bottom of the window shows "Plain Text", "Tab Width: 8", "Ln 10, Col 37", and "INS".

ID	First Name	Last Name	Age	Profession
4000001	Kristina	Chung	55	Pilot
4000002	Paige	Chen	74	Teacher
4000003	Sherri	Melton	34	Firefighter
4000004	Gretchen	Hill	66	Computer hardware engineer
4000005	Karen	Puckett	74	Lawyer
4000006	Patrick	Song	42	Veterinarian
4000007	Elsie	Hamilton	43	Pilot
4000008	Hazel	Bender	63	Carpenter
4000009	Malcolm	Wagner	39	Artist
4000010	Dolores	McLaughlin	60	Writer

# Reduce side Join

- Create input file for transaction inside `/home/cloudera/workspace/ReduceJoin`



The screenshot shows a text editor window titled "\*trans (~/workspace/ReduceJoin) - gedit". The content of the file is as follows:

```
00000000,06-26-2011,4000001,040.33,Exercise & Fitness,Cardio Machine  
Accessories,Clarksville,Tennessee,credit  
00000001,05-26-2011,4000002,198.44,Exercise & Fitness,Weightlifting  
Gloves,Long Beach,California,credit  
00000002,06-01-2011,4000002,005.58,Exercise & Fitness,Weightlifting Machine  
Accessories,Anaheim,California,credit  
00000003,06-05-2011,4000003,198.19,Gymnastics,Gymnastics  
Rings,Milwaukee,Wisconsin,credit  
00000004,12-17-2011,4000002,098.81,Team Sports,Field  
Hockey,Nashville ,Tennessee,credit  
00000005,02-14-2011,4000004,193.63,Outdoor Recreation,Camping & Backpacking  
& Hiking,Chicago,Illinois,credit  
00000006,10-28-2011,4000005,027.89,Puzzles,Jigsaw Puzzles,Charleston,South  
Carolina,credit  
00000007,07-14-2011,4000006,096.01,Outdoor Play  
Equipment,Sandboxes,Columbus,Ohio,credit  
00000008,01-17-2011,4000006,010.44,Winter Sports,Snowmobiling,Des  
Moines,Iowa,credit  
00000009,05-17-2011,4000006,152.46,Jumping,Bungee Jumping,St.
```

The status bar at the bottom indicates: "Plain Text > Tab Width: 8 > Ln 60, Col 93 INS".

# Reduce side Join

- Create input directory in HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir ReduceJoin  
[cloudera@quickstart ~]$ hdfs dfs -mkdir ReduceJoin/input
```

# Reduce side Join

- Move input files to HDFS

```
[cloudera@quickstart ~]$ cd /home/cloudera/workspace/ReduceJoin
[cloudera@quickstart ReduceJoin]$ hdfs dfs -put cust ReduceJoin/input
[cloudera@quickstart ReduceJoin]$ hdfs dfs -put trans ReduceJoin/input
[cloudera@quickstart ReduceJoin]$ hdfs dfs -ls Reduce/input
ls: `Reduce/input': No such file or directory
[cloudera@quickstart ReduceJoin]$ hdfs dfs -ls ReduceJoin/input
Found 2 items
-rw-r--r--  1 cloudera cloudera      356 2020-09-28 04:45 ReduceJoin/input/cust
-rw-r--r--  1 cloudera cloudera    5383 2020-09-28 04:46 ReduceJoin/input/trans
[cloudera@quickstart ReduceJoin]$ █
```

# Reduce side Join

- Check the content of **cust** file

```
[cloudera@quickstart ReduceJoin]$ hdfs dfs -cat ReduceJoin/input/cust
4000001,Kristina,Chung,55,Pilot
4000002,Paige,Chen,74,Teacher
4000003,Sherri,Melton,34,Firefighter
4000004,Gretchen,Hill,66,Computer hardware engineer
4000005,Karen,Puckett,74,Lawyer
4000006,Patrick,Song,42,Veterinarian
4000007,Elsie,Hamilton,43,Pilot
4000008,Hazel,Bender,63,Carpenter
4000009,Malcolm,Wagner,39,Artist
4000010,Dolores,McLaughlin,60,Writer
[cloudera@quickstart ReduceJoin]$ █
```

# Reduce side Join

- Submit the job:

```
hadoop jar ReduceJoin.jar ReduceJoin ReduceJoin/input/cust ReduceJoin/input/trans ReduceJoin/output
```

```
[cloudera@quickstart ReduceJoin]$ hadoop jar ReduceJoin.jar ReduceJoin ReduceJoin/input/cust ReduceJoin/input/trans ReduceJoin/output
20/09/28 05:17:44 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/09/28 05:17:45 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
20/09/28 05:17:45 INFO input.FileInputFormat: Total input paths to process : 1
20/09/28 05:17:45 INFO input.FileInputFormat: Total input paths to process : 1
20/09/28 05:17:45 INFO mapreduce.JobSubmitter: number of splits:2
20/09/28 05:17:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1601275309549_0001
20/09/28 05:17:46 INFO impl.YarnClientImpl: Submitted application application_1601275309549_0001
20/09/28 05:17:46 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1601275309549_0001/
20/09/28 05:17:46 INFO mapreduce.Job: Running job: job_1601275309549_0001
20/09/28 05:17:54 INFO mapreduce.Job: Job job_1601275309549_0001 running in uber mode : false
20/09/28 05:17:54 INFO mapreduce.Job: map 0% reduce 0%
20/09/28 05:18:06 INFO mapreduce.Job: map 100% reduce 0%
20/09/28 05:18:14 INFO mapreduce.Job: map 100% reduce 100%
20/09/28 05:18:14 INFO mapreduce.Job: Job job_1601275309549_0001 completed successfully
```

# Reduce side Join

- Check the output directory in HDFS

```
[cloudera@quickstart ReduceJoin]$ hdfs dfs -ls ReduceJoin/output
Found 2 items
-rw-r--r--  1 cloudera cloudera          0 2020-09-28 05:18 ReduceJoin/output/_SUCCESS
-rw-r--r--  1 cloudera cloudera 214 2020-09-28 05:18 ReduceJoin/output/part-r-00000
[cloudera@quickstart ReduceJoin]$
```

# Reduce side Join

- Check the output directory in HDFS and the result inside it

```
[cloudera@quickstart ReduceJoin]$ hdfs dfs -ls ReduceJoin/output
Found 2 items
-rw-r--r-- 1 cloudera cloudera      0 2020-09-28 05:18 ReduceJoin/output/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 214 2020-09-28 05:18 ReduceJoin/output/part-r-00000
[cloudera@quickstart ReduceJoin]$ hdfs dfs -cat ReduceJoin/output/part*
Kristina      8 651.049999
Paige        6 706.970007
Sherri       3 527.589996
Gretchen     5 337.060005
Karen        5 325.150005
Patrick      5 539.380010
Elsie        6 699.550003
Hazel        10 859.419990
Malcolm      6 457.829996
Dolores      6 447.090004
[cloudera@quickstart ReduceJoin]$ █
```

# Assignment 2

- Write program to output
  - Min, Max, and Average age of users for each Game Type
  - The game types with lowest average age

File Input Format Counters

Bytes Read=665

File Output Format Counters

Bytes Written=633

```
[cloudera@quickstart ~]$ hdfs dfs -cat ReduceJoin/outputChain2/part*
Air Sports      [Min: 74 Max: 74 Avg: 74.0]
Combat Sports   [Min: 39 Max: 55 Avg: 47.0]
Exercise & Fitness [Min: 43 Max: 74 Avg: 61.2]
```

...

```
Team Sports     [Min: 43 Max: 74 Avg: 62.8]
```

```
Water Sports    [Min: 34 Max: 74 Avg: 53.285714285714285]
```

```
Winter Sports   [Min: 42 Max: 55 Avg: 48.5]
```

```
Game types with lowest average age of 47.0: Combat Sports, Gymnastics
```

```
[cloudera@quickstart ~]$
```

Cust ID	First Name	Last Name	Age	Profession
4000001	Kristina	Chung	55	Pilot
4000002	Paige	Chen	74	Teacher
4000003	Sherri	Melton	34	Firefighter
4000004	Gretchen	Hill	66	Engineer
.....	.....	.....	.....	.....

Fig: cust\_details

Trans ID	Date	Cust ID	Amount	Game Type	Equipment	City	State	Mode
0000000	06-26-2011	4000001	40.33	Exercise & Fitness	Cardio Machine Accessories	Clarksville	Tennessee	credit
0000001	05-05-2011	4000002	198.44	Exercise & Fitness	Weightlifting Gloves	Long Beach	California	credit
0000002	06-17-2011	4000002	5.58	Exercise & Fitness	Weightlifting Machine Accessories	Anaheim	California	credit
0000003	06-14-2011	4000003	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin	credit
0000004	12-28-2011	4000002	98.81	Team Sports	Field Hockey	Nashville	Tennessee	credit
0000005	02-14-2011	4000004	193.63	Outdoor Recreation	Camping & Backpacking & Hiking	Chicago	Illinois	credit
0000006	10-17-2011	4000005	27.89	Puzzles	Jigsaw Puzzles	Charleston	South Carolina	credit
.....	.....	.....	.....	.....	.....	.....	.....	.....

Fig: transaction\_details

Hint: you might need two MapReduce jobs for this task

# First MapReduce job

## Map

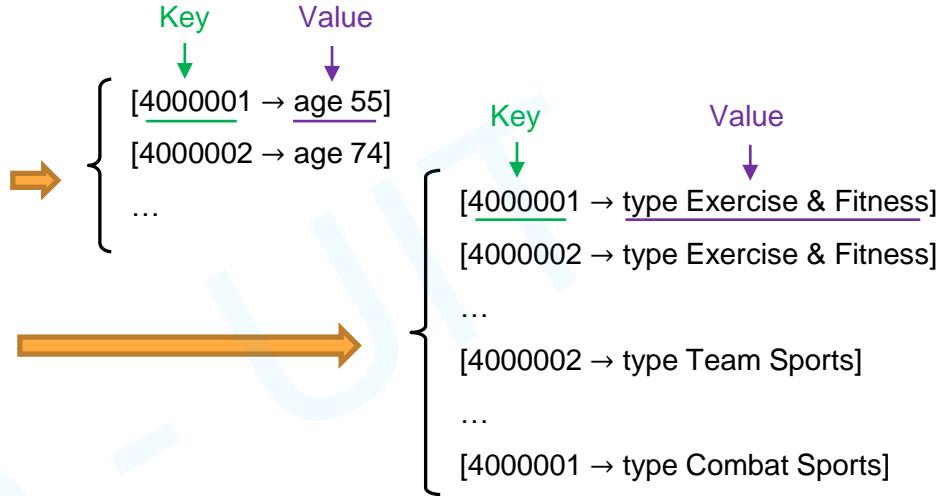
Cust ID	First Name	Last Name	Age	Profession
4000001	Kristina	Chung	55	Pilot
4000002	Paige	Chen	74	Teacher
4000003	Sherri	Melton	34	Firefighter
4000004	Gretchen	Hill	66	Engineer
.....	.....	.....	.....	.....

Fig: cust_details							
Trans ID	Date	Cust ID	Amount	Game Type	Equipment	City	State
0000000	06-26-2011	4000001	40.33	Exercise & Fitness	Cardio Machine Accessories	Clarksville	Tennessee
0000001	05-05-2011	4000002	198.44	Exercise & Fitness	Weightlifting Gloves Weightlifting Machine	Long Beach	California
0000002	06-17-2011	4000002	5.58	Exercise & Fitness	Accessories	Anaheim	California
0000003	06-14-2011	4000003	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin
0000004	12-28-2011	4000002	98.81	Team Sports	Field Hockey Camping & Backpacking	Nashville	Tennessee
0000005	02-14-2011	4000004	193.63	Outdoor Recreation	& Hiking	Chicago	Illinois
0000006	10-17-2011	4000005	27.89	Puzzles	Jigsaw Puzzles	Charleston	South Carolina
.....	.....	.....	.....	.....	.....	.....	.....

Fig: transaction_details							
Trans ID	Date	Cust ID	Amount	Game Type	Equipment	City	State
0000000	06-26-2011	4000001	40.33	Exercise & Fitness	Cardio Machine Accessories	Clarksville	Tennessee
0000001	05-05-2011	4000002	198.44	Exercise & Fitness	Weightlifting Gloves Weightlifting Machine	Long Beach	California
0000002	06-17-2011	4000002	5.58	Exercise & Fitness	Accessories	Anaheim	California
0000003	06-14-2011	4000003	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin
0000004	12-28-2011	4000002	98.81	Team Sports	Field Hockey Camping & Backpacking	Nashville	Tennessee
0000005	02-14-2011	4000004	193.63	Outdoor Recreation	& Hiking	Chicago	Illinois
0000006	10-17-2011	4000005	27.89	Puzzles	Jigsaw Puzzles	Charleston	South Carolina
.....	.....	.....	.....	.....	.....	.....	.....



## Sort and Shuffle

{4000001 → [(age 55) , (type Exercise & Fitness), (type Combat Sport), (type Water Sport), ..., (type Water Sport), ...]}

{4000002 → [(age 74) , (type Exercise & Fitness), (type Team Sports), ...]}

Sort and Shuffle will be done by the framework. Each of the line above will be the input for Reduce() function in the Reducer class. (i.e. the reducer will run the Reduce() function for each key).

## Reduce

Key                      Value  
↓                      ↓  
[55 → Exercise & Fitness, Combat Sport, Water Sport, ...]  
[74 → Exercise & Fitness, Team Sports, ...]

Note that there might be duplicate game types for each player. So remember to remove any duplicates.

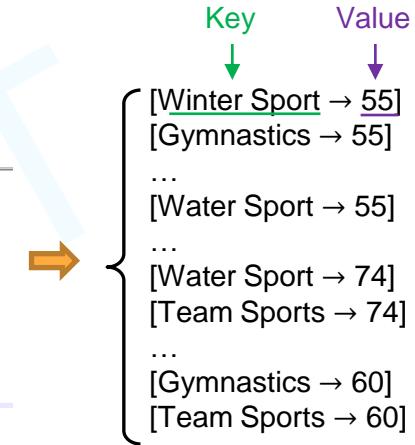
# Output of the first MapReduce job

```
[cloudera@quickstart ~]$ hdfs dfs -cat ReduceJoin/output/part*
55      Winter Sports,Gymnastics,Exercise & Fitness,Water Sports,Outdoor Recreation,Combat Sports,
74      Water Sports,Exercise & Fitness,Team Sports,Outdoor Recreation,
...
39      Outdoor Play Equipment,Indoor Games,Water Sports,Gymnastics,Combat Sports,
60      Gymnastics,Jumping,Exercise & Fitness,Games,Team Sports,
[cloudera@quickstart ~]$ █
```

# Second MapReduce job

## Map

```
55 → Winter · Sports, Gymnastics, Exercise & Fitness, Water · Sports, Outdoor · Recreation, Combat · Sports,  
74 → Water · Sports, Exercise & Fitness, Team · Sports, Outdoor · Recreation,  
34 → Water · Sports, Gymnastics, Outdoor · Recreation,  
...  
39 → Outdoor · Play · Equipment, Indoor · Games, Water · Sports, Gymnastics, Combat · Sports,  
60 → Gymnastics, Jumping, Exercise & Fitness, Games, Team · Sports,
```



## Sort and Shuffle

{Winter Sport → [55,...]}

{Water Sport → [55,74,...]}

{Gymnastics → [55,...,60]}

...

## Reduce

Key  
↓  
[Winter Sport → Min: 42 Max: 55 Avg: 58]  
Value  
↓

[Water Sport → Min: 34 Max: 74 Avg: 53.29]

...

# Output of the second MapReduce job

Run the second MapReduce job which takes the output of the first job as input

```
[cloudera@quickstart ~]$ hdfs dfs -cat ReduceJoin/outputjob2/part*
Air Sports      [Min: 74 Max: 74 Avg: 74.0]
Combat Sports   [Min: 39 Max: 55 Avg: 47.0]
Exercise & Fitness   [Min: 43 Max: 74 Ava: 61.21]
...
Team Sports     [Min: 43 Max: 74 Avg: 62.8]
Water Sports    [Min: 34 Max: 74 Avg: 53.285714285714285]
Winter Sports   [Min: 42 Max: 55 Avg: 48.5]
[cloudera@quickstart ~]$ █
```

# Output types with lowest average age (2<sup>nd</sup> job)

Set the number of Reducer to 1, keep track the Game Type with lowest average age, then write it to the output in **cleanup** function, which will run after all Reduce() function will have finished.

```
File Input Format Counters
    Bytes Read=665
File Output Format Counters
    Bytes Written=633
[clooudera@quickstart ~]$ hdfs dfs -cat ReduceJoin/outputChain2/part*
Air Sports      [Min: 74 Max: 74 Avg: 74.0]
Combat Sports   [Min: 39 Max: 55 Avg: 47.0]
Exercise & Fitness [Min: 43 Max: 74 Avg: 61.2]
Games          [Min: 60 Max: 63 Avg: 61.5]

...
Water Sports    [Min: 34 Max: 74 Avg: 53.285714285714285]
Winter Sports   [Min: 42 Max: 55 Avg: 48.5]
Game types with lowest average age of  47.0: Combat Sports, Gymnastics ←
[clooudera@quickstart ~]$ █
```

# Job chain

```
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
13 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15
16 public class ReduceJoinChain {
17     public static class CustsMapper extends Mapper<Object, Text, Text, Text>
18     {
19         public static class TxnsMapper extends Mapper<Object, Text, Text, Text>
20         {
21             public static class ReduceJoinReducer extends Reducer<Text, Text, Text, Text>
22             {
23                 public static class GameTypeStatMapper extends Mapper<Object, Text, Text, Text>
24                 {
25                     public static class GameTypeStatReducer extends Reducer<Text, Text, Text, Text>
26                     {
27                         public static void main(String[] args) throws Exception {
28                             Configuration conf = new Configuration();
29                             Job job = new Job(conf, "Reduce-side-join-job-1");
30                             job.setJarByClass(ReduceJoinChain.class);
31                             job.setReducerClass(ReduceJoinReducer.class);
32                             job.setOutputKeyClass(Text.class);
33                             job.setOutputValueClass(Text.class);
34                             MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, CustsMapper.class);
35                             MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, TxnsMapper.class);
36                             Path outputPath = new Path(args[2]);
37                             FileOutputFormat.setOutputPath(job, outputPath);
38                             outputPath.getFileSystem(conf).delete(outputPath);
39                             job.waitForCompletion(true);
40
41                             Configuration conf2 = new Configuration();
42                             Job job2 = Job.getInstance(conf2, "Reduce-side-join-job-2");
43                             job2.setJarByClass(ReduceJoinChain.class);
44                             job2.setMapperClass(GameTypeStatMapper.class);
45                             job2.setReducerClass(GameTypeStatReducer.class);
46                             job2.setOutputKeyClass(Text.class);
47                             job2.setOutputValueClass(Text.class);
48                             job2.setNumReduceTasks(1);
49                             FileInputFormat.addInputPath(job2, new Path(args[2]));
50                             FileOutputFormat.setOutputPath(job2, new Path(args[3]));
51                             System.exit(job2.waitForCompletion(true) ? 0 : 1);
52
53                         }
54                     }
55                 }
56             }
57         }
58     }
59 }
```

Instead of running two separate jobs, we can also run a job chain

# Job chain

Instead of running two separate jobs, we can also run a job chain

```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/workspace/ReduceJoinChain.jar ReduceJoinChain ReduceJoin/input/cust ReduceJoin/input/trans ReduceJoin/outputChain1 ReduceJoin/outputChain2  
21/03/31 02:24:29 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032  
21/03/31 02:24:29 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.  
21/03/31 02:24:29 INFO input.FileInputFormat: Total input paths to process : 1  
21/03/31 02:24:29 INFO input.FileInputFormat: Total input paths to process : 1  
21/03/31 02:24:29 WARN hdfs.DFSClient: Caught exception  
java.lang.InterruptedException  
    at java.lang.Object.wait(Native Method)  
    at java.lang.Thread.join(Thread.java:1281)  
    at java.lang.Thread.join(Thread.java:1355)  
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)  
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)  
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)  
21/03/31 02:24:30 INFO mapreduce.JobSubmitter: number of splits:2  
21/03/31 02:24:30 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1617150354712_0023  
21/03/31 02:24:30 INFO impl.YarnClientImpl: Submitted application application_1617150354712_0023
```

# Job chain

```
[cloudera@quickstart ~]$ hdfs dfs -cat ReduceJoin/outputChain1/part*
55      Winter Sports,Gymnastics,Exercise & Fitness,Water Sports,Outdoor Recreation,Combat Sports,
74      Water Sports,Exercise & Fitness,Team Sports,Outdoor Recreation,
34      Water Sports,Gymnastics,Outdoor Recreation,
66      Outdoor Recreation,Water Sports,Indoor Games,
```

...

```
[cloudera@quickstart ~]$ hdfs dfs -cat ReduceJoin/outputChain2/part*
Air Sports      [Min: 74 Max: 74 Avg: 74.0]
Combat Sports   [Min: 39 Max: 55 Avg: 47.0]
Exercise & Fitness   [Min: 43 Max: 74 Avg: 61.2]
Games      [Min: 60 Max: 63 Avg: 61.5]
```

...

```
Water Sports    [Min: 34 Max: 74 Avg: 53.285714285714285]
Winter Sports   [Min: 42 Max: 55 Avg: 48.5]
Game types with lowest average age of 47.0: Combat Sports, Gymnastics
[cloudera@quickstart ~]$ █
```

# Hadoop Streaming:

---

Writing A Hadoop MapReduce Program In Python

# Hadoop Streaming:

## What is Hadoop Streaming?

- Hadoop Streaming is a utility that comes with the Hadoop distribution. It can be used to execute programs for big data analysis. Hadoop streaming can be performed using languages like Python, Java, PHP, Scala, Perl, UNIX, and many more. The utility allows us to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer. For example:

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar
```

```
-input myInputDirs
```

```
-output myOutputDir
```

```
-mapper /bin/cat
```

```
-reducer /bin/wc
```

# Hadoop Streaming:

## What is Hadoop Streaming?

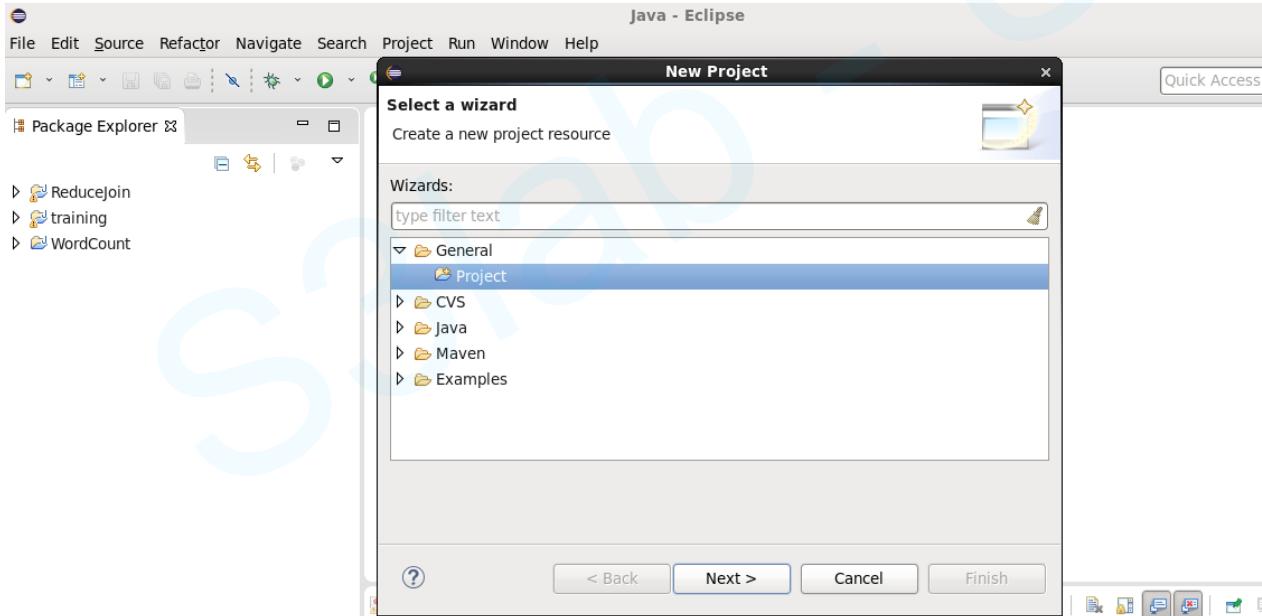
Parameter	Optional / Required	Description
-input directoryname or filename	Required	Input location for mapper
-output directoryname or filename	Required	Output location for mapper
-mapper executable	Required	Mapper executable
-reducer executable	Required	Reducer executable
-file filename	Optional	Make the mapper, reducer, or combiner executable available locally on the compute nodes

# Hadoop Streaming:

Hadoop streaming tutorial start from here

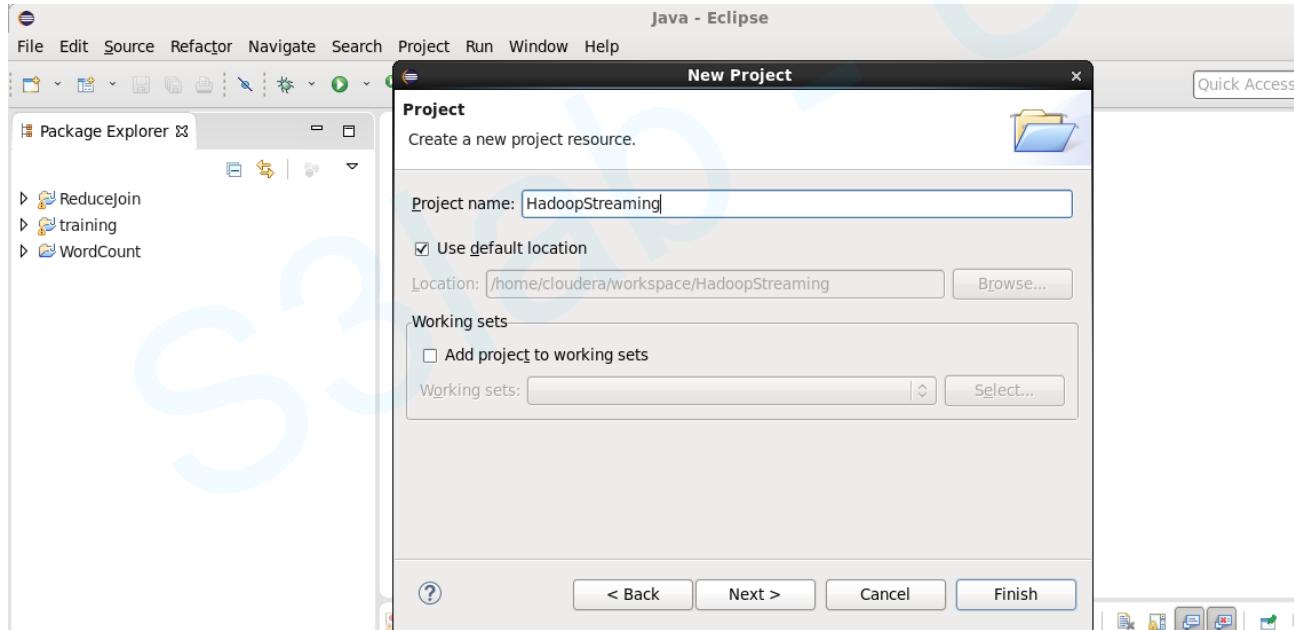
# Hadoop Streaming:

- Create new General Project in Eclipse



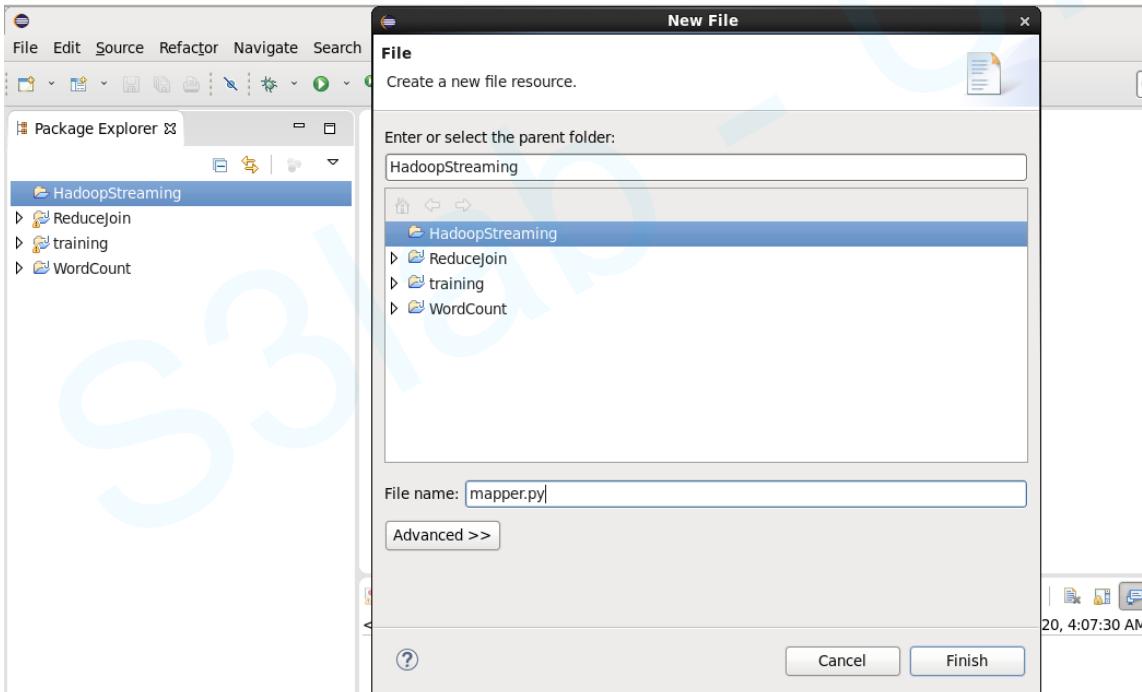
# Hadoop Streaming:

- Name the project HadoopStreaming



# Hadoop Streaming:

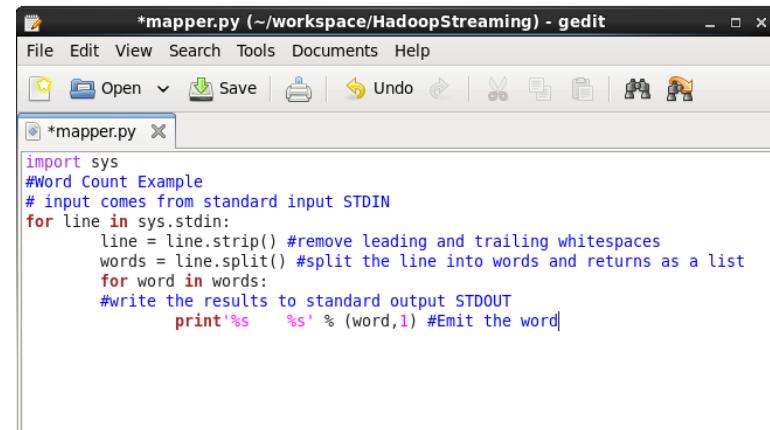
- Create new file **mapper.py**



# Hadoop Streaming:

## mapper.py

```
import sys
#Word Count Example
# input comes from standard input STDIN
for line in sys.stdin:
    line = line.strip() #remove leading and trailing whitespaces
    words = line.split() #split the line into words and returns as a list
    for word in words:
        #write the results to standard output STDOUT
        print '%s\t%s' % (word,1) #Emit the word
```

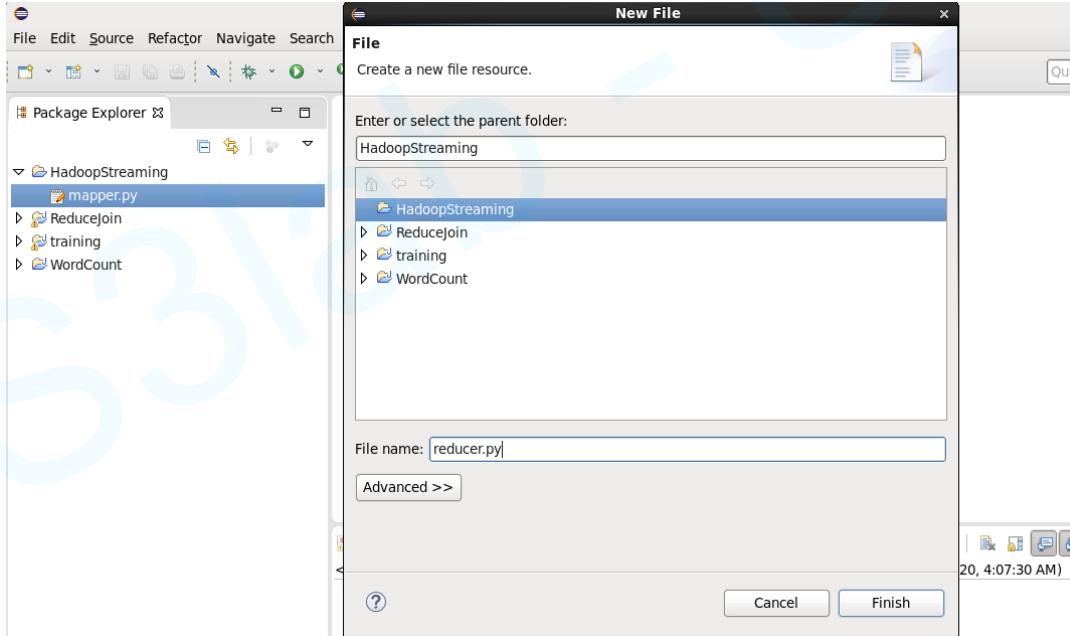


The screenshot shows a window titled "mapper.py (~/workspace/HadoopStreaming) - gedit". The window contains a text editor with the following Python code:

```
import sys
#Word Count Example
# input comes from standard input STDIN
for line in sys.stdin:
    line = line.strip() #remove leading and trailing whitespaces
    words = line.split() #split the line into words and returns as a list
    for word in words:
        #write the results to standard output STDOUT
        print '%s\t%s' % (word,1) #Emit the word
```

# Hadoop Streaming:

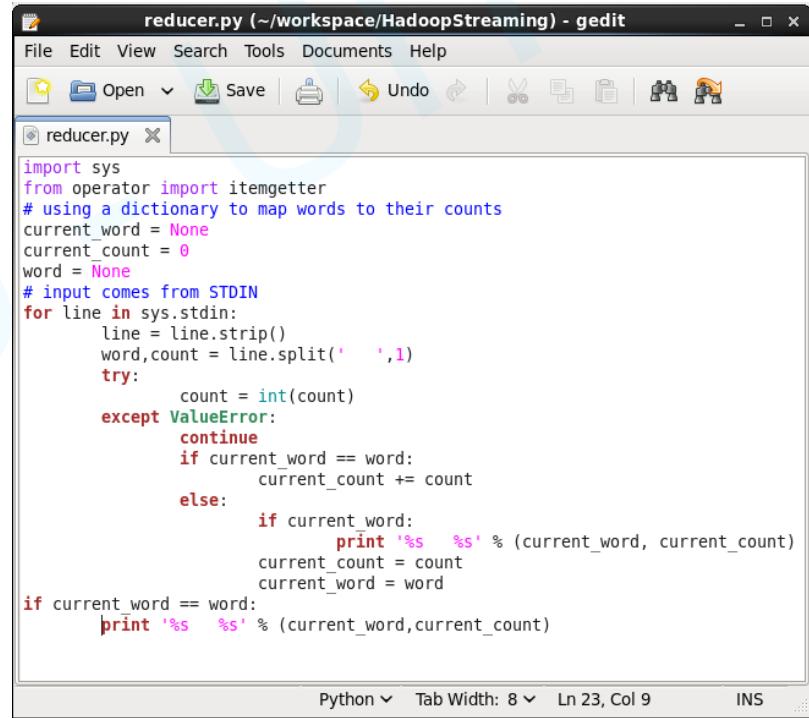
- Create new file **reducer.py**



# Hadoop Streaming:

## reducer.py

```
import sys
from operator import itemgetter
# using a dictionary to map words to their counts
current_word = None
current_count = 0
word = None
# input comes from STDIN
for line in sys.stdin:
    line = line.strip()
    word, count = line.split(' ', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s %s' % (current_word, current_count)
        current_count = count
        current_word = word
    if current_word == word:
        print '%s %s' % (current_word, current_count)
```



```
reducer.py (~/workspace/HadoopStreaming) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Cut Copy Paste Find Select
reducer.py
import sys
from operator import itemgetter
# using a dictionary to map words to their counts
current_word = None
current_count = 0
word = None
# input comes from STDIN
for line in sys.stdin:
    line = line.strip()
    word, count = line.split(' ', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s %s' % (current_word, current_count)
        current_count = count
        current_word = word
    if current_word == word:
        print '%s %s' % (current_word, current_count)

Python v Tab Width: 8 v Ln 23, Col 9      INS
```

# Hadoop Streaming:

- Locate the **HadoopStreaming** directory which contains **mapper.py** and **reducer.py**
- Create **word.txt** file

```
[cloudera@quickstart ~]$ cd /home/cloudera/workspace/HadoopStreaming  
[cloudera@quickstart HadoopStreaming]$ echo 'Cat mouse lion deer Tiger lion Elephant lion deer' > word.txt
```

# Hadoop Streaming:

- We can run mapper and reducer on local files (ex: word.txt). In order to run the Map and reduce on the Hadoop Distributed File System (HDFS), we need the Hadoop Streaming jar. So before we run the scripts on HDFS, let's run them locally to ensure that they are working fine.
- command: **cat word.txt | python mapper.py**

```
[cloudera@quickstart HadoopStreaming]$ cat word.txt | python mapper.py
Cat      1
mouse    1
lion     1
deer     1
Tiger    1
lion     1
Elephant 1
lion     1
deer     1
[cloudera@quickstart HadoopStreaming]$
```

# Hadoop Streaming:

- Run reducer.py
- command: `cat word.txt | python mapper.py | sort -k1,1 | python reducer.py`

```
[cloudera@quickstart HadoopStreaming]$ cat word.txt | python mapper.py | sort -k1,1 | python reducer.py
Cat    1
deer   2
Elephant 1
lion   3
mouse  1
Tiger  1
[cloudera@quickstart HadoopStreaming]$
```

# Hadoop Streaming:

Running the Python Code on Hadoop

- Create **HadoopStreaming** directory in HDFS
- Move **word.txt** to HDFS

```
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -mkdir HadoopStreaming  
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -put word.txt HadoopStreaming
```

# Hadoop Streaming:

Running the Python Code on Hadoop

- So locate the Hadoop Streaming jar on your terminal and copy the path.
- Command: **ls /usr/lib/hadoop-mapreduce/hadoop-streaming.jar**

```
[cloudera@quickstart HadoopStreaming]$ ls /usr/lib/hadoop-mapreduce/hadoop-streaming.jar  
/usr/lib/hadoop-mapreduce/hadoop-streaming.jar  
[cloudera@quickstart HadoopStreaming]$ █
```

# Hadoop Streaming:

## Run the MapReduce job

- Command: `hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -file /home/cloudera/workspace/HadoopStreaming/mapper.py -mapper 'python mapper.py' -file /home/cloudera/workspace/HadoopStreaming/reducer.py -reducer 'python reducer.py' -input HadoopStreaming/word.txt -output HSOutput`

```
[cloudera@quickstart ~]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -file /home/cloudera/workspace/HadoopStreaming/mapper.py -mapper 'python mapper.py' -file /home/cloudera/workspace/HadoopStreaming/reducer.py -reducer 'python reducer.py' -input HadoopStreaming/word.txt -output HSOutput
20/09/28 23:25:12 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/cloudera/workspace/HadoopStreaming/mapper.py, /home/cloudera/workspace/HadoopStreaming/reducer.py] [/usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.13.0.jar] /tmp/streamjob3457884780733202883.jar tm
pDir=null
20/09/28 23:25:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/09/28 23:25:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/09/28 23:25:14 INFO mapred.FileInputFormat: Total input paths to process : 1
20/09/28 23:25:14 INFO mapreduce.JobSubmitter: number of splits:2
20/09/28 23:25:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1601358520423_0002
```

Note: If we navigate to **HadoopStreaming** directory, we don't need to specify the full path `/home/cloudera/workspace/HadoopStreaming/mapper.py`

```
[cloudera@quickstart HadoopStreaming]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -file mapper.py -mapper 'python mapper.py' -file reducer.py -reducer 'python reducer.py' -input HadoopStreaming/word.txt -output HSOutput
```

# Hadoop Streaming:

- Hadoop provides a basic web interface for statistics and information

The screenshot shows a Mozilla Firefox browser window displaying the Hadoop MapReduce Job web interface. The title bar reads "MapReduce Job job\_1601275309549\_0011 - Mozilla Firefox". The address bar shows the URL "quickstart.cloudera:19888/jobhistory/job/job\_1601275309549\_0011". The page header includes links for Cloudera, Hue, Hadoop, HBase, Impala, Spark, Solr, Oozie, Cloudera Manager, and Getting Started. The main content area is titled "MapReduce Job job\_1601275309549\_0011".  
**Job Overview:**  
Job Name: streamjob2312088283050083692.jar  
User Name: hdfs  
Queue: root.hdfs  
State: SUCCEEDED  
Uberized: false  
Submitted: Mon Sep 28 10:18:48 PDT 2020  
Started: Mon Sep 28 10:18:52 PDT 2020  
Finished: Mon Sep 28 10:19:07 PDT 2020  
Elapsed: 14sec  
**Diagnostics:**  
Average Map Time 5sec  
Average Shuffle Time 3sec  
Average Merge Time 0sec  
Average Reduce Time 0sec  
**ApplicationMaster:**

Attempt Number	Start Time	Node	Logs
1	Mon Sep 28 10:18:49 PDT 2020	quickstart.cloudera:8042	logs

  
**Task Metrics:**

Task Type	Total	Complete
Map	2	2
Reduce	1	1

  
**Attempt Metrics:**

Attempt Type	Failed	Killed	Successful
Maps	0	0	2
Reduces	0	0	1

At the bottom right, there are buttons for "Activate Windows" and "Go to Settings to activate Windows".

# Hadoop Streaming:

- Check the output directory and the content of the result file

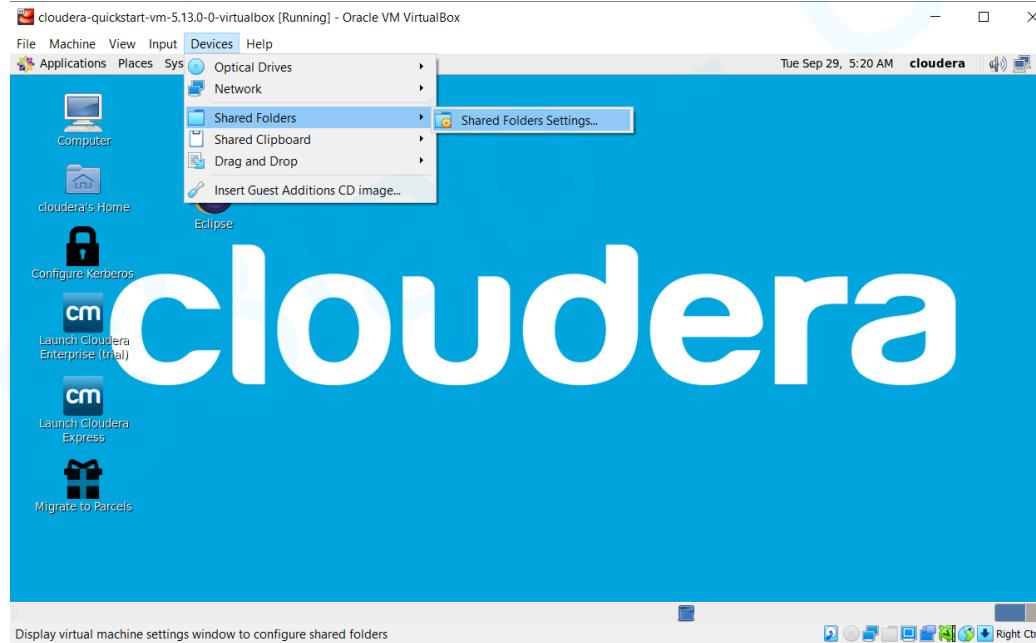
```
Bytes Read=75
File Output Format Counters
Bytes Written=65
20/09/28 10:19:08 INFO streaming.StreamJob: Output directory: /user/cloudera/HSOutput
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -ls
Found 7 items
drwxr-xr-x  - cloudera   cloudera      0 2020-09-28 10:19 HSOutput
drwxr-xr-x  - cloudera   cloudera      0 2020-09-28 10:03 HadoopStreaming
drwxr-xr-x  - cloudera   cloudera      0 2020-09-28 05:17 ReduceJoin
drwxr-xr-x  - cloudera   cloudera      0 2020-09-26 06:02 dataset
drwxr-xr-x  - cloudera   cloudera      0 2020-09-27 07:07 inputWC
drwxr-xr-x  - cloudera   cloudera      0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera   cloudera      0 2020-09-16 07:29 student
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -ls HSOutput
Found 2 items
-rw-r--r--  1 hdfs cloudera      0 2020-09-28 10:19 HSOutput/_SUCCESS
-rw-r--r--  1 hdfs cloudera    65 2020-09-28 10:19 HSOutput/part-00000
[cloudera@quickstart HadoopStreaming]$ hdfs dfs -cat HSOutput/part*
Cat 1
Elephant 1
Tiger 1
deer 2
lion 3
mouse 1
[cloudera@quickstart HadoopStreaming]$ █
```

# Share Folders between VM and Host

- Just in case you're finding the way to transfer files from Host to VM

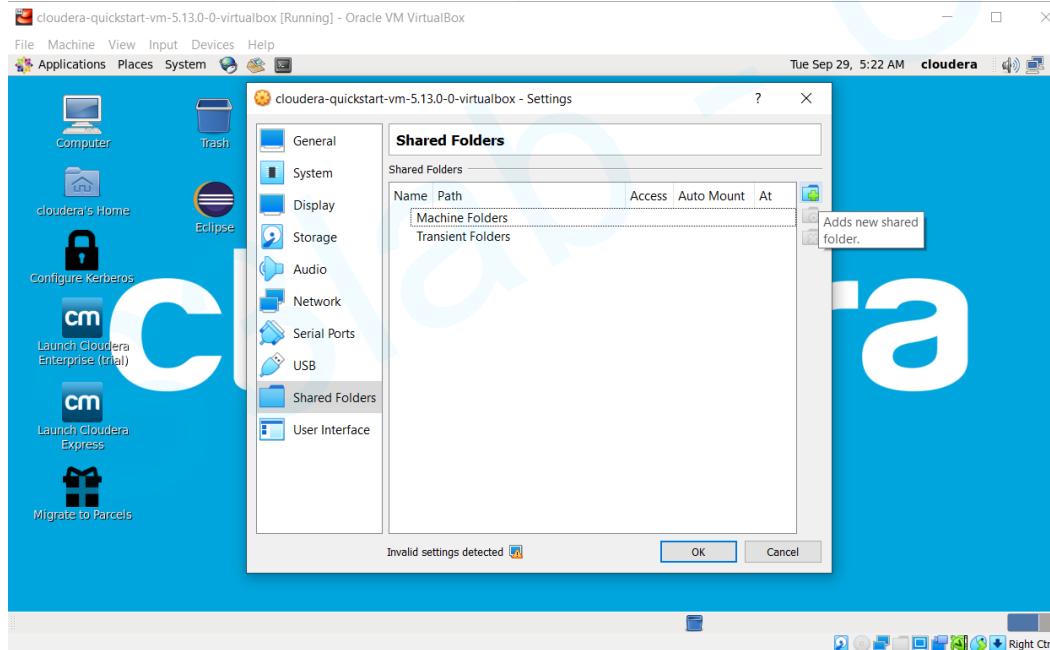
# Share Folders between VM and Host

- Device -> Shared Folders -> Shared Folders Setting



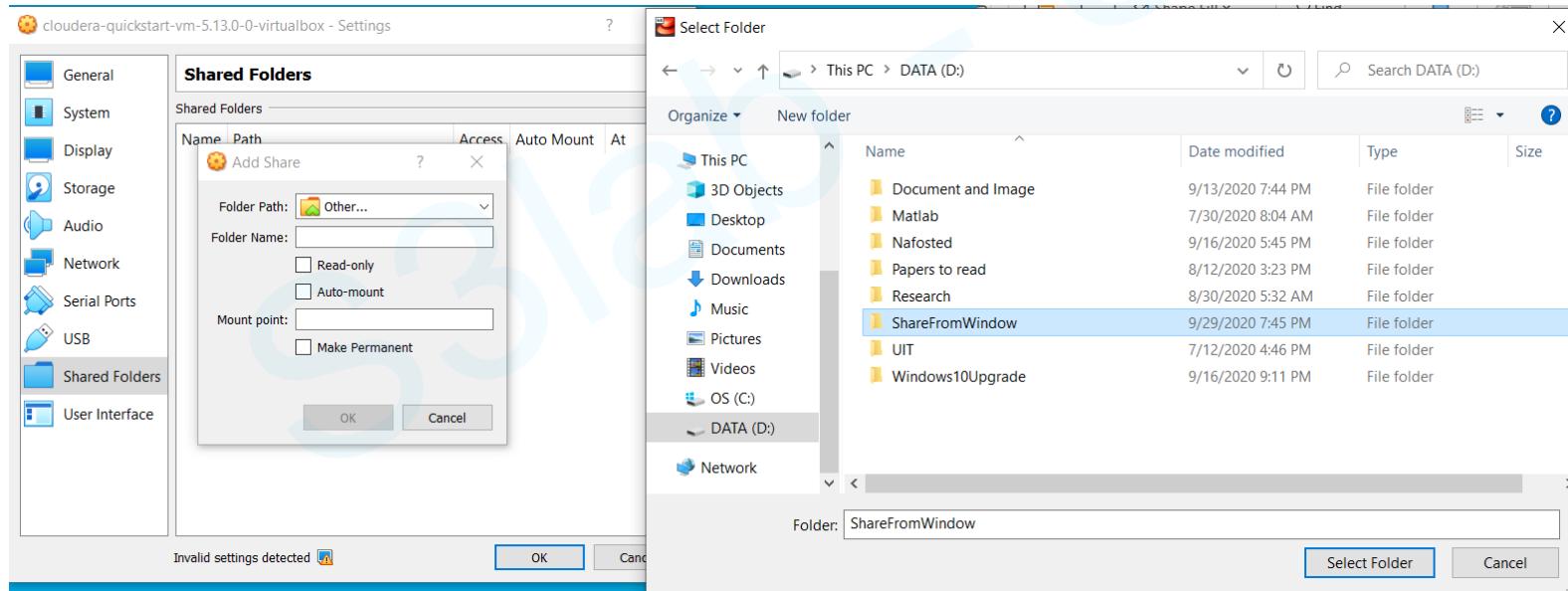
# Share Folders between VM and Host

- Click Adds new shared folder



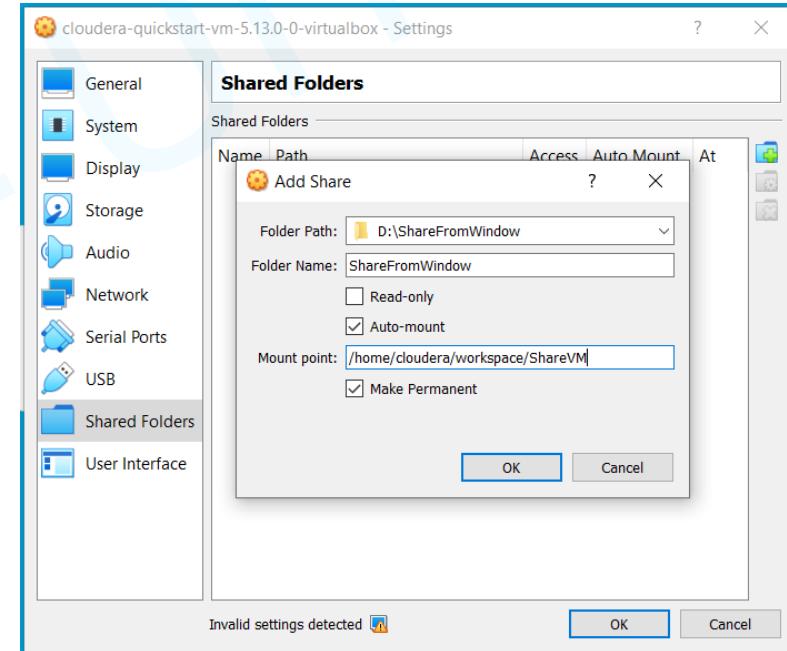
# Share Folders between VM and Host

- Navigate to any folder on Window



# Share Folders between VM and Host

- **Folder Path** - folder in the host (Windows 10)
- **Folder Name** - repeat the folder name from above
- **Auto-mount** - check this option
- **Mount point** - guest OS folder where the shared folder will mount (will be created if it does not exist)
- **Make Permanent** - check this option



# Share Folders between VM and Host

- The ShareVM folder will be created (if not exist) after you reboot the VM.
- You can create the **ShareVM** folder in guest OS (Cloudera) by yourself (so no need to reboot)
- Command: **mkdir /home/cloudera/workspace/ShareVM**

```
[cloudera@quickstart ~]$ ls /home/cloudera
cloudera-manager          Downloads           lib      student.java
cm_api.py                  eclipse            Music    Templates
codegen_categories.java    enterprise-deployment.json parcels  Videos
Desktop                   express-deployment.json Pictures workspace
Documents                 kerberos           Public
[cloudera@quickstart ~]$ ls /home/cloudera/workspace
training
[cloudera@quickstart ~]$ mkdir /home/cloudera/workspace/ShareVM
```

# Share Folders between VM and Host

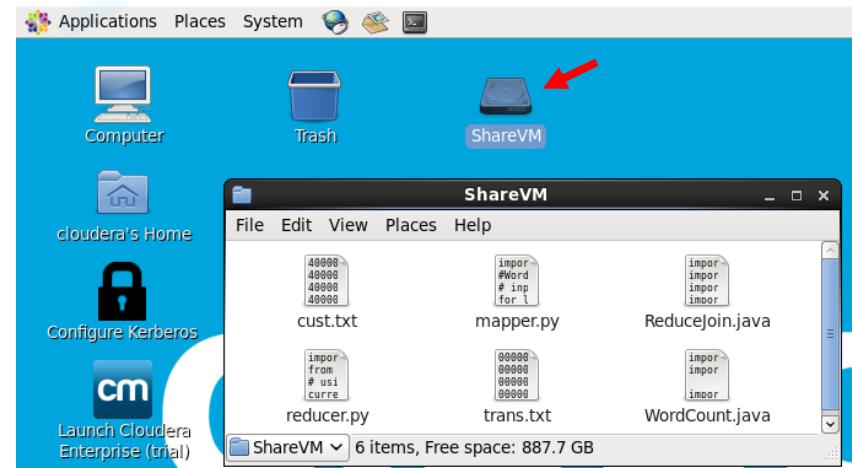
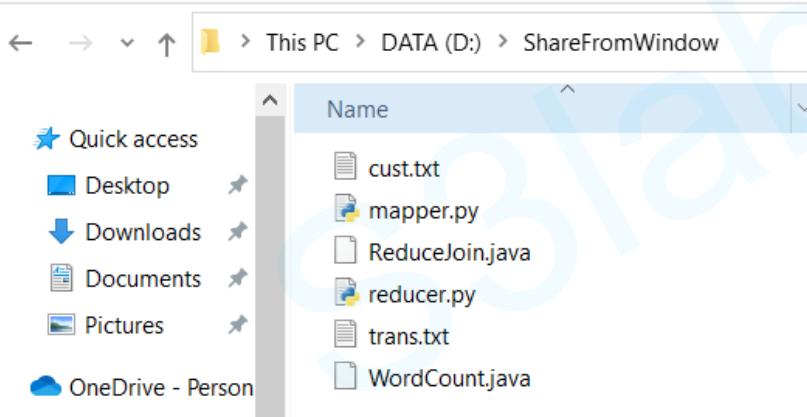
- Switch to root account (password: cloudera)
- Mount the shared folder:

Command: **Mount –t vboxsf ShareFromWindow /home/cloudera/workspace/ShareVM**

```
[cloudera@quickstart ~]$ mount -t vboxsf ShareFromWindow /home/cloudera/workspace/ShareVM
mount: only root can do that
[cloudera@quickstart ~]$ su
Password:
[root@quickstart cloudera]# mount -t vboxsf ShareFromWindow /home/cloudera/workspace/ShareVM
[root@quickstart cloudera]#
```

# Share Folders between VM and Host

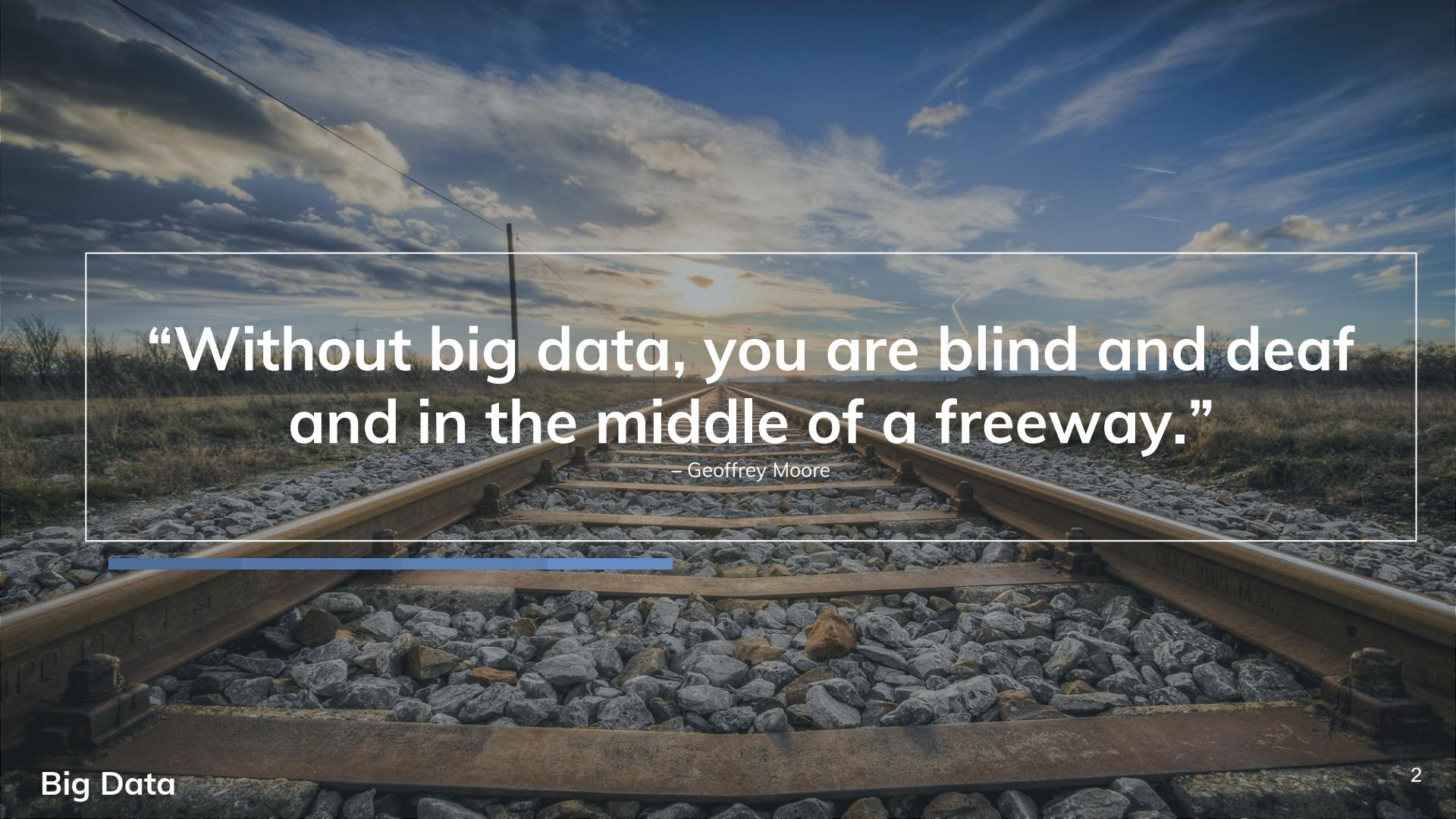
- Put some files to the shared folder in Window
- Click the newly created shortcut in Cloudera VM and check if the files appear



# Big Data

## Data Loading Tools

Trong-Hop Do

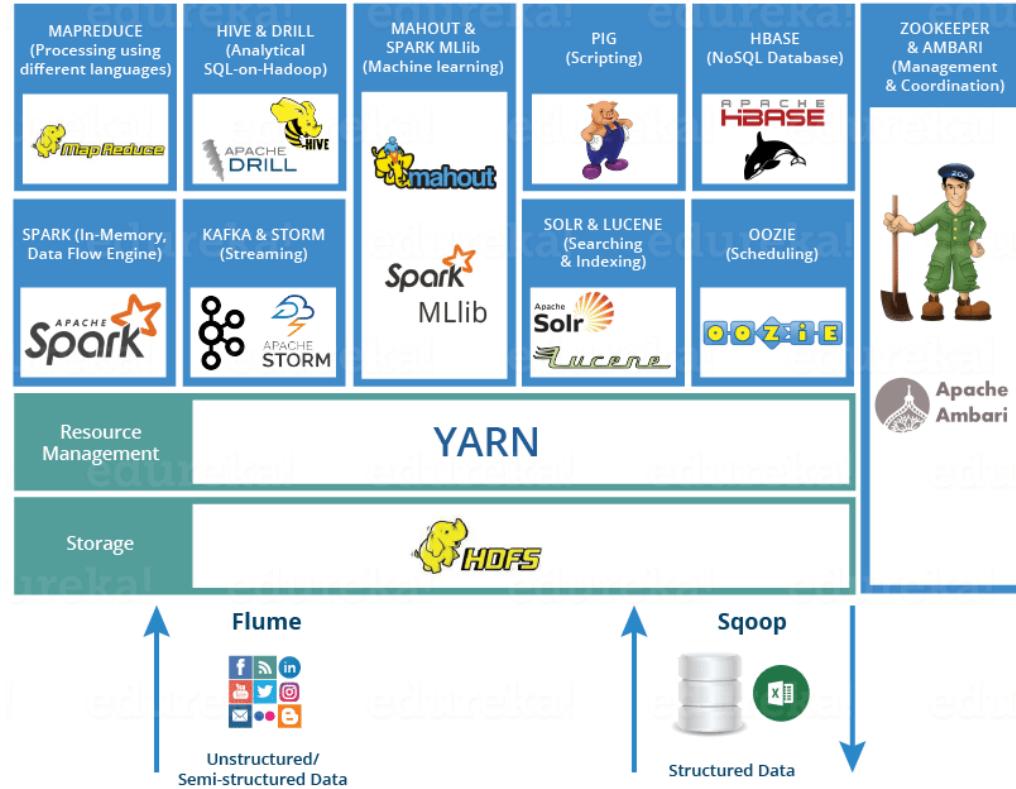
A photograph of a railway track receding into a cloudy sky. The track is made of steel rails and wooden sleepers, with gravel ballast. The sky is filled with scattered clouds, suggesting a sunset or sunrise. A white rectangular box contains the quote.

“Without big data, you are blind and deaf  
and in the middle of a freeway.”

– Geoffrey Moore



# Hadoop Ecosystem



# Apache Flume Tutorial

---

# Introduction to Apache Flume

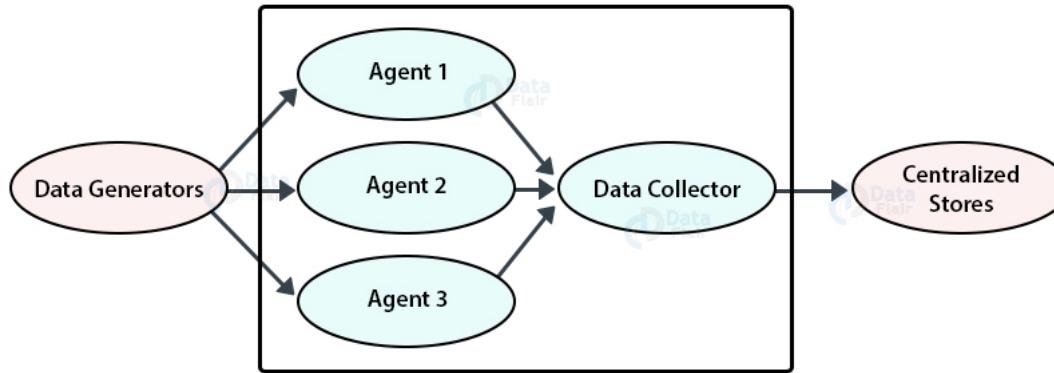
- Apache Flume is a tool for data ingestion in HDFS. It collects, aggregates and transports large amount of streaming data such as log files, events from various sources like network traffic, social media, email messages etc. to HDFS. Flume is a highly reliable & distributed.
- The main idea behind the Flume's design is to capture streaming data from various web servers to HDFS. It has simple and flexible architecture based on streaming data flows. It is fault-tolerant and provides reliability mechanism for Fault tolerance & failure recovery.





# Data transfer components

## Flume - How it works



- **Agent** nodes are typically installed on the machines that generate the logs and are data's initial point of contact with Flume. They forward data to the next tier of **collector** nodes, which aggregate the separate data flows and forward them to the final **storage** tier.



# Data transfer components

## Flume - How it works

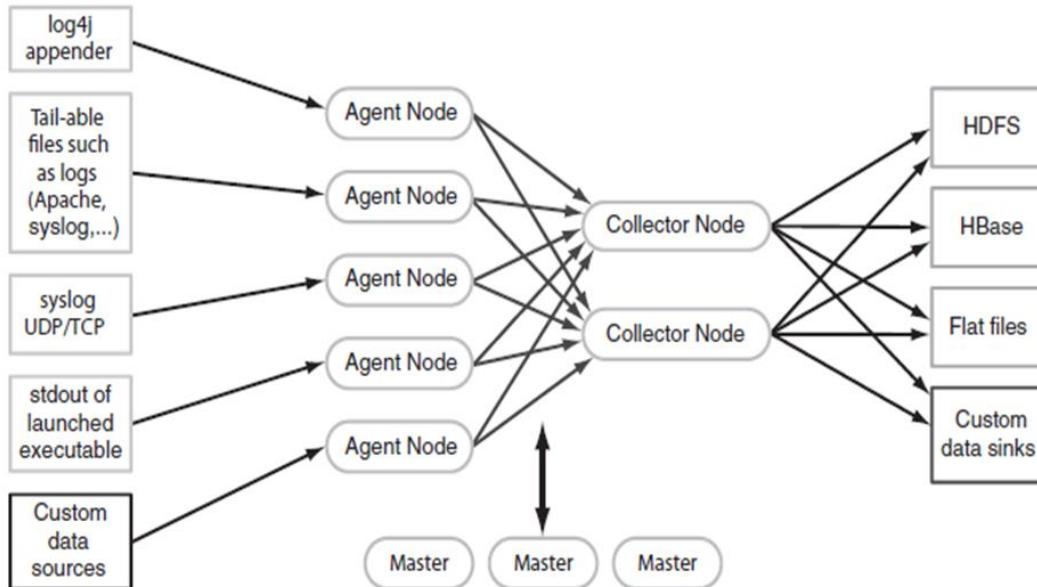


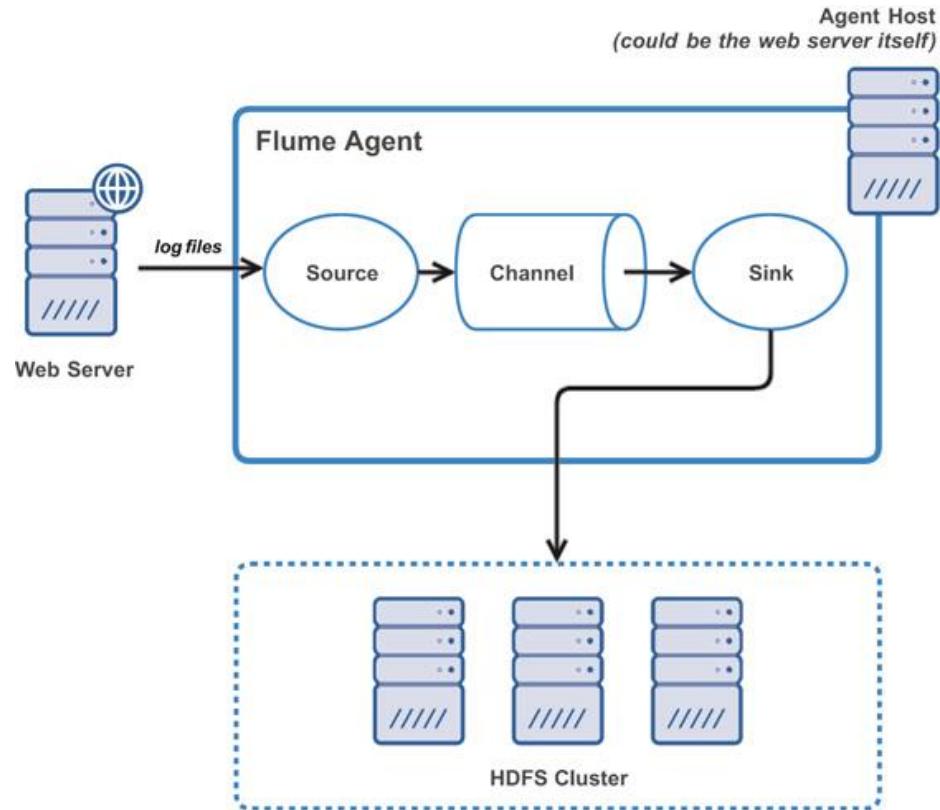
Figure 2.2 Flume architecture for collecting streaming data



# Data transfer components

## Flume - Agent architecture

- Sources:
  - HTTP, Syslog, JMS, Kafka, Avro, Twitter - stream api for tweets download, ...
- Sink:
  - HDFS, Hive, HBase, Kafka, Solr, ...
- Channel:
  - File, JDBC, Kafka, ...



# Start Flume on Cloudera Quickstart VM

- To add Flume to Cloudera Quickstart VM, you need to launch Cloudera Manager
- Configure the VM.
  - Allocate a minimum of 10023 MB memory.
  - Allocate 2 CPUs.
  - Allocate 20 MB video memory.
  - Consider setting the clipboard to bidirectional.

# Start Flume on Cloudera Quickstart VM

- Launch Cloudera Express

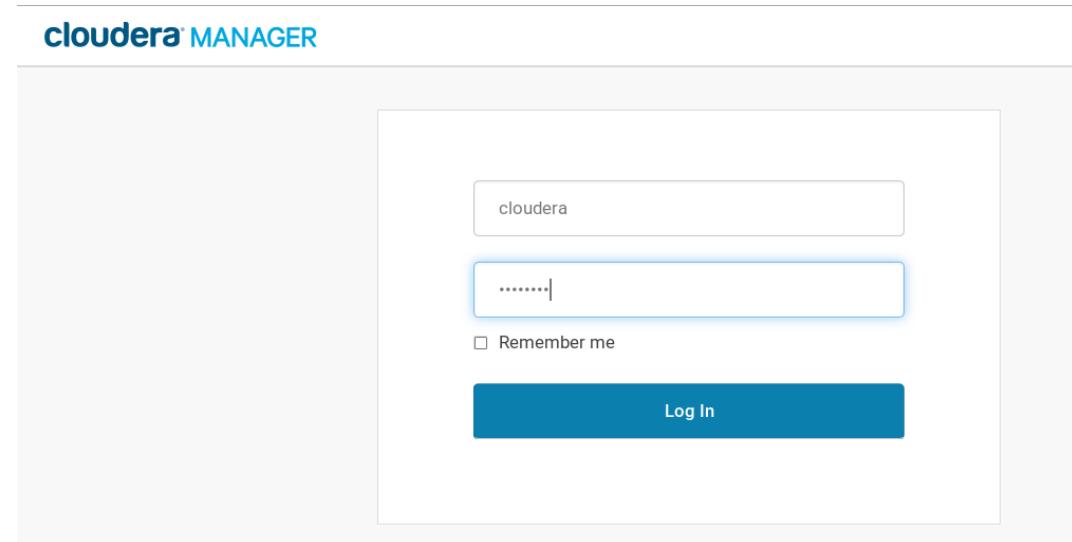


# Start Flume on Cloudera Quickstart VM

- Check the status of Namenode services
  - Command: **sudo service hadoop-hdfs-namenode status**
  - If namenode is **not** running, then start namenode service
  - Command: **sudo service hadoop-hdfs-namenode start**
- Check the status of Datanode services
  - Command: **sudo service hadoop-hdfs-datanode status**
  - If datanode is not running, then start datanode service
  - Command: **sudo service hadoop-hdfs-datanode start**

# Start Flume on Cloudera Quickstart VM

- Open Cloudera Manager in web browser
- Username: **cloudera**
- Password: **cloudera**



# Start Flume on Cloudera Quickstart VM

- After logging in to Cloudera Manager, click **Add Service**

The screenshot shows the Cloudera Manager interface. On the left, a sidebar lists services: Hosts, HBase, HDFS, Hive, Hue, Impala, Key-Value, Oozie, Solr, and others. A blue arrow points from the text 'After logging in to Cloudera Manager, click Add Service' to the 'Add Service' button in the central dialog box. The dialog box contains options: Start, Stop, Restart, Rolling Restart, Deploy Client Configuration, Deploy Kerberos Client Configuration, Upgrade Cluster, and Refresh Cluster. To the right of the dialog is a 'Charts' section with two graphs: 'Cluster CPU' and 'Cluster Disk IO'. The 'Cluster CPU' graph shows a sharp drop in usage around 07:15, followed by a fluctuating baseline at approximately 2.6%. The 'Cluster Disk IO' graph shows a single green spike peaking at 586K/s.

# Start Flume on Cloudera Quickstart VM

- Select Flume

## Add Service to Cloudera QuickStart

Select the type of service you want to add.

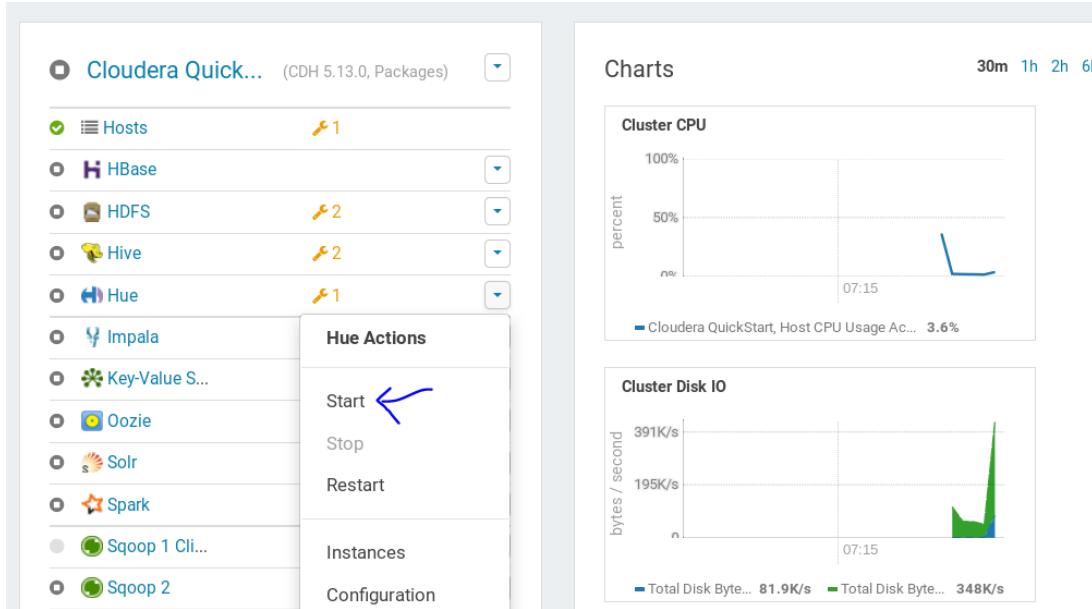
Service Type	Description
<input type="radio"/>  Accumulo	The Apache Accumulo sorted, distributed key/value store is a robust, scalable, high performance data storage and retrieval system. This service only works with releases based on Apache Accumulo 1.6 or later.
<input checked="" type="radio"/>  Flume	Flume collects and aggregates data from almost any source into a persistent store such as HDFS.
<input type="radio"/>  HBase	Apache HBase provides random, real-time, read/write access to large data sets (requires HDFS and ZooKeeper).
<input type="radio"/>  HDFS	Apache Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute hosts throughout a cluster to enable reliable, extremely rapid computations.

[Back](#)

[Continue](#)

# Start Flume on Cloudera Quickstart VM

- Start Hue



# Start Flume on Cloudera Quickstart VM

- Start Flume

The screenshot shows the Cloudera Manager interface for a cluster named "Cloudera Quick...".

**Services Status:**

- Hosts: 1 healthy
- Flume: 2 healthy
- HBase: 0 healthy
- HDFS: 2 healthy
- Hive: 2 healthy
- Hue: 1 healthy
- Impala: 0 healthy
- Key-Value S...: 0 healthy
- Oozie: 0 healthy

**Charts:**

Cluster CPU: 30m 1h 2h 6h 12h 1d 7d 30d

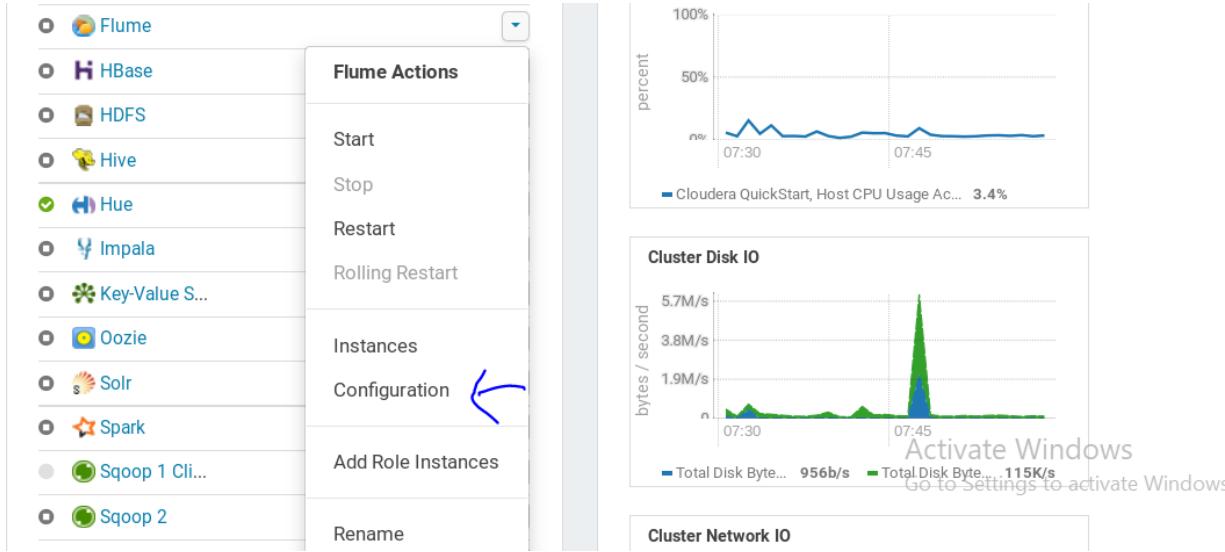
Cluster Disk IO: 1 second 2.9M/s 1.9M/s

Activate

Detailed description: The interface includes a navigation bar at the top with links like Home, Overview, Services, Clusters, Hosts, and Help. Below the navigation is a search bar. The main area has two tabs: "Overview" (selected) and "Logs". The "Overview" tab displays service status and metrics. The "Logs" tab shows log entries for various services. On the right side, there's a sidebar with "Quick Start" and "Help" sections.

# Start Flume on Cloudera Quickstart VM

- Check the configuration of Flume



# Start Flume on Cloudera Quickstart VM

- Check the port (9999 in this VM)

The screenshot shows the Cloudera Manager interface for managing a Flume agent. The left sidebar lists categories like Advanced, Flume-NG Solr Sink, Logs, Main, Monitoring, Performance, Ports and Addresses, Resource Management, Security, and Stacks Collection. The main panel shows the configuration for the 'tier1' agent, which is part of the 'Agent Default Group'. The configuration file contains the following code:

```
# standard properties.
tier1.sources.source1.type = netcat
tier1.sources.source1.bind = 127.0.0.1
tier1.sources.source1.port = 9999
tier1.sources.source1.channels = channel1
tier1.channels.channel1.type = memory
```

The 'Flume Home Directory' is set to '/var/lib/flume-ng'.

# Start Flume on Cloudera Quickstart VM

Use Telnet to test the default Flume implementation

- Firstly, let's install telnet
- Command: **sudo yum install telnet**

```
[cloudera@quickstart ~]$ sudo yum install telnet
Loaded plugins: fastestmirror, security
Setting up Install Process
Determining fastest mirrors
epel/metalink
| 5.1 kB    00:00
```

```
Installed:
  telnet.x86_64 1:0.17-49.el6_10

Complete!
[cloudera@quickstart ~]$ █
```

# Start Flume on Cloudera Quickstart VM

Use Telnet to test the default Flume implementation

- Launch Telnet with the command: **telnet localhost 9999**
- At the prompt, enter **Hello world ^.^**
- Press **Ctr+]** to escape
- Type **quit** to close telnet

```
[cloudera@quickstart ~]$ telnet localhost 9999
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Hello world ^.^
OK
^]

telnet> quit
Connection closed.
[cloudera@quickstart ~]$ █
```

# Start Flume on Cloudera Quickstart VM

Use Telnet to test the default Flume implementation

- Check the log
- Command: `cat /var/log/flume-ng/flume-cmf-flume-AGENT-quickstart.cloudera.log`

```
[cloudera@quickstart ~]$ cat /var/log/flume-ng/flume-cmf-flume-AGENT-quickstart.cloudera.log
2020-10-01 08:04:33,945 INFO org.apache.flume.node.PollingPropertiesFileConfigurationProvider: Configuration provider starting
2020-10-01 08:04:33,964 INFO org.apache.flume.node.PollingPropertiesFileConfigurationProvider: Reloading configuration file:/var/run/cloudera-scm-agent/process/8-flume-AGENT/flume.conf
2020-10-01 08:04:33,967 INFO org.apache.flume.conf.FlumeConfiguration: Processing:sink1

2020-10-01 08:16:43,778 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 48 65 6C 6C
6F 20 77 6F 72 6C 64 20 5E 2E 5E 0D Hello world ^.^. }
[cloudera@quickstart ~]$ █
```

# Writing from Flume to HDFS

Create the `/flume/events` directory

- In the VM web browser, open Hue
- Click File Browser
- In the `/user/cloudera` directory, click New->Directory
- Create a directory named **flume**

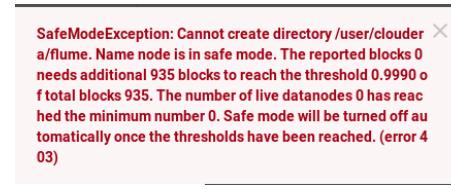
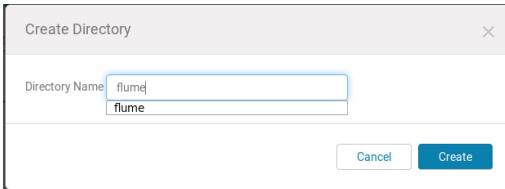
The screenshot shows the Hue File Browser interface. The top navigation bar includes 'HUE', 'Query', a search bar, and user information for 'cloudera'. The main area is titled 'File Browser' and shows the path '/ user / cloudera'. A modal dialog is open in the center, titled 'New', with options for 'File' and 'Directory'. A blue arrow points to the 'Directory' option. Below the modal, a table lists two items: a folder named 'flume' and a file named '.'. The table columns are 'Name', 'Size', 'User', 'Group', 'Permissions', and 'Date'. At the bottom of the page, there are pagination controls for 'Page 1 of 1'.

	Name	Size	User	Group	Permissions	Date
	flume		hdfs	supergroup	drwxr-xr-x	October 23, 2017 10:31 AM
	.		cloudera	cloudera	drwxr-xr-x	October 23, 2017 10:28 AM

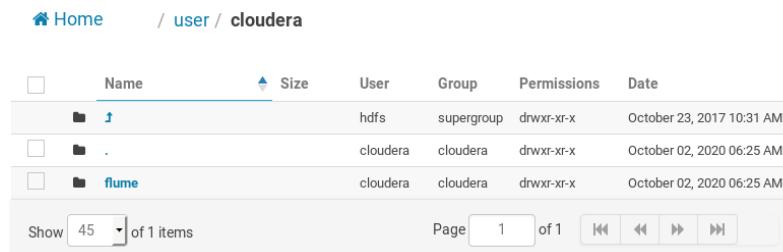
# Writing from Flume to HDFS

Create the /flume/events directory

- If you get this error when creating new directory



- Then run command: `sudo -u hdfs hdfs dfsadmin -safemode leave`



	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	flume		hdfs	supergroup	drwxr-xr-x	October 23, 2017 10:31 AM
<input type="checkbox"/>	.		cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:25 AM
<input type="checkbox"/>	flume		cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:25 AM

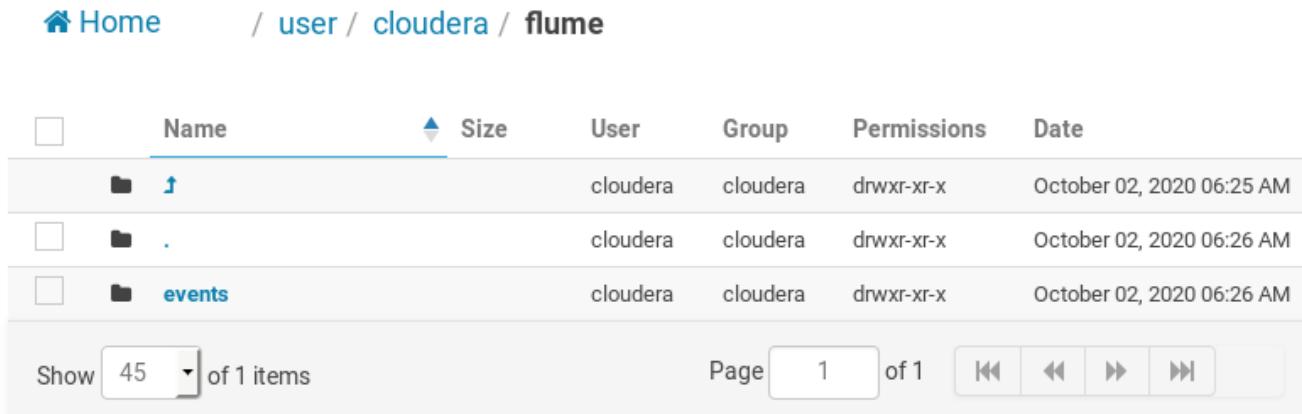
Show 45 of 1 items

Page 1 of 1

# Writing from Flume to HDFS

Create the /flume/events directory

- In the flume directory, create a directory named **events**



The screenshot shows a file browser interface with the following details:

Path: [Home](#) / [user](#) / [cloudera](#) / [flume](#)

	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>			cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:25 AM
<input type="checkbox"/>	.		cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:26 AM
<input type="checkbox"/>	events		cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:26 AM

Show 45 of 1 items

Page 1 of 1

Navigation icons: back, forward, search, etc.

# Writing from Flume to HDFS

Create the /flume/events directory

- Check the box to the left of the events directory, then click the Permissions setting

[Home](#) / user / cloudera / flume

	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	↑		cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:25 AM
<input type="checkbox"/>	.		cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:26 AM
<input checked="" type="checkbox"/>	events		cloudera	cloudera	drwxr-xr-x	October 02, 2020 06:26 AM

Show  of 1 items

Page  of 1

# Writing from Flume to HDFS

Create the /flume/events directory

- Enable Write access for Group and Other users
- Then click Submit

Change Permissions

	User	Group	Other
Read	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Write	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Execute	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sticky			<input type="checkbox"/>
Recursive			<input type="checkbox"/>

[Cancel](#) [Submit](#)

# Writing from Flume to HDFS

## Change the Flume configuration

- Open Cloudera Manager -> click Flume -> Click the Configuration tab
- Scroll or search for the **Configuration File** item.
- Append the following lines to the Configuration File settings
- Click **Save Changes**

```
tier1.sinks.sink1.type = HDFS
tier1.sinks.sink1.filetype = DataStream
tier1.sinks.sink1.channel = channel1
tier1.sinks.sink1.hdfs.path =
    hdfs://localhost:8020/user/cloudera/flume/events
```

The screenshot shows the 'Configuration File' section of the Cloudera Manager Flume configuration interface. The configuration file content is as follows:

```
tier1.sinks.sink1.type=HDFS
tier1.sinks.sink1.fileType=DataStream
tier1.sinks.sink1.channel = channel1
tier1.sinks.sink1.hdfs.path = hdfs://localhost:8020/user/cloudera
/flume/events|
```

Below the configuration file, there are sections for 'Flume Home Directory' (set to '/var/lib/flume-ng') and 'Plugin directories' (set to '/usr/lib/flume-ng/plugins.d'). At the bottom, there is a 'Save Changes' button, which is highlighted with a large orange arrow pointing to it.

# Writing from Flume to HDFS

Change the Flume configuration

- Restart Flume

The screenshot shows the Cloudera Manager interface for the Flume service. The top navigation bar includes links for Clusters, Hosts, Diagnostics, Audits, Charts, Administration, and various search and support options. The main content area displays the Flume configuration page. On the left, there's a sidebar with sections for Filters, SCOPE, and CATEGORY. The main panel shows tabs for Status, Instances, Configuration (which is selected), and other service details. A large Actions dropdown menu is open, listing Start, Stop, Restart (with an orange arrow pointing to it), Rolling Restart, Add Role Instances, Rename, Enter Maintenance Mode, and Update Config. To the right of the actions, there are sections for Flume (Service-Wide) settings, including a radio button for HDFS (selected) and an option for none. At the bottom right, there are Save Changes and Cancel buttons.

# Writing from Flume to HDFS

Change the Flume configuration

- Start YARN

The screenshot shows the Cloudera Manager interface. On the left, there's a sidebar with several service icons: SQuoop 1 Cli..., SQuoop 2 (with a yellow '1' badge), YARN (MR2 ...), ZooKeeper, and Cloudera M... (with a green checkmark). The YARN (MR2 Included) service is selected. A context menu titled "YARN (MR2 Included) Actions" is open over it, containing four options: Start, Stop, Restart, and Rolling Restart. A blue arrow points to the "Start" button. To the right of the service list are two monitoring charts. The top chart is titled "Cluster Network IO" and shows a single sharp blue peak reaching up to 391K/s, with a baseline around 195K/s. The bottom chart is titled "HDFS IO" and shows a flat line at approximately 91.3b/s. The Y-axis for both charts is labeled "bytes / second".

Cloudera Manager

SQuoop 1 Cli...

SQuoop 2

YARN (MR2 ...)

ZooKeeper

Cloudera M...

**YARN (MR2 Included) Actions**

- Start
- Stop
- Restart
- Rolling Restart

Total Disk Byte... 3.3M/s Total Disk Byte... 694K/s

Cluster Network IO

bytes / second

391K/s 195K/s 06:15 06:30

Total Bytes Re... 197b/s Total Bytes Tr... 91.3b/s

HDFS IO

# Writing from Flume to HDFS

## Writing to HDFS

- In a terminal window, launch Telnet with the command **telnet localhost 9999**
- At the prompt, enter some text

```
[cloudera@quickstart ~]$ telnet localhost 9999
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^].
Hello HDFS writing from Flume ^.^
OK
^]

telnet> quit
Connection closed.
[cloudera@quickstart ~]$ █
```

# Writing from Flume to HDFS

## Writing to HDFS

- Hue File Browser, open the /user/cloudera/flume/events directory
- Click the file name **FlumeData.xxxxxx** link to view the data sent by Flume to HDFS

The screenshot shows the Hue File Browser interface. The top navigation bar includes the Hue logo, a 'Query' dropdown, a search bar, and user information ('Jobs' and 'cloudera'). The main area is titled 'File Browser' and shows a single file named 'FlumeData.1601646911241'. The file content is displayed as a text log of binary data. A yellow arrow points to the end of the log, indicating where new data might be appended.

```
000000: 53 45 51 06 21 6f 72 67 2e 61 70 61 63 68 65 2e SEQ.!org.apache.
000010: 68 61 64 6f 6f 70 2e 66 6f 2e 4c 6f 6e 67 57 72 hadoop.io.LongWr
000020: 69 74 61 62 6c 65 22 6f 72 67 2e 61 70 61 63 68 itable"org.apach
000030: 65 2e 68 61 64 6f 6f 70 2e 69 6f 2e 42 79 74 65 e.hadoop.io.Byte
000040: 73 57 72 69 74 61 62 6c 65 00 00 00 00 00 00 2d sWritable.....-
000050: 01 4d 2d 5e e1 65 f3 f7 1e c0 3d 4d ec 01 2e 00 .M-^.e....-M....
000060: 00 00 2e 00 00 00 08 00 00 01 74 e9 98 5f 36 00 .....t...-6.
000070: 00 00 22 48 65 6c 6c 6f 20 48 44 46 53 20 77 72 .."Hello HDFS wr
000080: 69 74 69 6e 67 20 66 72 6f 6d 20 46 6c 75 6d 65 iting from Flume
000090: 20 5e 2e 5e 0d ff ff ff ff 2d 01 4d 2d 5e e1 65 ^.^.....-M-^.e
0000a0: f3 f7 1e c0 3d 4d ec 01 2e ....=M...
```

# Apache Sqoop Tutorial

---



# Apache Sqoop Tutorial

## Sqoop (Sql-to-hadoop)

- Command-line interface for **transforming** data between **RDBMS & Hadoop**
- Parallelized data transfer with **MapReduce**
- Support **incremental** imports
- **Imports** use to populate tables in Hadoop
- **Exports** use to put data from Hadoop into relational database
- **Sqoop2** -> Sqoop-as-a-Service: server-based implementation of Sqoop





# Apache Swoop Tutorial

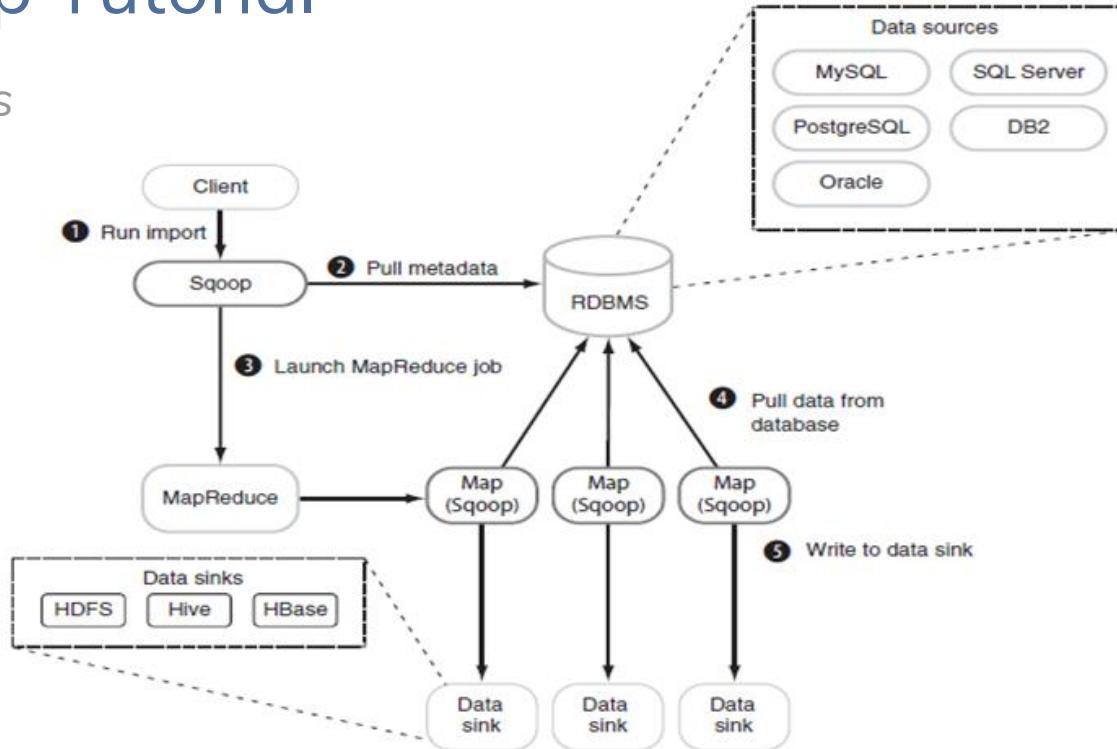
## Swoop - How it works

- The dataset being transferred is broken into small blocks.
- **Map only** job is launched.
- Individual mapper is responsible for transferring a block of the dataset.



# Apache Swoop Tutorial

## Swoop - How it works

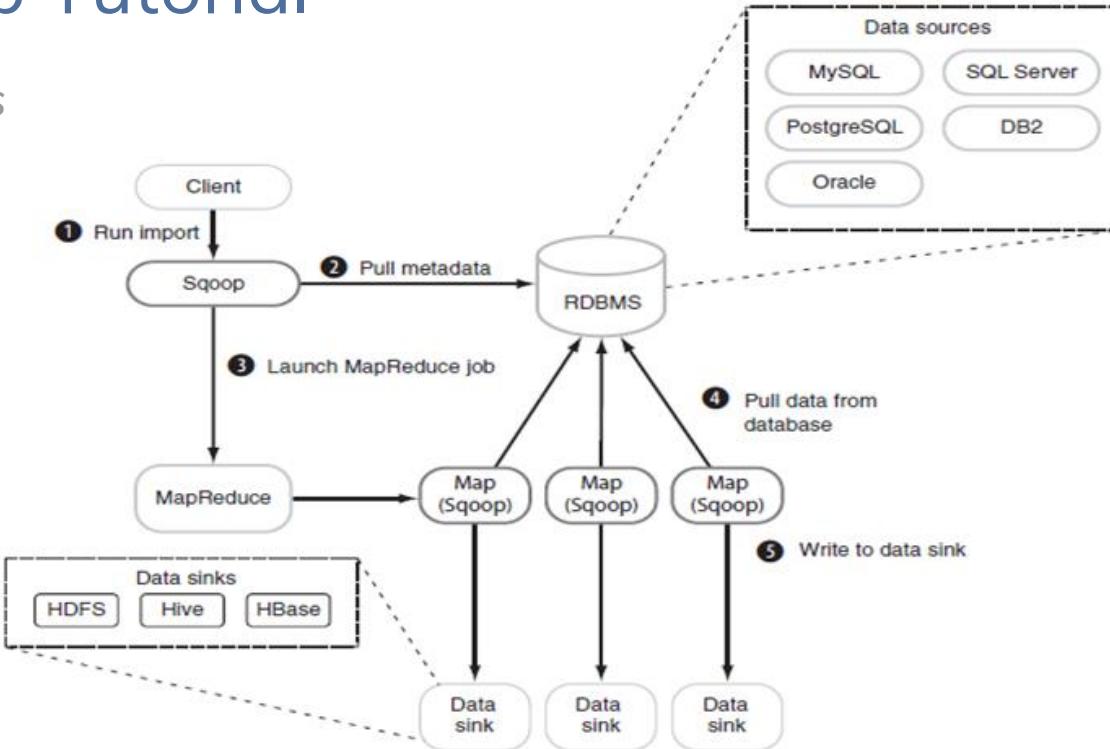


**Figure 2.20 Five-stage Swoop import overview: connecting to the data source and using MapReduce to write to a data sink**



# Apache Swoop Tutorial

## Swoop - How it works

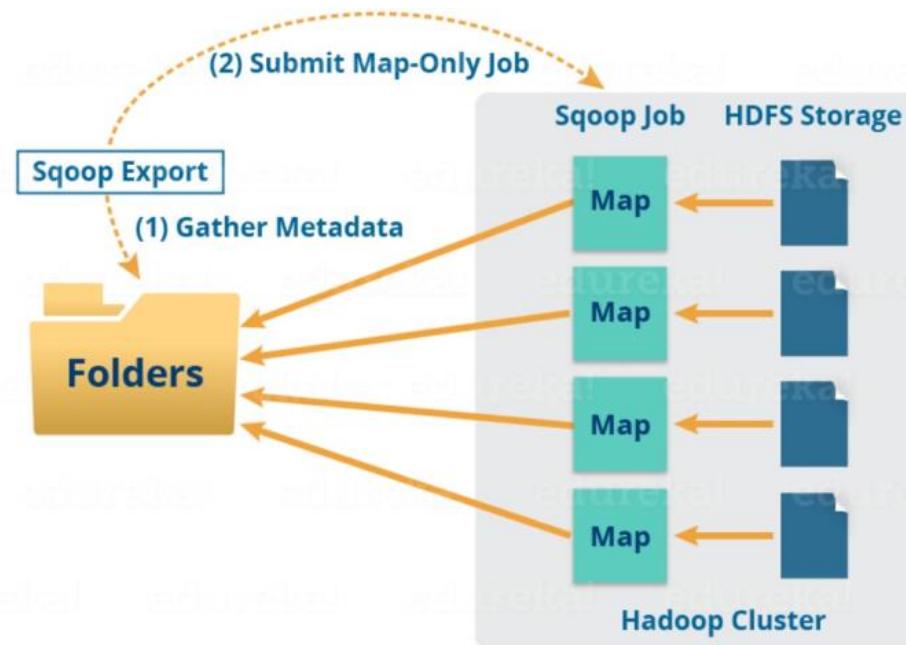


**Figure 2.20 Five-stage Swoop import overview: connecting to the data source and using MapReduce to write to a data sink**



# Apache Sqoop Tutorial

## Sqoop - How it works

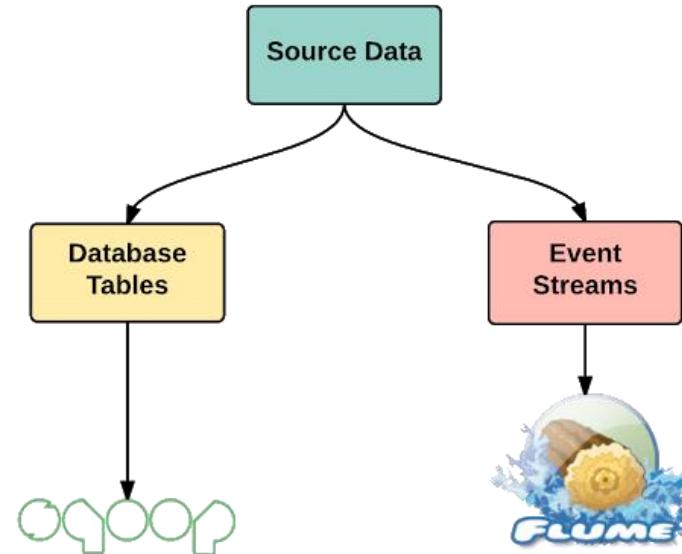




# Apache Swoop Tutorial

## Flume vs Swoop

- Flume only ingests unstructured data or semi-structured data into HDFS.
- Swoop can import as well as export structured data from RDBMS or Enterprise data warehouses to HDFS or vice versa.



# Apache Sqoop Tutorial

- Display a list of all available tools
- Command: **sqoop help**

```
[cloudera@quickstart ~]$ sqoop help

Available commands:
codegen          Generate code to interact with database records
create-hive-table Import a table definition into Hive
eval             Evaluate a SQL statement and display the results
export            Export an HDFS directory to a database table
help              List available commands
import             Import a table from a database to HDFS
import-all-tables Import tables from a database to HDFS
import-mainframe   Import datasets from a mainframe server to HDFS
job                Work with saved jobs
list-databases    List available databases on a server
list-tables        List available tables in a database
merge              Merge results of incremental imports
metastore          Run a standalone Sqoop metastore
version            Display version information
```

# Sqoop connecting to a Database Server

```
[cloudera@quickstart ~]$ sqoop help import
```

Argument	Description
--connect <jdbc-uri>	Specify JDBC connect string
--connection-manager <class-name>	Specify connection manager class to use
--driver <class-name>	Manually specify JDBC driver class to use
--hadoop-mapred-home <dir>	Override \$HADOOP_MAPRED_HOME
--help	Print usage instructions
--password-file	Set path for a file containing the authentication password
-P	Read password from console
--password <password>	Set authentication password
--username <username>	Set authentication username
--verbose	Print more information while working
--connection-param-file <filename>	Optional properties file that provides connection parameters
--relaxed-isolation	Set connection transaction isolation to read uncommitted for the mappers.

# Sqoop import RDBMS table into HDFS

- Login to mysql

```
[cloudera@quickstart ~]$ mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 19
```

# Sqoop import RDBMS table into HDFS

- Create database StudentInfo

```
mysql> create database StudentInfo;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| StudentInfo |
| cm |
| firehose |
| hue |
| metastore |
| mysql |
| nav |
| navms |
| oozie |
| retail_db |
| rman |
| sentry |
+-----+
13 rows in set (0.00 sec)
```

- Use the newly created database

```
mysql> use StudentInfo;
Database changed
```

```
mysql> ■
```

# Sqoop import RDBMS table into HDFS

- Create table student and insert some data into this table

```
|mysql> create table student(std_id integer, std_name varchar(43));  
|Query OK, 0 rows affected (0.01 sec)  
  
|mysql> insert into student values (101,'le'), (102,'pham'), (103,'tran'), (104,'  
|ngo'), (105,'vu'), (106,'dao');  
|Query OK, 6 rows affected (0.00 sec)  
|Records: 6  Duplicates: 0  Warnings: 0
```

# Sqoop import RDBMS table into HDFS

- Check the newly created table

```
mysql> select * from student;
+-----+-----+
| std_id | std_name |
+-----+-----+
|    101 | le
|    102 | pham
|    103 | tran
|    104 | ngo
|    105 | vu
|    106 | dao
+-----+-----+
6 rows in set (0.00 sec)

mysql> █
```

# Sqoop import RDBMS table into HDFS

- Use Sqoop to import table student into HDFS
- Command: [cloudera@quickstart ~]\$ sqoop import --connect jdbc:mysql://localhost/StudentInfo --table student --username root --password cloudera --split-by std\_id --m 1 --target-dir /user/cloudera/studentInfo/student;

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/StudentInfo --table student --username root  
--password cloudera --split-by std_id --m 1 --target-dir '/user/cloudera/studentInfo/student';  
Warning: /usr/lib/sqoop/..//accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO_HOME to the root of your Accumulo installation.  
20/09/29 21:39:27 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0  
20/09/29 21:39:27 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P  
instead.  
20/09/29 21:39:28 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
20/09/29 21:39:28 INFO tool.CodeGenTool: Beginning code generation  
20/09/29 21:39:28 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `student` AS t LIMIT 1  
20/09/29 21:39:28 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `student` AS t LIMIT 1  
20/09/29 21:39:28 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
```

```
File Input Format Counters  
    Bytes Read=0  
File Output Format Counters  
    Bytes Written=48
```

```
20/09/29 21:39:52 INFO mapreduce.ImportJobBase: Transferred 48 bytes in 20.273 seconds (2.3677 bytes/sec)  
20/09/29 21:39:52 INFO mapreduce.ImportJobBase: Retrieved 6 records.
```

# Sqoop import RDBMS table into HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 7 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 00:30 HSOutput
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 23:13 HadoopStreaming
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 05:17 ReduceJoin
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 03:20 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:07 inputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 21:39 studentInfo
[cloudera@quickstart ~]$ hdfs dfs -ls studentInfo
Found 1 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 21:39 studentInfo/student
[cloudera@quickstart ~]$ hdfs dfs -ls studentInfo/student
Found 2 items
-rw-r--r--  1 cloudera cloudera      0 2020-09-29 21:39 studentInfo/student/_SUCCESS
-rw-r--r--  1 cloudera cloudera    48 2020-09-29 21:39 studentInfo/student/part-m-00000
[cloudera@quickstart ~]$ █
```

# Sqoop import RDBMS table into HDFS

- Now let us see the output on our Command shell:

```
[cloudera@quickstart ~]$ hdfs dfs -cat studentInfo/student/part-m-00000  
101,le  
102,pham  
103,tran  
104,ngo  
105,vu  
106,dao  
[cloudera@quickstart ~]$ █
```

---

# Sqoop import RDBMS table into HDFS without target directory

- Import RDBMS table into HDFS without specifying target directory
- Command: [cloudera@quickstart ~]\$ sqoop import --connect jdbc:mysql://localhost/StudentInfo --table student --username root --password cloudera --split-by std\_id --m 1

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/StudentInfo --table student --username root  
--password cloudera --split-by std_id --m 1  
Warning: /usr/lib/sqoop/.../accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO_HOME to the root of your Accumulo installation.  
20/09/29 22:35:55 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0  
20/09/29 22:35:55 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P  
instead.  
20/09/29 22:35:55 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
20/09/29 22:35:55 INFO tool.CodeGenTool: Beginning code generation  
20/09/29 22:35:56 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `student` AS t LIMIT 1  
20/09/29 22:35:56 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `student` AS t LIMIT 1  
20/09/29 22:35:56 INFO mapreduce.ImportJobBase: Map input files: 1  
20/09/29 22:36:15 INFO mapreduce.ImportJobBase: Transferred 48 bytes in 16.5944 seconds (2.8926 bytes/sec)  
20/09/29 22:36:15 INFO mapreduce.ImportJobBase: Retrieved 6 records.
```

# Sqoop import RDBMS table into HDFS without target directory

- Check the newly created directory

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 8 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 00:30 HSOutput
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 23:13 HadoopStreaming
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 05:17 ReduceJoin
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 03:20 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:07 inputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 22:36 student
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 21:39 studentInfo
[cloudera@quickstart ~]$ hdfs dfs -ls student
Found 2 items
-rw-r--r--  1 cloudera cloudera      0 2020-09-29 22:36 student/_SUCCESS
-rw-r--r--  1 cloudera cloudera    48 2020-09-29 22:36 student/part-m-00000
[cloudera@quickstart ~]$ █
```

# Sqoop – IMPORT Command with Where Clause

- Command: `sqoop import --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera --table student --m 1 --where 'std_id > 103' --target-dir /user/cloudera/studentInfo/studentAfter103`

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/StudentIn
fo --username root --password cloudera --table student --m 1 --where 'std_id > 1
03' --target-dir /user/cloudera/studentInfo/studentAfter103
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
20/09/30 05:04:07 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
20/09/30 05:04:07 WARN tool.BaseSqoopTool: Setting your password on the command-
line is insecure. Consider using -P instead.
20/09/30 05:04:07 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
20/09/30 05:04:07 INFO tool.CodeGenTool: Beginning code generation
20/09/30 05:04:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
20/09/30 05:04:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
```

# Sqoop – IMPORT Command with Where Clause

- Check the result in HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls studentInfo
Found 2 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 21:39 studentInfo/student
drwxr-xr-x  - cloudera cloudera      0 2020-09-30 05:04 studentInfo/studentAfter103
[cloudera@quickstart ~]$ hdfs dfs -ls studentInfo/studentAfter103
Found 2 items
-rw-r--r--  1 cloudera cloudera      0 2020-09-30 05:04 studentInfo/studentAfter103/_SUCCESS
-rw-r--r--  1 cloudera cloudera    23 2020-09-30 05:04 studentInfo/studentAfter103/part-m-00000
[cloudera@quickstart ~]$ hdfs dfs -cat studentInfo/studentAfter103/part*
104,ngo
105,vu
106,dao
[cloudera@quickstart ~]$ █
```

# Free-form Query Imports

- Sqoop can also import the result set of an arbitrary SQL query.
- When importing a free-form query, you must specify --target-dir, --split-by, and include the token \$CONDITIONS.
- Command: `sqoop import --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera --query 'select std_name from student where std_id=103 and $CONDITIONS' --split-by std_name --target-dir /user/cloudera/studentInfo/studentName103`

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera  
--query 'select std_name from student where std_id=103 and $CONDITIONS' --split-by std_name --target-dir /user/cloudera  
/studentInfo/studentName103  
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO_HOME to the root of your Accumulo installation.  
20/09/30 05:22:53 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0  
20/09/30 05:22:53 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  
  
File Input Format Counters  
    Bytes Read=0  
File Output Format Counters  
    Bytes Written=5  
20/09/30 05:23:13 INFO mapreduce.ImportJobBase: Transferred 5 bytes in 16.4991 seconds (0.303 bytes/sec)  
20/09/30 05:23:13 INFO mapreduce.ImportJobBase: Retrieved 1 records.  
[cloudera@quickstart ~]$
```

# Free-form Query Imports

- Check the result in HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls studentInfo
Found 3 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 21:39 studentInfo/student
drwxr-xr-x  - cloudera cloudera      0 2020-09-30 05:04 studentInfo/studentAfter103
drwxr-xr-x  - cloudera cloudera      0 2020-09-30 05:23 studentInfo/studentName103
[cloudera@quickstart ~]$ hdfs dfs -ls studentInfo/studentName103
Found 2 items
-rw-r--r--  1 cloudera cloudera      0 2020-09-30 05:23 studentInfo/studentName103/_SUCCESS
-rw-r--r--  1 cloudera cloudera      5 2020-09-30 05:23 studentInfo/studentName103/part-m-00000
[cloudera@quickstart ~]$ hdfs dfs -cat studentInfo/studentName103/part*
tran
[cloudera@quickstart ~]$ █
```

# Sqoop - list all databases

- List all databases
- Command: **sqoop list-databases --connect jdbc:mysql://localhost/ --username root --password cloudera**

```
[cloudera@quickstart ~]$ sqoop list-databases --connect jdbc:mysql://localhost/ --username root --password cloudera
Warning: /usr/lib/sqoop/..../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
20/09/30 06:07:11 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
20/09/30 06:07:11 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
20/09/30 06:07:12 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
information_schema
StudentInfo
cm
firehose
hue
metastore
mysql
nav
navms
oozie
retail_db
rman
sentry
[cloudera@quickstart ~]$ █
```

# Sqoop - list all tables

- Let's create another table in the StudentInfo database

```
mysql> use StudentInfo;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

```
mysql> create table course (course_id integer,course_name varchar(43),num_credit integer);
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> insert into course values (11,'big data',3), (12,'deep learning',4), (13,'data structure',4), (14,'database', 3);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> ■
```

# Sqoop - list all tables

- List all tables of a particular database
- Command: `sqoop list-tables --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera`

```
[cloudera@quickstart ~]$ sqoop list-tables --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
20/09/30 06:11:31 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
20/09/30 06:11:31 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
20/09/30 06:11:31 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
course
student
[cloudera@quickstart ~]$
```

# Sqoop Imports All Tables

- Let's import all these tables into HDFS
- Command: `sqoop import-all-tables --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera --m 1`

```
[cloudera@quickstart ~]$ sqoop import-all-tables --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera --m 1
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
20/09/30 06:00:07 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
20/09/30 06:00:07 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
20/09/30 06:00:08 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
20/09/30 06:00:08 INFO tool.CodeGenTool: Beginning code generation
```

# Sqoop Imports All Tables

- Check if two folders appeared in HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 9 items
drwxr-xr-x  - cloudera cloudera          0 2020-09-29 00:30 HSOutput
drwxr-xr-x  - cloudera cloudera          0 2020-09-28 23:13 HadoopStreaming
drwxr-xr-x  - cloudera cloudera          0 2020-09-28 05:17 ReduceJoin
drwxr-xr-x  - cloudera cloudera          0 2020-09-30 06:00 course ←
drwxr-xr-x  - cloudera cloudera          0 2020-09-29 03:20 dataset
drwxr-xr-x  - cloudera cloudera          0 2020-09-27 07:07 inputWC
drwxr-xr-x  - cloudera cloudera          0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera cloudera          0 2020-09-30 06:00 student ←
drwxr-xr-x  - cloudera cloudera          0 2020-09-30 05:23 studentInfo
[cloudera@quickstart ~]$ █
```

# Sqoop Export data from HDFS to the RDBMS

- Create a new file in local file system
- Command: **cat > newstudent**
- Press ctr-d to save file
- Then put this file to HDFS

```
[cloudera@quickstart ~]$ cat > newstudent
107,nam
108,viet
109,quoc
[cloudera@quickstart ~]$ hdfs dfs -put newstudent
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 10 items
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 00:30 HSOutput
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 23:13 HadoopStreaming
drwxr-xr-x  - cloudera cloudera      0 2020-09-28 05:17 ReduceJoin
drwxr-xr-x  - cloudera cloudera      0 2020-09-30 06:00 course
drwxr-xr-x  - cloudera cloudera      0 2020-09-29 03:20 dataset
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:07 inputWC
-rw-r--r--  1 cloudera cloudera  26 2020-09-30 08:25 newstudent ←
drwxr-xr-x  - cloudera cloudera      0 2020-09-27 07:29 outputWC
drwxr-xr-x  - cloudera cloudera      0 2020-09-30 06:00 student
drwxr-xr-x  - cloudera cloudera      0 2020-09-30 05:23 studentInfo
[cloudera@quickstart ~]$ █
```

# Sqoop Export data from HDFS to the RDBMS

- Export data from file **newstudent** in HDFS to table **student** in mysql
- The table must exist in the target database
- Command: **sqoop export --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera --table student --export-dir /user/cloudera/newstudent**

```
[cloudera@quickstart ~]$ sqoop export --connect jdbc:mysql://localhost/StudentInfo --username root --password cloudera --table student --export-dir /user/cloudera/newstudent
Warning: /usr/lib/sqoop/.../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
20/09/30 08:34:00 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
20/09/30 08:34:31 MAPREDUCE-TASKS: Setting mapreduce.job.maps=1
20/09/30 08:34:31 INFO mapreduce.ExportJobBase: Transferred 708 bytes in 27.1772 seconds (26.0513 bytes/sec)
20/09/30 08:34:31 INFO mapreduce.ExportJobBase: Exported 3 records.
```

# Sqoop Export data from HDFS to the RDBMS

- Query table **student** in sql to check new rows are inserted

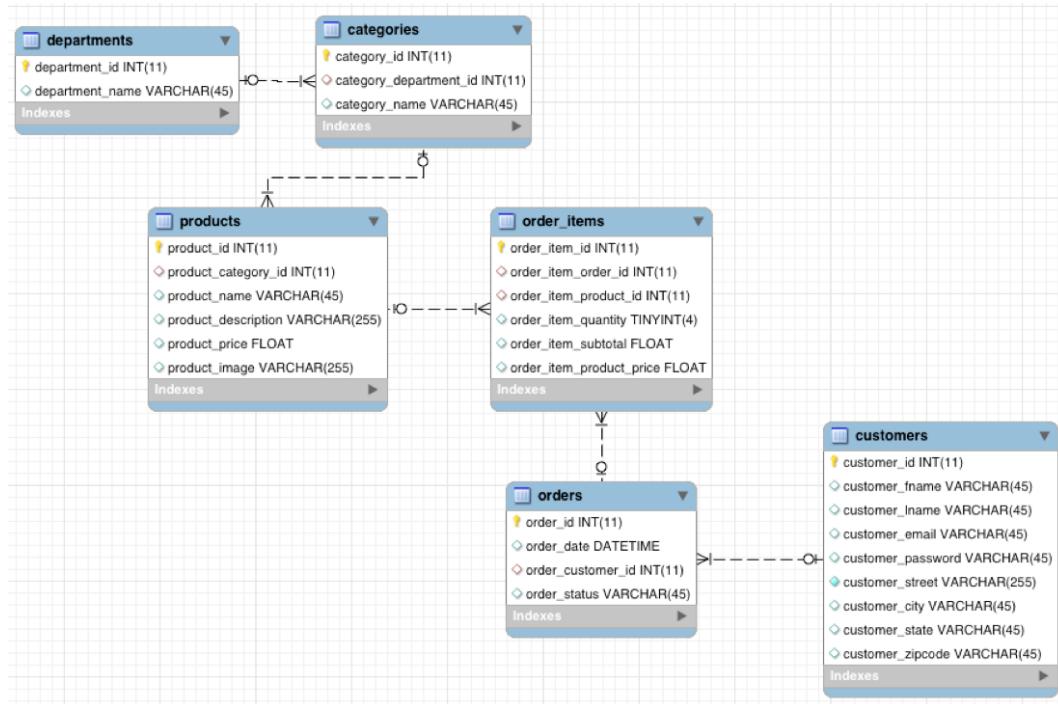
```
[cloudera@quickstart ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.

mysql> use StudentInfo;

mysql> select * from student;
+-----+-----+
| std_id | std_name |
+-----+-----+
| 101   | le
| 102   | pham
| 103   | tran
| 104   | ngo
| 105   | vu
| 106   | dao
| 107   | nam
| 108   | viet
| 109   | quoc
+-----+-----+
9 rows in set (0.00 sec)
```

- Show databases

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| cm |  
| firehose |  
| hue |  
| metastore |  
| mysql |  
| nav |  
| navms |  
| oozie |  
| retail_db | ←  
| rman |  
| sentry |  
+-----+  
12 rows in set (0.00 sec)
```



- Run

```
[cloudera@quickstart ~]$ sqoop import-all-tables \
  -m 1 \
  --connect jdbc:mysql://quickstart:3306/retail_db \
  --username=retail_dba \
  --password=cloudera \
  --compression-codec=snappy \
  --as-parquetfile \
  --warehouse-dir=/user/hive/warehouse \
  --hive-import
```

- This command may take a while to complete. It is launching MapReduce jobs to pull the data from our MySQL database and write the data to HDFS, distributed across the cluster in Apache Parquet format. It is also creating tables to represent the HDFS files in Impala / Apache Hive with matching schema.

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse
Found 6 items
drwxrwxrwx  - cloudera supergroup          0 2021-03-29 17:59 /user/hive/warehouse/categories
drwxrwxrwx  - cloudera supergroup          0 2021-03-29 18:00 /user/hive/warehouse/customers
drwxrwxrwx  - cloudera supergroup          0 2021-03-29 18:01 /user/hive/warehouse/departments
drwxrwxrwx  - cloudera supergroup          0 2021-03-29 18:01 /user/hive/warehouse/order_items
drwxrwxrwx  - cloudera supergroup          0 2021-03-29 18:02 /user/hive/warehouse/orders
drwxrwxrwx  - cloudera supergroup          0 2021-03-29 18:02 /user/hive/warehouse/products
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/customers
Found 3 items
drwxr-xr-x  - cloudera supergroup          0 2021-03-29 18:00 /user/hive/warehouse/customers/.metadata
drwxr-xr-x  - cloudera supergroup          0 2021-03-29 18:00 /user/hive/warehouse/customers/.signals
-rw-r--r--  1 cloudera supergroup  254648 2021-03-29 18:00 /user/hive/warehouse/customers/4d6563b0-549b-486b-be26-b538ed56b660.parquet
```

# Assignment 3

```
[cloudera@quickstart ~]$ sqoop export --connect jdbc:mysql://localhost/EmployeeInfo --username root --password cloudera --table employee --export-dir /user/cloudera/EmployeeNoheader.csv
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
21/04/18 09:11:25 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
21/04/18 09:11:25 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
21/04/18 09:11:25 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
21/04/18 09:11:25 INFO tool.CodeGenTool: Beginning code generation
21/04/18 09:11:26 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employee` AS t LIMIT 1

```

# Assignment 3

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/EmployeeInfo --username root --password cloudera --query 'select id, name, salary from employee where salary > 2000 and $CONDITIONS' --split-by id --target-dir /user/cloudera/Employee2000plus
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
21/04/18 09:42:32 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
21/04/18 09:42:32 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
21/04/18 09:42:32 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
21/04/18 09:42:32 INFO tool.CodeGenTool: Beginning code generation
21/04/18 09:42:33 INFO manager.SqlManager: Executing SQL statement: select id, name, salary from employee where salary > 2000 and (1 = 0)
21/04/18 09:42:33 INFO manager.SqlManager: Executing SQL statement: select id, name, salary from employee where salary > 2000 and (1 = 0)
21/04/18 09:42:33 INFO manager.SqlManager: Executing SQL statement: select id, name, salary from employee where salary > 2000 and (1 = 0)
21/04/18 09:42:33 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
```

```
[cloudera@quickstart ~]$ hdfs dfs -cat Employee2000plus/part*
20024,Vu,3000.0
20025,Dao,2500.0
20026,Nam,2500.0
20027,Viet,3000.0
[cloudera@quickstart ~]$ █
```



# Q & A



## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.





Query

Search data and saved documents...



Impala



Add a name...

Add a description...

&lt; default

Tables

(7) ▾ + ↻

categories

course

customers

departments

order\_items

orders

products

1| drop table course;



✓ Success.

Query History



Saved Queries



a few seconds ago



drop table course

# Writing from Flume to HDFS

- cd /usr/lib/flume-ng
- If you get this error: log4j:ERROR setFile(null,true) call failed.
- java.io.FileNotFoundException: /var/log/flume-ng/flume.log (Permission denied)
- Command: sudo chmod 777 –R /var/log/flume-ng/
- bin/flume-ng agent --n TwitterAgent --conf conf --f twitter.conf

A screenshot of a web browser displaying the Twitter Developer Apps page at developer.twitter.com/en/apps. The page has a purple header bar with various links: Developer, Use cases, Solutions, Products, Docs, Community, Updates, Support, Apply (which is circled in blue), Apps, and a user profile icon. Below the header, there's a navigation bar with 'Apps' selected (underlined) and a 'Create an app' button. The main content area displays the message 'No apps here.' followed by a descriptive text: 'You'll need an app and API key in order to authenticate and integrate with most Twitter developer products. Create an app to get your API key.'

← → C 🔒 developer.twitter.com/en/apps

Developer Use cases Solutions Products Docs Community Updates Support **Apply** Apps

**Apps**

Create an app

**No apps here.**

You'll need an app and API key in order to authenticate and integrate with most Twitter developer products. Create an app to get your API key.



Get started with Twitter APIs and tools

# Apply for access

---

All new developers must apply for a developer account to access Twitter APIs. Once approved, you can begin to use our standard APIs and our new premium APIs.

[Apply for a developer account](#)[Restricted used cases >](#)



Apps

Create an app

No apps here.

You'll need an app and API key in order to authenticate and integrate with most Twitter developer products. Create an app to get your API key.

# Get access to the Twitter API

Twitter @username > Organization > Intended use



## Team developer account

You are signing up for a team developer account.

These are typically used for:

**companies**  
**organizations**  
**educators**  
**group collaboration**

If you do not think you will need to invite other people to your account in the future to share API access or apps, you can [create an individual developer account](#) instead.



## #Welcome to the Twitter Developer Platform

Let's get you some keys. But first, you'll need to name your App. The App name needs to be unique. Don't take it too seriously, you can always change it later.

MSIS405.L11.CTTT.2020

11

Get keys

# Here are your keys.

These verify and allow you to make requests to the Twitter API.

## API key (i)

TrLcMgECOFO9ZqWacnPAzVl16



## API secret key (i)

O1JPS0xwFy1DCF0FQ2G3ZTXVGg5ycbTX6UdiN8eShyUTMa6wUC



## Bearer token (i)

AAAAAAAAAAAAAAAAAAAAANbglAEAAAAAmcR9CB84oZzT8JQ7jpnHKZJWxD8%3DdIP8ATjSv0ixEnNAkgRCPBI9L9Sn0o61X0g1G5m1d4iF14RAMD



### Here are your new key & secret. Have you saved them?

For security, we will be hiding these starting 01/12/2021. If something happens, you can always regenerate them.

API key: lbIT1MeVICsQDrhePxrS9icnL



API key secret: A6gJu7OS3y1ljLAScGL0LnH0RN9GfZnSxVTJNYoLtuiMOurGLx



[Yes, I saved them](#)

### Here are your new access token & secret. Have you saved them?

For security, this will be the last time we'll display these. If something happens, you can always regenerate them.

Access token: 1310978302480322560-fEkGm2r2cyc2NbjoGc9Lj68b4CsIM



Access token secret: gWCGF4QrKhcLCTs40hFax7ZNlcNvcdJLALbWtniBW0AN



[Yes, I saved them](#)

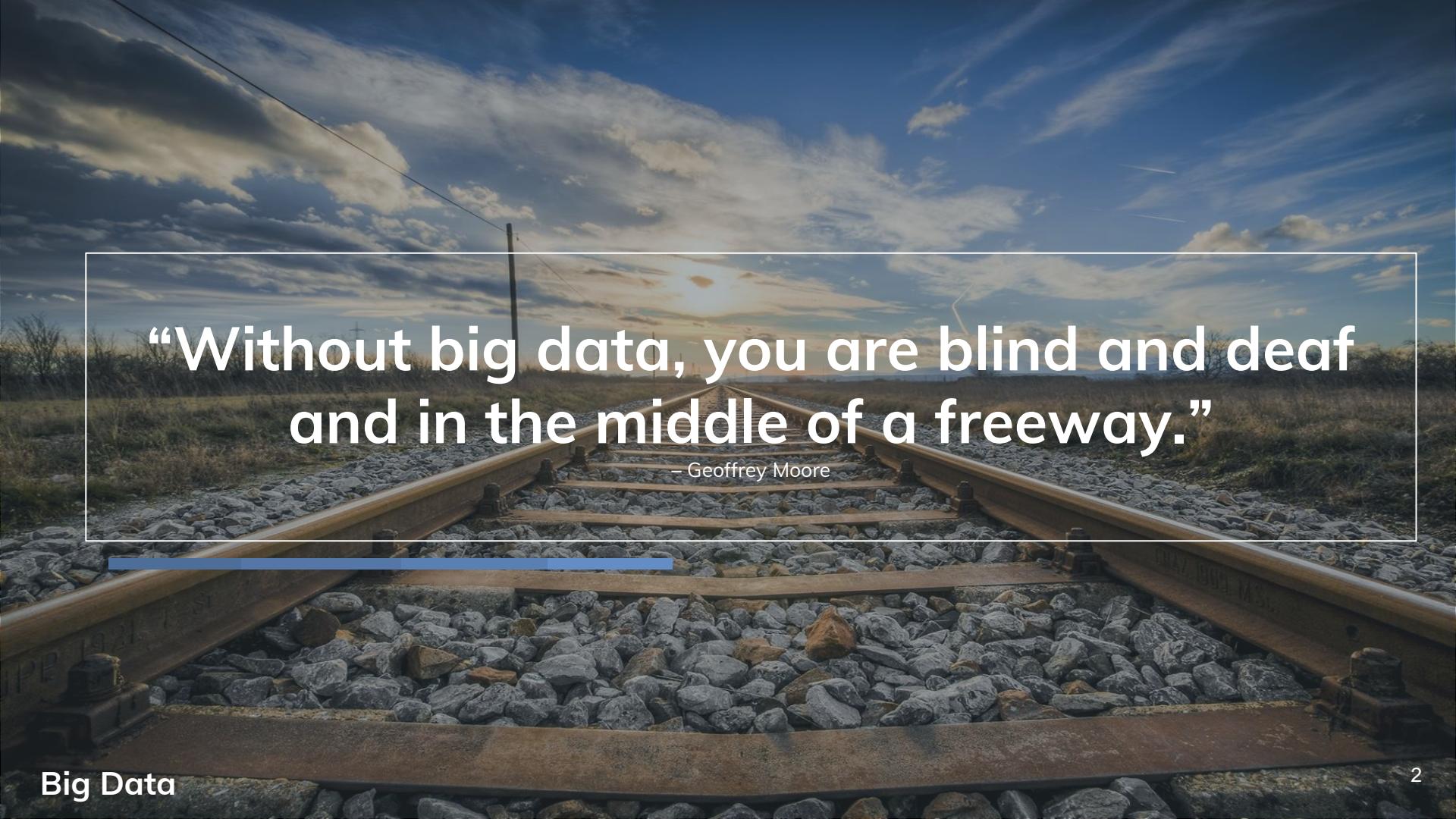
# Big Data

## Data Loading Tools

Trong-Hop Do

**S<sup>3</sup>Lab**

*Smart Software System Laboratory*

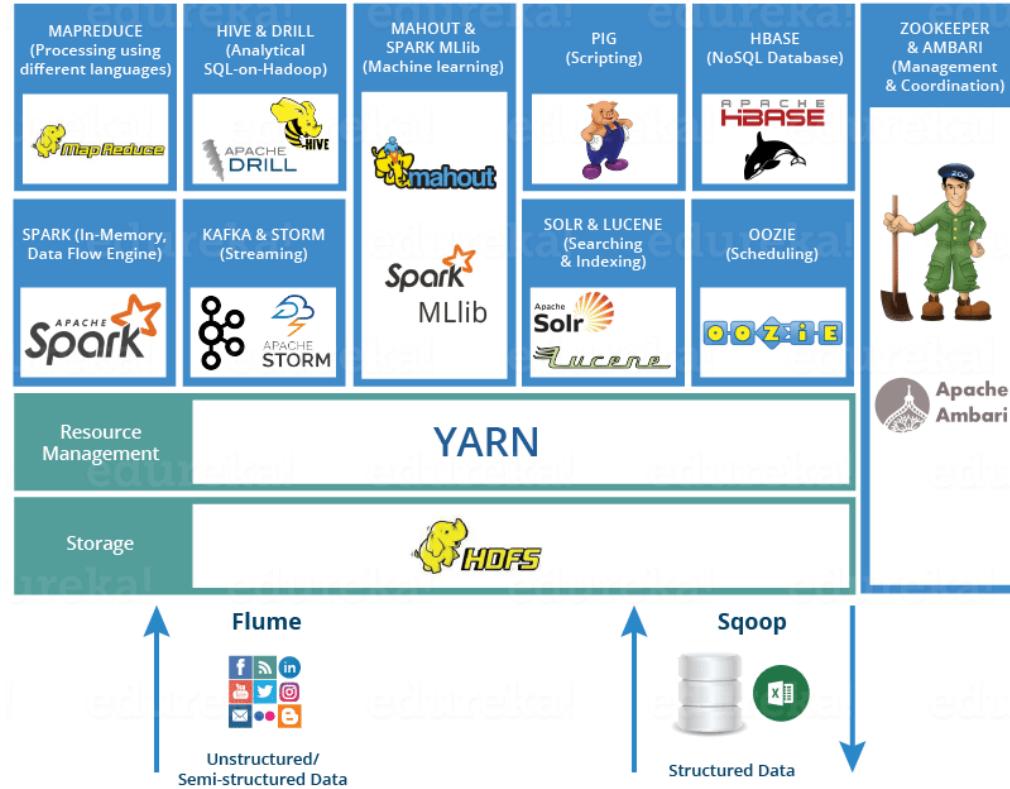


“Without big data, you are blind and deaf  
and in the middle of a freeway.”

– Geoffrey Moore



# Hadoop Ecosystem



# Apache Hive Tutorial

---



# High-Level Data Process Components

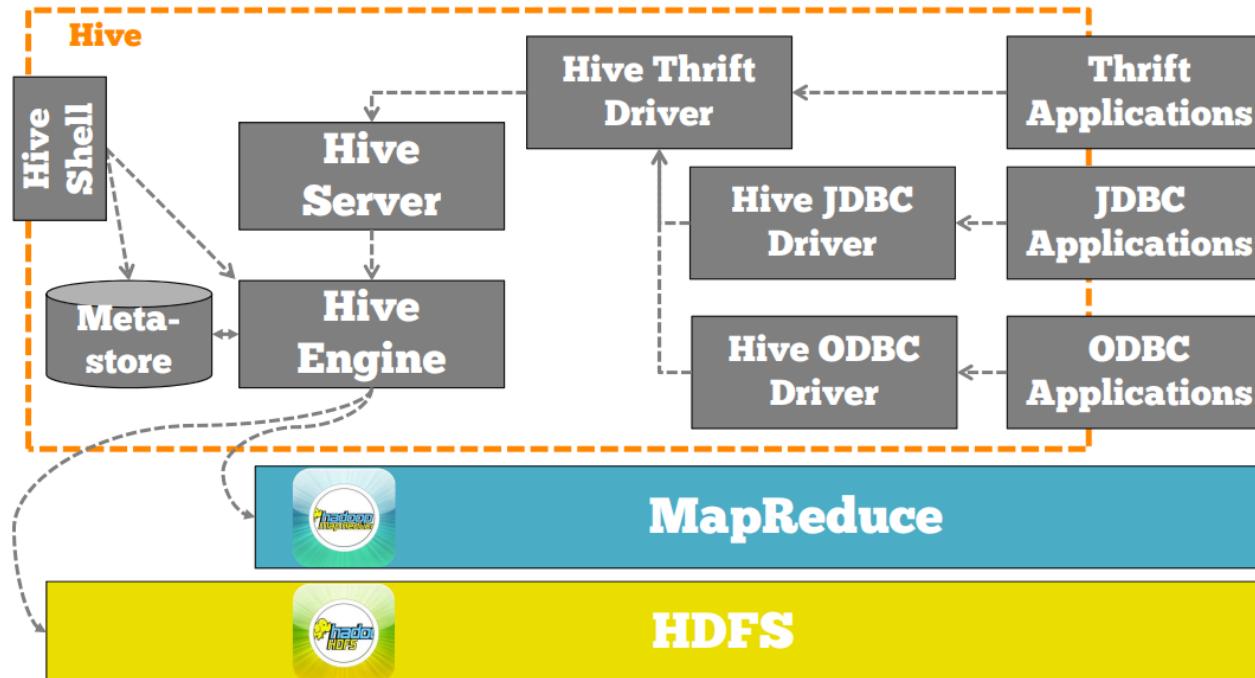
## Hive

- An **sql** like **interface** to Hadoop.
- Data warehouse infrastructure **built on top** of Hadoop
- Provide data **summarization, query** and **analysis**
- Query execution via **MapReduce**
- Hive interpreter convert the query to Map-reduce format.
- Open source project.
- Developed by Facebook
- Also used by Netflix, Cnet, Digg, eHarmony etc.



# High-Level Data Process Components

## Hive - architecture





# High-Level Data Process Components

## Hive

- HiveQL example:

```
SELECT customerId, max(total_cost) from hive_purchases GROUP BY  
customerId HAVING count(*) > 3;
```

# Hive tutorial

Let us get started with Command Line Interface(CLI)

Command: **hive**

```
[cloudera@quickstart ~]$ hive  
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-commo  
n-1.1.0-cdh5.13.0.jar!/hive-log4j.properties  
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.  
hive> █
```

# Hive tutorial

Create and show databases;

Command: **create database newdb;**

```
hive> create database newdb;
OK
Time taken: 0.398 seconds
hive> █
```

Command: **show databases;**

```
hive> show databases;
OK
default
newdb
Time taken: 0.168 seconds, Fetched: 2 row(s)
hive> █
```

# Hive tutorial

Two types of table in Hive: managed table and external table

- For managed table, Hive is responsible for managing the data of a managed table. If you load the data from a file present in HDFS into a Hive *Managed Table* and issue a DROP command on it, the table along with its metadata will be deleted. So, the data belonging to the dropped *managed\_table* no longer exist anywhere in HDFS and you can't retrieve it by any means. Basically, you are moving the data when you issue the LOAD command from the HDFS file location to the Hive warehouse directory.
- For *external table*, Hive is not responsible for managing the data. In this case, when you issue the LOAD command, Hive moves the data into its warehouse directory. Then, Hive creates the metadata information for the external table. Now, if you issue a DROP command on the *external table*, only metadata information regarding the external table will be deleted. Therefore, you can still retrieve the data of that very external table from the warehouse directory using HDFS commands.

# Hive tutorial

## Create managed table (internal table)

```
hive> create table employee (ID int, name string, Salary float, Age int)
      > row format delimited
      > fields terminated by ','
      > ;
OK
Time taken: 0.186 seconds
```

```
hive> describe employee;
OK
id                  int
name                string
salary              float
age                 int
Time taken: 0.058 seconds, Fetched: 4 row(s)
```

# Hive tutorial

## Describe table

```
hive> describe formatted employee;
OK
# col_name          data_type      comment
id                  int
name                string
salary              float
age                 int

# Detailed Table Information
Database:          default
Owner:              cloudera
CreateTime:         Mon Oct 12 05:27:24 PDT 2020
LastAccessTime:     UNKNOWN
Protect Mode:       None
Retention:          0
Location:           hdfs://quickstart.cloudera:8020/user/hive/warehouse/employee
Table Type:         MANAGED_TABLE ➔
Table Parameters:
    transient_lastDdlTime 1602505644

# Storage Information
Serde Library:      org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:         org.apache.hadoop.mapred.TextInputFormat
OutputFormat:        org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:         No
Num Buckets:        -1
Bucket Columns:     []
Sort Columns:        []
Storage Desc Params:
    field.delim      ,
    serialization.format   ,
Time taken: 0.063 seconds, Fetched: 30 row(s)
```

# Hive tutorial

## Create external table

- Let's try to create some external table

```
hive> create external table employee2 (ID int, name string, Salary float, Age int)
      > row format delimited
      > fields terminated by ','
      > stored as textfile;
OK
Time taken: 0.05 seconds
hive> describe employee2;
OK
id                  int
name                string
salary              float
age                 int
Time taken: 0.07 seconds, Fetched: 4 row(s)
```

# Hive tutorial

```
hive> describe formatted employee2;
OK
# col_name          data_type          comment
id                  int
name                string
salary              float
age                 int

# Detailed Table Information
Database:          default
Owner:              cloudera
CreateTime:        Mon Oct 12 05:35:01 PDT 2020
LastAccessTime:    UNKNOWN
Protect Mode:      None
Retention:         0
Location:          hdfs://quickstart.cloudera:8020/user/hive/warehouse/employee2
Table Type:        EXTERNAL_TABLE
Table Parameters:
                    EXTERNAL          TRUE
                    transient_lastDdlTime  1602506101

# Storage Information
SerDe Library:     org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:        org.apache.hadoop.mapred.TextInputFormat
OutputFormat:       org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:         No
Num Buckets:       -1
Bucket Columns:    []
Sort Columns:       []
Storage Desc Params:
                    field.delim      ,
                    serialization.format   ,
Time taken: 0.06 seconds, Fetched: 31 row(s)
```

# Hive tutorial

- Let's check the directory `/user/hive/warehouse` in HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse
Found 3 items
drwxrwxrwx  - cloudera supergroup      0 2020-10-12 05:27 /user/hive/warehouse/employee
drwxrwxrwx  - cloudera supergroup      0 2020-10-12 05:35 /user/hive/warehouse/employee2
drwxrwxrwx  - cloudera supergroup      0 2020-10-12 05:25 /user/hive/warehouse/newdb.db
[cloudera@quickstart ~]$ █
```

# Hive tutorial

- Let's create new external table and store data in home directory of HDFS

```
hive> create external table employee3 (ID int, name string, Salary float, Age int)
      > row format delimited
      > fields terminated by ','
      > location '/user/cloudera/emp';
OK
Time taken: 0.062 seconds
```

# Hive tutorial

- Let's rename the table and check the result

```
hive> Alter table employee3 RENAME TO emptable;
OK
Time taken: 0.126 seconds
```

```
hive> describe emptable;
OK
id                  int
name                string
salary              float
age                 int
Time taken: 0.068 seconds, Fetched: 4 row(s)
```

# Hive tutorial

- Let's add one more column to the table and check the result

```
|hive> Alter table emptable add columns (surname string);  
|OK  
|Time taken: 0.113 seconds
```

```
|hive> describe emptable;  
|OK  
|id          int  
|name        string  
|salary      float  
|age         int  
|surname    string  
|Time taken: 0.064 seconds, Fetched: 5 row(s)
```

# Hive tutorial

- Let's change one column of the table and check the result

```
hive> Alter table emptable change name first_name string;  
OK  
Time taken: 0.084 seconds
```

```
hive> describe emptable;  
OK  
id                  int  
first_name          string  
salary              float  
age                 int  
surname             string  
Time taken: 0.07 seconds, Fetched: 5 row(s)
```

# Hive tutorial

## LOAD Data from Local into Hive Managed Table

- Command: LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/Employee.csv' INTO TABLE employee;

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/Employee.csv' INTO TABLE employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, totalSize=172]
OK
Time taken: 0.649 seconds
```

```
hive> select * from employee;
OK
NULL      name    NULL      NULL    ← Why NULL?
20021     Le       1500.0   22
20022     Pham    1000.0   23
20023     Tran    2000.0   24
20024     Vu      3000.0   25
20025     Dao     2500.0   21
20026     Nam    2500.0   22
20027     Viet   3000.0   24
20028     Quoc   1000.0   22
Time taken: 0.292 seconds, Fetched: 9 row(s)
```

# Hive tutorial

## LOAD Data from Local into Hive Managed Table

- Check the schema of the table and the .csv file

```
hive> describe employee;
OK
id          int
name        string
salary      float
age         int
Time taken: 0.082 seconds, Fetched: 4 row(s)
hive> 
```

```
[cloudera@quickstart ~]$ cat /home/cloudera/Desktop/Employee.csv
ID,name,Salary,Age
20021,Le,1500,22
20022,Pham,1000,23
20023,Tran,2000,24
20024,Vu,3000,25
20025,Dao,2500,21
20026,Nam,2500,22
20027,Viet,3000,24
20028,Quoc,1000,22[cloudera@quickstart ~]$ 
```

# Hive tutorial

## LOAD Data from Local into Hive Managed Table

```
hive> drop table employee;
OK
Time taken: 0.074 seconds
hive> create table employee (ID int, name string,Salary float, Age int)
    > row format delimited
    > fields terminated by ','
    > ;
OK
Time taken: 0.051 seconds
```

*Employee.csv			
20021	Le	1500	22
20022	Pham	1000	23
20023	Tran	2000	24
20024	Vu	3000	25
20025	Dao	2500	21
20026	Nam	2500	22
20027	Viet	3000	24
20028	Quoc	1000	22

# Hive tutorial

## LOAD Data from Local into Hive Managed Table

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/Employee.csv' INTO TABLE employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, totalSize=153]
OK
Time taken: 0.154 seconds
hive> select * from employee;
OK
20021    Le      1500.0  22
20022    Pham    1000.0  23
20023    Tran    2000.0  24
20024    Vu      3000.0  25
20025    Dao     2500.0  21
20026    Nam     2500.0  22
20027    Viet    3000.0  24
20028    Quoc    1000.0  22
Time taken: 0.038 seconds, Fetched: 8 row(s)
```

# Hive tutorial

## LOAD Data from Local into Hive External Table

- Command: LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/Employee.csv' INTO TABLE employee2;

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/Employee.csv' INTO TABLE employee2;
Loading data to table default.employee2
Table default.employee2 stats: [numFiles=1, totalSize=153]
OK
Time taken: 0.155 seconds
```

```
hive> select * from employee2;
OK
20021    Le      1500.0  22
20022    Pham    1000.0  23
20023    Tran    2000.0  24
20024    Vu      3000.0  25
20025    Dao     2500.0  21
20026    Nam     2500.0  22
20027    Viet    3000.0  24
20028    Quoc    1000.0  22
Time taken: 0.041 seconds, Fetched: 8 row(s)
```

# Hive tutorial

## Difference between managed and external table

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse
Found 3 items
drwxrwxrwx  - cloudera supergroup          0 2020-10-12 05:27 /user/hive/warehouse/employee
drwxrwxrwx  - cloudera supergroup          0 2020-10-12 05:35 /user/hive/warehouse/employee2
drwxrwxrwx  - cloudera supergroup          0 2020-10-12 05:25 /user/hive/warehouse/newdb.db
[cloudera@quickstart ~]$ █
```

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/employee
Found 1 items
-rwxrwxrwx  1 cloudera supergroup      153 2020-10-12 07:26 /user/hive/warehouse/employee/Employee.csv
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/employee2
Found 1 items
-rwxrwxrwx  1 cloudera supergroup      153 2020-10-12 07:31 /user/hive/warehouse/employee2/Employee.csv
```

# Hive tutorial

## Difference between managed and external table

- Let's drop the external table

```
hive> drop table employee2;
OK
Time taken: 0.054 seconds
hive> select * from employee2;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'employee2'
```

- Then check the directory associated with this external table in HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/employee2
Found 1 items
-rwxrwxrwx  1 cloudera supergroup      153 2020-10-12 07:31 /user/hive/warehouse/employee2/Employee.csv
```

# Hive tutorial

## Difference between managed and external table

- Let's drop the internal table

```
hive> drop table employee;
OK
Time taken: 0.091 seconds
hive> select * from employee;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'employee'
```

- Then check the directory associated with this internal table in HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/employee
ls: `/user/hive/warehouse/employee': No such file or directory
```

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse
Found 3 items
-rw-r--r-- 1 cloudera supergroup      1381 2020-10-12 06:05 /user/hive/warehouse/StudentReport.csv
drwxrwxrwx  - cloudera supergroup          0 2020-10-12 07:31 /user/hive/warehouse/employee2
drwxrwxrwx  - cloudera supergroup          0 2020-10-12 05:25 /user/hive/warehouse/newdb.db
```

# Hive tutorial

## LOAD Data from HDFS to Hive Table

- Let's create some internal table

```
hive> create table student (ID int, Name string,Course string, Age int)
      > row format delimited fields terminated by ',' tblproperties('skip.header.line.count'='1');
OK
Time taken: 0.07 seconds
```



declare this Hive's property to skip the header in Student.csv file

```
hive> describe student;
OK
id                  int
name                string
course              string
age                 int
Time taken: 0.04 seconds, Fetched: 4 row(s)
```

# Hive tutorial

## LOAD Data from HDFS to Hive Table

- Put file Student.csv to HDFS

```
[cloudera@quickstart ~]$ cd /home/cloudera/Desktop
[cloudera@quickstart Desktop]$ ls
Department.csv  Employee2.csv  Employee.csv~      Express.desktop  Parcels.desktop  StudentReport.csv
Eclipse.desktop  Employee.csv  Enterprise.desktop  Kerberos.desktop  Student.csv
[cloudera@quickstart Desktop]$ hdfs dfs -put Student.csv
[cloudera@quickstart Desktop]$ hdfs dfs -ls
Found 2 items
-rw-r--r--  1 cloudera cloudera      249 2020-10-12 08:01 Student.csv
drwxr-xr-x  - cloudera cloudera          0 2020-10-12 05:46 emp
```

# Hive tutorial

## LOAD Data from HDFS to Hive Table

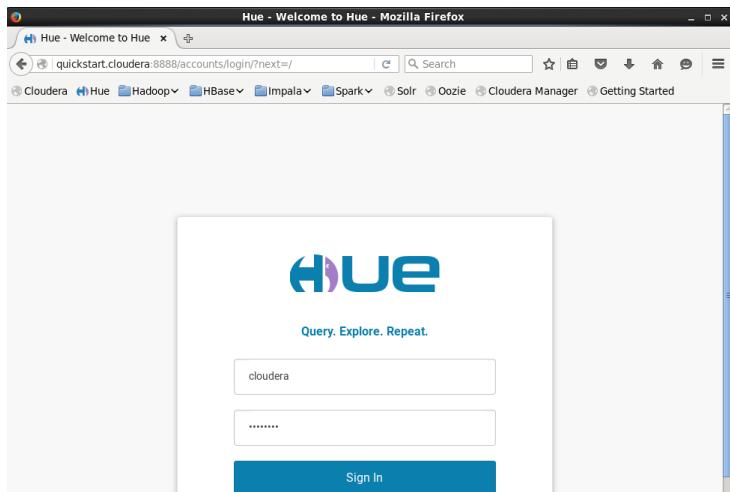
```
hive> load data inpath 'Student.csv' into table student;
Loading data to table default.student
Table default.student stats: [numFiles=1, totalSize=249]
OK
Time taken: 0.217 seconds
```

```
hive> select * from student;
OK
123451 Quynh    Hadoop   22
123452 Tai      Java     22
123453 Truong   Python   23
123454 Nghia   Hadoop   24
123455 Thuy     Java     23
123456 Hao      Python   24
123457 Hien    Hadoop   22
123458 Phuong   Java     23
123459 Hai      Python   23
123460 Phuong   Hadoop   24
Time taken: 0.042 seconds, Fetched: 10 row(s)
```

# Hive tutorial

## Hive command using HUE

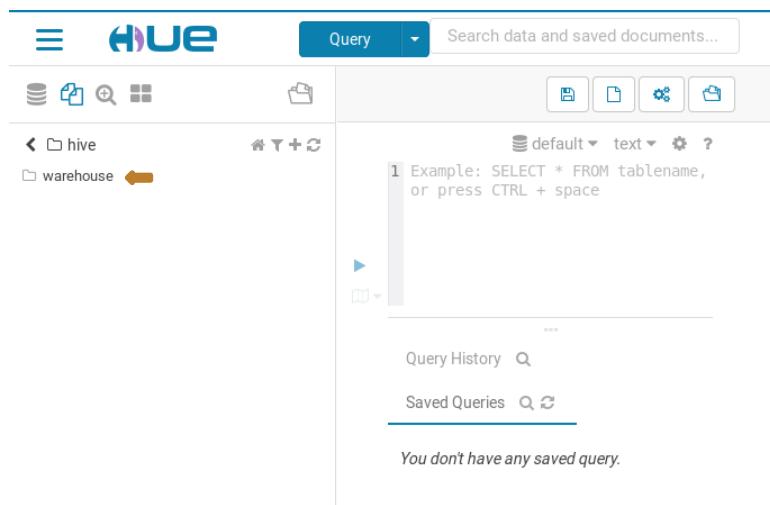
- Login to HUE



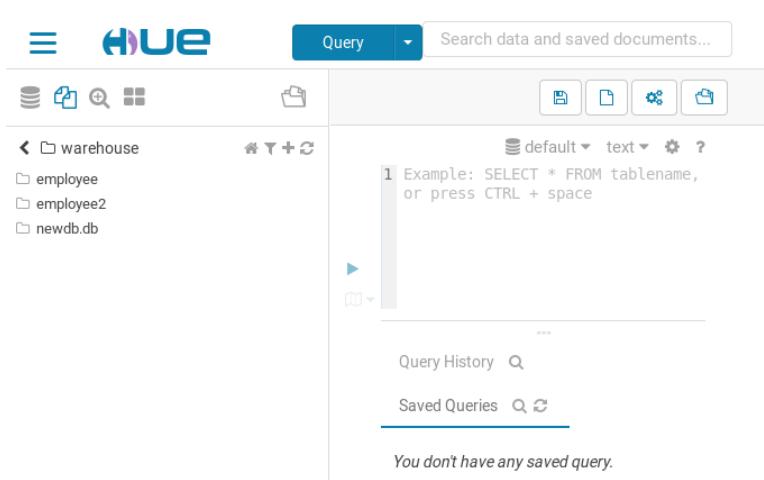
A screenshot of the Hue home interface. At the top, there's a navigation bar with icons for Cloudera, Hue, Hadoop, HBase, Impala, Spark, Solr, Oozie, Cloudera Manager, and Getting Started. The main area has a header 'hue' with a search bar 'Search data and saved documents...'. On the left is a sidebar with a tree view of the environment: 'user' (selected), 'cloudera', 'history', 'hive' (with an orange arrow pointing to it), 'hue', 'jenkins', 'oozie', 'root', and 'spark'. To the right of the sidebar is a 'Query' section with a text input field containing 'default text', a help message 'Example: SELECT \* FROM tablename, or press CTRL + space', and buttons for 'Run', 'Save', 'Edit', and 'Delete'. Below the query section is a 'Query History' section with a search bar and a message 'You don't have any saved query.'.

# Hive tutorial

## Hive command using HUE



The screenshot shows the Hue interface for Hive. The top navigation bar includes the Hue logo, a 'Query' dropdown, and a search bar. On the left, there's a sidebar with icons for databases (hive, warehouse), a file browser, and a help section. The main area displays a query editor with a placeholder message: 'Example: SELECT \* FROM tablename, or press CTRL + space'. Below the editor are sections for 'Query History' and 'Saved Queries'. A note at the bottom states: 'You don't have any saved query.'



This screenshot shows the Hue interface after navigating to the 'warehouse' database. The sidebar now lists 'warehouse' as the active database, along with 'employee', 'employee2', and 'newdb.db'. The main query editor area remains the same as in the previous screenshot, with the placeholder message: 'Example: SELECT \* FROM tablename, or press CTRL + space'. Below the editor are sections for 'Query History' and 'Saved Queries'. A note at the bottom states: 'You don't have any saved query.'

# Hive tutorial

## Hive command using HUE

- Let's check the file Employee.csv

The screenshot shows the Hue web interface. At the top, there is a navigation bar with icons for Home, Search, and Jobs, along with a Cloudera logo. Below the navigation bar, the main area has a sidebar on the left containing icons for databases, queries, and files, and a list of files under 'employee2' including 'Employee.csv'. The main content area shows the details for 'Employee.csv'. It includes a breadcrumb trail: Home > Page 1 to 1 of 1. Below the breadcrumb, the full file path is shown: / user / hive / warehouse / employee2 / Employee.csv. The file content is displayed as a list of CSV rows:

ID	Name	Salary	Age
20021	Le	1500	22
20022	Pham	1000	23
20023	Tran	2000	24
20024	Vu	3000	25
20025	Dao	2500	21
20026	Nam	2500	22
20027	Viet	3000	24
20028	Quoc	1000	22

# Hive tutorial

## Hive command using HUE

- Let's check the file Student.csv

The screenshot shows the Hue web interface. At the top, there is a navigation bar with icons for Home, Search, and Jobs, along with a Cloudera logo. Below the navigation bar, the main area has a sidebar on the left containing a tree view with a 'student' folder expanded, showing 'Student.csv'. The main content area displays the contents of 'Student.csv' in a table format. The table has columns for 'ID', 'Name', 'Course', and 'Age'. The data rows are:

ID	Name	Course	Age
123451	Quynh	Hadoop	22
123452	Tai	Java	22
123453	Truong	Python	23
123454	Nghia	Hadoop	24
123455	Thuy	Java	23
123456	Hao	Python	24
123457	Hien	Hadoop	22
123458	Phuong	Java	23
123459	Hai	Python	23

# Hive tutorial

## Hive command using HUE

- Let's make some Hive query

The screenshot shows the Hue web interface. On the left, there is a sidebar with icons for databases, tables, and queries, and a list of 'student' tables containing 'Student.csv'. The main area is titled 'Query' and has a search bar. A query is being typed into the editor:

```
1 select name from student where age > 23;
```

The results section shows the output of the query:

name
1 Nghia
2 Hao
3 Phuong

# Hive tutorial

## Hive command using HUE

- Let's make some Hive query

The screenshot shows the Hue web interface for Apache Hive. The top navigation bar includes 'Query' (selected), a search bar, and links for 'Jobs', 'cloudera', and other system icons. The main area has tabs for 'Hive' (selected) and 'Add a name...', 'Add a description...'. On the left, there's a sidebar with icons for tables, queries, and saved documents, and a list of tables: 'default' (selected), 'department', 'emptable', 'student'. The 'student' table is detailed with columns: 'id (int)', 'name (string)', 'course (string)', and 'age (int)'. The central workspace shows a query editor with the following content:

```
1| INSERT INTO TABLE student VALUES (123461, "Anh", "Java", 22);
```

Execution details: 25.4s, default, text. A success message at the bottom says: "Success."

# Hive tutorial

## Hive command using HUE

- Check newly created file in HDFS

The screenshot shows the Hue interface with the following details:

- Header:** Includes the Hue logo, a "Query" dropdown, a search bar ("Search data and saved documents..."), and links for "Jobs" and "cloudera".
- Left Sidebar:** Shows a file tree with the following structure:
  - student
    - .hive-staging\_hive\_2020-10-12\_18-01-40\_929\_17508991635335906
    - 000000\_0 (highlighted with a yellow arrow)
    - Student.csv
- File Browser Panel:** Displays the contents of the selected file "000000\_0".
  - Actions:** View as binary, Edit file, Download, View file location, Refresh.
  - Location:** Home / user / hive / warehouse / student / 000000\_0
  - Content Preview:** 123461 , Anh, Java, 22

# Hive tutorial

## Hive command using HUE

- Let's create some table in HUE
- Command: **create table department (DepartmentID int, DepartmentName string) row format delimited fields terminated by ',' tblproperties('skip.header.line.count'='1');**

The screenshot shows the Hue web interface for Apache Hive. At the top, there is a navigation bar with icons for HDFS, Hue, and a search bar labeled "Search data and saved documents...". On the right side of the header, there are links for "Jobs", a clock icon, and "cloudera". Below the header, the main area has tabs for "Hive" and "Add a name...". There are also buttons to "Add a description..." and a set of icons for file operations. On the left, a sidebar shows a tree view of the "warehouse" database, containing "employee2", "newdb.db", and "student". The main workspace contains the following code:

```
1 create table department (DepartmentID int, DepartmentName string)
2 row format delimited fields terminated by ',' tblproperties('skip.header.line.count'='1');
```

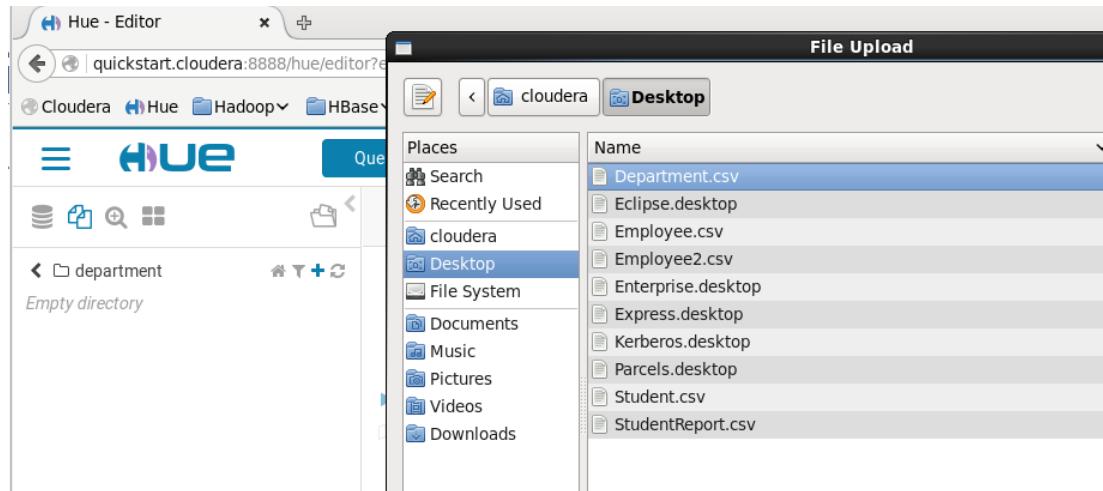
Below the code, a message says "Success." and there are links for "Query History" and "Saved Queries". A recent query is listed at the bottom:

a few seconds ago ➔ create table department (DepartmentID int, DepartmentName string)  
row format delimited fields terminated by ','  
tblproperties('skip.header.line.count'='1')

# Hive tutorial

## Hive command using HUE

- Upload file Department.csv in to department directory (use + button)



# Hive tutorial

## Hive command using HUE

- Query this newly created table

The screenshot shows the Hue web interface for Apache Hive. At the top, there's a navigation bar with icons for Home, Search, and Jobs, along with a user icon for 'cloudera'. Below the navigation is a toolbar with various icons for file operations like Open, Save, and Refresh.

The main area is divided into sections: 'department' (containing 'Department.csv') and 'Hive'. In the 'Hive' section, there's a search bar and fields for 'Add a name...' and 'Add a description...'. A query editor window contains the following code:

```
1|select * from department;
```

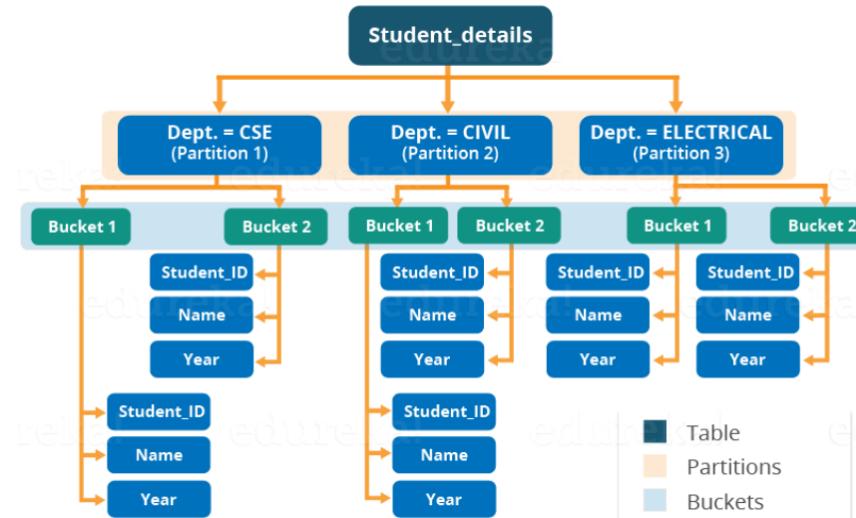
Below the editor, there are buttons for 'Run' and 'Cancel'. The results section shows a table with two columns: 'department.departmentid' and 'department.departmentname'. The data is as follows:

department.departmentid	department.departmentname
1 20021	Development
2 20022	Testing
3 20024	Product
4 20023	Relationship
5 20025	Admin and IT Support

# Hive tutorial

## Partition in Hive

- Hive organizes tables into partitions for grouping similar type of data together based on a column or partition key. Each Table can have one or more partition keys to identify a particular partition. This allows us to have a faster query on slices of the data.



# Hive tutorial

## Static partition

- Create new database

```
hive> create database studentdb;  
OK  
Time taken: 0.054 seconds  
hive> use studentdb;  
OK  
Time taken: 0.019 seconds
```

- Create new table with partition

```
hive> create table student (ID int, Name string, Age int) partitioned by (Course string)  
      > row format delimited fields terminated by ',' tblproperties('skip.header.line.count'='1');  
OK  
Time taken: 0.047 seconds
```

ID	Name	Age	Course
123451	Quynh	22	Hadoop
123452	Tai	22	Java
123453	Truong	23	Python
123454	Nghia	24	Hadoop
123455	Thuy	23	Java
123456	Hao	24	Python
123457	Hien	22	Hadoop
123458	Phuong	23	Java
123459	Hai	23	Python
123460	Phuong	24	Hadoop

# Hive tutorial

## Static partition

- Check the format of the table

```
hive> describe student;
OK
id                  int
name                string
age                 int
course              string

# Partition Information
# col_name          data_type          comment
course              string
Time taken: 0.055 seconds, Fetched: 9 row(s)
```

# Hive tutorial

## Static partition

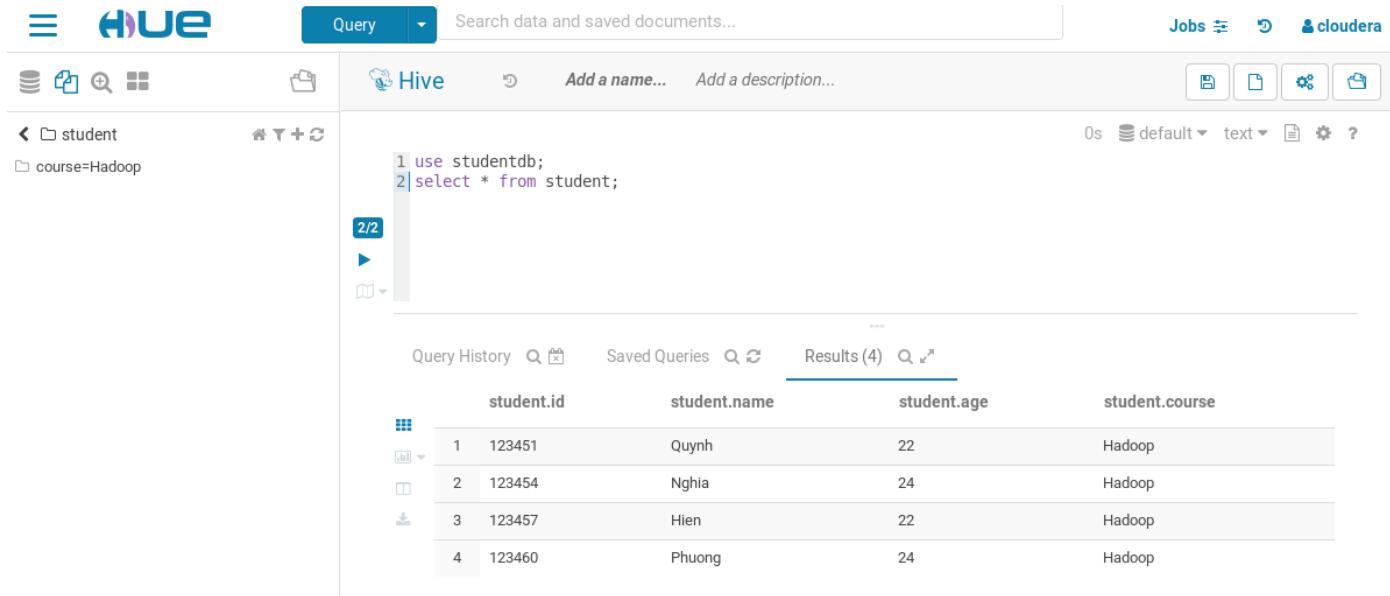
- Load data from file **StudentHadoop.csv** to partition (course=Hadoop)

```
hive> load data local inpath '/home/cloudera/Desktop/StudentHadoop.csv' into table student partition(course= "Hadoop");
Loading data to table studentdb.student partition (course=Hadoop)
Partition studentdb.student{course=Hadoop} stats: [numFiles=1, numRows=0, totalSize=116, rawDataSize=0]
OK
Time taken: 0.163 seconds
```

# Hive tutorial

## Static partition

- Check new directory **course=Hadoop** in HDFS



The screenshot shows the Hue web interface for Apache Hive. At the top, there's a navigation bar with icons for Home, Applications, and Help, followed by the Hue logo and a search bar labeled "Search data and saved documents...". On the right side of the header are links for "Jobs", a user icon, and "cloudera". Below the header, the main area has tabs for "Hive" (which is selected), "Add a name...", and "Add a description...". To the left, there's a sidebar with a tree view showing a "student" folder and a "course=Hadoop" folder under it. The main content area displays a query window with the following code:

```
1 use studentdb;
2 select * from student;
```

Below the code, it says "2/2" and has navigation arrows. The results section is titled "Results (4)" and shows the following table:

	student.id	student.name	student.age	student.course
1	123451	Quynh	22	Hadoop
2	123454	Nghia	24	Hadoop
3	123457	Hien	22	Hadoop
4	123460	Phuong	24	Hadoop

# Hive tutorial

## Static partition

- Continue to load data to other partition

```
hive> load data local inpath '/home/cloudera/Desktop/StudentJava.csv' into table student partition(course= "Java");
Loading data to table studentdb.student partition (course=Java)
Partition studentdb.student{course=Java} stats: [numFiles=1, numRows=0, totalSize=82, rawDataSize=0]
OK
Time taken: 0.227 seconds
hive> load data local inpath '/home/cloudera/Desktop/StudentPython.csv' into table student partition(course= "Python");
Loading data to table studentdb.student partition (course=Python)
Partition studentdb.student{course=Python} stats: [numFiles=1, numRows=0, totalSize=89, rawDataSize=0]
OK
Time taken: 0.27 seconds
```

# Hive tutorial

## Static partition

- Check all created directories in HDFS

The screenshot shows the Hue web interface. At the top, there's a navigation bar with icons for Home, Logout, and Cloudera. Below it is a search bar labeled "Search data and saved documents...". The main area is titled "Hive" and has buttons for "Add a name..." and "Add a description...". On the left, there's a sidebar with a tree view of datasets: "student", "course=Hadoop", "course=Java", and "course=Python". The main pane contains a text input field with the placeholder "Example: SELECT \* FROM tablename, or press CTRL + space". There are also icons for a play button, a bookmark, and other query-related functions.

# Hive tutorial

## Dynamic partition

- Create new database

```
hive> create database newstudentdb;
OK
Time taken: 0.027 seconds
hive> use newstudentdb;
OK
Time taken: 0.009 seconds
hive> set hive.exec.dynamic.partition=true;
hive> set hive.exec.dynamic.partition.mode=nonstrict;
```

# Hive tutorial

## Dynamic partition

- Create table student (same like before)

```
hive> create table student (ID int, Name string, Course string, Age int) row format delimited fields terminated by ','  
tblproperties('skip.header.line.count='1');  
OK  
Time taken: 0.054 seconds  
hive> describe student;  
OK  
id          int  
name        string  
course      string  
age         int  
Time taken: 0.034 seconds, Fetched: 4 row(s)
```

# Hive tutorial

## Dynamic partition

- Load data to table student (same like before)

```
hive> load data local inpath '/home/cloudera/Desktop/Student.csv' into table student;
Loading data to table newstudentdb.student
Table newstudentdb.student stats: [numFiles=1, totalSize=249]
OK
Time taken: 0.127 seconds
```

# Hive tutorial

## Dynamic partition

- Create table student\_partition

```
hive> create table student_partition (ID int, Name string, Age int) partitioned by (Course string) row format delimited fields terminated by ',';
OK
Time taken: 0.036 seconds
hive> describe student_partition;
OK
id          int
name        string
age          int
course      string

# Partition Information
# col_name      data_type            comment
course      string
Time taken: 0.041 seconds, Fetched: 9 row(s)
```

# Hive tutorial

## Dynamic partition

- Command: `insert into student_partition partition(Course) select ID, Name, Age, Course from student;`

```
hive> insert into student_partition partition(Course) select ID, Name, Age, Course from student;
Query ID = cloudera_20201012112020_58eb9cc8-6a9a-4e6b-ba6f-a96fae6e366b
Total jobs = 3 ←
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1602505280766_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1602505280766_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1602505280766_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-10-12 11:20:33,403 Stage-1 map = 0%,  reduce = 0%
2020-10-12 11:20:41,979 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.05 sec
MapReduce Total cumulative CPU time: 1 seconds 50 msec
Ended Job = job_1602505280766_0003
```

```
Stage-Stage-1: Map: 1  Cumulative CPU: 1.35 sec  HDFS Read: 4800 HDFS Write: 369 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 350 msec
OK
Time taken: 24.234 seconds
```

# Hive tutorial

## Dynamic partition

- Check created directories in HDFS

The screenshot shows the Hue web interface for Apache Hive. The top navigation bar includes the Hue logo, a search bar, and a 'Jobs' section. On the left, there's a sidebar with icons for tables, queries, and saved documents, and a tree view of database structures under 'student\_partition'. The main area is titled 'Hive' and contains a query editor with the following code:

```
1 use newstudentdb;
2 select * from student_partition;
```

The status bar indicates '2/2' rows processed. Below the query editor is a 'Results (10)' tab, which displays the following table:

	student_partition.id	student_partition.name	student_partition.age	student_partition.course
1	123451	Quynh	22	Hadoop
2	123454	Nghia	24	Hadoop
3	123457	Hien	22	Hadoop
4	123460	Phuong	24	Hadoop
5	123452	Tai	22	Java

# Apache Pig Tutorial

---



# High-Level Data Process Components

## Pig

- A **scripting platform** for processing and analyzing large data sets
- Apache Pig allows to write complex **MapReduce programs** using a simple **scripting** language.
- Made of two components:
  - High level language: **Pig Latin** (data flow language).
  - Pig translate Pig Latin script into MapReduce to execute within Hadoop.
- Open source project
- Developed by Yahoo



# High-Level Data Process Components

## Pig

- Pig Latin example:

```
A = LOAD 'student' USING PigStorage() AS (name:chararray,  
age:int, gpa:float);
```

```
X = FOREACH A GENERATE name,$2;
```

```
DUMP X;
```



# Pig tutorial

## Pig data model

### 1. Basic Pig Data Types :

Data Types	Description	How can we use use Pig Data Types?
int	It is a Signed 32-bit integer value	Int can hold any integer value Ex: 23
long	It is a Signed 64-bit integer value	long can hold any Long value i.e bigger than Integer value and it can be displayed as 23L
float	It is a 32-bit floating point value	Float can be represented as 4.5F or 4.5.5f or 4.5e2f or 4.5E2F
double	It is a 64-bit floating point value	We can use this if the value is bigger than float and it can be represented as 08.5 or 08.5e2 or 08.5E2



# Pig tutorial

## Pig data model

### 1. Basic Pig Data Types :

Data Types	Description	How can we use use Pig Data Types?
chararray	It is a Character array (string) in Unicode UTF-8 format value	JavaChain.com
bytearray	The default datatype in pig is Byte array	
boolean	boolean represents true or false values.	It could be either true/false and it is case sensitive.
datetime	It displays the datetime	2016-01-14T00:00:00.000+00:00
biginteger	It displays the BigInteger	70409060802
bigdecimal	It displays the BigDecimal	198.78946646131311211



# Pig tutorial

## Pig data model

### 2.Complex Data Types

tuple	it is the collections of one or more fields	(Connecticut, Newjersey, Newyork)
bag	it is the collection of one or more tuples	{(Jack, Jill, JavaChain.com), (Connecticut, Newjersey, Newyork)}
map	It has Key and value pair data	[websitename#javachain.com]



# Pig tutorial

## Tuple and Bag

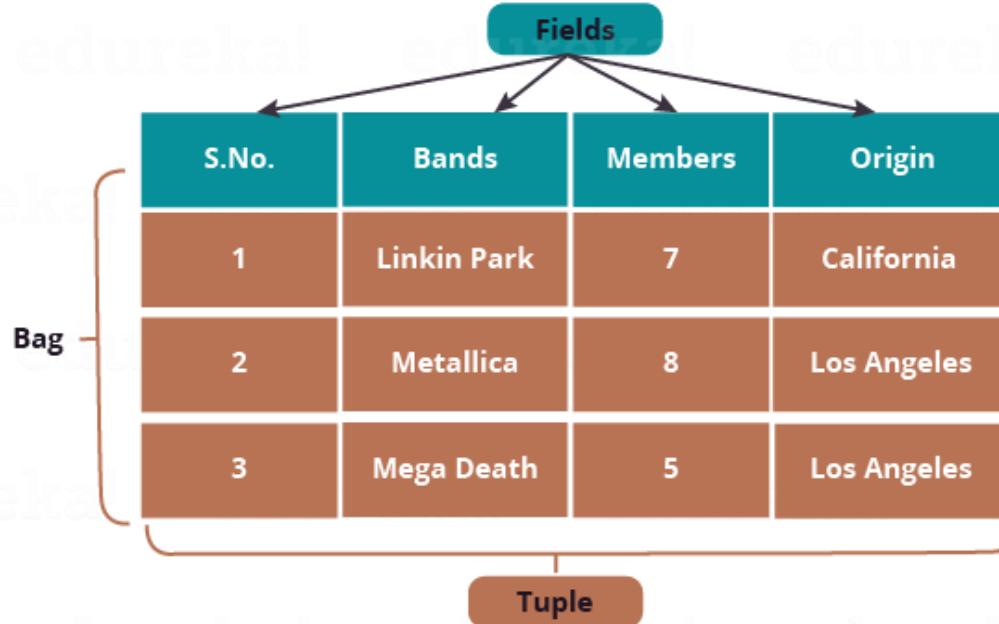


Figure: Apache Pig Data Model



# Pig tutorial

## Tuple

- Tuple is an ordered set of fields which may contain different data types for each field. You can understand it as the records stored in a row in a relational database. A Tuple is a set of cells from a single row as shown in the above image. The elements inside a tuple does not necessarily need to have a schema attached to it.
- A tuple is represented by '()' symbol.
- Example of tuple – (1, Linkin Park, 7, California)
- Since tuples are ordered, we can access fields in each tuple using indexes of the fields, like \$1 form above tuple will return a value 'Linkin Park'. You can notice that above tuple doesn't have any schema attached to it.



# Pig tutorial

## Bag

- A bag is a collection of a set of tuples and these tuples are subset of rows or entire rows of a table. A bag can contain duplicate tuples, and it is not mandatory that they need to be unique.
- The bag has a flexible schema i.e. tuples within the bag can have different number of fields. A bag can also have tuples with different data types.
- A bag is represented by ‘{}’ symbol.
- Example of a bag – {(Linkin Park, 7, California), (Metallica, 8), (Mega Death, Los Angeles)}



# Pig tutorial

## Bag

- For Apache Pig to effectively process bags, the fields and their respective data types need to be in the same sequence.
- Set of bags –
- `{(Linkin Park, 7, California), (Metallica, 8), (Mega Death, Los Angeles)}`,
- `{(Metallica, 8, Los Angeles), (Mega Death, 8), (Linkin Park, California)}`



# Pig tutorial

**Two types of Bag: Outer Bag and Inner Bag.**

- **Outer bag** or **relation** is noting but a bag of tuples. Here relations are similar as relations in relational databases. To understand it better let us take an example:
- $\{(Linkin\ Park, California), (Metallica, Los\ Angeles), (Mega\ Death, Los\ Angeles)\}$
- This above bag explains the relation between the Band and their place of Origin.



# Pig tutorial

## Pig data model

- **Inner bag** contains a bag inside a tuple. For Example, if we sort Band tuples based on Band's Origin, we will get:
  - (Los Angeles, {(Metallica, Los Angeles), (Mega Death, Los Angeles)})
  - (California,{(Linkin Park, California)})
- Here, first field type is a string while the second field type is a bag, which is an inner bag within a tuple.



# Pig tutorial

## Map

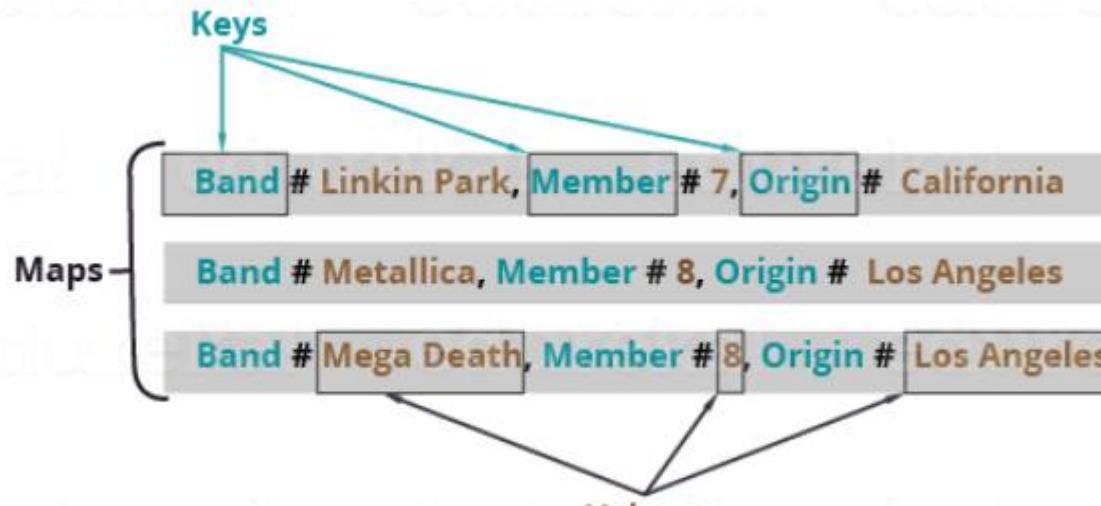


Figure: Map Example



# Pig tutorial

## Map

- A map is key-value pairs used to represent data elements. The key must be a chararray [] and should be unique like column name, so it can be indexed and value associated with it can be accessed on basis of the keys. The value can be of any data type.
- Maps are represented by '[]' symbol and key-value are separated by '#' symbol, as you can see in the above image.
- Example of maps– [band#Linkin Park, members#7 ], [band#Metallica, members#8 ]



# Pig tutorial

## Schema

- Schema assigns name to the field and declares data type of the field. Schema is optional in Pig Latin but Pig encourage you to use them whenever possible, as the error checking becomes efficient while parsing the script which results in efficient execution of program. Schema can be declared as both simple and complex data types. During LOAD function, if the schema is declared it is also attached with the data.



# Pig tutorial

## Schema

- Few Points on Schema in Pig:
- If the schema only includes the field name, the data type of field is considered as byte array.
- If you assign a name to the field you can access the field by both, the field name and the positional notation. Whereas if field name is missing we can only access it by the positional notation i.e. \$ followed by the index number.
- If you perform any operation which is a combination of relations (like JOIN, COGROUP, etc.) and if any of the relation is missing schema, the resulting relation will have null schema.
- If the schema is null, Pig will consider it as byte array and the real data type of field will be determined dynamically.



# Pig tutorial

## *Open Pig grunt shell*

- Command: pig

```
[cloudera@quickstart ~]$ pig
```



# Pig tutorial

## Open Pig grunt shell

- Invoke the ls command of Linux shell from the Grunt shell
- Command: sh ls

```
|grunt> sh ls
cloudera-manager
cm_api.py
Desktop
Documents
Downloads
eclipse
enterprise-deployment.json
express-deployment.json
kerberos
lib
Music
parcels
Pictures
Public
Templates
Videos
workspace
```



# Pig tutorial

## Load data into Apache Pig from the file system (HDFS/ Local)

### Syntax

The load statement consists of two parts divided by the “=” operator. On the left-hand side, we need to mention the name of the relation **where** we want to store the data, and on the right-hand side, we have to define **how** we store the data. Given below is the syntax of the **Load** operator.

```
Relation_name = LOAD 'Input file path' USING function as schema;
```

Where,

- **relation\_name** – We have to mention the relation in which we want to store the data.
- **Input file path** – We have to mention the HDFS directory where the file is stored. (In MapReduce mode)
- **function** – We have to choose a function from the set of load functions provided by Apache Pig (**BinStorage**, **JsonLoader**, **PigStorage**, **TextLoader**).
- **Schema** – We have to define the schema of the data. We can define the required schema as follows –

```
(column1 : data type, column2 : data type, column3 : data type);
```

**Note** – We load the data without specifying the schema. In that case, the columns will be addressed as \$01, \$02, etc... (check).



# Pig tutorial

## Load data from HDFS into Pig relation

- Open gedit and create a txt file and save it in home directory

101 Anto 20000 Architect

102 Bob 7000 SoftwareEngineer

103 Jack 4000 Programmer

104 Bil 3000 ITConsultant

105 Henry 5000 Manager

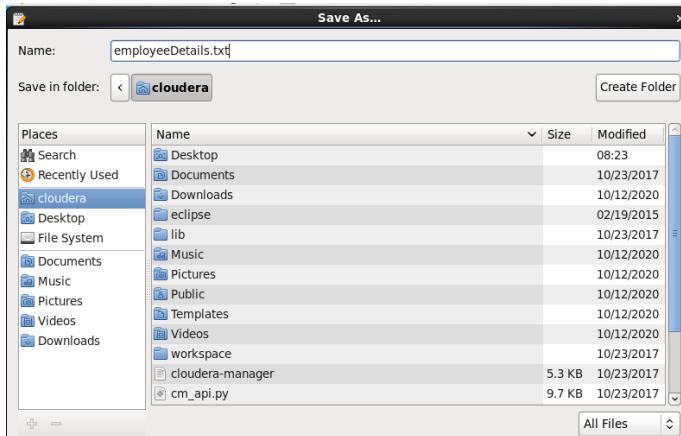
106 Isac 9000 Sr.Manager

107 David 7000 VP

108 Kingston 9000 Sr.VP

109 Balmer 19923 CEO

```
employeeDetails.txt
101 Anto 20000 Architect
102 Bob 7000 SoftwareEngineer
103 Jack 4000 Programmer
104 Bil 3000 ITConsultant
105 Henry 5000 Manager
106 Isac 9000 Sr.Manager
107 David 7000 VP
108 Kingston 9000 Sr.VP
109 Balmer 19923 CEO
```





# Pig tutorial

## *Load data from HDFS into Pig relation*

- Put the file to HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -put employeeDetails.txt
```



# Pig tutorial

## Load data from HDFS into Pig relation

- Let's load data into Pig Relation using Pig Data Types.
- Command: employee = load 'employeeDetails.txt' using PigStorage(' ') as (id:int, name:chararray,salary:float,task:chararray);

```
grunt> employee = load 'employeeDetails.txt' using PigStorage(' ') as (id:int, name:chararray,salary:float,task:  
chararray);  
2020-10-19 09:38:21,715 [main] INFO  hive.metastore - Trying to connect to metastore with URI thrift://127.0.0.1  
:9083
```



# Pig tutorial

## Load data from HDFS into Pig relation

- Let's DESCRIBE the relation to see the Data type names.

```
grunt> describe employee;
2020-10-19 09:48:30,077 [main] INFO  hive.metastore - Trying to connect to metastore with URI thrift://127.0.0.1
:9083
2020-10-19 09:48:30,078 [main] INFO  hive.metastore - Opened a connection to metastore, current connections: 1
2020-10-19 09:48:30,078 [main] INFO  hive.metastore - Connected to metastore.
2020-10-19 09:48:30,098 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is depre
cated. Instead, use fs.defaultFS
2020-10-19 09:48:30,098 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is de
precated. Instead, use mapreduce.jobtracker.address
employee: {id: int,name: chararray,salary: float,task: chararray}
```



# Pig tutorial

## Load data from HDFS into Pig relation

- Let's use dump operator to display the result

```
grunt> dump employee;
2020-10-19 09:38:31,176 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
```

```
2020-10-19 09:38:49,888 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(101,Anto,20000.0,Architect)
(102,Bob,7000.0,SoftwareEngineer)
(103,Jack,4000.0,Programmer)
(104,Bil,3000.0,ITConsultant)
(105,Henry,5000.0,Manager)
(106,Isac,9000.0,Sr.Manager)
(107,David,7000.0,VP)
(108,Kingston,9000.0,Sr.VP)
(109,Balmer,19923.0,CEO)
```



# Pig tutorial

## Load data from Local File System into Pig relation

- Open pig shell in local mode by pig -x local

```
grunt> quit;  
[cloudera@quickstart ~]$ pig -x local
```

- Load file

```
grunt> employee2 = load '/home/cloudera/employeeDetails.txt' using PigStorage(' ') as (id:int, name:chararray,salary:float,task:chararray);  
grunt>
```



# Pig tutorial

## Load data from Local File System into Pig relation

- Check the result

```
grunt> dump employee2;
2020-10-19 11:34:30,221 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script
: UNKNOWN
```

```
2020-10-19 11:34:42,816 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input
paths to process : 1
(101,Anto,20000.0,Architect)
(102,Bob,7000.0,SoftwareEngineer)
(103,Jack,4000.0,Programmer)
(104,Bil,3000.0,ITConsultant)
(105,Henry,5000.0,Manager)
(106,Isac,9000.0,Sr.Manager)
(107,David,7000.0,VP)
(108,Kingston,9000.0,Sr.VP)
(109,Balmer,19923.0,CEO)
```



# Pig tutorial

## LOAD Data from HIVE Table into PIG Relation.

- Let us consider that we have the Hive table called student with some data in it

```
hive> show tables;
OK
department
emptable
newstuden
student
Time taken: 0.707 seconds, Fetched: 4 row(s)
hive> select * from student;
OK
123451 Quynh    Hadoop   22
123452 Tai      Java     22
123453 Truong   Python   23
123454 Nghia   Hadoop   24
123455 Thuy     Java     23
123456 Hao      Python   24
123457 Hien    Hadoop   22
123458 Phuong   Java     23
123459 Hai      Python   23
123460 Phuong   Hadoop   24
Time taken: 0.706 seconds, Fetched: 10 row(s)
```



# Pig tutorial

## LOAD Data from HIVE Table into PIG Relation.

- The command below will load the data from HIVE Table into PIG Relation called pigdataStudent
- Command: **pigdataStudent = load 'student' using org.apache.hive.hcatalog.pig.HCatLoader();**

```
[grunt> pigdataStudent = load 'student' using org.apache.hive.hcatalog.pig.HCatLoader();
2020-10-19 09:51:23,159 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-10-19 09:51:23,159 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2020-10-19 09:51:23,194 [main] INFO  hive.metastore - Closed a connection to metastore, current connections: 0
2020-10-19 09:51:23,194 [main] INFO  hive.metastore - Trying to connect to metastore with URI thrift://127.0.0.1
:9083
```



# Pig tutorial

## LOAD Data from HIVE Table into PIG Relation.

- Check the content of the relation
- Command: **dump pigdataStudent;**

```
grunt> dump pigdataStudent;
2020-10-19 09:52:49,835 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
```

```
(123451,Quynh,Hadoop,22)
(123452,Tai,Java,22)
(123453,Truong,Python,23)
(123454,Nghia,Hadoop,24)
(123455,Thuy,Java,23)
(123456,Hao,Python,24)
(123457,Hien,Hadoop,22)
(123458,Phuong,Java,23)
(123459,Hai,Python,23)
(123460,Phuong,Hadoop,24)
(123466,Min,Hadoop,25)
(123464,Den,Hadoop,25)
(123461,Anh,Java,22)
(123462,An,Java,24)
```



# Pig tutorial

## *Filter operation*

- Now the a1 relation has all the data, Let us try to filter only the values where age > 23.
- Command: **plus23 = filter pigdataStudent by age > 23;**

```
grunt> plus23 = filter pigdataStudent by age > 23;
2020-10-19 09:57:58,418 [main] INFO  hive.metastore - Trying to connect to metastore with URI thrift://127.0.0.1
:9083
2020-10-19 09:57:58,419 [main] INFO  hive.metastore - Opened a connection to metastore, current connections: 1
2020-10-19 09:57:58,419 [main] INFO  hive.metastore - Connected to metastore.
2020-10-19 09:57:58,437 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is depre
cated. Instead, use fs.defaultFS
```



# Pig tutorial

## Filter operation

- Let's DESCRIBE the relation to see the Data type names

```
grunt> describe plus23;
2020-10-19 09:58:51,153 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-10-19 09:58:51,154 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2020-10-19 09:58:51,213 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-10-19 09:58:51,213 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2020-10-19 09:58:51,300 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-10-19 09:58:51,300 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
plus23: {id: int,name: chararray,course: chararray,age: int}
```



# Pig tutorial

## Filter operation

- Check the result

```
grunt> dump plus23;
2020-10-19 10:00:48,159 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is depre
cated. Instead, use fs.defaultFS
2020-10-19 10:00:48,159 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is de

2020-10-19 10:01:12,260 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input
paths to process : 1
(123454,Nghia,Hadoop,24)
(123456,Hao,Python,24)
(123460,Phuong,Hadoop,24)
(123466,Min,Hadoop,25)
(123464,Den,Hadoop,25)
(123462,An,Java,24)
grunt> ■
```



# Pig tutorial

## *Storing Data from PIG Relation*

### Syntax

Given below is the syntax of the Store statement.

```
STORE Relation_name INTO ' required_directory_path ' [USING function];
```



# Pig tutorial

## Store PIG Relation into HDFS

```
grunt> store plus23 into '/user/cloudera/plus23' using PigStorage(',');
2020-10-19 11:53:51,902 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
```

Counters:

```
Total records written : 6
Total bytes written : 128
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0
```

Job DAG:

```
job_1603115446431_0017
```

```
2020-10-19 11:54:12,429 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
```



# Pig tutorial

## Store PIG Relation into HDFS

- Check if the plus23 directory has been created in HDFS

```
grunt> ls
hdfs://quickstart.cloudera:8020/user/cloudera/emp      <dir>
hdfs://quickstart.cloudera:8020/user/cloudera/employeeDetails.txt<r 1> 217
hdfs://quickstart.cloudera:8020/user/cloudera/plus23    <dir>
grunt> cd plus23
grunt> ls
hdfs://quickstart.cloudera:8020/user/cloudera/plus23/_SUCCESS<r 1>      0
hdfs://quickstart.cloudera:8020/user/cloudera/plus23/part-m-00000<r 1> 128
```



# Pig tutorial

## Store PIG Relation into HDFS

- Check the content of the file

```
grunt> cat part-m-00000
123454,Nghia,Hadoop,24
123456,Hao,Python,24
123460,Phuong,Hadoop,24
123466,Min,Hadoop,25
123464,Den,Hadoop,25
123462,An,Java,24
```



# Pig tutorial

## STORE Data from PIG Relation Into HIVE Table

- Create a new Hive table

```
hive> create table plus23student (id int, name string, course string, age int);
OK
Time taken: 1.023 seconds
hive> describe plus23student;
OK
id          int
name        string
course      string
age         int
Time taken: 0.104 seconds, Fetched: 4 row(s)
```



# Pig tutorial

## STORE Data from PIG Relation Into HIVE Table

- Store data from Pig relation into the newly created Hive table

```
grunt> store plus23 into 'plus23student' using org.apache.hive.hcatalog.pig.HCatStorer();
2020-10-19 10:04:59,906 [main] INFO  hive.metastore - Trying to connect to metastore with URI thrift://127.0.0.1
:9083
```

```
Successfully stored 6 records (128 bytes) in: "plus23student"
```

```
Counters:
```

```
Total records written : 6
Total bytes written : 128
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0
```

```
Job DAG:
```

```
job_1603115446431_0012
```

```
2020-10-19 10:13:46,233 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLaunc
her - Success!
```



# Pig tutorial

## STORE Data from PIG Relation Into HIVE Table

- Check the Hive table

```
hive> select * from plus23student;
OK
123454 Nghia    Hadoop   24
123456 Hao       Python   24
123460 Phuong   Hadoop   24
123466 Min      Hadoop   25
123464 Den      Hadoop   25
123462 An       Java     24
Time taken: 0.325 seconds, Fetched: 6 row(s)
```



# Pig tutorial

## Create Your First Apache Pig Script

- Create and open an Apache Pig script file in an editor (e.g. gedit)

```
[cloudera@quickstart ~]$ sudo gedit /home/cloudera/test.pig
```

The screenshot shows a window of the gedit text editor. The title bar says '\*test.pig X'. The main area contains the following Pig Latin script:

```
employ = load 'employeeDetails.txt' using PigStorage(' ') as (id:int,
name:chararray,salary:float,task:chararray);

employSalary = FOREACH employ generate id, salary;

dump employSalary;
```



# Pig tutorial

## Create Your First Apache Pig Script

- Run the script in linux terminal

```
[cloudera@quickstart ~]$ pig test.pig
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell)
.
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more in
fo

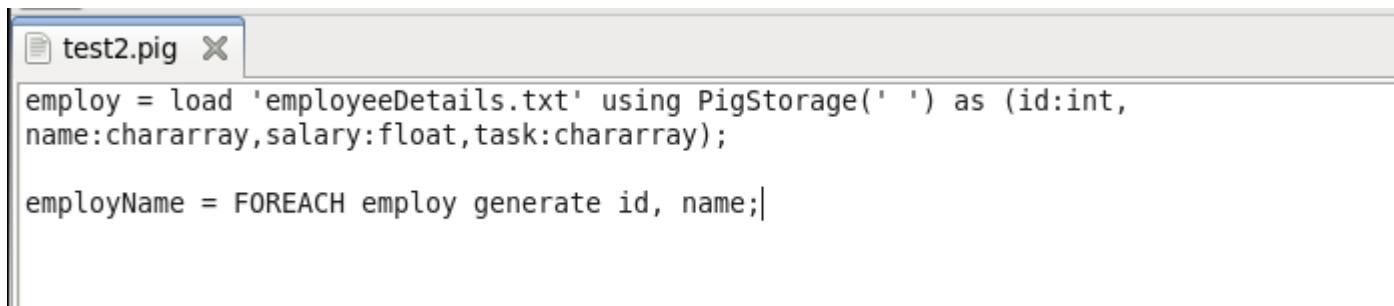
2020-10-19 10:53:21,121 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.util.MapRedUtil - Total input paths to process : 1
(101,20000.0)
(102,7000.0)
(103,4000.0)
(104,3000.0)
(105,5000.0)
(106,9000.0)
(107,7000.0)
(108,9000.0)
(109,19923.0)
2020-10-19 10:53:21,248 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
```



# Pig tutorial

## Create Your First Apache Pig Script

- Create file **test2.pig**



```
test2.pig X
employ = load 'employeeDetails.txt' using PigStorage(' ') as (id:int,
name:chararray,salary:float,task:chararray);

employName = FOREACH employ generate id, name;
```



# Pig tutorial

## Create Your First Apache Pig Script

- Run the script in grunt shell

```
grunt> run test2.pig
2020-10-19 11:05:32,010 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
```



# Pig tutorial

## Create Your First Apache Pig Script

- Check the result

```
grunt> dump employName;
2020-10-19 11:06:32,619 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2020-10-19 11:06:32,619 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
```

```
2020-10-19 11:06:51,266 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(101,Anto)
(102,Bob)
(103,Jack)
(104,Bil)
(105,Henry)
(106,Isac)
(107,David)
(108,Kingston)
(109,Balmer)
```



# Pig tutorial

## Positional notation reference

- So far, we fields in Pig Relation are referred by name (e.g. id, name, salary, task, etc.,)
- Names are assigned by you using schemas
- Positional notation is generated by the system. Positional notation is indicated with the dollar sign (\$) and begins with zero (0); for example, \$0, \$1, \$2.

	First Field	Second Field	Third Field
Data type	chararray	int	float
Positional notation (generated by system)	\$0	\$1	\$2
Possible name (assigned by you using a schema)	name	age	gpa
Field value (for the first tuple)	John	18	4.0



# Pig tutorial

## Positional notation reference

- In this example, the field **task** is referenced by position notation **\$3**

```
grunt> employee = load 'employeeDetails.txt' using PigStorage(' ') as (id:int, name:chararray, salary:float, task:chararray);
```

```
grunt> describe employee;
employee: {id: int, name: chararray, salary: float, task: chararray}
```

```
grunt> empTask = foreach employee generate id,$3;
grunt> describe empTask;
empTask: {id: int, task: chararray}
```

```
grunt> dump empTask;
2020-10-19 23:20:36,052 [main] INFO org.apache.pig.tools.pigstats.ScriptState -
```



# Pig tutorial

## Positional notation reference

- Check the result

```
grunt> dump empTask;
2020-10-19 23:20:36,052 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
```

```
2020-10-19 23:21:02,992 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.util.MapRedUtil - Total input paths to process : 1
(101,Architect)
(102,SoftwareEngineer)
(103,Programmer)
(104,ITConsultant)
(105,Manager)
(106,Sr.Manager)
(107,VP)
(108,Sr.VP)
(109,CEO)
grunt>
```



# Pig tutorial

## Schema Handling

- You can define a schema that includes both the field name and field type.
- You can define a schema that includes the field name only; in this case, the field type defaults to bytearray.
- You can choose not to define a schema; in this case, the field is un-named and the field type defaults to bytearray.



# Pig tutorial

## Schema Handling

- The field data types are not specified (defaults type is bytearray)

```
grunt> employee = load 'employeeDetails.txt' using PigStorage(' ') as (id, name, salary, task);
grunt> describe employee;
employee: {id: bytearray, name: bytearray, salary: bytearray, task: bytearray}
```



# Pig tutorial

## Schema Handling

- Unknow schema

```
grunt> employee = load 'employeeDetails.txt' using PigStorage(' ');
grunt> describe employee;
Schema for employee unknown.
```

```
grunt> bonusSalary = foreach employee generate $0,$2+500;
2020-10-20 00:04:31,169 [main] WARN  org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 3 time(s).
```

```
grunt> describe bonusSalary;
2020-10-20 00:04:47,411 [main] WARN  org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 1 time(s).
bonusSalary: {bytarray,int}
```



# Pig tutorial

## Schema Handling

- Check the result

```
grunt> dump bonusSalary;
2020-10-20 00:03:47,623 [main] WARN  org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 1 time(s).
2020-10-20 00:03:47,624 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: UNKNOWN
```

```
2020-10-20 00:04:06,072 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process
: 1
(101,20500)
(102,7500)
(103,4500)
(104,3500)
(105,5500)
(106,9500)
(107,7500)
(108,9500)
(109,20423)
```



# Pig tutorial

## Schema Handling

- Declare the schema of the result

```
grunt> bonusSalary = foreach employee generate $0,$2+500 as salary:float;
2020-10-20 00:03:30,144 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 2 time(s).
grunt> describe bonusSalary;
2020-10-20 00:04:17,802 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 1 time(s).
bonusSalary: {bytearray,salary: float}
```

```
grunt> bonusSalary = foreach employee generate $0 as id:int, $2+500 as salary:float;
2020-10-20 00:12:55,336 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 4 time(s).
grunt> describe bonusSalary;
2020-10-20 00:13:07,130 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 1 time(s).
bonusSalary: {id: int,salary: float}
```



# High-Level Data Process Components

## Hive & Pig

- Both require compiler to generate Mapreduce jobs
- Hence high latency queries when used for real time responses to ad-hoc queries
- Both are good for **batch processing** and **ETL** jobs
- Fault tolerant



# High-Level Data Process Components

## Impala

- Cloudera Impala is a query engine that runs on Apache Hadoop.
- Similar to HiveQL.
- Does not use Map-reduce
- Optimized for low latency queries
- Open source apache project
- Developed by Cloudera
- Much faster than Hive or pig

# Hive

1. Create an internal table name **InternalEmployee** with the schema

ID	int
Name	string
Salary	float
Department	string
Age	int

- a. Put file **EmployeeInfo.csv** to **HDFS**, then load data from this file to table InternalEmp
  - b. Login to HUE and query EmpName and EmpAge of all employees with salary- $\geq$  2000
  - c. In HUE, insert a row with the content [20029,Quang,1000,Support,20] to the table and then show the content of the newly created file which stores the data of the new row
2. Create a dynamic partition table named **EmployeePartition** to store the data from **EmployeeInfo.csv** file. This table is partitioned by **Department**.

- a. Load data to EmployeePartition table
- b. Open HUE and query to show all the rows of the table |

## Sqoop and Pig

1. Create table **Employee** inside database **EmployeeInfo** and load the content from **EmployeeNoheader.csv** to this table. Then query the created database to show the info of employees with **salary >1500**

id	name	salary	department	age
20023	Tran	2000	Product	24
20024	Vu	3000	Product	25
20025	Dao	2500	Testing	21
20026	Nam	2500	Support	22
20027	Viet	3000	Product	24

2. Use Sqoop import to get id, name, and salary of employees in Employee table **with salary > 2000** and save results in **Employee2000plus** directory. Then show the imported results.

3. Load data from file **EmployeeNoheader.csv** to a relation named **PigEmployee** and show the result

4. Create a relation named **PigEmployee1500plus** to store employees with **salary > 1500**

5. Store data in **PigEmployee1500plus** to a Hive table name **HiveEmployee1500plus** and show the result

6. Load data in **EmployeeNoheader.csv** to a relation named **PigEmployeeNoschema** without declaring datatype. Then from this relation, create another relation named **BonusSalary** with all data is same as that in **PigEmployeeNoschema** but **salary = salary + 500**

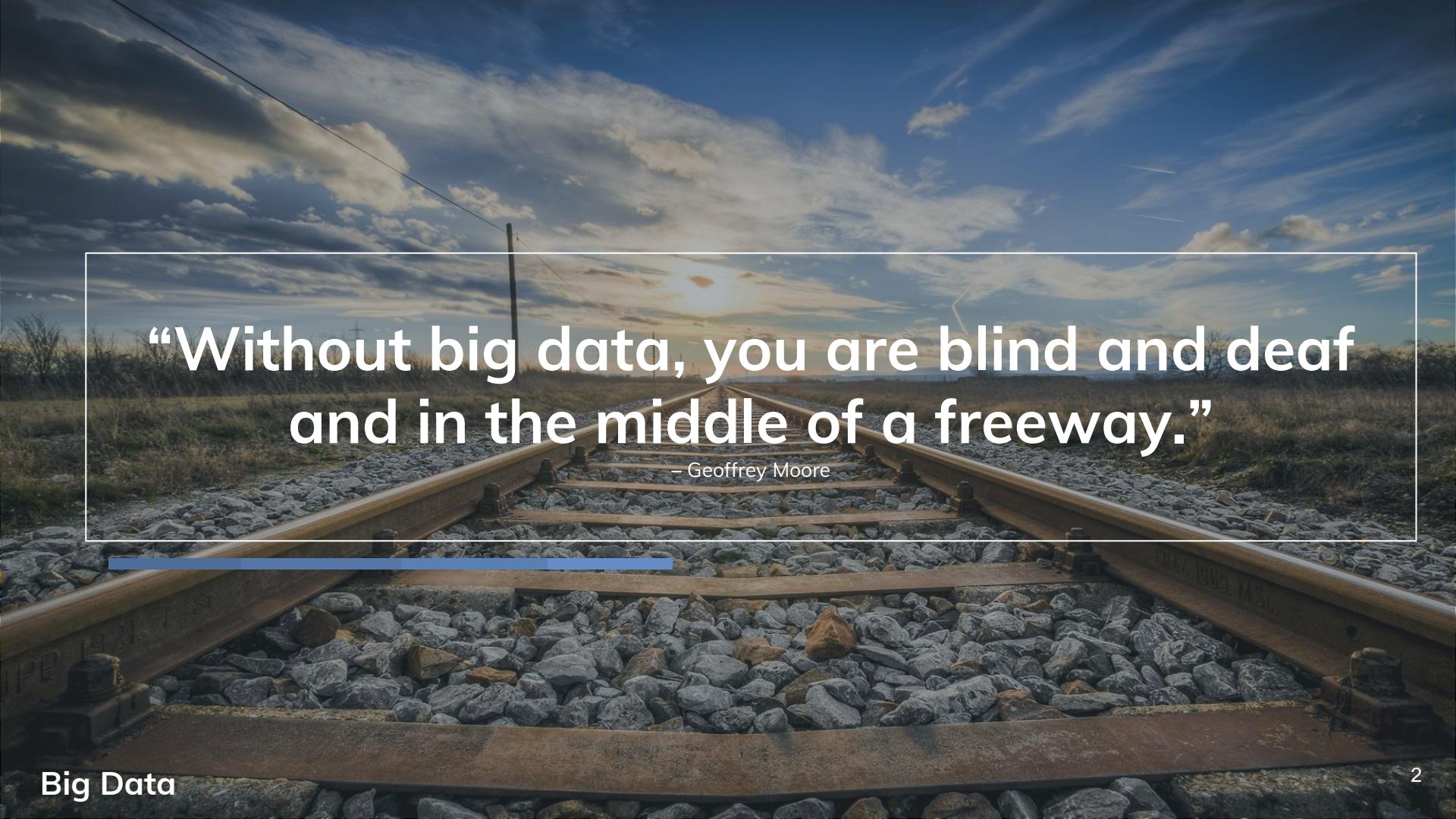
# Big Data

NoSQL database and Hbase tutorial

Trong-Hop Do

**S<sup>3</sup>Lab**

*Smart Software System Laboratory*

A photograph of a railway track receding into a cloudy sky. The track is made of steel rails and wooden sleepers, with gravel ballast. The sky is filled with scattered clouds, suggesting a sunset or sunrise. A white rectangular box contains the quote.

“Without big data, you are blind and deaf  
and in the middle of a freeway.”

– Geoffrey Moore

# NoSQL

---



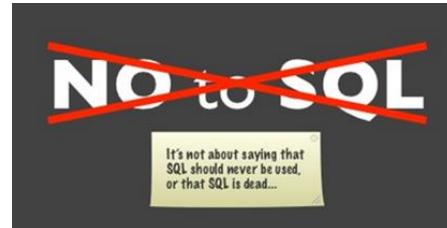
# Background

- Relational databases mainstay of business
- Web-based applications caused spikes
- Explosion of social media sites (Facebook, Twitter) with large data needs
- rise of cloud-based solutions such as Amazon S3 (simple storage solution)
- Hooking RDBMS to web-based application becomes trouble



# What is NoSQL?

- This name stands for **Not Only SQL**
- The term NOSQL was introduced by Carl Strozzi in 1998 to name his file-based database
- It was again re-introduced by Eric Evans when an event was organized to discuss open source distributed databases
  - Eric states that "... but the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for. ..."





# What is NoSQL?

## Key features (Advantages)

- non-relational
- don't require schema
- data are replicated to multiple nodes (so, identical & fault-tolerant)  
and can be partitioned:
  - down nodes easily replaced
  - no single point of failure
- horizontal scalable





# What is NoSQL?

## Key features (Advantages)

- cheap, easy to implement (open-source)
- massive write performance
- fast key-value access





# What is NoSQL?

## Disadvantages

- Don't fully support relational features
  - no join, group by, order by operations (except within partitions)
  - no referential integrity constraints across partitions
- No declarative query language (e.g., SQL) more programming
- Relaxed **ACID** (see **CAP** theorem) fewer guarantees
- No easy integration with other applications that support SQL



# Who is using them?





# 3 major papers for NoSQL

- Three major papers were the “seeds” of the NOSQL movement:
  - BigTable (Google)
  - DynamoDB (Amazon)
    - Ring partition and replication
    - Gossip protocol (discovery and error detection)
    - Distributed key-value data stores
    - Eventual consistency
  - CAP Theorem



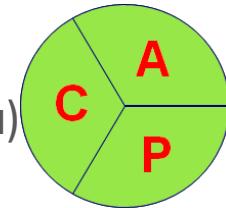
# The perfect storm

- Large datasets, acceptance of alternatives, and dynamically-typed data has come together in a “perfect storm”
- Not a backlash against RDBMS
- SQL is a rich query language that cannot be rivaled by the current list of NOSQL offerings



# CAP Theorem

- Suppose three properties of a **distributed system** (sharing data)
  - Consistency:**
    - Reads and writes are always executed atomically and are strictly consistent ([linearizable](#)). Put differently, all clients have the same view on the data at all times.
  - Availability:**
    - Every non-failing node in the system can always accept read and write requests by clients and will eventually return with a meaningful response, i.e. not with an error message.
  - Partition-tolerance:**
    - system properties (consistency and/or availability) hold even when network failures prevent some machines from communicating with others. A system can continue to operate in the presence of a network partitions





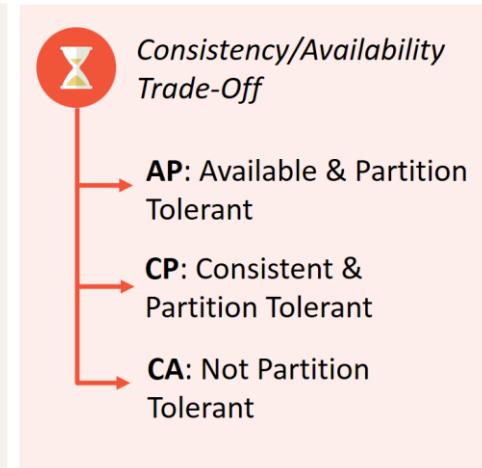
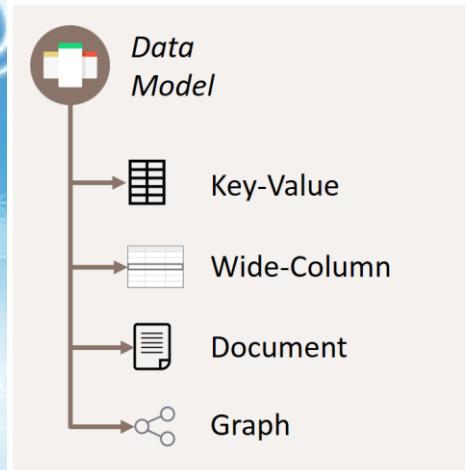
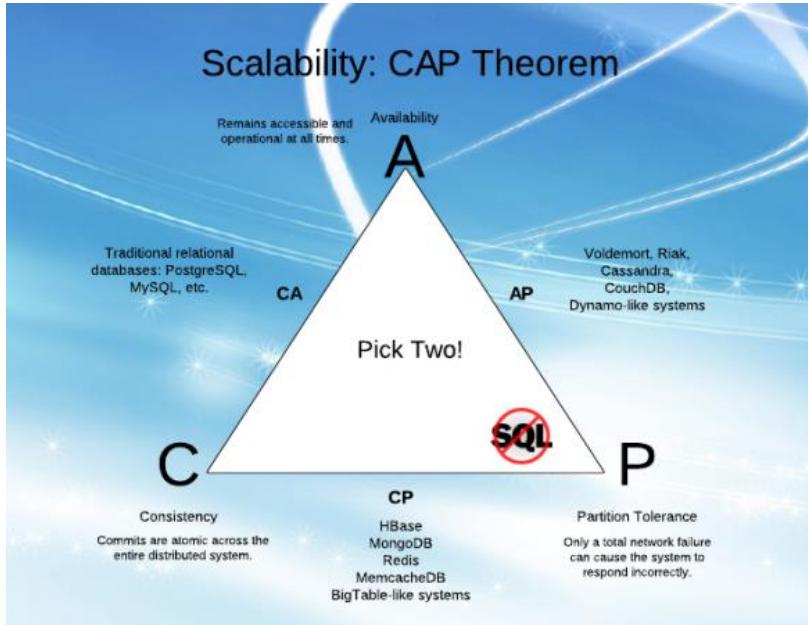
# CAP Theorem

- Brewer's CAP Theorem:
  - For any system sharing data, it is “impossible” to guarantee simultaneously all of these three properties
  - You can have at most two of these three properties for any shared-data system
- Very large systems will “partition” at some point:
  - That leaves either **C** or **A** to choose from (traditional DBMS prefers **C** over **A** and **P**)
  - In almost all cases, you would choose **A** over **C** (except in specific applications such as order processing)



# CAP Theorem

## Consistency



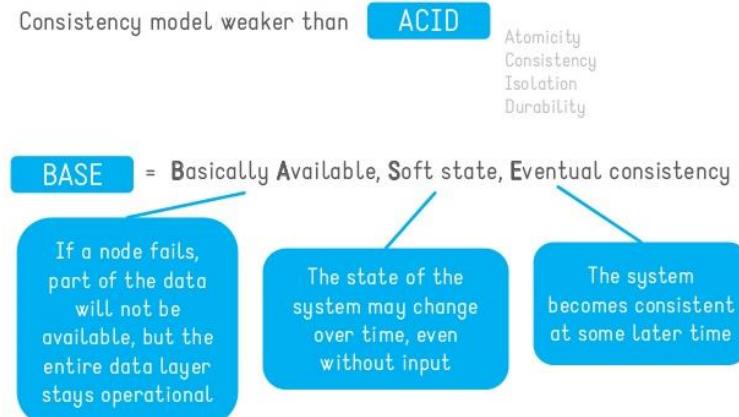
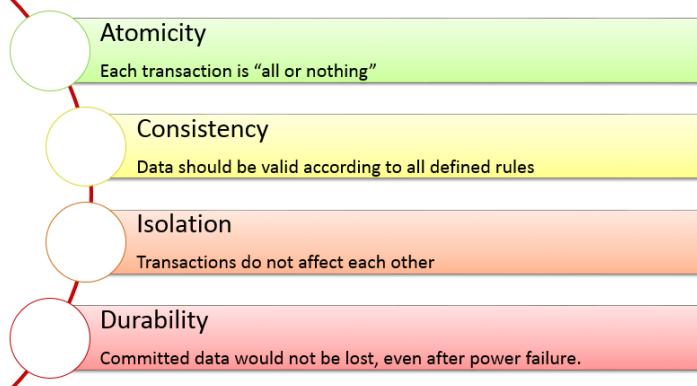


# CAP Theorem

## Consistency

- Have 2 types of consistency:
  - Strong consistency – ACID (Atomicity, Consistency, Isolation, Durability)
  - Weak consistency – BASE (Basically Available Soft-state Eventual consistency)

### ACID Properties





# CAP Theorem

## Consistency

- A consistency model determines rules for visibility and apparent order of updates
- Example:
  - Row X is replicated on nodes M and N
  - Client A writes row X to node N
  - Some period of time t elapses
  - Client B reads row X from node M
  - **Does client B see the write from client A?**
  - Consistency is a continuum with tradeoffs
  - **For NOSQL, the answer would be: “maybe”**
  - CAP theorem states: “strong consistency can't be achieved at the same time as availability and partition-tolerance”



# NoSQL

- “No-schema” is a common characteristics of most NOSQL storage systems
- Provide “flexible” data types
- Other or additional query languages than SQL
- Distributed – horizontal scaling
- Less structured data
- Supports big data



# NoSQL Categories

## Key Value



Example:  
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

## Document-Based



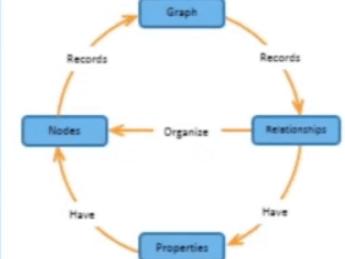
Example:  
MongoDB, CouchDB, OrientDB, RavenDB

## Column-Based



Example:  
BigTable, Cassandra, Hbase, Hypertable

## Graph-Based



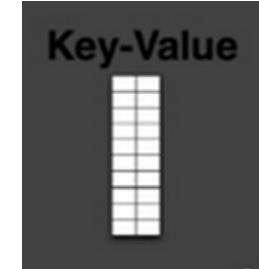
Example:  
Neo4J, InfoGrid, Infinite Graph, Flock DB



# NoSQL Categories

## Key-value

- Focus on scaling to huge amounts of data
- Designed to handle massive load
- Based on Amazon's dynamo paper
- Data model: (global) collection of Key-value pairs
- Dynamo ring partitioning and replication
- Example: (DynamoDB)
  - items having one or more attributes (name, value)
  - An attribute can be single-valued or multivalued like set.
  - items are combined into a table





# NoSQL Categories

## Key-value

- Basic API access:
  - get(key): extract the value given a key
  - put(key, value): create or update the value given its key
  - delete(key): remove the key and its associated value
  - execute(key, operation, parameters): invoke an operation to the value (given its key) which is a special data structure (e.g. List, Set, Map .... etc)



# NoSQL Categories

## Key-value

- Pros:
  - very fast
  - very scalable (horizontally distributed to nodes based on key)
  - simple data model
  - eventual consistency
  - fault-tolerance
- Cons:
  - Can't model more complex data structure such as objects



# NoSQL Categories

## Key-value

Name	Producer	Data model	Querying
SimpleDB	Amazon	set of couples (key, {attribute}), where attribute is a couple (name, value)	restricted SQL; select, delete, GetAttributes, and PutAttributes operations
Redis	Salvatore Sanfilippo	set of couples (key, value), where value is simple typed value, list, ordered (according to ranking) or unordered set, hash value	primitive operations for each value type
Dynamo	Amazon	like SimpleDB	simple get operation and put in a context
Voldemort	LinkedIn	like SimpleDB	similar to Dynamo



# NoSQL Categories

## Key-value

employee_id	first_name	last_name	address
1	John	Doe	New York
2	Benjamin	Button	Chicago
3	Mycroft	Holmes	London

```
employee:$employee_id:$attribute_name = $value
```

```
employee:1:first_name = "John"  
employee:1:last_name = "Doe"  
employee:1:address = "New York"
```

```
employee:2:first_name = "Benjamin"  
employee:2:last_name = "Button"  
employee:2:address = "Chicago"
```

```
employee:3:first_name = "Mycroft"  
employee:3:last_name = "Holmes"  
employee:3:address = "London"
```



# NoSQL Categories

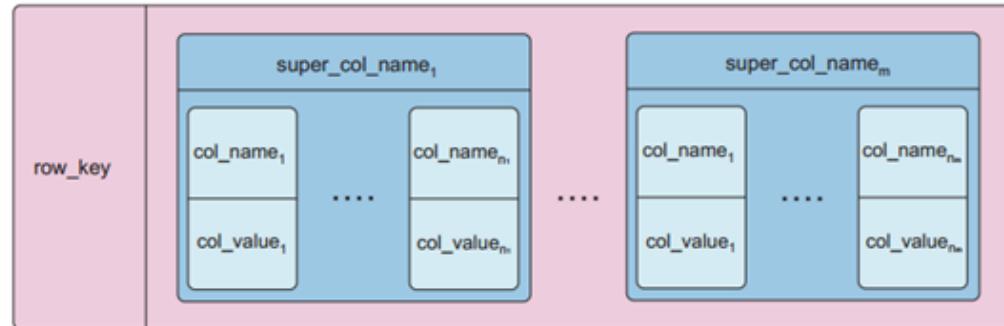
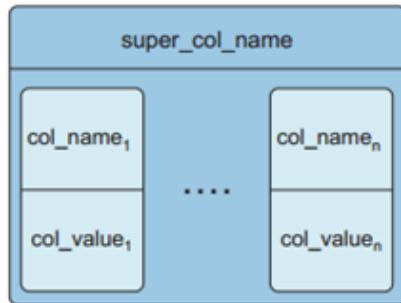
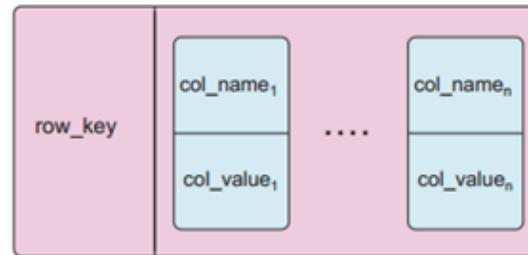
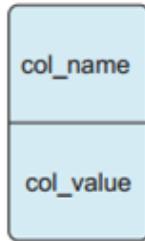
## Column-based

- Based on Google's BigTable paper
- Like column oriented relational databases (store data in column order) but with a twist
- Tables similarly to RDBMS, but handle semi-structured
- Data model:
  - Collection of Column Families
  - Column family = (key, value) where value = set of related columns (standard, super)
  - indexed by row key, column key and timestamp



# NoSQL Categories

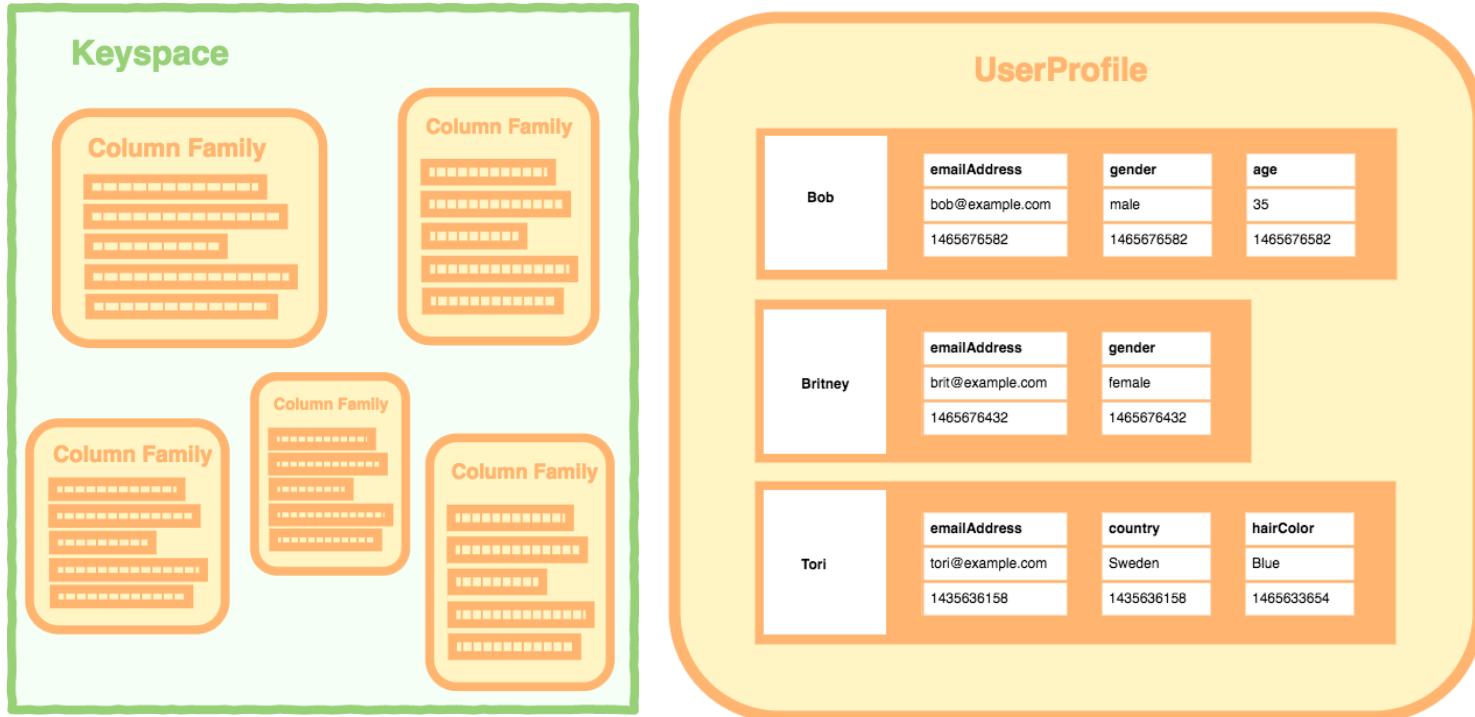
## Column-based





# NoSQL Categories

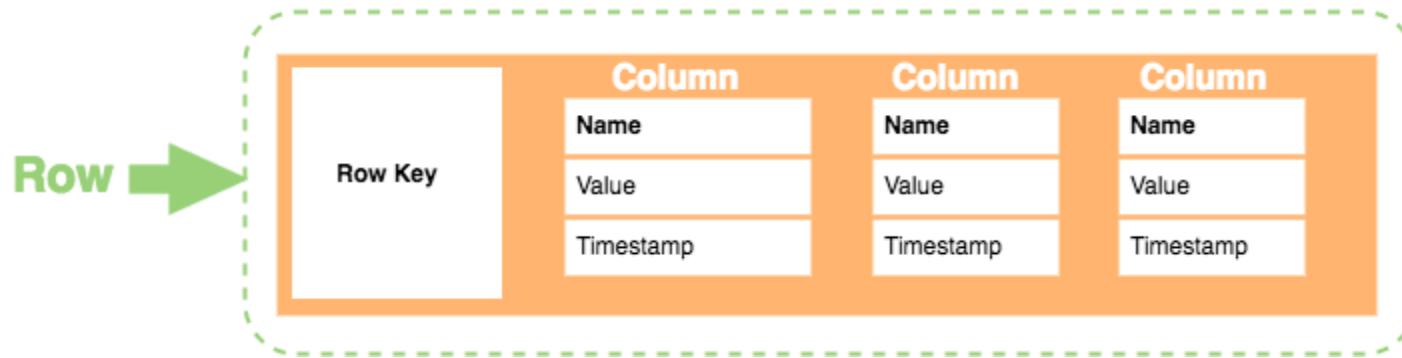
Column-based: Keyspace ~ Schema, Column Family ~ Table





# NoSQL Categories

Column-based: Row structure

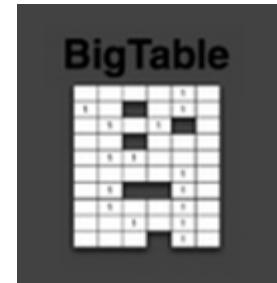




# NoSQL Categories

## Column-based

- One column family can have variable numbers of columns
- Cells within a column family are sorted “physically”
- Very sparse, most cells have null values
- Comparison: RDBMS vs column-based NOSQL
  - Query on multiple tables
    - RDBMS: must fetch data from several places on disk and glue together
    - Column-based NOSQL: only fetch column families of those columns that are required by a query (all columns in a column family are stored together on the disk, so multiple rows can be retrieved in one read operation data locality)





# NoSQL Categories

## Column-based

Operation	Column-Oriented Database	Row-Oriented Database
Aggregate Calculation of Single Column e.g. sum(price)	✓ fast	slow
Compression	✓ Higher. As stores similar data together	-
Retrieval of a few columns from a table with many columns	✓ Faster	has to skip over unnecessary data
Insertion/Updating of single new record	Slow	✓ Fast
Retrieval of a single record	Slow	✓ Fast



# NoSQL Categories

## Column-based

- Example: (Cassandra column family--timestamps removed for simplicity)

```
UserProfile = {  
    Cassandra = {  
        age:"20"  
    }  
    TerryCho = {  
        gender:"male"  
    }  
    Cath = {  
        emailAddress:"cassandra@apache.org",  
        emailAddress:"terry.cho@apache.org",  
        emailAddress:"cath@apache.org",  
        age:"20",gender:"female",address:"Seoul"  
    }  
}
```



# NoSQL Categories

## Column-based

Name	Producer	Data model	Querying
BigTable	Google	set of couples (key, {value})	selection (by combination of row, column, and time stamp ranges)
HBase	Apache	groups of columns (a BigTable clone)	JRUBY IRB-based shell (similar to SQL)
Hypertable	Hypertable	like BigTable	HQL (Hypertext Query Language)
CASSANDRA	Apache (originally Facebook)	columns, groups of columns corresponding to a key (supercolumns)	simple selections on key, range queries, column or columns ranges
PNUTS	Yahoo	(hashed or ordered) tables, typed arrays, flexible schema	selection and projection from a single table (retrieve an arbitrary single record by primary key, range queries, complex predicates, ordering, top-k)



# NoSQL Categories

## Document-based

- Can model more complex objects
- Inspired by Lotus Notes
- Data model: collection of documents
- Document: JSON (JavaScript Object Notation is a data model, key-value pairs, which supports objects, records, structs, lists, array, maps, dates, Boolean with nesting), XML, other semi-structured formats.



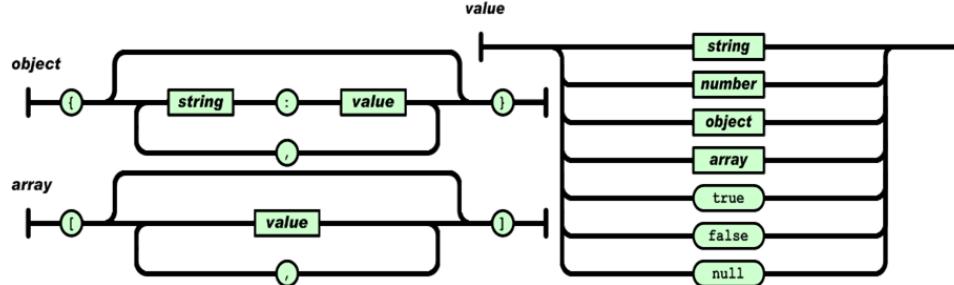


# NoSQL Categories

## Document-based

- Example: (MongoDB) document

```
{  
    Name:"Jaroslav",  
    Address:"Malostranske nám. 25, 118 00 Praha 1",  
    Grandchildren: {Claire: "7", Barbara: "6", "Magda: "3", "Kirsten: "1", "Otis: "3", Richard: "1"}  
    Phones: [ "123-456-7890", "234-567-8963" ]  
}
```



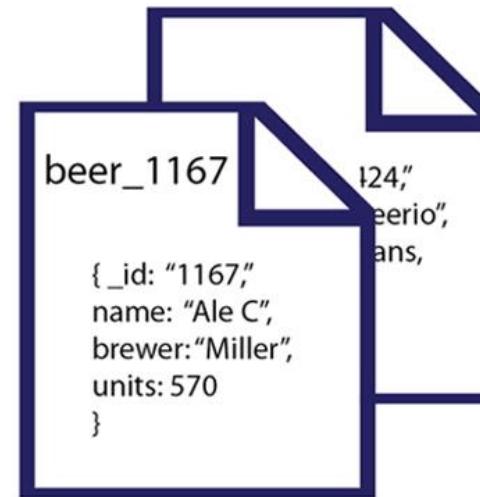


# NoSQL Categories

## Document-based

Beers Table			
1167	Ale C	Miller	570
3424	Beerio	Ians	340
5612	Amstel	Amtel	121
2409	Colt's	BeerCo	98

Beer Documents

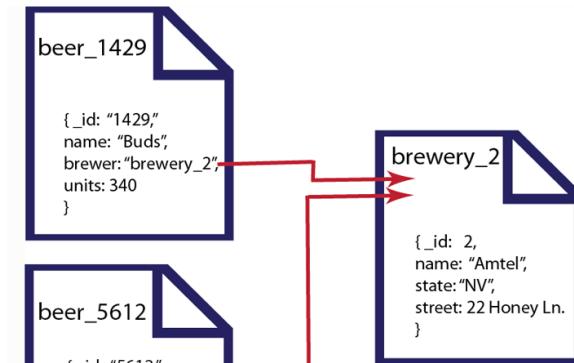




# NoSQL Categories

## Document-based

Beers Table				Brewery Table			
1167	Ale C	Miller	570				
1429	Buds	Amtel	340	1	Abe's	MA	
5612	Amstel	Amtel	121	2	Amtel	NV	
2409	Colt's	BeerCo	98	3	Bubba	TX	





# NoSQL Categories

## Document-based

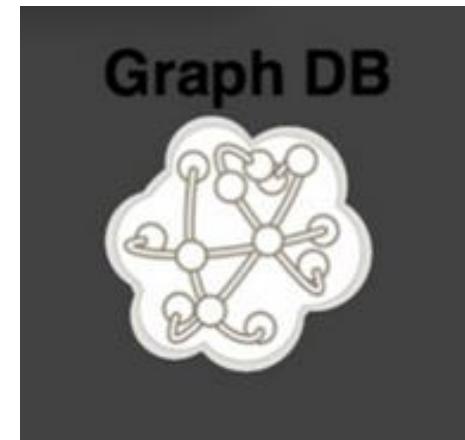
Name	Producer	Data model	Querying
MongoDB	10gen	object-structured documents stored in collections; each object has a primary key called ObjectId	manipulations with objects in collections (find object or objects via simple selections and logical expressions, delete, update,)
Couchbase	Couchbase <sup>1</sup>	document as a list of named (structured) items (JSON document)	by key and key range, views via Javascript and MapReduce



# NoSQL Categories

## Graph-based

- Focus on modeling the structure of data (interconnectivity)
- A **graph** is composed of two elements: a node and a relationship.
- Scales to the complexity of data
- Inspired by mathematical Graph Theory ( $G=(E,V)$ )
- Data model:
  - (Property Graph) nodes and edges
    - Nodes may have properties (including ID)
    - Edges may have labels or roles
  - Key-value pairs on both

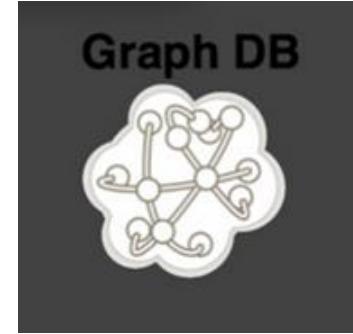




# NoSQL Categories

## Graph-based

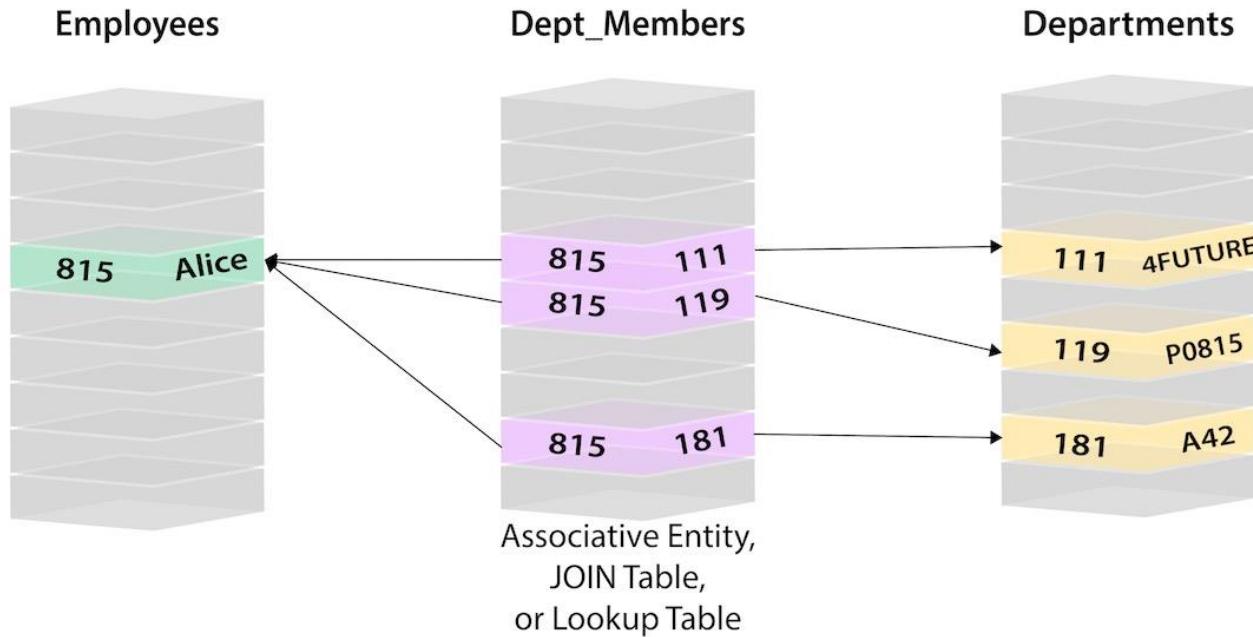
- Interfaces and query languages vary
- Single-step vs path expressions vs full recursion
- Example:
  - Neo4j, FlockDB, Pregel, InfoGrid ...





# NoSQL Categories

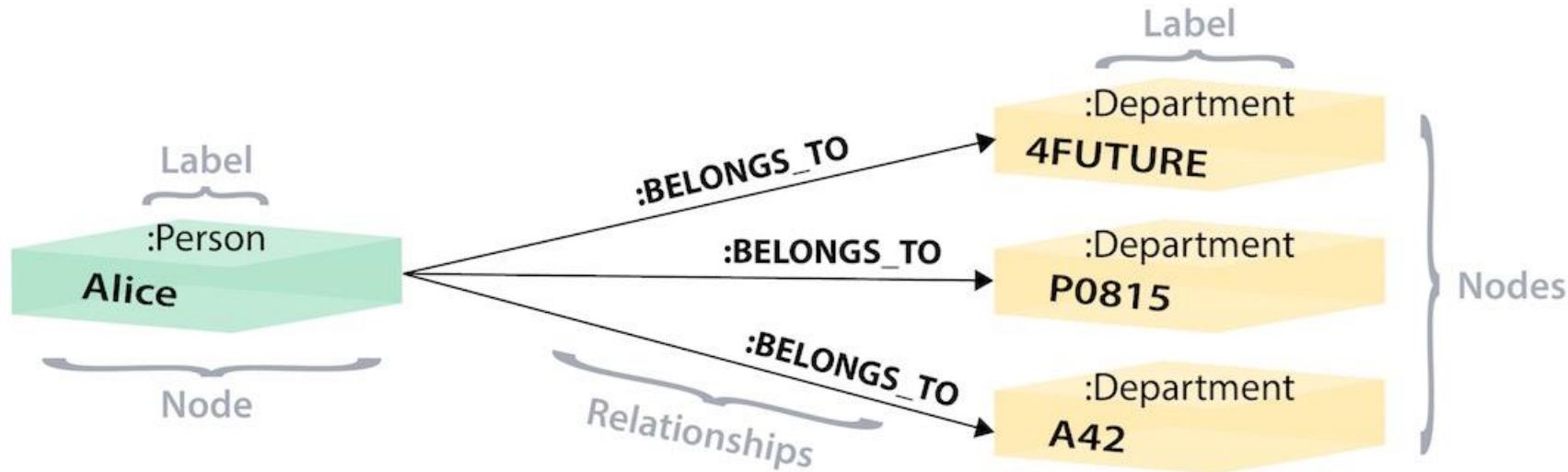
Graph-based





# NoSQL Categories

Graph-based





# NoSQL Categories

## Comparison

Attributes		NoSQL Databases								
Database model		Document-Stored		Wide-Column Stored			Key-Value Stored		Graph-oriented	
Design & Features	Features	MongoDB	CouchDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j
	Data storage	Volatile memory File System	Volatile memory File System	SSD	HDFS		Hadoop	Volatile memory File System	Bitcask LevelDB Volatile memory	File System
	Query language	Volatile memory File System	JavaScript Memcached protocol	API calls	API calls REST XML Thrift	API calls COL Thrift		API calls	HTTP REST Erlang	API calls REST SparQL Cypher Tinkerpop Gremlin
	Protocol	Custom, binary (BSON)	HTTP, REST	-	HTTP/REST Thrift	Thrift & custom binary CQL3	Thrift	Telnet-like	HTTP, REST	HTTP/REST Embedded in Java
	Conditional entry updates	Yes	Yes	Yes	Yes	No	Yes	No	No	
	MapReduce	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No
	Unicode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	TTL for Entries	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	
	Compression	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	Yes
	Integrity model	BASE	MVCC	ASID	Log Replication	BASE	MVCC	-	BASE	ASID
Integrity	Atomicity	Conditional	Yes	Yes	Yes	Yes	Condition al	Yes	No	Yes
	Consistency	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
	Isolation	No	Yes	Yes	No	No		Yes	Yes	Yes
	Durability (data storage)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes
	Transactions	No	No	No	Yes	No	Yes	Yes	No	Yes
	Referential integrity	No	No	No	No	No	No	Yes	No	Yes
	Revision control	No	Yes	Yes	Yes	No	Yes	No	Yes	No
	Secondary Indexes	Yes	Yes	No	Yes	Yes	Yes	-	Yes	-
	Composite keys	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	-
	Full text search	No	No	No	No	No	Yes	No	Yes	Yes
Indexing	Geospatial Indexes	Yes	No	No	No	No	Yes	-	-	Yes
	Graph support	No	No	No	No	No	Yes	No	Yes	Yes
	Horizontal scalable	Yes	Yes	Yes	Yes	Yes	Yes		Yes	No
	Replication	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes
	Replication mode	Master-Slave Replica tion	Master-Slave Replica tion	-	Master-Slave Replica tion	Master-Slave Replica tion	-	Master-Slave Replica tion	Multi-master replicati on	-
	Sharding	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
	Shared nothing architecture	Yes	Yes	Yes	Yes	Yes	-	-	Yes	-
	Value size max.	16MB	20MB	64KB	2TB	2GB	1EB	-	64MB	
	Operating system	Cross-platform	Ubuntu Red Hat Windows Mac OS X	Cross-platform	Cross-platform	Cross-platform	NIX 32 entries Operating system	Linux *NIX Mac OS X Window s	Cross-platform	Cross-platform
	Programming language	C++	Erlang C++ C Python	Java	Java	Java	S C C++	Erlang	Java	



# Conclusion

- NOSQL database cover only a part of data-intensive cloud applications (mainly Web applications)
  - Problems with cloud computing:
    - SaaS (Software as a Service or on-demand software) applications require enterprise-level functionality, including ACID transactions, security, and other features associated with commercial RDBMS technology, i.e. NOSQL should not be the only option in the cloud
    - Hybrid solutions:
      - Voldemort with MySQL as one of storage backend
      - deal with NOSQL data as semi-structured data
- Big Data -> integrating RDBMS and NOSQL via SQL/XML

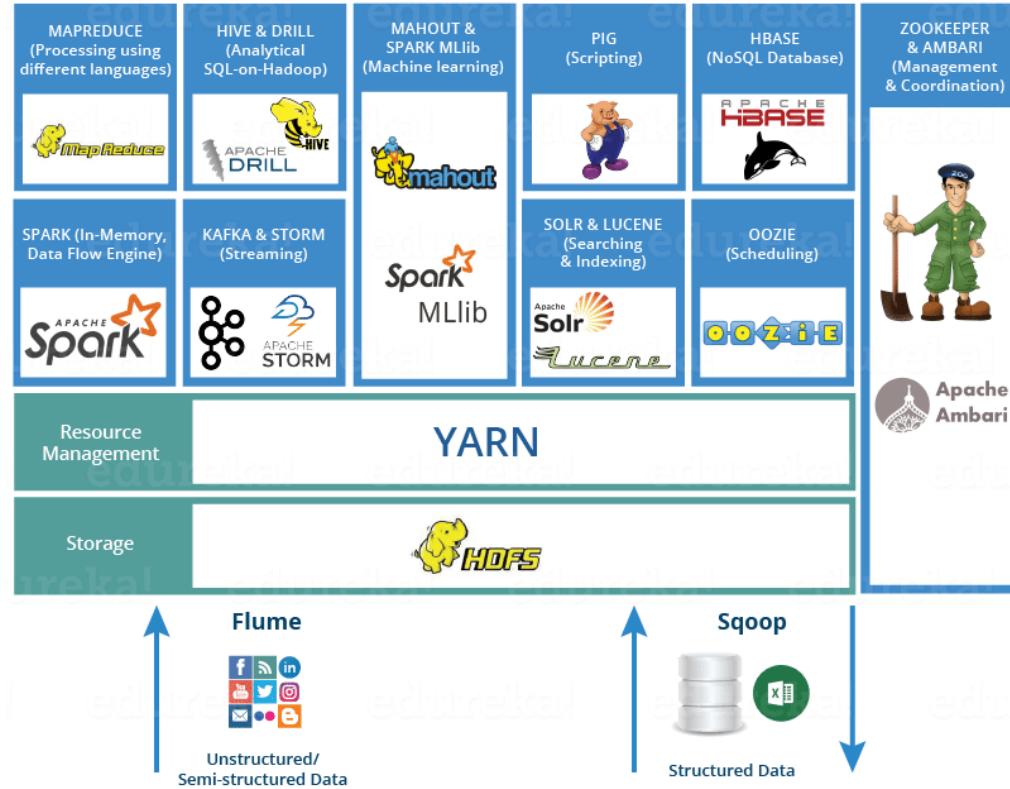


# Conclusion

- next generation of highly scalable and elastic RDBMS: NewSQL databases (from April 2011)
  - they are designed to scale out horizontally on shared nothing machines,
  - still provide ACID guarantees,
  - applications interact with the database primarily using SQL,
  - the system employs a lock-free concurrency control scheme to avoid user shut down,
  - the system provides higher performance than available from the traditional systems.
- Examples: MySQL Cluster (most mature solution), VoltDB, Clustrix, ScalArc, etc.



# Hadoop Ecosystem



# HBase tutorial

---



# Hbase tutorial

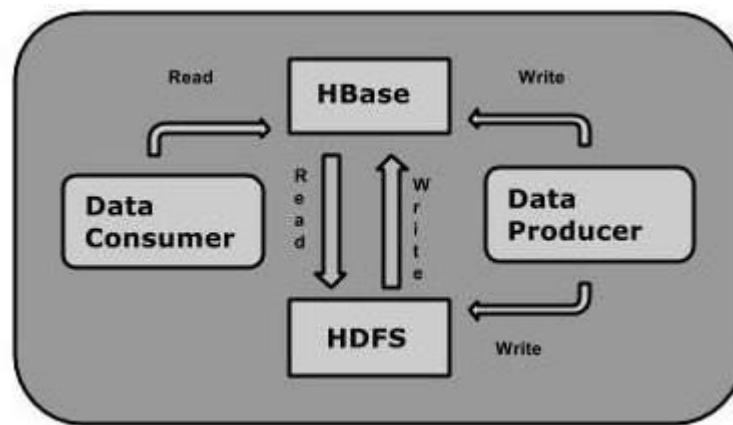
## What is HBase?

- HBase is a distributed column-oriented database built on top of the Hadoop file system.
- HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).
- It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.
- One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access



# Hbase tutorial

## What is HBase?





# Hbase tutorial

## HDFS vs HBase

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.



# Hbase tutorial

## What is HBase?

- HBase is a column-oriented database and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs.
- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

Rowid	Column Family											
	col1	col2	col3									
1												
2												
3												



# Hbase tutorial

## Column Oriented and Row Oriented

Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.



# Hbase tutorial

## HBase and RDBMS

HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.
It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.
It has de-normalized data.	It will have normalized data.
It is good for semi-structured as well as structured data.	It is good for structured data.



# Hbase tutorial

## Features of HBase

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.
- It has easy java API for client.
- It provides data replication across clusters.



# Hbase tutorial

## Where to Use HBase

- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top of Hadoop and HDFS.

# Hbase tutorial





# Hbase tutorial

- Accessing HBase by using the HBase Shell
- Command: **hbase shell**

```
[cloudera@quickstart ~]$ hbase shell
2020-10-13 06:35:21,354 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> █
```



# Hbase tutorial

- Check the shell functioning before proceeding further. Use the **list** command for this purpose. List is a command used to get the list of all the tables in HBase.

```
hbase(main):001:0> list
TABLE
0 row(s) in 0.1460 seconds

=> []
hbase(main):002:0> █
```



# Hbase tutorial

- Command: **status**

This command returns the status of the system including the details of the servers running on the system. Its syntax is as follows:

```
|hbase(main):002:0> status  
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load
```

- Command: **table\_help**

This command guides you what and how to use table-referenced commands.

Given below is the syntax to use this command.



# Hbase tutorial

## Creating a Table using HBase Shell

Command: create '<table name>','<column family>'

Given below is a sample schema of a table named emp. It has two column families: "personal data" and "professional data".

Row key	personal data	professional data

```
hbase(main):006:0> create 'emp', 'personal data', 'professional data'  
0 row(s) in 1.4800 seconds
```

```
=> Hbase::Table - emp  
hbase(main):007:0> list ← verify whether the table is created using the list command  
TABLE  
emp  
1 row(s) in 0.0110 seconds  
  
=> ["emp"]
```



# Hbase tutorial

## Creating a Table using HBase Shell

Check the table

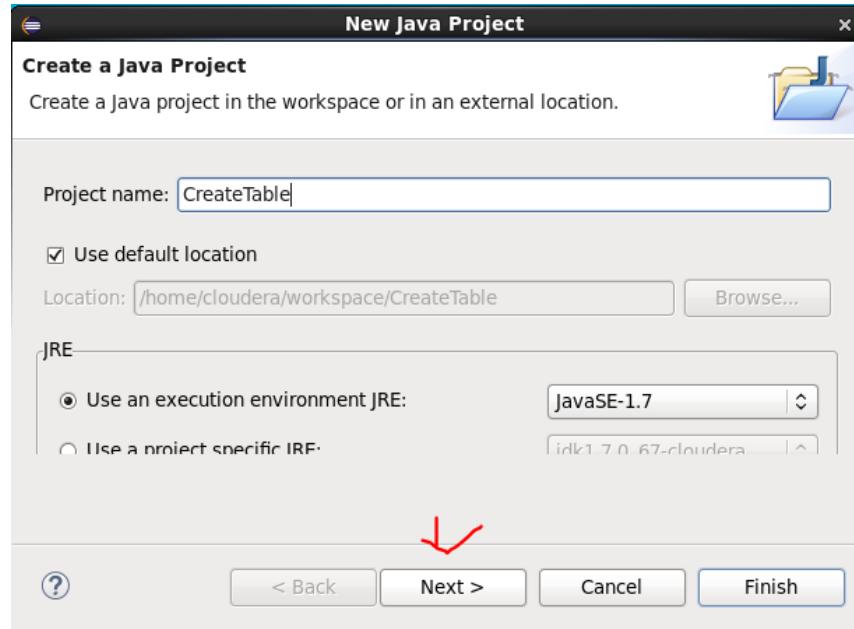
```
hbase(main):009:0> describe 'emp'
Table emp is ENABLED
emp
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal data', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'professional data', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
2 row(s) in 0.0490 seconds
```



# Hbase tutorial

## Creating a Table Using java API

- Create Java Project

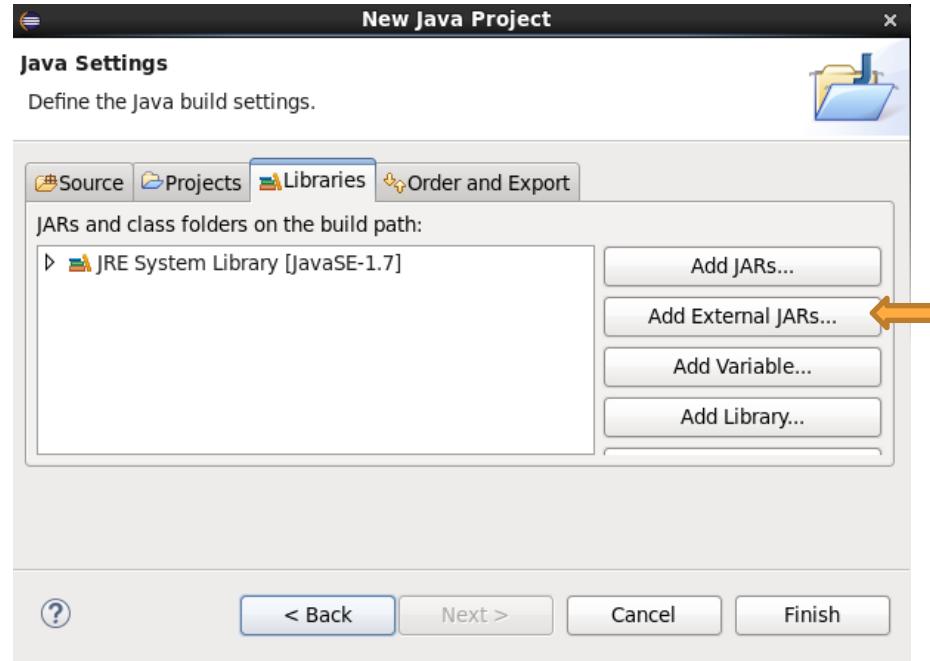




# Hbase tutorial

## Creating a Table Using java API

- Add External JARs

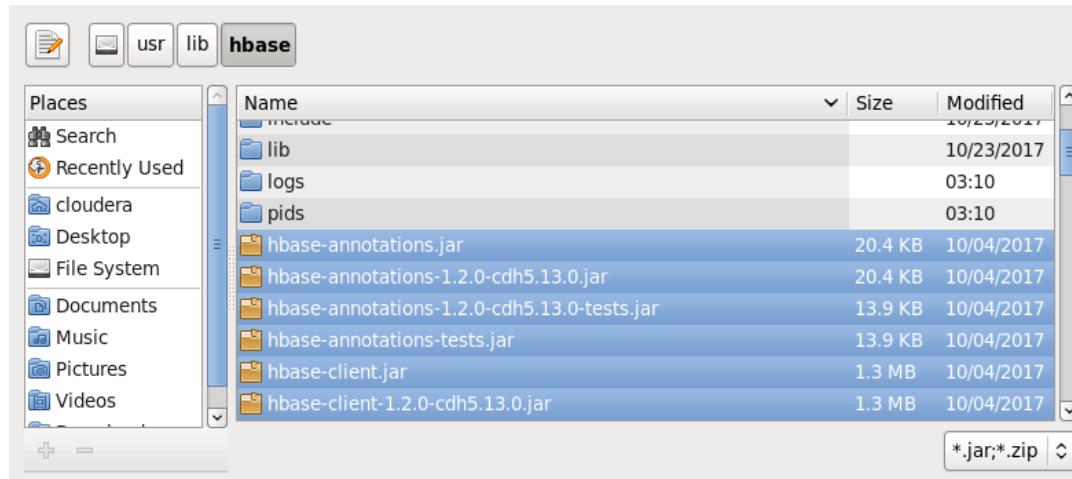




# Hbase tutorial

## Creating a Table Using java API

- Add all .jar files in **/usr/lib/hbase**

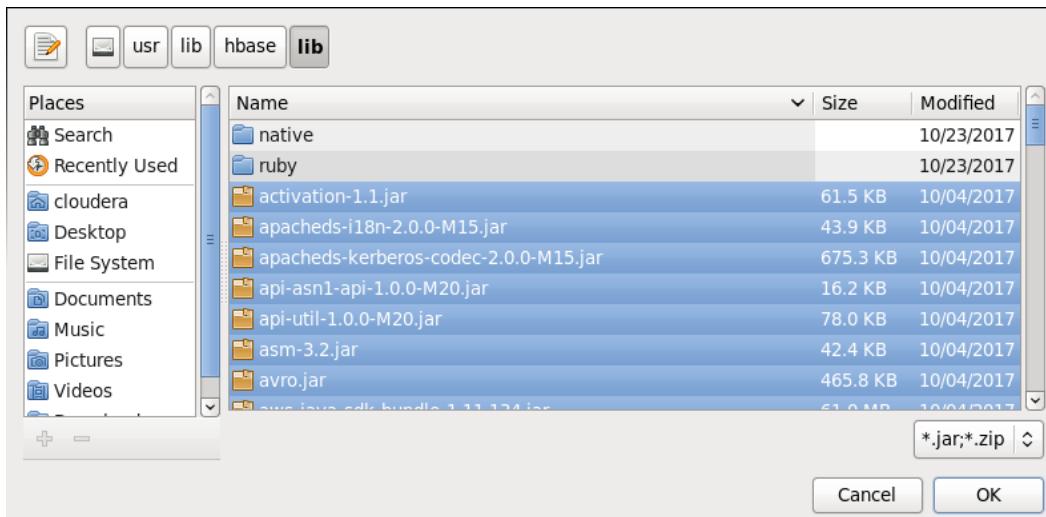




# Hbase tutorial

## Creating a Table Using java API

- Add all .jar files in **/usr/lib/hbase/lib**

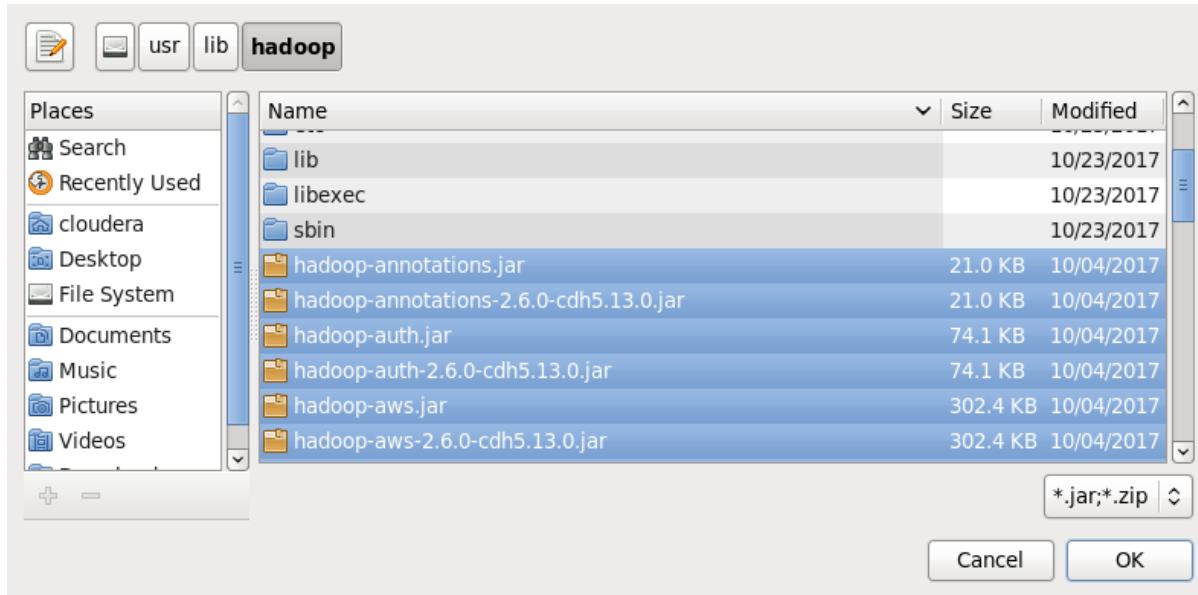




# Hbase tutorial

## Creating a Table Using java API

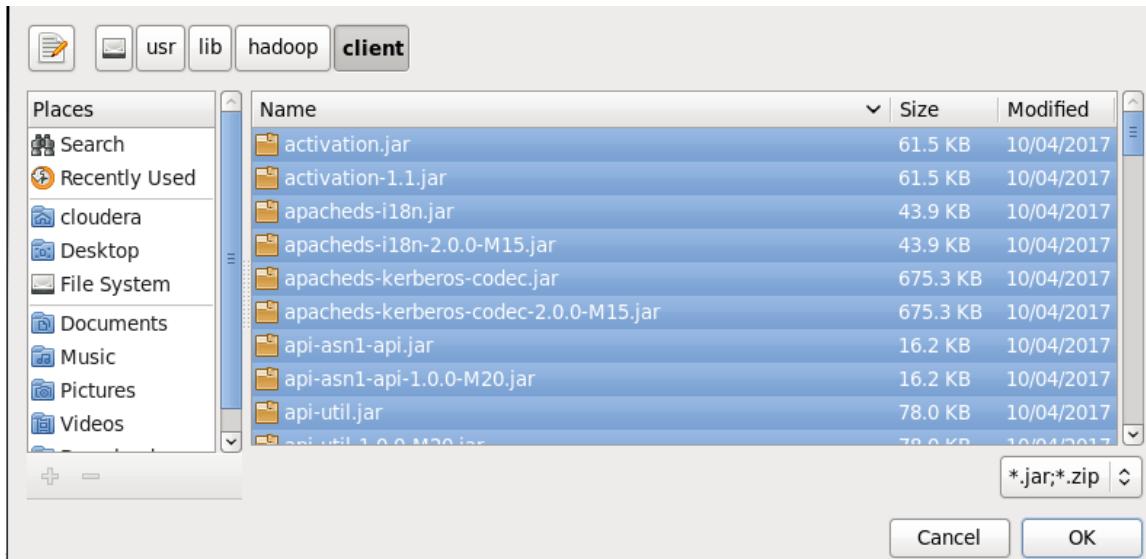
- Add all .jar files in **/usr/lib/hadoop**





# Hbase tutorial

- Add all .jar files in **/usr/lib/hadoop/client**

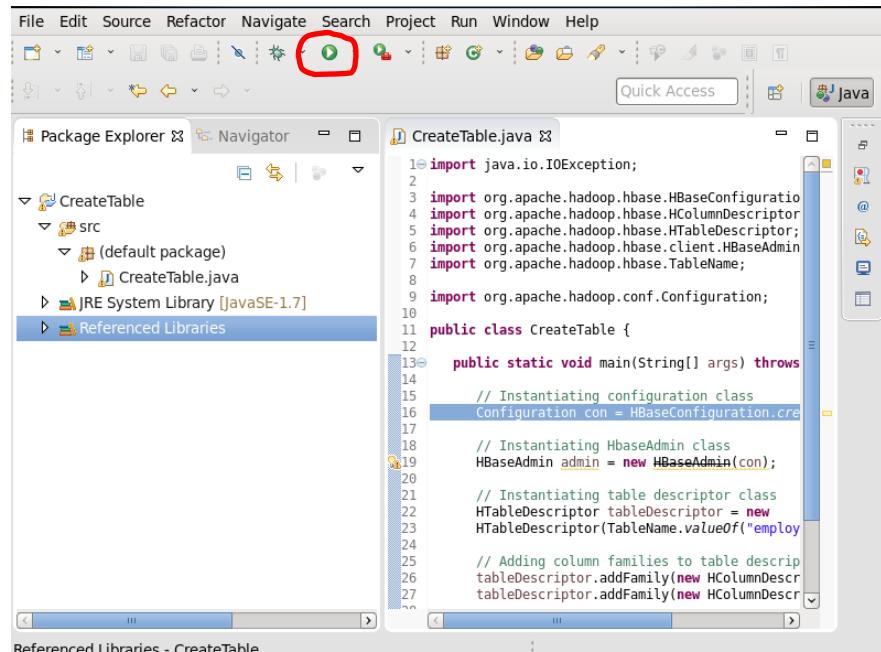




# Hbase tutorial

## Creating a Table Using java API

- Create new .java file and run it

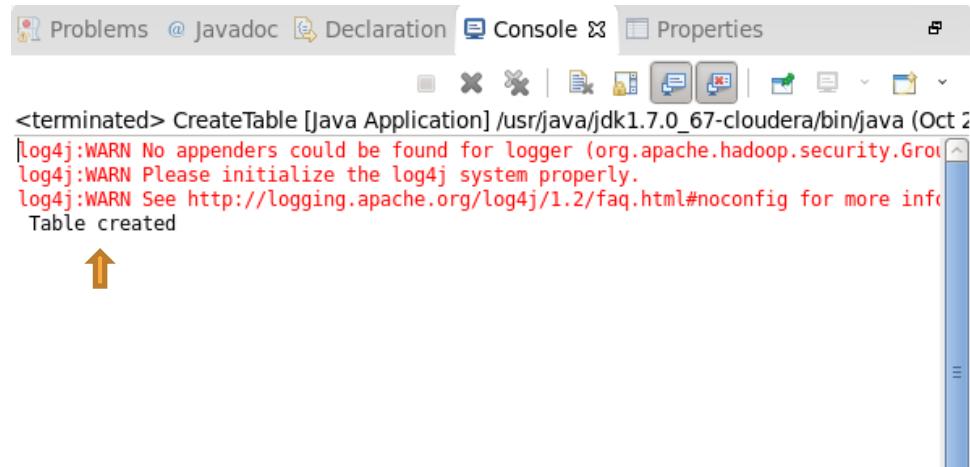


```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java
Package Explorer Navigator CreateTable.java
CreateTable
src (default package)
CreateTable.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
import java.io.IOException;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.conf.Configuration;
public class CreateTable {
    public static void main(String[] args) throws
        // Instantiating configuration class
        Configuration con = HBaseConfiguration.create();
        // Instantiating HbaseAdmin class
        HBaseAdmin admin = new HBaseAdmin(con);
        // Instantiating table descriptor class
        HTableDescriptor tableDescriptor = new
        HTableDescriptor(TableName.valueOf("employ"));
        // Adding column families to table descriptor
        tableDescriptor.addFamily(new HColumnDescriptor());
        tableDescriptor.addFamily(new HColumnDescriptor());
}
```



# Hbase tutorial

- The console output should be like this



A screenshot of an IDE interface showing the 'Console' tab selected. The output window displays the following log message:

```
<terminated> CreateTable [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Oct 2
[Log4j:WARN No appenders could be found for logger (org.apache.hadoop.security.GroupInfo).
[Log4j:WARN Please initialize the log4j system properly.
[Log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more information.
Table created
```

An orange arrow points upwards from the bottom left towards the 'Table created' message.



# Hbase tutorial

## *Creating a Table Using java API*

- Check if the table **employee** has been created

```
hbase(main):001:0> list
TABLE
emp
employee
2 row(s) in 0.2330 seconds

=> ["emp", "employee"]
hbase(main):002:0> █
```

# Hbase tutorial

## Creating a Table Using java API

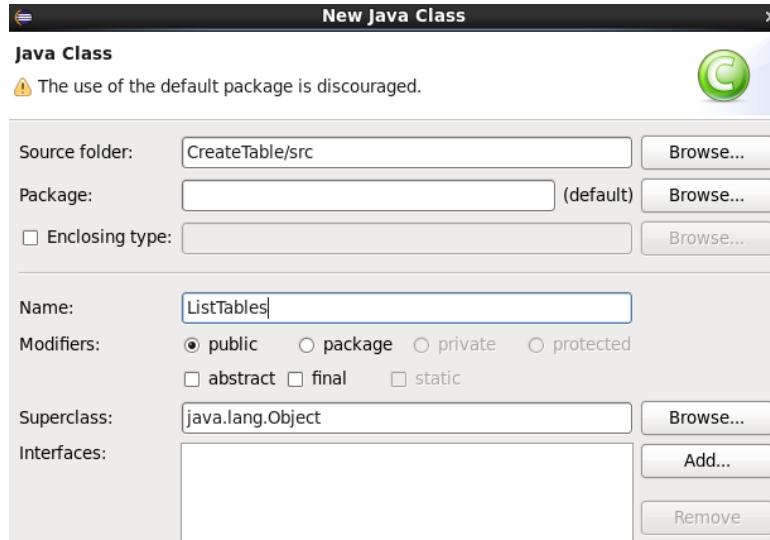
```
hbase(main):010:0> describe 'employee'
Table employee is ENABLED
employee
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'professional', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
2 row(s) in 0.0200 seconds
```



# Hbase tutorial

## *Listing Tables Using Java API*

- Create new Java file in the same project





# Hbase tutorial

## *Listing Tables Using Java API*

- Paste the code and execute it

```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java
Package Explorer Navigator
CreateTable
  src
    (default package)
      CreateTable.java
      ListTables.java
JRE System Library [JavaSE-1.7]
Referenced Libraries

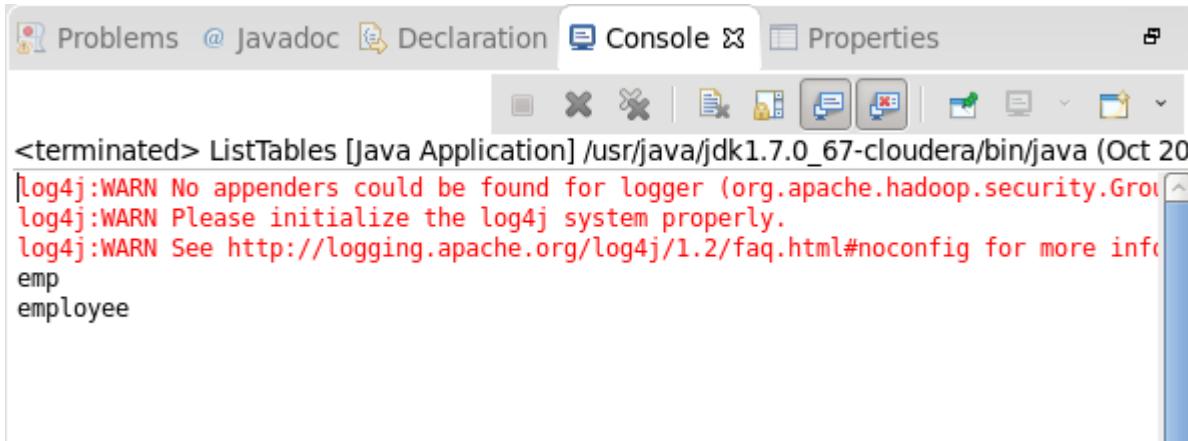
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.MasterNotRunningException;
import org.apache.hadoop.hbase.client.HBaseAdmin;
public class ListTables {
    public static void main(String args[]) throws
        // Instantiating a configuration class
        Configuration conf = HBaseConfiguration.create();
        // Instantiating HBaseAdmin class
        HBaseAdmin admin = new HBaseAdmin(conf);
        // Getting all the list of tables using HB
        HTableDescriptor[] tableDescriptor = admin.listTables();
        // printing all the table names.
        for (int i=0; i<tableDescriptor.length;i++)
            System.out.println(tableDescriptor[i].getNameAsString());
}
```



# Hbase tutorial

## *Listing Tables Using Java API*

- The console output should be like this



The screenshot shows a Java application window with the title 'ListTables [Java Application]'. The application has terminated, as indicated by the '<terminated>' prefix. The output in the console window is as follows:

```
<terminated> ListTables [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Oct 20,
log4j:WARN No appenders could be found for logger (org.apache.hadoop.security.GroupInfo).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more information.
emp
employee
```

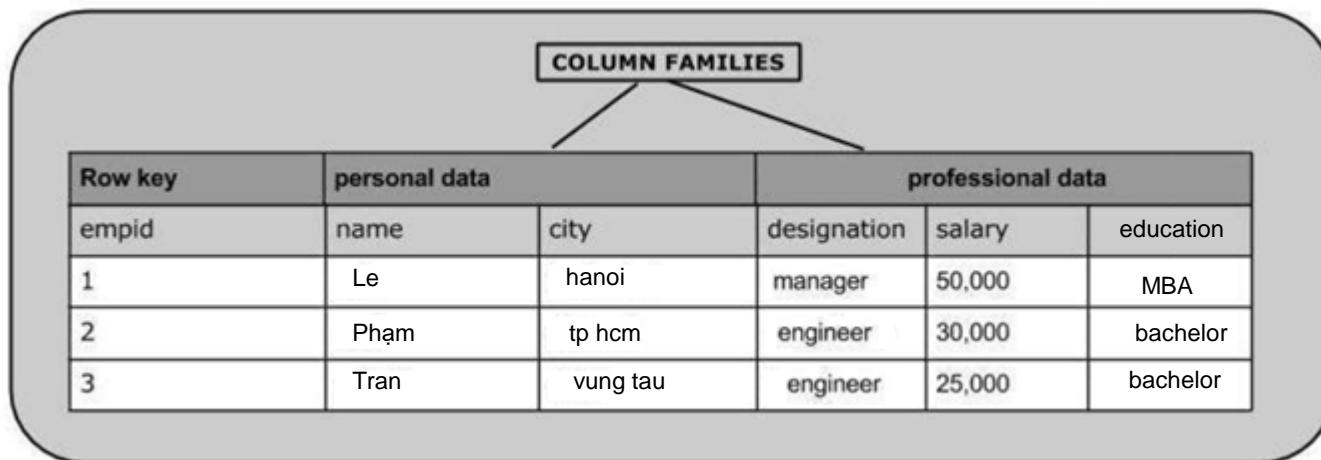


# Hbase tutorial

## Writing Data to HBase

Using **put** command, you can insert rows into a table. Its syntax is as follows:

```
put '<table name>','<row name>','<colfamily:colname>','<value>'
```





# Hbase tutorial

## Writing Data to HBase

- Let us insert the first row values into the emp table as shown below

```
hbase(main):011:0> put 'emp','1','personal data:name','Le'  
0 row(s) in 0.1250 seconds  
  
hbase(main):012:0> put 'emp','1','personal data:city','hanoi'  
0 row(s) in 0.0050 seconds  
  
hbase(main):013:0> put 'emp','1','professional data:task','manager'  
0 row(s) in 0.0130 seconds  
  
hbase(main):015:0> put 'emp','1','professional data:salary','5000'  
0 row(s) in 0.0060 seconds  
  
hbase(main):016:0> put 'emp','1','professional data:education','MBA'  
0 row(s) in 0.0080 seconds
```



# Hbase tutorial

## Writing Data to HBase

- Check the content of the table

```
hbase(main):017:0> scan 'emp'
ROW                                COLUMN+CELL
1                                  column=personal data:city, timestamp=1603200548353, value=hanoi
1                                  column=personal data:name, timestamp=1603200496978, value=Le
1                                  column=professional data:education, timestamp=1603200731642, value=MBA
1                                  column=professional data:salary, timestamp=1603200695604, value=5000
1                                  column=professional data:task, timestamp=1603200595474, value=manager
1 row(s) in 0.0150 seconds
```



# Hbase tutorial

## Writing Data to HBase

- Copy these lines and paste in the Hbase shell (you can type them if you want)

```
hbasePut
put 'emp','1','personal data:name','Le'
put 'emp','1','personal data:city','hanoi'
put 'emp','1','professional data:task','manager'
put 'emp','1','professional data:salary','5000'
put 'emp','1','professional data:education','MBA'
put 'emp','2','personal data:name','Pham'
put 'emp','2','personal data:city','tp hcm'
put 'emp','2','professional data:task','engineer'
put 'emp','2','professional data:salary','3000'
put 'emp','2','professional data:education','bachelor'
put 'emp','3','personal data:name','Tran'
put 'emp','3','personal data:city','tp hcm'
put 'emp','3','professional data:task','vung tau'
put 'emp','3','professional data:salary','2500'
put 'emp','3','professional data:education','bachelor'
```



# Hbase tutorial

## Writing Data to HBase

- Check the content of the table

```
| hbase(main):028:0> scan 'emp'
ROW                         COLUMN+CELL
 1      column=personal data:city, timestamp=1603201878949, value=hanoi
 1      column=personal data:name, timestamp=1603201878923, value=Le
 1      column=professional data:education, timestamp=1603201880630, value=MBA
 1      column=professional data:salary, timestamp=1603201878997, value=5000
 1      column=professional data:task, timestamp=1603201878979, value=manager
 2      column=personal data:city, timestamp=1603201992646, value=tp hcm
 2      column=personal data:name, timestamp=1603201992611, value=Pham
 2      column=professional data:education, timestamp=1603201992708, value=bachelor
 2      column=professional data:salary, timestamp=1603201992686, value=3000
 2      column=professional data:task, timestamp=1603201992671, value=engineer
 3      column=personal data:city, timestamp=1603201992738, value=tp hcm
 3      column=personal data:name, timestamp=1603201992721, value=Tran
 3      column=professional data:education, timestamp=1603201992787, value=bachelor
 3      column=professional data:salary, timestamp=1603201992774, value=2500
 3      column=professional data:task, timestamp=1603201992756, value=vung tau
3 row(s) in 0.0490 seconds
```



# Hbase tutorial

## Writing Data to HbaseUsing Java API

**Step 1:** Instantiate the Configuration Class

```
Configuration conf = HbaseConfiguration.create();
```

**Step 2:**Instantiate the HTable Class

```
HTable hTable = new HTable(conf, tableName);
```

**Step 3:** Instantiate the PutClass

```
Put p = new Put(Bytes.toBytes("row id"));
```

This class requires the row name you want to  
insert the data into, in string format.

**Step 4:** Insert Data

```
p.add(Bytes.toBytes("column family "), Bytes.toBytes("column name"), Bytes.toBytes("value"));
```

**Step 5:** Save the Data in Table

```
hTable.put(p);
```

**Step 6:** Close the HTable Instance

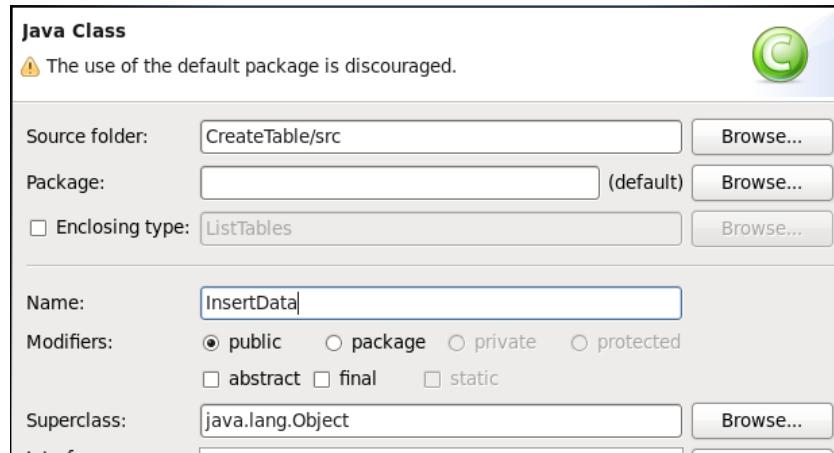
```
hTable.close();
```



# Hbase tutorial

## Writing Data to HbaseUsing Java API

- Create new .java file





# Hbase tutorial

## Writing Data to HbaseUsing Java API

- Paste the code, save, and run it

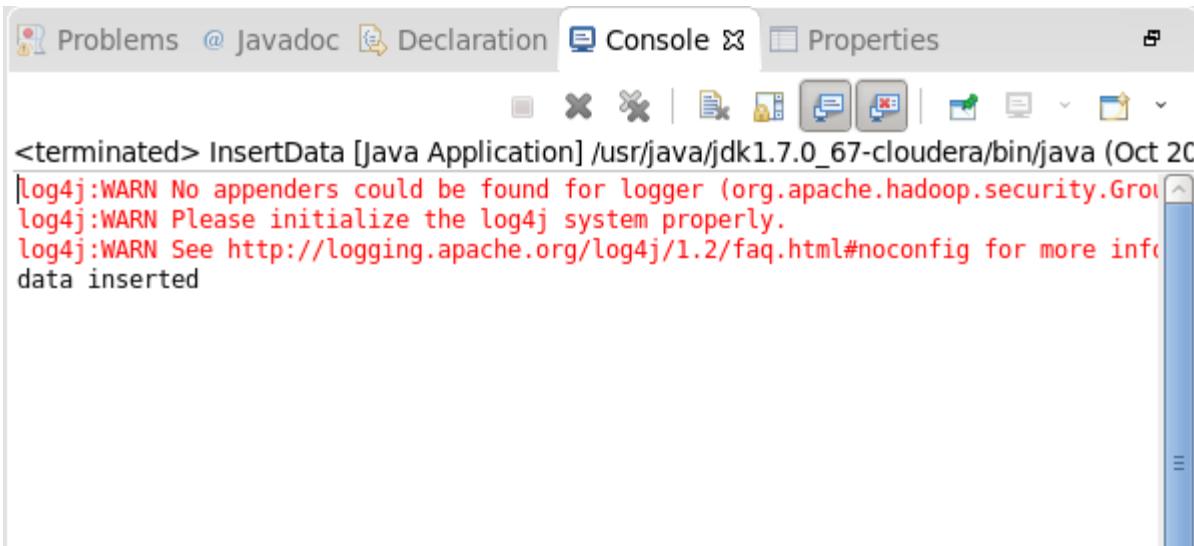
The screenshot shows a Java development environment with the following details:

- Project Structure:** A package named "CreateTable" is selected. It contains a "src" folder which includes three files: "CreateTable.java", "InsertData.java" (highlighted in blue), and "ListTables.java". There are also "JRE System Library [Java]" and "Referenced Libraries" entries.
- Code Editor:** The "InsertData.java" file is open and displayed. The code is as follows:

```
5 import org.apache.hadoop.hbase.HBaseConfiguration;
6 import org.apache.hadoop.hbase.client.HTable;
7 import org.apache.hadoop.hbase.util.Bytes;
8
9 public class InsertData{
10
11     public static void main(String[] args) throws IOException {
12         // Instantiating Configuration class
13         Configuration config = HBaseConfiguration.create();
14
15         // Instantiating HTable class
16         HTable hTable = new HTable(config, "emp");
17
18         // Instantiating Put class
19         // accepts a row name.
20         Put p = new Put(Bytes.toBytes("row4"));
21
22         // adding values using add() method
23         // accepts column family name, qualifier/row name ,value
24         p.add(Bytes.toBytes("personal_data"),
25               Bytes.toBytes("name"),Bytes.toBytes("Ngo"));
26
27
28
29         p.add(Bytes.toBytes("personal_data"),
30               Bytes.toBytes("city"),Bytes.toBytes("da lat"));
31 }
```



# Hbase tutorial



The screenshot shows a Java application window titled "InsertData [Java Application]". The console tab is active, displaying the following log output:

```
<terminated> InsertData [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Oct 20
log4j:WARN No appenders could be found for logger (org.apache.hadoop.security.GroupInfo).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more information.
data inserted
```



# Hbase tutorial

- Check the result

```
hbase(main):029:0> scan 'emp'
ROW                                COLUMN+CELL
 1                                  column=personal data:city, timestamp=1603201878949, value=hanoi
 1                                  column=personal data:name, timestamp=1603201878923, value=Le
 1                                  column=professional data:education, timestamp=1603201880630, value=MBA
 1                                  column=professional data:salary, timestamp=1603201878997, value=5000
 1                                  column=professional data:task, timestamp=1603201878979, value=manager
 2                                  column=personal data:city, timestamp=1603201992646, value=tp hcm
 2                                  column=personal data:name, timestamp=1603201992611, value=Pham
 2                                  column=professional data:education, timestamp=1603201992708, value=bachelor
 2                                  column=professional data:salary, timestamp=1603201992686, value=3000
 2                                  column=professional data:task, timestamp=1603201992671, value=engineer
 3                                  column=personal data:city, timestamp=1603201992738, value=tp hcm
 3                                  column=personal data:name, timestamp=1603201992721, value=Tran
 3                                  column=professional data:education, timestamp=1603201992787, value=bachelor
 3                                  column=professional data:salary, timestamp=1603201992774, value=2500
 3                                  column=professional data:task, timestamp=1603201992756, value=vung tau
row4                               column=personal data:city, timestamp=1603204408960, value=da lat
row4                               column=personal data:name, timestamp=1603204408960, value=Ngo
row4                               column=professional data:education, timestamp=1603204408960, value=bachelor
row4                               column=professional data:salary, timestamp=1603204408960, value=2000
row4                               column=professional data:task, timestamp=1603204408960, value=developer
4 row(s) in 0.0400 seconds
```



# Hbase tutorial

- Edit the code

```
Put p = new Put(Bytes.toBytes("row4")); ➔ Put p = new Put(Bytes.toBytes("4"));
```

- Then run the code



# Hbase tutorial

## Writing Data to HbaseUsing Java API

- Scan the table to see the result.
- In the table, “4” and “row4” are different rows

```
hbase(main):030:0> scan 'emp'
ROW          COLUMN+CELL
1            column=personal data:city, timestamp=1603201878949, value=hanoi
1            column=personal data:name, timestamp=1603201878923, value=Le
1            column=professional data:education, timestamp=1603201880630, value=MBA
1            column=professional data:salary, timestamp=1603201878997, value=5000
1            column=professional data:task, timestamp=1603201878979, value=manager
2            column=personal data:city, timestamp=1603201992646, value=tp hcm
2            column=personal data:name, timestamp=1603201992611, value=Phan
2            column=professional data:education, timestamp=1603201992708, value=bachelor
2            column=professional data:salary, timestamp=1603201992686, value=3000
2            column=professional data:task, timestamp=1603201992671, value=engineer
3            column=personal data:city, timestamp=1603201992738, value=tp hcm
3            column=personal data:name, timestamp=1603201992721, value=Tran
3            column=professional data:education, timestamp=1603201992787, value=bachelor
3            column=professional data:salary, timestamp=1603201992774, value=2500
3            column=professional data:task, timestamp=1603201992756, value=vung tau
4            column=personal data:city, timestamp=1603205278909, value=da lat
4            column=personal data:name, timestamp=1603205278909, value=Ngo
4            column=professional data:education, timestamp=1603205278909, value=bachelor
4            column=professional data:salary, timestamp=1603205278909, value=2000
4            column=professional data:task, timestamp=1603205278909, value=developer
row4          column=personal data:city, timestamp=1603204408960, value=da lat
row4          column=personal data:name, timestamp=1603204408960, value=Ngo
row4          column=professional data:education, timestamp=1603204408960, value=bachelor
row4          column=professional data:salary, timestamp=1603204408960, value=2000
row4          column=professional data:task, timestamp=1603204408960, value=developer

5 row(s) in 0.1000 seconds
```



# Hbase tutorial

## Reading Data using HBase Shell

- Command: get ‘<table name>’ , ‘<row id>’

```
hbase(main):032:0> get 'emp','row4'
COLUMN          CELL
personal data:city    timestamp=1603204408960, value=da lat
personal data:name     timestamp=1603204408960, value=Ngo
professional data:education timestamp=1603204408960, value=bachelor
n
professional data:salary   timestamp=1603204408960, value=2000
professional data:task      timestamp=1603204408960, value=developer
5 row(s) in 0.0150 seconds
```



# Hbase tutorial

## Reading a Specific Column using HBase Shell

- Command: get 'table name', 'row id', {COLUMN => 'column family:column name '}

```
hbase(main):034:0> get 'emp', 'row4', {COLUMN => 'personal data:city'}
COLUMN          CELL
personal data:city      timestamp=1603204408960, value=da lat
1 row(s) in 0.0050 seconds
```



# Hbase tutorial

## Updating Data using HBase Shell

- Command: put 'table name','row id','Column family:column name','new value'

```
hbase(main):040:0> put 'emp','1','personal data:city','haiphong'
0 row(s) in 0.0190 seconds

hbase(main):041:0> scan 'emp'
ROW                                COLUMN+CELL
1                                  column=personal data:city, timestamp=1603208058069, value=haiphong
1                                  column=personal data:name, timestamp=1603201878923, value=Le
1                                  column=professional data:education, timestamp=1603201880630, value=MBA
1                                  column=professional data:salary, timestamp=1603201878997, value=5000
1                                  column=professional data:task, timestamp=1603201878979, value=manager
2                                  column=personal data:city, timestamp=1603201992646, value=tp hcm
-
```



# Hbase tutorial

## Deleting a Specific Cell in a Table

- Command: delete '<table name>', '<row id>', '<column name >', '<time stamp>'

```
hbase(main):035:0> delete 'emp', 'row4', 'personal data:city'
0 row(s) in 0.0320 seconds

hbase(main):036:0> scan 'emp'
ROW                                COLUMN+CELL
 1                                  column=personal data:city, timestamp=1603201878949, value=hanoi
 1                                  column=personal data:name, timestamp=1603201878923, value=Le
 1                                  column=professional data:education, timestamp=1603201880630, value=MBA
 1                                  column=professional data:salary, timestamp=1603201878997, value=5000
 1                                  column=professional data:task, timestamp=1603201878979, value=manager

row4                               column=personal data:name, timestamp=1603204408960, value=Ngo
row4                               column=professional data:education, timestamp=1603204408960, value=bachelor
row4                               column=professional data:salary, timestamp=1603204408960, value=2000
row4                               column=professional data:task, timestamp=1603204408960, value=developer
5 row(s) in 0.0230 seconds
```



# Hbase tutorial

## Deleting all the cells in a row

- Command: deleteall '<table name>', '<row id>'

```
hbase(main):037:0> deleteall 'emp', 'row4'
0 row(s) in 0.0080 seconds

hbase(main):038:0> scan 'emp'
ROW                                COLUMN+CELL
1                                  column=personal data:city, timestamp=1603201878949, value=hanoi
1                                  column=personal data:name, timestamp=1603201878923, value=Lê
1                                  column=professional data:education, timestamp=1603201880630, value=MBA
1                                  column=professional data:salary, timestamp=1603201878997, value=5000
1                                  column=professional data:task, timestamp=1603201878979, value=manager
2                                  column=personal data:city, timestamp=1603201992646, value=tp hcm
2                                  column=personal data:name, timestamp=1603201992611, value=Phan
2                                  column=professional data:education, timestamp=1603201992708, value=bachelor
2                                  column=professional data:salary, timestamp=1603201992686, value=3000
2                                  column=professional data:task, timestamp=1603201992671, value=engineer
3                                  column=personal data:city, timestamp=1603201992738, value=tp hcm
3                                  column=personal data:name, timestamp=1603201992721, value=Tran
3                                  column=professional data:education, timestamp=1603201992787, value=bachelor
3                                  column=professional data:salary, timestamp=1603201992774, value=2500
3                                  column=professional data:task, timestamp=1603201992756, value=vung tau
4                                  column=personal data:city, timestamp=1603205278909, value=da lat
4                                  column=personal data:name, timestamp=1603205278909, value=Ngo
4                                  column=professional data:education, timestamp=1603205278909, value=bachelor
4                                  column=professional data:salary, timestamp=1603205278909, value=2000
4                                  column=professional data:task, timestamp=1603205278909, value=developer

4 row(s) in 0.0230 seconds
```



# Hbase tutorial

## *Deleting a Column Family*

- Command: alter ‘<table name>’, ‘delete’ ⇒ ‘<column family>’

```
hbase(main):044:0> alter 'emp' , 'delete' => 'personal data'
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.1930 seconds

hbase(main):045:0> scan 'emp'
ROW                                COLUMN+CELL
 1                                  column=professional data:education, timestamp=1603201880630, value=MBA
 1                                  column=professional data:salary, timestamp=1603201878997, value=5000
 1                                  column=professional data:task, timestamp=1603201878979, value=manager
 2                                  column=professional data:education, timestamp=1603201992708, value=bachelor
 2                                  column=professional data:salary, timestamp=1603201992686, value=3000
 2                                  column=professional data:task, timestamp=1603201992671, value=engineer
 3                                  column=professional data:education, timestamp=1603201992787, value=bachelor
-
```



# Hbase tutorial

## VERSION

- When you put data into HBase, a timestamp is required.
- The timestamp can be generated automatically by the RegionServer or can be supplied by you.
- The timestamp must be unique per version of a given cell, because the timestamp identifies the version.
- To modify a previous version of a cell, for instance, you would issue a Put with a different value for the data itself, but the **same timestamp**.

Command: **put 'table name','row id','Column family:column name','new value', timestamp**



# Hbase tutorial

## VERSION

- Doing a put always creates a new version of a cell, at a certain timestamp.

```
hbase(main):005:0> get 'emp', '1' , {COLUMN => 'professional data:task'}
COLUMN          CELL
 professional data:task    timestamp=1603201878979, value=manager
1 row(s) in 0.0160 seconds
```

- Default update

```
hbase(main):008:0> put 'emp', '1' , 'professional data:task', 'CEO'
0 row(s) in 0.0770 seconds
```

```
hbase(main):009:0> get 'emp', '1' , {COLUMN => 'professional data:task',VERSION => 2}
COLUMN          CELL
 professional data:task    timestamp=1603248439909, value=CEO
1 row(s) in 0.0060 seconds
```



# Hbase tutorial

## Change the maximum number of versions

- Get 2 versions of that cell

```
hbase(main):013:0> get 'emp', '1' , {COLUMN => 'professional data:task', VERSIONS => 2}
COLUMN          CELL
  professional data:task      timestamp=1603248439909, value=CEO
1 row(s) in 0.0050 seconds
```

- We receive only the latest version of that cell (which is 'CEO')
- The reason is because the maximum number of versions defaults to 1
- Use **alter** command to change the maximum number of versions of that family column

```
hbase(main):014:0> alter 'emp', {NAME => 'professional data', VERSIONS => 3}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.3850 seconds
```



# Hbase tutorial

## Get all versions of a cell

- Let's try again to get 2 versions of that cell

```
hbase(main):015:0> get 'emp', '1' , {COLUMN => 'professional data:task', VERSIONS => 2}
COLUMN          CELL
  professional data:task    timestamp=1603248439909, value=CEO
  professional data:task    timestamp=1603201878979, value=manager
2 row(s) in 0.0080 seconds
```



# Hbase tutorial

## Update a specific version

Let's get all versions of the cell

```
hbase(main):015:0> get 'emp', '1' , {COLUMN => 'professional data:task', VERSIONS => 2}
COLUMN          CELL
 professional data:task    timestamp=1603248439909, value=CEO
 professional data:task    timestamp=1603201878979, value=manager
2 row(s) in 0.0080 seconds
```

Command: put 'table name','row id','Column family:column name','new value', timestamp

```
hbase(main):018:0> put 'emp', '1' , 'professional data:task', 'helpdesk',1603248439909
0 row(s) in 0.0110 seconds
```

```
hbase(main):020:0> get 'emp', '1' , {COLUMN => 'professional data:task', VERSIONS => 3}
COLUMN          CELL
 professional data:task    timestamp=1603248439909, value=helpdesk
 professional data:task    timestamp=1603201878979, value=manager
2 row(s) in 0.0060 seconds
```



# Hbase tutorial

## Load CSV file from HDFS to HBase

- Create a csv file (e.g. using **gedit**)

```
empl.csv X
1,Le,hanoi,manager,5000,MBA
2,Pham,tp hcm,engineer,3000,bachelor
3,Tran,vung tau, engineer,2500,bachelor|
```

- Put the file to HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -put empl.csv
```



# Hbase tutorial

## Load CSV file from HDFS to HBase

- Navigate to HBase directory

```
[cloudera@quickstart ~]$ cd /usr/lib/hbase  
[cloudera@quickstart hbase]$ █
```

- Command: `hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns= ...`

```
[cloudera@quickstart hbase]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns='HBASE_ROW_KEY,personal:name,personal:city,professional:task,professional:salary,professional:education' employee /user/cloudera/emp1.csv  
2020-10-20 22:30:53,376 INFO  [main] zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x451b2519 connecting to ZooKeeper ensemble=localhost:2181  
2020-10-20 22:30:53,387 INFO  [main] zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.5-cdh5.13.0--1, built on 10/04/2017 18:04 GMT
```

```
ImportTsv  
    Bad Lines=0  
    File Input Format Counters  
        Bytes Read=105  
    File Output Format Counters  
        Bytes Written=0
```

Important: no space before and after ',' when listing columns

```
-Dimporttsv.columns='HBASE_ROW_KEY, personal:name, personal:city, professional:task, won't run
```



# Hbase tutorial

## Load CSV file from HDFS to HBase

- Scan the table in hbase shell

```
hbase(main):031:0> scan 'employee'
ROW                                COLUMN+CELL
 1                                  column=personal:city, timestamp=1603256907282, value=hanoi
 1                                  column=personal:name, timestamp=1603256907282, value=Le
 1                                  column=professional:education, timestamp=1603256907282, value=MBA
 1                                  column=professional:salary, timestamp=1603256907282, value=5000
 1                                  column=professional:task, timestamp=1603256907282, value=manager
 2                                  column=personal:city, timestamp=1603256907282, value=tp hcm
 2                                  column=personal:name, timestamp=1603256907282, value=Pham
 2                                  column=professional:education, timestamp=1603256907282, value=bachelor
 2                                  column=professional:salary, timestamp=1603256907282, value=3000
 2                                  column=professional:task, timestamp=1603256907282, value=engineer
 3                                  column=personal:city, timestamp=1603256907282, value=vung tau
 3                                  column=personal:name, timestamp=1603256907282, value=Tran
 3                                  column=professional:education, timestamp=1603256907282, value=bachelor
 3                                  column=professional:salary, timestamp=1603256907282, value=2500
 3                                  column=professional:task, timestamp=1603256907282, value= engineer
3 row(s) in 0.0270 seconds
```



# Hbase tutorial

## Load data from Hive to HBase

- Check the Hive table **student**

```
hive> select * from student;
OK
123451 Quynh    Hadoop   22
123452 Tai      Java     22
123453 Truong   Python   23
123454 Nghia   Hadoop   24
123455 Thuy    Java     23
123456 Hao      Python   24
123457 Hien    Hadoop   22
123458 Phuong  Java     23
123459 Hai      Python   23
123460 Phuong  Hadoop   24
Time taken: 0.882 seconds, Fetched: 10 row(s)
hive> describe student;
OK
id          int
name        string
course      string
age         int
Time taken: 0.079 seconds, Fetched: 4 row(s)
```



# Hbase tutorial

## Create HBase-Hive Mapping table

- Create another hive table which actually points to an HBase table
- **hbase\_student** is the name of Hive table
- **studen\_hbase** is the name of Hbase table linked to the Hive table above
- Command: `create table hbase_student (id int,name string,course string,age int) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,personal:name,personal:course,additional:age") TBLPROPERTIES ("hbase.table.name" = "studen_hbase");`

```
hive> create table hbase_student (id int,name string,course string,age int) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,personal:name,personal:course,additional:age") TBLPROPERTIES ("hbase.table.name" = "studen_hbase");
OK
Time taken: 3.204 seconds
```



# Hbase tutorial

## Load data from Hive to HBase

- Check if the new table has been created in Hbase, then check its schema

```
hbase(main):034:0> list
TABLE
emp
employee
studen_hbase
3 row(s) in 0.0040 seconds

=> ["emp", "employee", "studen_hbase"]
hbase(main):035:0> describe 'studen_hbase'
Table studen_hbase is ENABLED
studen_hbase
COLUMN FAMILIES DESCRIPTION
{NAME => 'additional', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'personal', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
2 row(s) in 0.0130 seconds
```



# Hbase tutorial

## Load data from Hive to HBase

- Migrate hive table data to HBase

```
hive> insert into table hbase_student select * from student;
Query ID = cloudera_20201020231515_7295e24d-11a1-4350-81b3-7d1f193d0ef3
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603246922983_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1603246922983_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1603246922983_0003
Hadoop job information for Stage-0: number of mappers: 1; number of reducers: 0
2020-10-20 23:15:19,749 Stage-0 map = 0%,  reduce = 0%
2020-10-20 23:15:28,451 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 1.69 sec
MapReduce Total cumulative CPU time: 1 seconds 690 msec
Ended Job = job_1603246922983_0003
MapReduce Jobs Launched:
Stage-Stage-0: Map: 1  Cumulative CPU: 1.69 sec  HDFS Read: 12230 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 690 msec
OK
```



# Hbase tutorial

## Load data from Hive to HBase

- Check the table in Hive

```
hive> select * from hbase_student;
OK
123451 Quynh    Hadoop   22
123452 Tai      Java     22
123453 Truong   Python   23
123454 Nghia   Hadoop   24
123455 Thuy     Java     23
123456 Hao      Python   24
123457 Hien    Hadoop   22
123458 Phuong   Java     23
123459 Hai      Python   23
123460 Phuong   Hadoop   24
Time taken: 0.115 seconds, Fetched: 10 row(s)
```



# Hbase tutorial

## Load CSV file from HDFS to HBase

- Check the Hbase table

```
hbase(main):037:0> scan 'studen_hbase'
ROW                                     COLUMN+CELL
123451                                    column=additional:age, timestamp=1603260927982, value=22
123451                                    column=personal:course, timestamp=1603260927982, value=Hadoop
123451                                    column=personal:name, timestamp=1603260927982, value=Quynh
123452                                    column=additional:age, timestamp=1603260927982, value=22
123452                                    column=personal:course, timestamp=1603260927982, value=Java
123452                                    column=personal:name, timestamp=1603260927982, value=Tai
123453                                    column=additional:age, timestamp=1603260927982, value=23
123453                                    column=personal:course, timestamp=1603260927982, value=Python
123453                                    column=personal:name, timestamp=1603260927982, value=Truong
123454                                    column=additional:age, timestamp=1603260927982, value=24
123454                                    column=personal:course, timestamp=1603260927982, value=Hadoop
123454                                    column=personal:name, timestamp=1603260927982, value=Nghia
123455                                    column=additional:age, timestamp=1603260927982, value=23
123455                                    column=personal:course, timestamp=1603260927982, value=Java
123455                                    column=personal:name, timestamp=1603260927982, value=Thuy
123456                                    column=additional:age, timestamp=1603260927982, value=24
123456                                    column=personal:course, timestamp=1603260927982, value=Python
123456                                    column=personal:name, timestamp=1603260927982, value=Hao
123457                                    column=additional:age, timestamp=1603260927982, value=22
```



# Hbase tutorial

## Dropping a Table using HBase Shell

- Using the drop command, you can delete a table. Before dropping a table, you have to disable it.

```
hbase(main):039:0> disable 'emp'  
0 row(s) in 2.4110 seconds  
  
hbase(main):040:0> drop 'emp'  
0 row(s) in 1.3980 seconds  
  
hbase(main):041:0> list  
TABLE  
employee  
studen_hbase  
2 row(s) in 0.0240 seconds  
  
=> ["employee", "studen_hbase"]
```



# Hbase tutorial

## Hfile stored in HDFS

- Open HUE and use File Browsers

The screenshot shows the Hue interface with a sidebar on the left. At the top of the sidebar is a navigation bar with icons for Home, Logout, and a search bar. Below the navigation bar is a list of HDFS locations:

- default
- emp
- employee
- studen\_hbase

The 'studen\_hbase' folder is expanded, revealing its contents.

The screenshot shows the Hue interface with a sidebar on the left. At the top of the sidebar is a navigation bar with icons for Home, Logout, and a search bar. Below the navigation bar is a list of HDFS locations:

- Editor
- Dashboard
- Scheduler

Below these are sections for 'Browsers':

- Documents
- Files ←
- Tables
- Indexes
- Jobs



# Hbase tutorial

## Hfile stored in HDFS

- Navigate to `/hbase/data/default`

File Browser

Search for file name Actions Move to trash Upload New

Home / hbase / data / default

	Name	Size	User	Group	Permissions	Date
	↑		hbase	supergroup	drwxr-xr-x	October 12, 2020 05:22 AM
	.		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:09 PM
	emp		hbase	supergroup	drwxr-xr-x	October 15, 2020 06:39 PM
	employee		hbase	supergroup	drwxr-xr-x	October 20, 2020 04:58 AM
	studen_hbase		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:09 PM

Show 45 of 3 items Page 1 of 1



# Hbase tutorial

## Hfile stored in HDFS

Home / hbase / data / default / employee

	Name	Size	User	Group	Permissions	Date
	↑		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:09 PM
	.		hbase	supergroup	drwxr-xr-x	October 20, 2020 04:58 AM
	.tabledesc		hbase	supergroup	drwxr-xr-x	October 20, 2020 04:58 AM
	.tmp		hbase	supergroup	drwxr-xr-x	October 20, 2020 04:58 AM
	1edfd27630e52c45377b6b92d1bb90b0		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:12 PM

Show 45 of 3 items

Page 1 of 1

◀ ▶ ▷ ▷



# Hbase tutorial

## Hfile stored in HDFS

/ hbase / data / default / employee / 1edfd27630e52c45377b6b92d1bb90b0

	Name	Size	User	Group	Permissions	Date
	↑		hbase	supergroup	drwxr-xr-x	October 20, 2020 04:58 AM
	.		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:12 PM
	.regioninfo	43 bytes	hbase	supergroup	-rw-r--r--	October 20, 2020 04:58 AM
	.tmp		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:12 PM
	personal		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:12 PM
	professional		hbase	supergroup	drwxr-xr-x	October 20, 2020 11:12 PM
	recovered.edits		hbase	supergroup	drwxr-xr-x	October 20, 2020 07:23 PM

Important note: different column families are stored separately. When you query a row, the region server will have to grab data in multiple places (which will slow down your system).



# Hbase tutorial

## Hfile stored in HDFS

- Check the content of the Hfile (shown in binary and text format)

File Browser

A View as text

Home Page 1 to 1 of 1

Edit file Download View file location Refresh

Last modified 10/21/2020 6:12 AM User hbase Group supergroup Size 1.39 KB Mode 100644

/ hbase / data / default / employee / 1edfd27630e52c45377b6b92d1bb90b0 / professional / cabe02ffec744111b019995287a4d27d

```
000000: 44 41 54 41 42 4c 4b 2a 00 00 01 a6 00 00 01 a2 DATABLK*.....
0000010: ff ff ff ff ff ff 02 00 00 40 00 00 00 00 01 .....@....
0000020: c3 00 00 00 22 00 00 03 00 01 31 00 70 72 6f ....".....1.pro
0000030: 66 65 73 73 69 6f 6e 61 6c 65 64 75 63 61 74 69 fessionaleducati
0000040: 6f 6e 00 00 01 75 49 a3 7c 5c 04 4d 42 41 06 00 on...uI.|..MBA..
0000050: 00 00 1f 00 00 00 04 00 01 31 0c 78 72 6f 66 65 .....1.profe
0000060: 73 73 69 6f 6e 61 6c 73 61 6c 61 72 79 00 00 01 ssionalsalary...
0000070: 75 49 a3 7c c5 04 35 30 30 06 00 00 00 1d 00 uI.|..5000.....
0000080: 00 00 07 00 01 31 0c 70 72 6f 66 65 73 73 69 6f .....1.professio
0000090: 6e 61 6c 74 61 67 3b 00 00 01 75 49 a3 7c c5 04 naltask...uI.|..
00000a0: 6d 61 6e 61 67 65 72 06 00 00 00 22 00 00 00 08 manager...."....
00000b0: 00 01 32 0c 70 72 6f 66 65 73 73 69 6f 6e 61 6c ..2.professiona
00000c0: 65 64 75 63 61 74 69 6f 6e 00 00 01 75 49 a3 7c education...uI.|.
00000dd0: c5 04 62 61 63 68 65 6c 6f 72 06 00 00 00 1f 00 ..bachelor.....
00000e0: 00 00 04 00 01 32 0c 70 72 6f 66 65 73 73 69 6f .....2.professio
```



# Big Data

(Spark)

Instructor: Trong-Hop Do

November 24<sup>th</sup> 2020

**S<sup>3</sup>Lab**

*Smart Software System Laboratory*



**“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”**

– Chris Lynch, Vertica Systems



# Introduction

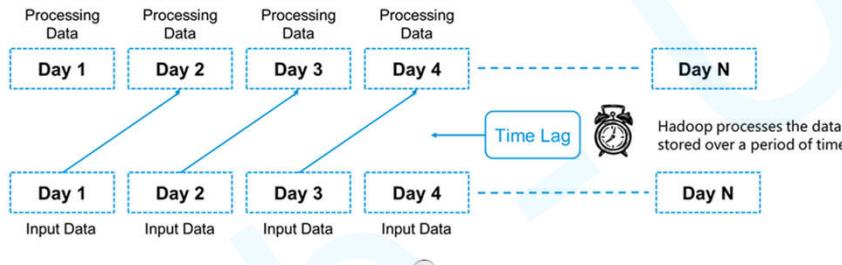
- Apache Spark is an open-source cluster computing framework for real-time processing
- Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.
- Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming



# Introduction



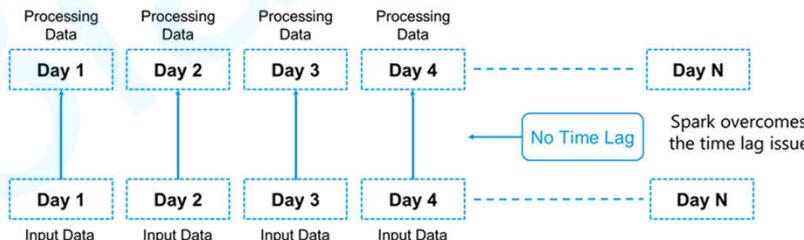
Processing Data Using MapReduce



VS



Real Time Processing in Spark



# Introduction

Hadoop implements **Batch processing** on Big Data.  
It thus cannot deliver to our **Real-Time** use case needs.



## Our Requirements:

Process data in real-time

Handle input from multiple sources

Easy to use

Faster processing





# Applications



Banking



Government



Healthcare



Telecommunications

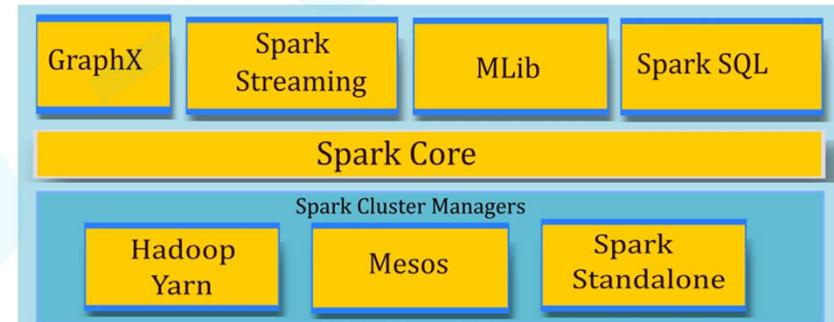


Stock Market



# Components

- Spark Core and Resilient Distributed Datasets or RDDs
- Spark SQL
- Spark Streaming
- Machine Learning Library or MLlib
- GraphX





# Components





# Components

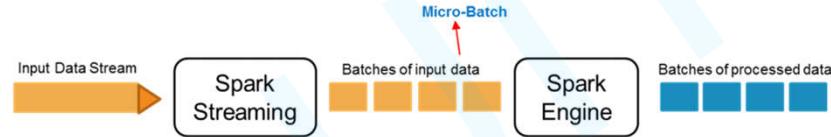
## Spark Core

- The base engine for large-scale parallel and distributed data processing.  
The core is the distributed execution engine and the Java, Scala, and Python APIs offer a platform for distributed ETL application development. Further, additional libraries which are built atop the core allow diverse workloads for streaming, SQL, and machine learning. It is responsible for:
  - Memory management and fault recovery
  - Scheduling, distributing and monitoring jobs on a cluster
  - Interacting with storage systems



# Components

## Spark Streaming



- Used to process real-time streaming data.
- It enables high-throughput and fault-tolerant stream processing of live data streams. The fundamental stream unit is **DStream** which is basically a series of **RDDs** (Resilient Distributed Datasets) to process the real-time data.

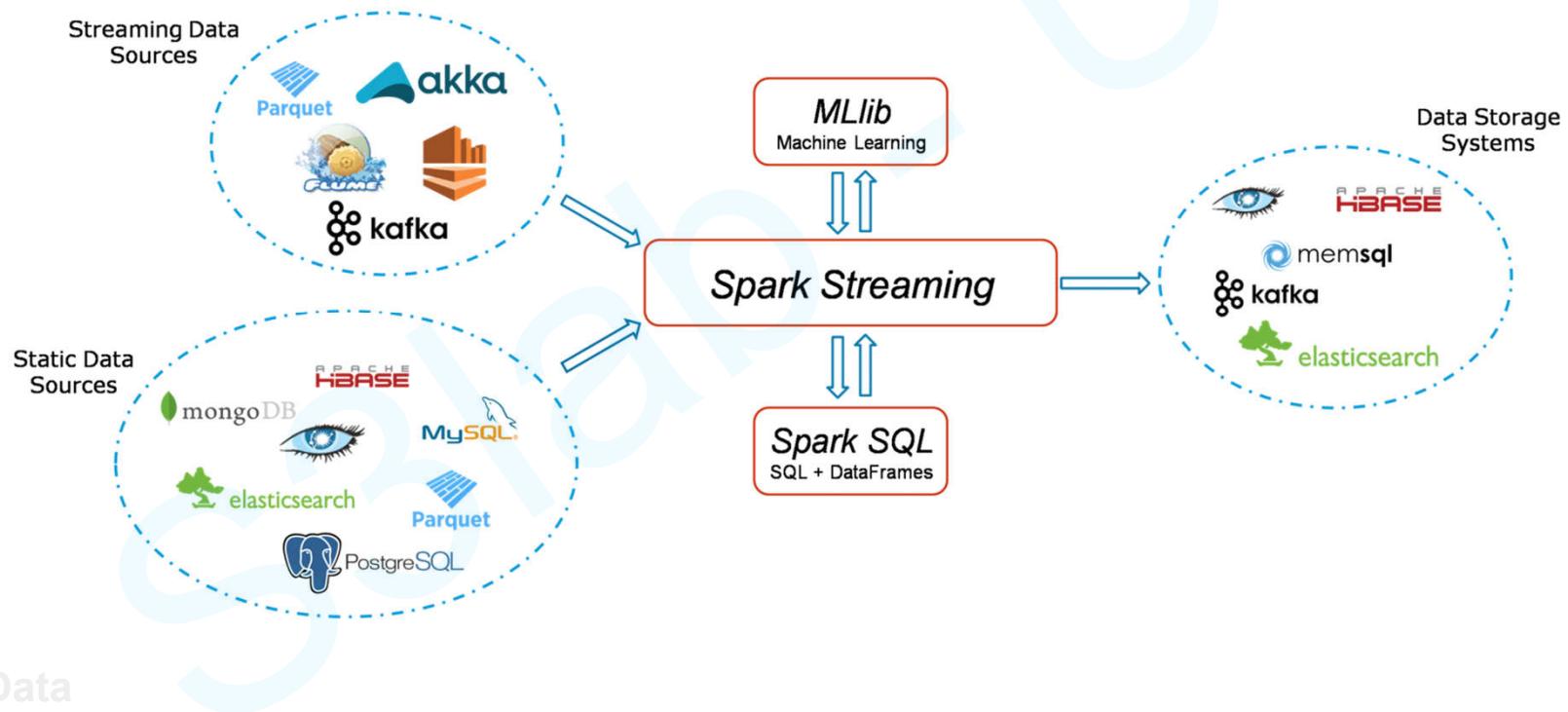


Spark Streaming is used to stream real-time data from various sources like Twitter, Stock Market and Geographical Systems and perform powerful analytics to help businesses.



# Components

## Spark Streaming - Workflow





# Components

## Spark Streaming - Fundamentals

- Streaming Context
- DStream (Discretized Stream)
- Caching
- Accumulators, Broadcast Variables and Checkpoints

# Components

## Spark Streaming - Fundamentals



**Figure:** Spark Streaming Context

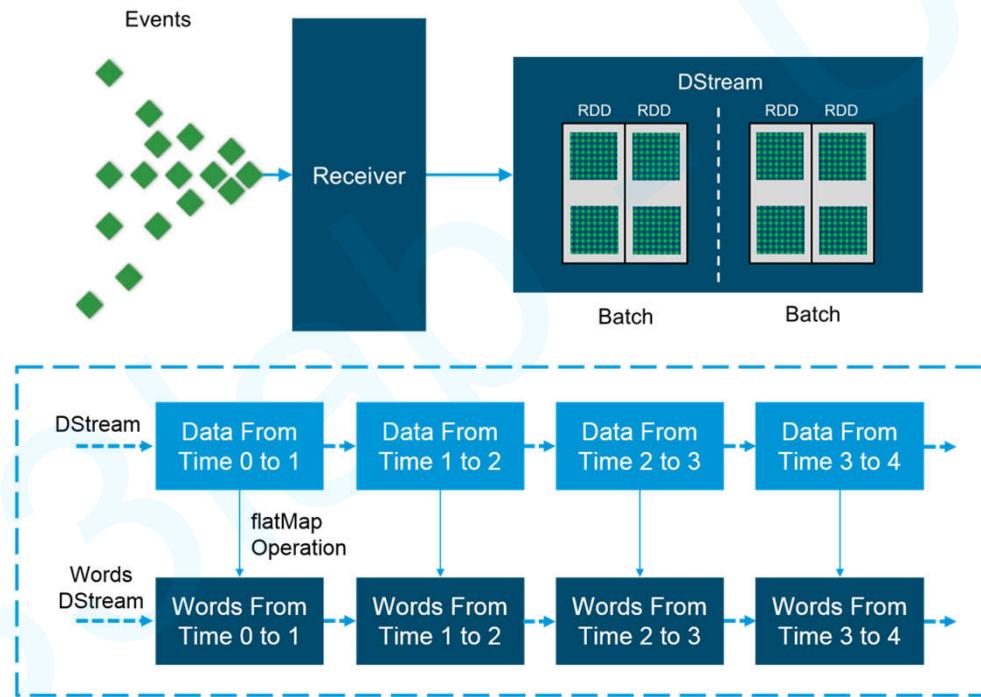


**Figure:** Default Implementation Sources



# Components

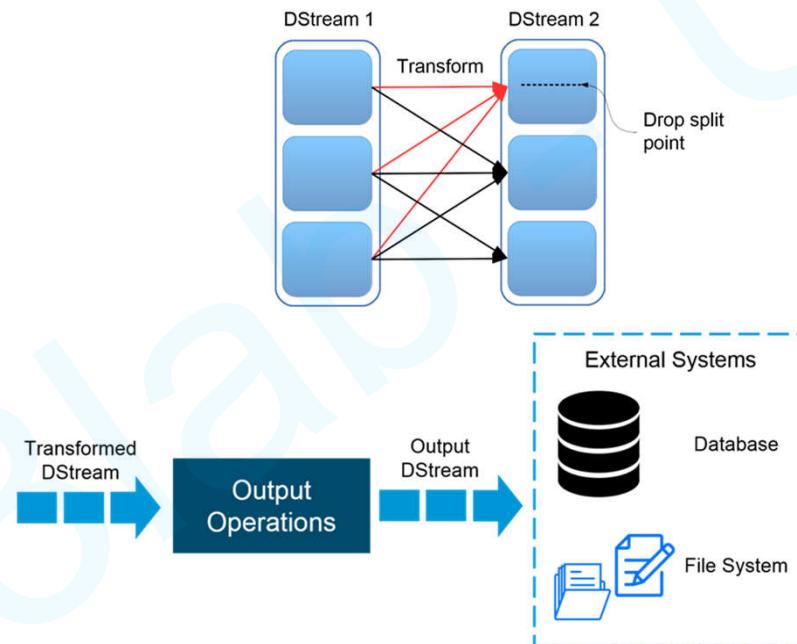
## Spark Streaming - Fundamentals





# Components

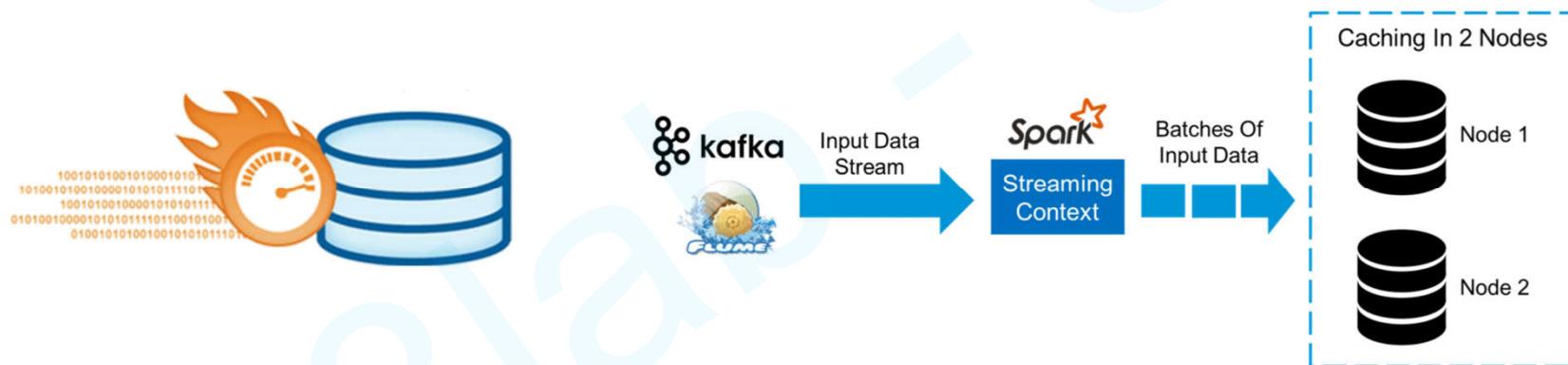
## Spark Streaming - Fundamentals





# Components

## Spark Streaming - Fundamentals





# Components

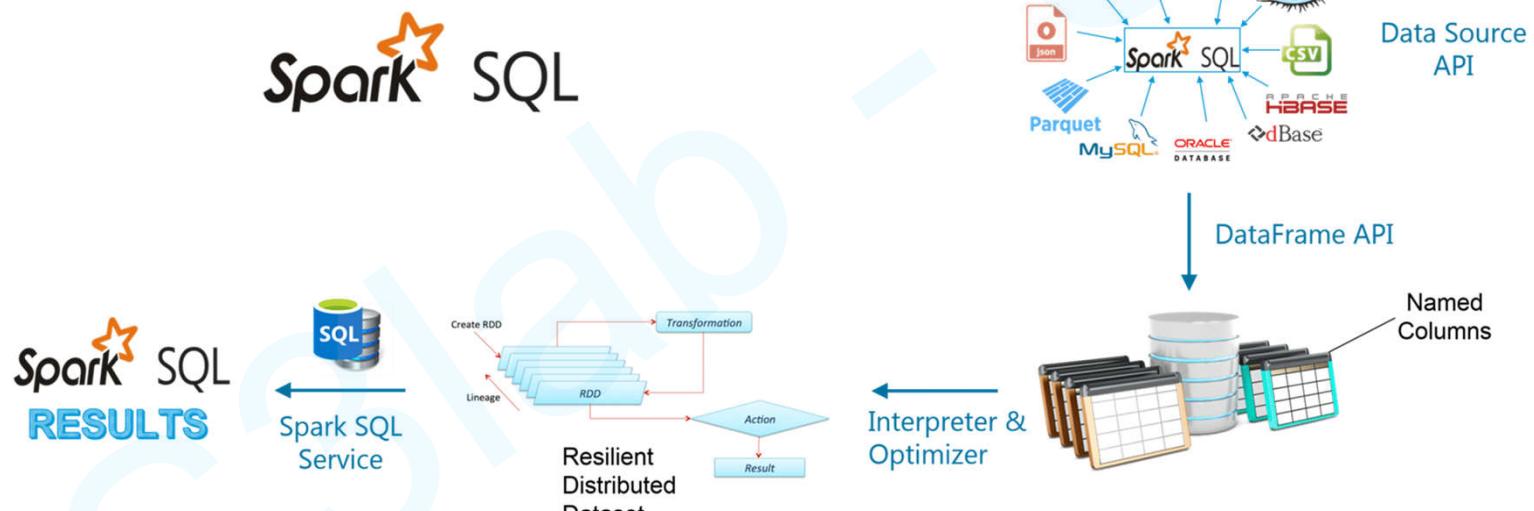
## Spark SQL

- Spark SQL integrates relational processing with Spark functional programming. Provides support for various data sources and makes it possible to weave SQL queries with code transformations thus resulting in a very powerful tool. The following are the four libraries of Spark SQL.
  - Data Source API
  - DataFrame API
  - Interpreter & Optimizer
  - SQL Service



# Components

## Spark SQL



**Figure:** The flow diagram represents a Spark SQL process using all the four libraries in sequence



# Components

## Spark SQL - Data Source API

- Universal API for loading and storing structured data.
  - Built in support for Hive, JSON, Avro, JDBC, Parquet, etc.
  - Support third party integration through spark packages
  - Support for smart sources
  - Data Abstraction and Domain Specific Language (DSL) applicable on structure and semi-structured data
  - Supports different data formats (Avro, CSV, Elastic Search and Cassandra) and storage systems (HDFS, HIVE Tables, MySQL, etc.)
  - Can be easily integrated with all Big Data tools and frameworks via Spark-Core.
  - It processes the data in the size of Kilobytes to Petabytes on a single-node cluster to multi-node clusters.



# Components

## Spark SQL - DataFrame API

- A Data Frame is a distributed collection of data organized into named column. It is equivalent to a relational table in SQL used for storing data into tables.



# Components

## Spark SQL - SQL Interpreter And Optimizer

- based on functional programming constructed in Scala.
  - provides a general framework for transforming trees, which is used to perform analysis/evaluation, optimization, planning, and run time code spawning.
  - This supports cost based optimization (run time and resource utilization is termed as cost) and rule based optimization, making queries run much faster than their RDD (Resilient Distributed Dataset) counterparts.



# Components

## Spark SQL - SQL Service

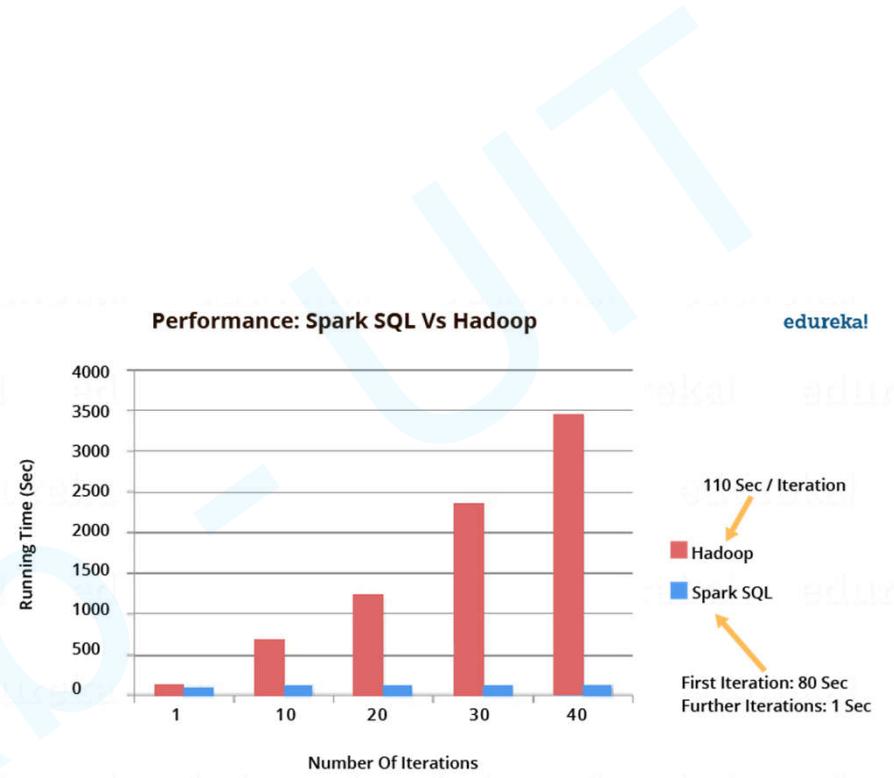
- SQL Service is the entry point for working along structured data in Spark.  
It allows the creation of DataFrame objects as well as the execution of SQL queries.



# Components

## Spark SQL - Features

- Integration With Spark
- Uniform Data Access
- Hive Compatibility
- Standard Connectivity
- Performance And Scalability
- User Defined Functions





# Components

## GraphX

- GraphX is the Spark API for graphs and graph-parallel computation. Thus, it extends the Spark RDD with a Resilient Distributed Property Graph. The property graph is a directed multigraph which can have multiple edges in parallel. Every edge and vertex have user defined properties associated with it. Here, the parallel edges allow multiple relationships between the same vertices.



# Components

## GraphX

- GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and mapReduceTriplets) as well as an optimized variant of the Pregel API.
- In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.
- GraphX unifies ETL (Extract, Transform & Load) process, exploratory analysis and iterative graph computation within a single system.



# Components

GraphX - use cases

- Disaster Detection System



- Page Rank



- Financial Fraud Detection



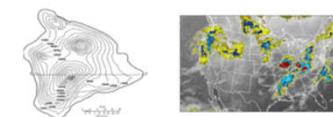
- Business Analysis

- Machine Learning, understanding the customer purchase trends



- Geographic Information Systems

- Watershed delineation and weather prediction



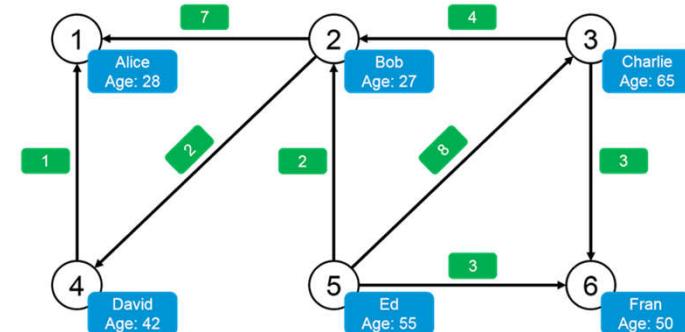
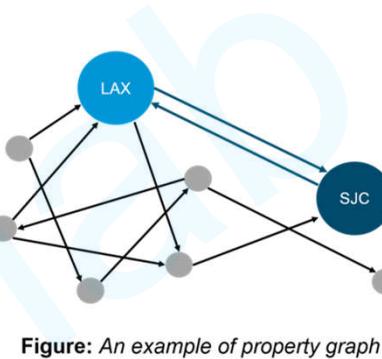
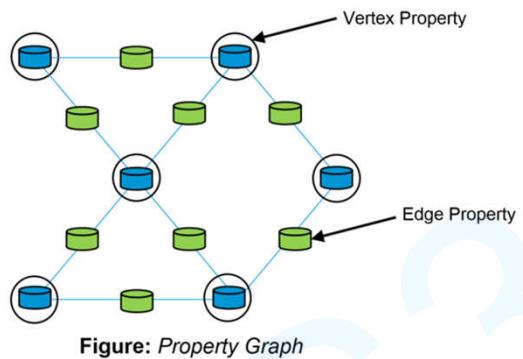
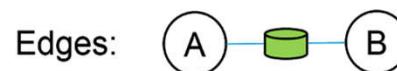
- Google Pregel





# Components

## GraphX - Graph and Examples





# Components

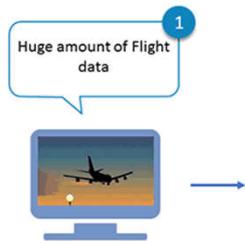
GraphX - Flight Data Analysis using Spark GraphX

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	dOfM	dOfW	carrier	tailNum	flNum	origin_id	origin	dest_id	dest	crsdeptime	deptime	depdelaymins	crsarrtime	arrtime	arrdelaymins	crselapsedetime	dist
2	1	3	AA	N338AA	1	12478	JFK	12892	LAX	900	914	14	1225	1238	13	385	2475
3	2	4	AA	N338AA	1	12478	JFK	12892	LAX	900	857	0	1225	1226	1	385	2475
4	4	6	AA	N327AA	1	12478	JFK	12892	LAX	900	1005	65	1225	1324	59	385	2475
5	5	7	AA	N323AA	1	12478	JFK	12892	LAX	900	1050	110	1225	1415	110	385	2475
6	6	1	AA	N319AA	1	12478	JFK	12892	LAX	900	917	17	1225	1217	0	385	2475
7	7	2	AA	N328AA	1	12478	JFK	12892	LAX	900	910	10	1225	1212	0	385	2475
8	8	3	AA	N323AA	1	12478	JFK	12892	LAX	900	923	23	1225	1215	0	385	2475
9	9	4	AA	N339AA	1	12478	JFK	12892	LAX	900	859	0	1225	1204	0	385	2475
10	10	5	AA	N319AA	1	12478	JFK	12892	LAX	900	929	29	1225	1245	20	385	2475



# Components

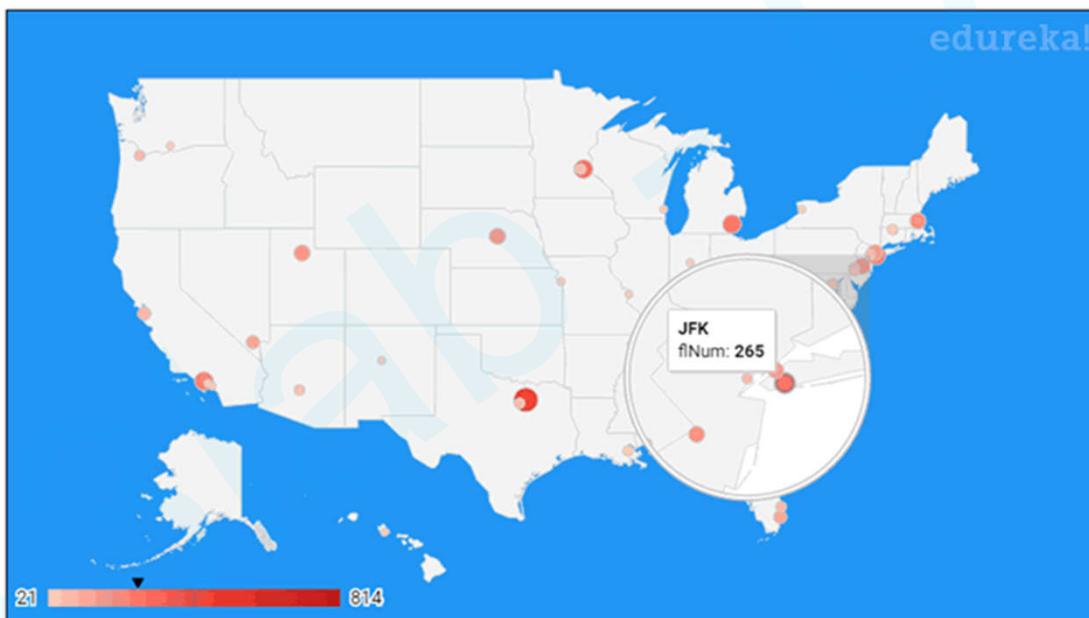
GraphX - Flight Data Analysis using Spark GraphX





# Components

GraphX - Flight Data Analysis using Spark GraphX



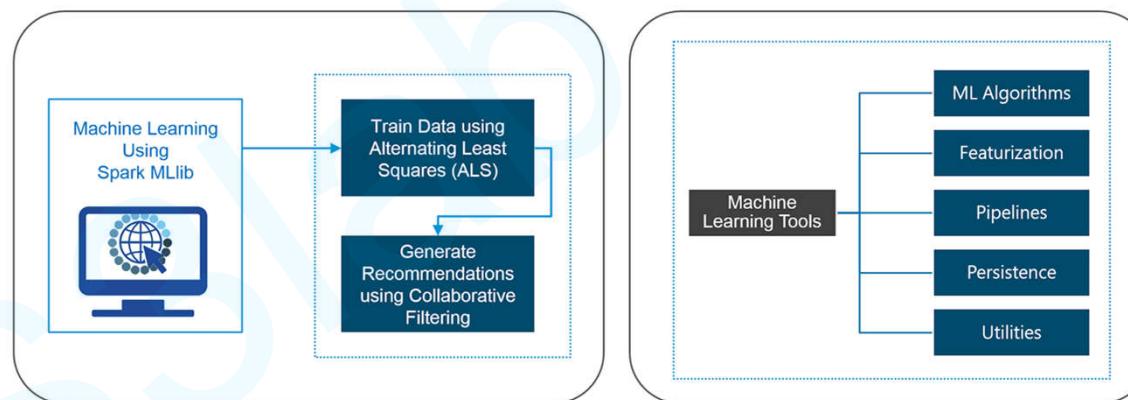
**Figure:** Total Number of Flights from New York



# Components

## MLLib

- stands for Machine Learning Library. Spark MLLib is used to perform machine learning in Apache Spark.



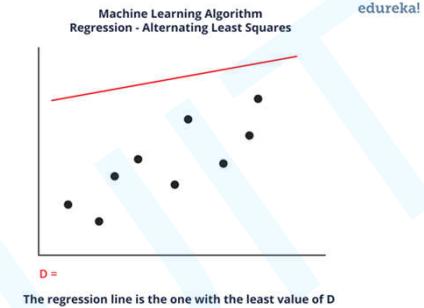
**Figure: Machine Learning Flow Diagram**

**Figure: Machine Learning Tools**

# Components

## MLLib - Algorithms

- **Basic Statistics:** Summary, Correlation, Stratified Sampling, Hypothesis Testing, Random Data Generation.
- Regression
- Classification
- Recommendation System: Collaborative, Content-Based
- Clustering
- Dimensionality Reduction: Feature Selection, Feature Extraction
- Feature Extraction
- Optimization

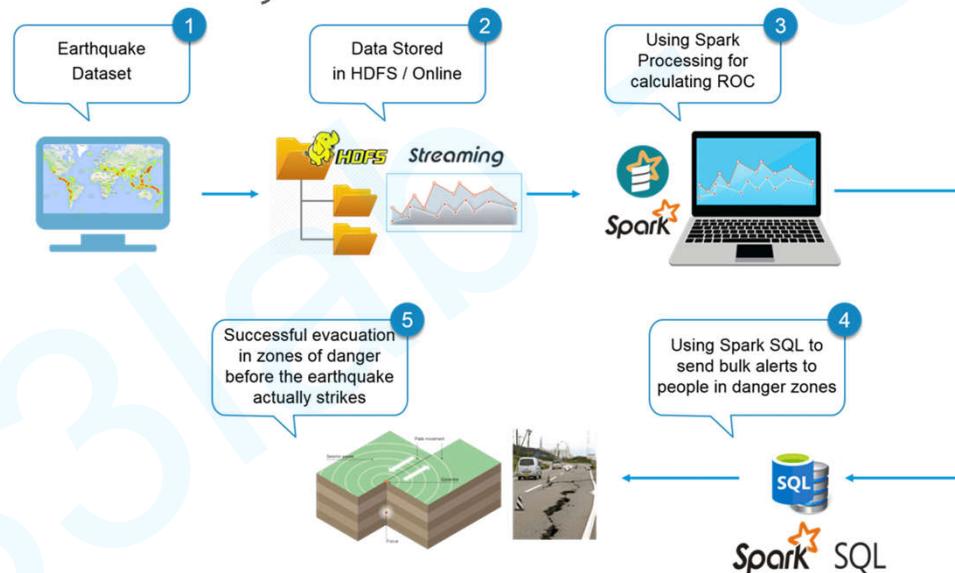




# Components

## MLLib - Use case

- Earthquake Detection System





# Components

MLLib - Use case

- Movie Recommendation System

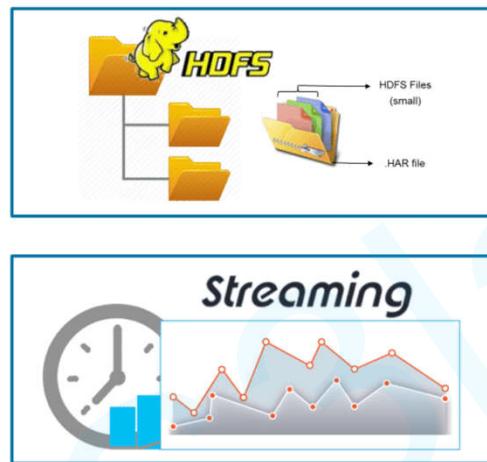




# Components

MLLib - Use case

- Movie Recommendation System



**Figure:** Various File Formats



# Challenges of Distributed computing

- How to divide the input data
- How to assign the divided data to machines in the cluster
- How to check and monitor a machine in the cluster is live and has resources to perform its duty
- How to retry or reassign failed chunks to another machine or worker



# Challenges of Distributed computing

- If the computation involves any aggregation operation like a sum, how to collate results from many workers and compute the aggregation
- Efficient use of memory , cpu and network
- Monitoring the tasks
- Overall job coordination
- Keeping a global time

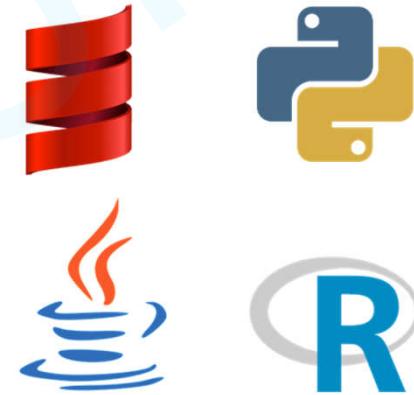


## Usecases

- ETL
- Analytics
- Machine Learning
- Graph processing
- SQL queries on large data sets
- Batch processing
- Stream processing

## Features

- **Multiple languages support** namely Java, R, Scala, Python for building applications. Spark provides high-level APIs in **Java, Scala, Python** and **R**. Spark code can be written in any of these four languages. It provides a **shell** in **Scala** and **Python**. The Scala shell can be accessed through `./bin/spark-shell` and Python shell through `./bin/pyspark` from the installed directory.





## Features

- **Fast Speed** in Data Processing, 10 times faster on Disk and 100 times swifter in Memory (compare to Hadoop). Spark is able to achieve this speed through controlled partitioning. It manages data using partitions that help parallelize distributed data processing with minimal network traffic.
- Spark with abstraction-RDD provides **Fault Tolerance** with ensured Zero Data loss



## Features

- Increase in system efficiency due to **Lazy Evaluation** of transformation in RDD: Apache Spark delays its evaluation till it is absolutely necessary. This is one of the key factors contributing to its speed. For **transformations**, Spark adds them to a DAG (Directed Acyclic Graph) of computation and only when the driver requests some data, does this DAG actually gets executed.





# Features

## Lazy evaluation

As we have seen in the log analytics example, transformations on RDDs are lazily evaluated to optimize disk and memory usage in Spark.

- RDDs are empty when they are created, only the type and ID are determined
- Spark will not begin to execute the job until it sees an action.
- Lazy evaluation is used to reduce the number of passes it has to take over the data by grouping operations together
- In MapReduce, developers have to spend a lot of time thinking about how to group together operations to minimize the number of MR passes
- A task and all its dependencies are largely omitted if the partition it generates is already cached.



## Features

- **In-Memory Processing** resulting in high computation speed and acyclic data-flow
- With **80+ high level operators** it is easy to develop **Dynamic & Parallel applications**
- **Real-time data stream processing** with Spark Streaming



## Features

- Flexible to run **Independently** and can be **integrated with Hadoop** Yarn Cluster Manager. Apache Spark provides smooth compatibility with Hadoop. This is a boon for all the Big Data engineers who started their careers with Hadoop. Spark is a potential replacement for the MapReduce functions of Hadoop, while Spark has the ability to run on top of an existing Hadoop cluster using YARN for resource scheduling.



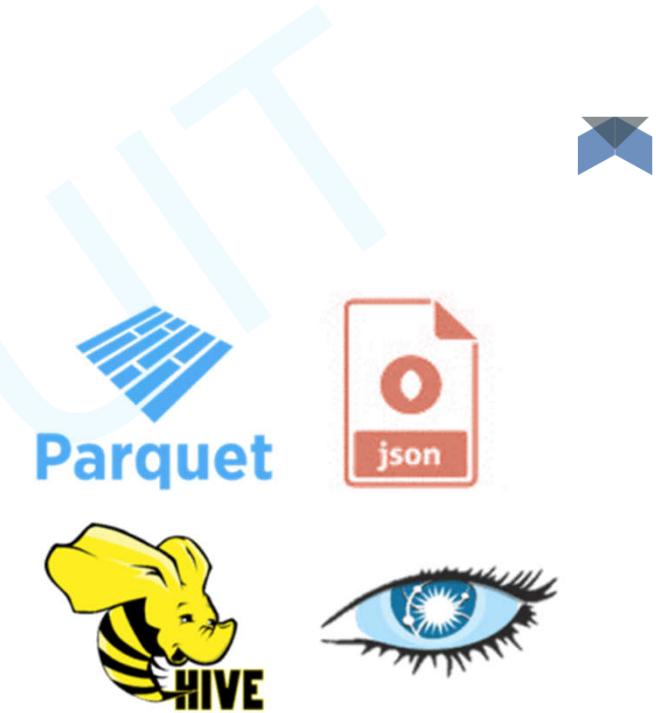


## Features

- **Cost Efficient** for Big data as minimal need of storage and data center
- Futuristic analysis with **built-in tools** for machine learning (MLLib), interactive queries & data streaming
- **Persistence** and **Immutable** in nature with data parallel processing over the cluster
- Graphx simplifies Graph Analytics by collecting algorithm and builders
- **Re-using Code** for batch processing and to run ad-hoc queries
- Progressive and expanding Apache community active for **Quick Assistance**

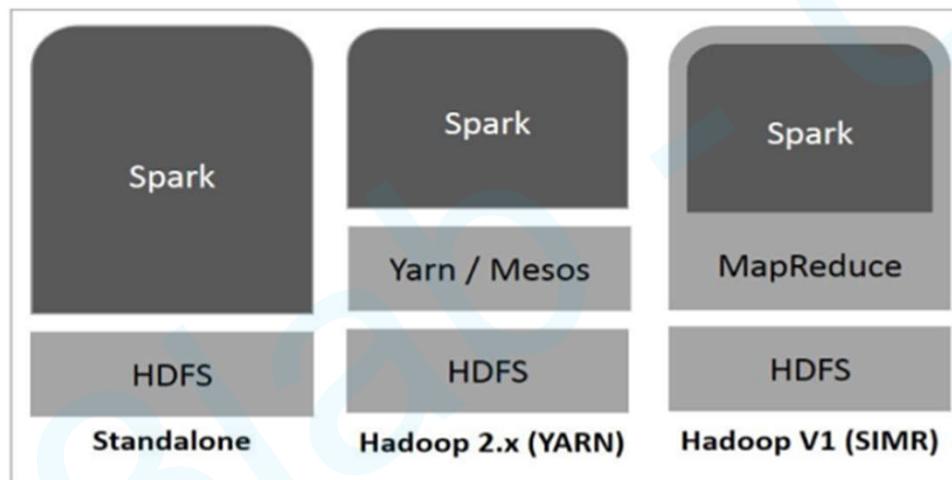
## Features

- Spark supports **multiple data sources** such as Parquet, JSON, Hive and Cassandra apart from the usual formats such as text files, CSV and RDBMS tables. The Data Source API provides a pluggable mechanism for accessing structured data through Spark SQL. Data sources can be more than just simple pipes that convert data and pull it into Spark.





# Spark build on Hadoop





# RDD - Resilient Distributed Dataset

## Definition

- Resilient Distributed Dataset is the fundamental data structure abstraction of Spark
- RDD is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. For instance you can create an RDD of integers and these gets partitioned and divided and assigned to various nodes in the cluster for parallel processing.
- RDDs can contain any type of **Python**, **Java**, or **Scala** objects, including **user-defined** classes.



# RDD - Resilient Distributed Dataset

## Definition

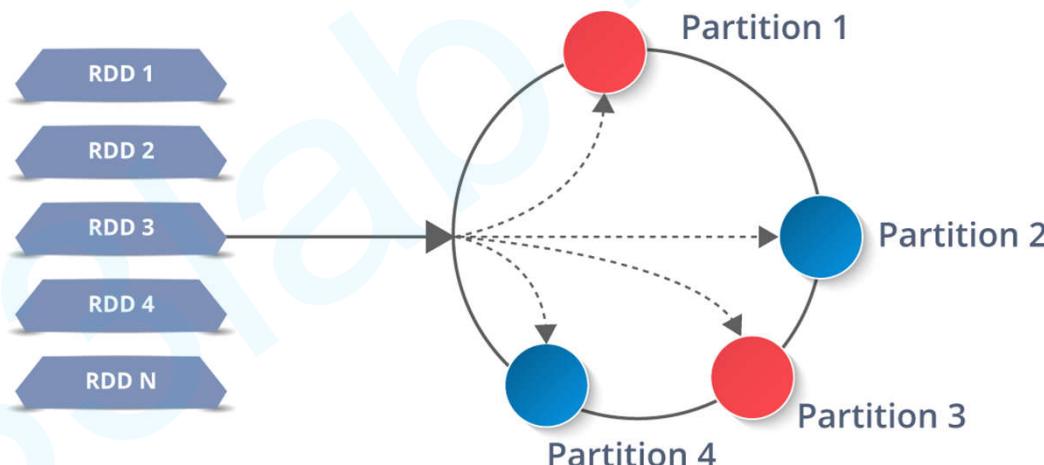
- **Resilient** - They are fault tolerant and able to detect and recompute missing or damaged partitions of an RDD due to node or network failures.
- **Distributed** - Data is partitioned and resides on multiple nodes depending on the cluster size, type and configuration
- **In Memory Data Structure** - Mostly they are in memory so that iterative operations runs faster and performs way better than traditional hadoop programs in executing iterative algorithms



# RDD - Resilient Distributed Dataset

## Definition

- **Dataset** - it can represent any form of data be it reading from a csv file, loading data from a table using rdbms, text file, json , xml.

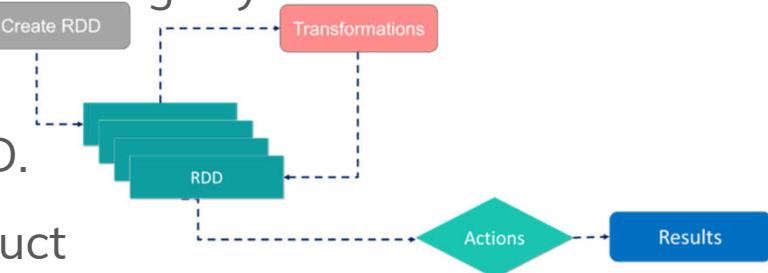




# RDD - Resilient Distributed Dataset

## Workflow

- Create RDDs
  - An existing collection in your **driver program**.
  - Referencing a dataset in an external storage system.
- 2 types of operations
  - **Transformation**: to create new RDD.
  - **Actions**: applied on an RDD to instruct Spark to apply computation and pass the result back to the driver.





# RDD - Resilient Distributed Dataset

## Partition and Parallelism

- **Partition** - is a logical chunk of a large distributed data set. By default. Spark tries to read data into an RDD from the nodes that are close to it.

```
Welcome to
          ____          _ 
         / \ \        / | |
        /   \ \      /  | 
       /     \ \    /   | 
      /       \ \  /    | 
     /         \ \|     | 
    /           \|    | 
   /             \|   | 
  /               \|  | 
 /                 \|_| 
version 2.1.1

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala> sc.parallelize(1 to 100, 5).collect()
```

```
res4: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72,
73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)
```

```
sc.parallelize(1 to 100, 5).collect()
```

5 here, represents the number of partitions the RDD is split into which in turn Runs them parallelly.



# RDD - Resilient Distributed Dataset

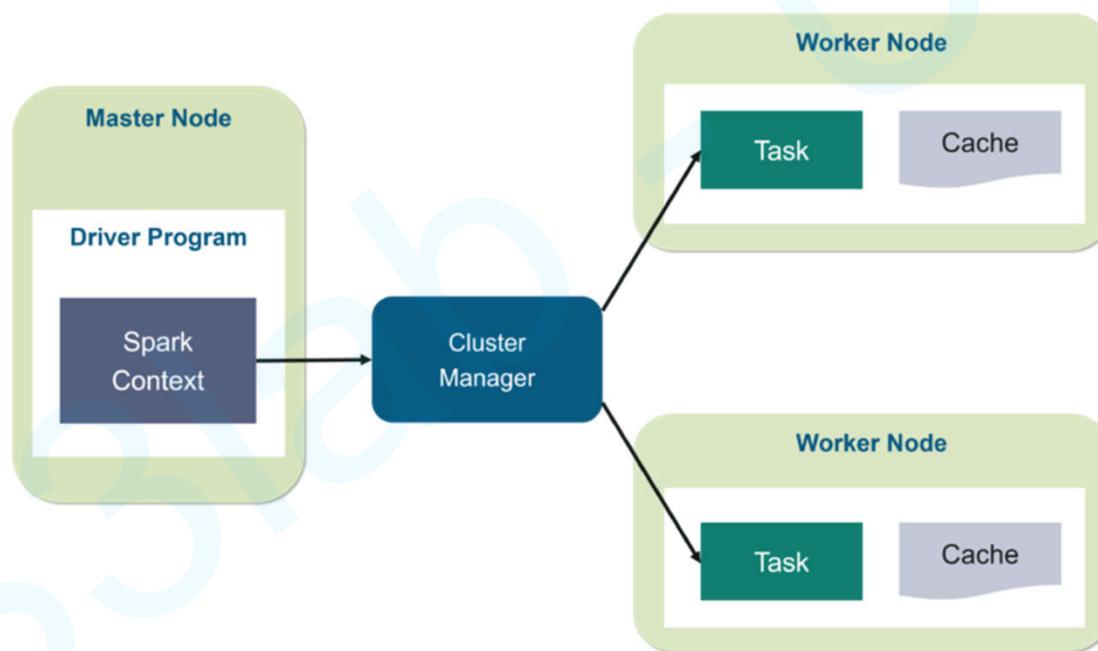
## Partition and Parallelism





# Spark Program

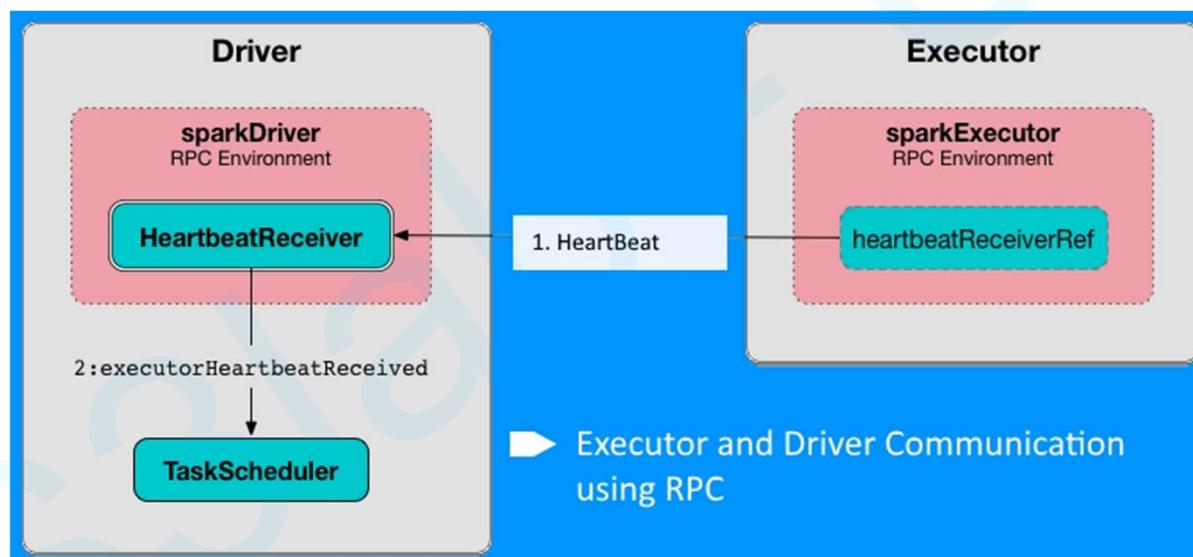
## Architecture





# Spark Program

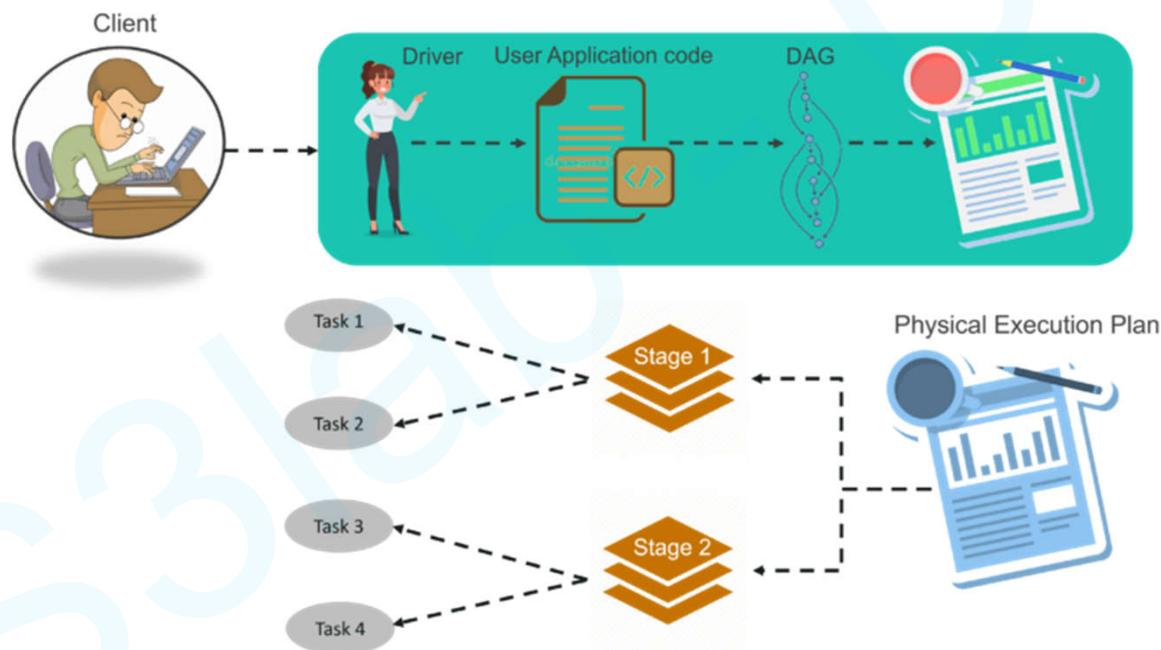
## Architecture





# Spark Program

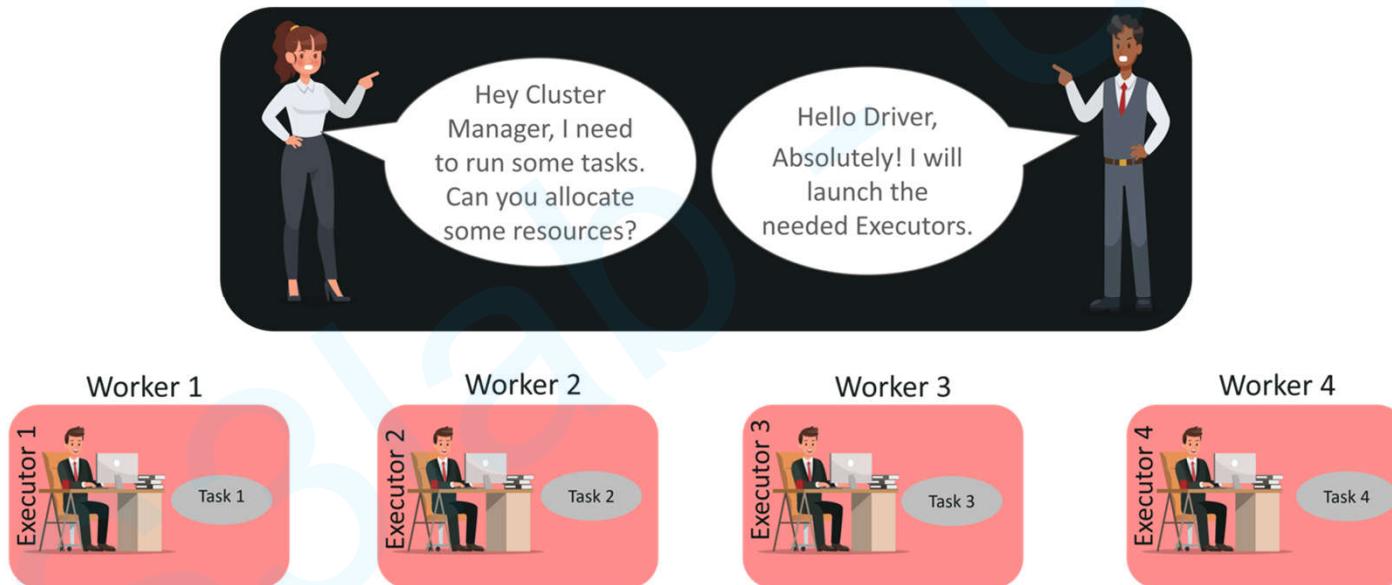
## Architecture





# Spark Program

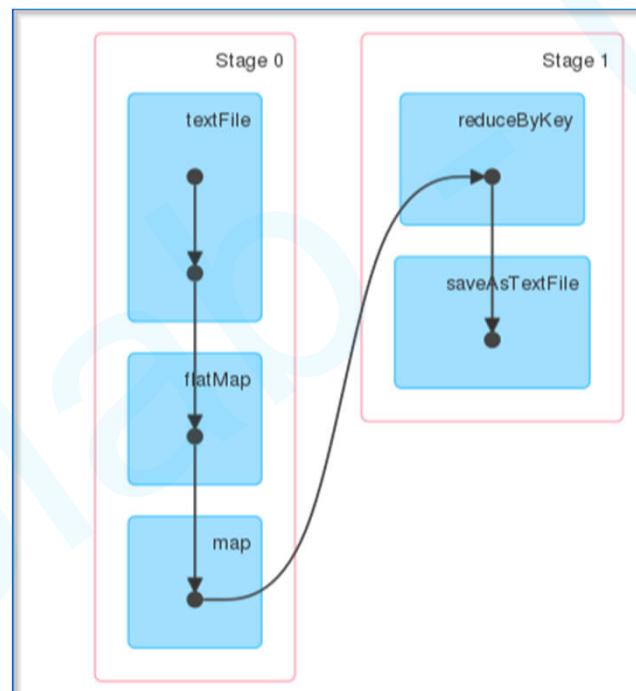
## Architecture





# Spark Program

Directed Acyclic Graph (DAG) Visualization





## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.



# Big Data

(Spark)

Instructor: Trong-Hop Do

May 5<sup>th</sup> 2021

**S<sup>3</sup>Lab**

*Smart Software System Laboratory*



**“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”**

– Chris Lynch, Vertica Systems

# Spark shell

- Open Spark shell
- Command: **spark-shell**

```
[cloudera@quickstart ~]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/avro/avro-tools-1.7.6-cdh5.13.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to

    / \ / \
    \ V \ / \
     \ . / / \
      / \ \ \
        /   \
version 1.6.0

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
20/11/28 00:15:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context available as sc (master = local[*], app id = local-1606551332403).
20/11/28 00:15:35 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
SQL context available as sqlContext.

scala> █
```

## What is SparkContext?

- Spark SparkContext is an entry point to Spark and defined in org.apache.spark package and used to programmatically create Spark RDD, accumulators, and broadcast variables on the cluster.
- Its object **sc** is default variable available in spark-shell and it can be programmatically created using SparkContext class.
- Most of the operations/methods or functions we use in Spark come from SparkContext for example accumulators, broadcast variables, parallelize, and more.

# What is SparkContext?

- Test some methods with the default SparkContext object **sc** in Spark shell

```
scala> sc.applicationId
res9: String = local-1606562891233

scala> sc.appName
res10: String = Spark shell

scala> sc.applicationId
res11: String = local-1606562891233

scala> var map = sc.textFile("/user/cloudera/sample.txt")
map: org.apache.spark.rdd.RDD[String] = /user/cloudera/sample.txt MapPartitionsRDD[8] at textFile at <console>:27
```

- There are many more methods of SparkContext

## Spark RDD

- Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark, It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster.
- Spark RDD can be created in several ways using Scala & Pyspark languages, for example, It can be created by using `sparkContext.parallelize()`, from text file, from another RDD, DataFrame, and Dataset.

# Spark RDD

## Create an RDD through Parallelized Collection

- Spark shell provides SparkContext variable “sc”, use **sc.parallelize()** to create an RDD[Int].

```
scala> val no = Array(1,2,3,4,5,6,7,8,9,10)
no: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
scala> val noData = sc.parallelize(no)
noData: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[9] at parallelize at <console>:29
```

```
scala> noData.foreach(println)
1
2
3
4
5
6
7
8
9
10
```

Note: the printed list might be unordered as foreach runs in parallel across the partitions which creates non-deterministic ordering. The action **collect()** gives you an array of the partitions concatenated in their sorted order. Or you might set the number of partitions to 1.

# Spark RDD

## Create an RDD through Parallelized Collection

- Spark shell provides SparkContext variable “sc”, use **sc.parallelize()** to create an RDD[Int].

```
scala> var pairData = sc.parallelize(Seq(("Java",20000),("Python",100000),("Scala",3000)))
pairData: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[30] at parallelize at <console>:27

scala> pairData.foreach(println)
(Java,20000)
(Python,100000)
(Scala,3000)
```

# Spark RDD

Read single or multiple text files into single RDD?

- Create files and put them to HDFS

FILE NAME	FILE CONTENTS
text01.txt	One,1
text02.txt	Two,2
text03.txt	Three,3
text04.txt	Four,4
invalid.txt	Invalid,I

```
[cloudera@quickstart ~]$ cat > text01.txt  
One,1  
[cloudera@quickstart ~]$ cat > text02.txt  
Two,2  
[cloudera@quickstart ~]$ cat > text03.txt  
Three,3  
[cloudera@quickstart ~]$ cat > text04.txt  
Four,4  
[cloudera@quickstart ~]$ cat > invalid.txt  
Invalid,I
```

```
[cloudera@quickstart ~]$ hdfs dfs -put text01.txt  
[cloudera@quickstart ~]$ hdfs dfs -put text02.txt  
[cloudera@quickstart ~]$ hdfs dfs -put text03.txt  
[cloudera@quickstart ~]$ hdfs dfs -put text04.txt  
[cloudera@quickstart ~]$ hdfs dfs -put invalid.txt
```

# Spark RDD

Read single or multiple text files into single RDD?

- Read single file and create an RDD[String]

```
scala> var textData = sc.textFile("/user/cloudera/text01.txt")
textData: org.apache.spark.rdd.RDD[String] = /user/cloudera/text01.txt MapPartitionsRDD[21] at textFile at <console>:27
scala> textData.foreach(f => {println(f)})
One,1
```

# Spark RDD

Read single or multiple text files into single RDD?

- Read multiple files and create an RDD[String]

```
scala> var textData = sc.textFile("/user/cloudera/text*")
textData: org.apache.spark.rdd.RDD[String] = /user/cloudera/text* MapPartitionsRDD[23] at textFile at <console>:27

scala> textData.foreach(f => {println(f)})
One,1
Two,2
Three,3
Four,4
```

# Spark RDD

Read single or multiple text files into single RDD?

- Read multiple specific files and create an RDD[String]

```
scala> var textData = sc.textFile("/user/cloudera/text01.txt,/user/cloudera/text03.txt")
textData: org.apache.spark.rdd.RDD[String] = /user/cloudera/text01.txt,/user/cloudera/text03.txt MapPartitionsRDD[25] at
textFile at <console>:27

scala> textData.foreach(f => {println(f)})
One,1
Three,3
```

# Spark RDD

## Load CSV File into RDD

- Create .csv files and put them to HDFS

FILE NAME	FILE CONTENTS
text01.csv	Col1,Col2 One,1 Eleven,11
text02.csv	Col1,Col2 Two,2 Twenty One,21

```
[cloudera@quickstart ~]$ cat > text01.csv
Col1,Col2
One,1
Eleven,11
[cloudera@quickstart ~]$ cat > text02.csv
Col1,Col2
Two,2
Twenty One,21
[cloudera@quickstart ~]$ hdfs dfs -put text01.csv
[cloudera@quickstart ~]$ hdfs dfs -put text02.csv
```

# Spark RDD

## Load CSV File into RDD

- `textFile()` method read an entire CSV record as a String and returns `RDD[String]`

```
scala> var textData = sc.textFile("/user/cloudera/text01.txt,/user/cloudera/text01.csv")
textData: org.apache.spark.rdd.RDD[String] = /user/cloudera/text01.txt,/user/cloudera/text01.csv MapPartitionsRDD[27] at
textFile at <console>:27
```

# Spark RDD

## Load CSV File into RDD

- we would need every record in a CSV to split by comma delimiter and store it in RDD as multiple columns, In order to achieve this, we should use map() transformation on RDD where we will convert RDD[String] to RDD[Array[String]] by splitting every record by comma delimiter. map() method returns a new RDD instead of updating existing.

```
scala> var csvData = textData.map(f=>{f.split(",")})
csvData: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[28] at map at <console>:29

scala> csvData.foreach(f=>{println("Column1:"+f(0)+",Column2:"+f(1))})
Column1:One,Column2:1
Column1:Col1,Column2:Col2
Column1:One,Column2:1
Column1:Eleven,Column2:11
```

Each record in the csvData RDD is an **Array[String]**  
f(0), f(1) are the first and second element in the array f

# Spark RDD

## Load CSV File into RDD

- Skip header from csv file
- Command: `rdd.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }`

```
scala> val csvData = textData.map(row => row.split(",")).mapPartitionsWithIndex{ (idx,iter) => if (idx ==0) iter.drop(1) else iter}
csvData: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[7] at mapPartitionsWithIndex at <console>:25

scala> csvData.foreach( row => println("Column 1:" + row(0) + ",Column 2:" + row(1)))
Column 1:two,Column 2:2
Column 1:one,Column 2:1
```

# Spark RDD Operations

- <https://spark.apache.org/docs/latest/api/python/reference/pyspark.html>
- <https://data-flair.training/blogs/spark-rdd-operations-transformations-actions/>

# Spark RDD Operations

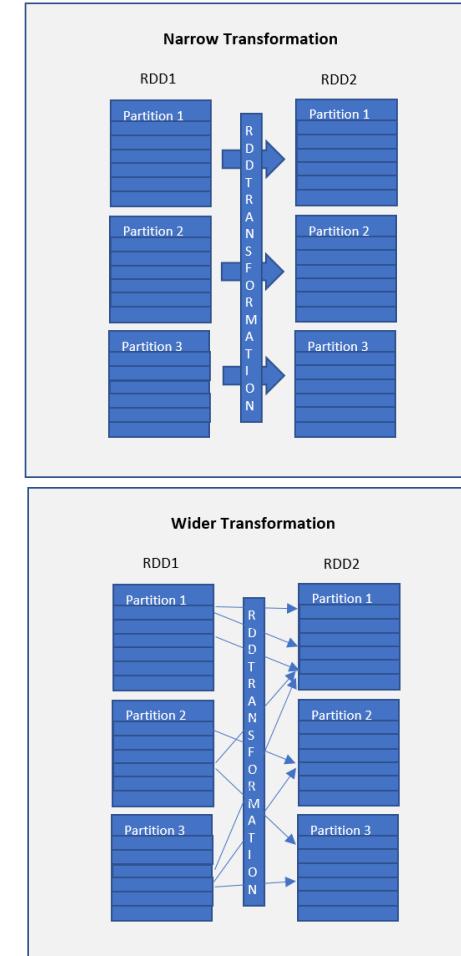
Two type of RDD operations

- Apache Spark RDD supports two types of Operations-
  - Transformations
  - Actions
- **Spark Transformation** is a function that produces new RDD from the existing RDDs. It takes RDD as input and produces one or more RDD as output. Each time it creates new RDD when we apply any transformation.
- **RDD actions** are operations that return the raw values, In other words, any RDD function that returns other than RDD is considered as an action in spark programming.

# Spark RDD Operations

## RDD Transformation

- Two types of RDD Transformation
  - **Narrow transformations** are the result of `map()` and `filter()` functions and these compute data that live on a single partition meaning there will not be any data movement between partitions to execute narrow transformations.
  - **Wider transformations** are the result of `groupByKey()` and `reduceByKey()` functions and these compute data that live on many partitions meaning there will be data movements between partitions to execute wider transformations.



# Spark RDD Operations

## RDD Transformation

<code>cache()</code>	Caches the RDD
<code>filter()</code>	Returns a new RDD after applying filter function on source dataset.
<code>flatMap()</code>	Returns flatten map meaning if you have a dataset with array, it converts each elements in a array as a row. In other words it return 0 or more items in output for each element in dataset.
<code>map()</code>	Applies transformation function on dataset and returns same number of elements in distributed dataset.
<code>mapPartitions()</code>	Similar to map, but executes transformation function on each partition, This gives better performance than map function
<code>mapPartitionsWithIndex()</code>	Similar to map Partitions, but also provides func with an integer value representing the index of the partition.
<code>randomSplit()</code>	Splits the RDD by the weights specified in the argument. For example <code>rdd.randomSplit(0.7,0.3)</code>

<code>union()</code>	Comines elements from source dataset and the argument and returns combined dataset. This is similar to union function in Math set operations.
<code>sample()</code>	Returns the sample dataset.
<code>intersection()</code>	Returns the dataset which contains elements in both source dataset and an argument
<code>distinct()</code>	Returns the dataset by eliminating all duplicated elements.
<code>repartition()</code>	Return a dataset with number of partition specified in the argument. This operation reshuffles the RDD randomlly, It could either return lesser or more partitioned RDD based on the input supplied.
<code>coalesce()</code>	Similar to repartition by operates better when we want to decrease the partitions. Betterment acheives by reshuffling the data from fewer nodes compared with all nodes by repartition.

# Spark RDD Operations

## RDD Action

RDD ACTION METHODS	METHOD DEFINITION
<a href="#">aggregate[U](zeroValue: U)(seqOp: (U, T) ⇒ U, combOp: (U, U) ⇒ U)(implicit arg0: ClassTag[U]): U</a>	Aggregate the elements of each partition, and then the results for all the partitions.
<a href="#">collect():Array[T]</a>	Return the complete dataset as an Array.
<a href="#">count():Long</a>	Return the count of elements in the dataset.
<a href="#">countApprox(timeout: Long, confidence: Double = 0.95): PartialResult[BoundedDouble]</a>	Return approximate count of elements in the dataset, this method returns incomplete when execution time meets timeout.
<a href="#">countApproxDistinct(relativeSD: Double = 0.05): Long</a>	Return an approximate number of distinct elements in the dataset.
<a href="#">countByValue(): Map[T, Long]</a>	Return Map[T,Long] key representing each unique value in dataset and value represent count each value present.
<a href="#">countByValueApprox(timeout: Long, confidence: Double = 0.95)(implicit ord: Ordering[T] = null): PartialResult[Map[T, BoundedDouble]]</a>	Same as countByValue() but returns approximate result.
<a href="#">first():T</a>	Return the first element in the dataset.
<a href="#">fold(zeroValue: T)(op: (T, T) ⇒ T): T</a>	Aggregate the elements of each partition, and then the results for all the partitions.
<a href="#">foreach(f: (T) ⇒ Unit): Unit</a>	Iterates all elements in the dataset by applying function f to all elements.
<a href="#">foreachPartition(f: (Iterator[T]) ⇒ Unit): Unit</a>	Similar to foreach, but applies function f for each partition.

<a href="#">reduce(f: (T, T) ⇒ T): T</a>	Reduces the elements of the dataset using the specified binary operator.
<a href="#">saveAsObjectFile(path: String): Unit</a>	Saves RDD as a serialized object's to the storage system.
<a href="#">saveAsTextFile(path: String, codec: Class[_ &lt;: CompressionCodec]): Unit</a>	Saves RDD as a compressed text file.
<a href="#">saveAsTextFile(path: String): Unit</a>	Saves RDD as a text file.
<a href="#">take(num: Int): Array[T]</a>	Return the first num elements of the dataset.
<a href="#">takeOrdered(num: Int)(implicit ord: Ordering[T]): Array[T]</a>	Return the first num (smallest) elements from the dataset and this is the opposite of the take() action. Note: Use this method only when the resulting array is small, as all the data is loaded into the driver's memory.
<a href="#">takeSample(withReplacement: Boolean, num: Int, seed: Long = Utils.random.nextLong()): Array[T]</a>	Return the subset of the dataset in an Array. Note: Use this method only when the resulting array is small, as all the data is loaded into the driver's memory.
<a href="#">toLocalIterator(): Iterator[T]</a>	Return the complete dataset as an Iterator. Note: Use this method only when the resulting array is small, as all the data is loaded into the driver's memory.
<a href="#">top(num: Int)(implicit ord: Ordering[T]): Array[T]</a>	Note: Use this method only when the resulting array is small, as all the data is loaded into the driver's memory.
<a href="#">treeAggregate</a>	Aggregates the elements of this RDD in a multi-level tree pattern.
<a href="#">treeReduce</a>	Reduces the elements of this RDD in a multi-level tree pattern.

# Spark RDD Operations

## RDD Transformation – Word count example

- Create a .txt file and put it to HDFS

```
[cloudera@quickstart ~]$ cat > test.txt
This is a test file for testing RDD transformation first line
This is the second line of the same file
[cloudera@quickstart ~]$ hdfs dfs -put test.txt
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Read the file and create an RDD[String]

```
scala> var testfile = sc.textFile("/user/cloudera/test.txt")
testfile: org.apache.spark.rdd.RDD[String] = /user/cloudera/test.txt MapPartitionsRDD[3] at textFile at <console>:27
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Apply **flatMap()** Transformation
- **flatMap()** transformation flattens the RDD after applying the function and returns a new RDD. On the below example, first, it splits each record by space in an RDD and finally flattens it. Resulting RDD consists of a single word on each record.

```
scala> var testfile2 = testfile.flatMap(f=>f.split(" "))  
testfile2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at flatMap at <console>:29
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Apply **map()** Transformation
- **map()** transformation is used to apply any complex operations like adding a column, updating a column etc, the output of map transformations would always have the same number of records as input.
- In this word count example, we are adding a new column with value 1 for each word, the result of the RDD is **PairRDD** which contains **key-value pairs**, word of type String as Key and 1 of type Int as value.

```
scala> var testfile3 = testfile2.map(m=>(m,1))
testfile3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[5] at map at <console>:31
```

Each record in testfile3 RDD is a **tuple [String,Int]** (not an array)

# Spark RDD Operations

## RDD Transformation – Word count example

- Let's print out the content of these RDDs (each record is printed in a single line)

```
scala> testfile.foreach(println)
This is a test file for testing RDD transformation first line
This is the second line of the same file
```

```
scala> testfile2.foreach(println)
This
is
a
test
file
for
testing
RDD
transformation
first
line
This
is
the
second
line
of
the
same
file
```

```
scala> testfile3.foreach(println)
(Key, Value)
((This,1), (is,1))
((a,1), (test,1))
((file,1), (for,1))
((for,1), (testing,1))
((testing,1), (RDD,1))
((RDD,1), (transformation,1))
((transformation,1), (first,1))
((first,1), (line,1))
((line,1), (This,1))
((This,1), (is,1))
((is,1), (the,1))
((the,1), (second,1))
((second,1), (line,1))
((line,1), (of,1))
((of,1), (the,1))
((the,1), (same,1))
((same,1), (file,1))
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Apply **reduceByKey()** Transformation
- **reduceByKey()** merges the values for each key with the function specified. In our example, it reduces the word string by applying the sum function on **value**. The result of our RDD contains unique words and their count.

```
scala> var count = testfile3.reduceByKey(_ + _)
count: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[6] at reduceByKey at <console>:33
```

- Note: **reduceByKey()** is a transformation, so **count** is a RDD

# Spark RDD Operations

## RDD Transformation – Word count example

- Let's print out the result

```
scala> count.foreach(println)
(is,2)
(second,1)
(same,1)
(a,1)
(line,2)
(This,2)
(first,1)
(testing,1)
(of,1)
(transformation,1)
(for,1)
(RDD,1)
(file,2)
(the,2)
(test,1)
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Another way to use **reduceByKey**

```
scala> testfile3.reduceByKey( (value1,value2) => value1+value2).foreach(println)
(a,1)
(is,2)
(second,1)
(same,1)
(line,2)
(This,2)
(first,1)
(testing,1)
(RDD,1)
(file,2)
(test,1)
(of,1)
(transformation,1)
(for,1)
(the,2)
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Try **sortByKey()** Transformation
- **sortByKey()** transformation is used to sort RDD elements on **key**.
- In this example, we want to sort RDD elements on the number of each word. Therefore, we need convert RDD[(String,Int)] to RDD[(Int,String)] using map transformation and then apply sortByKey which ideally does sort on an integer key value.

```
scala> var count2 = count.map(a=>(a._2,a._1))
count2: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[13] at map at <console>:35
```

Notice how each element in a tuple is accessed. If a is an array, the second and first element are accessed by a(1) and a(0). In this case, a is a tuple, the second and first element are accessed by a.\_2 and a.\_1

# Spark RDD Operations

## RDD Transformation – Word count example

- Let's print out the result (and see that the count number of each word now become the **key**)

```
scala> count.foreach(println)
(is,2)
(second,1)
(same,1)
(a,1)
(line,2)
(This,2)
(first,1)
(testing,1)
(of,1)
(transformation,1)
(for,1)
(RDD,1)
(file,2)
(the,2)
(test,1)
```



```
scala> count2.foreach(println)
(2,is)
(1,second)
(1,same)
(1,a)
(2,line)
(2,This)
(1,first)
(1,testing)
(1,of)
(1,transformation)
(1,for)
(1,RDD)
(2,file)
(2,the)
(1,test)
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Apply **sortByKey()** transformation

```
scala> var count3 = count2.sortByKey()
count3: org.apache.spark.rdd.RDD[(Int, String)] = ShuffledRDD[14] at sortByKey at <console>:37
```

```
scala> count3.foreach(println)
(1,second)
(1,same)
(1,a)
(1,first)
(1,testing)
(1,of)
(1,transformation)
(1,for)
(1,RDD)
(1,test)
(2,is)
(2,line)
(2,This)
(2,file)
(2,the)
```

If there are more than 1 partitions, records in each partitions are sorted, but the sorted results of multiple partitions might be mixed up.

To avoid this, use **sortByKey(numPartitions=1)**

# Spark RDD Operations

## RDD Transformation – Word count example

- You can also apply **map** transformation to switch the word back to the key

```
scala> var count4 = count3.map(a=>(a._2,a._1))
count4: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[15] at map at <console>:39

scala> count4.foreach(println)
(second,1)
(same,1)
(a,1)
(first,1)
(testing,1)
(of,1)
(transformation,1)
(for,1)
(RDD,1)
(test,1)
(is,2)
(line,2)
(This,2)
(file,2)
(the,2)
```

# Spark RDD Operations

## RDD Transformation – Word count example

- You can do all these steps in one line of code

```
scala> var count5 = count.map(a=>(a._2,a._1)).sortByKey().map(a=>(a._2,a._1))
count5: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[18] at map at <console>:35

scala> count5.foreach(println)
(second,1)
(same,1)
(a,1)
(first,1)
(testing,1)
(of,1)
(transformation,1)
(for,1)
(RDD,1)
(test,1)
(is,2)
(line,2)
(This,2)
(file,2)
(the,2)
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Or you can use `sortBy`

```
scala> count.sortBy(_.value, numPartitions=1).foreach(println)
(second,1)
(same,1)
(first,1)
(testing,1)
(RDD,1)
(test,1)
(a,1)
(of,1)
(transformation,1)
(for,1)
(is,2)
(line,2)
(This,2)
(file,2)
(the,2)
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Apply **filter()** Transformation
- filter() transformation is used to filter the records in an RDD. In this example we are filtering all words starts with “t”.

```
scala> var count6 = count.filter(a=>a._1.startsWith("t"))
count6: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[21] at filter at <console>:35

scala> count6.foreach(println)
(testing,1)
(transformation,1)
(the,2)
(test,1)
```

# Spark RDD Operations

## RDD Transformation – Word count example

- Merge two RDD

```
scala> var count7 = count.filter(a=>a._1.startsWith("T"))
count7: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[22] at filter at <console>:35

scala> var merged = count6.union(count7)
merged: org.apache.spark.rdd.RDD[(String, Int)] = PartitionerAwareUnionRDD[23] at union at <console>:39

scala> merged.foreach(println)
(testing,1)
(transformation,1)
(the,2)
(test,1)
(This,2)
```

# Spark RDD Operations

## RDD Action – Easiest Wordcount using `countByValue`

```
scala> testfile2.foreach(println)
This
is
the
second
line
of
the
same
file
This
is
a
test
file
for
testing
```

```
scala> testfile2.countByValue.foreach(println)
(for,1)
(test,1)
(is,2)
(This,2)
(line,2)
(a,1)
(transformation,1)
(testing,1)
(second,1)
(same,1)
(RDD,1)
(first,1)
(file,2)
(of,1)
(the,2)
```

# Spark RDD Operations

## RDD Action

- Let's create some RDD

```
scala> var inputrdd = sc.parallelize(List(("Z",1),("A",2),("B",30),("C",40),("B",30),("B",60)))
inputrdd: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[24] at parallelize at <console>:27

scala> var listrdd = sc.parallelize(List(1,2,3,4,5,3,2))
listrdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[25] at parallelize at <console>:27
```

# Spark RDD Operations

## RDD Action

- **reduce()** - Reduces the elements of the dataset using the specified binary operator.

```
scala> println("reduce : "+listrdd.reduce(_ + _))
reduce : 20

scala> println("reduce alternate : "+listrdd.reduce((x,y) => x + y))
reduce alternate : 20

scala> println("reduce : "+inputrdd.reduce((x,y) => ("Total",x._2 + y._2)))
reduce : (Total,163)
```

# Spark RDD Operations

## RDD Action

- **collect()** -Return the complete dataset as an **Array**.

```
scala> var data = listrdd.collect()  
data: Array[Int] = Array(1, 2, 3, 4, 5, 3, 2)
```



# Spark RDD Operations

## RDD Action

- **count()** – Return the count of elements in the dataset.
- **countApprox()** – Return approximate count of elements in the dataset, this method returns incomplete when execution time meets timeout.
- **countApproxDistinct()** – Return an approximate number of distinct elements in the dataset.

```
scala> println("Count : "+listrdd.count)
Count : 7

scala> println("CountApprox : "+listrdd.countApprox(1200))
CountApprox : (final: [7.000, 7.000])

scala> println("CountApproxDistinct : "+listrdd.countApproxDistinct())
CountApproxDistinct : 5

scala> println("CountApproxDistinct : "+inputrdd.countApproxDistinct())
CountApproxDistinct : 5
```

# Spark RDD Operations

## RDD Action

- `first()` – Return the first element in the dataset.
- `top()` – Return top n elements from the dataset.
- `min()` – Return the minimum value from the dataset.
- `max()` – Return the maximum value from the dataset.
- `take()` – Return the first num elements of the dataset.
- `takeOrdered()` – Return the first num (smallest) elements from the dataset and this is the opposite of the `take()` action.
- `takeSample()` – Return the subset of the dataset in an Array.

# Spark Pair RDD Functions

## Pair RDD Transformation

Spark defines **PairRDDFunctions** class with several functions to work with Pair RDD or RDD key-value pair

PAIR RDD FUNCTIONS	FUNCTION DESCRIPTION
aggregateByKey	Aggregate the values of each key in a data set. This function can return a different result type than the values in input RDD.
combineByKey	Combines the elements for each key.
combineByKeyWithClassTag	Combines the elements for each key.
flatMapValues	It's flatten the values of each key with out changing key values and keeps the original RDD partition.
foldByKey	Merges the values of each key.
groupByKey	Returns the grouped RDD by grouping the values of each key.
mapValues	It applied a map function for each value in a pair RDD with out changing keys.
reduceByKey	Returns a merged RDD by merging the values of each key.
reduceByKeyLocally	Returns a merged RDD by merging the values of each key and final result will be sent to the master.

sampleByKey	Returns the subset of the RDD.
subtractByKey	Return an RDD with the pairs from this whose keys are not in other.
keys	Returns all keys of this RDD as a RDD[T].
values	Returns an RDD with just values.
partitionBy	Returns a new RDD after applying specified partitioner.
fullOuterJoin	Return RDD after applying fullOuterJoin on current and parameter RDD
join	Return RDD after applying join on current and parameter RDD
leftOuterJoin	Return RDD after applying leftOuterJoin on current and parameter RDD
rightOuterJoin	Return RDD after applying rightOuterJoin on current and parameter RDD

# Spark Pair RDD Functions

## Pair RDD Action

Spark defines **PairRDDFunctions** class with several functions to work with Pair RDD or RDD key-value pair

PAIR RDD ACTION FUNCTIONS	FUNCTION DESCRIPTION
collectAsMap	Returns the pair RDD as a Map to the Spark Master.
countByKey	Returns the count of each key elements. This returns the final result to local Map which is your driver.
countByKeyApprox	Same as countByKey but returns the partial result. This takes a timeout as parameter to specify how long this function to run before returning.
lookup	Returns a list of values from RDD for a given input key.
reduceByKeyLocally	Returns a merged RDD by merging the values of each key and final result will be sent to the master.

saveAsHadoopDataset	Saves RDD to any hadoop supported file system (HDFS, S3, ElasticSearch, e.t.c), It uses Hadoop JobConf object to save.
saveAsHadoopFile	Saves RDD to any hadoop supported file system (HDFS, S3, ElasticSearch, e.t.c), It uses Hadoop OutputFormat class to save.
saveAsNewAPIHadoopDataset	Saves RDD to any hadoop supported file system (HDFS, S3, ElasticSearch, e.t.c) with new Hadoop API, It uses Hadoop Configuration object to save.
saveAsNewAPIHadoopFile	Saves RDD to any hadoop supported file system (HDFS, S3, ElasticSearch, e.t.c), It uses new Hadoop API OutputFormat class to save.

# Spark Pair RDD Functions

## Wordcount using `reduceByKey`

```
[cloudera@quickstart ~]$ cat > sample.txt  
This is a sample file to demonstrate a sample wordcount programe using Spark  
[cloudera@quickstart ~]$ hdfs dfs -put sample.txt
```

```
scala> var map = sc.textFile("/user/cloudera/sample.txt").flatMap(line => line.split(" ")).map(word => (word,1));  
map: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[12] at map at <console>:27  
  
scala> var counts = map.reduceByKey(_+_);  
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[13] at reduceByKey at <console>:29
```

# Spark Pair RDD Functions

Wordcount using `reduceByKey`

- Save the output in a text file

```
scala> counts.saveAsTextFile("/user/cloudera/outputWCspark");
```

# Spark Pair RDD Functions

Wordcount using `reduceByKey`

- Check the result

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 7 items
-rw-r--r--  1 cloudera cloudera      217 2020-10-19 09:33 employeeDetails.txt

[cloudera@quickstart ~]$ hdfs dfs -ls outputWCspark
Found 2 items
-rw-r--r--  1 cloudera cloudera       0 2020-11-28 00:58 outputWCspark/_SUCCESS
-rw-r--r--  1 cloudera cloudera    112 2020-11-28 00:58 outputWCspark/part-00000
[cloudera@quickstart ~]$ hdfs dfs -cat outputWCspark/part*
(Spark,1)
(is,1)
(wordcount,1)
(demonstrate,1)
(a,2)
(to,1)
(This,1)
(using,1)
(programme,1)
(file,1)
(sample,2)
[cloudera@quickstart ~]$ █
```

# Spark Pair RDD Functions

Another usage of `reduceByKey`

```
scala> val testRDD = sc.parallelize(List(("key1",(1,10)),("key2", (1,15)), ("key2", (2,20)), ("key3", (3,30))))  
testRDD: org.apache.spark.rdd.RDD[(String, (Int, Int))] = ParallelCollectionRDD[654] at parallelize at <console>:24
```

```
scala> testRDD.foreach(println)  
(key1,(1,10))  
(key3,(3,30))  
(key2,(1,15))  
(key2,(2,20))
```

```
scala> testRDD.reduceByKey( (row1,row2) => (row1._1+row2._1, row1._2+row2._2)).foreach(println)  
(key3,(3,30))  
(key1,(1,10))  
(key2,(3,35))
```

```
scala> testRDD.reduceByKey( (row1,row2) => (row1._1*row2._2, row1._2*row2._1)).foreach(println)  
(key3,(3,30))  
(key1,(1,10))  
(key2,(20,30))
```

# Spark Pair RDD Functions

Wordcount using **countByKey** action

```
scala> val countNonRDD = testfile3.countByKey()
countNonRDD: scala.collection.Map[String,Long] = Map(for -> 1, test -> 1, is -> 2, This -> 2, line -> 2, a -> 1,
transformation -> 1, testing -> 1, second -> 1, same -> 1, RDD -> 1, first -> 1, file -> 2, of -> 1, the -> 2)

scala> countNonRDD.foreach(println)
(for,1)
(test,1)
(is,2)
(This,2)
(line,2)
(a,1)
(transformation,1)
(testing,1)
(second,1)
(same,1)
(RDD,1)
(first,1)
(file,2)
(of,1)
(the,2)
```

# Spark Pair RDD Functions

## Join two RDDs

- Create two RDDs from text files

```
scala> rdd1.foreach(println)
Cats, 6
Dogs, 5
```

```
scala> rdd2.foreach(println)
Dogs, Spain
Dogs, Italy
Cats, Italy
```

 text1.txt - Notepad	 text2.txt - Notepad							
File	Edit	Format	View	File	Edit	Format	View	Help
Dogs, 5				Dogs, Italy				
Cats, 6				Cats, Italy				
				Dogs, Spain				

# Spark Pair RDD Functions

Join two RDDs

- Check the result

```
scala> val data1 = rdd1.map(line => line.split(",").map(word => word.trim))
data1: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[678] at map at <console>:25
```

```
scala> val data2 = rdd2.map(line => line.split(",").map(word => word.trim))
data2: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[679] at map at <console>:25
```

```
scala> data1.map(line => (line(0),line(1))).join(data2.map(line => (line(0),line(1)))).foreach(println)
(Dogs,(5,Italy))
(Dogs,(5,Spain))
(Cats,(6,Italy))
```

If you have duplicates key in any RDD, join will result in cartesian product.

# Spark Pair RDD Functions

Join two RDDs

- Use **cogroup** or **groupByKey** to get unique keys in the joining result

```
scala> data1.map(line => (line(0),line(1))).cogroup(data2.map(line => (line(0),line(1)))).  
map(row => (row._1,List.concat(row._2._1.toList,row._2._2.toList))).foreach(println)  
(Dogs,List(5, Italy, Spain))  
(Cats,List(6, Italy))
```

```
scala> data1.map(line => (line(0),line(1))).groupByKey().join(data2.map(line => (line(0),line(1))).  
groupByKey()).map(row => (row._1,List.concat(row._2._1.toList,row._2._2.toList))).foreach(println)  
(Dogs,List(5, Italy, Spain))  
(Cats,List(6, Italy))
```

# Spark shell Web UI

Accessing the Web UI of Spark

- Open <http://quickstart.cloudera:4040> in web browser

The screenshot shows the Spark shell application UI at <http://quickstart.cloudera:4040/jobs/>. The top navigation bar includes links for HBase, Impala, Spark, Solr, Oozie, Cloudera Manager, and Getting Started. The main content area displays the "Spark Jobs" section with the following information:

- Total Uptime: 9.6 min
- Scheduling Mode: FIFO
- Completed Jobs: 1

A link to "Event Timeline" is also present. Below this, the "Completed Jobs (1)" section lists one job:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	saveAsTextFile at <console>:32	2020/11/28 03:32:47	0.9 s	2/2	2/2

# Spark shell Web UI

## Explore Spark shell Web UI

- Click DAG Visualization

Details for Job 0

Status: SUCCEEDED  
Completed Stages: 2

► Event Timeline  
► DAG Visualization

Completed Stages (2)

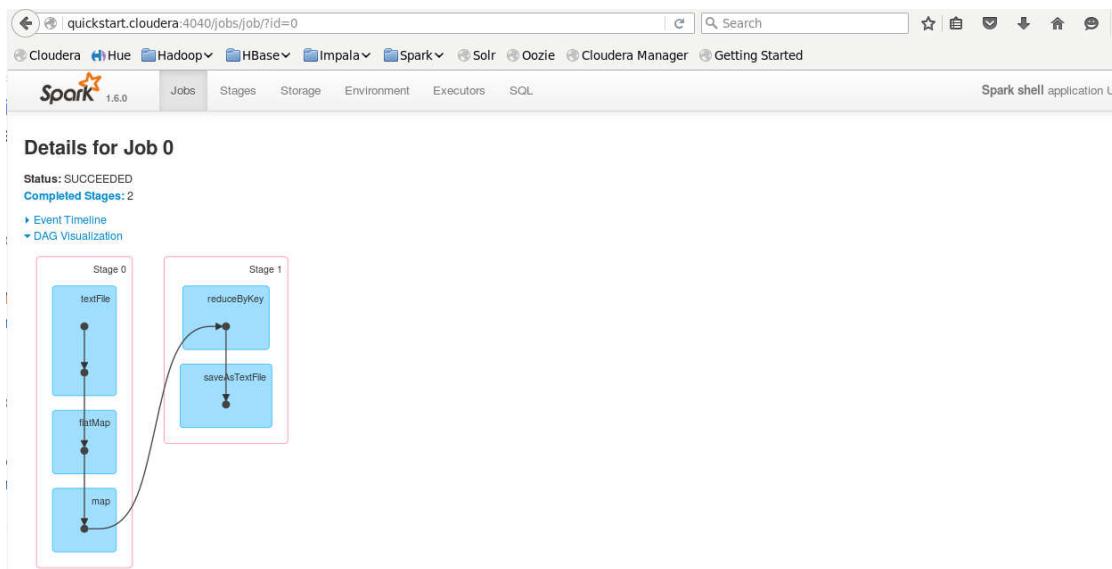
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	saveAsTextFile at <console>:32	+details 2020/11/28 03:32:47	0.6 s	1/1		112.0 B	218.0 B	
0	map at <console>:27	+details 2020/11/28 03:32:47	0.1 s	1/1	77.0 B			218.0 B

Number of partitions

# Spark shell Web UI

Explore Spark shell Web UI

- View the Direct Acyclic Graph (DAG) of the completed job



# Spark shell Web UI

## Partition and parallelism in RDDs

Now, let's understand about partitions and parallelism in RDDs.

- A **partition** is a *logical chunk* of a *large distributed data set*.
- By default, Spark tries to *read data into an RDD* from the *nodes* that are *close to it*.

# Spark shell Web UI

## Partition and parallelism in RDDs

- Execute a parallel task in the shell

```
scala> sc.parallelize(1 to 100, 5).collect()
res1: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)
```

# Spark shell Web UI

Partition and parallelism in RDDs

- Open Spark shell Web UI

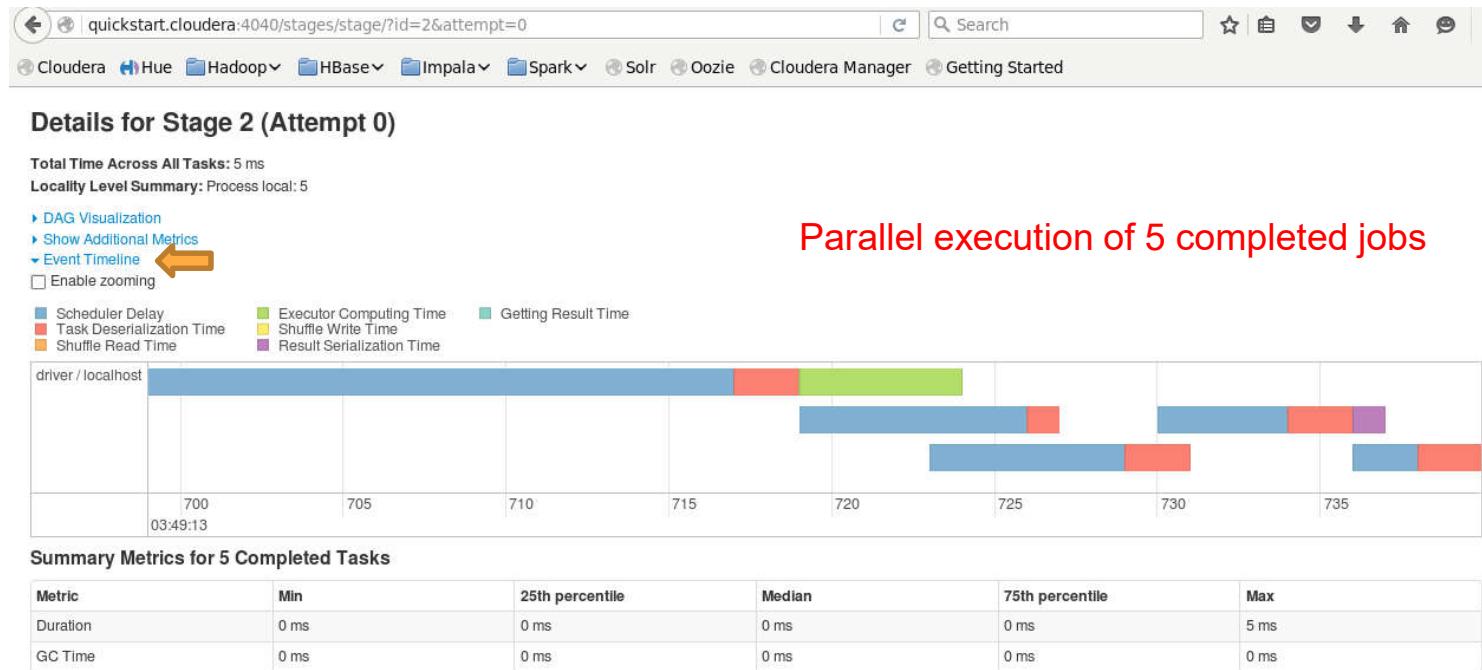
The screenshot shows the Spark shell application UI for a job with ID 1. The top navigation bar includes links for Cloudera, Hue, Hadoop, HBase, Impala, Spark, Solr, Oozie, Cloudera Manager, and Getting Started. The main content area is titled "Details for Job 1". It displays the status as "SUCCEEDED" and "Completed Stages: 1". Below this, there are two collapsed sections: "Event Timeline" and "DAG Visualization". The "DAG Visualization" section shows a single stage labeled "Stage 2" with one task labeled "parallelize". A red box highlights this task. Below the DAG, a table titled "Completed Stages (1)" lists the completed stage details. An orange arrow points from the text "Totally 5 partitions are done" to the progress bar in the table row for Stage 2.

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	collect at <console>:28	+details 2020/11/28 03:49:13	41 ms	5/5				

Totally 5 partitions are done

# Spark shell Web UI

## Partition and parallelism in RDDs



## Q & A



## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.

```
!pip install pyspark==3.0.1

import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

transRDD = spark.sparkContext.textFile("trans.txt")
custRDD = spark.sparkContext.textFile("cust.txt")

#cau 1: Show IDs and number of transactions of each customer
transRDD.map(lambda line: line.split(',')).map(lambda array: (array[2],1)).reduceByKey(lambda a,b : int(a)+int(b)).collect()
#transRDD.map(lambda x : x.split(",")).map(lambda x : (x[2],1)).collect()

[('4000004', 5),
 ('4000007', 6),
 ('4000008', 10),
 ('4000001', 8),
 ('4000002', 6),
 ('4000003', 3),
 ('4000005', 5),
 ('4000006', 5),
 ('4000009', 6),
 ('4000010', 6)]

#cau 2: Show IDs and number of transactions of each customer, sorted by customer ID
transRDD.map(lambda line: line.split(',')).map(lambda array: (array[2],1)).reduceByKey(lambda a,b : int(a)+int(b)).sortBy(lambda x: x[1]).collect()

[('4000003', 3),
 ('4000004', 5),
 ('4000005', 5),
 ('4000006', 5),
 ('4000007', 6),
 ('4000002', 6),
 ('4000009', 6),
 ('4000010', 6),
 ('4000001', 8),
 ('4000008', 10)]

#cau 3: Show IDs and total cost of transactions of each customer, sorted by total cost
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],x[3])).reduceByKey(lambda x1,x2:float(x1)+float(x2) ).sortBy(lambda x: x[1]).collect()

[('4000005', 325.15),
 ('4000004', 337.06),
 ('4000010', 447.0900000000003),
 ('4000009', 457.83),
 ('4000003', 527.589999999999),
 ('4000006', 539.380000000001),
 ('4000001', 651.050000000001),
 ('4000007', 699.55),
 ('4000002', 706.97),
 ('4000008', 859.42)]
```

```
#câu 4: Show ID, number of transactions, and total cost for each customer, sorted by customer ID
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],(1,x[3]))).reduceByKey(lambda x1,x2: ( x1[0]+x2[0],    float(x1[1]) + float(x2[1])   )).sortBy(lambda x: x[1][1]).collect()

⇒ [('4000005', (5, 325.15)),
 ('4000004', (5, 337.06)),
 ('4000010', (6, 447.0900000000003)),
 ('4000009', (6, 457.83)),
 ('4000003', (3, 527.5899999999999)),
 ('4000006', (5, 539.3800000000001)),
 ('4000001', (8, 651.0500000000001)),
 ('4000007', (6, 699.55)),
 ('4000002', (6, 706.97)),
 ('4000008', (10, 859.42))]

#câu 5: Show name, number of transactions, and total cost for each customer, sorted by total cost
custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1])).collect()

transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],(1,x[3]))).reduceByKey(lambda x1,x2: ( x1[0]+x2[0],    float(x1[1]) + float(x2[1])   )).join( custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1])) ).map(lambda x: (x[1][1], x[1][0][0] , x[1][0][1] )    ).collect()

[('Gretchen', 5, 337.06),
 ('Elsie', 6, 699.55),
 ('Sherri', 3, 527.5899999999999),
 ('Malcolm', 6, 457.83),
 ('Hazel', 10, 859.42),
 ('Kristina', 8, 651.0500000000001),
 ('Paige', 6, 706.97),
 ('Karen', 5, 325.15),
 ('Patrick', 5, 539.3800000000001),
 ('Dolores', 6, 447.0900000000003)]

#6 Show name, game types played by each customer

transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().reduceByKey(lambda x1,x2: x1+";" + x2).join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))).map(lambda x: (x[1][1]    , x[1][0])).collect()

[('Elsie', 'Team Sports;Exercise & Fitness;Outdoor Recreation'),
 ('Gretchen', 'Indoor Games;Water Sports;Outdoor Recreation'),
 ('Sherri', 'Gymnastics;Outdoor Recreation;Water Sports'),
 ('Malcolm',
 'Gymnastics;Combat Sports;Outdoor Play Equipment;Indoor Games;Water Sports'),
```

```

# 7 Show ID, name, game types of all players who play 5 or more game types
transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().reduceByKey(lambda x1,x2: x1 + ";" + x2).map(lambda x: (x[0], x[1].split(";"))).filter(lambda x: len(x[1])>4).join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))).map(lambda x: (x[1][1],x[1][0])).collect()

[('Malcolm',
 ['Gymnastics',
 'Combat Sports',
 'Outdoor Play Equipment',
 'Indoor Games',
 'Water Sports']),
 ['Malcolm']]

#8 Show name of all distinct players of each game types

transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))).map(lambda x: (x[1][0],x[1][1])).reduceByKey(lambda x1,x2: x1+";" + x2).collect()

[('Water Sports', 'Gretchen;Sherri;Malcolm;Hazel;Patrick;Kristina;Paige'),
 ('Winter Sports', 'Patrick;Kristina'),
 ('Gymnastics', 'Sherri;Malcolm;Dolores;Kristina'),
 ('Team Sports', 'Elsie;Hazel;Dolores;Paige;Karen'),
 ('Exercise & Fitness', 'Elsie;Dolores;Kristina;Hazel;Karen')]

#9 Show all game types which don't have player under 40

custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],float(x[3]))).collect()

[('4000001', 55.0),
 ('4000002', 74.0),
 ('4000003', 34.0),
 ('4000004', 66.0),
 ('4000005', 74.0),
 ('4000006', 42.0),
 ('4000007', 43.0),
 ('4000008', 63.0),
 ('4000009', 39.0),
 ('4000010', 60.0)]

transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],float(x[3])))).map(lambda x: x[1]).reduceByKey(min).filter(lambda x:x[1]>40).collect()

[('Winter Sports', 42.0),
 ('Team Sports', 43.0),
 ('Exercise & Fitness', 43.0),
 ('Games', 60.0),
 ('Puzzles', 74.0),
 ('Air Sports', 74.0),
 ('Jumping', 42.0)]

#10 show average age of players of all gametypes
transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],float(x[3])))).map(lambda x: (x[1][0], (1,x[1][1]))).reduceByKey(lambda x1,x2: (x1[0]+x2[0] ,x1[1]+x2[1] ) ).map(lambda x: (x[0] , x[1][1]/x[1][0] ) ).collect()

[('Water Sports', 53.285714285714285),
 ('Winter Sports', 48.5),
 ('Gymnastics', 47.0),
 ('Team Sports', 62.8),
 ('Exercise & Fitness', 61.2),
 ('Indoor Games', 52.5),
 ('Outdoor Play Equipment', 54.5),
 ('Puzzles', 74.0),
 ('Air Sports', 74.0),
 ('Jumping', 42.0)]

```



# Big Data

## PySpark

Instructor: Trong-Hop Do

April 24<sup>th</sup> 2021

A photograph of a railway track receding into a landscape under a dramatic, cloudy sky.

**“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”**

– Chris Lynch, Vertica Systems

# Install Spark on Windows



# Install Java 8 or Later

- To install Apache Spark on windows, you would need Java 8 or later version hence download the Java version from Oracle and install it on your system.
- <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

Windows x64

166.79 MB



jdk-8u271-windows-x64.exe

# Apache Spark Installation on Windows

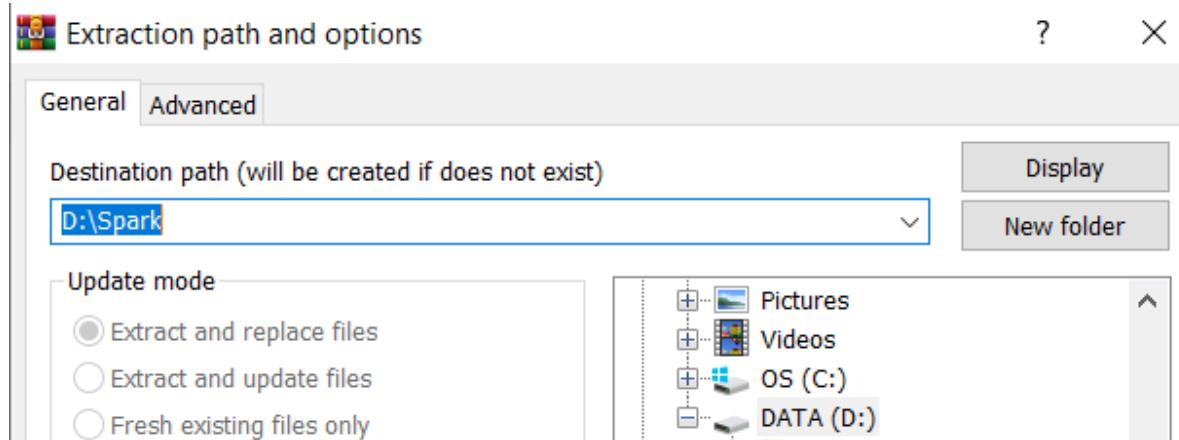
- Download Apache spark
- <https://spark.apache.org/downloads.html>

## Download Apache Spark™

1. Choose a Spark release:
2. Choose a package type:
3. Download Spark: [spark-3.0.1-bin-hadoop2.7.tgz](#)
4. Verify this release using the 3.0.1 [signatures](#), [checksums](#) and [project release KEYS](#).

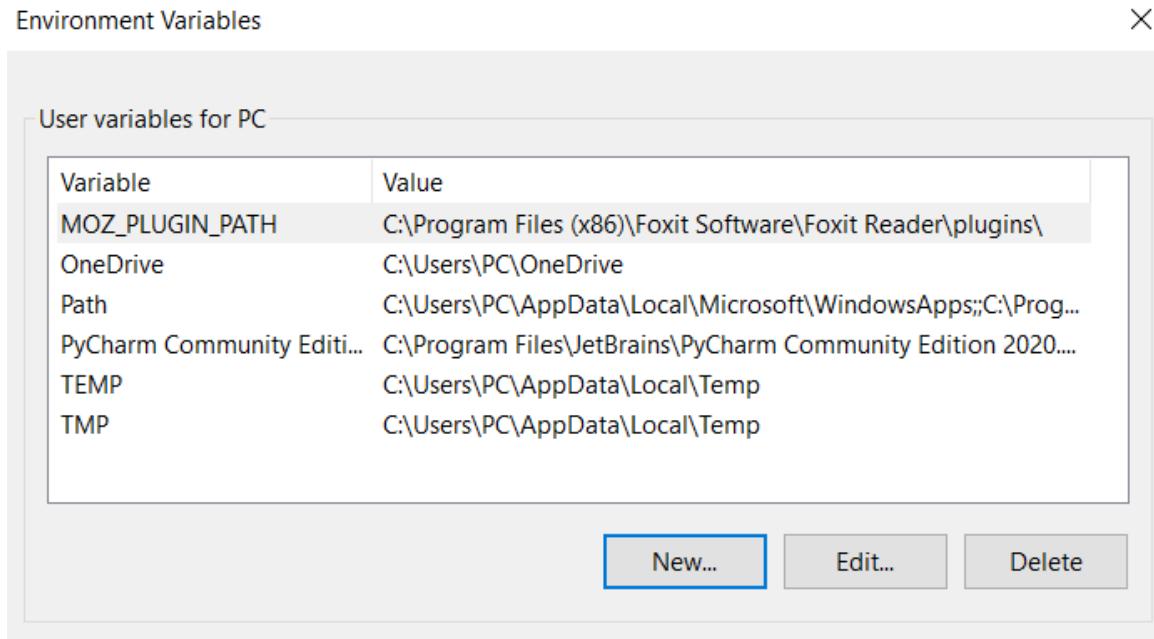
# Apache Spark Installation on Windows

- Extract the zip file to any folder

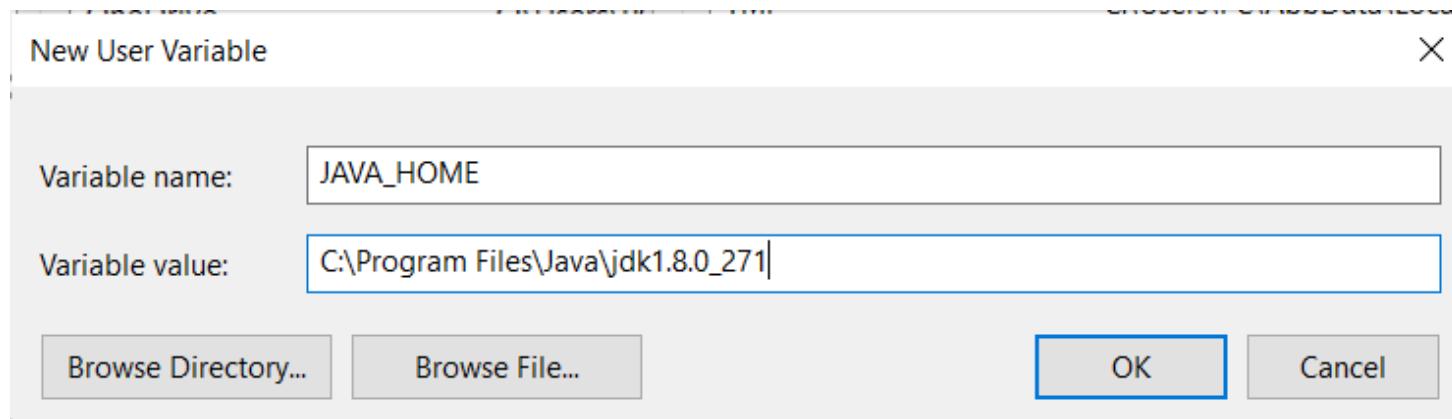


# Environment Variables Setting

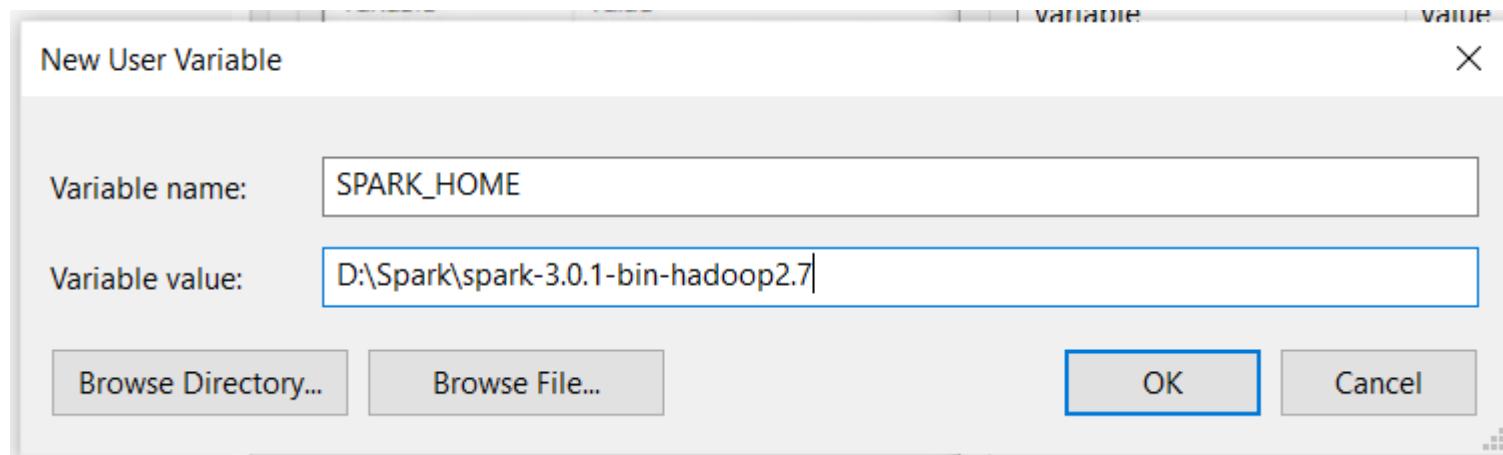
- Open System Environment Variables window and select Environment Variables.



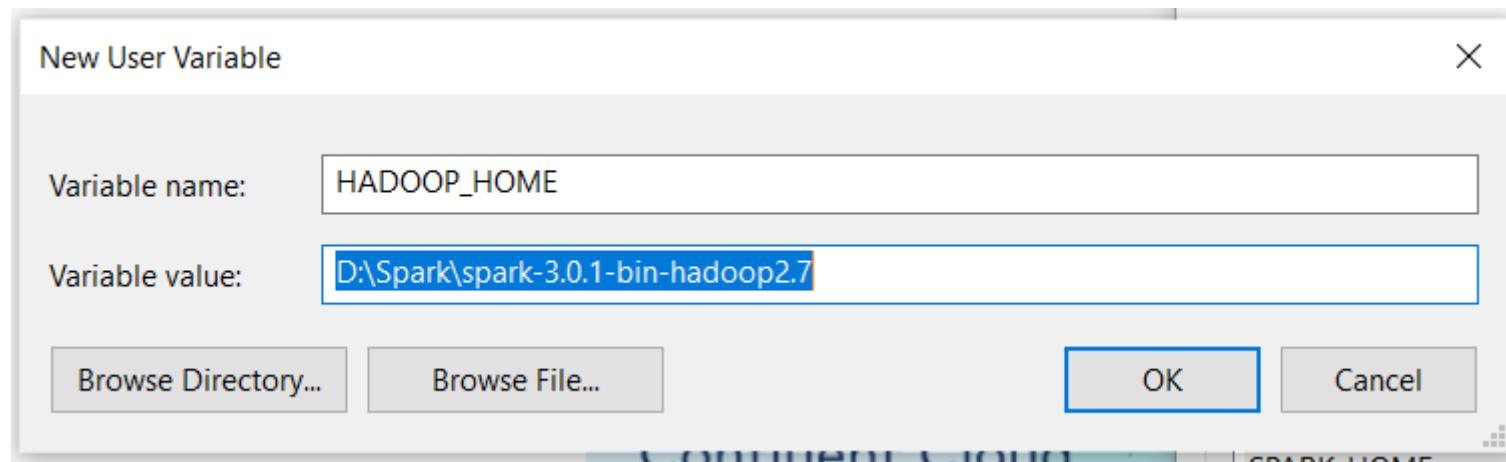
# Environment Variables Setting



# Apache Spark Installation on Windows

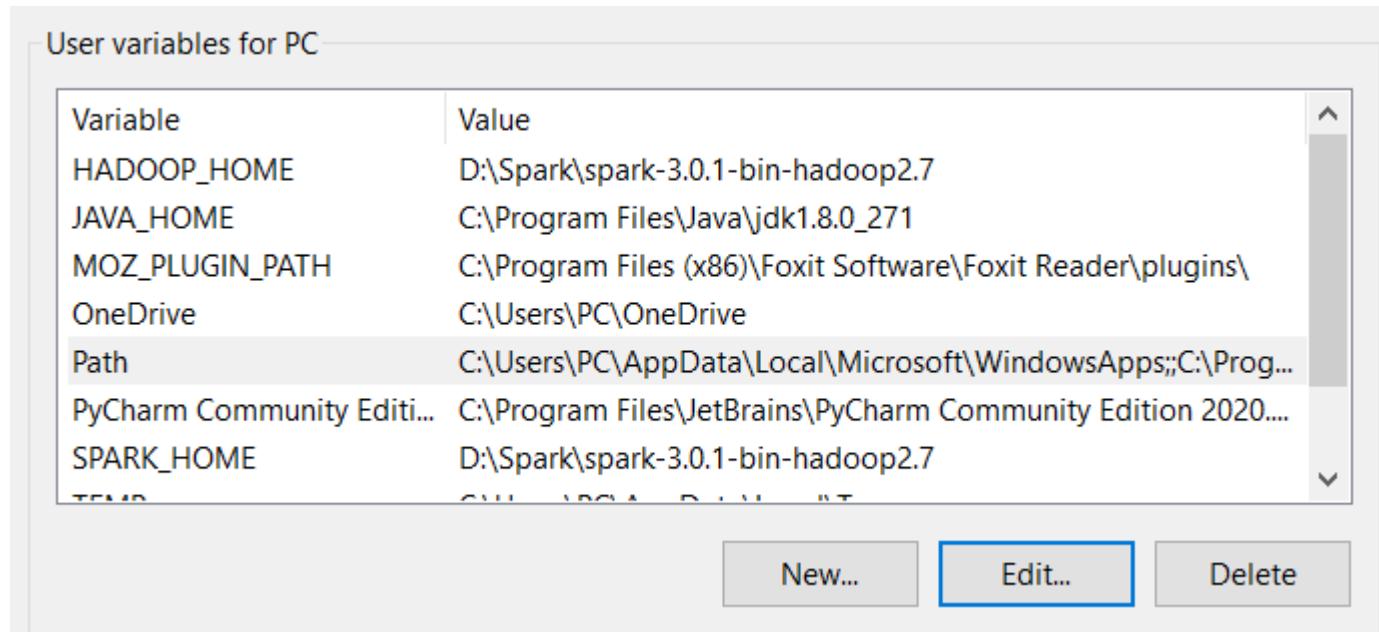


# Apache Spark Installation on Windows



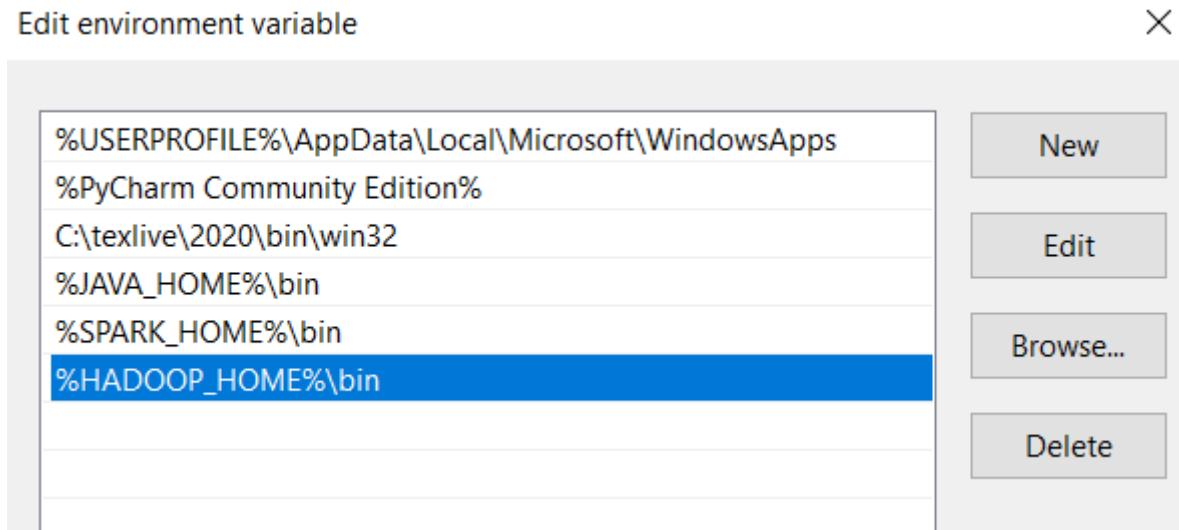
# Apache Spark Installation on Windows

- Now Edit the PATH variable



# Apache Spark Installation on Windows

- Add Spark, Java, and Hadoop bin location by selecting New option.



# Test apache Spark shell

```
C:\Users\PC>spark-shell
20/12/07 16:50:25 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
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://192.168.56.1:4040
Spark context available as 'sc' (master = local[*], app id = local-1607334630777).
Spark session available as 'spark'.
Welcome to

   _/\_ _/\_ _/\_ _/\_ _/\_
  / \ / \ / \ / \ / \ / \
 /_ \_. / \_, /_ \ /_ \ \_ \
  /_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \
   version 3.0.1

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_271)
Type in expressions to have them evaluated.
Type :help for more information.

scala> 20/12/07 16:50:45 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of P
rocessTree metrics is stopped

scala>
```

# Run PySpark on PySpark Shell

Type **pyspark** on command prompt

# Run PySpark on Jupyter lab

- Open Anaconda prompt and type “**python -m pip install findspark**”. This package is necessary to run spark from Jupyter notebook.

```
(base) D:\Spark\spark-3.0.1-bin-hadoop2.7>python -m pip install findspark
Collecting findspark
  Downloading findspark-1.4.2-py2.py3-none-any.whl (4.2 kB)
Installing collected packages: findspark
Successfully installed findspark-1.4.2
```

# Run PySpark on Jupyter lab

- Open jupyter notebook
- New -> Python 3

```
In [1]: import findspark  
  
findspark.init()
```

```
In [2]: import pyspark  
  
from pyspark.sql import SparkSession  
  
spark = SparkSession.builder.getOrCreate()  
  
df = spark.sql("select 'spark' as hello ")  
  
df.show()
```

```
+----+  
|hello|  
+----+  
|spark|  
+----+
```

# Run PySpark on Google Colab

✓ [1] !pip install pyspark==3.0.1  
38s

```
Collecting pyspark==3.0.1
  Downloading pyspark-3.0.1.tar.gz (204.2 MB)
    |████████| 204.2 MB 34 kB/s
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
    |████████| 198 kB 60.5 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.0.1-py2.py3-none-any.whl size=204612243 sha256=a1989b33b84227
  Stored in directory: /root/.cache/pip/wheels/5e/34/fa/b37b5cef503fc5148b478b2495043ba61b079120b7ff379f9b
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.0.1
```

✓ 11s

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
df = spark.sql("select 'spark' as hello")
df.show()
```

```
+---+
|hello|
+---+
|spark|
+---+
```

# Big Data Analytics with PySpark SQL



# What is PySpark

PySpark is a Spark library written in Python to run Python application using Apache Spark capabilities, using PySpark we can run applications parallelly on the distributed cluster (multiple nodes).

In other words, PySpark is a Python API for Apache Spark. Apache Spark is an analytical processing engine for large scale powerful distributed data processing and machine learning applications.



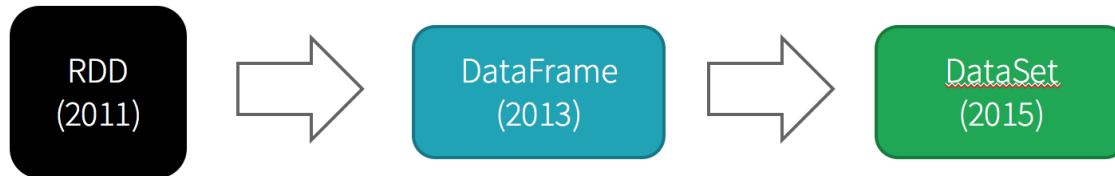
source: <https://databricks.com/>

# PySpark Modules and Packages

- PySpark RDD ([pyspark.RDD](#))
- PySpark DataFrame and SQL ([pyspark.sql](#))
- PySpark Streaming ([pyspark.streaming](#))
- PySpark MLlib ([pyspark.ml](#), [pyspark.mllib](#))
- PySpark GraphFrames ([GraphFrames](#))
- PySpark Resource ([pyspark.resource](#)) It's new in PySpark 3.0

# RDD vs DataFrame vs DataSet

## History of Spark APIs



Distribute collection  
of JVM objects

Functional Operators (map,  
filter, etc.)

Distribute collection  
of Row objects

Expression-based operations  
and UDFs

Internally rows, externally  
JVM objects

Almost the “Best of both  
worlds”: type safe + fast

Logical plans and optimizer

Fast/efficient internal  
representations

But slower than DF  
Not as good for interactive  
analysis, especially Python



In version 2.0, DataSet and DataFrame APIs are unified to provide a single API for developers. A DataFrame is a specific Dataset[T], where T=Row type, so DataFrame shares the same methods as Dataset.

# RDD vs DataFrame vs DataSet

Feature	RDD	DataFrame	DataSet
Immutable	Yes	Yes	Yes
Fault tolerant	Yes	Yes	Yes
Type-safe	Yes	No	Yes
Schema	No	Yes	Yes
Execution optimization	No	Yes	Yes
Level	Low	High	High

# What is SparkSession?

- Since Spark 2.0 SparkSession has become an entry point to PySpark to work with RDD, DataFrame. Prior to 2.0, SparkContext used to be an entry point.
- Spark Session also includes all the APIs available in different contexts –
  - Spark Context,
  - SQL Context,
  - Streaming Context,
  - Hive Context.

# SparkSession in PySpark shell

- By default PySpark shell provides “**spark**” object; which is an instance of SparkSession class. We can directly use this object where required in spark-shell.

```
>>> spark.version  
'3.0.1'  
>>> spark.createDataFrame([('Java', '20000'), ('Python', '10000'), ('Scala', '5000')]).show()  
+---+---+  
| _1| _2|  
+---+---+  
| Java|20000|  
| Python|10000|  
| Scala| 5000|  
+---+---+
```

# Create SparkSession in Jupyter lab

```
[1]: import findspark  
findspark.init()  
import pyspark  
from pyspark.sql import SparkSession  
  
spark = SparkSession.builder.appName("VeryFirstSparkExample").getOrCreate()
```

```
[2]: spark.version
```

```
[2]: '3.0.1'
```

```
[3]: spark.createDataFrame([('Java','20000'), ('Python','10000'), ('Scala','5000')]).show()
```

```
+----+----+  
| _1|_2|  
+----+----+  
| Java|20000|  
| Python|10000|  
| Scala| 5000|  
+----+----+
```

# SparkSession Commonly Used Methods

**version** – Returns Spark version where your application is running, probably the Spark version you cluster is configured with.

**createDataFrame()** – This creates a DataFrame from a collection and an RDD

**getActiveSession()** – returns an active Spark session.

**read()** – Returns an instance of DataFrameReader class, this is used to read records from csv, parquet, avro and more file formats into DataFrame.

**readStream()** – Returns an instance of DataStreamReader class, this is used to read streaming data. that can be used to read streaming data into DataFrame.

**sparkContext()** – Returns a SparkContext.

**sql** – Returns a DataFrame after executing the SQL mentioned.

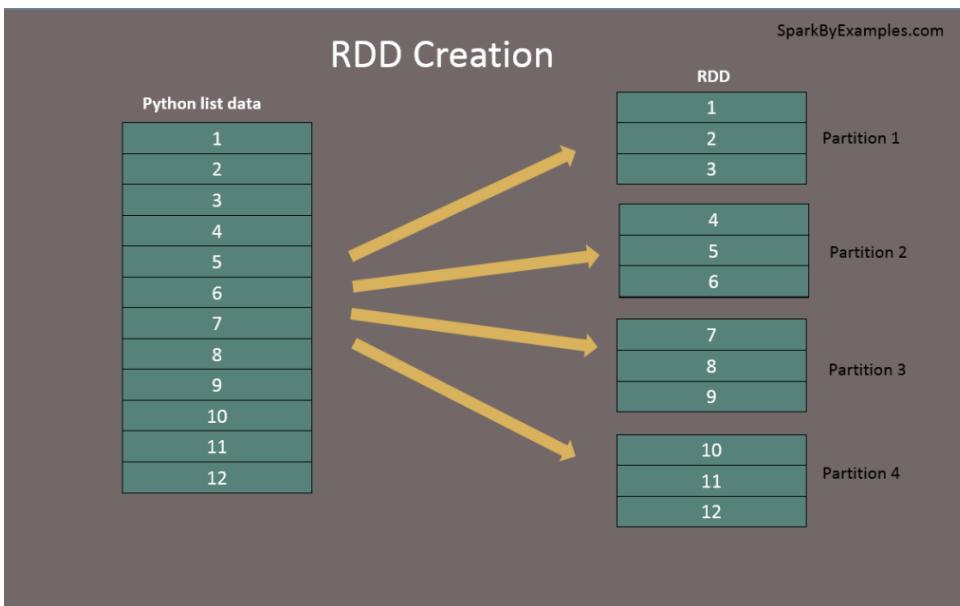
**sqlContext()** – Returns SQLContext.

**stop()** – Stop the current SparkContext.

**table()** – Returns a DataFrame of a table or view.

**udf()** – Creates a PySpark UDF to use it on DataFrame, Dataset, and SQL.

# Create RDD using sparkContext.parallelize()



By using `parallelize()` function of `SparkContext` (`sparkContext.parallelize()`) you can create an RDD. This function loads the existing collection from your driver program into parallelizing RDD. This is a basic method to create RDD and used when you already have data in memory that either loaded from a file or from a database. and it required all data to be present on the driver program prior to creating RDD.

# Create RDD using sparkContext.parallelize()

```
[4]: #Create RDD from parallelize  
data = [1,2,3,4,5,6,7,8,9,10,11,12]  
rdd=spark.sparkContext.parallelize(data)
```

```
[6]: rdd.collect()
```

```
[6]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

# Create RDD using sparkContext.textFile()

```
[7]: #Create RDD from external Data source  
rdd2 = spark.sparkContext.textFile("/path/textFile.txt")
```

```
[8]: #Reads entire file into a RDD as single record.  
rdd3 = spark.sparkContext.wholeTextFiles("/path/textFile.txt")
```

# PySpark RDD Operations

- **RDD transformations** – Transformations are lazy operations, instead of updating an RDD, these operations return another RDD.
- **RDD actions** – operations that trigger computation and return non-RDD values.
- Transformations on PySpark RDD returns another RDD and transformations are lazy meaning they don't execute until you call an action on RDD. Some transformations on RDD's are flatMap(), map(), reduceByKey(), filter(), sortByKey() and return new RDD instead of updating the current.

# RDD transformation: flatMap

**flatMap** – flatMap() transformation flattens the RDD after applying the function and returns a new RDD. On the below example, first, it splits each record by space in an RDD and finally flattens it. Resulting RDD consists of a single word on each record.

```
[9]: #Create RDD from external Data source
```

```
transRDD = spark.sparkContext.textFile("trans.txt")
```

```
[10]: transRDD.collect()
```

```
[10]: ['00000000,06-26-2011,4000001,040.33,Exercise & Fitness,Cardio Machine Accessories,Clarksville,Tennessee,credit',
      '00000001.05-26-2011.4000002.198.44.Exercise & Fitness.Weightlifting Gloves.Long Beach.California.credit'.
```

```
[11]: transRDD.flatMap(lambda x: x.split(",")).collect()
```

```
[11]: ['00000000',
      '06-26-2011',
      '4000001',
      '040.33',
      'Exercise & Fitness',
      'Cardio Machine Accessories',
      'Clarksville',
```

# RDD transformation: map

**map** – `map()` transformation is used to apply any complex operations like adding a column, updating a column e.t.c, the output of map transformations would always have the same number of records as input.

```
[11]: transRDD.flatMap(lambda x: x.split(",")).collect()
```

```
[11]: ['00000000',
       '06-26-2011',
       '4000001',
       '040.33',
       'Exercise & Fitness',
       'Cardio Machine Accessories',
       'Clarksville',
       'Tennessee',
       'credit',
       '00000001',
       '05-26-2011',
```

```
[16]: transRDD.flatMap(lambda x: x.split(",")).count()
```

```
[16]: 540
```

```
[14]: transRDD.map(lambda x: x.split(",")).collect()
```

```
[14]: [['00000000',
         '06-26-2011',
         '4000001',
         '040.33',
         'Exercise & Fitness',
         'Cardio Machine Accessories',
         'Clarksville',
         'Tennessee',
         'credit'],
        ['00000001',
         '05-26-2011',
```

```
[15]: transRDD.map(lambda x: x.split(",")).count()
```

```
[15]: 60
```

# RDD transformation: map

```
[22]: #Show customer ID and amount of each transaction  
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],x[3])).collect()
```

```
[22]: [ ('4000001', '040.33'),  
      ('4000002', '198.44'),  
      ('4000002', '005.58'),  
      ('4000003', '198.19'),  
      ('4000002', '098.81'),  
      ('4000004', '193.63'),  
      ('4000005', '027.89'),  
      ('4000006', '096.01'),  
      ('4000006', '010.44'),  
      ('4000006', '152.46'),  
      ('4000007', '180.28'),  
      ('4000009', '121.39'),
```

# RDD transformation: reduceByKey

**reduceByKey** – `reduceByKey()` merges the values for each key with the function specified. In our example, it reduces the word string by applying the sum function on value. The result of our RDD contains unique words and their count.

```
[27]: #Show ID and total cost of all transactions of each customer
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],x[3])).reduceByKey(lambda amount1,amount2: float(amount1)+float(amount2) ).collect()

[27]: [('4000004', 337.06),
       ('4000007', 699.55),
       ('4000008', 859.42),
       ('4000001', 651.050000000001),
       ('4000002', 706.97),
       ('4000003', 527.589999999999),
       ('4000005', 325.15),
       ('4000006', 539.380000000001),
       ('4000009', 457.83),
       ('4000010', 447.090000000003)]
```

# RDD transformation: sortByKey

```
[32]: #Show ID and total cost of all transctions of each customer, sorted by customer ID
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],x[3])).reduceByKey(lambda amount1,amount2: float(amount1)+float(amount2) ).sortByKey().collect()
<
[32]: [('4000001', 651.0500000000001),
('4000002', 706.97),
('4000003', 527.589999999999),
('4000004', 337.06),
('4000005', 325.15),
('4000006', 539.380000000001),
('4000007', 699.55),
('4000008', 859.42),
('4000009', 457.83),
('4000010', 447.0900000000003)]
```

# RDD transformation: filter

```
[40]: #Show customer IDs and games of all transaction where games include 'Sport'  
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],x[4])).filter(lambda x: 'Sport' in x[1]).collect()
```

```
[40]: [('4000002', 'Team Sports'),  
       ('4000006', 'Winter Sports'),  
       ('4000010', 'Team Sports'),  
       ('4000001', 'Combat Sports'),  
       ('4000008', 'Water Sports'),  
       ('4000008', 'Team Sports'),  
       ('4000008', 'Water Sports'),  
       ('4000005', 'Air Sports'),  
       ('4000009', 'Water Sports'),  
       ('4000003', 'Water Sports'),  
       ('4000009', 'Combat Sports'),  
       ('4000008', 'Team Sports'),  
       ('4000001', 'Water Sports'),  
       ('4000008', 'Team Sports'),  
       ('4000008', 'Team Sports'),  
       ('4000007', 'Team Sports'),  
       ('4000005', 'Team Sports'),  
       ('4000004', 'Water Sports'),
```

# RDD functions

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.html>

# Exercises

- Show the ID and all game types played by customers who play “Water Sports”.

Hint: use **reduceByKey()** to concatenate the game types of each customer IDs and then apply **filter()**. To remove duplicate game types for each ID, use **distinct()** function

```
[('4000008', 'Water Sports;Team Sports;Games;Outdoor Play Equipment;Outdoor Recreation'),  
 ('4000004', 'Indoor Games;Water Sports;Outdoor Recreation'),  
 ('4000003', 'Gymnastics;Outdoor Recreation;Water Sports'),  
 ('4000006', 'Jumping;Outdoor Play Equipment;Winter Sports;Water Sports'),  
 ('4000001', 'Combat Sports;Outdoor Recreation;Gymnastics;Exercise & Fitness;Water Sports;Winter Sports'),  
 ('4000009', 'Gymnastics;Combat Sports;Outdoor Play Equipment;Indoor Games;Water Sports'),  
 ('4000002', 'Outdoor Recreation;Exercise & Fitness;Team Sports;Water Sports')]
```

- Other exercises
1. Show IDs and number of transactions of each customer
  2. Show IDs and number of transactions of each customer, sorted by customer ID
  3. Show IDs and total cost of transactions of each customer, sorted by total cost
  4. Show ID, number of transactions, and total cost for each customer, sorted by customer ID
  5. Show name, number of transactions, and total cost for each customer, sorted by total cost
  6. Show ID, name, game types played by each customer
  7. Show ID, name, game types of all players who play 5 or more game types
  8. Show name of all distinct players of each game types
  9. Show all game types which don't have player under 40
  10. Show min, max, average age of players of all game types

# Create DataFrame from RDD

SPARKSESSION	RDD	DATAFRAME
createDataFrame(rdd)	toDF()	toDF(*cols)
createDataFrame(dataList)	toDF(*cols)	
createDataFrame(rowData,columns)		
createDataFrame(dataList,schema)		

# Create DataFrame from RDD

## Using toDF() function

```
[54]: columns = ["language", "users_count"]
       data = [("Java", "20000"), ("Python", "100000"), ("Scala", "3000")]
       rdd = spark.sparkContext.parallelize(data)
```

```
[55]: dfFromRDD1 = rdd.toDF()
       dfFromRDD1.printSchema()

root
 |-- _1: string (nullable = true)
 |-- _2: string (nullable = true)
```

```
[56]: dfFromRDD1 = rdd.toDF(columns)
       dfFromRDD1.printSchema()

root
 |-- language: string (nullable = true)
 |-- users_count: string (nullable = true)
```

# Create DataFrame from RDD

## Using `createDataFrame()` from `SparkSession`

- Calling `createDataFrame()` from `SparkSession` is another way to create PySpark DataFrame manually, it takes a list object as an argument. and chain with `toDF()` to specify names to the columns.

```
[57]: dfFromRDD2 = spark.createDataFrame(rdd).toDF(*columns)
dfFromRDD2.printSchema()
```

```
root
 |-- language: string (nullable = true)
 |-- users_count: string (nullable = true)
```

# Create DataFrame from List Collection

Using `createDataFrame()` from `SparkSession`

```
[67]: data = [("Java", "20000"), ("Python", "100000"), ("Scala", "3000")]
      dfFromData2 = spark.createDataFrame(data).toDF(*columns)
      dfFromData2.show()
```

language	users_count
Java	20000
Python	100000
Scala	3000

# Create DataFrame from List Collection

## Using `createDataFrame()` with the Row type

`createDataFrame()` has another signature in PySpark which takes the collection of Row type and schema for column names as arguments. To use this first we need to convert our “data” object from the list to list of Row.

```
[72]: from pyspark.sql import Row
rowData = map(lambda x: Row(*x), data)
dfFromData3 = spark.createDataFrame(rowData,columns)
dfFromData3.show()
```

language	users_count
Java	20000
Python	100000
Scala	3000

# Create DataFrame from List Collection

## Create DataFrame with schema

If you wanted to specify the column names along with their data types, you should create the StructType schema first and then assign this while creating a DataFrame.

```
[78]: from pyspark.sql.types import StructType, StructField, StringType, IntegerType
data2 = [("James","","Smith","36636","M",3000),
        ("Maria","Anne","Jones","39192","F",4000),
        ("Jen","Mary","Brown","", "F",-1)
      ]
schema = StructType([
    StructField("firstname",StringType(),True), \
    StructField("middlename",StringType(),True), \
    StructField("lastname",StringType(),True), \
    StructField("id", StringType(), True), \
    StructField("gender", StringType(), True), \
    StructField("salary", IntegerType(), True) \
  ])
df = spark.createDataFrame(data=data2,schema=schema)
df.printSchema()
df.show()
```

```
root
|-- firstname: string (nullable = true)
|-- middlename: string (nullable = true)
|-- lastname: string (nullable = true)
|-- id: string (nullable = true)
|-- gender: string (nullable = true)
|-- salary: integer (nullable = true)

+-----+-----+-----+-----+
|firstname|middlename|lastname|   id|gender|salary|
+-----+-----+-----+-----+
|    James|          |  Smith|36636|     M|  3000|
|   Maria|       Anne|  Jones|39192|     F|  4000|
|     Jen|       Mary|  Brown||     F|   -1|
+-----+-----+-----+-----+
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV

```
[2]: df = spark.read.csv("zipcodes.csv")
df.printSchema()
df.show()
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: string (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
 |-- _c12: string (nullable = true)
 |-- _c13: string (nullable = true)
 |-- _c14: string (nullable = true)
 |-- _c15: string (nullable = true)
 |-- _c16: string (nullable = true)
 |-- _c17: string (nullable = true)
 |-- _c18: string (nullable = true)
 |-- _c19: string (nullable = true)
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV

- Using fully qualified data source name, you can alternatively do the following.

```
[7]: df = spark.read.format("csv").load("zipcodes.csv")
df.show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      _c0|      _c1|      _c2|      _c3|      _c4|      _c5|      _c6|      _c7|      _c8|      _c9|      _c10|      _c11|      _c12|
|_c13|      _c14|      _c15|      _c16|      _c17|      _c18|      _c19|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|RecordNumber|Zipcode|ZipCodeType|          City|State| LocationType|  Lat|  Long|Xaxis|Yaxis|Zaxis|WorldRegion|Country|
|LocationText|          Location|Decommissioned|TaxReturnsFiled|EstimatedPopulation|TotalWages|      Notes|
|           1|    704| STANDARD|      PARC PARQUE|     PR|NOT ACCEPTABLE|17.96| -66.22| 0.38|-0.87|  0.3|        NA|       US|
|Parc Parque, PR|NA-US-PR-PARC PARQUE|      FALSE|         null|         null|         null|         null|
|           2|    704| STANDARD|PASEO COSTA DEL SUR|     PR|NOT ACCEPTABLE|17.96| -66.22| 0.38|-0.87|  0.3|        NA|       US|P
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV - Using Header Record For Column Names

```
[11]: df2 = spark.read.option("header",True).csv("zipcodes.csv")
df2.printSchema()
```

```
root
 |-- RecordNumber: string (nullable = true)
 |-- Zipcode: string (nullable = true)
 |-- ZipCodeType: string (nullable = true)
 |-- City: string (nullable = true)
 |-- State: string (nullable = true)
 |-- LocationType: string (nullable = true)
 |-- Lat: string (nullable = true)
 |-- Long: string (nullable = true)
 |-- Xaxis: string (nullable = true)
 |-- Yaxis: string (nullable = true)
 |-- Zaxis: string (nullable = true)
 |-- WorldRegion: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- LocationText: string (nullable = true)
 |-- Location: string (nullable = true)
 |-- Decommissioned: string (nullable = true)
 |-- TaxReturnsFiled: string (nullable = true)
 |-- EstimatedPopulation: string (nullable = true)
 |-- TotalWages: string (nullable = true)
 |-- Notes: string (nullable = true)
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV - Read Multiple CSV Files

- df = spark.read.csv("path1,path2,path3")

# Create DataFrame from Data sources

Creating DataFrame from CSV - Read all CSV Files in a Directory

- df = spark.read.csv("Folder path")

# Create DataFrame from Data sources

## Creating DataFrame from CSV - Options While Reading CSV File

- **delimiter** option is used to specify the column delimiter of the CSV file. By default, it is comma (,) character, but can be set to any character like pipe(|), tab (\t), space using this option.

```
df3 = spark.read.options(delimiter=',').csv("zipcodes.csv")
df3.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: string (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
 |-- _c12: string (nullable = true)
 |-- _c13: string (nullable = true)
 |-- _c14: string (nullable = true)
 |-- _c15: string (nullable = true)
 |-- _c16: string (nullable = true)
 |-- _c17: string (nullable = true)
 |-- _c18: string (nullable = true)
 |-- _c19: string (nullable = true)
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV - Options While Reading CSV File

- **inferSchema:** The default value set to this option is False when setting to true it automatically infers column types based on the data. Note that, it requires reading the data one more time to infer the schema.

```
[21]: df4 = spark.read.options(inferSchema='True', delimiter=',').csv("zipcodes.csv")
df4.printSchema()
```

```
root
|-- _c0: string (nullable = true)
|-- _c1: string (nullable = true)
|-- _c2: string (nullable = true)
|-- _c3: string (nullable = true)
|-- _c4: string (nullable = true)
|-- _c5: string (nullable = true)
|-- _c6: string (nullable = true)
|-- _c7: string (nullable = true)
|-- _c8: string (nullable = true)
|-- _c9: string (nullable = true)
|-- _c10: string (nullable = true)
|-- _c11: string (nullable = true)
|-- _c12: string (nullable = true)
|-- _c13: string (nullable = true)
|-- _c14: string (nullable = true)
|-- _c15: string (nullable = true)
|-- _c16: string (nullable = true)
|-- _c17: string (nullable = true)
|-- _c18: string (nullable = true)
|-- _c19: string (nullable = true)
```

Why're all String?

```
[28]: df3 = spark.read.options(inferSchema='True', delimiter=',').csv("zipcodesNoHeader.csv")
df3.printSchema()
```

```
root
|-- _c0: integer (nullable = true)
|-- _c1: integer (nullable = true)
|-- _c2: string (nullable = true)
|-- _c3: string (nullable = true)
|-- _c4: string (nullable = true)
|-- _c5: string (nullable = true)
|-- _c6: double (nullable = true)
|-- _c7: double (nullable = true)
|-- _c8: double (nullable = true)
|-- _c9: double (nullable = true)
|-- _c10: double (nullable = true)
|-- _c11: string (nullable = true)
|-- _c12: string (nullable = true)
|-- _c13: string (nullable = true)
|-- _c14: string (nullable = true)
|-- _c15: boolean (nullable = true)
|-- _c16: integer (nullable = true)
|-- _c17: integer (nullable = true)
|-- _c18: integer (nullable = true)
|-- _c19: string (nullable = true)
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV - Options While Reading CSV File

- **header:** This option is used to read the first line of the CSV file as column names. By default the value of this option is False , and all column types are assumed to be a string.

```
[22]: df3 = spark.read.options(header='True', inferSchema='True', delimiter=',').csv("zipcodes.csv")
df3.printSchema()
```

```
root
 |-- RecordNumber: integer (nullable = true)
 |-- Zipcode: integer (nullable = true)
 |-- ZipCodeType: string (nullable = true)
 |-- City: string (nullable = true)
 |-- State: string (nullable = true)
 |-- LocationType: string (nullable = true)
 |-- Lat: double (nullable = true)
 |-- Long: double (nullable = true)
 |-- Xaxis: double (nullable = true)
 |-- Yaxis: double (nullable = true)
 |-- Zaxis: double (nullable = true)
 |-- WorldRegion: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- LocationText: string (nullable = true)
 |-- Location: string (nullable = true)
 |-- Decommissioned: boolean (nullable = true)
 |-- TaxReturnsFiled: integer (nullable = true)
 |-- EstimatedPopulation: integer (nullable = true)
 |-- TotalWages: integer (nullable = true)
 |-- Notes: string (nullable = true)
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV – user specified custom schema

- We can specify schema by using the **schema** option belonging to `read.csv()`

```
s = spark.read.schema(user_schema)
```

- Where **user\_schema** is a
  - `pyspark.sql.types.StructType` object
  - or
  - DDL-formatted string

# Create DataFrame from Data sources

## Creating DataFrame from CSV - StructType custom schema

```
from pyspark.sql.types import *
schema = StructType() \
    .add("RecordNumber",IntegerType(),True) \
    .add("Zipcode",IntegerType(),True) \
    .add("ZipCodeType",StringType(),True) \
    .add("City",StringType(),True) \
    .add("State",StringType(),True) \
    .add("LocationType",StringType(),True) \
    .add("Lat",DoubleType(),True) \
    .add("Long",DoubleType(),True) \
    .add("Xaxis",IntegerType(),True) \
    .add("Yaxis",DoubleType(),True) \
    .add("Zaxis",DoubleType(),True) \
    .add("WorldRegion",StringType(),True) \
    .add("Country",StringType(),True) \
    .add("LocationText",StringType(),True) \
    .add("Location",StringType(),True) \
    .add("Decommissioned",BooleanType(),True) \
    .add("TaxReturnsFiled",StringType(),True) \
    .add("EstimatedPopulation",IntegerType(),True) \
    .add("TotalWages",IntegerType(),True) \
    .add("Notes",StringType(),True)

df_with_schema = spark.read.format("csv").option("header", True).schema(schema).load("zipcodes.csv")
df_with_schema.printSchema()
```

```
root
|-- RecordNumber: integer (nullable = true)
|-- Zipcode: integer (nullable = true)
|-- ZipCodeType: string (nullable = true)
|-- City: string (nullable = true)
|-- State: string (nullable = true)
|-- LocationType: string (nullable = true)
|-- Lat: double (nullable = true)
|-- Long: double (nullable = true)
|-- Xaxis: integer (nullable = true)
|-- Yaxis: double (nullable = true)
|-- Zaxis: double (nullable = true)
|-- WorldRegion: string (nullable = true)
|-- Country: string (nullable = true)
|-- LocationText: string (nullable = true)
|-- Location: string (nullable = true)
|-- Decommissioned: boolean (nullable = true)
|-- TaxReturnsFiled: string (nullable = true)
|-- EstimatedPopulation: integer (nullable = true)
|-- TotalWages: integer (nullable = true)
|-- Notes: string (nullable = true)
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV – DLL formatted string custom schema

```
transDF = spark.read.options(delimiter=',').schema('trans_id INT, date STRING, cust_ID INT, amount DOUBLE, game STRING, equipment STRING, city STRING, state STRING, mode STRING').csv("trans.txt")
```

```
transDF.printSchema
transDF.show()
```

---

```
root
|-- trans_id: integer (nullable = true)
|-- date: string (nullable = true)
|-- cust_ID: integer (nullable = true)
|-- amount: double (nullable = true)
|-- game: string (nullable = true)
|-- equipment: string (nullable = true)
|-- city: string (nullable = true)
|-- state: string (nullable = true)
|-- mode: string (nullable = true)

+-----+-----+-----+-----+-----+-----+-----+-----+
|trans_id|      date|cust_ID|amount|      game|    equipment|      city|    state| mode|
+-----+-----+-----+-----+-----+-----+-----+-----+
|      0|06-26-2011|4000001| 40.33|Exercise & Fitness|Cardio Machine Ac...|Clarksville| Tennessee|credit|
|      1|05-26-2011|4000002|198.44|Exercise & Fitness|Weightlifting Gloves|Long Beach| California|credit|
|      2|06-01-2011|4000002|   5.58|Exercise & Fitness|Weightlifting Mac...|Anaheim| California|credit|
|      3|06-05-2011|4000003|198.19|          Gymnastics|Gymnastics Rings|Milwaukee| Wisconsin|credit|
|      4|12-17-2011|4000002| 98.81|        Team Sports|Field Hockey|Nashville | Tennessee|credit|
|      5|02-14-2011|4000004|193.63|Outdoor Recreation|Camping & Backpac...|Chicago| Illinois|credit|
|      6|10-28-2011|4000005| 27.89|          Puzzles|Jigsaw Puzzles|Charleston|South Carolina|credit|
|      7|07-14-2011|4000006| 96.01|Outdoor Play Equi...|   Sandboxes|Columbus|       Ohio|credit|
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV - Write PySpark DataFrame to CSV file-

- Use the write() method of the PySpark DataFrameWriter object to write PySpark DataFrame to a CSV file.

```
df.write.option("header",True).csv("newzipcodes")
```

- While writing a CSV file you can use several options. for example, header to output the DataFrame column names as header record and delimiter to specify the delimiter on the CSV output file.

```
df2.write.options(header='True', delimiter=',').csv("newzipcodes")
```

# Create DataFrame from Data sources

## Creating DataFrame from CSV - Write PySpark DataFrame to CSV file-

### Saving modes

PySpark DataFrameWriter also has a method mode() to specify saving mode.

**overwrite** – mode is used to overwrite the existing file.

**append** – To add the data to the existing file.

**ignore** – Ignores write operation when the file already exists.

**error** – This is a default option when the file already exists, it returns an error.

```
df2.write.mode('overwrite').csv("newzipcodes")
```

#you can also use this

```
df2.write.format("csv").mode('overwrite').save("newzipcodes")
```

# Create DataFrame from Data sources

## Creating DataFrame from text file

You can use `.text()`

```
[41]: df = spark.read.text("zipcodes.txt")
df.printSchema()
df.collect()
```

```
root
 |-- value: string (nullable = true)
```

```
[41]: [Row(value='RecordNumber\ zipcode\ zipCodeType\ city\ state\ locationType'),
Row(value='1\t704\tSTANDARD\tPARQUE\tPR\tNOT ACCEPTABLE'),
Row(value='2\t704\tSTANDARD\tPASEO COSTA DEL SUR\tPR\tNOT ACCEPTABLE'),
Row(value='10\t709\tSTANDARD\tBDA SAN LUIS\tPR\tNOT ACCEPTABLE'),
Row(value='61391\t76166\tUNIQUE\tCINGULAR WIRELESS\tTX\tNOT ACCEPTABLE'),
Row(value='61392\t76177\tSTANDARD\tFORT WORTH\tTX\tPRIMARY'),
Row(value='61393\t76177\tSTANDARD\tFORT WORTH\tTX\tACCEPTABLE'),
Row(value='4\t704\tSTANDARD\tBURE EUGENE RICE\tPR\tNOT ACCEPTABLE'),
Row(value='39827\t85209\tSTANDARD\tMESA\tAZ\tPRIMARY'),
Row(value='39828\t85210\tSTANDARD\tMESA\tAZ\tPRIMARY'),
Row(value='49345\t32046\tSTANDARD\tHILLIARD\tFL\tPRIMARY'),
Row(value='49346\t34445\tPO BOX\THOLDER\tFL\tPRIMARY'),
Row(value='49347\t32564\tSTANDARD\tHOLT\tFL\tPRIMARY'),
Row(value='49348\t34487\tPO BOX\THOMOSASSA\tFL\tPRIMARY'),
Row(value='10\t708\tSTANDARD\tBDA SAN LUIS\tPR\tNOT ACCEPTABLE'),
Row(value='3\t704\tSTANDARD\tSECT LANAUSSIE\tPR\tNOT ACCEPTABLE'),
Row(value='54354\t36275\tPO BOX\SPRING GARDEN\tAL\tPRIMARY'),
Row(value='54355\t35146\tSTANDARD\tSPRINGVILLE\tAL\tPRIMARY'),
Row(value='54356\t35585\tSTANDARD\tSPRUCE PINE\tAL\tPRIMARY'),
Row(value='76511\t27007\tSTANDARD\tASH HILL\tNC\tNOT ACCEPTABLE'),
Row(value='76512\t27203\tSTANDARD\tASHEBORO\tNC\tPRIMARY'),
Row(value='76513\t27204\tPO BOX\ASHEBORO\tNC\tPRIMARY')]
```

But `.csv()` is still much better

```
[48]: df = spark.read.options(header='True',inferSchema='True', delimiter='\t').csv("zipcodes.txt")
df.printSchema()
```

```
root
 |-- RecordNumber: integer (nullable = true)
 |-- Zipcode: integer (nullable = true)
 |-- ZipCodeType: string (nullable = true)
 |-- City: string (nullable = true)
 |-- State: string (nullable = true)
 |-- LocationType: string (nullable = true)
```

# PySpark dataframe function

## Select Columns From DataFrame

```
transDF = spark.read.options(delimiter=',').schema('trans_id INT, date STRING, cust_id INT, amount DOUBLE, game STRING, equipment STRING, city STRING, state STRING, mode STRING').csv("trans.txt")
```

```
transDF.printSchema()
transDF.show()
#you have several way to select columns
transDF.select('cust_id','amount').show()
transDF.select(transDF.cust_id,transDF.amount).show()
transDF.select(transDF['cust_id'],transDF['amount']).show()
#select from a list
twocolumns = ['cust_id','amount']
transDF.select(twocolumns).show()
#select all column
transDF.select([col for col in transDF.columns]).show()
transDF.select('*').show()
```

# PySpark dataframe function

## PySpark withColumn()

- PySpark **withColumn()** is a transformation function of DataFrame which is used to change the value, convert the datatype of an existing column, create a new column, and many more.
- You can use withColumn() to
  - [Change DataType using PySpark withColumn\(\)](#)
  - [Update The Value of an Existing Column](#)
  - [Create a Column from an Existing](#)
  - [Add a New Column using withColumn\(\)](#)
  - [Rename Column Name](#)

# PySpark dataframe function

## withColumn() - Change DataType

```
[13]: from pyspark.sql.functions import col  
transDF.withColumn('trans_id',col('trans_id').cast('String')).printSchema()
```

root  
|-- trans\_id: string (nullable = true)  
|-- date: string (nullable = true)  
|-- cust\_id: integer (nullable = true)  
|-- amount: double (nullable = true)  
|-- game: string (nullable = true)  
|-- equipment: string (nullable = true)  
|-- city: string (nullable = true)  
|-- state: string (nullable = true)  
|-- mode: string (nullable = true)

**pyspark.sql.functions.col(col)**  
Returns a **Column** based on the given column name.'

**Column.cast(dataType)**  
Convert the column into type **dataType**.

# PySpark dataframe function

## withColumn() - Update The Value of an Existing Column

```
[15]: from pyspark.sql.functions import col  
transDF.withColumn('amount',col('amount')*2).show()
```

trans_id	date	cust_id	amount	game	equipment	city	state	mode
0 06-26-2011 4000001  80.66  Exercise & Fitness Cardio Machine Ac...  Clarksville  Tennessee credit								
1 05-26-2011 4000002 396.88  Exercise & Fitness Weightlifting Gloves  Long Beach  California credit								
2 06-01-2011 4000002  11.16  Exercise & Fitness Weightlifting Mac...  Anaheim  California credit								
3 06-05-2011 4000003 396.38  Gymnastics  Gymnastics Rings  Milwaukee  Wisconsin credit								
4 12-17-2011 4000002 197.62  Team Sports  Field Hockey  Nashville   Tennessee credit								
5 02-14-2011 4000004 387.26  Outdoor Recreation Camping & Backpac...  Chicago  Illinois credit								
6 10-28-2011 4000005  55.78  Puzzles  Jigsaw Puzzles  Charleston South Carolina credit								
7 07-14-2011 4000006 192.02 Outdoor Play Equi...  Sandboxes  Columbus  Ohio credit								
8 01-17-2011 4000006  20.88  Winter Sports  Snowmobiling  Des Moines  Iowa credit								
9 05-17-2011 4000006 304.92  Jumping  Bungee Jumping St. Petersburg  Florida credit								
10 05-29-2011 4000007 360.56  Outdoor Recreation  Archery  Reno  Nevada credit								
11 06-18-2011 4000009 242.78 Outdoor Play Equi...  Swing Sets  Columbus  Ohio credit								
12 02-08-2011 4000009  83.04  Indoor Games  Bowling  San Francisco  California credit								
13 03-13-2011 4000010  215.6  Team Sports  Field Hockey  Honolulu   Hawaii credit								
14 02-25-2011 4000010  73.62  Gymnastics  Vaulting Horses  Los Angeles  California credit								
15 10-20-2011 4000001 275.28  Combat Sports  Fencing  Honolulu   Hawaii credit								
16 05-28-2011 4000010  71.12  Exercise & Fitness  Free Weight Bars  Columbia South Carolina credit								
17 10-18-2011 4000008  151.1  Water Sports Scuba Diving & Sn...  Omaha  Nebraska credit								
18 11-18-2011 4000008  177.3  Team Sports  Baseball Salt Lake City  Utah credit								
19 08-28-2011 4000008 103.62  Water Sports  Life Jackets  Newark  New Jersey credit								

# PySpark dataframe function

## withColumn() - Create a Column from an Existing

```
[10]: from pyspark.sql.functions import col  
transDF.withColumn("new amount", col("amount")*2).show()
```

trans_id	date	cust_id	amount	game	equipment	city	state	mode	new amount
0 06-26-2011 4000001  40.33  Exercise & Fitness Cardio Machine Ac...  Clarksville  Tennessee credit  80.66	1 05-26-2011 4000002 198.44  Exercise & Fitness Weightlifting Gloves  Long Beach  California credit  396.88	2 06-01-2011 4000002  5.58  Exercise & Fitness Weightlifting Mac...  Anaheim  California credit  11.16	3 06-05-2011 4000003 198.19  Gymnastics  Gymnastics Rings  Milwaukee  Wisconsin credit  396.38	4 12-17-2011 4000002  98.81  Team Sports  Field Hockey  Nashville   Tennessee credit  197.62	5 02-14-2011 4000004 193.63  Outdoor Recreation Camping & Backpack...  Chicago  Illinois credit  387.26	6 10-28-2011 4000005  27.89  Puzzles  Jigsaw Puzzles  Charleston South Carolina credit  55.78	7 07-14-2011 4000006  96.01 Outdoor Play Equi...  Sandboxes  Columbus  Ohio credit  192.02	8 01-17-2011 4000006  10.44  Winter Sports  Snowmobiling  Des Moines  Iowa credit  20.88	
9 05-17-2011 4000006 152.46  Jumping  Bungee Jumping St. Petersburg  Florida credit  304.92	10 05-29-2011 4000007 180.28  Outdoor Recreation  Archery  Reno  Nevada credit  360.56	11 06-18-2011 4000009 121.39 Outdoor Play Equi...  Swing Sets  Columbus  Ohio credit  242.78	12 02-08-2011 4000009  41.52  Indoor Games  Bowling  San Francisco  California credit  83.04	13 03-13-2011 4000010  107.8  Team Sports  Field Hockey  Honolulu   Hawaii credit  215.6	14 02-25-2011 4000010  36.81  Gymnastics  Vaulting Horses  Los Angeles  California credit  73.62	15 10-20-2011 4000001 137.64  Combat Sports  Fencing  Honolulu   Hawaii credit  275.28	16 05-28-2011 4000010  35.56  Exercise & Fitness  Free Weight Bars  Columbia South Carolina credit  71.12	17 10-18-2011 4000008  75.55  Water Sports Scuba Diving & Sn...  Omaha  Nebraska credit  151.1	18 11-18-2011 4000008  88.65  Team Sports  Baseball Salt Lake City  Utah credit  177.3
19 08-28-2011 4000008  51.81  Water Sports  Life Jackets  Newark  New Jersey credit  103.62									

# PySpark dataframe function

## withColumn() - Add a New Column

pyspark.sql.functions.lit(col)

Creates a **Column** of literal value.

```
[19]: from pyspark.sql.functions import lit  
transDF.withColumn("Country", lit("USA")).show()
```

trans_id	date	cust_id	amount	game	equipment	city	state	mode	Country
0	06-26-2011	4000001	40.33	Exercise & Fitness	Cardio Machine Ac...	Clarksville	Tennessee	credit	USA
1	05-26-2011	4000002	198.44	Exercise & Fitness	Weightlifting Gloves	Long Beach	California	credit	USA
2	06-01-2011	4000002	5.58	Exercise & Fitness	Weightlifting Mac...	Anaheim	California	credit	USA
3	06-05-2011	4000003	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin	credit	USA
4	12-17-2011	4000002	98.81	Team Sports	Field Hockey	Nashville	Tennessee	credit	USA
5	02-14-2011	4000004	193.63	Outdoor Recreation	Camping & Backpac...	Chicago	Illinois	credit	USA
6	10-28-2011	4000005	27.89	Puzzles	Jigsaw Puzzles	Charleston	South Carolina	credit	USA
7	07-14-2011	4000006	96.01	Outdoor Play Equi...	Sandboxes	Columbus	Ohio	credit	USA
8	01-17-2011	4000006	10.44	Winter Sports	Snowmobiling	Des Moines	Iowa	credit	USA
9	05-17-2011	4000006	152.46	Jumping	Bungee Jumping	St. Petersburg	Florida	credit	USA
10	05-29-2011	4000007	180.28	Outdoor Recreation	Archery	Reno	Nevada	credit	USA
11	06-18-2011	4000009	121.39	Outdoor Play Equi...	Swing Sets	Columbus	Ohio	credit	USA
12	02-08-2011	4000009	41.52	Indoor Games	Bowling	San Francisco	California	credit	USA
13	03-13-2011	4000010	107.8	Team Sports	Field Hockey	Honolulu	Hawaii	credit	USA
14	02-25-2011	4000010	36.81	Gymnastics	Vaulting Horses	Los Angeles	California	credit	USA
15	10-20-2011	4000011	137.64	Combat Sports	Fencing	Honolulu	Hawaii	credit	USA
16	05-28-2011	4000010	35.56	Exercise & Fitness	Free Weight Bars	Columbia	South Carolina	credit	USA
17	10-18-2011	4000008	75.55	Water Sports	Scuba Diving & Sn...	Omaha	Nebraska	credit	USA
18	11-18-2011	4000008	88.65	Team Sports	Baseball	Salt Lake City	Utah	credit	USA
19	08-28-2011	4000008	51.81	Water Sports	Life Jackets	Newark	New Jersey	credit	USA

only showing top 20 rows

# PySpark dataframe function

## withColumn() - Rename Column Name

```
[21]: from pyspark.sql.functions import lit  
transDF.withColumnRenamed('amount', 'cost').show()
```

trans_id	date	cust_id	cost	game	equipment	city	state	mode
0 06-26-2011 4000001  40.33  Exercise & Fitness Cardio Machine Ac...  Clarksville  Tennessee credit	1 05-26-2011 4000002 198.44  Exercise & Fitness Weightlifting Gloves  Long Beach  California credit	2 06-01-2011 4000002  5.58  Exercise & Fitness Weightlifting Mac...  Anaheim  California credit	3 06-05-2011 4000003 198.19  Gymnastics  Gymnastics Rings  Milwaukee  Wisconsin credit	4 12-17-2011 4000002  98.81  Team Sports  Field Hockey  Nashville   Tennessee credit	5 02-14-2011 4000004 193.63  Outdoor Recreation Camping & Backpac...  Chicago  Illinois credit	6 10-28-2011 4000005  27.89  Puzzles  Jigsaw Puzzles  Charleston South Carolina credit		

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

```
from pyspark.sql.types import StructType,StructField
from pyspark.sql.types import StringType, IntegerType, ArrayType
data = [
    ("James","","Smith"),["Java","Scala","C++"],"OH","M"),
    ("Anna","Rose","",["Spark","Java","C++"],"NY","F"),
    ("Julia","","Williams"),["CSharp","VB"],"OH","F"),
    ("Maria","Anne","Jones"),["CSharp","VB"],"NY","M"),
    ("Jen","Mary","Brown"),["CSharp","VB"],"NY","M"),
    ("Mike","Mary","Williams"),["Python","VB"],"OH","M")
]

schema = StructType([
    StructField('name', StructType([
        StructField('firstname', StringType(), True),
        StructField('middlename', StringType(), True),
        StructField('lastname', StringType(), True)
    ])),
    StructField('languages', ArrayType(StringType()), True),
    StructField('state', StringType(), True),
    StructField('gender', StringType(), True)
])

df = spark.createDataFrame(data = data, schema = schema)
df.printSchema()
df.show(truncate=False)
```

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

```
# Using equals condition
df.filter(df.state == "OH").show(truncate=False)

# not equals condition
df.filter(df.state != "OH").show(truncate=False)
df.filter(~(df.state == "OH")).show(truncate=False)

from pyspark.sql.functions import col
df.filter(col("state") == "OH").show(truncate=False)
```

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

```
[*]: #Using SQL Expression  
      df.filter("gender == 'M'").show()  
      #For not equal  
      df.filter("gender != 'M'").show()  
      df.filter("gender <> 'M'").show()
```

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

```
[29]: #Filter multiple condition  
df.filter( (df.state == "OH") & (df.gender == "M") ).show(truncate=False)
```

name	languages	state	gender
[James, , Smith]	[Java, Scala, C++]	OH	M
[Mike, Mary, Williams]	[Python, VB]	OH	M

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

```
[30]: #Filter IS IN List values
      li=["OH","CA","DE"]
      df.filter(df.state.isin(li)).show()

      # Filter NOT IS IN List values
      #These show all records with NY (NY is not part of the list)
      df.filter(~df.state.isin(li)).show()
      df.filter(df.state.isin(li)==False).show()
```

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

- You can also filter DataFrame rows by using startswith(), endswith() and contains() methods of Column class.

```
[31]: # Using startswith
df.filter(df.state.startswith("N")).show()

#using endswith
df.filter(df.state.endswith("H")).show()

#contains
df.filter(df.state.contains("H")).show()
```

```
+-----+-----+-----+
|       name|     languages|state|gender|
+-----+-----+-----+
| [Anna, Rose, ]|[Spark, Java, C++]|   NY|      F|
|[Maria, Anne, Jones]|      [CSharp, VB]|   NY|      M|
| [Jen, Mary, Brown]|      [CSharp, VB]|   NY|      M|
+-----+-----+-----+
```

```
+-----+-----+-----+
|       name|     languages|state|gender|
+-----+-----+-----+
| [James, , Smith]|[Java, Scala, C++]|   OH|      M|
|[Julia, , Williams]|      [CSharp, VB]|   OH|      F|
|[Mike, Mary, Will...|[Python, VB]|   OH|      M|
+-----+-----+-----+
```

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

```
[32]: data2 = [(2,"Michael Rose"),(3,"Robert Williams"),
      (4,"Rames Rose"),(5,"Rames rose")]
df2 = spark.createDataFrame(data = data2, schema = ["id","name"])

# Like - SQL LIKE pattern
df2.filter(df2.name.like("%rose%")).show()

# rlike - SQL RLIKE pattern (LIKE with Regex)
#This check case insensitive
df2.filter(df2.name.rlike("(?i)^*rose$")).show()
```

```
+---+-----+
| id|     name|
+---+-----+
|  5|Rames rose|
+---+-----+
```

```
+---+-----+
| id|     name|
+---+-----+
|  2|Michael Rose|
|  4|Rames Rose|
|  5|Rames rose|
+---+-----+
```

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

### Filter on an Array column

```
[33]: from pyspark.sql.functions import array_contains  
df.filter(array_contains(df.languages,"Java")).show(truncate=False)
```

name	languages	state	gender
[James, , Smith]	[Java, Scala, C++]	OH	M
[Anna, Rose, ]	[Spark, Java, C++]	NY	F

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

### Filtering on Nested Struct columns

```
[34]: #Struct condition  
df.filter(df.name.lastname == "Williams").show(truncate=False)
```

name	languages	state	gender
[Julia, , Williams]	[CSharp, VB]	OH	F
[Mike, Mary, Williams]	[Python, VB]	OH	M

# PySpark dataframe function

## Where Filter Function | Multiple Conditions

How about Where()?

pyspark.sql.DataFrame.where

`DataFrame.Where(condition)`

`where()` is an alias for `filter()`.

# PySpark dataframe function

Get Distinct Rows (By Comparing All Columns)

```
[45]: transDF.select('cust_id','game').count()
```

```
[45]: 60
```

```
[46]: transDF.select('cust_id','game').distinct().count()
```

```
[46]: 43
```

# PySpark dataframe function

## Distinct of Selected Multiple Columns

```
[47]: dropDisDF = transDF.dropDuplicates(['cust_id','game'])
print("Distinct count of customer ID & game : "+str(dropDisDF.count()))
dropDisDF.show(truncate=False)
```

Distinct count of customer ID & game : 43

trans_id	date	cust_id	amount	game	equipment	city	state	mode
13	03-13-2011	4000010	107.8	Team Sports	Field Hockey	Honolulu	Hawaii	credit
48	09-27-2011	4000007	157.94	Exercise & Fitness	Exercise Bands	Philadelphia	Pennsylvania	credit
20	06-29-2011	4000005	41.55	Exercise & Fitness	Weightlifting Belts	New Orleans	Louisiana	credit
33	06-15-2011	4000008	154.15	Outdoor Recreation	Lawn Games	Nashville	Tennessee	credit
49	07-12-2011	4000010	144.59	Jumping	Jumping Stilts	Cambridge	Massachusetts	credit
46	05-27-2011	4000001	52.29	Gymnastics	Vaulting Horses	Cleveland	Ohio	credit
55	12-16-2011	4000006	106.11	Water Sports	Swimming	New York	New York	credit
3	06-05-2011	4000003	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin	credit
6	10-28-2011	4000005	27.89	Puzzles	Jigsaw Puzzles	Charleston	South Carolina	credit
12	02-08-2011	4000009	41.52	Indoor Games	Bowling	San Francisco	California	credit
10	05-29-2011	4000007	180.28	Outdoor Recreation	Archery	Reno	Nevada	credit
47	10-23-2011	4000008	100.1	Outdoor Play Equipment	Swing Sets	Everett	Washington	credit
24	06-10-2011	4000003	151.2	Water Sports	Surfing	Plano	Texas	credit
52	02-04-2011	4000005	44.82	Outdoor Play Equipment	Lawn Water Slides	Hampton	Virginia	cash
15	10-20-2011	4000001	137.64	Combat Sports	Fencing	Honolulu	Hawaii	credit
43	04-22-2011	4000004	32.34	Water Sports	Water Polo	Las Vegas	Nevada	cash
16	05-28-2011	4000010	35.56	Exercise & Fitness	Free Weight Bars	Columbia	South Carolina	credit
14	02-25-2011	4000010	36.81	Gymnastics	Vaulting Horses	Los Angeles	California	credit
22	10-10-2011	4000009	19.64	Water Sports	Kitesurfing	Saint Paul	Minnesota	credit
7	07-14-2011	4000006	96.01	Outdoor Play Equipment	Sandboxes	Columbus	Ohio	credit

# PySpark dataframe function

## Sort()

```
transDF = spark.read.options(delimiter=',')\
.schema('trans_id INT, date STRING, cust_ID INT, amount DOUBLE, game STRING, equipment STRING, city STRING, state STRING, mode STRING')\
.csv("trans.txt")
```

```
transDF.show(2,truncate=False)
```

```
+-----+-----+-----+-----+-----+-----+-----+
|trans_id|date      |cust_ID|amount|game           |equipment          |city        |state       |mode   |
+-----+-----+-----+-----+-----+-----+-----+
|0      |06-26-2011|4000001|40.33 |Exercise & Fitness|Cardio Machine Accessories|Clarksville|Tennessee |credit|
|1      |05-26-2011|4000002|198.44|Exercise & Fitness|Weightlifting Gloves    |Long Beach |California|credit|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 2 rows
```

```
transDF.sort('amount').show(10,truncate=False)
```

```
+-----+-----+-----+-----+-----+-----+-----+
|trans_id|date      |cust_ID|amount|game           |equipment          |city        |state       |mode   |
+-----+-----+-----+-----+-----+-----+-----+
|31     |11-28-2011|4000008|5.03  |Games            |Dice & Dice Sets    |Los Angeles |California |credit|
|2      |06-01-2011|4000002|5.58  |Exercise & Fitness|Weightlifting Machine Accessories|Anaheim    |California |credit|
|8      |01-17-2011|4000006|10.44 |Winter Sports     |Snowmobiling      |Des Moines  |Iowa        |credit|
|22    |10-10-2011|4000009|19.64 |Water Sports      |Kitesurfing       |Saint Paul   |Minnesota  |credit|
|32    |01-29-2011|4000008|20.13 |Team Sports       |Soccer            |Springfield |Illinois   |credit|
|37    |04-19-2011|4000007|20.2   |Outdoor Recreation|Shooting Games    |San Diego   |California |credit|
|59    |11-07-2011|4000001|21.43 |Winter Sports     |Snowboarding      |Philadelphia|Pennsylvania |cash  |
|6      |10-28-2011|4000005|27.89 |Puzzles           |Jigsaw Puzzles    |Charleston  |South Carolina|credit|
|41    |04-16-2011|4000004|28.11 |Indoor Games      |Bowling           |Westminster |Colorado   |cash  |
|26    |10-11-2011|4000009|31.58 |Combat Sports     |Wrestling         |Orange      |California |credit|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 10 rows
```

# PySpark dataframe function

## groupBy() and aggregate functions

- **groupBy()** function is used to collect the identical data into groups on DataFrame and perform aggregate functions on the grouped data.
- **Aggregate functions** operate on a group of rows and calculate a single return value for every group:
  - approx\_count\_distinct
  - avg
  - collect\_list
  - collect\_set
  - countDistinct
  - count
  - grouping
  - first
  - last
  - kurtosis
  - max
  - min
  - mean
  - skewness
  - stddev
  - stddev\_samp
  - stddev\_pop
  - sum
  - sumDistinct
  - variance

# PySpark dataframe function

## groupBy() and aggregate functions

```
#count number of transactions of each user  
transDF.groupBy('cust_ID').count().show(5)
```

```
+-----+  
| cust_ID | count |  
+-----+  
| 4000009 |     6 |  
| 4000001 |     8 |  
| 4000006 |     5 |  
| 4000005 |     5 |  
| 4000008 |    10 |  
+-----+  
only showing top 5 rows
```

```
transDF.groupBy('cust_ID').min().show(5)
```

```
+-----+-----+-----+  
| cust_ID | min(trans_id) | min(cust_ID) | min(amount) |  
+-----+-----+-----+  
| 4000009 |          11 | 4000009 |      19.64 |  
| 4000001 |           0 | 4000001 |      21.43 |  
| 4000006 |           7 | 4000006 |      10.44 |  
| 4000005 |           6 | 4000005 |      27.89 |  
| 4000008 |          17 | 4000008 |       5.03 |  
+-----+-----+-----+  
only showing top 5 rows
```

```
: #show min amount of each user  
transDF.groupBy('cust_ID').min('amount').show(5)
```

```
+-----+-----+  
| cust_ID | min(amount) |  
+-----+-----+  
| 4000009 |      19.64 |  
| 4000001 |      21.43 |  
| 4000006 |      10.44 |  
| 4000005 |      27.89 |  
| 4000008 |       5.03 |  
+-----+-----+  
only showing top 5 rows
```

# PySpark dataframe function

## groupBy() and aggregate functions

- Aggregate function: `agg()`

```
from pyspark.sql import functions as f  
  
#show min amount of each user and rename the result column to min_transaction_amount  
transDF.groupBy('cust_ID').agg(f.min('amount').alias('min_transaction_amount')).show(5)
```

```
+-----+  
|cust_ID|min_transaction_amount|  
+-----+  
|4000009|           19.64|  
|4000001|           21.43|  
|4000006|           10.44|  
|4000005|           27.89|  
|4000008|            5.03|  
+-----+  
only showing top 5 rows
```

# PySpark dataframe function

## groupBy() and aggregate functions

- Aggregate function: `agg()`

```
#show sum amount of each user
transDF.groupBy('cust_ID').sum('amount').show(5)
```

```
+-----+-----+
|cust_ID|sum(amount)|
+-----+-----+
|4000009|      457.83|
|4000001|      651.05|
|4000006|      539.38|
|4000005|      325.15|
|4000008|      859.42|
+-----+-----+
only showing top 5 rows
```

```
#show sum amount of each user, rename result to 'total_amount'
transDF.groupBy('cust_ID').agg(f.sum('amount').alias('total_amount')).show(5)
```

```
+-----+-----+
|cust_ID|total_amount|
+-----+-----+
|4000009|      457.83|
|4000001|      651.05|
|4000006|      539.38|
|4000005|      325.15|
|4000008|      859.42|
+-----+-----+
only showing top 5 rows
```

# PySpark dataframe function

## groupBy() and aggregate functions

- Many aggregate function can be used only inside `agg()`

```
#aggregate function first() can only be used inside agg() function
transDF.groupBy('cust_ID').first('game').show(5,truncate=False)
```

```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-51-d85a43ff06b3> in <module>
      1 #aggregate function first() can only be used inside agg() function
----> 2 transDF.groupBy('cust_ID').first('game').show(5,truncate=False)

AttributeError: 'GroupedData' object has no attribute 'first'
```

```
#show first game of each user
transDF.groupBy('cust_ID').agg(f.first('game')).show(5,truncate=False)
```

cust_ID	first(game)
4000009	Outdoor Play Equipment
4000001	Exercise & Fitness
4000006	Outdoor Play Equipment
4000005	Puzzles
4000008	Water Sports

# PySpark dataframe function

## groupBy() and aggregate functions

- Many aggregate function can be used only inside `agg()`

```
transDF.groupBy('cust_ID').countDistinct('game').show(5)
```

```
-----
AttributeError                                 Traceback (most recent call last)
<ipython-input-52-96ba5f81a49c> in <module>
----> 1 transDF.groupBy('cust_ID').countDistinct('game').show(5)

AttributeError: 'GroupedData' object has no attribute 'countDistinct'
```

```
#count the number of game played by each user
transDF.groupBy('cust_ID').agg(f.countDistinct('game')).show(5)
```

cust_ID	count(game)
4000009	5
4000001	6
4000006	4
4000005	5
4000008	5

# PySpark dataframe function

## groupBy() and aggregate functions

- **collect\_list()** and **collect\_set()**

```
#show list of games played by each user
transDF.groupBy('cust_ID').agg(f.collect_list('game')).show(5,truncate=False)

+-----+
|cust_ID|collect_list(game)|
+-----+
|4000009|[Outdoor Play Equipment, Indoor Games, Water Sports, Gymnastics, Indoor Games, Combat Sports]|
|4000001|[Exercise & Fitness, Combat Sports, Outdoor Recreation, Water Sports, Water Sports, Exercise & Fitness, Gymnastics, Winter Sports]|
|4000006|[Outdoor Play Equipment, Winter Sports, Jumping, Outdoor Play Equipment, Water Sports]|
|4000005|[Puzzles, Exercise & Fitness, Air Sports, Team Sports, Outdoor Play Equipment]|
|4000008|[Water Sports, Team Sports, Water Sports, Team Sports, Games, Team Sports, Outdoor Recreation, Team Sports, Games, Outdoor Play Equipment]|
+-----+
only showing top 5 rows
```

```
#show list of distinct games played by each user
transDF.groupBy('cust_ID').agg(f.collect_set('game')).show(5,truncate=False)

+-----+
|cust_ID|collect_set(game)|
+-----+
|4000009|[Combat Sports, Water Sports, Indoor Games, Gymnastics, Outdoor Play Equipment]|
|4000001|[Combat Sports, Water Sports, Outdoor Recreation, Gymnastics, Winter Sports, Exercise & Fitness]|
|4000006|[Water Sports, Jumping, Winter Sports, Outdoor Play Equipment]|
|4000005|[Puzzles, Team Sports, Air Sports, Exercise & Fitness, Outdoor Play Equipment]|
|4000008|[Team Sports, Water Sports, Outdoor Recreation, Games, Outdoor Play Equipment]|
+-----+
only showing top 5 rows
```

# PySpark built-in function

- <https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html>

## Functions

<code>abs(col)</code>	Computes the absolute value.
<code>acos(col)</code>	New in version 1.4.0.
<code>acosh(col)</code>	Computes inverse hyperbolic cosine of the input column.
<code>add_months(start, months)</code>	Returns the date that is <i>months</i> months after <i>start</i>
<code>aggregate(col, initialValue, merge[, finish])</code>	Applies a binary operator to an initial state and all elements in the array, and reduces this to a single state.
<code>approxCountDistinct(col[, rsd])</code>	Deprecated since version 2.1.0.
<code>approx_count_distinct(col[, rsd])</code>	Aggregate function: returns a new <code>Column</code> for approximate distinct count of column <i>col</i> .
<code>array(*cols)</code>	Creates a new array column.
<code>array_contains(col, value)</code>	Collection function: returns null if the array is null, true if the array contains the given value, and false otherwise.
<code>array_distinct(col)</code>	Collection function: removes duplicate values from the array.
<code>array_except(col1, col2)</code>	Collection function: returns an array of the elements in <i>col1</i> but not in <i>col2</i> , without duplicates.
<code>array_intersect(col1, col2)</code>	Collection function: returns an array of the elements in the intersection of <i>col1</i> and <i>col2</i> , without duplicates.

<code>array_join(col, delimiter[, null_replacement])</code>	Concatenates the elements of <i>column</i> using the <i>delimiter</i> .
<code>array_max(col)</code>	Collection function: returns the maximum value of the array.
<code>array_min(col)</code>	Collection function: returns the minimum value of the array.
<code>array_position(col, value)</code>	Collection function: Locates the position of the first occurrence of the given value in the given array.
<code>array_remove(col, element)</code>	Collection function: Remove all elements that equal to <i>element</i> from the given array.
<code>array_repeat(col, count)</code>	Collection function: creates an array containing a column repeated <i>count</i> times.
<code>array_sort(col)</code>	Collection function: sorts the input array in ascending order.
<code>array_union(col1, col2)</code>	Collection function: returns an array of the elements in the union of <i>col1</i> and <i>col2</i> , without duplicates.
<code>arrays_overlap(a1, a2)</code>	Collection function: returns true if the arrays contain any common non-null element; if not, returns null if both the arrays are non-empty and any of them contains a null element; returns false otherwise.
<code>arrays_zip(*cols)</code>	Collection function: Returns a merged array of structs in which the N-th struct contains all N-th values of input arrays.
<code>asc(col)</code>	Returns a sort expression based on the ascending order of the given column name.
<code>asc_nulls_first(col)</code>	Returns a sort expression based on the ascending order of the given column name, and null values return before non-
<code>upper(col)</code>	Converts a string expression to upper case.
<code>var_pop(col)</code>	Aggregate function: returns the population variance of the values in a group.
<code>var_samp(col)</code>	Aggregate function: returns the unbiased sample variance of the values in a group.
<code>variance(col)</code>	Aggregate function: alias for <code>var_samp</code>
<code>weekofyear(col)</code>	Extract the week number of a given date as integer.
<code>when(condition, value)</code>	Evaluates a list of conditions and returns one of multiple possible result expressions.
<code>window(timeColumn, windowDuration[, ...])</code>	Bucketize rows into one or more time windows given a timestamp specifying column.
<code>xxhash64(*cols)</code>	Calculates the hash code of given columns using the 64-bit variant of the xxHash algorithm, and returns the result as a long column.
<code>year(col)</code>	Extract the year of a given date as integer.
<code>years(col)</code>	Partition transform function: A transform for timestamps and dates to partition data into years.
<code>zip_with(left, right, f)</code>	Merge two given arrays, element-wise, into a single array using a function.
<code>from_avro(data, jsonFormatSchema[, options])</code>	Converts a binary column of Avro format into its corresponding catalyst value.
<code>to_avro(data[, jsonFormatSchema])</code>	Converts a column into binary of avro format.

# PySpark built-in function

- array\_contains()

```
#show List of users who play Jumping
transDF.groupBy('cust_ID').agg(f.collect_set('game')).filter(f.array_contains(f.col('collect_set(game)'), 'Jumping')).show(5,truncate=False)

+-----+-----+
| cust_ID | collect_set(game) |
+-----+-----+
| 4000006 | [Water Sports, Jumping, Winter Sports, Outdoor Play Equipment] |
| 4000010 | [Team Sports, Jumping, Gymnastics, Games, Exercise & Fitness] |
+-----+-----+
```

# PySpark built-in function

- concat\_ws()

```
transDF.groupBy('cust_ID').agg(f.collect_set('game')).withColumn('game_list_string',f.concat_ws(',',f.col('collect_set(game))))).show(5,truncate=False)

+-----+-----+
|cust_ID|collect_set(game)           |game_list_string
+-----+-----+
|4000009|[Combat Sports, Water Sports, Indoor Games, Gymnastics, Outdoor Play Equipment] |Combat Sports,Water Sports,Indoor Games,Gymnastics,Outdoor Play Equipment
|4000001|[Combat Sports, Water Sports, Outdoor Recreation, Gymnastics, Winter Sports, Exercise & Fitness]|Combat Sports,Water Sports,Outdoor Recreation,Gymnastics,Winter Sports,Exercise & Fitness
|4000006|[Water Sports, Jumping, Winter Sports, Outdoor Play Equipment] |Water Sports,Jumping,Winter Sports,Outdoor Play Equipment
|4000005|[Puzzles, Team Sports, Air Sports, Exercise & Fitness, Outdoor Play Equipment] |Puzzles,Team Sports,Air Sports,Exercise & Fitness,Outdoor Play Equipment
|4000008|[Team Sports, Water Sports, Outdoor Recreation, Games, Outdoor Play Equipment] |Team Sports,Water Sports,Outdoor Recreation,Games,Outdoor Play Equipment
+-----+-----+
only showing top 5 rows
```

# PySpark built-in function

- `split()`

```
transGameStringDF = transDF.groupBy('cust_ID').agg(f.collect_set('game')).withColumn('game_string',f.concat_ws(',',f.col('collect_set(game)'))).select('cust_ID','game_string')
transGameStringDF.show(5,truncate=False)

transGameStringDF.withColumn('game_array',f.split('game_string','')).show(5,truncate=False)
```

```
+-----+-----+
|cust_ID|game_string
+-----+-----+
|4000009|Combat Sports,Water Sports,Indoor Games,Gymnastics,Outdoor Play Equipment
|4000001|Combat Sports,Water Sports,Outdoor Recreation,Gymnastics,Winter Sports,Exercise & Fitness
|4000006|Water Sports,Jumping,Winter Sports,Outdoor Play Equipment
|4000005|Puzzles,Team Sports,Air Sports,Exercise & Fitness,Outdoor Play Equipment
|4000008|Team Sports,Water Sports,Outdoor Recreation,Games,Outdoor Play Equipment
+-----+
only showing top 5 rows
```

```
+-----+-----+-----+
|cust_ID|game_string          |game_array
+-----+-----+-----+
|4000009|Combat Sports,Water Sports,Indoor Games,Gymnastics,Outdoor Play Equipment |[[Combat Sports, Water Sports, Indoor Games, Gymnastics, Outdoor Play Equipment]|
|4000001|Combat Sports,Water Sports,Outdoor Recreation,Gymnastics,Winter Sports,Exercise & Fitness |[[Combat Sports, Water Sports, Outdoor Recreation, Gymnastics, Winter Sports, Exercise & Fitness]|
|4000006|Water Sports,Jumping,Winter Sports,Outdoor Play Equipment |[[Water Sports, Jumping, Winter Sports, Outdoor Play Equipment]|
|4000005|Puzzles,Team Sports,Air Sports,Exercise & Fitness,Outdoor Play Equipment |[[Puzzles, Team Sports, Air Sports, Exercise & Fitness, Outdoor Play Equipment]|
|4000008|Team Sports,Water Sports,Outdoor Recreation,Games,Outdoor Play Equipment |[[Team Sports, Water Sports, Outdoor Recreation, Games, Outdoor Play Equipment]|
+-----+
only showing top 5 rows
```

# PySpark built-in function

- size()

```
transGameStringDF.withColumn('game_array',f.split('game_string','')).withColumn('num_game',f.size('game_array')).show(5,truncate=True)

+-----+-----+-----+
|cust_ID| game_string| game_array|num_game|
+-----+-----+-----+
|4000009|Combat Sports,Wat...|[Combat Sports, W...|      5|
|4000001|Combat Sports,Wat...|[Combat Sports, W...|      6|
|4000006|Water Sports,Jump...|[Water Sports, Ju...|      4|
|4000005|Puzzles,Team Spor...|[Puzzles, Team Sp...|      5|
|4000008|Team Sports,Water...|[Team Sports, Wat...|      5|
+-----+-----+-----+
only showing top 5 rows
```

# PySpark built-in function

- element\_at()

```
#show first and last game play by each user
transGameStringDF.withColumn('game_array',f.split('game_string',','))    \
.withColumn('first_game',f.element_at('game_array',1))                  \
.withColumn('last_game',f.element_at('game_array',-1)).show(5)
```

cust_ID	game_string	game_array	first_game	last_game
4000009	Combat Sports,Wat...	[Combat Sports, W...	Combat Sports	Outdoor Play Equi...
4000001	Combat Sports,Wat...	[Combat Sports, W...	Combat Sports	Exercise & Fitness
4000006	Water Sports,Jump...	[Water Sports, Ju...	Water Sports	Outdoor Play Equi...
4000005	Puzzles,Team Spor...	[Puzzles, Team Sp...	Puzzles	Outdoor Play Equi...
4000008	Team Sports,Water...	[Team Sports, Wat...	Team Sports	Outdoor Play Equi...

only showing top 5 rows

# PySpark built-in function

- explode()

```
transGameStringDF.withColumn('game_array',f.split('game_string','')).withColumn('single_game',f.explode('game_array')).show(10)
```

cust_ID	game_string	game_array	single_game
4000009	Combat Sports,Wat...	[Combat Sports, W...]	Combat Sports
4000009	Combat Sports,Wat...	[Combat Sports, W...]	Water Sports
4000009	Combat Sports,Wat...	[Combat Sports, W...]	Indoor Games
4000009	Combat Sports,Wat...	[Combat Sports, W...]	Gymnastics
4000009	Combat Sports,Wat...	[Combat Sports, W...]	Outdoor Play Equi...
4000001	Combat Sports,Wat...	[Combat Sports, W...]	Combat Sports
4000001	Combat Sports,Wat...	[Combat Sports, W...]	Water Sports
4000001	Combat Sports,Wat...	[Combat Sports, W...]	Outdoor Recreation
4000001	Combat Sports,Wat...	[Combat Sports, W...]	Gymnastics
4000001	Combat Sports,Wat...	[Combat Sports, W...]	Winter Sports

# PySpark built-in function

- `substring()`

```
transDF.show(5)
transDF.withColumn('day_moth',f.substring('date',0,5)).show(5)

+-----+-----+-----+-----+-----+-----+-----+
|trans_id|    date|cust_ID|amount|      game|   equipment|     city|   state| mode|
+-----+-----+-----+-----+-----+-----+-----+
|  0|06-26-2011|4000001| 40.33|Exercise & Fitness|Cardio Machine Ac...|Clarksville| Tennessee|credit|
|  1|05-26-2011|4000002|198.44|Exercise & Fitness|Weightlifting Gloves| Long Beach| California|credit|
|  2|06-01-2011|4000002|  5.58|Exercise & Fitness|Weightlifting Mac...| Anaheim| California|credit|
|  3|06-05-2011|4000003|198.19|      Gymnastics|Gymnastics Rings| Milwaukee| Wisconsin|credit|
|  4|12-17-2011|4000002| 98.81|    Team Sports|       Field Hockey|Nashville | Tennessee|credit|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

+-----+-----+-----+-----+-----+-----+-----+-----+
|trans_id|    date|cust_ID|amount|      game|   equipment|     city|   state| mode|day_moth|
+-----+-----+-----+-----+-----+-----+-----+-----+
|  0|06-26-2011|4000001| 40.33|Exercise & Fitness|Cardio Machine Ac...|Clarksville| Tennessee|credit| 06-26|
|  1|05-26-2011|4000002|198.44|Exercise & Fitness|Weightlifting Gloves| Long Beach| California|credit| 05-26|
|  2|06-01-2011|4000002|  5.58|Exercise & Fitness|Weightlifting Mac...| Anaheim| California|credit| 06-01|
|  3|06-05-2011|4000003|198.19|      Gymnastics|Gymnastics Rings| Milwaukee| Wisconsin|credit| 06-05|
|  4|12-17-2011|4000002| 98.81|    Team Sports|       Field Hockey|Nashville | Tennessee|credit| 12-17|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

# PySpark column function

- alias()

```
transDF.select(f.col('date')).show(5)
transDF.select(f.col('date').alias('day_month_year')).show(5)
```

```
+-----+
|      date|
+-----+
|06-26-2011|
|05-26-2011|
|06-01-2011|
|06-05-2011|
|12-17-2011|
+-----+
only showing top 5 rows
```

```
+-----+
|day_month_year|
+-----+
|    06-26-2011|
|    05-26-2011|
|    06-01-2011|
|    06-05-2011|
|    12-17-2011|
+-----+
only showing top 5 rows
```

# PySpark column function

- `isIn()`

```
transDF.withColumn('moth_in_05_06',f.substring('date',0,2).isin(['05','06'])).show(5)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|trans_id| date|cust_ID|amount| game| equipment| city| state| mode|moth_in_05_06|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0|06-26-2011|4000001| 40.33|Exercise & Fitness|Cardio Machine Ac...|[Clarksville| Tennessee|credit| true|
| 1|05-26-2011|4000002|198.44|Exercise & Fitness|Weightlifting Gloves| Long Beach|California|credit| true|
| 2|06-01-2011|4000002| 5.58|Exercise & Fitness|Weightlifting Mac...|[Anaheim|California|credit| true|
| 3|06-05-2011|4000003|198.19| Gymnastics| Gymnastics Rings| Milwaukee| Wisconsin|credit| true|
| 4|12-17-2011|4000002| 98.81| Team Sports| Field Hockey|Nashville | Tennessee|credit| false|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
transDF.filter(f.substring('date',0,2).isin(['05','06'])).show(5)
```

```
+-----+-----+-----+-----+-----+-----+-----+
|trans_id| date|cust_ID|amount| game| equipment| city| state| mode|
+-----+-----+-----+-----+-----+-----+-----+
| 0|06-26-2011|4000001| 40.33|Exercise & Fitness|Cardio Machine Ac...|[Clarksville| Tennessee|credit|
| 1|05-26-2011|4000002|198.44|Exercise & Fitness|Weightlifting Gloves| Long Beach|California|credit|
| 2|06-01-2011|4000002| 5.58|Exercise & Fitness|Weightlifting Mac...|[Anaheim|California|credit|
| 3|06-05-2011|4000003|198.19| Gymnastics| Gymnastics Rings| Milwaukee| Wisconsin|credit|
| 9|05-17-2011|4000006|152.46| Jumping| Bungee Jumping|St. Petersburg| Florida|credit|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

# PySpark column function

- `cast()`

```
transDF.withColumn('moth',f.substring('date',0,2)).printSchema()
transDF.withColumn('moth',f.substring('date',0,2).cast('int')).printSchema()

root
|-- trans_id: integer (nullable = true)
|-- date: string (nullable = true)
|-- cust_ID: integer (nullable = true)
|-- amount: double (nullable = true)
|-- game: string (nullable = true)
|-- equipment: string (nullable = true)
|-- city: string (nullable = true)
|-- state: string (nullable = true)
|-- mode: string (nullable = true)
|-- moth: string (nullable = true)

root
|-- trans_id: integer (nullable = true)
|-- date: string (nullable = true)
|-- cust_ID: integer (nullable = true)
|-- amount: double (nullable = true)
|-- game: string (nullable = true)
|-- equipment: string (nullable = true)
|-- city: string (nullable = true)
|-- state: string (nullable = true)
|-- mode: string (nullable = true)
|-- moth: integer (nullable = true)
```

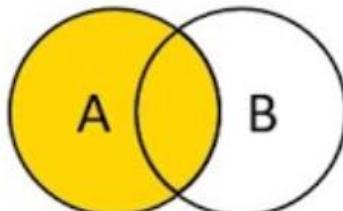
# PySpark column function

The screenshot shows a web browser displaying the Apache Spark API Reference for Python. The URL is [spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html](https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html). The page title is "Column APIs". The left sidebar contains links for "Spark SQL", "Structured Streaming", "MLlib (DataFrame-based)", "Spark Streaming", "MLlib (RDD-based)", "Spark Core", and "Resource Management". The main content area lists various column functions with their descriptions. A sidebar on the right lists "On this page" categories: Core Classes, Spark Session APIs, Configuration, Input and Output, DataFrame APIs, Column APIs (which is selected), Data Types, Row, Functions, Window, and Grouping.

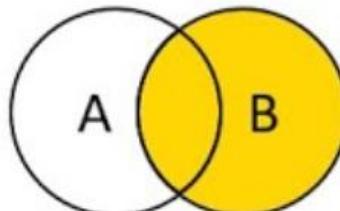
Method	Description
<code>column.alias(*alias, **kwargs)</code>	Returns this column aliased with a new name or names (in the case of expressions that return more than one column, such as <code>explode</code> ).
<code>column.asc()</code>	Returns a sort expression based on ascending order of the column.
<code>column.asc_nulls_first()</code>	Returns a sort expression based on ascending order of the column, and null values return before non-null values.
<code>column.asc_nulls_last()</code>	Returns a sort expression based on ascending order of the column, and null values appear after non-null values.
<code>column.astype(dataType)</code>	<code>astype()</code> is an alias for <code>cast()</code> .
<code>column.between(lowerBound, upperBound)</code>	A boolean expression that is evaluated to true if the value of this expression is between the given columns.
<code>column.bitwiseAND(other)</code>	Compute bitwise AND of this expression with another expression.
<code>column.bitwiseOR(other)</code>	Compute bitwise OR of this expression with another expression.
<code>column.bitwiseXOR(other)</code>	Compute bitwise XOR of this expression with another expression.
<code>column.cast(dataType)</code>	Convert the column into type <code>dataType</code> .
<code>column.contains(other)</code>	Contains the other element.

# PySpark SQL JOIN

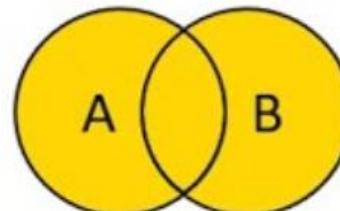
## JOIN Types



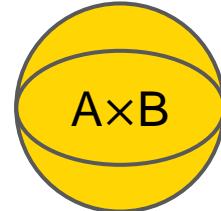
Left Outer



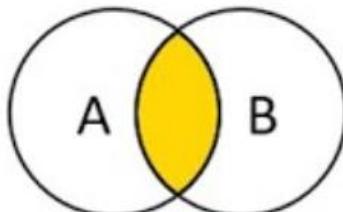
Right Outer



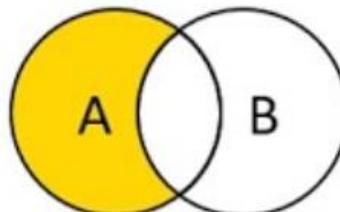
Full Outer



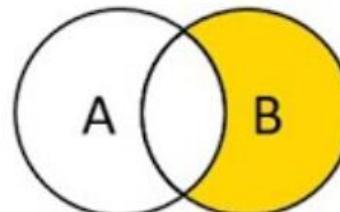
Cross



Inner



Left Anti



Right Anti

# PySpark SQL JOIN

## pyspark.sql.DataFrame.join

`DataFrame.join(other, on=None, how=None)`

[source]

Joins with another `DataFrame`, using the given join expression.

*New in version 1.3.0.*

**Parameters:** `other : DataFrame`

Right side of the join

`on : str, list or column, optional`

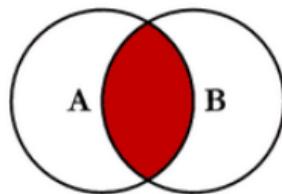
a string for the join column name, a list of column names, a join expression (Column), or a list of Columns. If `on` is a string or a list of strings indicating the name of the join column(s), the column(s) must exist on both sides, and this performs an equi-join.

`how : str, optional`

default `inner`. Must be one of: `inner`, `cross`, `outer`, `full`, `fullouter`, `full_outer`, `left`, `leftouter`, `left_outer`, `right`, `rightouter`, `right_outer`, `semi`, `leftsemi`, `left_semi`, `anti`, `leftanti` and `left_anti`.

# PySpark SQL JOIN

Inner Join in pyspark is the simplest and most common type of join. It is also known as simple join or Natural Join. Inner join returns the rows when matching condition is met.

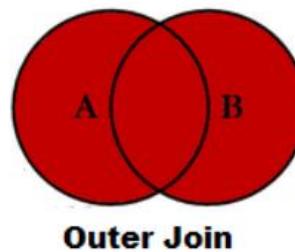


**Inner Join**

```
1 | ### Inner join in pyspark
2 |
3 | df_inner = df1.join(df2, on=['Roll_No'], how='inner')
4 | df_inner.show()
```

# PySpark SQL JOIN

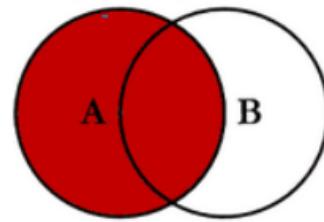
outer Join in pyspark combines the results of both left and right outer **joins**. The joined table will contain all records from both the tables



```
1 | ### Outer join in pyspark
2 |
3 | df_outer = df1.join(df2, on=['Roll_No'], how='outer')
4 | df_outer.show()
```

# PySpark SQL JOIN

The **LEFT JOIN in pyspark** returns all records from the **left** dataframe (A), and the matched records from the right dataframe (B)

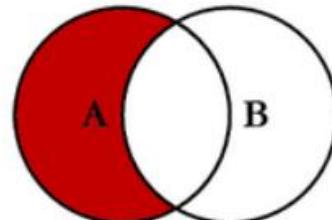


**Left join**

```
1 | ### Left join in pyspark
2 |
3 | df_left = df1.join(df2, on=['Roll_No'], how='left')
4 | df_left.show()
```

# PySpark SQL JOIN

This is like inner join, with only the left dataframe columns and values are selected

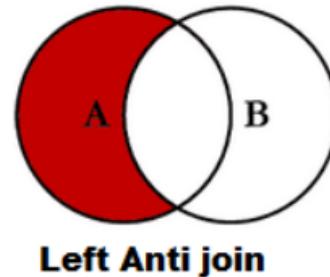


**Left semi join**

```
1 | ### Left Semi join in pyspark
2 |
3 | df_left_semi = df1.join(df2, on=['Roll_No'], how='left_semi')
4 | df_left_semi.show()
```

# PySpark SQL JOIN

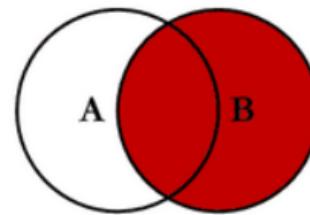
This join is like df1-df2, as it selects all rows from df1 that are not present in df2.



```
1 | ### Left Anti join in pyspark
2 |
3 | df_left_anti = df1.join(df2, on=['Roll_No'], how='left_anti')
4 | df_left_anti.show()
```

# PySpark SQL JOIN

The **RIGHT JOIN** in pyspark returns all records from the **right** dataframe (B), and the matched records from the left dataframe (A)

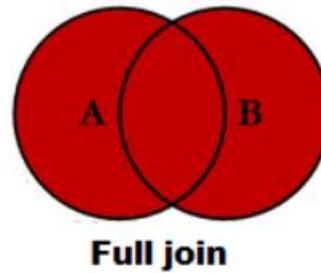


**Right Join**

```
1 | ### Right join in pyspark
2 |
3 | df_right = df1.join(df2, on=['Roll_No'], how='right')
4 | df_right.show()
```

# PySpark SQL JOIN

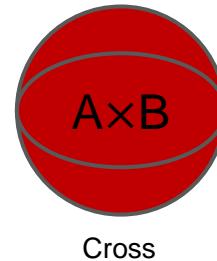
Full Join in pyspark combines the results of both left and right outer **joins**. The joined table will contain all records from both the tables



```
1 , how='full')  
2 df_full.show()
```

# PySpark SQL JOIN

- Cross join returns the cartesian product with another DataFrame.



# PySpark function

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html>

# PySpark SLQ tutorial 1

```
[3]: import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql import functions as f

spark = SparkSession.builder.appName("PySparkTutorial").getOrCreate()
```

# PySpark SLQ tutorial 1

To be able to perform comparison on the timestamp, we need to convert its data type from string to timestamp type. The third transformation will modify the values in the timestamp column using the values from this very column. We use the function `to_timestamp` to convert a string to timestamp data type.

```
ratings = (
    spark.read.csv(
        path="ratings_small.csv",
        sep=",",
        header=True,
        quote='"',
        schema="userId INT, movieId INT, rating DOUBLE, timestamp INT",
    )
    #.withColumnRenamed("timestamp", "timestamp_unix")
    #.withColumn("timestamp", f.from_unixtime("timestamp_unix"))
    #.withColumn("timestamp", f.to_timestamp("timestamp"))
)
ratings.show(5)
ratings.printSchema()

+-----+-----+-----+
|userId|movieId|rating|timestamp|
+-----+-----+-----+
| 1 | 1 | 4.0 | 964982703 |
| 1 | 3 | 4.0 | 964981247 |
| 1 | 6 | 4.0 | 964982224 |
| 1 | 47 | 5.0 | 964983815 |
| 1 | 50 | 5.0 | 964982931 |
+-----+-----+-----+
only showing top 5 rows

root
|-- userId: integer (nullable = true)
|-- movieId: integer (nullable = true)
|-- rating: double (nullable = true)
|-- timestamp: integer (nullable = true)
```

# PySpark SLQ tutorial 1

You can create the new column "timestamp" using the column "timestamp\_unix" and covert it to timestamp type in a single command.

```
ratings = (
    spark.read.csv(
        path="ratings_small.csv",
        sep=",",
        header=True,
        quote='',
        schema="userId INT, movieId INT, rating DOUBLE, timestamp INT",
    )
    .withColumnRenamed("timestamp", "timestamp_unix")
    .withColumn("timestamp", f.to_timestamp(f.from_unixtime("timestamp_unix")))
)
```

```
ratings.show(5)
ratings.printSchema()
```

```
+----+-----+-----+-----+-----+
|userId|movieId|rating|timestamp_unix|      timestamp|
+----+-----+-----+-----+-----+
|   1|     1|  4.0| 964982703|2000-07-31 01:45:03|
|   1|     3|  4.0| 964981247|2000-07-31 01:20:47|
|   1|     6|  4.0| 964982224|2000-07-31 01:37:04|
|   1|    47|  5.0| 964983815|2000-07-31 02:03:35|
|   1|    50|  5.0| 964982931|2000-07-31 01:48:51|
+----+-----+-----+-----+
only showing top 5 rows
```

```
root
|-- userId: integer (nullable = true)
|-- movieId: integer (nullable = true)
|-- rating: double (nullable = true)
|-- timestamp_unix: integer (nullable = true)
|-- timestamp: timestamp (nullable = true)
```

# PySpark SLQ tutorial 1

```
: #lets try to drop a column
#it's ok to add some collumnns which don't exist

ratings.drop("timestamp_unix", "foobar").show(5)

+-----+-----+-----+-----+
|userId|movieId|rating|      timestamp|
+-----+-----+-----+-----+
|     1|      1|   4.0|2000-07-31 01:45:03|
|     1|      3|   4.0|2000-07-31 01:20:47|
|     1|      6|   4.0|2000-07-31 01:37:04|
|     1|     47|   5.0|2000-07-31 02:03:35|
|     1|     50|   5.0|2000-07-31 01:48:51|
+-----+-----+-----+-----+
only showing top 5 rows
```

# PySpark SLQ tutorial 1

```
#count the number of review of each user, sorted by userId  
ratings.groupBy("userId").count().sort("userId").show(5)
```

```
+-----+-----+  
|userId|count|  
+-----+-----+  
|     1|  232|  
|     2|   29|  
|     3|   39|  
|     4|  216|  
|     5|   44|  
+-----+-----+  
only showing top 5 rows
```

# PySpark SLQ tutorial 1

```
: #show min rating of each user
ratings.groupBy("userId").agg(f.min("rating").alias("min_rating")).sort("userId").show(5)

+-----+-----+
|userId|min_rating|
+-----+-----+
|      1|        1.0|
|      2|        2.0|
|      3|        0.5|
|      4|        1.0|
|      5|        1.0|
+-----+-----+
only showing top 5 rows
```

# PySpark SLQ tutorial 2

```
import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql import functions as f

spark = SparkSession.builder.appName("PySparkTutorial").getOrCreate()
```

```
movies = (
    spark.read.csv(
        path="movies_small.csv",
        sep=",",
        header=True,
        quote="'",
        schema="movieId INT, title STRING, genres STRING",
    )
)
movies.show(5, truncate=False)
movies.printSchema()
```

# PySpark SLQ tutorial 2

```
movies.show(5, truncate=False)
movies.printSchema()
```

```
+-----+-----+
|movieId|title                |genres
+-----+-----+
|1      |Toy Story (1995)       |Adventure|Animation|Children|Comedy|Fantasy|
|2      |Jumanji (1995)          |Adventure|Children|Fantasy
|3      |Grumpier Old Men (1995)|Comedy|Romance
|4      |Waiting to Exhale (1995)|Comedy|Drama|Romance
|5      |Father of the Bride Part II (1995)|Comedy
+-----+
only showing top 5 rows
```

```
root
 |-- movieId: integer (nullable = true)
 |-- title: string (nullable = true)
 |-- genres: string (nullable = true)
```

# PySpark SLQ tutorial 2

```
: movies.where(f.col("genres") == "Action").show(5, False)
movies.where("genres == 'Action'").show(5, False)
```

```
+-----+-----+-----+
|movieId|title                                |genres|
+-----+-----+-----+
|9      |Sudden Death (1995)                  |Action|
|71     |Fair Game (1995)                     |Action|
|204    |Under Siege 2: Dark Territory (1995) |Action|
|251    |Hunted, The (1995)                   |Action|
|667    |Bloodsport 2 (a.k.a. Bloodsport II: The Next Kumite) (1996)|Action|
+-----+-----+-----+
only showing top 5 rows
```

# PySpark SLQ tutorial 2

```
#convert genres string to genres array and store to new column
movies.withColumn("genres_array",f.split(f.col("genres"),"\|")).show(5,False)
```

```
+-----+-----+-----+
|movieId|title           |genres          |genres array      |
+-----+-----+-----+
|1     |Toy Story (1995) |Adventure|Animation|Children|Comedy|Fantasy|[Adventure, Animation, Children, Comedy, Fantasy]|
|2     |Jumanji (1995)   |Adventure|Children|Fantasy          |[Adventure, Children, Fantasy]|
|3     |Grumpier Old Men (1995)|Comedy|Romance          |[Comedy, Romance]|
|4     |Waiting to Exhale (1995)|Comedy|Drama|Romance          |[Comedy, Drama, Romance]|
|5     |Father of the Bride Part II (1995)|Comedy          |[Comedy]|
+-----+-----+-----+
only showing top 5 rows
```

# PySpark SLQ tutorial 2

```
#use explode function to get a new row for each element in the genres_array
movies.withColumn("genres_array", f.split("genres", "\|")).withColumn("genre", f.explode("genres_array")).show(15, False)
```

movieId	title	genres	genres_array	genre
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	[Adventure, Animation, Children, Comedy, Fantasy]	Adventure
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	[Adventure, Animation, Children, Comedy, Fantasy]	Animation
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	[Adventure, Animation, Children, Comedy, Fantasy]	Children
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	[Adventure, Animation, Children, Comedy, Fantasy]	Comedy
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	[Adventure, Animation, Children, Comedy, Fantasy]	Fantasy
2	Jumanji (1995)	Adventure Children Fantasy	[[Adventure, Children, Fantasy]]	Adventure
2	Jumanji (1995)	Adventure Children Fantasy	[[Adventure, Children, Fantasy]]	Children
2	Jumanji (1995)	Adventure Children Fantasy	[[Adventure, Children, Fantasy]]	Fantasy
3	Grumpier Old Men (1995)	Comedy Romance	[[Comedy, Romance]]	Comedy
3	Grumpier Old Men (1995)	Comedy Romance	[[Comedy, Romance]]	Romance
4	Waiting to Exhale (1995)	Comedy Drama Romance	[[Comedy, Drama, Romance]]	Comedy
4	Waiting to Exhale (1995)	Comedy Drama Romance	[[Comedy, Drama, Romance]]	Drama
4	Waiting to Exhale (1995)	Comedy Drama Romance	[[Comedy, Drama, Romance]]	Romance
5	Father of the Bride Part II (1995)	Comedy	[[Comedy]]	Comedy
6	Heat (1995)	Action Crime Thriller	[[Action, Crime, Thriller]]	Action

only showing top 15 rows

# PySpark SLQ tutorial 2

```
#show final listed genres of each movie
movies.withColumn("genres_array", f.split("genres", "\|")).withColumn("last_genre", f.element_at("genres_array", -1)).show(15, False)
```

movieId	title	genres	genres_array	last_genre
1	Toy Story (1995)	[Adventure Animation Children Comedy Fantasy]	[Adventure, Animation, Children, Comedy, Fantasy]	Fantasy
2	Jumanji (1995)	[Adventure Children Fantasy]	[Adventure, Children, Fantasy]	Fantasy
3	Grumpier Old Men (1995)	[Comedy Romance]	[Comedy, Romance]	Romance
4	Waiting to Exhale (1995)	[Comedy Drama Romance]	[Comedy, Drama, Romance]	Romance
5	Father of the Bride Part II (1995)	[Comedy]	[Comedy]	Comedy
6	Heat (1995)	[Action Crime Thriller]	[Action, Crime, Thriller]	Thriller
7	Sabrina (1995)	[Comedy Romance]	[Comedy, Romance]	Romance
8	Tom and Huck (1995)	[Adventure Children]	[Adventure, Children]	Children
9	Sudden Death (1995)	[Action]	[Action]	Action
10	GoldenEye (1995)	[Action Adventure Thriller]	[Action, Adventure, Thriller]	Thriller
11	American President, The (1995)	[Comedy Drama Romance]	[Comedy, Drama, Romance]	Romance
12	Dracula: Dead and Loving It (1995)	[Comedy Horror]	[Comedy, Horror]	Horror
13	Balto (1995)	[Adventure Animation Children]	[Adventure, Animation, Children]	Children
14	Nixon (1995)	[Drama]	[Drama]	Drama
15	Cutthroat Island (1995)	[Action Adventure Romance]	[Action, Adventure, Romance]	Romance

only showing top 15 rows

# PySpark SLQ tutorial 3

```
links = spark.read.csv(  
    path = "links_small.csv",  
    sep=",",  
    header=True,  
    quote='',  
    inferSchema=True,  
    schema="movieId INT, imdbId STRING, tmdbId INT",  
)  
tags = spark.read.csv(  
    path="tags_small.csv",  
    sep=",",  
    header=True,  
    quote='',  
    inferSchema=True,  
    schema="userId INT, movieId INT,tag STRING, timestamp INT",  
).withColumn("timestamp",f.to_timestamp(f.from_unixtime("timestamp")))
```

# PySpark SLQ tutorial 3

```
links.show(5, True)  
links.printSchema()
```

```
+-----+-----+  
|movieId|imdbId|tmdbId|  
+-----+-----+  
| 1|0114709| 862|  
| 2|0113497| 8844|  
| 3|0113228| 15602|  
| 4|0114885| 31357|  
| 5|0113041| 11862|  
+-----+-----+  
only showing top 5 rows
```

```
root  
|-- movieId: integer (nullable = true)  
|-- imdbId: string (nullable = true)  
|-- tmdbId: integer (nullable = true)
```

```
tags.show(5, True)  
tags.printSchema()
```

```
+-----+-----+-----+  
|userId|movieId| tag| timestamp|  
+-----+-----+-----+  
| 2| 60756| funny|2015-10-25 02:29:54|  
| 2| 60756| Highly quotable|2015-10-25 02:29:56|  
| 2| 60756| will ferrell|2015-10-25 02:29:52|  
| 2| 89774| Boxing story|2015-10-25 02:33:27|  
| 2| 89774| MMA|2015-10-25 02:33:20|  
+-----+-----+-----+  
only showing top 5 rows
```

```
root  
|-- userId: integer (nullable = true)  
|-- movieId: integer (nullable = true)  
|-- tag: string (nullable = true)  
|-- timestamp: timestamp (nullable = true)
```

# PySpark SLQ tutorial 3

```
opinions = movies.join(tags, movies["movieId"] == tags["movieId"])
opinions.show(10)
```

movieId	title	genres	userId	movieId	tag	timestamp
1	Toy Story (1995)	Adventure Animation Children Comedy	567	1	fun	2018-05-03 01:33:33
1	Toy Story (1995)	Adventure Animation Children Comedy	474	1	pixar	2006-01-14 09:47:05
1	Toy Story (1995)	Adventure Animation Children Comedy	336	1	pixar	2006-02-04 16:36:04
2	Jumanji (1995)	Adventure Children Comedy Romance	474	2	game	2006-01-16 08:39:12
2	Jumanji (1995)	Adventure Children Comedy Romance	62	2	Robin Williams	2018-06-13 05:51:47
2	Jumanji (1995)	Adventure Children Comedy Romance	62	2	magic board game	2018-06-13 05:52:12
2	Jumanji (1995)	Adventure Children Comedy Romance	62	2	fantasy	2018-06-13 05:52:09
3	Grumpier Old Men ...	Comedy Romance	289	3	old	2006-03-27 09:01:00
3	Grumpier Old Men ...	Comedy Romance	289	3	moldy	2006-03-27 09:01:00
5	Father of the Bride ...	Comedy	474	5	remake	2006-01-16 08:11:43

# PySpark SLQ tutorial 3

```
#the following command will result in error (two movieId in the dataframe)
```

```
#opinions.select("movieId")
```

```
#you need to change the code
```

```
movies.join(tags,[ "movieId"], "inner").show(5)
movies.join(tags,[ "movieId"], "outer").sort("movieId").show(5)
movies.join(tags,[ "movieId"], "left").show(5)
movies.join(tags,[ "movieId"], "right").sort("movieId").show(5)
```

```
+-----+-----+-----+-----+-----+
|movieId|      title|      genres|userId|      tag|      timestamp|
+-----+-----+-----+-----+-----+
|     1|Toy Story (1995)|Adventure|Animati...|    567|      fun|2018-05-03 01:33:33|
|     1|Toy Story (1995)|Adventure|Animati...|    474|  pixar|2006-01-14 09:47:05|
|     1|Toy Story (1995)|Adventure|Animati...|    336|  pixar|2006-02-04 16:36:04|
|     2| Jumanji (1995)|Adventure|Childre...|    474|    game|2006-01-16 08:39:12|
|     2| Jumanji (1995)|Adventure|Childre...|     62|Robin Williams|2018-06-13 05:51:47|
+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
+-----+-----+-----+-----+-----+
|movieId|      title|      genres|userId|      tag|      timestamp|
+-----+-----+-----+-----+-----+
|     1|Toy Story (1995)|Adventure|Animati...|    567|      fun|2018-05-03 01:33:33|
|     1|Toy Story (1995)|Adventure|Animati...|    474|  pixar|2006-01-14 09:47:05|
|     1|Toy Story (1995)|Adventure|Animati...|    336|  pixar|2006-02-04 16:36:04|
|     2| Jumanji (1995)|Adventure|Childre...|    474|    game|2006-01-16 08:39:12|
|     2| Jumanji (1995)|Adventure|Childre...|     62|fantasy|2018-06-13 05:52:09|
+-----+-----+-----+-----+-----+
only showing top 5 rows
```

# PySpark SLQ tutorial 3

```
movies.join(tags,[ "movieId"], "inner").join(ratings,[ "movieId", "userId"]).show(10)
```

movieId	userId	title	genres	tag	timestamp	rating	timestamp
1	567	Toy Story (1995)	Adventure Animation Children Comedy	fun	2018-05-03 01:33:33	3.5	2018-05-03 01:33:21
1	474	Toy Story (1995)	Adventure Animation Children Comedy	pixar	2006-01-14 09:47:05	4.0	2001-01-04 09:36:00
1	336	Toy Story (1995)	Adventure Animation Children Comedy	pixar	2006-02-04 16:36:04	4.0	2005-07-25 00:48:49
2	474	Jumanji (1995)	Adventure Children Family Science Fiction	game	2006-01-16 08:39:12	3.0	2003-03-06 00:53:34
2	62	Jumanji (1995)	Adventure Children Family Science Fiction	Robin Williams	2018-06-13 05:51:47	4.0	2018-06-13 05:51:30
2	62	Jumanji (1995)	Adventure Children Family Science Fiction	magic board game	2018-06-13 05:52:12	4.0	2018-06-13 05:51:30
2	62	Jumanji (1995)	Adventure Children Family Science Fiction	fantasy	2018-06-13 05:52:09	4.0	2018-06-13 05:51:30
3	289	Grumpier Old Men ...	Comedy Romance	old	2006-03-27 09:01:00	2.5	2006-03-27 08:57:37
3	289	Grumpier Old Men ...	Comedy Romance	moldy	2006-03-27 09:01:00	2.5	2006-03-27 08:57:37
5	474	Father of the Bride (1991)	Comedy	remake	2006-01-16 08:11:43	1.5	2003-05-16 01:06:22

only showing top 10 rows

# PySpark SLQ tutorial 3

```
movies.join(tags,[ "movieId"], "inner").withColumnRenamed("timestamp","tag_timestamp").join(ratings,[ "movieId", "userId"]).show(10)
```

movieId	userId	title	genres	tag	tag_timestamp	rating	timestamp
1	567	Toy Story (1995)	Adventure Animation Children Comedy	fun	2018-05-03 01:33:33	3.5	2018-05-03 01:33:21
1	474	Toy Story (1995)	Adventure Animation Children Comedy	pixar	2006-01-14 09:47:05	4.0	2001-01-04 09:36:00
1	336	Toy Story (1995)	Adventure Animation Children Comedy	pixar	2006-02-04 16:36:04	4.0	2005-07-25 00:48:49
2	474	Jumanji (1995)	Adventure Children Comedy	game	2006-01-16 08:39:12	3.0	2003-03-06 00:53:34
2	62	Jumanji (1995)	Adventure Children Comedy	Robin Williams	2018-06-13 05:51:47	4.0	2018-06-13 05:51:30
2	62	Jumanji (1995)	Adventure Children Comedy	magic board game	2018-06-13 05:52:12	4.0	2018-06-13 05:51:30
2	62	Jumanji (1995)	Adventure Children Comedy	fantasy	2018-06-13 05:52:09	4.0	2018-06-13 05:51:30
3	289	Grumpier Old Men ...	Comedy Romance	old	2006-03-27 09:01:00	2.5	2006-03-27 08:57:37
3	289	Grumpier Old Men ...	Comedy Romance	moldy	2006-03-27 09:01:00	2.5	2006-03-27 08:57:37
5	474	Father of the Bride ...	Comedy	remake	2006-01-16 08:11:43	1.5	2003-05-16 01:06:22

only showing top 10 rows

# PySpark SLQ tutorial 3

```
: ratings.groupBy("movieId").agg(  
    f.count("*"),  
    f.min("rating"),  
    f.max("rating"),  
    f.avg("rating"),  
    f.min("timestamp"),  
    f.max("timestamp"),  
) .show(10)
```

movieId	count(1)	min(rating)	max(rating)	avg(rating)	min(timestamp)	max(timestamp)
1580	165	0.5	5.0	3.4878787878788	1997-07-07 19:07:18	2018-07-22 20:30:52
2366	25	1.5	5.0	3.64	1999-11-04 22:23:49	2018-02-20 17:20:35
3175	75	1.0	5.0	3.58	1999-12-26 21:01:31	2018-06-25 12:07:19
1088	42	1.0	5.0	3.369047619047619	1997-04-07 14:36:08	2018-01-17 08:52:47
32460	4	3.5	5.0	4.25	2011-12-19 02:21:21	2017-04-22 03:12:30
44022	23	1.0	4.5	3.217391304347826	2006-10-26 01:02:59	2018-03-07 14:38:56
96488	4	4.0	4.5	4.25	2014-11-08 23:17:07	2018-04-02 06:12:59
1238	9	3.0	5.0	4.055555555555555	1997-05-31 04:00:50	2013-06-02 06:27:29
1342	11	1.0	4.0	2.5	2000-08-08 10:22:32	2017-06-27 05:39:33
1591	26	1.0	5.0	2.6346153846153846	1999-11-19 00:37:57	2018-08-01 15:54:59

only showing top 10 rows

# PySpark SLQ tutorial 3

```
tags.groupBy("movieId").agg(  
    f.collect_set("tag"),  
    f.count("tag"),  
    f.collect_set("userId"),  
    f.count("userId"),  
    f.min("timestamp"),  
    f.max("timestamp"),  
  
) .show(10)  
  
+-----+-----+-----+-----+-----+-----+  
|movieId| collect_set(tag)|count(tag)|collect_set(userId)|count(userId)| min(timestamp)| max(timestamp)|  
+-----+-----+-----+-----+-----+-----+  
| 471| [hula hoop]| 1| [474]| 1| 2006-01-16 08:39:07| 2006-01-16 08:39:07|  
| 1088| [music, dance]| 2| [474]| 2| 2006-01-27 03:20:56| 2006-01-27 03:20:56|  
| 1580| [aliens]| 1| [474]| 1| 2006-01-14 09:25:19| 2006-01-14 09:25:19|  
| 1645| [lawyers]| 1| [474]| 1| 2006-01-16 08:14:55| 2006-01-16 08:14:55|  
| 1959| [adultery, Africa]| 2| [474]| 2| 2006-01-23 22:58:43| 2006-01-23 22:58:43|  
| 2122| [Stephen King]| 1| [474]| 1| 2006-01-16 08:08:16| 2006-01-16 08:08:16|  
| 3175| [spoof]| 1| [474]| 1| 2006-01-23 23:05:15| 2006-01-23 23:05:15|  
| 6466| [In Netflix queue]| 1| [474]| 1| 2006-01-14 08:27:07| 2006-01-14 08:27:07|  
| 6620| [cancer]| 1| [474]| 1| 2006-01-14 09:28:35| 2006-01-14 09:28:35|  
| 7833| [Nick and Nora Ch...]| 1| [474]| 1| 2006-01-14 08:21:01| 2006-01-14 08:21:01|  
+-----+-----+-----+-----+-----+  
only showing top 10 rows
```

# PySpark Dataframe Exercise

Load data from **movies\_small.csv** to a dataframe named **movies** and perform the following tasks:

1. Show ID, title, and genres of all movies with ‘Action’ included in their genres
2. Show ID, title, and the number of genres of each movie, sorted by movieId
3. Show the number of movies of each genre, sorted by the count number
4. Show the list of all movies associated with each genres.
5. Show the years of the first\_appearance of ‘Animation’ and ‘Sci-Fi’ movies

# Run PySpark on Kaggle Notebook

- Click Code -> click New Notebook



# Run PySpark on Kaggle Notebook

- Make sure Internet is turned On on Kaggle Notebook
- Run !pip install pyspark

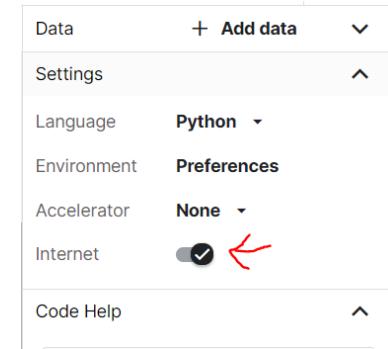
```
▶ !pip install pyspark
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("sparkOnKaggle").getOrCreate()

Collecting pyspark
  Downloading pyspark-3.1.2.tar.gz (212.4 MB)
    |██████████| 212.4 MB 57 kB/s eta 0:00:01   ||██████████| 6.2 MB 10.2 MB/s eta 0:00:21
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
    |██████████| 198 kB 39.6 MB/s eta 0:00:01
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl size=212880768 sha256=0b1eec842dd65803d8c4a025067a20de79057b3a1fc32a1ab74a785eb56b25
  Stored in directory: /root/.cache/pip/wheels/a5/0a/c1/9561f6fecb759579a7d863dc846daaa95f598744e71b02c77
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.2

[8]: + Code + Markdown

[8]: spark.version
```





# Q & A



## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.



# Big Data

## Spark ML

Instructor: Trong-Hop Do

August 15<sup>th</sup> 2021

**S<sup>3</sup>Lab**  
*Smart Software System Laboratory*



“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”

– Chris Lynch, Vertica Systems

# Spark Machine Learning



# Spark Mllib Utilities

## Linear Algebra

- API Guide

<http://spark.apache.org/docs/3.0.1/api/python/pyspark.ml.html#module-pyspark.ml.linalg>

- MLlib RDD-Based Guide

<http://spark.apache.org/docs/latest/mllib-data-types.html>

# Spark Mllib Utilities

## Linear Algebra

⚠ Not secure | [spark.apache.org/docs/3.0.1/api/python/pyspark.ml.html#module-pyspark.ml.linalg](http://spark.apache.org/docs/3.0.1/api/python/pyspark.ml.html#module-pyspark.ml.linalg)

### pyspark.ml.linalg module

MLlib utilities for linear algebra. For dense vectors, MLlib uses the NumPy `array` type, so you can simply pass NumPy arrays around. For sparse vectors, users can construct a `SparseVector` object from MLlib or pass SciPy `scipy.sparse` column vectors if SciPy is available in their environment.

`class pyspark.ml.linalg.Vector`

[\[source\]](#)

`toArray()`

[\[source\]](#)

Convert the vector into an numpy.ndarray

**Returns:** numpy.ndarray

`class pyspark.ml.linalg.DenseVector(ar)`

[\[source\]](#)

A dense vector represented by a value array. We use numpy array for storage and arithmetics will be delegated to the underlying numpy array.

```
>>> v = Vectors.dense([1.0, 2.0])
>>> u = Vectors.dense([3.0, 4.0])
>>> v + u
DenseVector([4.0, 6.0])
>>> 2 * v
DenseVector([1.0, 0.0])
>>> v / 2
DenseVector([0.5, 1.0])
>>> v * u
DenseVector([3.0, 8.0])
>>> u / v
DenseVector([3.0, 2.0])
>>> u % 2
```

# Spark Mllib Utilities

## Linear Algebra

The screenshot shows the Apache Spark 3.1.1 documentation page for "Data Types - RDD-based API". The page has a header with navigation links for Overview, Programming Guides, API Docs, Deploying, More, and a search bar. A red arrow points from the "Data types" link in the sidebar to the "Local vector" item in the list of data types.

Not secure | spark.apache.org/docs/latest/mllib-data-types.html

Apache Spark 3.1.1

Overview Programming Guides API Docs Deploying More Search the docs

**Guide**

- Basic statistics
- Data sources
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

**Mlib: RDD-based API Guide**

- Data types**
- Basic statistics
- Classification and regression
- Collaborative filtering
- Clustering
- Dimensionality reduction

**Data Types - RDD-based API**

- Local vector
- Labeled point
- Local matrix
- Distributed matrix
  - RowMatrix
  - IndexedRowMatrix
  - CoordinateMatrix
  - BlockMatrix

MLlib supports local vectors and matrices stored on a single machine, as well as distributed matrices backed by one or more RDDs. Local vectors and local matrices are simple data models that serve as public interfaces. The underlying linear algebra operations are provided by [Breeze](#). A training example used in supervised learning is called a "labeled point" in MLlib.

### Local vector

A local vector has integer-typed and 0-based indices and double-typed values, stored on a single machine. MLlib supports two types of local vectors: dense and sparse. A dense vector is backed by a double array representing its entry values, while a sparse vector is backed by two parallel arrays: indices and values. For example, a vector (1.0, 0.0, 3.0) can be represented in dense format as [1.0, 0.0, 3.0] or in sparse format as (3, [0, 2], [1.0, 3.0]), where 3 is the size of the vector.

# Spark Mllib Utilities

## Linear Algebra

### Vectors

- Vectors (main class to use)
- DenseVector: MLlib uses the NumPy array type
- SparseVector : You can pass SciPy `scipy.sparse` column vectors
- VectorUDT
- Vector

### Matrices

- Matrices (\_main class to use\_)
- DenseMatrix: Column-major dense matrix
- SparseMatrix : Sparse matrix stored in CSC format
- MatrixUDT
- Matrix

Vectors and matrices are important data types and provide integration with NumPy and SciPy

# Spark Mllib Utilities

## Linear Algebra

```
[ ]: import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

# Spark Mllib Utilities

## Linear Algebra

### Sparse Vector Example

```
[ ]: import numpy as np
import scipy.sparse as sps
from pyspark.ml.linalg import Vectors

# Use a NumPy array as a dense vector.
dv1 = np.array([1.0, 0.0, 3.0])

# Create a SparseVector.
sv1 = Vectors.sparse(3, [0, 2], [1.0, 3.0])

print(dv1)
print(sv1)
```

### Dense Vector Example

```
[ ]: # Use a Python List as a dense vector.
dv2 = [1.0, 0.0, 3.0]

# Use a single-column SciPy csc_matrix as a sparse vector.
sv2 = sps.csc_matrix(
    (np.array([1.0, 3.0]), np.array([0, 2]), np.array([0, 2])), shape=(3, 1)
)

print(dv2)
print(sv2)
```

# Spark Mllib Utilities

## Linear Algebra

### Dense Matrix

```
[ ]: from pyspark.ml.linalg import Matrix, Matrices  
  
# Create a dense matrix ((1.0, 2.0), (3.0, 4.0), (5.0, 6.0))  
dm2 = Matrices.dense(3, 2, [1, 3, 5, 2, 4, 6])  
print(dm2)
```

### Sparse Matrix

```
[ ]: # Create a sparse matrix ((9.0, 0.0), (0.0, 8.0), (0.0, 6.0))  
sm = Matrices.sparse(3, 2, [0, 1, 3], [0, 2, 1], [9, 6, 8])  
print(sm)
```

# Spark Mllib Utilities

## Data Sources

<http://spark.apache.org/docs/3.0.1/api/python/pyspark.ml.html#module-pyspark.ml.image>

<https://spark.apache.org/docs/latest/ml-datasource#image-data-source>

# Spark Mllib Utilities

## Data Sources

→ C Not secure | spark.apache.org/docs/3.0.1/api/python/pyspark.ml.html#module-pyspark.ml.image

### pyspark.ml.image module

#### pyspark.ml.image.Imageschema

An attribute of this module that contains the instance of `_ImageSchema`.

#### class pyspark.ml.image.\_ImageSchema

[source]

Internal class for `pyspark.ml.image.ImageSchema` attribute. Meant to be private and not to be instantized. Use `pyspark.ml.image.ImageSchema` attribute to access the APIs of this class.

#### property columnSchema

Returns the schema for the image column.

**Returns:** a `structType` for image column, `struct<origin:string, height:int, width:int, nChannels:int, mode:int, data:binary>`.

*New in version 2.4.0.*

#### property imageFields

Returns field names of image columns.

**Returns:** a list of field names.

*New in version 2.3.0.*

#### property imageSchema

Returns the image schema.

**Returns:** a `structType` with a single column of images named "image" (nullable) and having the same type returned by `columnSchema()`.

# Spark Mllib Utilities

## Data Sources

The screenshot shows a web browser displaying the Apache Spark 3.1.1 documentation. The URL in the address bar is [spark.apache.org/docs/latest/ml-datasource#image-data-source](https://spark.apache.org/docs/latest/ml-datasource#image-data-source). The page title is "Data sources". On the left, there's a sidebar with two main sections: "MLlib: Main Guide" and "MLlib: RDD-based API Guide". The "MLlib: Main Guide" section has a list of topics including "Basic statistics", "Data sources" (which is highlighted in blue), "Pipelines", etc. The "MLlib: RDD-based API Guide" section also lists various topics. At the bottom of the sidebar, there are tabs for "Scala", "Java", "Python" (which is highlighted in blue), and "R". The main content area starts with a heading "Data sources" and a brief introduction about using data sources in ML to load data. It then lists specific data sources: "Image data source" and "LIBSVM data source". Below this, there's a section titled "Image data source" with a detailed description of its functionality and schema.

← → C spark.apache.org/docs/latest/ml-datasource#image-data-source

APACHE Spark 3.1.1 Overview Programming Guides API Docs Deploying More Search the docs

**MLlib: Main Guide**

- Basic statistics
- **Data sources**
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

**MLlib: RDD-based API Guide**

- Data types
- Basic statistics
- Classification and regression
- Collaborative filtering
- Clustering
- Dimensionality reduction

## Data sources

In this section, we introduce how to use data source in ML to load data. Besides some general data sources such as Parquet, CSV, JSON and JDBC, we also provide some specific data sources for ML.

### Table of Contents

- Image data source
- LIBSVM data source

### Image data source

This image data source is used to load image files from a directory, it can load compressed image (jpeg, png, etc.) into raw image representation via `ImageIO` in Java library. The loaded DataFrame has one `StructType` column: "image", containing image data stored as image schema. The schema of the `image` column is:

- `origin: StringType` (represents the file path of the image)
- `height: IntegerType` (height of the image)
- `width: IntegerType` (width of the image)
- `nChannels: IntegerType` (number of image channels)
- `mode: IntegerType` (OpenCV-compatible type)
- `data: BinaryType` (Image bytes in OpenCV-compatible order: row-wise BGR in most cases)

Scala Java Python R

# Spark Mllib Utilities

## Data Sources

```
[3]: import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

Example of loading image data using the `kittens` dataset that ships with Spark

Refer to ML Main Guide for more info: <https://spark.apache.org/docs/latest/ml-datasource#image-data-source>

The schema of the image column is:

- origin: StringType (represents the file path of the image)
- height: IntegerType (height of the image)
- width: IntegerType (width of the image)
- nChannels: IntegerType (number of image channels)
- mode: IntegerType (OpenCV-compatible type)
- data: BinaryType (Image bytes in OpenCV-compatible order: row-wise BGR in most cases)

# Spark Mllib Utilities

## Data Sources

```
[22]: PATH = "D:/Spark/spark-3.0.1-bin-hadoop2.7/data/mllib/images/origin/kittens"
df = (
    spark.read.format("image")
    .option("dropInvalid", True)
    .load(PATH)
    .select("image.origin", "image.height", "image.width", "image.nChannels", "image.mode", "image.data")
)
df.toPandas()
```

```
[22]:
```

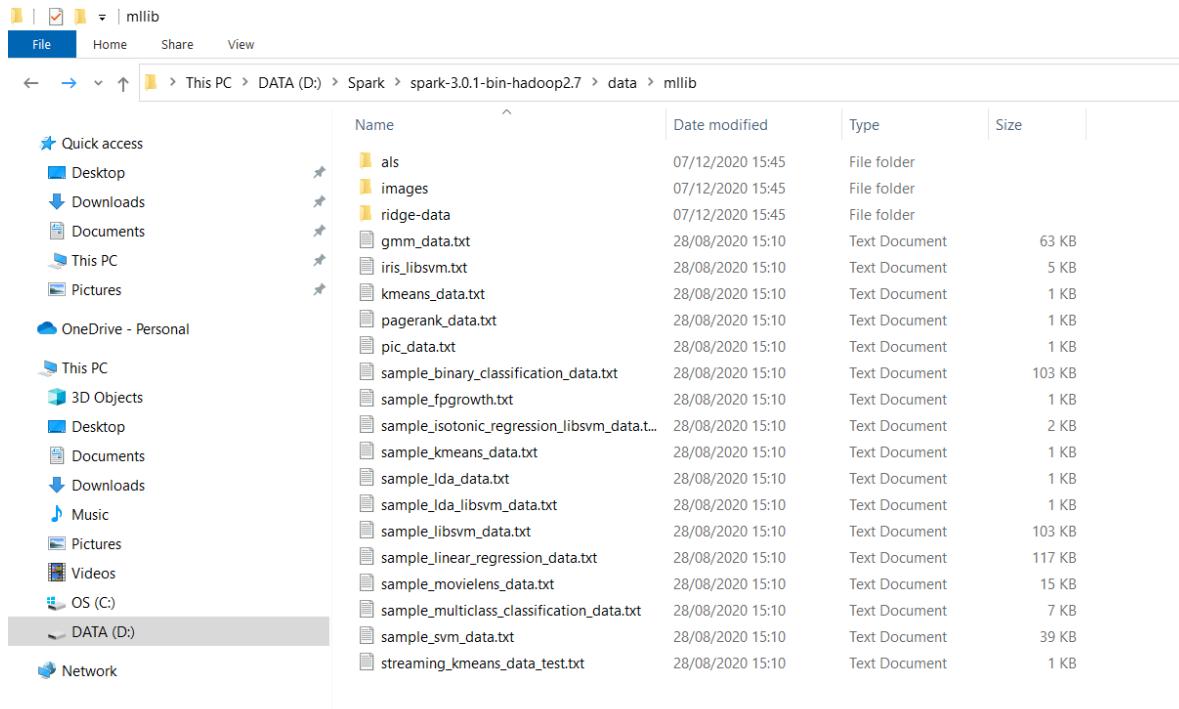
	origin	height	width	nChannels	mode	data
0	file:///usr/local/spark-2.4.3-bin-hadoop2.7/da...	311	300	3	16	[193, 193, 193, 194, 194, 194, 194, 194, 194, ...]
1	file:///usr/local/spark-2.4.3-bin-hadoop2.7/da...	313	199	3	16	[208, 229, 237, 202, 223, 231, 210, 231, 239, ...]
2	file:///usr/local/spark-2.4.3-bin-hadoop2.7/da...	200	300	3	16	[88, 93, 96, 88, 93, 96, 88, 93, 96, 89, 94, 9...
3	file:///usr/local/spark-2.4.3-bin-hadoop2.7/da...	296	300	3	16	[203, 230, 244, 202, 229, 243, 201, 228, 242, ...]

```
[13]: df.printSchema()
```

```
root
 |-- image: struct (nullable = true)
 |   |-- origin: string (nullable = true)
 |   |-- height: integer (nullable = true)
 |   |-- width: integer (nullable = true)
 |   |-- nChannels: integer (nullable = true)
 |   |-- mode: integer (nullable = true)
 |   |-- data: binary (nullable = true)
```

# Spark Mllib Utilities

## Sample Data

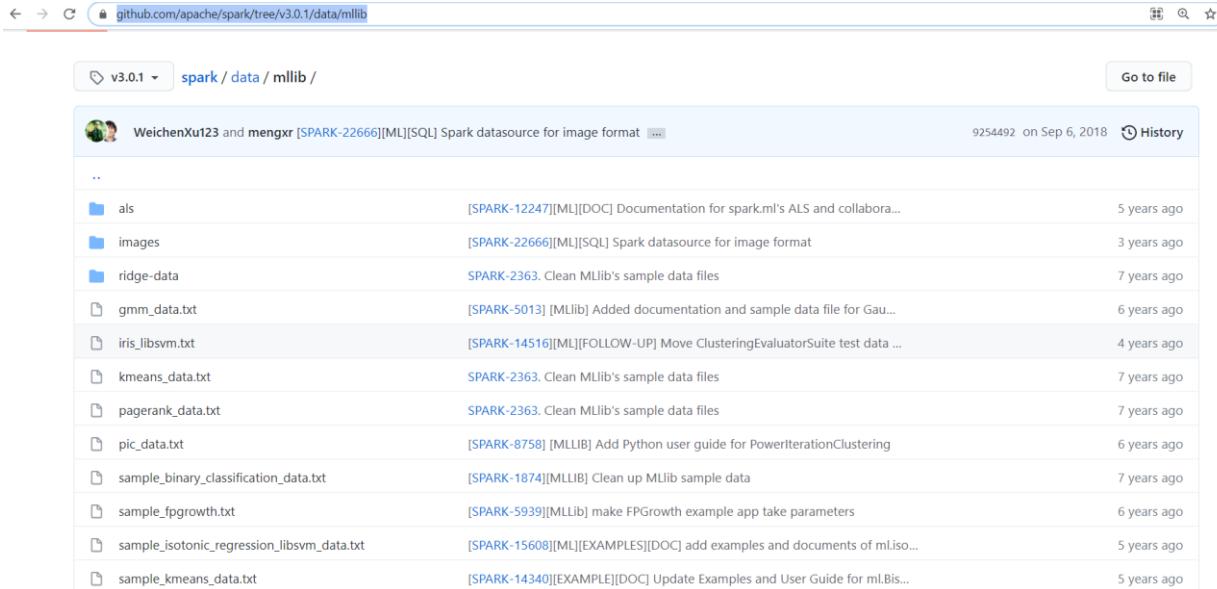


Name	Date modified	Type	Size
als	07/12/2020 15:45	File folder	
images	07/12/2020 15:45	File folder	
ridge-data	07/12/2020 15:45	File folder	
gmm_data.txt	28/08/2020 15:10	Text Document	63 KB
iris_libsvm.txt	28/08/2020 15:10	Text Document	5 KB
kmeans_data.txt	28/08/2020 15:10	Text Document	1 KB
pagerank_data.txt	28/08/2020 15:10	Text Document	1 KB
pic_data.txt	28/08/2020 15:10	Text Document	1 KB
sample_binary_classification_data.txt	28/08/2020 15:10	Text Document	103 KB
sample_fpgrowth.txt	28/08/2020 15:10	Text Document	1 KB
sample_isotonic_regression_libsvm_data.t...	28/08/2020 15:10	Text Document	2 KB
sample_kmeans_data.txt	28/08/2020 15:10	Text Document	1 KB
sample_lda_data.txt	28/08/2020 15:10	Text Document	1 KB
sample_lda_libsvm_data.txt	28/08/2020 15:10	Text Document	1 KB
sample_libsvm_data.txt	28/08/2020 15:10	Text Document	103 KB
sample_linear_regression_data.txt	28/08/2020 15:10	Text Document	117 KB
sample_movielens_data.txt	28/08/2020 15:10	Text Document	15 KB
sample_multiclass_classification_data.txt	28/08/2020 15:10	Text Document	7 KB
sample_svm_data.txt	28/08/2020 15:10	Text Document	39 KB
streaming_kmeans_data_test.txt	28/08/2020 15:10	Text Document	1 KB

# Spark Mllib Utilities

## Sample Data

Github: <https://github.com/apache/spark/tree/v3.0.1/data/mllib>



The screenshot shows a GitHub repository page for the 'spark / data / mllib' directory in the 'v3.0.1' branch. The page lists several sample data files with their commit history:

File	Commit Message	Author	Date
als	[SPARK-12247][ML][DOC] Documentation for spark.ml's ALS and collabor...	WeichenXu123 and mengxr [SPARK-22666][ML][SQL]	on Sep 6, 2018
images	[SPARK-22666][ML][SQL] Spark datasource for image format		5 years ago
ridge-data	SPARK-2363. Clean MLlib's sample data files		3 years ago
gmm_data.txt	[SPARK-5013] [MLlib] Added documentation and sample data file for Gau...		7 years ago
iris_libsvm.txt	[SPARK-14516][ML][FOLLOW-UP] Move ClusteringEvaluatorSuite test data ...		6 years ago
kmeans_data.txt	SPARK-2363. Clean MLlib's sample data files		4 years ago
pagerank_data.txt	SPARK-2363. Clean MLlib's sample data files		7 years ago
pic_data.txt	[SPARK-8758] [MLLIB] Add Python user guide for PowerIterationClustering		7 years ago
sample_binary_classification_data.txt	[SPARK-1874][MLLIB] Clean up MLlib sample data		6 years ago
sample_fprowth.txt	[SPARK-5939][MLlib] make FPGrowth example app take parameters		5 years ago
sample_isotonic_regression_libsvm_data.txt	[SPARK-15608][ML][EXAMPLES][DOC] add examples and documents of ml.iso...		7 years ago
sample_kmeans_data.txt	[SPARK-14340][EXAMPLE][DOC] Update Examples and User Guide for ml.Bis...		5 years ago

# Spark Mllib Utilities

## Heppers

- <http://spark.apache.org/docs/3.0.1/api/python/pyspark.ml.html#module-pyspark.ml.util>

- |                         |                       |
|-------------------------|-----------------------|
| ● BaseReadWrite         | ● JavaMLReadable      |
| ● DefaultParamsReadable | ● JavaMLReader        |
| ● DefaultParamsReader   | ● JavaMLWritable      |
| ● DefaultParamsWritable | ● JavaMLWriter        |
| ● DefaultParamsWriter   | ● JavaPredictionModel |
| ● GeneralJavaMLWritable | ● MLReadable          |
| ● GeneralJavaMLWriter   | ● MLReader            |
| ● GeneralMLWriter       | ● MLWritable          |
| ● Identifiable          | ● MLWriter            |

# Spark Mllib

## Learning Resources

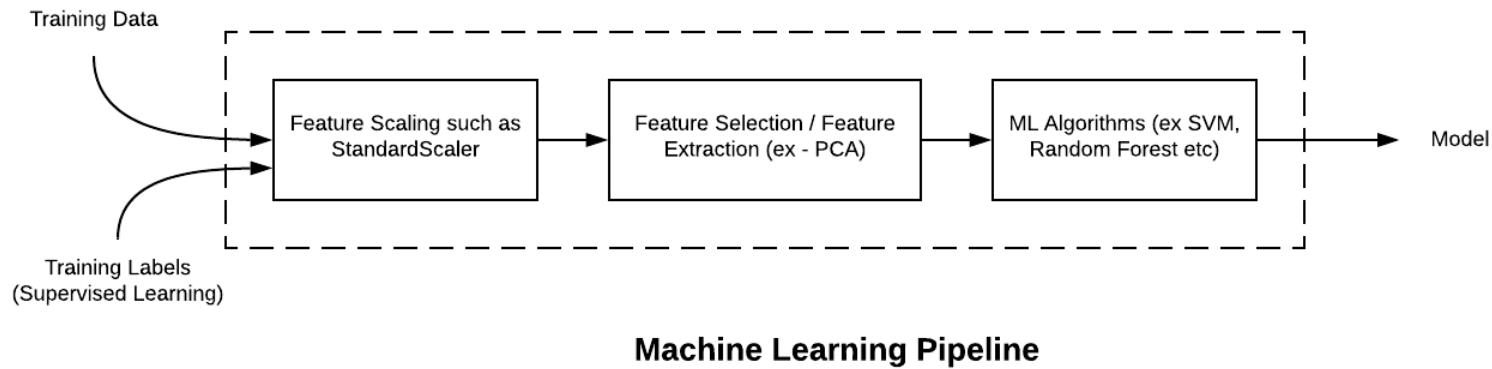
- Main guide
  - <http://spark.apache.org/docs/3.0.1/ml-guide.html>
- API guide
  - <http://spark.apache.org/docs/3.0.1/api/python/pyspark.ml.html>
- Github
  - <https://github.com/apache/spark/tree/v3.0.1/python/pyspark/ml>

# Spark MLlib Pipelines

- Pipelines API
  - Provide a Uniform Set of High-level APIs Built on Top of DataFrames that Help Users Create and Tune Practical Machine Learning Pipelines
  - <https://spark.apache.org/docs/latest/ml-pipeline.html>

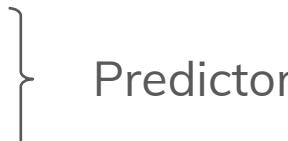
# Spark MLlib Pipelines

- Algorithms Working Together: Assembled in a Pipeline



# Spark MLlib Pipelines

## 5 Key Concepts

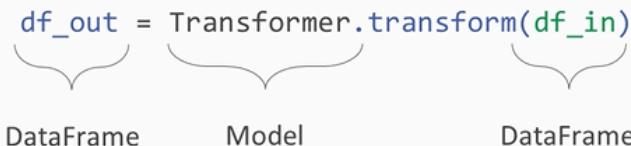
- DataFrame
  - Estimator
  - Transformer
  - Parameters
  - Pipeline
- 
- The diagram shows a brace on the right side of the list, grouping 'Estimator' and 'Transformer' together under the label 'Predictor'. This indicates that these two components are part of the Predictor stage in a machine learning pipeline.

# Spark MLlib Pipelines

## Transformer

- A Transformer is an abstraction that includes feature transformers and learned models
- Technically, a transformer implements a method `.transform()`, which converts one DataFrame into another, generally by appending one or more columns
- In code:

```
df_out = Transformer.transform(df_in)
```



The diagram illustrates the transformation process. It shows the code `df_out = Transformer.transform(df_in)`. Brackets under `df_out` and `df_in` are labeled "DataFrame". A bracket under `Transformer` is labeled "Model".

# Spark MLlib Pipelines

## Estimator

- An estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data
- Technically, an Estimator implements a method `.fit()`, which accepts a DataFrame and produces a model which is a transformer
- In code:

```
df_out = Estimator.fit(df_in)
```

Transformer

DataFrame

# Spark MLlib Pipelines

## Estimator vs Transformer

```
model_out = Estimator.fit(df_in)
```

Transformer



DataFrame

```
df_out = Transformer.transform(df_in)
```

  
DataFrame  
Model  
DataFrame

# Spark MLlib Pipelines

## Pipeline

- Stages:
  - Pipelines consist of a sequence of stages that run in order
  - Each stage is either a transformer or an estimator
    - .transform()
    - .fit()

```
class Pipeline  
    .fit(DataFrame)
```

Acts as an Estimator, consists of stages

Stages execute in order when `.fit()` is called

```
class PipelineModel  
    .transform(DataFrame)
```

Represents a compiled pipeline with transformers and fitted models

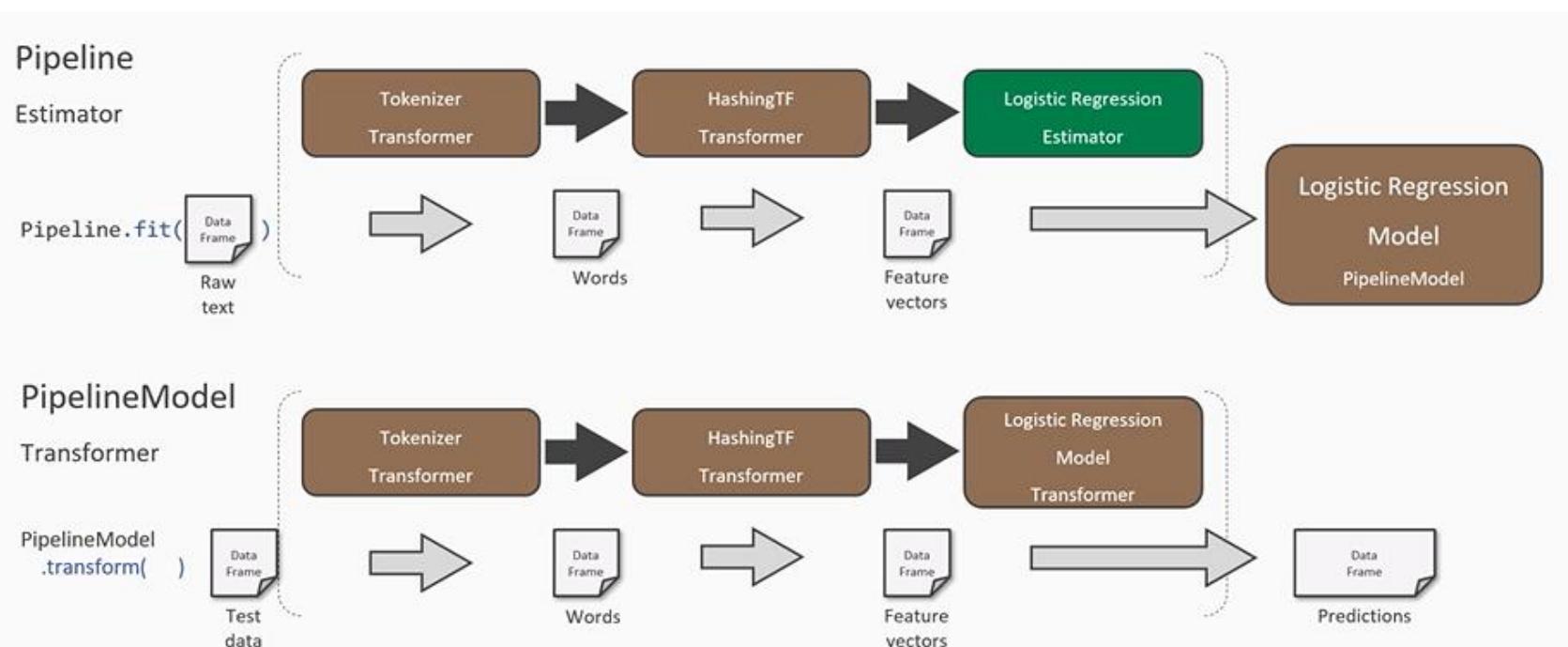
A pipeline's `fit()` method, produces a `PipelineModel`

Acts as a transformer, consists of stages

Pc

# Spark MLlib Pipelines

## Example



# Spark MLlib Pipelines

- Spark MLlib pipeline API:
  - MLlib standardizes APIs for its many machine learning algorithms to make it easier to combine multiple algorithms
  - The key concepts of pipelines are: Estimator, transformer, and parameter
  - Pipelines are a sequence of stages
  - Pipelines produce PipelineModels

# Spark MLlib Pipelines

- Spark MLlib pipeline API:
  - A PipelineModel represents a compiled pipeline with transformers and fitted models
  - PipelineModels perform the same transformations before running data through the compiled model

# Spark MLlib Pipelines

## Classification and regression

- <https://spark.apache.org/docs/latest/ml-classification-regression.html>

This page covers algorithms for Classification and Regression. It also includes sections discussing specific classes of algorithms, such as linear methods, trees, and ensembles.

### Table of Contents

- Classification
  - Logistic regression
    - Binomial logistic regression
    - Multinomial logistic regression
  - Decision tree classifier
  - Random forest classifier
  - Gradient-boosted tree classifier
  - Multilayer perceptron classifier
  - Linear Support Vector Machine
  - One-vs-Rest classifier (a.k.a. One-vs-All)
  - Naive Bayes
  - Factorization machines classifier
- Regression
  - Linear regression
  - Generalized linear regression
    - Available families
  - Decision tree regression
  - Random forest regression
  - Gradient-boosted tree regression
  - Survival regression
  - Isotonic regression

# Spark MLlib Pipelines

## Classification and regression

```
[2]: import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

*# Load and parse the data file, converting it to a DataFrame.*

```
sample_libsvm_data = spark.read.format("libsvm").load("D:/Spark/spark-3.0.1-bin-hadoop2.7/data/mllib/sample_libsvm_data.txt")
```

# Spark MLlib Pipelines

## DecisionTreeClassifier

```
: from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import StringIndexer, VectorIndexer
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

sample_libsvm_data.show(5)
```

```
+----+-----+
|label|    features|
+----+-----+
| 0.0|(692,[127,128,129...|
| 1.0|(692,[158,159,160...|
| 1.0|(692,[124,125,126...|
| 1.0|(692,[152,153,154...|
| 1.0|(692,[151,152,153...|
+----+-----+
only showing top 5 rows
```

# Spark MLlib Pipelines

## DecisionTreeClassifier – Feature Transformers

StringIndexer encodes a string column of labels to a column of label indices

```
label_indexer = StringIndexer(inputCol="label", outputCol="indexedLabel").fit(sample_libsvm_data)

print(type(label_indexer))
label_indexer.transform(sample_libsvm_data).show()
```

id	category
0	a
1	b
2	c
3	a
4	a
5	c



id	category	categoryIndex
0	a	0.0
1	b	2.0
2	c	1.0
3	a	0.0
4	a	0.0
5	c	1.0

```
<class 'pyspark.ml.feature.StringIndexerModel'>
+-----+-----+
|label|          features|indexedLabel|
+-----+-----+
| 0.0|(692,[127,128,129,130,131,154,155,156,157,158,1...| 1.0|
| 1.0|(692,[158,159,160,161,185,186,187,188,189,213,2...| 0.0|
| 1.0|(692,[124,125,126,127,151,152,153,154,155,179,1...| 0.0|
| 1.0|(692,[152,153,154,180,181,182,183,208,209,210,2...| 0.0|
| 1.0|(692,[151,152,153,154,179,180,181,182,208,209,2...| 0.0|
+-----+-----+
```

# Spark MLlib Pipelines

## DecisionTreeClassifier – Feature Transformers

[Vectorizer](#) helps index categorical features in datasets of Vectors. It can both automatically decide which features are categorical and convert original values to category indices.

```
# Automatically identify categorical features, and index them.  
# We specify maxCategories so features with > 4 distinct values are treated as continuous.  
feature_indexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4).fit(sample_libsvm_data)  
  
feature_indexer.transform(sample_libsvm_data).show(5,truncate=50)
```

label	features	indexedFeatures
0.0	(692,[127,128,129,130,131,154,155,156,157,158,1...]	(692,[127,128,129,130,131,154,155,156,157,158,1...]
1.0	(692,[158,159,160,161,185,186,187,188,189,213,2...]	(692,[158,159,160,161,185,186,187,188,189,213,2...]
1.0	(692,[124,125,126,127,151,152,153,154,155,179,1...]	(692,[124,125,126,127,151,152,153,154,155,179,1...]
1.0	(692,[152,153,154,180,181,182,183,208,209,210,2...]	(692,[152,153,154,180,181,182,183,208,209,210,2...]
1.0	(692,[151,152,153,154,179,180,181,182,208,209,2...]	(692,[151,152,153,154,179,180,181,182,208,209,2...]

# Spark MLlib Pipelines

## DecisionTreeClassifier

```
: # Creating our stages:

# STAGE 1:
# Index labels, adding metadata to the label column.
# Fit on whole dataset to include all labels in index.
label_indexer = StringIndexer(inputCol="label", outputCol="indexedLabel").fit(
    sample_libsvm_data
)

# STAGE 2:
# Automatically identify categorical features, and index them.
# We specify maxCategories so features with > 4 distinct values are treated as continuous.
feature_indexer = VectorIndexer(
    inputCol="features", outputCol="indexedFeatures", maxCategories=4
).fit(sample_libsvm_data)

# STAGE 3:
# Train a DecisionTree model.
decission_tree_classifier_model = DecisionTreeClassifier(
    labelCol="indexedLabel", featuresCol="indexedFeatures"
)

print(type(decission_tree_classifier_model))
<class 'pyspark.ml.classification.DecisionTreeClassifier'>
```

# Spark MLlib Pipelines

## DecisionTreeClassifier

```
# Creating our Pipeline:  
  
# Chain indexers and tree in a Pipeline  
pipeline = Pipeline(  
    stages=[  
        label_indexer, # STAGE 1  
        feature_indexer, # STAGE 2  
        decision_tree_classifier_model, # STAGE 3  
    ]  
)
```

# Spark MLlib Pipelines

## DecisionTreeClassifier

```
# Split the data into training and test sets (30% held out for testing)
(training_data, test_data) = sample_libsvm_data.randomSplit([0.7, 0.3])

# Train model. This also runs the indexers.
model = pipeline.fit(training_data)

# Make predictions.
predictions = model.transform(test_data)

# Select example rows to display.
predictions.select("prediction", "indexedLabel", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(
    labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy"
)
accuracy = evaluator.evaluate(predictions)
print(f"Test Error = {1.0 - accuracy:.5f} ")
```

```
+-----+-----+-----+
|prediction|indexedLabel|      features|
+-----+-----+-----+
|     1.0|        1.0|(692,[95,96,97,12...
|     1.0|        1.0|(692,[123,124,125...
|     1.0|        1.0|(692,[124,125,126...
|     0.0|        1.0|(692,[124,125,126...
|     1.0|        1.0|(692,[126,127,128...
+-----+-----+-----+
only showing top 5 rows
```

Test Error = 0.19355

# Spark MLlib Pipelines

## DecisionTreeClassifier

```
# You can see that the Pipeline and the PipelineModel have the same stages
print(pipeline.getStages())
print(model.stages)
```

```
[StringIndexerModel: uid=StringIndexer_bde466415729, handleInvalid=error, VectorIndexerModel: uid=VectorIndexer_5be5f8abe571, numFeatures=692,
handleInvalid=error, DecisionTreeClassifier_2381fa7fa6a8]
[StringIndexerModel: uid=StringIndexer_bde466415729, handleInvalid=error, VectorIndexerModel: uid=VectorIndexer_5be5f8abe571, numFeatures=692,
handleInvalid=error, DecisionTreeClassificationModel: uid=DecisionTreeClassifier_2381fa7fa6a8, depth=2, numNodes=5, numClasses=2, numFeatures=6
92]
```

# Spark MLlib Pipelines

## DecisionTreeClassifier

```
decission_tree_classifier_model = DecisionTreeClassifier(labelCol="label", featuresCol="features")

pipeline = Pipeline( stages=[decission_tree_classifier_model, ])

# Train model. This also runs the indexers.
model = pipeline.fit(training_data)

# Make predictions.
predictions = model.transform(test_data)

predictions.show()

# You can see that the Pipeline and the PipelineModel have the same stages
print(pipeline.getStages())
print(model.stages)

[DecisionTreeClassifier_7387b0a8a12c]
[DecisionTreeClassificationModel: uid=DecisionTreeClassifier_7387b0a8a12c, depth=2, numNodes=5, numClasses=2, numFeatures=692]
```

prediction label	features
1.0  0.0  (692,[98,99,100,1...	
0.0  0.0  (692,[100,101,102...	
0.0  0.0  (692,[122,123,124...	
0.0  0.0  (692,[123,124,125...	
0.0  0.0  (692,[124,125,126...	

# Spark MLlib Pipelines

## Random Forest Regression

```
from pyspark.ml import Pipeline
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator
```

```
sample_libsvm_data.show(5)
```

```
+-----+
|label|      features|
+-----+
| 0.0|(692,[127,128,129...|
| 1.0|(692,[158,159,160...|
| 1.0|(692,[124,125,126...|
| 1.0|(692,[152,153,154...|
| 1.0|(692,[151,152,153...|
+-----+
only showing top 5 rows
```

# Spark MLlib Pipelines

## Random Forest Regression

```
# Creating our stages:  
  
# STAGE 1:  
# Automatically identify categorical features, and index them.  
feature_indexer = VectorIndexer(  
    inputCol="features",  
    outputCol="indexedFeatures",  
    # Set maxCategories so features with > 4 distinct values are treated as continuous.  
    maxCategories=4,  
).fit(sample_libsvm_data)  
  
# STAGE 2:  
# Train a RandomForest model.  
random_forest_model = RandomForestRegressor(featuresCol="indexedFeatures")
```

# Spark MLlib Pipelines

## Random Forest Regression

```
# Creating our Pipeline:  
# Chain indexer and forest in a Pipeline  
pipeline = Pipeline(stages=[feature_indexer, random_forest_model])
```

# Spark MLlib Pipelines

## Random Forest Regression

```
# Split the data into training and test sets (30% held out for testing)
(training_data, test_data) = sample_libsvm_data.randomSplit([0.7, 0.3])

# Train model. This also runs the indexer.
model = pipeline.fit(training_data)

# Make predictions.
predictions = model.transform(test_data)

# Select example rows to display.
predictions.select("prediction", "label", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="rmse"
)
rmse = evaluator.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
+-----+-----+
|prediction|label|      features|
+-----+-----+
|     0.05|  0.0|(692,[122,123,148...|
|      0.0|  0.0|(692,[123,124,125...|
|     0.05|  0.0|(692,[124,125,126...|
|      0.0|  0.0|(692,[125,126,127...|
|      0.0|  0.0|(692,[126,127,128...|
+-----+-----+
only showing top 5 rows
```

Root Mean Squared Error (RMSE) on test data = 0.0310087

# Spark MLlib Pipelines

## Random Forest Regression

```
# You can see that the Pipeline and the PipelineModel have the same stages
print(pipeline.getStages())
print(model.stages)
```

```
[VectorIndexerModel: uid=VectorIndexer_98f042bd5bbd, numFeatures=692, handleInvalid=error, RandomForestRegressor_77ce3e646d07]
[VectorIndexerModel: uid=VectorIndexer_98f042bd5bbd, numFeatures=692, handleInvalid=error, RandomForestRegressionModel: uid=RandomForestRegressor_77ce3e646d07, numTrees=20, numFeatures=692]
```

```
# The last stage in a PipelineModel is usually the most informative
print(model.stages[-1])
```

```
RandomForestRegressionModel: uid=RandomForestRegressor_77ce3e646d07, numTrees=20, numFeatures=692
```

```
# Here you can see that pipeline and model are Pipeline and PipelineModel classes
print("pipeline:", type(pipeline))
print("model:", type(model))
```

```
pipeline: <class 'pyspark.ml.pipeline.Pipeline'>
model: <class 'pyspark.ml.pipeline.PipelineModel'>
```

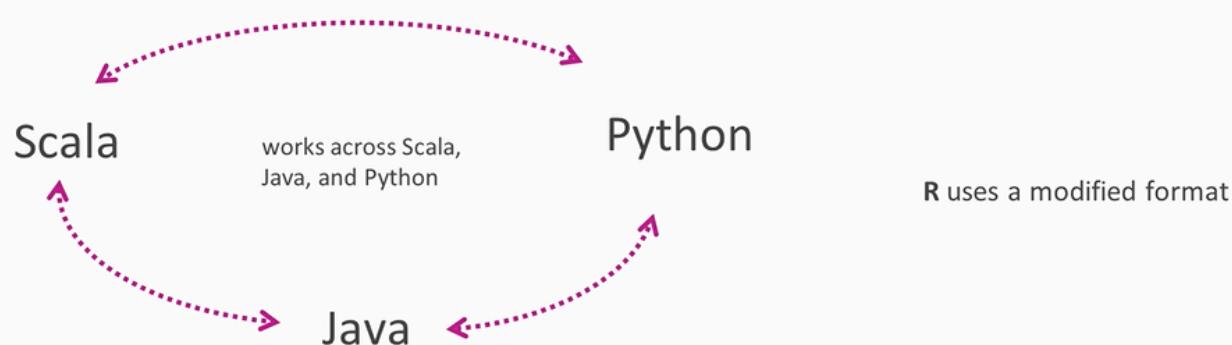
# Spark MLlib Pipelines

Classification and regression

- <https://spark.apache.org/docs/latest/ml-classification-regression.html>

# Spark MLlib Persistence

- Saving and loading of algorithms, models, and pipelines



# Spark MLlib Persistence

## Saving/Writing

```
Pipeline.write().save(path)  
Pipeline.save(path)
```



class MLWriter

## Loading/Reading

```
Pipeline.read().load(path)  
Pipeline.load(path)
```



class MLReader

# Spark MLlib Parameters

- Param: A named parameter with self-contained documentation

`.extractParams()`

- ParamMap: a set of (parameter, value) pairs

`.extractParamMap()`

# Spark MLlib Parameters

- MLlib Estimators and Transformers use a uniform API for specifying parameters.
- A Param is a named parameter with self-contained documentation. A ParamMap is a set of (parameter, value) pairs.
- There are two main ways to pass parameters to an algorithm:
  - Set parameters for an instance. E.g., if lr is an instance of LogisticRegression, one could call lr.setMaxIter(10) to make lr.fit() use at most 10 iterations. This API resembles the API used in spark.mllib package.
  - Pass a ParamMap to .fit() or .transform(). Any parameters in the ParamMap will override parameters previously specified via setter methods.
- Parameters belong to specific instances of Estimators and Transformers. E.g. if we have two instances lr1 and lr2, then we can build a ParamMap with both maxIter parameters: ParamMap({lr1.maxIter: 10, lr2.maxIter: 20}). This is useful if there are two algorithms with the maxIter parameter in a Pipeline.

# Spark MLlib Parameters

```
from IPython.core.display import display, HTML
from pyspark.sql import SparkSession
from pyspark.ml.linalg import Vectors
from pyspark.ml.classification import LogisticRegression

spark = SparkSession.builder.getOrCreate()

# Prepare training data from a list of (label, features) tuples.
training = spark.createDataFrame(
    [
        (1.0, Vectors.dense([0.0, 1.1, 0.1])),
        (0.0, Vectors.dense([2.0, 1.0, -1.0])),
        (0.0, Vectors.dense([2.0, 1.3, 1.0])),
        (1.0, Vectors.dense([0.0, 1.2, -0.5])),
        ...
    ],
    ["label", "features"],
)
# Prepare test data
test = spark.createDataFrame(
    [
        (1.0, Vectors.dense([-1.0, 1.5, 1.3])),
        (0.0, Vectors.dense([3.0, 2.0, -0.1])),
        (1.0, Vectors.dense([0.0, 2.2, -1.5])),
        ...
    ],
    ["label", "features"],
)
```

# Spark MLlib Parameters

```
# Create a LogisticRegression instance. This instance is an Estimator.  
# maxIter and regParam are parameters  
lr = LogisticRegression(maxIter=10, regParam=0.01)  
  
# Print out the parameters, documentation, and any default values.  
print(f"LogisticRegression parameters:\n{lr.explainParams()}"")
```

```
LogisticRegression parameters:  
aggregationDepth: suggested depth for treeAggregate (>= 2). (default: 2)  
elasticNetParam: the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty is an L2 penalty. For alpha = 1, it is an L1 penalty. (default: 0.0)  
family: The name of family which is a description of the label distribution to be used in the model. Supported options: auto, binomial, multinomial (default: auto)  
featuresCol: features column name. (default: features)  
fitIntercept: whether to fit an intercept term. (default: True)  
labelCol: label column name. (default: label)  
lowerBoundsOnCoefficients: The lower bounds on coefficients if fitting under bound constrained optimization. The bound matrix must be compatible with the shape (1, n  
umber of features) for binomial regression, or (number of classes, number of features) for multinomial regression. (undefined)  
lowerBoundsOnIntercepts: The lower bounds on intercepts if fitting under bound constrained optimization. The bounds vector size must be equal with 1 for binomial regr  
ession, or the number of classes for multinomial regression. (undefined)  
maxIter: max number of iterations (>= 0). (default: 100, current: 10)  
predictionCol: prediction column name. (default: prediction)  
probabilityCol: Column name for predicted class conditional probabilities. Note: Not all models output well-calibrated probability estimates! These probabilities sho  
uld be treated as confidences, not precise probabilities. (default: probability)  
rawPredictionCol: raw prediction (a.k.a. confidence) column name. (default: rawPrediction)  
regParam: regularization parameter (>= 0). (default: 0.0, current: 0.01)  
standardization: whether to standardize the training features before fitting the model. (default: True)  
threshold: Threshold in binary classification prediction, in range [0, 1]. If threshold and thresholds are both set, they must match.e.g. if threshold is p, then thr  
esholds must be equal to [1-p, p]. (default: 0.5)  
thresholds: Thresholds in multi-class classification to adjust the probability of predicting each class. Array must have length equal to the number of classes, with  
values > 0, excepting that at most one value may be 0. The class with largest value p/t is predicted, where p is the original probability of that class and t is the  
class's threshold. (undefined)
```

# Spark MLlib Parameters

```
print_explainParams(lr)
```

Parameters for: LogisticRegression\_49aa5a3b7809

**aggregationDepth**

suggested depth for treeAggregate (>= 2). (default: 2)

**elasticNetParam**

the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty is an L2 penalty. For alpha = 1, it is an L1 penalty. (default: 0.0)

**family**

The name of family which is a description of the label distribution to be used in the model. Supported options: auto, binomial, multinomial (default: auto)

**featuresCol**

features column name. (default: features)

**fitIntercept**

whether to fit an intercept term. (default: True)

**labelCol**

label column name. (default: label)

**lowerBoundsOnCoefficients**

The lower bounds on coefficients if fitting under bound constrained optimization. The bound matrix must be compatible with the shape (1, number of features) for binomial regression, or (number of classes, number of features) for multinomial regression. (undefined)

**lowerBoundsOnIntercepts**

The lower bounds on intercepts if fitting under bound constrained optimization. The bounds vector size must be equal with 1 for binomial regression, or the number of classes for multinomial regression. (undefined)

# Spark MLlib Parameters

```
# Learn a LogisticRegression model. This uses the parameters stored in lr.  
model1 = lr.fit(training)  
  
# Since model1 is a Model (i.e., a transformer produced by an Estimator),  
# we can view the parameters it used during fit().  
# This prints the parameter (name: value) pairs, where names are unique IDs for this  
# LogisticRegression instance.  
print("Model 1 was fit using parameters: ")  
print(model1.extractParamMap())
```

Model 1 was fit using parameters:

```
{Param(parent='LogisticRegression_49aa5a3b7809', name='aggregationDepth', doc='suggested depth for treeAggregate (>= 2)': 2, Param(parent='LogisticRegression_49aa5a3b7809', name='elasticNetParam', doc='the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty is an L2 penalty. For alpha = 1, it is an L1 penalty'): 0.0, Param(parent='LogisticRegression_49aa5a3b7809', name='family', doc='The name of family which is a description of the label distribution to be used in the model. Supported options: auto, binomial, multinomial.'): 'auto', Param(parent='LogisticRegression_49aa5a3b7809', name='featuresCol', doc='features column name'): 'features', Param(parent='LogisticRegression_49aa5a3b7809', name='fitIntercept', doc='whether to fit an intercept term'): True, Param(parent='LogisticRegression_49aa5a3b7809', name='labelCol', doc='label column name'): 'label', Param(parent='LogisticRegression_49aa5a3b7809', name='maxIter', doc='maximum number of iterations (>= 0)': 10, Param(parent='LogisticRegression_49aa5a3b7809', name='predictionCol', doc='prediction column name'): 'prediction', Param(parent='LogisticRegression_49aa5a3b7809', name='probabilityCol', doc='Column name for predicted class conditional probabilities. Note: Not all models output well-calibrated probability estimates! These probabilities should be treated as confidences, not precise probabilities'): 'probability', Param(parent='LogisticRegression_49aa5a3b7809', name='rawPredictionCol', doc='raw prediction (a.k.a. confidence) column name'): 'rawPrediction', Param(parent='LogisticRegression_49aa5a3b7809', name='regParam', doc='regularization parameter (>= 0)': 0.01, Param(parent='LogisticRegression_49aa5a3b7809', name='standardization', doc='whether to standardize the training features before fitting the model'): True, Param(parent='LogisticRegression_49aa5a3b7809', name='threshold', doc='threshold in binary classification prediction, in range [0, 1]'): 0.5, Param(parent='LogisticRegression_49aa5a3b7809', name='tol', doc='the convergence tolerance for iterative algorithms (>= 0)': 1e-06}
```

# Spark MLlib Parameters

```
print_explainParamMap(model1)
```

## Parameter Map

LogisticRegressionModel: uid = LogisticRegression\_49aa5a3b7809, numClasses = 2, numFeatures = 3

### aggregationDepth

doc: *suggested depth for treeAggregate (>= 2)*

value: 2

### elasticNetParam

doc: *the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty is an L2 penalty. For alpha = 1, it is an L1 penalty*

value: 0.0

### family

doc: *The name of family which is a description of the label distribution to be used in the model. Supported options: auto, binomial, multinomial.*

value: auto

### featuresCol

doc: *features column name*

value: features

### fitIntercept

doc: *whether to fit an intercept term*

value: True

# Spark MLlib Parameters

```
# We may alternatively specify parameters using a Python dictionary as a paramMap
paramMap = {lr.maxIter: 20}
paramMap[lr.maxIter] = 30 # Specify 1 Param, overwriting the original maxIter.
paramMap.update({lr.regParam: 0.1, lr.threshold: 0.55}) # Specify multiple Params.

# You can combine paramMaps, which are python dictionaries.
paramMap2 = {lr.probabilityCol: "myProbability"} # Change output column name
paramMap2 = {lr.maxIter: 10}
paramMapCombined = paramMap.copy()
paramMapCombined.update(paramMap2)

# Now Learn a new model using the paramMapCombined parameters.
# paramMapCombined overrides all parameters set earlier via lr.set* methods.
model2 = lr.fit(training, paramMapCombined)

print("Model 2 fit used these parameters: ")
print_explainParamMap(model2, False)
```

Model 2 fit used these parameters:

- **aggregationDepth**: 2
- **elasticNetParam**: 0.0
- **family**: auto
- **featuresCol**: features
- **fitIntercept**: True
- **labelCol**: label
- **maxIter**: 10
- **predictionCol**: prediction
- **probabilityCol**: probability
- **rawPredictionCol**: rawPrediction
- **regParam**: 0.01
- **standardization**: True
- **threshold**: 0.5
- **tol**: 1e-06

# Spark MLlib Parameters

```
# Make predictions on test data using the Transformer.transform() method.  
# LogisticRegression.transform will only use the 'features' column.  
# Note that model2.transform() outputs a "myProbability" column instead of the usual  
# 'probability' column since we renamed the lr.probabilityCol parameter previously.  
prediction = model2.transform(test)  
result = prediction.select("features", "label", "myProbability", "prediction").collect()  
  
for row in result:  
    print(  
        "features=%s, label=%s -> prob=%s, prediction=%s"  
        % (row.features, row.label, row.myProbability, row.prediction)  
    )  
  
features=[-1.0,1.5,1.3], label=1.0 -> prob=[0.057073041710340174,0.9429269582896599], prediction=1.0  
features=[3.0,2.0,-0.1], label=0.0 -> prob=[0.9238522311704104,0.07614776882958973], prediction=0.0  
features=[0.0,2.2,-1.5], label=1.0 -> prob=[0.10972776114779419,0.8902722388522057], prediction=1.0
```

# Spark MLlib Featurization

- This module contains algorithms for working with features, roughly divided into these groups:
  - Feature extractors: Extracting features from “raw” data
  - Feature transformers: Scaling, converting, or modifying features
  - Feature selectors: Selecting a subset from a larger set of features
  - Locality Sensitive Hashing (LSH): This group of algorithms combines aspects of feature transformation with other algorithms

# Spark MLlib Featurization

## Feature Extractors

- HashingTF
- IDF & IDFModel
- Word2Vec & Word2VecModel
- CountVectorizer & CountVectorizerModel
- FeatureHasher

## Feature Transformers

- Tokenizer
- RegexTokenizer
- StopWordsRemover
- NGram
- Binarizer
- PCA
- PCAModel
- PolynomialExpansion
- DCT
- StringIndexer & StringIndexerModel
- IndexToString
- OneHotEncoder & OneHotEncoderModel
- OneHotEncoderEstimator
- VectorIndexer
- VectorIndexerModel
- Normalizer
- StandardScaler & StandardScalerModel
- MinMaxScaler & MinMaxScalerModel
- MaxAbsScaler & MaxAbsScalerModel
- Bucketizer
- ElementwiseProduct
- SQLTransformer
- VectorAssembler
- VectorSizeHint
- QuantileDiscretizer
- Imputer & ImputerModel

Not available in Python:

- Interaction

## Feature Selectors

- VectorSlicer
- RFormula & RFormulaModel
- ChiSqSelector & ChiSqSelectorModel

## Locality Sensitive Hashing (LSH)

- BucketedRandomProjectionLSH & BucketedRandomProjectionLSHModel
- MinHashLSH & MinHashLSHModel

# Spark MLlib Featurization

```
from pyspark.ml.feature import QuantileDiscretizer

data = [(0, 18.0), (1, 19.0), (2, 8.0), (3, 5.0), (4, 2.2)]
df = spark.createDataFrame(data, ["id", "hour"])

discretizer = QuantileDiscretizer(numBuckets=3, inputCol="hour", outputCol="result")

result = discretizer.fit(df).transform(df)
result.show()
```

```
+---+---+---+
| id|hour|result|
+---+---+---+
|  0|18.0|  2.0|
|  1|19.0|  2.0|
|  2| 8.0|  1.0|
|  3| 5.0|  1.0|
|  4| 2.2|  0.0|
+---+---+---+
```

# Spark MLlib Featurization

```
: from pyspark.ml.feature import Word2Vec

# Input data: Each row is a bag of words from a sentence or document.
documentDF = spark.createDataFrame(
    [
        ("Hi I heard about Spark".split(" ")),
        ("I wish Java could use case classes".split(" ")),
        ("Logistic regression models are neat".split(" ")),
    ],
    ["text"],
)

# Learn a mapping from words to Vectors.
word2Vec = Word2Vec(vectorSize=3, minCount=0, inputCol="text", outputCol="result")
model = word2Vec.fit(documentDF)

result = model.transform(documentDF)
for row in result.collect():
    text, vector = row
    print(f"Text: {text} => \nVector: {vector}\n")

Text: ['Hi', 'I', 'heard', 'about', 'Spark'] =>
Vector: [0.01781592555344105, 0.02203895077109337, 0.02248857170343399]

Text: ['I', 'wish', 'Java', 'could', 'use', 'case', 'classes'] =>
Vector: [-0.00022016598709991998, -0.0291545108027224, -0.01409794549856867]

Text: ['Logistic', 'regression', 'models', 'are', 'neat'] =>
Vector: [-0.03180776406079531, 0.0591656094416976, 0.005205358564853668]
```

# Spark MLlib Featurization

```
from pyspark.ml.feature import FeatureHasher

dataset = spark.createDataFrame(
    [
        (2.2, True, "1", "foo"),
        (3.3, False, "2", "bar"),
        (4.4, False, "3", "baz"),
        (5.5, False, "4", "foo"),
    ],
    ["real", "bool", "stringNum", "string"],
)

hasher = FeatureHasher(
    inputCols=["real", "bool", "stringNum", "string"], outputCol="features"
)

featurized = hasher.transform(dataset)
featurized.show(truncate=False)
```

```
+-----+-----+
|real|bool |stringNum|string|features           |
+-----+-----+
|2.2 |true |1       |foo    |((262144,[174475,247670,257907,262126],[2.2,1.0,1.0,1.0])|
|3.3 |false|2       |bar    |((262144,[70644,89673,173866,174475],[1.0,1.0,1.0,3.3]) |
|4.4 |false|3       |baz    |((262144,[22406,70644,174475,187923],[1.0,1.0,4.4,1.0]) |
|5.5 |false|4       |foo    |((262144,[70644,101499,174475,257907],[1.0,1.0,5.5,1.0]) |
+-----+-----+
```

# Spark MLlib Featurization

```
from pyspark.ml.feature import StringIndexer

df = spark.createDataFrame(
    [
        (0, "a"),
        (1, "b"),
        (2, "c"),
        (3, "a"),
        (4, "a"),
        (5, "c"),
        (6, "d"),
        (7, "a"),
        (8, "d"),
    ],
    ["id", "category"],
)

indexer = StringIndexer(inputCol="category", outputCol="categoryIndex")
indexed = indexer.fit(df).transform(df)
indexed.show()
```

```
+---+-----+
| id|category|categoryIndex|
+---+-----+
|  0|      a|          0.0|
|  1|      b|          3.0|
|  2|      c|          1.0|
|  3|      a|          0.0|
|  4|      a|          0.0|
|  5|      c|          1.0|
|  6|      d|          2.0|
|  7|      a|          0.0|
|  8|      d|          2.0|
+---+-----+
```

# Spark MLlib Featurization

```
from pyspark.ml.feature import StringIndexer

df = spark.createDataFrame(
    [
        (0, "a"),
        (1, "b"),
        (2, "c"),
        (3, "a"),
        (4, "a"),
        (5, "c"),
        (6, "d"),
        (7, "a"),
        (8, "d"),
    ],
    ["id", "category"],
)
indexer = StringIndexer(inputCol="category", outputCol="categoryIndex")
indexed = indexer.fit(df).transform(df)
indexed.show()
```

+-----+-----+ <th>  id category categoryIndex <th>+-----+-----+</th></th>	id category categoryIndex  <th>+-----+-----+</th>	+-----+-----+
	0      a       0.0	
	1      b       3.0	
	2      c       1.0	
	3      a       0.0	
	4      a       0.0	
	5      c       1.0	
	6      d       2.0	
	7      a       0.0	
	8      d       2.0	

# Spark MLlib Featurization

```
from pyspark.ml.feature import VectorIndexer

data = spark.read.format("libsvm").load(
    "/usr/local/spark-2.4.3-bin-hadoop2.7/data/mllib/sample_libsvm_data.txt"
)

indexer = VectorIndexer(inputCol="features", outputCol="indexed",
    maxCategories=10)
indexerModel = indexer.fit(data)

categoricalFeatures = indexerModel.categoryMaps
print(
    f"Chose {len(categoricalFeatures)} categorical "
    f"features: {[str(k) for k in categoricalFeatures.keys()]}"
)

# Create new column "indexed" with categorical values transformed to indices
indexedData = indexerModel.transform(data)
indexedData.show()
```

label	features	indexed
0.0	(692,[127,128,129...])	(692,[127,128,129...])
1.0	(692,[158,159,160...])	(692,[158,159,160...])
1.0	(692,[124,125,126...])	(692,[124,125,126...])
1.0	(692,[152,153,154...])	(692,[152,153,154...])
1.0	(692,[151,152,153...])	(692,[151,152,153...])
0.0	(692,[129,130,131...])	(692,[129,130,131...])
1.0	(692,[158,159,160...])	(692,[158,159,160...])
1.0	(692,[99,100,101,...])	(692,[99,100,101,...])
0.0	(692,[154,155,156...])	(692,[154,155,156...])
0.0	(692,[127,128,129...])	(692,[127,128,129...])
1.0	(692,[154,155,156...])	(692,[154,155,156...])
0.0	(692,[153,154,155...])	(692,[153,154,155...])
0.0	(692,[151,152,153...])	(692,[151,152,153...])
1.0	(692,[129,130,131...])	(692,[129,130,131...])
0.0	(692,[154,155,156...])	(692,[154,155,156...])
1.0	(692,[150,151,152...])	(692,[150,151,152...])
0.0	(692,[124,125,126...])	(692,[124,125,126...])
0.0	(692,[152,153,154...])	(692,[152,153,154...])
1.0	(692,[97,98,99,12...])	(692,[97,98,99,12...])
1.0	(692,[124,125,126...])	(692,[124,125,126...])

# Spark MLlib Featurization

```
from pyspark.ml.feature import Bucketizer

splits = [-float("inf"), -0.5, 0.0, 0.5, float("inf")]

data = [
    (-999.9,),
    (-0.5,),
    (-0.3,),
    (-0.4,),
    (-0.2,),
    (0.0,),
    (0.1,),
    (0.2,),
    (0.3,),
    (999.9,),
]
dataFrame = spark.createDataFrame(data, ["features"])

bucketizer = Bucketizer(
    splits=splits, inputCol="features", outputCol="bucketedFeatures"
)

# Transform original data into its bucket index.
bucketedData = bucketizer.transform(dataFrame)

print(f"Bucketizer output with {len(bucketizer.getSplits())-1} buckets")
bucketedData.show()
```

Bucketizer output with 4 buckets

features	bucketedFeatures
-999.9	0.0
-0.5	1.0
-0.3	1.0
-0.4	1.0
-0.2	1.0
0.0	2.0
0.1	2.0
0.2	2.0
0.3	2.0
999.9	3.0

# Spark MLlib Statistic

- A module used for correlation calculations, hypothesis testing, and column-wise summary statistics for vectors using summarizer
- Correlation:
  - `class pyspark.ml.stat.Correlation`: Supports Pearson's and Spearman's correlation to calculate correlation between two series of data

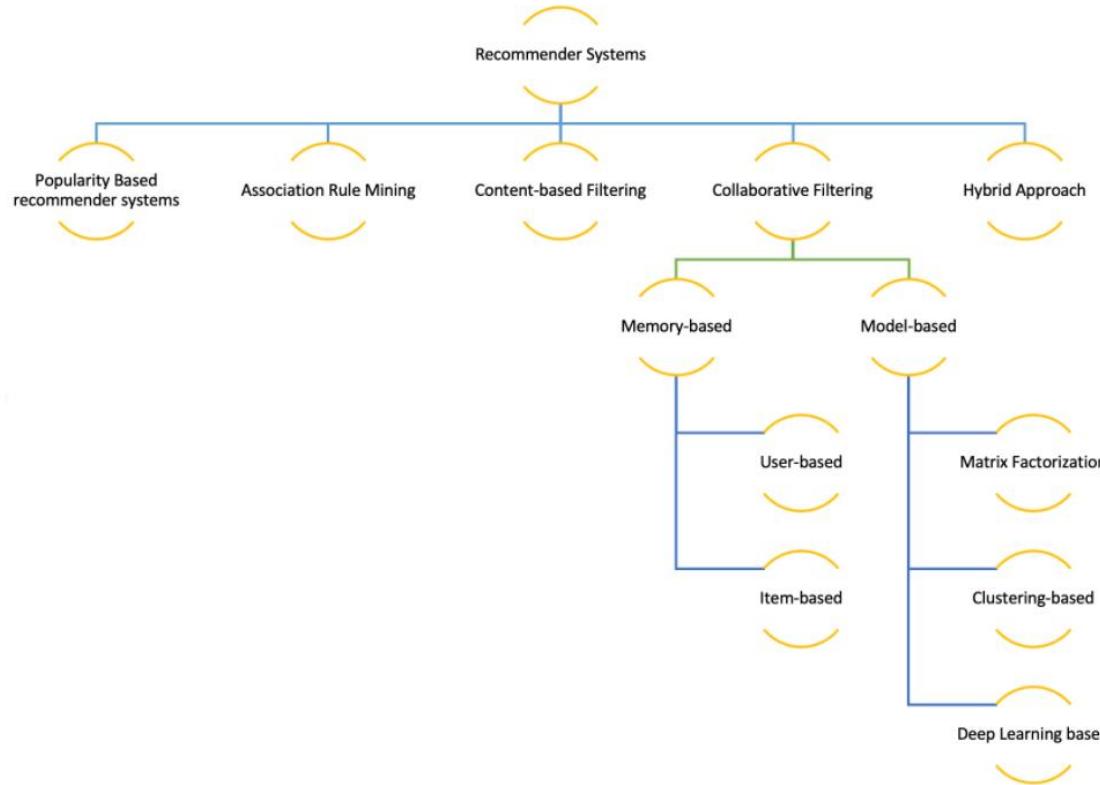
# Spark MLlib Statistic

- Hypothesis testing:
  - `class pyspark.ml.ChiSquareTest`: Conducts Pearson's independence test for every feature against the label
  - `class pyspark.ml.KolmogorovSmirnovTest`: Conduct the two-sided Kolmogorov-Smirnov (KS) test for data sampled from a continuous distribution
- Summarizer:
  - `class pyspark.ml.Summarizer` and `class pyspark.ml.SummaryBuilder`: Column-wise summary statistics for vectors

# Spark MLlib Statistic

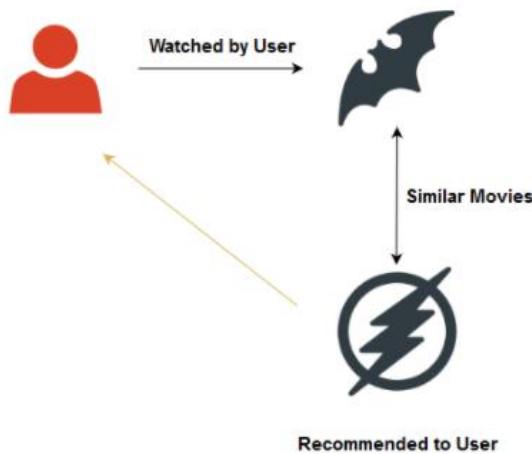
- Summarizer:
  - `class pyspark.ml.Summarizer` and  
`class pyspark.ml.SummaryBuilder`: Column-wise summary statistics for vectors

# Spark MLlib Recommendation System

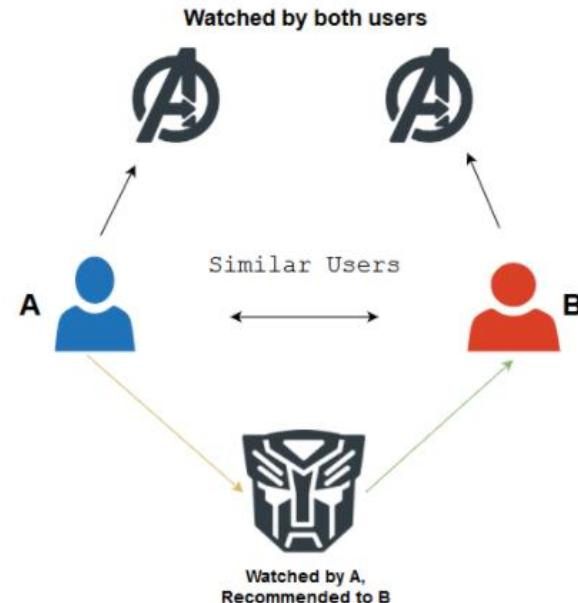


# Spark MLlib Recommendation System

**Content-based Filtering**



**Collaborative Filtering**



# Spark MLlib Recommendation System

Content-based recommendation system

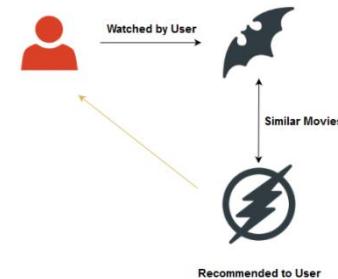
- Content-based recommendations based on a user's likes/dislikes
- Advantage:
  - Highly personalized for the user as it learns the user's preferences and tastes

# Spark MLlib Recommendation System

## Content-based recommendation system

Content-based Filtering

- Disadvantage
  - Finding appropriate features is hard
  - Recommendations for new users (how to know their profile)
  - Low quality recommendations might occurs as it doesn't consider what the neighbors think
  - Unless the user indicates their preference for a new category (e.g. genre of movie), no new categories will ever get recommended to the user

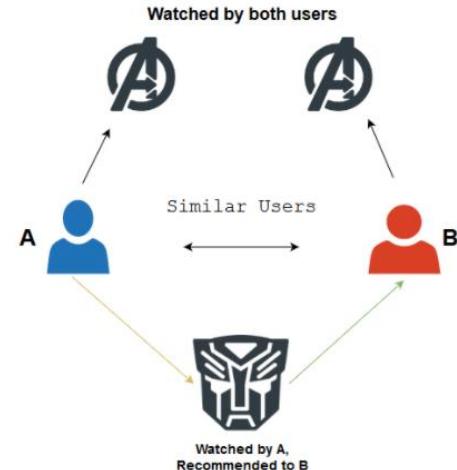


# Spark MLlib Recommendation System

## Collaborative filtering

- Make filtered predictions around the interest of users by collecting preferences or information from many users
- Working under the assumption that user might like items popular among their neighbors

Collaborative Filtering



# Spark MLlib Recommendation System

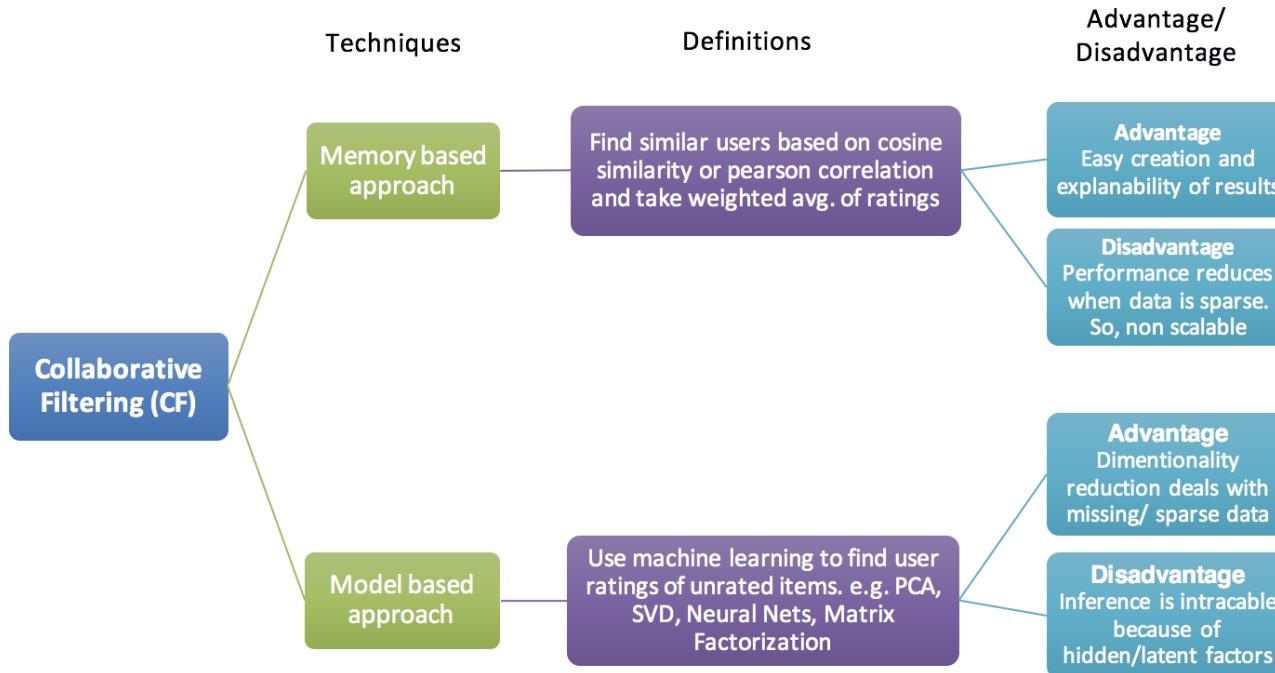
## Collaborative filtering

- Our use case:
  - Building a recommender system based on users rating movies
- The challenge:
  - We are dealing with an extremely sparse matrix; where more than 99% of entries are missing values

		Movie			
		E	F	G	H
User	A	1.0	4.0	3.5	
	B	2.0		5.0	
	C		3.5		4.0
	D		2.0	4.5	

# Spark MLlib Recommendation System

## Collaborative filtering



# Spark MLlib Recommendation System

Collaborative filtering – memory based

$$sim(u, u') = \cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{\|\mathbf{r}_u\| \|\mathbf{r}_{u'}\|} = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2} \sqrt{\sum_i r_{u'i}^2}}$$

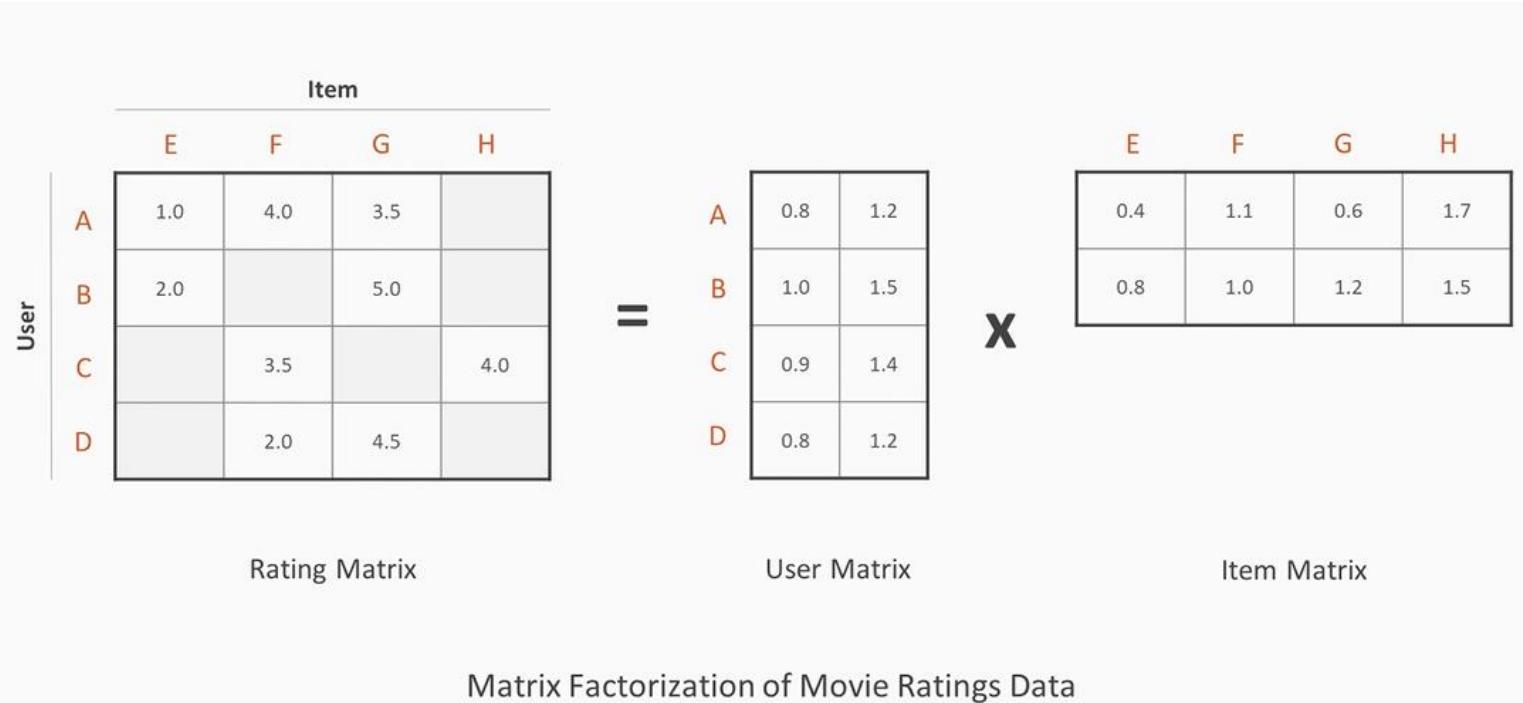
$$\hat{r}_{ui} = \sum_{u'} sim(u, u') r_{u'i}$$

		Movie			
		E	F	G	H
User	A	1.0	4.0	3.5	?
	B	2.0	?	5.0	
	C	1.0	3.5	4.0	4.0
	D		2.0	4.5	



# Spark MLlib Recommendation System

Collaborative filtering – model based



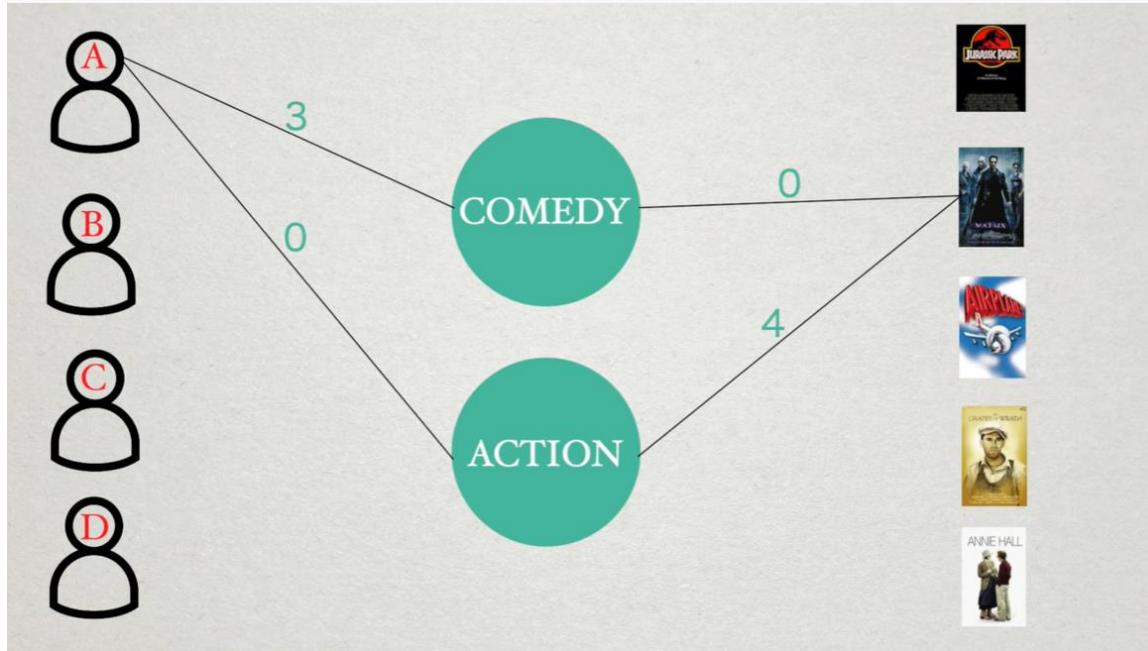
# Spark MLlib Recommendation System

Collaborative filtering – model based

	Jurassic Park	Star Wars	Avatar	Indiana Jones	Anne Hall
A		4		0	1
B	3	1	2		
C		2	3	1	
D	0			4	

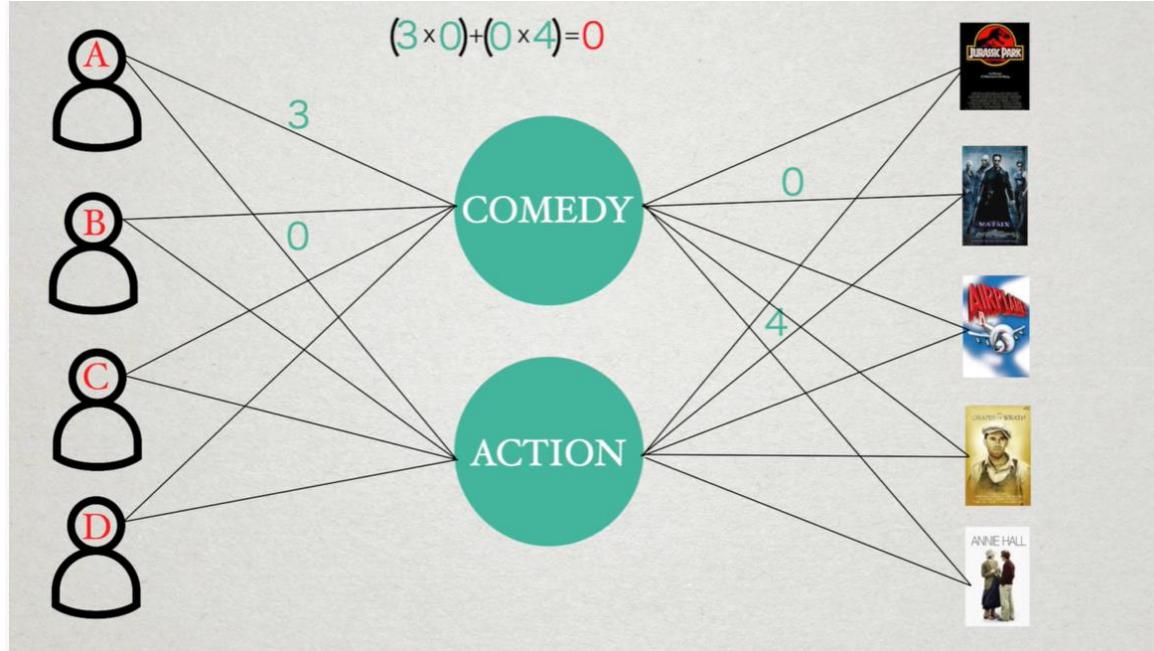
# Spark MLlib Recommendation System

Collaborative filtering – model based



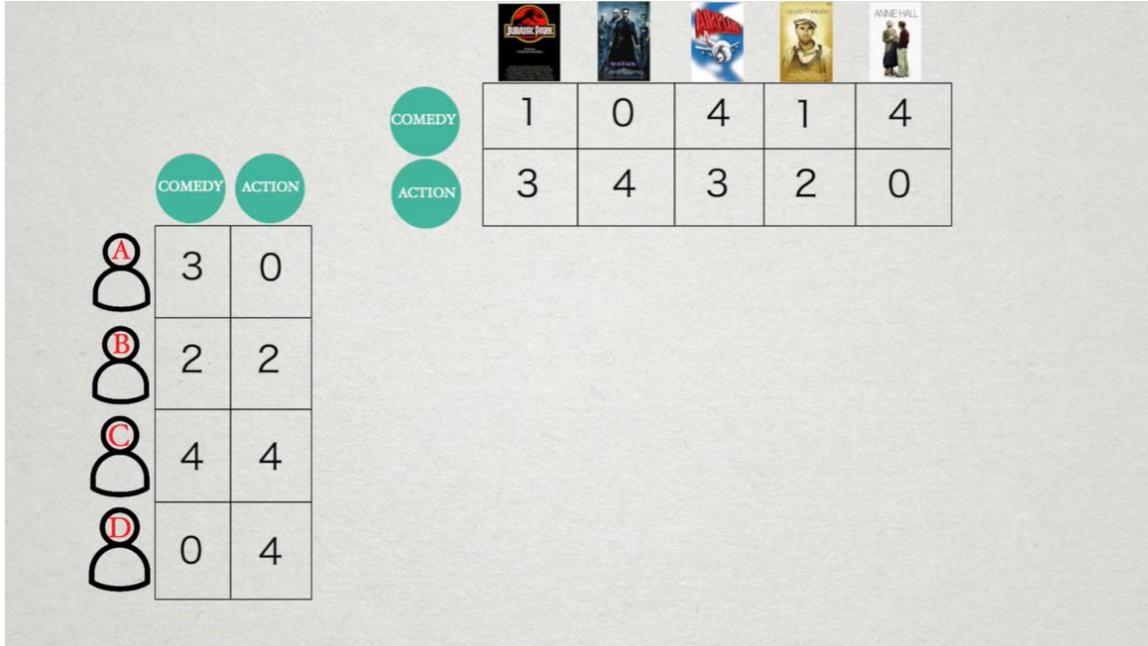
# Spark MLlib Recommendation System

Collaborative filtering – model based



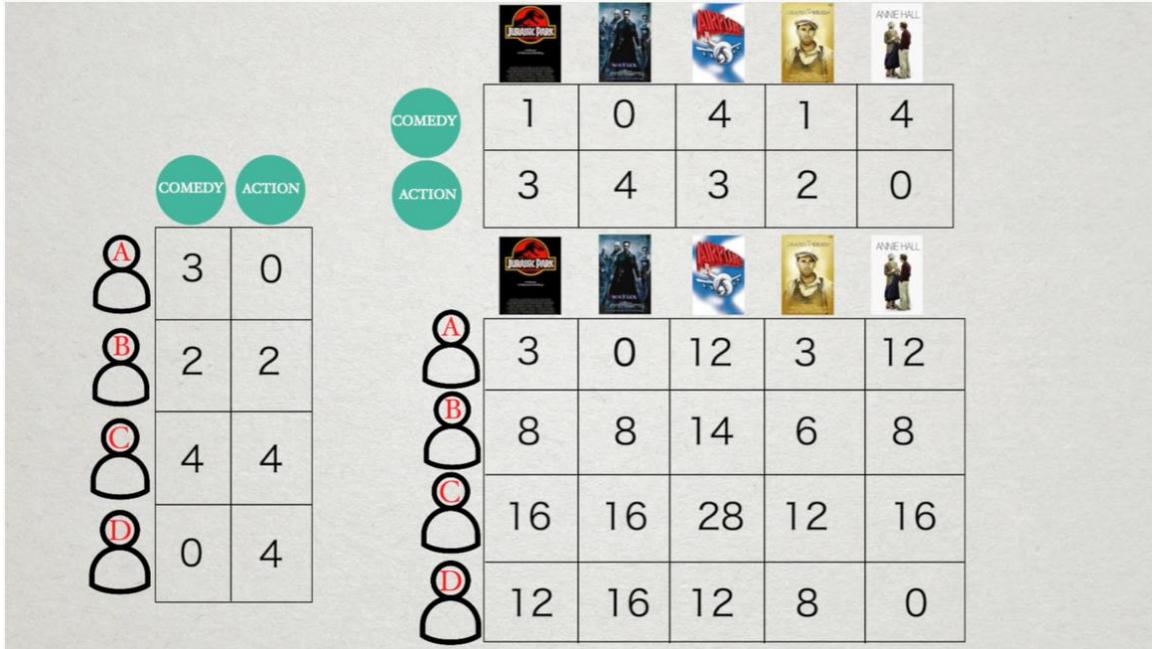
# Spark MLlib Recommendation System

Collaborative filtering – model based



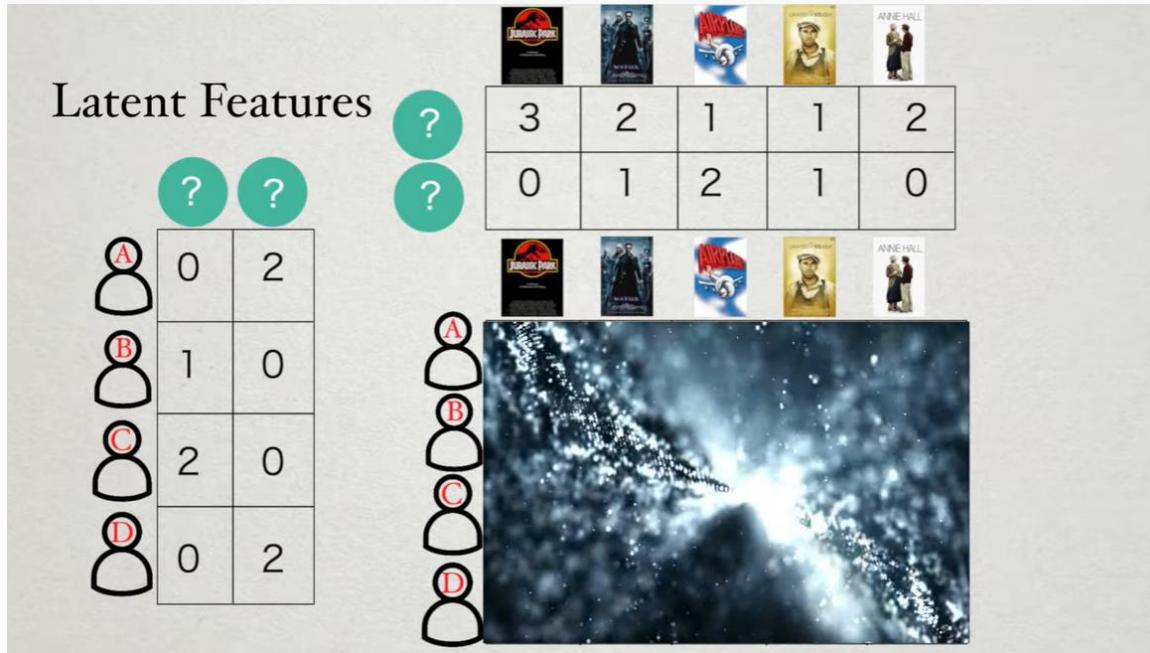
# Spark MLlib Recommendation System

Collaborative filtering – model based



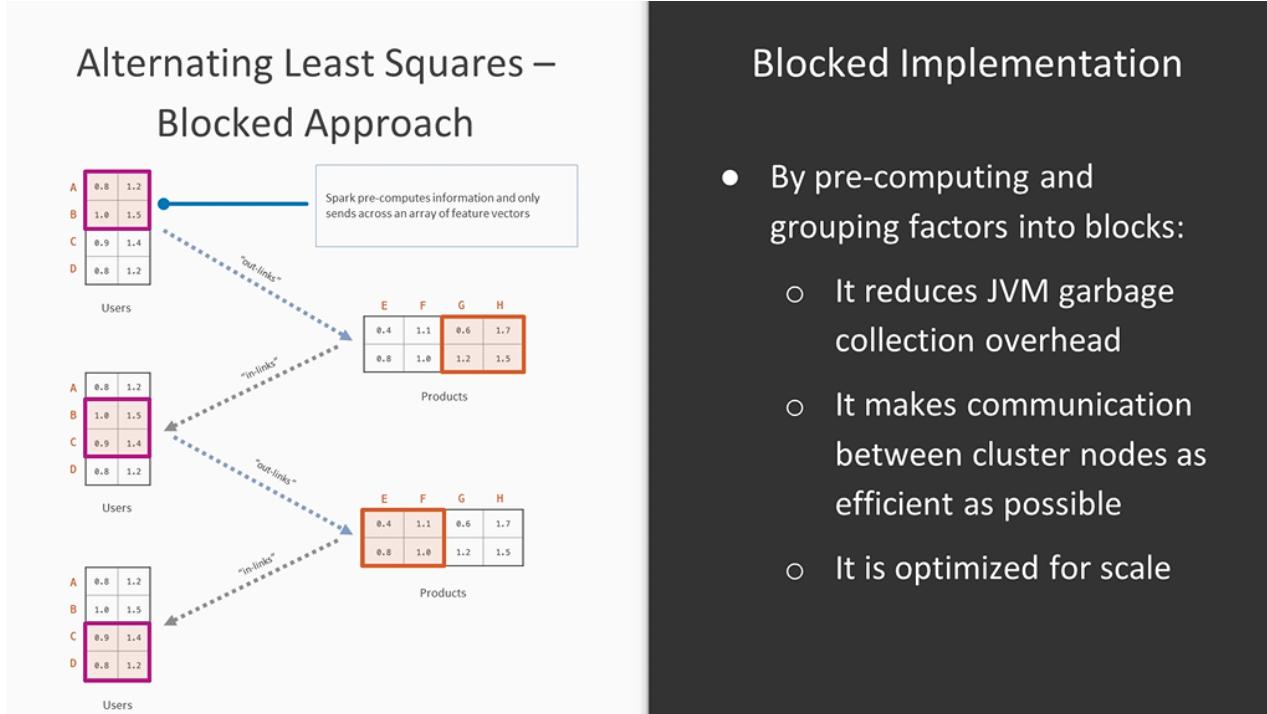
# Spark MLlib Recommendation System

Collaborative filtering – model based



# Spark MLlib Recommendation System

## Collaborative filtering – Alternating Least Squares



# Spark MLlib Recommendation System

## Collaborative filtering – pyspark.ml

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
from pyspark.sql import functions as f

spark = SparkSession.builder.appName("movieRecommendationPySpark").getOrCreate()

ratings = (
    spark.read.csv(
        path="ratings_small.csv",
        sep=",",
        header=True,
        quote='',
        schema="userId INT, movieId INT, rating DOUBLE, timestamp INT",
    )
    # .withColumn("timestamp", f.to_timestamp(f.from_unixtime("timestamp")))
    .select("userId", "movieId", "rating")
    .cache()
)
```

The ALS class has this signature:

```
class pyspark.ml.recommendation.ALS(
    rank=10,
    maxIter=10,
    regParam=0.1,
    numUserBlocks=10,
    numItemBlocks=10,
    implicitPrefs=False,
    alpha=1.0,
    userCol="user",
    itemCol="item",
    seed=None,
    ratingCol="rating",
    nonnegative=False,
    checkpointInterval=10,
    intermediateStorageLevel="MEMORY_AND_DISK",
    finalStorageLevel="MEMORY_AND_DISK",
    coldStartStrategy="nan",
)
```

# Spark MLlib Recommendation System

Collaborative filtering – pyspark.ml

```
ratings.show(10, False)
```

```
+---+---+---+
|userId|movieId|rating|
+---+---+---+
|1    |1      |4.0   |
|1    |3      |4.0   |
|1    |6      |4.0   |
|1    |47     |5.0   |
|1    |50     |5.0   |
|1    |70     |3.0   |
|1    |101    |5.0   |
|1    |110    |4.0   |
|1    |151    |5.0   |
|1    |157    |5.0   |
+---+---+---+
```

```
ratings.summary().show()
```

```
+-----+-----+-----+-----+
|summary|       userId|       movieId|       rating|
+-----+-----+-----+-----+
|  count| 100836| 100836| 100836|
|  mean |326.12756356856676|19435.2957177992| 3.501556983616962|
| stddev| 182.6184914635004|35530.9871987003|1.0425292390606342|
|  min  |          1|          1|      0.5|
| 25%  |         177|        1199|      3.0|
| 50%  |         325|        2991|      3.5|
| 75%  |         477|        8092|      4.0|
|  max  |         610|       193609|      5.0|
+-----+-----+-----+-----+
```

# Spark MLlib Recommendation System

## Collaborative filtering – pyspark.ml

```
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator

als = ALS(
    userCol="userId",
    itemCol="movieId",
    ratingCol="rating",
)
(training_data, validation_data) = ratings.randomSplit([8.0, 2.0])

evaluator = RegressionEvaluator(
    metricName="rmse", labelCol="rating", predictionCol="prediction"
)

model = als.fit(training_data)
predictions = model.transform(validation_data)
```

```
predictions.show(10, False)
```

```
+----+----+----+-----+
|userId|movieId|rating|prediction|
+----+----+----+-----+
| 91 | 471 | 1.0 | 3.0327234 |
| 409 | 471 | 3.0 | 4.679796 |
| 218 | 471 | 4.0 | 3.8422227 |
| 387 | 471 | 3.0 | 3.2297335 |
| 312 | 471 | 4.0 | 3.8340437 |
| 414 | 471 | 5.0 | 3.7297359 |
| 541 | 471 | 3.0 | 3.7804155 |
| 260 | 471 | 4.5 | 4.155776 |
| 609 | 833 | 3.0 | 1.6748803 |
| 111 | 1088 | 3.0 | 2.5603733 |
+----+----+----+-----+
only showing top 10 rows
```

```
rmse = evaluator.evaluate(predictions.na.drop())
```

```
print(rmse)
```

```
0.8786288108243542
```

# Spark MLlib Recommendation System

## Collaborative filtering – pyspark.ml

```
userFactors = model.userFactors
itemFactors = model.itemFactors
userFactors.sort('id').show(5, False)
itemFactors.sort('id').show(5, False)
import numpy as np

user91Feature = model.userFactors.filter(f.col('id') == 91).select(f.col('features')).rdd.flatMap(lambda x: x).collect()[0]
item471Feature = model.itemFactors.filter(f.col('id') == 471).select(f.col('features')).rdd.flatMap(lambda x: x).collect()[0]

print(user91Feature)
print(item471Feature)
print('Predicted rating of user 91 for movie 471: ' + str(np.dot(user91Feature, item471Feature)))

+---+
| id | features
+---+
|1 |[-1.1396486, 0.046824947, 0.08606943, -0.89530176, 0.381437, -0.2385697, 0.62992984, -1.1538332, 0.22461006, -0.9601014]
|2 |[-0.57027817, -0.54838854, 0.80637234, -0.97872186, 0.6183535, -0.042785455, 0.7865455, -0.4062991, -0.098041765, -0.47591898]
|3 |[0.24463238, 0.9600048, 0.98998576, -0.9759907, -0.40930456, 0.09272141, 1.3343368, -0.30400163, -0.33486682, -0.39383462]
|4 |[-0.16313235, -0.59598106, -0.5850466, -0.39807796, 0.89300704, -0.3094769, -0.053823743, -1.4678935, 0.88977605, -0.539779]
|5 |[-0.5790872, -0.30325407, -0.04732976, -0.60516685, 0.08863028, 0.18239452, 0.10536681, -1.547137, 0.7643438, -0.72799474]
+---+
only showing top 5 rows

+---+
| id | features
+---+
|1 |[-0.94216657, -0.7023109, 0.009909666, -0.79786617, 0.6519753, -0.20649733, 0.8735492, -1.1982708, 0.55822855, -0.54331917]
|2 |[-0.9681279, -0.47008696, 0.17003863, -0.4119318, 0.38166243, 0.20443514, 1.0768436, -1.1046029, 0.116564624, -0.63136214]
|3 |[-0.83121145, -0.31397244, 0.55451936, -0.7868329, 0.79168385, 0.24812174, 0.16610332, -1.0381806, -0.52568763, -0.93329155]
|4 |[-0.600918, -0.61637694, 0.5325797, -0.23614328, 0.7943548, 0.12294494, 0.17635046, -0.6797515, -0.04617397, -0.41124898]
|5 |[-0.5220082, -0.17868853, 0.27958348, -0.8080285, 0.7605523, 0.37655824, 0.3978313, -1.0887363, -0.10682703, -0.62507665]
+---+
only showing top 5 rows

[ -0.3994467258453369, -0.274524986743927, 0.34672558307647705, -0.8329096436500549, 0.2576223611831665, -0.07359780371189117, 1.2851195335388184, -0.4822205603122711, 0.11742987483739853, 0.8312146663665771]
[ -0.618400514125824, -0.36738458275794983, -0.43478521704673767, -0.10723729431629181, 0.4398992359638214, 0.1055564135313034, 0.754115641117096, -0.8633681535720825, 0.513541579246521, -1.7102802991867065]
Predicted rating of user 91 for movie 471: 3.032723343608766
```

```
predictions.show(10, False)
```

userId	movieId	rating	prediction
91	471	1.0	3.0327234

# Spark MLlib Recommendation System

## Collaborative filtering – pyspark.ml

```
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

parameter_grid = (
    ParamGridBuilder()
    .addGrid(als.rank, [5, 10])
    .addGrid(als.maxIter, [20])
    .addGrid(als.regParam, [0.05, 0.1])
    .build()
)

type(parameter_grid)

<class 'list'>

from pprint import pprint

pprint(parameter_grid)

[{'Param(parent='ALS_82b6ebc09a93', name='maxIter', doc='max number of iterations (>= 0).'): 20,
 Param(parent='ALS_82b6ebc09a93', name='rank', doc='rank of the factorization'): 5,
 Param(parent='ALS_82b6ebc09a93', name='regParam', doc='regularization parameter (>= 0).'): 0.05},
 {'Param(parent='ALS_82b6ebc09a93', name='maxIter', doc='max number of iterations (>= 0).'): 20,
 Param(parent='ALS_82b6ebc09a93', name='rank', doc='rank of the factorization'): 5,
 Param(parent='ALS_82b6ebc09a93', name='regParam', doc='regularization parameter (>= 0).'): 0.1},
 {'Param(parent='ALS_82b6ebc09a93', name='maxIter', doc='max number of iterations (>= 0).'): 20,
 Param(parent='ALS_82b6ebc09a93', name='rank', doc='rank of the factorization'): 10,
 Param(parent='ALS_82b6ebc09a93', name='regParam', doc='regularization parameter (>= 0).'): 0.05},
 {'Param(parent='ALS_82b6ebc09a93', name='maxIter', doc='max number of iterations (>= 0).'): 20,
 Param(parent='ALS_82b6ebc09a93', name='rank', doc='rank of the factorization'): 10,
 Param(parent='ALS_82b6ebc09a93', name='regParam', doc='regularization parameter (>= 0).'): 0.1}]
```

# Spark MLlib Recommendation System

## Collaborative filtering – pyspark.ml

```
rmse = evaluator.evaluate(predictions.na.drop())
print(rmse)
```

```
0.9151240171424524
```

```
model = crossval_model.bestModel
```

```
model.userFactors.show(5, False)
```

```
+-----+
|id |features
+-----+
|10 |[1.2530712, -0.2791498, -1.7960297, 0.48610744, 0.71415794]
|20 |[-0.015548448, -1.44367, -1.3121722, 0.062264383, -0.9449332]
|30 |[0.65103686, -0.57082814, -1.7171123, 0.38074148, -0.8411453]
|40 |[1.2579815, -1.3042885, -0.27397826, 0.42265755, -1.0877987]
|50 |[0.560652, -1.1323379, -0.21379222, -0.14114618, -0.7281288]
+-----+
only showing top 5 rows
```

# Spark MLlib Recommendation System

Collaborative filtering – pyspark.ml

```
predictions.toPandas()
```

	userId	movieId	rating	prediction
0	91	471	1.0	2.535147
1	409	471	3.0	4.819777
2	218	471	4.0	3.222253
3	387	471	3.0	3.149618
4	312	471	4.0	4.164871
...	...	...	...	...
19500	204	79008	3.5	4.002772
19501	522	79008	4.5	4.013596
19502	177	84374	3.5	2.969700
19503	432	84374	4.0	3.434984
19504	249	84374	3.5	3.237358

19505 rows × 4 columns

# Spark MLlib Recommendation System

Collaborative filtering – pyspark.ml

Top 5 Movies for each user, for all users

```
userID = 91

rec_all_users = model.recommendForAllUsers(5).cache()
rec_all_users.show(5, False)
rec_all_users.printSchema()

+-----+-----+
|userId|recommendations
+-----+-----+
|471   |[[[158783, 5.590863], [6818, 5.5682793], [8477, 5.514393], [68945, 5.241924], [96004, 5.241924]]]
|463   |[[[86377, 5.835226], [299, 5.645237], [26865, 5.3744984], [187593, 5.302268], [96004, 5.238351]]]
|496   |[[[70946, 7.0032015], [26865, 6.577416], [6818, 6.3157024], [3819, 6.225935], [306, 6.0877237]]]
|148   |[[[4256, 5.8173766], [112804, 5.343978], [49932, 5.2747917], [74946, 5.143712], [4442, 5.1404366]]]
|540   |[[[299, 6.086707], [87234, 5.9714355], [158872, 5.9553595], [68945, 5.949146], [96004, 5.949146]]]
+-----+
only showing top 5 rows

root
|-- userId: integer (nullable = false)
|-- recommendations: array (nullable = true)
|   |-- element: struct (containsNull = true)
|   |   |-- movieId: integer (nullable = true)
|   |   |-- rating: float (nullable = true)
```

# Spark MLlib Recommendation System

Collaborative filtering – pyspark.ml

```
recommendations_for_user91.show(5, False)
```

movieId	title	year
86377	Louis C.K.: Shameless	2007
70946	Troll 2	1990
26865	Fist of Legend (Jing wu ying xiong)	1994
299	Priest	1994
3819	Tampopo	1985

# Spark MLlib Recommendation System

Collaborative filtering – pyspark.ml

## Top unrated movies for a user

```
userID = 91

ratedMovies = ratings.filter(f.col('userId')==91).select('movieId').rdd.flatMap(lambda x: x).collect()

movies_to_be_rated = (
    ratings
    .filter(~ f.col('movieID').isin(ratedMovies))
    .select('movieId').distinct()
    .withColumn('userId',f.lit(userID))
)
movies_to_be_rated.show(5)
user_movie_predictions = model.transform(movies_to_be_rated)
user_movie_predictions.filter(~f.isnan('prediction')).orderBy('prediction',ascending=False).show(5)
```

```
+-----+-----+
|movieId|userId|
+-----+-----+
|  4| 91|
|  5| 91|
|  7| 91|
|  8| 91|
|  9| 91|
+-----+
only showing top 5 rows
```

```
+-----+-----+-----+
|movieId|userId|prediction|
+-----+-----+-----+
| 86377| 91| 5.7544503|
| 70946| 91| 5.4921703|
| 26865| 91| 5.4447317|
| 299| 91| 5.2966266|
| 3819| 91| 5.2113304|
+-----+
only showing top 5 rows
```

# Spark MLlib NLP

## MLlib: Main Guide

- Basic statistics
- Data sources
- Pipelines
- **Extracting, transforming and selecting features**
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

- Feature Extractors
  - TF-IDF
  - Word2Vec
  - CountVectorizer
  - FeatureHasher
- Feature Transformers
  - Tokenizer
  - StopWordsRemover
  - *n*-gram
  - Binarizer
  - PCA
  - PolynomialExpansion
  - Discrete Cosine Transform (DCT)
  - StringIndexer
- IndexToString
- OneHotEncoder
- VectorIndexer
- Interaction
- Normalizer
- StandardScaler
- RobustScaler
- MinMaxScaler
- MaxAbsScaler
- Bucketizer
- ElementwiseProduct
- SQLTransformer
- VectorAssembler
- VectorSizeHint
- QuantileDiscretizer
- Imputer

# Spark MLlib NLP

The screenshot shows the homepage of the Spark NLP website, [nlp.johnsnowlabs.com](https://nlp.johnsnowlabs.com). The page has a blue background with a network graph pattern. At the top, there is a navigation bar with links for Home, Docs, Learn, Models, Demo, and a search icon. Below the navigation bar, the John Snow LABS logo is displayed. The main content area features the text "Spark NLP: State of the Art Natural Language Processing" in large white font, followed by a subtitle "The first production grade versions of the latest deep learning NLP research". At the bottom, there are three buttons: "Get Started" (orange), "View Demo" (white), and "GitHub" (white).

nlp.johnsnowlabs.com

John Snow LABS

Home Docs Learn Models Demo

Spark NLP:  
State of the Art Natural Language Processing

The first production grade versions of the latest deep learning NLP research

Get Started View Demo GitHub

# Spark MLlib NLP

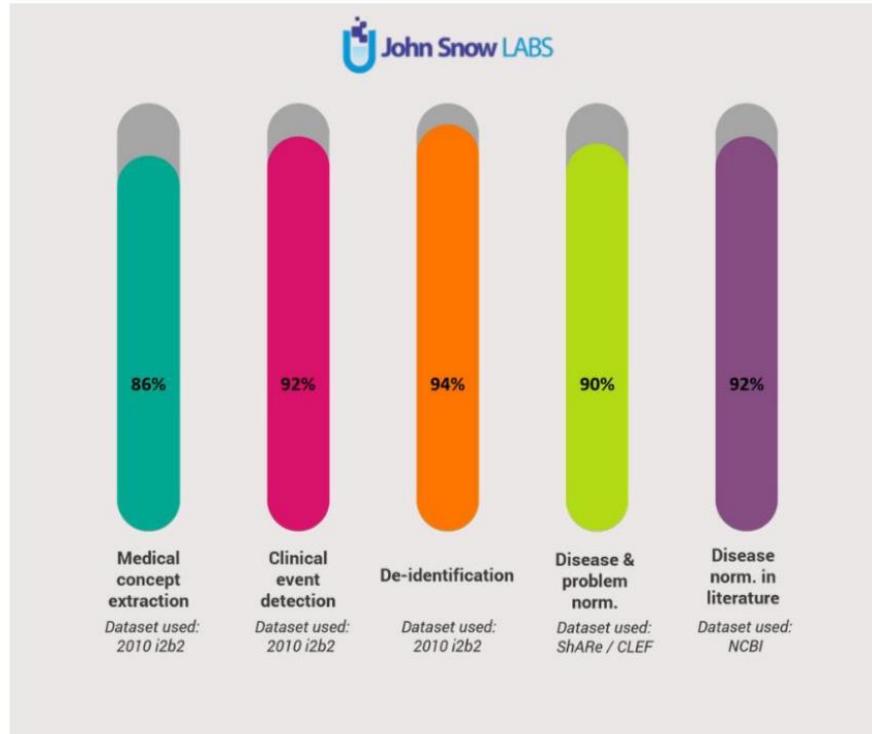


*Johnson & Johnson*



Spark NLP is already in use in enterprise projects for various use cases (2019)

# Spark MLlib NLP



# Spark MLlib NLP

## Data exploration

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
from pyspark.sql import functions as f

import pandas as pd
from IPython.core.display import display
import seaborn as sns

spark = SparkSession.builder.getOrCreate()

schema = "polarity FLOAT, id LONG, date_time STRING, query STRING, user STRING, text STRING"
```

# Spark MLlib NLP

## Data exploration: test data

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	4	3	Mon May	kindle2	tpryan	@stellargirl I loooooooooooooo my Kindle2. Not that the DX is cool, but the 2 is fantastic in its own right.											
2	4	4	Mon May	kindle2	vcu451	Reading my kindle2... Love it... Lee childs is good read.											
3	4	5	Mon May	kindle2	chadfu	Ok, first assesment of the #kindle2 ...it fucking rocks!!!											
4	4	6	Mon May	kindle2	SIX15	@kenburbary You'll love your Kindle2. I've had mine for a few months and never looked back. The new big one is huge! No need for remorse! :)											
5	4	7	Mon May	kindle2	yamarama @mikefish	Fair enough. But i have the Kindle2 and I think it's perfect :)											
6	4	8	Mon May	kindle2	GeorgeVH @richardebaker	no. it is too big. I'm quite happy with the Kindle2.											
7	0	9	Mon May	aig	Seth937	Fuck this economy. I hate aig and their non loan given asses.											
8	4	10	Mon May	jquery	dcostalis	jQuery is my new best friend.											
9	4	11	Mon May	twitter	PJ_King	Loves twitter											
10	4	12	Mon May	obama	mandanicc	how can you not love Obama? he makes jokes about himself.											
11	2	13	Mon May	obama	jpeb	Check this video out -- President Obama at the White House Correspondents' Dinner <a href="http://bit.ly/IMXUM">http://bit.ly/IMXUM</a>											
12	0	14	Mon May	obama	kylesellers @Karoli	I firmly believe that Obama/Pelosi have ZERO desire to be civil. It's a charade and a slogan, but they want to destroy conservatism											
13	4	15	Mon May	obama	theviewfa	House Correspondents dinner was last night whoopi, barbara & sheri went, Obama got a standing ovation											
14	4	16	Mon May	nike	MumsFP	Watchin Espn..Jus seen this new Nike Commerical with a Puppet Lebron..sh*t was hilarious...LMAO!!!											
15	0	17	Mon May	nike	vincentx24	dear nike, stop with the flywire. that shit is a waste of science. and ugly. love, @vincentx24x											
16	4	18	Mon May	lebron	cameronw	#lebron best athlete of our generation, if not all time (basketball related) I don't want to get into inter-sport debates about __1/2											
17	0	19	Mon May	lebron	luv8242	I was talking to this guy last night and he was telling me that he is a die hard Spurs fan. He also told me that he hates LeBron James.											
18	4	20	Mon May	lebron	mattwill111	I love lebron <a href="http://bit.ly/2dDfUu">http://bit.ly/2dDfUu</a>											

# Spark MLlib NLP

## Data exploration: test data

Overall data count: 498

	summary	polarity	id	date_time	query	user	text
0	count	498	498	498	498	498	498
1	mean	2.0200803212851404	1867.2269076305222	None	46.0	None	None
2	stddev	1.6996858490577658	2834.891681137318	None	5.163977794943222	None	None
3	min	0.0	3	Fri May 15 06:45:54 UTC 2009	""booz allen""	5x1llz	""The Republican party is a bunch of anti-abortion zealots who couldn't draw flies to a dump." -- Neal Boortz (just now)
4	25%	0.0	388	None	40.0	None	None
5	50%	2.0	1013	None	50.0	None	None
6	75%	4.0	2367	None	50.0	None	None
7	max	4.0	14076	Wed May 27 23:59:18 UTC 2009	yanekees zedomax		zomg!!! I have a G2!!!!!!

# Spark MLlib NLP

## Data exploration: test data

	polarity	id	date_time	query	user	text
0	4.0	3	Mon May 11 03:17:40 UTC 2009	kindle2	tpryan	@stellargirl I loooooooooooooo my Kindle2. Not that the DX is cool, but the 2 is fantastic in its own right.
1	4.0	4	Mon May 11 03:18:03 UTC 2009	kindle2	vcu451	Reading my kindle2... Love it... Lee childs is good read.
2	4.0	5	Mon May 11 03:18:54 UTC 2009	kindle2	chadfu	Ok, first assesment of the #kindle2 ...it fucking rocks!!!
3	4.0	6	Mon May 11 03:19:04 UTC 2009	kindle2	SIX15	@kenburbary You'll love your Kindle2. I've had mine for a few months and never looked back. The new big one is huge! No need for remorse! :)
4	4.0	7	Mon May 11 03:21:41 UTC 2009	kindle2	yamarama	@mikefish Fair enough. But i have the Kindle2 and I think it's perfect :)
5	4.0	8	Mon May 11 03:22:00 UTC 2009	kindle2	GeorgeVHulme	@richardebaker no. it is too big. I'm quite happy with the Kindle2.

# Spark MLlib NLP

## Data exploration: train data

A	B	C	D	E	F
1 0	1467810369	Mon Apr 06 22:19:45	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda
2 0	1467810672	Mon Apr 06 22:19:49	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by texting it... and might cry as a res
3 0	1467810917	Mon Apr 06 22:19:53	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Managed to save 50% The rest go
4 0	1467811184	Mon Apr 06 22:19:57	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
5 0	1467811193	Mon Apr 06 22:19:57	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because
6 0	1467811372	Mon Apr 06 22:20:00	NO_QUERY	joy_wolf	@Kwesidei not the whole crew
7 0	1467811592	Mon Apr 06 22:20:03	NO_QUERY	mybirch	Need a hug
8 0	1467811594	Mon Apr 06 22:20:03	NO_QUERY	coZZ	@LOLTrish hey long time no see! Yes.. Rains a bit ,only a bit LOL ,I'm fine tha
9 0	1467811795	Mon Apr 06 22:20:05	NO_QUERY	2Hood4Hollywood	@Tatiana_K nope they didn't have it
10 0	1467812025	Mon Apr 06 22:20:09	NO_QUERY	mimismo	@twittera que me muera ?
11 0	1467812416	Mon Apr 06 22:20:16	NO_QUERY	erinx3leannexo	spring break in plain city... it's snowing
12 0	1467812579	Mon Apr 06 22:20:17	NO_QUERY	pardonlauren	I just re-pierced my ears
13 0	1467812723	Mon Apr 06 22:20:19	NO_QUERY	TLeC	@caregiving I couldn't bear to watch it. And I thought the UA loss was embarr
14 0	1467812771	Mon Apr 06 22:20:19	NO_QUERY	robobbierobert	@octolinz16 It it counts, idk why I did either. you never talk to me anymore
15 0	1467812784	Mon Apr 06 22:20:20	NO_QUERY	bayofwolves	@smarrison i would've been the first, but i didn't have a gun. not really thou

# Spark MLlib NLP

## Data exploration: train data

Overall data count: 1600000

	summary	polarity	id	date_time	query	user	text
0	count	1600000	1600000	1600000	1600000	1600000	1600000
1	mean	2.0	1.9988175522956276E9	None	None	4.325887521835714E9	None
2	stddev	2.000000625000293	1.9357607362267897E8	None	None	5.16273321845489E10	None
3	min	0.0	1467810369	Fri Apr 17 20:30:31 PDT 2009	NO_QUERY	000catnap000	exhausted
4	25%	0.0	1956912114	None	None	32508.0	None
5	50%	0.0	2002096128	None	None	130587.0	None
6	75%	4.0	2177066219	None	None	1100101.0	None
7	max	4.0	2329205794	Wed May 27 07:27:38 PDT 2009	NO_QUERY	zzzzeus111	?????5?0??*??&lt;&lt;----I DID NOT KNOW I CUD or HOW TO DO ALL DAT ON MY PHONE TIL NOW. WOW..MY LIFE IS NOW COMPLETE. JK.

# Spark MLlib NLP

## Data exploration: train data

polarity	id	date_time	query	user	text
0	0.0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_ @switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
1	0.0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
2	0.0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus @Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
3	0.0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF my whole body feels itchy and like its on fire
4	0.0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli @nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
5	0.0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf @Kwesidei not the whole crew
6	0.0	1467811592	Mon Apr 06 22:20:03 PDT 2009	NO_QUERY	mybirch Need a hug

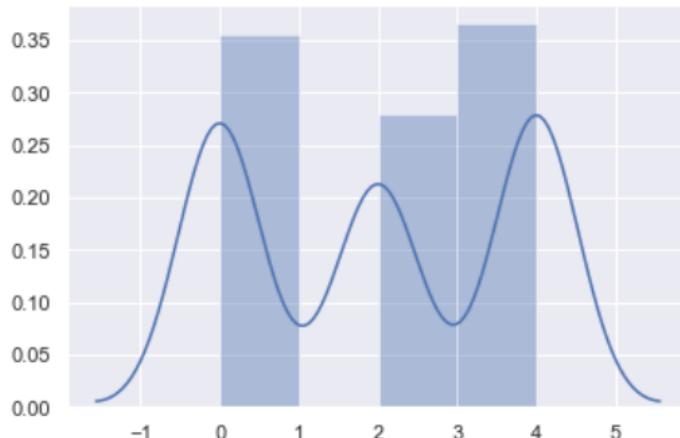
# Spark MLlib NLP

## Data exploration: distribution of polarity

```
df = raw_test_data.select("polarity").na.drop()  
print(f"No of rows with Polarity: {df.count()}/{raw_test_data.count()}")  
  
sns.distplot(df.toPandas())
```

No of rows with Polarity: 498/498

<matplotlib.axes.\_subplots.AxesSubplot at 0x26b618bd1d0>

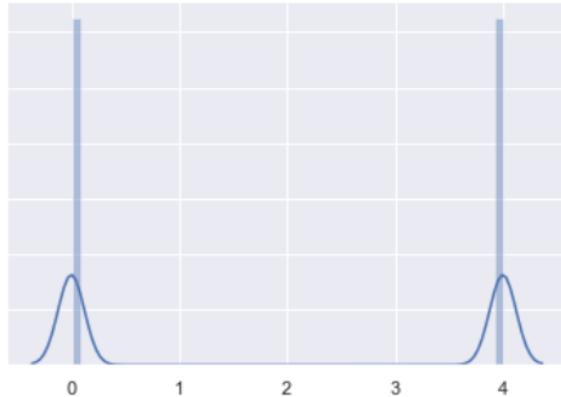


# Spark MLlib NLP

## Data exploration: distribution of polarity

```
df = raw_training_data.select("polarity").na.drop()  
print(f"No of rows with Polarity: {df.count()} / {raw_training_data.count()}")  
sns.distplot(df.toPandas())
```

No of rows with Polarity: 1600000 / 1600000  
<matplotlib.axes.\_subplots.AxesSubplot at 0x26b61bc9358>



```
polarity_df = raw_training_data.select("polarity").cache()  
polarity_df.groupBy("polarity").count().toPandas()
```

polarity	count
0	4.0 800000
1	0.0 800000

# Spark MLlib NLP

## Data exploration

Now it's time for us to write the raw data we intend to use to disk.

We're going to:

- keep the format CSV
- partition the data by polarity, this will create 2 subfolders inside our output folder
- repartition the data in 20 partitions: This will ensure that we have 20 smaller csv files per partition

```
raw_training_data.repartition(20).write.partitionBy("polarity").csv(OUTPUT_PATH, mode="overwrite")
```

Name

RAW
testdata.manual.2009.06.14.csv
training.1600000.processed.noemoticon.csv

# Spark MLlib NLP

## Data cleaning - Converting Date column

```
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")

schema_ddl = " polarity STRING, id LONG, date TIMESTAMP, query STRING , user string, text string "
spark_reader = spark.read.schema(schema_ddl)

simple_date_format = "EEE MMM dd HH:mm:ss zzz yyyy"

raw_data = spark_reader.csv(RAW_PATH, timestampFormat=simple_date_format)
raw_data.show(10)
raw_data.printSchema()

raw_data.summary().show()
```

# Spark MLlib NLP

## Data cleaning - Converting Date column

```
+-----+-----+-----+-----+-----+
|      id|      date|    query|     user|      text|polarity|
+-----+-----+-----+-----+-----+
|1833543437|2009-05-18 13:37:41|NO_QUERY| Kbrodes|Hi all. Have woke...|      0.0|
|1692704388|2009-05-04 10:35:20|NO_QUERY|spentwent|@kristindoll YOU ...|      0.0|
+-----+-----+-----+-----+-----+
only showing top 2 rows
```

```
root
|-- id: long (nullable = true)
|-- date: timestamp (nullable = true)
|-- query: string (nullable = true)
|-- user: string (nullable = true)
|-- text: string (nullable = true)
|-- polarity: string (nullable = true)
```

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

- Remove email-addresses and URLs
- Extract and then remove username (@mentions)
- Extract and then remove hashtag (#hash-tag)

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
# extract up to 6 twitter user names/handles to the output column `users_mentioned`  
  
user_regex = r"(@\w{1,15})"  
  
raw_data.select(  
    f.array_remove(  
        f.array(  
            f regexp_extract(f.col("text"), user_regex, 1),  
            f regexp_extract(  
                f.col("text"), """.join([f"{user_regex}.*?" for i in range(0, 2)])), 2  
            ),  
            f regexp_extract(  
                f.col("text"), """.join([f"{user_regex}.*?" for i in range(0, 3)])), 3  
            ),  
            f regexp_extract(  
                f.col("text"), """.join([f"{user_regex}.*?" for i in range(0, 4)])), 4  
            ),  
            f regexp_extract(  
                f.col("text"), """.join([f"{user_regex}.*?" for i in range(0, 5)])), 5  
            ),  
            f regexp_extract(  
                f.col("text"), """.join([f"{user_regex}.*?" for i in range(0, 6)])), 6  
            ),  
            "",  
            ).alias("users_mentioned"),  
            "text",  
            ).toPandas().head(35)
```

	users_mentioned	text
0	[]	Hi all. Have woken up with a cold. Boo! No tim...
1	[@kristindoll]	@kristindoll YOU NEVER ANSER ME
2	[]	Craaaaamps
3	[]	Our human mom just called. Her brand new Pruis...
4	[]	stomach growling. would give anything to be ab...
5	[]	I miss the sea, nak gi diving badly....but no ...
6	[@gradytwin]	@gradytwin i wish it was true for me too
7	[]	Why is it so hot??
8	[]	I need a hug
9	[]	Just saw a guy with the same (except black) &q...
10	[]	doinnnn' some hmwk, projects, studying, what a...
11	[]	Gosh there is nothing on tv
12	[@rrunyan]	I'm ignoring @rrunyan this morning because it'...
13	[@lesley007]	@lesley007 scales are evil. FACT!!! xxx

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
raw_data.select(  
    f.regexp_replace(f.col("text"), user_regex, "").alias("text"),  
    f.col("text").alias("original_text"),  
).toPandas().head(20)
```

	text	original_text
0	Hi all. Have woken up with a cold. Boo! No tim...	Hi all. Have woken up with a cold. Boo! No tim...
1	YOU NEVER ANSER ME	@kristindoll YOU NEVER ANSER ME
2	Craaaaamps	Craaaaamps
3	Our human mom just called. Her brand new Pruis...	Our human mom just called. Her brand new Pruis...
4	stomach growling. would give anything to be ab...	stomach growling. would give anything to be ab...
5	I miss the sea, nak gi diving badly....but no ...	I miss the sea, nak gi diving badly....but no ...
6	i wish it was true for me too	@gradytwin i wish it was true for me too
7	Why is it so hot??	Why is it so hot??
8	I need a hug	I need a hug
9	Just saw a guy with the same (except black) &q...	Just saw a guy with the same (except black) &q...

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

text	hashtags
@JasonCalacanis hey! we have to wait 'til sunday in the UK for the finale #lost  ... and then something mega urgent and mega important gatecrashes the list #GTD  Darn, my earphone cable snapped. Can't listen to music on my #iPhone while walking  wow #revision3.com is one big ad	[#lost]  [#GTD]  [#iPhone]  [#revision3]
@AlexLJ alas I think me may have the see flu that be going round this fine ship arhhh #Twittarrr  @ScottMonty wishing we had an official #SXSEMPIA Ford Car. Stuck with crappy Dodge Charger rental.  grr, latelatetatee! i have to shower, then get ready and leave at 10:15 to get into school. i FU...  @desertsong1 i only said #shitstack will be over because it's nearly 3am and i need sleep before ...  Where's the sunshine gone!???? #fb	[#Twittarrr]  [#SXSEMPIA]  [#mcfly]  [#shitstack]  [#fb]
still sad the #Mavs season is over. someone cheer me up  House small biz meeting not on CSPAN online #CPSIA	[#Mavs]  [#CPSIA]
@CXXG good night! #fixreplies #twitterfail #fixreplies #twitterfail #fixreplies #twitterfail #f... [#fixreplies, #twitterfail]	null
@gerrity What's more, we get a slew of hideous adverts every 23 seconds. The race is as much adv...	null
@robbarry @bugabundo We'll soon find out if #ubuntu uses one core or not on the PS3 soon. I used ...  @machineplay I'm so sorry you're having to go through this. Again. #therapyfail	[#ubuntu]  [#therapyfail]

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
hashtag_replace_regex = "#(\w{1,})"

_.select(f regexp_replace(f.col("text"), hashtag_replace_regex, "$1"), "hashtags").show(35, 100)

+-----+-----+
|           regexp_replace(text, #(\w{1,}), $1)| hashtags|
+-----+-----+
| @JasonCalacanis hey! we have to wait 'til sunday in the UK for the finale lost| [#lost]|
| ... and then something mega urgent and mega important gatecrashes the list GTD| [#GTD]|
| Darn, my earphone cable snapped. Can't listen to music on my iPhone while walking| [#iPhone]|
|                               wow revision3.com is one big ad| [#revision3]|
| @AlexLJ alas I think me may have the see flu that be going round this fine ship arhhh Twittarrr| [#Twittarrr]|
| @ScottMonty wishing we had an official SXSEMIA Ford Car. Stuck with crappy Dodge Charger rental.| [#SXSEMIA]|
| grr, latelatetatee! i have to shower, then get ready and leave at 10:15 to get into school. i FU...| [#mcfly]|
| @desertsong1 i only said shitstack will be over because it's nearly 3am and i need sleep before u...| [#shitstack]|
|                               Where's the sunshine gone!???? fb| [#fb]|
| still sad the Mavs season is over. someone cheer me up| [#Mavs]|
| House small biz meeting not on CSPAN online CPSIA| [#CPSIA]|
| @CXXG good night! fixreplies twitterfail fixreplies twitterfail fixreplies twitterfail fixrepli...| [#fixreplies, #twitterfail]|
| @gerrity What's more, we get a slew of hideous adverts every 23 seconds. The race is as much adv...| null|
| @robberry @bugabundo We'll soon find out if ubuntu uses one core or not on the PS3 soon. I used a...| [#ubuntu]|
| @machineplay I'm so sorry you're having to go through this. Again. therapyfail| [#therapyfail]|
| (via @etanowitz) There is an @orlandotweetup photo gallery http://tr.im/kf7h otweet OH YEAH PIC...| [#otweet]|
```

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
url_regex=r"((https?|ftp|file):\/\/{2,3})+([-\\w+@#/=%~|$?!:,.]*)|(www.)+([-\\w+@#/=%~|$?!:,.]*)"  
email_regex=r"[\w.-]+@[\\w.-]+\.[a-zA-Z]{1,}"
```

```
raw_data.select(  
    f regexp_replace(f.col("text"), email_regex, "").alias("text_no_email"),  
    f regexp_replace(f.col("text"), url_regex, "").alias("text_no_url"),  
    f.col("text").alias("original_text"),  
).toPandas().head(20)
```

	text_no_email	text_no_url	original_text
0	Hi all. Have woken up with a cold. Boo! No tim...	Hi all. Have woken up with a cold. Boo! No tim...	Hi all. Have woken up with a cold. Boo! No tim...
1	@kristindoll YOU NEVER ANSER ME	@kristindoll YOU NEVER ANSER ME	@kristindoll YOU NEVER ANSER ME
2	Craaaaamps	Craaaaamps	Craaaaamps
3	Our human mom just called. Her brand new Pruis...	Our human mom just called. Her brand new Pruis...	Our human mom just called. Her brand new Pruis...
4	stomach growling. would give anything to be ab...	stomach growling. would give anything to be ab...	stomach growling. would give anything to be ab...
5	I miss the sea, nak gi diving badly....but no ...	I miss the sea, nak gi diving badly....but no ...	I miss the sea, nak gi diving badly....but no ...
6	@gradytwin i wish it was true for me too	@gradytwin i wish it was true for me too	@gradytwin i wish it was true for me too
7	Why is it so hot??	Why is it so hot??	Why is it so hot??
8	I need a hug	I need a hug	I need a hug
9	Just saw a guy with the same (except black) &q...	Just saw a guy with the same (except black) &q...	Just saw a guy with the same (except black) &q...

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

started to think that Citi is in really deep s&t. Are they gonna survive the turmoil or are they gonna be the next AIG?  
"I'm listening to ""P.Y.T"" by Danny Gokey &lt;3 &lt;3 Aww

```
from pyspark.sql.functions import udf
import html

@udf
def html_unescape(s: str):
    return html.unescape(s)

raw_data.select(html_unescape("text")).show(35, 150)
```

html_unescape(text)
Hi all. Have woken up with a cold. Boo! No time to feel sorry for myself. My family are coming today. Hot lemon for me.  @kristindoll YOU NEVER ANSER ME  Craaaaamps  Our human mom just called. Her brand new Pruis just died on the causeway!!! stomach growling. would give anything to be able to open mouth wide and chomp down on a burger, or pizza, or even rice!  I miss the sea, nak gi diving badly....but no cuti  @gradytwin i wish it was true for me too  Why is it so hot??  I need a hug  Just saw a guy with the same (except black) "toppu" t-shirt as I have. I'm no longer unique!!  doinnnn' some hmwk, projects, studying, what a nice way to end this long weekend  Gosh there is nothing on tv  I'm ignoring @rrunyan this morning because it's Sunday and that's his weigh-in day...and he didn't do well this week.  @lesley007 scales are evil. FACT!!! xxx

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
raw_data = spark.read.schema(schema).csv(RAW_PATH)
clean_data = cleaning_process(raw_data)
clean_data.show()
clean_data.select("text").show(50, False)
```

id	date	query	user	text	polarity	users_mentioned	hashtags	original_text
1833543437	Sun May 17 23:37:...	NO_QUERY	Kbrodes	Hi all. Have woke...	0.0	null	null	Hi all. Have woke...
1692704388	Sun May 03 20:35:...	NO_QUERY	spentwent	YOU NEVER ANSE...	0.0	[@kristindoll]	null	@kristindoll YOU ...
1677671697	Sat May 02 03:21:...	NO_QUERY	Honey_Nut	Craaaaamps	0.0	null	null	Craaaaamps
1573908755	Tue Apr 21 03:25:...	NO_QUERY	dirtydogsoftsobe	Our human mom jus...	0.0	null	null	Our human mom jus...
1836234330	Mon May 18 07:37:...	NO_QUERY	chiemwei	stomach growling....	0.0	null	null	stomach growling....
1833057837	Sun May 17 22:10:...	NO_QUERY	tini_hotfm	I miss the sea, n...	0.0	null	null	I miss the sea, n...
1695573341	Mon May 04 06:27:...	NO_QUERY	D3vouring	i wish it was tr...	0.0	[@gradytwin]	null	@gradytwin i wish...
1834138334	Mon May 18 01:46:...	NO_QUERY	jessyflores	Why is it so hot??	0.0	null	null	Why is it so hot??
1760069887	Sun May 10 20:35:...	NO_QUERY	DarianFroseth	I need a hug	0.0	null	null	I need a hug
1825085705	Sun May 17 04:16:...	NO_QUERY	mmazur	Just saw a guy wi...	0.0	null	null	Just saw a guy wi...
1836140621	Mon May 18 07:27:...	NO_QUERY	meljonasxo	doinnnn' some hmw...	0.0	null	null	doinnnn' some hmw...
1556303497	Sat Apr 18 22:20:...	NO_QUERY	Bridgetgarz	Gosh there is not...	0.0	null	null	Gosh there is not...
1686547885	Sun May 03 05:53:...	NO_QUERY	willow1999	I'm ignoring thi...	0.0	[@rrunyan]	null	I'm ignoring @rru...
1879980021	Thu May 21 23:43:...	NO_QUERY	iSlayer2009	scales are evil....	0.0	[@lesley007]	null	@lesley007 scales...
1826030215	Sun May 17 07:31:...	NO_QUERY	gaminette	omg sinus infecti...	0.0	null	null	omg sinus infecti...

# Spark MLlib NLP

## Data exploration

```
+-----+  
|text  
+-----+  
|Hi all. Have woken up with a cold. Boo! No time to feel sorry for myself. My family are coming today. Hot lemon for me.  
| YOU NEVER ANSER ME  
|Craaaaamps  
|Our human mom just called. Her brand new Pruis just died on the causeway!!  
|stomach growling. would give anything to be able to open mouth wide and chomp down on a burger, or pizza, or even rice!  
|I miss the sea, nak gi diving badly....but no cuti  
| i wish it was true for me too  
|Why is it so hot??  
|I need a hug  
|Just saw a guy with the same (except black) "toppu" t-shirt as I have. I'm no longer unique!  
|doinnnn some hmwk, projects, studying, what a nice way to end this long weekend  
|Gosh there is nothing on tv  
|I'm ignoring this morning because it's Sunday and that's his weigh-in day...and he didn't do well this week.  
| scales are evil. FACT!!! xxx  
|omg sinus infection, so that was you lurking behind yesterday's headache. just don't stay too long okay?  
|alone and sad  
|Working until close tonight. I'm going to miss the Hell's Kitchen finale...nooo  
|i miss my ugly wight!!!  
|I think I lost my best friend today...feeling blue  
| Its also known as glandular fever () I got ill about 2 weeks ago, still feeling miserable  
|Gahh, I just spilt hot chocolate all down my top and burnt myself a bit  
| hiiiiiiiiiiiiii!!! i am following u! LOL ps. i miss u guys and my neighbor! LOL dark knight isn't following me on twitter!
```

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
raw_data.count()
```

```
1600000
```

```
clean_data.count()
```

```
1600000
```

```
clean_data.filter("text == ''").show(1000)
```

id	date	query	user	text	polarity	users_mentioned	hashtags	original_text
1823933200	Sat May 16 23:27:....	NO_QUERY	NOTjanelle		0.0	[@chriswantsfood]	null	@chriswantsfood
1753166696	Sun May 10 00:24:....	NO_QUERY	Sweet_Candii		0.0	[@Stealth_Tricia]	null	@Stealth_Tricia
1794571767	Thu May 14 06:06:....	NO_QUERY	msfitznham		0.0	[@demongirly]	null	@demongirly
1693024371	Sun May 03 21:21:....	NO_QUERY	juuleeya		0.0	[@fanficaholic]	null	@fanficaholic
1823473643	Sat May 16 22:04:....	NO_QUERY	michelle_dunlap		0.0	[@smoulderinsea]	null	@smoulderinsea
1556285000	Sat Apr 18 22:16:....	NO_QUERY	xcassiegottox		0.0	[@staticxage]	null	@staticxage
1565687892	Mon Apr 20 07:20:....	NO_QUERY	7arfal3ain		0.0	[@pearly_uea]	null	@pearly_uea
1834489373	Mon May 18 03:07:....	NO_QUERY	rehmxo		0.0	[@shaunjumpnow]	null	@shaunjumpnow
1468112539	Mon Apr 06 23:46:....	NO_QUERY	MissPassion		0.0	[@thecoolestout]	null	@thecoolestout
1956202634	Thu May 28 21:23:....	NO_QUERY	BLAK_OUT		0.0	[@shortyjunior]	null	@shortyjunior
1978920897	Sun May 31 00:56:....	NO_QUERY	desplesda		0.0	[@TheRealBnut]	null	@TheRealBnut
1982409707	Sun May 31 11:09:....	NO_QUERY	lpt21		0.0	[@The_Brew_Co]	null	@The_Brew_Co

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
: df = (
    df_clean
    # Remove all numbers
    .withColumn("text", f regexp_replace(f.col("text"), "[^a-zA-Z]", " "))
    # Remove all double/multiple spaces
    .withColumn("text", f regexp_replace(f.col("text"), " +", " "))
    # Remove leading and trailing whitespaces
    .withColumn("text", f.trim(f.col("text"))))
    # Ensure we don't end up with empty rows
    .filter("text != ''")
)
data = df.select("text", "polarity").coalesce(3).cache()
```

# Spark MLlib NLP

## Data cleaning - Cleaning the tweet text

```
print(df_clean.count())
print(df.count())
```

1600000  
1596232

df.toPandas()								
polarity	id	date_time	query	user	text	original_text		
0	4.0	3	2009-05-11 03:17:40	kindle2	tpryan	I loooooooooooooo my Kindle Not that the DX is cool but the is fantastic in its own right	@stellargirl I loooooooooooooo my Kindle2. Not that the DX is cool, but the 2 is fantastic in its own right.	
1	4.0	4	2009-05-11 03:18:03	kindle2	vcu451	Reading my Kindle Love it Lee childs is good read	Reading my kindle2... Love it... Lee childs is good read.	
2	4.0	5	2009-05-11 03:18:54	kindle2	chadfu	Ok first assessment of the kindle it fucking rocks	Ok, first assesment of the #kindle2 ...it fucking rocks!!!	
3	4.0	6	2009-05-11 03:19:04	kindle2	SIX15	You'll love your Kindle I've had mine for a few months and never looked back The new big one is huge No need for remorse	@kenburbary You'll love your Kindle2. I've had mine for a few months and never looked back. The new big one is huge! No need for remorse! :)	
4	4.0	7	2009-05-11 03:21:41	kindle2	yamarama	Fair enough But i have the Kindle and I think it's perfect	@mikefish Fair enough. But i have the Kindle2 and I think it's perfect :)	
...	...	...	...	...	...	...	...	...
493	2.0	14072	2009-06-14 04:31:43	latex	proggit	Ask Programming LaTeX or InDesign submitted by calcio1 link comment	Ask Programming: LaTeX or InDesign?: submitted by calcio1 [link] [1 comment] http://tinyurl.com/myfmf7	
494	0.0	14073	2009-06-14 04:32:17	latex	sam33r	On that note I hate Word I hate Pages I hate LaTeX There I said it I hate LaTeX All you TEXN RDS can come kill me now	On that note, I hate Word. I hate Pages. I hate LaTeX. There, I said it. I hate LaTeX. All you TEXN3RDS can come kill me now.	
495	4.0	14074	2009-06-14 04:36:34	latex	iamtheonlyjosie	Ahhh back in a real text editing environment I LaTeX	Ahhh... back in a *real* text editing environment. I &lt;3 LaTeX.	
496	0.0	14075	2009-06-14 21:36:07	iran	plutopup7	Trouble in Iran I see Hmm Iran Iran so far away flockofseagulls were geopolitically correct	Trouble in Iran, I see. Hmm. Iran. Iran so far away. #flockofseagulls were geopolitically correct	
497	0.0	14076	2009-06-14 21:36:17	iran	captain_pete	Reading the tweets coming out of Iran The whole thing is terrifying and incredibly sad	Reading the tweets coming out of Iran... The whole thing is terrifying and incredibly sad...	

# Spark MLlib NLP

## Building model

```
(training_data, validation_data, test_data) = data.randomSplit([0.98, 0.01, 0.01], seed=2020)
```

+ Code + Markdown

```
%time
from pyspark.ml.feature import (
    StopWordsRemover,
    Tokenizer,
    HashingTF,
    IDF,
)
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline

tokenizer = Tokenizer(inputCol="text", outputCol="words1")
stopwords_remover = StopWordsRemover(
    inputCol="words1",
    outputCol="words2",
    stopWords=StopWordsRemover.loadDefaultStopWords("english")
)
hashing_tf = HashingTF(
    inputCol="words2",
    outputCol="term_frequency",
)
idf = IDF(
    inputCol="term_frequency",
    outputCol="features",
    minDocFreq=5,
)
lr = LogisticRegression(labelCol="polarity")

semantic_analysis_pipeline = Pipeline(
    stages=[tokenizer, stopwords_remover, hashing_tf, idf, lr]
)
```

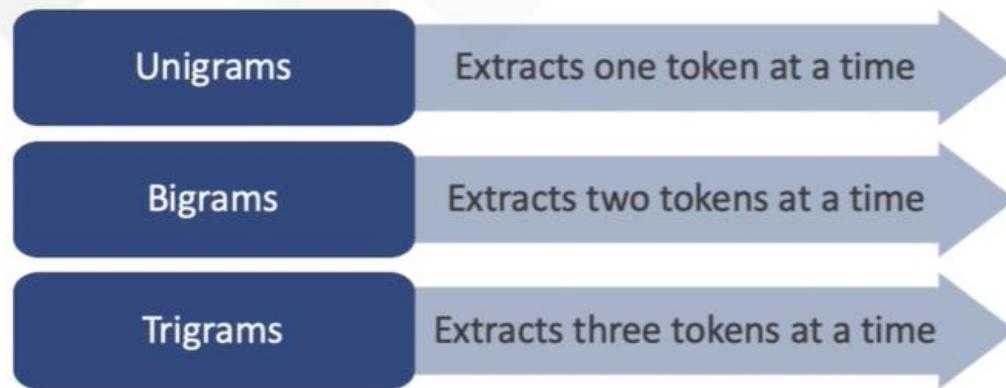
# Tokenization

Tokenization refers to the procedure of splitting a sentence into its constituent words.

Example:



Tokenization is of three types:



# Spark MLlib NLP

## Building model

```
df1 = tokenizer.transform(validation_data)
df1.show()
```

```
+-----+-----+
|      text|polarity|      words1|
+-----+-----+
|'AYYE PASS ME THE...|    4.0|['ayye, pass, me,...|
|'Honey' the chick...|    0.0|['honey', the, ch...|
|'What Canadians H...|    4.0|['what, canadians...|
|'allo Davina welc...|    4.0|['allo, davina, w...|
|'course it's not ...|    4.0|['course, it's, n...|
|'s bus just broke...|    0.0|['s, bus, just, b...|
|'s happy thought ...|    0.0|['s, happy, thoug...|
|'s screen size is...|    0.0|['s, screen, size...|
|'s voice makes me...|    4.0|['s, voice, makes...|
|'tis ok I posted ...|    4.0|['tis, ok, i, pos...|
|A BIG Welcome to ...|    4.0|[a, big, welcome,...|
|     A BOMB INATION|    4.0|[a, bomb, ination]|
|A Demi I hope no ...|    0.0|[a, demi, i, hope...|
|A Don't worry abo...|    4.0|[a, don't, worry,...|
|A GIRL IS SO LUCK...|    0.0|[a, girl, is, so,...|
|A Gabrielle wishe...|    0.0|[a, gabrielle, wi...|
|A How babies just...|    4.0|[a, how, babies, ...|
|A I am sorry abou...|    0.0|[a, i, am, sorry,...|
|A I hella miss yo...|    0.0|[a, i, hella, mis...|
|A I hope they cal...|    0.0|[a, i, hope, they...|
+-----+-----+
only showing top 20 rows
```

# Stop-Word Removal

Examples of stop-words are:

am	the	are	yourself
ourselves	her	between	until
there	is	other	most
below	while	does	being
against	under	which	whom
because	than	very	this



Support words and  
sentences

They help us to construct sentences. But they do not affect the meaning of the sentence in which they are present. Thus, we can safely ignore their presence.

# Spark MLlib NLP

## Building model

```
df2 = stopwords_remover.transform(df1)
df2.show()
```

	text polarity	words1	words2
'AYYE PASS ME THE...	4.0 ['ayye, pass, me,... ['ayye, pass, mon...		
'Honey' the chick...	0.0 ['honey', the, ch... ['honey', chicken...		
'What Canadians H...	4.0 ['what, canadians... ['what, canadians...		
'allo Davina welc...	4.0 ['allo, davina, w... ['allo, davina, w...		
'course it's not ...	4.0 ['course, it's, n... ['course, quantit...		
's bus just broke...	0.0 ['s, bus, just, b... ['s, bus, broke]		
's happy thought ...	0.0 ['s, happy, thoug... ['s, happy, thoug...		
's screen size is...	0.0 ['s, screen, size... ['s, screen, size...		
's voice makes me...	4.0 ['s, voice, makes... ['s, voice, makes...		
'tis ok I posted ...	4.0 ['tis, ok, i, pos... ['tis, ok, posted...		
A BIG Welcome to ...	4.0 [a, big, welcome,... [big, welcome, ne...		
A BOMB INATION	4.0  [a, bomb, ination]  [bomb, ination]		
A Demi I hope no ...	0.0 [a, demi, i, hope... [demi, hope, one,...		
A Don't worry abo...	4.0 [a, don't, worry,... [worry, hun, wors...		
A GIRL IS SO LUCK...	0.0 [a, girl, is, so,... [girl, lucky, won...		
A Gabrielle wishe...	0.0 [a, gabrielle, wi... [gabrielle, wishe...		
A How babies just...	4.0 [a, how, babies, ... [babies, light, g...		
A I am sorry abou...	0.0 [a, i, am, sorry,...  [sorry]		
A I hella miss yo...	0.0 [a, i, hella, mis...  [hella, miss]		
A I hope they cal...	0.0 [a, i, hope, they... [hope, calm, fun,...		

only showing top 20 rows

## TF-IDF

TF-IDF, or Term Frequency-Inverse Document Frequency, is a method of representing text data in a matrix format using numbers that quantify how much information these terms carry in the given documents.

The IDF for a given term is given by the following formula:

$$\text{term } j \text{ (idf}_j\text{)} = \log_{10} (N/\text{df}_j)$$

Here,  $\text{df}_j$  refers to the number of documents with term  $j$ .  $N$  is the total number of documents. Thus, the TF-IDF score for term  $j$  in document  $i$  will be as follows:

$$a_{ij} = \text{tf-idf}_{ij} = \text{tf}_{ij} \times \text{idf}_j = \text{tf}_{ij} \times \log_{10} (N/\text{df}_j)$$

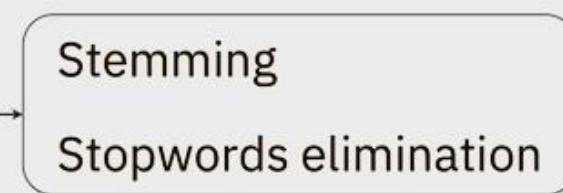
The TF-IDF value of a term  $t$  in given document:

$$\text{TFIDF}(t, d) = \boxed{\text{Term count within the document}} \times \boxed{\text{Document counts across the corpus}}$$
$$\text{TFIDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

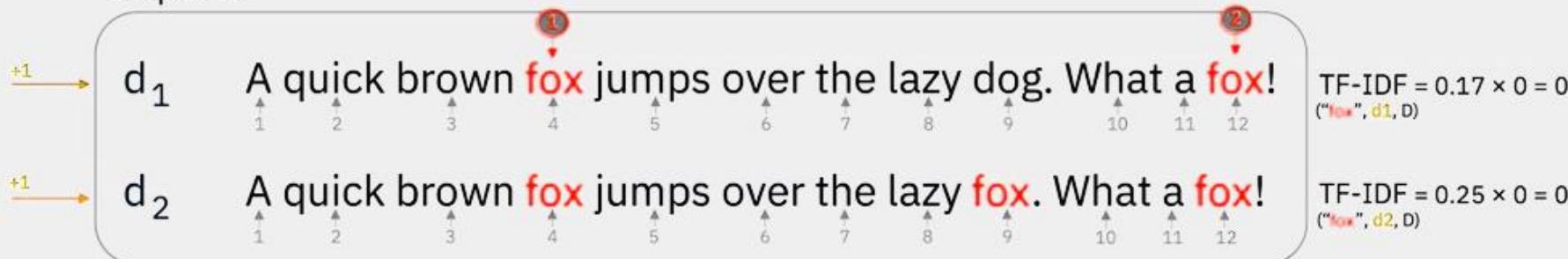
TFIDF value is specific to a single document  $d$ .

IDF depends on entire corpus.

Bag of Words representation



## Corpus D



Question: How word **fox** is relevant to corpus D documents?

Solution:

TF-IDF

TF is the frequency of any "term" in a given "document".

IDF is constant per corpus, and accounts for the ratio of documents that include that specific "term".

$$\text{TF}(\text{"fox"}, d_1) = 2 / 12 = 0.17$$

$$\text{TF}(\text{"fox"}, d_2) = 3 / 12 = 0.25$$

$$\text{IDF}(\text{"fox"}, D) = \log(2/2) = 0$$

# Spark MLlib NLP

## Building model

```
df3 = hashing_tf.transform(df2)
df3.show()
```

text polarity	words1	words2	term_frequency
'AYYE PASS ME THE...	4.0 [ 'ayye, pass, me,...	[ 'ayye, pass, mon... (262144,[31536,64...	
'Honey' the chick...	0.0 [ 'honey', the, ch...	[ 'honey', chicken... (262144,[55627,64...	
'What Canadians H...	4.0 [ 'what, canadians...	[ 'what, canadians... (262144,[16415,10...	
'allo Davina welc...	4.0 [ 'allo, davina, w...	[ 'allo, davina, w... (262144,[1512,124...	
'course it's not...	4.0 [ 'course, it's, n...	[ 'course, quantit... (262144,[43890,70...	
's bus just broke...	0.0 [ 's, bus, just, b...	[ 's, bus, broke (262144,[91694,92...	
's happy thought ...	0.0 [ 's, happy, thoug...	[ 's, happy, thoug... (262144,[12409,29...	
's screen size is...	0.0 [ 's, screen, size...	[ 's, screen, size... (262144,[92492,14...	
's voice makes me...	4.0 [ 's, voice, makes...	[ 's, voice, makes... (262144,[92492,10...	
'tis ok I posted ...	4.0 [ 'tis, ok, i, pos...	[ 'tis, ok, posted... (262144,[21894,64...	
A BIG Welcome to ...	4.0 [ a, big, welcome,...	[ big, welcome, ne... (262144,[64358,10...	
A BOMB INATION	4.0 [ a, bomb, ination]	[ bomb, ination) (262144,[26648,66...	
A Demi I hope no ...	0.0 [ a, demi, i, hope...	[ demi, hope, one,... (262144,[21823,61...	
A Don't worry abo...	4.0 [ a, don't, worry,...	[ worry, hun, wors... (262144,[117975,1...	
A GIRL IS SO LUCK...	0.0 [ a, girl, is, so,...	[ girl, lucky, won... (262144,[13781,33...	
A Gabrielle wishe...	0.0 [ a, gabrielle, wi...	[ gabrielle, wishe... (262144,[991,2078...	
A How babies just...	4.0 [ a, how, babies, ...	[ babies, light, g... (262144,[109208,2...	
A I am sorry abou...	0.0 [ a, i, am, sorry,...	[ sorry) (262144,[144961],...	
A I hella miss yo...	0.0 [ a, i, hella, mis...	[ hella, miss) (262144,[197515,2...	
A I hope they cal...	0.0 [ a, i, hope, they...	[ hope, calm, fun,... (262144,[23087,46...	

only showing top 20 rows

# Spark MLlib NLP

## Building model

```
df4 = idf.fit(df3).transform(df3)
df4.show()
```

	text polarity	words1	words2	term_frequency	features
'AYYE PASS ME THE...	4.0 ['ayye, pass, me,...	['ayye, pass, mon... (262144,[31536,64...	(262144,[31536,64...		
'Honey' the chick...	0.0 ['honey', the, ch...	['honey', chicken... (262144,[55627,64...	(262144,[55627,64...		
'What Canadians H...	4.0 ['what, canadians...	['what, canadians... (262144,[16415,10...	(262144,[16415,10...		
'allo Davina welc...	4.0 ['allo, davina, w...	['allo, davina, w... (262144,[1512,124...	(262144,[1512,124...		
'course it's not ...	4.0 ['course, it's, n...	['course, quantit... (262144,[43890,70...	(262144,[43890,70...		
's bus just broke...	0.0 ['s, bus, just, b...	['s, bus, broke] (262144,[91694,92...	(262144,[91694,92...		
's happy thought ...	0.0 ['s, happy, thoug...	['s, happy, thoug... (262144,[12409,29...	(262144,[12409,29...		
's screen size is...	0.0 ['s, screen, size...	['s, screen, size... (262144,[92492,14...	(262144,[92492,14...		
's voice makes me...	4.0 ['s, voice, makes...	['s, voice, makes... (262144,[92492,10...	(262144,[92492,10...		
'tis ok I posted ...	4.0 ['tis, ok, i, pos...	['tis, ok, posted... (262144,[21894,64...	(262144,[21894,64...		
A BIG Welcome to ...	4.0 [a, big, welcome,...	[big, welcome, ne... (262144,[64358,10...	(262144,[64358,10...		
A BOMB INATION	4.0 [a, bomb, ination]	[bomb, ination] (262144,[26648,66...	(262144,[26648,66...		
A Demi I hope no ...	0.0 [a, demi, i, hope...	[demi, hope, one,... (262144,[21823,61...	(262144,[21823,61...		
A Don't worry abo...	4.0 [a, don't, worry,...	[worry, hun, wors... (262144,[117975,1...	(262144,[117975,1...		
A GIRL IS SO LUCK...	0.0 [a, girl, is, so,...	[girl, lucky, won... (262144,[13781,33...	(262144,[13781,33...		
A Gabrielle wishe...	0.0 [a, gabrielle, wi...	[gabrielle, wishe... (262144,[991,2078...	(262144,[991,2078...		
A How babies just...	4.0 [a, how, babies, ...	[babies, light, g... (262144,[109208,2...	(262144,[109208,2...		
A I am sorry abou...	0.0 [a, i, am, sorry,...	[sorry] (262144,[144961],... (262144,[144961],...	(262144,[144961],...		
A I hella miss yo...	0.0 [a, i, hella, mis...	[hella, miss] (262144,[197515,2...	(262144,[197515,2...		
A I hope they cal...	0.0 [a, i, hope, they...	[hope, calm, fun,... (262144,[23087,46...	(262144,[23087,46...		

only showing top 20 rows

# Spark MLlib NLP

## Building model

```
lr.fit(df4).transform(df4).show()
```

text polarity	words1	words2	term_frequency	features	rawPrediction	probability prediction
'AYYE PASS ME THE...  4.0 ['ayye, pass, me,... [ 'ayye, pass, mon... (262144,[31536,64... (262144,[31536,64... [12.9780656017038... [0.21901779229395...  4.0						
'Honey' the chick...  0.0 ['honey', the, ch... [ 'honey', chicken... (262144,[55627,64... (262144,[55627,64... [12.0945489104097... [0.94421689167222...  0.0						
'What Canadians H...  4.0 ['what, canadians... [ 'what, canadians... (262144,[16415,10... (262144,[16415,10... [5.99369343435033... [0.18325323124123...  4.0						
'allo Davina welc...  4.0 ['allo, davina, w... [ 'allo, davina, w... (262144,[1512,124... (262144,[1512,124... [4.37111819424014... [0.00369593761715...  4.0						
'course it's not ...  4.0 ['course, it's, n... [ 'course, quantit... (262144,[43890,70... (262144,[43890,70... [7.52256788373932... [0.01878302997629...  4.0						
's bus just broke...  0.0 ['s, bus, just, b... [ 's, bus, broke... (262144,[91694,92... (262144,[91694,92... [10.5011690396112... [0.99354789740347...  0.0						
's happy thought ...  0.0 ['s, happy, thoug... [ 's, happy, thoug... (262144,[12409,29... (262144,[12409,29... [11.7400887292877... [0.55704370276780...  0.0						
's screen size is...  0.0 ['s, screen, size... [ 's, screen, size... (262144,[92492,14... (262144,[92492,14... [10.0581323946005... [0.60877308779168...  0.0						
's voice makes me...  4.0 ['s, voice, makes... [ 's, voice, makes... (262144,[92492,10... (262144,[92492,10... [8.03725845045519... [0.17077618299793...  4.0						
'tis ok I posted ...  4.0 ['tis, ok, i, pos... [ 'tis, ok, posted... (262144,[21894,64... (262144,[21894,64... [12.1513795255169... [0.89768486529536...  0.0						
[A BIG Welcome to ...  4.0 ['a, big, welcome,... [big, welcome, ne... (262144,[64358,10... (262144,[64358,10... [4.74536197045959... [7.71752216152728...  4.0						
[ A BOMB INATION  4.0 ['a, bomb, ination]  [bomb, ination](262144,[26648,66... (262144,[26648,66... [6.63379263319305... [0.40468526673953...  4.0						
[A Demi I hope no ...  0.0 [a, demi, i, hope... [demi, hope, one,... (262144,[21823,61... (262144,[21823,61... [11.731530643212... [0.95018414511760...  0.0						
[A Don't worry abo...  4.0 [a, don't, worry,... [worry, hun, wors... (262144,[117975,1... (262144,[117975,1... [9.12771068870963... [0.12421053641147...  4.0						
[A GIRL IS SO LUCK...  0.0 [a, girl, is, so,... [girl, lucky, won... (262144,[13781,33... (262144,[13781,33... [12.4060325488055... [0.42720958546993...  4.0						
[A Gabrielle wishe...  0.0 [a, gabrielle, wi... [gabrielle, wishe... (262144,[991,2078... (262144,[991,2078... [12.856609799347... [0.8697594962995...  0.0						
[A How babies just...  4.0 [a, how, babies, ... [babies, light, g... (262144,[109208,2... (262144,[109208,2... [5.62931364182358... [0.00755680436664...  4.0						
[A I am sorry abou...  0.0 [a, i, am, sorry,...  [sorry](262144,[144961,... (262144,[144961,... [7.97099816820809... [0.91272636828932...  0.0						
[A I hella miss yo...  0.0 [a, i, hella, mis... [hella, miss](262144,[197515,2... (262144,[197515,2... [8.58203798611087... [0.88237741144112...  0.0						
[A I hope they cal...  0.0 [a, i, hope, they... [hope, calm, fun... (262144,[23087,46... (262144,[23087,46... [7.65951633148478... [0.08779867011967...  4.0						

only showing top 20 rows

# Spark MLlib NLP

## Building model

```
: semantic_analysis_pipeline = Pipeline(  
    stages=[tokenizer, stopwords_remover, hashing_tf, idf, lr]  
)  
  
semantic_analysis_model = semantic_analysis_pipeline.fit(training_data)
```

```
spark = (  
    SparkSession.builder.appName("ModelTraining")  
        .config("spark.executor.memory", "4g")  
        .getOrCreate()  
)
```

# Spark MLlib NLP

# Building model

```
%time
trained_df = semantic_analysis_model.transform(training_data)
val_df = semantic_analysis_model.transform(validation_data)
test_df = semantic_analysis_model.transform(test_data)

trained_df.show()
val_df.show()
test_df.show()
```

text polarity	words1	words2	term_frequency	features	rawPrediction	probability prediction
'   0.0	[]	[]	(262144, [186171], ...)	(262144, [186171], ...)	[8.0781565229097, ...]	[0.46734275935458, ...] 4.0
'   4.0	[]	[]	(262144, [186171], ...)	(262144, [186171], ...)	[8.0781565229097, ...]	[0.46734275935458, ...] 4.0
'   4.0	[' ', ' ', ' ', ' ', '	[' ', ' ', ' ', ' ', '	(262144, [186171], ...)	(262144, [186171], ...)	[8.4514140491861, ...]	[0.64774113228464, ...] 0.8
'   PIMS	[], [pims]	[], [pims]	(262144, [186171, 2, ...)	(262144, [186171, 2, ...)	[2.504814608305, ...]	[0.54281514973836, ...] 0.0
'   cute hug facey	0.0 [], ' , cute, hug, ...	[], ' , cute, hug, ...	(262144, [23837, 65, ...)	(262144, [23837, 65, ...)	[7.979375133558, ...]	[0.41702838723787, ...] 4.0
i didn't mean...	0.0 [], ' , i, didn't, ...	[], ' , mean, any..., ...	(262144, [991, 3710, ...)	(262144, [991, 3710, ...)	[8.77058491428643, ...]	[0.77414425733478, ...] 0.8
it's a monkey...	4.0 [], ' , it's, a, m..., ...	[], ' , monkey, re..., ...	(262144, [31536, 60, ...)	(262144, [31536, 60, ...)	[9.034775371085, ...]	[0.85420003704538, ...] 0.0
it's good to ...	4.0 [], ' , it's, good, ...	[], ' , good, clea..., ...	(262144, [3955, 113, ...)	(262144, [3955, 113, ...)	[7.75654967633660, ...]	[0.3140966357582, ...] 4.0
is actually ...	4.0 [], ' , is, actu..., ...	[], ' , actually, ...	(262144, [109068, 1, ...)	(262144, [109068, 1, ...)	[7.88455743582298, ...]	[0.3721105979897, ...] 4.0
AMAZING heey ho...	4.0 [], amazing, heey, ...	[], ' , amazing, heey, ...	(262144, [23087, 51, ...)	(262144, [23087, 51, ...)	[6.27953674659944, ...]	[0.02321840068064, ...] 4.0
And she broke m...	0.0 [], and, she, bro..., ...	[], ' , broke, heart	(262144, [112352, 1, ...)	(262144, [112352, 1, ...)	[9.17251117294008, ...]	[0.88644382647143, ...] 0.0
Birthday Sex i...	4.0 [], birthday, sex, ...	[], ' , birthday, sex, ...	(262144, [86553, 12, ...)	(262144, [86553, 12, ...)	[5.79186750576178, ...]	[0.24673277052895, ...] 4.0
Bored on aim wa...	4.0 [], bored, on, ai..., ...	[], ' , bored, aim, w..., ...	(262144, [9958, 180, ...)	(262144, [9958, 180, ...)	[7.1616128850605, ...]	[0.12072903391416, ...] 4.0
But u were doin...	0.0 [], but, u, were, ...	[], ' , something, ...	(262144, [51783, 57, ...)	(262144, [51783, 57, ...)	[3.78601856335490, ...]	[0.17946376057583, ...] 4.0
' CRAZY IN LOVE I...	0.0 [], crazy, in, lo..., ...	[], ' , crazy, love, ...	(262144, [113024, 1, ...)	(262144, [113024, 1, ...)	[7.0967852033637, ...]	[0.1089459517827, ...] 4.0
'   D	0.0 [], ' , d	[], ' , d	(262144, [89538, 10, ...)	(262144, [89538, 10, ...)	[8.05038771619375, ...]	[0.45314700809016, ...] 4.0
' DVD out in US t...	0.0 [], dvd, out, in,...	[], ' , dvd, us, mont,...	(262144, [39216, 62, ...)	(262144, [39216, 62, ...)	[8.8983213333979, ...]	[0.81765739791268, ...] 0.0
' Dancing with th...	4.0 [], dancing, with, ...	[], ' , dancing, devi..., ...	(262144, [12409, 17, ...)	(262144, [12409, 17, ...)	[7.72910547164212, ...]	[0.30075676094310, ...] 4.0
' GERD is acting ...	0.0 [], gerd, is, act..., ...	[], ' , gerd, acting, ...	(262144, [62224, 95, ...)	(262144, [62224, 95, ...)	[9.02670468309538, ...]	[0.85322284832704, ...] 0.0
' Go to settings ...	4.0 [], go, to, setti..., ...	[], ' , go, settings, ...	(262144, [64465, 87, ...)	(262144, [64465, 87, ...)	[7.4473772490499, ...]	[0.19552947334757, ...] 4.0

Only showing top 20 rows

	text polarity	words1	words2	term_frequency	features	rawPrediction	probability prediction
!"AYYE PASS ME THE..."	4.0 [ayye, pass, me,...	[ 'ayye, pass, mon...	(262144, [31536, 64,...	{262144, [31536, 64,...	[7.54757712507432, ...	[0.29977054253765,...	4.0
"'Honey' the chick..."	0.0 [honey', the, ch...	[ 'honey', chicken...	(262144, [55627, 64,...	{262144, [55627, 64,...	[11.633477126232,...	[0.9990529148233,...	0.0
'What Canadians H...	4.0 [what, canadians...	[ 'what, canadians...	(262144, [16415, 10,...	{262144, [16415, 10,...	[6.92111820388093,	[0.0794177506000000...	4.0
'allo Davina weic...	4.0 [allo, davina, w...	[ 'allo, davina, w...	(262144, [1512, 124,...	{262144, [1512, 124,...	[7.32487646757031,...	[0.16216385641909,...	4.0
'course it's not ...	4.0 [course, it's, n...	[ 'course, quantit...	(262144, [43890, 70,...	{262144, [43890, 70,...	[5.4439718307222,...	[0.00445302843612,...	4.0
'bus just broke...	0.0 [s, bus, just, b...	[ 's, bus, broke...	(262144, [91694, 92,...	{262144, [91694, 92,...	[9.11167004686925,...	[0.87355932453613,...	0.0
's happy thought ...	0.0 [s, happy, thoug...	[ 's, happy, thoug...	(262144, [12409, 29,...	{262144, [12409, 29,...	[8.31312185212257,	[0.58091120318997,...	0.0
's screen size is...	0.0 [s, screen, size...	[ 's, screen, size...	(262144, [92492, 14,...	{262144, [92492, 14,...	[8.35212241137533,...	[0.60059380009412,...	0.0
's voice makes me...	4.0 [s, voice, makes...	[ 's, voice, makes...	(262144, [92492, 10,...	{262144, [92492, 10,...	[7.3993190449596,...	[0.18311487947805,...	4.0

# Spark MLlib NLP

## Building model

```
%time
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(labelCol="polarity", metricName="accuracy")
accuracy_val = evaluator.evaluate(val_df)
accuracy_test = evaluator.evaluate(test_df)
print("Validation Data:")
print(f"Accuracy: {accuracy_val*100:.5f}%")
print("Testing Data:")
print(f"Accuracy: {accuracy_test*100:.5f}%")
```

```
Validation Data:
Accuracy: 77.33275%
Testing Data:
Accuracy: 76.87971%
CPU times: user 14.1 ms, sys: 1.91 ms, total: 16 ms
Wall time: 19.7 s
```

# Spark MLlib NLP

## Building model

```
final_model = semantic_analysis_pipeline.fit(data)
final_model.save(MODEL_PATH)
```

# Run PySpark on Kaggle Notebook

- Click Code -> New Notebook

The screenshot shows the Kaggle competition interface for 'UIT Spring 2021 DS200.L11 Assignment 7'. The title 'UIT Spring 2021 DS200.L11 Assignment 7' is displayed prominently, along with the subtitle 'Predict the acceptability of cars'. Below the title, it says '18 teams · 2 days to go'. The navigation bar includes links for Overview, Data, Code, Discussion, Leaderboard, Rules, Team, and Host. The 'Code' link is highlighted with a red circle and underline. A red arrow points from the 'Code' link towards the 'New Notebook' button. The search bar at the bottom left contains the placeholder 'Search notebooks'. The bottom navigation bar includes links for All, Your Work, Shared With You, Bookmarks, and Hotness.

# Run PySpark on Kaggle Notebook

- Make sure Internet is turned On on Kaggle Notebook
- Run !pip install pyspark

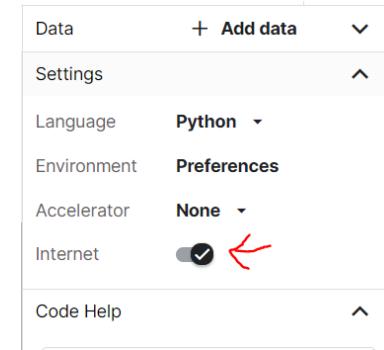
```
▶ !pip install pyspark
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("sparkOnKaggle").getOrCreate()

Collecting pyspark
  Downloading pyspark-3.1.2.tar.gz (212.4 MB)
    |██████████| 212.4 MB 57 kB/s eta 0:00:01   |██████████| 6.2 MB 10.2 MB/s eta 0:00:21
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
    |██████████| 198 kB 39.6 MB/s eta 0:00:01
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl size=212880768 sha256=0b1eec842dd65803d8c4a025067a20de79057b3a1fc32a1ab74a785eb56b25
  Stored in directory: /root/.cache/pip/wheels/a5/0a/c1/9561f6fecb759579a7d863dc846daaa95f598744e71b02c77
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.2

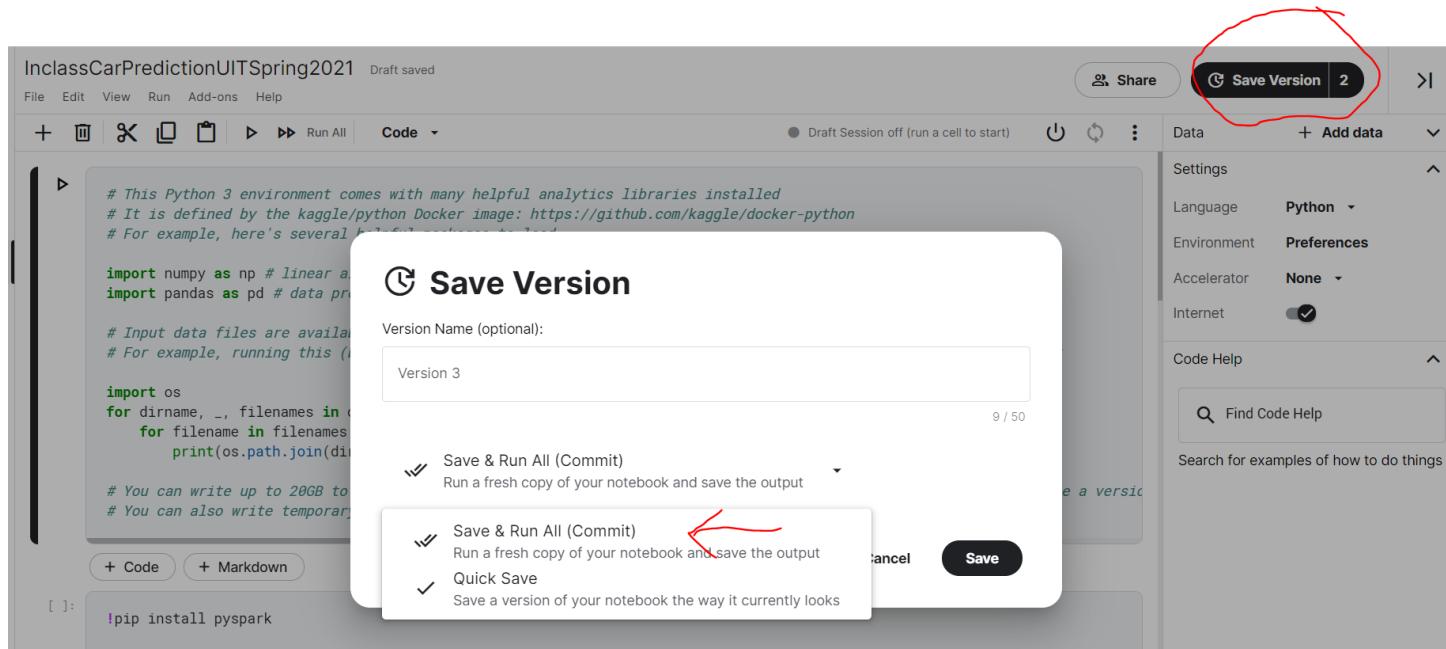
[8]: + Code + Markdown

[8]: spark.version
```



# Run PySpark on Kaggle Notebook

- After finishing -> click Save Version -> click Save & Run All (Commit)



# Run PySpark on Kaggle Notebook

- Scroll down your notebook or click on Output to see your output file -> click Submit

The screenshot shows a Kaggle Notebook interface. On the left, there's a sidebar with 'Output' selected, showing a file named 'mysolutions.csv' (2.24 KB). Below it is a link to download the file. The main area displays a table titled 'mysolutions.csv (2.24 KB)' containing 10 rows of data. A red arrow points from the 'Output' link in the sidebar to the 'Submit' button in the table preview. To the right, a vertical sidebar shows 'Version 2 of 2' with sections for 'Notebook', 'Input (1)', 'Output' (which is circled in red), 'Execution Info', 'Log', and 'Comments (0)'. The 'Output' section is currently active.

car_id	acceptability
11429	acc
11430	acc
11431	vgood
11432	acc
11433	vgood
11434	unacc
11435	unacc
11436	acc

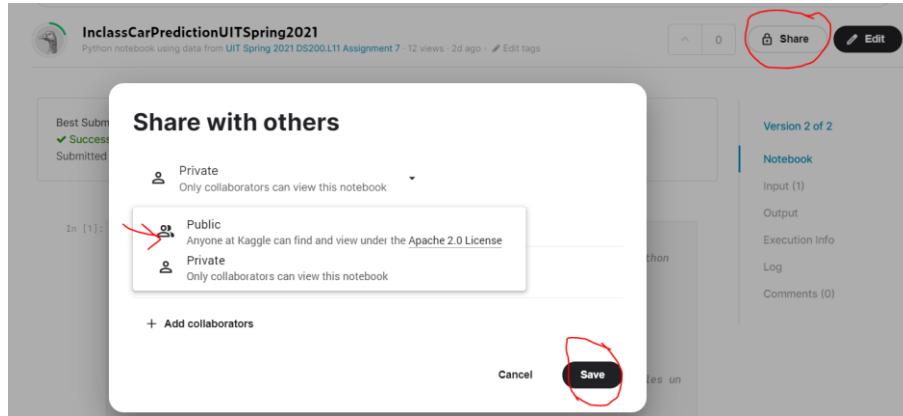
# Run PySpark on Kaggle Notebook

- Check if your submission is Successful and check your Score

A screenshot of a Kaggle notebook submission page. At the top, there is a search bar with a magnifying glass icon and the word "Search". Below the search bar, a user profile picture of a duck is shown next to the notebook title "InclassCarPredictionUITSpring2021". The title is bolded. Below the title, it says "Python notebook using data from [UIT Spring 2021 DS200.L11 Assignment 7](#) · 12 views · 2d ago · [Edit tags](#)". To the right of the notebook title, there is a button with an upward arrow and the number "0". In the bottom left corner of the page, there is a box containing the text "Best Submission" followed by "✓ Successful" and "Submitted 2 days ago". In the bottom right corner, there is a box containing the text "Public Score".

# Run PySpark on Kaggle Notebook

- After the competition finished, click Share -> Public -> Save

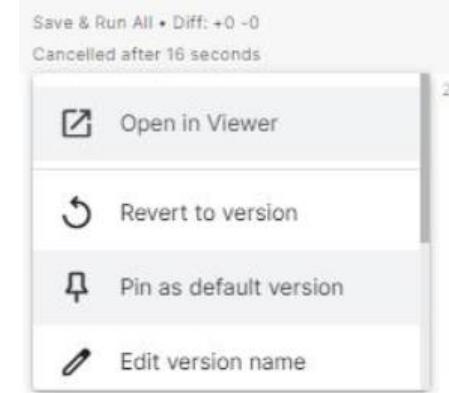
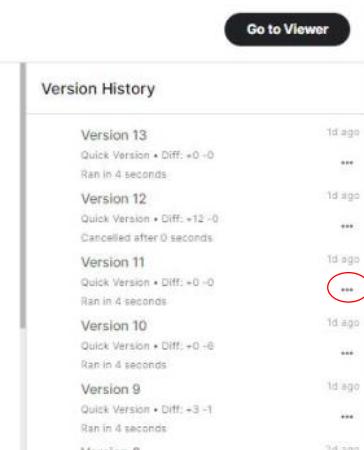


# Run PySpark on Kaggle Notebook

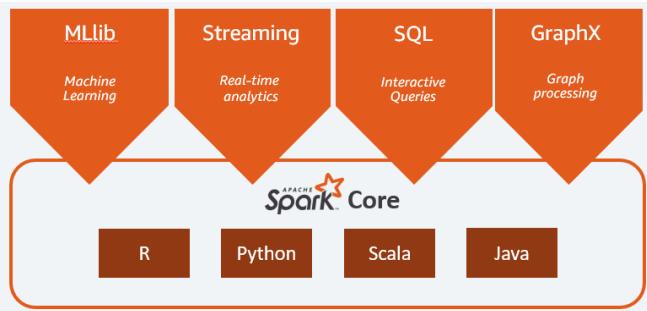
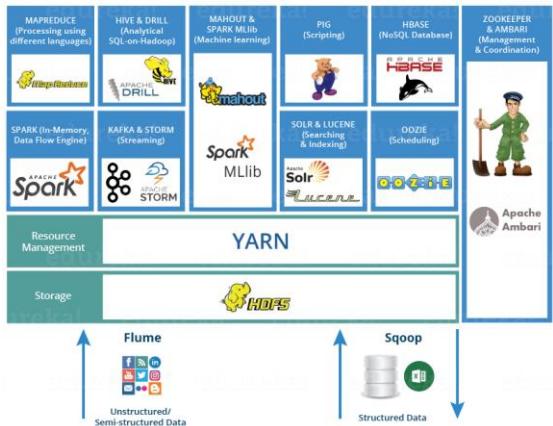
- **Pin default version**

Viewing Version 1: ✓ Save & Run All • June 5, 2021, 8:46 PM

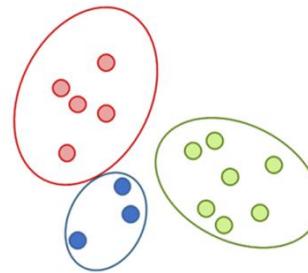
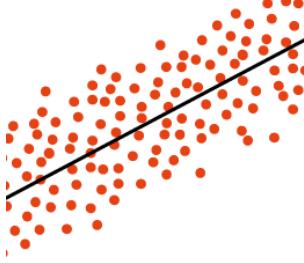
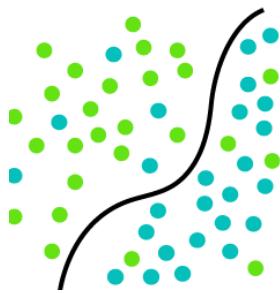
```
In [1]: # This Python 3 environment comes with many helpful analytics libraries installed  
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python  
# For example, here's several helpful packages to load  
  
import numpy as np # linear algebra  
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)  
  
# Input data files are available in the read-only "../input/" directory  
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under  
# the input directory  
  
import os  
for dirname, _, filenames in os.walk('/kaggle/input'):  
    for filename in filenames:  
        print(os.path.join(dirname, filename))
```



# Projects



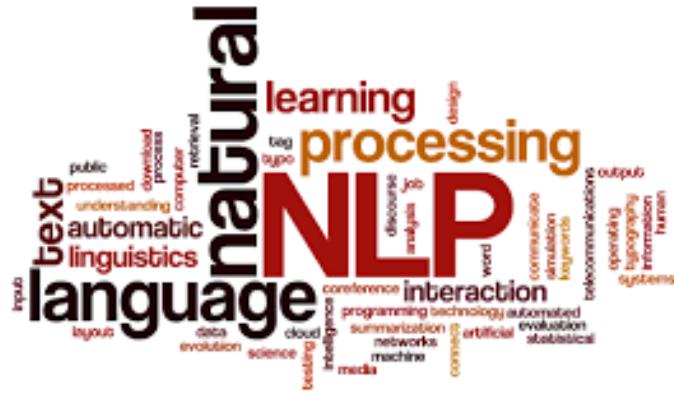
## Projects



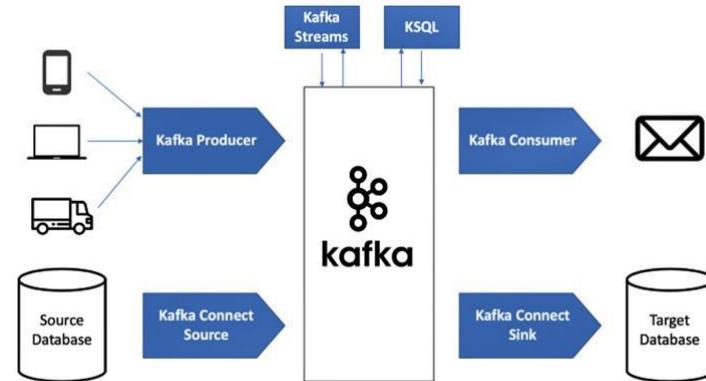
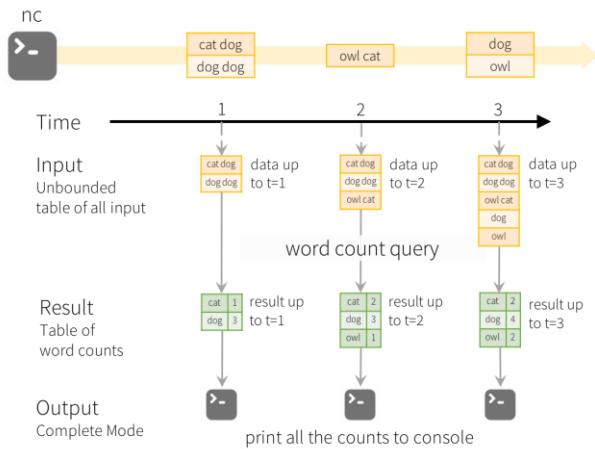
## Projects



## Projects



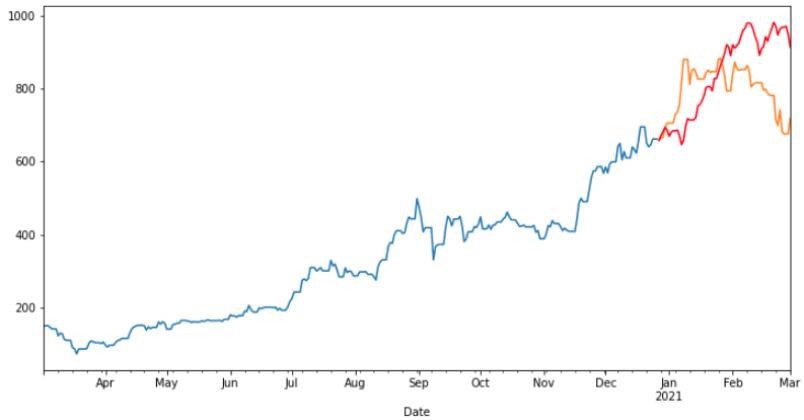
# Projects



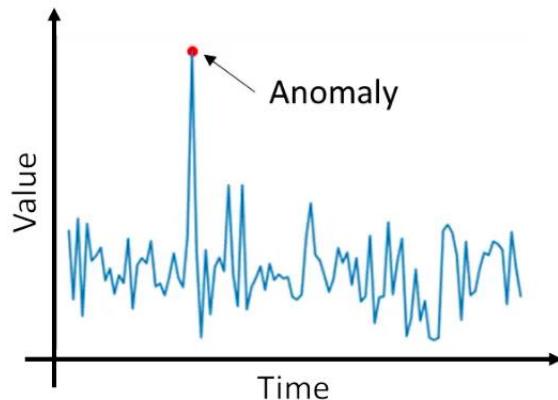
## Projects



# Projects



## Projects





# Q & A



## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.

# Big Data

## Spark for streaming data

Instructor: Trong-Hop Do

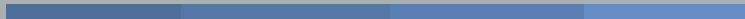
September 05<sup>th</sup> 2021



**“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”**

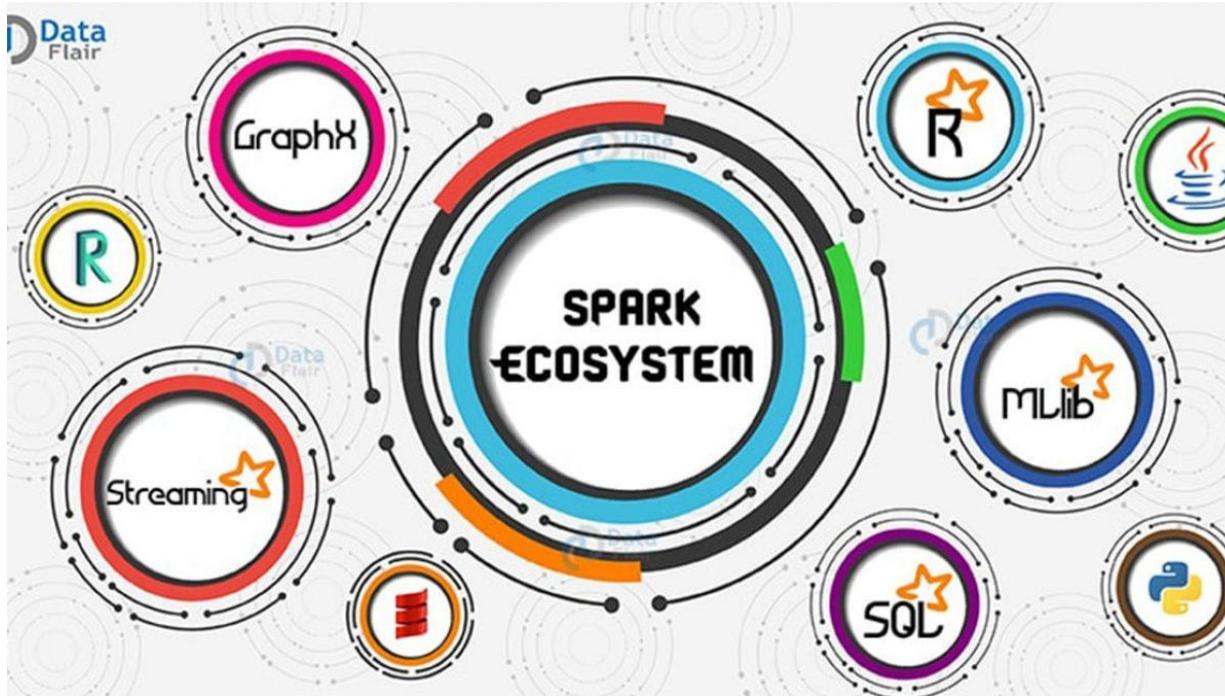
– Chris Lynch, Vertica Systems

# Spark Structure Streaming



# Streaming in Spark

Spark Streaming vs Structure Streaming



# Streaming in Spark

## Spark Streaming vs Structure Streaming

Spark



```
from pyspark.sql import streaming
```

SQL Structured Streaming

versus



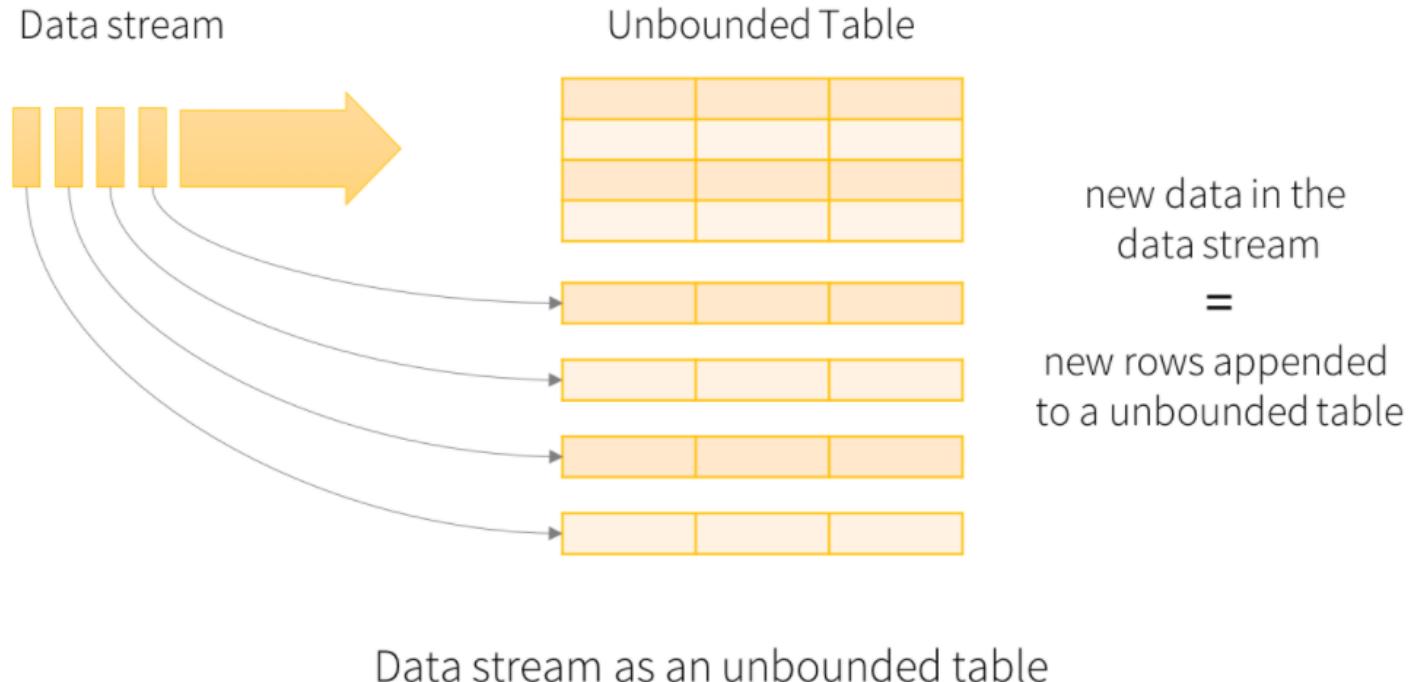
```
from pyspark import streaming
```

Streaming Component

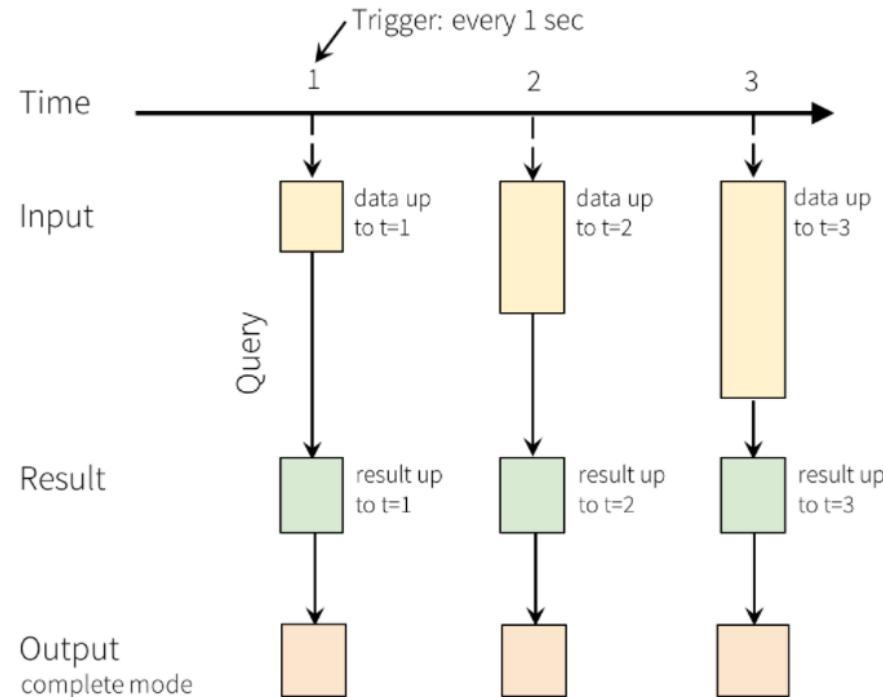
And remember,

Spark SQL Structured Streaming ≠ Spark Streaming (module)

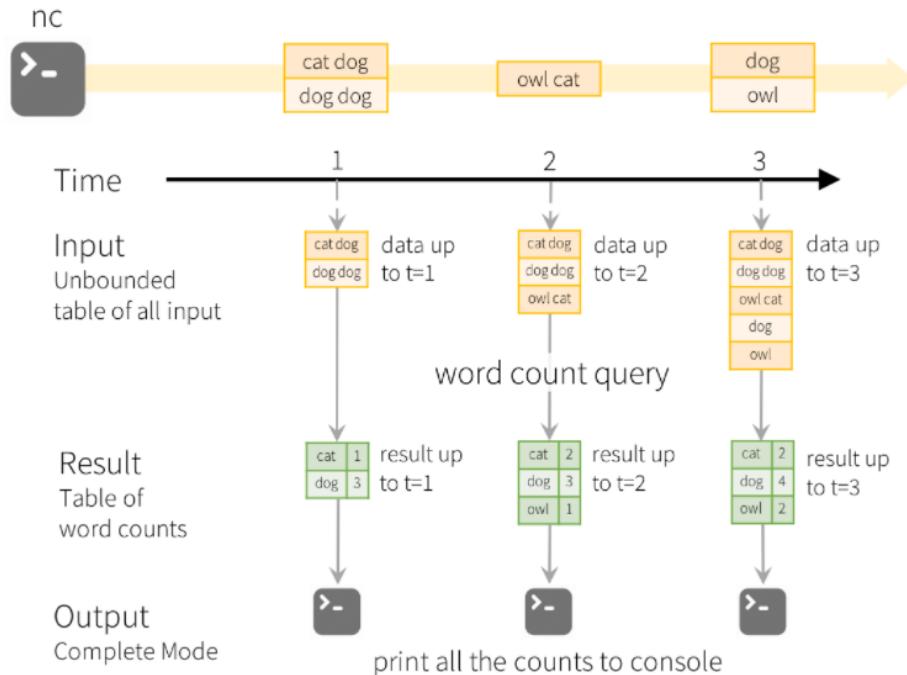
# Streaming in Spark



# Streaming in Spark



# Streaming in Spark



# Streaming in Spark

## Input Source

- File source
- Kafka source
- Socket source (for testing)
- Rate source (for testing)

# Streaming in Spark

## Output Sinks

- File sink
- Kafka sink
- Memory sink
- Console sink (for debugging)
- Foreach sink

# Streaming in Spark

<https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html>

# Structure streaming

## Quick example

```
import findspark
findspark.init()
import pyspark
from IPython.display import display, clear_output
from pyspark.sql import SparkSession
from pyspark.sql import functions as f
import pandas as pd
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode
from pyspark.sql.functions import split

spark = SparkSession \
    .builder \
    .appName("StructuredNetworkWordCount") \
    .getOrCreate()
```

# Structure streaming

## Quick example

```
# Create DataFrame representing the stream of input lines from connection to localhost:9999
lines = spark \
    .readStream \
    .format("socket") \
    .option("host", "localhost") \
    .option("port", 9999) \
    .load()

# Split the lines into words
words = lines.select(
    explode(
        split(lines.value, " ")
    ).alias("word")
)

# Generate running word count
wordCounts = words.groupBy("word").count()
```

# Structure streaming

## Quick example – Console sink

```
# Start running the query that prints the running counts to the console
query = wordCounts \
    .writeStream \
    .outputMode("complete") \
    .format("console") \
    .start()

query.awaitTermination()
```

# Run netcat and Spark application

Quick example – Console sink

```
D:\Spark\nc112>nc -l -p 9999  
cacat  
dog  
cat dog fish  
dog fish rat
```

```
D:\Spark\spark-3.0.1-bin-hadoop2.7>spark-submit code/quickexample.py  
Batch: 3  
-----  
+---+---+  
| word|count|  
+---+---+  
| rat| 1|  
| dog| 4|  
| cat| 1|  
| cacat| 1|  
| fish| 2|  
+---+---+
```

# Structure streaming

## Quick example – Memory sink

```
# Start running the query that write the running counts to memory
query = wordCounts \
    .writeStream \
    .queryName("wordCounts") \
    .outputMode("complete") \
    .format("memory") \
    .start()
```

# Structure streaming

## Quick example – Memory sink

Command Prompt - nc -l -p 9999

```
D:\Spark\nc111nt>nc -l -p 9999
dog cat fish
dog cat
```

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

word	count
dog	1
cat	1
fish	1

# Structure streaming

## Quick example – Memory sink

The screenshot shows a Jupyter Notebook interface with a command prompt and several code cells.

**Command Prompt:**

```
D:\Spark\nc111int>nc -l -p 9999
dog cat fish
dog cat
dog cat cat
cat cat fish
chicken
```

**Code Cell 1:**

```
[7]: display(spark.sql(f"SELECT * from {query.name}").show())
```

word	count
dog	1
cat	1
fish	1

**Code Cell 2:**

```
[*]: # show Live results for 2 minutes, refreshed every 1 second
from time import sleep
for x in range(0, 120):
    # spark.sql can be used to request how the query is performing
    display(spark.sql(f"SELECT * from {query.name}").toPandas())
    sleep(1)
    clear_output(wait=True)
else:
    print("Live view ended...")
```

word	count
dog	3
cat	6
chicken	1
fish	2

# Structure streaming

## Streaming from files

```
import pyspark
from IPython.display import display, clear_output
from pyspark.sql import SparkSession, DataFrame
from pyspark.sql import functions as f
import pandas as pd
from pyspark.ml import PipelineModel
from pyspark.sql.functions import udf
from pyspark.sql.streaming import DataStreamReader
import html

# SETTINGS
IN_PATH = "/kaggle/input/twitter-data-for-spark-streaming/"

timestampformat = "EEE MMM dd HH:mm:ss zzzz yyyy"
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
spark = SparkSession.builder.appName("StructuredStreamingExample").getOrCreate()
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
schema = spark.read.json(IN_PATH).limit(10).schema

spark_reader = spark.readStream.schema(schema)
```

# Structure streaming

## Streaming from files

```
streaming_data_raw = (
    spark_reader.json(IN_PATH)
    .select(
        "id",
        # extract proper timestamp from created_at column
        f.to_timestamp(f.col("created_at"), timestampformat).alias("timestamp"),
        # extract user information
        f.col("user.screen_name").alias("user"),
        "text",
    )
    .coalesce(1)
)
streaming_data_clean = clean_data(streaming_data_raw)

stream_writer = (streaming_data_clean.writeStream.queryName("data").trigger(once=True).outputMode("append").format("memory"))

query = stream_writer.start()
```

# Structure streaming

## Streaming from files

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

id	timestamp	user	text	original_text
1250972456673857538	2020-04-17 02:20:50	dibeckss	RT me after check...	RT @itsxdianaa: m...
1250972454035611649	2020-04-17 02:20:49	meanmediumode2	RT I love intra p...	RT @theroguecreat...
1250972456736526336	2020-04-17 02:20:50	mathoeee	RT won t say i m ...	RT @ArianaGrande:...
1250972455625125888	2020-04-17 02:20:50	Atalarania00	RT won t say i m ...	RT @ArianaGrande:...
1250972455834812416	2020-04-17 02:20:50	kaluvbot	RT y all really o...	RT @solsticemark:...
1250972455457312768	2020-04-17 02:20:49	_larri	RT HappY Birthday...	RT @btbsinluv777: ...
1250972453670547457	2020-04-17 02:20:49	twiliight_omen	Sorry but I won't...	Sorry, but I won'...
1250972453440028674	2020-04-17 02:20:49	HooverAthletics	RT Thank you for ...	RT @mj_thomas10: ...
1250972456061480960	2020-04-17 02:20:50	WaterBottle199	RT God Bless Amer...	RT @jakecoco: God...
1250972453582573572	2020-04-17 02:20:49	tj02008	RT MS ARIANA GRAN...	RT @francisdomini...
1250972453695676417	2020-04-17 02:20:49	nidzjordan	RT Today together...	RT @ADNFOREVER167...
1250972456493309958	2020-04-17 02:20:50	emmaaaa_blythe	RT Love my Instan...	RT @atdanwhite: L...
1250972456292155392	2020-04-17 02:20:50	cloutchaserprt	RT Imma go take a ...	RT @RandomPocket1...
1250972454777786368	2020-04-17 02:20:49	cakesiroe	RT True love's kiss	RT @Pillow_boi: T...
1250972454823948288	2020-04-17 02:20:49	kthnsbno	RT Full video cov...	RT @ArianaToday: ...
1250972454261977089	2020-04-17 02:20:49	sunshciner	RT You're smile m...	RT @_jenible: "Yo...
1250972455759462402	2020-04-17 02:20:50	sincerelyyoolie	I just love when ...	I just love when ...
1250972453398097920	2020-04-17 02:20:49	yjunlionness	RT Well does that...	RT @crackheadtxt_...
1250972456858329088	2020-04-17 02:20:50	KballeroandanT	RT Dear we love y...	RT @That_Radish_t...
1250972455008493568	2020-04-17 02:20:49	KhitLm1	RT No amount can ...	RT @MingErs_Intl:...

# Structure streaming

## Streaming from files

```
distinct_user_count = streaming_data_clean.select(f.approx_count_distinct("user"), f.current_timestamp())

stream_writer = (distinct_user_count.writeStream.queryName("data").trigger(once=True).outputMode("complete").format("memory"))

query = stream_writer.start()
```

+ Code

+ Markdown

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

```
+-----+-----+
|approx_count_distinct(user)| current_timestamp()|
+-----+-----+
|          49|2021-06-20 02:47:...|
+-----+-----+
```

# Structure streaming

## Streaming from files

```
sentiment_model = PipelineModel.load("/kaggle/input/pyspark-nlp/MODEL")
raw_sentiment = sentiment_model.transform(streaming_data_clean)

# Select downstream columns
sentiment = raw_sentiment.select(
    "id", "timestamp", "user", "text", f.col("prediction").alias("user_sentiment")
)
```

```
stream_writer = (sentiment.writeStream.queryName("data").trigger(once=True).outputMode("append").format("memory"))

query = stream_writer.start()
```

# Structure streaming

## Streaming from files

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

	id	timestamp	user	text	user_sentiment
1250972456673857538	2020-04-17 02:20:50	dibeckss	RT me after check...	4.0	
1250972454035611649	2020-04-17 02:20:49	meanmediummode2	RT I love intra p...	4.0	
1250972456736526336	2020-04-17 02:20:50	mathoeee	RT won t say i m ...	4.0	
1250972455625125888	2020-04-17 02:20:50	Atalaranaria00	RT won t say i m ...	4.0	
1250972455834812416	2020-04-17 02:20:50	kaluvbot	RT y all really o...	4.0	
1250972455457312768	2020-04-17 02:20:49	_larrri	RT HappY Birthday...	4.0	
1250972453670547457	2020-04-17 02:20:49	twiliight_omen	Sorry but I won't...	0.0	
1250972453440028674	2020-04-17 02:20:49	HooverAthletics	RT Thank you for ...	4.0	
1250972456061480960	2020-04-17 02:20:50	WaterBottle199	RT God Bless Amer...	4.0	
1250972453582573572	2020-04-17 02:20:49	tj02008	RT MS ARIANA GRAN...	4.0	
1250972453695676417	2020-04-17 02:20:49	nidzjordan	RT Today together...	4.0	
1250972456493309958	2020-04-17 02:20:50	emmaaaa_blythe	RT Love my Instan...	4.0	
1250972456292155392	2020-04-17 02:20:50	cloutchaserprt	RT Imma go take a...	0.0	
1250972454777786368	2020-04-17 02:20:49	cakesiroe	RT True love's kiss	4.0	
1250972454823948288	2020-04-17 02:20:49	kthnsbno	RT Full video cov...	4.0	
1250972454261977089	2020-04-17 02:20:49	sunshciner	RT You're smile m...	4.0	
1250972455759462402	2020-04-17 02:20:50	sincerelyyoolie	I just love when ...	4.0	
1250972453398097920	2020-04-17 02:20:49	yjunlionness	RT Well does that...	4.0	
1250972456858329088	2020-04-17 02:20:50	KballeroandanT	RT Dear we love y...	4.0	
1250972455008493568	2020-04-17 02:20:49	KhitLm1	RT No amount can ...	4.0	

# Structure streaming

## Streaming from files

```
negative_sentiment_count = (
    sentiment.filter("user_sentiment == 0.0")
    .select(f.col("user_sentiment").alias("negative_sentiment"))
    .agg(f.count("negative_sentiment"))
)

positive_sentiment_count = (
    sentiment.filter("user_sentiment == 4.0")
    .select(f.col("user_sentiment").alias("positive_sentiment"))
    .agg(f.count("positive_sentiment"))
)

average_sentiment = sentiment.agg(f.avg("user_sentiment"))
```

```
data_to_stream = average_sentiment
```

# Structure streaming

## Streaming from files

```
if isinstance(spark_reader, DataStreamReader):
    stream_writer = (
        data_to_stream.writeStream.queryName("streaming_table")
        .trigger(processingTime="20 seconds")
        #.trigger(once=True)
        .outputMode("complete")
        .format("memory")
    )
    # Calling .start on a DataStreamWriter return an instance of StreamingQuery
    query = stream_writer.start()
```

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

avg(user_sentiment)
+-----+   3.68   +-----+

# Structure streaming

## Streaming from files

```
# Let's see what we are outputting
if streaming_data_clean.isStreaming:
    from time import sleep
    for x in range(0, 200):
        try:
            if not query.isActive:
                break
            print("Showing live view refreshed every 10 seconds")
            print(f"Seconds passed: {x*10}")
            result = spark.sql(f"SELECT * from {query.name}")
            # spark.sql can be used to request how the query is performing
            display(result.toPandas())
            sleep(10)
            clear_output(wait=True)
        except KeyboardInterrupt:
            break
    print("Live view ended...")
else:
    print("Not streaming, showing static output instead")
    result = data_to_stream
    display(result.limit(10).toPandas())
```

# Create Twitter App

The screenshot shows a web browser window for the Twitter Developer website at developer.twitter.com/en/apps. The page has a purple header with navigation links like 'Developer', 'Use cases', 'Solutions', 'Products', 'Docs', 'Community', 'Updates', 'Support', 'Apply', and 'Apps'. A user profile icon is visible in the top right. Below the header, there's a main content area with a blue underline under the 'Apps' tab and a 'Create an app' button. The central text reads 'No apps here.' followed by a descriptive paragraph about API keys.

← → ⌂ 🔒 developer.twitter.com/en/apps

Developer Use cases Solutions Products Docs Community Updates Support Apply Apps

Apps Create an app

**No apps here.**

You'll need an app and API key in order to authenticate and integrate with most Twitter developer products. Create an app to get your API key.

# Create Twitter App



## #ApplicationReceived

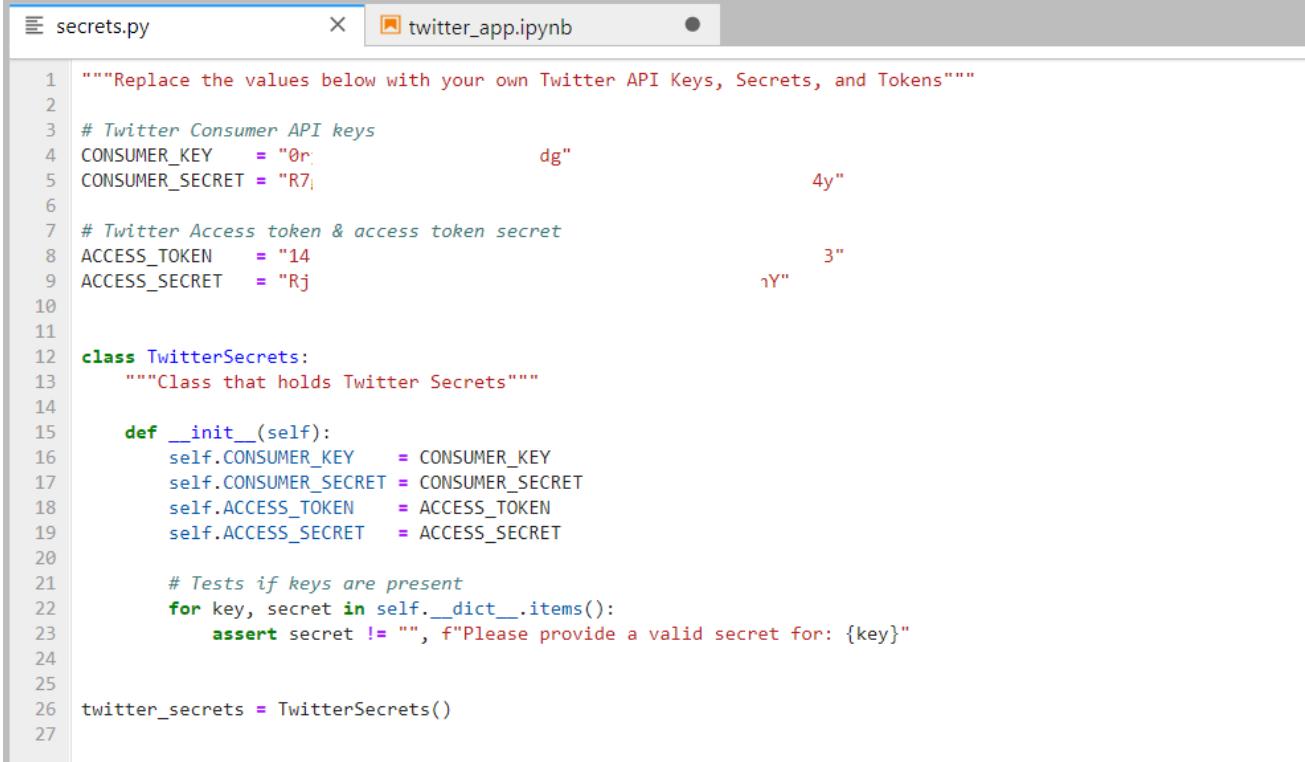
Your email has been verified and your application is officially under review!

We'll let you know when it's done, or if we need any additional information from you by sending an email

# Create Twitter App

The screenshot shows the Twitter Developer Portal interface. On the left, there's a sidebar with a dark background. At the top of the sidebar is the Twitter logo followed by "Developer Portal". Below this are three main items: "Dashboard", "Projects & Apps" (which is currently selected, indicated by an upward arrow icon), and "Overview". Under "Projects & Apps", there's a section for "Project 1" and a "TwitterNLUIT" project, which is highlighted with a blue vertical bar. Below "TwitterNLUIT", there are two collapsed sections: "Products" (with a "NEW" badge) and "Account". The main content area has a light gray background. At the top right of this area are two tabs: "Settings" and "Keys and tokens", with "Keys and tokens" being the active tab, indicated by a blue underline. Below the tabs, there are two sections: "Consumer Keys" and "Authentication Tokens". The "Consumer Keys" section contains a card for "API Key and Secret" with a "Regenerate" button. The "Authentication Tokens" section contains two cards: one for "Bearer Token" generated on June 20, 2021, with "Regenerate" and "Revoke" buttons; and another for "Access Token and Secret" generated on June 20, 2021, for the user "@dotronghop", also with "Regenerate" and "Revoke" buttons. A note below the second card states "Created with Read Only permissions".

# Create Twitter App



The screenshot shows a Jupyter Notebook interface with two tabs: 'secrets.py' and 'twitter\_app.ipynb'. The 'secrets.py' tab is active, displaying Python code for managing Twitter API keys and tokens.

```
1 """Replace the values below with your own Twitter API Keys, Secrets, and Tokens"""
2
3 # Twitter Consumer API keys
4 CONSUMER_KEY      = "0r"           dg"
5 CONSUMER_SECRET   = "R7;"          4y"
6
7 # Twitter Access token & access token secret
8 ACCESS_TOKEN      = "14"          3"
9 ACCESS_SECRET     = "Rj"          1Y"
10
11
12 class TwitterSecrets:
13     """Class that holds Twitter Secrets"""
14
15     def __init__(self):
16         self.CONSUMER_KEY      = CONSUMER_KEY
17         self.CONSUMER_SECRET   = CONSUMER_SECRET
18         self.ACCESS_TOKEN      = ACCESS_TOKEN
19         self.ACCESS_SECRET     = ACCESS_SECRET
20
21     # Tests if keys are present
22     for key, secret in self.__dict__.items():
23         assert secret != "", f"Please provide a valid secret for: {key}"
24
25
26 twitter_secrets = TwitterSecrets()
27
```

# Create Twitter App

```
from secrets import twitter_secrets as ts

OUT_PATH = "twitterdata"

QUERY = "euro 2021"

STOP_AFTER = 500
```

```
import json
import tempfile
import requests
import pathlib
from datetime import datetime as dt
from uuid import uuid4
from requests_oauthlib import OAuth1Session
```

# Create Twitter App

```
pathlib.Path(OUT_PATH).mkdir(parents = True, exist_ok = True)

query_data = {
    "track": f"#{QUERY}".replace("#", "").lower(), "language":"en",
}

twitter = OAuth1Session(
    client_key = ts.CONSUMER_KEY,
    client_secret = ts.CONSUMER_SECRET,
    resource_owner_key = ts.ACCESS_TOKEN,
    resource_owner_secret = ts.ACCESS_SECRET,
)

url = "https://stream.twitter.com/1.1/statuses/filter.json"
query_url = f"{url}?{'&'.join([f'{k}={v}' for k, v in query_data.items()])}"

print(f"STREAMING {STOP_AFTER} TWEETS")

with twitter.get(query_url, stream = True) as response:
    for i, raw_tweet in enumerate(response.iter_lines()):
        if i == STOP_AFTER:
            break
        try:
            tweet = json.loads(raw_tweet)
            print(f"{i+1}/{STOP_AFTER}: {tweet['user']['screen_name']} @ {tweet['created_at']}: {tweet['text']}\n")
        except (json.JSONDecodeError, KeyError) as err:
            print("ERROR")
            continue
    with pathlib.Path(OUT_PATH) / f"{dt.now().timestamp()}-{uuid4()}.json" as F:
        F.write_bytes(raw_tweet)
```

# Create Twitter App

The screenshot shows a Jupyter Notebook interface. On the left, there is a file browser window titled 'Streaming / twitterdata /' showing a list of files. The main area is a code editor with tabs for 'secrets.py' and 'twitter\_app.ipynb'. The code editor displays a stream of tweets from a Twitter API endpoint. The tweets are listed below:

```
STREAMING 500 TWEETS
1/500: press24newslive @ Sun Jun 20 16:22:11 +0000 2021: Italy v Wales: Euro 2020 - live! | Football https://t.co/UTGYAvFdKi

2/500: Anisa_News @ Sun Jun 20 16:22:30 +0000 2021: Updated Euro 2021 group scenarios: How each team can advance to the Round of 16 https://t.co/o0G2ckcb5p

ERROR
4/500: 9jaheadies @ Sun Jun 20 16:22:59 +0000 2021: #sportingnewsheadlines
    Euro 2021 tiebreakers: How groups are decided if teams are tied on points
    https://t.co/y2Pf8iuTOH

5/500: emerald_fm @ Sun Jun 20 16:23:17 +0000 2021: RT @ecb: (THREAD) A digital euro would be a powerful push for digitalisation, offering a safe, c
ostless means of payment and allowing inter...
    via @GoogleNews

ERROR
7/500: henson49_50 @ Sun Jun 20 16:23:47 +0000 2021: Fantasy Football Scout: The latest EURO 2020 injury updates and team news for Matchday 3.
    https://t.co/yEHfVVEis1

8/500: TelegraphSport @ Sun Jun 20 16:24:04 +0000 2021: #ITA 0 #WAL 0: A couple of nervy moments for Wales as Italy start strongly. Live updates wit
h @alantyers here.. https://t.co/1hb5barWaF

9/500: TeleFootball @ Sun Jun 20 16:24:04 +0000 2021: #ITA 0 #WAL 0: A couple of nervy moments for Wales as Italy start strongly. Live updates with
@alantyers here.. https://t.co/kmI8ly4T8l

10/500: henson40 @ Sun Jun 20 16:24:10 +0000 2021: Fantasy Football Scout: The latest EURO 2020 injury updates and team news for Matchday 3.
    https://t.co/go2d5RJCYe

    via @GoogleNews

11/500: BluMyst @ Sun Jun 20 16:24:14 +0000 2021: RT @newsmax: "[Biden] was all carrot and no stick," former Secretary of State @MikePompeo said, "T
hat won't work with Vladimir Putin, but I...
```

# Create Twitter App

The screenshot shows a Jupyter Notebook environment. On the left, there is a file browser window titled 'Streaming / twitterdata /' displaying a list of JSON files. The files are sorted by name and last modified time, with the most recent file at the top. The file names are truncated versions of tweet IDs. In the center, there is a code editor window titled 'secrets.py'. The code editor displays a Python dictionary structure representing a tweet object. The root node contains fields like 'created\_at', 'id', 'id\_str', 'text', 'source', 'truncated', and various reply-related fields. It also contains nested objects for 'user', 'retweeted\_status', and 'entities'. The entities field includes 'favorited', 'retweeted', 'filter\_level', 'lang', and 'timestamp\_ms'. The code editor has a toolbar at the top with icons for file operations and a status bar at the bottom.

```
root:
    created_at: "Sun Jun 20 16:24:48 +0000 2021"
    id: 1406649241255170049
    id_str: "1406649241255170049"
    text: "RT @LabyMod: Support your favorite team in the 2021 European football championship with the brand new flag cosmetic and new country texture."
    source: "<a href=\"http://twitter.com/download/android\" rel=\"nofollow">Twitter for Android</a>"
    truncated: false
    in_reply_to_status_id: null
    in_reply_to_status_id_str: null
    in_reply_to_user_id: null
    in_reply_to_user_id_str: null
    in_reply_to_screen_name: null
    user:
        geo: null
        coordinates: null
        place: null
        contributors: null
    retweeted_status:
        is_quote_status: false
        quote_count: 0
        reply_count: 0
        retweet_count: 0
        favorite_count: 0
    entities:
        favorited: false
        retweeted: false
        filter_level: "low"
        lang: "en"
        timestamp_ms: "1624206288051"
```

# Real-time sentiment analysis

The screenshot shows a Jupyter Notebook interface running on localhost:8888/lab. There are two code cells visible:

- Left Cell (File List):** A table showing a list of files in the directory / ... / Streaming / euro2021tweets /. The columns are Name and Last Modified. The files listed are all timestamped and end with .py or .pyc.
- Right Cell (Code):** Python code for real-time sentiment analysis. It imports secrets, json, tempfile, requests, and pathlib. It defines constants OUT\_PATH, QUERY, and STOP\_AFTER. It uses OAuthSession to interact with Twitter's stream API, specifying CONSUMER\_KEY, CONSUMER\_SECRET, ACCESS\_TOKEN, and ACCESS\_SECRET. It constructs URLs for the stream and query, and prints a message indicating the start of streaming.

```
from secrets import twitter_secrets as ts
OUT_PATH = "euro2021tweets"
QUERY = "euro 2021"
STOP_AFTER = 500
import json
import tempfile
import requests
import pathlib
from datetime import datetime as dt
from uuid import uuid4
from requests_oauthlib import OAuth1Session
pathlib.Path(OUT_PATH).mkdir(parents = True, exist_ok = True)
query_data = {
    "track": f"#{QUERY}".replace("#", "").lower(), "language": "en",
}
twitter = OAuth1Session(
    client_key = ts.CONSUMER_KEY,
    client_secret = ts.CONSUMER_SECRET,
    resource_owner_key = ts.ACCESS_TOKEN,
    resource_owner_secret = ts.ACCESS_SECRET,
)
url = "https://stream.twitter.com/1.1/statuses/filter.json"
query_url = f"{url}?{'&'.join([f'{k}={v}' for k, v in query_data.items()])}"
print(f"STREAMING {STOP_AFTER} TWEETS")
```

# Real-time sentiment analysis

The screenshot shows a web browser displaying a John Snow Labs page. The URL in the address bar is [nlp.johnsnowlabs.com/2021/01/18/analyze\\_sentimentdl\\_use\\_twitter\\_en.html](https://nlp.johnsnowlabs.com/2021/01/18/analyze_sentimentdl_use_twitter_en.html). The page title is "Sentiment Analysis of tweets Pipeline (analyze\_sentimentdl\_use\_twitter)". The top navigation bar includes links for Home, Docs, Learn, Models, Demo, and a gear icon for settings. Below the title, there are four small blue buttons labeled "en", "sentiment", "pipeline", and "open\_source". A "Description" section follows, containing a brief text about the pipeline and three orange buttons: "Live Demo", "Open in Colab", and "Download". An orange arrow points from the text "Check this" to the "Open in Colab" button.

Check this

and also: <https://nlp.johnsnowlabs.com/docs/en/install>

# Real-time sentiment analysis

```
stream_writer1 = (sentiment_result.writeStream.queryName("sentiment_result").trigger(processingTime="5 seconds").outputMode("append").format("memory"))

query1 = stream_writer1.start()

stream_writer2 = (sentiment_count_result.writeStream.queryName("data3").trigger(processingTime="5 seconds").outputMode("complete").format("memory"))

query2 = stream_writer2.start()

if streaming_data_clean.isStreaming:
    from time import sleep
    for x in range(0, 2000):
        try:
            if not query1.isActive:
                print("Query not active")
                break
            print("Showing live view refreshed every 5 seconds")
            print(f"Seconds passed: {x*5}")
            result1 = spark.sql(f"SELECT * from {query1.name}")
            result2 = spark.sql(f"SELECT * from {query2.name}")

            display(result1.toPandas())
            display(result2.toPandas())
            sleep(5)
            clear_output(wait=True)
        except KeyboardInterrupt:
            print("break")
            break
    print("Live view ended...")
else:
    print("Not streaming")
```

# Real-time sentiment analysis

Showing live view refreshed every 5 seconds  
Seconds passed: 605

	timestamp	user	document	sentiment	
0	2021-06-22 19:18:08	barbrady1	RT So the Government wouldn't relax rules to hold the Champions league final at Wembley and sent fans to Portugal wi	negative	
1	2021-06-22 18:45:44	fsams	RT UEFA says no surprising absolutely nobody But Munich should do it anyway pay the fine Make UEFA answer all the ques	negative	
2	2021-06-22 18:52:52	DuncMcKay	RT Because Public Health England a professional fucking body of experts deemed it so Grow up	negative	
3	2021-06-22 18:40:53	LuluBisserier		RT Shame	neutral
4	2021-06-22 19:09:39	F1Emz	RT UEFA is a joke Acknowledging that LGBTQ people exist and deserve equal treatment is not a political statement	negative	
...	...	...		...	
285	2021-06-22 19:30:22	cfcc Connor	RT Jack Grealish to START for England tonight Praise the Lord Team news story here ENG AVFC	positive	
286	2021-06-22 19:30:22	AkumiahJ	RT Jack Grealish to START for England tonight Praise the Lord Team news story here ENG AVFC	positive	
287	2021-06-22 19:30:23	InnoBystander		What could possibly go wrong	negative
288	2021-06-22 19:30:24	WesthamDeku_		Saucy	positive
289	2021-06-22 19:30:25	DistinctToday	Euro Denmark's incredible moment as players find out they've reached the last	positive	

290 rows × 4 columns

	sentiment	count
0	positive	173
1	neutral	15
2	negative	93

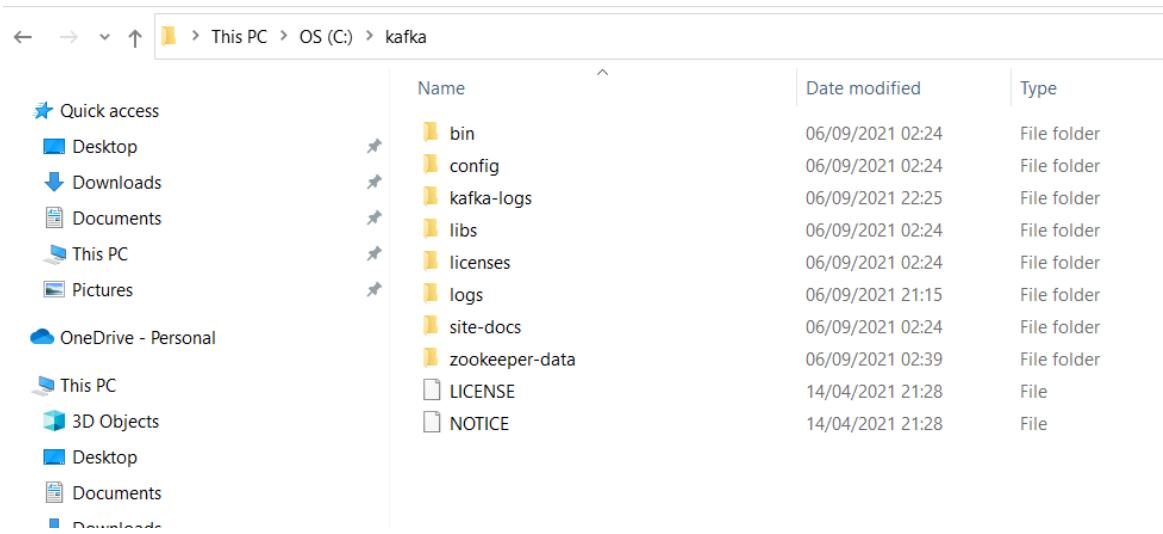
# Connect Spark Structured Streaming to Kafka

- Firstly, you need to install Kafka

<https://kafka.apache.org/>

# Kafka installation on Window

- Download Kafka from <https://kafka.apache.org/>
- Unzip the download file
- Rename the kafka to “kafka” and move it to C:\ drive



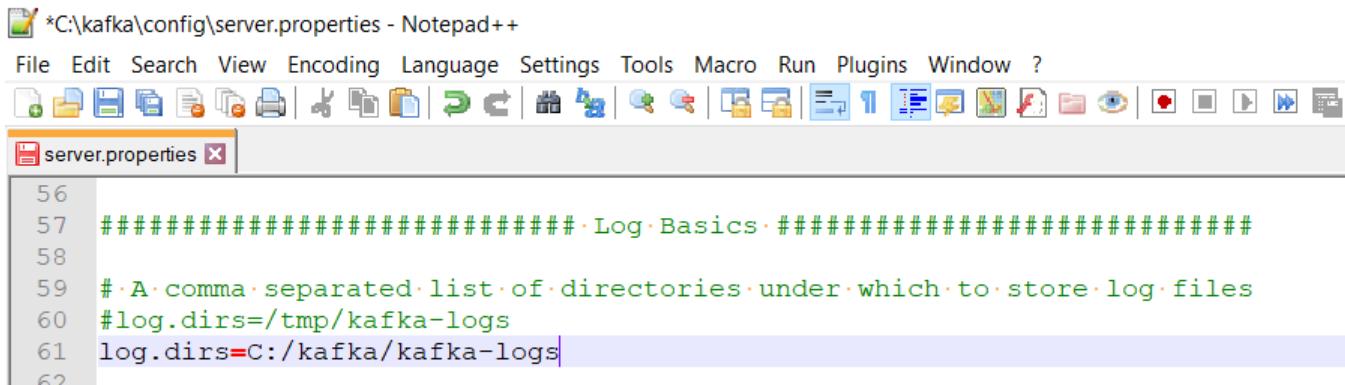
The screenshot shows a Windows File Explorer window with the following details:

Path: This PC > OS (C:) > kafka

Name	Date modified	Type
bin	06/09/2021 02:24	File folder
config	06/09/2021 02:24	File folder
kafka-logs	06/09/2021 22:25	File folder
libs	06/09/2021 02:24	File folder
licenses	06/09/2021 02:24	File folder
logs	06/09/2021 21:15	File folder
site-docs	06/09/2021 02:24	File folder
zookeeper-data	06/09/2021 02:39	File folder
LICENSE	14/04/2021 21:28	File
NOTICE	14/04/2021 21:28	File

# Kafka installation on Window

- Open C:\kafka\config\server.properties
- Change the path of log.dir

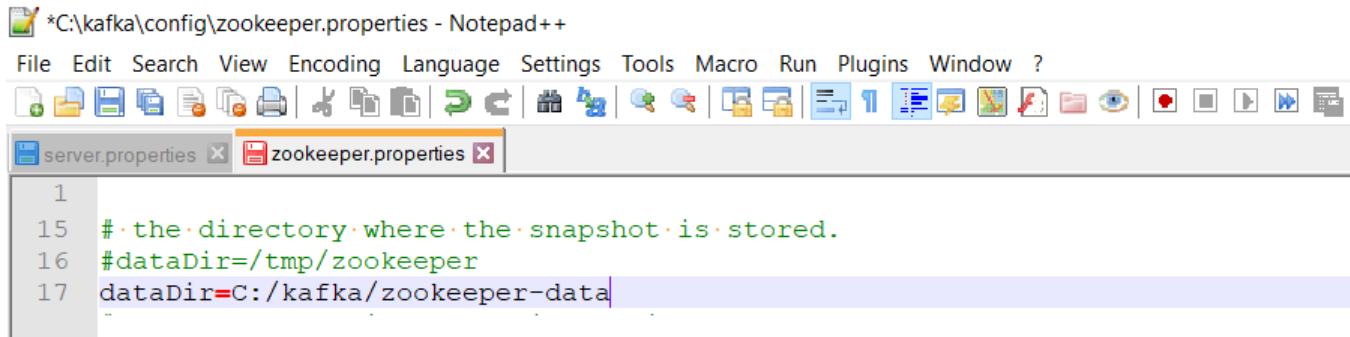


The screenshot shows the Notepad++ application window with the file 'server.properties' open. The window title is 'C:\kafka\config\server.properties - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar below the menu bar contains various icons for file operations like Open, Save, Print, and Find. The main editor area displays the configuration file with the following content:

```
56 ###### Log Basics #####
57 # A comma separated list of directories under which to store log files
58 #log.dirs=/tmp/kafka-logs
59 log.dirs=C:/kafka/kafka-logs
60
61
62
```

# Kafka installation on Window

- Open C:\kafka\config\zookeeper.properties
- Change the path of dataDir



\*C:\kafka\config\zookeeper.properties - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

server.properties x zookeeper.properties x

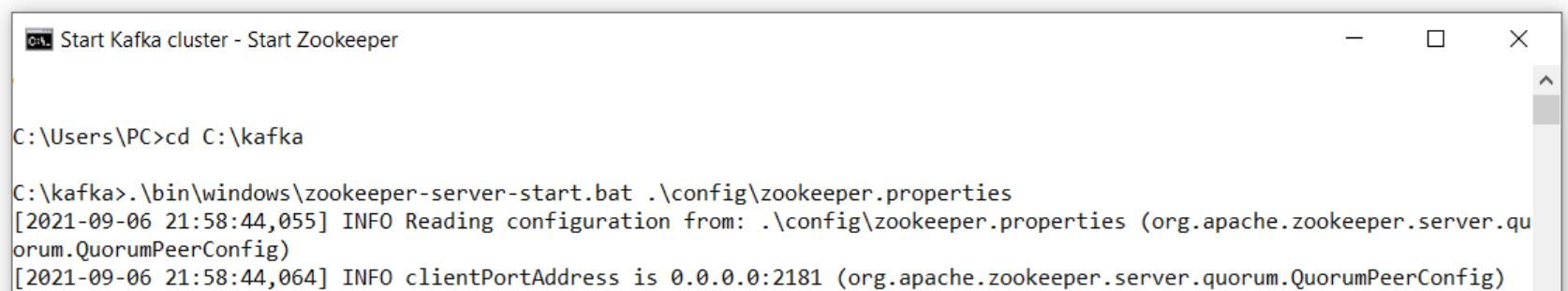
```
1
15 # the directory where the snapshot is stored.
16 #dataDir=/tmp/zookeeper
17 dataDir=C:/kafka/zookeeper-data
```

- By default Apache Kafka will run on port 9092 and Apache Zookeeper will run on port 2181.

# Test Apache Kafka on Windows

- Start the Kafka cluster
  - Run the following command to start ZooKeeper:

```
cd C:\kafka\  
.\\bin\\windows\\zookeeper-server-start.bat .\\config\\zookeeper.properties
```



```
C:\Users\PC>cd C:\kafka  
  
C:\kafka>.\\bin\\windows\\zookeeper-server-start.bat .\\config\\zookeeper.properties  
[2021-09-06 21:58:44,055] INFO Reading configuration from: .\\config\\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)  
[2021-09-06 21:58:44,064] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
```

# Test Apache Kafka on Windows

- Start the Kafka cluster
  - Run the following command to start the Kafka broker:  
`cd C:\kafka  
.\bin\windows\kafka-server-start.bat .\config\server.properties`



```
Start Kafka cluster - Start Kafka broker
C:\Users\PC>cd C:\kafka
C:\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties
[2021-09-06 22:03:25,045] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
```

# Test Apache Kafka on Windows

- Produce and consume some messages
  - Run the kafka-topics command to create a Kafka topic named TestTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic TestTopic
```

- Let's create another topic named NewTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic NewTopic
```

- Let's show list of created topics

```
.\bin\windows\kafka-topics.bat --list --zookeeper localhost:2181
```

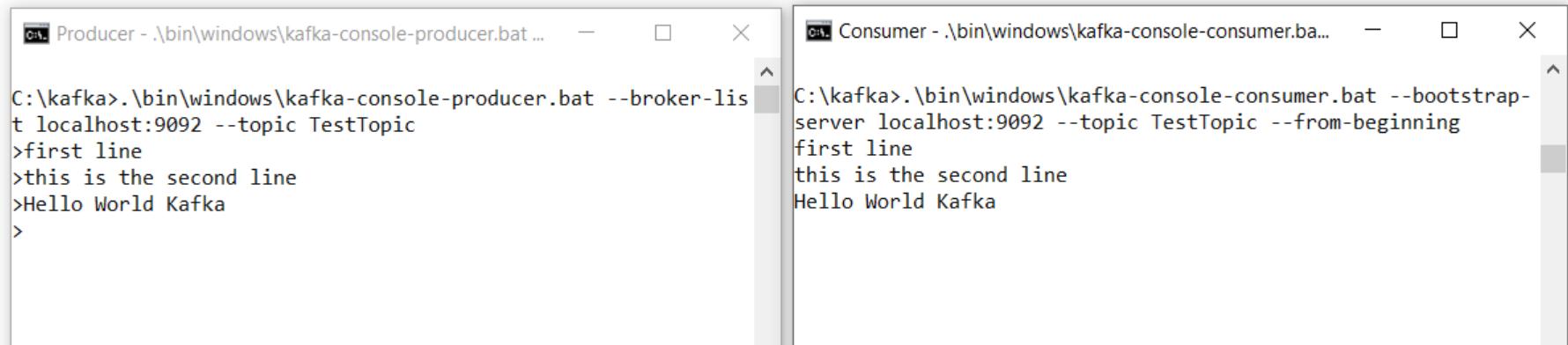
# Test Apache Kafka on Windows

- Produce and consume some messages

- Run the producer and consumer on separate Command Prompt:

```
.\bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic TestTopic
```

```
.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TestTopic --from-beginning
```



The image shows two separate Command Prompt windows running on Windows. The left window, titled 'Producer - .\bin\windows\kafka-console-producer.bat ...', displays the command being run and the message 'Hello World Kafka'. The right window, titled 'Consumer - .\bin\windows\kafka-console-consumer.bat ...', displays the command being run and the received message 'Hello World Kafka'.

```
Producer - .\bin\windows\kafka-console-producer.bat ...
C:\kafka>.\bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic TestTopic
>first line
>this is the second line
>Hello World Kafka
>
```

```
Consumer - .\bin\windows\kafka-console-consumer.bat ...
C:\kafka>.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TestTopic --from-beginning
first line
this is the second line
Hello World Kafka
```

# Connect Kafka and Spark Structure Streaming

- Step1: Start Kafka cluster using Terminal
- Step 2: Run KafkaProducer in Jupyter Notebook

```
from kafka import KafkaProducer
from json import dumps
from time import sleep

topic_name = 'RandomNumber'
kafka_server = 'localhost:9092'

producer = KafkaProducer(bootstrap_servers=kafka_server,value_serializer = lambda
x:dumps(x).encode('utf-8'))

for e in range(1000):
    data = {'number' : e}
    producer.send(topic_name, value=data)
    print(str(data) + " sent")
    sleep(5)

producer.flush()
```

- Open another Jupyter Notebook

```
[1]: import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession

scala_version = '2.12' # your scala version
spark_version = '3.0.1' # your spark version
packages = [
    f'org.apache.spark:spark-sql-kafka-0-10_{scala_version}:{spark_version}',
    'org.apache.kafka:kafka-clients:2.8.0' #your kafka version
]
spark = SparkSession.builder.master("local").appName("kafka-example").config("spark.jars.packages", ",".join(packages)).getOrCreate()
spark
```

[1]: **SparkSession - in-memory**

SparkContext

Spark UI

Version	v3.0.1
Master	local
AppName	kafka-example

- You will reading data from Kafka in two ways:
  - Batch query
  - Streaming query
- See more at <https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html>

# Creating a Kafka Source for Batch Queries

- Create dataframe from Kafka data

```
topic_name = 'RandomNumber'  
kafka_server = 'localhost:9092'  
kafkaDf = spark.read.format("kafka").option("kafka.bootstrap.servers", kafka_server).option("subscribe", topic_name).option("startingOffsets", "earliest").load()
```

- Show data (converting dataframe to pandas for cleaner view of data)

```
kafkaDf.toPandas()
```

	key	value	topic	partition	offset	timestamp	timestampType
0	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	0	2022-10-05 15:18:37.301	0
1	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	1	2022-10-05 15:18:42.314	0
2	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	2	2022-10-05 15:18:47.327	0
3	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	3	2022-10-05 15:18:52.341	0
4	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	4	2022-10-05 15:18:57.352	0
5	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	5	2022-10-05 15:19:02.363	0
6	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	6	2022-10-05 15:19:07.376	0
7	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	7	2022-10-05 15:19:12.395	0
8	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	8	2022-10-05 15:19:17.411	0
9	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	9	2022-10-05 15:19:22.417	0

- Show streaming data using for loop

```
batchDF = kafkaDf.select(col('topic'), col('offset'), col('value').cast('string').substr(12,1).alias('rand_number'))  
from time import sleep  
from IPython.display import display, clear_output  
for x in range(0, 2000):  
    try:  
        print("Showing live view refreshed every 5 seconds")  
        print(f"Seconds passed: {x*5}")  
        display(batchDF.toPandas())  
        sleep(5)  
        clear_output(wait=True)  
    except KeyboardInterrupt:  
        print("break")  
        break  
print("Live view ended...")
```

Showing live view refreshed every 5 seconds  
Seconds passed: 10

	topic	offset	rand_number
0	RandomNumber	0	2
1	RandomNumber	1	4
2	RandomNumber	2	7
3	RandomNumber	3	7
4	RandomNumber	4	3
5	RandomNumber	5	7
6	RandomNumber	6	6

- Perform some data aggregation and show live results

```
batchCountDF = batchDF.groupBy('rand_number').count()
for x in range(0, 2000):
    try:
        print("Showing live view refreshed every 5 seconds")
        print(f"Seconds passed: {x*5}")
        display(batchCountDF.toPandas())
        sleep(5)
        clear_output(wait=True)
    except KeyboardInterrupt:
        print("break")
        break
print("Live view ended...")
```

Showing live view refreshed every 5 seconds

Seconds passed: 5

	rand_number	count
0	7	5
1	3	2
2	8	1
3	0	1
4	5	1
5	6	3
6	9	1
7	1	1
8	4	1
9	2	1

break

Live view ended...

# Creating a Kafka Source for Streaming Queries

- Create Streaming dataframe from Kafka

```
streamRawDf = spark.readStream.format("kafka").option("kafka.bootstrap.servers",  
kafka_server).option("subscribe", topic_name).load()  
streamDF =  
streamRawDf.select(col('topic'),col('offset'),col('value').cast('string').substr(12,1).alias('rand_number'))  
checkEvenDF = streamDF.withColumn('Is_Even',col('rand_number').cast('int') % 2 == 0 )
```

- Write stream

```
from random import randint  
randNum=str(randint(0,10000))  
q1name = "queryNumber"+randNum  
q2name = "queryCheckEven"+randNum  
  
stream_writer1 = (streamDF.writeStream.queryName(q1name).trigger(processingTime="5  
seconds").outputMode("append").format("memory"))  
stream_writer2 = (checkEvenDF.writeStream.queryName(q2name).trigger(processingTime="5  
seconds").outputMode("append").format("memory"))  
  
query1 = stream_writer1.start()  
query2 = stream_writer2.start()
```

## ● View streaming result

```
for x in range(0, 2000):
    try:
        print("Showing live view refreshed every 5 seconds")
        print(f"Seconds passed: {x*5}")
        result1 = spark.sql(f"SELECT * from {query1.name}")
        result2 = spark.sql(f"SELECT * from {query2.name}")
        display(result1.toPandas())
        display(result2.toPandas())
        sleep(5)
        clear_output(wait=True)
    except KeyboardInterrupt:
        print("break")
        break
print("Live view ended...")
```

Showing live view refreshed every 5 seconds  
Seconds passed: 20

	topic	offset	rand_number
0	RandomNumber	21	0
1	RandomNumber	22	8
2	RandomNumber	23	2
3	RandomNumber	24	1

	topic	offset	rand_number	Is_Even
0	RandomNumber	21	0	True
1	RandomNumber	22	8	True
2	RandomNumber	23	2	True
3	RandomNumber	24	1	False

break  
Live view ended...



# Q & A



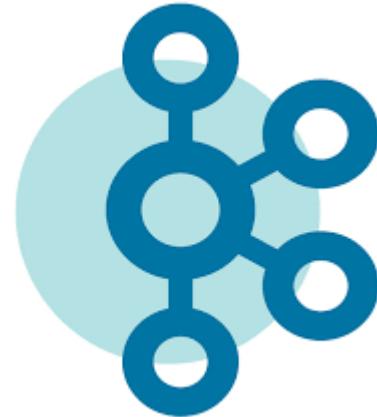
## Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.

# Distributed and Parallel Computing

Trong-Hop Do

# Kafka – A distributed event streaming platform



# What is event streaming?

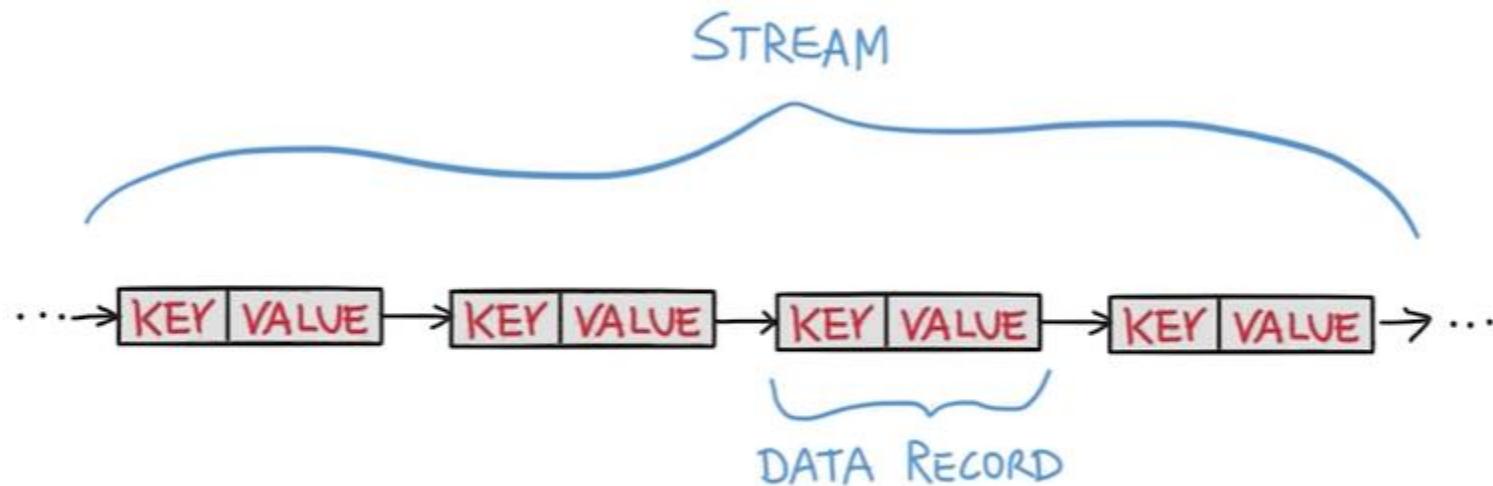


# What can I use event streaming for?

- To process payments and financial transactions in real-time, such as in stock exchanges, banks, and insurances.
- To track and monitor cars, trucks, fleets, and shipments in real-time, such as in logistics and the automotive industry.
- To continuously capture and analyze sensor data from IoT devices or other equipment, such as in factories and wind parks.
- To collect and immediately react to customer interactions and orders, such as in retail, the hotel and travel industry, and mobile applications.
- To monitor patients in hospital care and predict changes in condition to ensure timely treatment in emergencies.
- To connect, store, and make available data produced by different divisions of a company.
- To serve as the foundation for data platforms, event-driven architectures, and microservices.

# What is a stream?

- Think of a stream as an unbounded, continuous real-time flow of records
  - You don't need to explicitly request new records, you just receive them
- Records are key-value pairs



## Motivation

The Shift to Event-driven Systems has Already Begun...

**From a static snapshot...**



Occasional call to a friend

**...to a continuous stream of events**



A constant feed about the activities of all your friends



Daily news reports



Real time news feeds, accessible online anytime, anywhere

## Motivation

This leads us to...



**Single platform** to connect everyone to every event



**Real-time** stream of events



**All events stored** for historical view

# Motivation

Successful Digital Businesses are Inherently Event-driven

Born cloud-native...



Social Networks  
Enabling Event  
Sharing

Traditional ones that adapt...



Newspaper  
Provide a single  
Source of Truth

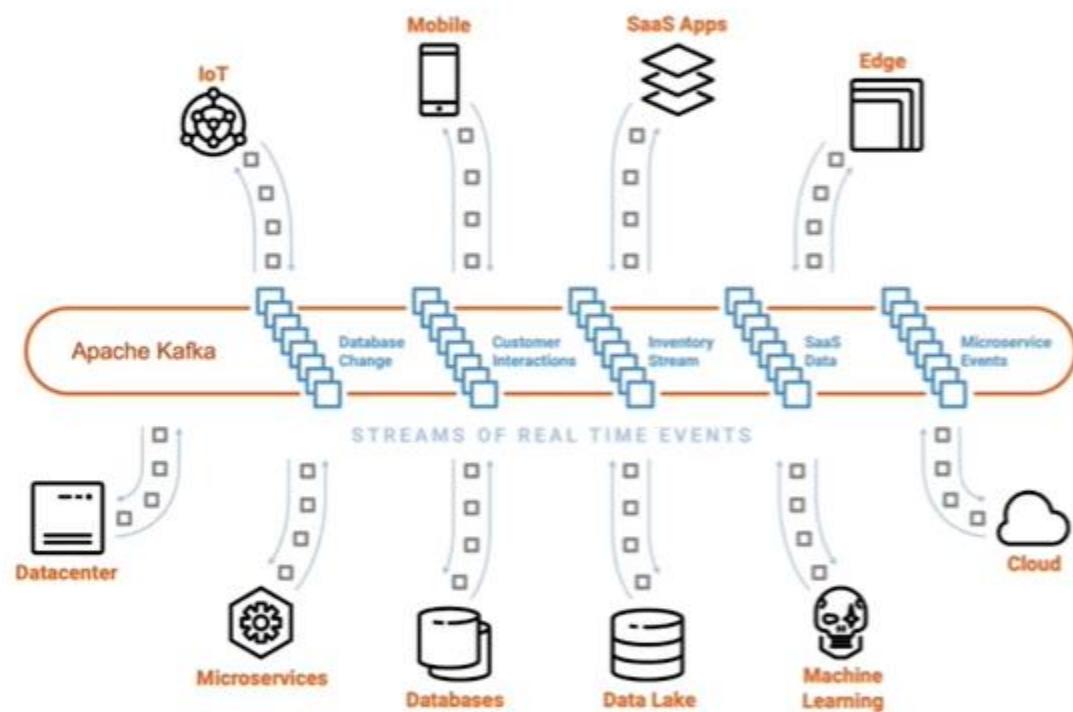


Streaming Provider  
On-demand Digital  
Content



Credit Card Payments  
Microservices  
Architecture

# Motivation



**Apache Kafka®: the De-facto Standard for Real-Time Event Streaming**

- Global-scale
- Real-time
- Persistent Storage
- Stream Processing

## Motivation

Thousands of Companies Worldwide trust Kafka for their Journey towards "Event-driven"





Travel



6 of top 10



Global banks



7 of top 10



Insurance



8 of top 10



Telecom



9 of top 10

## Real-time Fraud Detection

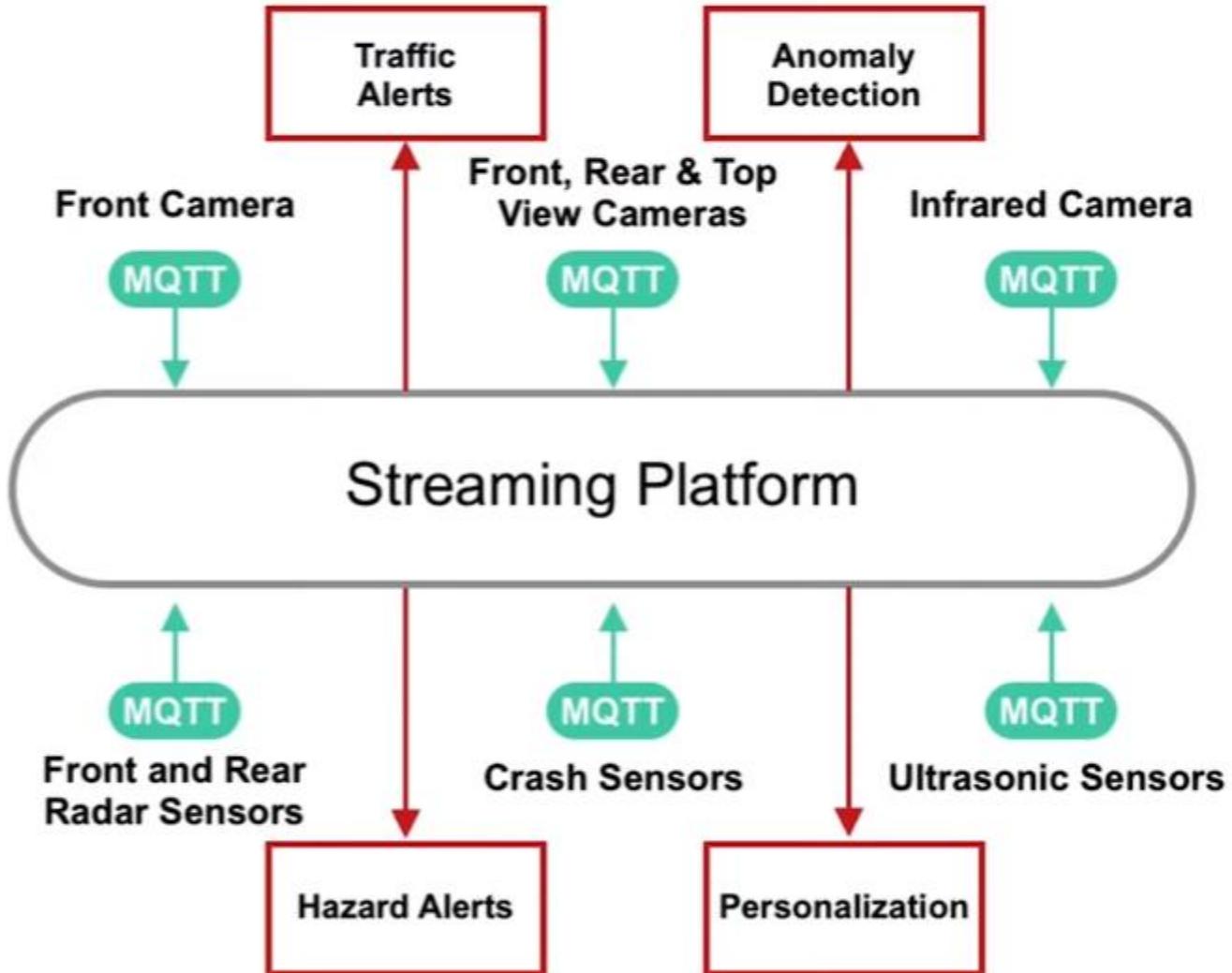


- Act in real-time
- Detect fraud
- Minimize risk
- Improve customer experience

# Automotive



The Future of the Automotive Industry is a Real Time Data Cluster



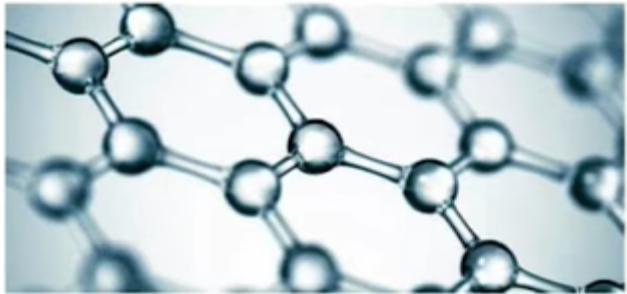
## Real-time e-Commerce



### Rewards Program

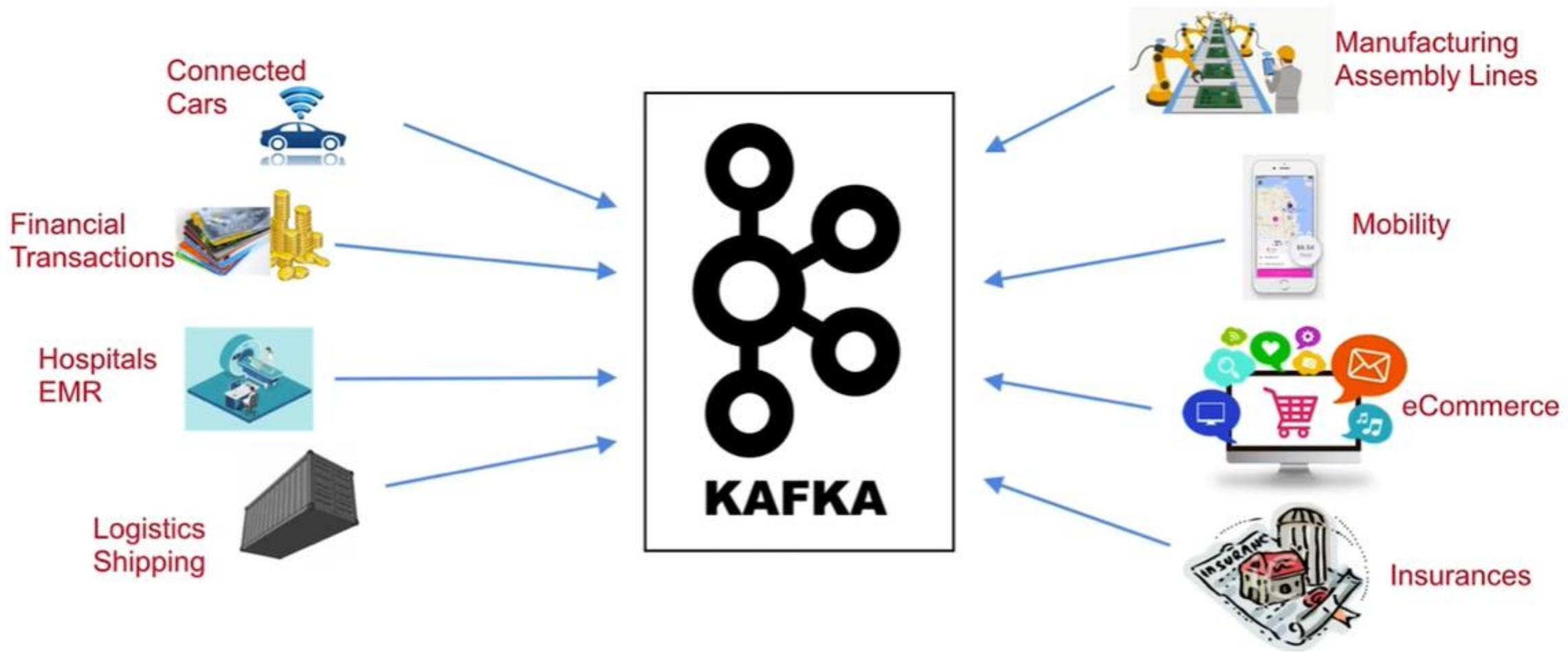
- Onboarding new merchants faster
- Increased speed at which mobile applications are delivered to customers
- Enabled a full 360 view of customers
- Enhanced performance and monitoring
- Projected savings of millions of dollars

## Health Care

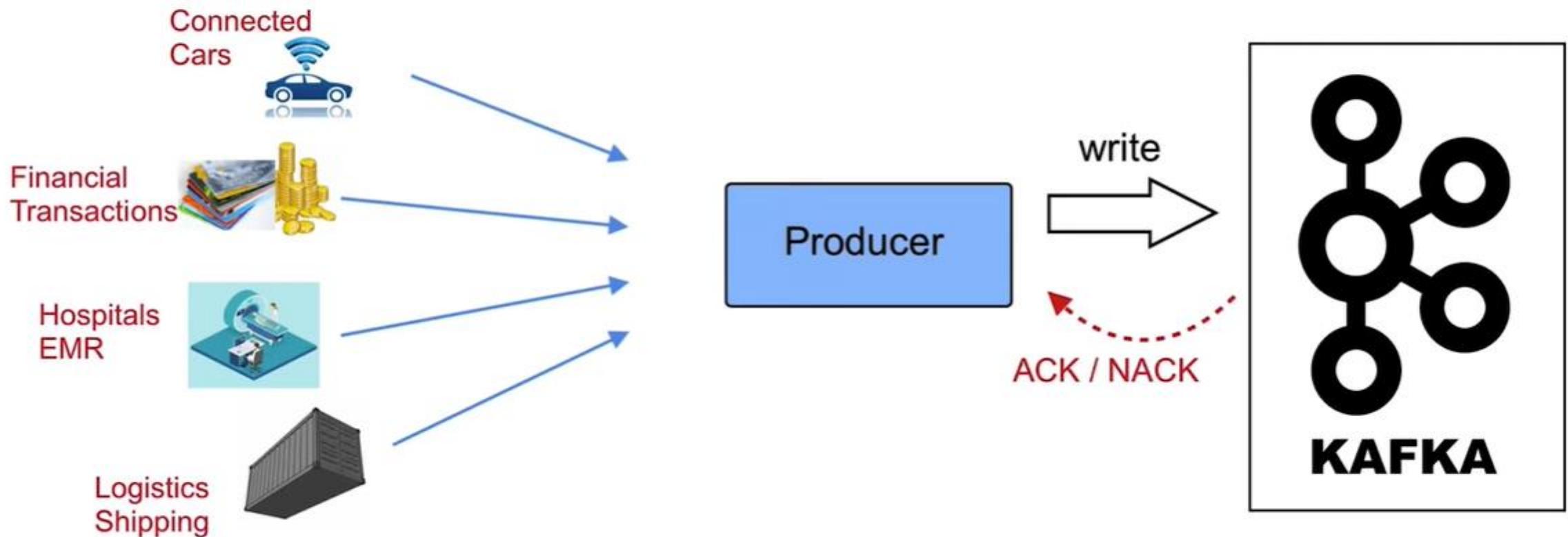


- Microservices
- Internet of Things

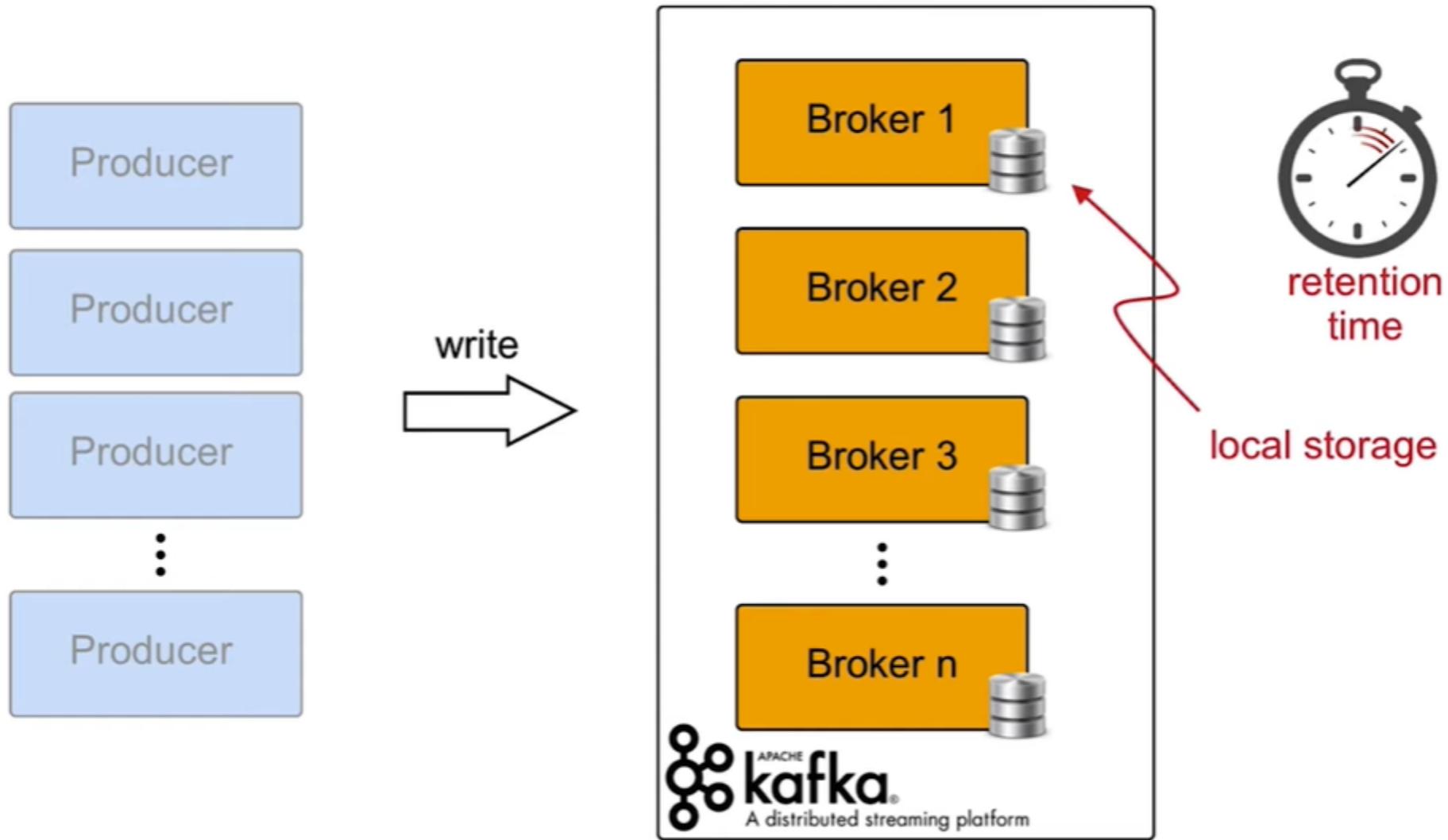
# The World Produces Data



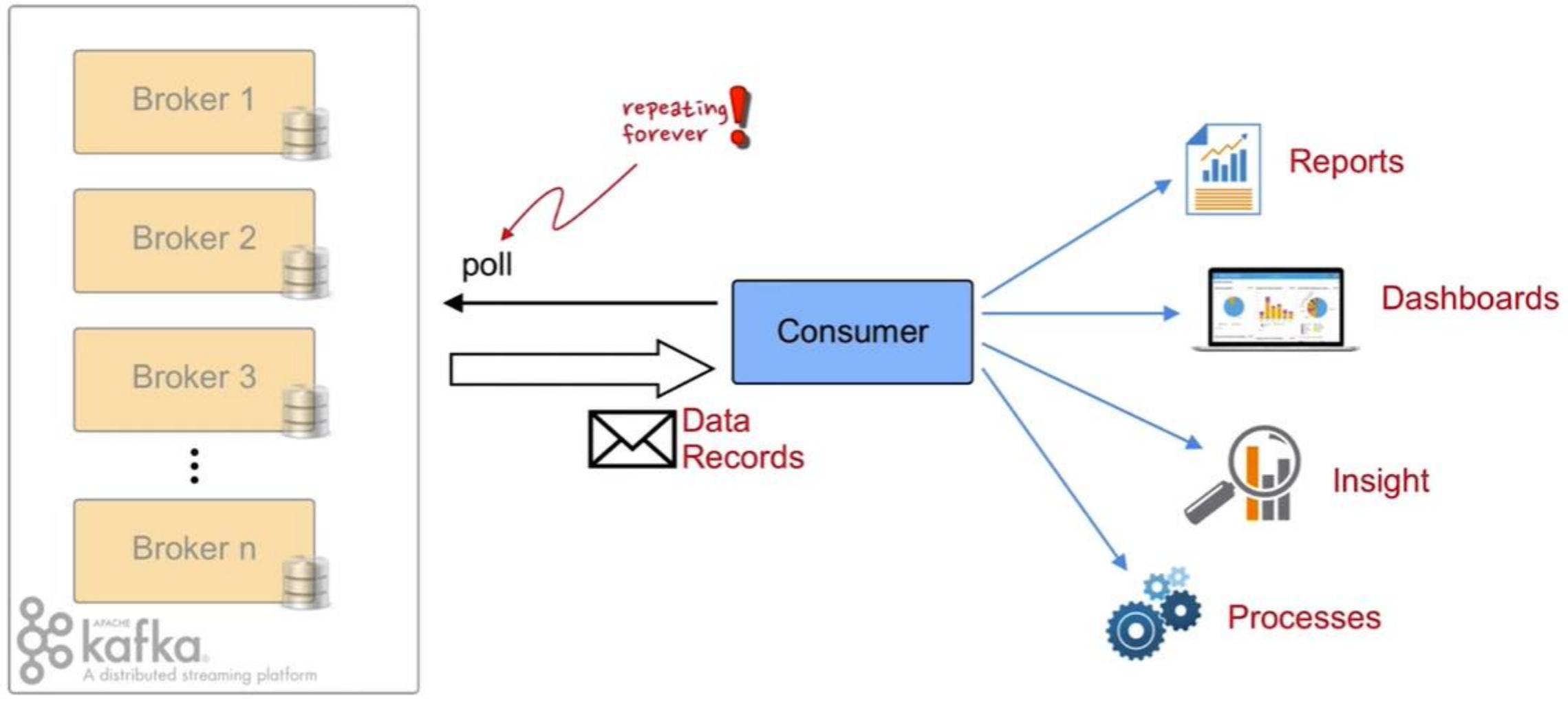
# Producers



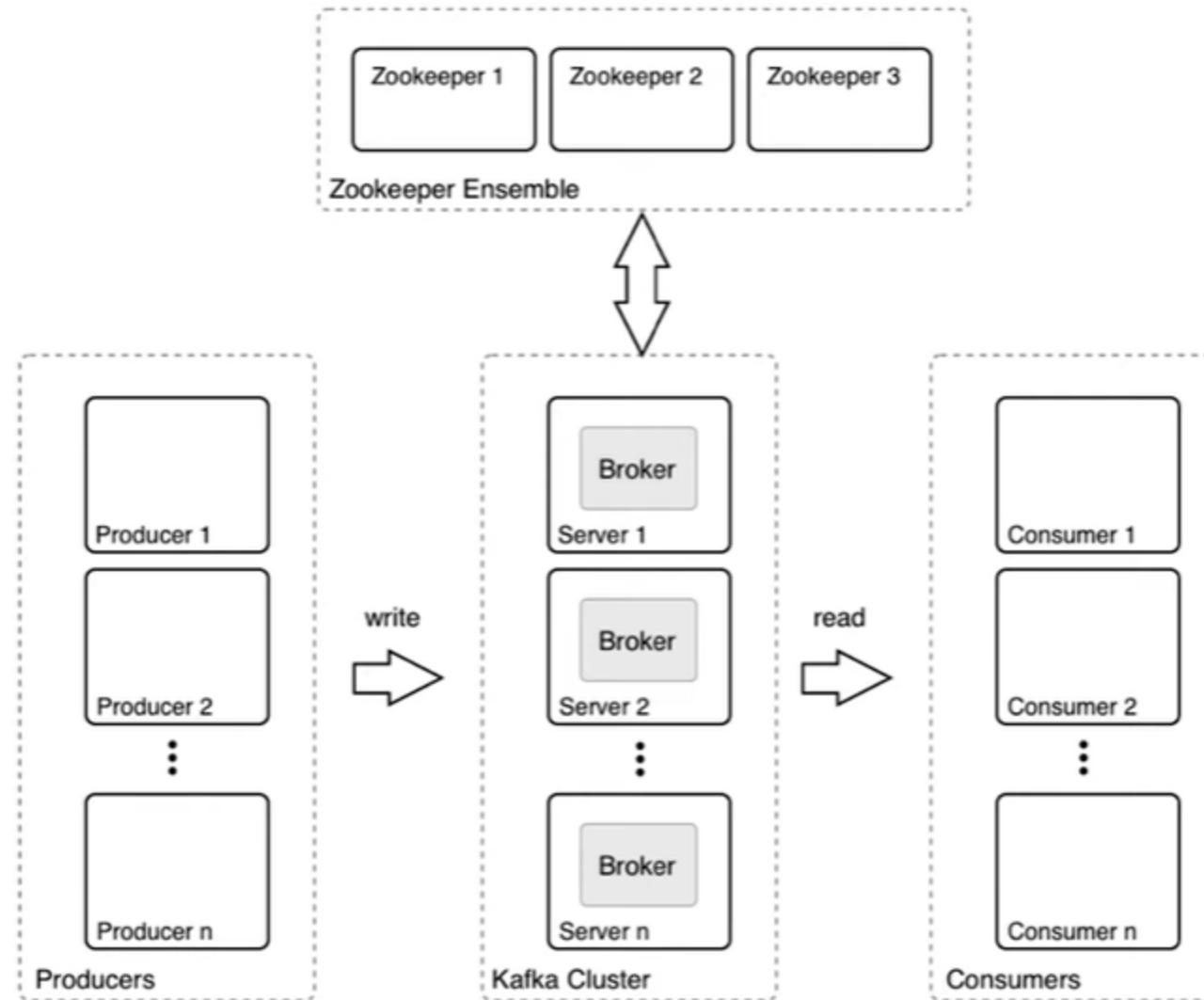
## Kafka Brokers



## Consumers

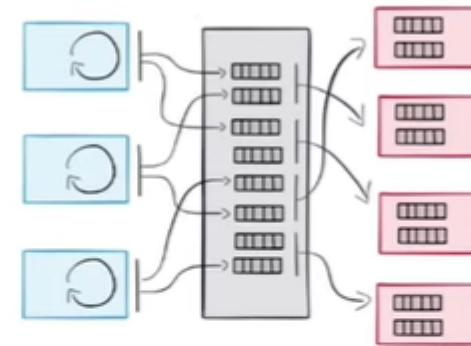


# Architecture

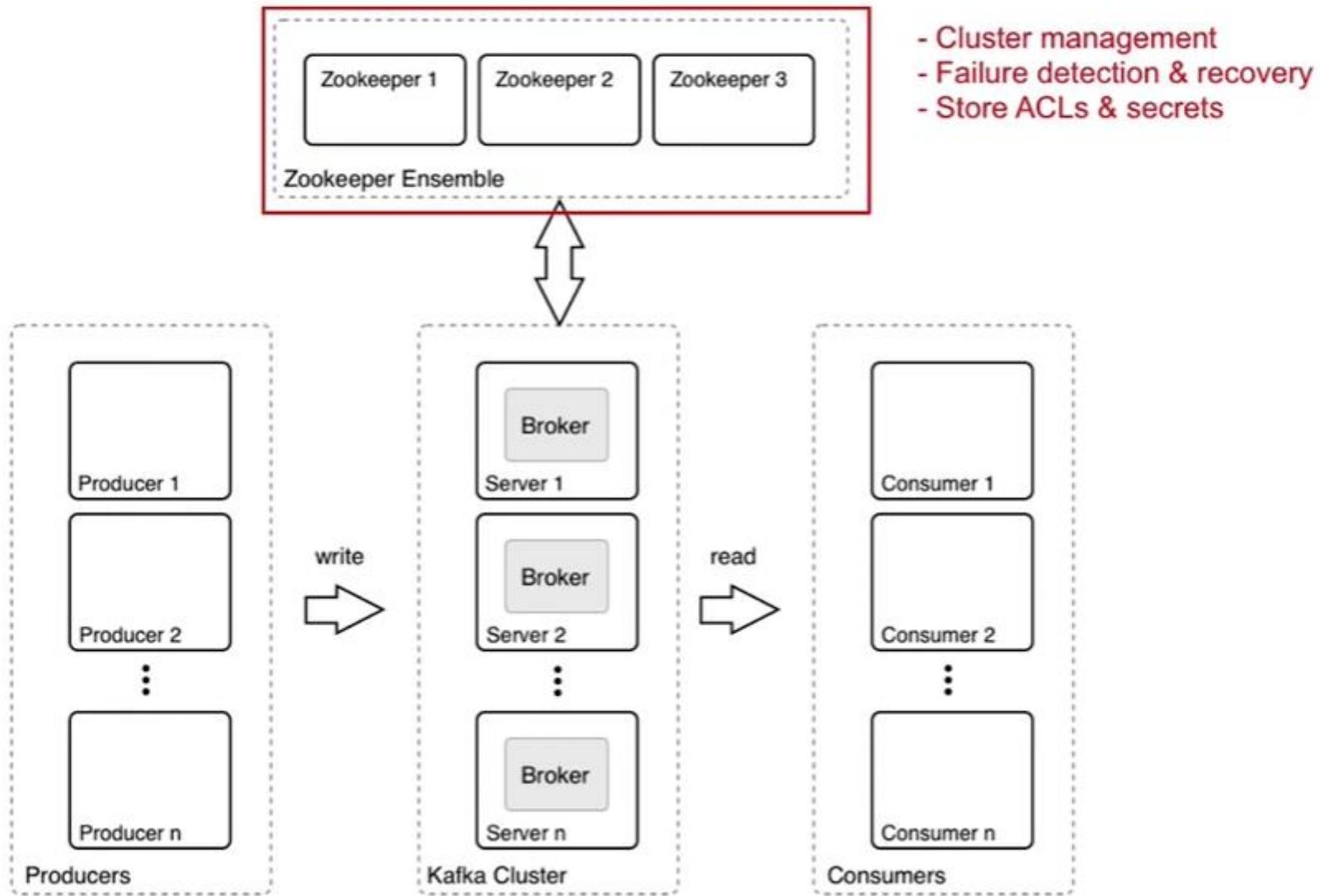


## Decoupling Producers and Consumers

- Producers and Consumers are decoupled
- Slow Consumers do not affect Producers
- Add Consumers without affecting Producers
- Failure of Consumer does not affect System

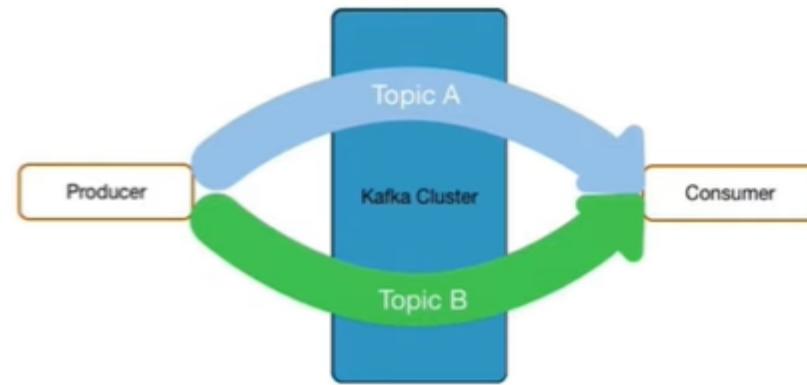


# How Kafka Uses ZooKeeper

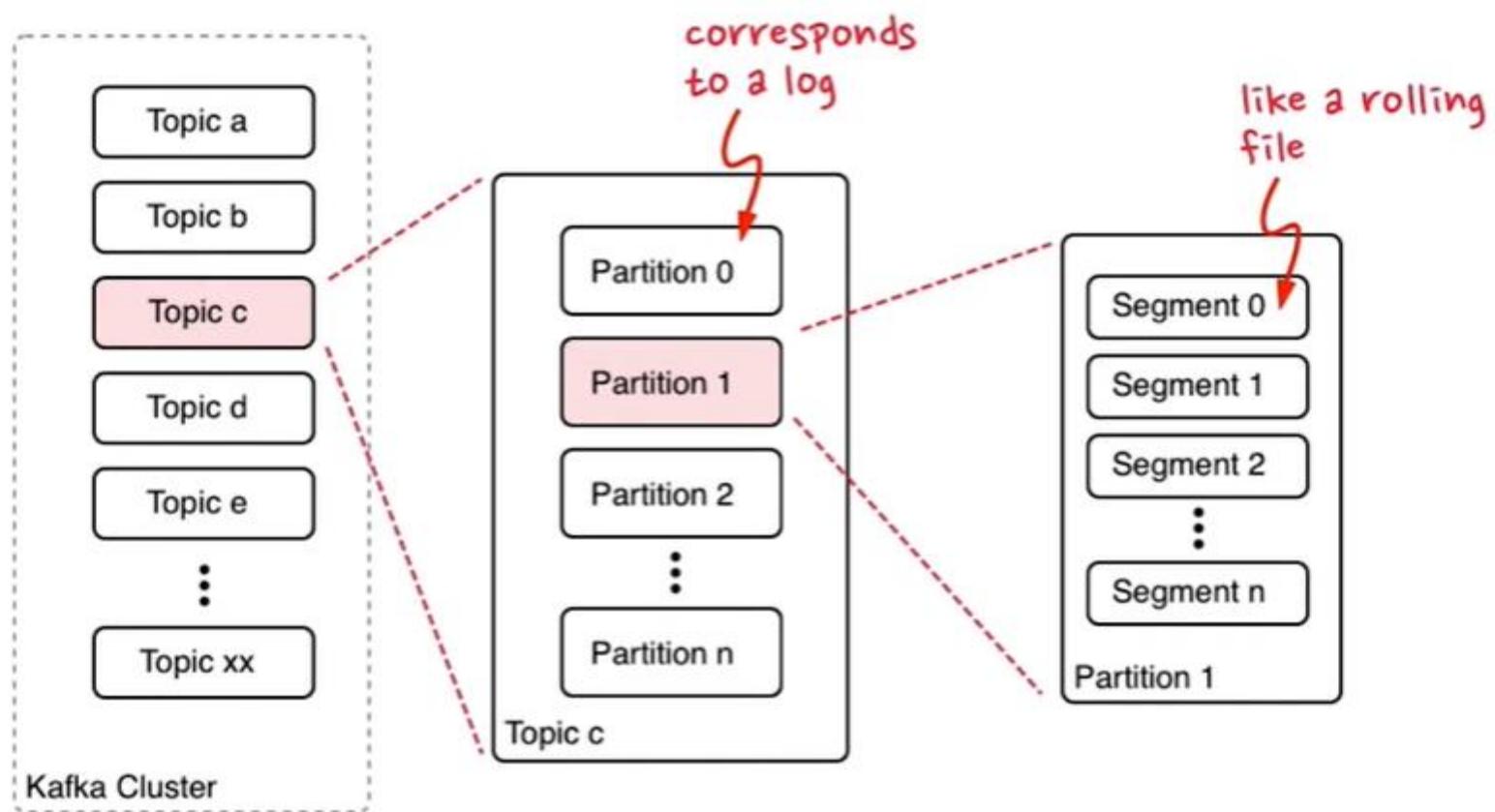


## Topics

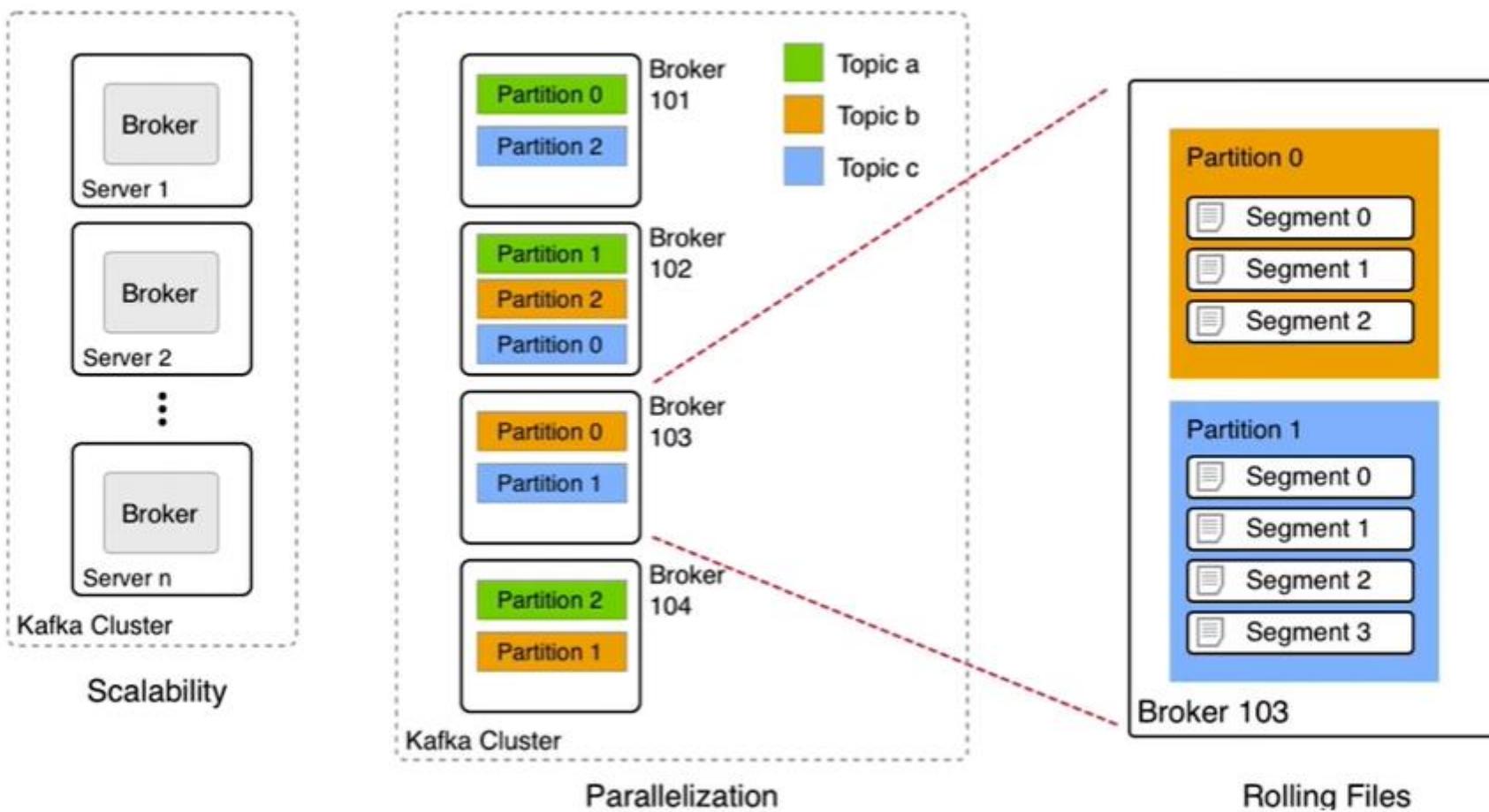
- **Topics:** Streams of "related" Messages in Kafka
  - Is a **Logical Representation**
  - **Categorizes Messages** into Groups
- Developers define Topics
- Producer ↔ Topic: N to N Relation
- Unlimited Number of Topics



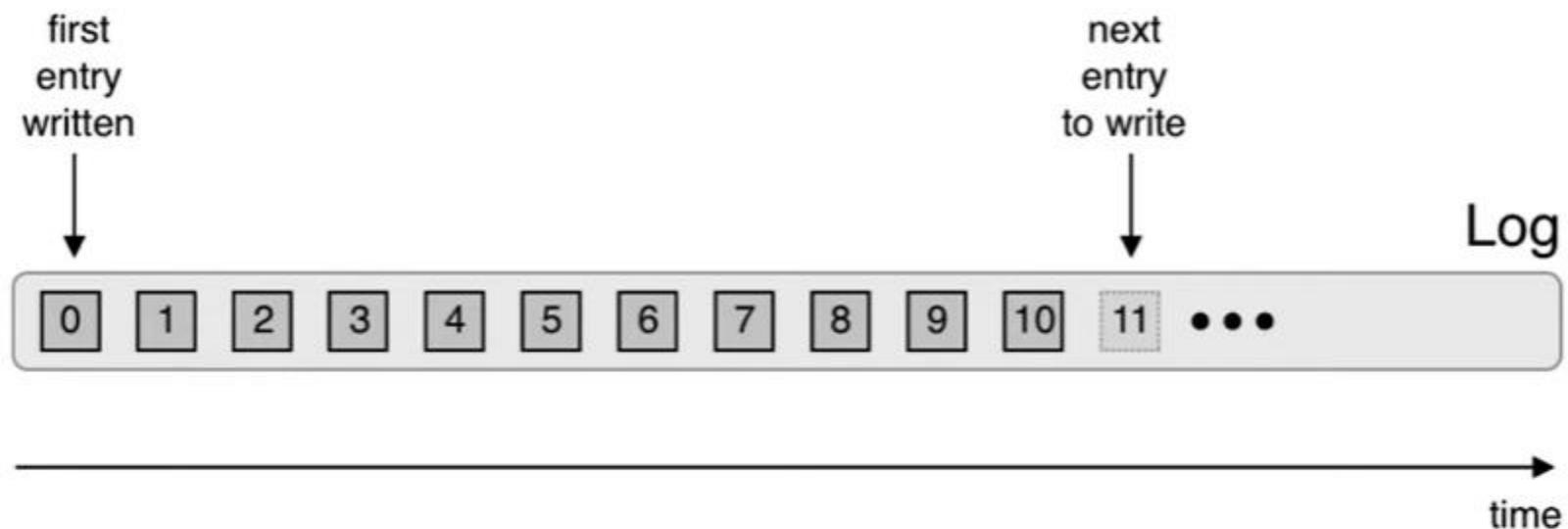
# Topics, Partitions and Segments



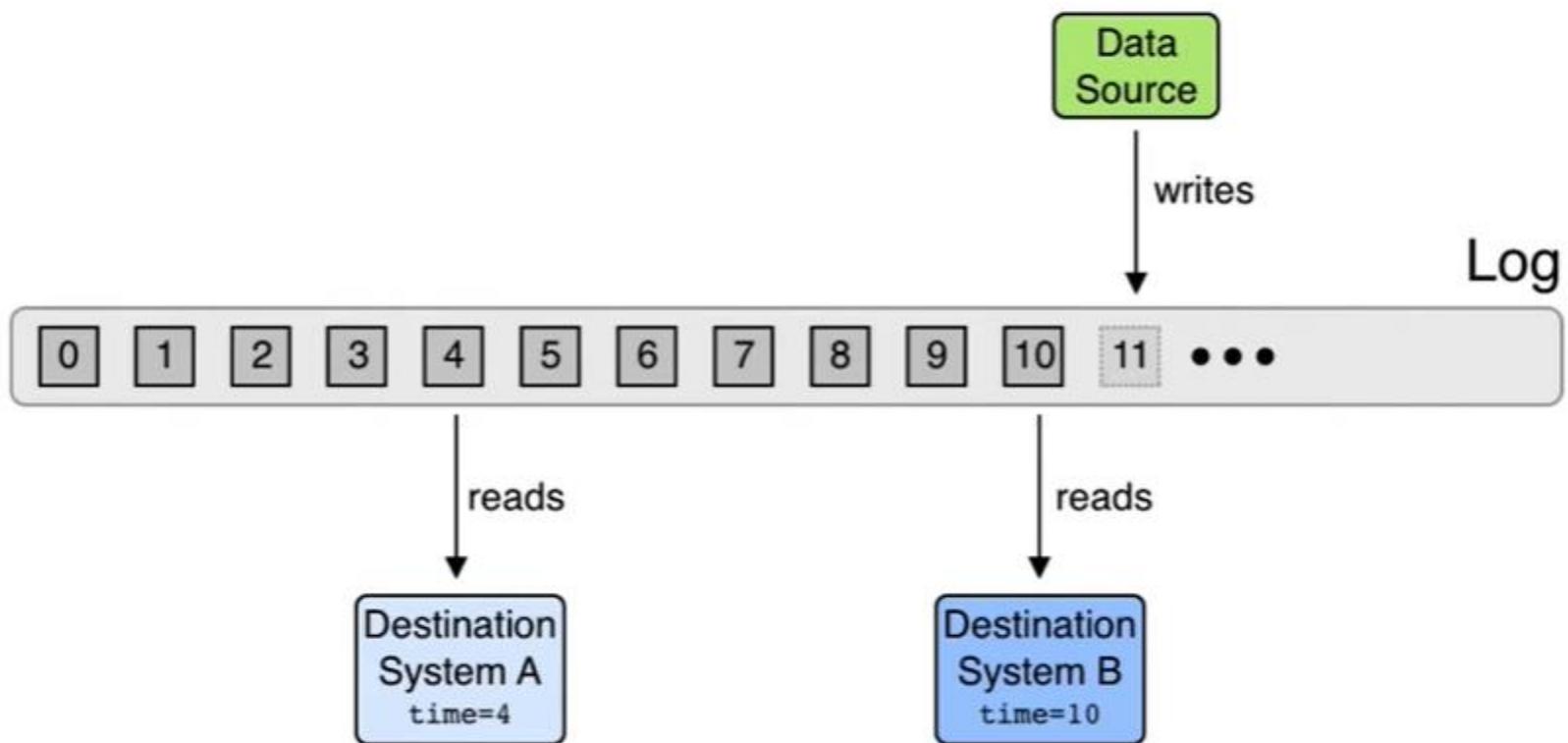
# Topics, Partitions and Segments



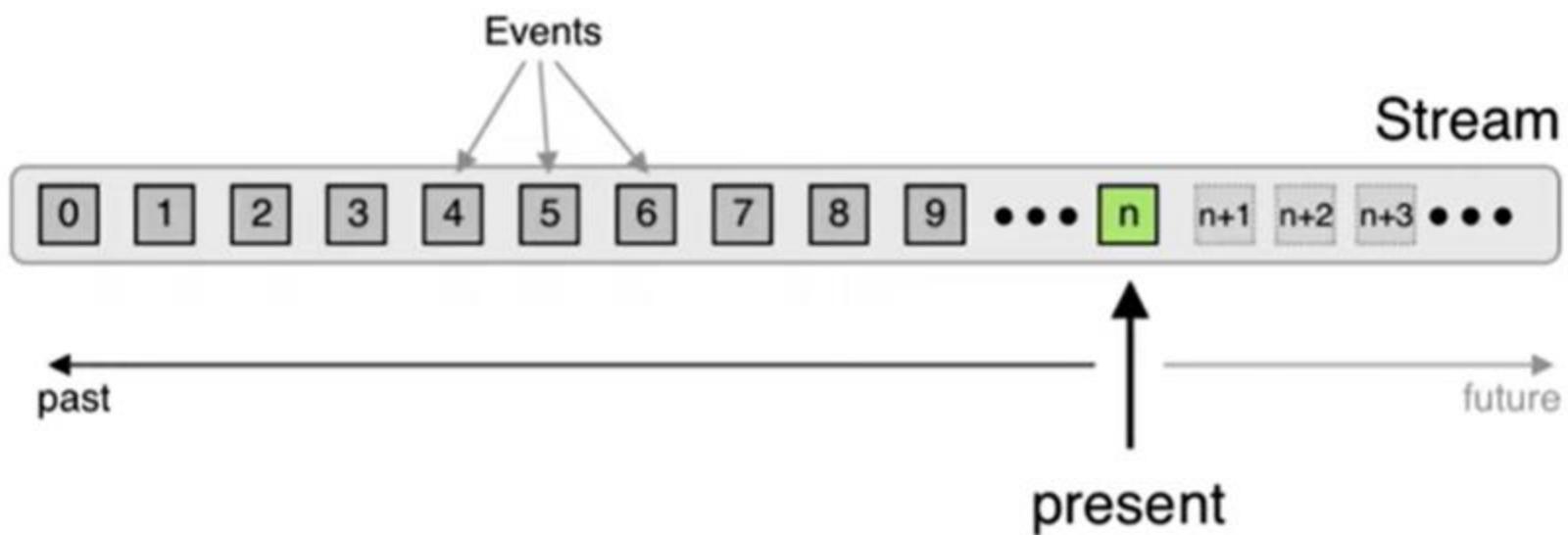
## The Log



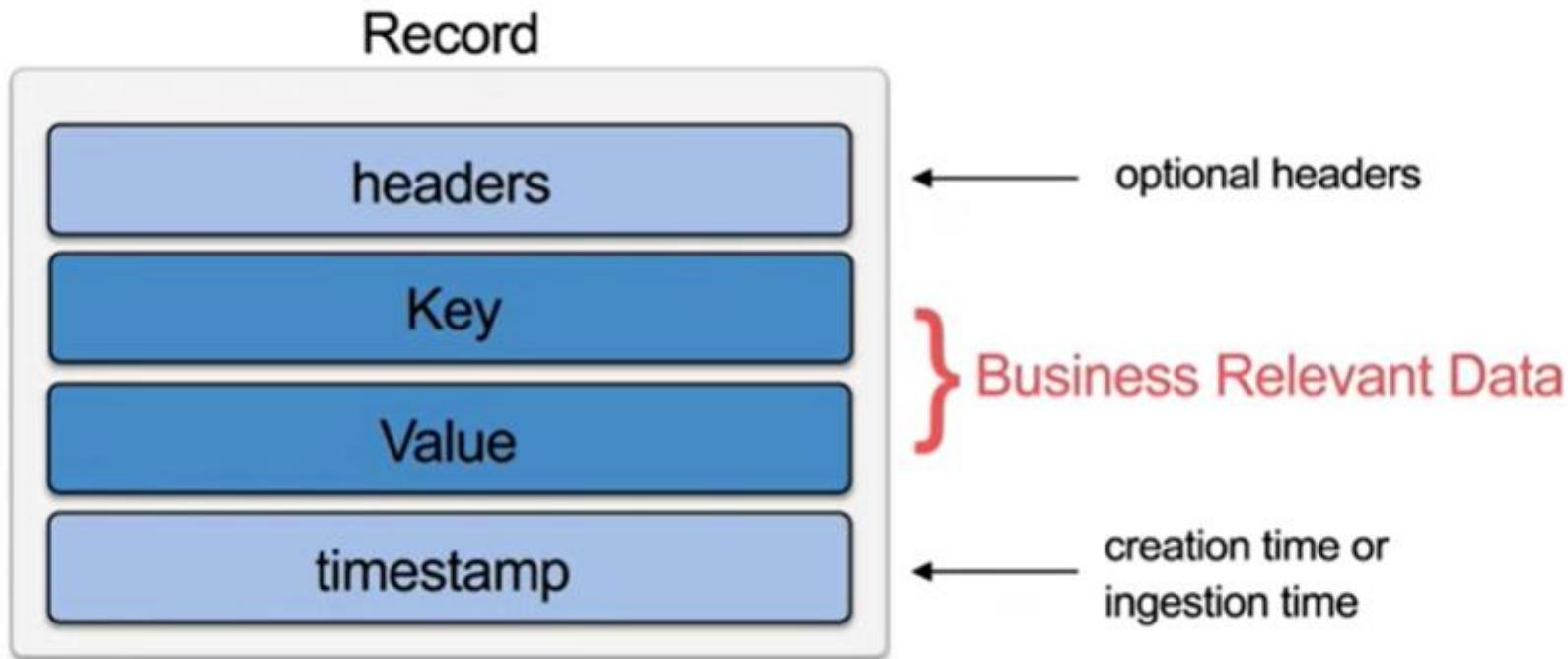
## Log Structured Data Flow



## The Stream



## Data Elements

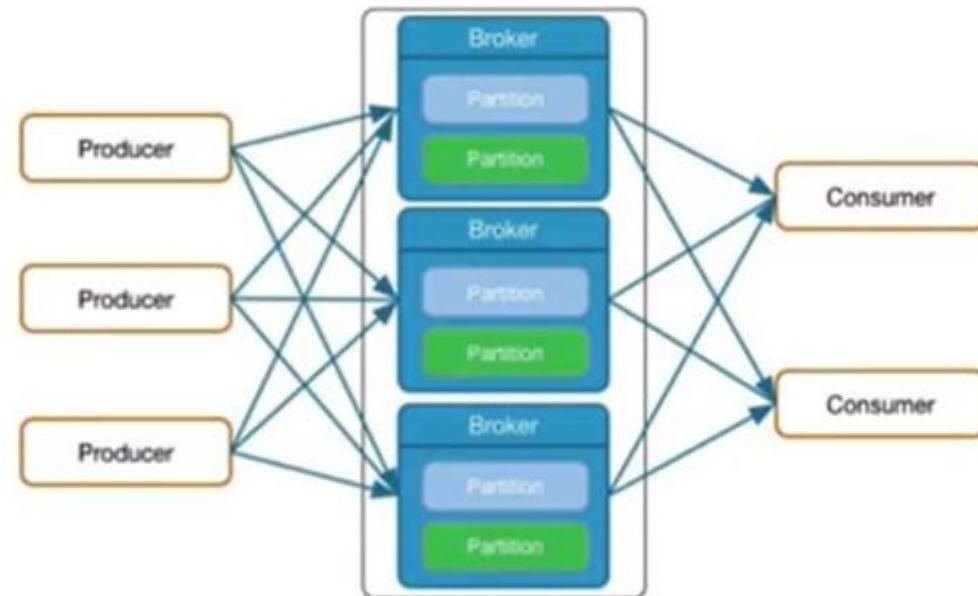


## Brokers Manage Partitions

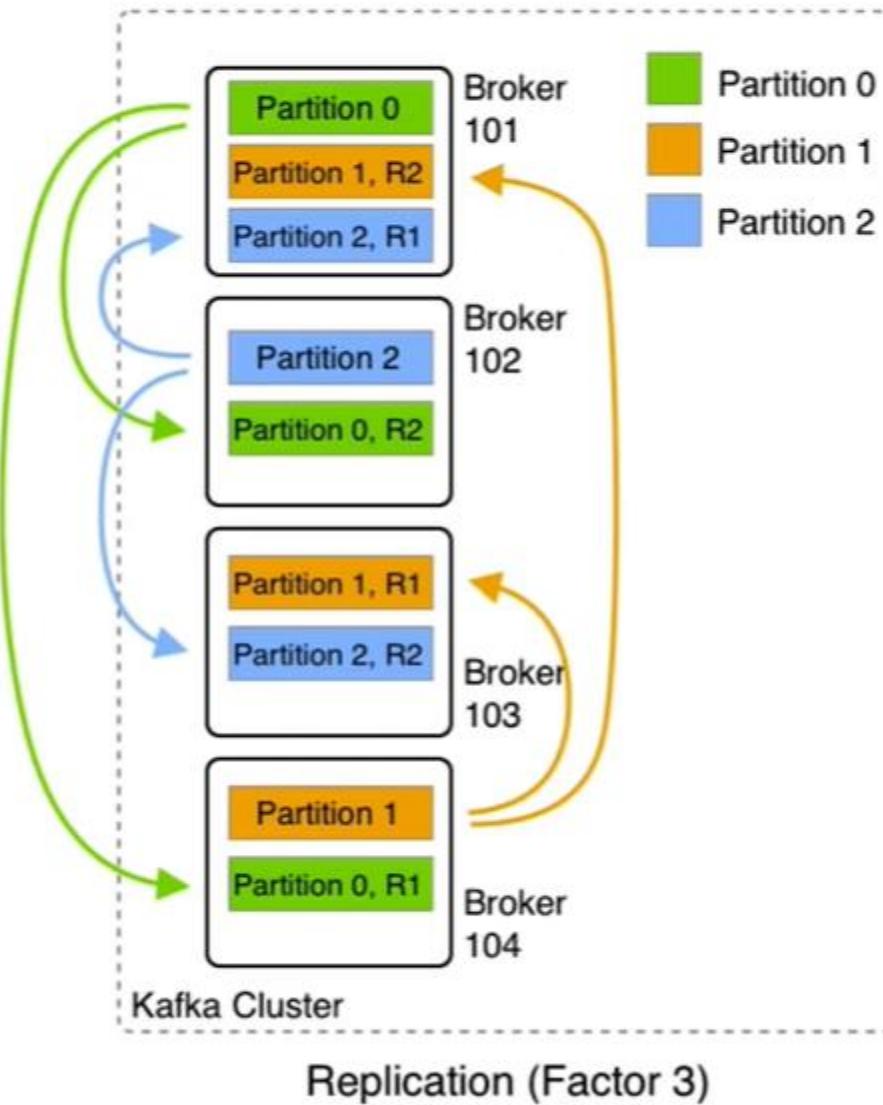
- Messages of Topic spread across Partitions
- Partitions spread across Brokers
- Each Broker handles many Partitions
- Each Partition stored on Broker's disk
- Partition: 1..n **log** files
- Each message in Log identified by Offset
- Configurable Retention Policy

## Broker Basics

- Producer sends Messages to Brokers
- Brokers receive and store Messages
- A Kafka Cluster can have many Brokers
- Each Broker manages multiple Partitions



## Broker Replication



## Producer Basics

- Producers write Data as Messages
- Can be written in any language
  - Native: Java, C/C++, Python, Go, .NET, JMS
  - More Languages by Community
  - REST Proxy for any unsupported Language
- Command Line Producer Tool

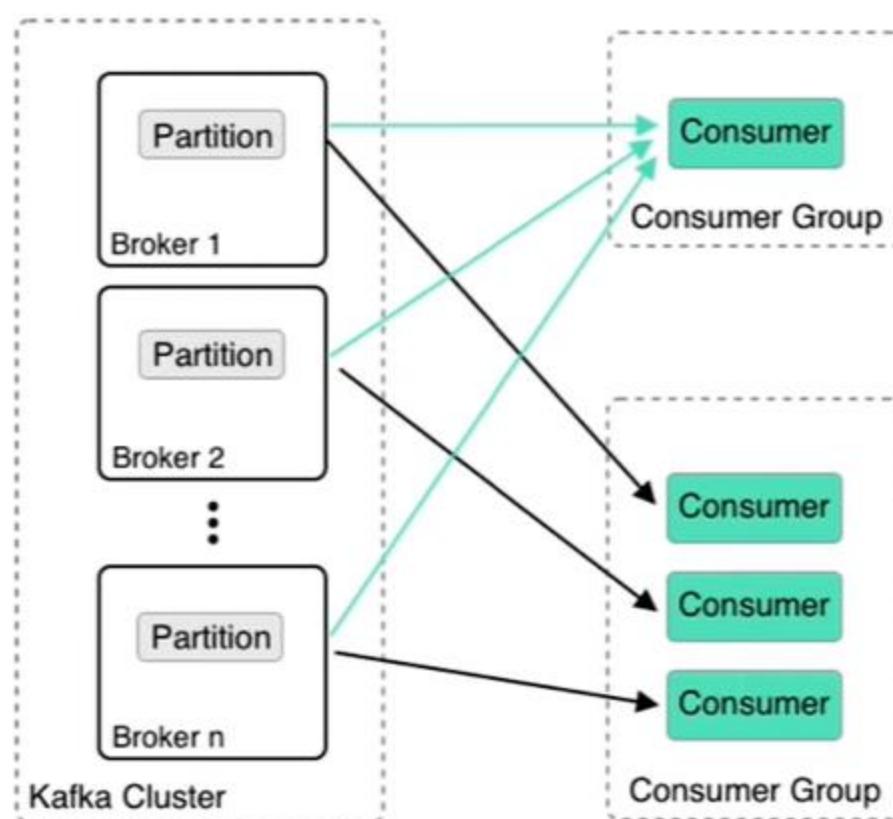
## Load Balancing and Semantic Partitioning

- Producers use a Partitioning Strategy to assign each Message to a Partition
- Two Purposes:
  - Load Balancing
  - Semantic Partitioning
- Partitioning Strategy specified by Producer
  - Default Strategy: `hash(key) % number_of_partitions`
  - No Key → Round-Robin
- Custom Partitioner possible

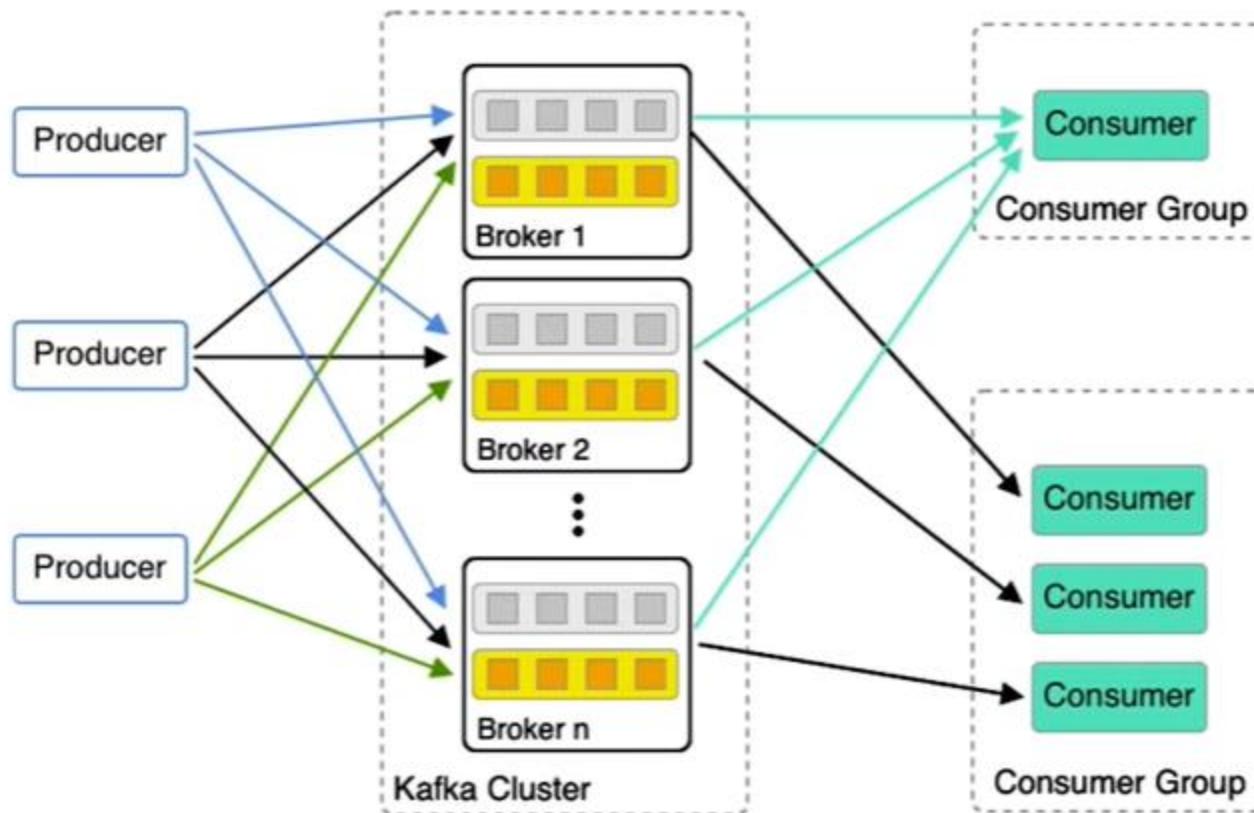
## Consumer Basics

- Consumers **pull** messages from 1...n topics
- New inflowing messages are automatically retrieved
- Consumer offset
  - keeps track of the last message read
  - is stored in special topic
- CLI tools exist to read from cluster

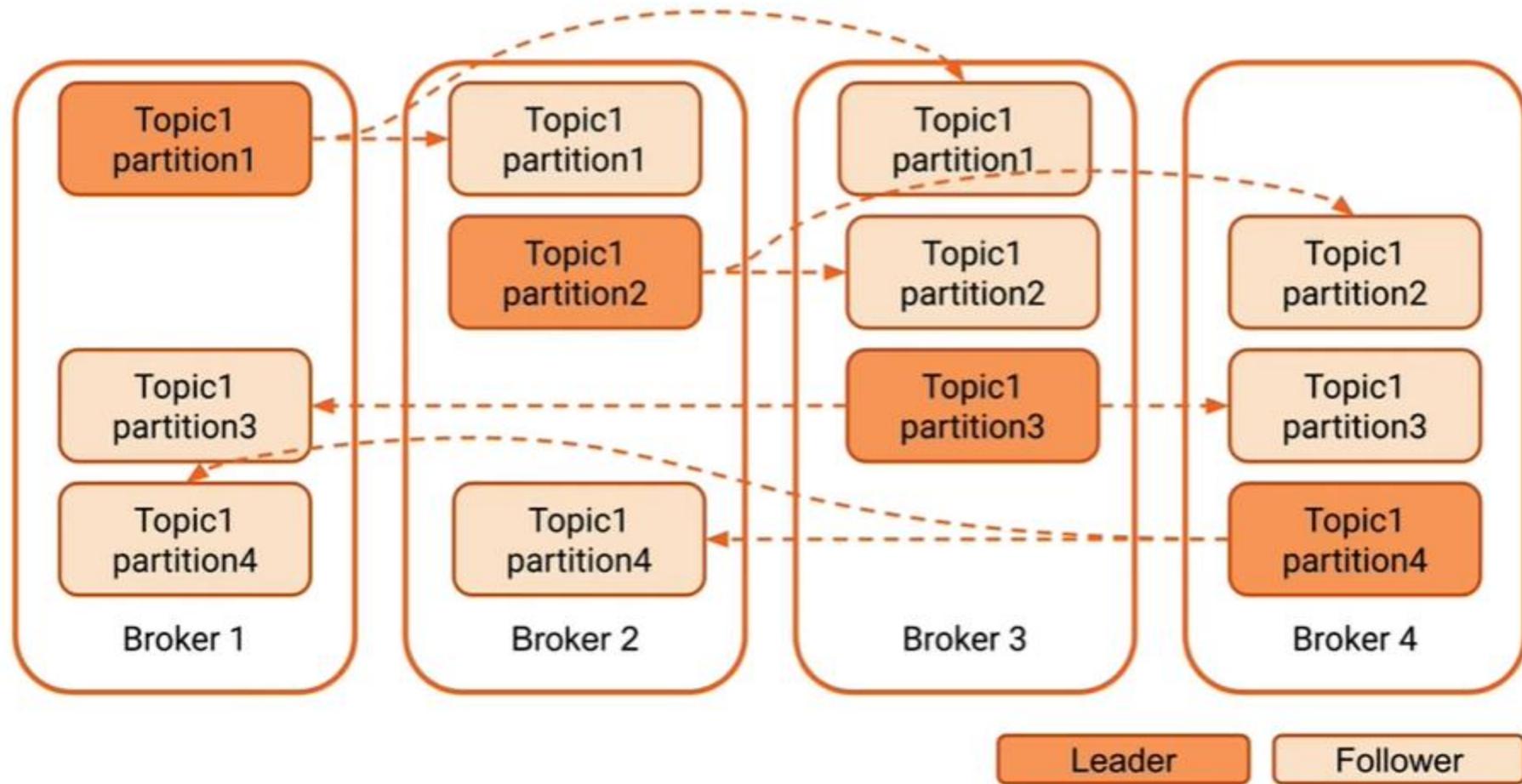
## Distributed Consumption



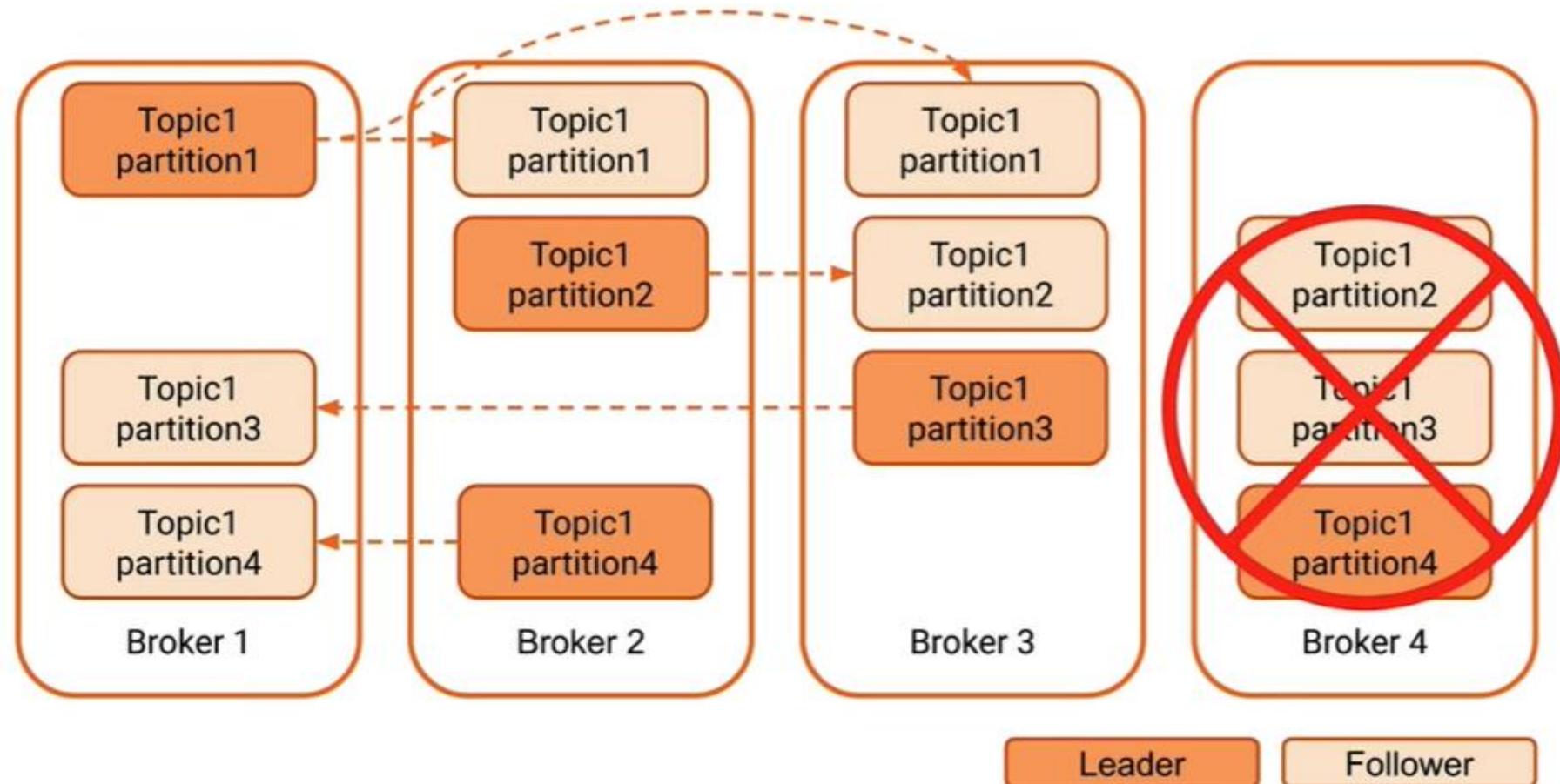
# Scalable Data Pipeline



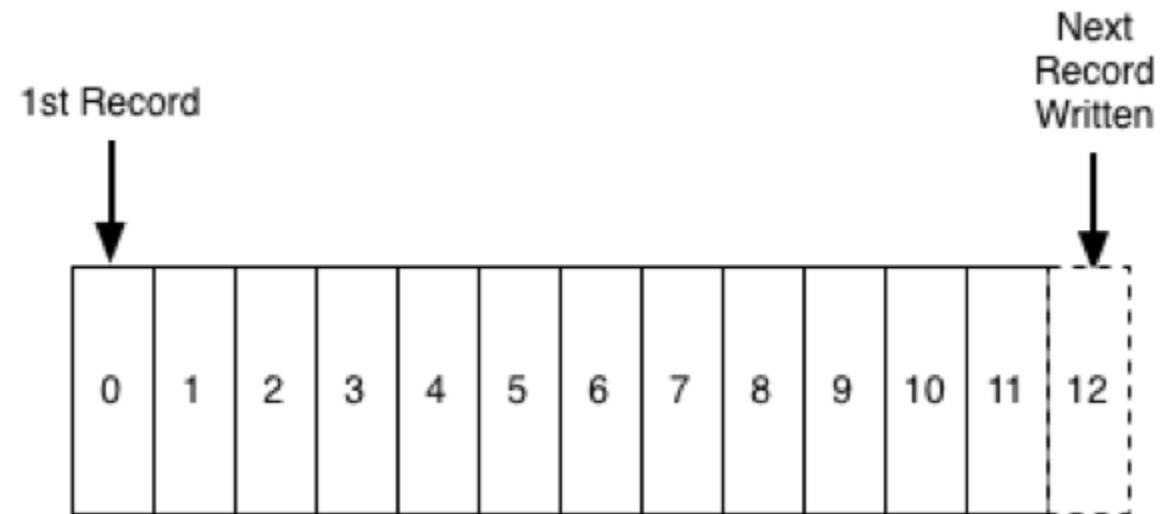
## Partition Leadership & Replication



## Partition Leadership & Replication



# Store data in Kafka?



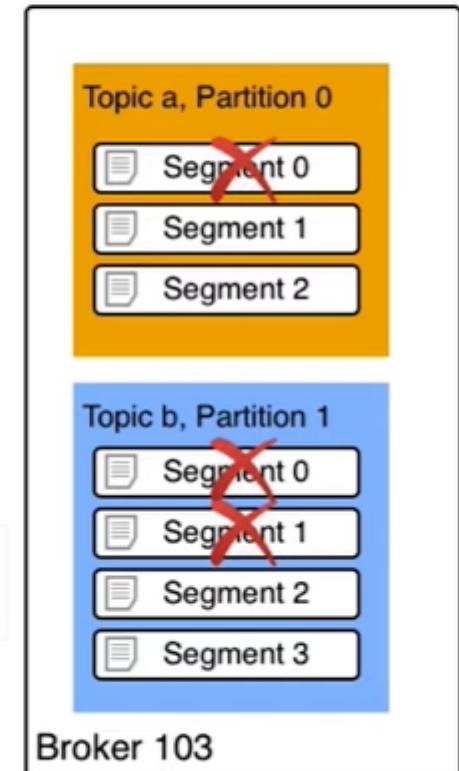
## Data Retention Policy

How long do I want or can I store my data?

- How long (default: 1 week)
- Set globally or per topic
- Business decision
- Cost factor
- Compliance factor → GDPR

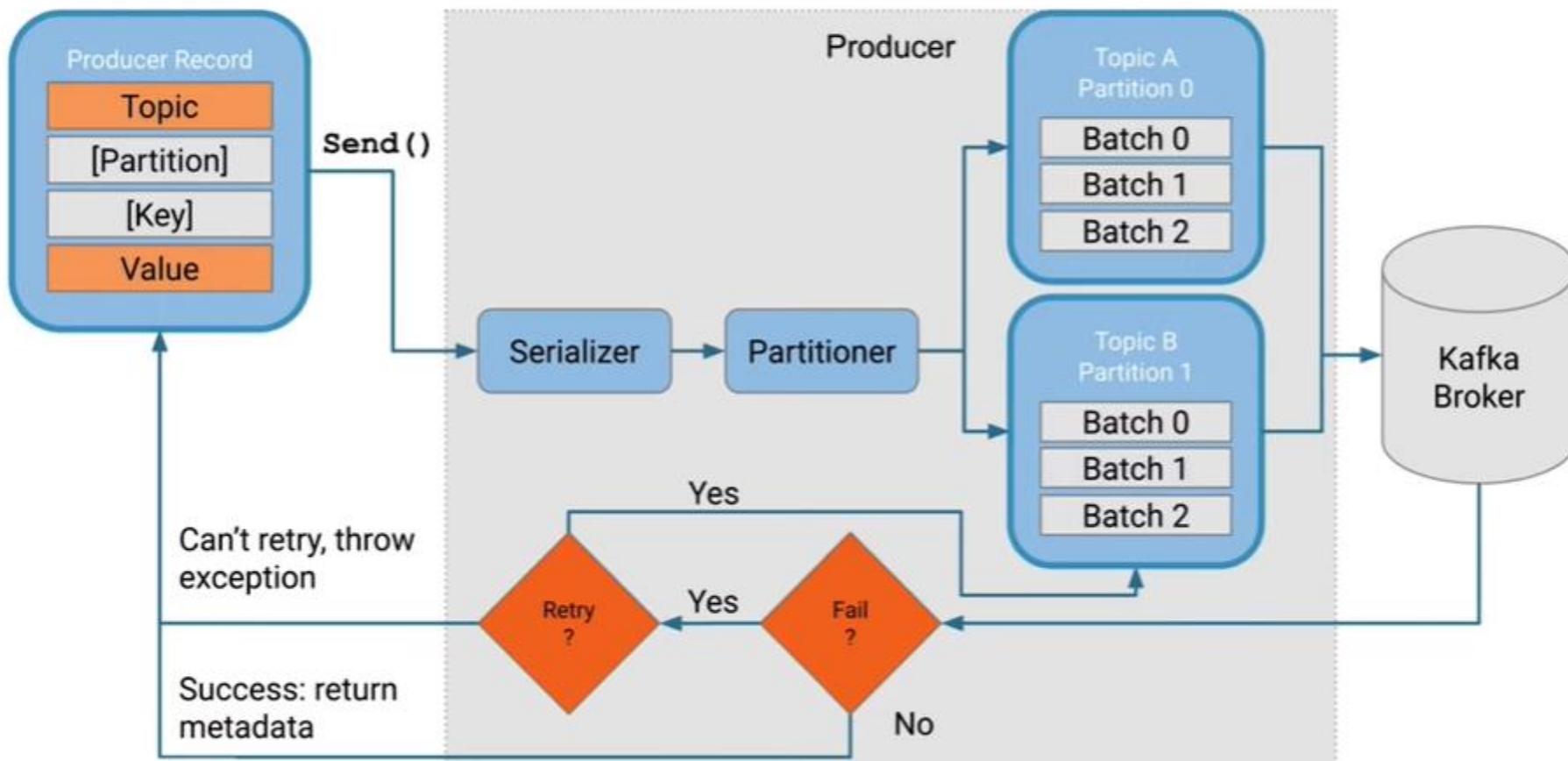


Data purged per segment

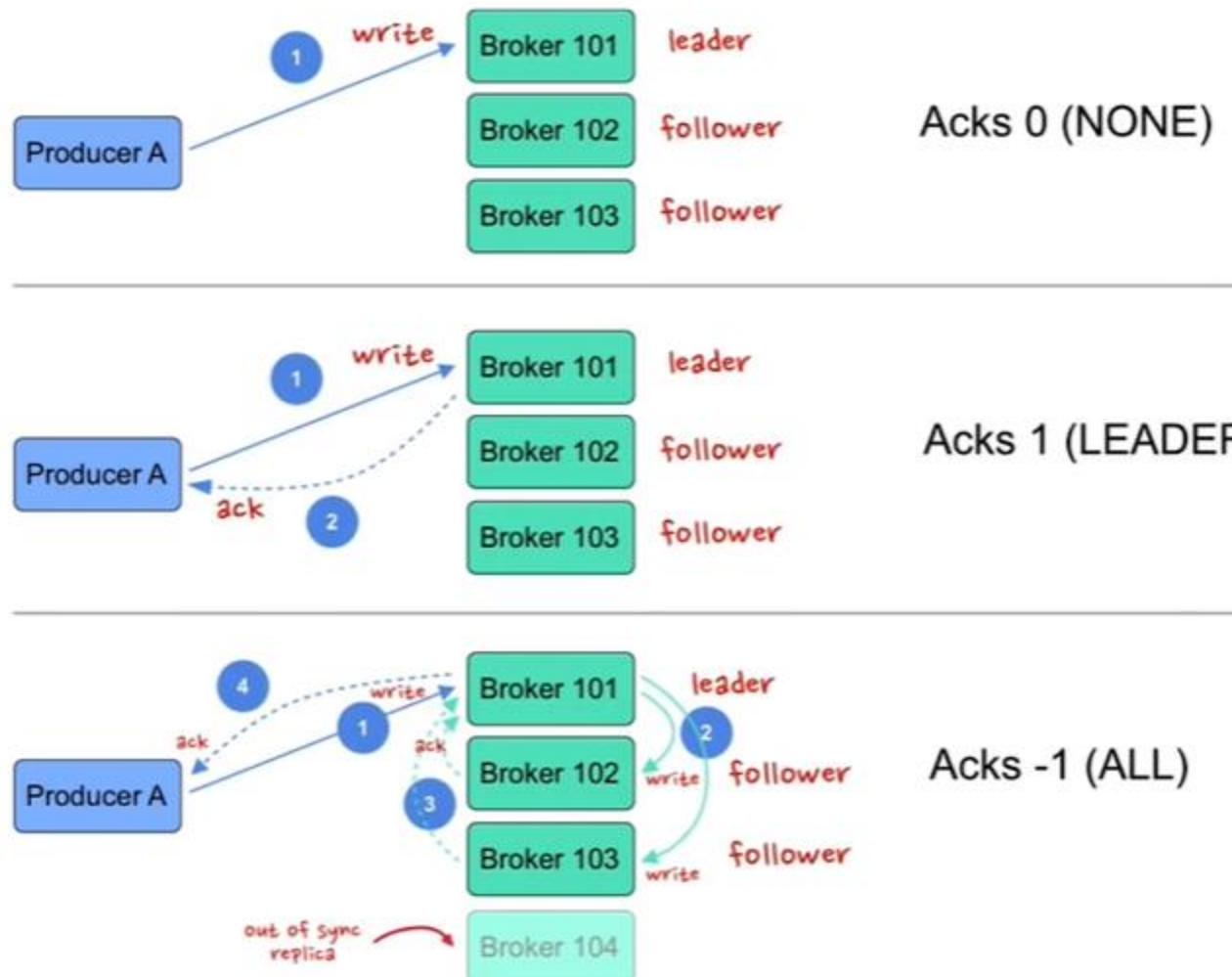


Retention Policy

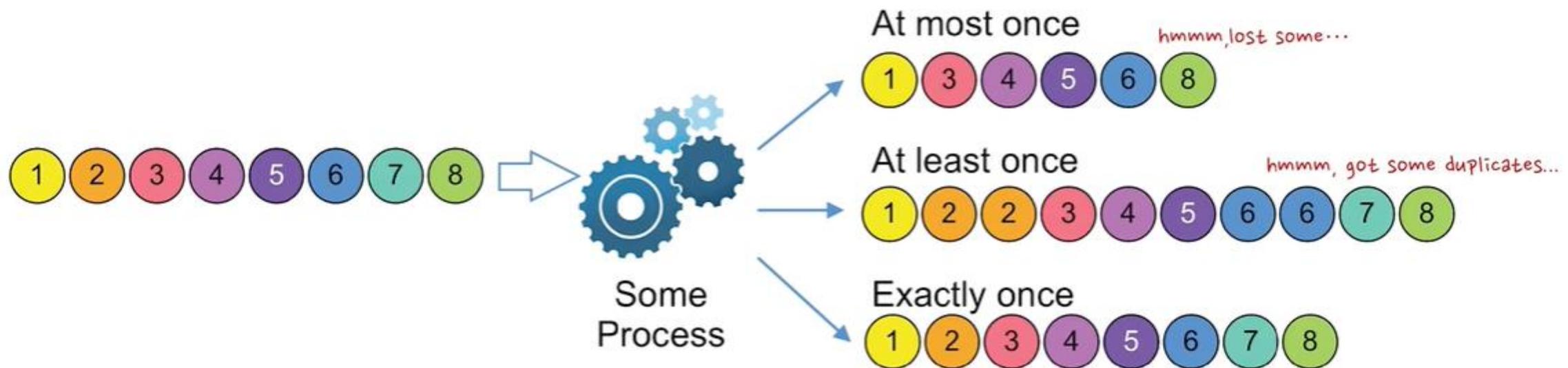
## Producer Design



## Producer Guarantees



## Delivery Guarantees

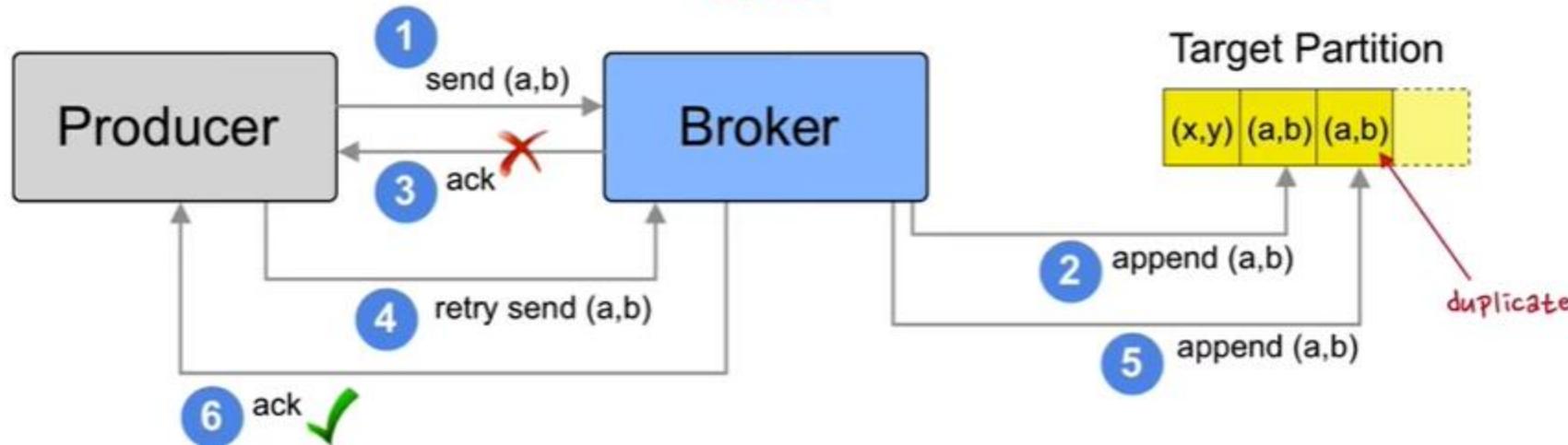


## Idempotent Producers

**GOOD**



**BAD**



## Exactly Once Semantics

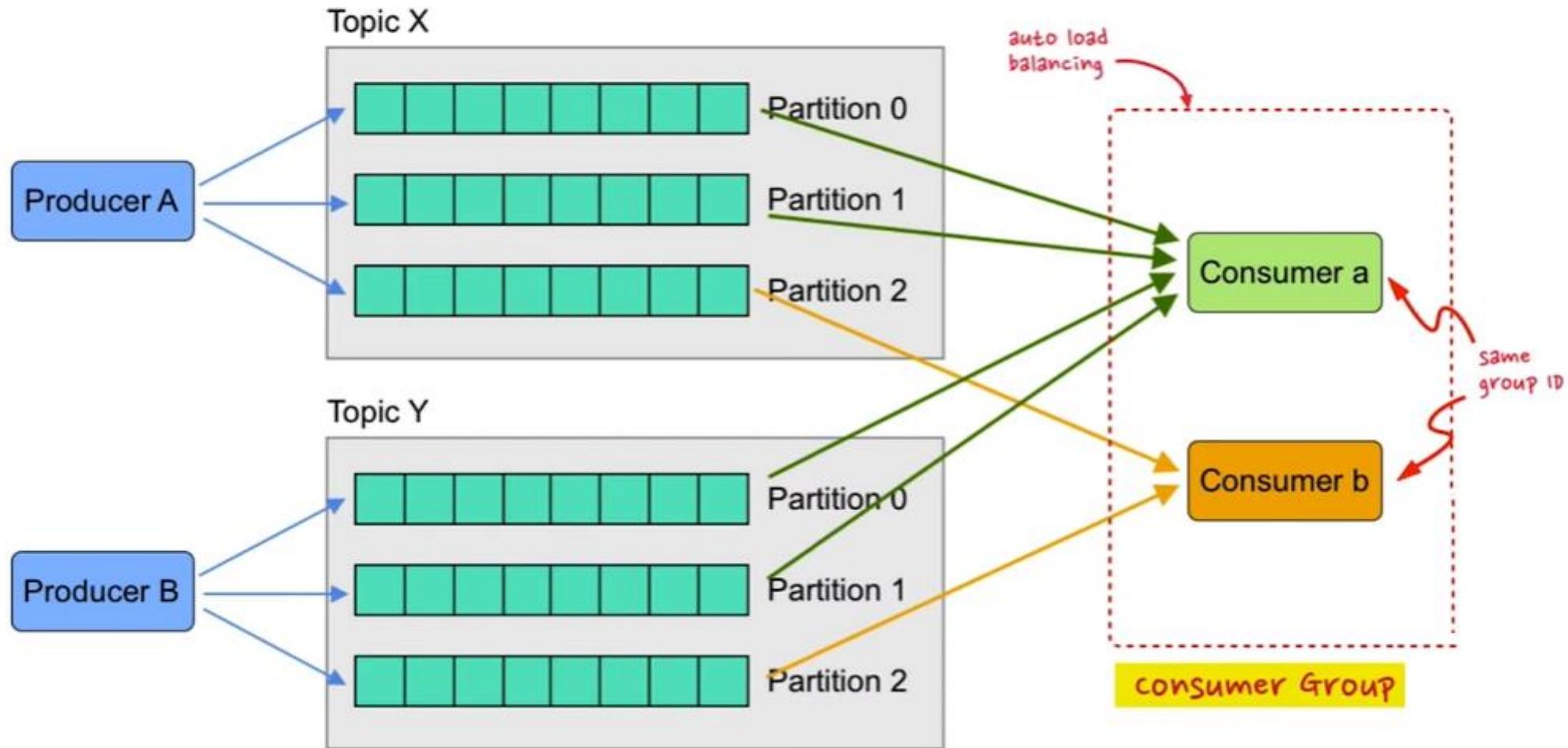
### What?

- Strong **transactional guarantees** for Kafka
- Prevents clients from processing duplicate messages
- Handles failures gracefully

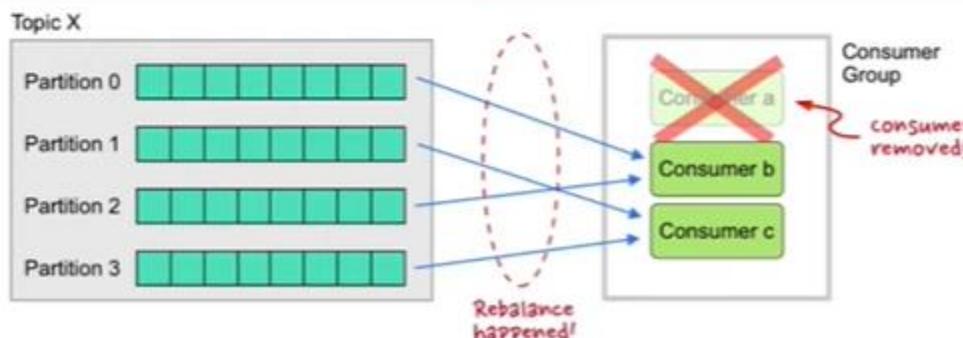
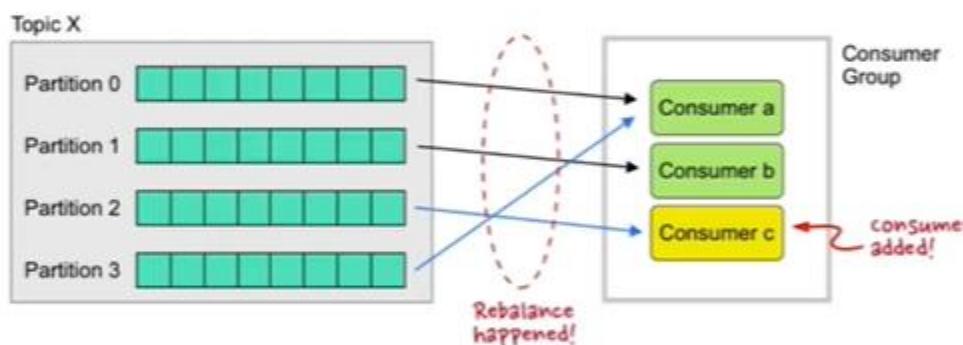
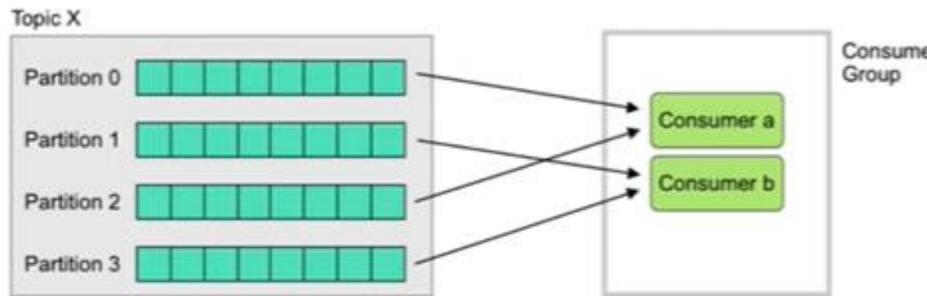
### Use Cases

- Tracking ad views
- Processing financial transactions
- Stream processing

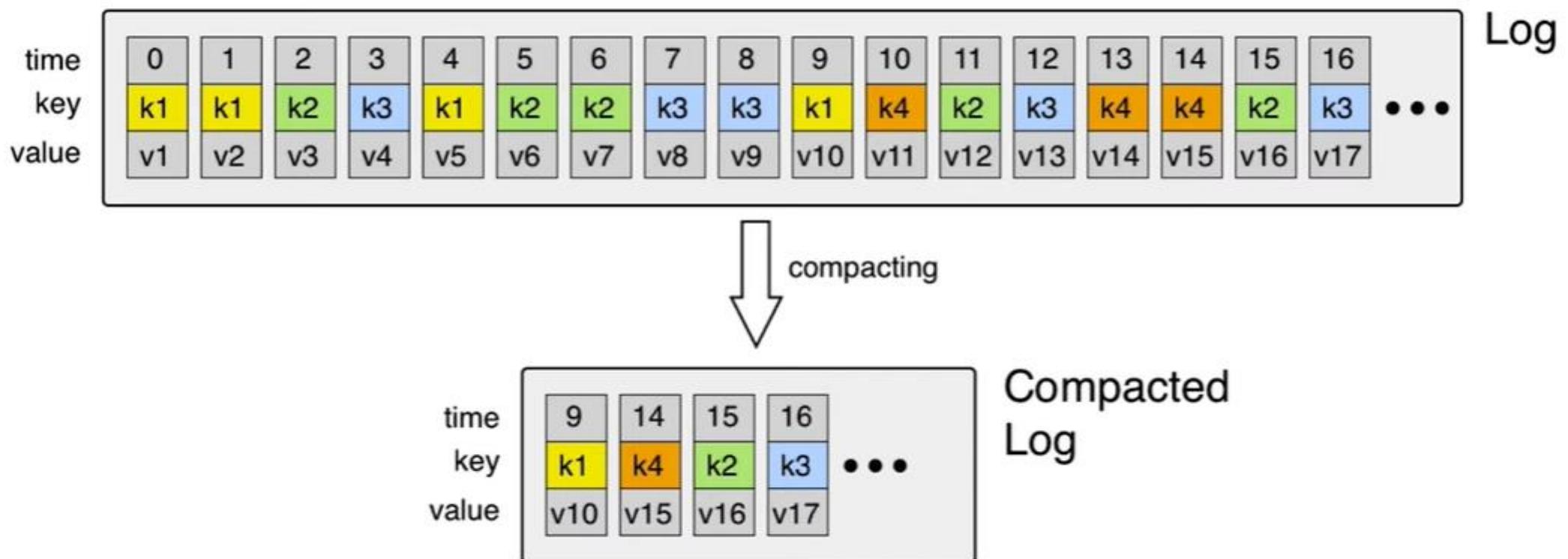
## Consumer Groups



# Consumer Rebalances

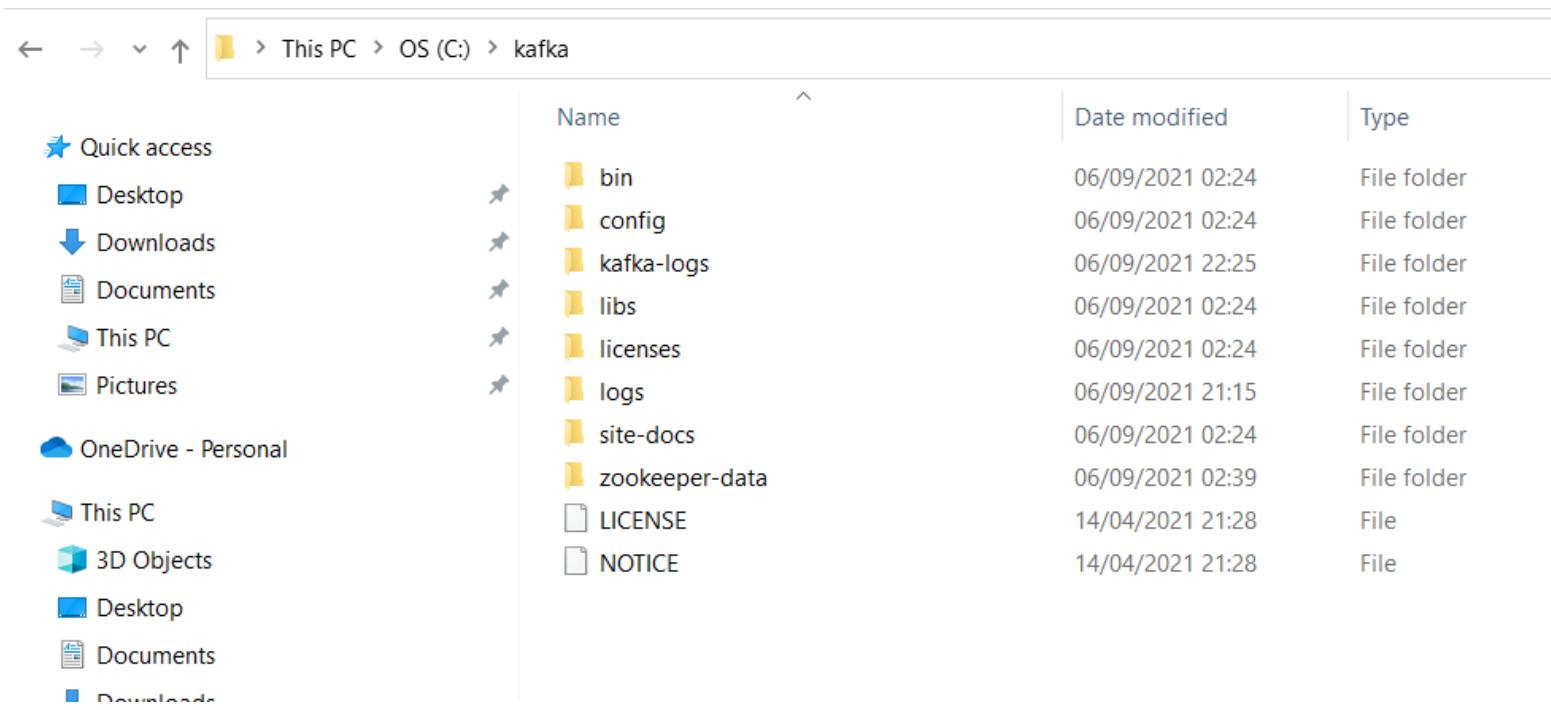


## Compacted Topics



# Tutorial 1: Kafka installation on Window

- Download Kafka from <https://kafka.apache.org/>
- Unzip the download file
- Rename the kafka to “kafka” and move it to C:\ drive



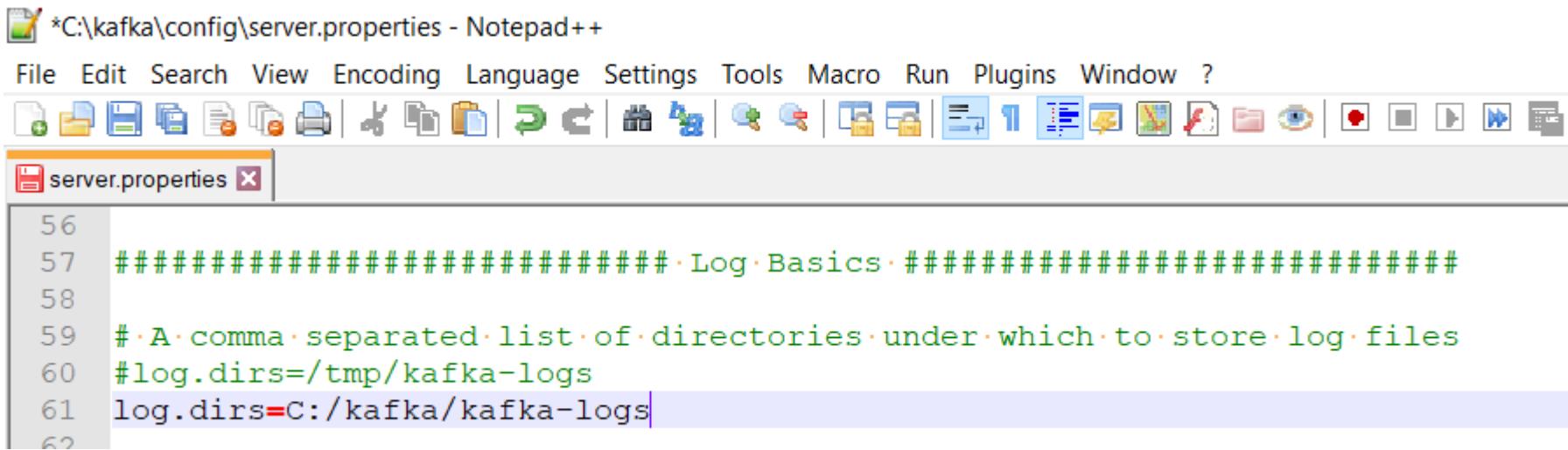
The screenshot shows a Windows File Explorer window with the following details:

Path: This PC > OS (C:) > kafka

Name	Date modified	Type
bin	06/09/2021 02:24	File folder
config	06/09/2021 02:24	File folder
kafka-logs	06/09/2021 22:25	File folder
libs	06/09/2021 02:24	File folder
licenses	06/09/2021 02:24	File folder
logs	06/09/2021 21:15	File folder
site-docs	06/09/2021 02:24	File folder
zookeeper-data	06/09/2021 02:39	File folder
LICENSE	14/04/2021 21:28	File
NOTICE	14/04/2021 21:28	File

# Kafka installation on Window

- Open C:\kafka\config\server.properties
- Change the path of log.dir

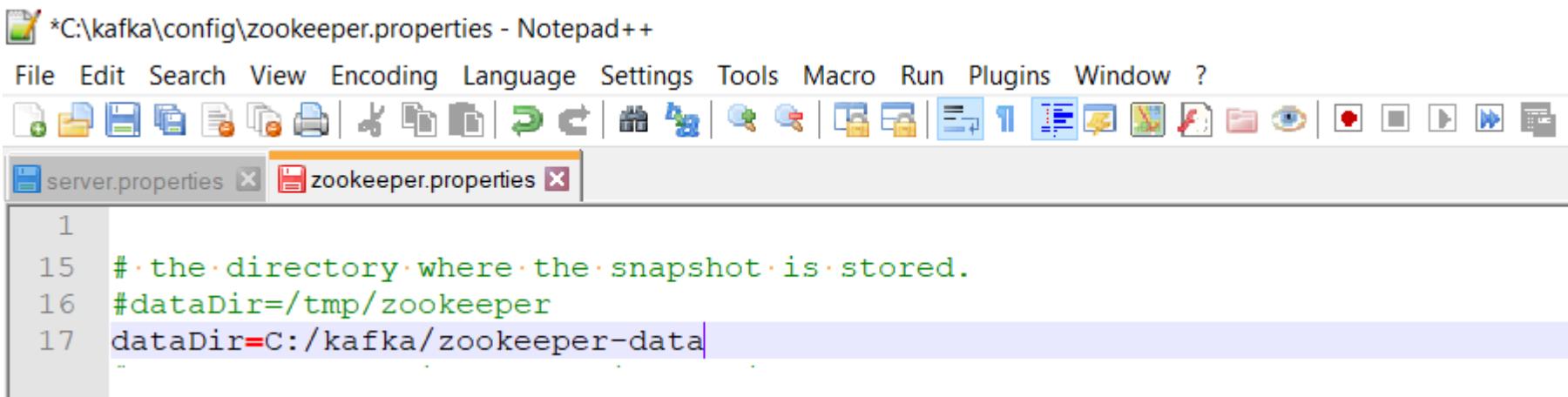


The screenshot shows the Notepad++ application window with the file 'server.properties' open. The window title is '\*C:\kafka\config\server.properties - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar below the menu bar contains various icons for file operations like Open, Save, Print, Find, and Replace. The status bar at the bottom shows line numbers from 56 to 62. The code in the editor is as follows:

```
56 ###### Log Basics #####
57 # A comma separated list of directories under which to store log files
58 #log.dirs=/tmp/kafka-logs
59 log.dirs=C:/kafka/kafka-logs
```

# Kafka installation on Window

- Open C:\kafka\config\zookeeper.properties
- Change the path of dataDir



\*C:\kafka\config\zookeeper.properties - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

server.properties x zookeeper.properties x

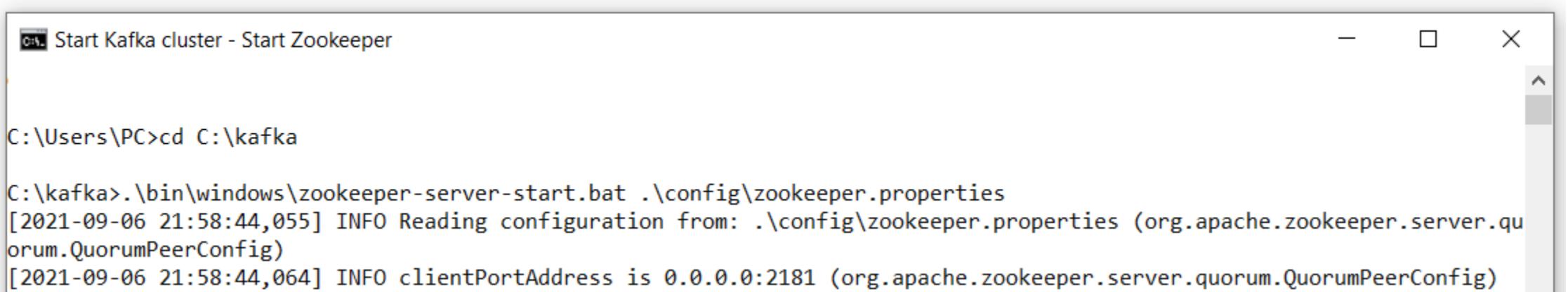
```
1
15 # the directory where the snapshot is stored.
16 #dataDir=/tmp/zookeeper
17 dataDir=C:/kafka/zookeeper-data
```

- By default Apache Kafka will run on port 9092 and Apache Zookeeper will run on port 2181.

# Tutorial 2: Run Apache Kafka on Windows

- Start the Kafka cluster
  - Run the following command to start ZooKeeper:

```
cd C:\kafka\  
.\\bin\\windows\\zookeeper-server-start.bat .\\config\\zookeeper.properties
```



```
C:\Users\PC>cd C:\kafka  
  
C:\kafka>.\\bin\\windows\\zookeeper-server-start.bat .\\config\\zookeeper.properties  
[2021-09-06 21:58:44,055] INFO Reading configuration from: .\\config\\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)  
[2021-09-06 21:58:44,064] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
```

# Run Apache Kafka on Windows

- Start the Kafka cluster
  - Run the following command to start the Kafka broker:

```
cd C:\kafka\  
.\\bin\\windows\\kafka-server-start.bat .\\config\\server.properties
```



The screenshot shows a Windows Command Prompt window with the title "Start Kafka cluster - Start Kafka broker". The command entered is:

```
C:\\Users\\PC>cd C:\\kafka  
C:\\kafka>.\\bin\\windows\\kafka-server-start.bat .\\config\\server.properties
```

Output from the command:

```
[2021-09-06 22:03:25,045] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
```

# Run Apache Kafka on Windows

- Produce and consume some messages
  - Run the kafka-topics command to create a Kafka topic named TestTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic TestTopic
```

- Let's create another topic named NewTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic NewTopic
```

- Let's show list of created topics

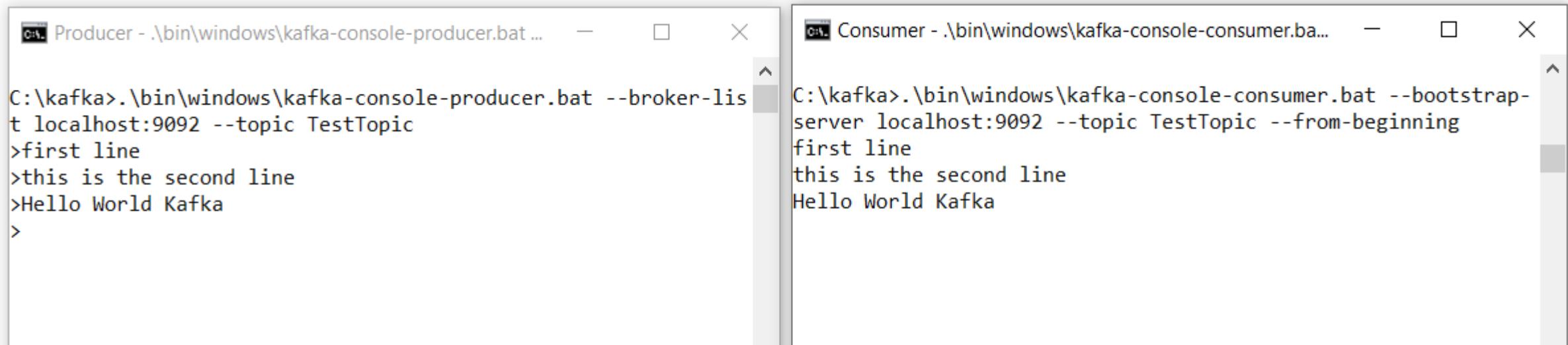
```
.\bin\windows\kafka-topics.bat --list --zookeeper localhost:2181
```

# Run Apache Kafka on Windows

- Produce and consume some messages
  - Run the producer and consumer on separate Command Prompt:

```
.\bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic TestTopic
```

```
.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TestTopic --from-beginning
```



The image shows two separate Command Prompt windows running on Windows. The left window, titled 'Producer - .\bin\windows\kafka-console-producer.bat ...', displays the command being run and three lines of text being produced: 'first line', 'this is the second line', and 'Hello World Kafka'. The right window, titled 'Consumer - .\bin\windows\kafka-console-consumer.bat ...', displays the command being run and the same three lines of text being consumed: 'first line', 'this is the second line', and 'Hello World Kafka'.

```
C:\kafka>.\bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic TestTopic
>first line
>this is the second line
>Hello World Kafka
>
```

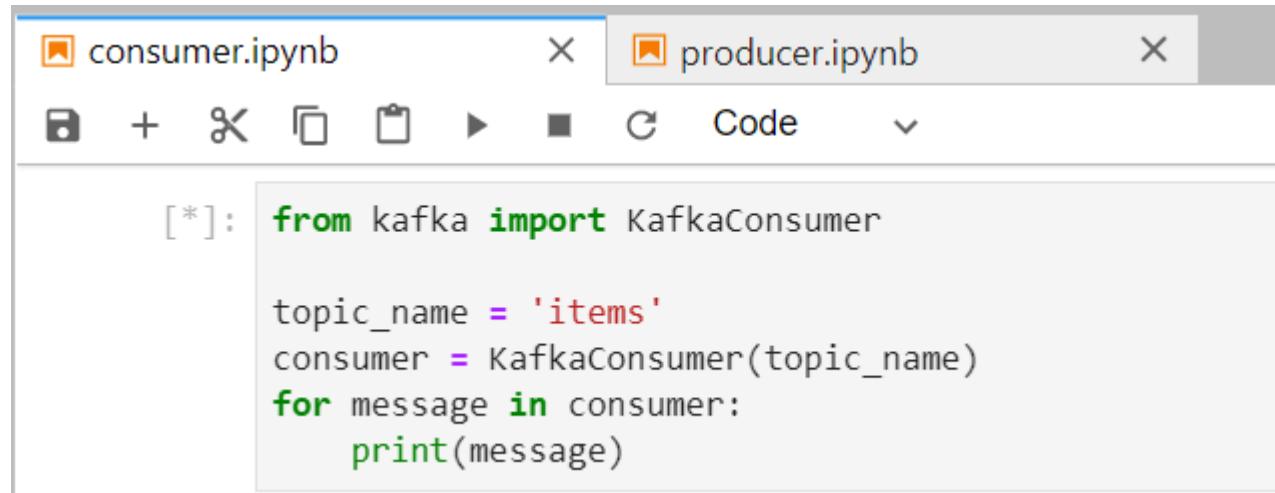
```
C:\kafka>.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TestTopic --from-beginning
first line
this is the second line
Hello World Kafka
```

# Tutorial 3: Kafka Python client

- <https://kafka-python.readthedocs.io/en/master/index.html>
- Install Kafka-Python
  - pip install kafka-python
- Start Zookeeper server and Kafka broker
  - Zookeeper is running default on localhost:2181 and Kafka on localhost:9092

# Kafka-Python

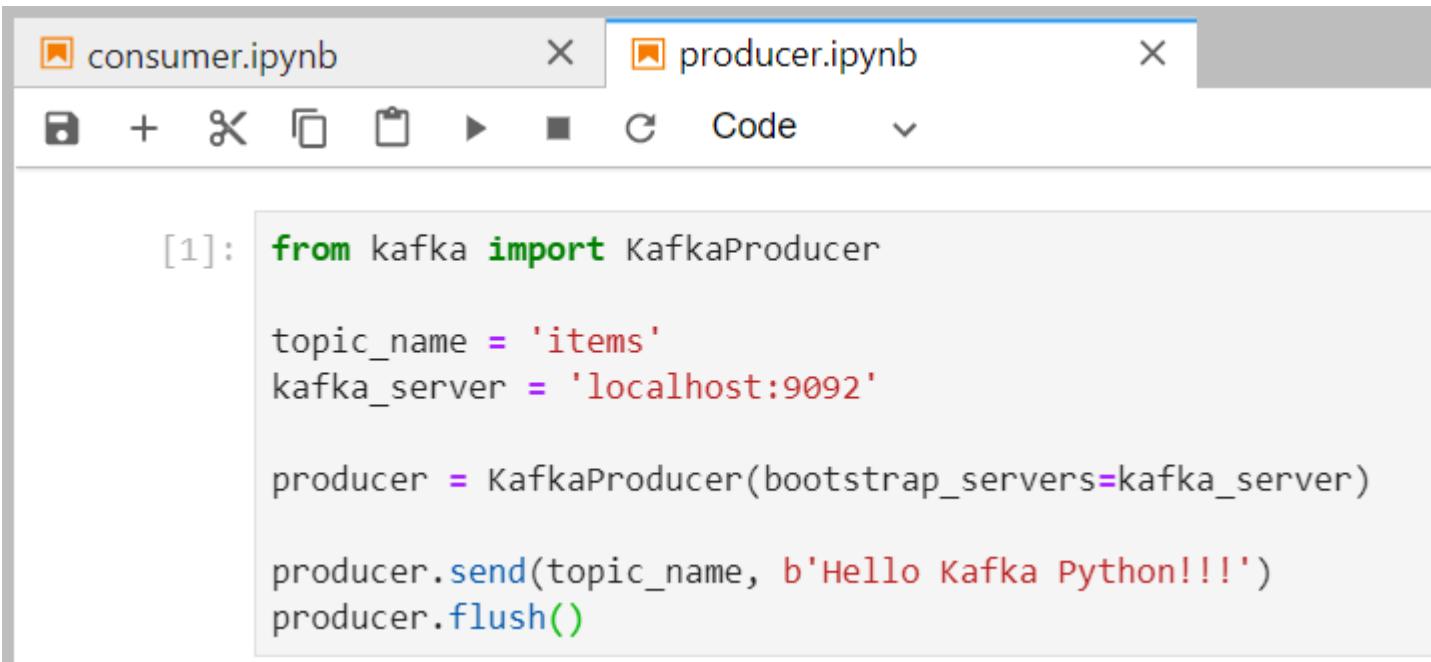
- Run consumer code



```
[*]: from kafka import KafkaConsumer  
  
topic_name = 'items'  
consumer = KafkaConsumer(topic_name)  
for message in consumer:  
    print(message)
```

# Kafka-Python

- Run producer code



```
[1]: from kafka import KafkaProducer  
  
topic_name = 'items'  
kafka_server = 'localhost:9092'  
  
producer = KafkaProducer(bootstrap_servers=kafka_server)  
  
producer.send(topic_name, b'Hello Kafka Python!!!')  
producer.flush()
```

# Kafka-Python

- Check the result

The screenshot shows a Jupyter Notebook interface with two tabs open: "consumer.ipynb" and "producer.ipynb". The "producer.ipynb" tab is currently selected. The code cell contains the following Python code:

```
[*]: from kafka import KafkaConsumer  
  
topic_name = 'items'  
consumer = KafkaConsumer(topic_name)  
for message in consumer:  
    print(message)
```

The output of the code is displayed below the cell, showing a single message object:

```
ConsumerRecord(topic='items', partition=0, offset=0, timestamp=1630946348692, timestamp_type=0, key=None, value=b'Hello Kafka P  
ython!!!', headers=[], checksum=None, serialized_key_size=-1, serialized_value_size=21, serialized_header_size=-1)
```

# Tutorial 4: Run Kafka on Colab

- Download Kafka and unzip

```
!curl -sSOL https://downloads.apache.org/kafka/3.3.1/kafka_2.13-3.3.1.tgz  
!tar -xzf kafka_2.13-3.3.1.tgz
```

- Start zookeeper server and kafka server

```
!./kafka_2.13-3.3.1/bin/zookeeper-server-start.sh -daemon ./kafka_2.13-3.3.1/config/zookeeper.properties  
!./kafka_2.13-3.3.1/bin/kafka-server-start.sh -daemon ./kafka_2.13-3.3.1/config/server.properties
```

- Create a topic

```
!./kafka_2.13-3.3.1/bin/kafka-topics.sh --create --bootstrap-server 127.0.0.1:9092 --replication-factor 1 --partitions 1 --topic TestTopic
```

# Run Kafka on Colab

- Describe the created topic

```
!./kafka_2.13-3.3.1/bin/kafka-topics.sh --describe --bootstrap-server 127.0.0.1:9092 --topic TestTopic
```

- Write some event in the topic

```
!./kafka_2.13-3.3.1/bin/kafka-console-producer.sh --topic TestTopic --bootstrap-server 127.0.0.1:9092
```

- Read the event

```
!./kafka_2.13-3.3.1/bin/kafka-console-consumer.sh --topic TestTopic --from-beginning --bootstrap-server 127.0.0.1:9092
```

# Run Kafka on Colab

- You can run cells sequentially and get the result (not really streaming)

```
✓ [12] !./kafka_2.13-3.3.1/bin/kafka-console-producer.sh --topic TestTopic --bootstrap-server 127.0.0.1:9092
```

```
4m  
>hello  
>second line  
>
```

```
✓ [13] !./kafka_2.13-3.3.1/bin/kafka-console-consumer.sh --topic TestTopic --from-beginning --bootstrap-server 127.0.0.1:9092
```

```
9s  
test input from kafka producer  
hello  
second line
```

# Run Kafka on Colab

- Or you can run producer and consumer parallelly in different terminals

```
✓ [15] !pip install colab-xterm  
      %load_ext colabxterm
```

- Open terminal using Xterm and run consumer (it will be empty at first)

```
✓ [21] %xterm  
  
Launching Xterm...  
  
/content# ./kafka_2.13-3.3.1/bin/kafka-console-consumer.sh --topic TestTopic --from-beginning --bootstrap-server 127.0.0.1:9092  
second line in terminal  
third line in terminal  
fourth line in terminal  
fifth line in terminal  
[]
```

- Open terminal using Xterm and run producer, write some lines and they will appear on the consumer's terminal

```
✓ [20] %xterm  
  
Launching Xterm...  
  
/content# ./kafka_2.13-3.3.1/bin/kafka-console-producer.sh --topic TestTopic --bootstrap-server 127.0.0.1:9092  
>second line in terminal  
>third line in terminal  
>fourth line in terminal  
>fifth line in terminal  
>[]
```

# Run Kafka on Colab

- Use kafka-python on Colab

```
[30] !pip install kafka-python
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: kafka-python in /usr/local/lib/python3.7/dist-packages (2.0.2)

[27] from kafka import KafkaProducer

topic_name = 'TestTopic'
kafka_server = 'localhost:9092'

producer = KafkaProducer(bootstrap_servers=kafka_server)

producer.send(topic_name, b'Hello Kafka Python!!!')
producer.flush()

from kafka import KafkaConsumer

topic_name = 'TestTopic'
consumer = KafkaConsumer(topic_name,bootstrap_servers='127.0.0.1:9092',auto_offset_reset='earliest')
print(consumer)
for message in consumer:
    print(message)
```



The image shows a Jupyter Notebook interface on Google Colab. At the top, there's a toolbar with icons for file operations like up, down, back, forward, and search. Below the toolbar, the code cell [30] shows the command to install kafka-python and its output indicating the requirement is already satisfied. The code cell [27] contains Python code to produce a message to a Kafka topic and then consume it. The output of this cell shows the produced message and the consumed message, which is identical to the produced one. The code cell for the consumer has a play button icon to its left.

```
WARNING:kafka.coordinator.consumer:group_id is None: disabling auto-commit.
<kafka.consumer.group.KafkaConsumer object at 0x7fb44a7e2350>
ConsumerRecord(topic='TestTopic', partition=0, offset=0, timestamp=1664896978186, timestamp_type=0, key=None, value=b'ww', headers=[], checksum=1
ConsumerRecord(topic='TestTopic', partition=0, offset=1, timestamp=1664896985895, timestamp_type=0, key=None, value=b'cc', headers=[], checksum=1
- - - - -
```

# Tutorial 5: Test Kafka and Spark Structure Streaming on Colab

- Start kafka

```
[ ] !curl -ssLO https://downloads.apache.org/kafka/3.3.1/kafka_2.13-3.3.1.tgz  
!tar -xzf kafka_2.13-3.3.1.tgz  
  
[ ] !./kafka_2.13-3.3.1/bin/zookeeper-server-start.sh -daemon ./kafka_2.13-3.3.1/config/zookeeper.properties  
!./kafka_2.13-3.3.1/bin/kafka-server-start.sh -daemon ./kafka_2.13-3.3.1/config/server.properties
```

- Install PySpark

```
#currently, 3.3.0 is the latest version. However, you still need to specify this.  
!pip install pyspark==3.3.0  
  
from pyspark.sql import SparkSession  
scala_version = '2.13'  
spark_version = '3.3.0'  
packages = [ f'org.apache.spark:spark-sql-kafka-0-10_{scala_version}:{spark_version}' , 'org.apache.kafka:kafka-clients:3.3.1' ]  
spark = SparkSession.builder.master("local").appName("kafka-example").config("spark.jars.packages", ",".join(packages)).getOrCreate()  
spark
```

- Install kafka-python

```
!pip install kafka-python
```

```
from kafka import KafkaProducer
from json import dumps
topic_name = 'Number'
kafka_server = 'localhost:9092'
producer = KafkaProducer(bootstrap_servers=kafka_server,value_serializer = lambda x:dumps(x).encode('utf-8'))
for e in range(1000):
    data = {'number' : e}
    producer.send(topic_name, value=data)
producer.flush()
```

- You can test if the topic is sent sucessfully

```
[15] !./kafka_2.13-3.3.1/bin/kafka-console-consumer.sh --topic Number --from-beginning --bootstrap-server 127.0.0.1:9092
{"number": 937}
{"number": 938}
{"number": 939}
{"number": 940}
{"number": 941}
{"number": 942}
{"number": 943}
```

- Create datafram from Kafka topic

```
producer.flush()  
  
kafkaDf = spark.read.format("kafka") \  
    .option("kafka.bootstrap.servers", "localhost:9092") \  
    .option("subscribe", topic_name) \  
    .option("startingOffsets", "earliest") \  
    .load()  
  
kafkaDf.show()
```

- Show the dataframe in a formatted way

```
from pyspark.sql.functions import col, concat, lit  
  
kafkaDf.select(  
    concat(col("topic"), lit(':'),  
    col("partition").cast("string")).alias("topic_partition"), col("offset"), col("value").cast("string"))  
    .show()
```

topic_partition	offset	value
Number:0	0	{"number": 0}
Number:0	1	{"number": 1}
Number:0	2	{"number": 2}
Number:0	3	{"number": 3}
Number:0	4	{"number": 4}
Number:0	5	{"number": 5}
Number:0	6	{"number": 6}

# Tutorial 6: Test Kafka and Spark Structure Streaming on Local

- Step1: Start Kafka cluster using Terminal
- Step 2: Run KafkaProducer in Jupyter Notebook

```
from kafka import KafkaProducer
from json import dumps
from time import sleep

topic_name = 'RandomNumber'
kafka_server = 'localhost:9092'

producer = KafkaProducer(bootstrap_servers=kafka_server,value_serializer = lambda x:dumps(x).encode('utf-8'))

for e in range(1000):
    data = {'number' : e}
    producer.send(topic_name, value=data)
    print(str(data) + " sent")
    sleep(5)

producer.flush()
```

- Open another Jupyter Notebook

```
[1]: import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession

scala_version = '2.12' # your scala version
spark_version = '3.0.1' # your spark version
packages = [
    f'org.apache.spark:spark-sql-kafka-0-10_{scala_version}:{spark_version}',
    'org.apache.kafka:kafka-clients:2.8.0' #your kafka version
]
spark = SparkSession.builder.master("local").appName("kafka-example").config("spark.jars.packages", ",".join(packages)).getOrCreate()
spark
```

[1]: **SparkSession - in-memory**

**SparkContext**

[Spark UI](#)

**Version** v3.0.1

**Master** local

**AppName** kafka-example

- You will reading data from Kafka in two ways:
  - Batch query
  - Streaming query
- See more at <https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html>

# Creating a Kafka Source for Batch Queries

- Create dataframe from Kafka data

```
topic_name = 'RandomNumber'  
  
kafka_server = 'localhost:9092'  
  
kafkaDf = spark.read.format("kafka").option("kafka.bootstrap.servers", kafka_server).option("subscribe", topic_name).option("startingOffsets", "earliest").load()
```

- Show data (converting dataframe to pandas for cleaner view of data)

	key	value	topic	partition	offset	timestamp	timestampType
0	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	0	2022-10-05 15:18:37.301	0
1	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	1	2022-10-05 15:18:42.314	0
2	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	2	2022-10-05 15:18:47.327	0
3	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	3	2022-10-05 15:18:52.341	0
4	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	4	2022-10-05 15:18:57.352	0
5	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	5	2022-10-05 15:19:02.363	0
6	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	6	2022-10-05 15:19:07.376	0
7	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	7	2022-10-05 15:19:12.395	0
8	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	8	2022-10-05 15:19:17.411	0
9	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	9	2022-10-05 15:19:22.417	0

- Show streaming data using for loop

```
batchDF = kafkaDF.select(col('topic'), col('offset'), col('value').cast('string').substr(12,1).alias('rand_number'))  
from time import sleep  
from IPython.display import display, clear_output  
  
for x in range(0, 2000):  
  
    try:  
  
        print("Showing live view refreshed every 5 seconds")  
  
        print(f"Seconds passed: {x*5}")  
  
        display(batchDF.toPandas())  
  
        sleep(5)  
  
        clear_output(wait=True)  
  
    except KeyboardInterrupt:  
  
        print("break")  
  
        break  
  
print("Live view ended...")
```

Showing live view refreshed every 5 seconds  
Seconds passed: 10

	topic	offset	rand_number
0	RandomNumber	0	2
1	RandomNumber	1	4
2	RandomNumber	2	7
3	RandomNumber	3	7
4	RandomNumber	4	3
5	RandomNumber	5	7
6	RandomNumber	6	6

- Perform some data aggregation and show live results

```
batchCountDF = batchDF.groupBy('rand_number').count()
for x in range(0, 2000):
    try:
        print("Showing live view refreshed every 5 seconds")
        print(f"Seconds passed: {x*5}")
        display(batchCountDF.toPandas())
        sleep(5)
        clear_output(wait=True)
    except KeyboardInterrupt:
        print("break")
        break
print("Live view ended...")
```

Showing live view refreshed every 5 seconds  
Seconds passed: 5

	rand_number	count
0	7	5
1	3	2
2	8	1
3	0	1
4	5	1
5	6	3
6	9	1
7	1	1
8	4	1
9	2	1

break  
Live view ended...

# Creating a Kafka Source for Streaming Queries

- Create Streaming dataframe from Kafka

```
streamRawDf = spark.readStream.format("kafka").option("kafka.bootstrap.servers", kafka_server).option("subscribe", topic_name).load()  
streamDF = streamRawDf.select(col('topic'), col('offset'), col('value').cast('string').substr(12,1).alias('rand_number'))  
checkEvenDF = streamDF.withColumn('Is_Even', col('rand_number').cast('int') % 2 == 0 )
```

- Write stream

```
from random import randint  
randNum=str(randint(0,10000))  
q1name = "queryNumber"+randNum  
q2name = "queryCheckEven"+randNum  
  
stream_writer1 = (streamDF.writeStream.queryName(q1name).trigger(processingTime="5 seconds").outputMode("append").format("memory"))  
stream_writer2 = (checkEvenDF.writeStream.queryName(q2name).trigger(processingTime="5 seconds").outputMode("append").format("memory"))  
  
query1 = stream_writer1.start()  
query2 = stream_writer2.start()
```

- View streaming result

```
for x in range(0, 2000):
    try:
        print("Showing live view refreshed every 5 seconds")
        print(f"Seconds passed: {x*5}")
        result1 = spark.sql(f"SELECT * from {query1.name}")
        result2 = spark.sql(f"SELECT * from {query2.name}")
        display(result1.toPandas())
        display(result2.toPandas())
        sleep(5)
        clear_output(wait=True)
    except KeyboardInterrupt:
        print("break")
        break
print("Live view ended...")
```

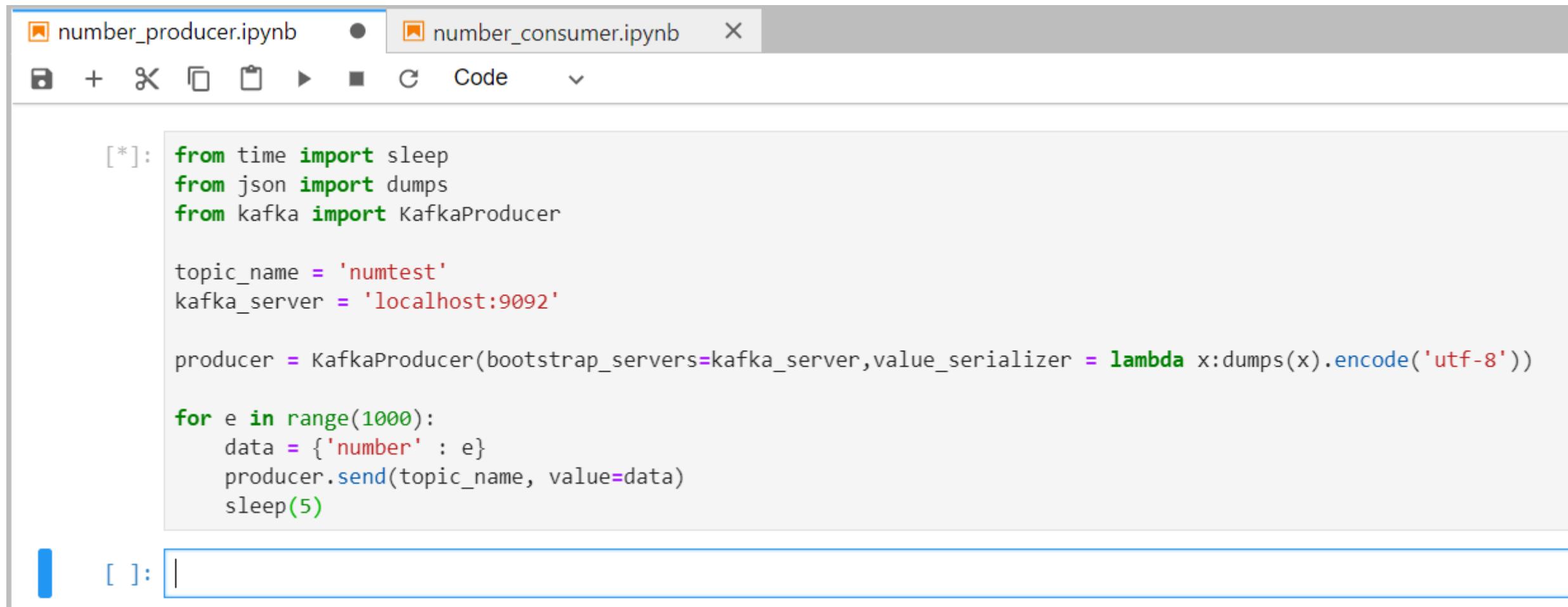
Showing live view refreshed every 5 seconds  
Seconds passed: 20

	topic	offset	rand_number
0	RandomNumber	21	0
1	RandomNumber	22	8
2	RandomNumber	23	2
3	RandomNumber	24	1

	topic	offset	rand_number	Is_Even
0	RandomNumber	21	0	True
1	RandomNumber	22	8	True
2	RandomNumber	23	2	True
3	RandomNumber	24	1	False

break  
Live view ended...

# Tutorial 7: Kafka and MongoDB on Window



The screenshot shows a Jupyter Notebook interface with two tabs at the top: 'number\_producer.ipynb' and 'number\_consumer.ipynb'. The 'number\_producer.ipynb' tab is active, indicated by a blue border around its title bar. Below the tabs is a toolbar with icons for file operations like new, open, save, and run, followed by a 'Code' dropdown menu.

The main content area displays a Python code cell (cell 1) containing the following code:

```
[*]: from time import sleep
from json import dumps
from kafka import KafkaProducer

topic_name = 'numtest'
kafka_server = 'localhost:9092'

producer = KafkaProducer(bootstrap_servers=kafka_server,value_serializer = lambda x:dumps(x).encode('utf-8'))

for e in range(1000):
    data = {'number' : e}
    producer.send(topic_name, value=data)
    sleep(5)
```

Below this cell is another cell indicator '[ ]:' with an empty input field.

```
[*]: from kafka import KafkaConsumer
      from pymongo import MongoClient
      from json import loads

      topic_name = 'numtest'
      consumer = KafkaConsumer(
          topic_name,
          bootstrap_servers=['localhost:9092'],
          auto_offset_reset='earliest',
          enable_auto_commit=True,
          group_id='my-group',
          value_deserializer=lambda x: loads(x.decode('utf-8')))

      client = MongoClient('localhost:27017')
      collection = client.numtest.numtest

      for message in consumer:
          message = message.value
          collection.insert_one(message)
          print('{} added to {}'.format(message, collection))
```

```
{'number': 0, '_id': ObjectId('61365919e8fecaa7b75f57e0')} added to Collection(Database(MongoClient(host=['localhost:27017']), document_class=dict, tz_aware=False, connect=True), 'numtest')
{'number': 1, '_id': ObjectId('61365919e8fecaa7b75f57e1')} added to Collection(Database(MongoClient(host=['localhost:27017']), document_class=dict, tz_aware=False, connect=True), 'numtest')
{'number': 2, '_id': ObjectId('61365919e8fecaa7b75f57e2')} added to Collection(Database(MongoClient(host=['localhost:27017']), document_class=dict, tz_aware=False, connect=True), 'numtest')
{'number': 3, '_id': ObjectId('61365919e8fecaa7b75f57e3')} added to Collection(Database(MongoClient(host=['localhost:27017']), document_class=dict, tz_aware=False, connect=True), 'numtest')
```



# Tutorial 8

<https://towardsdatascience.com/make-a-mock-real-time-stream-of-data-with-python-and-kafka-7e5e23123582>

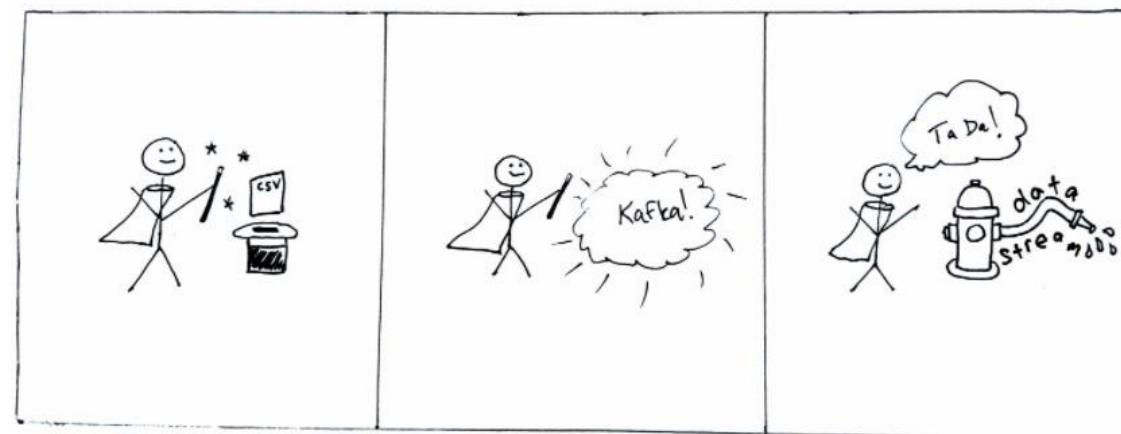
🔒 towardsdatascience.com/make-a-mock-real-time-stream-of-data-with-python-and-kafka-7e5e23123582

## Make a mock “real-time” data stream with Python and Kafka

A Dockerized tutorial with everything you need to turn a .csv file of timestamped data into a Kafka stream



Maria Patterson Aug 30 · 5 min read ★



# Tutorial 8: streaming from CSV

- sendStream.py

```
while True:

    try:

        if firstline is True:
            line1 = next(rdr, None)
            timestamp, value = line1[0], float(line1[1])
            # Convert csv columns to key value pair
            result = {}
            result[timestamp] = value
            # Convert dict to json as message format
            jresult = json.dumps(result)
            firstline = False

            producer.produce(topic, key=p_key, value=jresult, callback=acked)

        else:
            line = next(rdr, None)
            d1 = parse(timestamp)
            d2 = parse(line[0])
            diff = ((d2 - d1).total_seconds()) / args.speed
            time.sleep(diff)
            timestamp, value = line[0], float(line[1])
            result = {}
            result[timestamp] = value
            jresult = json.dumps(result)

            producer.produce(topic, key=p_key, value=jresult, callback=acked)

    producer.flush()
```

# Tutorial 8: streaming from CSV

- processStream.py

```
def main():
    parser = argparse.ArgumentParser(description=__doc__)
    parser.add_argument('topic', type=str, help='Name of the Kafka topic to stream.')

    args = parser.parse_args()

    conf = {'bootstrap.servers': 'localhost:9092', 'default.topic.config': {'auto.offset.reset': 'smallest'},
            'group.id': socket.gethostname()}

    consumer = Consumer(conf)

    running = True

    try:
        while running:
            consumer.subscribe([args.topic])

            msg = consumer.poll(1)
```

# Tutorial 8: streaming from CSV

Start the consumer

```
c:\ Command Prompt - CALL conda.bat activate base - python processStream.py my-stream  
(base) C:\Users\PC>cd PythonCodes\Kafka\csvstreaming  
  
(base) C:\Users\PC\PythonCodes\Kafka\csvstreaming>python processStream.py my-stream  
Topic unknown, creating my-stream topic
```

# Tutorial 8: streaming from CSV

Start the producer

```
cmd Command Prompt - CALL conda.bat activate base - python sendStream.py data.csv my-stream

(base) C:\Users\PC\PythonCodes\Kafka\csvstreaming>python sendStream.py data.csv my-stream
Message produced: b'{"2021-01-01 00:00:00": 51.0}'
Message produced: b'{"2021-01-01 00:00:04": 60.0}'
Message produced: b'{"2021-01-01 00:00:06": 82.0}'
Message produced: b'{"2021-01-01 00:00:07": 86.0}'
Message produced: b'{"2021-01-01 00:00:11": 99.0}'
Message produced: b'{"2021-01-01 00:00:12": 23.0}'
Message produced: b'{"2021-01-01 00:00:21": 63.0}'
```

```
cmd Command Prompt - CALL conda.bat activate base - python processStream.py my-stream

(base) C:\Users\PC>cd PythonCodes\Kafka\csvstreaming

(base) C:\Users\PC\PythonCodes\Kafka\csvstreaming>python processStream.py my-stream
Topic unknown, creating my-stream topic
2021-09-08 22:54:11 {'2021-01-01 00:00:00': 51.0}
2021-09-08 22:54:15 {'2021-01-01 00:00:04': 60.0}
2021-09-08 22:54:17 {'2021-01-01 00:00:06': 82.0}
2021-09-08 22:54:18 {'2021-01-01 00:00:07': 86.0}
2021-09-08 22:54:22 {'2021-01-01 00:00:11': 99.0}
2021-09-08 22:54:23 {'2021-01-01 00:00:12': 23.0}
2021-09-08 22:54:32 {'2021-01-01 00:00:21': 63.0}
```

# Tutorial 8: streaming from CSV

If you terminate the consumer and then restart it, the streaming will be resumed from where it stop

```
Command Prompt - CALL conda.bat activate base
Message produced: b'{"2021-01-01 00:00:39": 46.0}'
Message produced: b'{"2021-01-01 00:00:40": 61.0}'
Message produced: b'{"2021-01-01 00:00:41": 50.0}'
Message produced: b'{"2021-01-01 00:00:44": 2.0}'
Message produced: b'{"2021-01-01 00:00:47": 20.0}'
Message produced: b'{"2021-01-01 00:00:49": 38.0}'
```

```
Command Prompt - CALL conda.bat activate base
2021-09-08 22:58:46 {'2021-01-01 00:00:38': 61.0}
2021-09-08 22:58:47 {'2021-01-01 00:00:39': 46.0}
2021-09-08 22:58:48 {'2021-01-01 00:00:40': 61.0}
2021-09-08 22:58:49 {'2021-01-01 00:00:41': 50.0}

(base) C:\Users\PC\PythonCodes\Kafka\csvstreaming>python processStream.py my-stream
2021-09-08 22:58:57 {'2021-01-01 00:00:44': 2.0}
2021-09-08 22:58:58 {'2021-01-01 00:00:47': 20.0}
2021-09-08 22:58:58 {'2021-01-01 00:00:49': 38.0}
2021-09-08 22:59:00 {'2021-01-01 00:00:52': 88.0}
2021-09-08 22:59:01 {'2021-01-01 00:00:53': 59.0}
```

# Tutorial 9



[medium.com/@kevin.michael.horan/distributed-video-streaming-with-python-and-kafka-551de69fe1dd](https://medium.com/@kevin.michael.horan/distributed-video-streaming-with-python-and-kafka-551de69fe1dd)

**Kevin Horan**

89 Followers

About

Follow



...



## Distributed Video Streaming with Python and Kafka



Kevin Horan Mar 26, 2018 · 9 min read



...

<https://medium.com/@kevin.michael.horan/distributed-video-streaming-with-python-and-kafka-551de69fe1dd>

# Tutorial 9: Video streaming using Kafka

- Producer.py

```
def publish_video(video_file):

    # Start up producer
    producer = KafkaProducer(bootstrap_servers='localhost:9092')

    # Open file
    video = cv2.VideoCapture(video_file)

    while(video.isOpened()):
        success, frame = video.read()

        # Convert image to png
        ret, buffer = cv2.imencode('.jpg', frame)

        # Convert to bytes and send to kafka
        producer.send(topic, buffer.tobytes())

        time.sleep(0.2)
```

# Tutorial 9: Video streaming using Kafka

- Producer.py

```
def publish_camera():

    # Start up producer
    producer = KafkaProducer(bootstrap_servers='localhost:9092')

    camera = cv2.VideoCapture(0)
    try:
        while(True):
            success, frame = camera.read()

            ret, buffer = cv2.imencode('.jpg', frame)
            producer.send(topic, buffer.tobytes())

            time.sleep(0.2)

    except:
        print("\nExiting.")
        sys.exit(1)
```

# Tutorial 9: Video streaming using Kafka

- Producer.py

```
if __name__ == '__main__':
    """
    Producer will publish to Kafka Server a video file given as a system arg.
    Otherwise it will default by streaming webcam feed.
    """
    if(len(sys.argv) > 1):
        video_path = sys.argv[1]
        publish_video(video_path)
    else:
        print("publishing feed!")
        publish_camera()
```

---

# Tutorial 9: Video streaming using Kafka

- consumer.py

```
# Fire up the Kafka Consumer
topic = "distributed-video1"

consumer = KafkaConsumer(topic,bootstrap_servers=['localhost:9092'])

# Set the consumer in a Flask App
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/video_feed', methods=['GET'])
def video_feed():
    return Response( get_video_stream(), mimetype='multipart/x-mixed-replace; boundary=frame')

def get_video_stream():
    for msg in consumer:
        yield (b"--frame\r\n" b'Content-Type: image/jpg\r\n\r\n' + msg.value + b'\r\n\r\n')

if __name__ == '__main__':
    app.run(debug=True)
```

# Tutorial 9: Video streaming using Kafka

- Run consumer.py

```
(base) C:\Users\PC\PythonCodes\Kafka\videostreaming>python consumer.py
* Serving Flask app "consumer" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 244-614-898
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



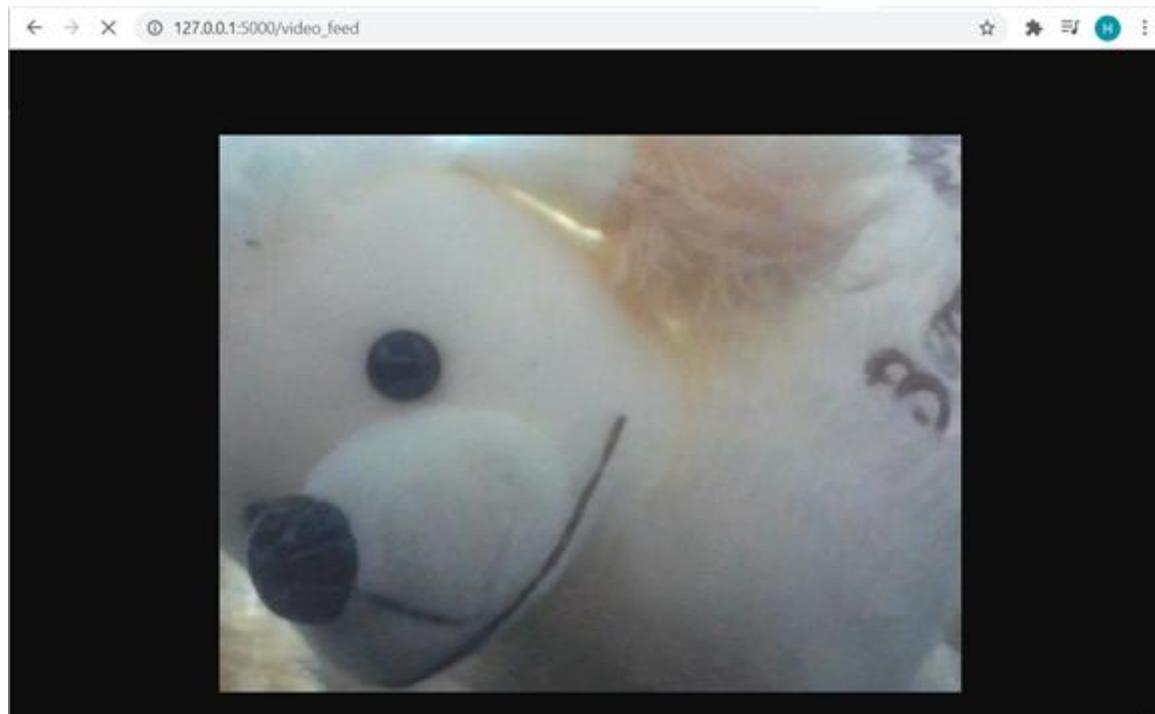
127.0.0.1:5000

## Video Streaming Demonstration

# Tutorial 9: Video streaming using Kafka

- Stream video from webcam

```
(base) C:\Users\PC\PythonCodes\Kafka\videostreaming>python producer.py  
publishing feed!
```



# Tutorial 9: Video streaming using Kafka

- Stream a video entitled Countdown1.mp4

```
(base) C:\Users\PC\PythonCodes\Kafka\videostreaming>python producer.py videos/Countdown1.mp4  
publishing video...
```



# Tutorial 10

<https://towardsdatascience.com/real-time-anomaly-detection-with-apache-kafka-and-python-3a40281c01c9>

## Real-time anomaly detection with Apache Kafka and Python

Learn how to make predictions over streaming data coming from Kafka using Python.



Rodrigo Arenas Jun 18 · 5 min read



# Tutorial 10: real-time anomaly detection

```
C:\kafka>.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic transaction
Created topic transaction.

C:\kafka>.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic anomalies
Created topic anomalies.
```

# Tutorial 10: real-time anomaly detection

- Producer.py

```
if producer is not None:
    while True:
        # Generate some abnormal observations
        if random.random() <= OUTLIERS_GENERATION_PROBABILITY:
            X_test = np.random.uniform(low=-4, high=4, size=(1, 2))
        else:
            X = 0.3 * np.random.randn(1, 2)
            X_test = (X + np.random.choice(a=[2, -2], size=1, p=[0.5, 0.5]))

        X_test = np.round(X_test, 3).tolist()

        current_time = datetime.utcnow().isoformat()

        record = {"id": _id, "data": X_test, "current_time": current_time}
        record = json.dumps(record).encode("utf-8")
        print('produce message')
        print(record)
        producer.produce(topic=TRANSACTIONS_TOPIC,
                          value=record)
        producer.flush()
        _id += 1
        time.sleep(Delay)
```

# Tutorial 10: real-time anomaly detection

- train.py

```
import random
from joblib import dump

import numpy as np
from sklearn.ensemble import IsolationForest

rng = np.random.RandomState(42)

# Generate train data
X = 0.3 * rng.randn(500, 2)
X_train = np.r_[X + 2, X - 2]
X_train = np.round(X_train, 3)

# fit the model
clf = IsolationForest(n_estimators=50, max_samples=500, random_state=rng, contamination=0.01)
clf.fit(X_train)

dump(clf, './isolation_forest.joblib')
print('finished training')
```

# Tutorial 10: real-time anomaly detection

- detector.py

```
while True:
    message = consumer.poll(timeout=50)
    if message is None:
        continue
    if message.error():
        logging.error("Consumer error: {}".format(message.error()))
        continue

    # Message that came from producer
    record = json.loads(message.value().decode('utf-8'))
    data = record["data"]
    print(data)
    prediction = clf.predict(data)
    if prediction[0] == 1:
        print('Normal')

    # If an anomaly comes in, send it to anomalies topic
    if prediction[0] == -1:
        print('Abnormal')
        score = clf.score_samples(data)
        record["score"] = np.round(score, 3).tolist()

        _id = str(record["id"])
        record = json.dumps(record).encode("utf-8")

        producer.produce(topic=ANOMALIES_TOPIC,
                          value=record)
        producer.flush()
        print(record)
        print('Alert sent!')
# consumer.commit() # Uncomment to process all messages, not just new ones
```

# Tutorial 10: real-time anomaly detection

```
(base) C:\Users\PC\PythonCodes\Kafka\abnomaldetection>python train.py  
finished training
```

# Tutorial 10: real-time anomaly detection

```
(base) C:\Users\PC\PythonCodes\Kafka\abnomalddetection>python producer.py
produce message
b'{"id": 0, "data": [[2.098, 2.606]], "current_time": "2021-09-09T15:03:21.134716"}'
produce message
b'{"id": 1, "data": [[2.159, 2.785]], "current_time": "2021-09-09T15:03:22.177714"}'
produce message
b'{"id": 2, "data": [[-1.014, 3.463]], "current_time": "2021-09-09T15:03:23.204923"}'
produce message
b'{"id": 3, "data": [[2.102, -3.905]], "current_time": "2021-09-09T15:03:24.223482"}'
produce message
b'{"id": 4, "data": [[1.2, 1.708]], "current_time": "2021-09-09T15:03:25.252457"}'
produce message
```

# Tutorial 10: real-time anomaly detection

```
C:\kafka>.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic transactions
{"id": 42, "data": [[2.149, 2.192]], "current_time": "2021-09-09T15:04:04.194174"}
{"id": 43, "data": [[-1.713, -2.004]], "current_time": "2021-09-09T15:04:05.213219"}
{"id": 44, "data": [[2.128, 2.349]], "current_time": "2021-09-09T15:04:06.244390"}
{"id": 45, "data": [[-2.026, -2.136]], "current_time": "2021-09-09T15:04:07.266721"}
{"id": 46, "data": [[1.473, 1.82]], "current_time": "2021-09-09T15:04:08.293584"}
 {"id": 47, "data": [[-1.583, -1.949]], "current_time": "2021-09-09T15:04:09.325670"}
 {"id": 48, "data": [[1.394, -1.956]], "current_time": "2021-09-09T15:04:10.354479"}
 {"id": 49, "data": [[-2.348, -1.901]], "current_time": "2021-09-09T15:04:11.384494"}
```

# Tutorial 10: real-time anomaly detection

```
(base) C:\Users\PC\PythonCodes\Kafka\abnomalddetection>python detector.py  
[[-1.915, 1.71]]  
Abnormal  
b'{"id": 725, "data": [[-1.915, 1.71]], "current_time": "2021-09-09T14:01:07.985942", "score": [-0.711]}'  
Alert sent  
[[2.516, 1.881]]  
Normal  
[[-1.472, -2.328]]  
Normal  
[[2.597, 1.787]]  
Normal
```

# Tutorial 10: real-time anomaly detection

```
C:\kafka>.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic anomalies
{"id": 181, "data": [[-2.883, -3.107]], "current_time": "2021-09-09T15:06:26.751928", "score": [-0.733]}
 {"id": 184, "data": [[-0.804, 1.628]], "current_time": "2021-09-09T15:06:29.830897", "score": [-0.699]}
 {"id": 192, "data": [[-0.899, -3.483]], "current_time": "2021-09-09T15:06:38.020883", "score": [-0.711]}
```

# Tutorial 11: Tensorflow-IO and Kafka

<https://www.tensorflow.org/io/tutorials/kafka>

TensorFlow > Resources > TensorFlow I/O > Guide & Tutorials

Was this helpful?  

## Robust machine learning on streaming data using Kafka and Tensorflow-IO

### Overview

This tutorial focuses on streaming data from a [Kafka](#) cluster into a `tf.data.Dataset` which is then used in conjunction with `tf.keras` for training and inference.

Kafka is primarily a distributed event-streaming platform which provides scalable and fault-tolerant streaming data across data pipelines. It is an essential technical component of a plethora of major enterprises where mission-critical data delivery is a primary requirement.

```
!pip install tensorflow-io==0.25.0  
!pip install kafka-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/  
Requirement already satisfied: tensorflow-io==0.25.0 in /usr/local/lib/python3.7/dist-packages (0.25.0)  
Requirement already satisfied: tensorflow-io-gcs-filesystem==0.25.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow-io==0.25.0) (0.25.0)  
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/  
Requirement already satisfied: kafka-python in /usr/local/lib/python3.7/dist-packages (2.0.2)
```

## Import packages

```
[ ] import os  
from datetime import datetime  
import time  
import threading  
import json  
from kafka import KafkaProducer  
from kafka.errors import KafkaError  
from sklearn.model_selection import train_test_split  
import pandas as pd  
import tensorflow as tf  
import tensorflow_io as tfio
```

## Download and setup Kafka and Zookeeper instances

For demo purposes, the following instances are setup locally:

- Kafka (Brokers: 127.0.0.1:9092)
- Zookeeper (Node: 127.0.0.1:2181)

```
[ ] !curl -sSOL https://downloads.apache.org/kafka/3.3.1/kafka\_2.13-3.3.1.tgz
!tar -xzf kafka_2.13-3.3.1.tgz
```

Using the default configurations (provided by Apache Kafka) for spinning up the instances.

```
[ ] !./kafka_2.13-3.3.1/bin/zookeeper-server-start.sh -daemon ./kafka_2.13-3.3.1/config/zookeeper.properties
!./kafka_2.13-3.3.1/bin/kafka-server-start.sh -daemon ./kafka_2.13-3.3.1/config/server.properties
!echo "Waiting for 10 secs until kafka and zookeeper services are up and running"
!sleep 10
```

```
Waiting for 10 secs until kafka and zookeeper services are up and running
```

Create the kafka topics with the following specs:

- susy-train: partitions=1, replication-factor=1
- susy-test: partitions=2, replication-factor=1

```
[ ] !./kafka_2.13-3.3.1/bin/kafka-topics.sh --create --bootstrap-server 127.0.0.1:9092 --replication-factor 1 --partitions 1 --topic susy-train  
!./kafka_2.13-3.3.1/bin/kafka-topics.sh --create --bootstrap-server 127.0.0.1:9092 --replication-factor 1 --partitions 2 --topic susy-test
```

```
Error while executing topic command : Topic 'susy-train' already exists.  
[2022-10-04 15:41:03,437] ERROR org.apache.kafka.common.errors.TopicExistsException: Topic 'susy-train' already exists.  
(kafka.admin.TopicCommand$)  
Error while executing topic command : Topic 'susy-test' already exists.  
[2022-10-04 15:41:05,951] ERROR org.apache.kafka.common.errors.TopicExistsException: Topic 'susy-test' already exists.  
(kafka.admin.TopicCommand$)
```

- Just follow <https://www.tensorflow.org/io/tutorials/kafka>

Evaluate the performance of the batch training model on the test data

```
[ ] res = model.evaluate(test_ds)
print("test loss, test acc:", res)

1250/1250 [=====] - 16s 12ms/step - loss: 0.4328 - accuracy: 0.7994
test loss, test acc: [0.43278080224990845, 0.7993500232696533]
```

```
[ ] def decode_kafka_online_item(raw_message, raw_key):
    message = tf.io.decode_csv(raw_message, [[0.0] for i in range(NUM_COLUMNS)])
    key = tf.strings.to_number(raw_key)
    return (message, key)

for mini_ds in online_train_ds:
    mini_ds = mini_ds.shuffle(buffer_size=32)
    mini_ds = mini_ds.map(decode_kafka_online_item)
    mini_ds = mini_ds.batch(32)
    if len(mini_ds) > 0:
        model.fit(mini_ds, epochs=3)
```

```
Epoch 1/3
313/313 [=====] - 1s 4ms/step - loss: 0.4366 - accuracy: 0.7990
Epoch 2/3
313/313 [=====] - 2s 6ms/step - loss: 0.4329 - accuracy: 0.8000
Epoch 3/3
313/313 [=====] - 1s 4ms/step - loss: 0.4305 - accuracy: 0.8023
```

<https://www.analyticsvidhya.com/blog/2021/06/spotify-recommendation-system-using-pyspark-and-kafka-streaming/>

# Spotify Recommendation System using Pyspark and Kafka streaming

Siddharth1698 — June 23, 2021

Advanced Data Engineering Python

<https://github.com/rongpenl/order-book-simulation>

## Order book simulation with Kafka and Streamlit

---

- Order book simulation with Kafka and Streamlit
  - Part 1 Simulation with Kafka and Streamlit
  - Part 2 MLOps on your local machines and AWS

<https://aiven.io/blog/create-your-own-data-stream-for-kafka-with-python-and-faker>

The screenshot shows a web browser displaying a blog post from the Aiven website. The URL in the address bar is <https://aiven.io/blog/create-your-own-data-stream-for-kafka-with-python-and-faker>. The page features the Aiven logo and navigation links for Products, Pricing, Solutions, Company, Case Studies, and Blog. A prominent red "Get Started" button is visible. The main title of the post is "Create your own data stream for Kafka with Python and Faker". Below the title is a summary text: "How can you test an empty data pipeline? Well, you can't, really. Read on and let Aiven's Developer Advocate Francesco Tisiot walk you through creating pretend streaming data using Python and Faker." At the bottom, it says "10 FEBRUARY 2021" and includes a photo of the author, Francesco Tisiot, and a link to his RSS feed.

aiven

Products ▾ Pricing Solutions ▾ Company ▾ Case Studies Blog Get Started

# Create your own data stream for Kafka with Python and Faker

How can you test an empty data pipeline? Well, you can't, really. Read on and let Aiven's Developer Advocate Francesco Tisiot walk you through creating pretend streaming data using Python and Faker.

10 FEBRUARY 2021

 Francesco Tisiot | [Francesco Tisiot RSS Feed](#)  
Developer Advocate at Aiven