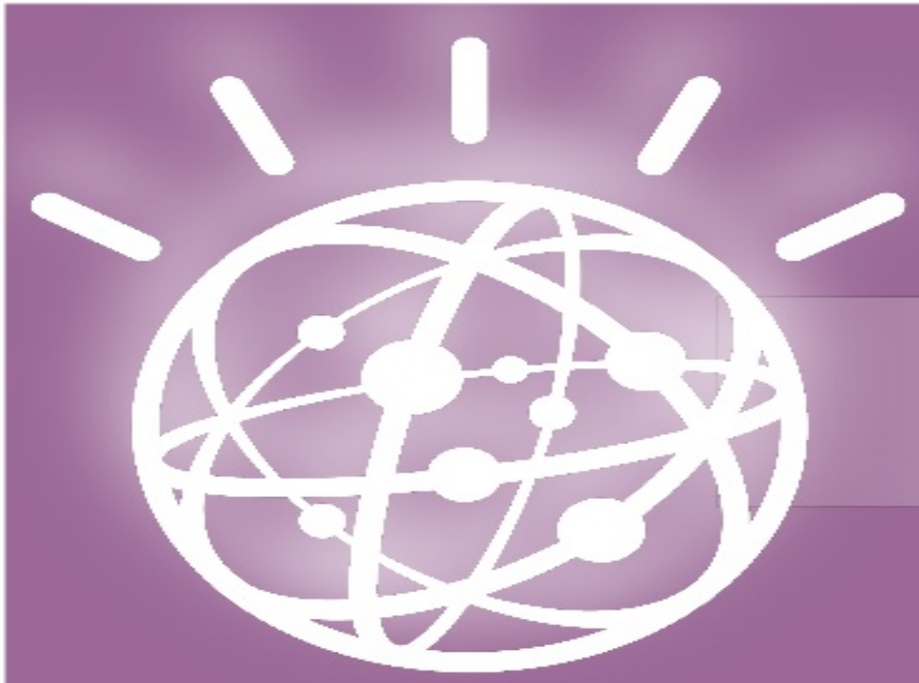


FROM SINGLE-TENANT HADOOP TO 3000 TENANTS IN APACHE SPARK

RUBEN PULIDO
BEHAR VELIQI

IBM WATSON ANALYTICS FOR SOCIAL MEDIA
IBM GERMANY R&D LAB

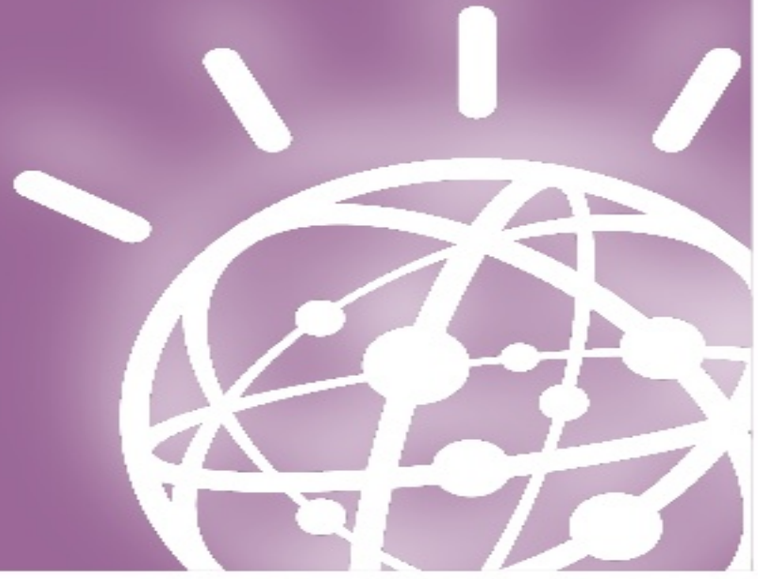
WHAT IS WATSON ANALYTICS FOR SOCIAL MEDIA

PREVIOUS ARCHITECTURE

THOUGHT PROCESS TOWARDS MULTITENANCY

NEW ARCHITECTURE

LESSONS LEARNED



WATSON ANALYTICS

A data **analytics solution** for **business users** in the **cloud**

WATSON ANALYTICS FOR SOCIAL MEDIA

Part of Watson Analytics
Allows users to
harvest and **analyze social media content**
To understand how brands, products, services, social issues, …
are **perceived**

Our previous architecture: a single-tenant "big data" ETL batch pipeline

Evolving Topics

Text Analytics

BigInsights

DB2

Cognos BI

Content is stored in HDFS...

...analyzed within our components...

...stored in HDFS again...

...and loaded into DB2

What people think about **Social Media data rates**
What **Stream Processing** is really good at

Photo by Ghislain Bonneau,gbphotodidactical.ca

What the size of a customer specific social media data-feed really is

Our requirement:
Efficiently processing
"trickle feeds" of data

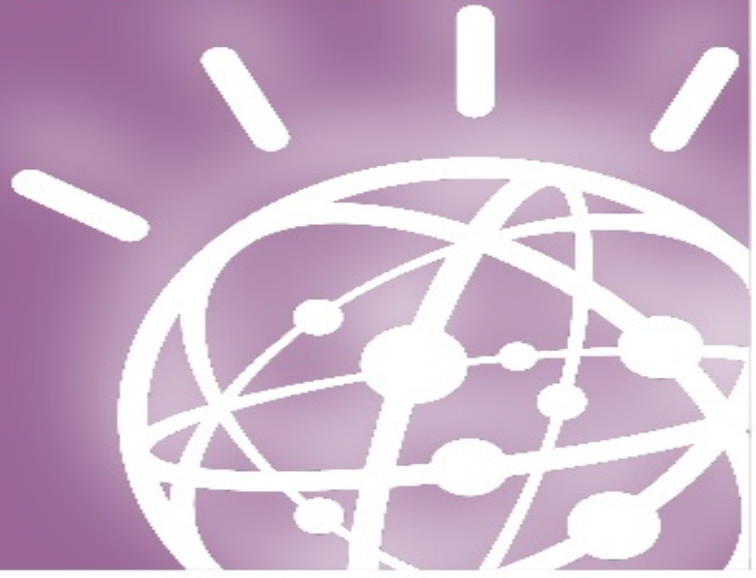
WATSON ANALYTICS FOR SOCIAL MEDIA

PREVIOUS ARCHITECTURE

THOUGHT PROCESS TOWARDS MULTITENANCY

NEW ARCHITECTURE

LESSONS LEARNED



Step 1: Revisit our analytics workload

Concept detection

Everyone I ask says my **PhoneA** is way better than my brother's **PhoneB**

Concept level sentiment detection

Everyone I ask says my **PhoneA** is way better than my brother's PhoneB → **PhoneA: Positive**

Everyone I ask says my PhoneA is way better than my brother's **PhoneB** → **PhoneB: Negative**

Author features detection

My wife's PhoneA died yesterday. → **Author**: Is Married = **true**

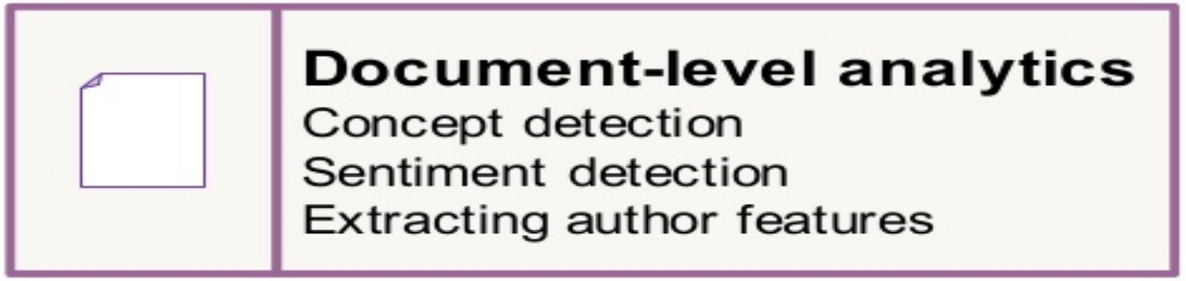
My son wants me to get him PhoneB. → **Author**: Has Children = **true**

Creation of author profiles

Author: Is Married = **true**
Author: Has Children = **true**

Author: Is Married = **true**, Has Children = **true**

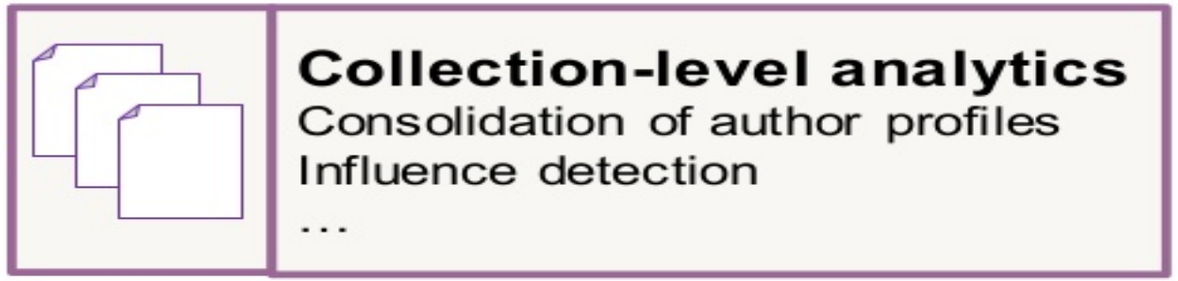
Step 2: Assign the right processing model for the right workload



Document-level analytics
Concept detection
Sentiment detection
Extracting author features

Can easily work on a data **stream**

User gets additional value from **every document**



Collection-level analytics
Consolidation of author profiles
Influence detection
…

Most algorithms need a **batch** of documents (or all documents)

User value is getting the "**big picture**"

Step 3: Turn “trickle feeds” into a stream through multitenancy

Document-level analytics makes sense on a stream…

….document analysis for a single tenant often does **not** justify stream processing…

…but what if we process all documents for **all tenants** in a **single system?**

Step 4: Address Multitenancy Requirements

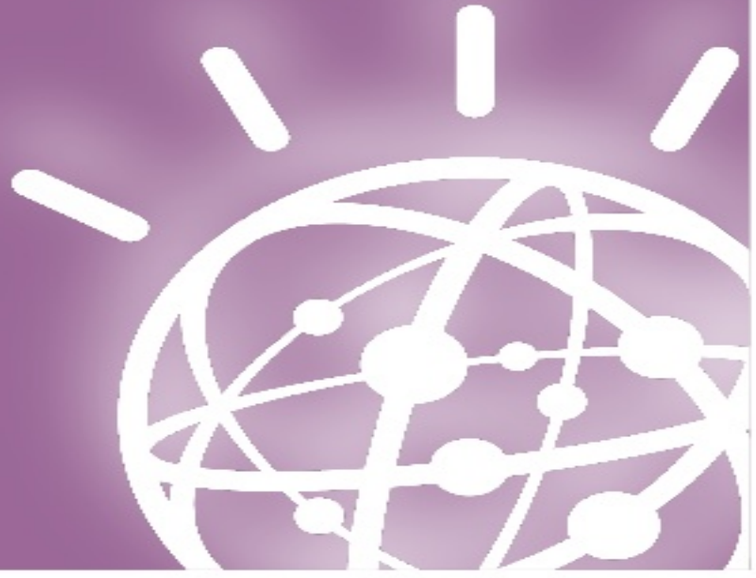
Minimize "switching costs" when analyzing data for different **tenants**

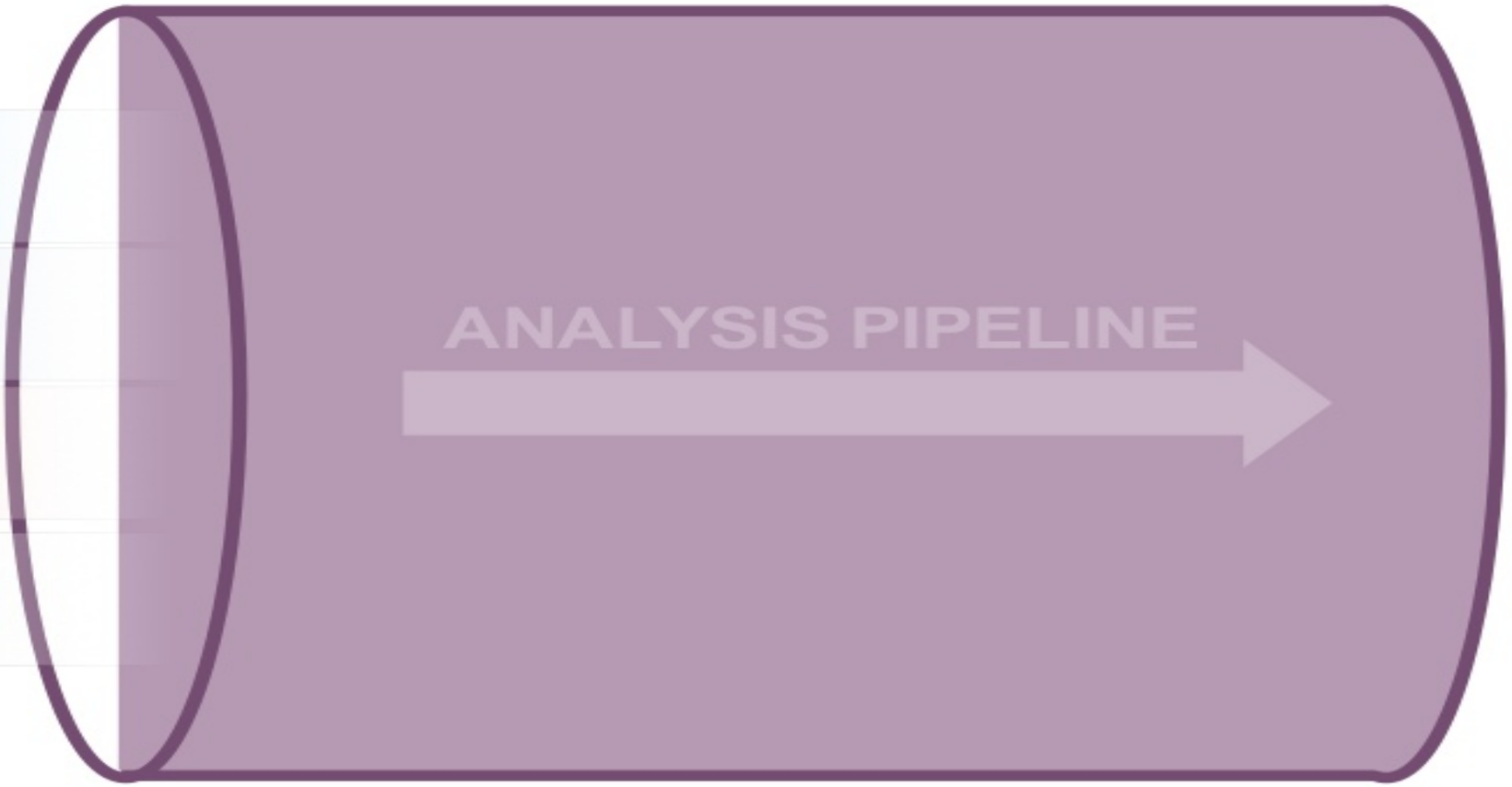
- **Separated** text analytics steps into:
  - **tenant-specific**
  - **language-specific**
- **Optimized** **tenant-specific** steps for **"low latency" switching**

Avoid storing tenant state in processing components

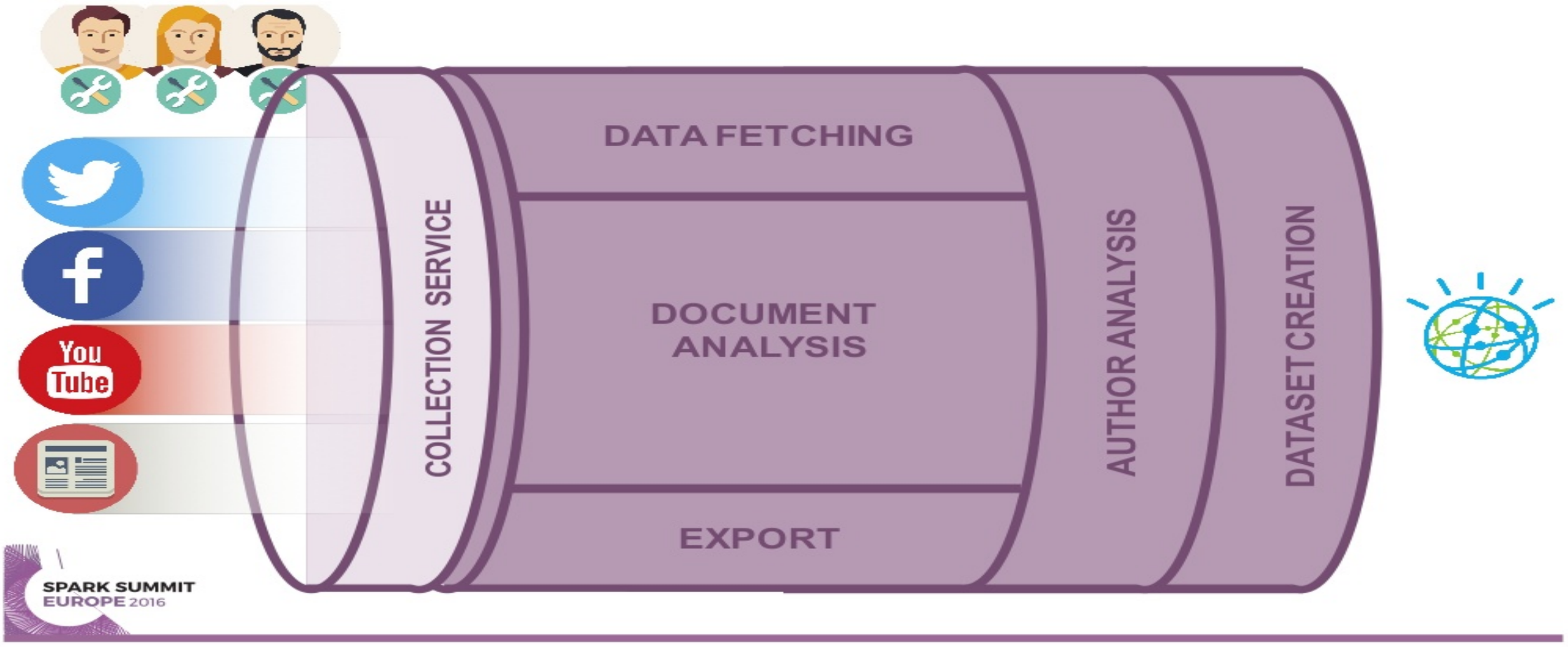
- Using Zookeeper as distributed configuration store

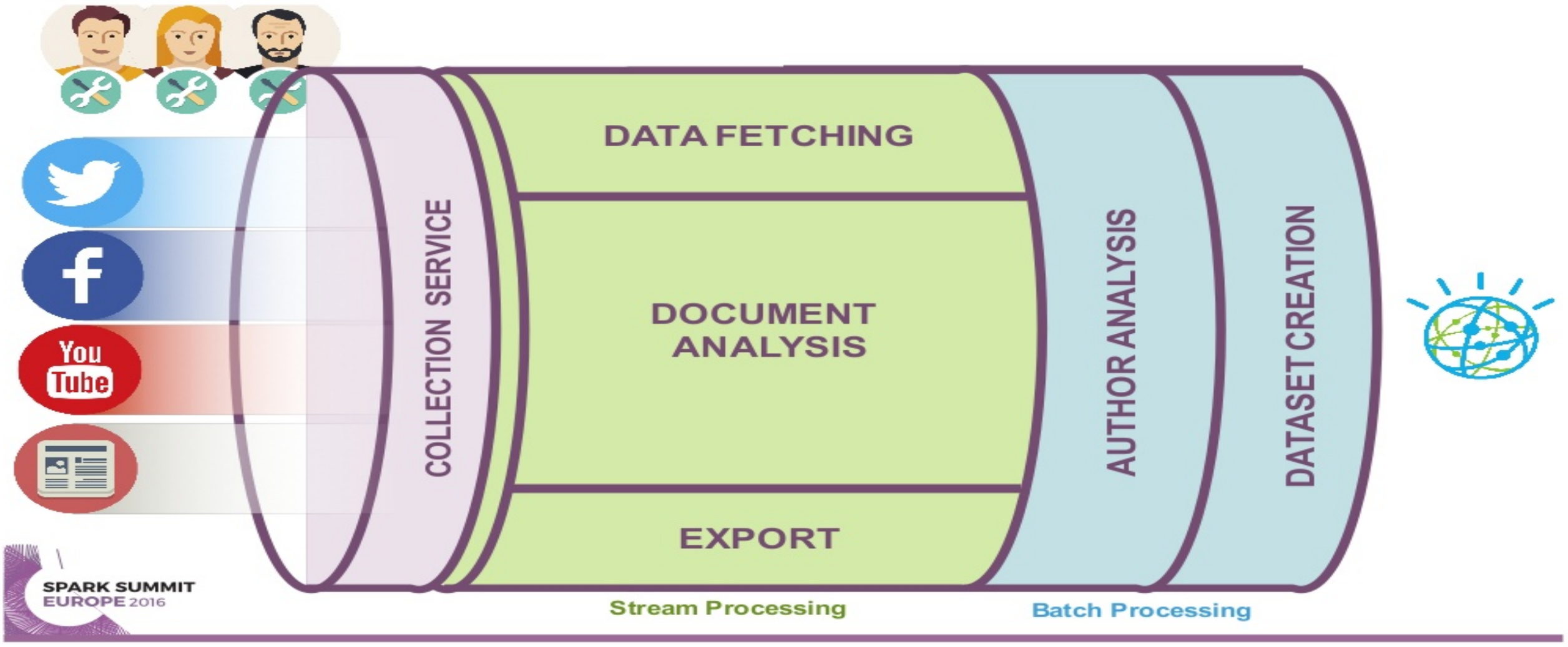
WATSON ANALYTICS FOR SOCIAL MEDIA
PREVIOUS ARCHITECTURE
THOUGHT PROCESS TOWARDS MULTITENANCY
NEW ARCHITECTURE
LESSONS LEARNED

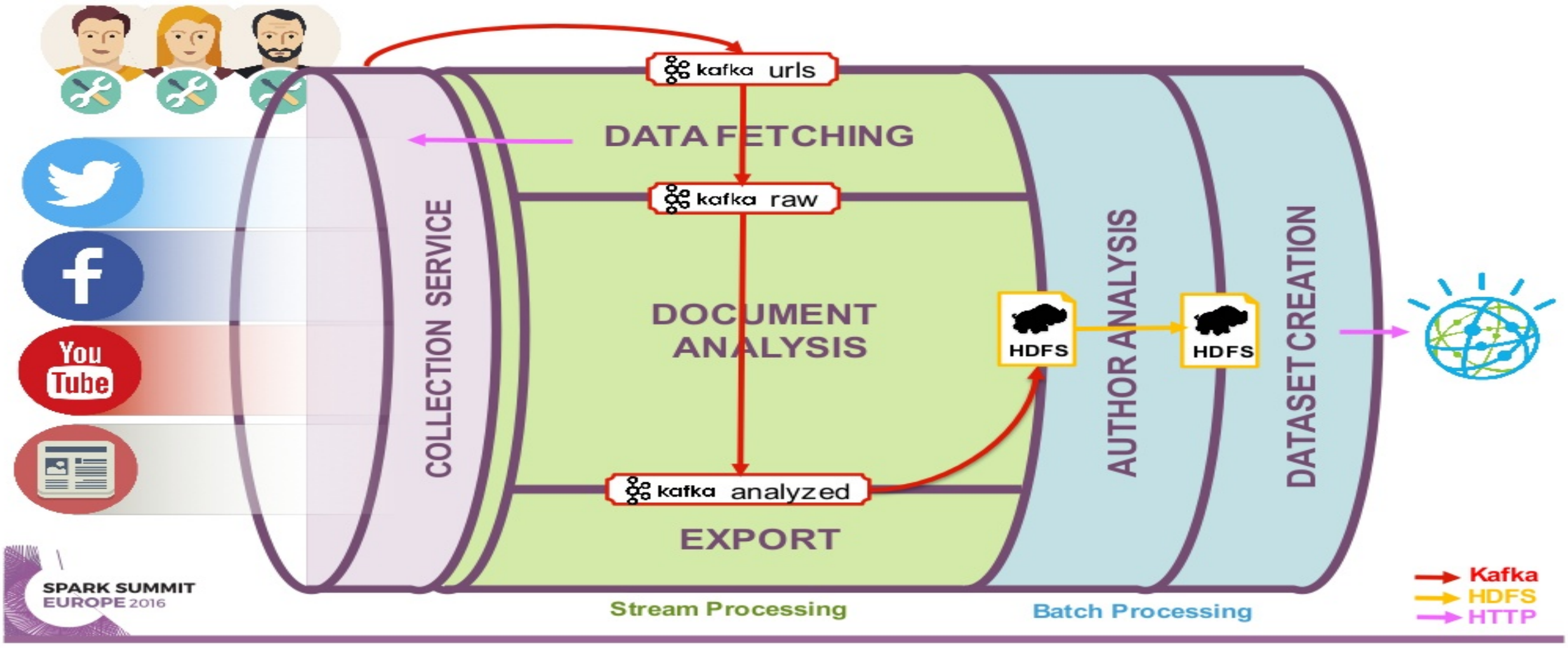


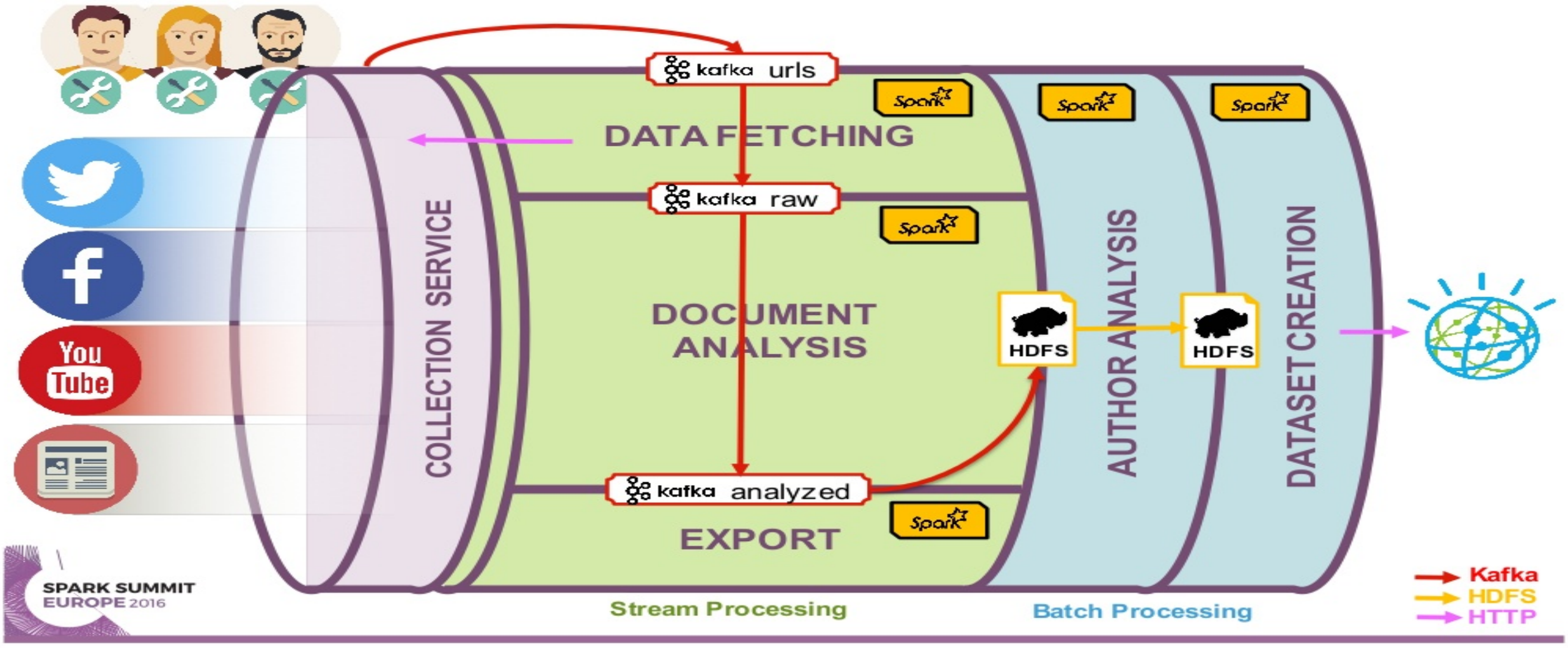

ANALYSIS PIPELINE

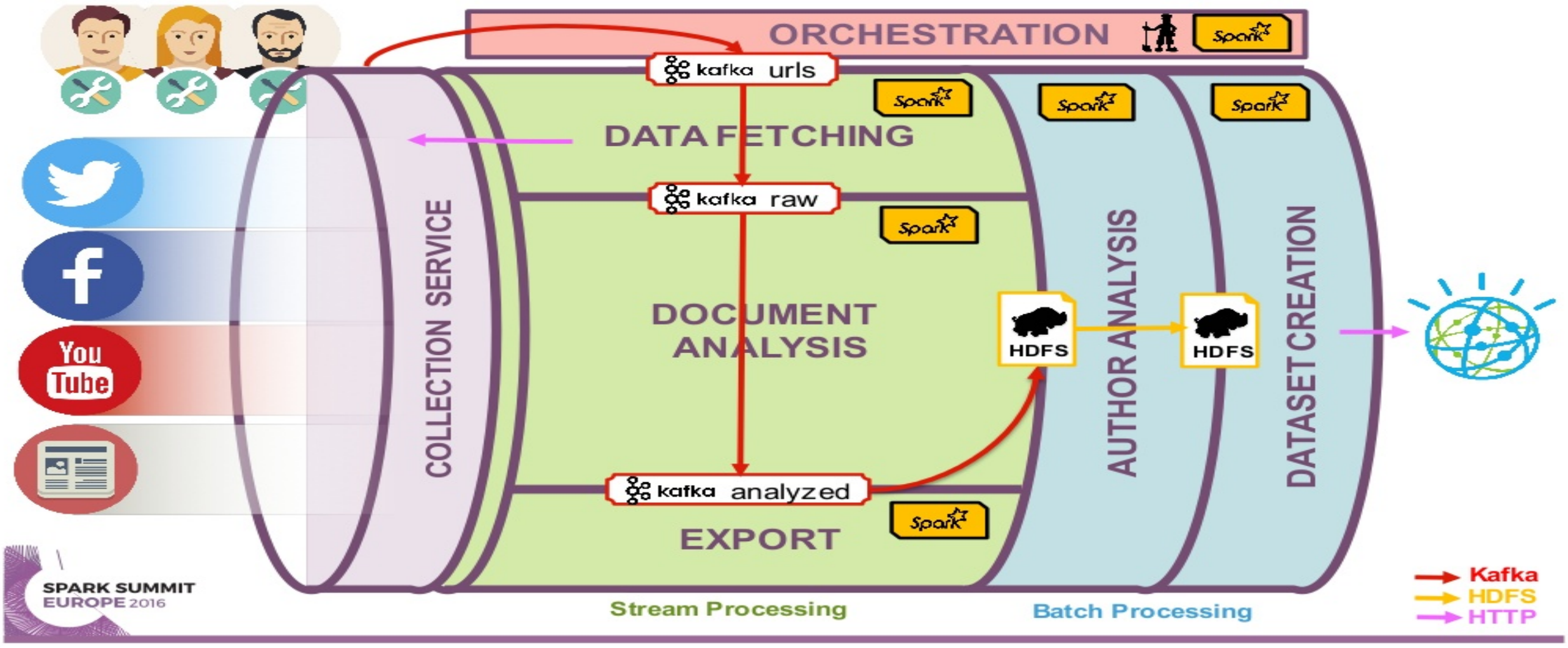

COLLECTION SERVICE
ANALYSIS PIPELINE
You
Tube

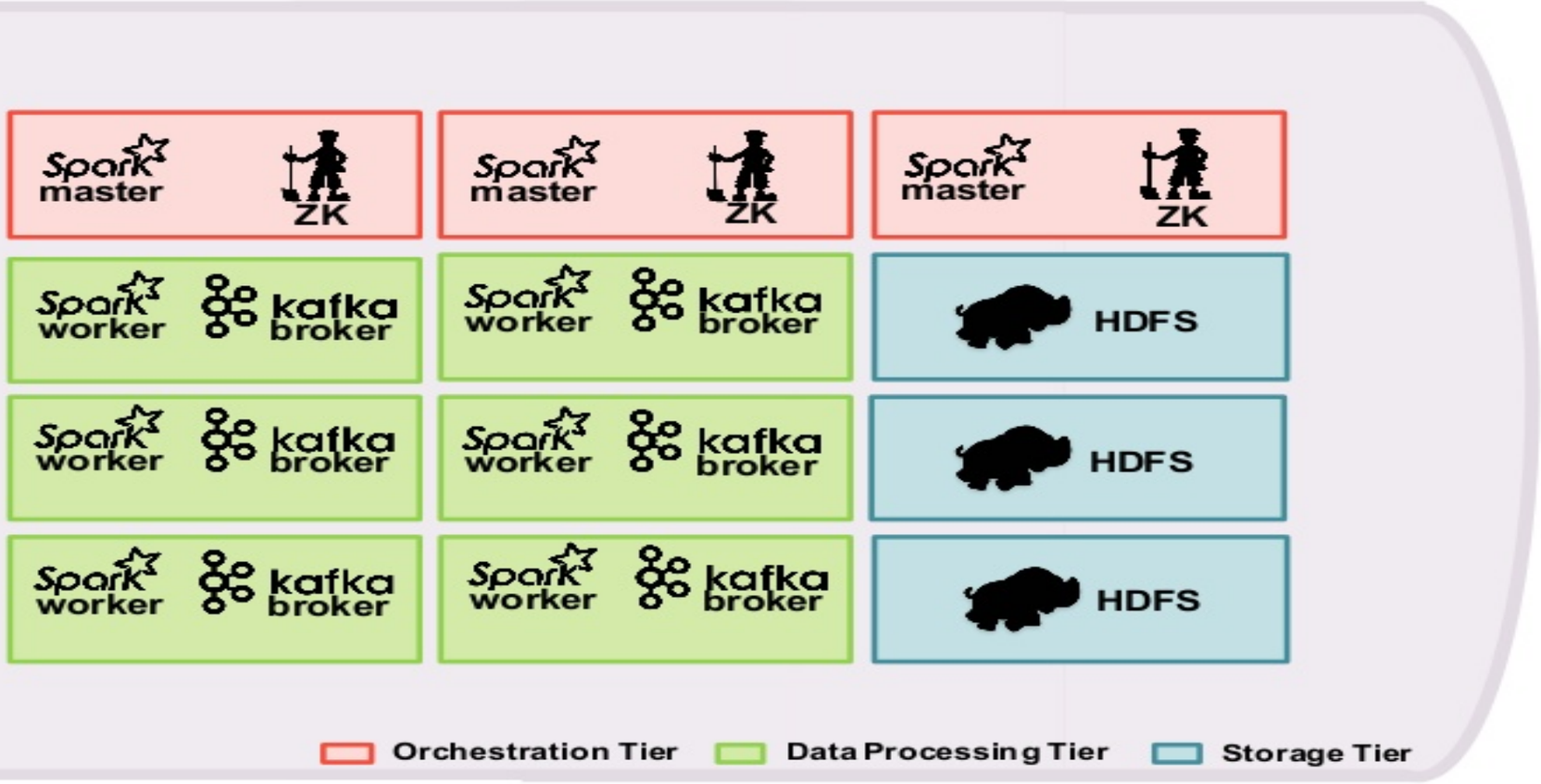

COLLECTION SERVICE
DATA FETCHING
DOCUMENT ANALYSIS
EXPORT
AUTHOR ANALYSIS
DATASET CREATION
You Tube

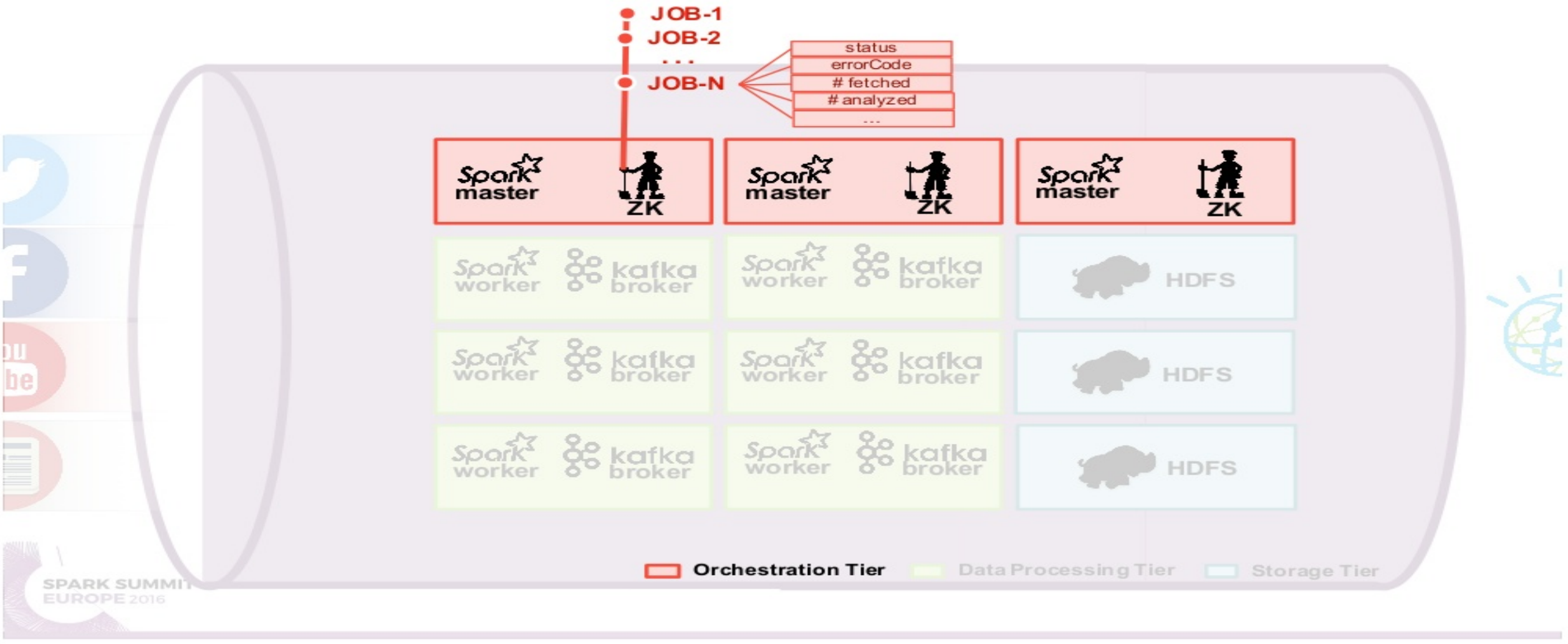

COLLECTION SERVICE
DATA FETCHING
DOCUMENT ANALYSIS
EXPORT
AUTHOR ANALYSIS
DATASET CREATION
Stream Processing
Batch Processing

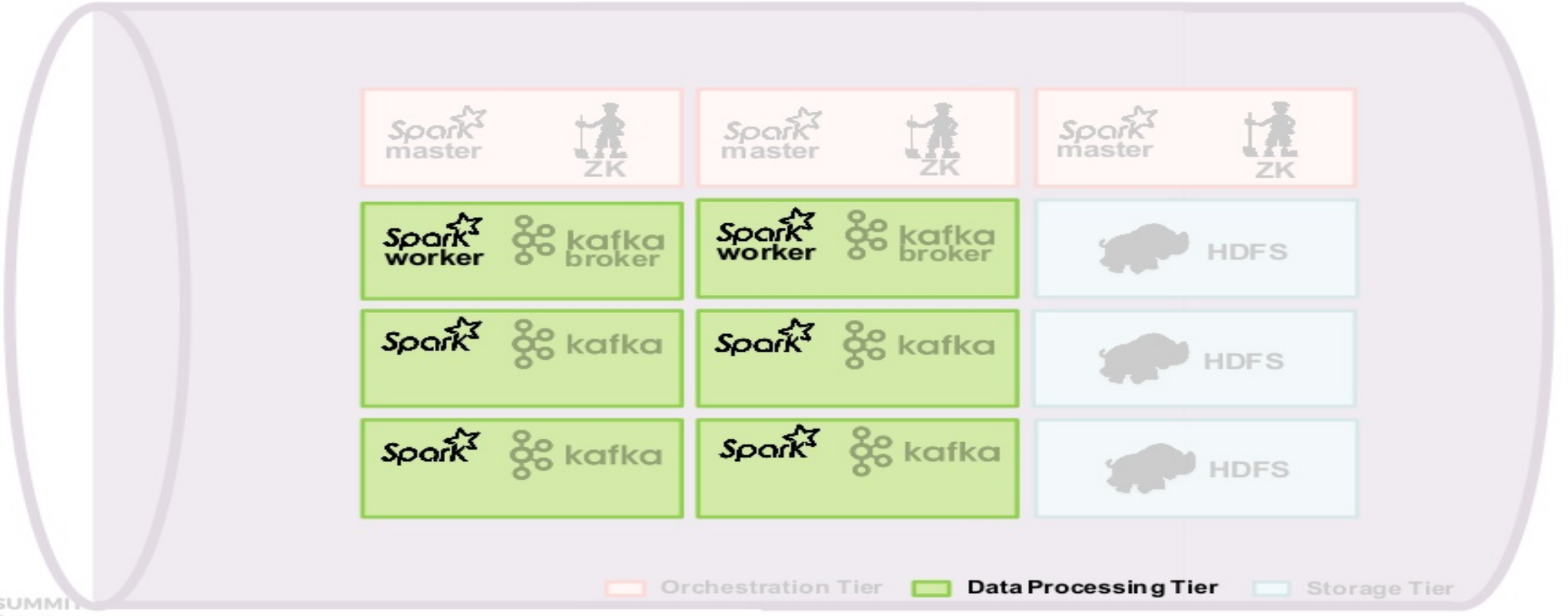

kafka urls
DATA FETCHING
kafka raw
DOCUMENT ANALYSIS
kafka analyzed
EXPORT
COLLECTION SERVICE
HDFS
HDFS
AUTHOR ANALYSIS
DATASET CREATION
You Tube
SPARK SUMMIT EUROPE 2016
Stream Processing
Batch Processing
Kafka
HDFS
HTTP


kafka urls
DATA FETCHING
kafka raw
DOCUMENT ANALYSIS
kafka analyzed
EXPORT
Spark
Spark
Spark
Spark
Spark
COLLECTION SERVICE
HDFS
HDFS
AUTHOR ANALYSIS
DATASET CREATION
You Tube
SPARK SUMMIT EUROPE 2016
Stream Processing
Batch Processing
Kafka
HDFS
HTTP


ORCHESTRATION
kafka urls
DATA FETCHING
kafka raw
DOCUMENT ANALYSIS
kafka analyzed
EXPORT
COLLECTION SERVICE
HDFS
HDFS
AUTHOR ANALYSIS
DATASET CREATION
Stream Processing
Batch Processing
Kafka
HDFS
HTTP
SPARK SUMMIT EUROPE 2016


Spark master
ZK
Spark master
ZK
Spark master
ZK
Spark worker
kafka broker
Spark worker
kafka broker
HDFS
Spark worker
kafka broker
Spark worker
kafka broker
HDFS
Spark worker
kafka broker
Spark worker
kafka broker
HDFS
Orchestration Tier
Data Processing Tier
Storage Tier


JOB-1
JOB-2
...
JOB-N
status
errorCode
fetched
analyzed
...
Spark master
ZK
Spark worker
kafka broker
HDFS
Orchestration Tier
Data Processing Tier
Storage Tier


Spark master
ZK
Spark master
ZK
Spark master
ZK
Spark worker
kafka broker
Spark worker
kafka broker
HDFS
Spark
kafka
Spark
kafka
HDFS
Spark
kafka
Spark
kafka
HDFS
Orchestration Tier
Data Processing Tier
Storage Tier

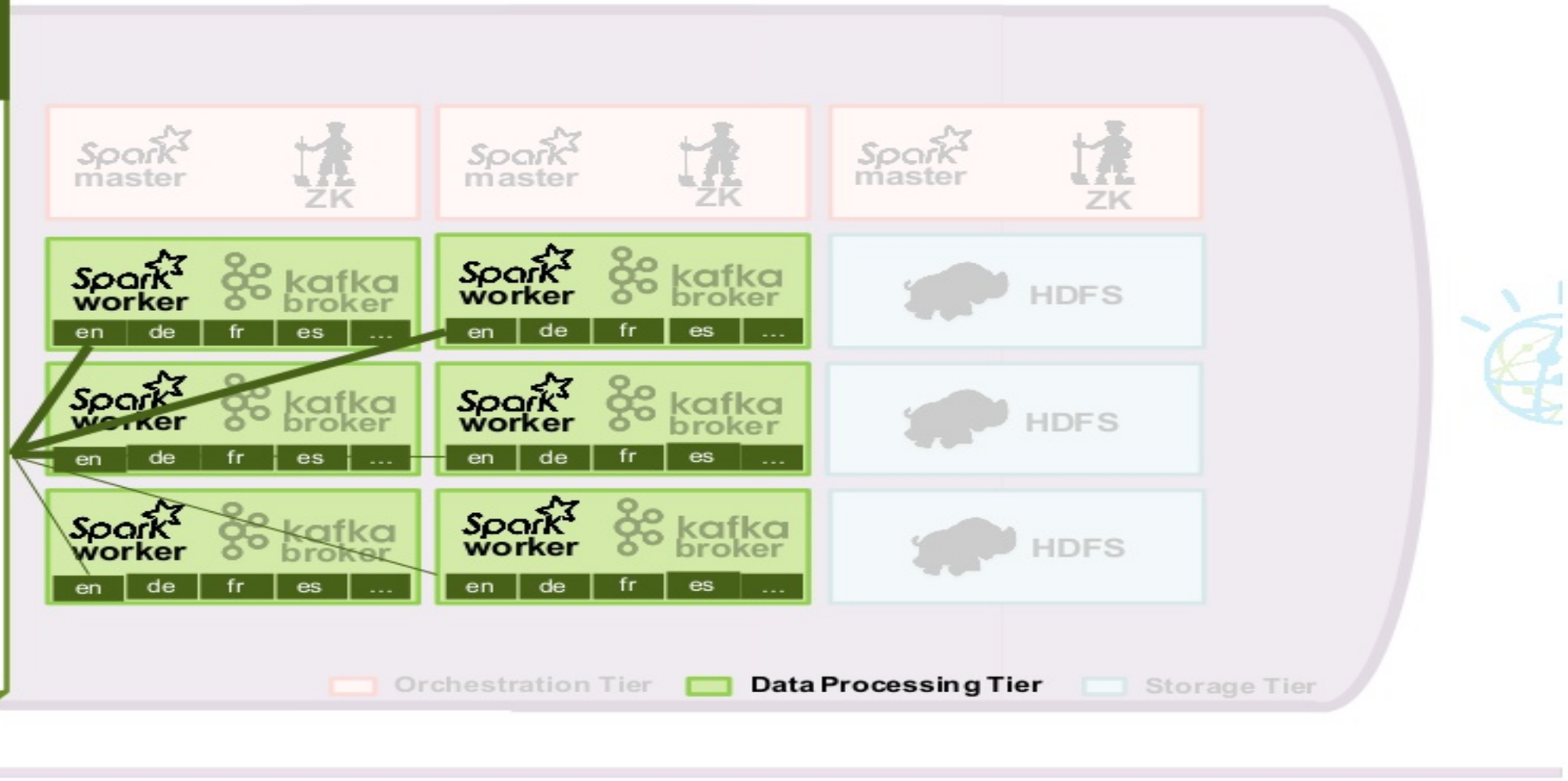
LANGUANGE SPECIFIC ANALYSIS

- **Expensive** in **Initialization**
- **High memory** usage

  → **Not appropriate for Caching**

- **Stable across tenants**
- **One analysis engine**
  - **per language**
  - **per Spark worker**

  → **created at deployment time**
- Processing **threads co-use these engines**



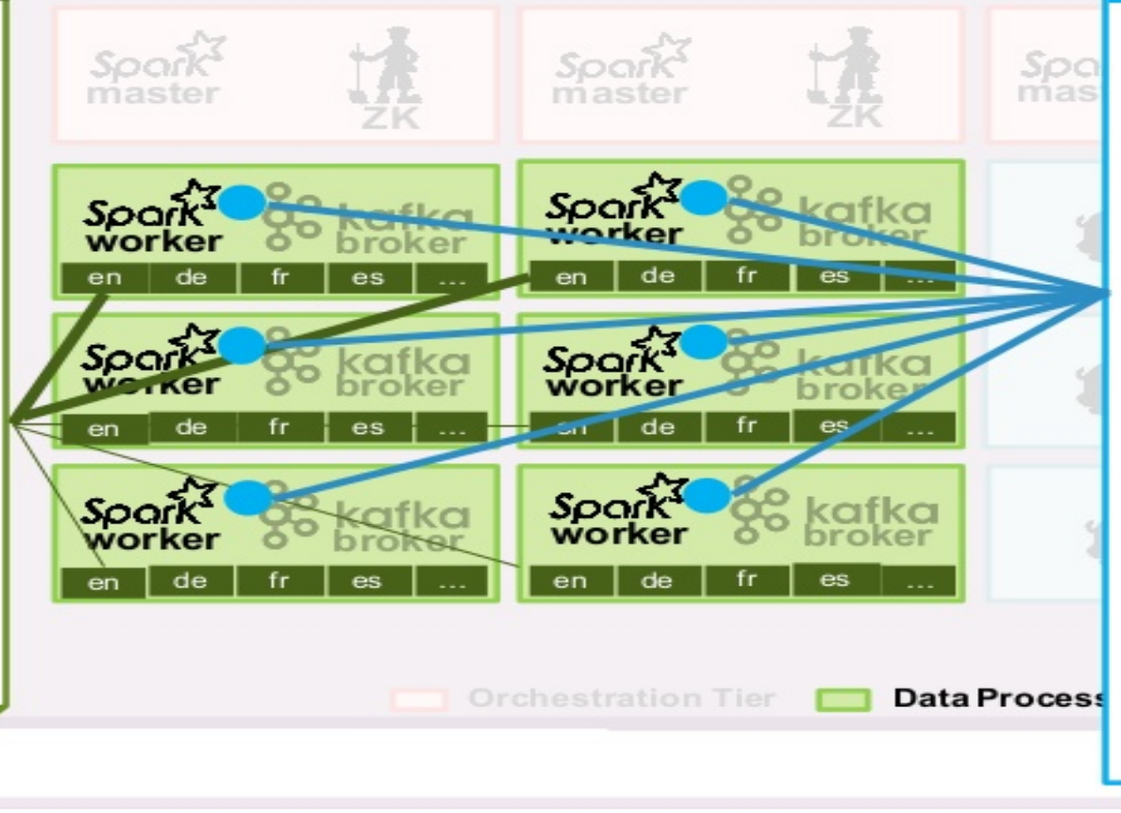
LANGUANGE SPECIFIC ANALYSIS

- **Expensive** in **Initialization**
- **High memory** usage

  → **Not appropriate for Caching**

- **Stable across tenants**
- **One analysis engine**
  - **per language**
  - **per Spark worker**

  → **created at deployment time**
- Processing **threads co-use these engines**



TENANT SPECIFIC ANALYSIS

- **Low memory** usage compared to language analysis engines

  → **LRU-cache** of **100** tenant specific analysis engines **per** Spark **worker**

- Very **low tenant-switching overhead**

kafka raw

READ

```
val documents = KafkaUtils.createDirectStream[ShippingLabel, AnalysisDocument, ...
```

kafka raw

READ

```
val documents = KafkaUtils.createDirectStream[ShippingLabel, AnalysisDocument, ...
```

ANALYZE

```
val analyzedDocuments = documents.map {
  case (label, doc) => analyze(label, doc)
}
```

AQL

```
create view Parent as
extract regex /
        (my | our)
        (sons?|daughters?|kids?|baby(  boy|girl)?|babies|child|children)
        /
with flags 'CASE_INSENSITIVE' on D.text as match from  Document D
```

Author Uses	Author is Marri...	Author is Parent	Conversation S...	Conve
	True	True		http:/
		True		http:/

kafka raw

READ

```
val documents = KafkaUtils.createDirectStream[ShippingLabel, AnalysisDocument, ...
```

ANALYZE

```
val analyzedDocuments = documents.map {
  case (label, doc) => analyze(label, doc)
}
```

kafka analyzed

WRITE

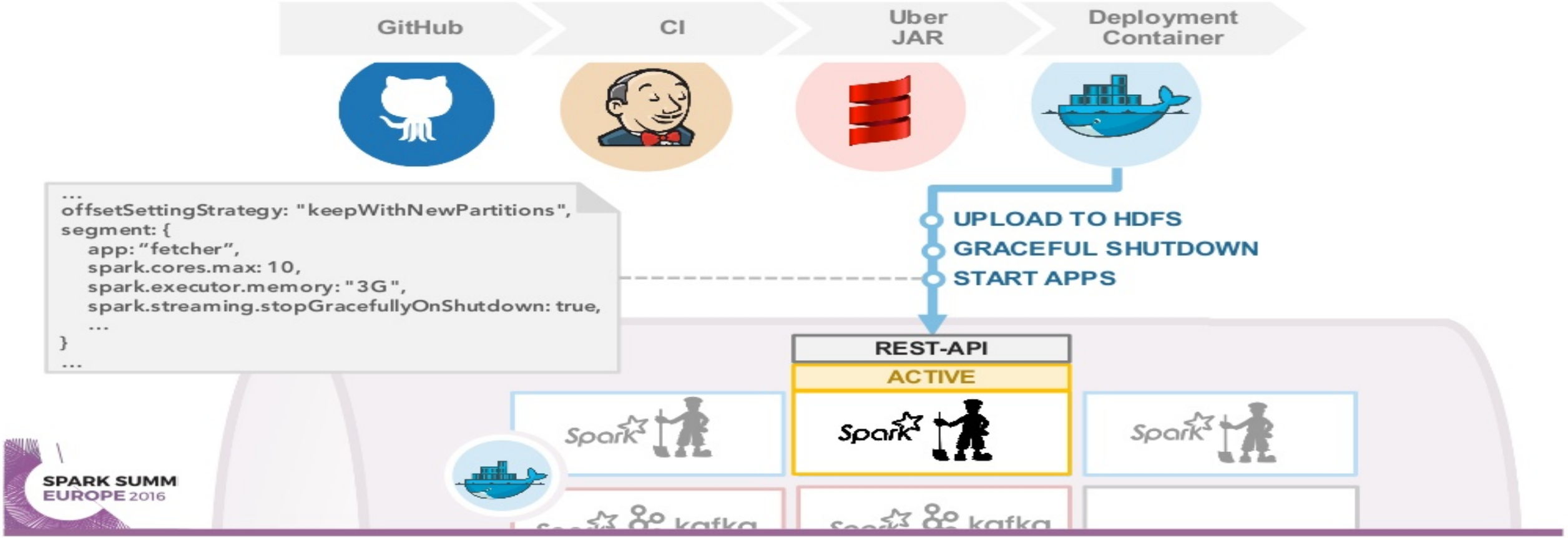
```
val results = kafka.write(messages)
trackProgess(results)
```

More than **3000 tenants**
Between **150-300 analysis jobs per day**
Between **25K-4M documents analyzed per job**
3 data centers in Toronto, Dallas, Amsterdam
Cluster of **10 VMs**: each with 16 Cores and 32G RAM

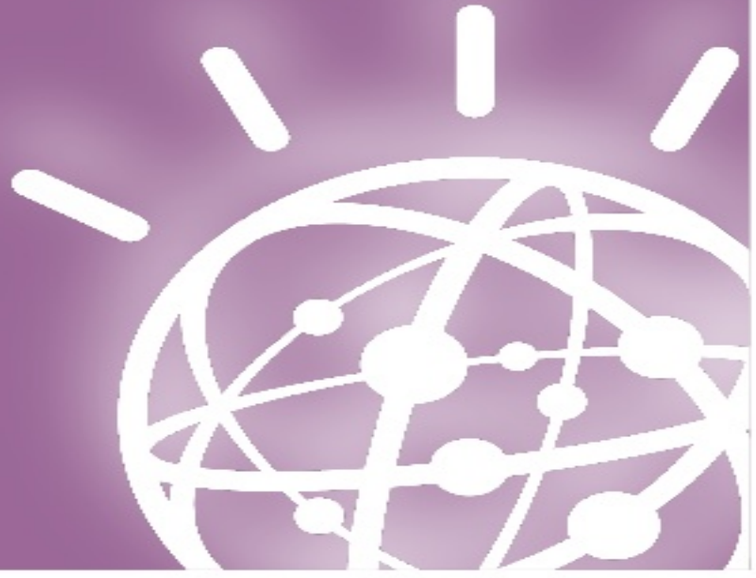
Text-Analytics Throughput:

Full Stack: from Tokenization to Sentiment / Behavior
Mixed Sources: ~60K documents per minute
Tweets only: 150K Tweets per minute

DEPLOYMENT

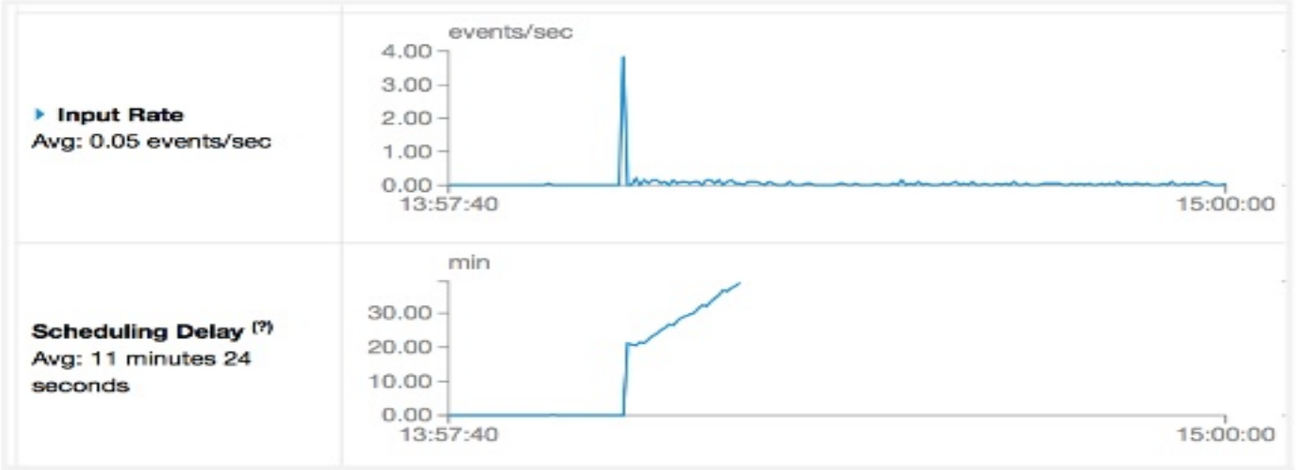

GitHub
CI
Uber JAR
Deployment Container
...
offsetSettingStrategy: "keepWithNewPartitions",
segment: {
app: “fetcher”,
spark.cores.max: 10,
spark.executor.memory: "3G",
spark.streaming.stopGracefullyOnShutdown: true,
...
}
...
UPLOAD TO HDFS
GRACEFUL SHUTDOWN
START APPS
REST-API
ACTIVE
Spark
Spark
Spark
kafka
kafka

WATSON ANALYTICS FOR SOCIAL MEDIA
PREVIOUS ARCHITECTURE
THOUGHT PROCESS TOWARDS MULTITENANCY
NEW ARCHITECTURE
LESSONS LEARNED

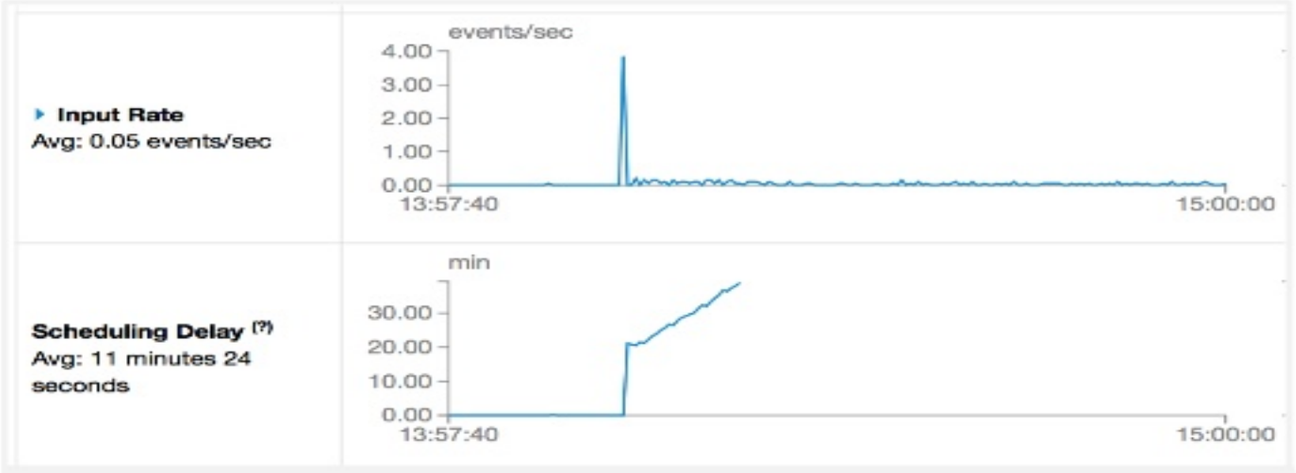


```
private class JobMonitor(segments: Seq[BatchSegment], zk: ZooKeeperClient, ctx: SparkContext,

              val tuples =
                for (segment <- segments; job <- zk.jobs if job.state == Running)
                  yield (segment, job)

              tuples
                ...
                // update progress on zookeeper
                // trigger the batch phases
```

"**Driver Only**" – No Stream or Batch Processing
Uses Spark's **Fail-Over** mechanisms
No implementation of custom **master election** necessary



KAFKA EVENT → HTTP REQUEST → DOCUMENT BATCH



KAFKA EVENT → HTTP REQUEST → DOCUMENT BATCH

wrong workload for Spark`s
micro-batch based stream-processing
Hard to find appropriate **scheduling interval**
builds high **scheduling delays**
hard to **distribute equally** across the cluster
No benefit of Spark settings like
maxRatePerPartition or **backpressure.enabled**
sometimes leading to **OOM**

ORCHESTRAION | DATA FETCHING | PARTITIONING | EXPORTING | AUTHOR ANALYSIS

KAFKA EVENT → HTTP REQUEST → DOCUMENT BATCH

NON-SPARK APPLICATION

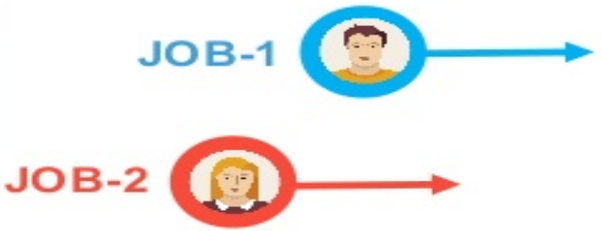
wrong workload for Spark`s
micro-batch based stream-processing
Hard to find appropriate **scheduling interval**
builds high **scheduling delays**
hard to **distribute equally** across the cluster
No benefit of Spark settings like
maxRatePerPartition or **backpressure.enabled**
sometimes leading to **OOM**

Run the fetcher as **separate application**
outside of Spark – "long running" thread
Fetch documents from social media providers
Write to raw documents to Kafka
Let the pipeline analyze these documents
backpressure.enabled = true

ORCHESTRAION | DATA FETCHING | **PARTITIONING** | EXPORTING | AUTHOR ANALYSIS

Job-ID

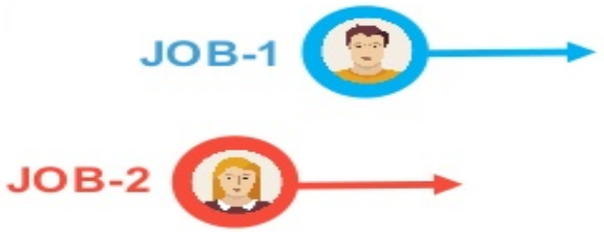
- Jobs don't influence each other
- Run fully in parallel
- Cluster is not fully/equally utilized



ORCHESTRAION | DATA FETCHING | **PARTITIONING** | EXPORTING | AUTHOR ANALYSIS

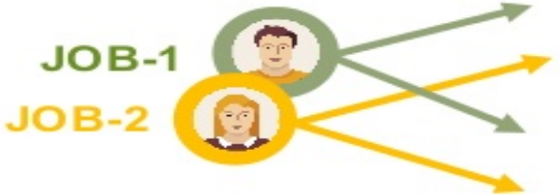
Job-ID

- Jobs don't influence each other
- Run fully in parallel
- Cluster is not fully/equally utilized



"Random" + All URLs At Once

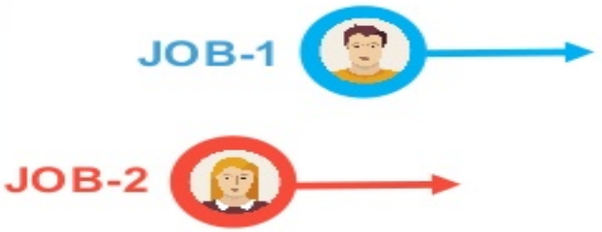
- Cluster is fully utilized
- Workload equally distributed
- Jobs run sequentially



ORCHESTRAION | DATA FETCHING | PARTITIONING | EXPORTING | AUTHOR ANALYSIS

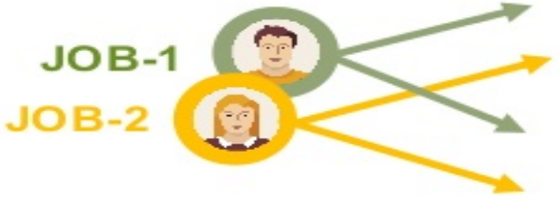
Job-ID

- Jobs don't influence each other
- Run fully in parallel
- Cluster is not fully/equally utilized



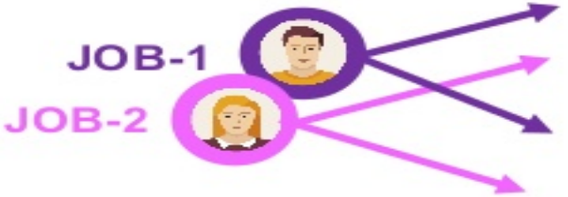
"Random" + All URLs At Once

- Cluster is fully utilized
- Workload equally distributed
- Jobs run sequentially

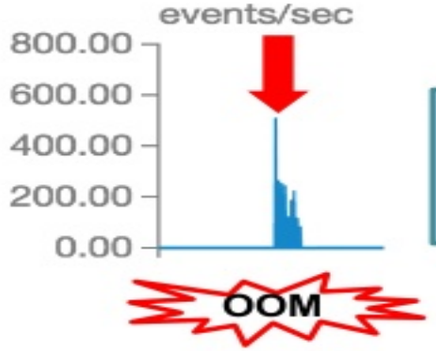


"Random" + "Next" URLs

- Cluster is fully utilized
- Workload equally distributed
- Jobs run in parallel



Write documents for each job to a separate HDFS directory.
First version: Group documents using Scala collections API



User Memory Region

```
jobIdDocumentDStream.foreachRDD {
  rdd => {
    rdd.foreachPartition {
      partition => {
        partition.toSeq
          .groupBy { case (jobId, document) => jobId }
          .foreach {
            case (jobId, documents) => exporter.writeToHdfs(jobId, documents)
          }
      }
    }
    ..
  }
}
```

Grouping happens in **User Memory** region

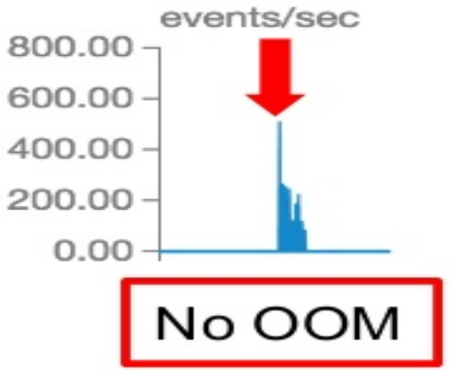
When a large batch of documents is processed:

Spark does does not spill to disk

OOM errors

→ configure a **small streaming batch interval** or set **maxRatePerPartition** to limit the events of the batch

Write documents for each job to a separate HDFS directory.
Second version: Let Spark do the grouping and write to HDFS



Spark Execution Memory Region

```
jobIdDocumentDStream.foreachRDD {
  rdd => {
    rdd.toDF("jobId", "document")
      .write.mode(SaveMode.Append)
      .partitionBy("jobId")
      .format("text")
      .save(pathToHdfsDirectory)
    ..
  }
}
```

Group and write job documents through **DataFrame APIs**

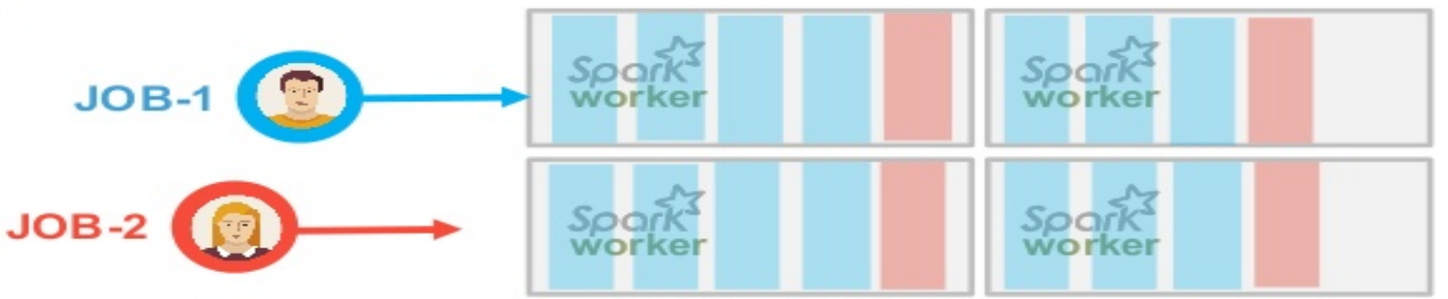
- Uses the **Execution Memory region** for computations
- Spark can prevent OOM errors by **borrowing space from the Storage memory region** or by **spilling to disk** if needed.

Within-Application Job Scheduling

FIFO SCHEDULER

First job gets available resources while its stages have tasks to launch

Big jobs “block” smaller jobs started later



FAIR SCHEDULER

Tasks between jobs assigned “round robin”

Smaller jobs submitted while a big job is running can start receiving resources right away

SUMMARY

Watson Analytics for Social Media allows harvesting and analyzing social media data

Becoming a **real-time multi-tenant** analytics pipeline required:

- Splitting analytics into tenant-specific and language-specific
- Aggregating all tenants trickle-feeds into a single stream
- Ensuring low-latency context switching between tenants
- Removing state from processing components

New pipeline based on **Spark**, **Kafka** and **Zookeeper**

- robust
- fulfills latency and throughput requirements
- additional analytics

THANK YOU.

RUBEN PULIDO

www.linkedin.com/in/ruben-pulido
@_rubenpulido

BEHAR VELIQI

www.linkedin.com/in/beharveliqi
@beveliqi

FREE TRIAL

www.watsonanalytics.com