

XebiCon'18

Build the future

Des notebooks pour le monitoring avec Zeppelin

Romain Sagean, Développeur
Xebia

Qui suis-je ?



Scauglog (Romain Sagean)

Développeur

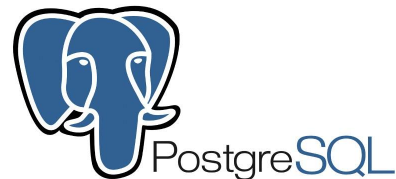
Xebia

 @scauglog

Let's Monitor

the performance of your machine learning model

Monitor?



- once a day
- mean errors
- sell
- data are on HDFS, SQL, Cassandra
- reality and prediction are in separate file
- result are for dev and business



Why not Jupyter?

Jupyter?

- notebook
- language oriented
- no database connection
- **can't hide code**
- **no security out of the box**
- no auto refresh
- doesn't free resource after use



What about kibana?

ELK?

- near real time monitoring
- too much stuff to install
- need creation of pipeline to feed the dashboard
- not enough flexibility



Zeppelin???

Zeppelin?

- big data notebook
- apache project
- written in Java (nobody is perfect)
- Backed by Hortonworks Engineer
- v0.8.0
- shipped with HDP
- designed for Big Data



Apache Zeppelin

There is an interpreter for that.

- Spark Scala, Python, R
- JDBC
- Markdown
- Shell
- Angular
- Many More
- Create Your own interpreter

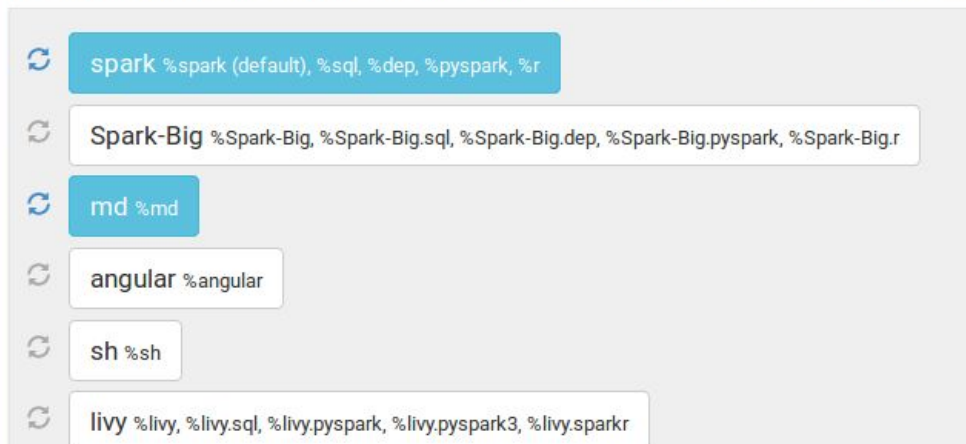


Interpreter

- Auto shutdown
- shared, scoped, isolated
- spark context is shared among language
- Configure interpreter
 - library
 - multiple spark conf

Interpreter binding

Bind interpreter for this note. Click to Bind/Unbind interpreter. Drag and drop to reorder interpreters. The first interpreter on the list becomes default. To create/remove interpreters, go to [Interprete](#)



Dashboarding

1. mix language

- JDBC
- Scala
- Cassandra

The screenshot shows the Zeppelin Notebook interface. At the top is a blue header with the Zeppelin logo, 'Notebook' and 'Job' dropdowns, a search bar, and a user indicator 'anonymous'. Below the header is a toolbar with icons for running, saving, and other actions. The main content area is titled 'rain model analysis' and shows a sequence of four code blocks, each with a 'FINISHED' status and control icons.

rain model analysis FINISHED

Took 2 sec. Last updated by anonymous at September 12 2018, 10:56:36 AM.

```
%spark
import org.apache.spark.sql.functions._
val real = spark.read.option("header", true).csv("/home/sagean/Documents/rain.csv").groupBy(col("year")).agg(sum("rain"))
```

FINISHED

```
%spark
z.show(real)
```

FINISHED

```
%cassandra
select * from corporate360.groupe_affaire_metrrique_annuelle;
```

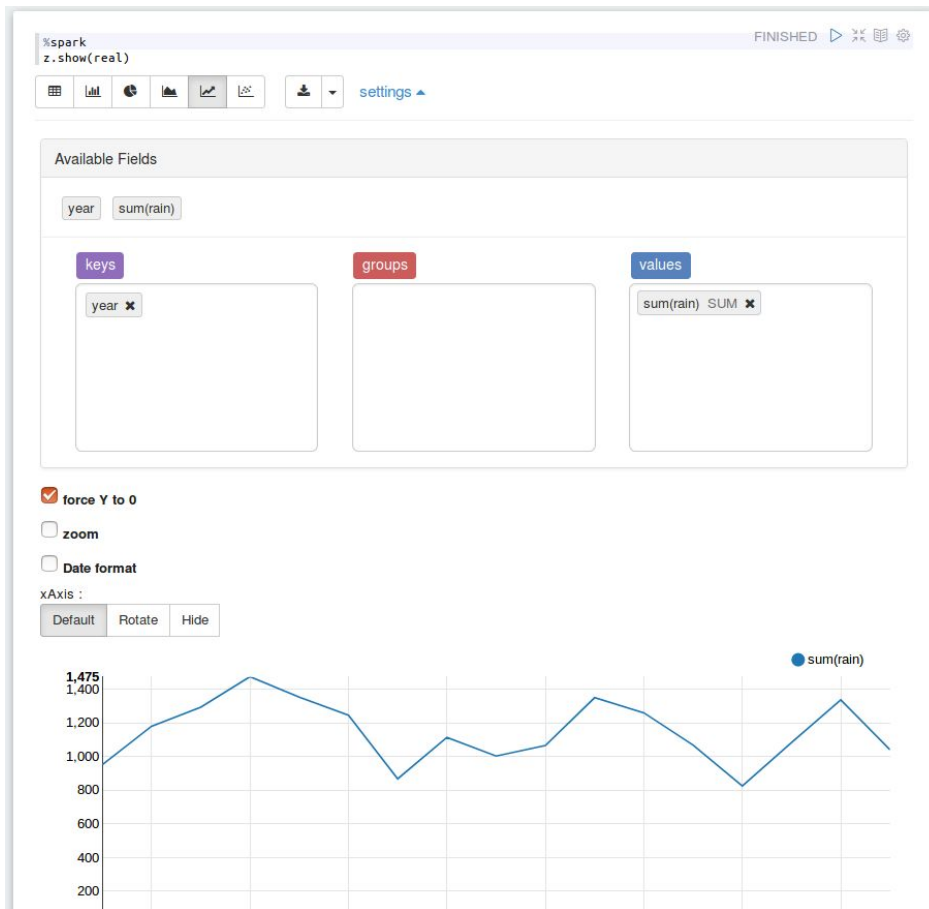
FINISHED

```
%jdbc
select * from prediction
```

FINISHED

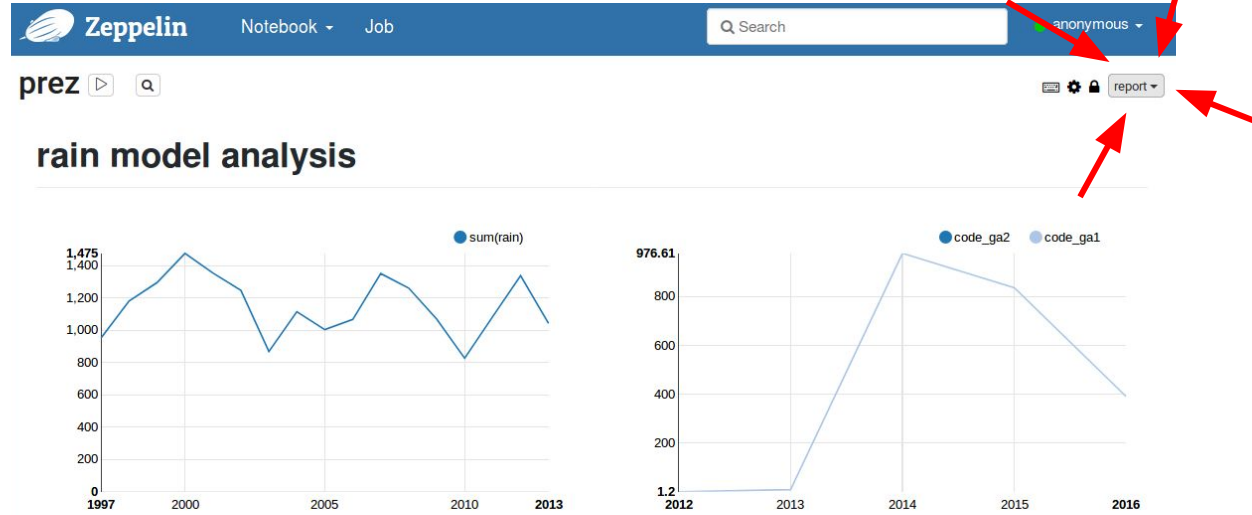
Dashboarding

1. mix language
2. **choose your dataviz**
 - table
 - bar
 - line
 - create your own dataviz



Dashboarding

1. mix language
2. choose your dataviz
3. **hide code**



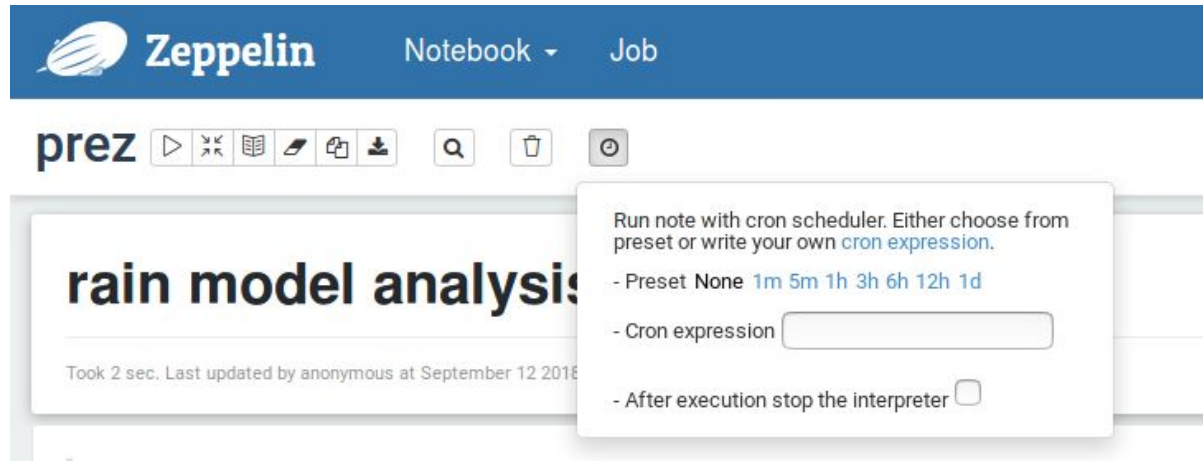
Dashboarding

1. mix language
2. choose your dataviz
3. hide code
4. **create form**



Production

1. schedule



The screenshot displays the Zeppelin Notebook interface. At the top, a blue header bar contains the Zeppelin logo, the text 'Notebook', and a 'Job' button. Below the header, the word 'prez' is followed by a series of icons for notebook actions. The main content area shows a notebook titled 'rain model analysis'. A modal dialog box is open, titled 'Run note with cron scheduler. Either choose from preset or write your own cron expression.' It contains three options: 'Preset None' with links for '1m', '5m', '1h', '3h', '6h', '12h', and '1d'; 'Cron expression' with a text input field; and 'After execution stop the interpreter' with an unchecked checkbox.

Zeppelin Notebook Job

prez

rain model analysis

Took 2 sec. Last updated by anonymous at September 12 2018

Run note with cron scheduler. Either choose from preset or write your own [cron expression](#).

- Preset None [1m](#) [5m](#) [1h](#) [3h](#) [6h](#) [12h](#) [1d](#)
- Cron expression
- After execution stop the interpreter ☐

Production

1. schedule
2. REST API

The screenshot displays a REST client interface with the following components:

- Method and URL:** A dropdown menu shows 'GET' and the URL is 'http://localhost:9995/api/notebook/export/2DAFX7NP4'.
- Buttons:** 'Params', 'Send' (highlighted in blue), and 'Save' buttons are visible.
- Tabs:** 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests' are present. 'Authorization' is the active tab.
- Authorization Section:**
 - TYPE:** A dropdown menu shows 'Inherit auth from parent'.
 - Description:** 'The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)'
 - Message:** 'This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.'
- Response Section:**
 - Tabs:** 'Body', 'Cookies', 'Headers (10)', and 'Test Results'. 'Body' is the active tab.
 - Status:** 'Status: 200 OK', 'Time: 84 ms', 'Size: 112.18 KB'.
 - Body Format:** A dropdown menu shows 'JSON'.
 - Body Content:** A JSON response is displayed in a code editor with syntax highlighting. The response includes fields like 'status', 'message', and 'body'.

Production

1. schedule
2. REST API
3. Secure
 - login



Login


User Name

Password

Login

Production

1. schedule
2. REST API
3. Secure
 - login
 - **manage interpreter credential**

 **Zeppelin** Notebook ▾ Job anonymous ▾

Credentials

? + Add

Manage your credentials. You can add new credential information.

Add new credential

Entity	Username	Password
<input type="text" value="[Interpreter Group].[Interpreter Name]"/>	<input type="text"/>	<input type="text"/>

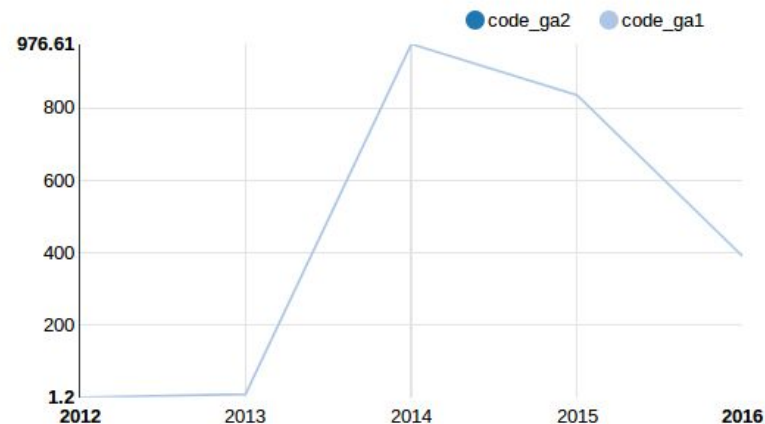
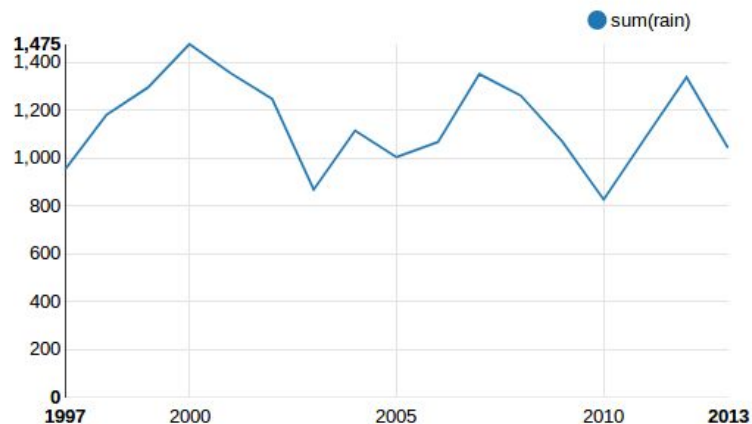
Save Cancel

Currently there is no credential information

prez ▶ 🔍

🔧 🔒 report ▾

rain model analysis



Thank you

Any questions?