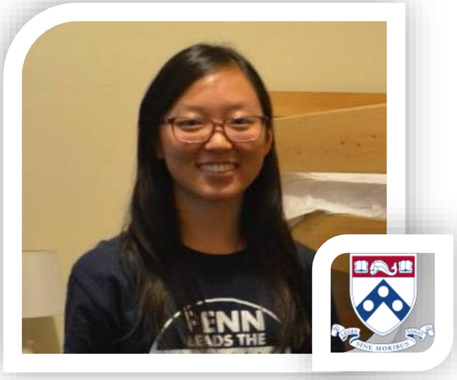
DataSpark

clarity redefined

SparkGen: Spark Job Configurator

Data Engineering Team

The Team



Karen Her
University of Pennsylvania
Computer Science

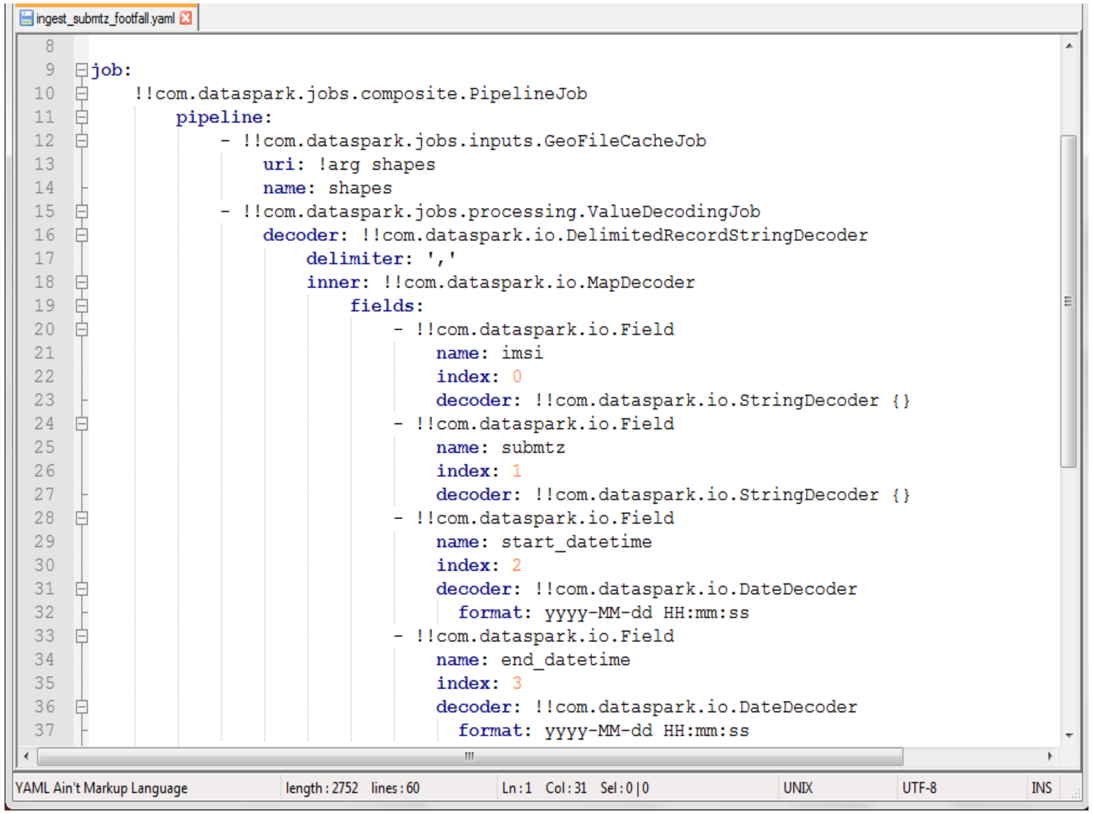
Skyler Sin
Stanford University
Symbolic Systems

Karthik Balasubramanian
Carnegie Mellon University
Information Systems Management

DataSpark

Executive Summary

- SparkGen is a Spark job configurator



ingest_submtz_footfall.yaml

```yaml

job:
    !!com.dataspark.jobs.composite.PipelineJob
        pipeline:
            - !!com.dataspark.jobs.inputs.GeoFileCacheJob
                uri: !arg shapes
                name: shapes
            - !!com.dataspark.jobs.processing.ValueDecodingJob
                decoder: !!com.dataspark.io.DelimitedRecordStringDecoder
                    delimiter: ','
                    inner: !!com.dataspark.io.MapDecoder
                        fields:
                            - !!com.dataspark.io.Field
                                name: imsi
                                index: 0
                                decoder: !!com.dataspark.io.StringDecoder {}
                            - !!com.dataspark.io.Field
                                name: submtz
                                index: 1
                                decoder: !!com.dataspark.io.StringDecoder {}
                            - !!com.dataspark.io.Field
                                name: start_datetime
                                index: 2
                                decoder: !!com.dataspark.io.DateDecoder
                                  format: yyyy-MM-dd HH:mm:ss
                            - !!com.dataspark.io.Field
                                name: end_datetime
                                index: 3
                                decoder: !!com.dataspark.io.DateDecoder
                                  format: yyyy-MM-dd HH:mm:ss
```

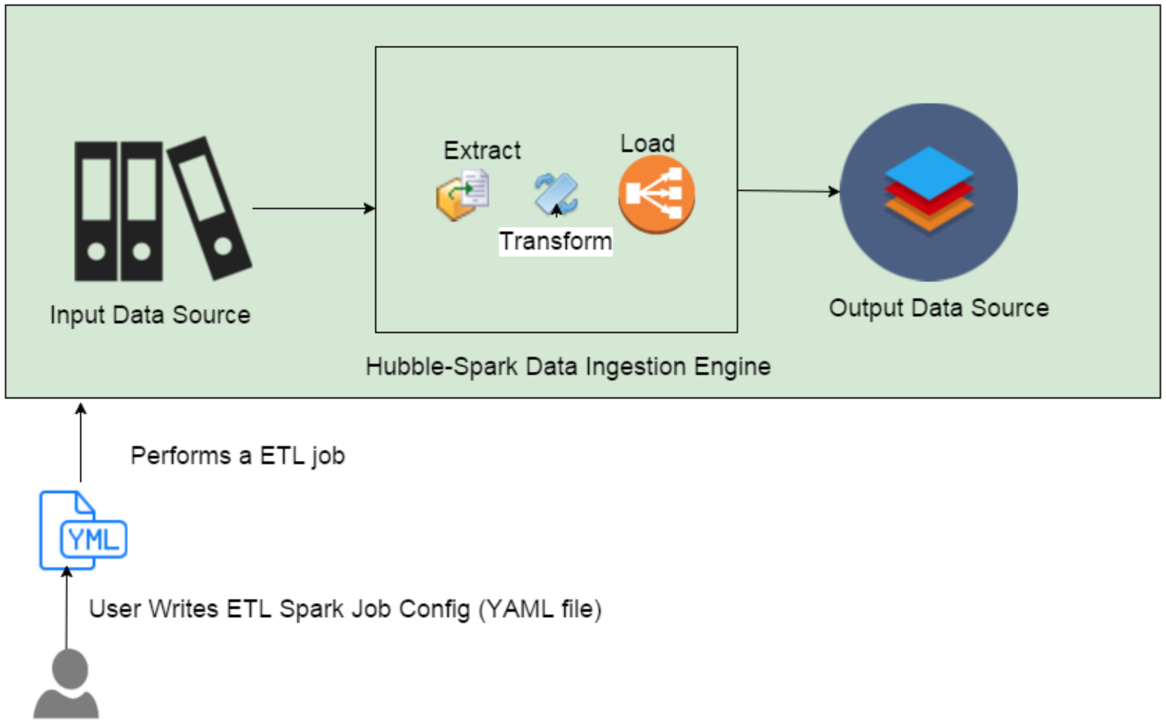
YAML Ain't Markup Language   length : 2752   lines : 60   Ln : 1   Col : 31   Sel : 0 | 0   UNIX   UTF-8   INS

Job Configuration File

DataSpark

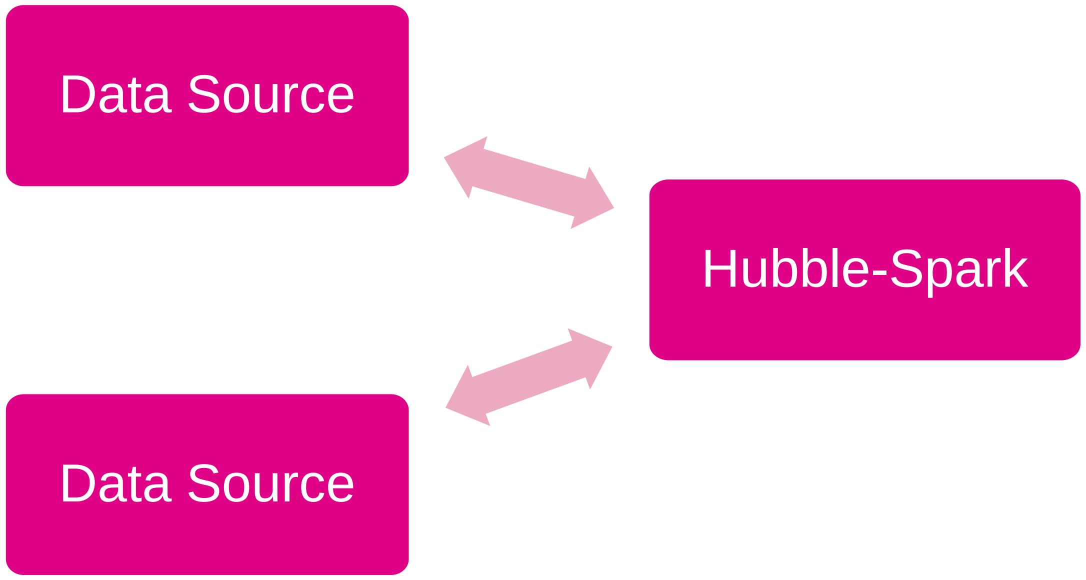
Executive Summary

- SparkGen is a Spark job configurator
- Hubble-Spark – A Data Ingestion Engine in DataSpark environment. It is written in Java.


Extract
Load
Transform
Input Data Source
Output Data Source
Hubble-Spark Data Ingestion Engine
Performs a ETL job
YML
User Writes ETL Spark Job Config (YAML file)

Executive Summary

- SparkGen is a Spark job configurator
- Hubble-Spark – A Data Ingestion Engine in DataSpark environment. It is written in Java.
- Has connectors to multiple varieties of data sources to read, write and perform Extract, Transform and Load (ETL) jobs.

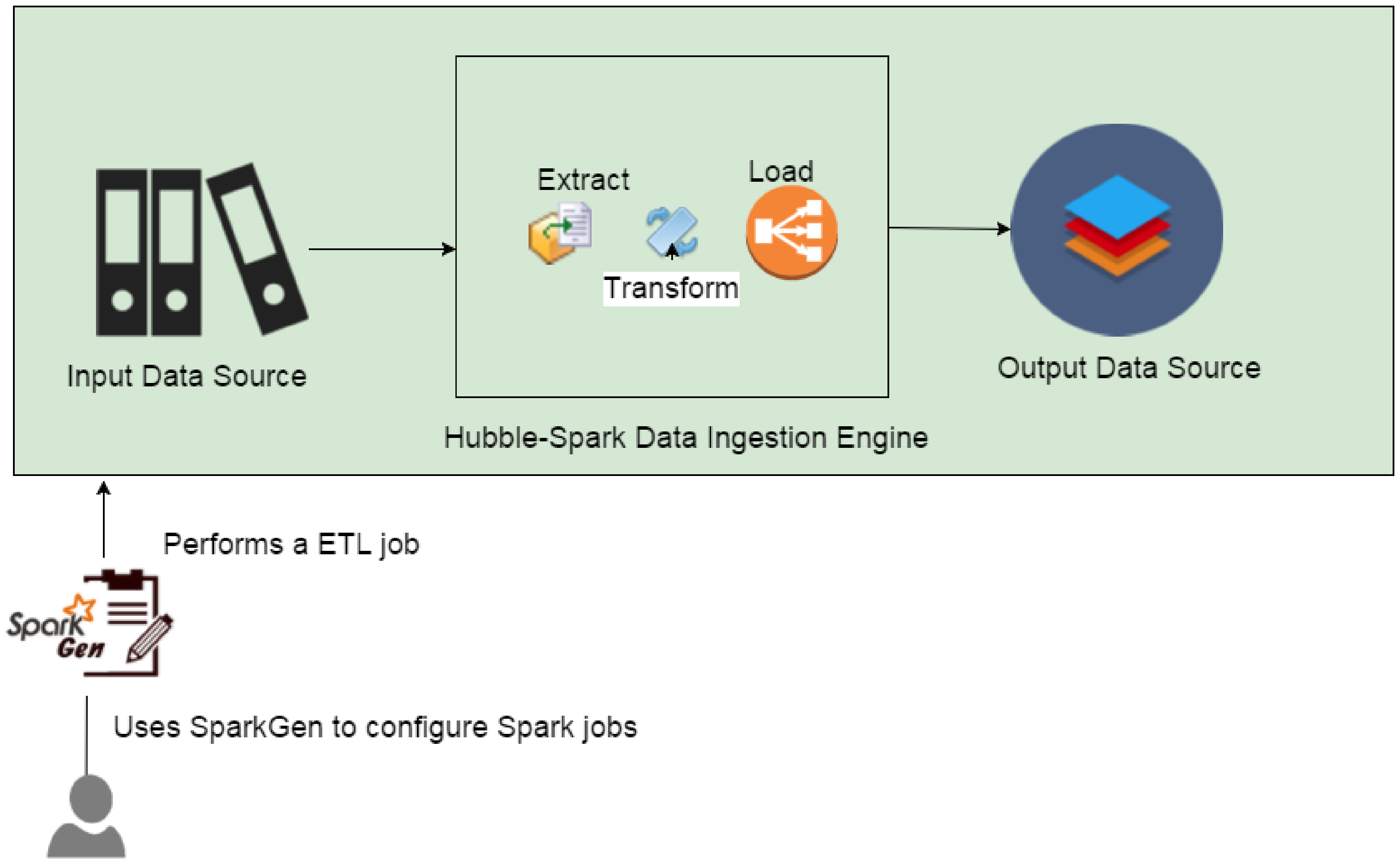


DataSpark

Executive Summary

- SparkGen is a Spark job configurator
- Hubble-Spark – A Data Ingestion Engine in DataSpark environment. It is written in Java.
- Has connectors to multiple varieties of data sources to read, write and perform Extract, Transform and Load (ETL) jobs.
- Uses markup language called YAML to configure ETL Spark Jobs
- We developed a SparkGen API component to expose internal components of Hubble-Spark ETL engine and a UI component to generate a Spark Job Configuration file (YAML file)

Executive Summary



DataSpark

Problem Statement

How do we simplify the robust configuration driven Hubble-Spark to a team member with limited expertise in Spark and achieve seamless ETL activities?

DataSpark

Problem Background

- Hubble Spark is an execution framework for any ETL Spark Job
- Has hundreds of components to extract, transform, and load terabytes of data which facilitate DataSpark’s business
- Ideology: One code base – Million jobs
- All you have to do is write your ETL configurations using the existing templates

Problem Background

However,

- Hubble-Spark requires deep understanding of Java-Spark to configure Spark ETL Jobs
- YAML file used to configure complex Spark Jobs is a new configuration markup, is created manually and error prone

Our Idea

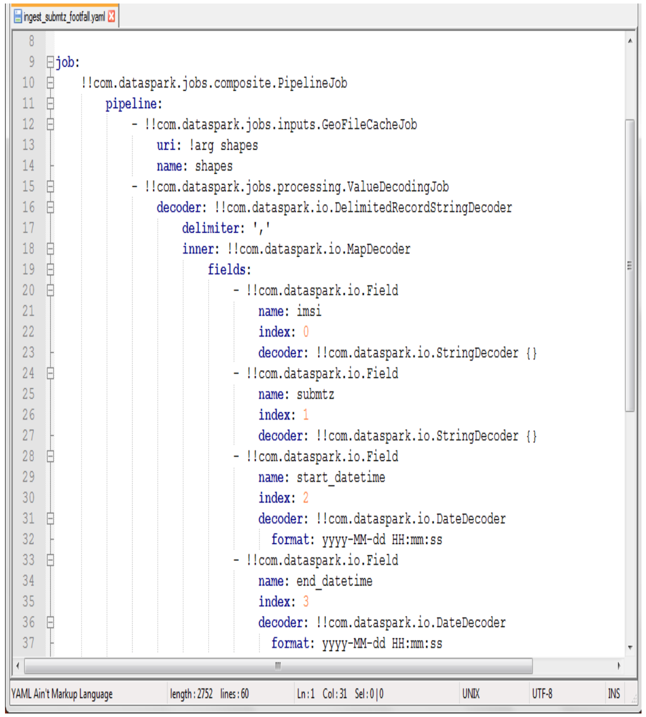
What if?

- *We provide an abstraction to Hubble-Spark as a Graphical User Interface and give users the limited information they need to know to run any ETL Jobs using Hubble-Spark*

DataSpark

Goal

- Create a user-friendly interface that will produce a YAML file to run Spark Jobs.

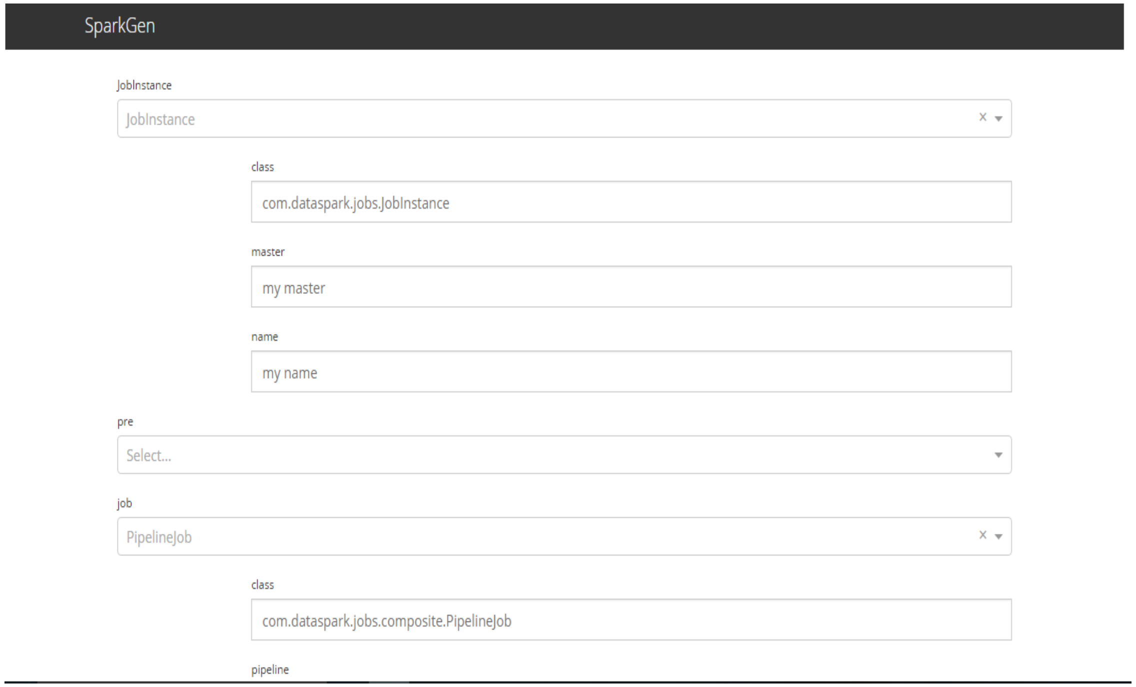


ingest_submtz_footfall.yaml

```yaml

job:
    !!com.dataspark.jobs.composite.PipelineJob
        pipeline:
            - !!com.dataspark.jobs.inputs.GeoFileCacheJob
                uri: !arg shapes
                name: shapes
            - !!com.dataspark.jobs.processing.ValueDecodingJob
                decoder: !!com.dataspark.io.DelimitedRecordStringDecoder
                    delimiter: ','
                    inner: !!com.dataspark.io.MapDecoder
                        fields:
                            - !!com.dataspark.io.Field
                                name: imsi
                                index: 0
                                decoder: !!com.dataspark.io.StringDecoder {}
                            - !!com.dataspark.io.Field
                                name: submtz
                                index: 1
                                decoder: !!com.dataspark.io.StringDecoder {}
                            - !!com.dataspark.io.Field
                                name: start_datetime
                                index: 2
                                decoder: !!com.dataspark.io.DateDecoder
                                  format: yyyy-MM-dd HH:mm:ss
                            - !!com.dataspark.io.Field
                                name: end_datetime
                                index: 3
                                decoder: !!com.dataspark.io.DateDecoder
                                  format: yyyy-MM-dd HH:mm:ss
```

YAML Ain't Markup Language | length : 2752 lines : 60 | Ln : 1 Col : 31 Sel : 0 | 0 | UNIX | UTF-8 | INS



SparkGen

JobInstance: JobInstance

class: com.dataspark.jobs.JobInstance

master: my master

name: my name

pre: Select...

job: PipelineJob

class: com.dataspark.jobs.composite.PipelineJob

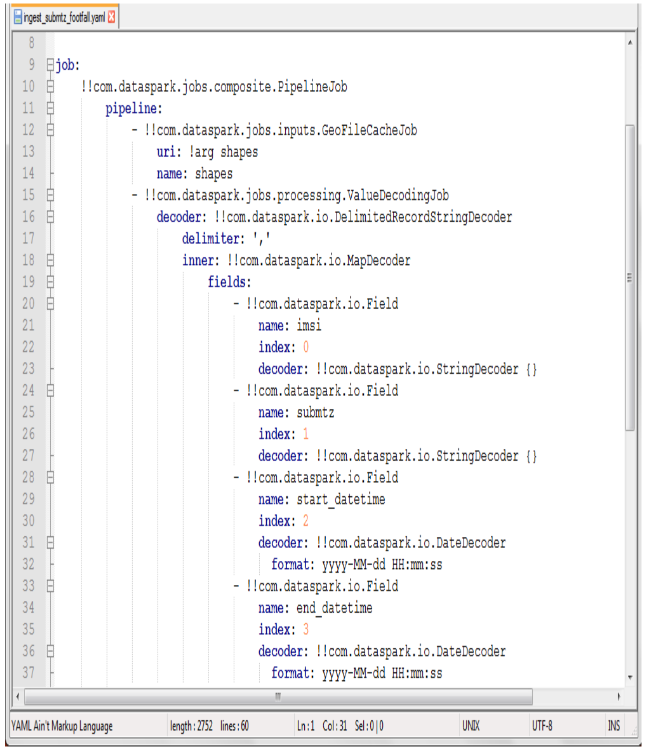
pipeline

Internal Components of Hubble-Spark

DataSpark

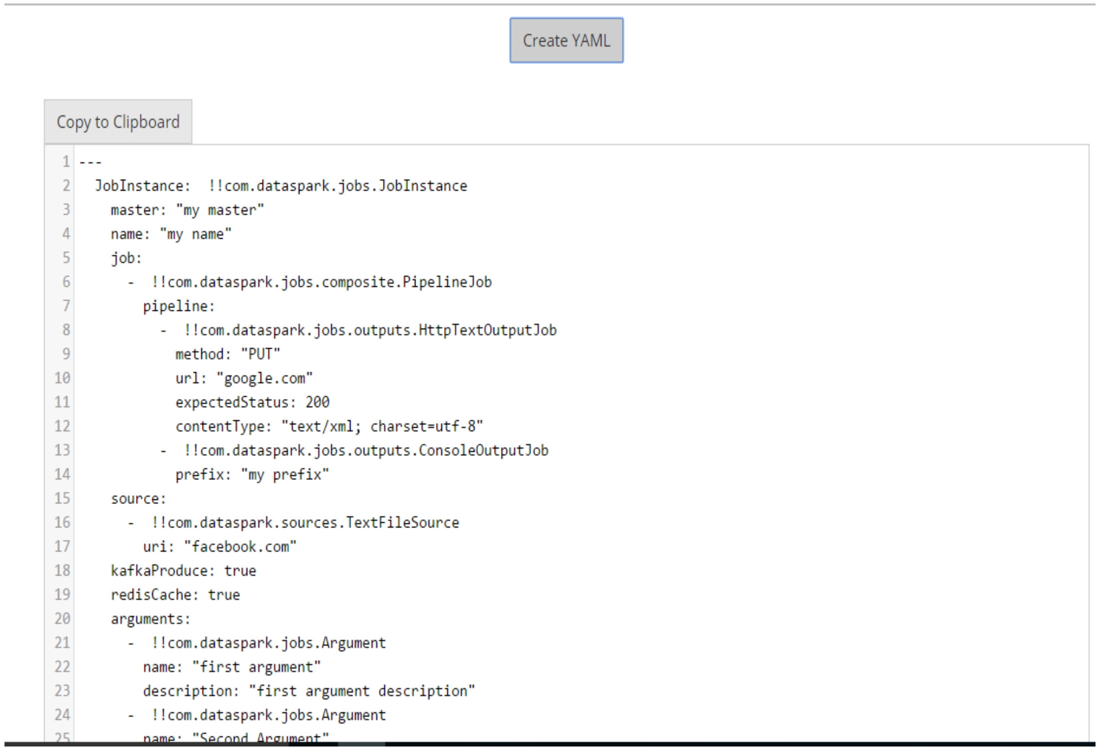
Goal

- Create a user-friendly interface that will produce a YAML file to run Spark Jobs.


ingest_submtz_footfall.yaml
YAML Ain't Markup Language
length : 2752 lines : 60
Ln : 1 Col : 31 Sel : 0 | 0
UNIX
UTF-8
INS

```yaml

job:
    !!com.dataspark.jobs.composite.PipelineJob
        pipeline:
            - !!com.dataspark.jobs.inputs.GeoFileCacheJob
                uri: !arg shapes
                name: shapes
            - !!com.dataspark.jobs.processing.ValueDecodingJob
                decoder: !!com.dataspark.io.DelimitedRecordStringDecoder
                    delimiter: ','
                    inner: !!com.dataspark.io.MapDecoder
                        fields:
                            - !!com.dataspark.io.Field
                                name: imsi
                                index: 0
                                decoder: !!com.dataspark.io.StringDecoder {}
                            - !!com.dataspark.io.Field
                                name: submtz
                                index: 1
                                decoder: !!com.dataspark.io.StringDecoder {}
                            - !!com.dataspark.io.Field
                                name: start_datetime
                                index: 2
                                decoder: !!com.dataspark.io.DateDecoder
                                    format: yyyy-MM-dd HH:mm:ss
                            - !!com.dataspark.io.Field
                                name: end_datetime
                                index: 3
                                decoder: !!com.dataspark.io.DateDecoder
                                    format: yyyy-MM-dd HH:mm:ss
```



```yaml
---
 JobInstance:  !!com.dataspark.jobs.JobInstance
   master: "my master"
   name: "my name"
   job:
     - !!com.dataspark.jobs.composite.PipelineJob
       pipeline:
         - !!com.dataspark.jobs.outputs.HttpTextOutputJob
           method: "PUT"
           url: "google.com"
           expectedStatus: 200
           contentType: "text/xml; charset=utf-8"
         - !!com.dataspark.jobs.outputs.ConsoleOutputJob
           prefix: "my prefix"
   source:
     - !!com.dataspark.sources.TextFileSource
       uri: "facebook.com"
   kafkaProduce: true
   redisCache: true
   arguments:
     - !!com.dataspark.jobs.Argument
       name: "first argument"
       description: "first argument description"
     - !!com.dataspark.jobs.Argument
       name: "Second Argument"
```

Produce YAML File

DataSpark

Solution Description

- Three components working together
  - Spark Jar File
    - An execution framework for all ETL Spark Jobs
  - SparkGen API
    - Exposes Hubble-Spark internal components to user interface
  - SparkGen UI
    - Creates ETL job configuration file (YAML)

Solution Steps

Spark Jar File

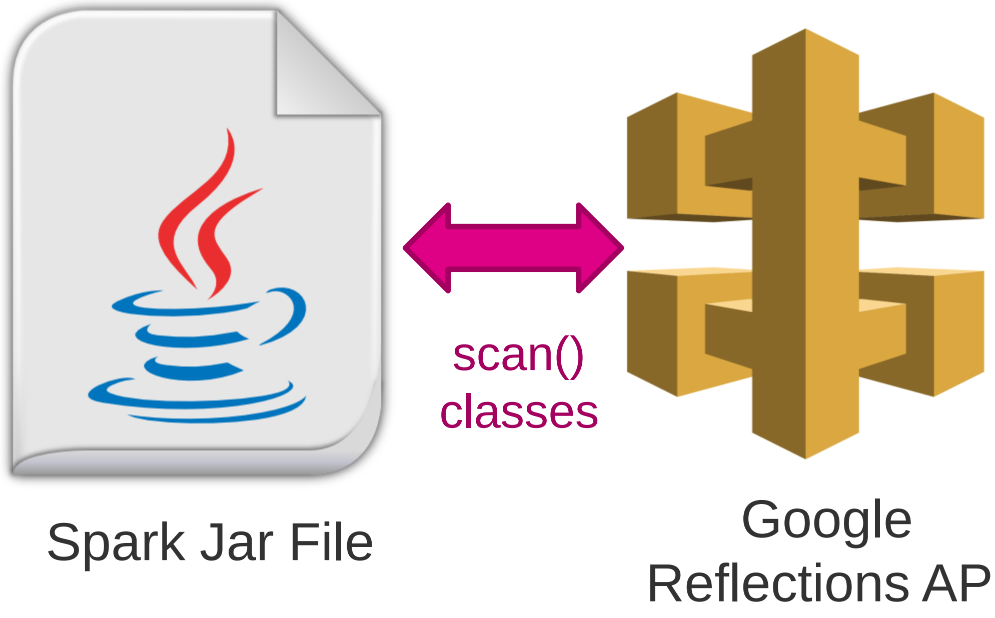
DataSpark

Solution Steps

Spark Jar File

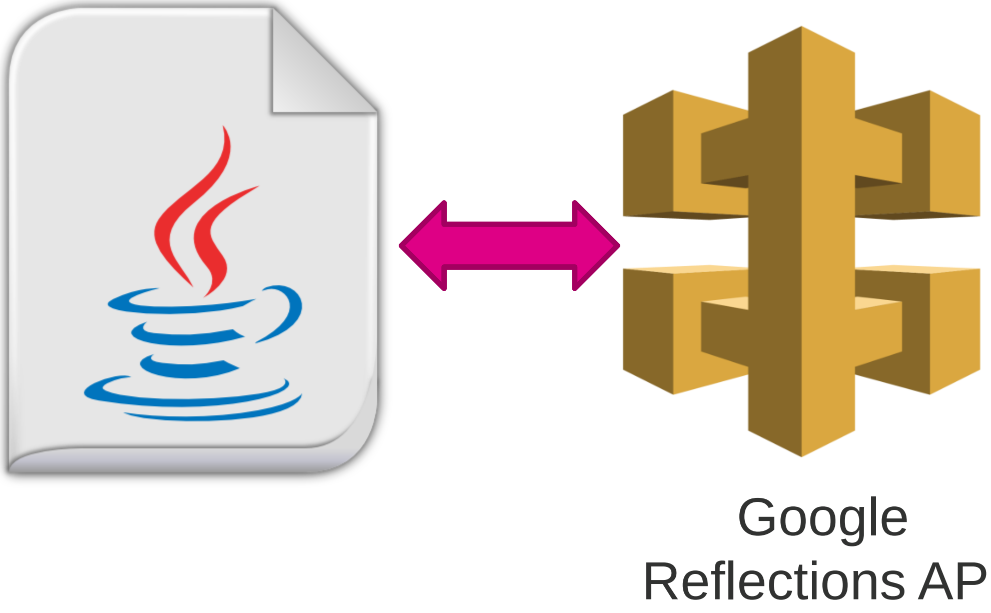
DataSpark

Solution Steps



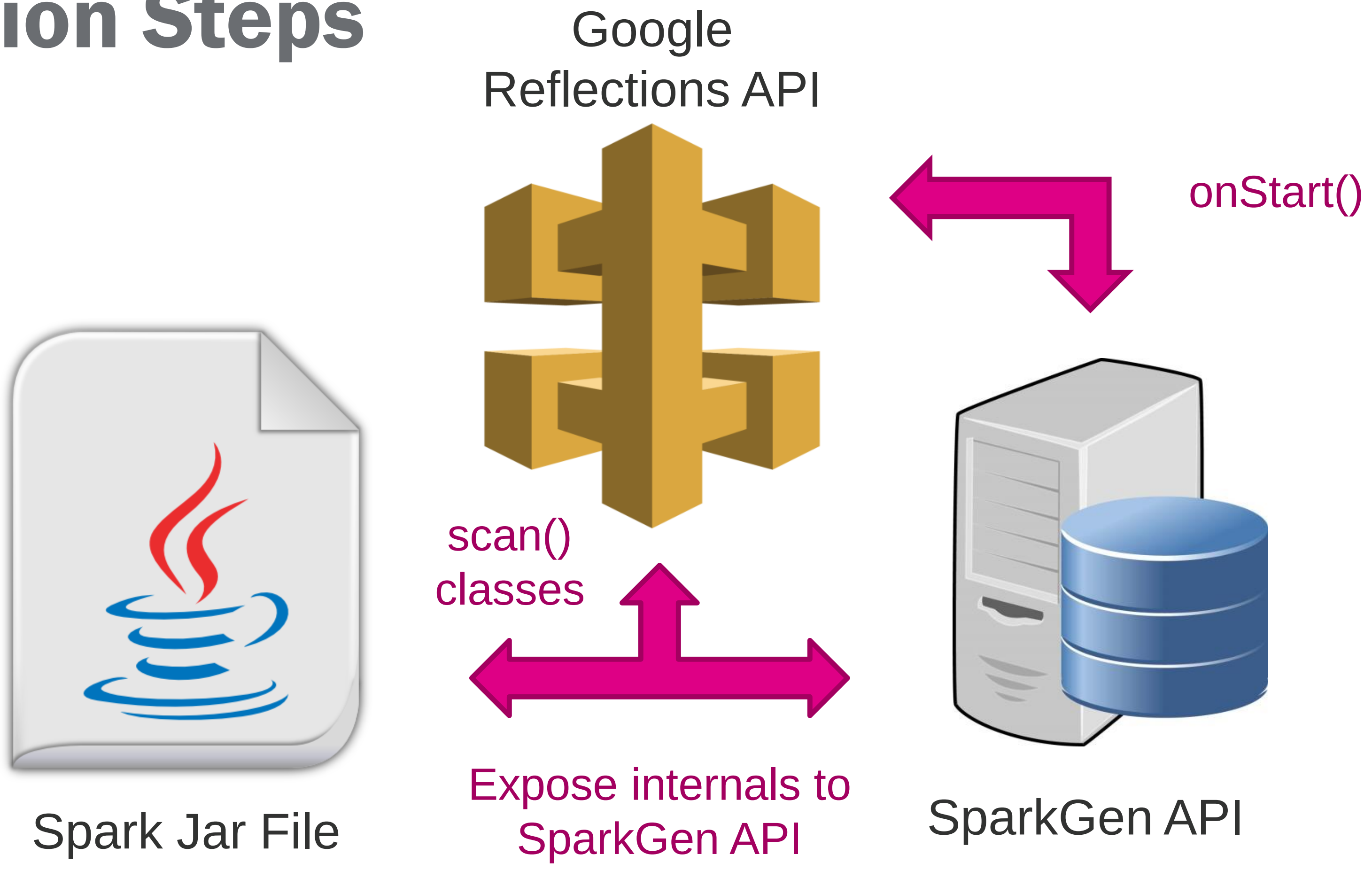
DataSpark

Solution Steps

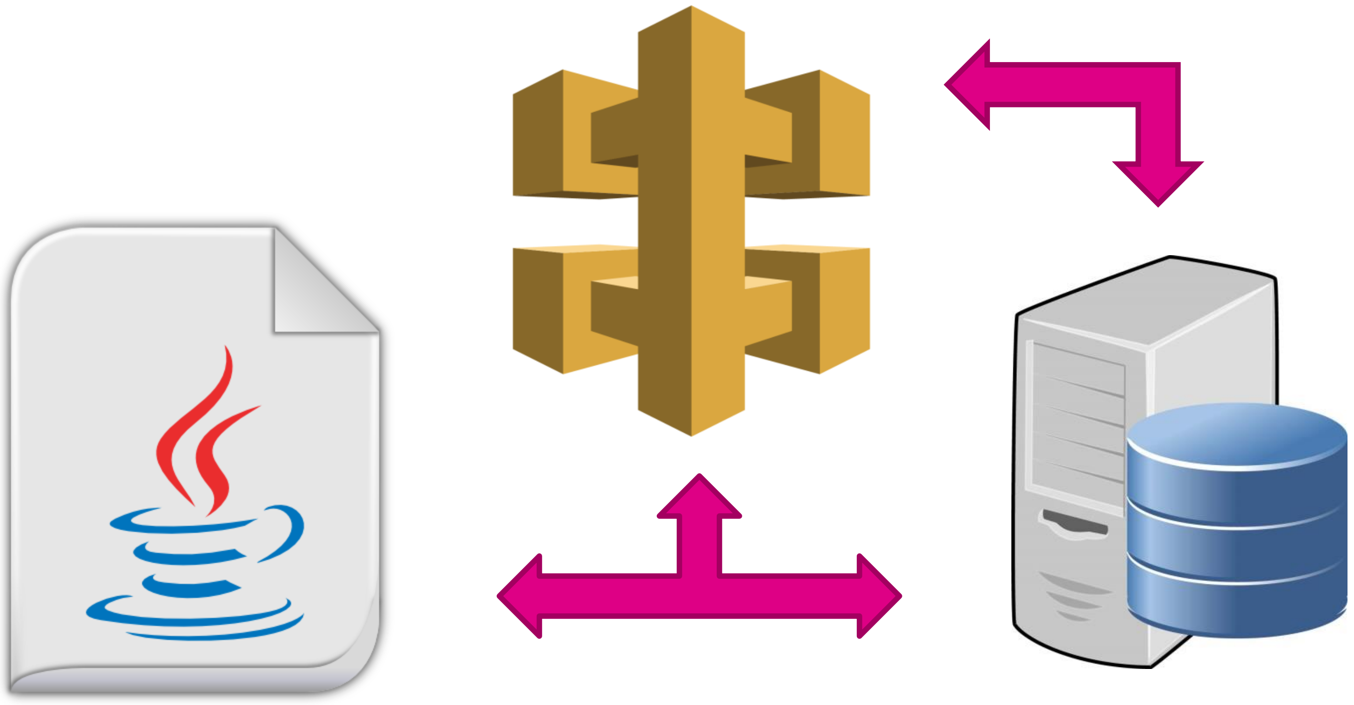


DataSpark

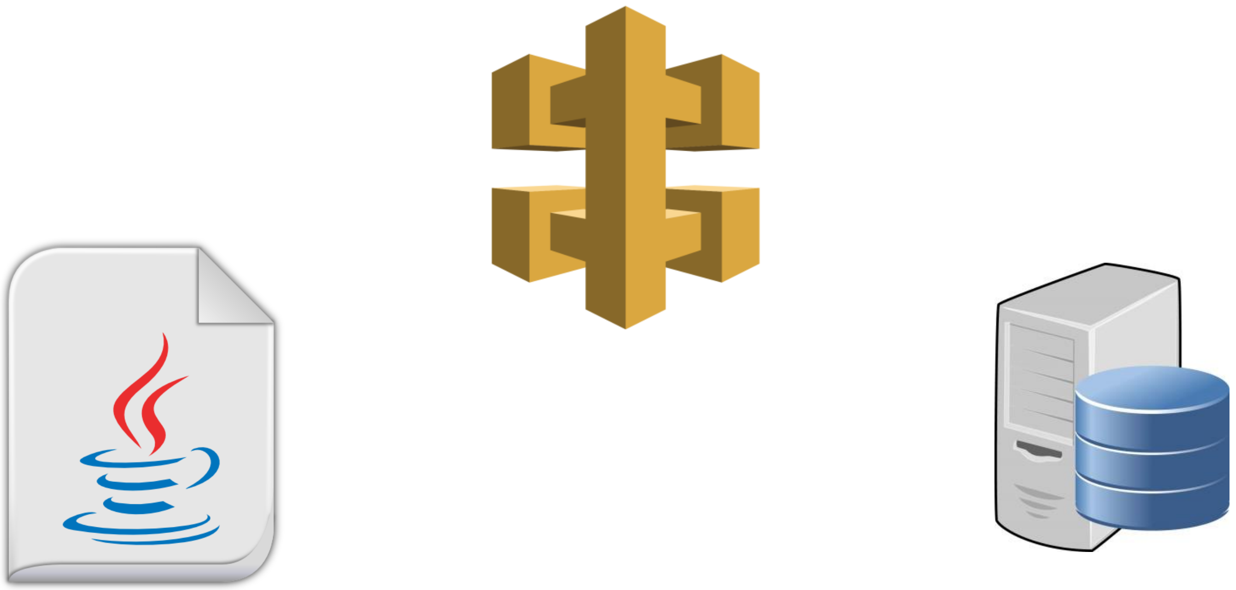
Solution Steps



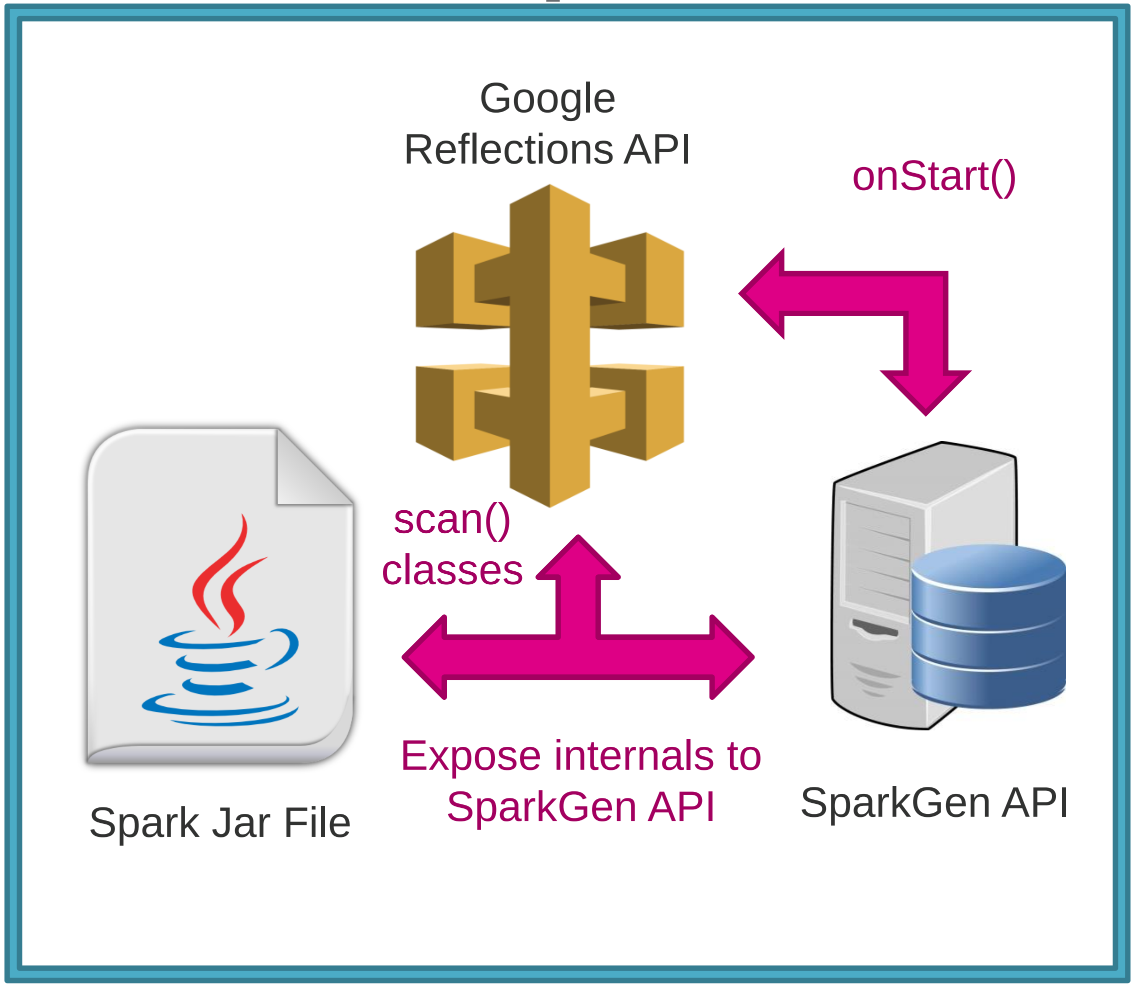
Solution Steps



Solution Steps

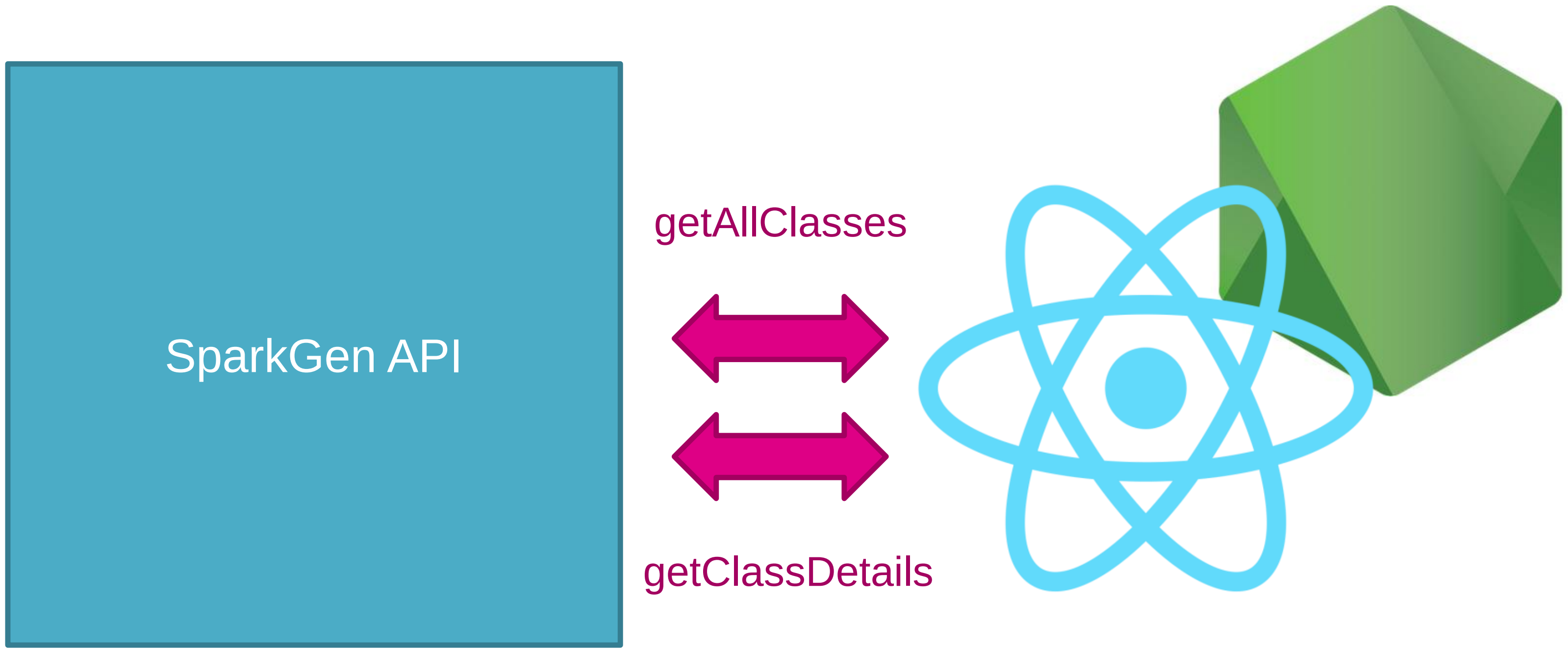


Solution Steps



DataSpark

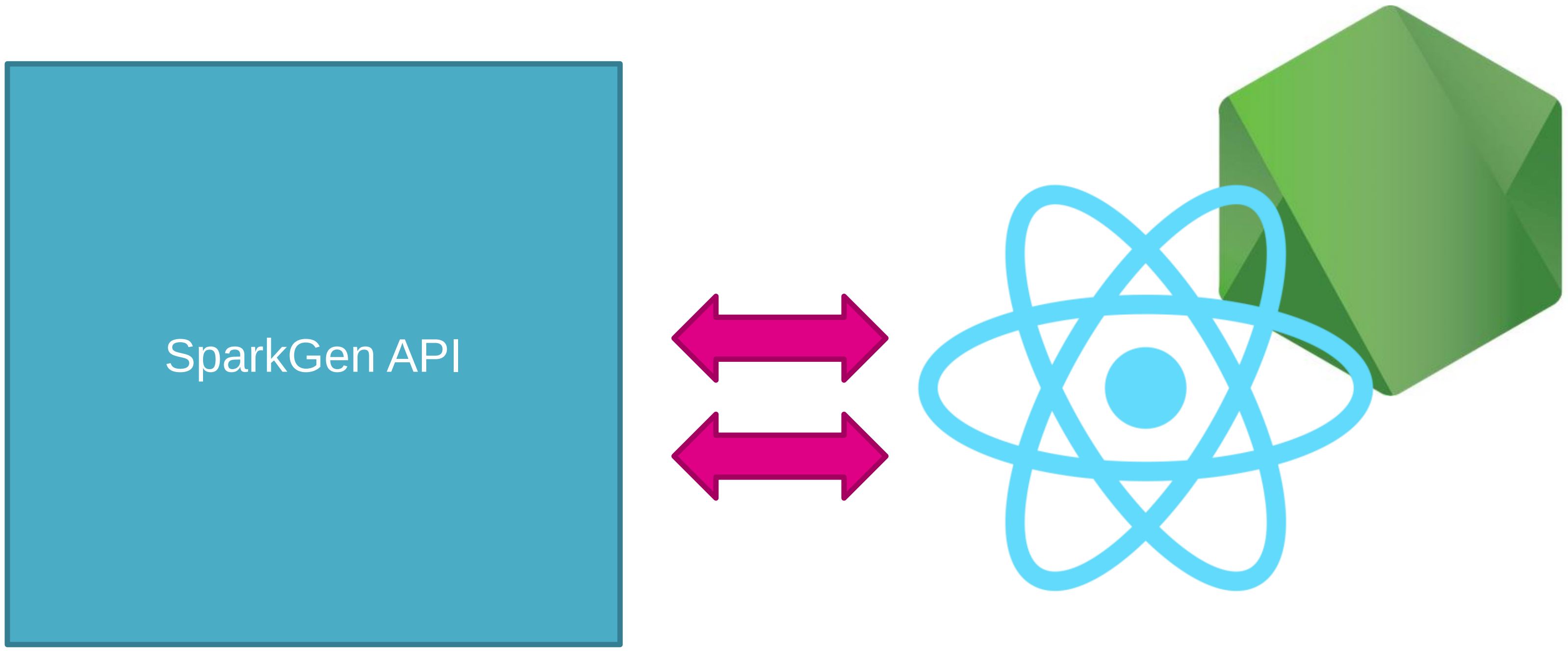
Solution Steps



React Form with
Node Server

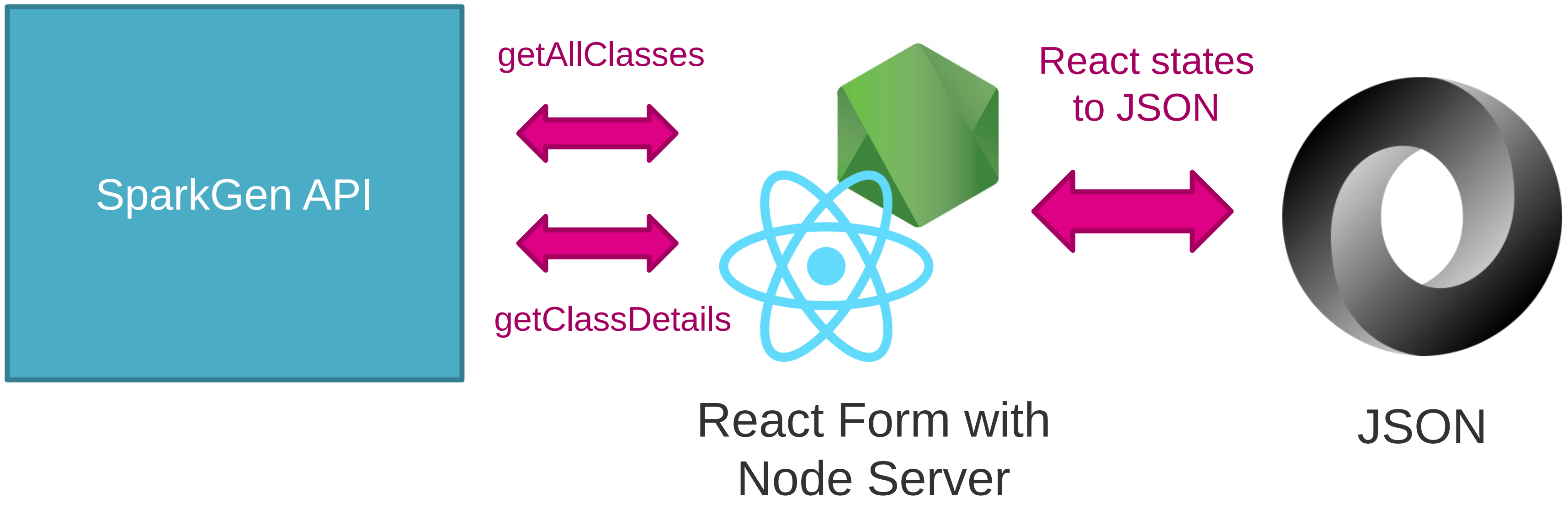
DataSpark

Solution Steps

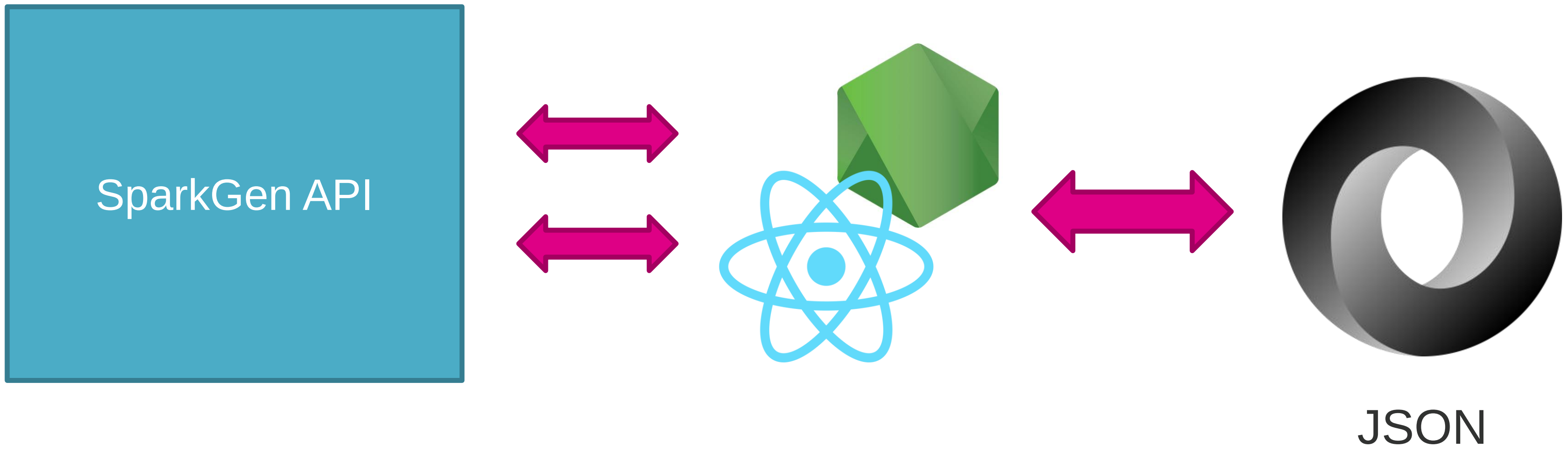


React Form with
Node Server

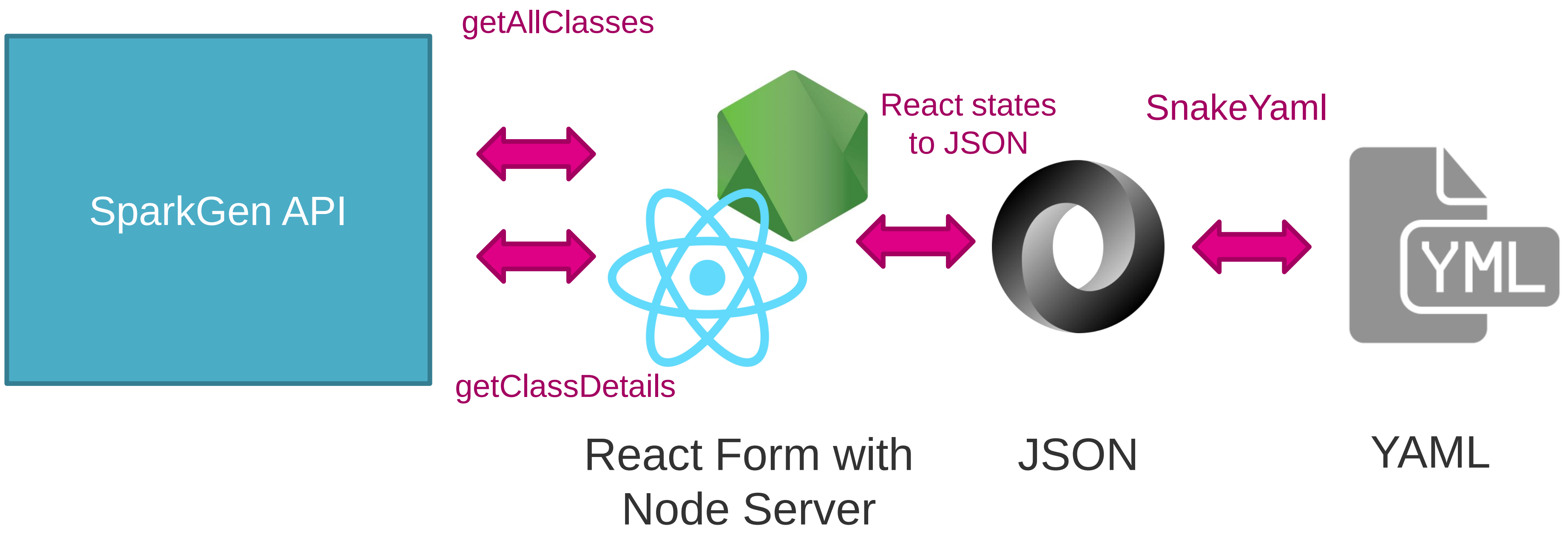
Solution Steps



Solution Steps

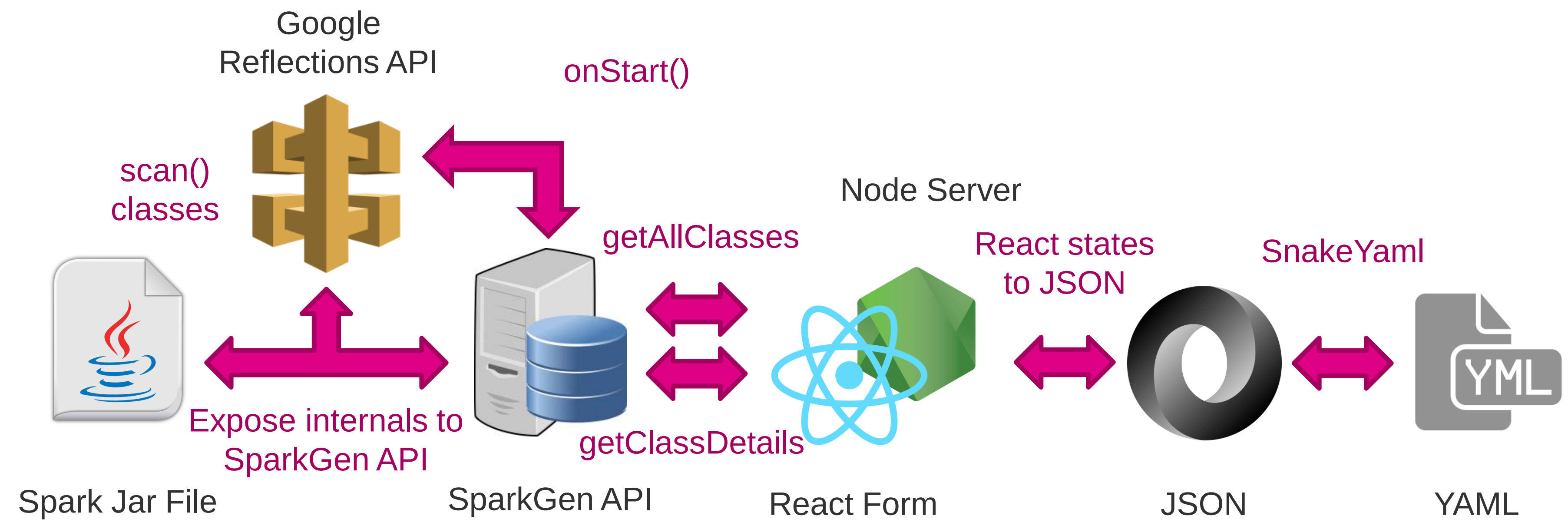


Solution Steps



Solution Overview

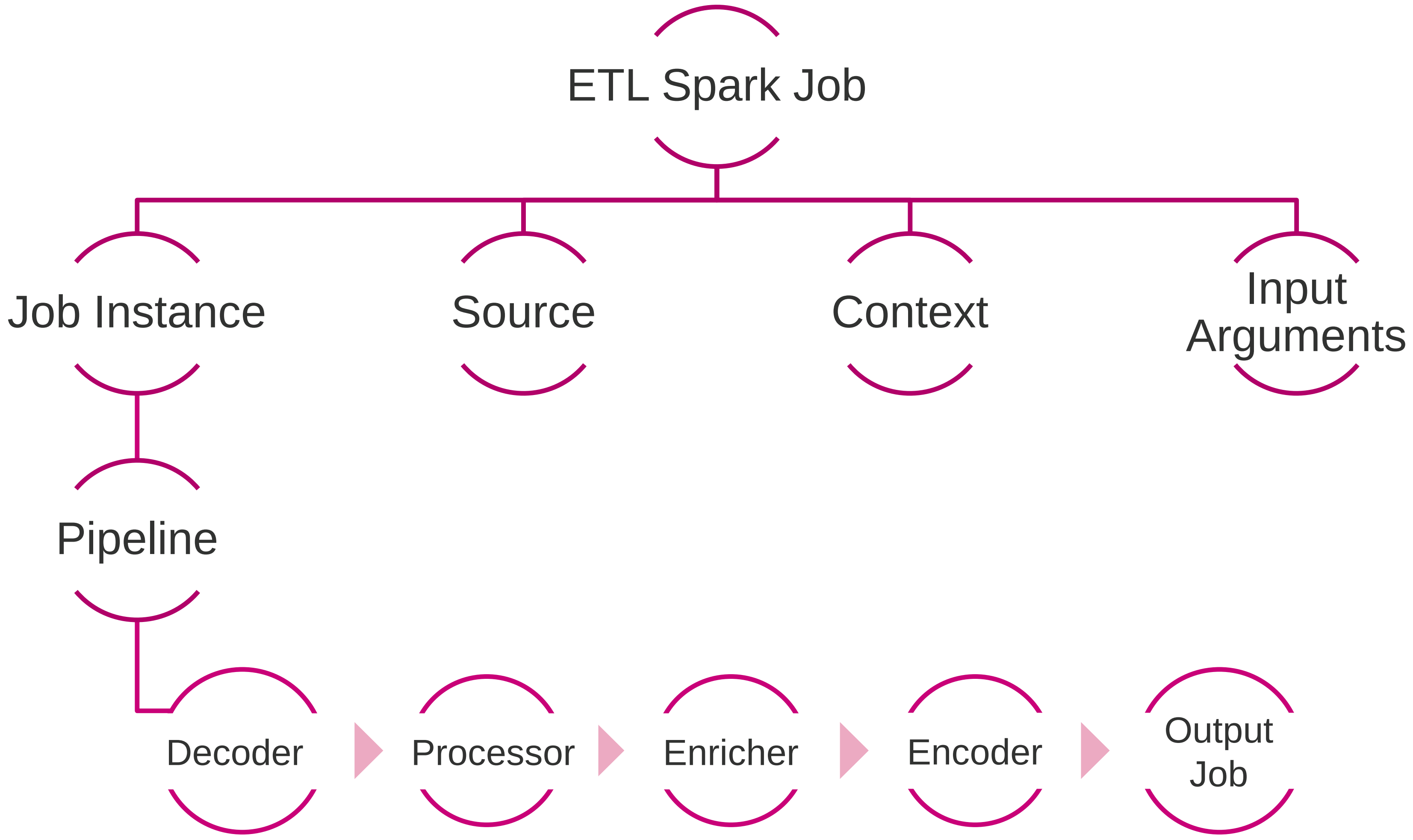
Flow Chart



SparkGen

- SparkGen is a dynamic form
  - Grows based on the user’s inputs
  - User’s inputs are recorded in the browser
  - On submit, the form is created and shows a YAML file in the CodeMirror palette of the UI
  - Users can copy to clipboard or edit the YAML directly in the palette

Components of a YAML File



Next Steps

- Refining YAML file
- Work with Product Team to validate YAML generation process

DataSpark

DEMO

DataSpark

DataSpark

Thank you