Adopting Dataframes and Parquet in an Already Existing Warehouse

Sol Ackerman & Franklyn D'souza


Spotify®

shopify

Search

Home

Orders

Products

Customers

Reports

Discounts

SALES CHANNELS

Online Store

Apps

Settings

Home

Welcome to Shopify,

Your trial just started

ADVICE

Add a product to see it in your store

Add a product

Customize the look of your website

Select a theme

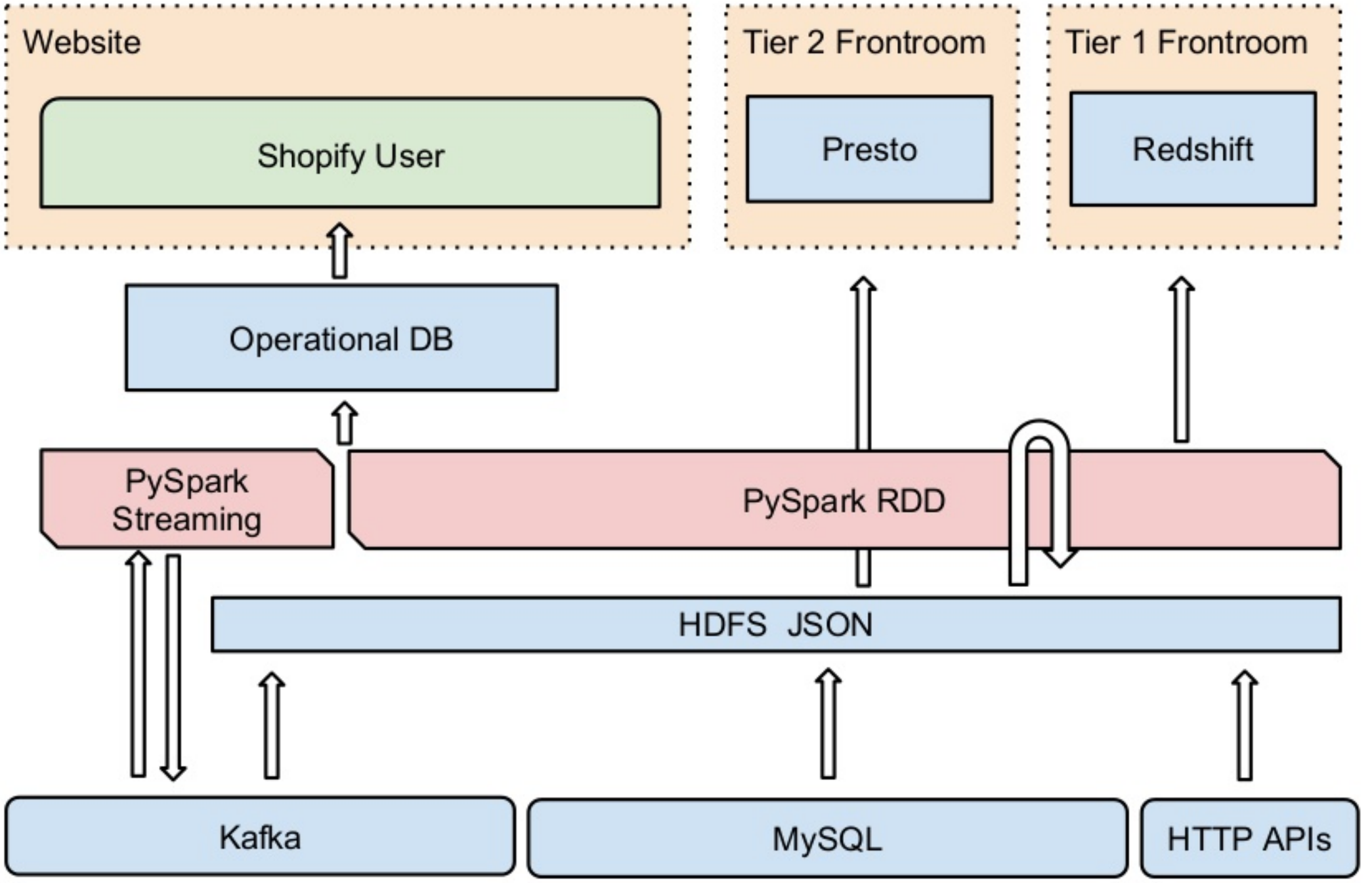
Set up a custom domain

EXCLUSIVE LAUNCH UPDATES ON KYLIE'S APP!

USD ACCOUNT 0

KYLIE COSMETICS

MATTE LIP KITS


Website
Shopify User
Tier 2 Frontroom
Presto
Tier 1 Frontroom
Redshift
Operational DB
PySpark Streaming
PySpark RDD
HDFS JSON
Kafka
MySQL
HTTP APIs

Warehouse Factoids

>300k
STORES

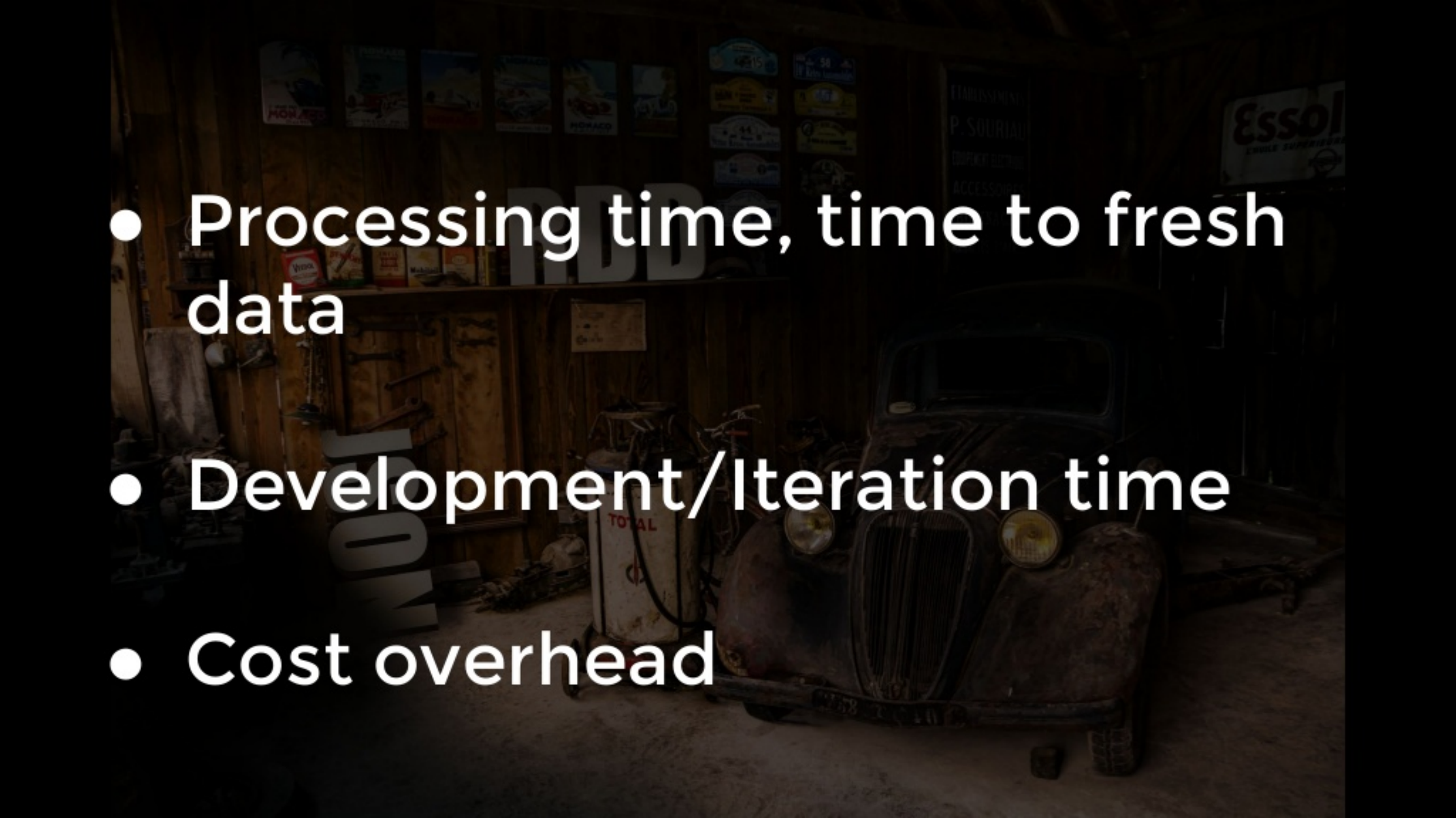
2 PB
DATA

4000
JOBS / DAY

1.5 TB
NEW DATA / DAY

Why Dataframes + Parquet ?


RDD
MOST
HONACO
ETABLISSEMENTS
P. SOURIAU
ACCESSOIRES
BOBINAGES
Essol
TOTAL



- Processing time, time to fresh data
- Development/Iteration time
- Cost overhead

- Columnar format
- Python simplicity, JVM performance
- Unlocks SQL on HDFS

Goals

- Zero down time for analysts / users
- Fully backwards compatible
- No data loss / corruption
- Incremental rollout, mitigating risk, downtime, revertability

Adoption Plan

- Keep old code
- Introduce Parquet + Dataframes
- Rewrite slow jobs as Dataframe pipelines

Backwards Compatible

Data agnostic reader / writer

Data agnostic reader

```
rdd1 = read('hdfs://file1.parquet', load_type='rdd')
rdd2 = read('hdfs://file2.json', load_type='rdd')
df1 = read('hdfs://file1.parquet', load_type='dataframe')
df2 = read('hdfs://file2.json', load_type='dataframe')
```

Configurable Output-format

```
sample-job:
 owner: franklyn.dsouza@shopify.com
 build:
   resource_class: large
   command_options:
     some-file-on-hdfs: hdfs://input/to/job
     output: hdfs://output/from/job
     output-format: parquet
   file:job.py
```

Configurable Output-format

```
sample-job:
 owner: franklyn.dsouza@shopify.com
 build:
   resource_class: large
   command_options:
     some-file-on-hdfs: hdfs://input/to/job
     output: hdfs://output/from/job
     output-format: parquet
   file:job.py
```

Configurable Output-format

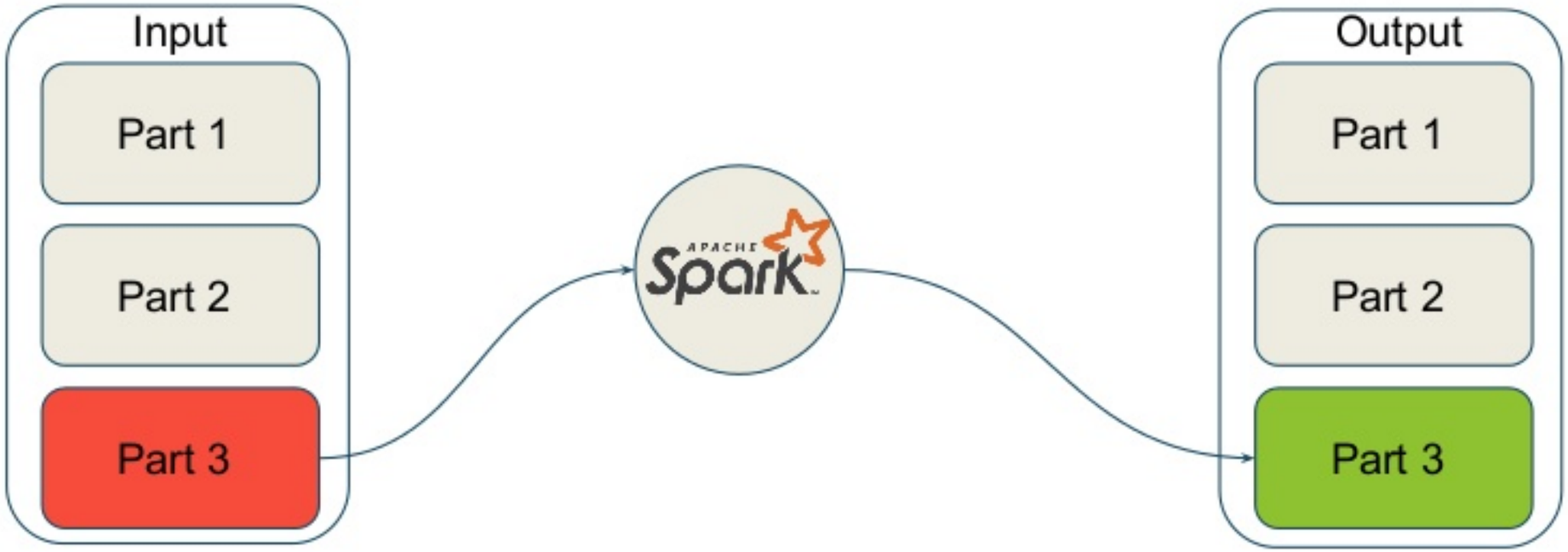
```
def write_output(self, path, out):
    if self.output_format == 'json':
        save_as_json_file(out, path)
    elif self.output_format == 'parquet':
        save_as_parquet_file(out, path)
```

Job Types

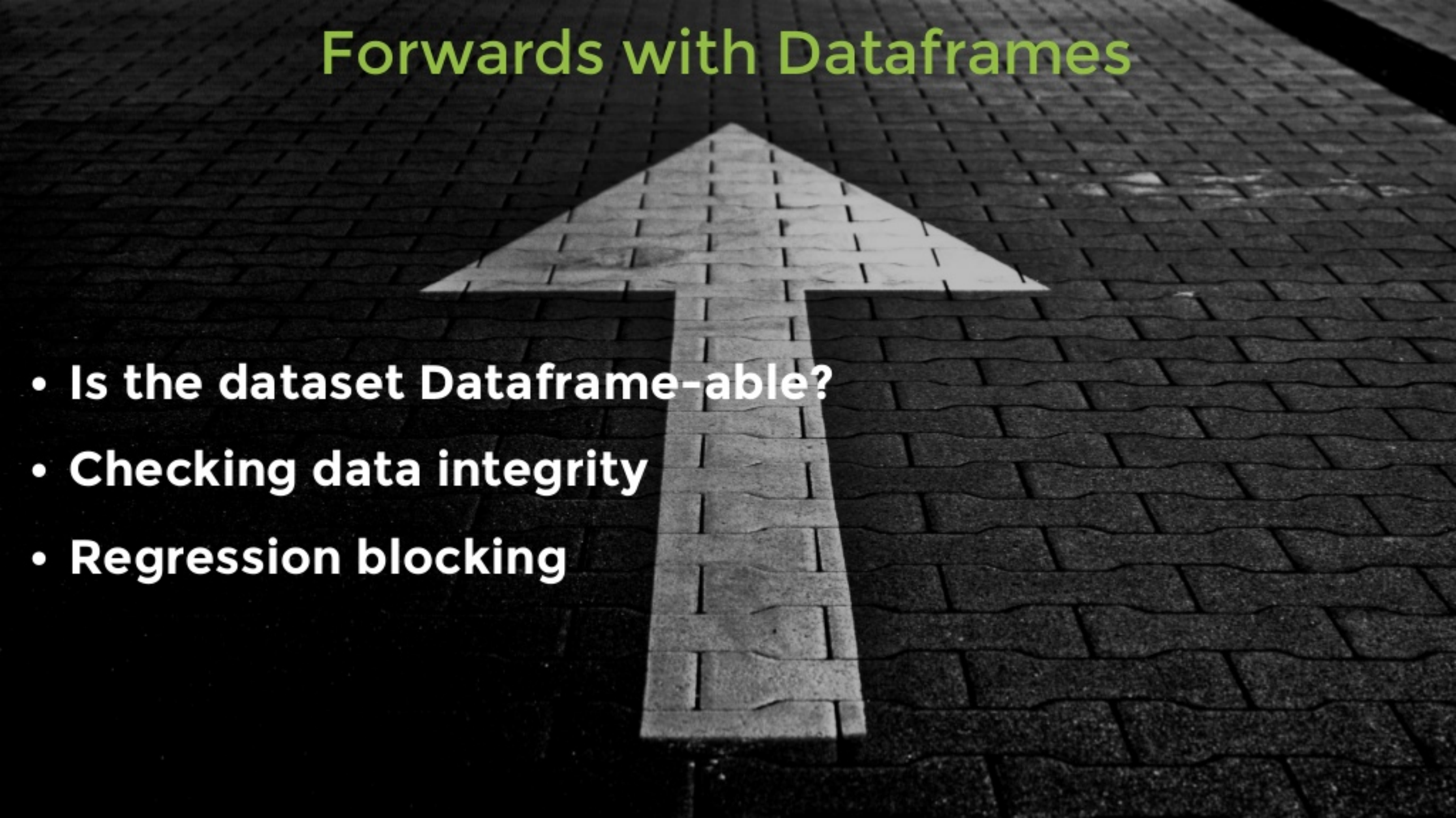
- Full Drop : Jobs that rebuild their output from scratch every time.

Job Types

- Incremental Drop : Jobs that build their output incrementally.



Forwards with Dataframes



- Is the dataset Dataframe-able?
- Checking data integrity
- Regression blocking

Will it Dataframe?

	Python	Dataframe
long	any size	sql.LongType (-2^63, 2^63-1)
Decimal	28 prec 28 sig digits	sql.DecimalType(38, 18) [20 digits].[18 digits]

Long:

```
if (v < -9223372036854775808 or v > 9223372036854775807):
  logger.warn('Field: {}, Value: {}. long will not fit inside sql.LongType'.format(k, v))
```

Decimal:

```
DECIMAL_CONTEXT = Context(prec=38, Emax=19, Emin=19, traps=[Inexact])
try:
  DECIMAL_CONTEXT.create_decimal(v)
except:
  logger.warn('Field: {}, Value: {}. Decimal will not fit inside sql.DecimalType(38,18)'.format(k, v))
```

Will it Dataframe?

Record based format

```
[{'a': 1},
{'a': 2, 'b': True} ...]
```

Column based format

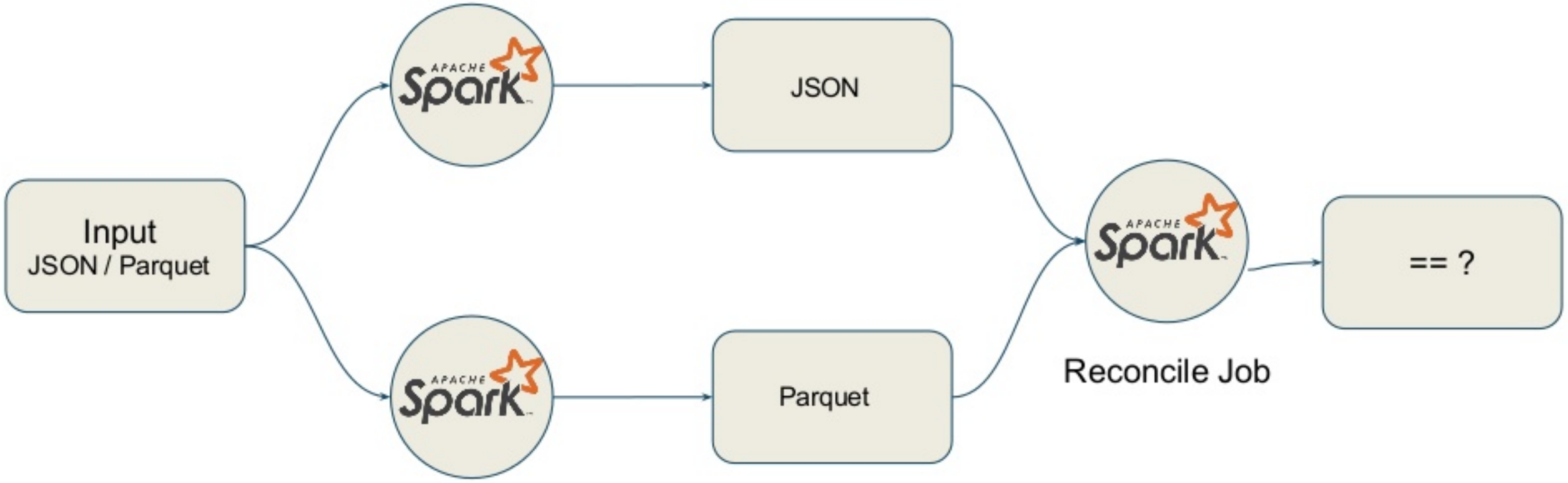
a	b
1	None
2	True

```
if 'b' in data:
  # DataFrame path
else:
  # RDD path
```

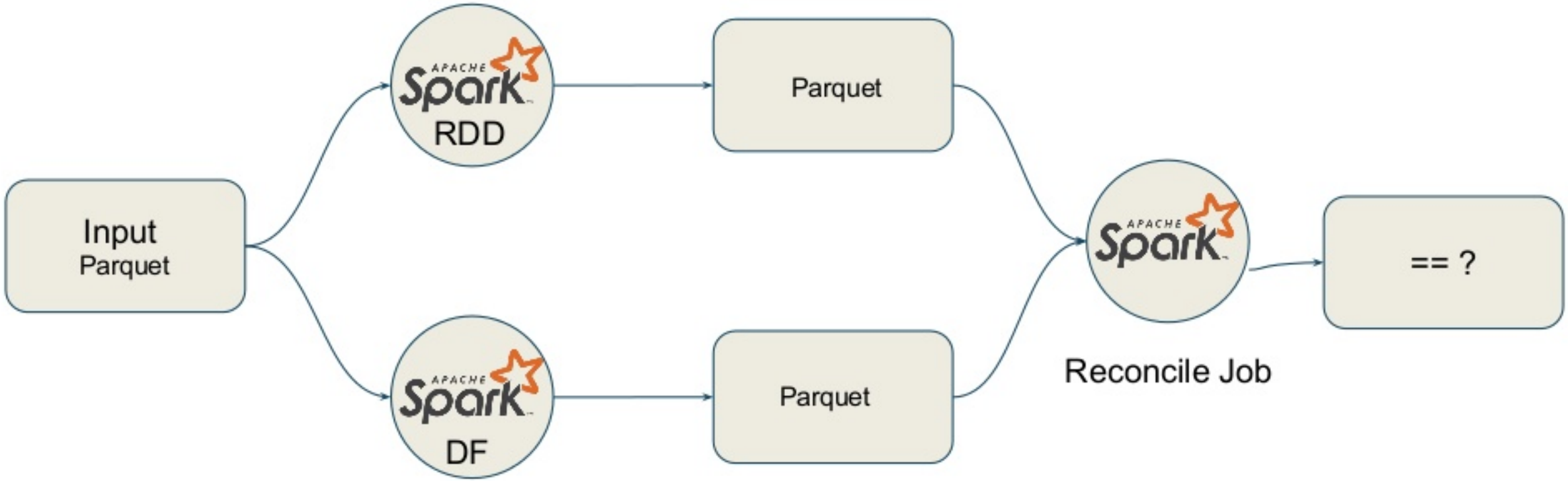
Optional Blocking

```
def test_no_new_optional_jobs():
    white_list = {...}
    jobs_with_optionals = find_jobs_with_optionals()
    new_jobs_with_optionals = jobs_with_optionals - white_list
    assert len(new_jobs_with_optionals) == 0
```

Checking Data Integrity



Checking Data Integrity



Regression Blocking

```
def test_no_new_json_jobs():
    white_list = {...}
    jobs_outputing_json = find_json_jobs()
    new_json_jobs = jobs_outputing_json - white_list
    assert len(new_json_jobs) == 0
```

Results

10x

FASTER

0.7x

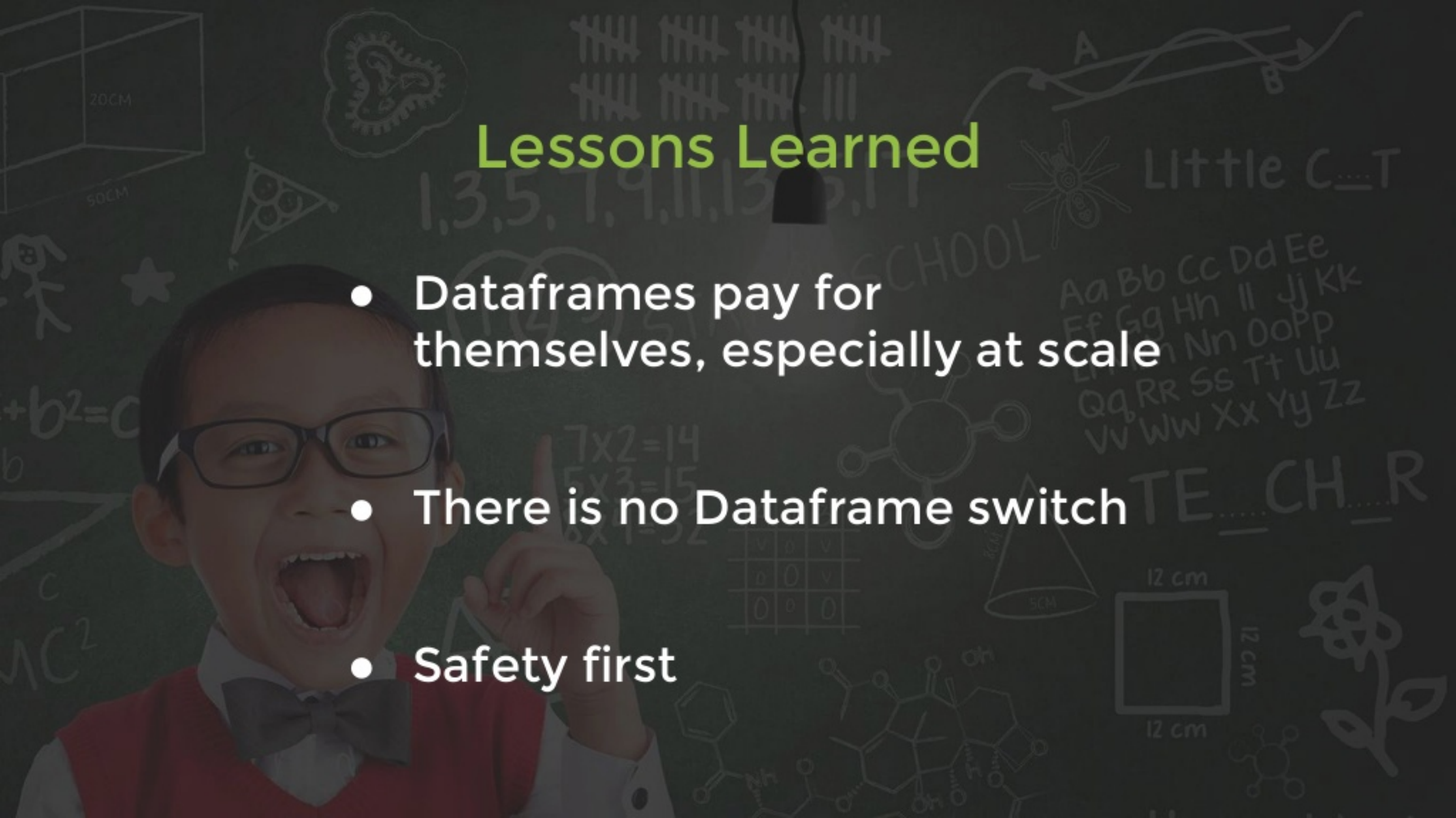
RESOURCES USED

1.3x

JAVA MEM

SQL

DATA ACCELERATED



Lessons Learned

- Dataframes pay for themselves, especially at scale
- There is no Dataframe switch
- Safety first


We're Hiring
shopify.com/careers
shopify