

# Data science lifecycle with Apache Flink and Apache Zeppelin (incubating)



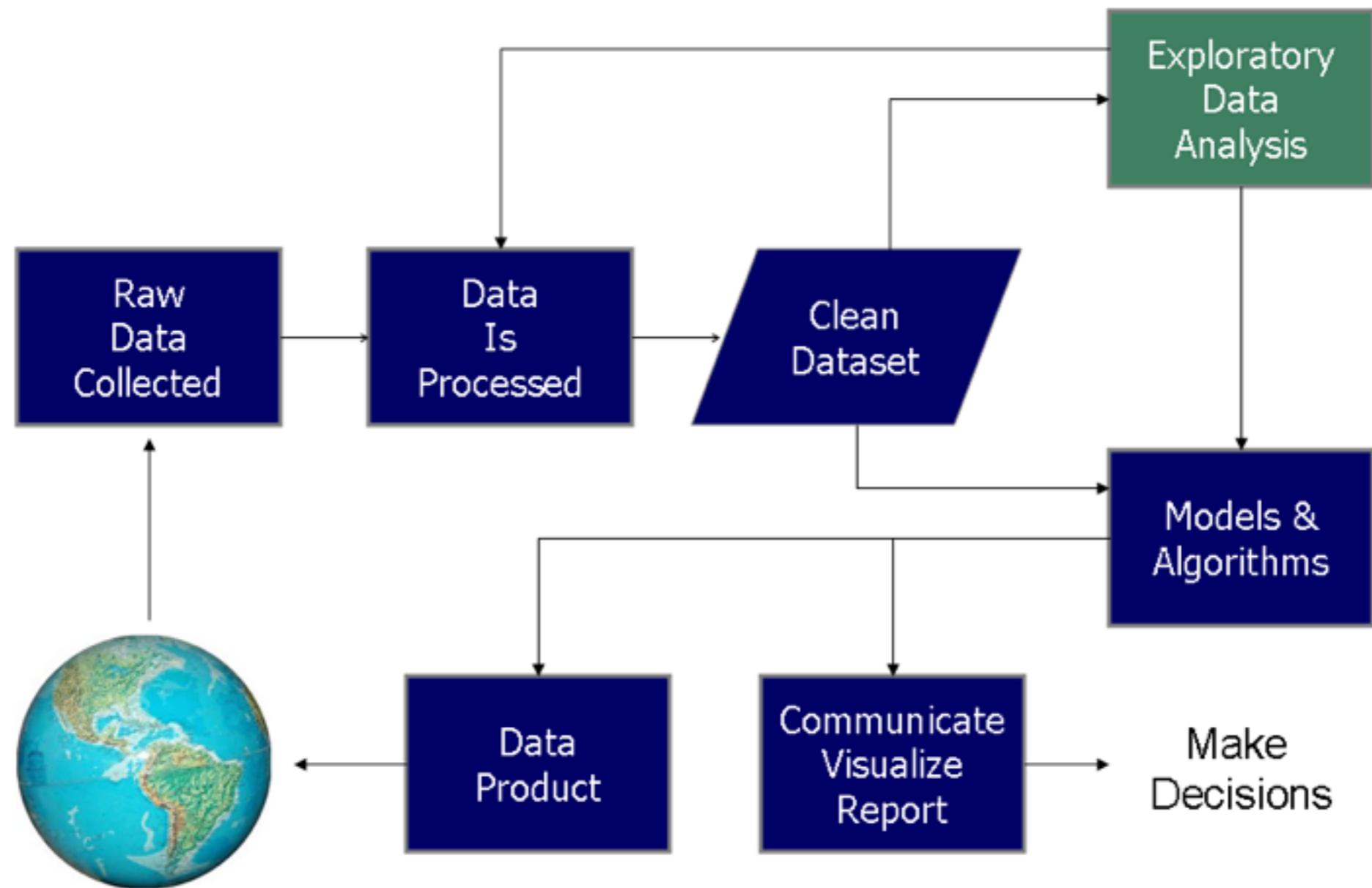
Moon [moon@nflabs.com](mailto:moon@nflabs.com)  
NFLabs [www.nflabs.com](http://www.nflabs.com)

# Content

1. **Data science lifecycle**
2. Zeppelin for data science
3. Zeppelin and Flink
4. Project Roadmap

# Data science lifecycle

# Data Science: process



# Data Science: tools



# Data Science: people



**Engineer**

**Data Scientist**



**DevOps**

**Business**

# Content

1. Data science lifecycle
- 2. Zeppelin for data science**
3. Zeppelin and Flink
4. Project Roadmap

# Zeppelin for data scientist

# Project Timeline



<http://zeppelin.incubator.apache.org>

- 12.2012    **Commercial Product for data analysis**
- 10.2013    **Open sourced a single feature**
- 08.2014    **Started getting adoption**
- 12.2014    **ASF Incubation**

# Hadoop Landscape

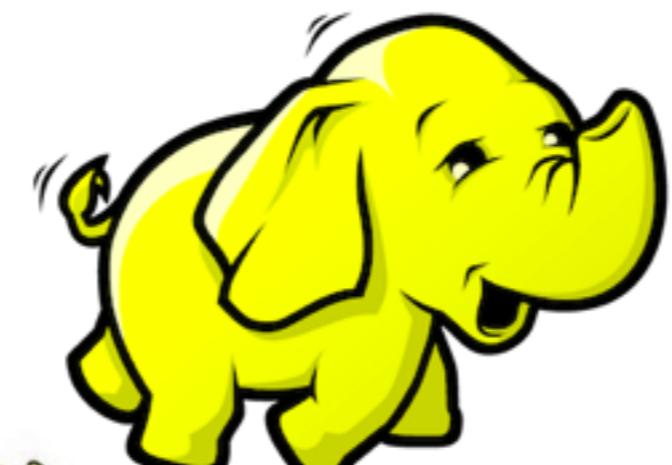
ML-base



MRQL



Cloudera-ML



Shark



# Commercial Product

12.2012



# Zeppelin

10.2013

Zeppelin ZQL ZAN

## Import bank marketing data

History [Latest](#)

```
14     housing STRING,  
15     loan STRING,  
16     contract STRING,  
17     day INT,  
18     month INT,  
19     duration INT,  
20     campain INT,  
21     pdays INT,  
22     previous INT,  
23     poutcome STRING,  
24     y STRING  
25 )  
26 ROW FORMAT DELIMITED  
27 FIELDS TERMINATED BY "59"  
28 STORED AS TEXTFILE  
29 ;  
30  
31 LOAD DATA LOCAL INPATH '/tmp/bank-full.csv' OVERWRITE INTO TABLE bank;
```

Run

Schedule - None | Share | Delete

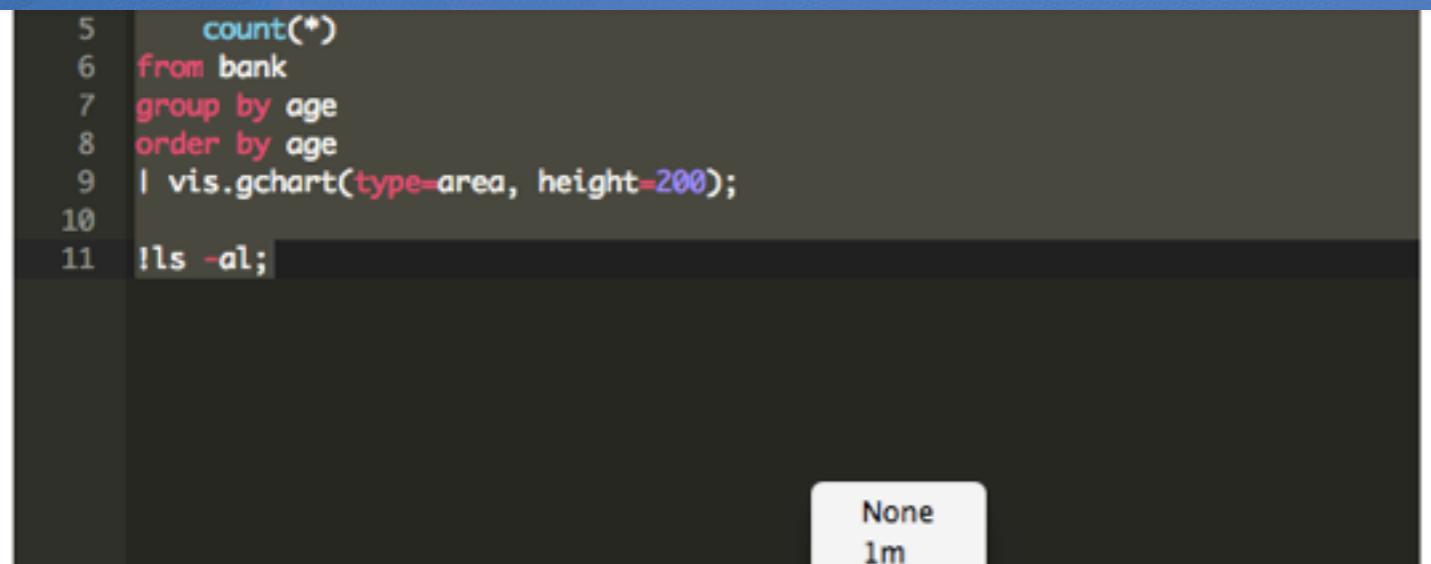
Last run Feb 5, 2014 10:25:53 AM, took 14.0 seconds

curl -s http://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.zip -o

# Zeppelin

10.2013

```
5   count(*)  
6   from bank  
7   group by age  
8   order by age  
9   | vis.gchart(type=area, height=200);  
10  
11 !ls -al;
```



Run

```
echo 'hello world';
```

```
hello world
```

vis.gchart

- Table
- Area
- Bar
- Bubble
- Candlestick
- Column
- Gauge
- Geo
- Line
- Map
- Pie
- Scatter



# Zeppelin

08.2014

The screenshot shows the Zeppelin web interface with several components:

- Scala REPL:** A code editor window containing Scala code for reading a CSV file and registering it as a table.

```
val textData = sc.textFile("/Users/moon/Projects/zeppelin/bank-full.csv")
val bank = textData.map(s=>s.split(",")).map(
    s=>Person(s(0).toInt, s(1).replaceAll("\"", ""), s(2).replaceAll("\"", ""), s(3).replaceAll("\"", ""), s(5).t
bank.registerAsTable("bank")
```

Output:

```
defined class Person
textData: org.apache.spark.rdd.RDD[String] = MappedRDD[1] at textFile at :15
bank: org.apache.spark.rdd.RDD[Person] = MappedRDD[3] at map at :19
```

- Parameterized Query:** A code editor window with a parameter `maxAge` set to 40, followed by a SQL query.

```
%sql select age, avg(balance), min(balance) from bank where age < ${maxAge=50} group by age order by age
```

- Chart:** A line chart visualizing the data from the query. The x-axis represents age (22 to 47) and the y-axis represents balance (from -8,019.0 to 1,459.0). The chart shows a general downward trend in average balance as age increases, with significant outliers at younger ages.

Markdown supported  
scala REPL.  
load some data and register  
as a table  
and we can do some query  
oops bug  
let me try again  
okay.

Parameterized query?  
yes!

# Zeppelin

08.2014



**Zeppelin** Notebook - Connected

## Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.  
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

**Notebook**

- [Create new note](#)
- [Data discovery and export](#)
- [Demo](#)
- [Export data to Peloton](#)
- [GameLog](#)
- [Gamelog view](#)
- [KISTI Web performance evaluation](#)
- [Netflix](#)
- [Plan](#)
- [Read data from Peloton](#)
- [Twitter Stream example](#)
- [Welcome to Zeppelin](#)

**Help**

Get started with [Zeppelin documentation](#)

**Community**

Please feel free to help us to improve Zeppelin,  
Any contribution are welcome!

- [Mailing list](#)
- [Issues tracking](#)
- [Github](#)

# Third-party Products

10.2014

localhost:8080/#/notebook/2A74MDXJC

STRATIO CROSSDATA Notebook Note 2A74MDXJC

DATALAYER Notebook /archive/icc.vcl.estiml/machine-learning-database/10

Load and analyse simple CSV Data

1. You will need this [file](#).  
2. You will need to change the path accordingly

```
case class Bank(age:Int, job:String, marital : String, education : String, balance : Integer)
Connected
```

```
val bank = bankText.map(s=>s.split(";")).filter(s=>s(0)!="age").map{
    s=>Bank(s(0).toInt,
        s(1).replaceAll("\\\\", ""),
        s(2).replaceAll("\\\\", ""),
        s(3).replaceAll("\\\\", ""),
        s(5).replaceAll("\\\\", "").toInt
    )
}
bank.take(10).foreach(x => println(x))
```

%sqlb select age, age+3 from bank limit 100

FINISHED

SETTINGS

Grouped Stacked

The chart displays the count of bank records for different age groups. The x-axis shows age ranges from 33 to 58, and the y-axis shows the count from 0.0 to 480.0. The counts generally increase with age, with a notable peak around age 58.

%sqlb select age, count(1) from bank where age <= 30 group by age order by count(1) desc

FINISHED

SETTINGS

18 19 20 21 22 23 24 25 26 27 28 29 30

A pie chart showing the distribution of ages for bank records up to age 30. The slices represent the percentage of records for each age group. The largest slice is for age 30, followed by 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, and 18.

%sqlb select age, count(1) from bank where age <= \${maxAge=30} group by age

FINISHED

maxAge 20

%sqlb select age, count(1) from bank where marital="\${marital=single,single}"

FINISHED

marital single

# Apache Incubation Proposal

11.2014

Incubator Wiki [Login](#)

[ZeppelinProposal](#)

Search

Titles

Text

[FrontPage](#) [RecentChanges](#) [FindPage](#) [HelpContents](#) [ZeppelinProposal](#)

[Immutable Page](#) [Info](#) [Attachments](#)

More Actions:



## Abstract

Zeppelin is a collaborative data analytics and visualization tool for distributed, general-purpose data processing systems such as Apache Spark, Apache Flink, etc.

## Proposal

Zeppelin is a modern web-based tool for the data scientists to collaborate over large-scale data exploration and visualization projects. It is a notebook style interpreter that enable collaborative analysis sessions sharing between users. Zeppelin is independent of the execution framework itself. Current version runs on top of Apache Spark but it has pluggable interpreter APIs to support other data processing systems. More execution frameworks could be added at a later date i.e Apache Flink, Crunch as well as SQL-like backends such as Hive, Tajo, MRQL.

We have a strong preference for the project to be called Zeppelin. In case that may not be feasible, alternative names could be: "Mir", "Yuga" or "Sora".

## Background

Large scale data analysis workflow includes multiple steps like data acquisition, pre-processing, visualization, etc and may include inter-operation of multiple different tools and technologies. With the widespread of the open source general-purpose data processing systems like Spark there is a lack of open source, modern user-friendly tools that combine strengths of interpreted language for data analysis with new in-browser visualization libraries and collaborative capabilities.

Zeppelin initially started as a GUI tool for diverse set of SQL-over-Hadoop systems like Hive, Presto, Shark, etc. It was open source since its inception in Sep 2013. Later, it became clear that there was a need for a greater web-based tool for data scientists to collaborate on data exploration over the large-scale projects, not limited to SQL. So Zeppelin integrated full support of Apache Spark while adding a collaborative environment with the ability to run and share interpreter sessions in-browser

## Rationale

# Acceptance by Incubator

23.12.2014



# Current Status

**68 Contributors worldwide**

**722 Stars on GH**

**300/900 Emails at users/dev @i.a.o**

**I Release**

# Interactive Notebooks

### Zeppelin Notebook - Interpreter

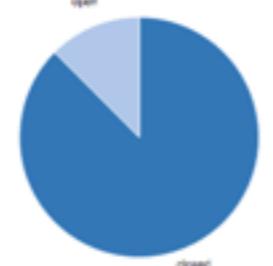


### Zeppelin Notebook - Interpreter

Connected

16 pull requests    8 Contributors    74 comments    4972 additions

closed    open



number	state	login	commits	comments	additions
20	closed	jongyoul	4	5	34
21	closed	jongyoul	1	1	0
22	closed	RamVenkatesh	4	5	42
23	closed	jongyoul	1	3	24
24	closed	langley	1	1	71
25	closed	jongyoul	2	4	8
26	closed	Leemoonsaoo	1	1	39
27	closed	Leemoonsaoo	21	24	4,676
30	open	bzz	1	2	1
99	missed	innovativedata	1	1	4

### Zeppelin Notebook - Interpreter

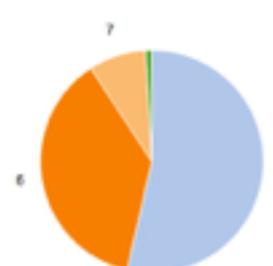
Disconnected

Task 2 seconds.

```
println("Notebook magnticcount")
earthQuake.map(s->Cs(5).toString.toInt, 1)).reduceByKey(_ + _).collect.foreach(s->println(s._1))
```

settings ▾

4 5 6 7 8 9



number	state	login	avatar	created_at	value
20	closed	jongyoul		2015-04-01T01:50:36Z	4
21	closed	jongyoul		2015-04-01T01:57:09Z	2
22	closed	RamVenkatesh		2015-04-01T06:09:37Z	2
23	closed	jongyoul		2015-04-01T07:32:02Z	2
24	closed	langley		2015-04-02T01:16:03Z	2
25	closed	jongyoul		2015-04-02T13:59:30Z	2
26	closed	Leemoonsaoo		2015-04-03T09:02:04Z	2
27	closed	Leemoonsaoo		2015-04-06T10:48:45Z	2
28	closed	bzz		2015-04-07T07:39:56Z	2
29	closed	corneadoug		2015-04-07T08:04:11Z	2
30	open	bzz		2015-04-07T09:02:26Z	0
31	closed	RamVenkatesh		2015-04-07T14:04:22Z	2015-04-16T18:02:50Z
32	closed	syspec		2015-04-08T21:31:19Z	2015-04-11T09:48:49Z

Task 1 seconds.

### Zeppelin Notebook - Interpreter

Disconnected

depth



### Zeppelin Notebook - Interpreter

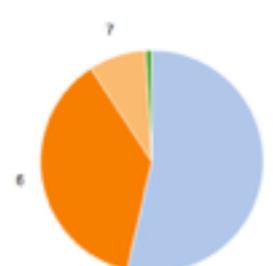
FINISHED

Task 2 seconds.

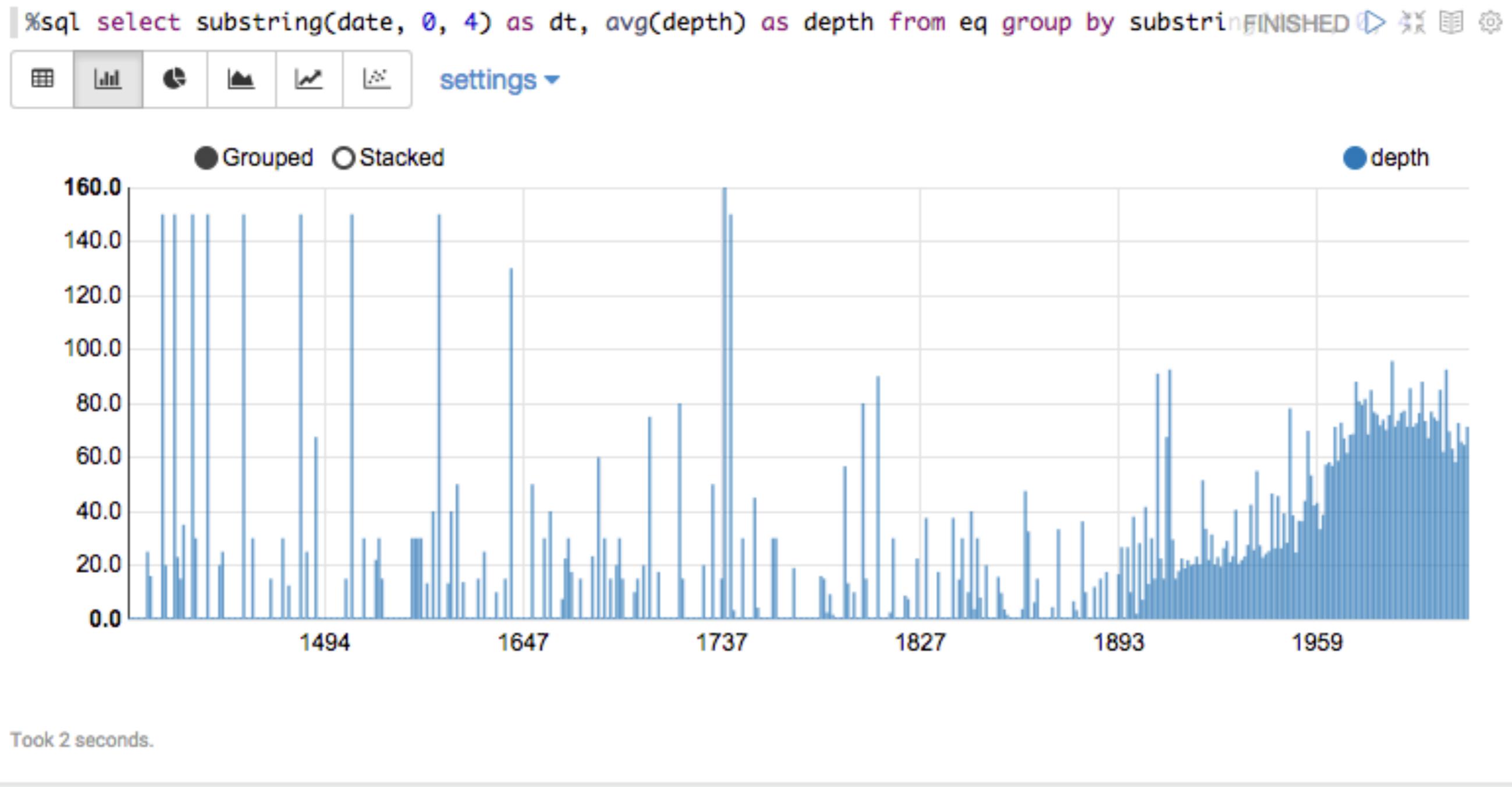
```
println("Notebook magnticcount")
earthQuake.map(s->Cs(5).toString.toInt, 1)).reduceByKey(_ + _).collect.foreach(s->println(s._1))
```

settings ▾

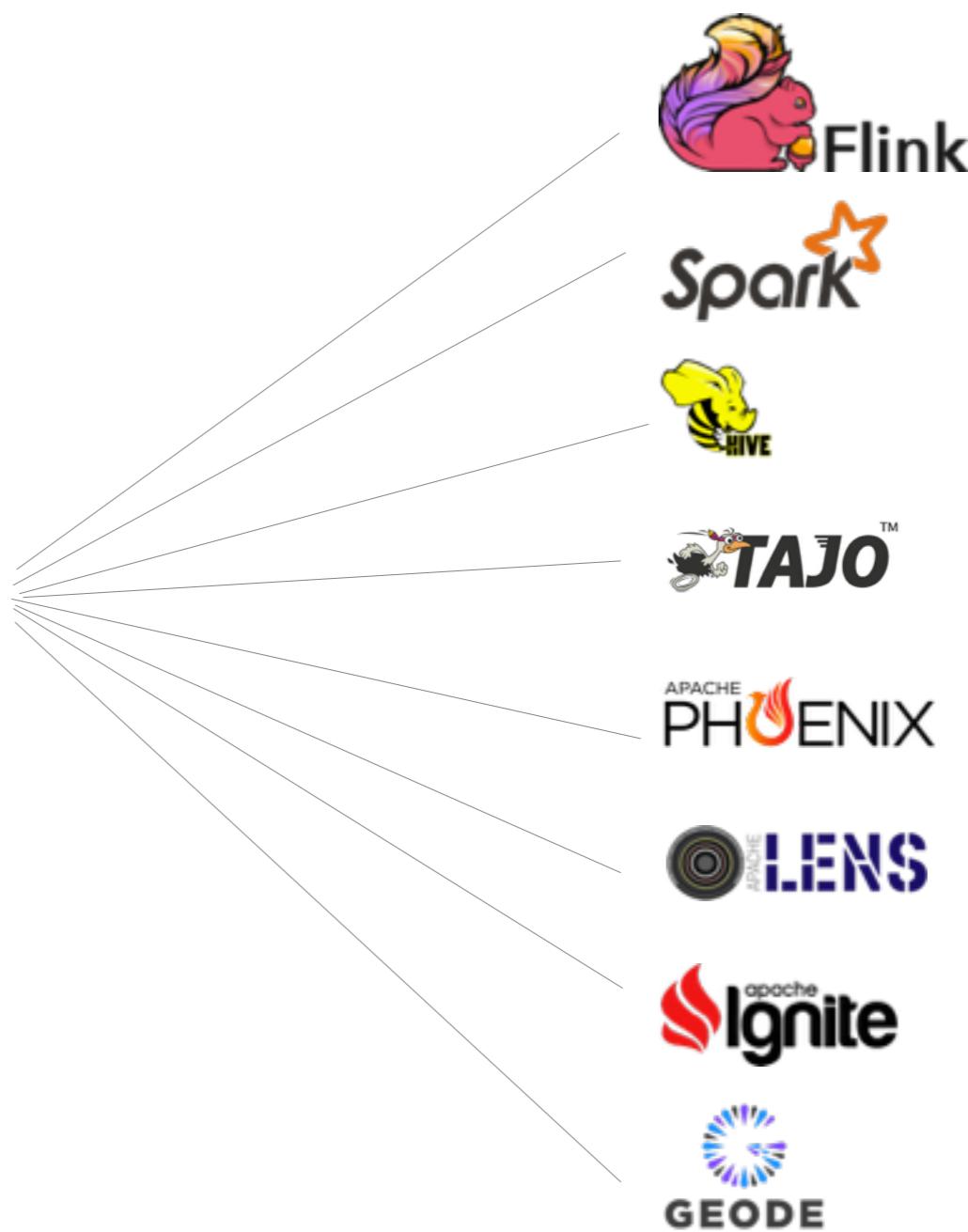
4 5 6 7 8 9



# Interactive Visualization



# Multiple Backends



# Zeppelin & Friends

## Collaboration/Sharing

### Packaging & Deployment



Z-Manager

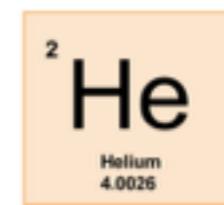


ZeppelinHub

### Zeppelin + Full stack on a cloud



### Packages



### Backend Integration



...



# Online Viewer

A screenshot of a web browser window. The address bar shows the URL <https://www.zeppelinhub.com/viewer/>. The page title is "Zeppelin Viewer". The main content area contains instructions for pasting a notebook link, a text input field for the URL, and a "view" button. Below this is a thank you message.

## Zeppelin Viewer

Paste a link to a notebook

This viewer currently supports direct urls or notebooks hosted on github and dropbox. We will support other methods in the very near future.

view

for example, [intro to Zeppelin](#)

We heard you and we want to thank you. As a gratitude to the wonderful community of Apache Zeppelin (incubating)—both users and contributors—we would like to offer this Zeppelin Viewer service. So go ahead, share your Zeppelin notebooks with anyone, anywhere. And please, share your [thoughts and feedback](#) as well. Thank you to the greatest open source community in the world

# Deployment

Ambari   Sandbox   0 ops   0 alerts   Dashboard   Services   Hosts   Alerts   Admin   admin   Connected

## Zeppelin

Notebook   Interpreter   Connected

### Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.  
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

**Notebook**

- [Create new note](#)
  - [Australian Dataset \(Hive example\)](#)
  - [Australian Dataset \(SparkSQL example\)](#)
  - [Zeppelin Tutorial](#)

**Help**

[Get started with Zeppelin documentation](#)

**Community**

Please feel free to help us to improve Zeppelin,  
Any contribution are welcome!

- [Mailing list](#)
- [Issues tracking](#)
- [Github](#)



<https://github.com/hortonworks-gallery/ambari-zeppelin-service>

# Deployment

JUJU    Sandbox    1    More    Get started 

spark 

Services Machines

Home

Added Services (2) >

Recommended (3)

 apache-spark  
Deployed 49 times  
trusty | Recommended

 apache-spark-...  
Deployed 9 times  
trusty | Recommended

(apache-ze...)   
... 

(apache-sp...) 



# As a Service



SALES 1-800-867-1389 ▾

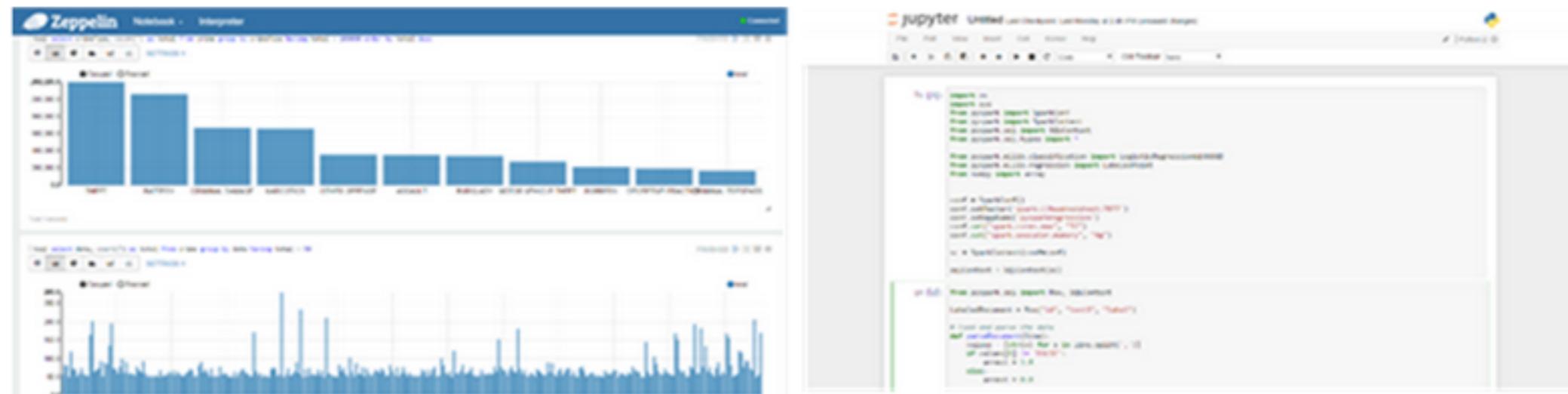
MY ACCOUNT

PORTAL

Why Azure   Products   Documentation   Pricing   Downloads   Partners   **Blog**   Community   Support

## Using open-source notebooks

Notebooks can also be used to visualize data running in Spark for Azure HDInsight. Notebooks are open source tools that give data scientists an ability to combine live code, statistical equations, narrative text, and visualizations to tell a story of their data. We have enabled popular Jupyter (IPython) and Zeppelin notebooks to run on Spark for Azure HDInsight. Jupyter will come with standard IPython visual libraries and ideal for those who code using Python. Zeppelin is ideal for those who write in Scala while also supporting Spark SQL and Markdown.



# Before

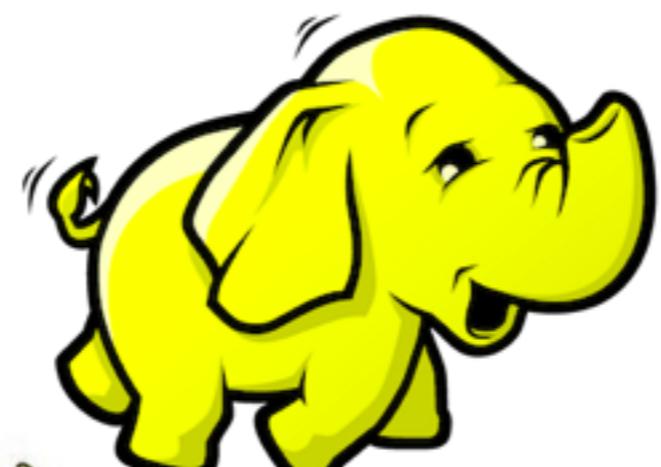
ML-base



MRQL



Cloudera-ML



Shark



# After

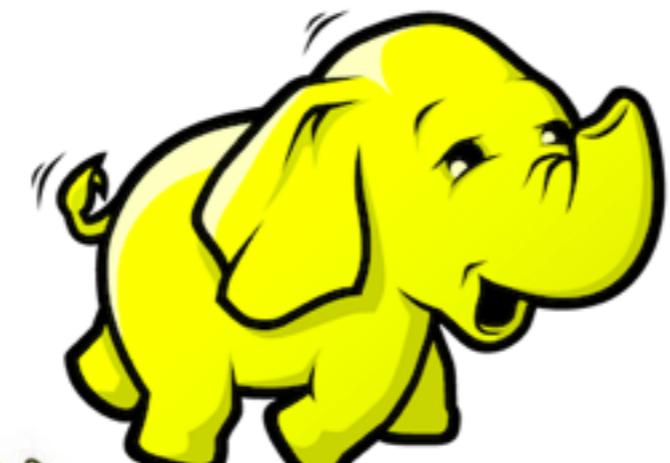
ML-base



MRQL



Cloudera-ML



Shark



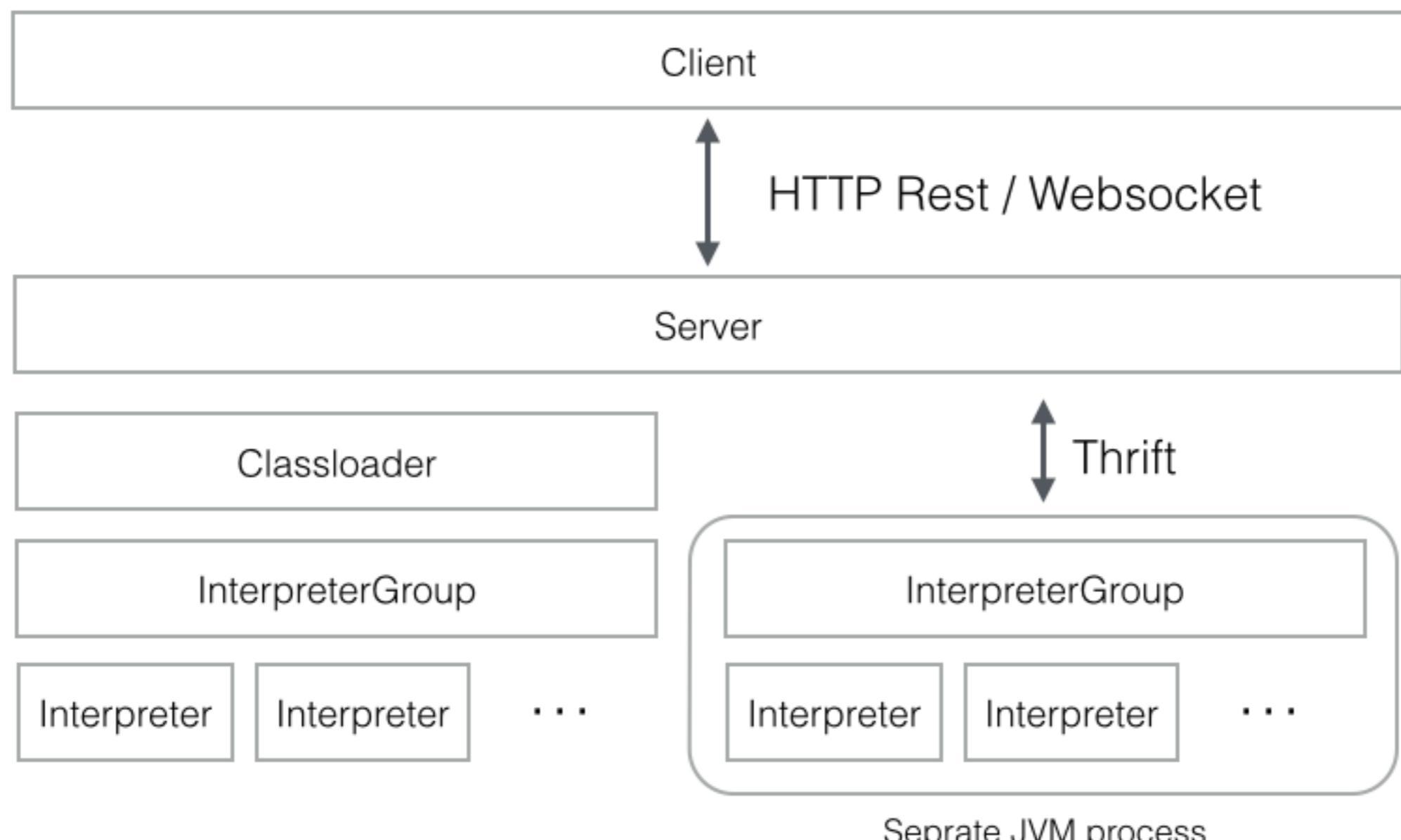
# Content

1. Data science lifecycle
2. Zeppelin for data science
- 3. Zeppelin and Flink**
4. Project Roadmap

# Flink integration

Integrated through Interpreter  
Data processing system abstraction in Zeppelin

# Interpreter



# Writing an Interpreter

public abstract void open();

Must have

public abstract void close();

public abstract InterpreterResult interpret(String st, InterpreterContext context);

public abstract void cancel(InterpreterContext context);

Good to have

public abstract int getProgress(InterpreterContext context);

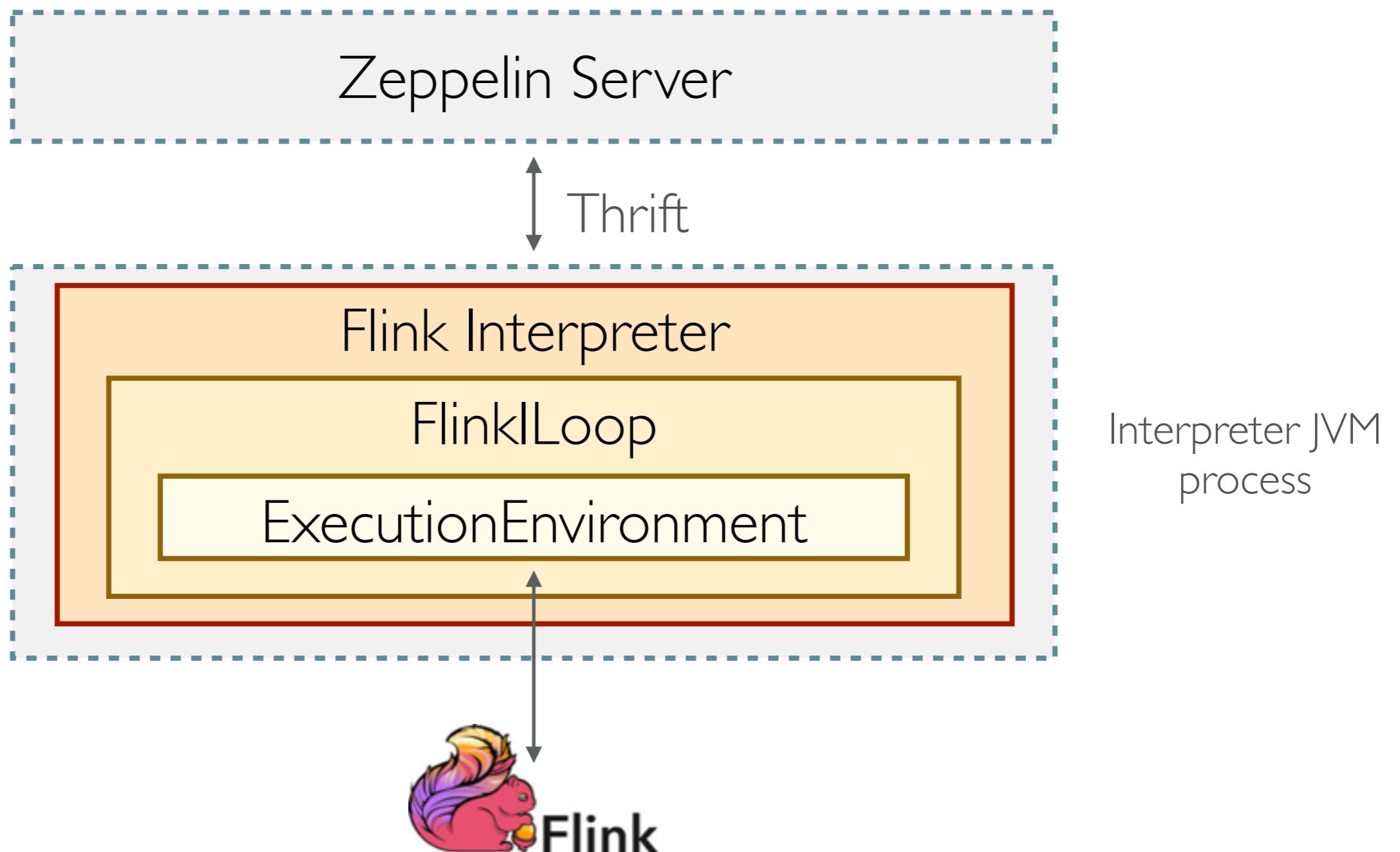
public abstract List<String> completion(String buf, int cursor);

public abstract FormType getFormType();

Advanced

public Scheduler getScheduler();

# Flink Interpreter



# Using interpreter

Configure

The screenshot shows the Zeppelin configuration interface. At the top, there's a blue header bar with the Zeppelin logo and the word "Zeppelin". Below it, a search bar contains the text "flink %flink". Underneath, there's a table titled "Properties" with the following rows:

name	value
host	local
port	6123
taskmanager.numberOfTaskSlots	2

Bind

The screenshot shows a Zeppelin notebook interface. The title bar says "Zeppelin Notebook". The main content area has the heading "FlinkForward 2015 Berlin". Below it, there's a section titled "Interpreter binding" with the sub-instruction: "Bind interpreter for this note. Click to Bind/Unbind int". A list of interpreters is shown in blue buttons: "flink %flink", "md %md", "angular %angular", and "sh %sh".

use

The screenshot shows the execution results of a Flink command. The input was "%flink println("Hello FLink")". The output is "Hello FLink" followed by the message "Took 18 seconds.".

# Using interpreter

Use different  
interpreters in the  
same notebook

The screenshot shows a Jupyter Notebook interface with three code cells. Red arrows point to the interpreter magic commands in each cell.

```
%md
## Flink Forward
- 2015
- 2016
```

## Flink Forward

- 2015
- 2016

Took 0 seconds. (outdated)

```
%flink
println("Hello FLink")
```

Hello FLink

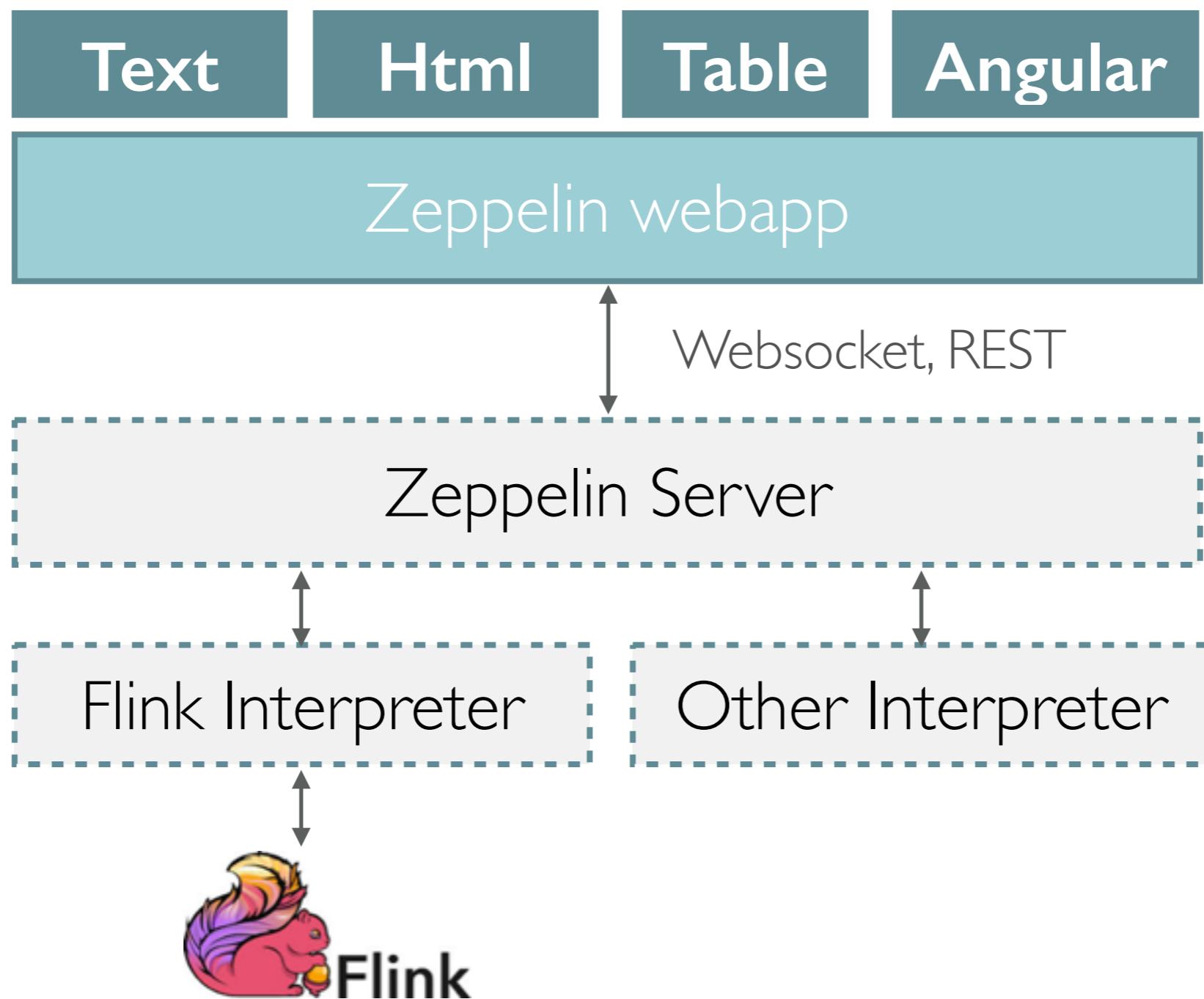
Took 18 seconds.

```
%sh hostname
```

Lees-MacBook.local

Took 1 seconds

# Display System



# Display System

Select display system through output

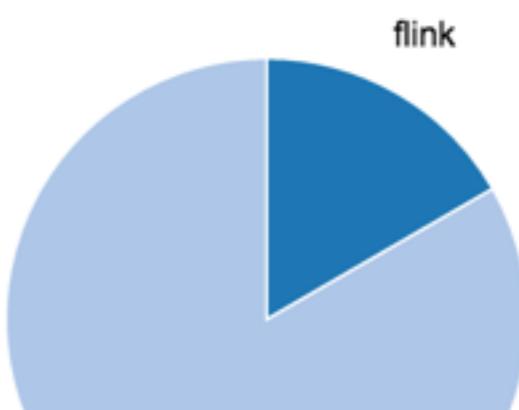
```
println("Default display system is text")
Default display system is text
Took 1 seconds.

println("%html <h4>html display system</h4>")
html display system
Took 1 seconds.

println(s"""%table key\tvalue
flink\t20
forward\t100""")

```

settings ▾



# Built in scheduler

Built-in scheduler runs your notebook with cron expression.

⌚ 3h

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

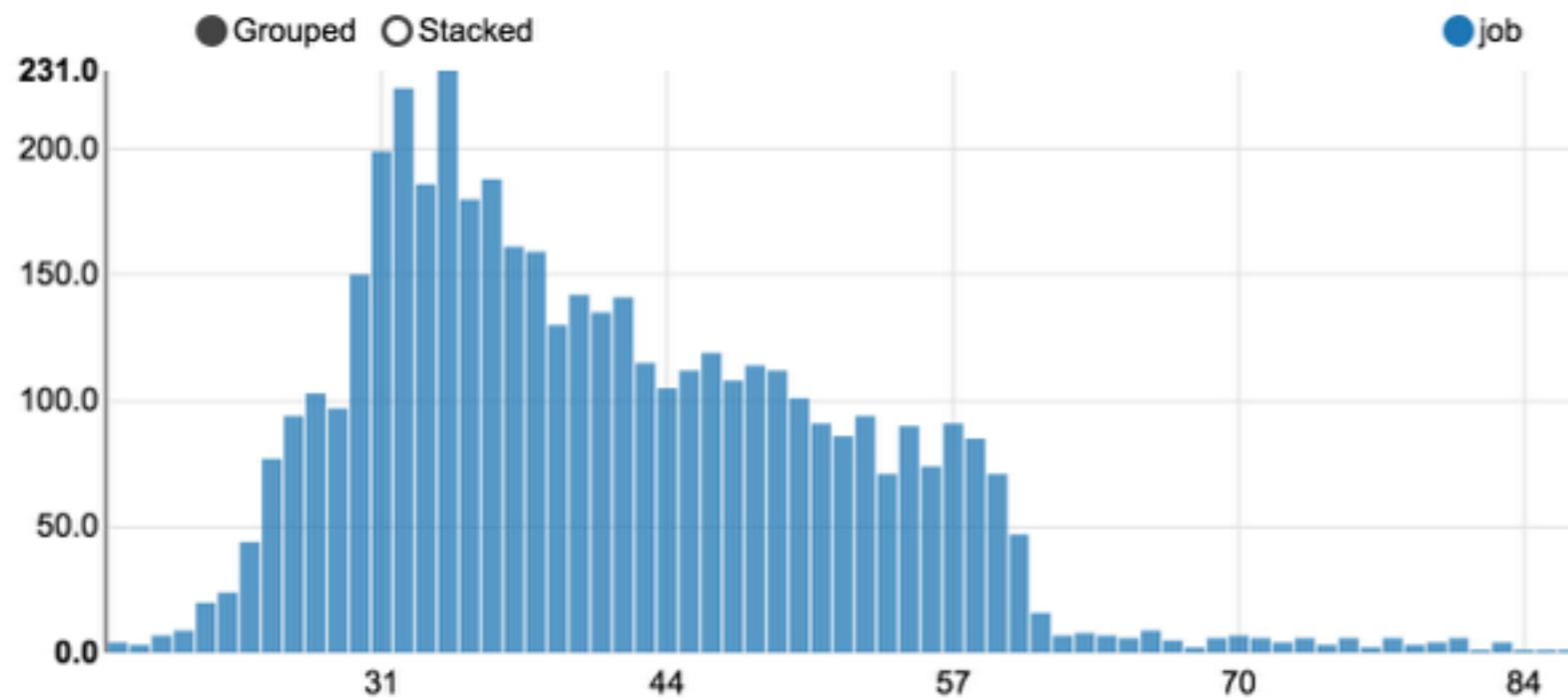
- Preset [None1m5m1h3h6h12h1d](#)
- Cron expression

# Flexible layout

## Flexible layout

```
println("%table " + Array("age", "job", "marital", "education", "balance"))
bank.collect.foreach(b=>
  println(Array(b.age, b.job, b.marital, b.education, b.balance).mkString("\t")))
```

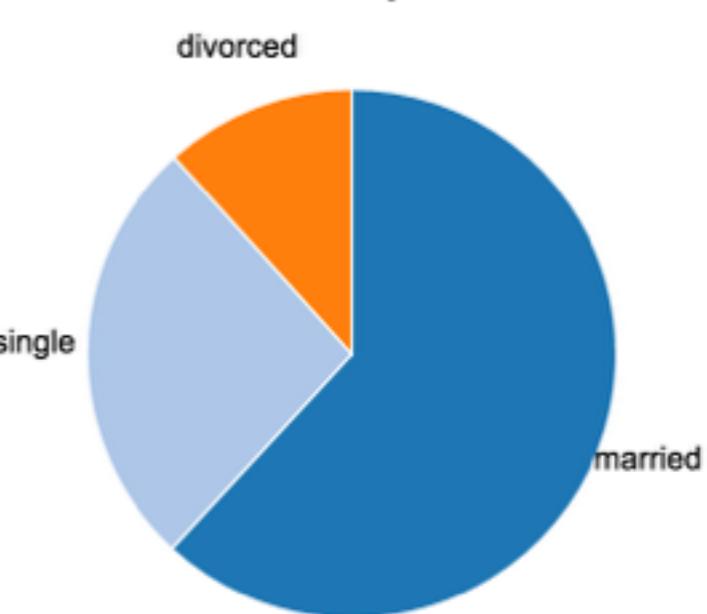
grid list pie bar line scatter settings ▾



```
println("%table " + Array("marital", "education", "balance")
.mkString("\t"))
bank.collect.foreach(b=>
```

grid list pie bar line scatter settings ▾

● married ○ single ● divorced



# DEMO

# Content

1. Data science lifecycle
2. Zeppelin for data science
3. Zeppelin and Flink
- 4. Project Roadmap**

# Flink Integration

- ZeppelinContext : Access to Zeppelin provided features
  - - Dynamic form
  - - Angular display system
- Dependency loading
- Auto completion
- Cancel
- Get progress information

# Thank you

## Q & A

<http://zeppelin.incubator.apache.org/>

Moon  
[moon@nflabs.com](mailto:moon@nflabs.com)

NFLabs  
[www.nflabs.com](http://www.nflabs.com)

# Project roadmap

# Multi-tenancy

Two approaches

- I. Implement authentication, ACL inside of Zeppelin

<https://github.com/apache/incubator-zeppelin/pull/53>

2. Run Zeppelin on top of Docker

<http://github.com/NFLabs/z-manager>

# Zeppelin for organizations

# An Engineer



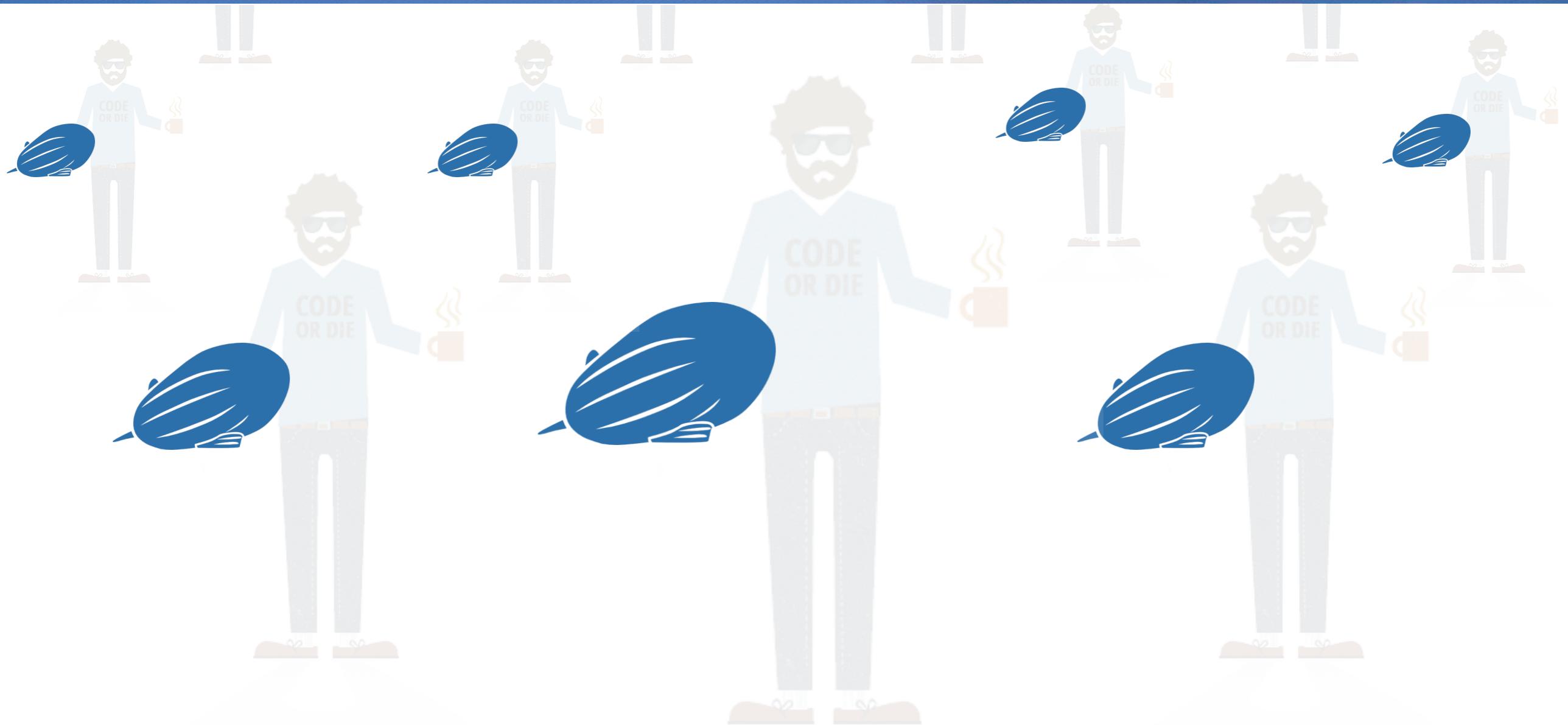
# A Team



# An Organization



# That's too many!



# What is the problem?

**Too much:**

**Install**

**Configure**

**Cluster resources**

# Solution?

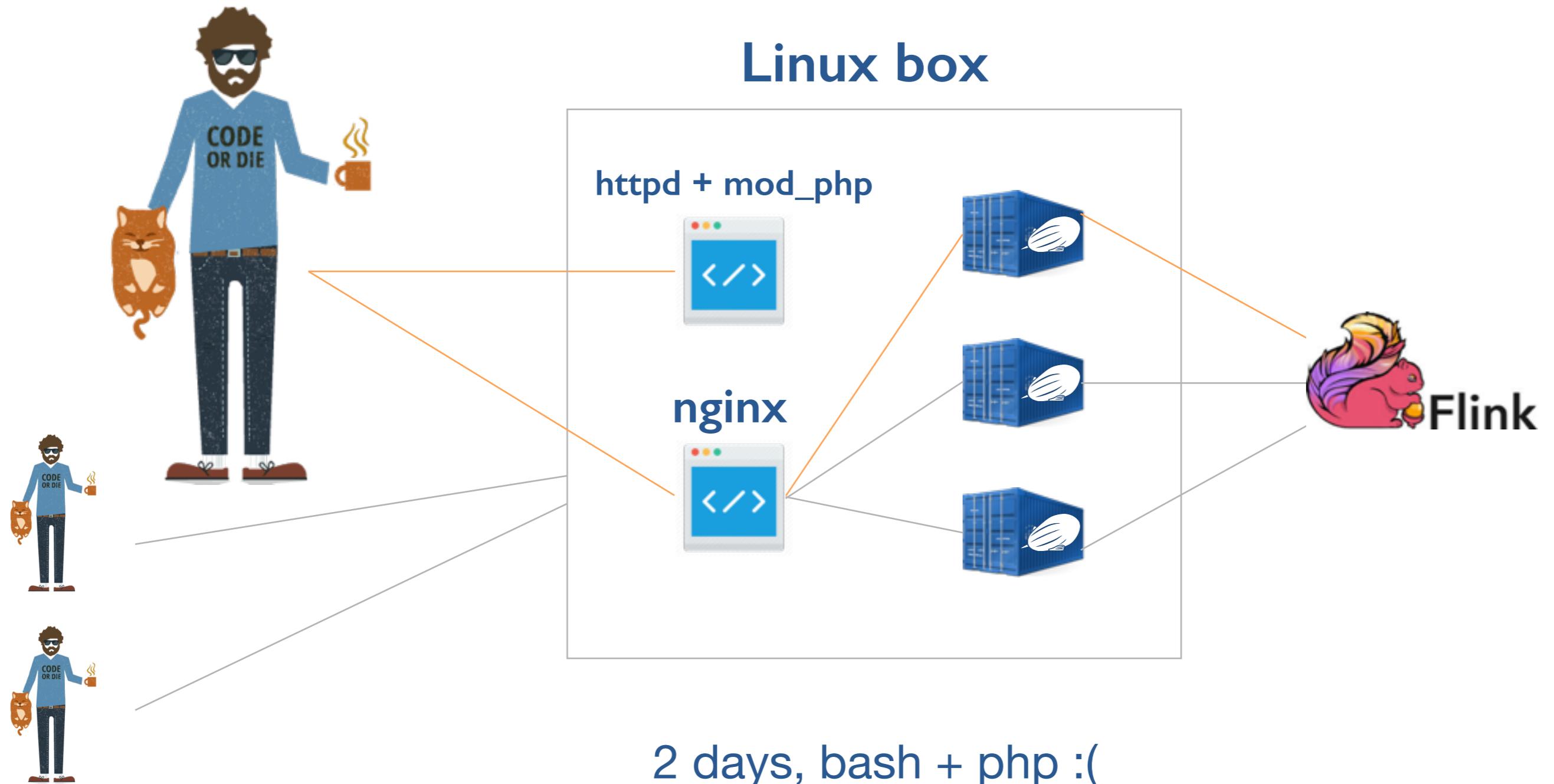
We have containers

+

reverse proxy



# Z Manager PoC



# Z Manager PoC

Please sign in

Username

Password

Remember me

Sign in

 Zeppelin instance manager Hello, moon Logout

## Cluster information

Core 209/544

209 20

Mem 1423GB/3.7TB

1423TB 0.00TB

## Start zeppelin instance

Number of cpu cores

20

Memory per worker

512 Mb Gb

Start

# Z Manager



Z Manager

<http://github.com/NFLabs/z-manager>

Single binary

Containerized deployment per user

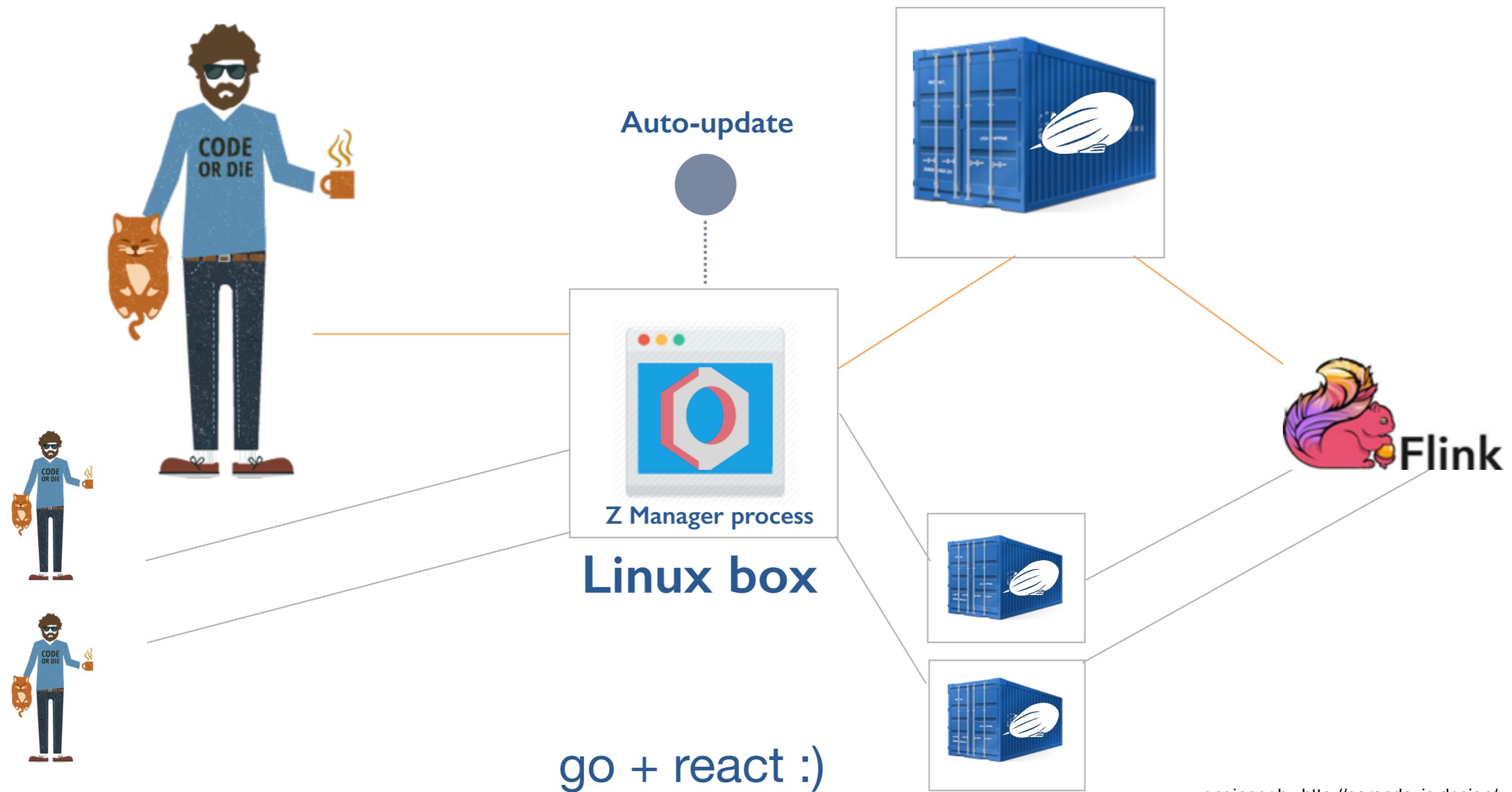
Reverse proxy

Simple web application

Apache 2.0 Licence

SGA to ASF coming \*

# Z Manager



# Z Manager

## Z-Manager

username

password

LOG IN

anthonycorbacho

LOGOUT

### Cores

Allocate	Free	Total
63	194	544

### Memory (GB)

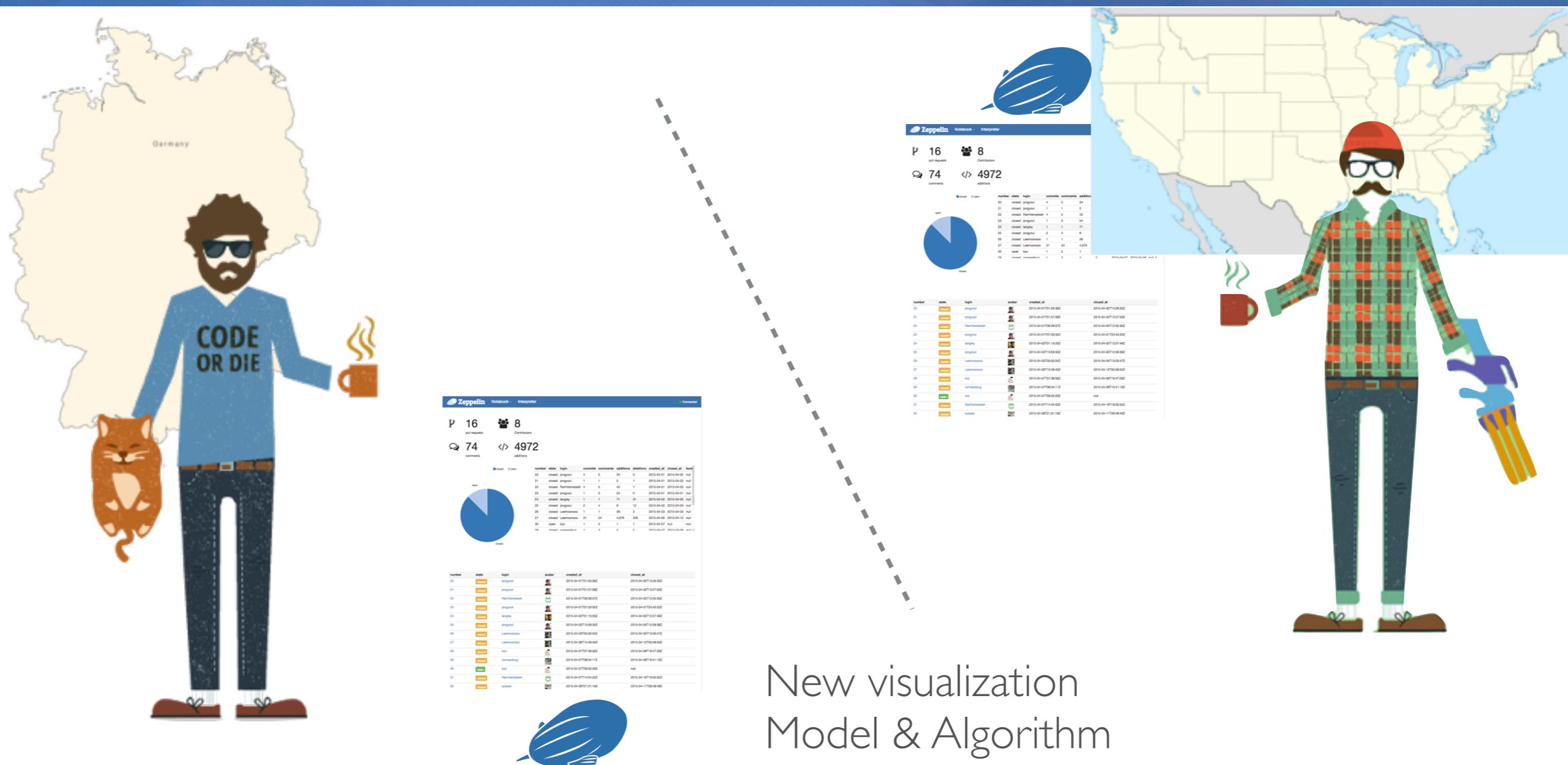
Allocate	Free	Total
600	1705.67	3808

RESET

CREATE

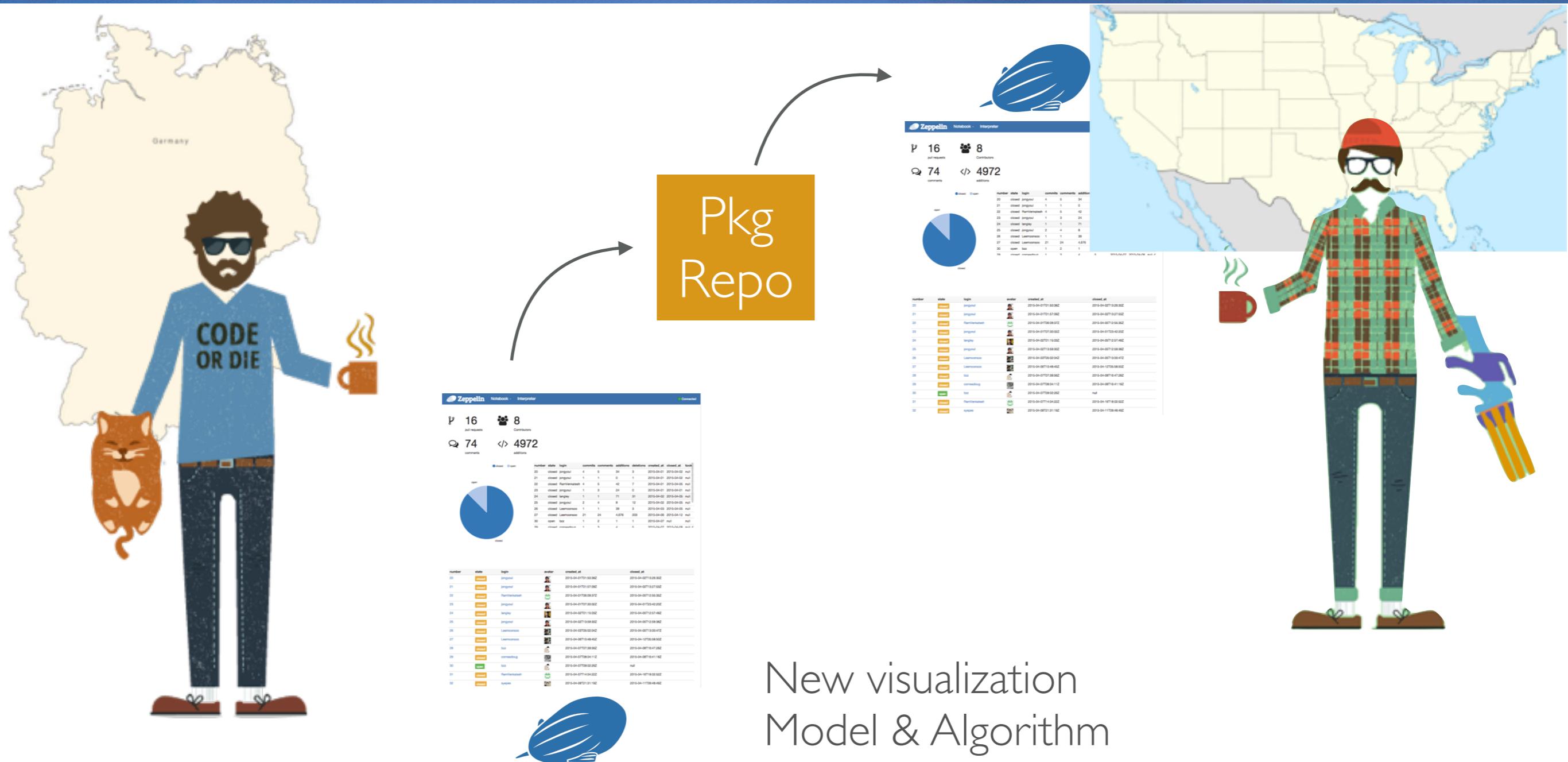
# Helium

# People do the similar work with different data

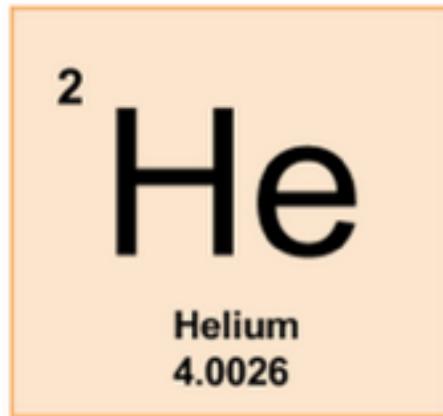


New visualization  
Model & Algorithm  
Data process pipeline

# Package and distribute work



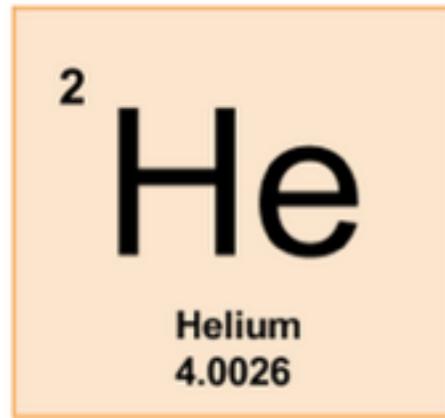
# Helium



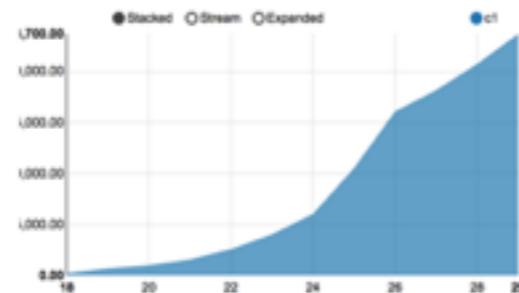
<https://s.apache.org/helium>

**Platform for  
Data Analytics Application  
on top of Apache Zeppelin**

# Helium Application



=



+

```
/** Configures the OAuth Credentials for accessing Twitter */
def configureTwitterCredentials(apiKey: String, apiSecret: String, accessToken: String) {
    val configs = new Map[String, String] +++
        Seq(
            "apiKey" -> apiKey,
            "apiSecret" -> apiSecret,
            "accessToken" -> accessToken
        )
    println("Configuring Twitter OAuth")
    configs.foreach{ case(key, value) =>
        if (value.trim.isEmpty) {
            throw new Exception("Error setting authentication - value for " + key)
        }
        val fullKey = "twitter4j.oauth." + key.replace("api", "consumer")
        System.setProperty(fullKey, value.trim)
        println("\tProperty " + fullKey + " set as [" + value.trim + "]")
    }
    println()
}
```

View



Algorithm



Zeppelin provided Resources

# Resources



## Data

- Result of last execution
- JDBC connection (from JDBC Interpreter)\*

## Computing

- SparkContext (from SparkInterpreter)
- Flink environment (from FlinkInterpreter)\*

## Any java object

- Provided by user created Interpreter
- Provided by user created Helium application

# Application Examples

## Data

- ex) get git commit log data

<https://github.com/Leemoonsoo/zeppelin-gitcommitdata>

## Computing

- ex) run cpu usage monitoring code across spark cluster, using SparkContext

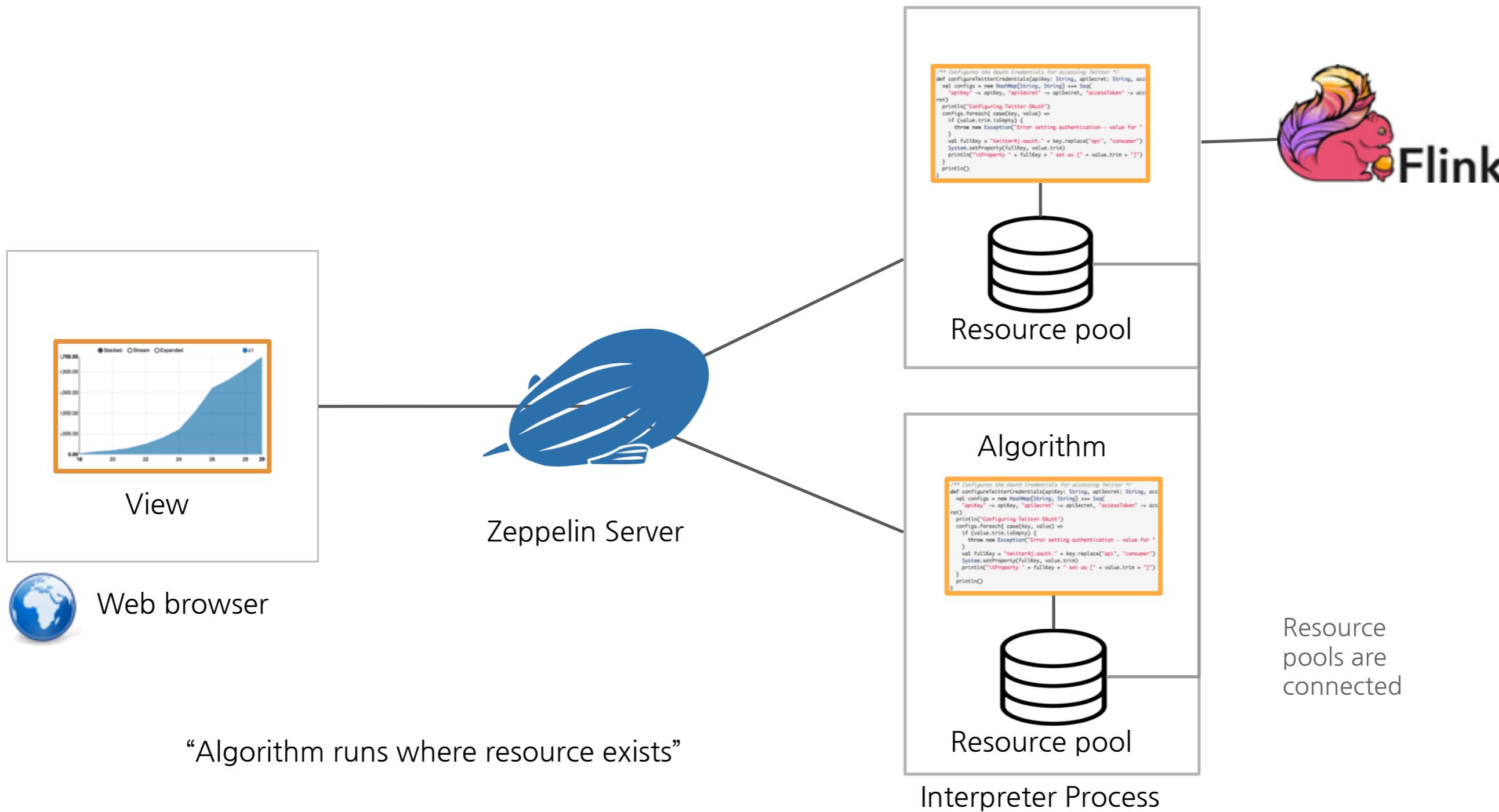
<https://github.com/Leemoonsoo/zeppelin-sparkmon>

## Visualization

- ex) display result data as a wordcloud

<https://github.com/Leemoonsoo/zeppelin-wordcloud>

# How it works



# API

## Easy API

```
class YourApplication extends org.apache.zeppelin.helium.Application {  
    @Override  
    public void run(ApplicationArgument arg, InterpreterContext context) {  
        ....  
    }  
}
```

Just extend helium.Application

# Application Spec

## Simple

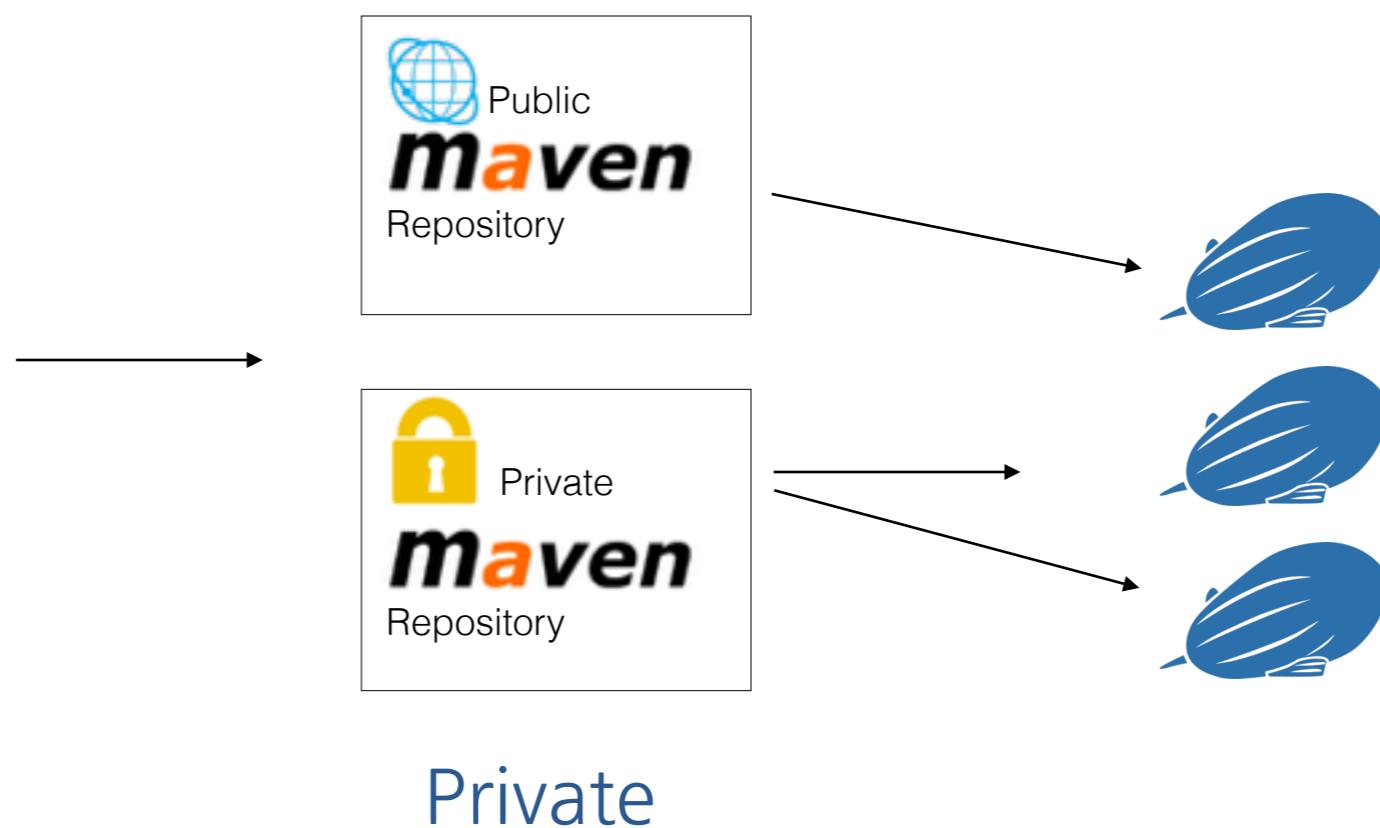
```
{  
  mavenArtifact : "groupId:artifactId:version",  
  className : "your.helium.application.Class",  
  icon : "fa fa-cloud",  
  name : "My app name",  
  description : "some description",  
  consume : [  
    "org.apache.spark.SparkContext"  
  ]  
}
```

Writing a spec file allow Zeppelin load application

# Deploy

Handy

Public



# Thank you

## Q & A

<http://zeppelin.incubator.apache.org/>

Moon  
[moon@nflabs.com](mailto:moon@nflabs.com)

NFLabs  
[www.nflabs.com](http://www.nflabs.com)