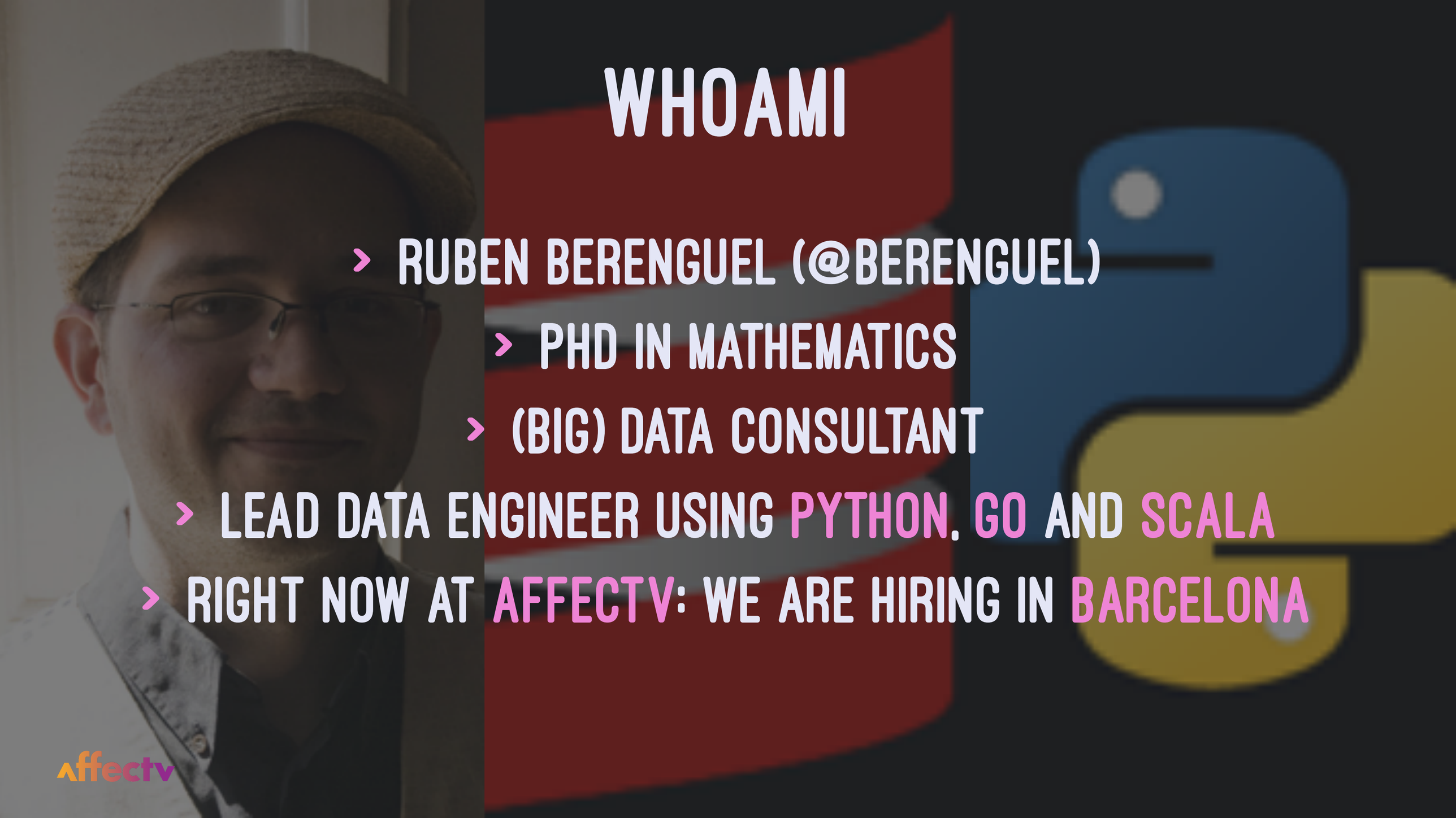
THE MAGIC OF PYSPARK

AN INTRODUCTION TO PYSPARK

affectv



WHOAMI

- RUBEN BERENGUEL (@BERENGUEL)
- PHD IN MATHEMATICS
- (BIG) DATA CONSULTANT
- LEAD DATA ENGINEER USING PYTHON, GO AND SCALA
- RIGHT NOW AT AFFECTV: WE ARE HIRING IN BARCELONA

affectv

WHAT IS SPARK?

WHAT IS SPARK?

- DISTRIBUTED COMPUTATION FRAMEWORK

WHAT IS SPARK?

- DISTRIBUTED COMPUTATION FRAMEWORK
- OPEN SOURCE

WHAT IS SPARK?

- DISTRIBUTED COMPUTATION FRAMEWORK
  - OPEN SOURCE
  - EASY TO USE

WHAT IS SPARK?

- DISTRIBUTED COMPUTATION FRAMEWORK
  - OPEN SOURCE
  - EASY TO USE
- SCALES HORIZONTALLY AND VERTICALLY

HOW DOES SPARK WORK?

affectv

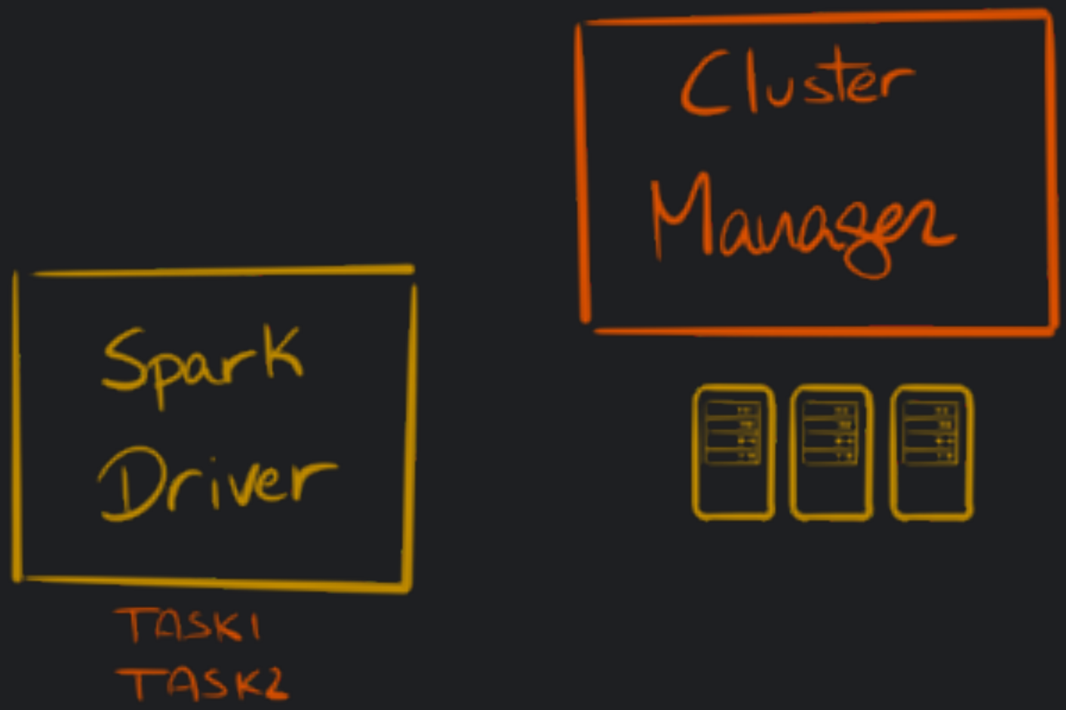
SPARK USUALLY RUNS ON TOP OF A CLUSTER MANAGER

AND A DISTRIBUTED STORAGE

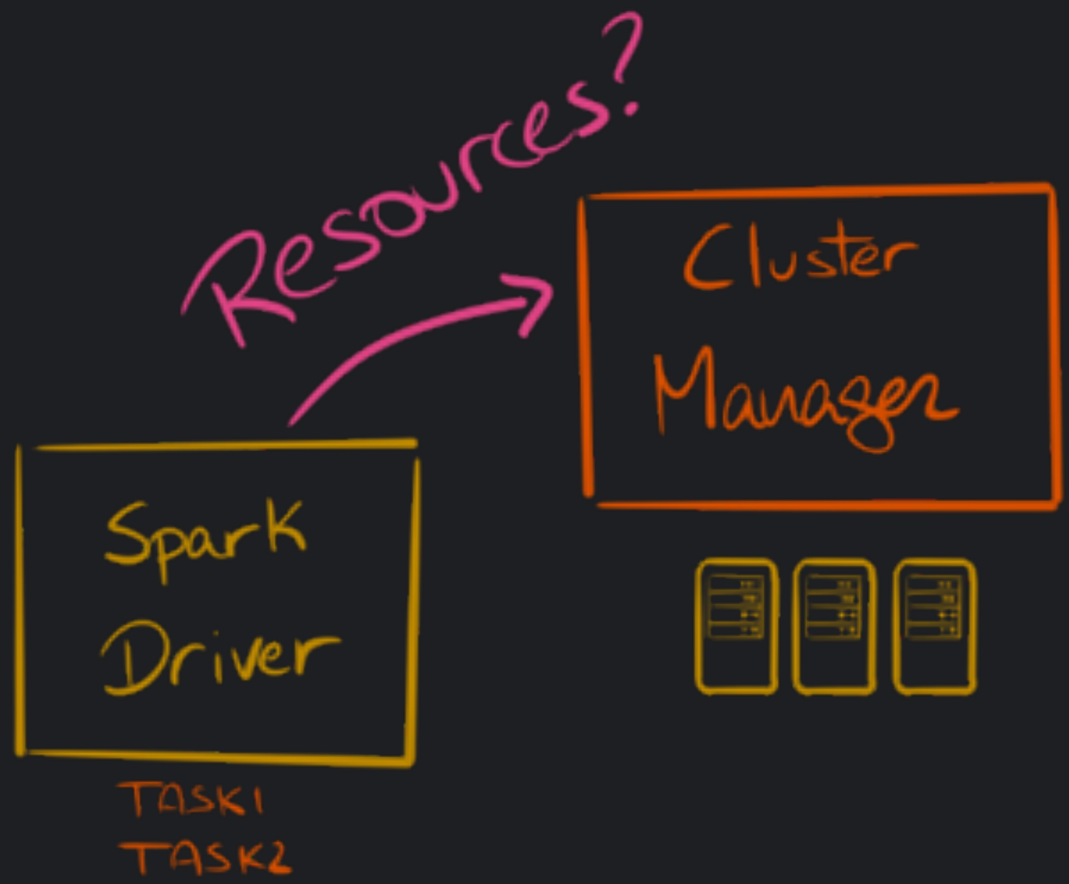
A SPARK PROGRAM
RUNS IN THE DRIVER

THE DRIVER REQUESTS RESOURCES FROM THE CLUSTER MANAGER TO RUN TASKS

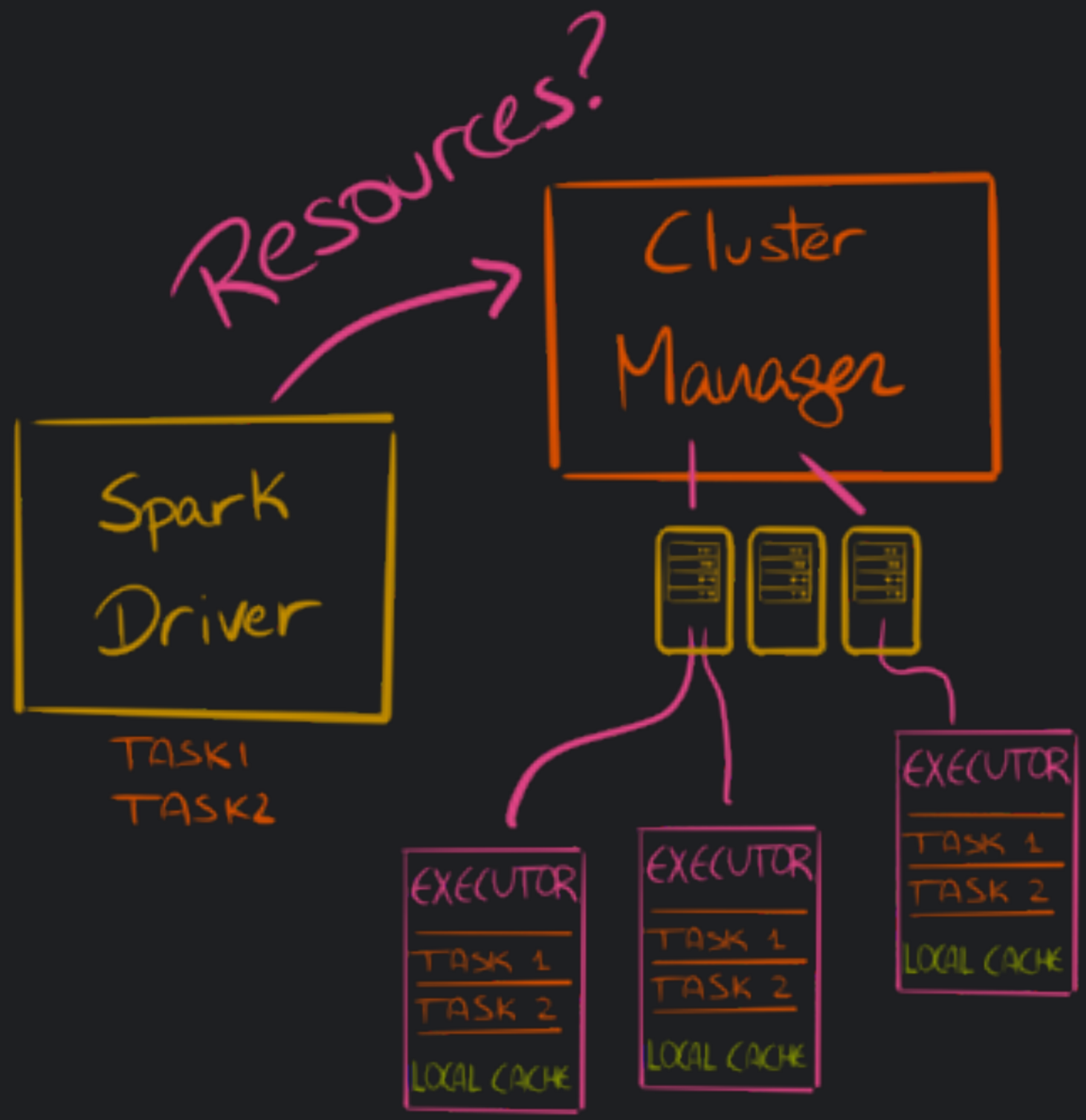
THE DRIVER REQUESTS RESOURCES FROM THE CLUSTER MANAGER TO RUN TASKS



THE DRIVER REQUESTS RESOURCES FROM THE CLUSTER MANAGER TO RUN TASKS



THE DRIVER REQUESTS RESOURCES FROM THE CLUSTER MANAGER TO RUN TASKS



affectv

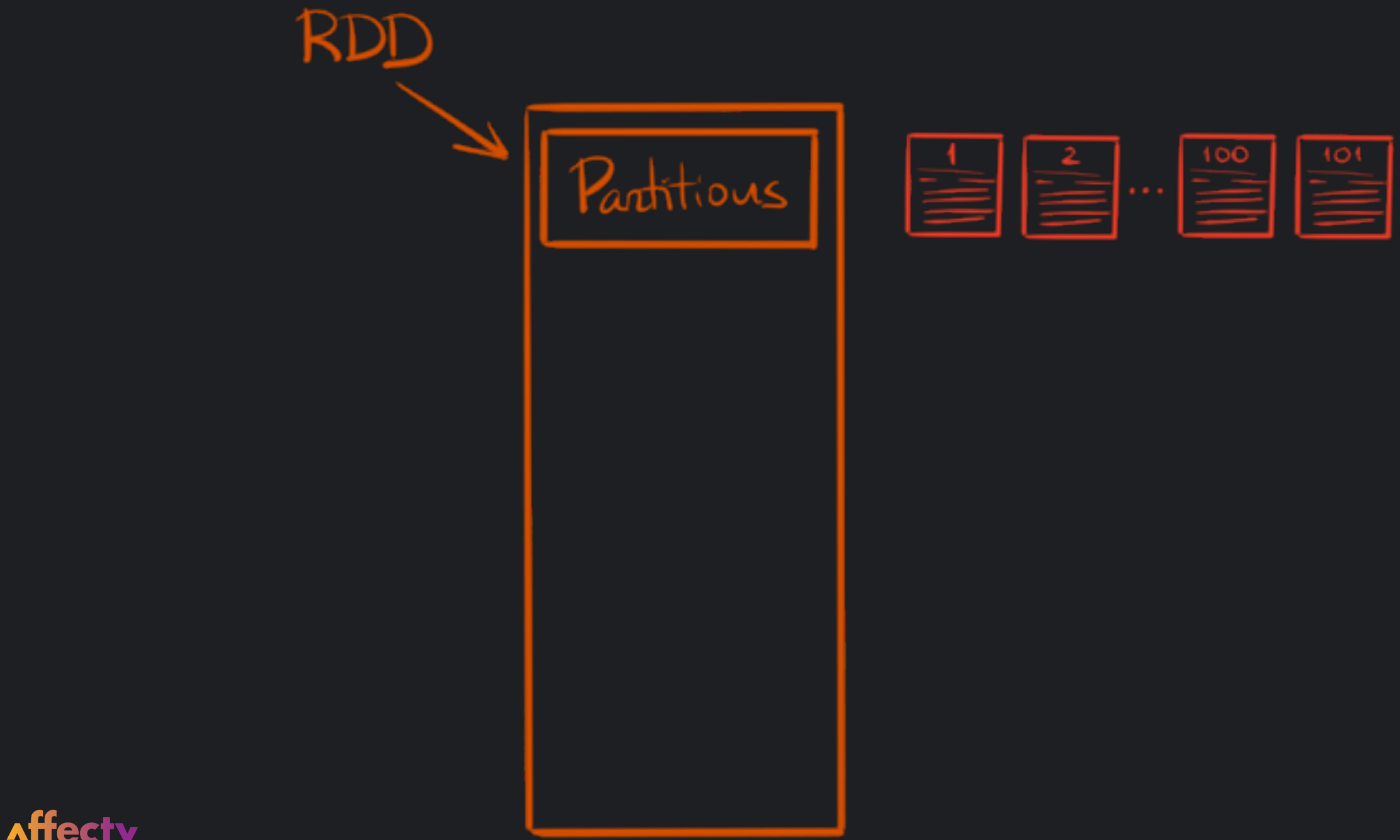
THE MAIN BUILDING BLOCK IS THE RDD: RESILIENT DISTRIBUTED DATASET

affectv

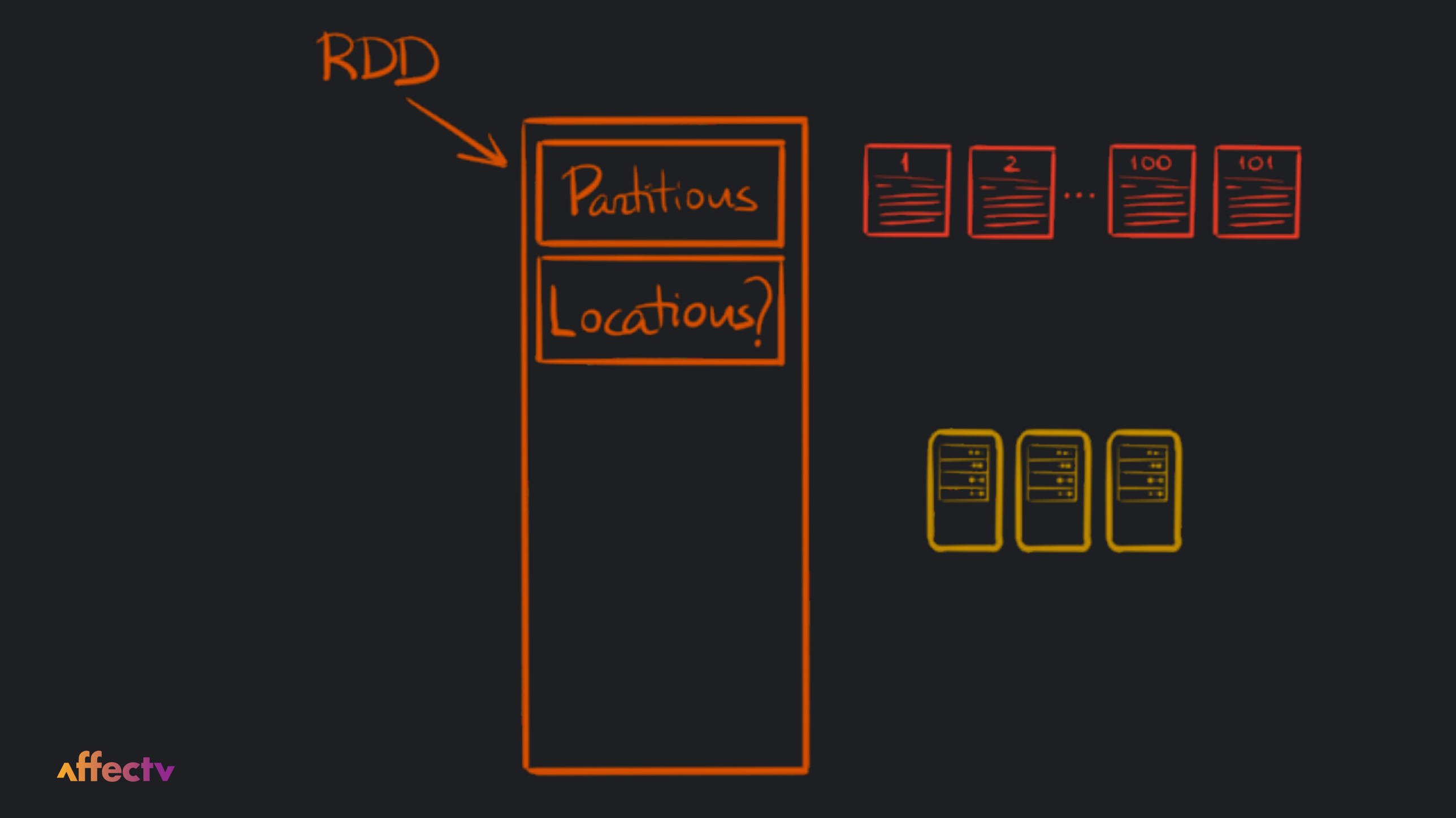
affectv

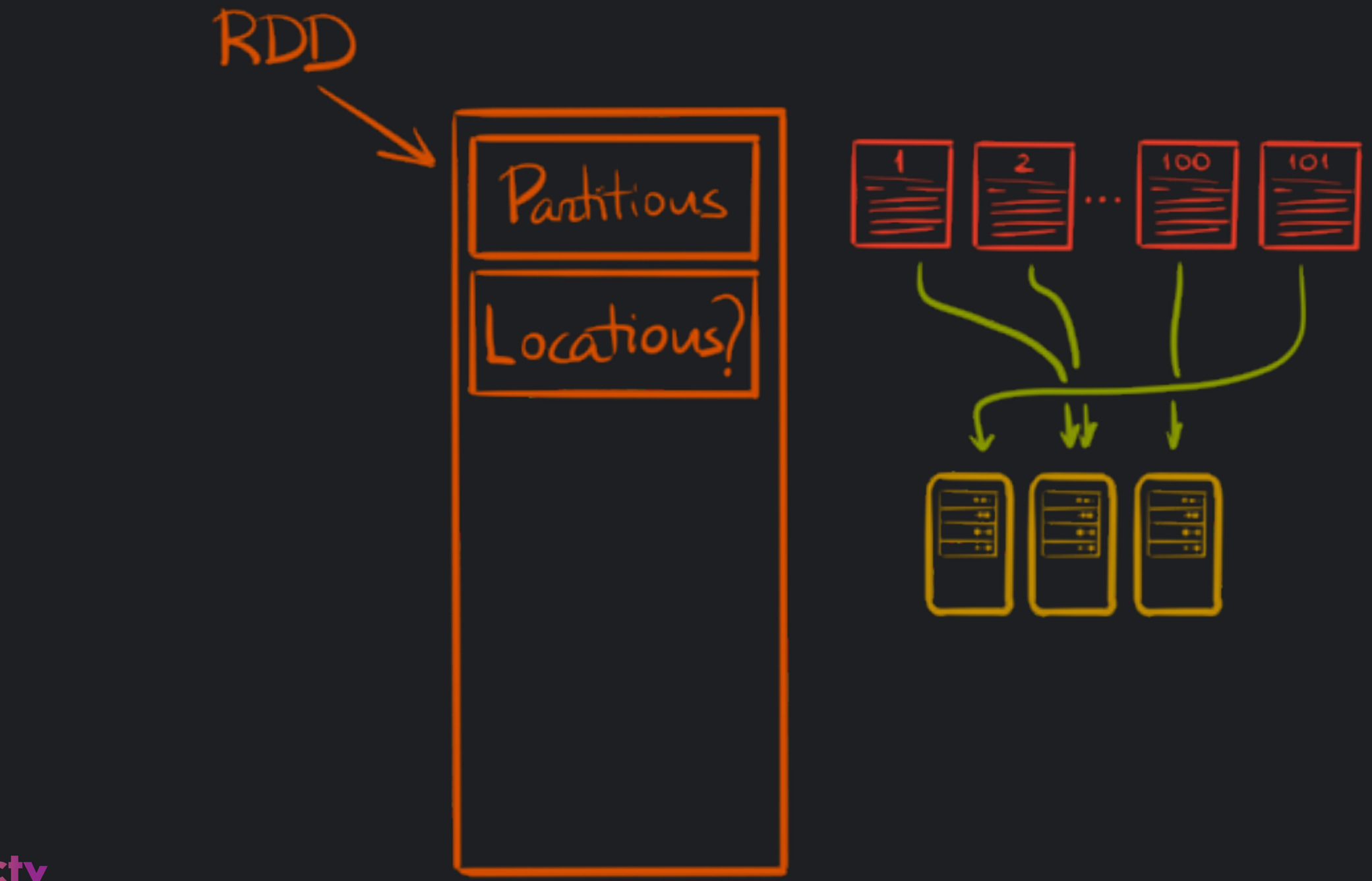

RDD

affectv

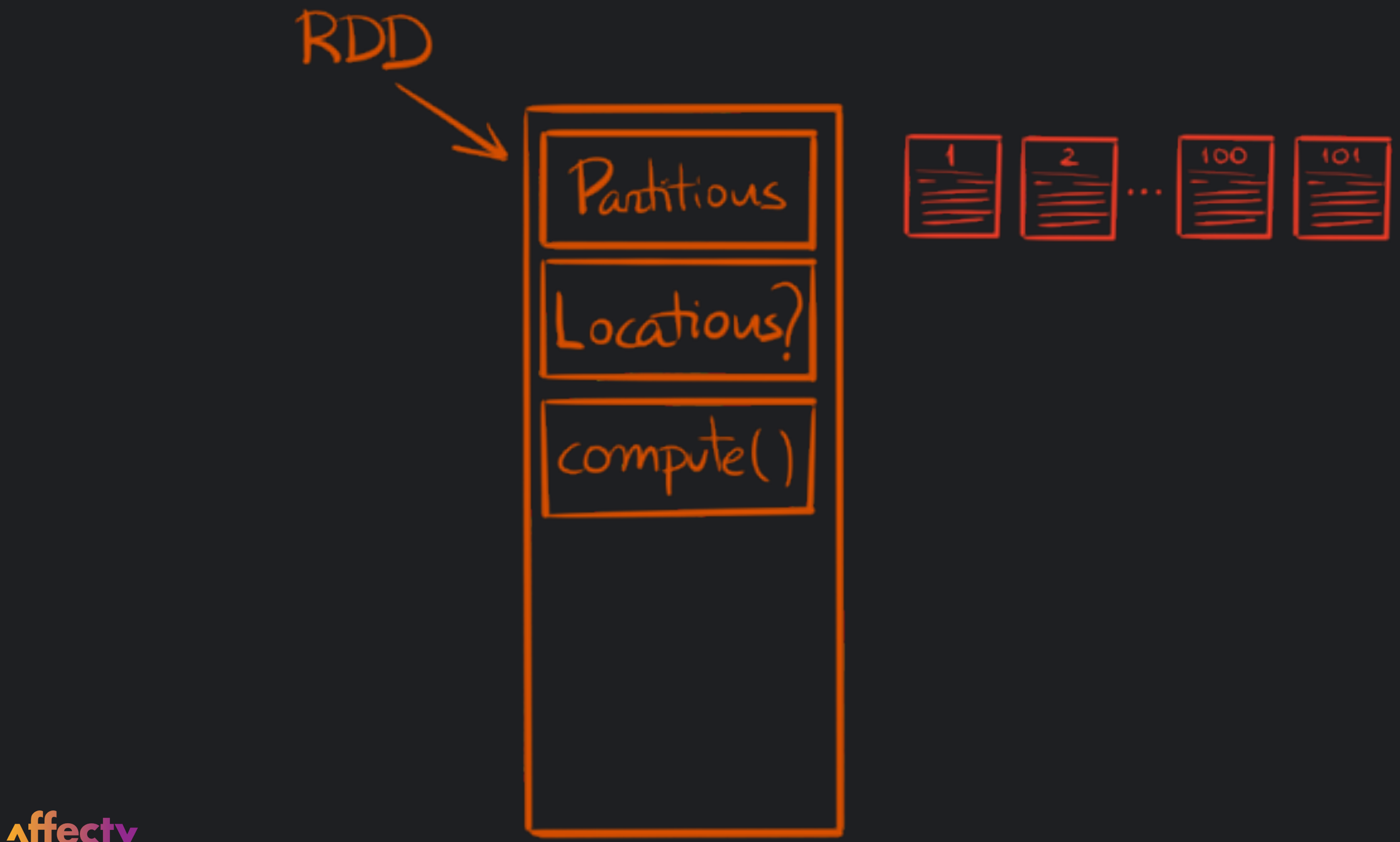


affectv

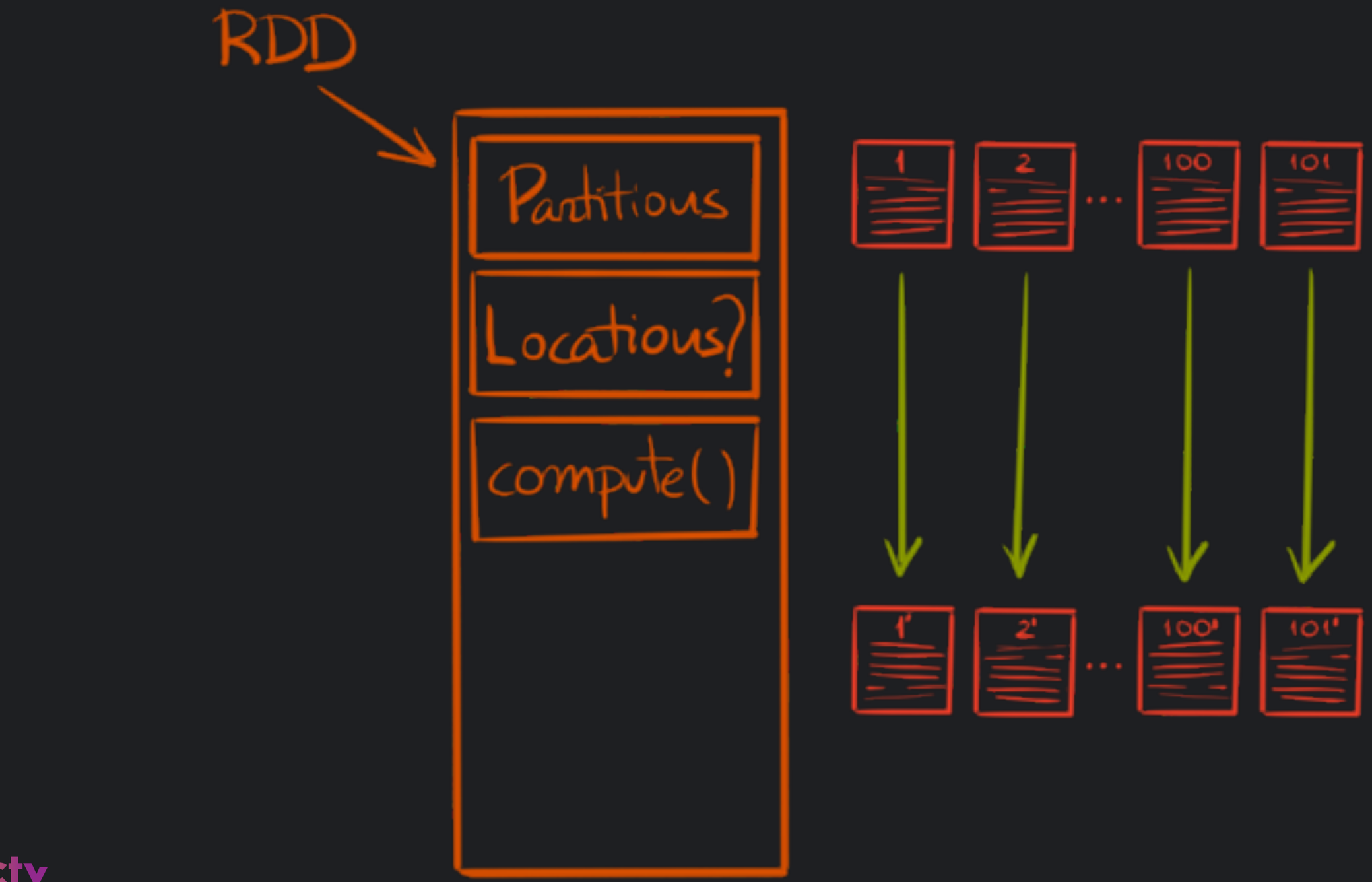

RDD
Partitions
Locations?
1
2
100
101



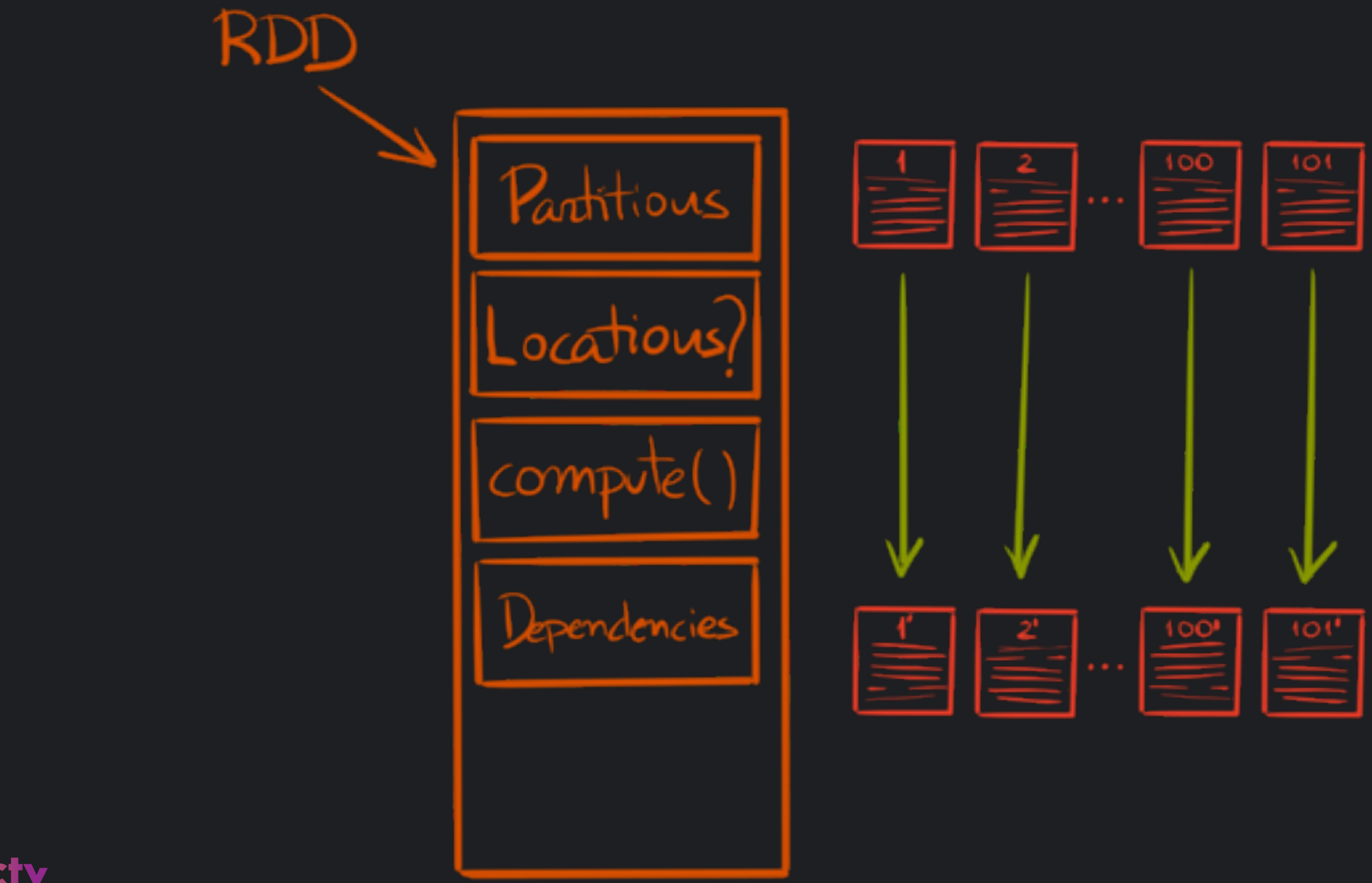
affectv



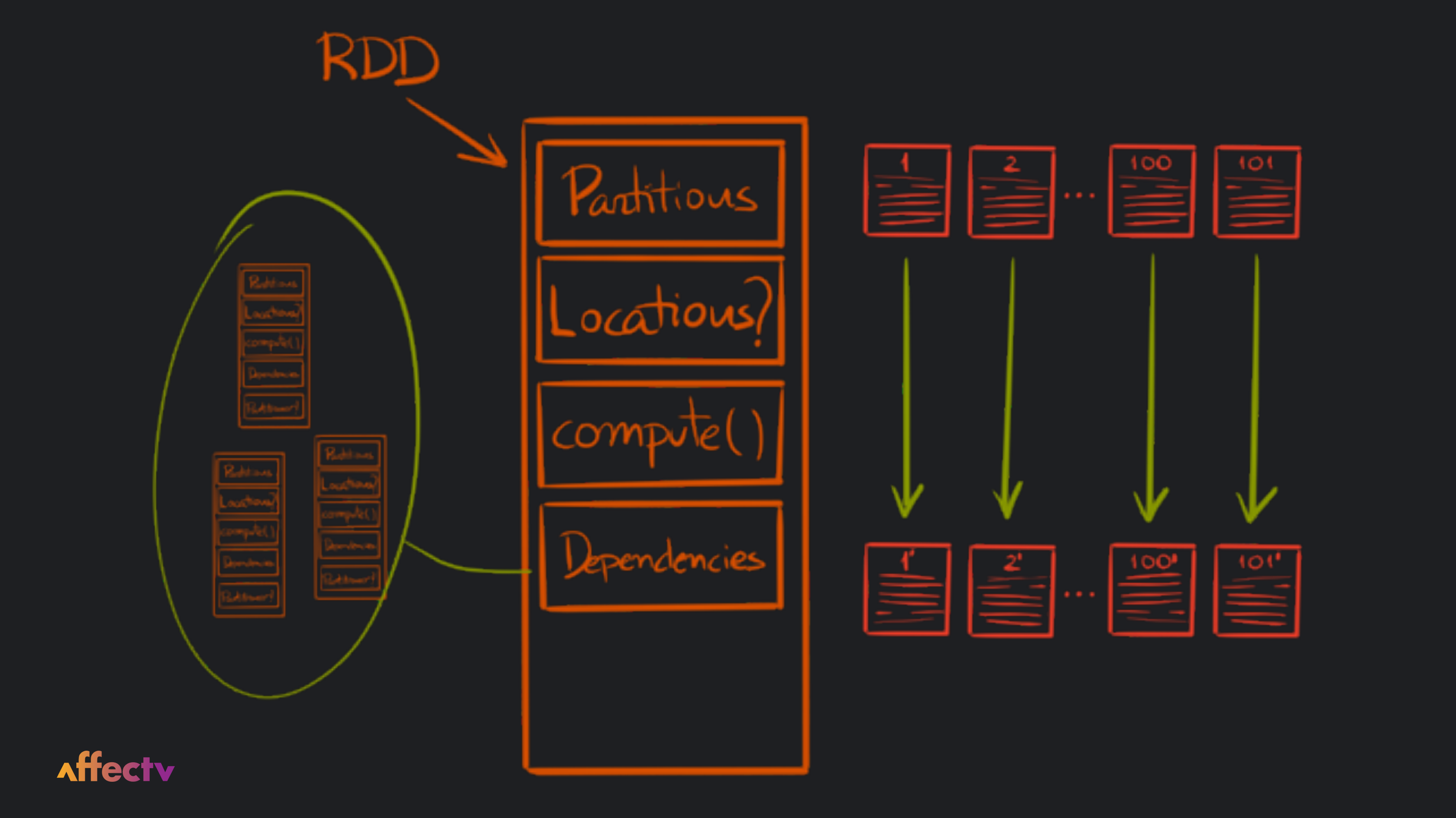
affectv

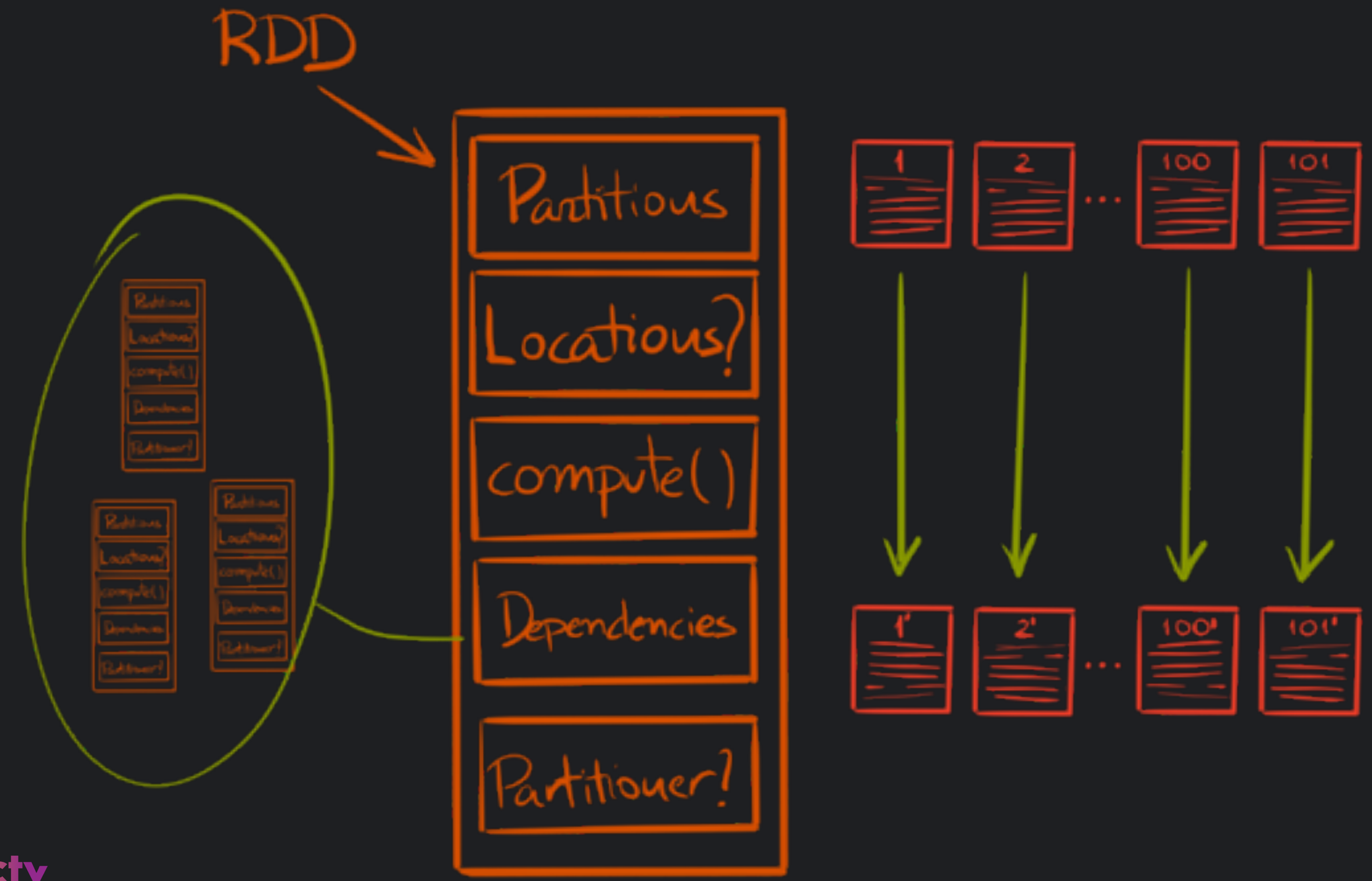


affectv


RDD
Partitions
Locations?
compute()
Dependencies
1
2
100
101
1'
2'
100'
101'

affectv


RDD
Partitions
Locations?
compute()
Dependencies
1
2
100
101
1'
2'
100'
101'


RDD
Partitions
Locations?
compute()
Dependencies
Partitioner?
1
2
100
101
1'
2'
100'
101'

affectv

TWO KIND OF OPERATIONS:

TRANSFORMATIONS

&

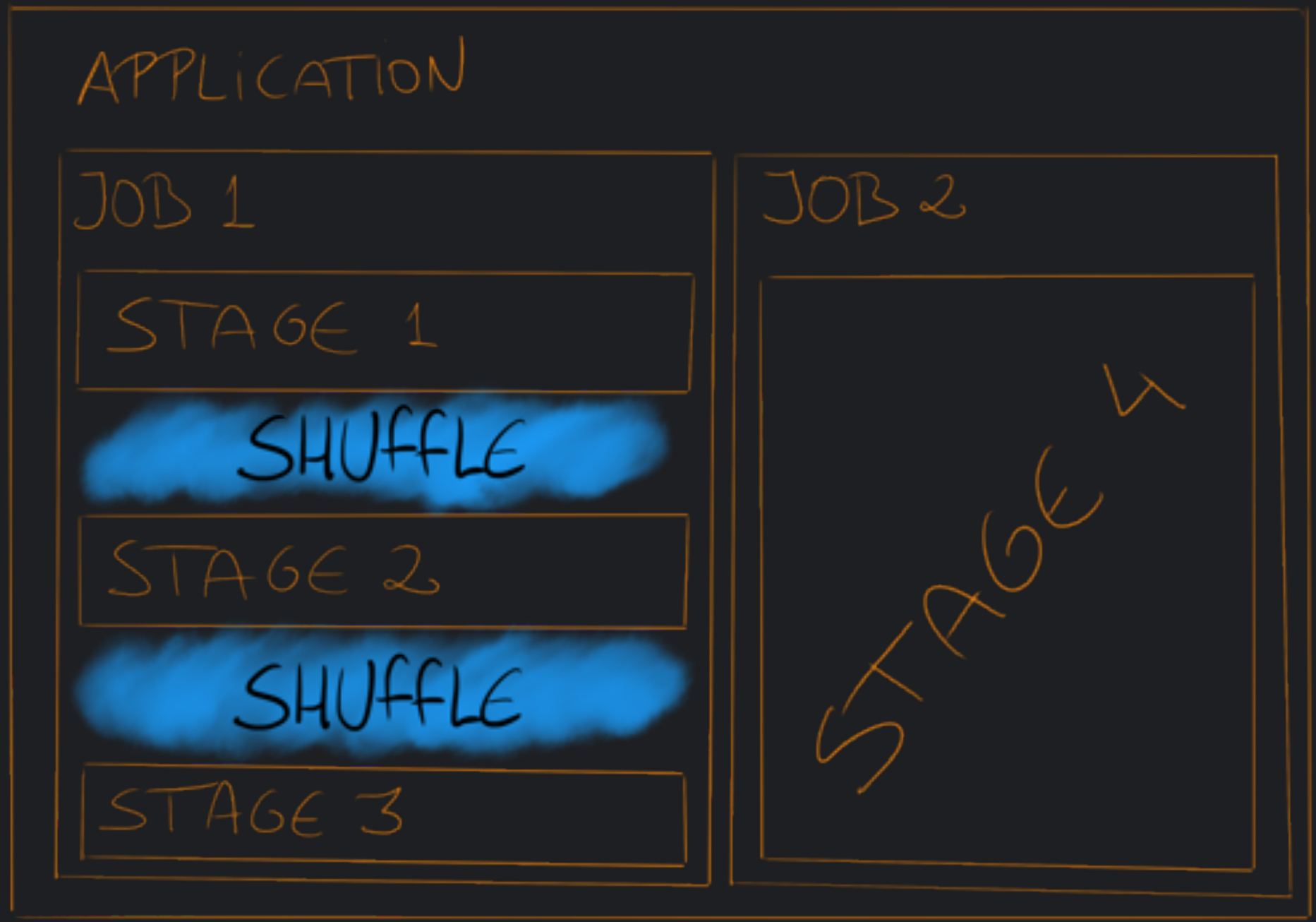
ACTIONS

TRANSFORMATIONS: RESHAPE THE DATA. THEY ARE PART OF STAGES

- FILTER
- MAP
- JOIN

ACTIONS: RETURN A RESULT. THEY DEFINE JOBS

- COUNT
- SHOW
- WRITE

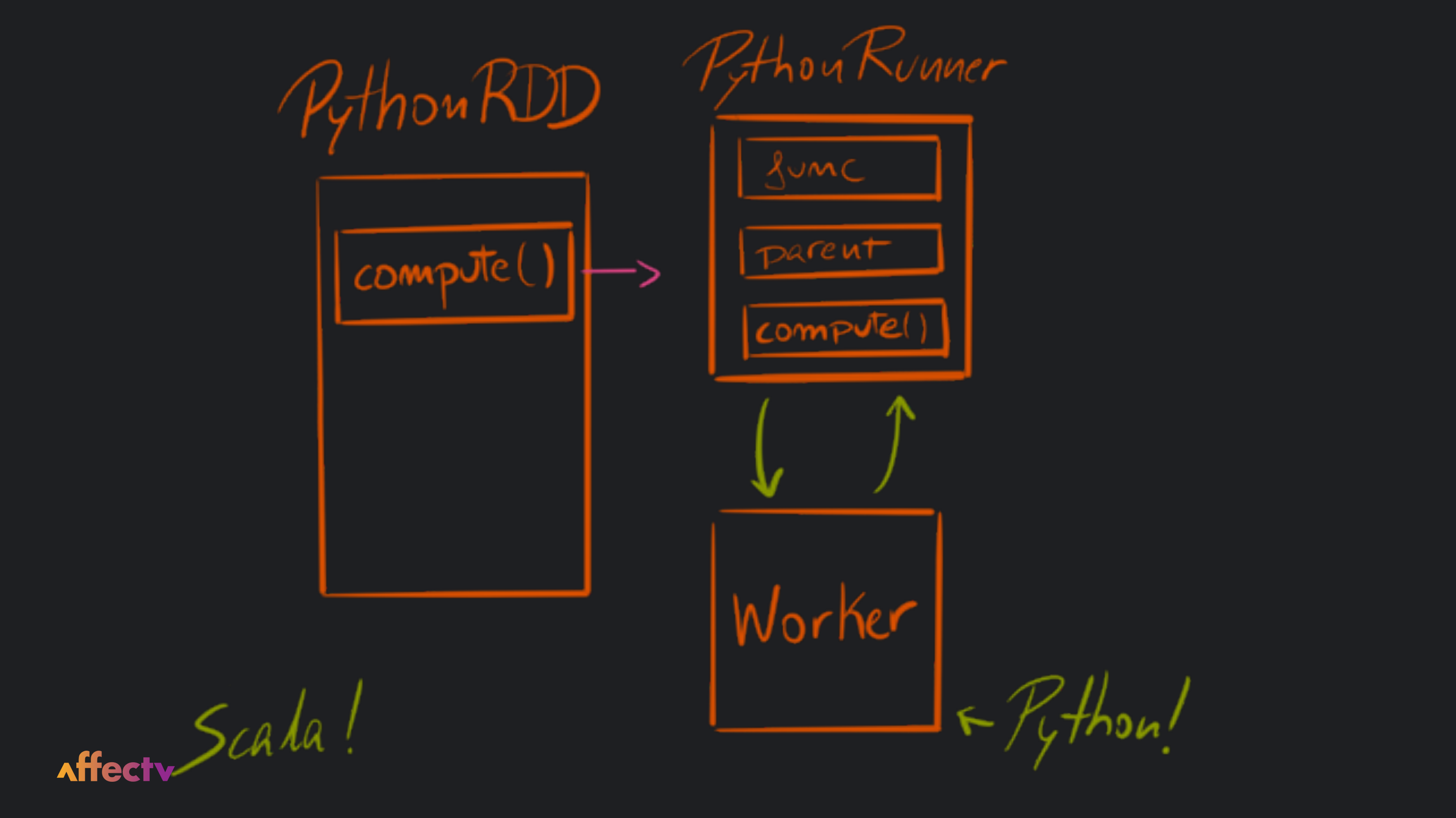

APPLICATION
JOB 1
STAGE 1
SHUFFLE
STAGE 2
SHUFFLE
STAGE 3
JOB 2
STAGE 4

PYSPARK

PYSPARK OFFERS A PYTHON API TO THE SCALA CORE OF SPARK

IT USES THE PY4J BRIDGE

affectv


PythonRDD
compute()
PythonRunner
func
parent
compute()
Worker
Scala!
Python!
affectv

WORKERS ACT AS STANDALONE PROCESSORS OF STREAMS OF DATA

affectv

WORKERS ACT AS STANDALONE PROCESSORS OF STREAMS OF DATA

- CONNECTS BACK TO THE JVM THAT STARTED IT

WORKERS ACT AS STANDALONE PROCESSORS OF STREAMS OF DATA

- CONNECTS BACK TO THE JVM THAT STARTED IT
- LOAD INCLUDED PYTHON LIBRARIES

WORKERS ACT AS STANDALONE PROCESSORS OF STREAMS OF DATA

- CONNECTS BACK TO THE JVM THAT STARTED IT
- LOAD INCLUDED PYTHON LIBRARIES
- DESERIALIZES THE PICKLED FUNCTION COMING FROM THE STREAM

WORKERS ACT AS STANDALONE PROCESSORS OF STREAMS OF DATA

- CONNECTS BACK TO THE JVM THAT STARTED IT
- LOAD INCLUDED PYTHON LIBRARIES
- DESERIALIZES THE PICKLED FUNCTION COMING FROM THE STREAM
- APPLIES THE FUNCTION TO THE DATA COMING FROM THE STREAM

WORKERS ACT AS STANDALONE PROCESSORS OF STREAMS OF DATA

- CONNECTS BACK TO THE JVM THAT STARTED IT
- LOAD INCLUDED PYTHON LIBRARIES
- DESERIALIZES THE PICKLED FUNCTION COMING FROM THE STREAM
- APPLIES THE FUNCTION TO THE DATA COMING FROM THE STREAM
- SENDS THE OUTPUT BACK

WHAT ABOUT DATAFRAMES?

THE RDD API EXPECTS YOU TO HANDLE EVERYTHING.

BUT SPARK CAN MAGICALLY OPTIMISE EVERYTHING..

affectv

..IF YOU USE SPARK DataFrame

SPARK WILL GENERATE
A PLAN

(A DIRECTED ACYCLIC GRAPH)

TO COMPUTE THE
RESULT

AND THE PLAN WILL BE OPTIMISED USING CATALYST

affectv

PLEASE. PLEASE. USE SPARK ≥ 2.3 (OR. THE LATEST YOU CAN)

YOU SHOULD TRY TO USE DataFrame UNLESS THERE IS A REASON NOT TO (LIKE SOME MLLIB METHOD)

affectv

YOU SHOULD ALSO ENABLE ARROW OPTIMISATIONS TO SPEED UP THE DATA TRANSFER FROM THE JVM TO PYTHON

WE'LL SEE WHY AND HOW LATER DURING THE WORKSHOP

NEW PROJECT: KOALAS[1]

OFFERS A UNIFIED API BETWEEN PANDAS AND SPARK DATAFRAMES (AS MUCH AS POSSIBLE)

[1] HTTPS://GITHUB.COM/DASK/DASK

affectv

PURE PYTHON ALTERNATIVE: DASK[1]

ANOTHER DISTRIBUTED FRAMEWORK

[1] HTTPS://GITHUB.COM/DASK/DASK

SHOULD YOU USE SPARK?

affectv

YES IF

affectv

YES IF

- THE DATA IS VERY LARGE (SIGNIFICANTLY LARGER THAN MEMORY)

YES IF

- THE DATA IS VERY LARGE (SIGNIFICANTLY LARGER THAN MEMORY)
- YOUR ORG ALREADY HAS SPARK CLUSTER OR CODEBASE

YES IF

- THE DATA IS VERY LARGE (SIGNIFICANTLY LARGER THAN MEMORY)
- YOUR ORG ALREADY HAS SPARK CLUSTER OR CODEBASE
- THERE IS NO BETTER ALTERNATIVE

NO IF

affectv

NO IF

- YOU WANT TO ADD SPARK TO YOUR CV

affectv

NO IF

- YOU WANT TO ADD SPARK TO YOUR CV
  - JAVA STACKTRACES SCARE YOU

RESOURCES

- SPARK DOCUMENTATION
- HIGH PERFORMANCE SPARK BY HOLDEN KARAU
- THE INTERNALS OF APACHE SPARK 2.4.2 BY JACEK LASKOWSKI
- SPARK'S GITHUB
- BECOME A CONTRIBUTOR


THANKS!
affectv

WORKSHOP TIME!

affectv

GET THE SLIDES AND NOTEBOOK WE WILL USE FROM MY GITHUB:

github.com/rberenguel/

THE REPOSITORY IS

pyspark_workshop

affectv

FURTHER REFERENCES

affectv

ARROW

ARROW'S HOME
ARROW'S GITHUB
ARROW SPEED BENCHMARKS
ARROW TO PANDAS CONVERSION BENCHMARKS
POST: STREAMING COLUMNAR DATA WITH APACHE ARROW
POST: WHY PANDAS USERS SHOULD BE EXCITED BY APACHE ARROW
CODE: ARROW-PANDAS COMPATIBILITY LAYER CODE
CODE: ARROW TABLE CODE
PYARROW IN-MEMORY DATA MODEL
BALLISTA: A POC DISTRIBUTED COMPUTE PLATFORM (RUST)
PYJAVA: POC ON JAVA/SCALA AND PYTHON DATA INTERCHANGE WITH ARROW

PANDAS

PANDAS' HOME
PANDAS' GITHUB
GUIDE: IDIOMATIC PANDAS
CODE: PANDAS INTERNALS
DESIGN: PANDAS INTERNALS
TALK: DEMYSTIFYING PANDAS' INTERNALS, BY MARC GARCIA
MEMORY LAYOUT OF MULTIDIMENSIONAL ARRAYS IN NUMPY

affectv

SPARK/PYSPARK

CODE: PYSPARK SERIALIZERS
JIRA: FIRST STEPS TO USING ARROW (ONLY IN THE PYSPARK DRIVER)
POST: SPEEDING UP PYSPARK WITH APACHE ARROW
ORIGINAL JIRA ISSUE: VECTORIZED UDFS IN SPARK
INITIAL DOC DRAFT
POST BY BRYAN CUTLER (LEADER FOR THE VEC UDFS PR)
POST: INTRODUCING PANDAS UDF FOR PYSPARK
CODE: ORG.APACHE.SPARK.SQL.VECTORIZED
POST BY BRYAN CUTLER: SPARK TOPANDAS() WITH ARROW, A DETAILED LOOK

PY4J

PY4J'S HOME
PY4J'S GITHUB
CODE: REFLECTION ENGINE

EOF

affectv