ODSC East • Boston • 2018

May 1st – 4th • Boston Convention Center

Under-the-hood: Creating Your Own Spark Data Sources

CONVERSANT | Engineering

About...

CONVERSANT

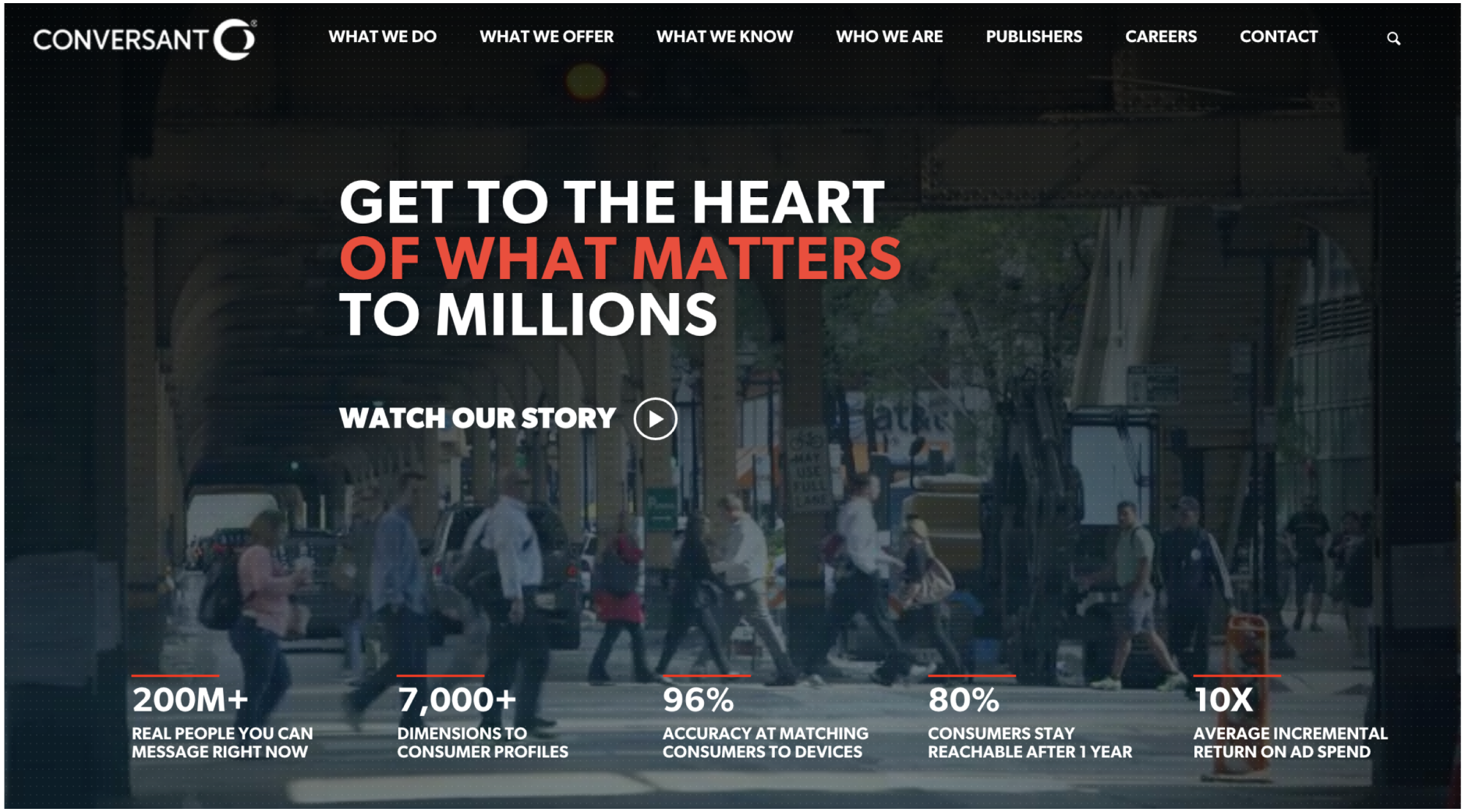
- Part of the Epsilon organization under Alliance Data
- Player in the programmatic ad bidding market (AdTech)
- Personalized messaging using non-PII anonymous profiles

  (PII = Personally Identifiable Information)
- EU General Data Protection Regulation (GDPR) ready

Myself (Jayesh Thakrar)

- Sr. Software Engineer (jthakrar@conversantmedia.com)
-  https://www.linkedin.com/in/jayeshthakrar/

More About CONVERSANT...

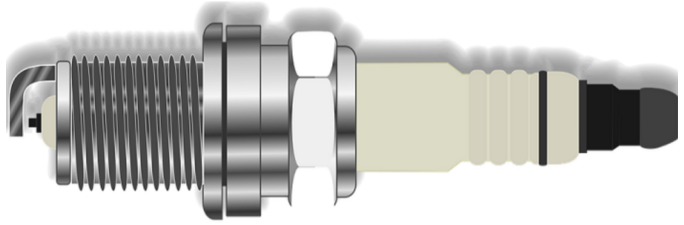


Workshop Agenda

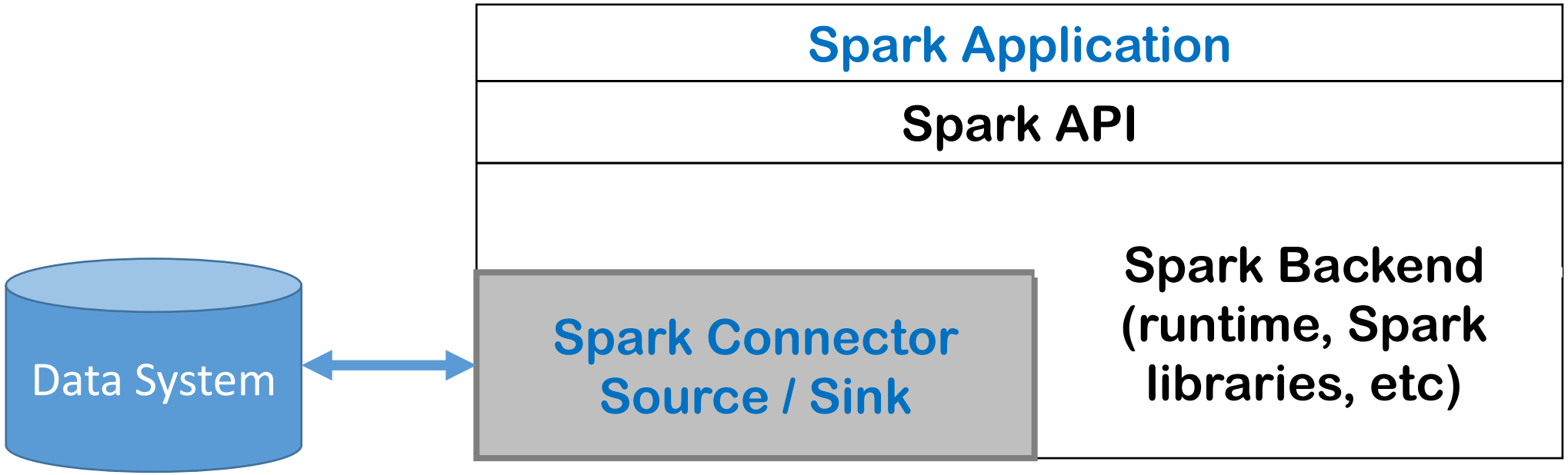
1. Introduction to Spark data sources
2. Walk-through sample code
3. Practical considerations

Slides and code on Github
https://github.com/JThakrar/sparkconn

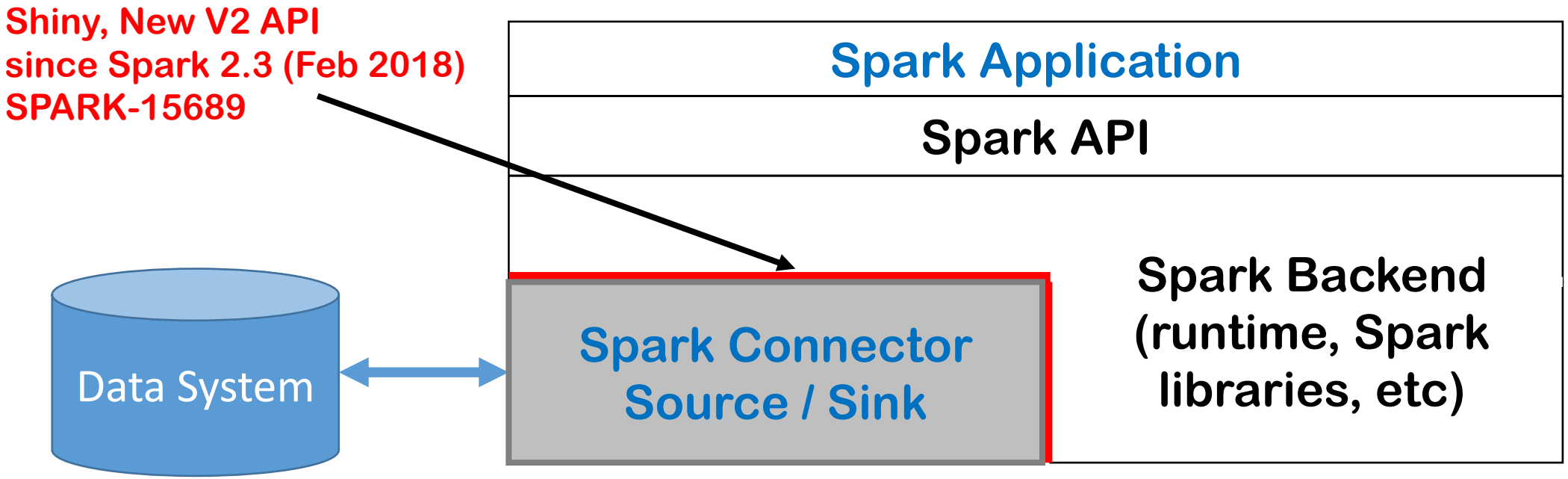
Introduction To Spark Data Source



Spark Data Source



Data Source V2 API



V2 API: Data Source Types

DataSource Type	Output
Batch	Dataset
Microbatch (successor to Dstreams)	Structured Stream = stream of bounded dataset
Continuous	Continuous Stream = continuous stream of Row(s)

V2 API

- **API = Java Interfaces**
- **Similar interfaces across all 3 data source types**
- **Write-ahead, recovery, checkpointing not ready**
- **Well-documented but still evolving**
  SPARK-20928, SPARK-22386

Reading From Data Source

```
val data = spark.read.format("DataSource").option("key", "value").schema(…).load()

data.show(100, false)
```

Step	Action
read	Instantiates a DataFrameReader for orchestration
format	Lazy lookup of data source class
schema	Optional, user-provided schema
load	Initiates actual setup of data source and instantiates the "connector"
show	Triggers actual data retrieval (“show” is an action)

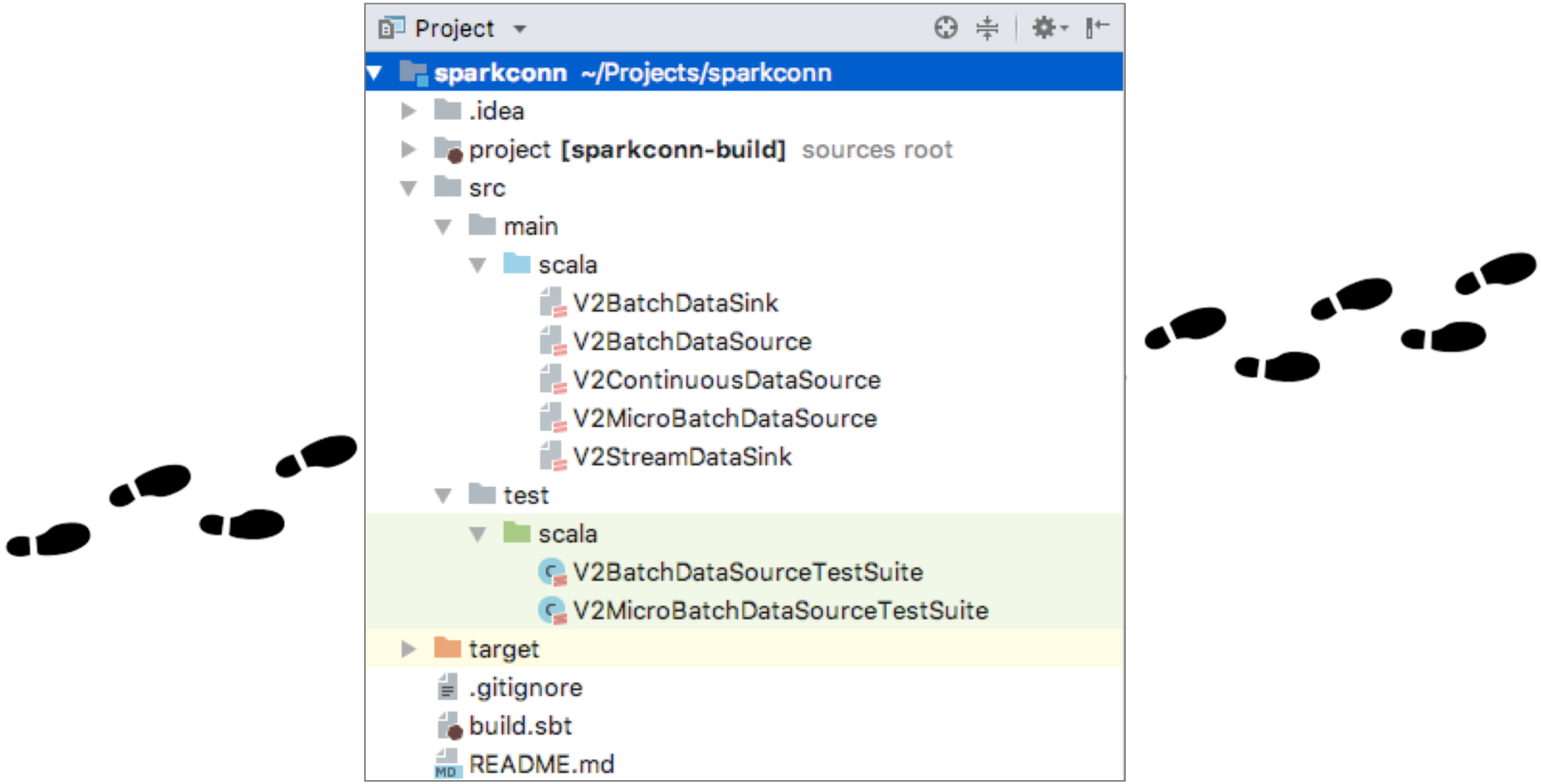
Reading: Interfaces to Implement

Java Interface	Purpose
org.apache.spark.sql.sources.v2.**DataSourceRegister** org.apache.spark.sql.sources.v2.**ReadSupport**	**DataSourceRegister** is the entry point for your connector. **ReadSupport** instantiates **DataSourceReader** below. Accepts options/parameters and optional schema from Spark application.
org.apache.spark.sql.sources.v2.reader.**DataSourceReader**	**DataSourceReader** needs to: • Provide the data schema • determine the number of partitions and creating that many reader factories below.
org.apache.spark.sql.sources.v2.reader.**DataReaderFactory**	This is the "handle" passed by the driver to each executor. It instantiates the readers below and controls data fetch. The **DataSourceRegister**, **DataSourceReader** and **DataReaderFactory** are instantiated at the driver. The driver then serializes **DataReaderFactory** and sends it to each of the executors.
org.apache.spark.sql.sources.v2.reader.**DataReader**	This does the actual work of fetching data

Writing: Interfaces to Implement

Java Interface	Purpose
org.apache.spark.sql.sources.v2.**DataSourceRegister** org.apache.spark.sql.sources.v2.**WriteSupport**	**DataSourceRegister** is the entry point for your connector. **WriteSupport** instantiates **DataSourceWriter**. Accepts options, savemode and schema.
org.apache.spark.sql.sources.v2.writer.**DataSourceWriter**	This interface needs to: • register commits and aborts across all executors • determine the number of partitions and creating that many reader factories below.
org.apache.spark.sql.sources.v2.writer.**DataWriterFactory**	This is the "handle" passed by the driver to each executor. It instantiates the writers below that control the write operations. The **DataSourceRegister**, **DataSourceWriter** and **DataWriterFactory** are instantiated at the driver. The driver then serializes **DataWriterFactory** and sends it to each of the executors.
org.apache.spark.sql.sources.v2.writer.**DataWriter**	Does writing data as well as commits and aborts
org.apache.spark.sql.sources.v2.writer.**WriterCommitMessage**	**Serializable empty interface** to pass commit message from DataWriter to DataSourceWriter

Code Walkthrough



Details

- **Project: https://github.com/JThakrar/sparkconn**
- **Requirements:**
  - **Spark 2.3 (http://spark.apache.org/downloads.html)**
  - **Scala 2.11 (https://www.scala-lang.org/download/2.11.8.html)**
  - **SBT (https://www.scala-sbt.org/download.html)**
  - **Editor or IDE (Intellij - https://www.jetbrains.com/idea/download/)**

Practical Considerations

Know Your Data Source

- Configuration
- Partitions
- Data schema
- Parallelism approach
- Batch and/or streaming
- Restart / recovery

V2 API IS STILL EVOLVING

- **SPARK-20928**
  SPIP: Continuous Processing Mode for Structured Streaming

- **Why use V2?**
  Alternative to V2 needs significantly more time and effort!
  See https://www.youtube.com/watch?v=O9kpduk5D48

- **Checkpointing, restartability, writeahead, etc**

- **Resource - dev@spark.apache.org**

Questions?