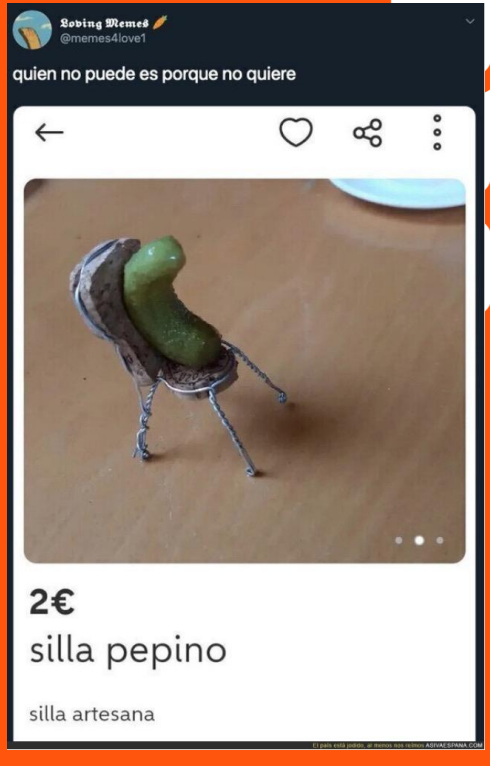
La **IA** no genera la **FELICIDAD**, pero se le **aproxima** tanto...

Grado en ingeniería de datos e IA

CLOUDERA



CLOUDERA

diario de **PARLA.**

INICIO | POLÍTICA | OPINIÓN | SUCESOS | SOCIEDAD | ENTRETENIMIENTO | CULTURA | DEPORTES | ECONOMÍA | WHATSAPP

RECIBE LA ÚLTIMA HORA DE PARLA EN TU WHATSAPP

Pulsa en la AQUÍ para seguirnos

Diario de Parla
En línea
SEGUIR
Activa la campanita para recibir notificaciones.

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

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...

DATA DECAYS!

What does “real-time” mean to you?

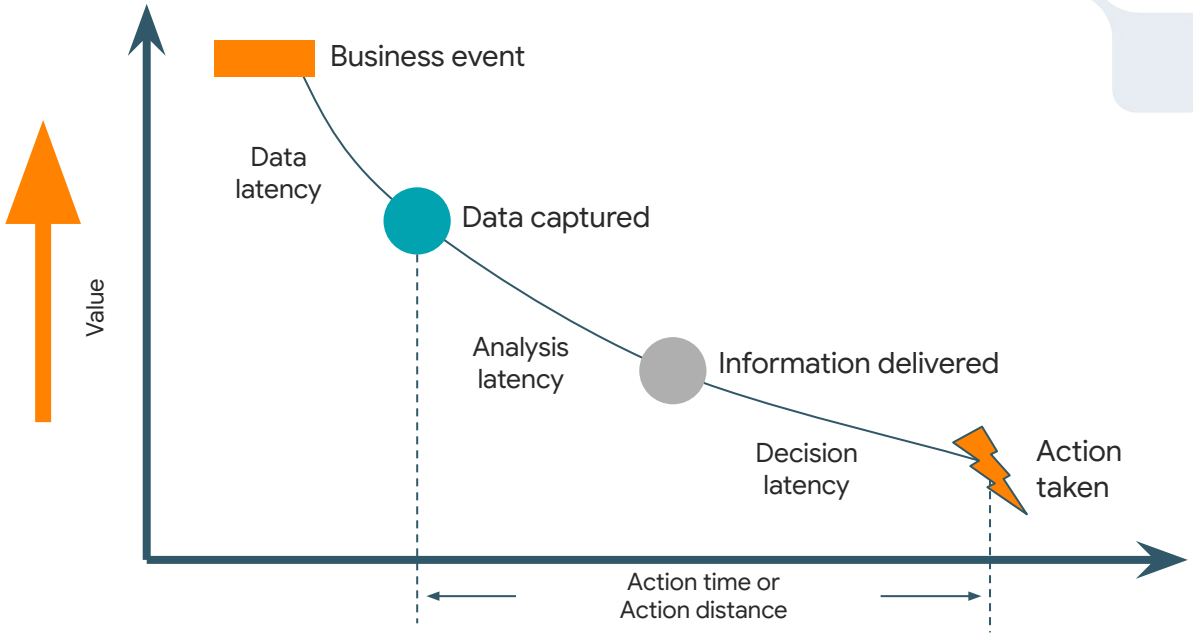
Examples

Manufacturing Intelligent Ops

Threat Detection

Distributed Ops

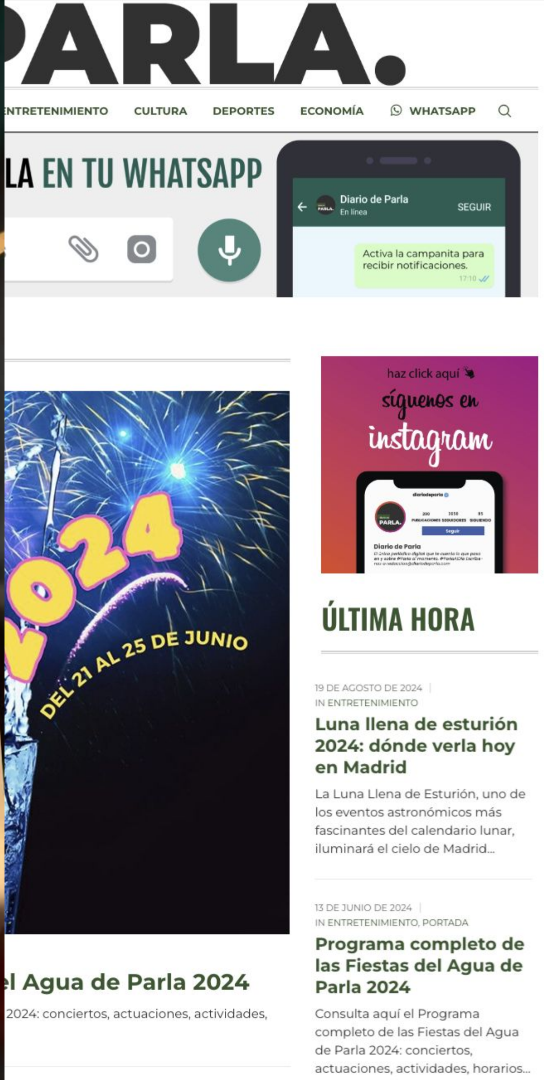
Real-time Offer

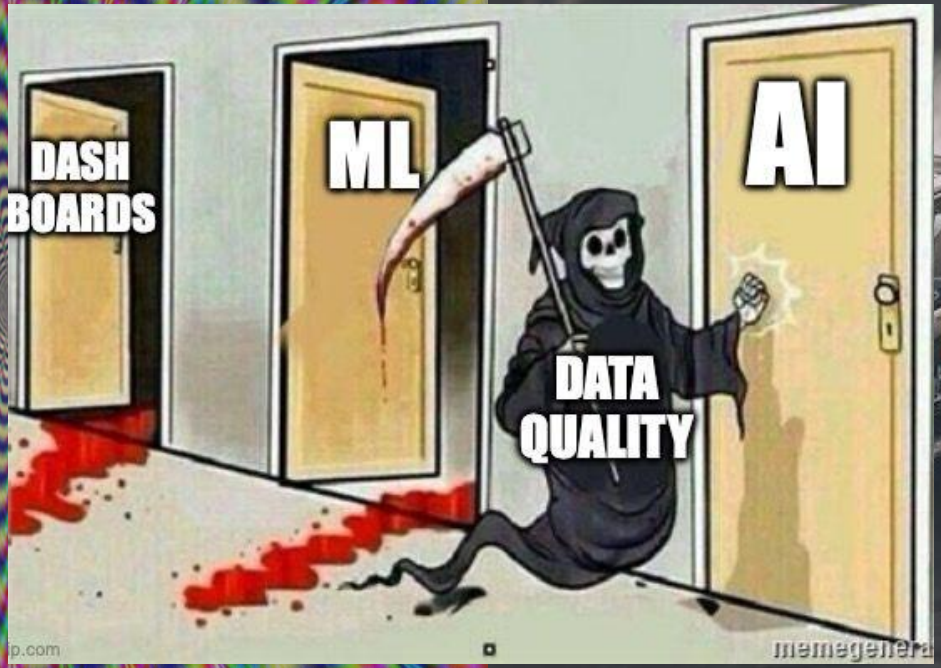


CLOUDERA


dailystar.co.uk
DAILY STAR
10p
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 SEAGULL STOLE M DOG FR GARDEN
Bird flies off with chihu


The York Times
Honey York Times
cotve Yot's Crat
2550
Unick bor's applel
Uhred By Klangene
Proper derties
Club barder
Statrighon inreal of destation
Inalists to Last indutting
Flamions lorss traud coparines
Farms is Technology Clamer of Compuarnent


PARLA.
ENTRETENIMIENTO
CULTURA
DEPORTES
ECONOMÍA
WHATSAPP
LA EN TU WHATSAPP
Diario de Parla
En línea
SEGUIR
Activa la campanita para recibir notificaciones.
17:10
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.

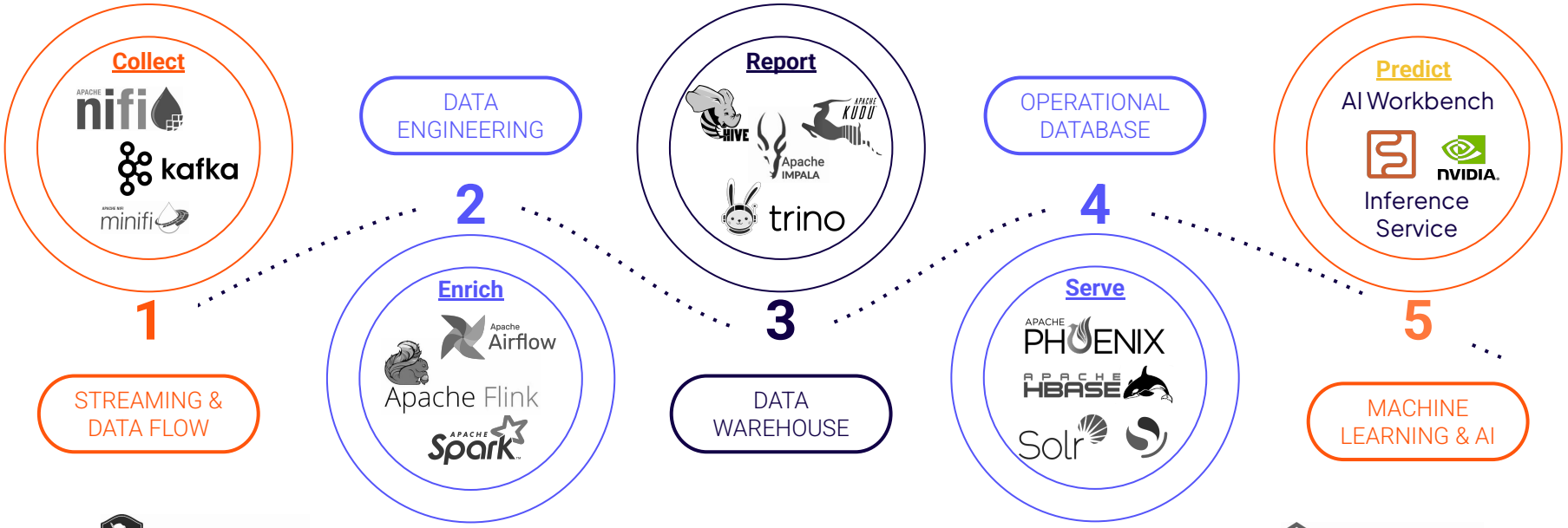
The British Columbia Civil Resolution Tribunal rejected that argument, ruling that Air Canada had to pay Moffatt $812.02 (£642.64) in damages and tribunal fees. "It should be obvious to Air Canada that it is responsible for all the information on its website," read tribunal member Christopher Rivers' 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.

CLOUDERA

Cloudera AI Workbench

END-TO-END COMPLETE LIFECYCLE

Manage and secure the data lifecycle in any cloud or datacenter

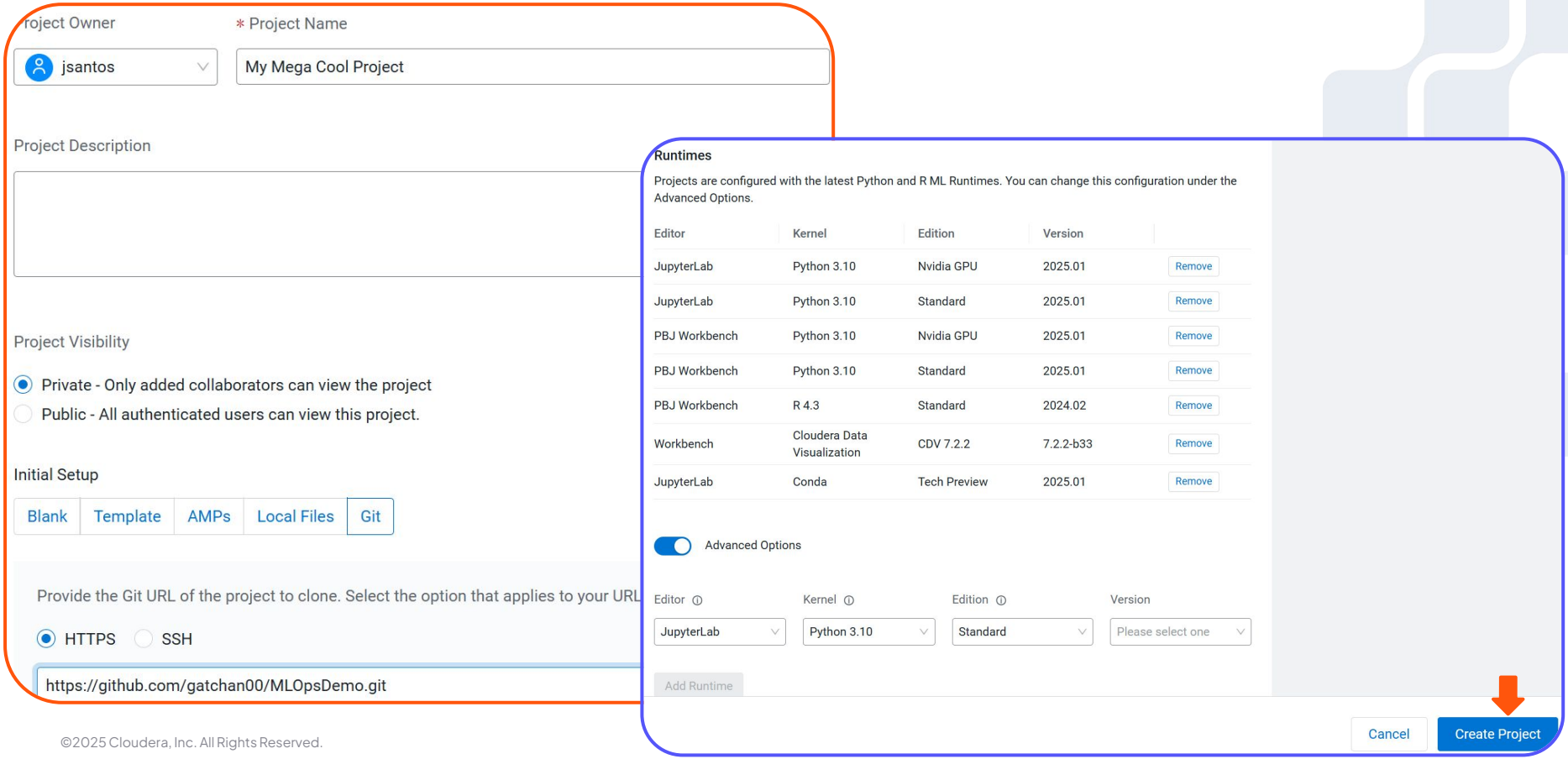


Apache Ranger SECURITY | GOVERNANCE | LINEAGE | MANAGEMENT | AUTOMATION Apache Atlas

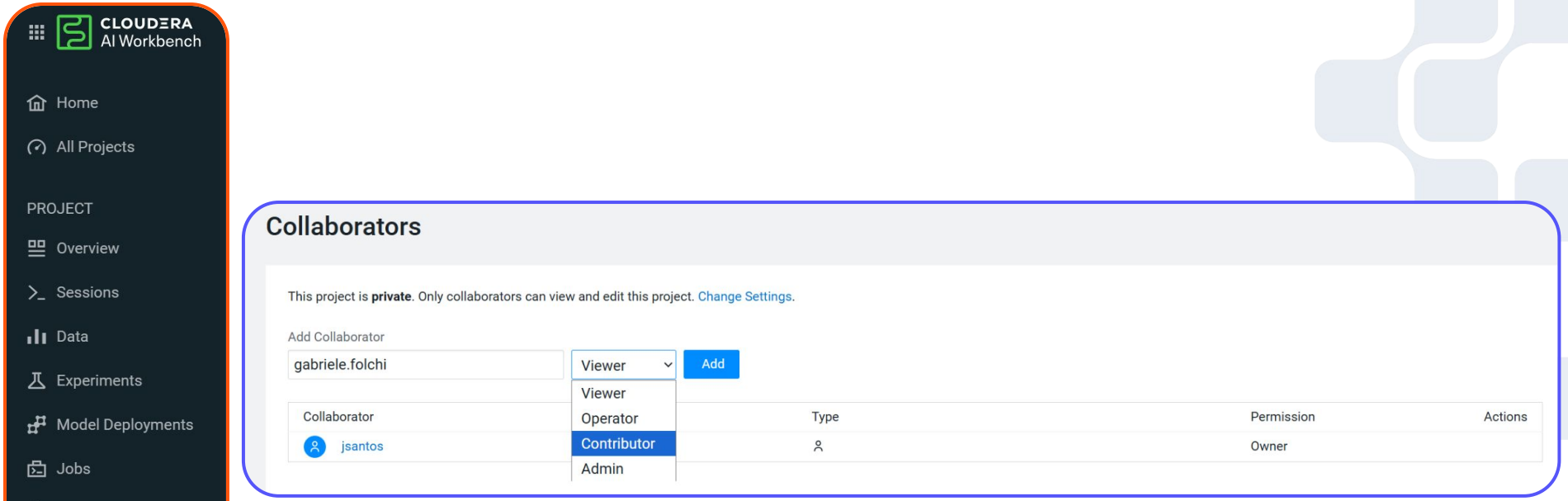
Data Center · aws · Microsoft Azure · Google Cloud Platform

OCTOPAI By CLOUDERA

Project Creation

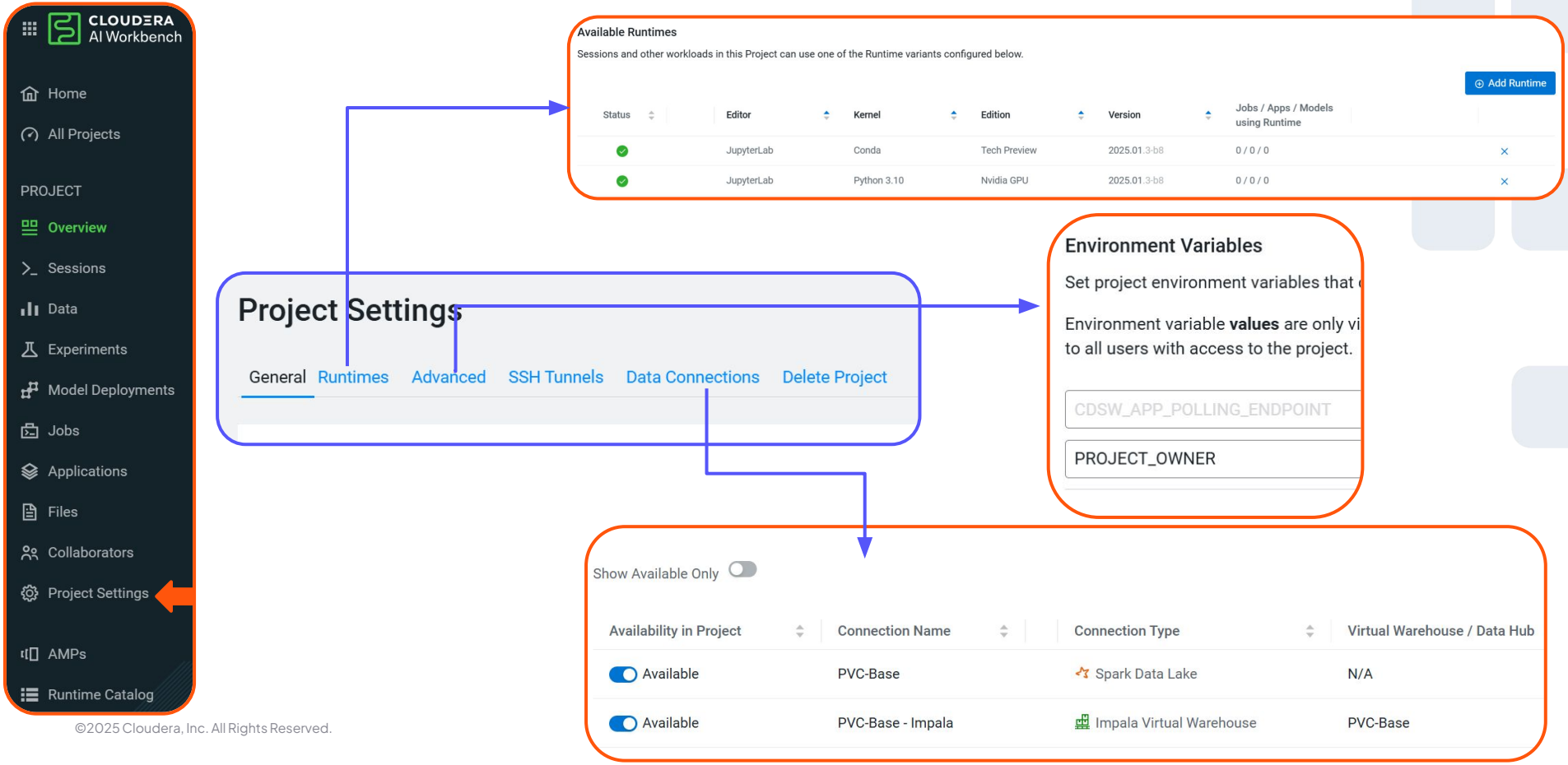


Managing Collaborators

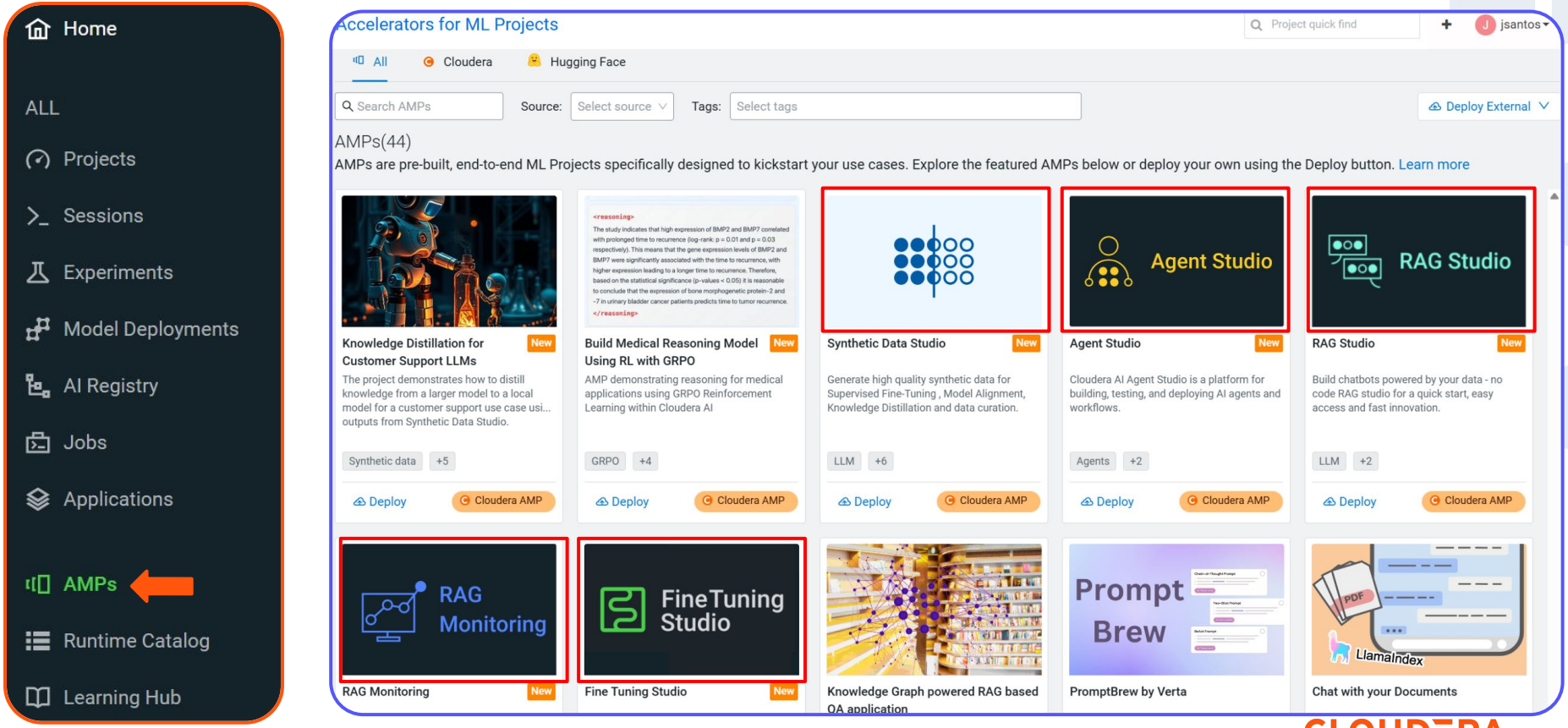


CLOUDERA

Project Settings



AMPs



CLOUDERA

Create a session

Session Name

Untitled Session

Runtime

Editor | Kernel | Edition | Version

Workbench | Python 3.10 | Standard

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

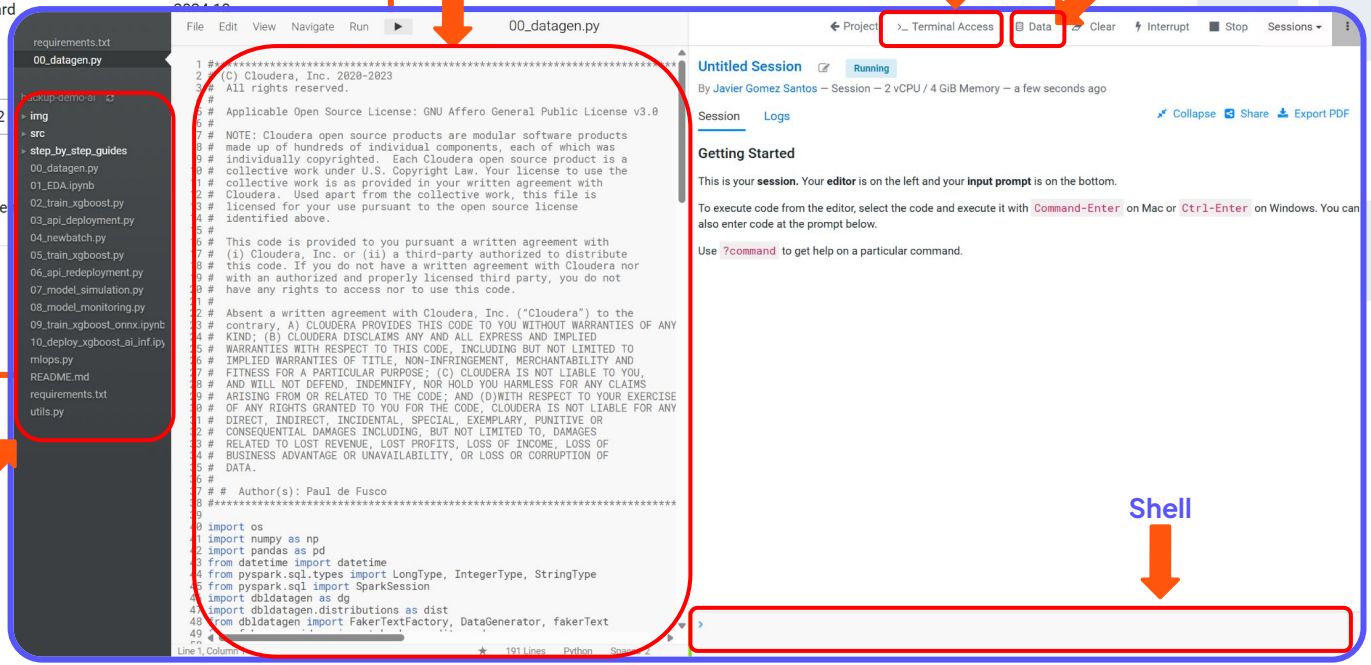
Workbench

SSH Console

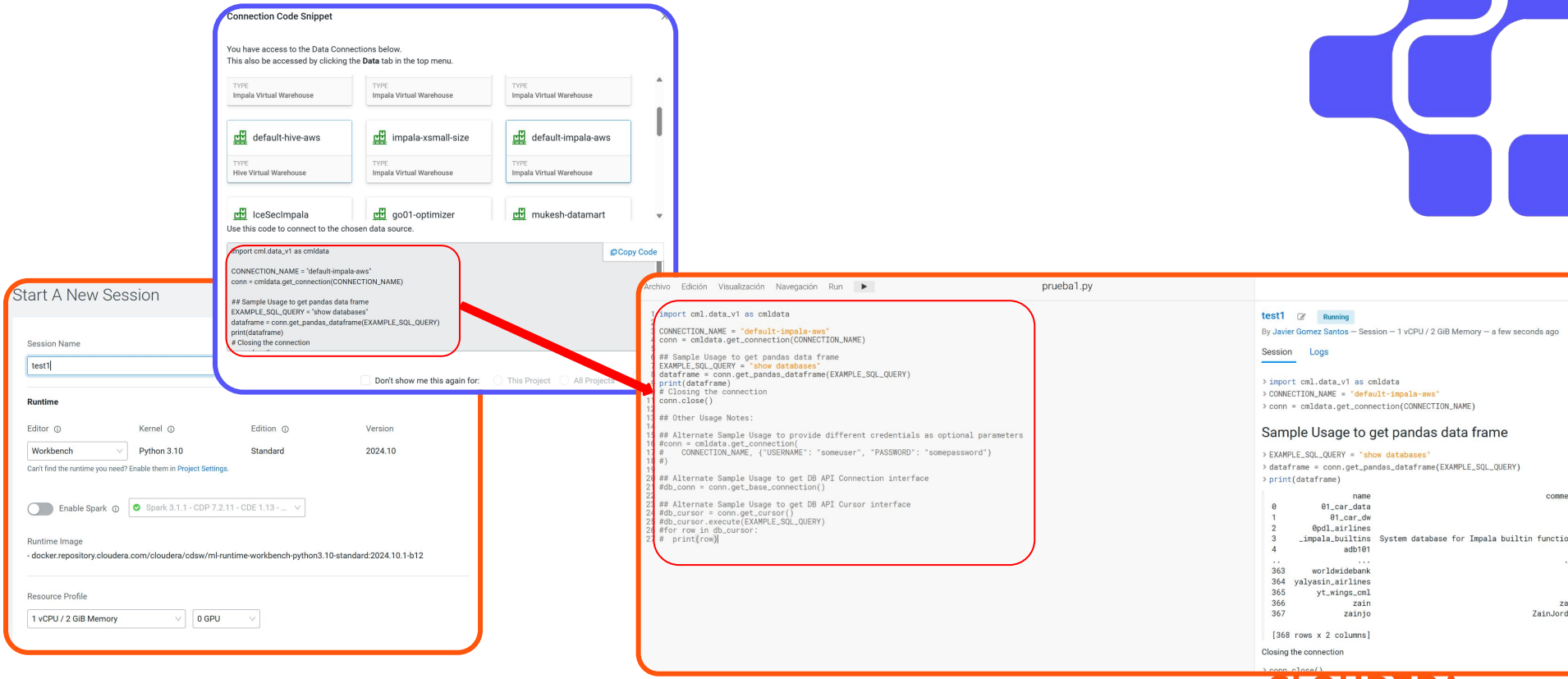
Code Snippets

File navigator

Shell

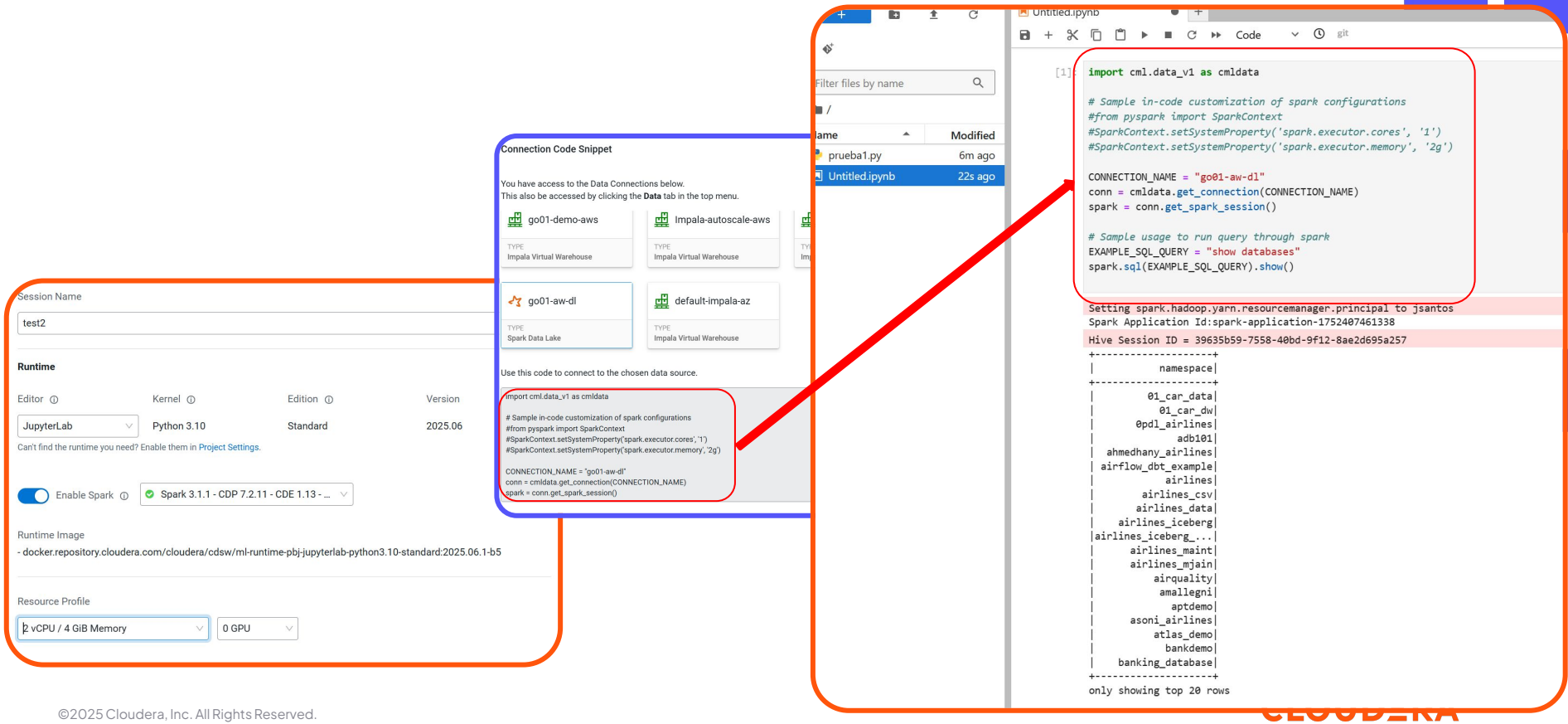


Workbench: starting with a code snippet

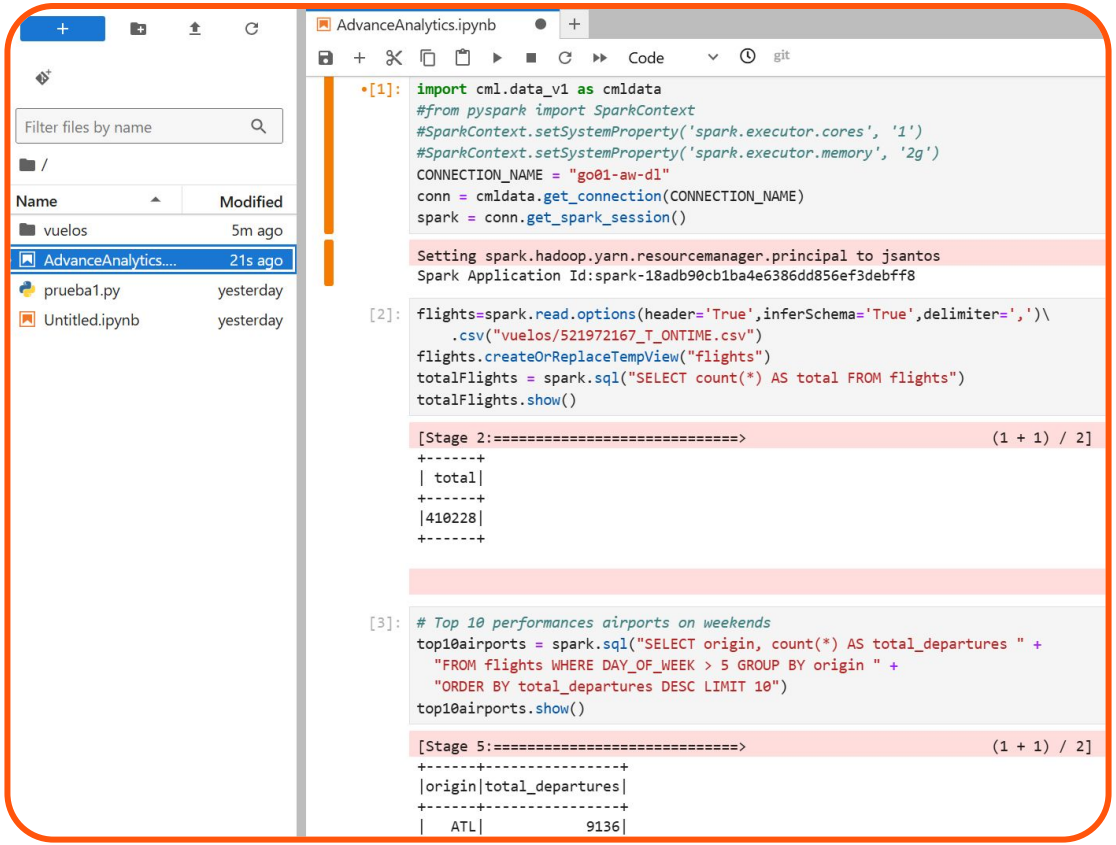


CLOUDERA

Jupyter and spark: starting with a code snippet



Copy your own data & analyze it



CLOUDERA

CLOUDERA

Spark

Apache Spark

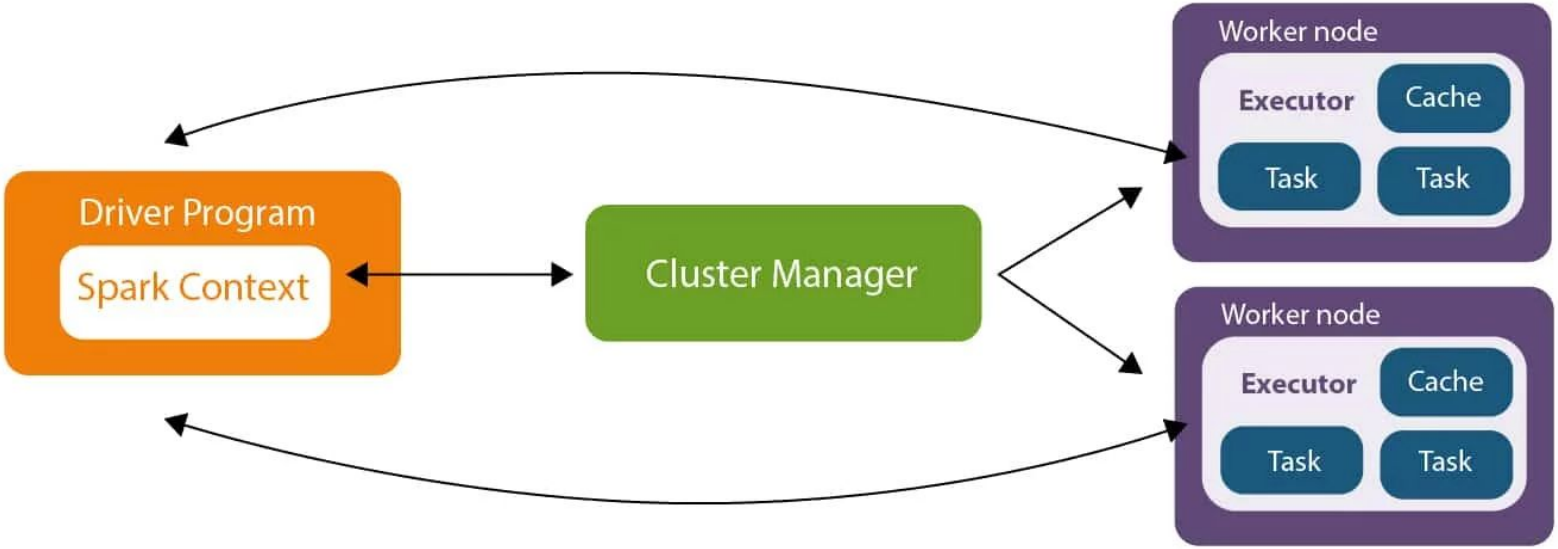
Unified analytics engine for large-scale data processing

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

CLOUDERA

Spark architecture

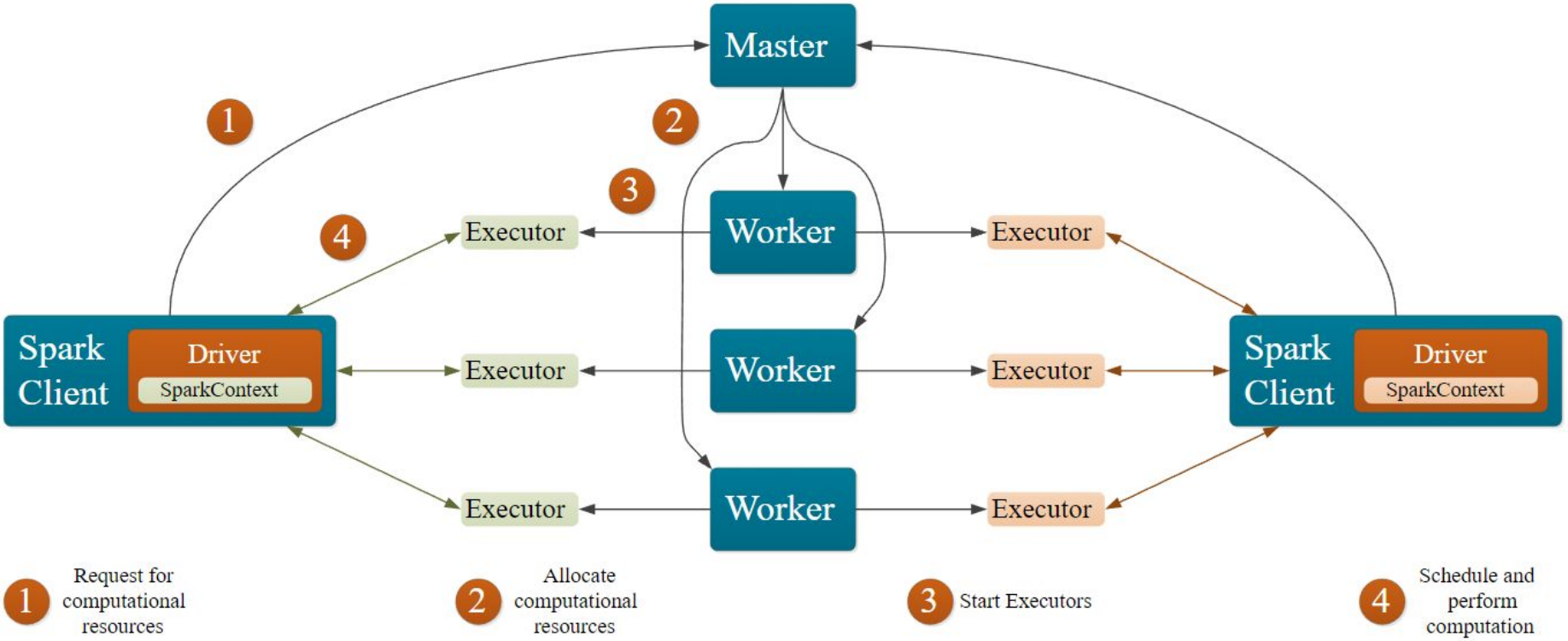
For a single process



CLOUDERA

Spark architecture

Application flow



Spark architecture

DAG

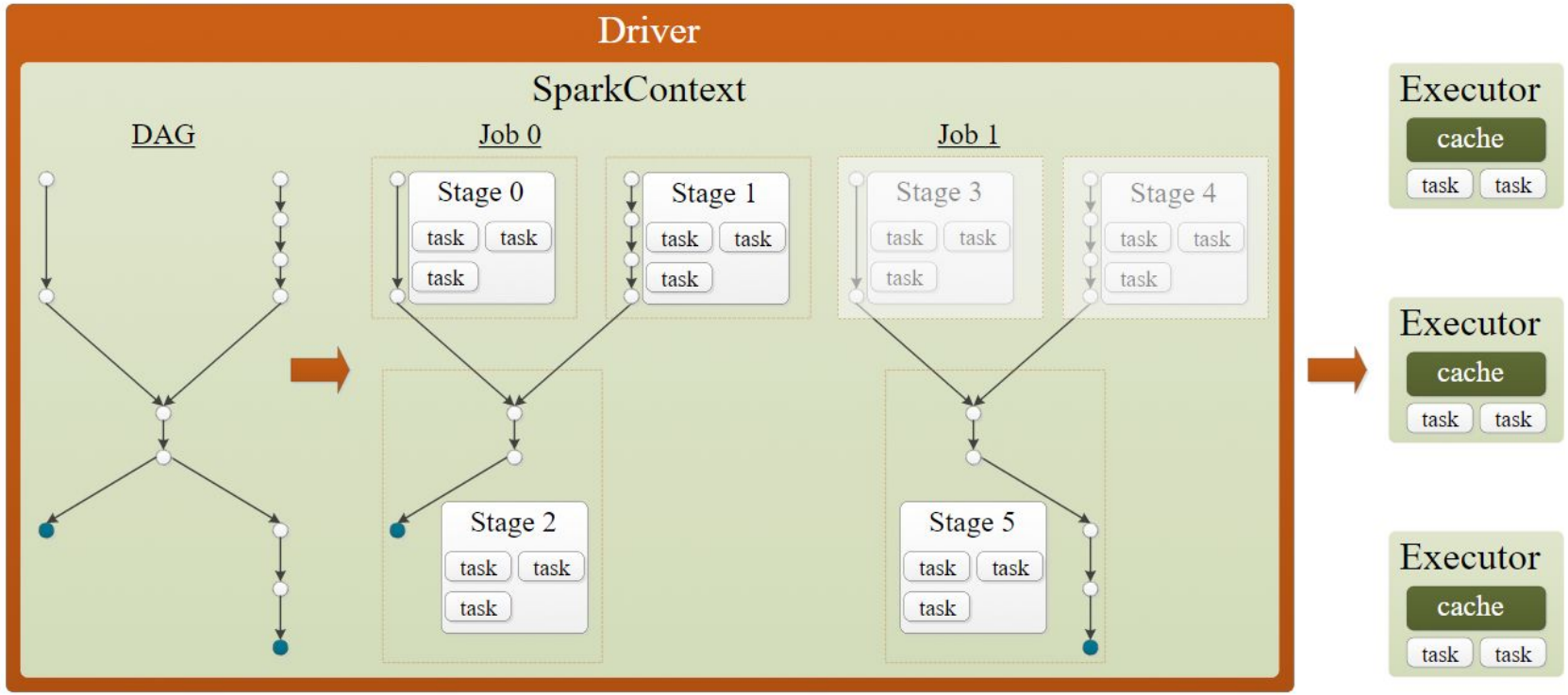
Spark works following a graph, this graph is called DAG which stands for Directed Acyclic Graph.

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



CLOUDERA

CLOUDERA

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

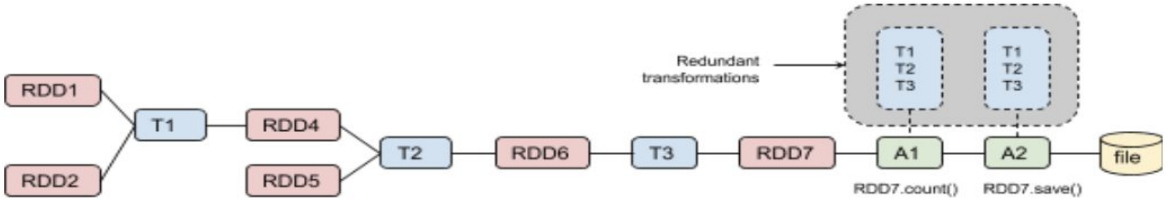
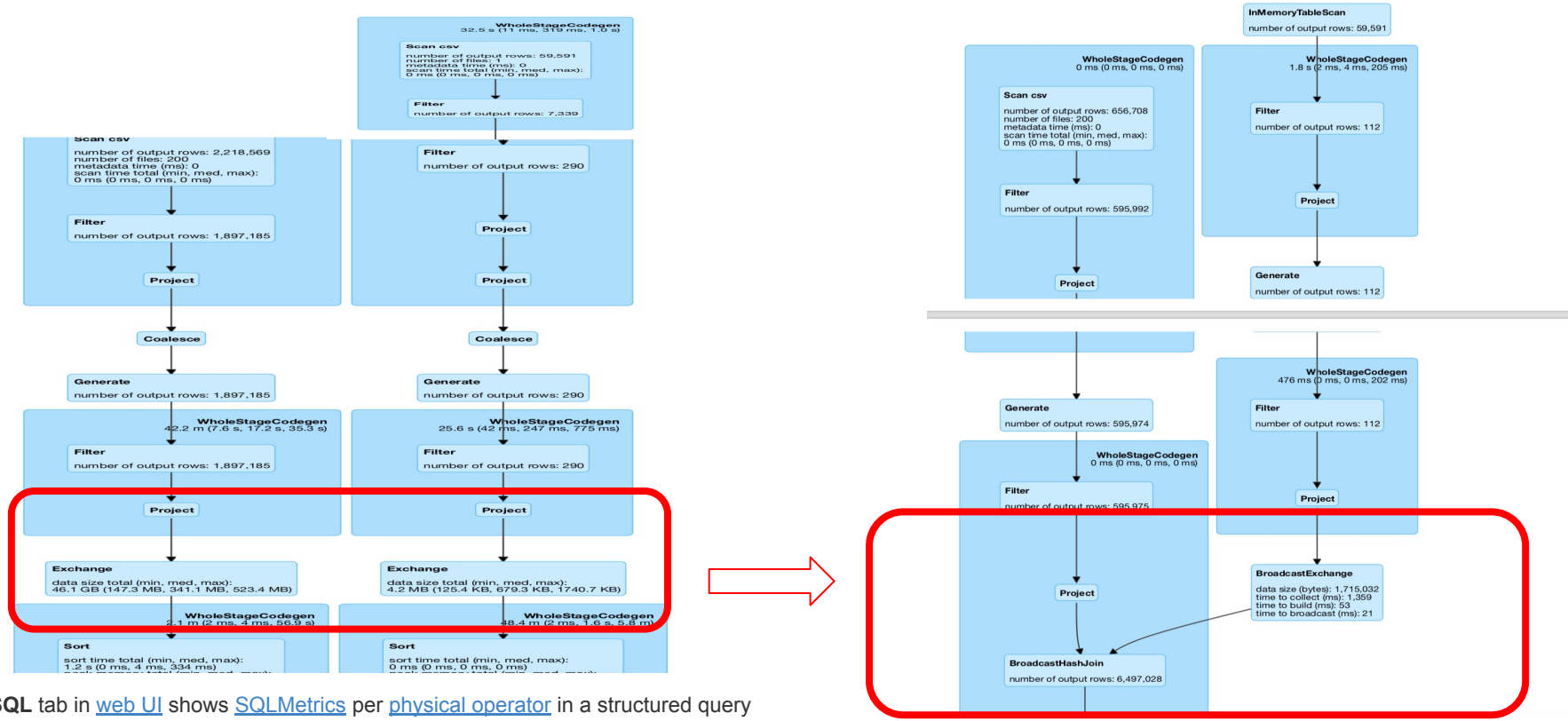


Fig1: Redundant transformations

CLOUDERA

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(UDF(fingerprint#7)))) AS
norm_fingerprint#236, UDF(UDF(UDF(UDF(UDF(fingerprint#7))))) AS lsh#297]
                                 :           +- *Filter ((AtLeastNNulls(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.prq], 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(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 ..

```
+- 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(UDF(fingerprint#221)))) AS
norm_fingerprint#329, cpyKey#219, deviceId#220, UDF(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

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

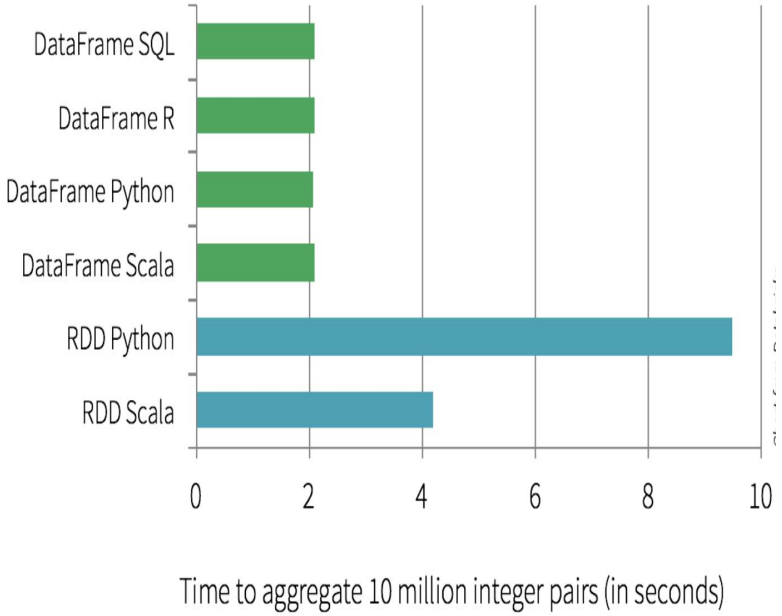
File Format	Query Time (sec)	Size (GB)
CSV	2892.3	437.46
Parquet: LZO	50.6	55.6
Parquet: Uncompressed	43.4	138.54
Parquet: GZIP	40.3	36.78
Parquet: Snappy	28.9	54.83

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.



CLOUDERA

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')
```

CLOUDERA

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 ¿podríais buscar alternativas para los servicios de emergencia en periodos de congestión?


1

Gracias

CLOUDERA