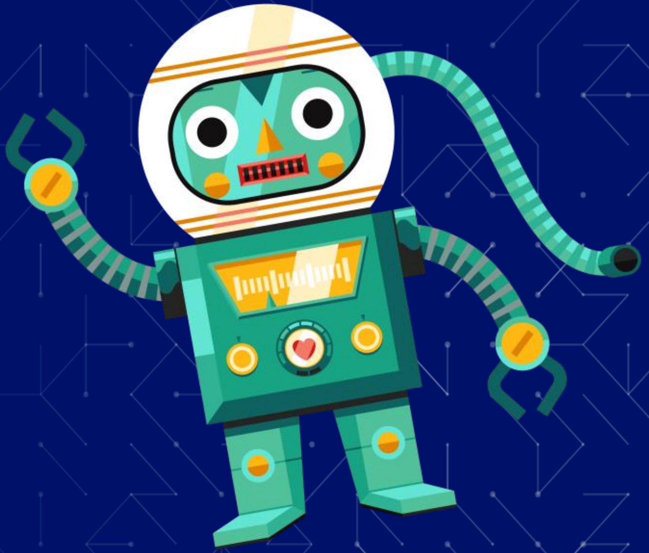


TECHORAMA

ADVANCING ANALYTICS

PYTHON PIPELINE PRIMER

ADDING A LITTLE SPARK TO YOUR WAREHOUSE PROCESS

Simon Whiteley
@MrSiWhiteley

MVP Microsoft® Most Valuable Professional https://github.com/SiWhiteley/DatabricksETL

Agenda

- What is Databricks?
- Patterns & Implementation
- Orchestration
- The Big Picture

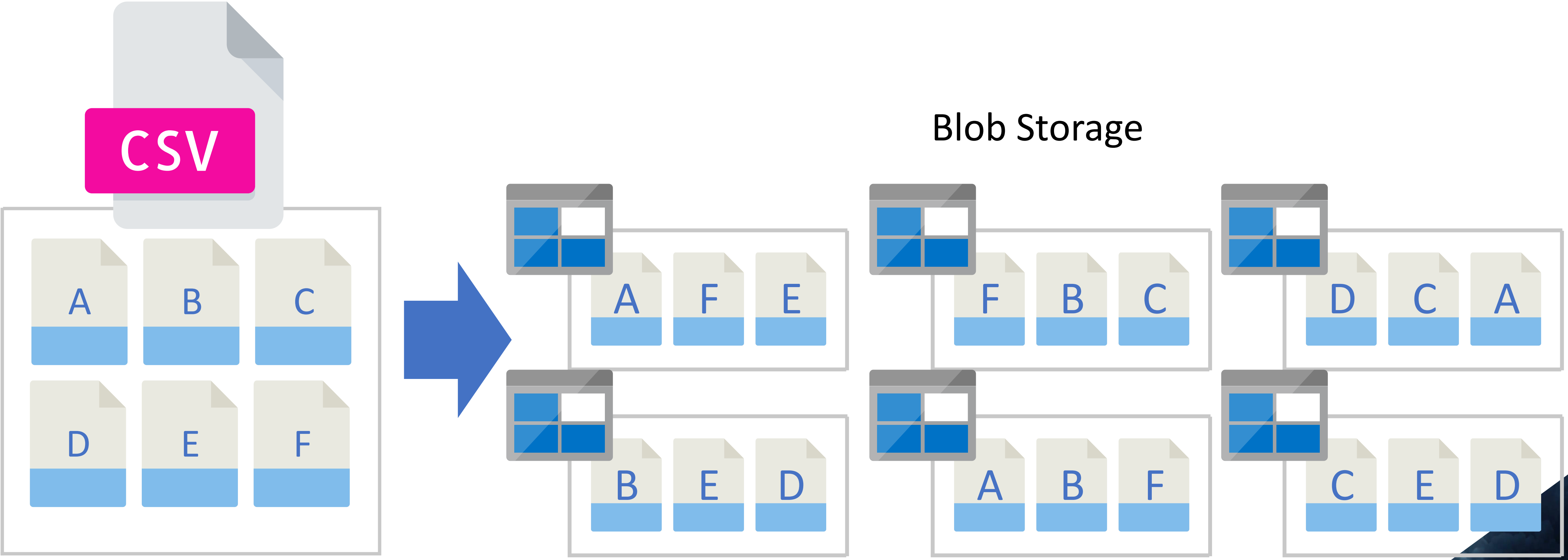
A HISTORY OF SPARK

Google

Google File System Papers Released

2003

HDFS IN AZURE



File Extents

PYTHON PIPELINE PRIMER

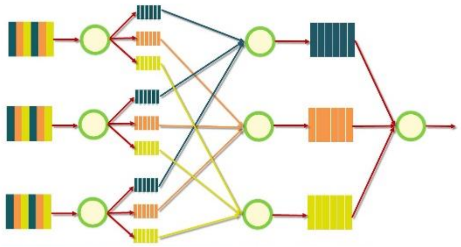
Google

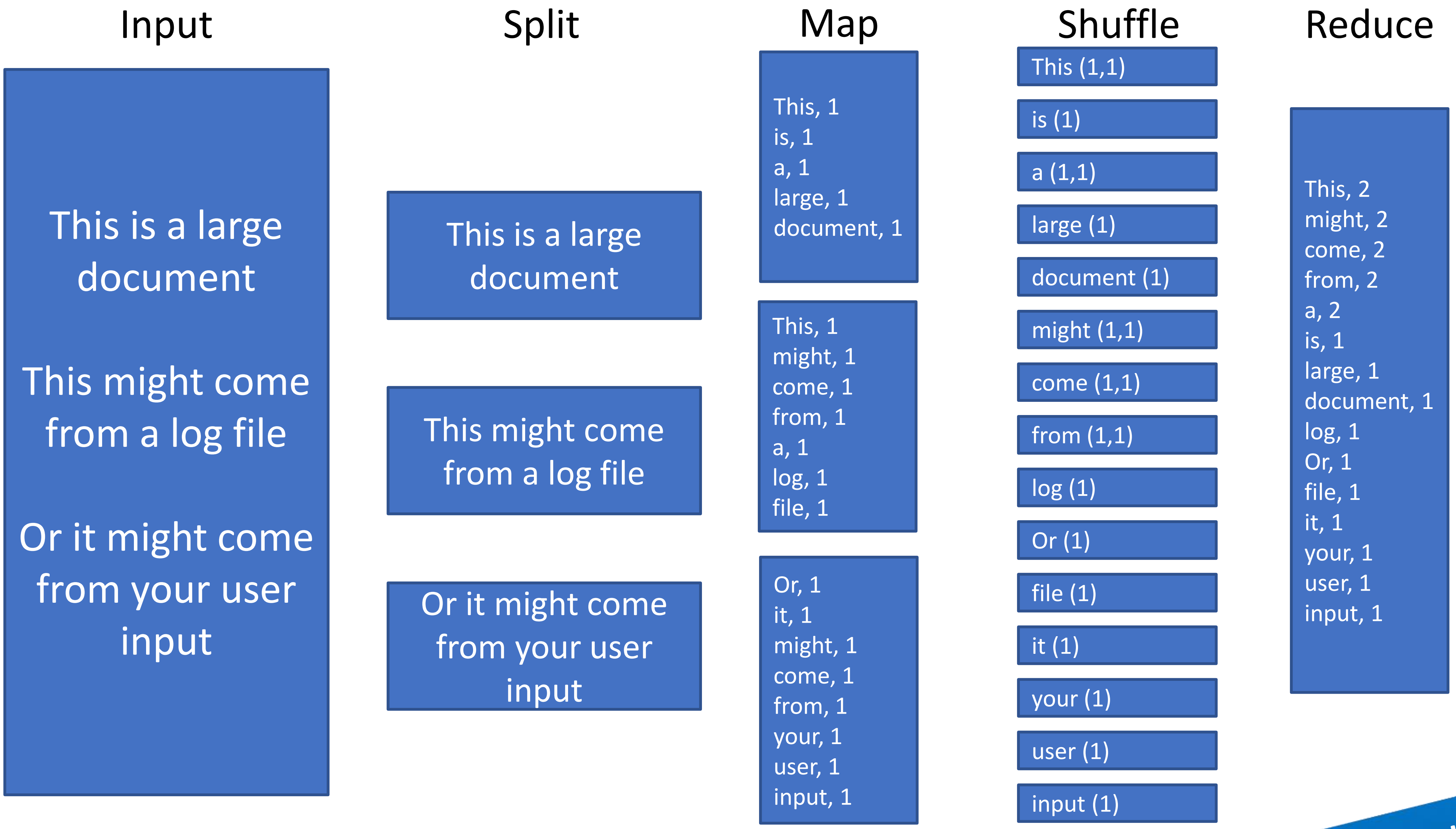
Google File System Papers Released

2003

Google MapReduce Papers

2004





Input

This is a large document

This might come from a log file

Or it might come from your user input

Split

This is a large document

This might come from a log file

Or it might come from your user input

Map

This, 1
is, 1
a, 1
large, 1
document, 1

This, 1
might, 1
come, 1
from, 1
a, 1
log, 1
file, 1

Or, 1
it, 1
might, 1
come, 1
from, 1
your, 1
user, 1
input, 1

Shuffle

This (1,1)
is (1)
a (1,1)
large (1)
document (1)
might (1,1)
come (1,1)
from (1,1)
log (1)
Or (1)
file (1)
it (1)
your (1)
user (1)
input (1)

Reduce

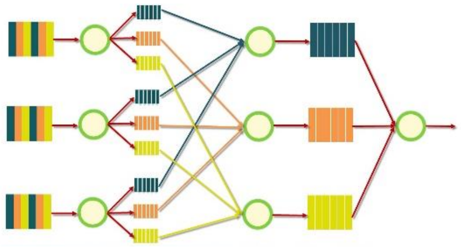
This, 2
might, 2
come, 2
from, 2
a, 2
is, 1
large, 1
document, 1
log, 1
Or, 1
file, 1
it, 1
your, 1
user, 1
input, 1

Google File System Papers Released

2003

Google MapReduce Papers

2004



2006

Apache Hadoop project created

Matei Zaharia starts Spark project

2012

Project donated to Apache Foundation

2013

Databricks founded by Matei

2013

Databricks is…

Apache Spark**, built by the guys who wrote Spark, made **super easy

amazon
web services

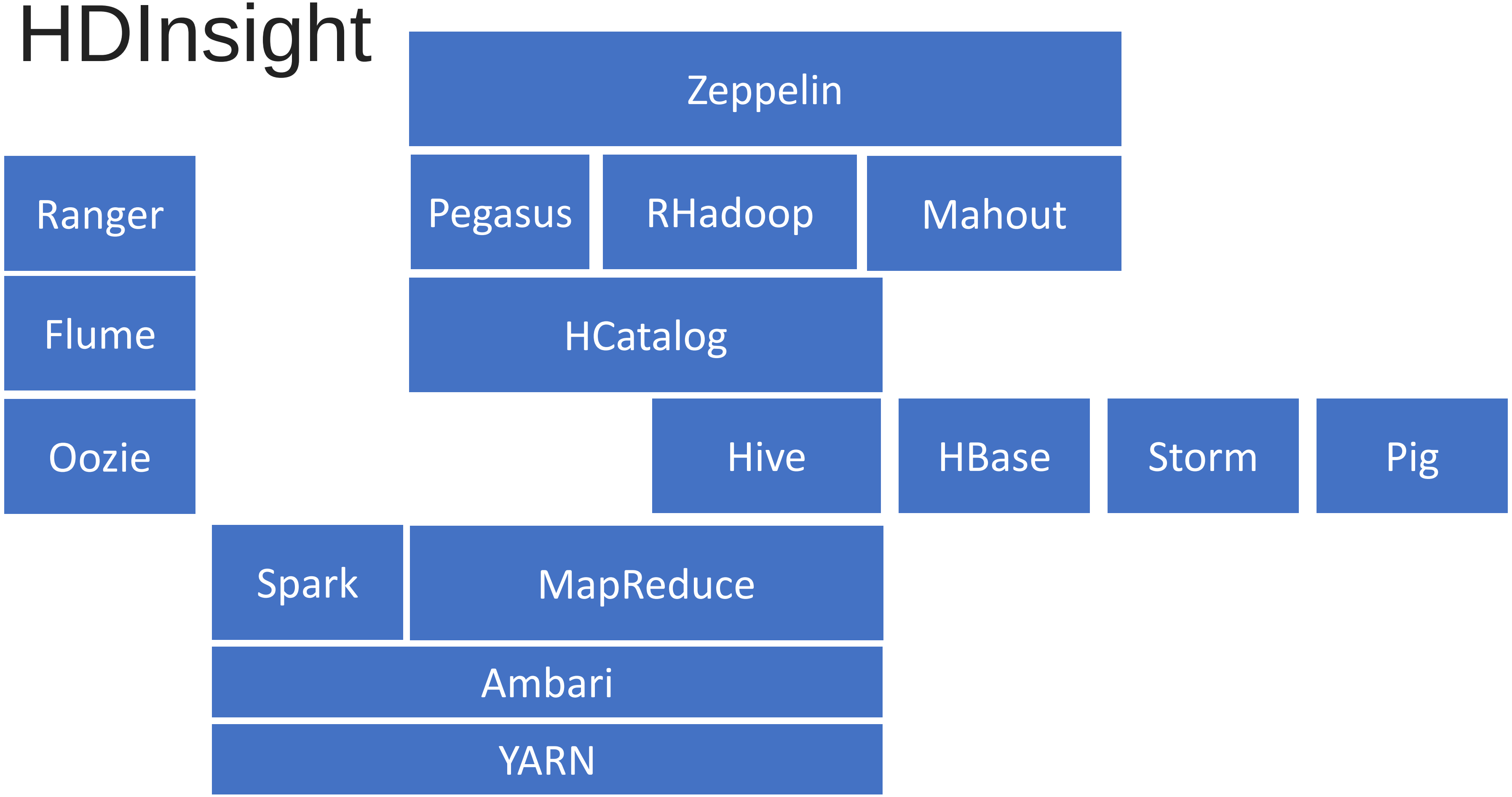
2016

It’s new to Azure, not to everyone else!

Microsoft
Azure

2018

HDInsight



Zeppelin

Ranger

Pegasus

RHadoop

Mahout

Flume

HCatalog

Oozie

Hive

HBase

Storm

Pig

Spark

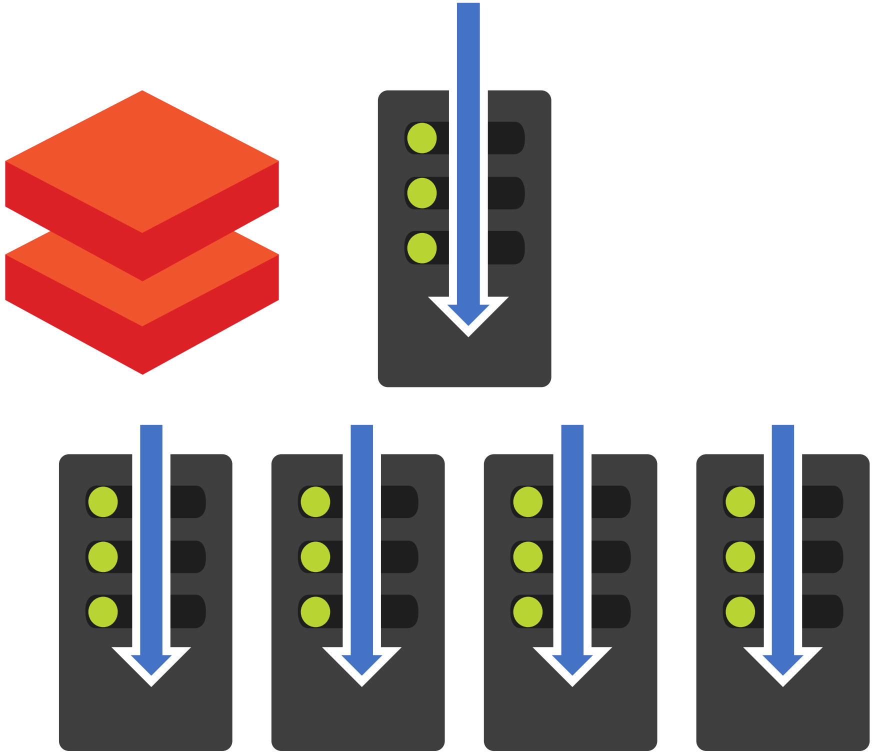
MapReduce

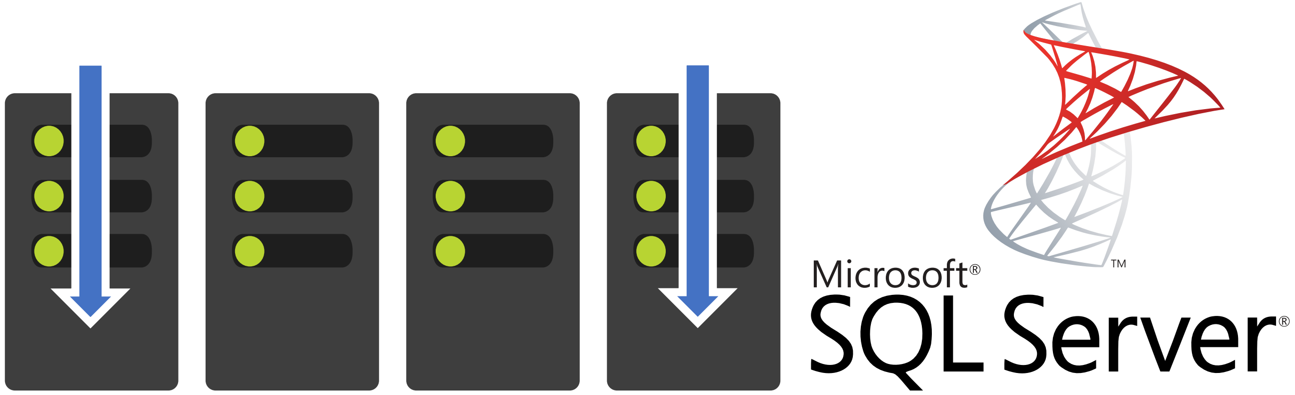
Ambari

YARN

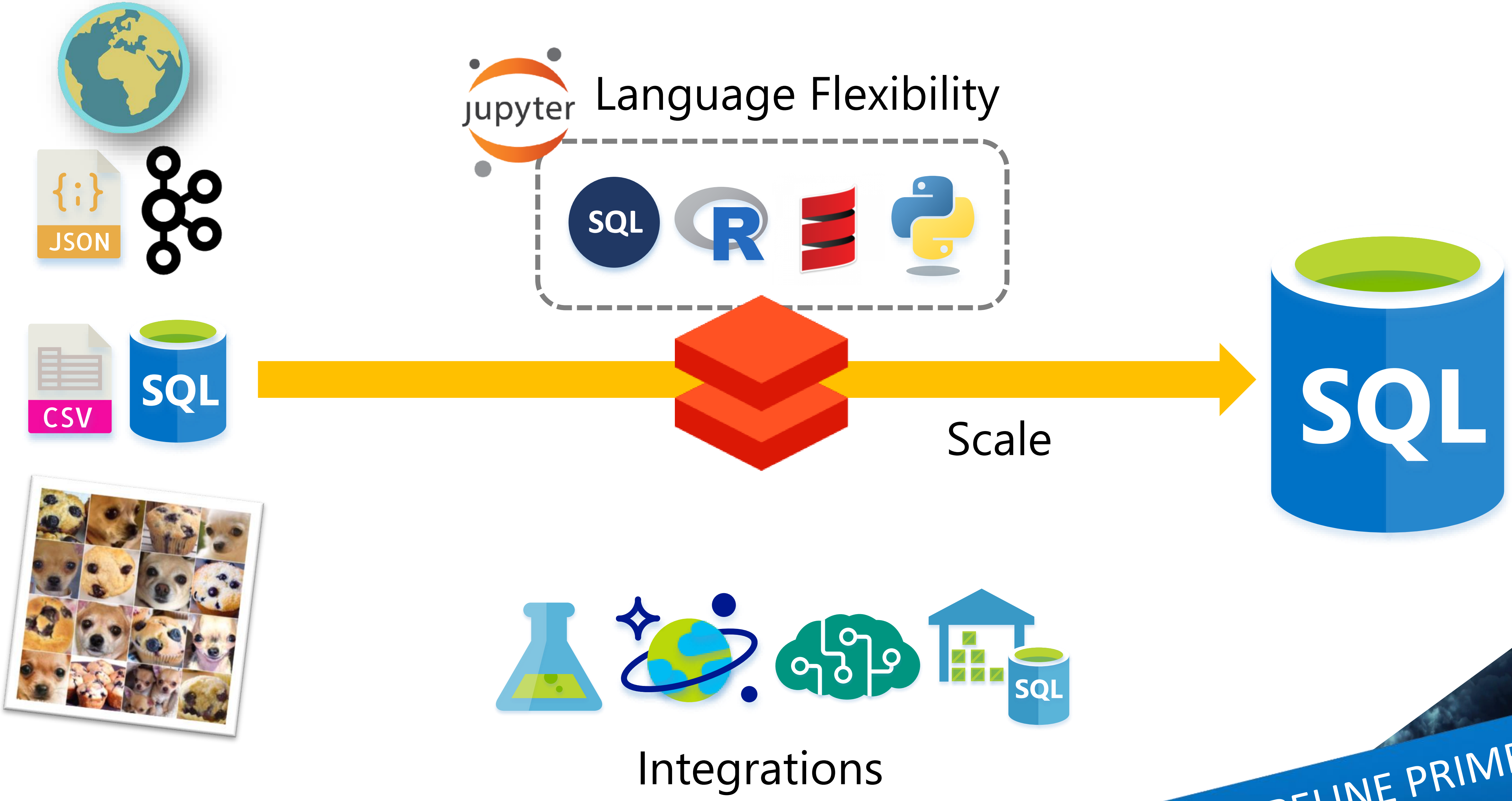
Open Source	Open Source	~~Proprietary~~
20 min provisioning	5 min provisioning	~~1 min provisioning~~
Integrates Well	Integrates Well	~~Integrates Poorly~~
Secure	Secure	~~Secure~~
Hadoop, Spark, Kafka, Hbase, HIVE, Storm…	Spark (Python/Scala/R)	~~U-SQL~~
Slow Release Cycle	Fast Release Cycle	~~Slow Release Cycle~~

1. Scale
2. Flexibility





WHAT PROBLEMS DOES IT SOLVE?



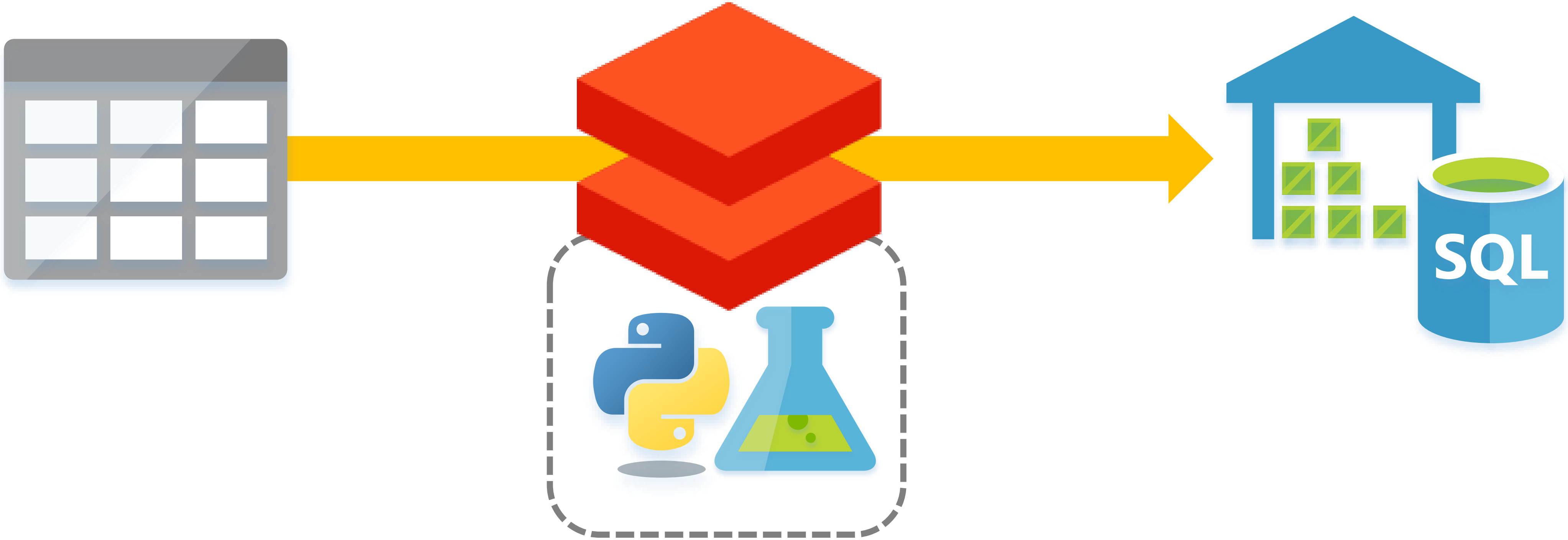
Language Flexibility

Scale

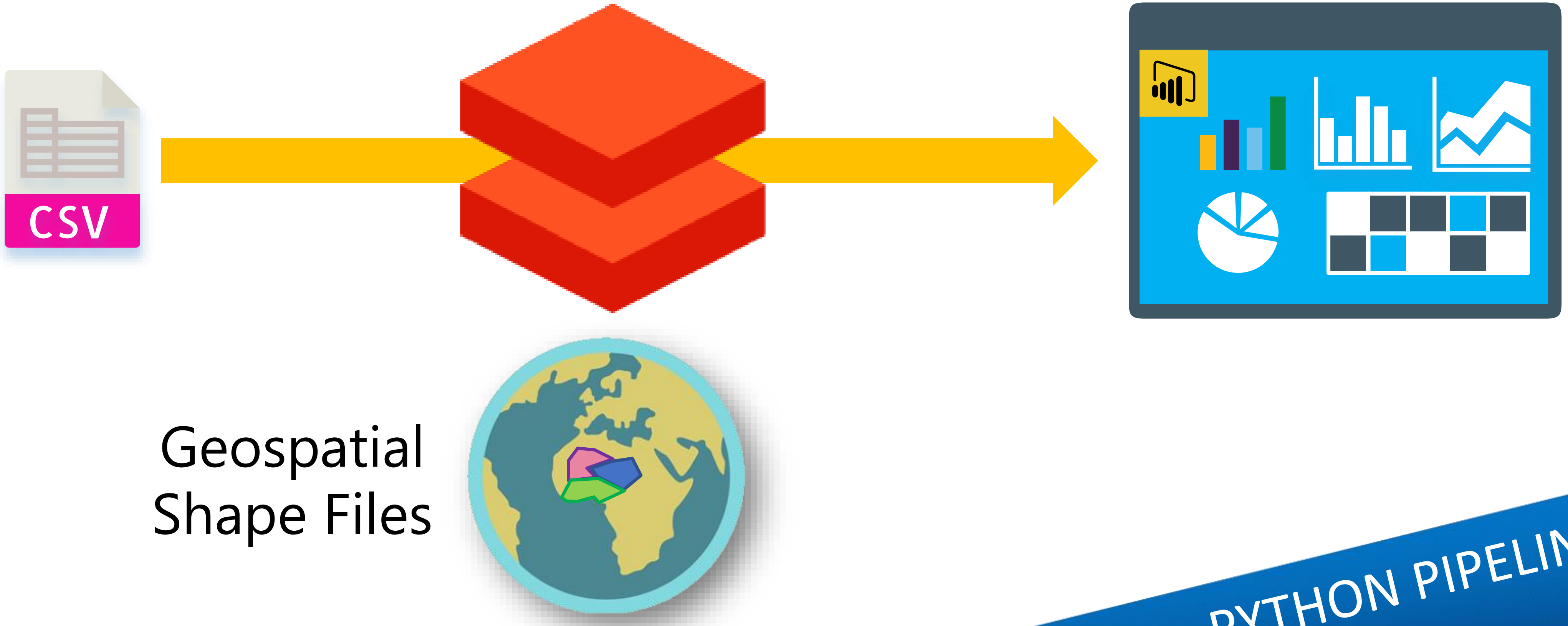
Integrations

File Types

MACHINE LEARNING AT SCALE

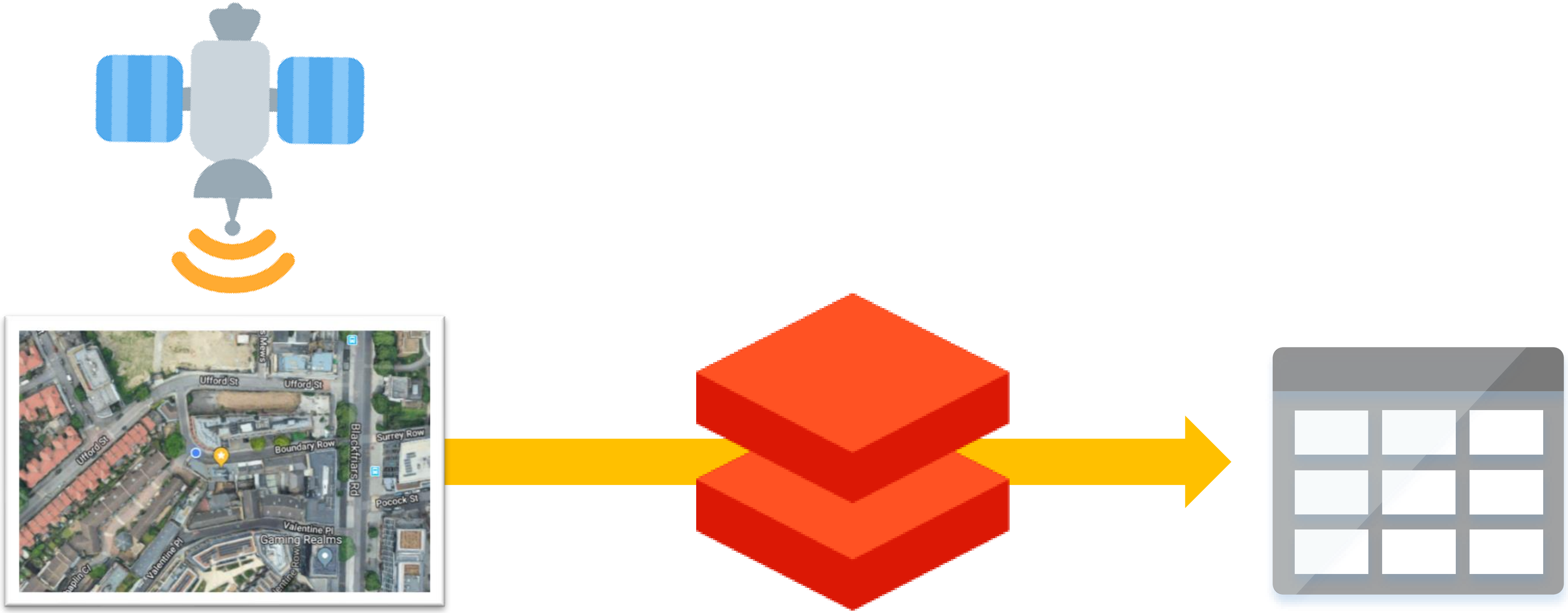


GEOSPATIAL MAPPING

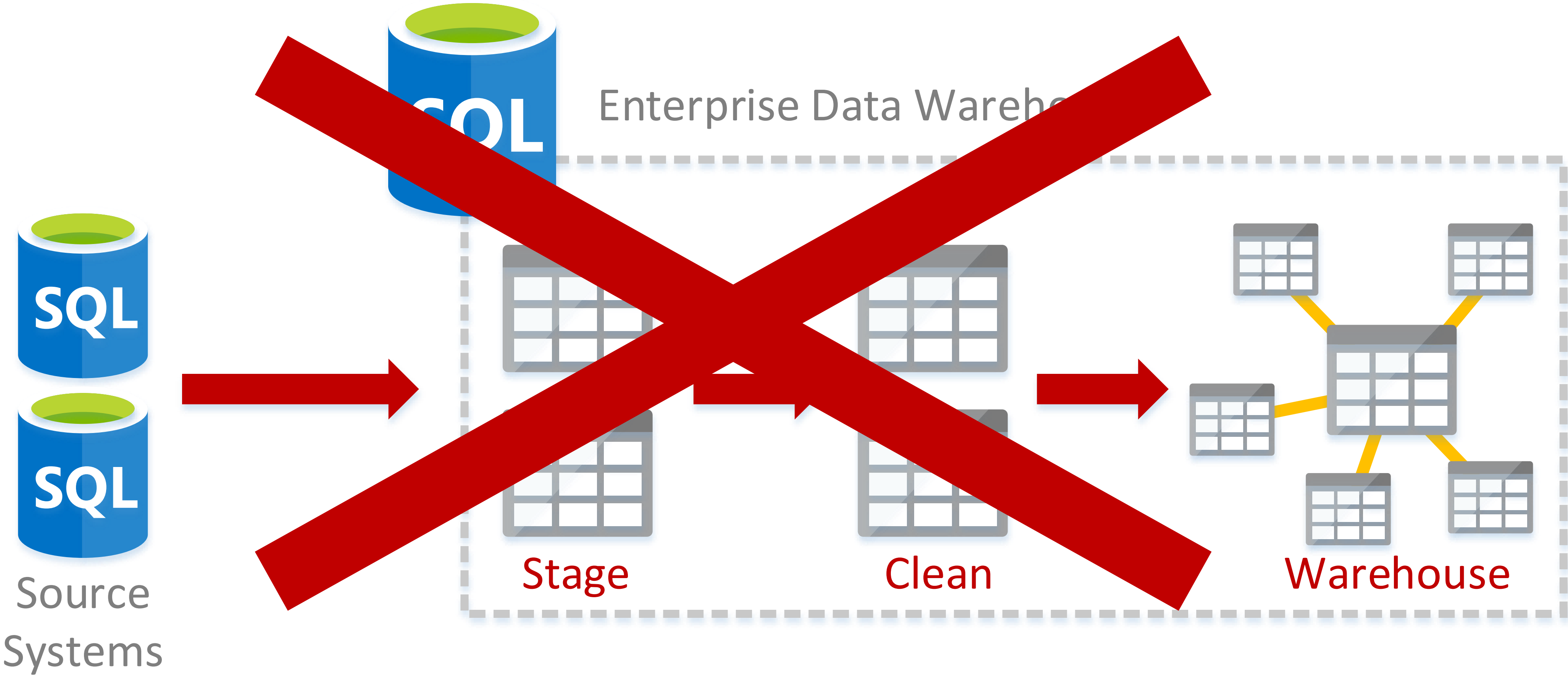


Geospatial Shape Files

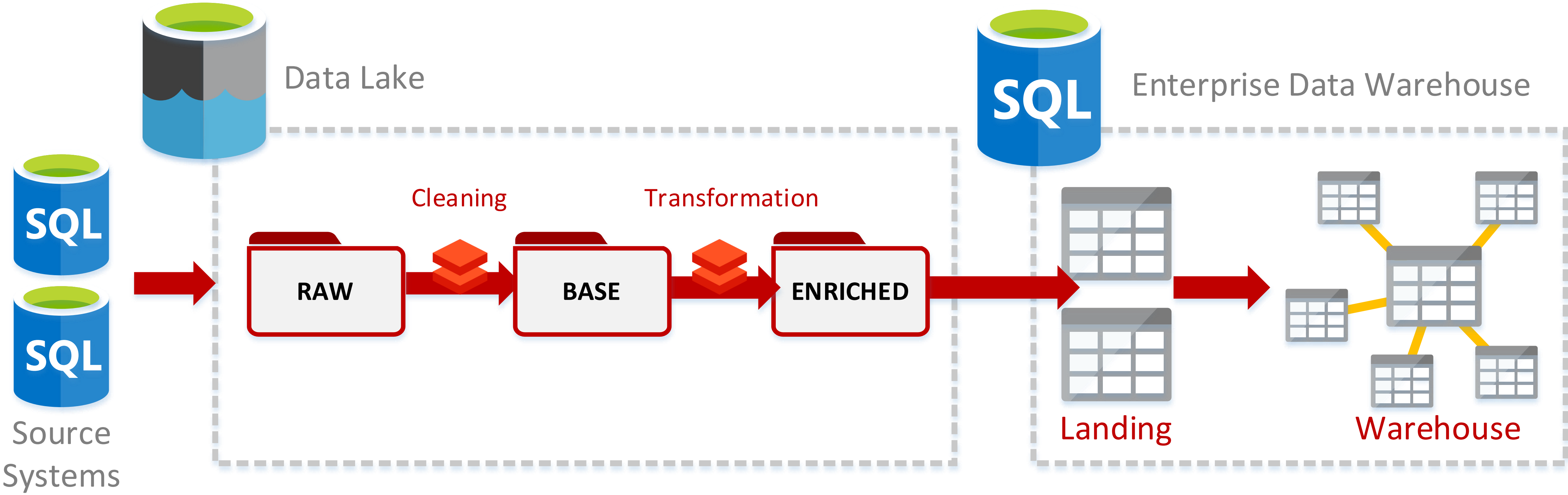
IMAGE PROCESSING



BUT MOST OF ALL...



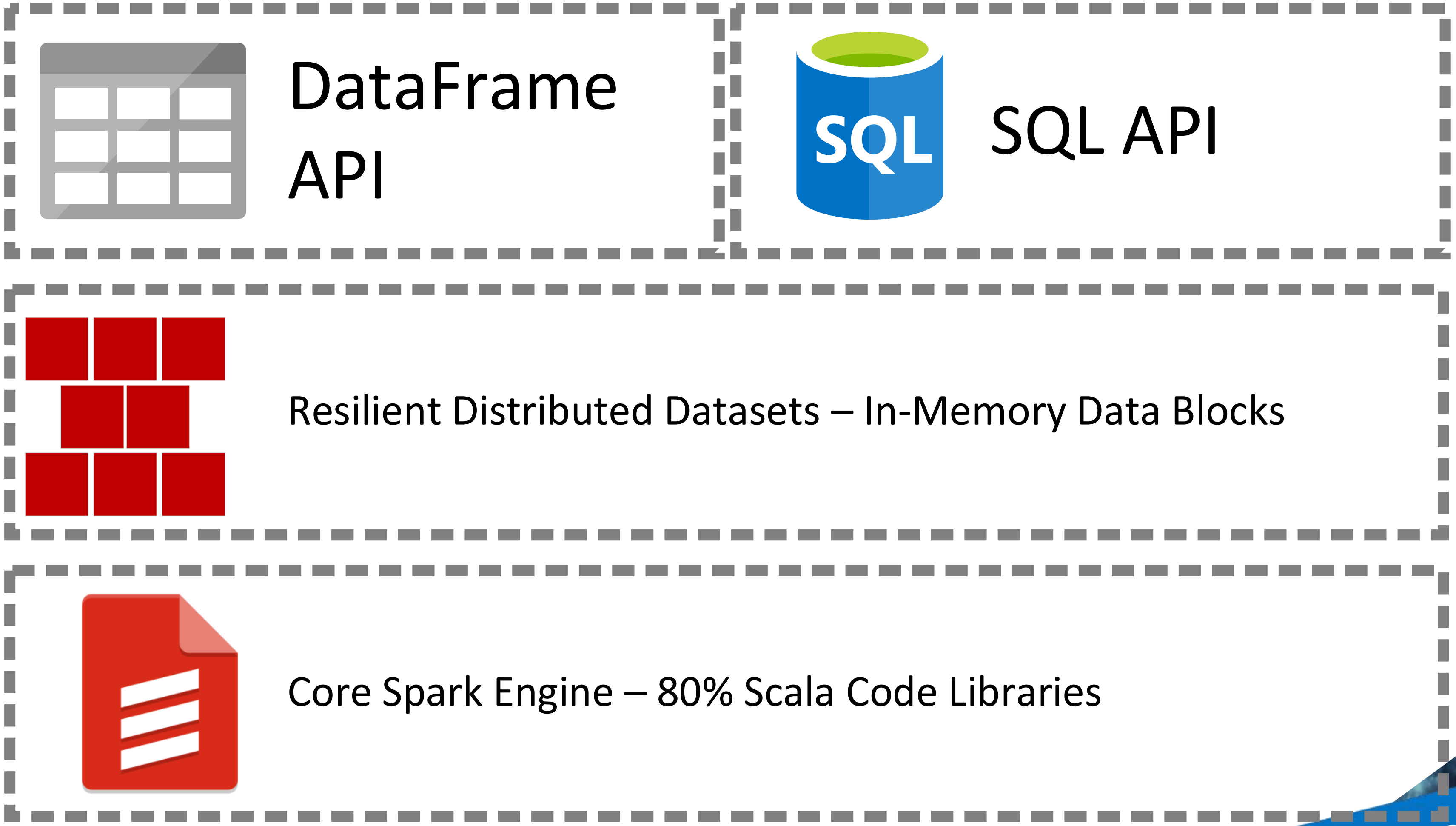
PYTHON PIPELINE PRIMER



PYTHON PIPELINE PRIMER

DATABRICKS BASICS

UNDER THE HOOD



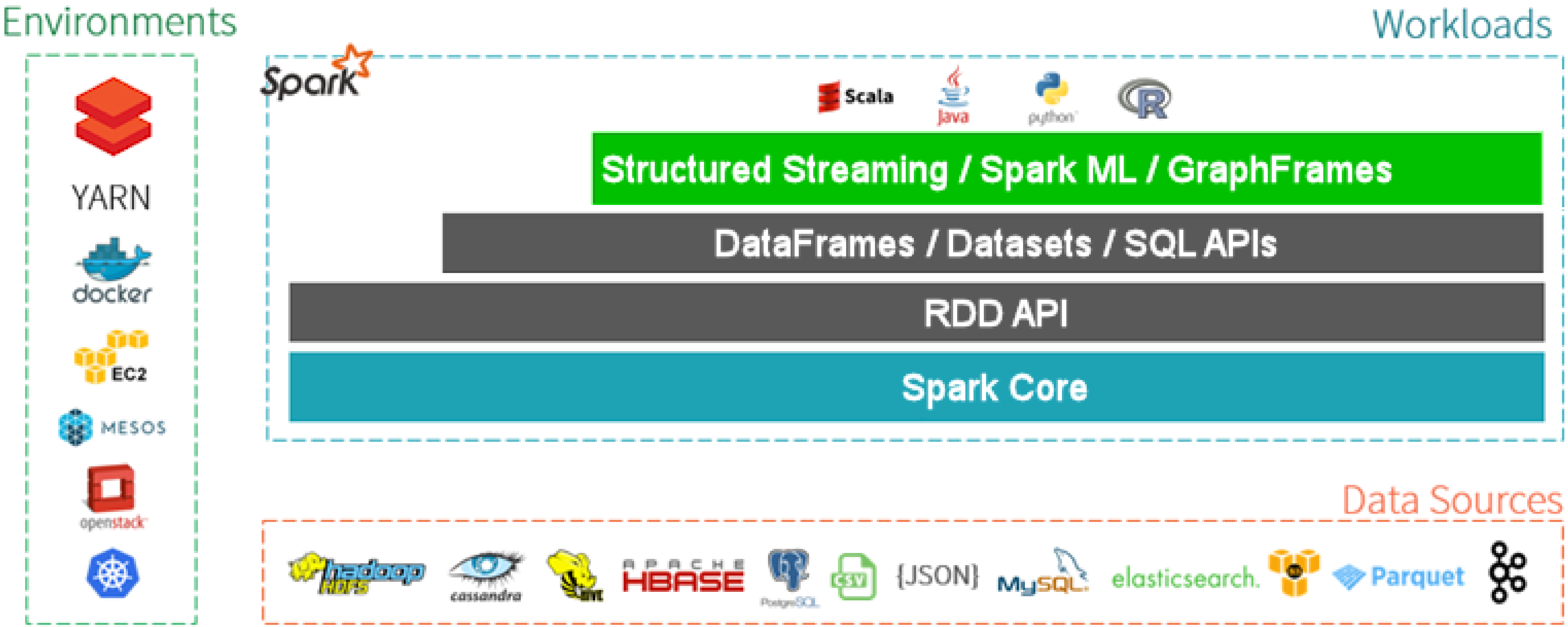
DataFrame API

SQL API

Resilient Distributed Datasets – In-Memory Data Blocks

Core Spark Engine – 80% Scala Code Libraries

UNDER THE HOOD



Environments

Workloads

Structured Streaming / Spark ML / GraphFrames

DataFrames / Datasets / SQL APIs

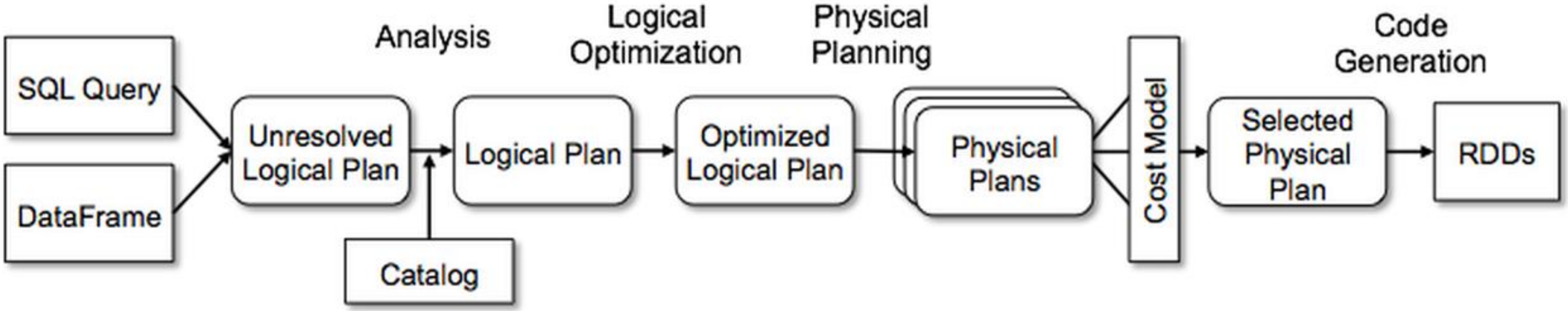
RDD API

Spark Core

Data Sources

LANGUAGE OPTIONS

THE CATALYST OPTIMISER



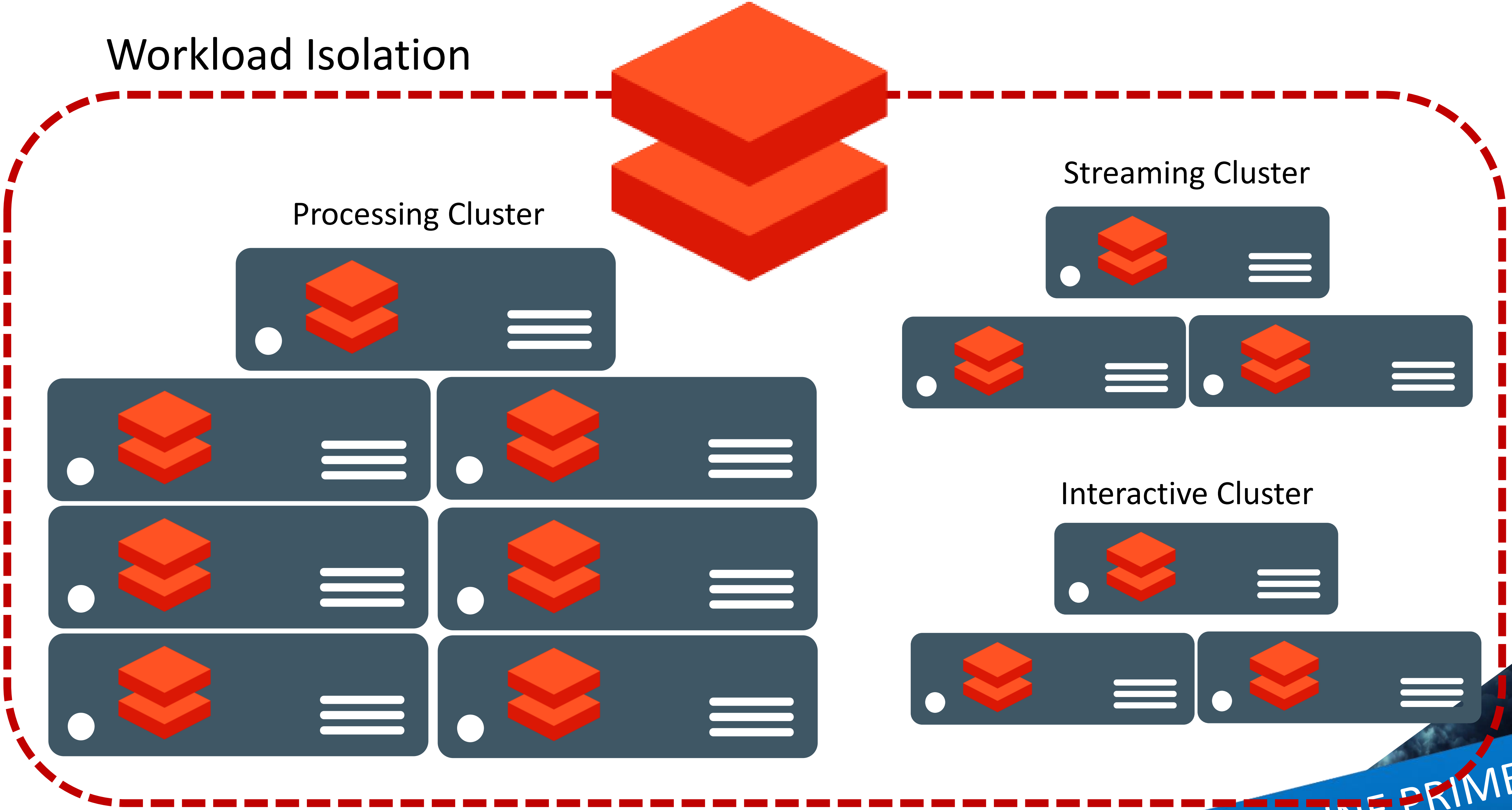
DataFrames → RDDs

DEMO: AZURE DATABRICKS

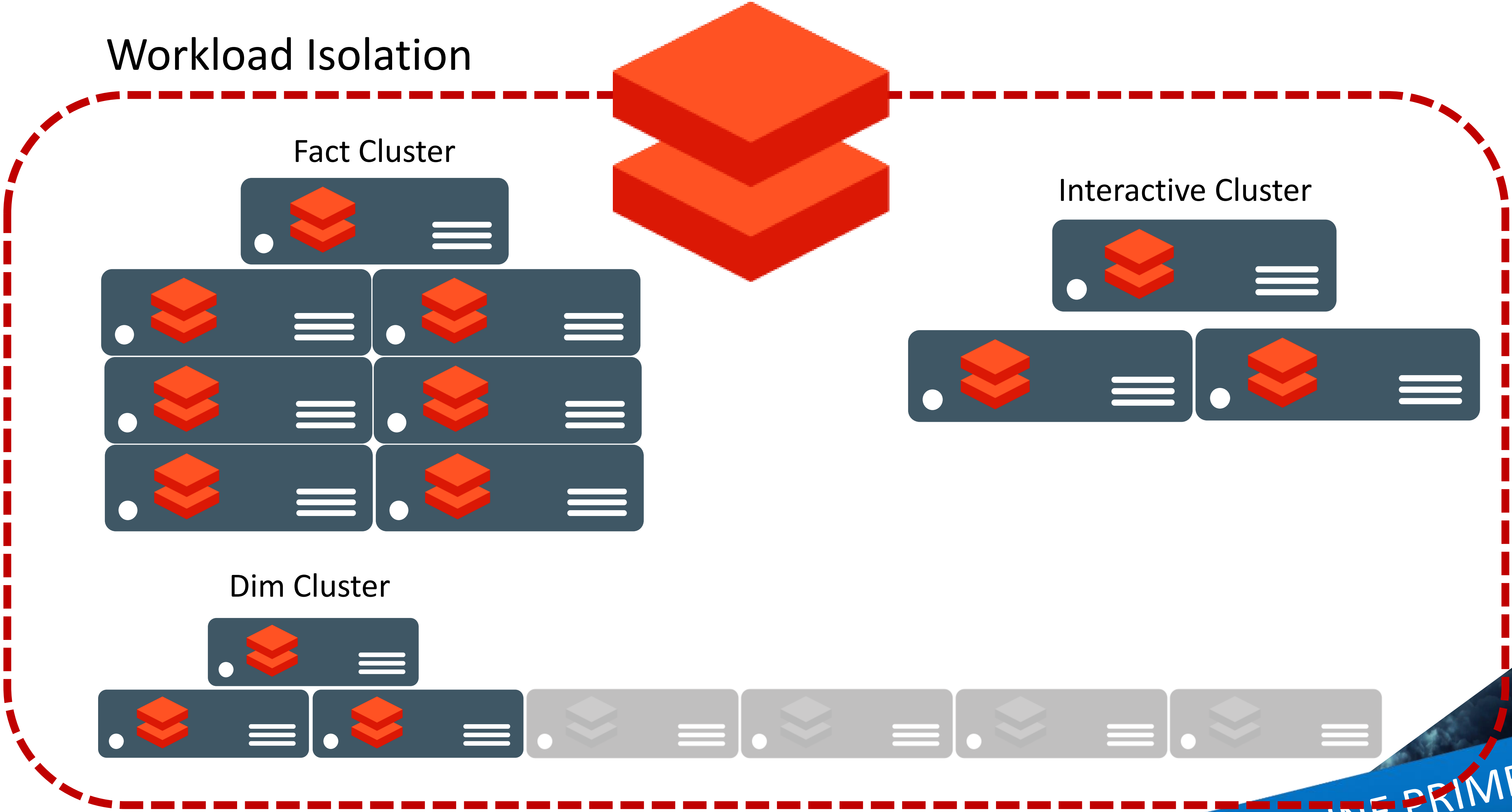
- Databricks Workspace
- Clusters

PATTERNS & IMPLEMENTATION

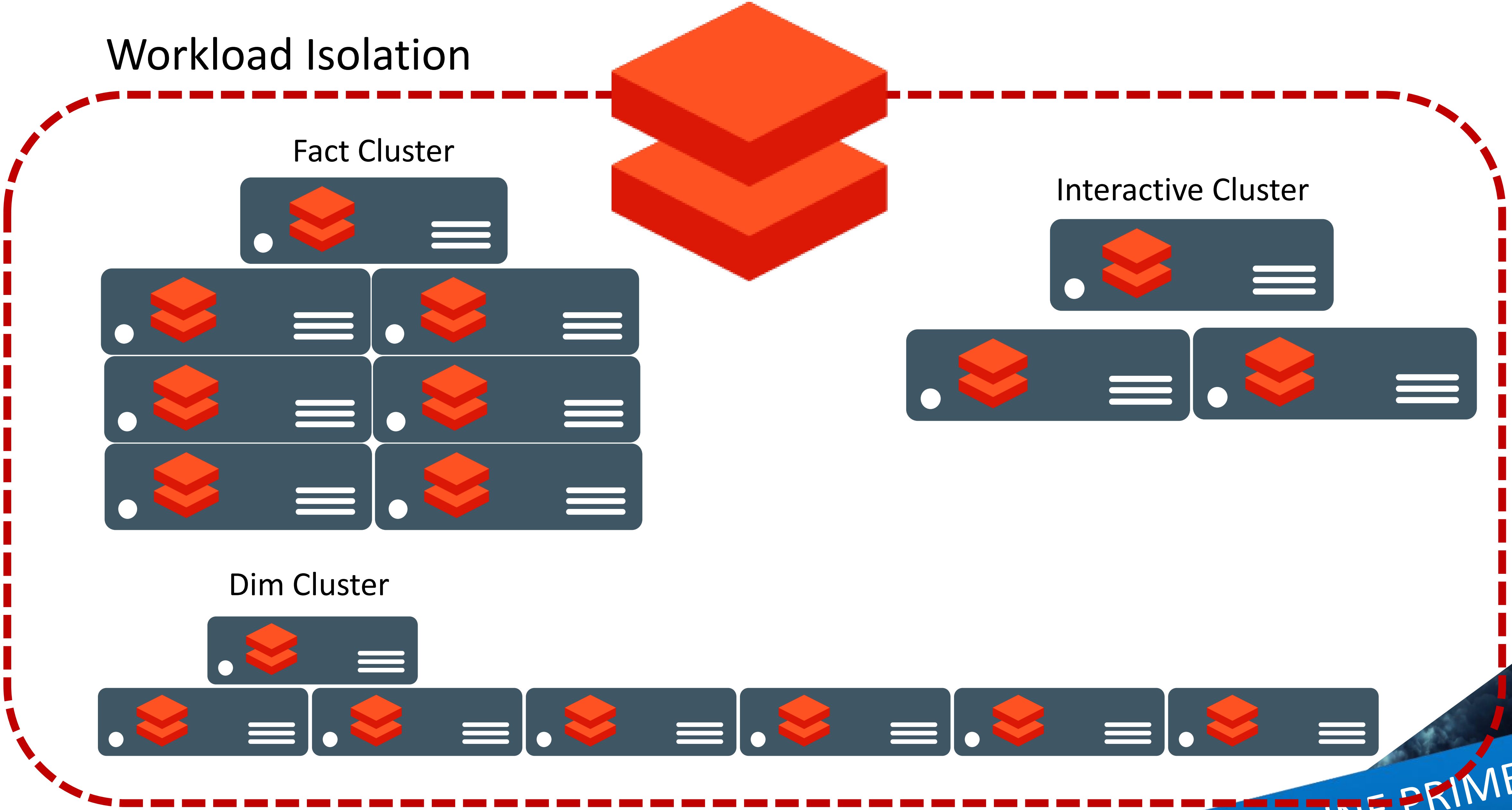
Workload Isolation



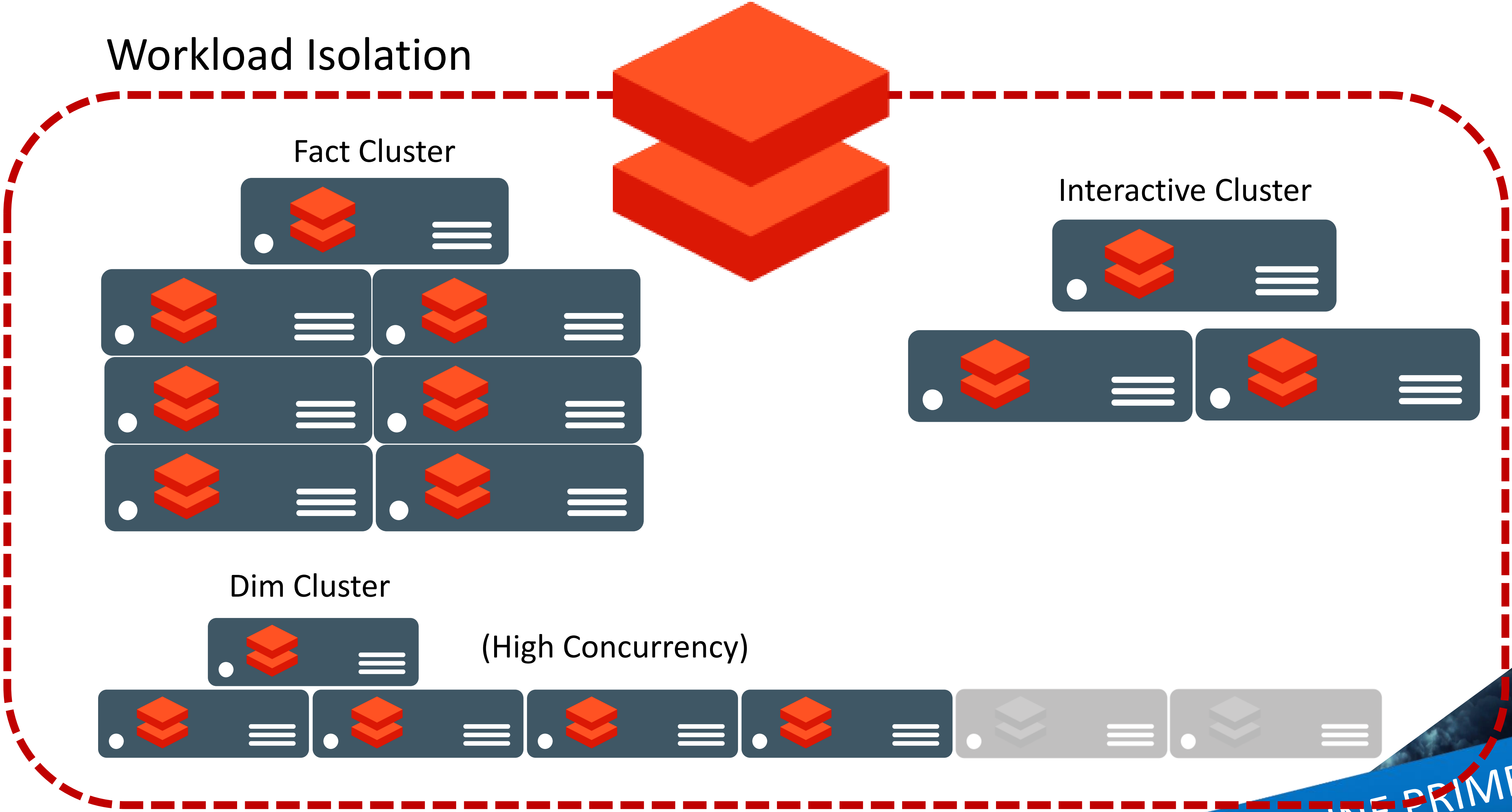
Workload Isolation



Workload Isolation

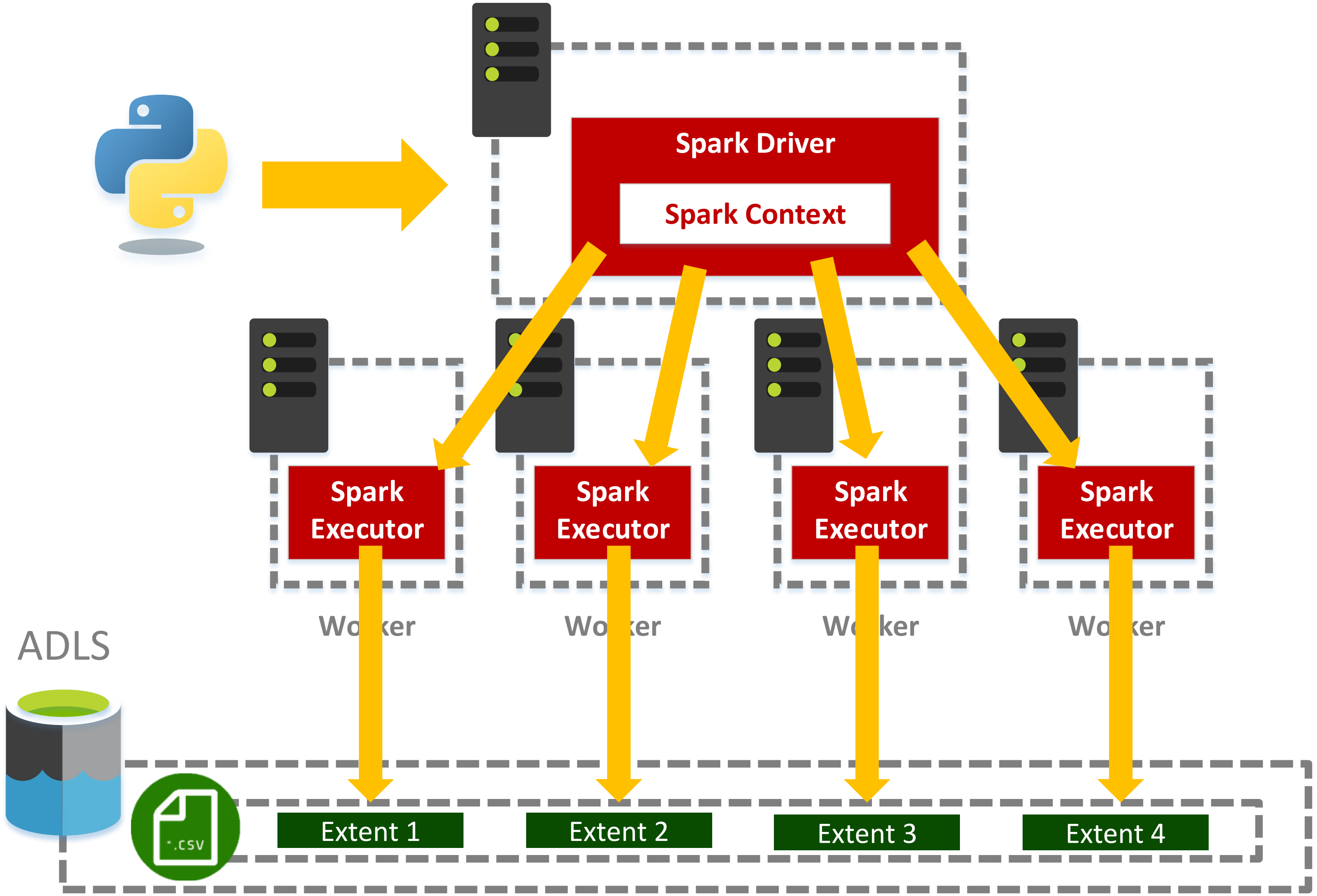


Workload Isolation



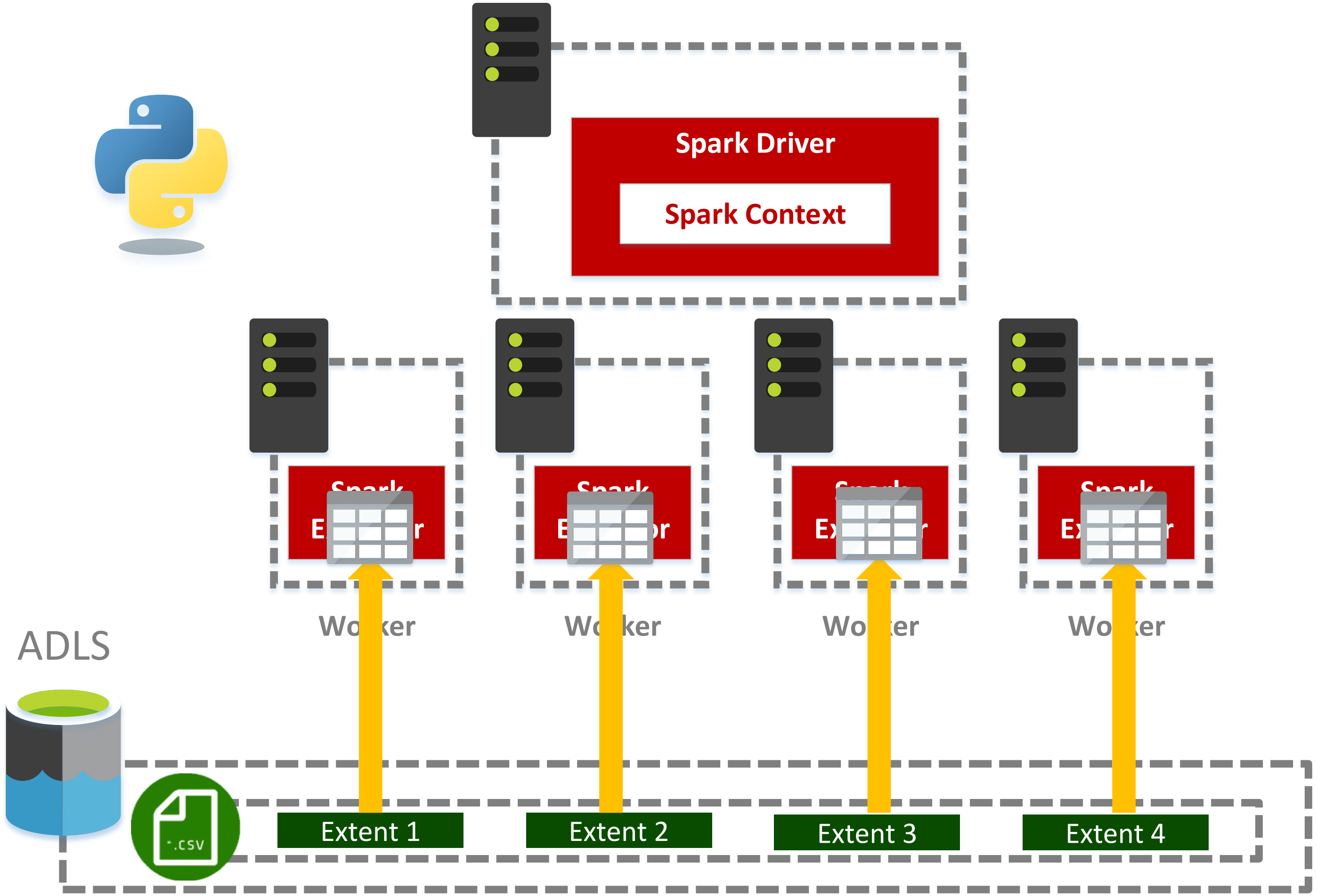
EXECUTIONS

DISTRIBUTED COMPUTE

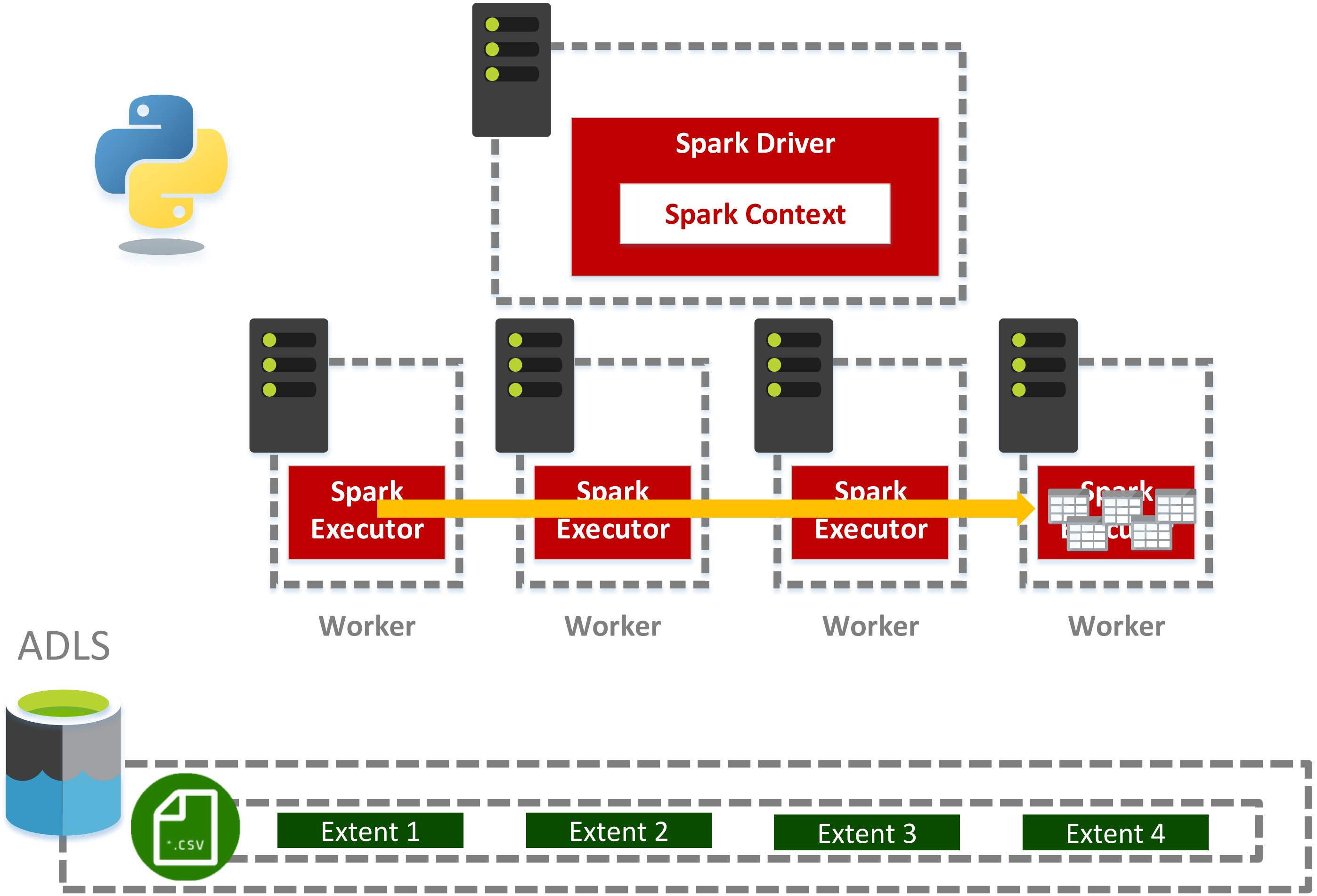


PYTHON PIPELINE PRIMER

DISTRIBUTED COMPUTE

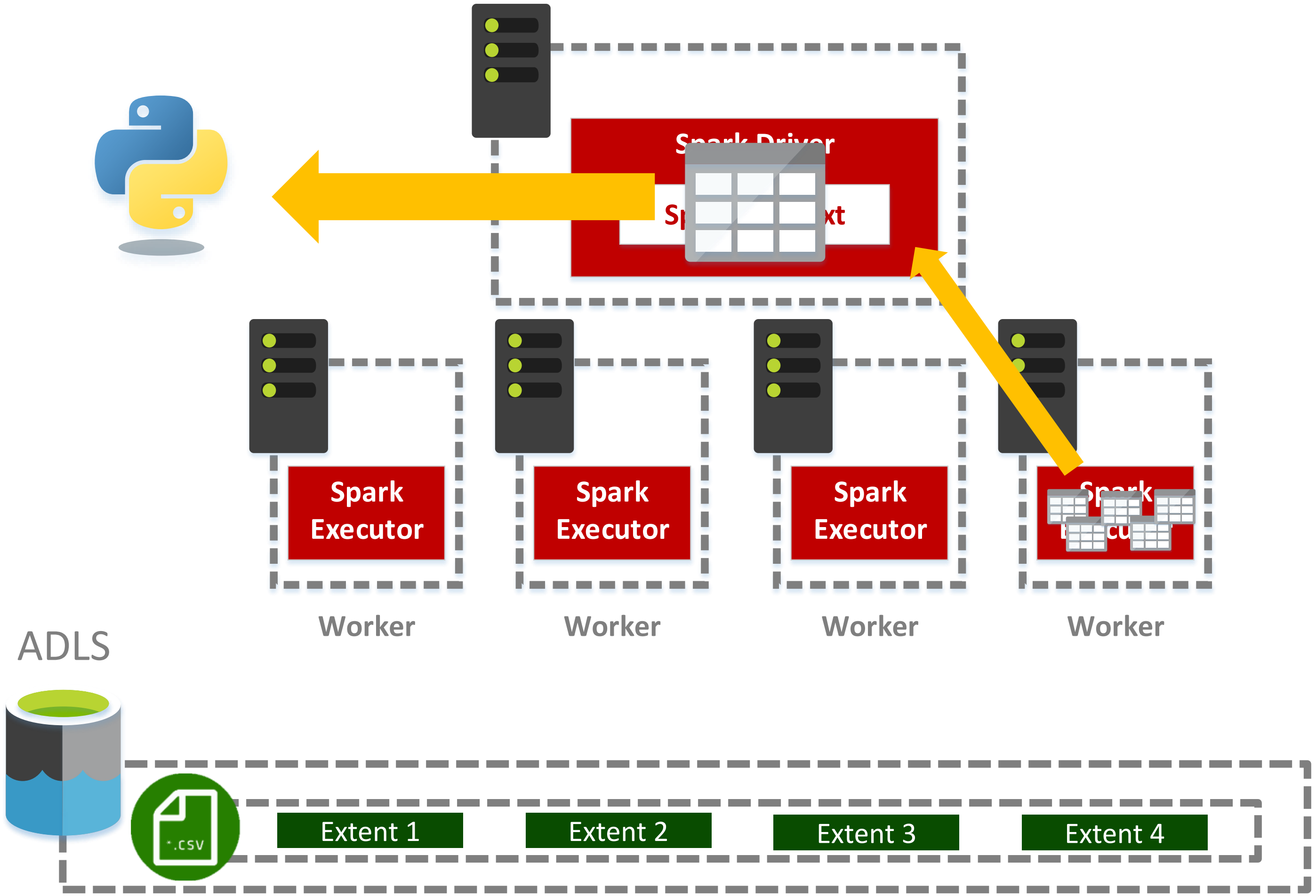


DISTRIBUTED COMPUTE



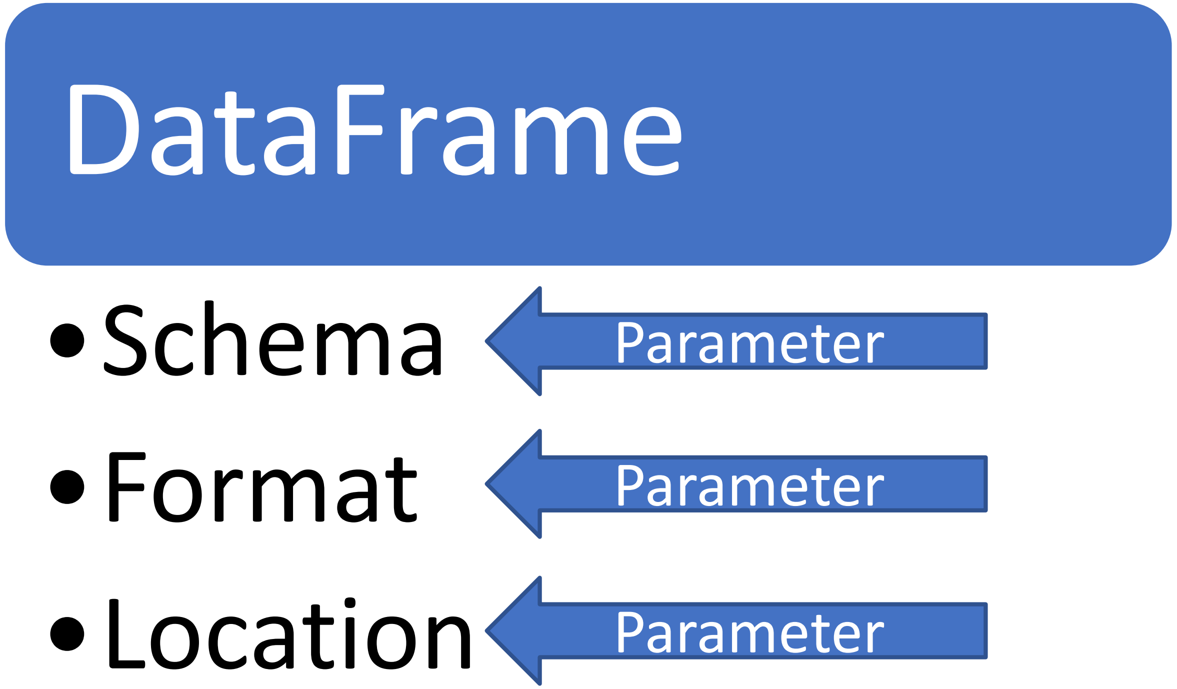
PYTHON PIPELINE PRIMER

DISTRIBUTED COMPUTE



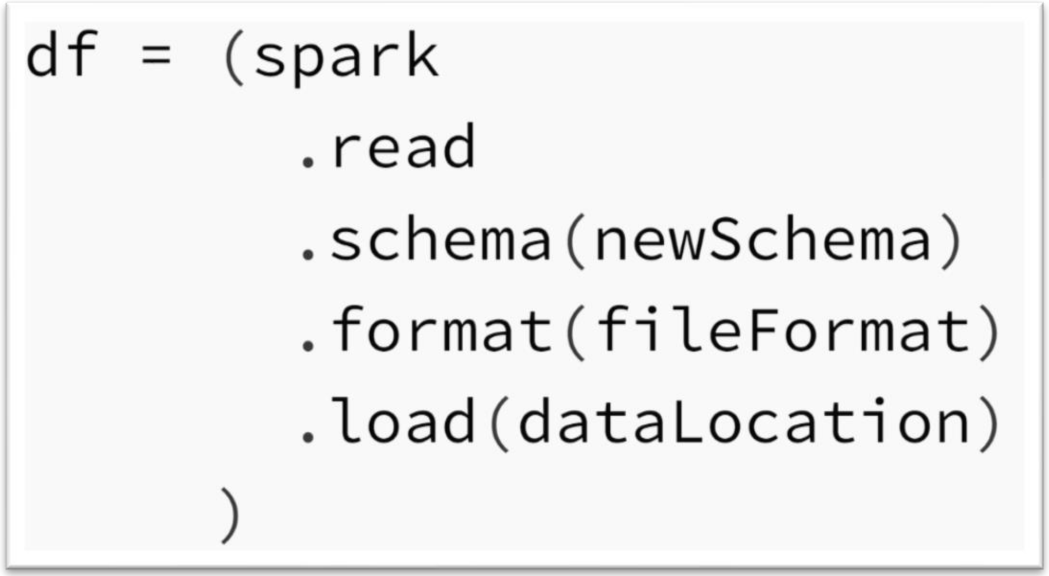
SO HOW DO WE USE IT?

THE DATA FRAME



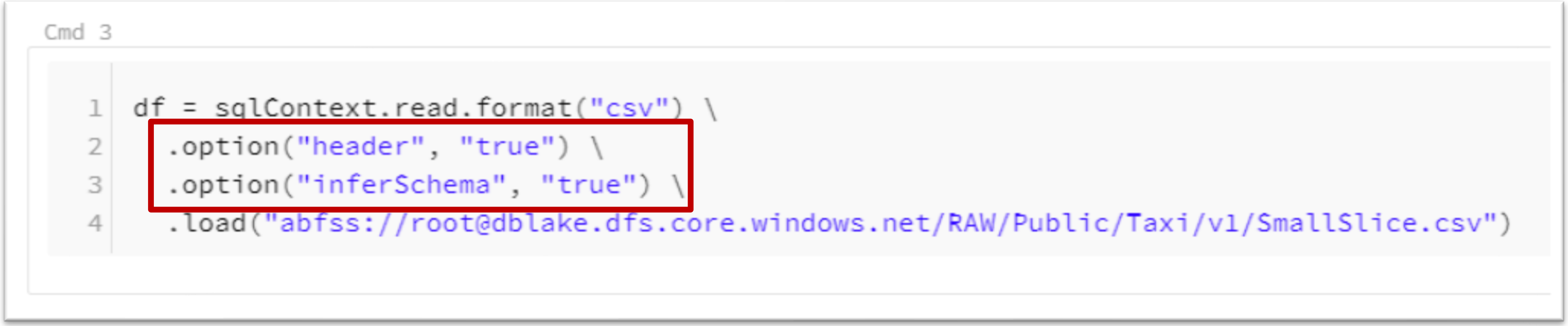
DataFrame

- Schema
- Format
- Location

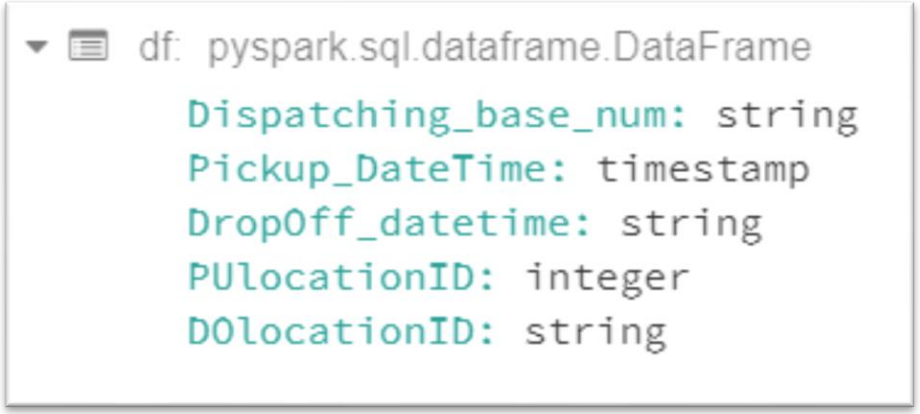


```
df = (spark
        .read
        .schema(newSchema)
        .format(fileFormat)
        .load(dataLocation)
     )
```

SCHEMA ON READ – INFER SCHEMA

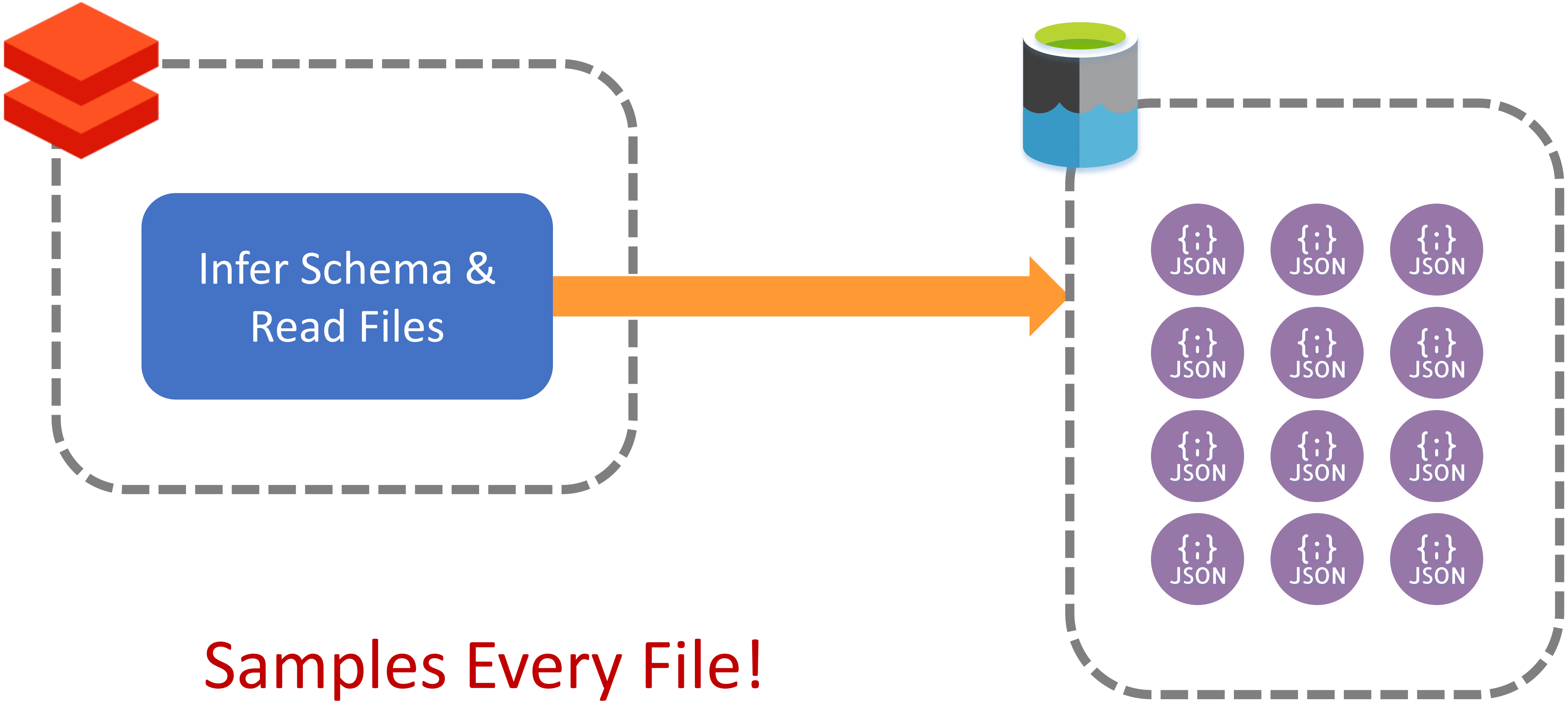


```
df = sqlContext.read.format("csv") \
  .option("header", "true") \
  .option("inferSchema", "true") \
  .load("abfss://root@dblake.dfs.core.windows.net/RAW/Public/Taxi/v1/SmallSlice.csv")
```



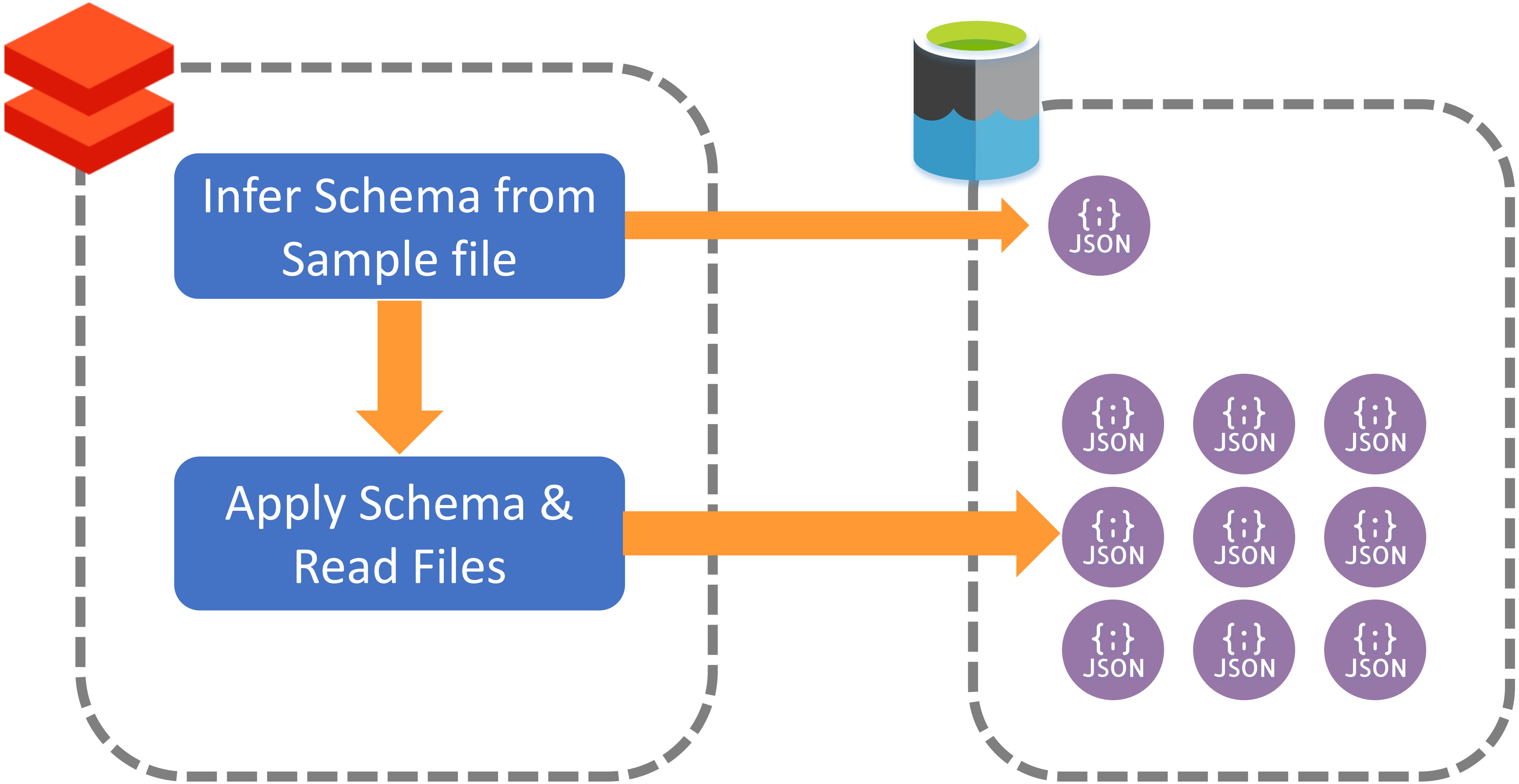
```
df:  pyspark.sql.dataframe.DataFrame
    Dispatching_base_num: string
    Pickup_DateTime: timestamp
    DropOff_datetime: string
    PUlocationID: integer
    DOlocationID: string
```

SCHEMA ON READ – INFER SCHEMA



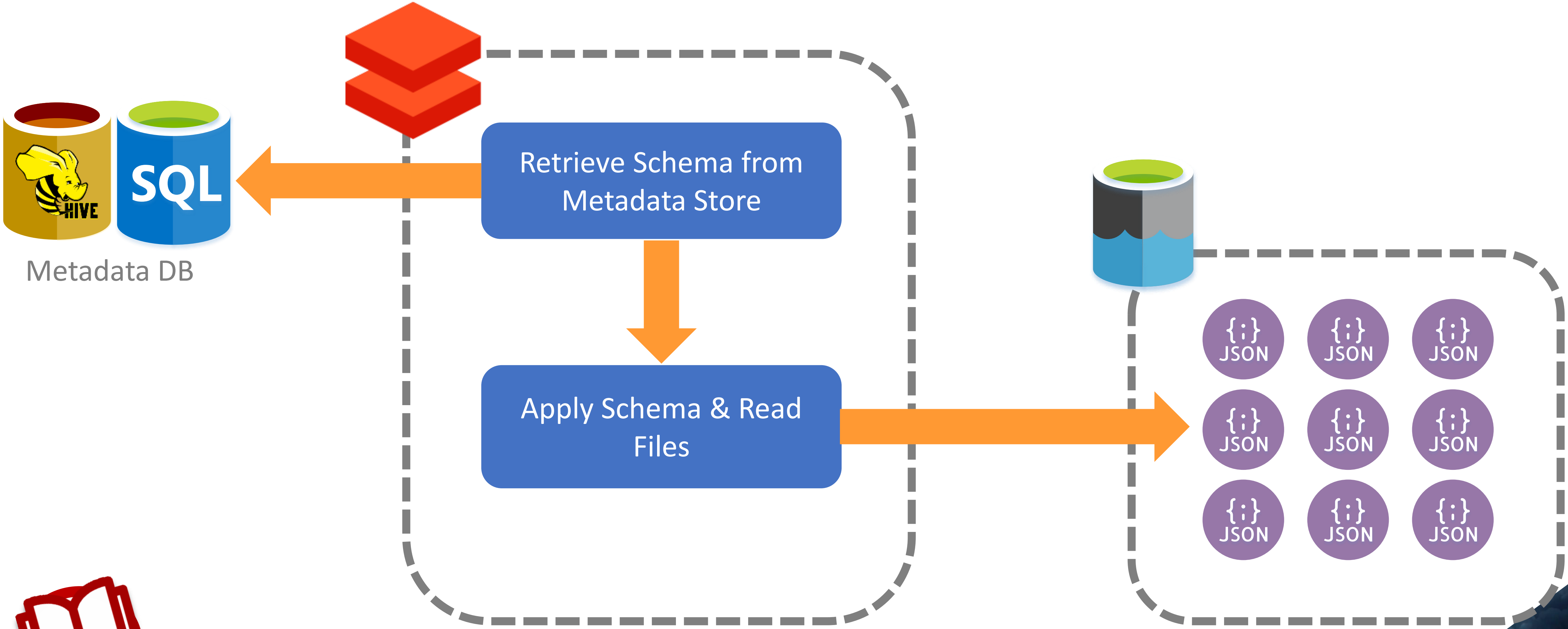
Samples Every File!

SCHEMA ON READ – SAMPLE FILES



Inferring schema from a small file sample before reading large datasets?

SCHEMA ON READ – METADATA STORE

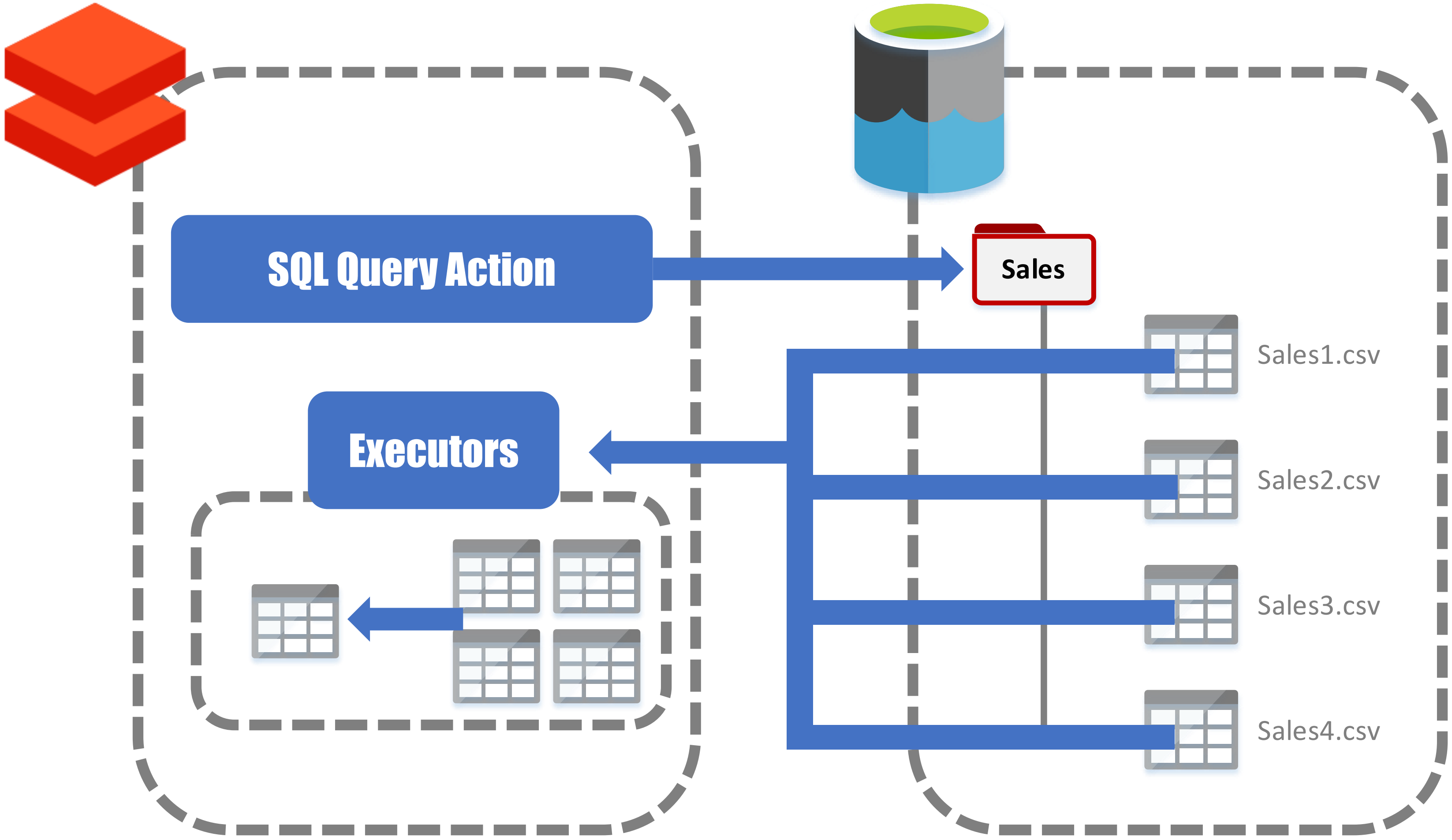


Files such as Parquet don’t need a schema to be supplied!

PYTHON PIPELINE PRIMER

READING FILES – NO PARTITIONS

*SELECT * FROM MyFiles WHERE Year = 2019 AND Month = 3*

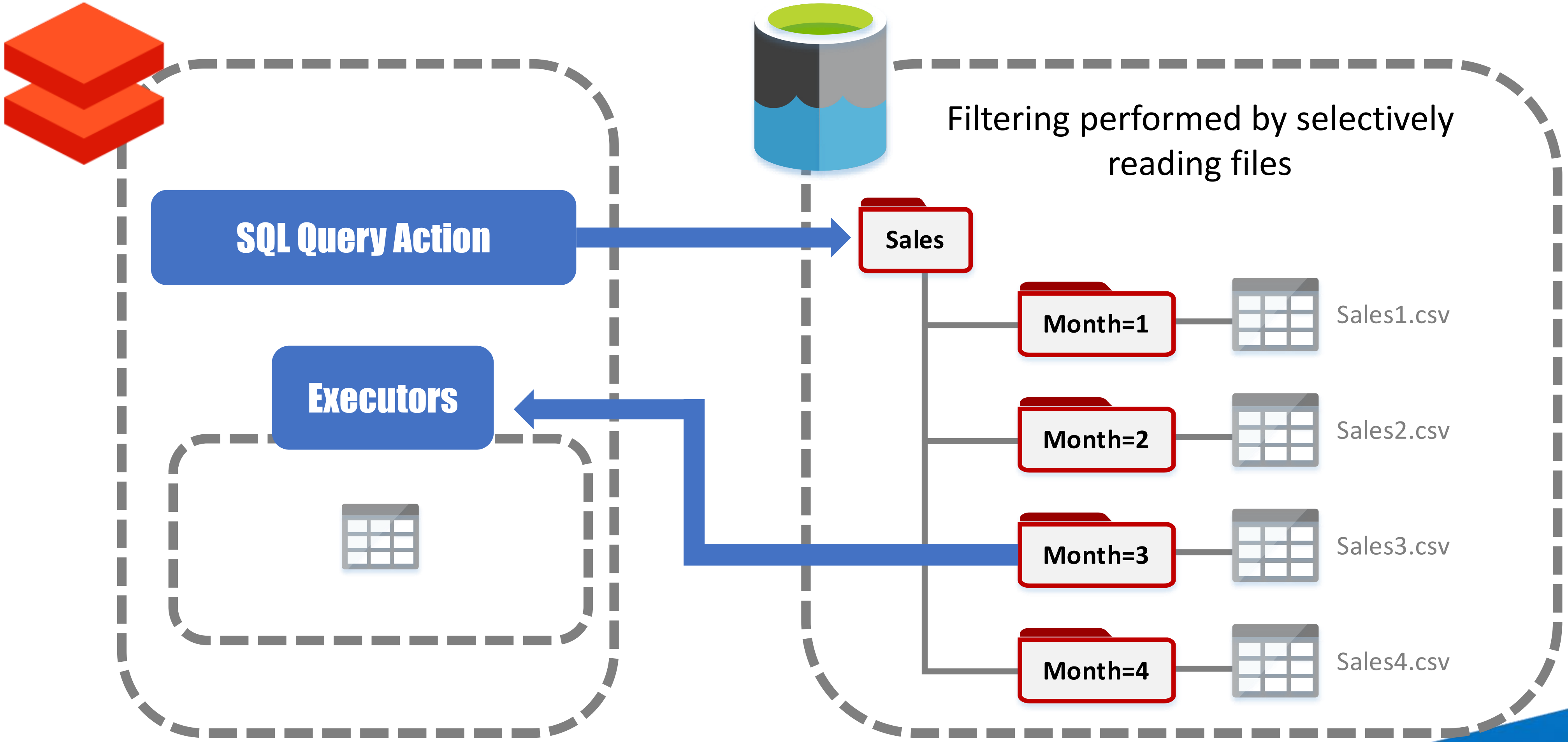


Filtering performed on executors
after reading all files

PYTHON PIPELINE PRIMER

READING FILES – PARTITIONED

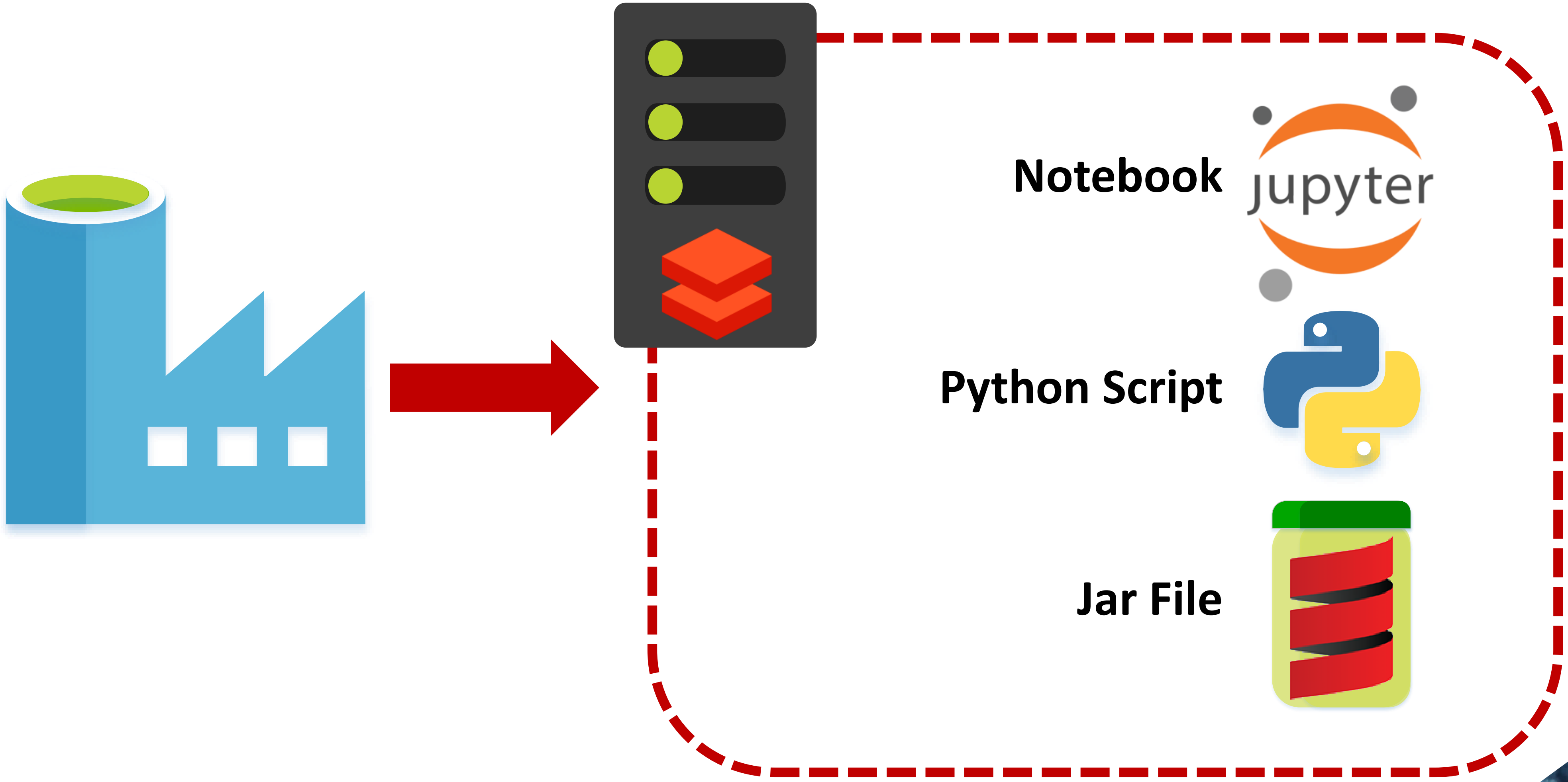
*SELECT * FROM MyFiles WHERE Year = 2019 AND Month = 3*



PYTHON PIPELINE PRIMER

ORCHESTRATION

AZURE DATA FACTORY

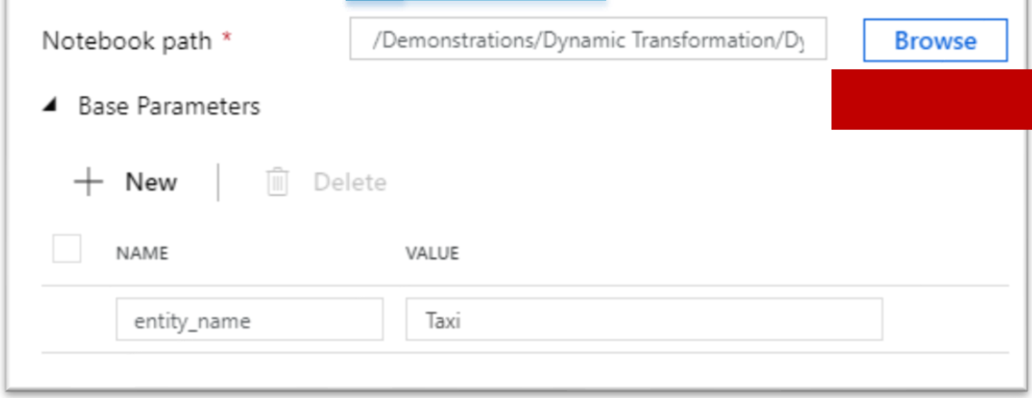


Notebook

Python Script

Jar File

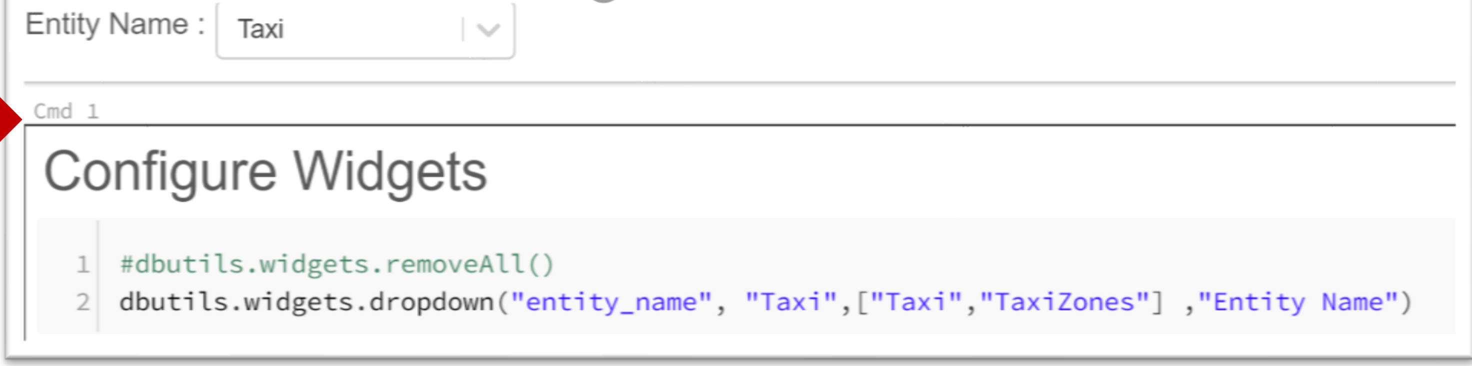
AZURE DATA FACTORY



Notebook path * /Demonstrations/Dynamic Transformation/Dy

◢ Base Parameters

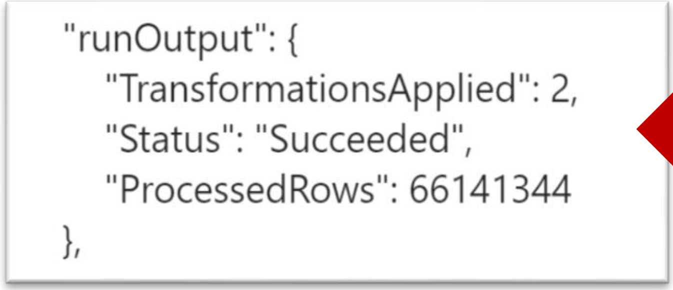
□	NAME	VALUE
	entity_name	Taxi



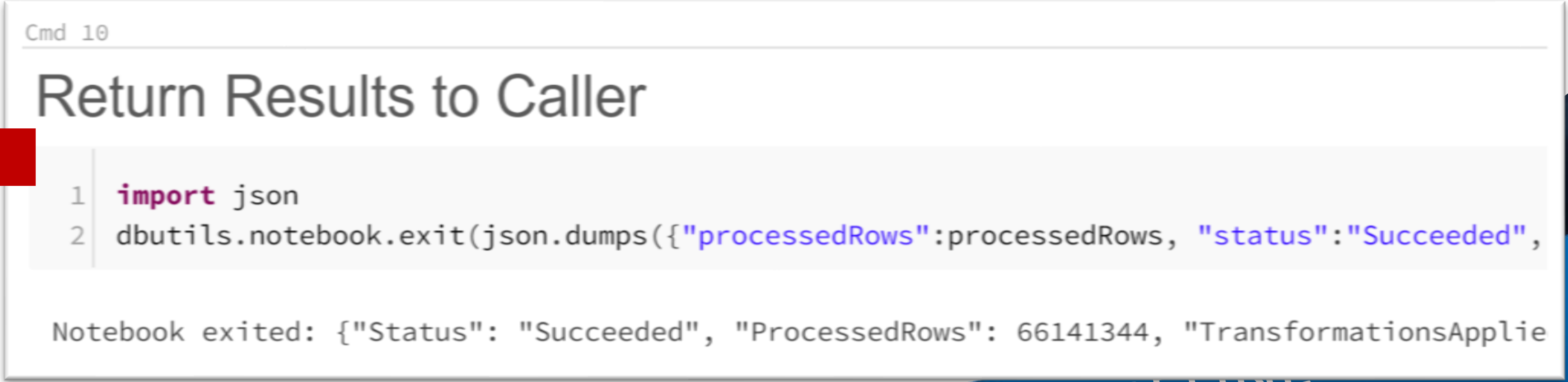
Entity Name : Taxi

Configure Widgets

```
#dbutils.widgets.removeAll()
dbutils.widgets.dropdown("entity_name", "Taxi",["Taxi","TaxiZones"] ,"Entity Name")
```



```
"runOutput": {
    "TransformationsApplied": 2,
    "Status": "Succeeded",
    "ProcessedRows": 66141344
},
```



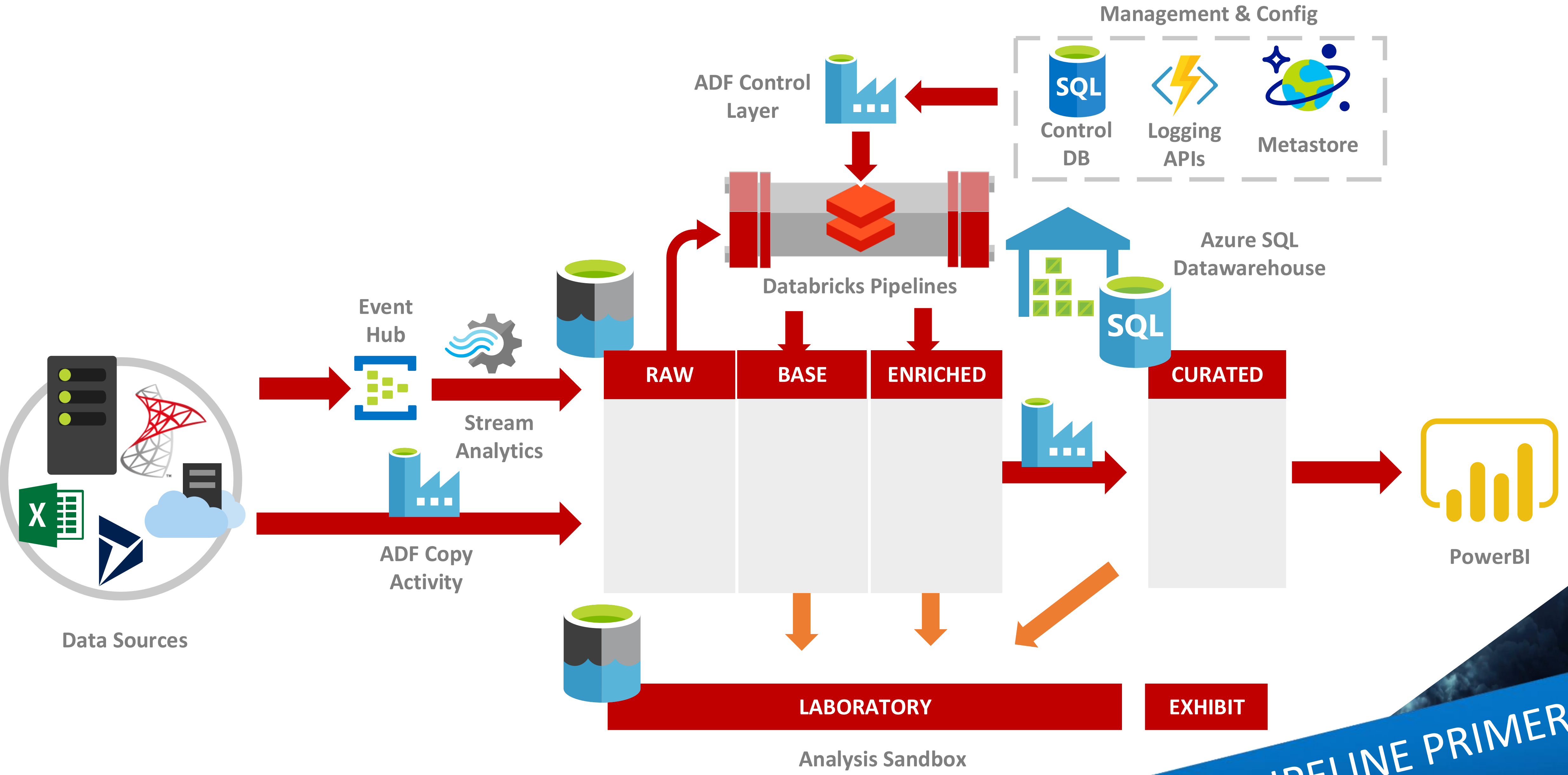
Return Results to Caller

```
import json
dbutils.notebook.exit(json.dumps({"processedRows":processedRows, "status":"Succeeded",
```

```
Notebook exited: {"Status": "Succeeded", "ProcessedRows": 66141344, "TransformationsApplie
```

PYTHON PIPELINE

THE BIG PICTURE

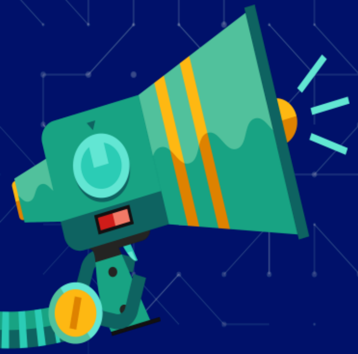


PYTHON PIPELINE PRIMER

SIMON WHITELEY

@MRSIWHITELEY

WWW.ADVANCINGANALYTICS.CO.UK



Questions?