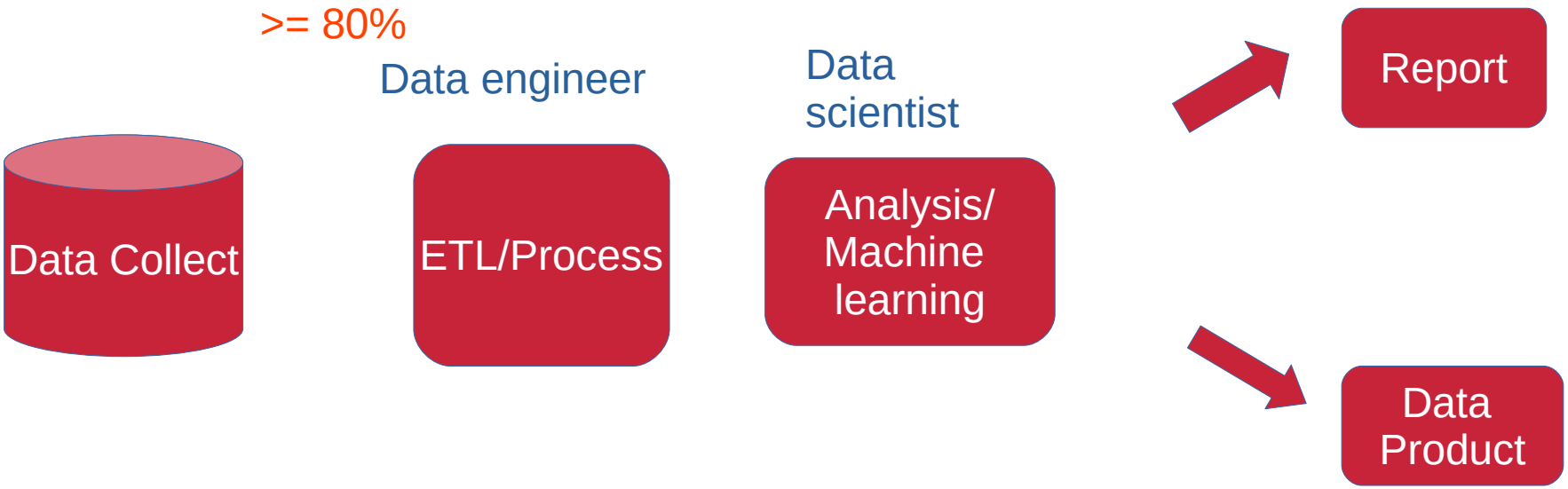
Data Science with Python workshop

- Fetch data: Nodejs Loopback API
- Data processing: PySpark
- Data visualization: Zeppelin
- Keras (optional)

The lifecycle of data (maybe science)



Data collection – Nodejs API Framework Loopback

What is Loopback?
A framework for creating APIs on frontend for data distributed and connect them with backend datasources. Loopback is all about serve side but it attaches client side to your serve.

Built on top of Express and can take a data model definition, easily generate a fully functional end-to-endREST API that can be called by any client.

- API Explorer
check the results of our data with different GET/POST/ etc methods

Data collection – Nodejs API Framework Loopback

Why?

The popularity about nodejs dev and its all about js. Easy setup (no coding skills required) and manage APIs, powerful db connectors.

Reality: many datasources and its formats varied

Our case today – API from Shopex (API integration provider for Tmall, JD….)

- API listen from Rest API
- load data to local warehouse MySQL
- use data as datasource for the business intelligence usage

Data collection – Nodejs Loopback API

Set up Loopback API models to match the database schema

EXPLORER

WORKSHOP2019

- api
  - client
  - common
    - models
      - sale-items.js
      - sale-items.json
      - saleslist.js
      - saleslist.json
  - node_modules
  - server
  - package-lock.json
  - package.json
  - README.md
  - yarn.lock
- client
  - README.md U
- cron
- server
- .editorconfig
- .eslintignore

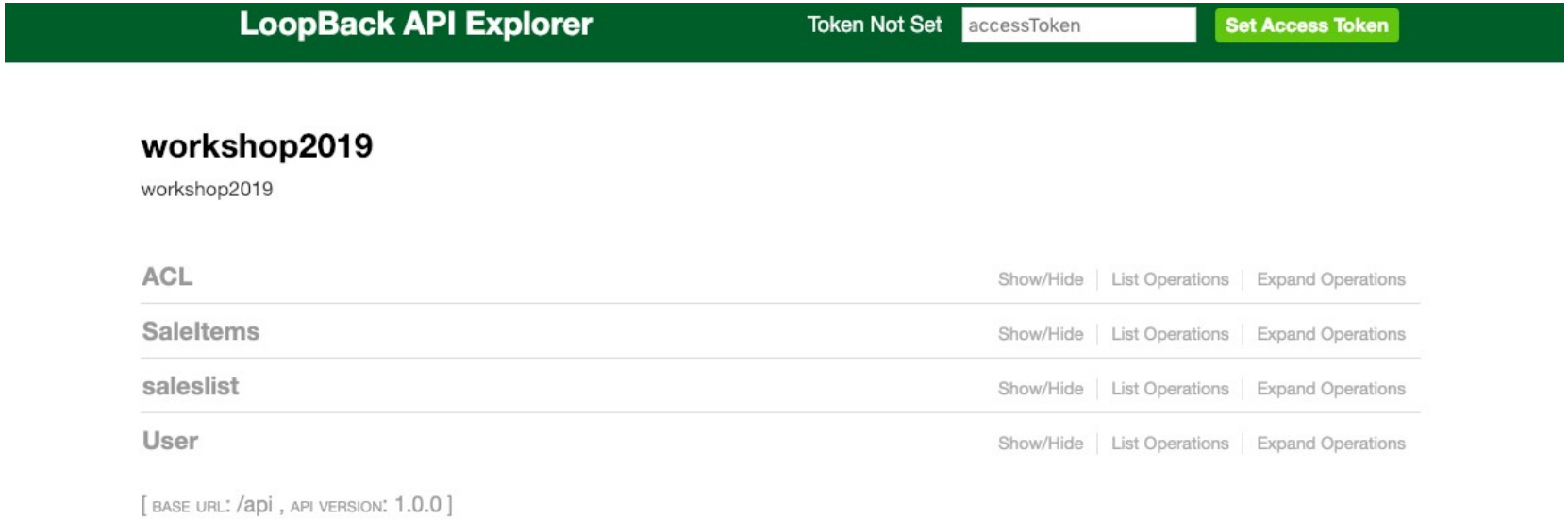
saleslist.json

```
{
  "name": "saleslist",
  "plural": "saleslist",
  "base": "PersistedModel",
  "idInjection": false,
  "options": {
    "validateUpsert": true
  },

    "properties": {
      "sale_no": {
        "type": "string",
        "id": true,
        "generated": false,
        "required": true,
        "comments": "Sale ID"
      },
      "shop_code": {
        "type": "string",
        "required": true,
        "comments": "Shop code"
      },
      "shop_name": {
        "type": "string",
        "required": true
      },
```

Data collection – Nodejs API Framework Loopback

Set up Loopback API models and check API from frontend



Data collection – Nodejs API Framework Loopback

LoopBack API Explorer Token Not Set accessToken **Set Access Token**

saleslist Show/Hide | List Operations | Expand Operations

PATCH /saleslist — Patch an existing model instance or insert a new one into the data source.

GET /saleslist — Find all instances of the model matched by filter from the data source.

Response Class (Status 200)

Request was successful

Model | Example Value


```
    "sale_time": "2019-01-25T09:06:14.572Z",
    "order_create_time": "2019-01-25T09:06:14.572Z",
    "pay_time": "2019-01-25T09:06:14.572Z",
    "order_check_time": "2019-01-25T09:06:14.572Z",
    "delivery_no": "string",
    "ship_time": "2019-01-25T09:06:14.572Z",
    "branch_name": "string",
    "branch_bn": "string",
    "consignee": "string",
    "consignee_area": "string",
    "consignee_addr": "string",
    "consignee_zip": "string"
```

Data collection – Nodejs API Framework Loopback

LoopBack API Explorer Token Not Set accessToken Set Access Token

must be a JSON-encoded string
({"something":"value"})

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:3000/api/saleslist'
```

Request URL

```
http://localhost:3000/api/saleslist
```

Response Body


```
[
  {
    "sale_no": "S201810042000001",
    "shop_code": "deita01",
    "shop_name": "deltaq旗舰店-天猫",
    "order_no": "232548224725334456",
    "member_name": "",
    "member_uname": "",
    "logi_name": "申通E物流",
    "logi_no": "3378416885881",
    "pay_method": "支付宝"
```

Data collection – Nodejs API Framework Loopback

Load data into local database MySQL/PostgreSQL (or other unstructured db)

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| delta_schema       |
| deltadb            |
| deltadb_test       |
| information_schema |
| mysql              |
| performance_schema |
| schema             |
| sys                |
| test               |
| workshop2019       |
+--------------------+
10 rows in set (0.00 sec)

mysql> use workshop2019;
Database changed
mysql> show tables;
+------------------------+
| Tables_in_workshop2019 |
+------------------------+
| ACL                    |
| saleitems              |
| saleslist              |
+------------------------+
3 rows in set (0.01 sec)

mysql>
```

```
mysql> describe saleslist;
+-------------------+--------------+------+-----+-------------------+-------+
| Field             | Type         | Null | Key | Default           | Extra |
+-------------------+--------------+------+-----+-------------------+-------+
| sale_no           | varchar(255) | NO   | PRI | NULL              |       |
| shop_code         | varchar(512) | NO   |     | NULL              |       |
| shop_name         | varchar(512) | NO   |     | NULL              |       |
| order_no          | varchar(60)  | YES  |     | NULL              |       |
| member_name       | varchar(512) | YES  |     | NULL              |       |
| member_uname      | varchar(512) | YES  |     | NULL              |       |
| logi_name         | varchar(512) | NO   |     | NULL              |       |
| logi_no           | varchar(512) | YES  |     | NULL              |       |
| pay_method        | varchar(512) | YES  |     | NULL              |       |
| sale_amount       | float        | NO   |     | NULL              |       |
| pmt_amount        | float        | NO   |     | NULL              |       |
| goods_amount      | float        | NO   |     | NULL              |       |
| freight_amount    | float        | NO   |     | NULL              |       |
| additional_amount | float        | YES  |     | NULL              |       |
| has_tax           | tinyint(1)   | YES  |     | NULL              |       |
| order_check_op    | varchar(512) | YES  |     | NULL              |       |
| sale_time         | datetime     | NO   |     | CURRENT_TIMESTAMP |       |
| order_create_time | datetime     | YES  |     | NULL              |       |
| pay_time          | datetime     | YES  |     | NULL              |       |
| order_check_time  | datetime     | YES  |     | NULL              |       |
| delivery_no       | varchar(512) | YES  |     | NULL              |       |
| ship_time         | datetime     | YES  |     | NULL              |       |
| branch_name       | varchar(512) | YES  |     | NULL              |       |
| branch_bn         | varchar(512) | YES  |     | NULL              |       |
| consignee         | varchar(512) | YES  |     | NULL              |       |
| consignee_area    | varchar(512) | YES  |     | NULL              |       |
| consignee_addr    | varchar(512) | YES  |     | NULL              |       |
| consignee_zip     | varchar(512) | YES  |     | NULL              |       |
| consignee_tel     | varchar(512) | YES  |     | NULL              |       |
| consignee_email   | varchar(512) | YES  |     | NULL              |       |
| consignee_mobile  | varchar(512) | YES  |     | NULL              |       |
| is_test           | varchar(512) | YES  |     | NULL              |       |
+-------------------+--------------+------+-----+-------------------+-------+
32 rows in set (0.19 sec)
```

PySpark : Data Processing (aka ETL)

What is Spark?

- Apache Open source project focus on **in-memory distributed, iterative computing and** DAG for data flow
- API – Scala, **Python**, Java and R
- Built-in datasource API with **JSON**, **JDBC**, Parquet, HDFS, **MySQL, PostgreSQL**, **S3**, H2 and etc

Why Spark?
Effective and fast for iterative computations and ML
Single machine for data munging and ML
Rich libraries such as for ML and runs everywhere Hadoop, Mesos, standalone, cloud
Powerful I/O, bring data from multiple datasource to join/query in Spark
Compare spark with pandas, when data exceeds the capacity on the single machine, slow

Jargon

Jobs: a pieces of code which reads some input from HDFS or local, perform some computation on the data and writes some output data

Stages: Jobs are divided into stages

Tasks: Each stage has some tasks, one task per partition

DAG: a directed acyclic graph for workflow

Executor: the process responsible for executing a task

Master: the machine on which the Driver program runs

Slave: the machine on which the executor program runs

Diagram?

RDD – Resilient Distributed Dataset
- Primary abstraction in Spark
An **Immutable (read only)** collection of objects that can be operated in parallel

- Distributed
Each RDD is composed of one or more partitions => parallelism

- Resilient
Recover from node failures
An RDD keeps its lineage information → it can recreated from its parent RDD

In Spark all work is expressed as creating new RDDs, transforming existing RDDs, or calling operations on RDDs to compute a result.

Spark learning material

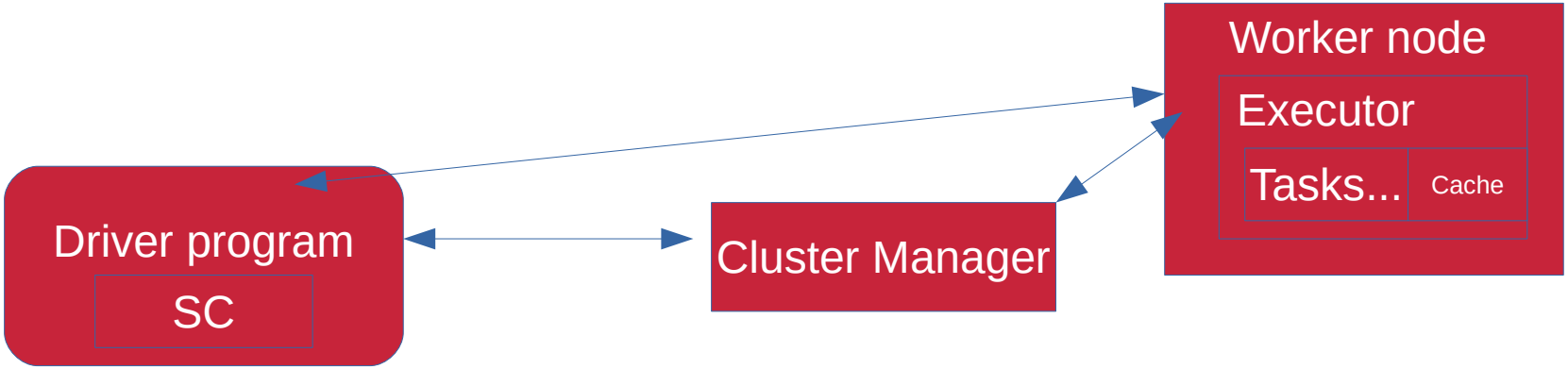
Operations – RDD Transformation (Lazy and object-oriented methods)
What is lazy, don’t create dataset until action is performed, RDD generation
Transformation create but not execute

Filter()
Map()
Distinct()
GroupBy()
SortBy()
Join()
Union()
Distinct()
….

Operations – RDD Actions (output to driver/file system)
Count()
Collect()
Take()/ TakeSample()
Reduce()
Aggregate()
ForEach()/forEachPartition()
Max()
Min()
Sum()
Mean()
Variance()
Stdev()
SaveTextFile()
CountByKey/Value()

Persist() /cache() => store in memory

df=sc.textFile(path) #sc is variable name for Spark Context, which is the main entry point for spark functionality and connect to Spark cluster, which also creates RDDs
df.filter(func)
….




```
df.show()
df.printSchema()
df.select(`fieldName`).show()
df.filter(df(`fieldName`)>num).show(10)
…..
```

Spark SQL

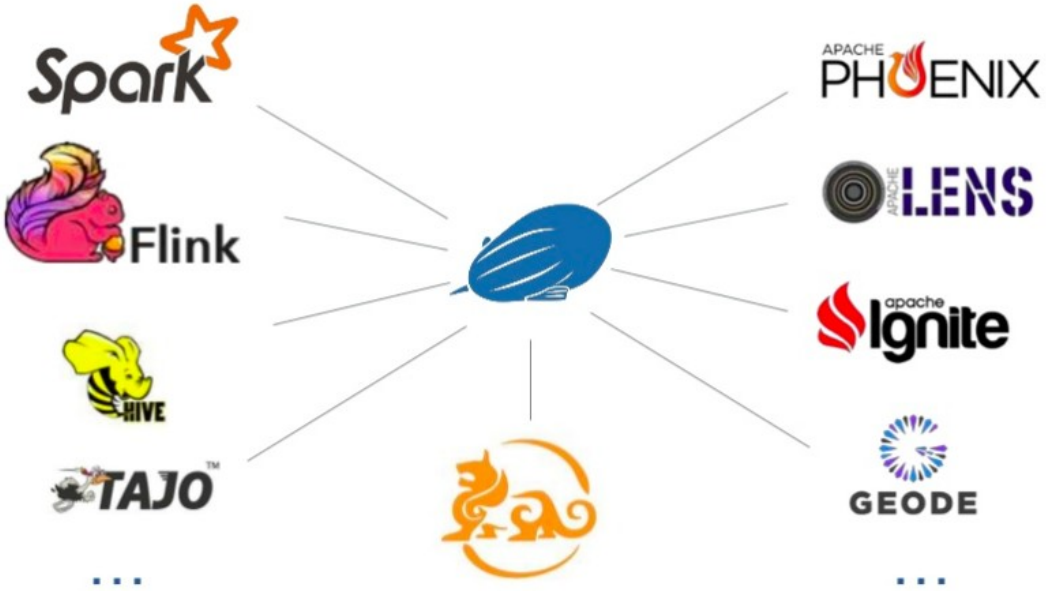
It offers interactive querying capabilities with low latency. module for structured data processing (db), easy manipulate data with their API

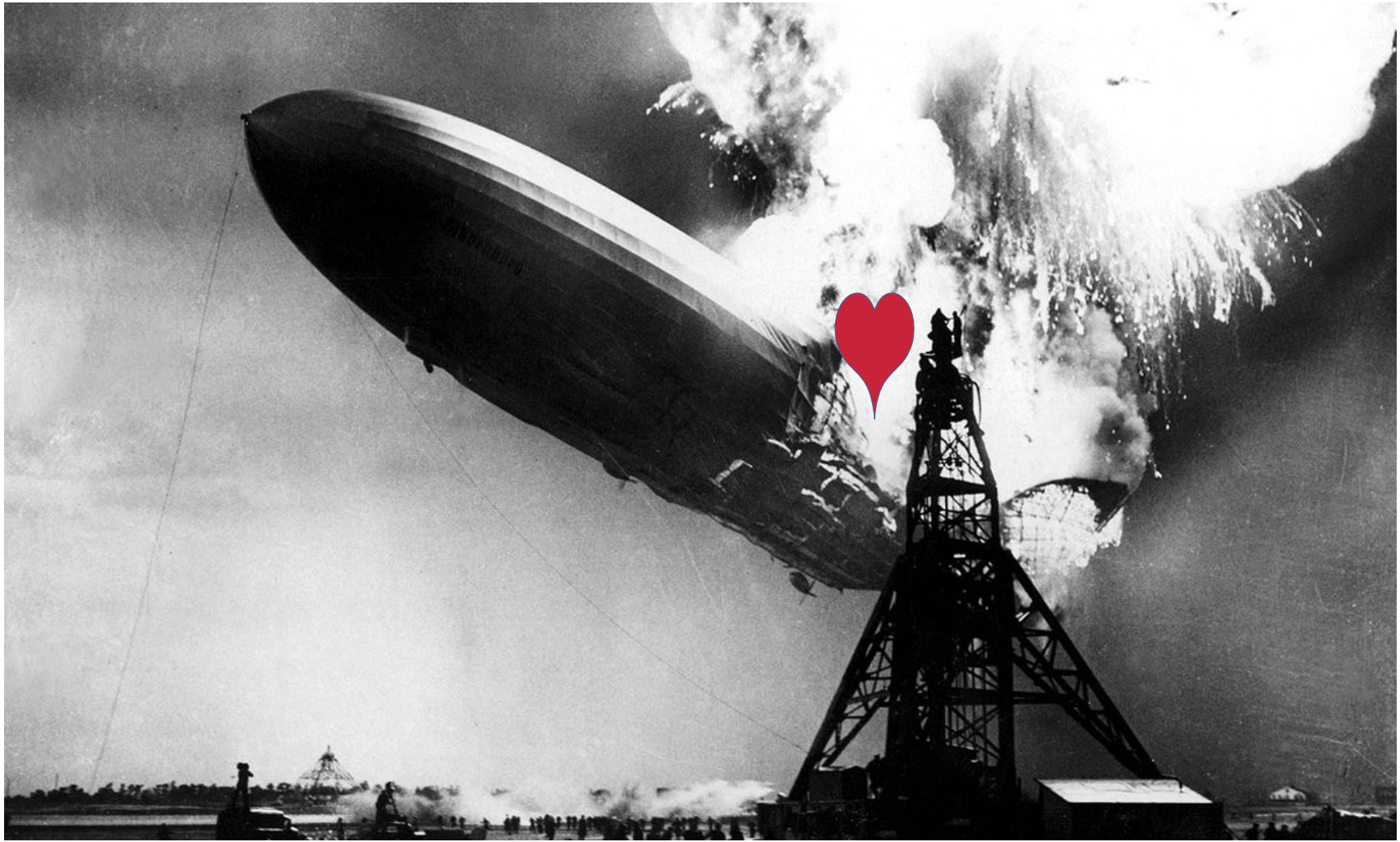
- DataFrames(abstractions) API
- SQL queries
- Datasets API)

```
df.registerTempTable(`tableName`)
SqlContext.sql("select * from tableName").show()
```

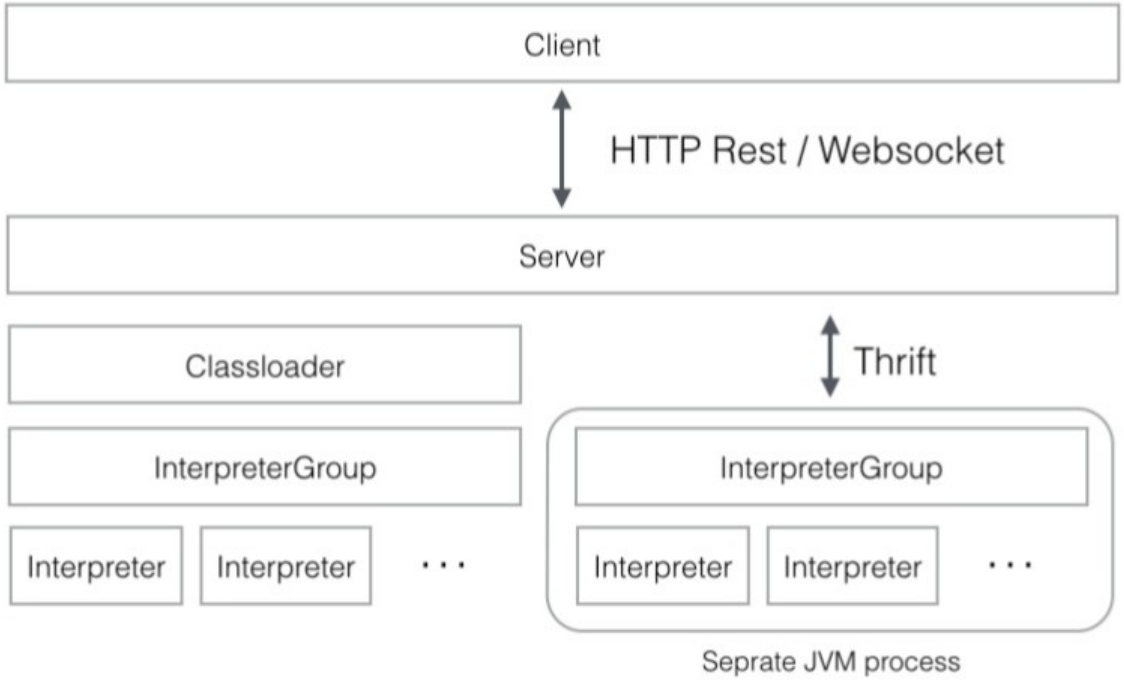
APACHE ZEPPELIN

Software





Zeppelin Architecture



APACHE ZEPPELIN

- A web-based notebook for interactive data viz and analytics, Ingestion, also version control And collaboration (notebook UI style), query and reports in one place

? why zeppelin?

?how to call and import data ?

- Zeppelin supports multi-language backend, such as %pyspark, %sql, %md, %sh

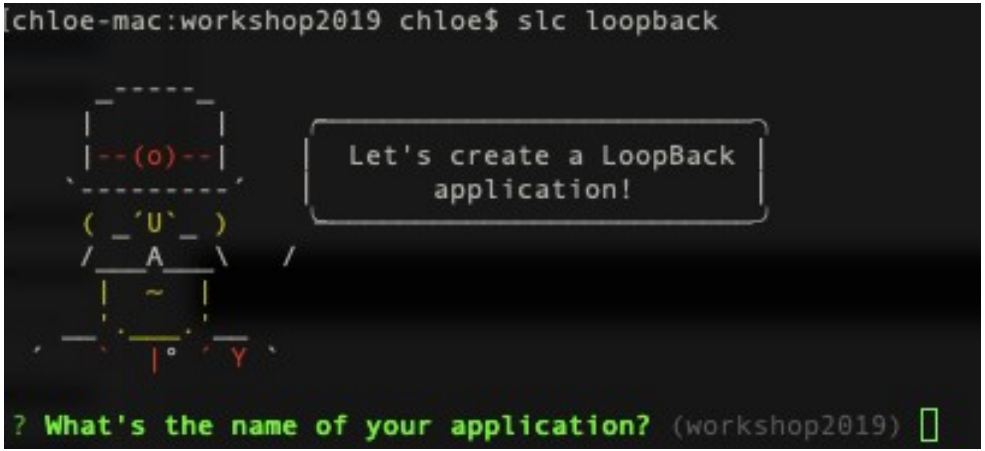
Online tutorial! (in few mins)

```
$ bin/zeppelin-daemon.sh start
```

Data collection – Nodejs API framework Loopback

Set up demo!

$ npm install -g loopback-cli (optionally use Yarn)
`$ slc loopback`



Set up guide:
- (the article I wrote about set up on medium)
- Github README.md
- https://github.com/strongloop/loopback

Data collection – Nodejs API framework Loopback

Add a model (table), property (field) and property_type(data_type)

`$ slc loopback:model (here I will give your sql schema, normally you can check with api manual )`

```
Setting the project root at: /Users/chloe/Desktop/dataplayground/workshop2019
? Enter the model name: order
? Select the data-source to attach order to: db (memory)
? Select model's base class PersistedModel
? Expose order via the REST API? Yes
? Custom plural form (used to build REST URL): order
? Common model or server only? common
Let's add some order properties now.

Enter an empty property name when done.
? Property name: order_no
   invoke   loopback:property
? Property type: string
? Required? Yes
? Default value[leave blank for none]:

Let's add another order property.
Enter an empty property name when done.
? Property name:
```

`common/models you will get json and js file`

Data collection – Nodejs API framework Loopback

```
$ slc loopback:relation
$ npm install --save loopback-connector-mysql
$ slc loopback:datasource
```

configure your datasource in `server/datasources.json`

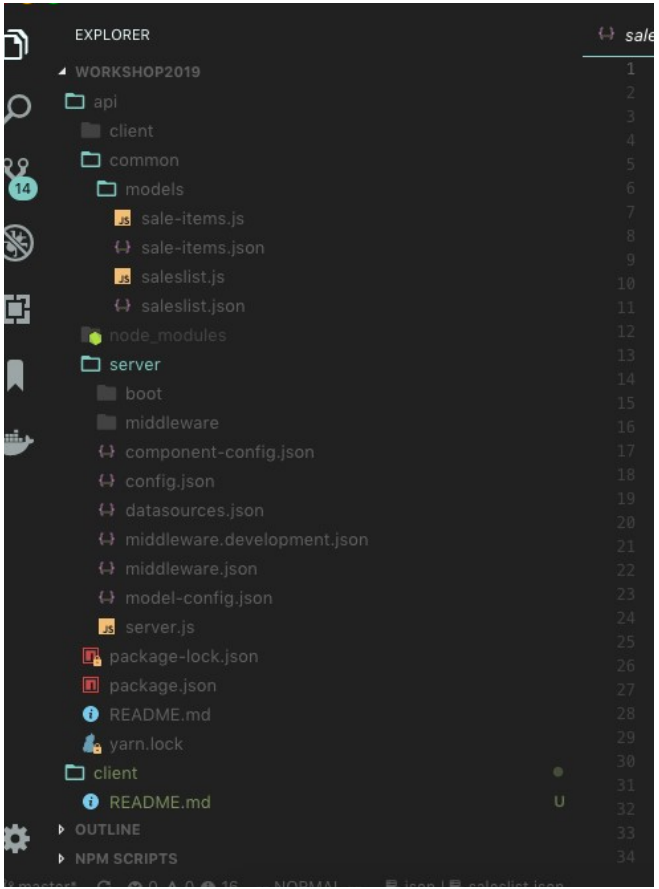
`server/model-config.json` and change the `datasource` for all entities

$node . => to open the Explore

Data collection – Nodejs API framework Loopback

```
You building env should look like
this
```

Package.json: standard npm package specification
Server.js: main application program file
Config.json: application setting
Datasource.json: datasource configuration file
Model-config.json
Middleware.json
/boot directory: add acripts to perform initialization and set up
/model directory



Data collection – Nodejs API framework Loopback

```
Config the datasource file with your localdb
$ yard add loopback-connector-mysql
```

datasources.json

```
{
  "workshop2019": {
    "host": "127.0.0.1",
    "port": 3306,
    "url": "",
    "database": "workshop2019",
    "password": "password",
    "name": "workshop2019",
    "user": "root",
    "connector": "mysql"
  }
}
```

Data collection – Nodejs API framework Loopback

```
$ slc loopback:relation
$ npm install --save loopback-connector-mysql
$ slc loopback:datasource
```

configure your datasource in `server/datasources.json`

`server/model-config.json` and change the `datasource` for all entities

$node . => to open the Explore

PySpark : Data Processing

Demo!

Install Spark on Mac

- First download Java
  note: please don't install Java version more than 8, which will created much bugs later such as java.lang.IllegalArgumentException, Unsupported class file major version 55, so better install from Java SE Development Kit 8 site and choose your os system and config.
- Go to the Apache Spark website
- Choose a Spark release and directly download
- Go to your home directory (command in bold below)
  `$ cd ~`
- Unzip the folder in your home directory using the following command

  `$ tar -zxvf spark-2.4.0-bin-hadoop2.7.tgz`
- Use the following command to see that you have a .bash_profile

  `$ ls -a`
- Config Spark to edit .bash_profile

PySpark : Data Processing

Demo!

```
$ vim .bash_profile
export SPARK_PATH=~/spark-2.4.0-bin-hadoop2.7 export PYSPARK_DRIVER_PYTHON="jupyter" export
PYSPARK_DRIVER_PYTHON_OPTS="notebook" #For python 3, have to add the line below or will get an error export
PYSPARK_PYTHON=python3 alias jupyter_notebook='$SPARK_PATH/bin/pyspark --master local[2]'
```

```
$ source .bash_profile
```

- then run the code to check if the pyspark installed

```
$ jupyter_notebook
```

- open jupyter notebook from command line

```
$ cd spark-2.4.0-bin-hadoop2.7 $ bin/pyspark
```

what’s workshop for?
- Learning from audience you :)
- Make some fun and friends to make strong community!