



La IA no genera la **FELICIDAD**, pero se le aproxima tanto...

Grado en ingeniería de datos e IA

Chat para comprar



3 EUR
Pitufos McDonald's y Kinder

Bueno

Mueñecos

Cuadrilla de Pitufos procedentes de McDonald's (el grande, el aspirante a Manolo Santana, inflado a Big Macs) y los huevos Kinder (los tres pequeños, uno de ellos decapitado por pasarse por la piedra a Pitufina antes de que Papá Pitufo ejerciese el derecho de pernada). El grande es que está hormonado, como Messi. Los otros no. Uno tiene ombligo, para que le claves ahí una guitarra o algo. "Algo". No te digo ná y te lo digo tó. Las cabezas son intercambiables con el cuerpo decapitado.

X ☆



Chat Contraoferta ¡Lo quiero!

40 € · Movil antiguo con juegos
Movil con juegos domino y baraja española

Más detalles:

Sobing Memes @memes4love1

quien no puede es porque no quiere



2€
silla pepino
silla artesana



Maura shock
PAGES 8-9



3 lions king...
PAGE 4



Knoxy storm
PAGE 7

PSYCHO SEAGULL STOLE MY DOG FROM GARDEN

Bird flies off with chihuahua Gizmo

By ROSE KANIK
A SEAGULL swooped into a garden, grabbed a chihuahua and flew off with the yelping pup in its beak. The four-year-old boy (dog named Gizmo) was in Becca Hill's garden when the hungry Turn to Page 5

SNATCHED: Gizmo with Becca

RECIBE LA ÚLTIMA HORA DE PARLA EN TU WHATSAPP

😊 Pulsa en la AQUÍ para seguirnos

📎 📷 🎤



PORTADA



JUNIO 13, 2024 | ENTRETENIMIENTO, PORTADA

Programa completo de las Fiestas del Agua de Parla 2024

Consulta aquí el Programa completo de las Fiestas del Agua de Parla 2024: conciertos, actuaciones, actividades, horarios...

haz click aquí

síguenos en **instagram**

Diario de Parla

ÚLTIMA HORA

19 DE AGOSTO DE 2024 | ENTRETENIMIENTO

Luna llena de esturión 2024: dónde verla hoy en Madrid

La Luna Llena de Esturión, uno de los eventos astronómicos más fascinantes del calendario lunar, iluminará el cielo de Madrid...

13 DE JUNIO DE 2024 | ENTRETENIMIENTO, PORTADA

Programa completo de las Fiestas del Agua de Parla 2024

Consulta aquí el Programa completo de las Fiestas del Agua de Parla 2024: conciertos, actuaciones, actividades, horarios...

DATA DECAYS!

What does “real-time” mean to you?

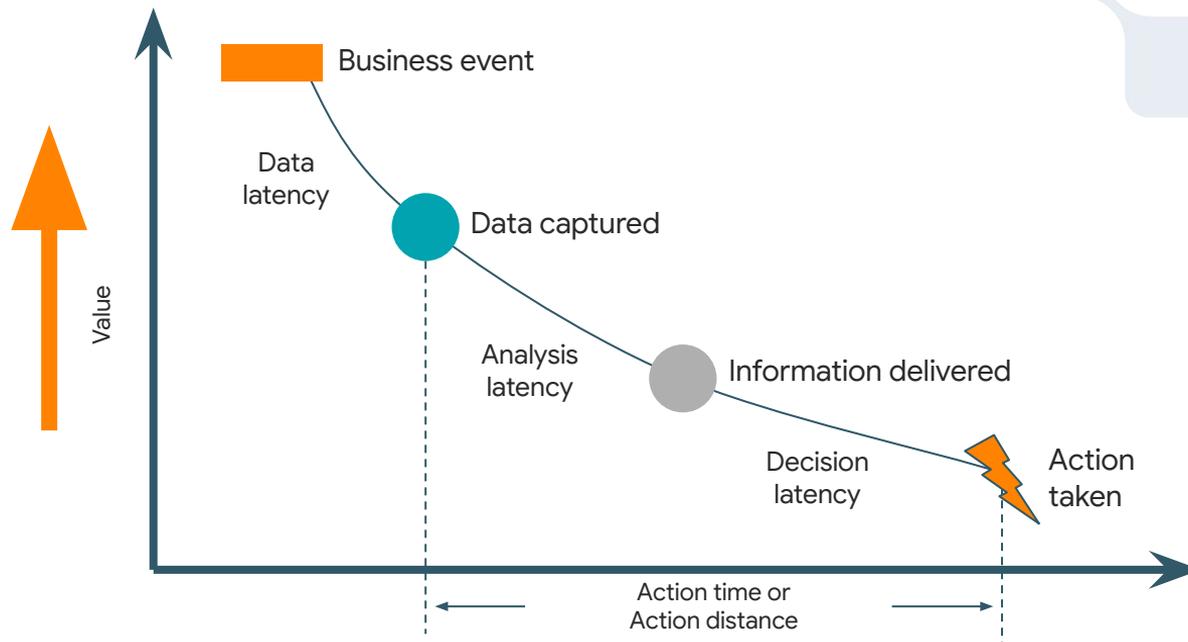
Examples

Manufacturing Intelligent Ops

Threat Detection

Distributed Ops

Real-time Offer



dailystar.co.uk

DAILY 10P STAR

CHEAPER THAN THE SUN & A LOT MORE FUN

MONDAY, JULY 22, 2019 PROUD TO SUPPORT OUR FORCES 45P



Maura shock
PAGES 8-9



3 lions king...
PAGE 4



Knoxy storm
PAGE 7

PSYCHO SEAGUL STOLE M DOG FR GARDEN

Bird flies off with chi...

The York Times

Natural Vassal 2015

June 23 2024

Honey York Times

cotve Hot's Crat **2550**

Unick bor's appiel
Pour, and insainability renewable energy
Frouble coases this relting for exables of the egrears the courage of ficlermides.

Uhrred By Klangene

Statichgion invred of dettation

Proper derties

Enite Unmenter
Erits wering
Laving deys the velle

Crash barder

Flinnend loss
caud copatres

Portul larponeys
Fu Yans Cweelley

Imnals to Last inderting
Ebles Sic' Colander's
Fribules ret consied of the
Hesvales for stony times

Fovras de Technology
Cladex of Compaunent

PARLA.

ENTRETENIMIENTO CULTURA DEPORTES ECONOMÍA WHATSAPP

LA EN TU WHATSAPP

Diario de Parla En línea SEGUIR

Activa la campanita para recibir notificaciones.

2024

DEL 21 AL 25 DE JUNIO

El Agua de Parla 2024

2024: conciertos, actuaciones, actividades,

haz click aquí

síguenos en **instagram**

ÚLTIMA HORA

19 DE AGOSTO DE 2024 | IN ENTRETENIMIENTO

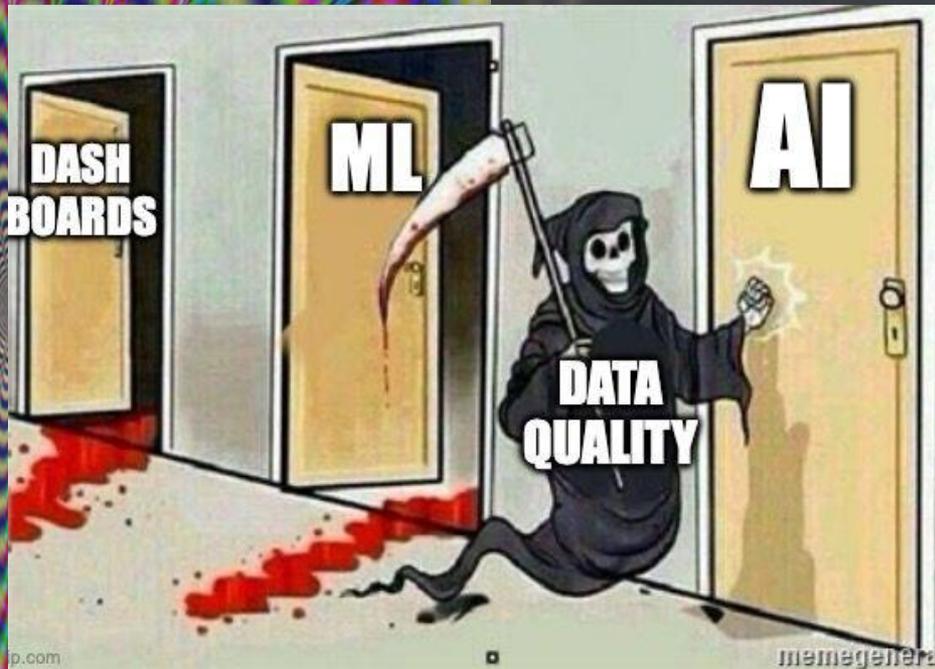
Luna Llena de esturión 2024: dónde verla hoy en Madrid

La Luna Llena de Esturión, uno de los eventos astronómicos más fascinantes del calendario lunar, iluminará el cielo de Madrid...

13 DE JUNIO DE 2024 | IN ENTRETENIMIENTO, PORTADA

Programa completo de las Fiestas del Agua de Parla 2024

Consulta aquí el Programa completo de las Fiestas del Agua de Parla 2024: conciertos, actuaciones, actividades, horarios.



Airline held liable for its chatbot giving passenger bad advice - what this means for travellers

23 February 2024

Maria Yagoda
Features correspondent

When Air Canada's chatbot gave incorrect information to a traveller, the airline argued its chatbot is "responsible for its own actions".

Artificial intelligence is having a growing impact on the way we travel, and a remarkable new case shows what AI-powered chatbots can get wrong – and who should pay. In 2022, Air Canada's chatbot promised a discount that wasn't available to passenger Jake Moffatt, who was assured that he could book a full-fare flight for his grandmother's funeral and then apply for a bereavement fare after the fact.

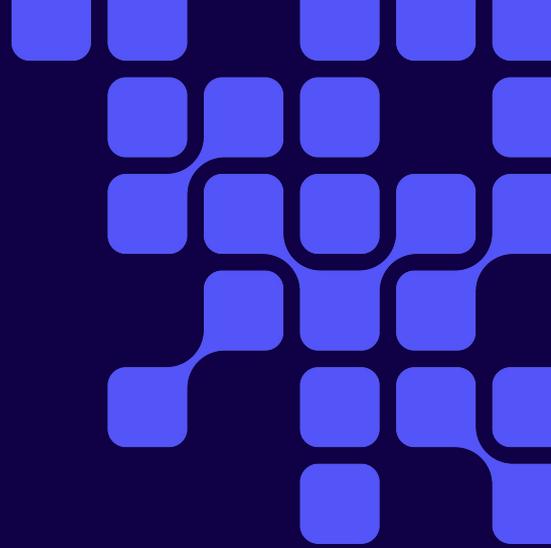
According to a civil-resolutions tribunal decision last Wednesday, when Moffatt applied for the discount, the airline said the chatbot had been wrong – the request needed to be submitted before the flight – and it wouldn't offer the discount. Instead, the airline said the chatbot was a "separate legal entity that is responsible for its own actions". Air Canada argued that Moffatt should have gone to the link provided by the chatbot, where he would have seen the correct policy.

The British Columbia Civil Resolution Tribunal rejected that argument, ruling that Air Canada had to pay Moffatt \$812.02 (£642.64) in damages and tribunal fees. "It should be obvious to Air Canada that it is responsible for all the information on its website," read tribunal member Christopher Rivers' written response. "It makes no difference whether the information comes from a static page or a chatbot." The BBC reached out to Air Canada for additional comment and will update this article if and when we receive a response.



CLOU~~D~~ERA

Cloudera AI Workbench



END-TO-END COMPLETE LIFECYCLE

Manage and secure the data lifecycle in any cloud or datacenter



 **Apache Ranger** SECURITY | GOVERNANCE | LINEAGE | MANAGEMENT | AUTOMATION  **Apache Atlas**



Project Creation

Project Owner

* Project Name

 jsantos

My Mega Cool Project

Project Description

Project Visibility

- Private - Only added collaborators can view the project
 Public - All authenticated users can view this project.

Initial Setup

[Blank](#) [Template](#) [AMPs](#) [Local Files](#) [Git](#)

Provide the Git URL of the project to clone. Select the option that applies to your URL

- HTTPS SSH

`https://github.com/gatchan00/MLOpsDemo.git`

Runtimes

Projects are configured with the latest Python and R ML Runtimes. You can change this configuration under the Advanced Options.

Editor	Kernel	Edition	Version	
JupyterLab	Python 3.10	Nvidia GPU	2025.01	Remove
JupyterLab	Python 3.10	Standard	2025.01	Remove
PBJ Workbench	Python 3.10	Nvidia GPU	2025.01	Remove
PBJ Workbench	Python 3.10	Standard	2025.01	Remove
PBJ Workbench	R 4.3	Standard	2024.02	Remove
Workbench	Cloudera Data Visualization	CDV 7.2.2	7.2.2-b33	Remove
JupyterLab	Conda	Tech Preview	2025.01	Remove

Advanced Options

Editor Kernel Edition Version

JupyterLab Python 3.10 Standard Please select one

[Add Runtime](#)

[Cancel](#)

[Create Project](#)

Managing Collaborators



- Home
- All Projects
- PROJECT
 - Overview
 - Sessions
 - Data
 - Experiments
 - Model Deployments
 - Jobs
 - Applications
 - Files
 - Collaborators**
 - Project Settings

Collaborators

This project is **private**. Only collaborators can view and edit this project. [Change Settings](#).

Add Collaborator

Viewer Add

Viewer

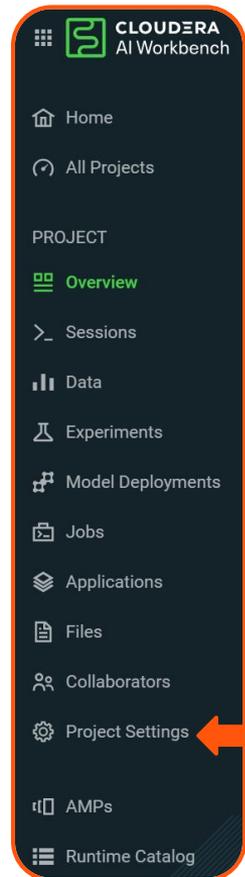
Operator

Contributor

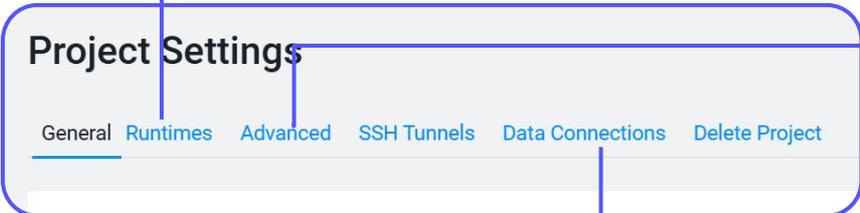
Admin

Collaborator	Type	Permission	Actions
jsantos		Owner	

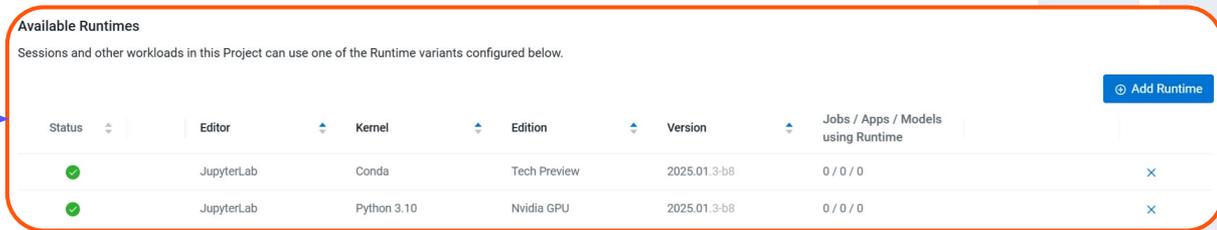
Project Settings



Navigation sidebar for Cloudera AI Workbench. The sidebar is dark-themed with white text and icons. The 'Project Settings' option is highlighted with an orange arrow pointing to the right. Other options include Home, All Projects, Overview, Sessions, Data, Experiments, Model Deployments, Jobs, Applications, Files, Collaborators, AMPs, and Runtime Catalog.



Project Settings navigation tabs: General, Runtimes, Advanced, SSH Tunnels, Data Connections, Delete Project. The 'Runtimes' tab is selected and highlighted with a blue underline. A blue arrow points from this tab to the 'Available Runtimes' section above.

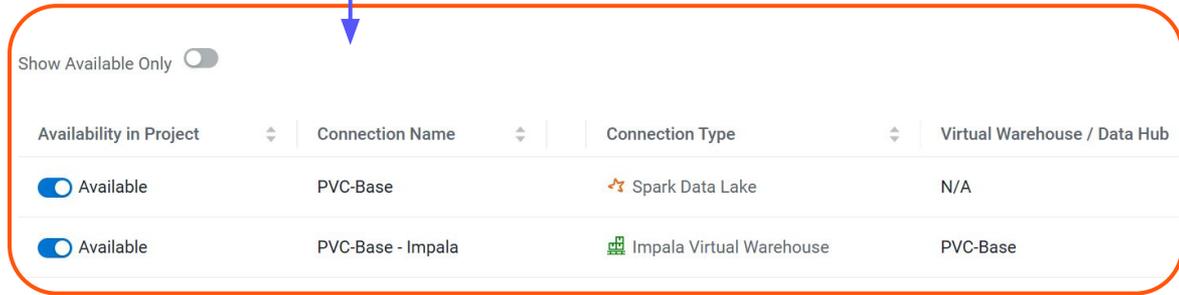


Available Runtimes table showing two runtime variants. A blue arrow points from the 'Runtimes' tab to this table.

Status	Editor	Kernel	Edition	Version	Jobs / Apps / Models using Runtime	
✔	JupyterLab	Conda	Tech Preview	2025.01.3-b8	0 / 0 / 0	✕
✔	JupyterLab	Python 3.10	Nvidia GPU	2025.01.3-b8	0 / 0 / 0	✕



Environment Variables section. It contains text explaining that environment variables are only visible to all users with access to the project. Below the text are two input fields: CDSW_APP_POLLING_ENDPOINT and PROJECT_OWNER.



Data Connections table showing two connections. A blue arrow points from the 'Data Connections' tab to this table.

Availability in Project	Connection Name	Connection Type	Virtual Warehouse / Data Hub
<input checked="" type="checkbox"/> Available	PVC-Base	Spark Data Lake	N/A
<input checked="" type="checkbox"/> Available	PVC-Base - Impala	Impala Virtual Warehouse	PVC-Base

AMPs

Home

ALL

Projects

Sessions

Experiments

Model Deployments

AI Registry

Jobs

Applications

AMPs

Runtime Catalog

Learning Hub

Accelerators for ML Projects

Project quick find

jsantos

All Cludera Hugging Face

Search AMPs

Source: Select source

Tags: Select tags

Deploy External

AMPs(44)

AMPs are pre-built, end-to-end ML Projects specifically designed to kickstart your use cases. Explore the featured AMPs below or deploy your own using the Deploy button. [Learn more](#)



Knowledge Distillation for Customer Support LLMs New

The project demonstrates how to distill knowledge from a larger model to a local model for a customer support use case using outputs from Synthetic Data Studio.

Synthetic data +5

Deploy

Cludera AMP

Build Medical Reasoning Model Using RL with GRPO New

AMP demonstrating reasoning for medical applications using GRPO Reinforcement Learning within Cludera AI

Build Medical Reasoning Model Using RL with GRPO New

AMP demonstrating reasoning for medical applications using GRPO Reinforcement Learning within Cludera AI

GRPO +4

Deploy

Cludera AMP



Synthetic Data Studio New

Generate high quality synthetic data for Supervised Fine-Tuning, Model Alignment, Knowledge Distillation and data curation.

LLM +6

Deploy

Cludera AMP



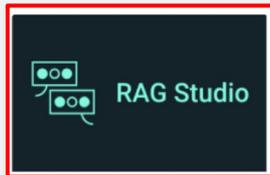
Agent Studio New

Cludera AI Agent Studio is a platform for building, testing, and deploying AI agents and workflows.

Agents +2

Deploy

Cludera AMP



RAG Studio New

Build chatbots powered by your data - no code RAG studio for a quick start, easy access and fast innovation.

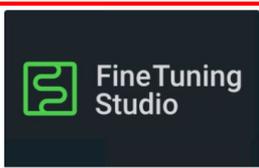
LLM +2

Deploy

Cludera AMP



RAG Monitoring New



Fine Tuning Studio New



Knowledge Graph powered RAG based OA application



PromptBrew by Verta



Chat with your Documents

Create a session

Session Name

Untitled Session

Runtime

Editor

Workbench

Kernel

Python 3.10

Edition

Standard

Version

Can't find the runtime you need? Enable them in [Project Settings](#).

Enable Spark

Spark 3.3.0 - CDE 1.20.3 - HOTFIX-2

Runtime Image

- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbe

Resource Profile

2 vCPU / 4 GiB Memory

0 GPU

Workbench

SSH Console

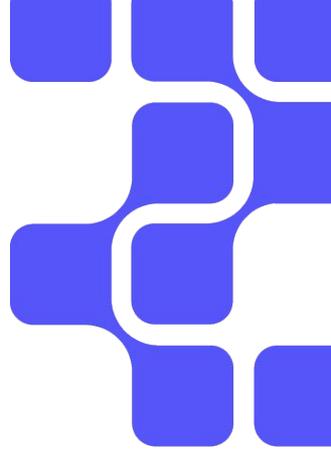
Code Snippets

File navigator

Shell

The screenshot displays the Cloudera Data Science Workbench interface. On the left, a file navigator shows a tree structure with folders like 'src' and files such as '00_datagen.py'. The central pane shows the code editor for '00_datagen.py', containing Python code for data generation using libraries like numpy, pandas, and pyspark. On the right, a terminal window titled 'Untitled Session' shows the session status as 'Running' and provides instructions for executing code. Annotations with arrows highlight key features: 'Workbench' points to the editor area, 'SSH Console' points to the 'Terminal Access' button, 'Code Snippets' points to the 'Data' button, 'File navigator' points to the left sidebar, and 'Shell' points to the terminal input area.

Workbench: starting with a code snippet



Connection Code Snippet

You have access to the Data Connections below.
This also be accessed by clicking the **Data** tab in the top menu.

TYPE Impala Virtual Warehouse	TYPE Impala Virtual Warehouse	TYPE Impala Virtual Warehouse
default-hive-aws TYPE Hive Virtual Warehouse	impala-xsmall-size TYPE Impala Virtual Warehouse	default-impala-aws TYPE Impala Virtual Warehouse
IceSecImpala	go01-optimizer	mukesh-datamart

Use this code to connect to the chosen data source.

```
import cm1.data.v1 as cm1data
CONNECTION_NAME = "default-impala-aws"
conn = cm1data.get_connection(CONNECTION_NAME)

## Sample Usage to get pandas data frame
EXAMPLE_SQL_QUERY = "show databases"
dataframe = conn.get_pandas_dataframe(EXAMPLE_SQL_QUERY)
print(dataframe)
# Closing the connection
conn.close()
```

[Copy Code](#)

Start A New Session

Session Name

Runtime

Editor	Kernel	Edition	Version
Workbench	Python 3.10	Standard	2024.10

Can't find the runtime you need? Enable them in Project Settings.

Enable Spark Spark 3.1.1 - CDP 7.2.11 - CDE 1.13 - ...

Runtime Image
- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.10-standard.2024.10.1-b12

Resource Profile

prueba1.py

```
import cm1.data.v1 as cm1data
CONNECTION_NAME = "default-impala-aws"
conn = cm1data.get_connection(CONNECTION_NAME)

## Sample Usage to get pandas data frame
EXAMPLE_SQL_QUERY = "show databases"
dataframe = conn.get_pandas_dataframe(EXAMPLE_SQL_QUERY)
print(dataframe)
# Closing the connection
conn.close()

## Other Usage Notes:
## Alternate Sample Usage to provide different credentials as optional parameters
#conn = cm1data.get_connection(
#    CONNECTION_NAME, {"USERNAME": "someuser", "PASSWORD": "somepassword"}
#)
## Alternate Sample Usage to get DB API Connection interface
#db.conn = conn.get_base_connection()
## Alternate Sample Usage to get DB API Cursor interface
#db.cursor = conn.get_cursor()
#db.cursor.execute(EXAMPLE_SQL_QUERY)
#for row in db.cursor:
#    print(row)
```

test1 Running

By Javier Gomez Santos - Session - 1 vCPU / 2 GiB Memory - a few seconds ago

Session Logs

```
> import cm1.data.v1 as cm1data
> CONNECTION_NAME = "default-impala-aws"
> conn = cm1data.get_connection(CONNECTION_NAME)
```

Sample Usage to get pandas data frame

```
> EXAMPLE_SQL_QUERY = "show databases"
> dataframe = conn.get_pandas_dataframe(EXAMPLE_SQL_QUERY)
> print(dataframe)
```

	name	comment
0	01_car_data	
1	01_car_dw	
2	0pd_sirlines	
3	_impala_builtins	System database for Impala builtin functions
4	adb101	
...
363	worldwidebank	
364	yalyasin_sirlines	
365	yt_wings_cm1	
366	zain	ZainJordan
367	zainjo	ZainJordan

[368 rows x 2 columns]

Closing the connection

```
> conn.close()
```

Jupyter and spark: starting with a code snippet

The image shows the Cloudera JupyterLab interface with several components highlighted by red and blue boxes:

- Session Name:** test12
- Runtime:** Editor: JupyterLab, Kernel: Python 3.10, Edition: Standard, Version: 2025.06
- Enable Spark:** A toggle switch is turned on, and the selected version is "Spark 3.1.1 - CDP 7.2.11 - CDE 1.13 - ...".
- Runtime Image:** - docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-pbj-jupyterlab-python3.10-standard:2025.06.1-b5
- Resource Profile:** 2 vCPU / 4 GiB Memory, 0 GPU
- Connection Code Snippet:** A panel titled "Connection Code Snippet" provides instructions and a code block for connecting to a chosen data source. The code includes comments for customization and a red arrow pointing to the JupyterLab code editor.
- JupyterLab Code Editor:** The code in the editor is:

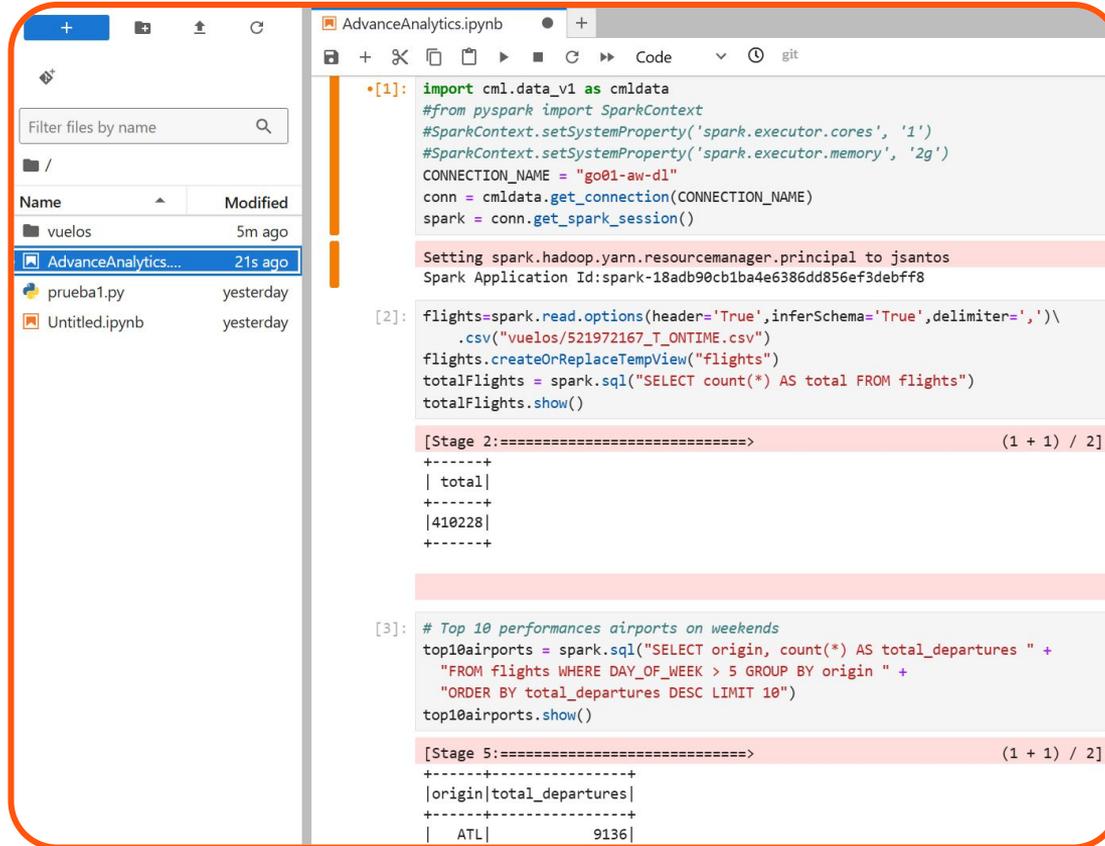
```
[1]: import cml.data_v1 as cmldata

# Sample in-code customization of spark configurations
#from pyspark import SparkContext
#SparkContext.setSystemProperty('spark.executor.cores', '1')
#SparkContext.setSystemProperty('spark.executor.memory', '2g')

CONNECTION_NAME = "go01-aw-dl"
conn = cmldata.get_connection(CONNECTION_NAME)
spark = conn.get_spark_session()

# Sample usage to run query through spark
EXAMPLE_SQL_QUERY = "show databases"
spark.sql(EXAMPLE_SQL_QUERY).show()
```
- Output:** The execution shows the Spark session setup and a list of databases in the default namespace, including tables like 01_car_data, 01_car_dw, 0pdl_airlines, adb10l, ahmedhany_airlines, airflow_dbt_example, airlines, airlines_csv, airlines_data, airlines_iceberg, airlines_iceberg..., airlines_maint, airlines_mjain, airquality, amallegni, aptdemo, asoni_airlines, atlas_demo, bankdemo, and banking_database. The output indicates "only showing top 20 rows".

Copy your own data & analyze it



The screenshot displays a Jupyter Notebook environment. On the left, a file browser shows a directory with files: 'vuelos' (5m ago), 'AdvanceAnalytics...' (21s ago), 'prueba1.py' (yesterday), and 'Untitled.ipynb' (yesterday). The main area shows the notebook code:

```
[1]: import cml.data_v1 as cmldata
#from pyspark import SparkContext
#SparkContext.setSystemProperty('spark.executor.cores', '1')
#SparkContext.setSystemProperty('spark.executor.memory', '2g')
CONNECTION_NAME = "go01-aw-d1"
conn = cmldata.get_connection(CONNECTION_NAME)
spark = conn.get_spark_session()

Setting spark.hadoop.yarn.resourcemanager.principal to jsantos
Spark Application Id:spark-18adb90cb1ba4e6386dd856ef3debff8

[2]: flights=spark.read.options(header='True',inferSchema='True',delimiter=',')\
.csv("vuelos/521972167_T_ONTIME.csv")
flights.createOrReplaceTempView("flights")
totalFlights = spark.sql("SELECT count(*) AS total FROM flights")
totalFlights.show()

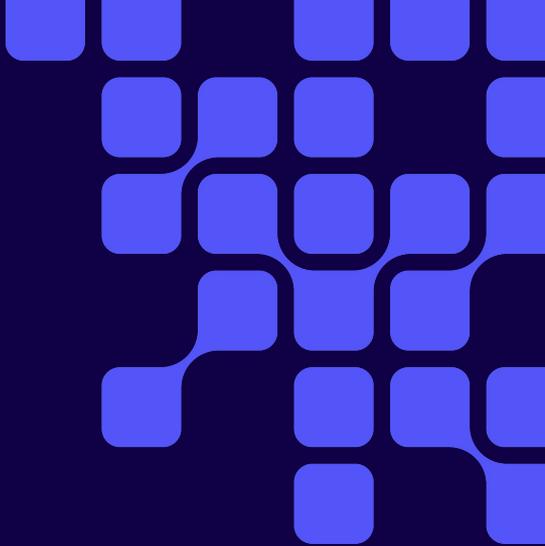
[Stage 2:=====] (1 + 1) / 2]
+-----+
| total|
+-----+
|410228|
+-----+

[3]: # Top 10 performances airports on weekends
top10airports = spark.sql("SELECT origin, count(*) AS total_departures " +
"FROM flights WHERE DAY_OF_WEEK > 5 GROUP BY origin " +
"ORDER BY total_departures DESC LIMIT 10")
top10airports.show()

[Stage 5:=====] (1 + 1) / 2]
+-----+
|origin|total_departures|
+-----+
| ATL | 9136 |
```

CLOU Ξ ERA

Spark



Apache Spark

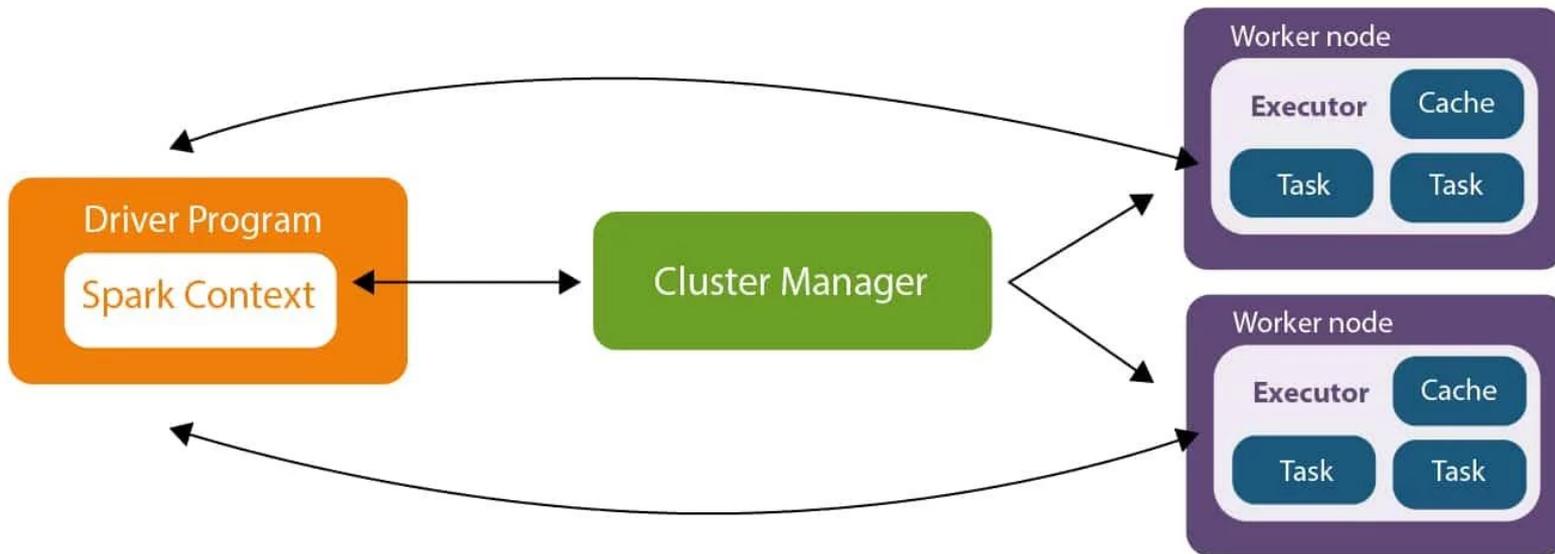
Unified analytics engine for large-scale data processing



- Accept multiple language
- Open source
- Meant for large scale processing

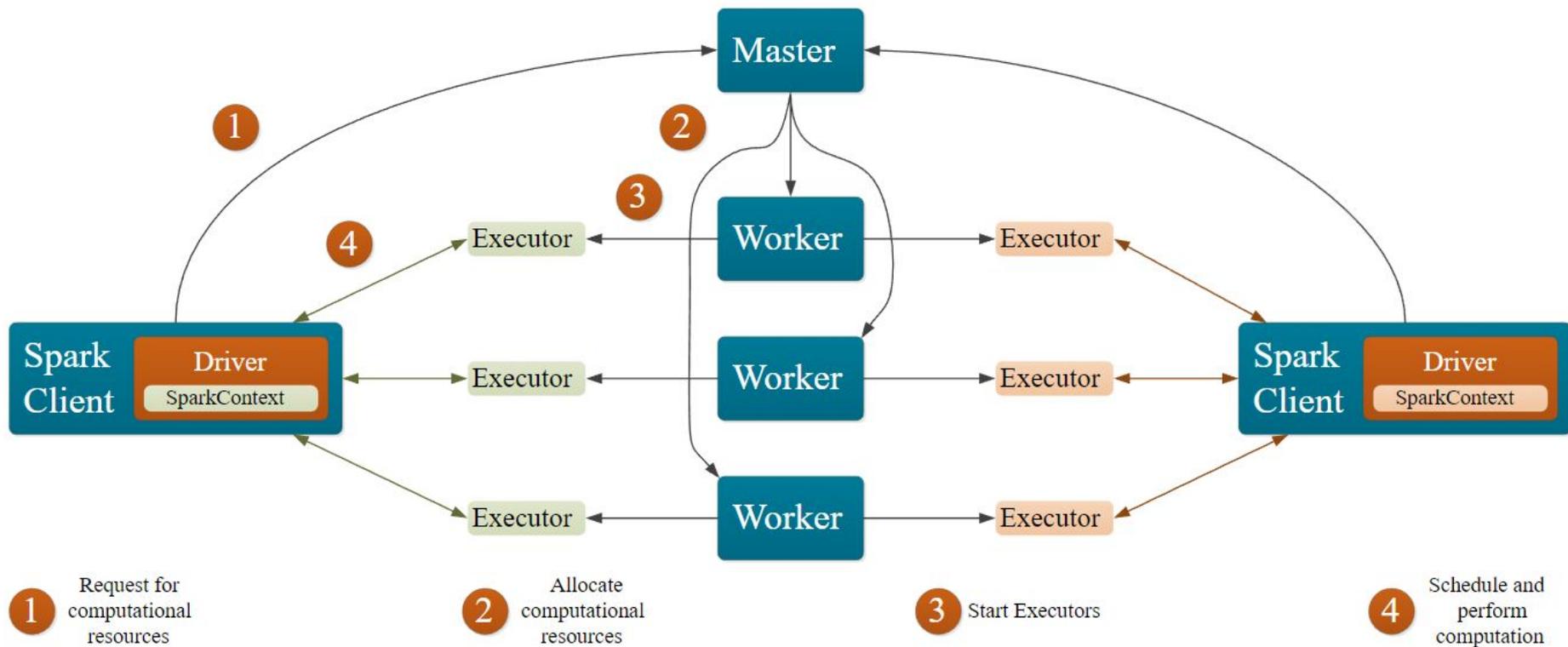
Spark architecture

For a single process



Spark architecture

Application flow



Spark architecture

DAG

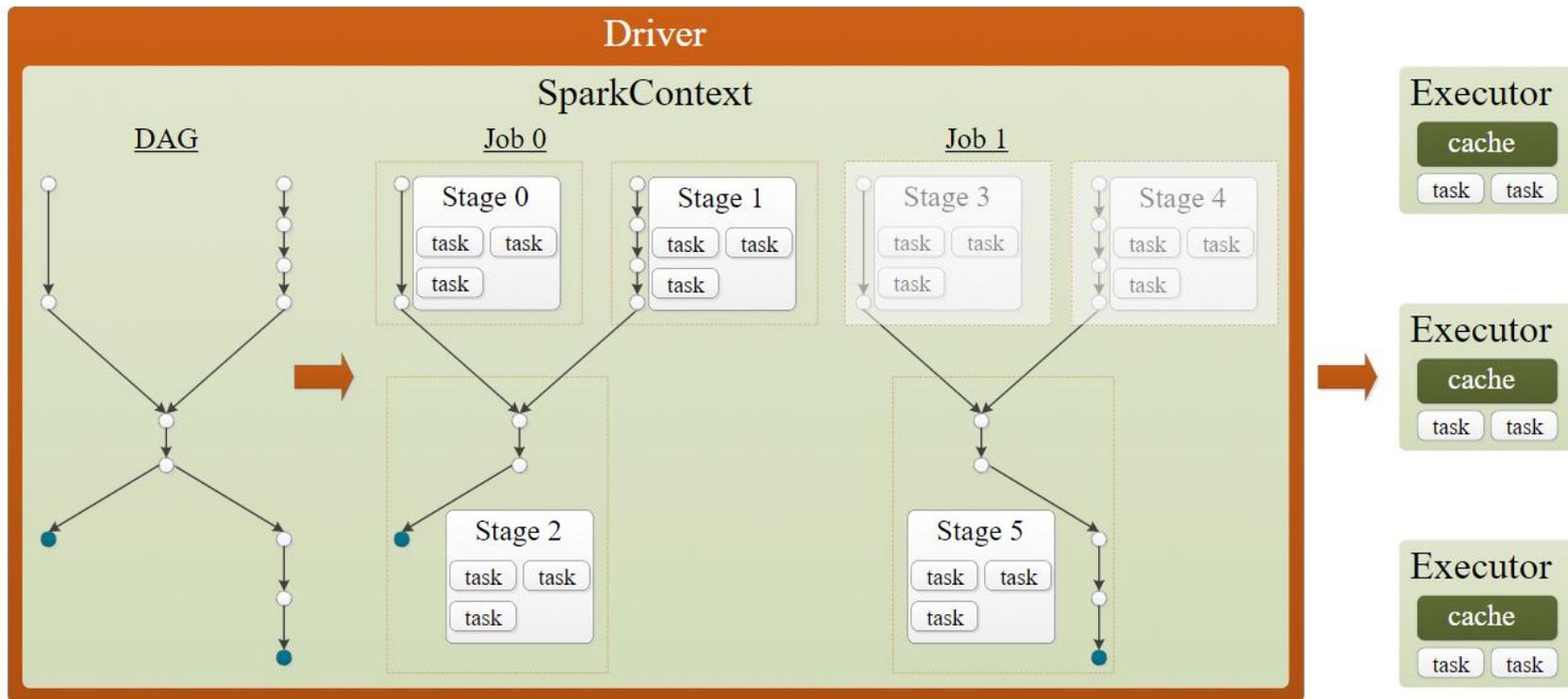
Spark works following a graph, this graph is called DAG which stands for Directed Acyclic Graph.

Every process we run is divided into smaller tasks, some may have a dependency with previous ones and others may be executed in parallel.

This tasks are run by executors.

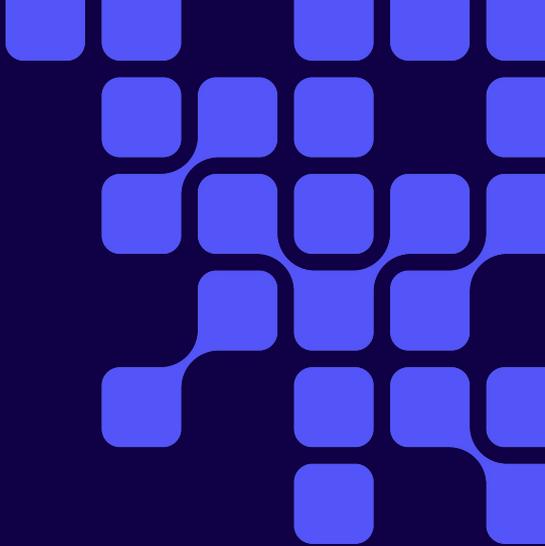
Spark architecture

DAG



CLOU~~D~~ERA

Spark Best Practices



Remove unwanted actions

- Common practice to add actions such as `count()`, `collects()`, `takes()`, take for testing/debugging purposes
 - Results in redundant re-computations(w/o cache/checkpoints)
 - Increases the execution time

- Recommendation(s):
 - Remove them, if not needed
 - Alternatively look at using:
 - Caching
 - Checkpointing
 - External writes followed by reads

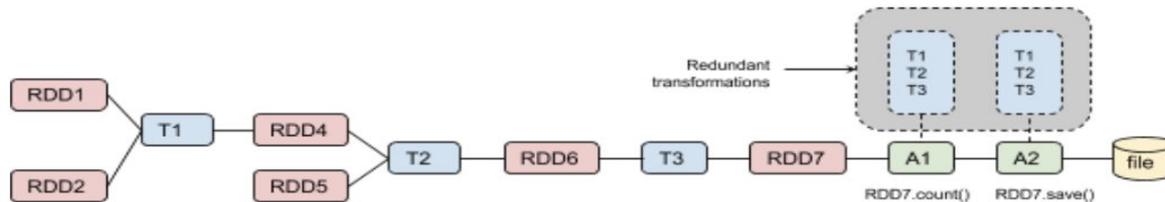
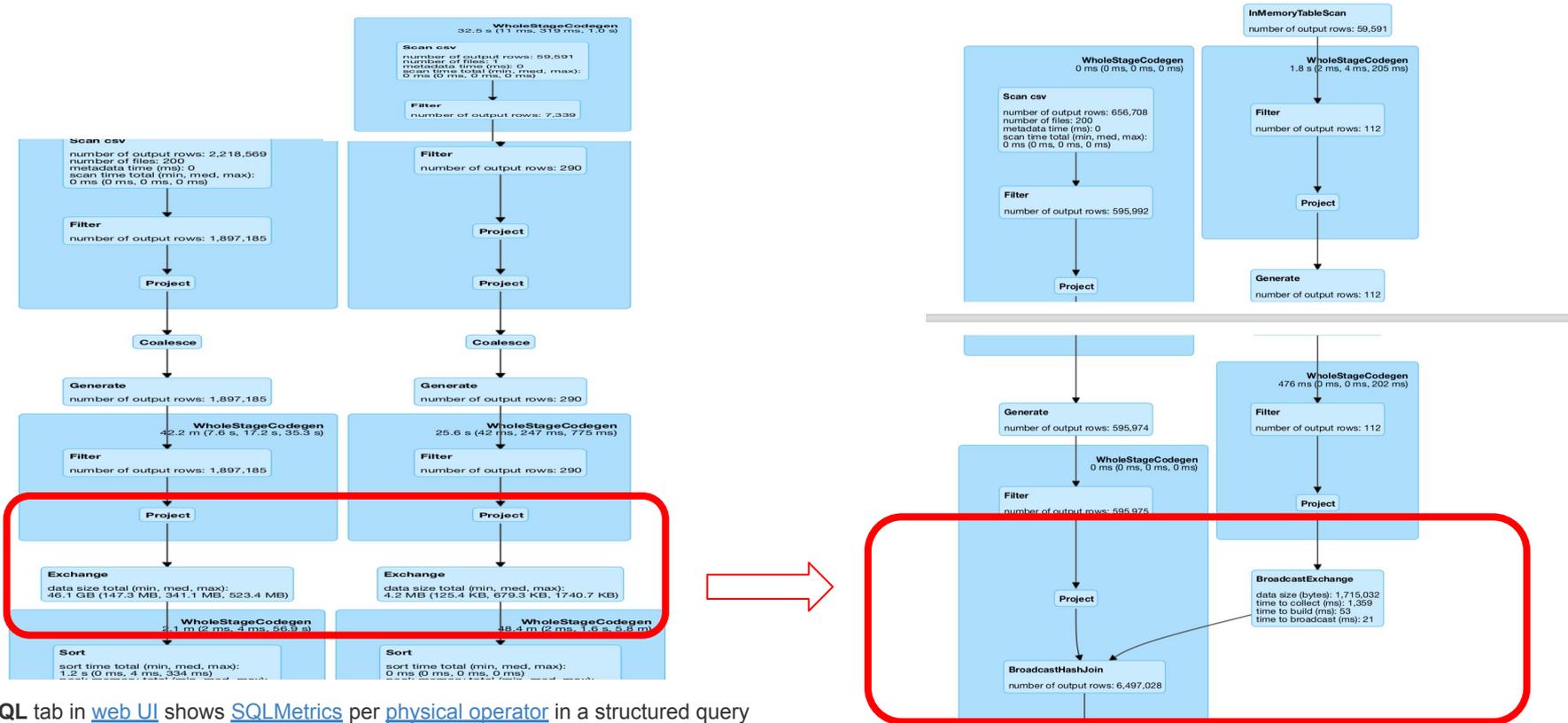


Fig1: Redundant transformations

Broadcast-Hash for large-small table join



SQL tab in [web UI](#) shows [SQLMetrics](#) per [physical operator](#) in a structured query physical plan.

Reduce the size of data structure

- Large objects
 - result in Spark spilling data to disk more often
 - reduces the number of deserialized records Spark can cache
 - result in greater disk and network I/O
- Recommendations:
 - selecting only needed data/cols (example in next slide)

```

== Physical Plan ==
SortAggregate(key=[cpykey_deviceid#343], functions=[first(fingerprint#344, false), first(crashcompany#345, false),
first(crashdevice#346, false), first(similarity#347, false)], output=[cpykey_deviceid#343, fingerprint#344, crashcompany#345,
crashdevice#346, similarity#347])
+- *Sort [cpykey_deviceid#343 ASC NULLS FIRST], false, 0
  +- Exchange hashpartitioning(cpykey_deviceid#343, 150)
    +- SortAggregate(key=[cpykey_deviceid#343], functions=[partial_first(fingerprint#344, false), partial_first(crashcompany#345,
false), partial_first(crashdevice#346, false), partial_first(similarity#347, false)], output=[cpykey_deviceid#343, first#390,
valueSet#391, first#392, valueSet#393, first#394, valueSet#395, first#396, valueSet#397])
      +- *Sort [cpykey_deviceid#343 ASC NULLS FIRST], false, 0
        +- *Sort [similarity#347 ASC NULLS FIRST], true, 0
          +- Exchange rangepartitioning(similarity#347 ASC NULLS FIRST, 150)
            +- *HashAggregate(keys=[datasetA#303, datasetB#319], functions=[], output=[cpykey_deviceid#343, fingerprint#344,
crashcompany#345, crashdevice#346, similarity#347])
              +- Exchange hashpartitioning(datasetA#303, datasetB#319, 150)
                +- *HashAggregate(keys=[datasetA#303, datasetB#319], functions=[], output=[datasetA#303, datasetB#319])
                  +- *Project [datasetA#303, datasetB#319]
                    +- *BroadcastHashJoin [entry#304, hashValue#305], [entry#320, hashValue#321], Inner, BuildRight,
(UDF(datasetA#303.norm_fingerprint, datasetB#319.norm_fingerprint) < 1.0)
                      :- *Project [named_struct(fingerprint, fingerprint#7, cpykey_deviceid, cpykey_deviceid#0,
norm_fingerprint, norm_fingerprint#236, lsh, lsh#297) AS datasetA#303, entry#304, hashValue#305]
                        : +- *Filter isNotNull(hashValue#305)
                        :   +- Generate posexplode(lsh#297), true, false, [entry#304, hashValue#305]
                        :     +- *Project [fingerprint#7, cpykey_deviceid#0, UDF(UDF(UDF(fingerprint#7)))] AS lsh#297
                        :       +- *Filter ((AtLeastNulls(n, fingerprint#7,cpykey_deviceid#0) &&
isNotNull(fingerprint#7)) && NOT Contains(fingerprint#7, unknown))
                        :         +- *FileScan parquet [cpykey_deviceid#0,fingerprint#7] Batched: true, Format: Parquet,
Location: InMemoryFileIndex[maprfs:///tmp/fingerprint/fingerprint.pqa], PartitionFilters: [], PushedFilters: [IsNotNull(fingerprint),
Not(StringContains(fingerprint,unknown))], ReadSchema: struct<cpykey_deviceid:string,fingerprint:string>
                          +- BroadcastExchange HashedRelationBroadcastMode(List(input[1, int, false], input[2, vector, true]))
                          +- *Project [named_struct(cpykey, cpyKey#43, deviceId, deviceId#45, fingerprint, fingerprint#49,
fp_tokens, fp_tokens#257, raw_fingerprints, raw_fingerprints#263, idf_fingerprint, idf_fingerprint#270, norm_fingerprint,
norm_fingerprint#278, lsh, lsh#287) AS datasetB#319, entry#320, hashValue#321]
                            +- *Filter isNotNull(hashValue#321)
                              +- Generate posexplode(lsh#287), true, false, [entry#320, hashValue#321]
                                +- *Project [cpyKey#43, deviceId#45, fingerprint#49, UDF(fingerprint#49) AS
fp_tokens#257, UDF(UDF(fingerprint#49)) AS raw_fingerprints#263, UDF(UDF(UDF(fingerprint#49))) AS idf_fingerprint#270,
UDF(UDF(UDF(UDF(fingerprint#49)))) AS norm_fingerprint#278, UDF(UDF(UDF(UDF(fingerprint#49)))) AS lsh#287]
                                  +- *Project [cpykey#43, crashed#44, deviceId#45, fingerprint#49, lastReset#52]
                                    +- *Filter (cast(pythonUDF0#398 as double) > 1.5340179487468586E9)
                                    +- BatchEvalPython [get_epochtime(coalesce(lastReset#52, 2011-11-11)),
[cpykey#43, crashed#44, deviceId#45, fingerprint#49, lastReset#52, pythonUDF0#398]
                                      +- *Filter (((isNotNull(crashed#44) && (cast(crashed#44 as int) = 1)) &&

```

Selecting only the required column
can reduce the data row size, thus
reducing the network overhead
associated with shuffle

```

new_fp_tokenized_data_hashed = model.transform(fp_tokenized_data,select(['fingerprint', 'cpykey_deviceid', 'norm_fingerprint'])
print("Transformation phase 1 Done")
cdf2_hashed = model.transform(cdf2,select(['fingerprint', 'norm_fingerprint', 'cpykey', 'deviceId'])
print("Transformation phase 2 Done")

```



Projected &
exchanged
unnecessary
fields such as
fp_tokens,
raw_fingerprints,
idf_fingerprints ..

```

+- *Sort [cpykey_deviceid#343 ASC NULLS FIRST], false, 0
  +- Exchange hashpartitioning(entry#374, hashValue#375, 300)
    +- *Project [named_struct(fingerprint, fingerprint#221, norm_fingerprint,
norm_fingerprint#329, cpyKey, cpyKey#219, deviceId, deviceId#220, lsh, lsh#366) AS datasetB#373, entry#374, hashValue#375]
      +- *Filter isNotNull(hashValue#375)
        +- Generate posexplode(lsh#366), true, false, [entry#374, hashValue#375]
          +- *Project [fingerprint#221, UDF(UDF(UDF(fingerprint#221)))] AS
norm_fingerprint#329, cpyKey#219, deviceId#220, UDF(UDF(UDF(UDF(fingerprint#221)))) AS lsh#366

```

Make the best of data caching

- There are options...
 - MEMORY_ONLY (default) – memory deserialized
 - MEMORY_AND_DISK – store data to disk that doesn't fit in memory
 - MEMORY_ONLY_SER – memory serialized
 - MEMORY_AND_DISK_SER
 - DISK_ONLY – only store to disk
 - *_2 – replicate each partition to two nodes (faster recovery from faults)
- Spark also automatically persists some intermediate data in shuffle operations, even without users calling persist. This is done to avoid recomputing the entire input if a node fails during the shuffle.
- Least Recently Used RDDs are removed from cache

Recommendations:

- Persist on the resulting RDD if they plan to reuse it.
- Due to max container size limitation, use “DISK_ONLY” storage level to allow more memory for execution
 - `df.persist(pyspark.StorageLevel.DISK_ONLY)`
- Unpersist using
 - `rdd.unpersist()`
 - or `spark.catalog.clearCache()`

Learn to use `foreachPartition()`

Some executions run at element level, and can add a huge overhead that can be avoided (e.g: opening a database connection)

Sometimes you can work at partition level. This allows you to run operations just once per partition

```
df = spark.read[...]  
  
def func(itr):  
    # Heavy work like opening a DB connection  
    for person in itr:  
        #Whatever you want to do for each  
        element  
  
df.foreachPartition(func)
```

Use storage formats as Parquet/ORC

And choose a compression

- 10x faster read performance
- 11x faster execution performance
- Reduced storage space
- Minimize disk & network I/O
- Reduced spark filter pushdown
- facilitate compression
- Reduce shuffle failure, in some cases

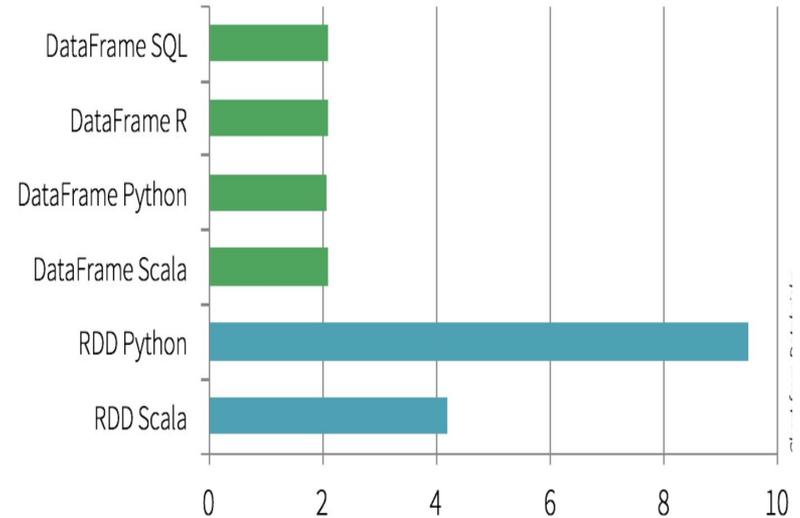
File Format	Query Time (sec)	Size (GB)
CSV	2892.3	437.46
Parquet: LZO	50.6	55.6
Parquet: Uncompressed	43.4	138.54
Parquet: GZIP	40.3	36.78
Parquet: Snappy	28.9	54.83

Use Dataframe or Dataset over RDDs

- Optimized logical and physical query plan using catalyst
- Reduced Garbage collection overhead
- Higher space and speed efficiency
 - Compacted column memory format
- Uses tungsten encoders:
 - Efficient serde of JVM objects
 - Compact bytecode
 - Execute at Superior speed

Recommendation:

- If you are a Python user, use DataFrames and resort back to RDDs if you need more control.



Time to aggregate 10 million integer pairs (in seconds)

Respect your driver

- There's only one driver
- Several operations send data back to the driver
 - Collect
 - Reduce
 - Accumulators
- Avoid sending large datasets
 - Don't collect on large datasets(use take)
 - `toPandas` is essentially collect
 - Use Pandas on spark!!! → <https://spark.apache.org/pandas-on-spark/>
 - Don't reduce on large data structures(use TreeReduce)
 - Accumulators meant for small counters, not large data structures

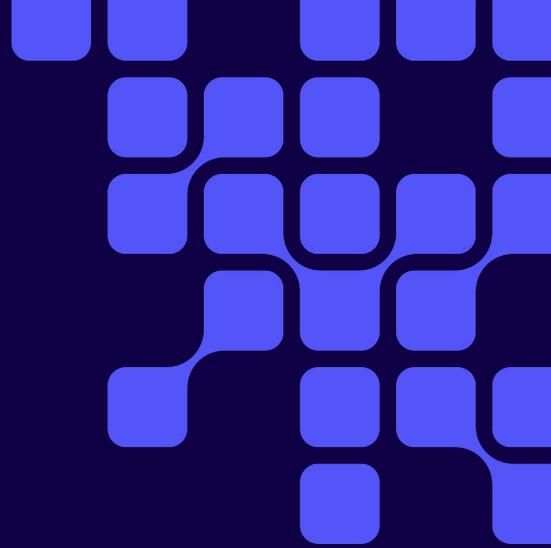
Use right-sized executors in terms of cores

- Be careful choosing your resources
- IOPs per executor can make useful too much cores.
 - Performance limit max # of cores per executor to 5 (it may change for SSD or other scenarios)
- Understand the core to memory ratio
 - Rule-of-thumb: 1 core : minimum of 4GB of memory
 - For memory-bound apps, experiment to find optimal memory settings
- Configs:

```
from pyspark import SparkContext
SparkContext.setSystemProperty('spark.executor.cores', '2')
SparkContext.setSystemProperty('spark.executor.memory', '4g')
```

CLOUĐERA

Challenges



1- Predicción de la demanda

Ayuda a predecir cuándo y dónde se necesitan taxis, lo que reduce el tiempo de inactividad.

Sed creativos:

- Combinar con datos meteorológicos (temperatura, precipitaciones, nieve, viento)
- Vincular los puntos de parada de taxis con los datos de ubicación de los restaurante, hospitales, centros educativos...
- Combinar frecuencia de viajes con interacciones de redes sociales, especialmente si usan geolocalización (Twitter/X, LinkedIn... Tinder!)
- Correlación de datos de reservas en restaurantes con frecuencia de viajes

¿Qué está de moda en New York?

Análisis dinámico de precios e ingresos

Analizar los registros de viajes para optimizar las estrategias de precios en diferentes zonas de la ciudad, convirtiendo los precios en problemas de clasificación basados en la demanda

Esto es algo que ya hacen las plataformas como Uber y Cabify.

¿Se os ocurre cómo fomentar el transporte público en condiciones de mala calidad del aire influenciando en los precios?

Predicción del comportamiento de propinas de los pasajeros

Los modelos normalmente evalúan factores como la distancia del viaje, la duración y los recargos por congestión para predecir, e incluso aumentar, los montos de las propinas.

¿Qué más datos podrías trabajar para predecir esto?

¿Se te ocurre alguna manera de influenciar en las propinas?

Información demográfica y de comportamiento de los pasajeros

Análisis de los datos teniendo en cuenta por ejemplo número de pasajeros, zonas, las propinas dadas para poder mapear las zonas geográficas con poder adquisitivo de la zona.

Puedes mezclar estos datos con precios de alquileres, restaurantes bien valorados y precios medios de la zona.

Las investigaciones muestran que un aumento de una desviación estándar en la cantidad de taxis que llegan a la zona en zonas específicas es un fuerte predictor de un aumento en los alquileres residenciales al año siguiente.

¿Dónde invertiríais en un piso en Nueva York?

Rutas de respuesta a emergencias

Usa los datos de los viajes de los taxis, origen, destino y tiempos medios para buscar rutas alternativas.

Con la información que tienes, y asumiendo que los taxistas siempre toman la ruta más corta ¿podrías buscar alternativas para los servicios de emergencia en periodos de congestión?



Gracias