Spark in the Dark 101

Code4Lib 2018, Washington D.C.
Tuesday, February 13th, 9AM - Noon
Omni Shoreham Hotel Senate Room

github.com/spark4lib/code4lib2018

Workshop Resources:

github.com/spark4lib/code4lib2018

Link to Slides, Datasets, & Other Workshop Materials

Schedule

9:00-9:10	Workshop Introduction, Logistics, Goals
9:10-9:25	Spark Theory: Optimal use cases for Spark
9:25-9:40	Spark Theory: Spark Architecture (sparkitecture?)
9:40-9:55	Spark Theory: RDD vs. DataFrame APIs
9:55-10:15	Spark Practice: Env/setup
10:15-10:30	*Break*
10:30-10:50	Spark Practice: Working with spark-shell
10:50-11:10	Spark Practice: Working with zeppelin
11:10-11:45	Spark Practice: Interacting with Real World Data
11:45-12:00	Examples & Wrap-Up

Your Facilitators & Helpers

- **Audrey Altman**, *@AudreyAltman, Digital Public Library of America*
- **Mark Breedlove**, *@markbreedlove, Digital Public Library of America*
- **Michael Della Bitta**, *@mdellabitta, Digital Public Library of America*
- **Scott Williams**, *@moltude, Digital Public Library of America*
- **Corey Harper**, *@chrpr, Elsevier Labs*
- **Justin Coyne**, *@jcoyne, Stanford University Libraries*
- **Erin Fahy**, *@eefahy, Stanford University Libraries*
- **Christina Harlow**, *@cmh2166, Stanford University Libraries*

Communication Channels & Asking for Help

- Alert a facilitator if you need help or have questions.
  - Please feel able to ask for help at any point!
  - Also see if your neighbor can help or needs help.
- Leverage our Code4Lib Slack #sparkinthedark channel.
  - Info about Code4Lib Slack: code4lib.org/slack.
- Open an Issue on Workshop GitHub Repo (for help later).
  - Workshop facilitators are watching this repository.

Our Expectations of You

Personal

- Follow the Code4Lib Code of Conduct
- Follow the Recurse Center Social Rules (a.k.a. "Hacker School Rules")

Technical

- Have Docker & our Docker Container ready to go on your computer
- Be ready to learn about Spark!

Code4Lib Code of Conduct

http://2018.code4lib.org/conduct/

Recurse Center Social Rules
(a.k.a. Hacker School Rules)

- No feigning surprise
- No well-actually's
- No back-seat driving
- No subtle -isms

More info:

- https://www.recurse.com/blog/38-subtle-isms-at-hacker-school
- https://www.recurse.com/manual#sub-sec-social-rules

Spark Technical Prep

1. Brought a Laptop
2. Installed latest stable Docker Community Edition on said laptop
3. In shell, ran (*we will cover this again before break*):
    a. `docker pull mbdpla/sparkworkshop:latest `
    b. `docker run -p 8080:8080 mbdpla/sparkworkshop:latest `
4. Have these slides & our workshop GitHub repository on your computer.

You should have been done beforehand - if not, we will quickly go through again before the break with facilitators able to help you.

Our Goals for this Workshop

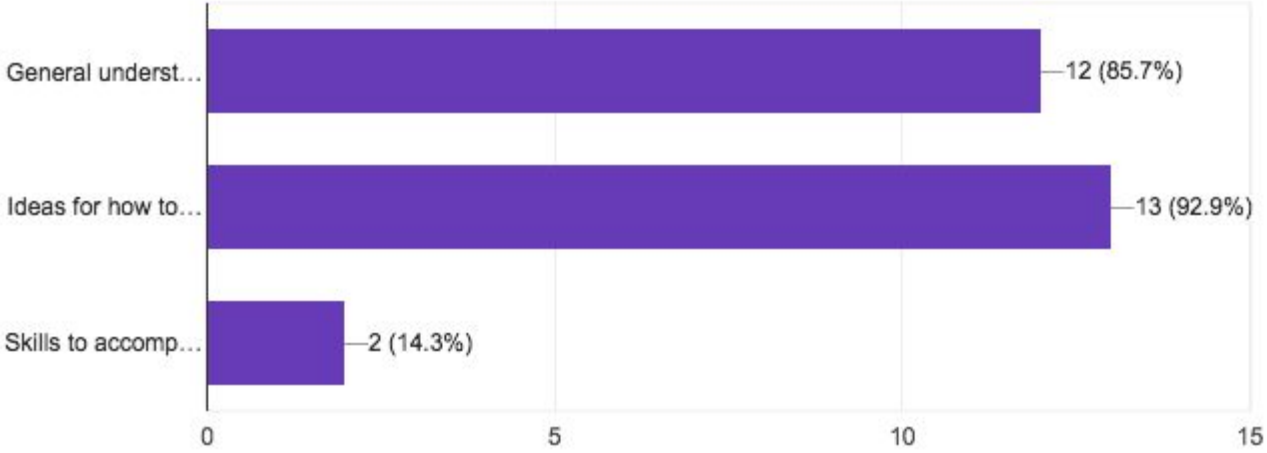
- Introduction to Spark data processing framework.
- High-level overview of Spark’s architecture.
- Learn about the optimal use cases for Spark.
- Practice loading common data formats.
- Learn how to perform out of the box analysis using SparkSQL.
- Learn how to visualize their SparkSQL data results.

Your Goals for this Workshop?

- Why are you attending this workshop?
- What are your goals - immediate or long-term?
- What's your level of comfort & experience with Spark?

What do you hope to get out of this workshop? (Select as many as apply)

14 responses



What are some things you want to do with data that you can't do now?

9 responses

- Mainly I'd like a reliable way to deal with and query very large datasets
- I don't have a specific use case, but I hope to learn about use cases & data operations I hadn't previously thought of.
- help researchers work with our data
- Parse through large datasets and easily identify clusters of subject terms or names that are slight variants.
- I've started exploring NLP and machine learning--I'm looking at potentially huge data sets that I'd like to process in various ways (analyze blocks of text for keywords, "meaning", erroneous entries, etc.).
- Effectively/efficiently work with it at scale; set up pipelines for data verification, validation, mapping, transforming, merging, etc in a hopefully more generalizable and less clunky way; anything useful with all the data locked up in Sierra
- better Spark based functional programming techniques
- No real problems that I can identify currently. Maybe "better" on-demand relationships among data?
- Auto-classification of the archival materials (photos and text) within a collection. Training of machine-learning algorithms on archival data. Running clustering algorithms on metrics extracted from images and docs.

Do you have any particular use cases for Spark & cluster computing, either in your current institution, or more generally?

11 responses

Using Spark to aid in analysis of giant bibliographic dataset(s).

I have vague use cases and hope to learn if this is a fruitful avenue, or I need to look at different tools/methods. One biggish idea is a sort of data warehouse that mirrors the non-sensitive data in the ILS in a way that we can build fast, flexible services (mainly staff-facing, with the ability to merge/map in other data sources like KBART or COUNTER or ...) on top of them for metadata quality analysis, analytics and assessment projects, etc.

mashing up large datasets to produce specialized/customized datasets

I've been trying out the Archives Unleashed Toolkit, which uses Spark for web archives analysis. I'm interested in using that as part of my work. Also possibly for working with social media data. I've done some data analysis experiments with color analysis that I think could run faster with Spark.

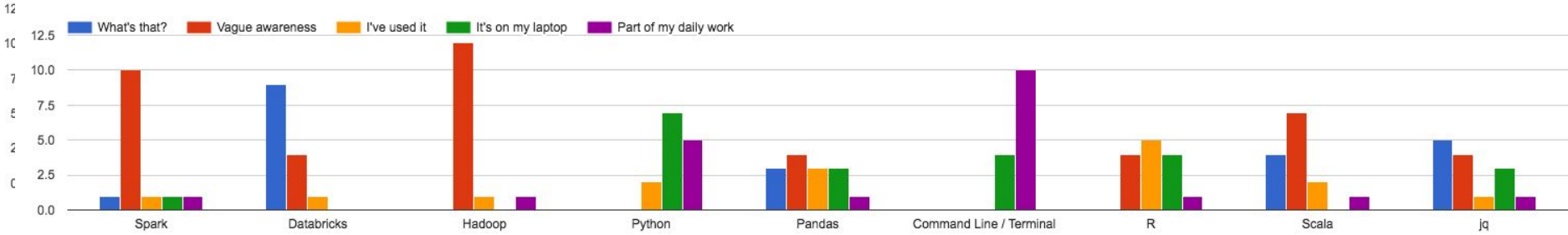
not yet; hoping that will change

Not particularly. More curious for ideas going forward.

We are developing new approaches to repository building and curation at UMD, so there are plenty of use cases. I think a good way to summarize would be to say that we want to put materials in the repository first, then perform the normal pre-ingest work, select etc.. in the repository. We also want to open up linked data to visiting researcher computational approach for discovery, etc.

Learning enough Spark to be able to give advice to students looking to get started with Spark

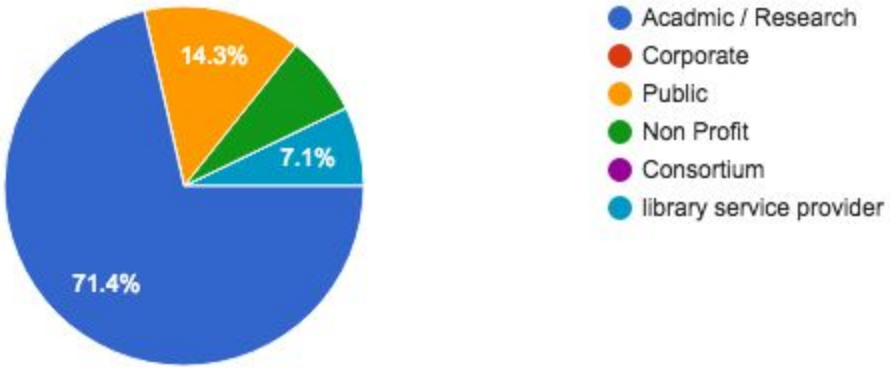
Rate your familiarity or comfort with the following technologies



github.com/spark4lib/code4lib2018

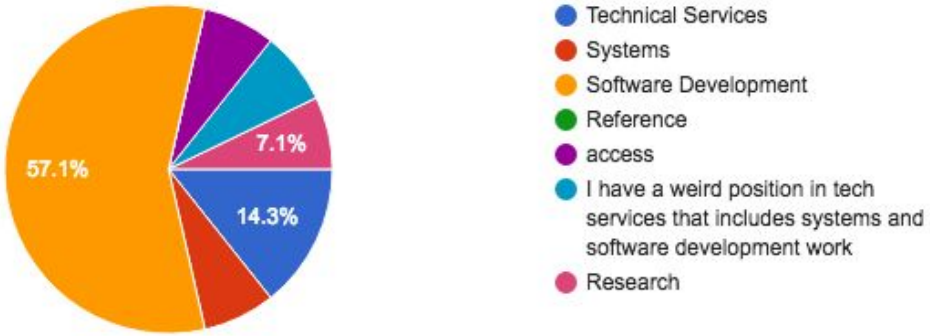
What type of library / institution are you in?

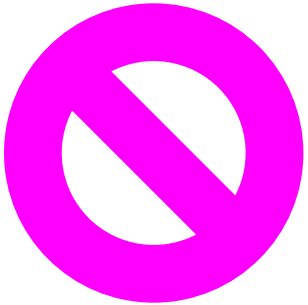
14 responses



What is your role in your library?

14 responses





Checkpoint: Any Questions or Comments?

Spark Theory: Optimal use cases for Spark

What is Spark?

Data processing engine

- Cluster computing
- Big data
- Open source
- Java, Scala, Python, R
- Alternative to Hadoop, MapReduce

History

- Started at UC Berkeley AMPLab in 2009
- Donated to Apache in 2014

Optimal use cases for Spark

- ETL (extract, transform, load) processes
- Batch computation
- Data integration
- Streaming
- Interactive analysis
  - Complex queries without sampling
  - Spark SQL, R, Python
- Machine learning
  - Clustering, classification, regression, etc.
  - Graph manipulation and analysis
- Integrating several of the above into one system

Consider using Spark if you have:

- High volume of data
- Need for speed
- Data from multiple sources (real-time or archived)
- Structured or unstructured data
- Access to sufficient RAM
- Parallelizable jobs
- Engineers to manage production-level software

Spark pros

- Performance is very good
- Scalable
- Fault-tolerant
- Easy to deploy
- Open source
- Interoperable with many other tools for file management, distributed computing, cloud computing, and data visualization
- APIs for Python, Scala, Java, R
- REPL
- Flexible data model
- Strong community of users and developers

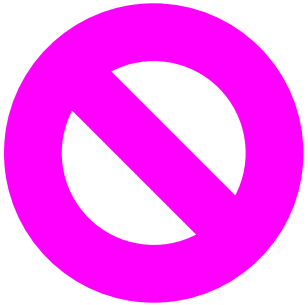
Spark cons

- Configuration for big jobs can be tricky
- Debugging can be a challenge, especially at scale
- Lacks file management system
- Lacks data visualization tools
- Can be inefficient when dealing with many small files
- Optimized for primitive data types
- Not optimal for multi-user environment
- Adolescent / changing software

Examples of industry use

- Recommendation systems
- Fraud detection
- Log processing and analysis
- Sentiment analysis
- Medical research
- More examples: https://spark.apache.org/powered-by.html

Examples of how we are using Spark are coming up in a later section.



Checkpoint: Any Questions or Comments?

“Sparkitecture”

Problem

It’s hard to write
concurrent, scalable systems
using the most standard libraries
built-in to most languages.

Solution

Imagine your programming environment offering a list or array type that could be spread out among many computers automatically.

Solution (cont’d)

Imagine if you needed to perform an
operation across every record
in that data.

It’d be great if you could broadcast
that operation out to the data and have
each computer run it
on the part it had.

Core Data Model:
Resilient Distributed Dataset

“A big array in the cloud”

Resilient Distributed Dataset*

Resilient: Durable against failure.
Can recover from computer / memory / network / storage outages

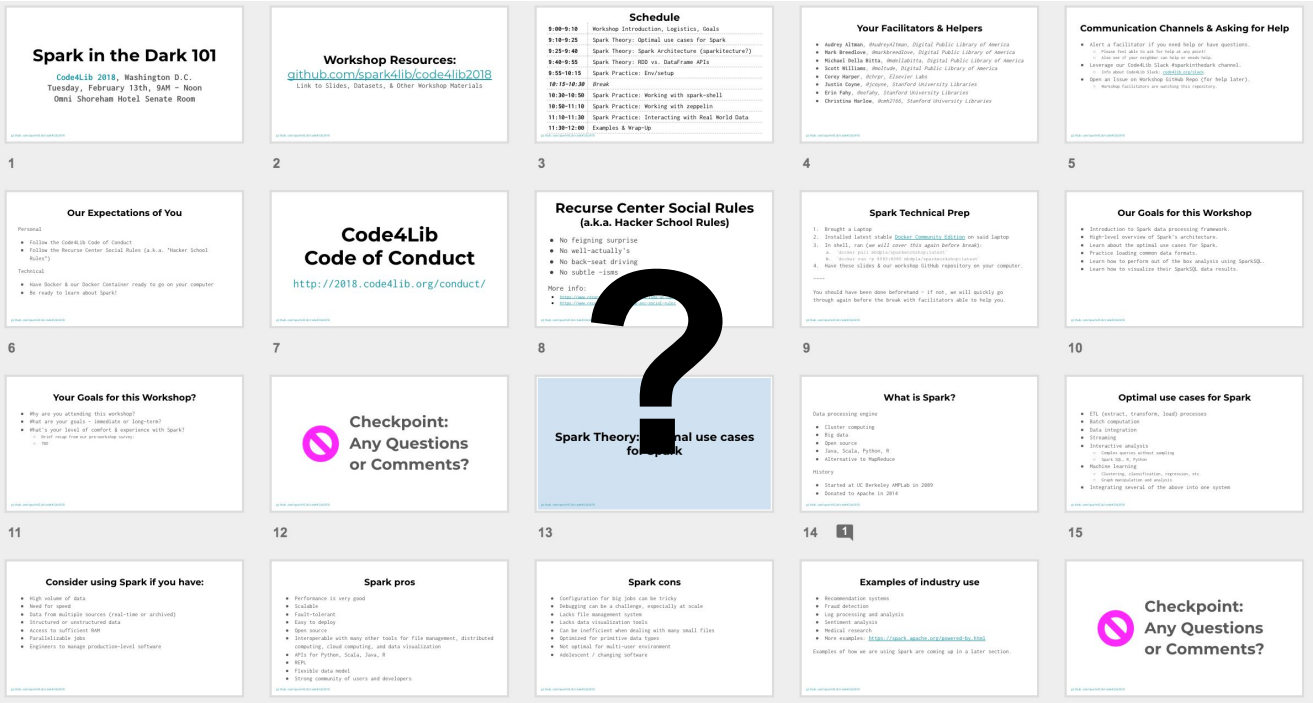
Distributed: Maximizes concurrency and scalability
by spreading the data out among a number of computers.

Dataset: A collection of partitioned data grouped
into structured rows of objects, primitive values or groups of values.
Like a List or an Array in your favorite programming language.†

* "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing"

† Your favorite language is not PHP, Pascal, Lua, or LISP because their arrays are weird.

RDD Metaphor: Presentation Slides



github.com/spark4lib/code4lib2018

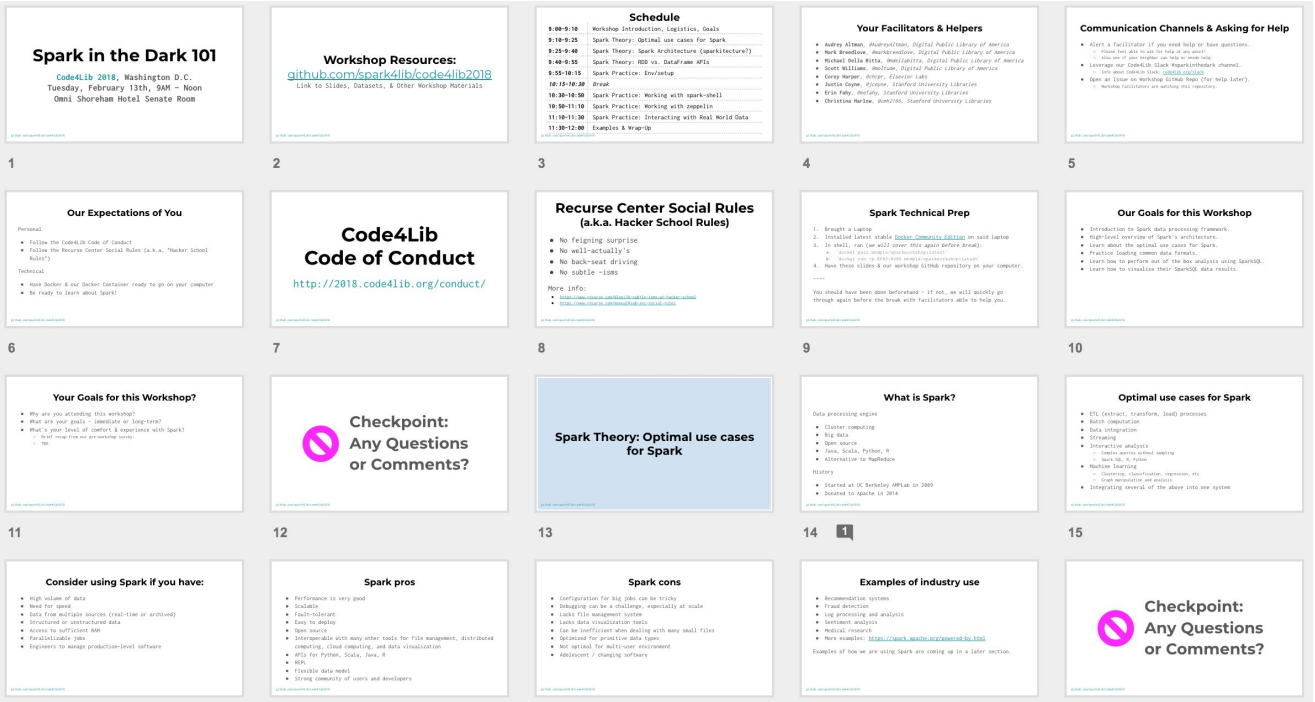
RDD Metaphor: Presentation Slides

Option 1. Single process: Do it all yourself. 🥷

Option 2. Database + Queue: Get some friends to work on it, but they have to ask you what to do for each slide. 🐢

Option 3. Spark: Let everybody be in charge of a range of slides and work in parallel because everyone knows what to do. 🚀

RDD Metaphor: Presentation Slides



github.com/spark4lib/code4lib2018

DataSource

- Spark uses DataSources to abstract over reading and writing from different data stores
- The same workflows you write against one DataSource are portable to others.
- DataSources are pluggable, so anybody can write one for what they need to get done

DataSource Examples

- Flat files:
  - Text
  - CSV
  - JSON
  - XML
  - Avro
  - Parquet
- Datastores:
  - SQL Databases (JDBC)
  - Solr
  - ElasticSearch
  - OAI-PMH
  - HBase
  - Cassandra
  - Redshift
  - Hive

More @ spark-packages.org

Execution Model

Execution Model: Read Only

- Nothing is modified in place
- Every operation that “changes” data does so by making another RDD with the changes applied
- We can go back to different versions of the data and compute other new versions of them with different changes

Execution Model: Lazy

- Data products are calculated not when you describe how to create them (“transformations”), but when you ask for them to be sent somewhere (“actions”).

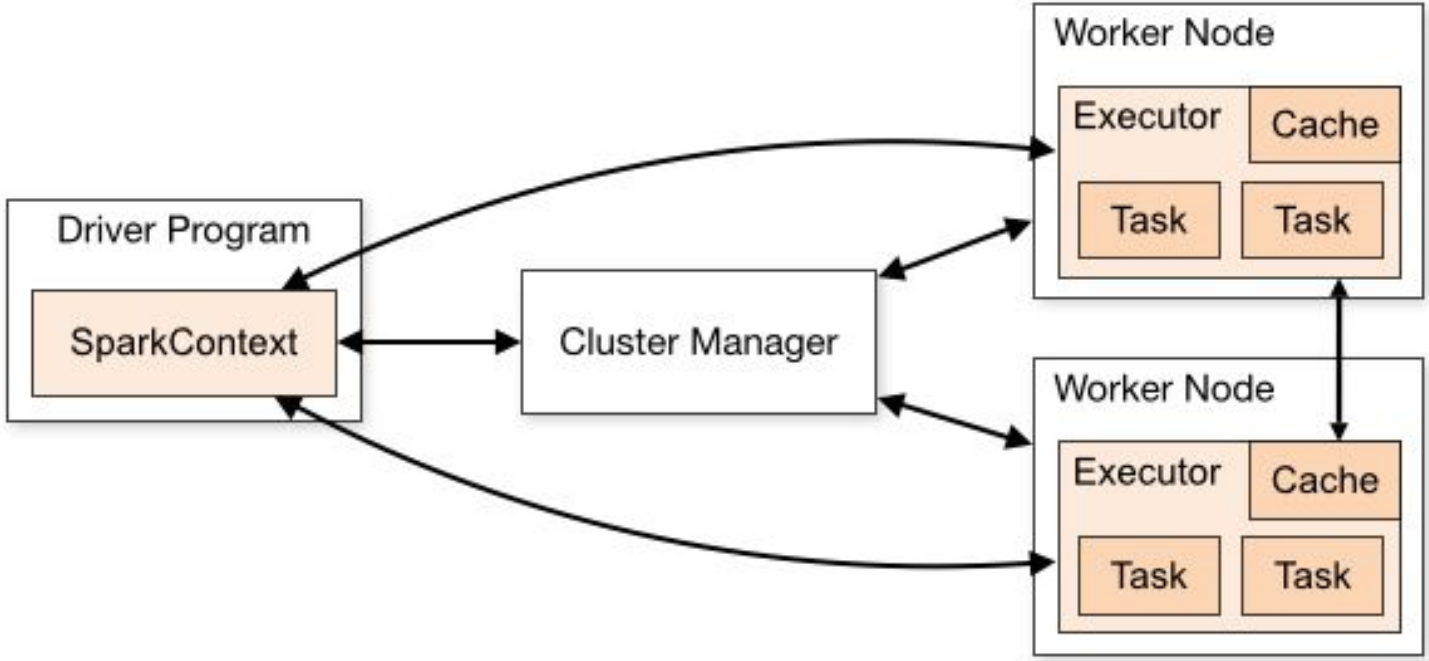
- Laziness lets Spark optimize storage so it can decide to hang on to multiple copies of the data with different transformations, or to build them on the fly if necessary

Performance Characteristics + Tricks

- It tries to run with all data in memory to avoid I/O
- Can cache to disk if you're out of RAM
- Transparent compression
- If you're using DataFrames or DataSets:
  - Code Generation
  - Built-in memory management

Takeaway: They're solving performance problems in ways you never will be able to

Service Layout



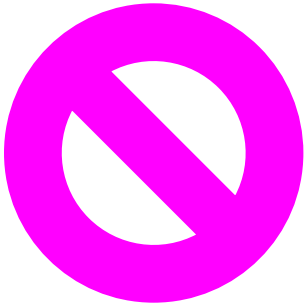
Spark vs. REST API + Message Queue

- Network Latency will swamp processing time for small requests

- Backend database can bottleneck (see Amdahl’s Law)

Spark vs. Hadoop

- Only Map then Reduce operations
- Worker performance unoptimized
  - Worker restarts
  - Heap storage
- Reduce output always written to HDFS



Checkpoint: Any Questions or Comments?

RDD vs DataFrame APIs

RDDs (Resilient Distributed Dataset)

They are...

- Immutable (**resilient**)
- Partitioned (**distributed**)
- Parallelizable
- Typed

They can perform…

- Transformations (map, flatMap, filter, groupByKey)
- Actions (count, first, take, collect, reduce)

RDD example

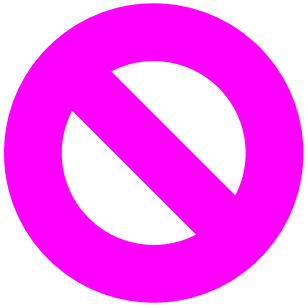
DataFrames

What is the difference?

- Schema with named columns (think of a database table)
- **Query/execution plan optimizations**

DataFrame example

Show read-in, count, group by



Checkpoint: Any Questions or Comments?

BREAK

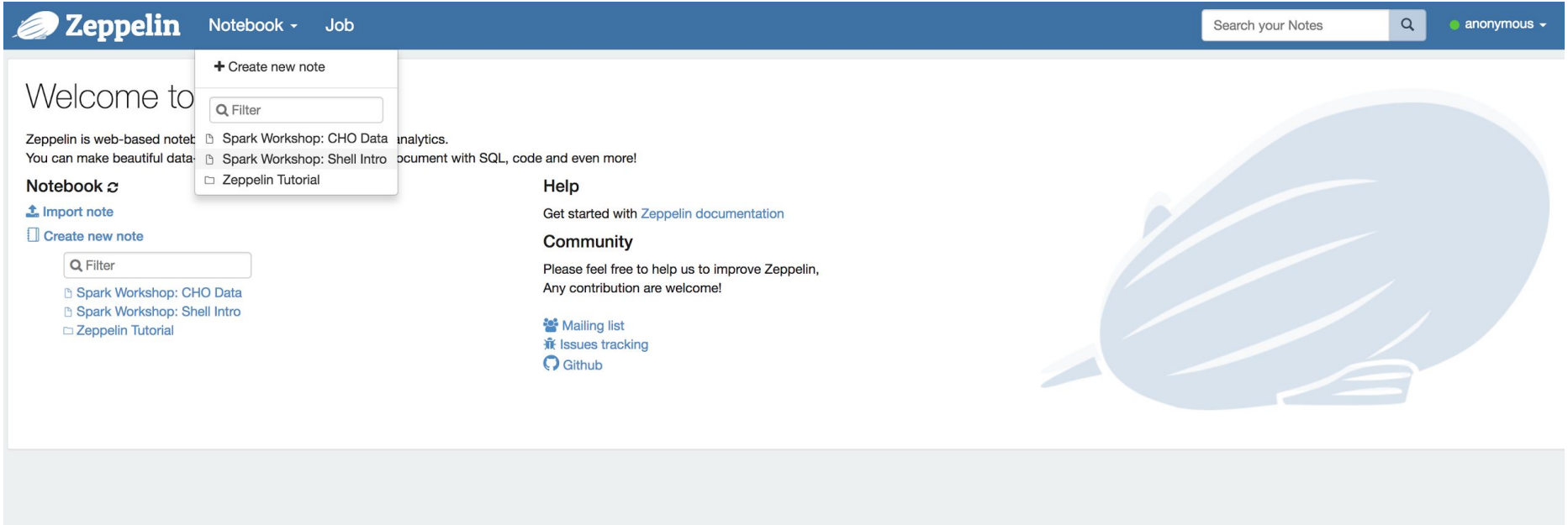
Return at 10:30

Spark “Practice”: Env Setup

Env/setup

- On your laptop:
  - Install latest stable Docker Community Edition (should've done before now!)
  - In your shell, run:
    - `docker pull mbdpla/sparkworkshop:latest `
    - `docker run -p 8080:8080 mbdpla/sparkworkshop:latest `
  - In your web browser, go to http://localhost:8080
  - If wifi is broken, ask a facilitator for a thumbdrive with the Docker image on it.
- What we are doing:
  - Pulling & running a Docker image w/Zeppelin, Spark, Notebooks, & Data preloaded
- Backup Options:
  - If wifi works, try the free Databricks Community account online.
  - If you're feeling up to it, get Zeppelin installed locally on your computer.
  - Either way, you'll need to pull data & notebooks from our Github repository.

Checkpoint: Everyone here? (Raise hands)



github.com/spark4lib/code4lib2018

Spark Practice: Working with spark-shell (pyspark)

Working with spark-shell: Getting Started

In GH Repo, see worksheets/working-with-spark-shell.md.

This is just a **watch + comment section**.

However, if you are feeling ambitious & want to run this Spark Shell walk through yourself, see details in the above worksheet.

Starting a PySpark Interpreter Shell

```
┌<sul.cmharlow @ li-dl-2628-db6b in ~/C/s/g/s/code4lib2018>
└< (add_shell_stuff)* >──» pyspark
Python 3.5.1 (default, May 24 2017, 21:07:43)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
18/02/11 10:53:35 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/02/11 10:53:41 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.2.1
      /_/

Using Python version 3.5.1 (default, May 24 2017 21:07:43)
SparkSession available as 'spark'.
>>>
```

github.com/spark4lib/code4lib2018

Review of Spark Standalone Dashboard

Spark Master at spark://172.17.0.3:7077

URL: spark://172.17.0.3:7077
REST URL: spark://172.17.0.3:6066 *(cluster mode)*
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory

Running Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Our Simple Sample Data

- Just to start, we are using the simple CSV represented below
- Run the first block in our *start-shell* notebook to retrieve this CSV

Name	Institution Type	Format	URL	Description	Informal Score
CMOA	Museum	JSON, CSV	https://github.com/cmoa/collection		10
Penn Museum	Museum	JSON, CSV, XML	https://www.penn.museum/collections/data.php	JSON is poorly structured	7
Met Museum	Museum	CSV		¯_(ツ)_/¯	3
DigitalNZ Te Puna Web Directory	Library	XML	https://natlib.govt.nz/files/data/tepunawebdirectory.xml	MARC XML	3
Canadian Subject Headings	Library	RDF/XML	http://www.collectionscanada.gc.ca/obj/900/f11/040004/csh.rdf	Ugh, rdf	4
DPLA	Aggregator	CSV,JSON,XML	dp.la		100

Reading from File into DataFrame Instance

```
>>> spark.read.csv("sample-data/small-sample.csv")
DataFrame[_c0: string, _c1: string, _c2: string, _c3: string, _c4: string, _c5: string]
>>>
```

Looking at Tabular Format Pretty Printed

```
>>> spark.read.csv("sample-data/small-sample.csv").show()
+--------------------+----------------+--------------+--------------------+--------------------+--------------+
|                 _c0|             _c1|           _c2|                 _c3|                 _c4|           _c5|
+--------------------+----------------+--------------+--------------------+--------------------+--------------+
|                Name|Institution Type|        Format|                 URL|         Description|Informal Score|
|                CMOA|          Museum|     JSON, CSV|https://github.co...|                  10|          null|
|          Penn Museum|          Museum|JSON, CSV, XML|https://www.penn....|JSON is poorly st...|             7|
|           Met Museum|          Museum|           CSV|                null|            ¯_(ツ)_/¯|          null|
|               ","3""|            null|          null|                null|                null|          null|
|DigitalNZ Te Puna...|         Library|           XML|https://natlib.go...|            MARC XML|             3|
|Canadian Subject ...|         Library|       RDF/XML|http://www.collec...|            Ugh, rdf|             4|
|                DPLA|      Aggregator|  CSV,JSON,XML|               dp.la|                 100|          null|
+--------------------+----------------+--------------+--------------------+--------------------+--------------+
```

Providing Dataframe Read Options for CSV

```
>>> spark.read.csv("sample-data/small-sample.csv", header=True, mode="DROPMALFORMED").show()
+--------------------+----------------+-------------+--------------------+--------------------+--------------+
|                Name|Institution Type|       Format|                 URL|         Description|Informal Score|
+--------------------+----------------+-------------+--------------------+--------------------+--------------+
|          Penn Museum|          Museum|JSON, CSV, XML|https://www.penn....|JSON is poorly st...|             7|
|DigitalNZ Te Puna...|         Library|          XML|https://natlib.go...|            MARC XML|             3|
|Canadian Subject ...|         Library|      RDF/XML|http://www.collec...|            Ugh, rdf|             4|
+--------------------+----------------+-------------+--------------------+--------------------+--------------+
```

Handling Multiline CSV (Spark 2.2.x ONLY)

```
>>> spark.read.csv("sample-data/small-sample.csv", header=True, multiLine=True).show()
+-------------------+---------------+-------------+-------------------+-------------------+-------------+
|               Name|Institution Type|       Format|                URL|        Description|Informal Score|
+-------------------+---------------+-------------+-------------------+-------------------+-------------+
|               CMOA|         Museum|     JSON, CSV|https://github.co...|                 10|         null|
|         Penn Museum|         Museum|JSON, CSV, XML|https://www.penn....|JSON is poorly st...|            7|
|          Met Museum|         Museum|          CSV|               null|          ¯_(ツ)_/¯
|             3|
|DigitalNZ Te Puna...|        Library|          XML|https://natlib.go...|           MARC XML|            3|
|Canadian Subject ...|        Library|      RDF/XML|http://www.collec...|           Ugh, rdf|            4|
|               DPLA|     Aggregator|  CSV,JSON,XML|              dp.la|                100|         null|
+-------------------+---------------+-------------+-------------------+-------------------+-------------+
```

Looking at CSV's Inferred Schema

```
>>> spark.read.csv("sample-data/small-sample.csv", header=True, multiLine=True).printSchema()
root
 |-- Name: string (nullable = true)
 |-- Institution Type: string (nullable = true)
 |-- Format: string (nullable = true)
 |-- URL: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Informal Score: string (nullable = true)
```

Providing a CSV's Schema

```
>>> from pyspark.sql.types import *
>>> customSchema = StructType([
...    StructField("Name", StringType(), True),
...   StructField("Institution Type", StringType(), True),
...   StructField("Format", StringType(), True),
...   StructField("URL", StringType(), True),
...   StructField("Description", StringType(), True),
...   StructField("Informal Score", DecimalType(), True)])
>>> sampleDf = spark.read.csv("sample-data/small-sample.csv", header=True, multiLine=True, schema=customSchema)
>>> sampleDf.show()
+-------------------+---------------+-------------+-------------------+-------------------+--------------+
|               Name|Institution Type|       Format|                URL|        Description|Informal Score|
+-------------------+---------------+-------------+-------------------+-------------------+--------------+
|               CMOA|         Museum|    JSON, CSV|https://github.co...|                 10|          null|
|         Penn Museum|         Museum|JSON, CSV, XML|https://www.penn....|JSON is poorly st...|             7|
|          Met Museum|         Museum|          CSV|               null|          ¯_(ツ)_/¯
|             3|
|DigitalNZ Te Puna...|        Library|          XML|https://natlib.go...|           MARC XML|             3|
|Canadian Subject ...|        Library|      RDF/XML|http://www.collec...|           Ugh, rdf|             4|
|               DPLA|     Aggregator|  CSV,JSON,XML|              dp.la|                100|          null|
+-------------------+---------------+-------------+-------------------+-------------------+--------------+
```

Check Rows Number (.count()) & Inspect Rows (.head())

```
>>> sampleDf.count()
6
>>> sampleDf.head(2)
[Row(Name='CMOA', Institution Type='Museum', Format='JSON, CSV', URL='https://github.com/cmoa/collection', Description
='10', Informal Score=None), Row(Name='Penn Museum', Institution Type='Museum', Format='JSON, CSV, XML', URL='https://
www.penn.museum/collections/data.php', Description='JSON is poorly structured', Informal Score=Decimal('7'))]
```

Check Columns & Column

```
>>> sampleDf.columns
['Name', 'Institution Type', 'Format', 'URL', 'Description', 'Informal Score']
>>> len(sampleDf.columns)
6
```

Inspect Specific Column(s)

```
>>> sampleDf.select("Name").show()
+--------------------+
|                Name|
+--------------------+
|                CMOA|
|          Penn Museum|
|           Met Museum|
|DigitalNZ Te Puna...|
|Canadian Subject ...|
|                DPLA|
+--------------------+

>>> sampleDf.select("Name", "Informal Score").show()
+--------------------+--------------+
|                Name|Informal Score|
+--------------------+--------------+
|                CMOA|          null|
|          Penn Museum|             7|
|           Met Museum|             3|
|DigitalNZ Te Puna...|             3|
|Canadian Subject ...|             4|
|                DPLA|          null|
+--------------------+--------------+
```

Filtering Columns

```
>>> scoresBooleanDf = sampleDf.select('Name', 'URL', sampleDf['Informal Score'] > 5)
>>> scoresBooleanDf.show()
+-------------------+-------------------+-------------------+
|               Name|                URL|(Informal Score > 5)|
+-------------------+-------------------+-------------------+
|               CMOA|https://github.co...|               null|
|         Penn Museum|https://www.penn....|               true|
|          Met Museum|               null|              false|
|DigitalNZ Te Puna...|https://natlib.go...|              false|
|Canadian Subject ...|http://www.collec...|              false|
|               DPLA|              dp.la|               null|
+-------------------+-------------------+-------------------+

>>> highScoresDf = sampleDf.filter(sampleDf['Informal Score'] > 5)
>>> highScoresDf.select('Name', 'URL', 'Informal Score').show()
+-----------+-------------------+--------------+
|       Name|                URL|Informal Score|
+-----------+-------------------+--------------+
|Penn Museum|https://www.penn....|             7|
+-----------+-------------------+--------------+
```

Ordering Results

```
>>> sampleDf.orderBy(sampleDf["Informal Score"].desc()).show()
+--------------------+----------------+--------------+--------------------+--------------------+--------------+
|                Name|Institution Type|        Format|                 URL|         Description|Informal Score|
+--------------------+----------------+--------------+--------------------+--------------------+--------------+
|          Penn Museum|          Museum|JSON, CSV, XML|https://www.penn....|JSON is poorly st...|             7|
|Canadian Subject ...|         Library|       RDF/XML|http://www.collec...|            Ugh, rdf|             4|
|           Met Museum|          Museum|           CSV|                null|           ¯_(ツ)_/¯
|              3|
|DigitalNZ Te Puna...|         Library|           XML|https://natlib.go...|            MARC XML|             3|
|                CMOA|          Museum|     JSON, CSV|https://github.co...|                  10|          null|
|                DPLA|      Aggregator|  CSV,JSON,XML|               dp.la|                 100|          null|
+--------------------+----------------+--------------+--------------------+--------------------+--------------+
```

Group Columns by Count

```
>>> sampleDf.groupBy("Format").count().show()
+--------------+-----+
|        Format|count|
+--------------+-----+
|           CSV|    1|
|           XML|    1|
|      JSON, CSV|    1|
|JSON, CSV, XML|    1|
|       RDF/XML|    1|
|  CSV,JSON,XML|    1|
+--------------+-----+
```

Show Distinct values

```
>>> sampleDf.select("Format").distinct().show()
+--------------+
|        Format|
+--------------+
|           CSV|
|           XML|
|     JSON, CSV|
|JSON, CSV, XML|
|       RDF/XML|
|  CSV,JSON,XML|
+--------------+
```

“Explode” Rows

```
>>> from pyspark.sql.functions import *
>>>
>>> sampleDf = sampleDf.withColumn("Format", explode(split("Format", ",")))
>>> sampleDf.show()
+--------------------+----------------+-------+--------------------+--------------------+--------------+
|                Name|Institution Type| Format|                 URL|         Description|Informal Score|
+--------------------+----------------+-------+--------------------+--------------------+--------------+
|                CMOA|          Museum|   JSON|https://github.co...|                  10|          null|
|                CMOA|          Museum|    CSV|https://github.co...|                  10|          null|
|         Penn Museum|          Museum|   JSON|https://www.penn....|JSON is poorly st...|             7|
|         Penn Museum|          Museum|    CSV|https://www.penn....|JSON is poorly st...|             7|
|         Penn Museum|          Museum|    XML|https://www.penn....|JSON is poorly st...|             7|
|          Met Museum|          Museum|    CSV|                null|           ¯_(ツ)_/¯
|             3|
|DigitalNZ Te Puna...|         Library|    XML|https://natlib.go...|            MARC XML|             3|
|Canadian Subject ...|         Library|RDF/XML|http://www.collec...|            Ugh, rdf|             4|
|                DPLA|      Aggregator|    CSV|               dp.la|                 100|          null|
|                DPLA|      Aggregator|   JSON|               dp.la|                 100|          null|
|                DPLA|      Aggregator|    XML|               dp.la|                 100|          null|
+--------------------+----------------+-------+--------------------+--------------------+--------------+
```

Create New Columns with Regex

```
>>> sampleDf = sampleDf.withColumn('CSV', sampleDf.Format.like("%CSV%"))
>>> sampleDf = sampleDf.withColumn('XML', sampleDf.Format.like("%XML%"))
>>> sampleDf = sampleDf.withColumn('RDF', sampleDf.Format.like("%RDF%"))
>>> sampleDf = sampleDf.withColumn('JSON', sampleDf.Format.like("%JSON%"))
>>> sampleDf.show()
+--------------------+----------------+-------+--------------------+--------------------+---------------+-----+-----+-----+-----+
|                Name|Institution Type| Format|                 URL|         Description|Informal Score|  CSV|  XML|  RDF| JSON|
+--------------------+----------------+-------+--------------------+--------------------+---------------+-----+-----+-----+-----+
|                CMOA|          Museum|   JSON|https://github.co...|                  10|           null|false|false|false| true|
|                CMOA|          Museum|    CSV|https://github.co...|                  10|           null| true|false|false|false|
|         Penn Museum|          Museum|   JSON|https://www.penn....|JSON is poorly st...|              7|false|false|false| true|
|         Penn Museum|          Museum|    CSV|https://www.penn....|JSON is poorly st...|              7| true|false|false|false|
|         Penn Museum|          Museum|    XML|https://www.penn....|JSON is poorly st...|              7|false| true|false|false|
|          Met Museum|          Museum|    CSV|                null|           ¯_(ツ)_/¯
|              3| true|false|false|false|
|DigitalNZ Te Puna...|         Library|    XML|https://natlib.go...|            MARC XML|              3|false| true|false|false|
|Canadian Subject ...|         Library|RDF/XML|http://www.collec...|            Ugh, rdf|              4|false| true| true|false|
|                DPLA|      Aggregator|    CSV|              dp.la|                 100|           null| true|false|false|false|
|                DPLA|      Aggregator|   JSON|              dp.la|                 100|           null|false|false|false| true|
|                DPLA|      Aggregator|    XML|              dp.la|                 100|           null|false| true|false|false|
+--------------------+----------------+-------+--------------------+--------------------+---------------+-----+-----+-----+-----+
```

github.com/spark4lib/code4lib2018

Format Repaired...?

```
>>> sampleDf.select("Format").distinct().show()
+-------+
| Format|
+-------+
|    CSV|
|    XML|
|    CSV|
|    XML|
|RDF/XML|
|   JSON|
+-------+
```

Stripping Whitespace Using Regex

```
>>> sampleDf = sampleDf.withColumn("Format", regexp_replace(sampleDf.Format, "\s+", ""))
>>> sampleDf.select("Format").distinct().show()
+-------+
| Format|
+-------+
|    CSV|
|    XML|
|RDF/XML|
|   JSON|
+-------+
```

Getting XML Format Datasets

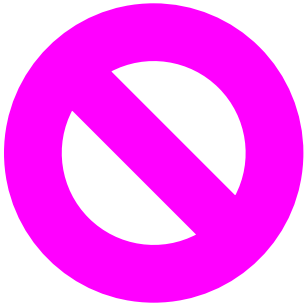
```
>>> sampleDf.where(col('Format') == 'XML').show()
+--------------------+----------------+------+--------------------+--------------------+--------------+-----+----+-----+-----+
|               Name|Institution Type|Format|                 URL|         Description|Informal Score|  CSV| XML|  RDF| JSON|
+--------------------+----------------+------+--------------------+--------------------+--------------+-----+----+-----+-----+
|         Penn Museum|          Museum|   XML|https://www.penn....|JSON is poorly st...|             7|false|true|false|false|
|DigitalNZ Te Puna...|         Library|   XML|https://natlib.go...|            MARC XML|             3|false|true|false|false|
|               DPLA|      Aggregator|   XML|               dp.la|                 100|          null|false|true|false|false|
+--------------------+----------------+------+--------------------+--------------------+--------------+-----+----+-----+-----+

>>> sampleDf.where(col('Format') == 'XML').count()
3
```

github.com/spark4lib/code4lib2018

Creating SQL Table View & Querying

```
>>> sampleDf.createOrReplaceTempView("sampleDataView")
>>> query = spark.sql("""SELECT `Institution Type`, COUNT(DISTINCT(Format)) As NumFormats
...                      FROM sampleDataView
...                      GROUP BY `Institution Type`""")
>>> query.collect()
[Row(Institution Type='Library', NumFormats=2), Row(Institution Type='Aggregator', NumFormats=3), Row(Institution Ty
pe='Museum', NumFormats=3)]
>>> for n in query.collect():
...     n
...
Row(Institution Type='Library', NumFormats=2)
Row(Institution Type='Aggregator', NumFormats=3)
Row(Institution Type='Museum', NumFormats=3)
```

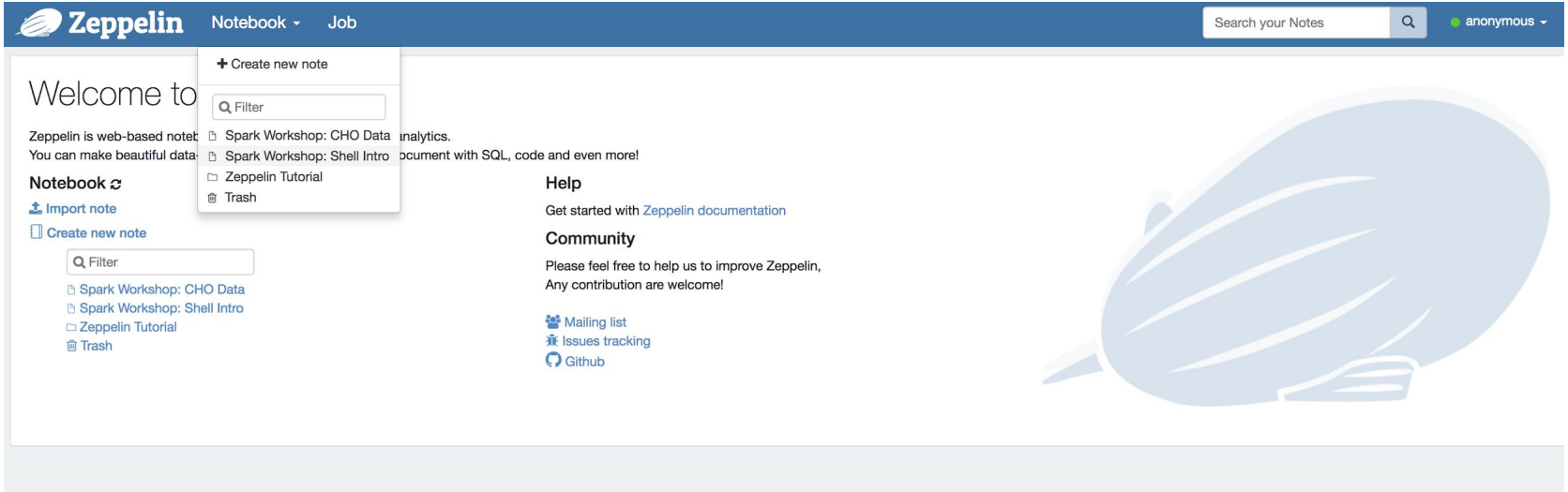


Checkpoint: Any Questions or Comments?

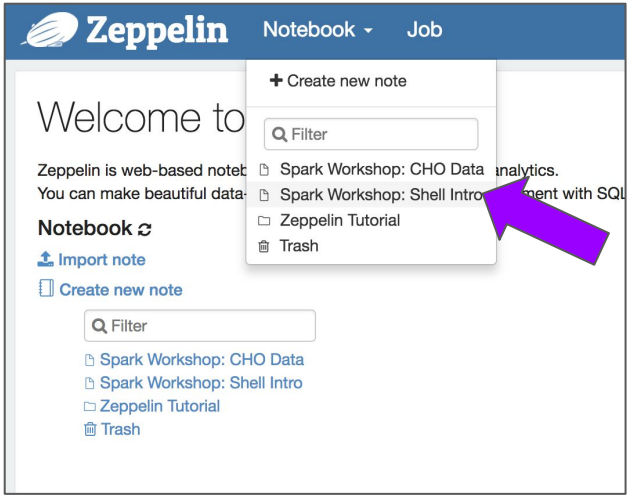
Spark Practice: Working with Zeppelin Notebooks

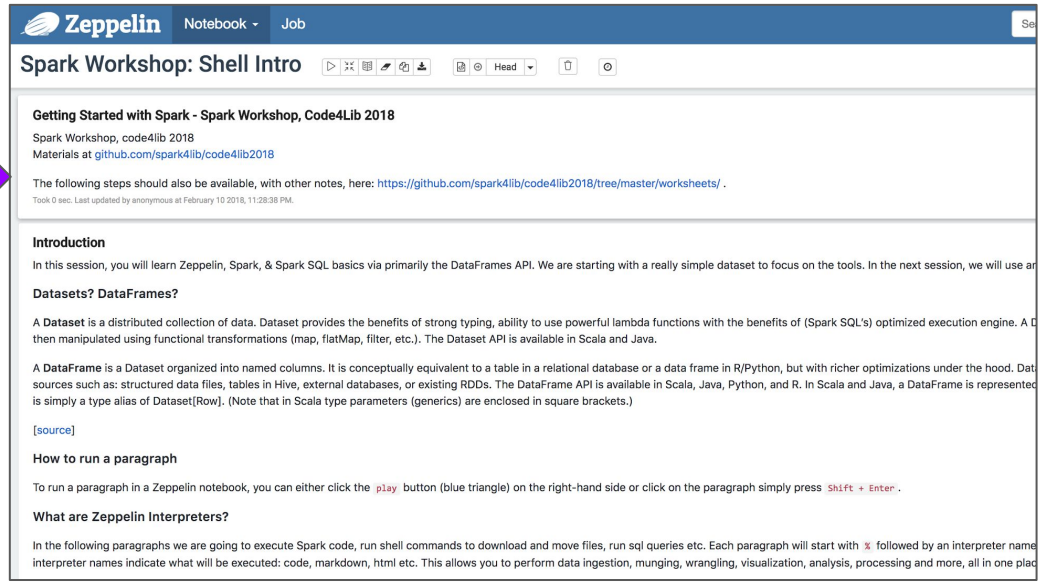
Working with Zeppelin Notebooks

- Moving from Watch & Learn to Hands-On with Zeppelin.
- Follow along in Zeppelin & at worksheets/working-with-zeppelin.md.
- Run Zeppelin w/our `Spark Workshop: Shell Intro` from workshop Docker Container & GH Repository: notebooks/SparkWorkshopShellIntro.json



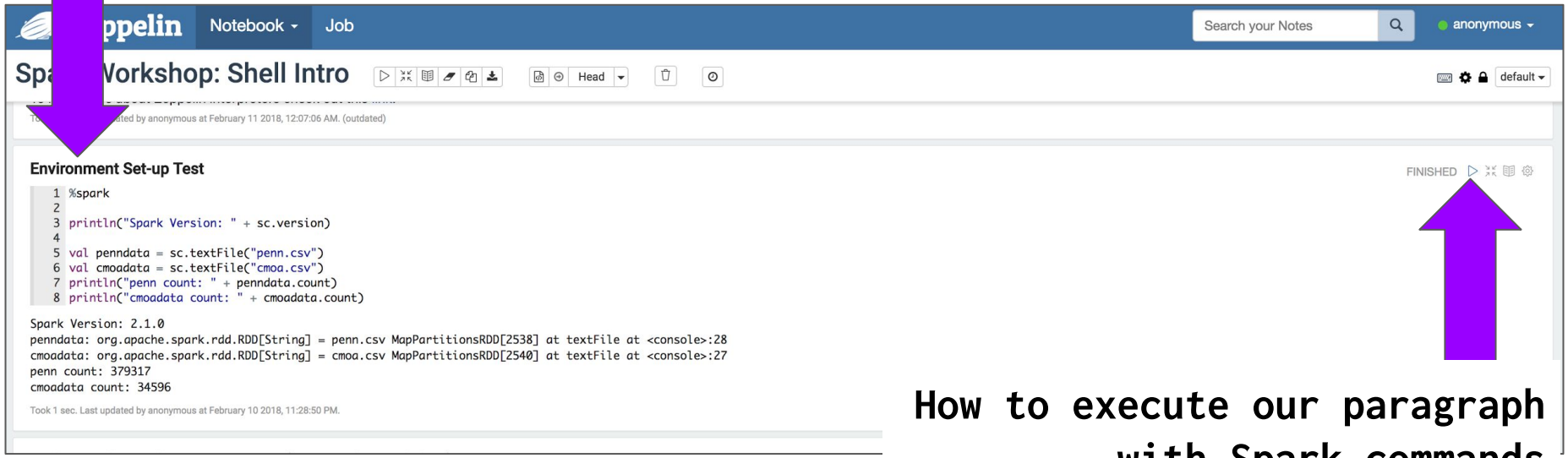
Getting to Our Zeppelin Notebook





github.com/spark4lib/code4lib2018

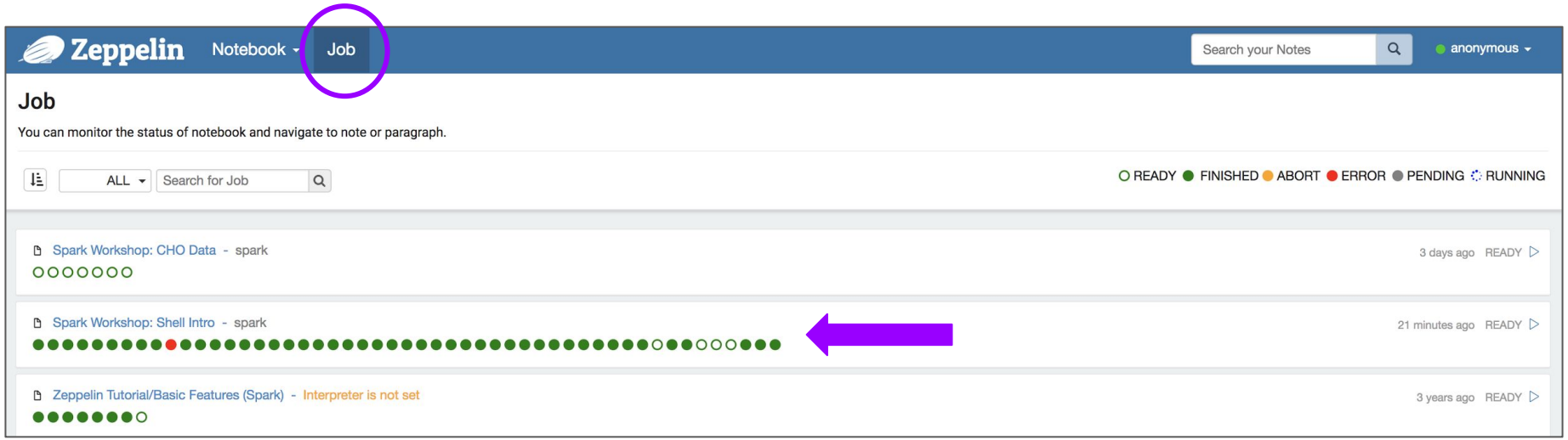
Our Spark commands



How to execute our paragraph with Spark commands

Checkpoint: Everyone (still) here? (Raise hands)

Working with Zeppelin: Jobs Dashboard



Noting Our Sample Data & Its Location

Grab Actual Sample Data & Preview (Remove for Workshop)

FINISHED

```
%sh
curl https://raw.githubusercontent.com/spark4lib/code4lib2018/master/sample-data/small-sample.csv > small-sample.csv
cat small-sample.csv
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0100   554  100   554    0     0   1207      0 --:--:-- --:--:-- --:--:--  1209
Name,Institution Type,Format,URL,Description,Informal Score
CMOA,Museum,"JSON, CSV",https://github.com/cmoa/collection,,10
Penn Museum,Museum,"JSON, CSV, XML",https://www.penn.museum/collections/data.php,JSON is poorly structured,7
Met Museum,Museum,CSV,,"¯\_(ツ)_/¯
",3
DigitalNZ Te Puna Web Directory,Library,XML,https://natlib.govt.nz/files/data/tepunawebdirectory.xml,MARC XML,3
Canadian Subject Headings,Library,RDF/XML,http://www.collectionscanada.gc.ca/obj/900/f11/040004/csh.rdf,"Ugh, rdf",4
DPLA,Aggregator ,"CSV,JSON,XML",dp.la,,100
```

Took 1 sec. Last updated by anonymous at February 10 2018, 11:30:55 PM.

Our Sample Data

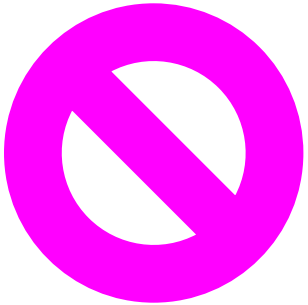
FINISHED

Name	Institution Type	Format	URL	Description	Informal Score
CMOA	Museum	“JSON, CSV”	https://github.com/cmoa/collection		10
Penn Museum	Museum	“JSON, CSV, XML”	https://www.penn.museum/collections/data.php	JSON is poorly structured	7
Met Museum	Museum	CSV		“¯_(ツ)_/¯ ”	3
DigitalNZ Te Puna Web Directory	Library	XML	https://natlib.govt.nz/files/data/tepunawebdirectory.xml	MARC XML	3
Canadian Subject Headings	Library	RDF/XML	http://www.collectionscanada.gc.ca/obj/900/f11/040004/csh.rdf	“Ugh, rdf”	4
DPLA	Aggregator	“CSV,JSON,XML”	dp.la		100

Took 0 sec. Last updated by anonymous at February 10 2018, 11:31:05 PM.

Shell Start Zeppelin Notebook Overview

- Review Sample Data
- Read from CSV into DataFrame
  - spark.read.csv Options
  - Repair of Oddities
  - DF Schemas - Provided & Inferred
  - View Tabular format of DF (.show())
- Simple Data Analysis
  - count(), head(), show()
  - describe() / describe(colName)
  - columns(), len()
  - select(), filter()
  - orderBy(), groupBy(), agg()
  - distinct(), countDistinct(colName)
- Derive New DataFrame(s) / Dataset(s)
  - dropDuplicates(), dropna(), fillna()
  - isNull()
  - withColumn()
  - split(), explode()
  - like(regex)
  - rdd, map(lambda), toDf()
  - drop()
  - regexp_replace()
  - where()
- SQL Queries
  - createOrReplaceTempView()
  - spark.sql(), collect()
- Simple SQL-driven Viz



Checkpoint: Any Questions or Comments?

Spark Practice: Working with Real World Data

What We've Learned So Far...

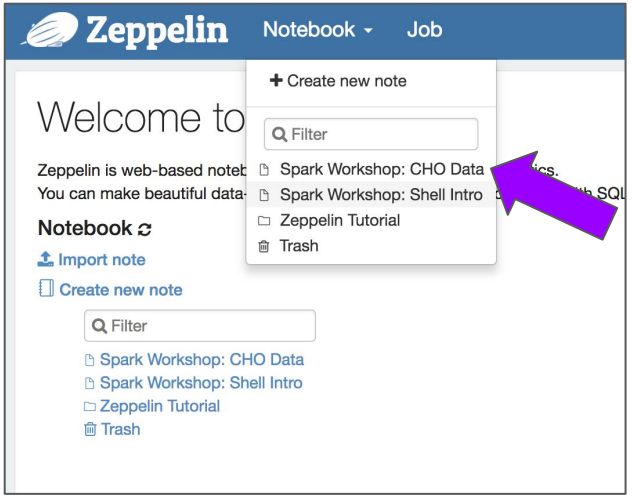
- Read from CSV into DataFrame
  - spark.read.csv Options
  - Repair of Oddities
  - DF Schemas - Provided & Inferred
  - View Tabular format of DF (.show())
- Simple Data Analysis
  - count(), head(), show()
  - describe() / describe(colName)
  - columns(), len()
  - select(), filter()
  - orderBy(), groupBy(), agg()
  - distinct(), countDistinct(colName)
- Derive New DataFrame(s) / Dataset(s)
  - dropDuplicates(), dropna(), fillna()
  - isNull()
  - withColumn()
  - split(), explode()
  - like(regex)
  - rdd, map(lambda), toDf()
  - drop()
  - regexp_replace()
  - where()
- SQL Queries
  - createOrReplaceTempView()
  - spark.sql(), collect()
- Simple SQL-driven Viz

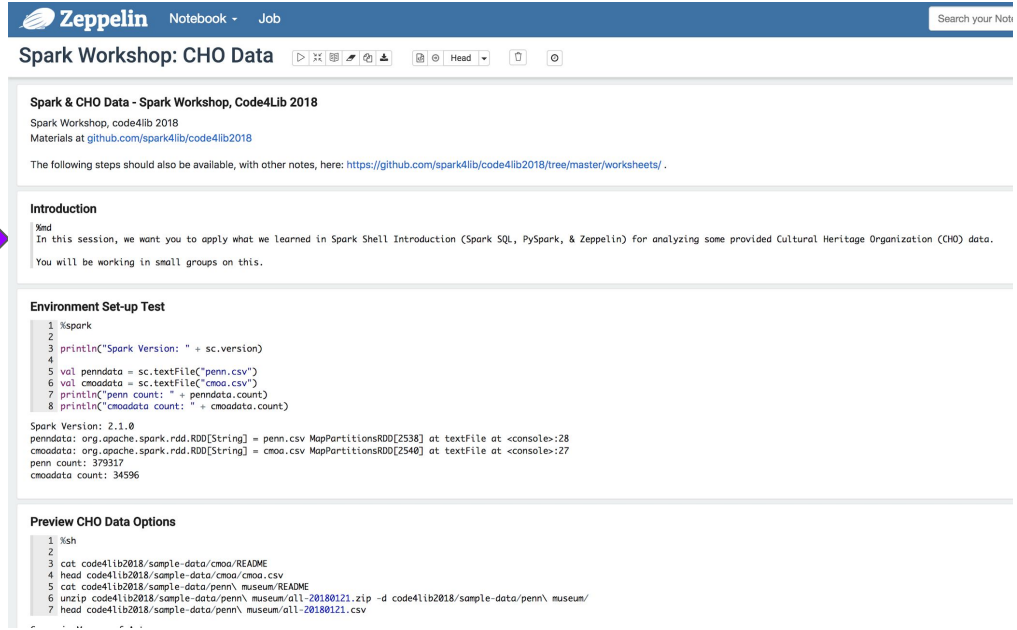
Spark Functions CheatSheet:

worksheets/functions-list.md

We’re Working in Groups Now & You’re (mostly) in the Driver’s Seat

Getting to Our Zeppelin Notebook





github.com/spark4lib/code4lib2018

CHO Data Options

See the CHO dataset files here in sample-data/ in our GH Repo.

FYI, you'll need to unzip the Penn dataset using the `%sh` interpreter in Zeppelin (so that it unzips in your Docker container instance).

Name	Institution Type	Format	URL	Description	Informal Score
CMOA	Museum	JSON, CSV	https://github.com/cmoa/collection		10
Penn Museum	Museum	JSON, CSV, XML	https://www.penn.museum/collections/data.php	JSON is poorly structured	7

Example: Select & Read New Data into DF

Create Our Own Schema & Read the CSV into a Dataframe Instance with our Schema

```
from pyspark.sql.types import *

pennSchema = StructType([
  StructField("emuIRN", IntegerType(), False),
  StructField("curatorial_section", StringType(), True),
  StructField("object_number", StringType(), True),
  StructField("object_name", StringType(), True),
  StructField("native_name", StringType(), True),
  StructField("culture", StringType(), True),
  StructField("provenience", StringType(), True),
  StructField("material", StringType(), True),
  StructField("period", StringType(), True),
  StructField("date_made", StringType(), True),
  StructField("date_made_early", StringType(), True),
  StructField("date_made_late", StringType(), True),
  StructField("accession_credit_line", StringType(), True),
  StructField("creator", StringType(), True),
  StructField("description", StringType(), True),
  StructField("manufacture_locationlocus", StringType(), True),
  StructField("culture_area", StringType(), True),
  StructField("technique", StringType(), True),
  StructField("iconography", StringType(), True),
  StructField("measurement_height", StringType(), True),
  StructField("measurement_length", StringType(), True),
  StructField("measurement_width", StringType(), True),
  StructField("measurement_outside_diameter", StringType(), True),
  StructField("measurement_tickness", StringType(), True),
  StructField("measurement_unit", StringType(), True),
  StructField("other_numbers", StringType(), True),
  StructField("url", StringType(), True)])


pennDf = spark.read.csv("code4lib2018/sample-data/penn\ museum/all-20180121.csv", header=True, schema=pennSchema)
pennDf.count()
```

379316

Took 1 sec. Last updated by anonymous at February 11 2018, 9:01:00 PM.

github.com/spark4lib/code4lib2018

Example: Sampling DataFrames

Get Sample Response of our DataFrame

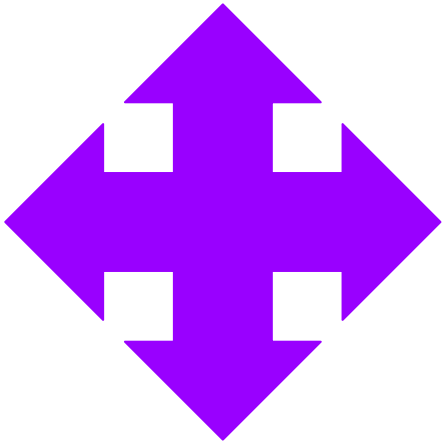
FINISHED

```
pennDf.sample(False, 0.10, 42).show()
```

```
+------+-----------------+-------------+-------------+-----------+-------+-----------------+--------+------+-----------+---------------+-------------+---------------------+-------+---------------------+-----------------
------+-----------------+-------------+-------+-----------+-----------+-----
```

In Groups: Select & Read New Data into DF

Use quick analysis to decide which data to read & if the DataFrame read it in okay



Checkpoint:

What did Groups do?

Example: Run Some Analyses on Your DF

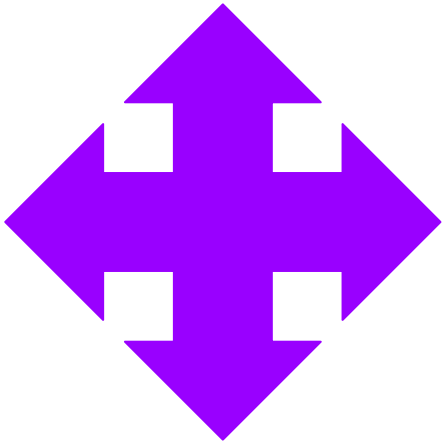
"Earliest" , Non-Null Dates for Objects?

```
pennDf.filter(pennDf.date_made.isNotNull()).orderBy("date_made_early").select('emuIRN', 'date_made', 'object_name', 'object_number', 'culture_area').show(10)
```

```
+------+--------------------+--------------------+-------------+------------+
|emuIRN|           date_made|         object_name|object_number|culture_area|
+------+--------------------+--------------------+-------------+------------+
|290887| 16th - 17th century|               Aegis|       AF2074|        null|
|340184|Late 18th/Early 19th|Hip-pendant|Penda...|       AF2085|        null|
|284193|        19th century|Hip-pendant|Penda...|       AF2086|        null|
|506858|   Late 19th Century|Game Board|Mancal...|   2015-22-18|        null|
|298979|  Early 19th century|Hip-pendant|Penda...|      AF2078A|        null|
|506888|Early to Mid- 20t...|Game Board|Mancal...|   2015-22-48|        null|
|506929|Early to Mid- 20t...|Game Board|Mancal...|   2015-22-88|        null|
|180915|            pre 1903|Kettledrum|Drumst...|      51-46-3|North Africa|
|506961|19th to Early 20t...|Game Board|Mancal...|  2015-22-116|        null|
|506880|  Early 20th Century|Game Board|Mancal...|   2015-22-40|        null|
+------+--------------------+--------------------+-------------+------------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at February 11 2018, 9:07:50 PM. (outdated)

In Groups: Run Some Simple Analyses



Checkpoint:

What did Groups do?

Example: Derive New Dataframes

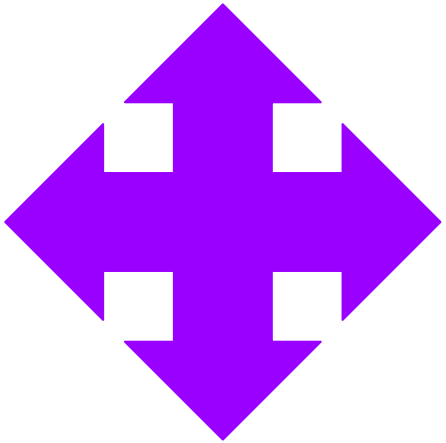
Get DataFrame with Records that Have no Title, Creator or Description

```
noDescDf = pennDf.filter(pennDf.creator.isNull()).filter(pennDf.description.isNull()).filter(pennDf.object_name.isNull())
print(noDescDf.count())
noDescDf.select('description', 'creator', 'emuIRN', 'curatorial_section', 'object_name').show(10, truncate=True)
```

```
53
+-----------+-------+------+-----------------+-----------+
|description|creator|emuIRN|curatorial_section|object_name|
+-----------+-------+------+-----------------+-----------+
|       null|   null|154595|          American|       null|
|       null|   null|160058|          American|       null|
|       null|   null| 82833|          American|       null|
|       null|   null|181698|          American|       null|
|       null|   null| 15540|          American|       null|
|       null|   null|204212|          American|       null|
|       null|   null|137368|          American|       null|
|       null|   null| 81009|          American|       null|
|       null|   null| 81010|          American|       null|
|       null|   null|524120|          American|       null|
+-----------+-------+------+-----------------+-----------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at February 11 2018, 9:58:53 PM. (outdated)

In Groups: Derive New Dataframes



Checkpoint:

What did Groups do?

Example: Run Spark SQL Queries on DFs

Query Penn DataFrame View for Number of Cultures per Curatorial Section

```
sparkQuery = spark.sql("""SELECT curatorial_section, COUNT(DISTINCT(culture)) AS NumCultures FROM curatorialView GROUP BY curatorial_section""")
for n in sparkQuery.collect():
    n
```

```
Row(curatorial_section=u'European', NumCultures=214)
Row(curatorial_section=u'African', NumCultures=21)
Row(curatorial_section=u'Near Eastern', NumCultures=123)
Row(curatorial_section=u'Historic', NumCultures=19)
Row(curatorial_section=u'Babylonian', NumCultures=7)
Row(curatorial_section=u'Egyptian', NumCultures=51)
Row(curatorial_section=u'Asian', NumCultures=134)
Row(curatorial_section=u'Mediterranean', NumCultures=194)
Row(curatorial_section=u'American', NumCultures=180)
```

Took 3 sec. Last updated by anonymous at February 11 2018, 10:29:53 PM.

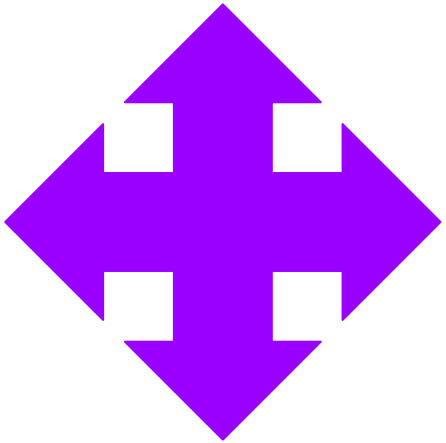
Query Penn DataFrame View for Number of No Title-Creator-Description Records per Curatorial Section

```
sparkQuery2 = spark.sql("""SELECT curatorial_section, COUNT(DISTINCT(emuIRN)) AS NumResources FROM noDescView GROUP BY curatorial_section""")
for n in sparkQuery.collect():
    n
```

```
Row(curatorial_section=u'European', NumCultures=214)
Row(curatorial_section=u'African', NumCultures=21)
Row(curatorial_section=u'Near Eastern', NumCultures=123)
Row(curatorial_section=u'Historic', NumCultures=19)
Row(curatorial_section=u'Babylonian', NumCultures=7)
Row(curatorial_section=u'Egyptian', NumCultures=51)
Row(curatorial_section=u'Asian', NumCultures=134)
Row(curatorial_section=u'Mediterranean', NumCultures=194)
Row(curatorial_section=u'American', NumCultures=180)
```

Took 2 sec. Last updated by anonymous at February 11 2018, 10:30:23 PM. (outdated)

In Groups: Run Some Spark SQL Queries on your DataFrames as Views



Checkpoint:

What did Groups do?

Example: Create a Simple Sql-Based Viz

```
%sql
SELECT curatorial_section,
COUNT(DISTINCT(culture)) AS NumCultures
FROM curatorialView
GROUP BY curatorial_section
```

FINISHED

settings

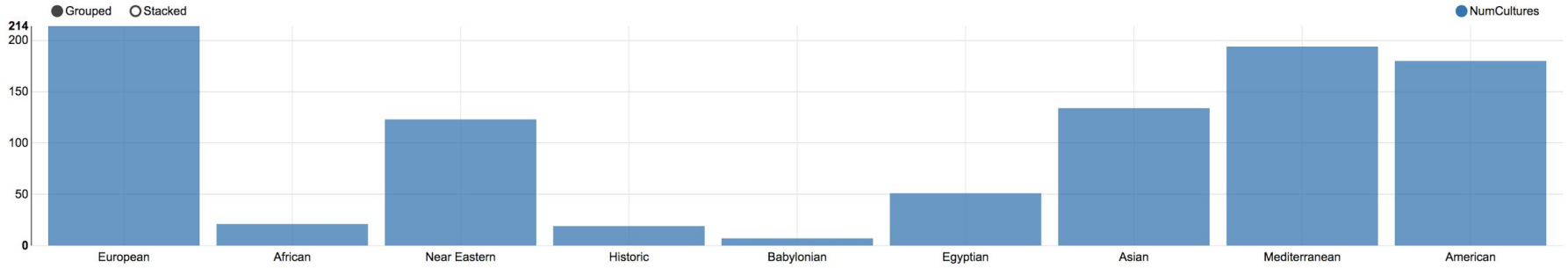
Bar Chart

curatorial_section | NumCultures

Keys: curatorial_section

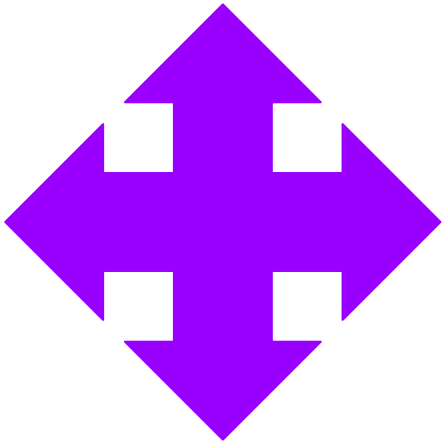
Groups:

Values: NumCultures SUM



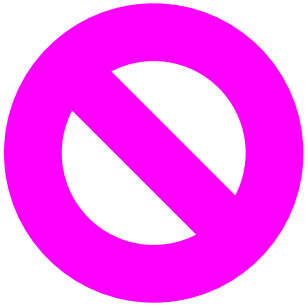
github.com/spark4lib/code4lib2018

In Groups: Create a Simple Sql-Based Viz with Your DataFrames, Views, Queries



Checkpoint:

What did Groups do?



Checkpoint: Any Questions or Comments?

Spark Example & Wrap-Up

Case studies from GLAM+

DPLA & Ingestion3

https://github.com/dpla/ingestion3

DPLA Ingestion 3

This project is an ETL system for cultural heritage metadata. The system has three primary components:

Harvesters

- OAI-PMH
- ResourceSync TBD
- API TBD
- File TBD
- Primo TBD

Mappers

- TBD

Enrichments

- String enrichments
- Linked data enrichments

OAI-PMH Harvester

For additional information please read the announcement blog post or the complete documentation on our

Clone the repo and build the JAR

Web Archiving

https://github.com/archivesunleashed/aut

https://github.com/archivesunleashed/auk

```
import io.archivesunleashed.spark.matchbox._
import io.archivesunleashed.spark.rdd.RecordRDD._

val r = RecordLoader.loadArchives("/aut-resources/Sample-Data/*.gz", sc)
.keepValidPages()
.map(r => ExtractDomain(r.getUrl))
.countItems()
.take(10)
```

Europeana QA

https://github.com/pkiraly/europeana-qa-spark

europeana-qa-spark

Spark interface

According to my measurements running the same code in Spark is 3-4 times faster than running it on Hadoop. For running it on Spark, you should install Hadoop, Scala and finally Spark. The JSON files are stored in Hadoop Distributed File System at `/europeana` directory. In my case Hadoop's core-site.xml has `fs.default.name` property value: `hdfs://localhost:54310`, so I Spark access the files as `hdfs://localhost:54310/europeana/*.json`. The result will go to HDFS's `/result` directory. If that directory exists Spark (such as Hadoop) stops, so you should remove it first. The run on all Europeana records takes roughly two hours, so it is worth to run it at background and with nohup.

```
hdfs dfs -rm -r -skipTrash /result
spark-submit --class com.nsdr.spark.CompletenessCount \
  --master local target/europeana-qa-spark-1.0-SNAPSHOT-jar-with-dependencies.jar \
  hdfs://localhost:54310/europeana/*.json \
  hdfs://localhost:54310/result \
  data-providers.txt \
  datasets.txt
hdfs dfs -getmerge /result result.csv
```

Repository Processing Back-Ends

https://github.com/trellis-ldp/trellis

https://github.com/duraspace/lambdora

https://github.com/sul-dlss-labs/taco

Keeping this Going

Keeping This Conversation Going…

- Use / Reuse our Workshop GitHub Repo!
- Spark in the Dark Channel & Informal Calls
- LDCX Data Architecture Track
- Data4Lib Conference?
- Look for Big Data Infrastructure Partnerships?
- … Something else? What do you think?

Thank you!

Code4Lib 2018, Washington D.C.
github.com/spark4lib/code4lib2018