

A WEB APPLICATION FOR INTERACTIVE DATA ANALYSIS WITH SPARK

Romain Rigaux

Spark Summit, Jul 1, 2014

cloudera®



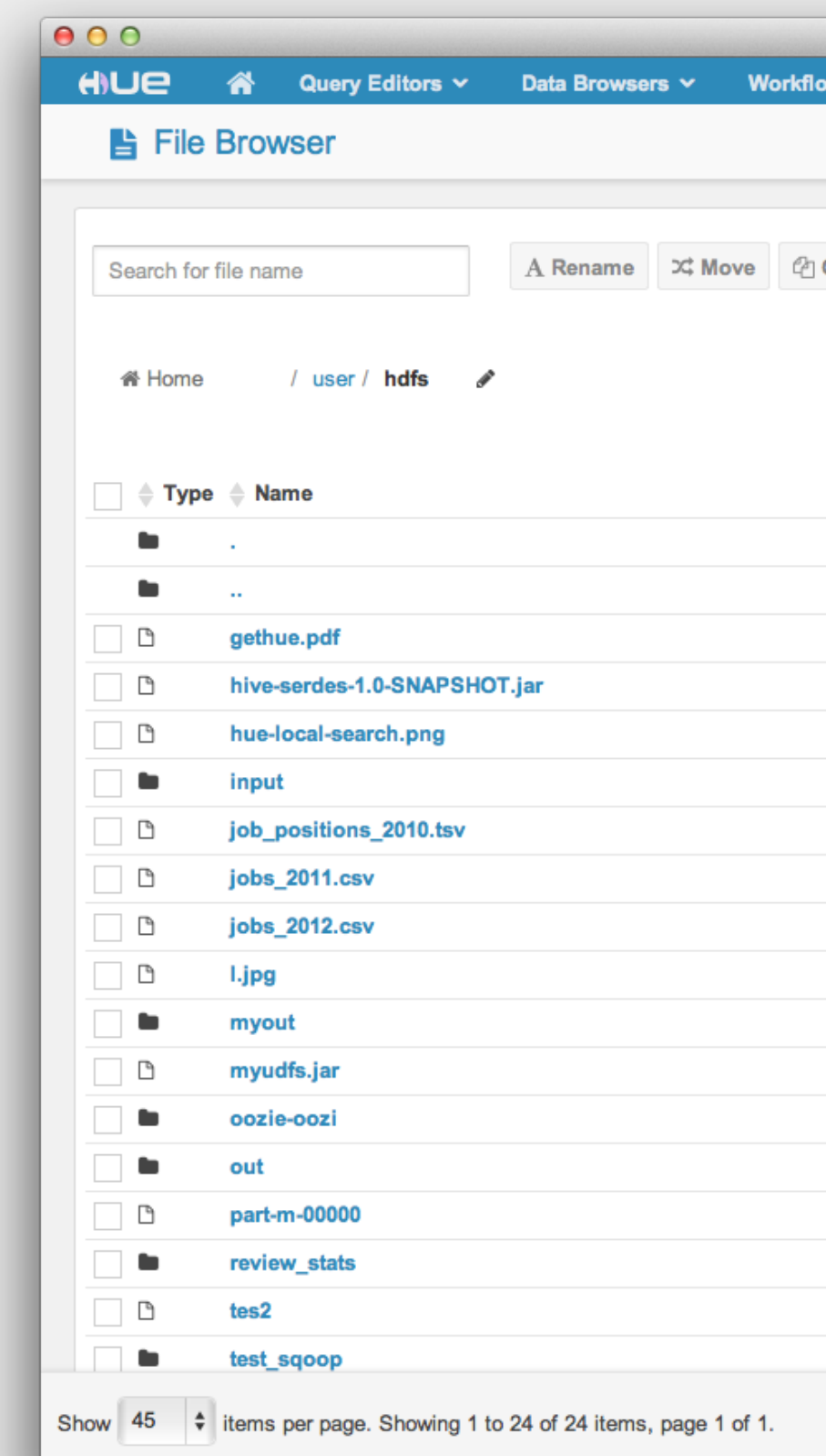
GOAL OF HUE

WEB INTERFACE FOR ANALYZING DATA
WITH APACHE HADOOP

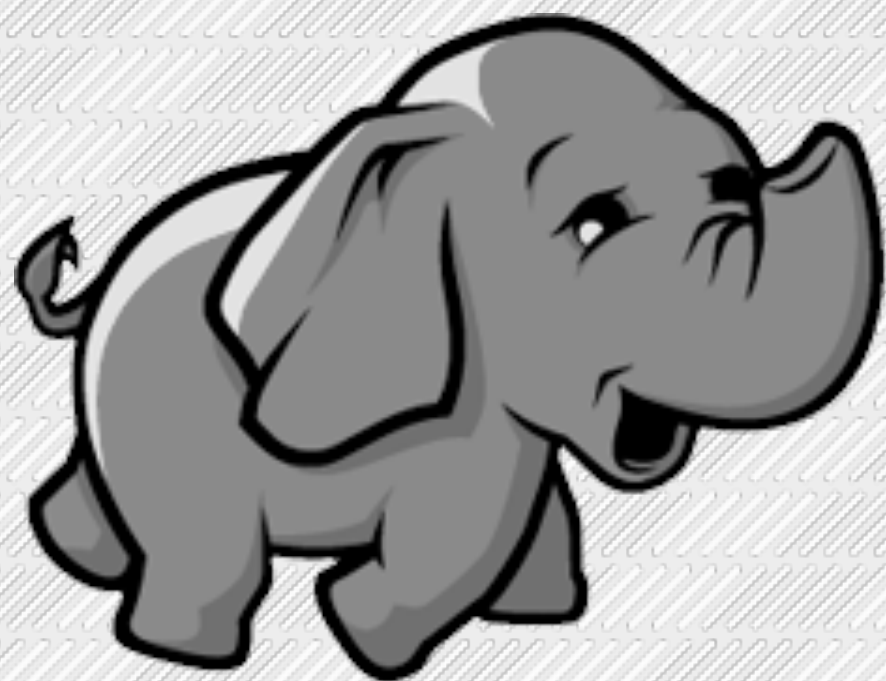
SIMPLIFY AND INTEGRATE

FREE AND OPEN SOURCE

—> OPEN UP BIG DATA



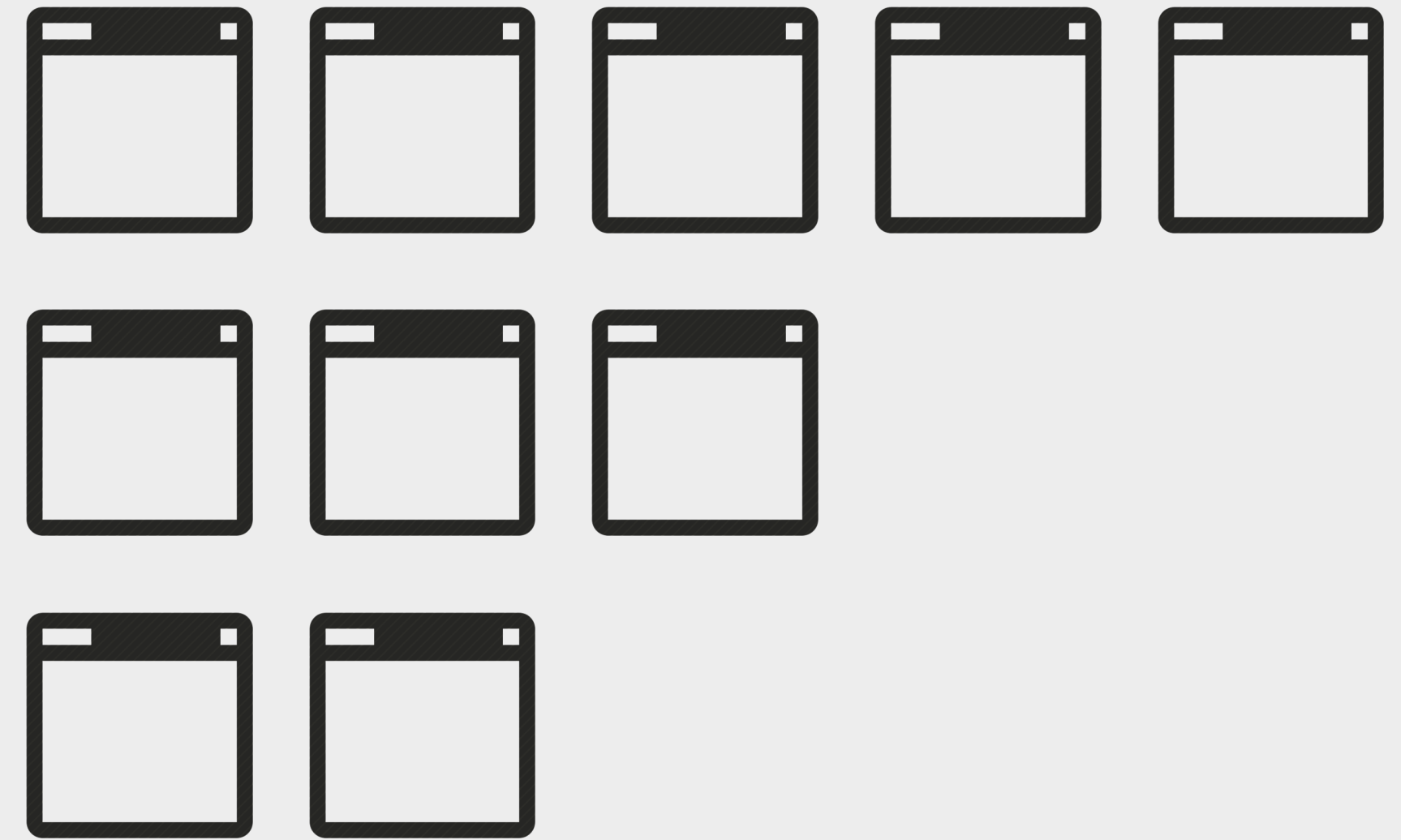
VIEW FROM 30K FEET



Hadoop



Web Server

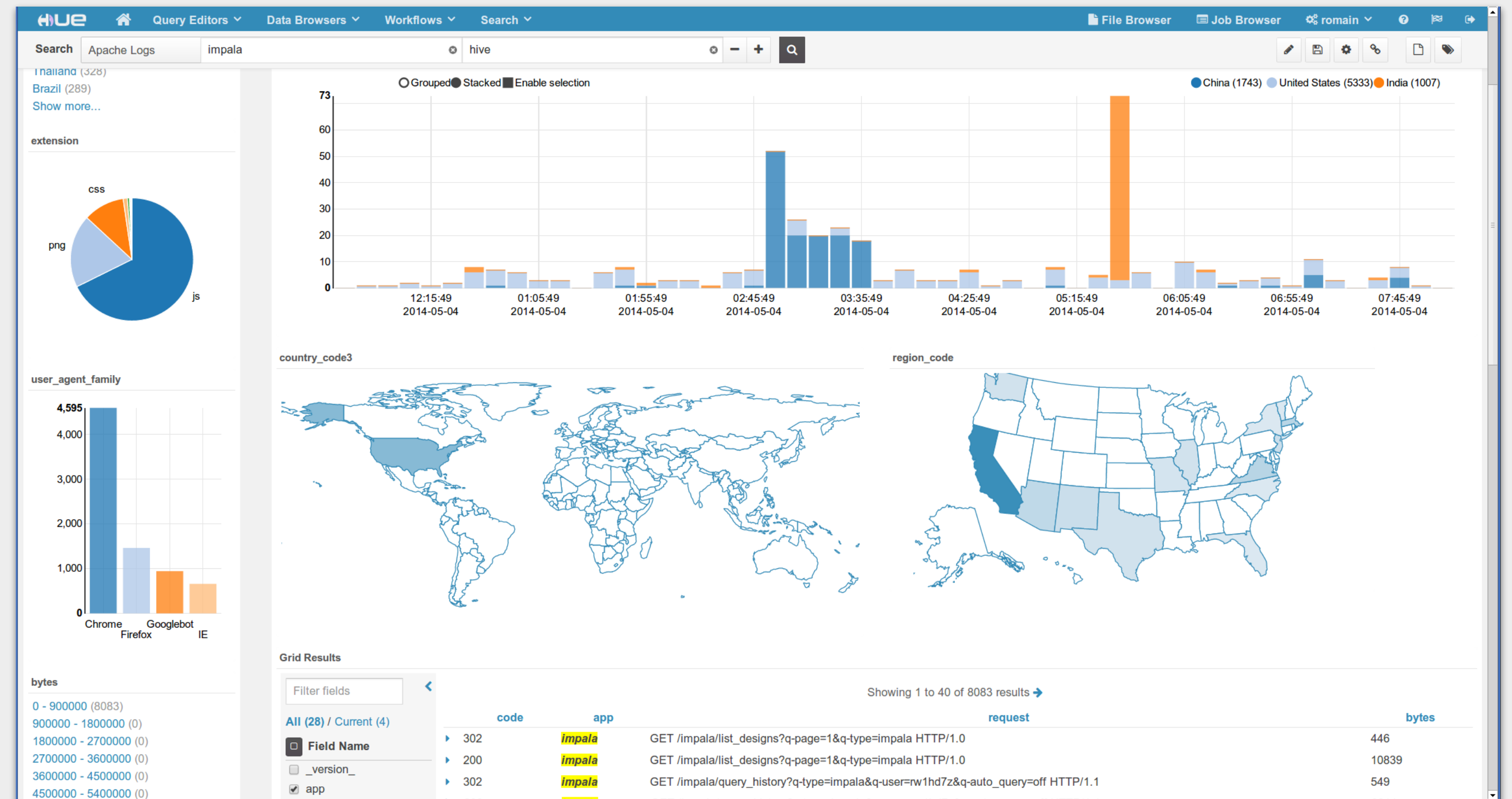


You, your colleagues and even that friend that uses IE9 ;)

LATEST HUE

HUE 3.6+

Where we are now, a brand new way to search and explore your data.



SPARK IGNITER

HUE

Home

Query Editors

Data Browsers

Workflows

Search

Other apps

File Browser

Job Browser

romain

?

Spark Igniter

Editor

Applications

Dashboard

Contexts

Uploads

App name

examples

Class path

spark.jobserver.WordCountExample

Context

Automatic

Create one?

Upload app

Create context

Parameters

Name

Value

input.string

a a a b c

..

Delete

Add

Execute

Save as...

or create a

New application

Download

Key

Value

a

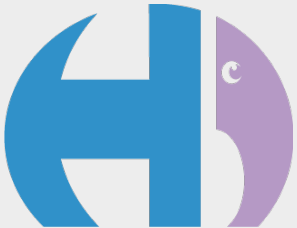
3

c

1

b

1

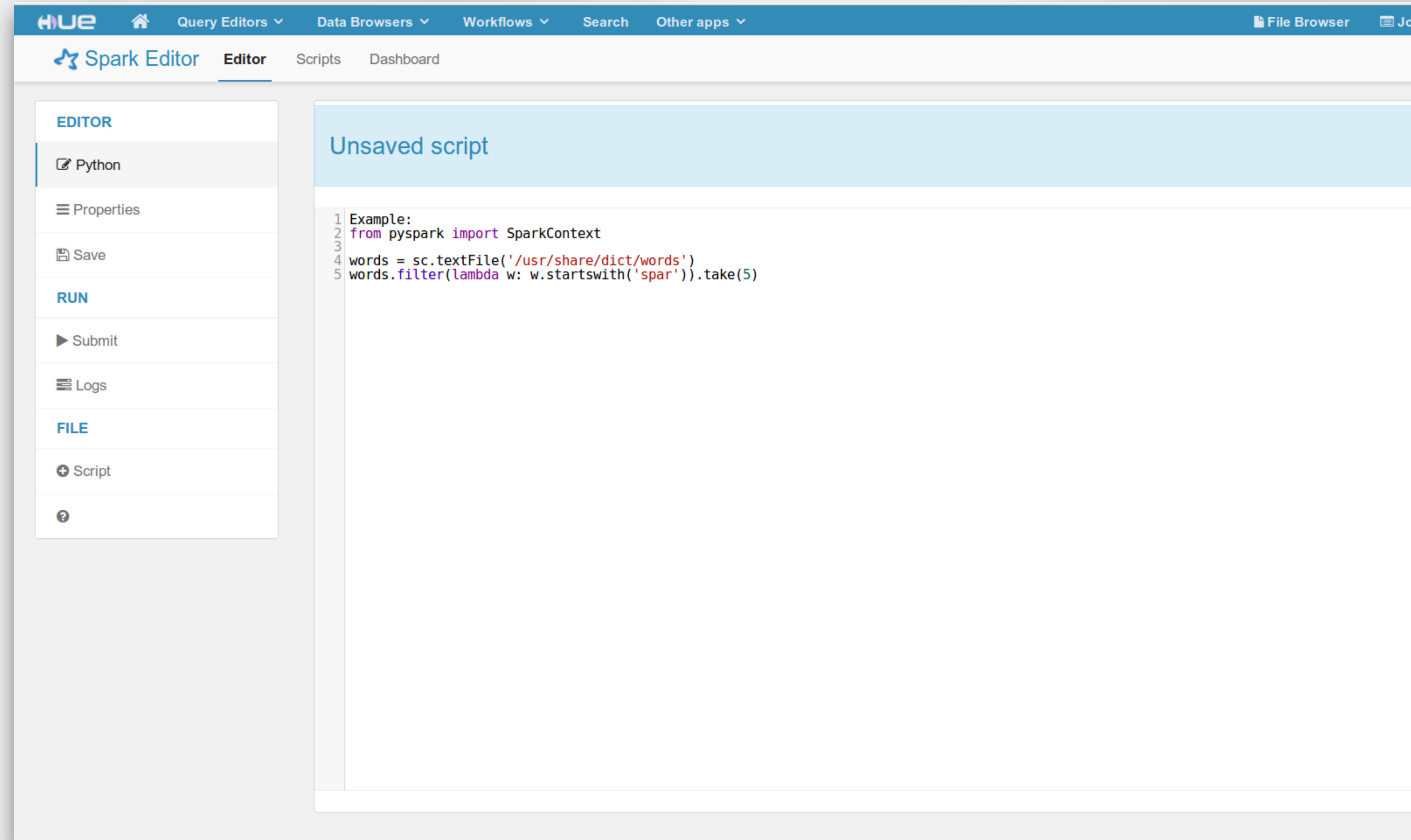


HISTORY

OCT 2013

Submit through Oozie

Shell like for Java, Scala, Python



“JUST A VIEW” ON TOP OF SPARK



Saved script metadata

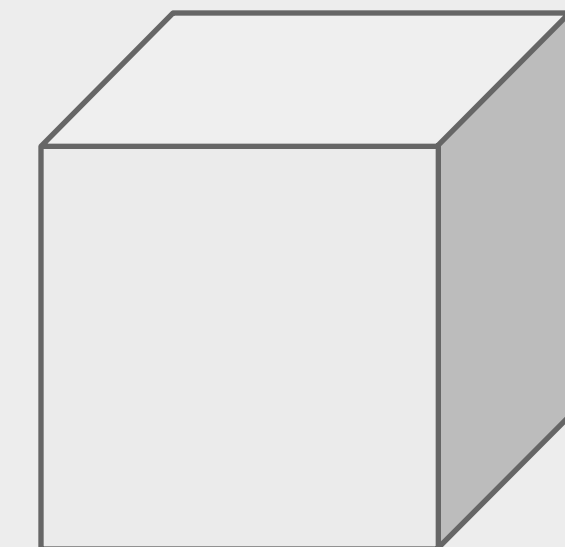
eg. name, args, classname, jar name...



Hue

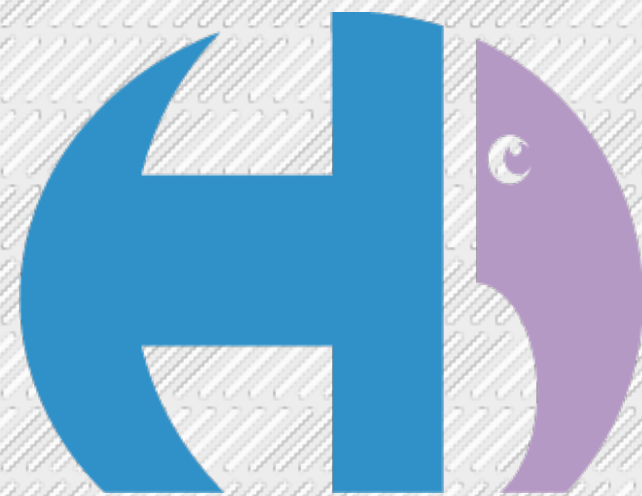


submit
list apps
list jobs
list contexts

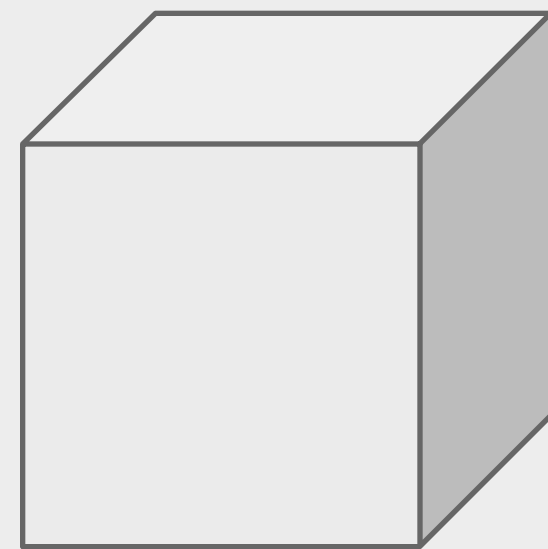


Job Server

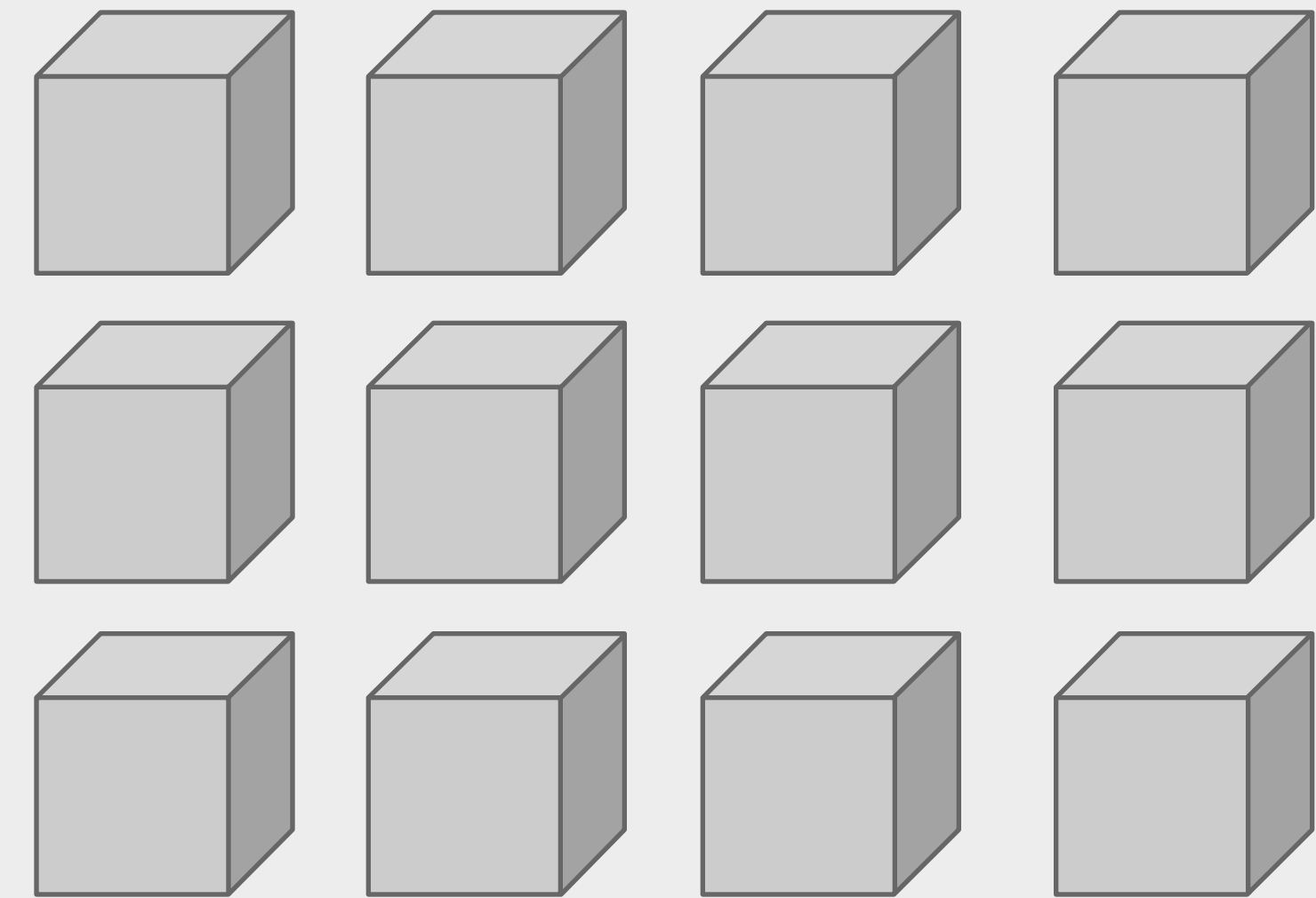
HOW TO TALK TO SPARK?



Hue

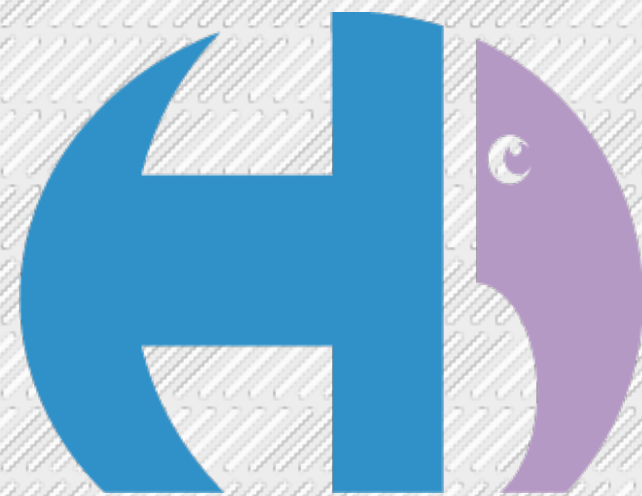


Spark Job Server

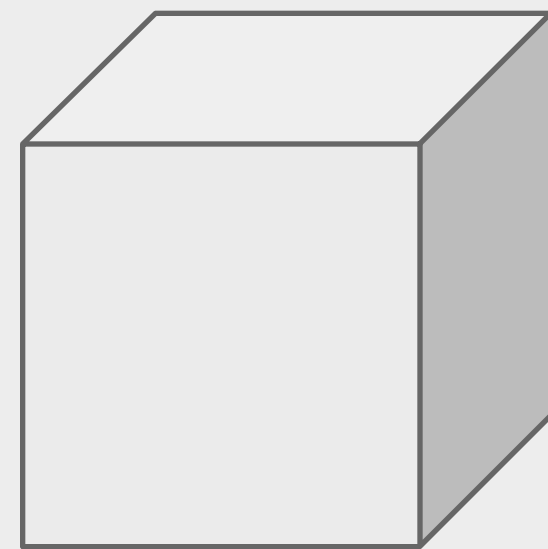


Spark

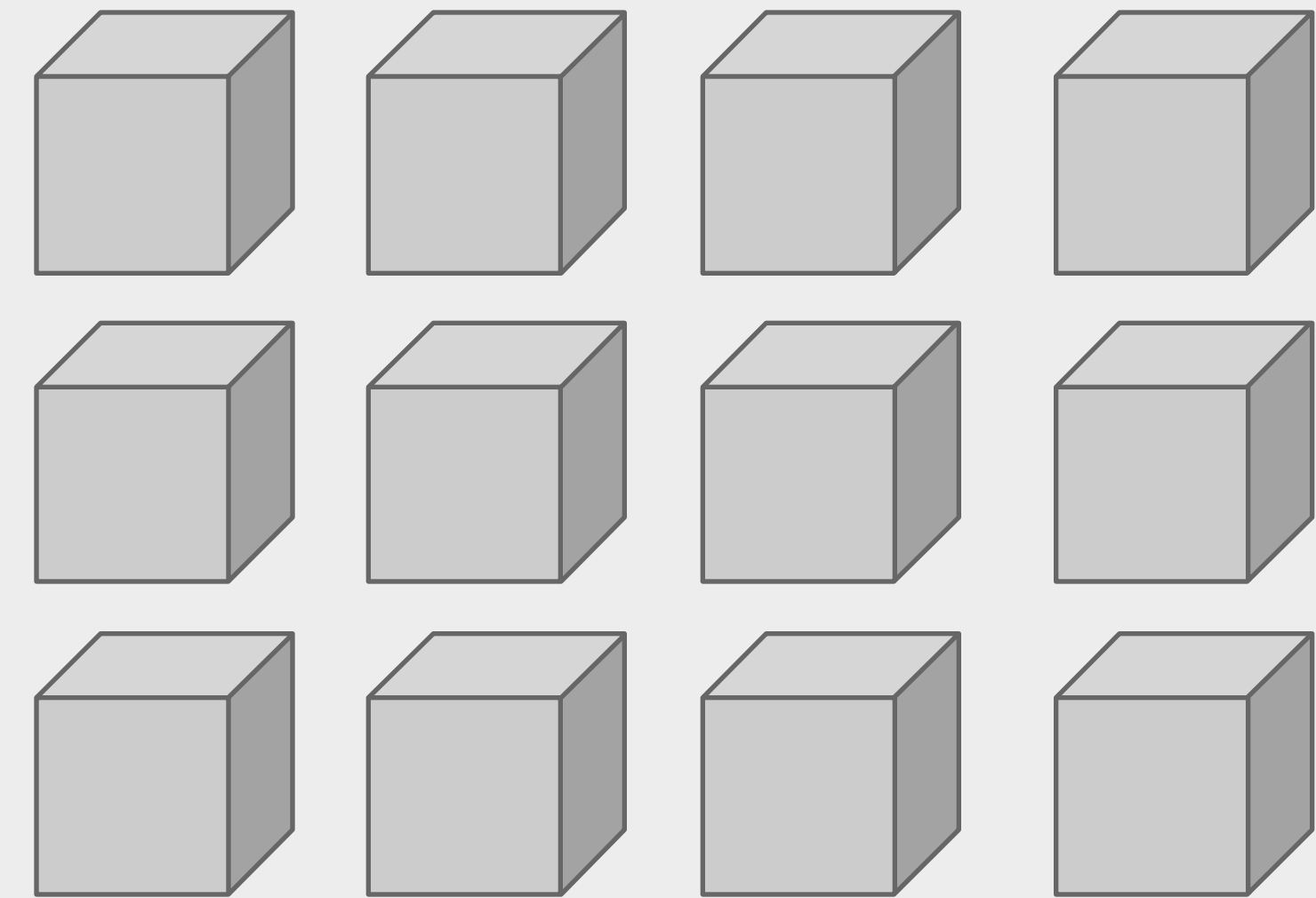
APP LIFE CYCLE



Hue



Spark Job Server



Spark

APP LIFE CYCLE

```
1 package spark.jobserver
2
3 import com.typesafe.config.{Config, ConfigFactory}
4 import org.apache.spark._
5 import scala.util.Try
6 import java.util.{Random, Date}
7
8 /**
9  * A long job for stress tests purpose.
10  * Iterative and randomized algorithm to compute Pi.
11  * Imagine a square centered at (1,1) of length 2 units.
12  * It tightly encloses a circle centered also at (1,1) of radius 1 unit.
13  * Randomly throw darts to them.
14  * We can use the ratio of darts inside the square and circle to approximate the Pi.
15  *
16  * stress.test.longpijob.duration controls how long it run in seconds.
17  * Longer duration increases precision.
18  *
19  */
20 object LongPiJob extends SparkJob {
21   private val rand = new Random(now)
22
23   def main(args: Array[String]) {
24     val sc = new SparkContext("local[4]", "LongPiJob")
25     val config = ConfigFactory.parseString("")
26     val results = runJob(sc, config)
27     println("Result is " + results)
28   }
29
30   override def validate(sc: SparkContext, config: Config): SparkJobValidation = {
31     SparkJobValid
32   }
33
34   override def runJob(sc: SparkContext, config: Config): Any = {
35     val duration = Try(config.getInt("stress.test.longpijob.duration")).getOrElse(5)
36     var hit:Long = 0
37     var total:Long = 0
```

... extend SparkJob

.scala

sbt _/package

JAR

Upload

Add

Execute

Save as...

or create a

New a

APP LIFE CYCLE

```
1 package spark.jobserver
2
3 import com.typesafe.config.{Config, ConfigFactory}
4 import org.apache.spark._
5 import scala.util.Try
6 import java.util.{Random, Date}
7
8 /**
9  * A long job for stress tests purpose.
10  * Iterative and randomized algorithm to compute Pi.
11  * Imagine a square centered at (1,1) of length 2 units.
12  * It tightly encloses a circle centered also at (1,1) of radius 1 unit.
13  * Randomly throw darts to them.
14  * We can use the ratio of darts inside the square and circle to approximate the Pi.
15  *
16  * stress.test.longpijob.duration controls how long it run in seconds.
17  * Longer duration increases precision.
18  *
19  */
20 object LongPiJob extends SparkJob {
21   private val rand = new Random(now)
22
23   def main(args: Array[String]) {
24     val sc = new SparkContext("local[4]", "LongPiJob")
25     val config = ConfigFactory.parseString("")
26     val results = runJob(sc, config)
27     println("Result is " + results)
28   }
29
30   override def validate(sc: SparkContext, config: Config): SparkJobValidation = {
31     SparkJobValid
32   }
33
34   override def runJob(sc: SparkContext, config: Config): Any = {
35     val duration = Try(config.getInt("stress.test.longpijob.duration")).getOrElse(5)
36     var hit:Long = 0
37     var total:Long = 0
```

... extend SparkJob

.scala

Context

sbt _/package

JAR

Upload

Add

Execute

Save as...

or create a

New app

create context: auto or manual

SPARK JOB SERVER

WHERE

<https://github.com/ooyala/spark-jobserver>

WHAT

REST job server for Spark

WHEN

Spark Summit talk Monday 5:45pm:

Spark Job Server: Easy Spark Job

Management by Ooyala

```
curl -d "input.string = a b c a b see" 'localhost:8090/jobs?
appName=test&classPath=spark.jobserver.WordCountExample'
{
  "status": "STARTED",
  "result": {
    "jobId": "5453779a-f004-45fc-a11d-a39dae0f9bf4",
    "context": "b7ea0eb5-spark.jobserver.WordCountExample"
  }
}
```


FOCUS ON UX

The screenshot shows the Cloudera Hue Spark Igniter interface. At the top, there's a navigation bar with tabs like Query Editors, Data Browsers, Workflows, Search, and Other apps. Below this, the 'Spark Igniter' section is active, showing the 'Editor' tab. The interface includes fields for 'App name' (set to 'examples'), 'Class path' (set to 'spark.jobserver.WordCountExample'), and 'Context' (set to 'Automatic'). There are buttons for 'Upload app' and 'Create context'. Below these, the 'Parameters' section is expanded, showing a table with columns 'Name' and 'Value'. The table contains one row: 'input.string' with value 'a a a b c'. There are buttons for 'Add', 'Delete', 'Execute', 'Save as...', 'or create a', 'New application', and 'Download'. At the bottom, there's a table with columns 'Key' and 'Value' showing the word counts: 'a' is 3, 'c' is 1, and 'b' is 1.

Name	Value
input.string	a a a b c

Key	Value
a	3
c	1
b	1

VS

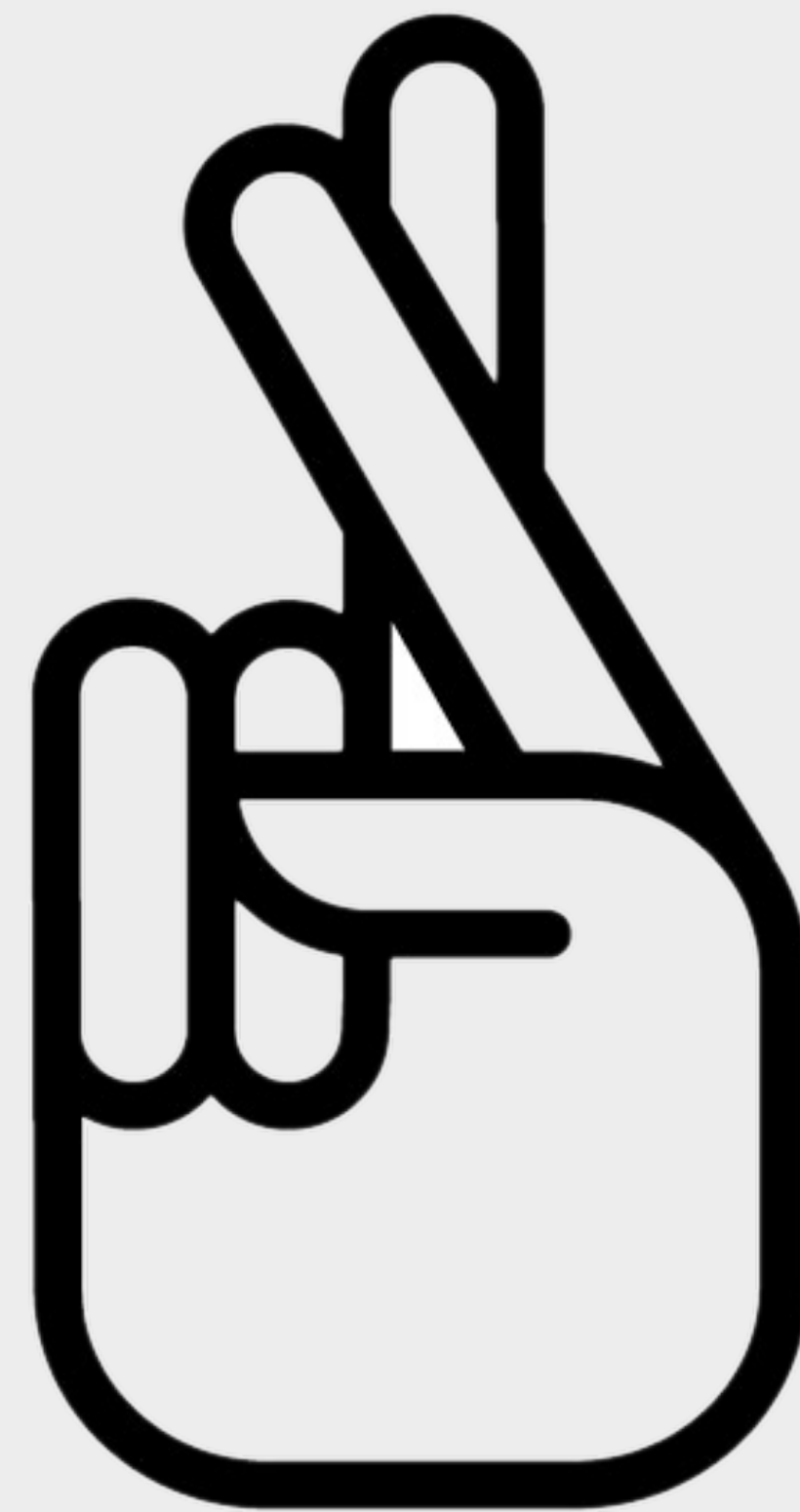
```
curl -d "input.string = a b c a b see" 'localhost:8090/jobs?appName=test&classPath=spark.jobserver.WordCountExample'
{
  "status": "STARTED",
  "result": {
    "jobId": "5453779a-f004-45fc-a11d-a39dae0f9bf4",
    "context": "b7ea0eb5-spark.jobserver.WordCountExample"
  }
}
```


TRAIT SPARKJOB

```
/**
 * This trait is the main API for Spark jobs submitted to the Job Server.
 */
trait SparkJob {
  /**
   * This is the entry point for a Spark Job Server to execute Spark jobs.
   */
  def runJob(sc: SparkContext, jobConfig: Config): Any

  /**
   * This method is called by the job server to allow jobs to validate their input and reject
   * invalid job requests. */
  def validate(sc: SparkContext, config: Config): SparkJobValidation
}
```


DEMO TIME



LIVE
DEMO

demo.gethue.com/spark

The screenshot displays the Hue web interface for the Spark Igniter application. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, and Search. The main header shows 'Spark Igniter' with the 'Editor' tab selected. Below the header, the 'App name' is set to 'examples' and the 'Class path' is 'spark.jobserver.WordCountExample'. The 'Parameters' section contains a table with one parameter: 'input.string' with the value 'a a a b c'. An 'Add' button is located below the table. At the bottom, there are buttons for 'Execute', 'Save as...', and 'New application'.

Name	Value
input.string	a a a b c

Execute Save as... or create a New application

Key	Value
a	3
c	1
b	1

CURRENT TECH SUM-UP

STANDALONE APP

SCALA 2.10

SPARK 0.9

HUE C5+

ROADMAP

WHAT

- **YARN**
- HUE-2134 [spark] App revamp and Job Server needs
 - Impersonation
 - Status report
 - Fetch N from result set
 - Python?
- Full Hue integration with HDFS, JobBrowser, Hive, charts...
- On the fly compile of Scala, Java?
- ?



THANK YOU!

WEBSITE

<http://gethue.com>

LEARN

<http://gethue.com/category/spark/>

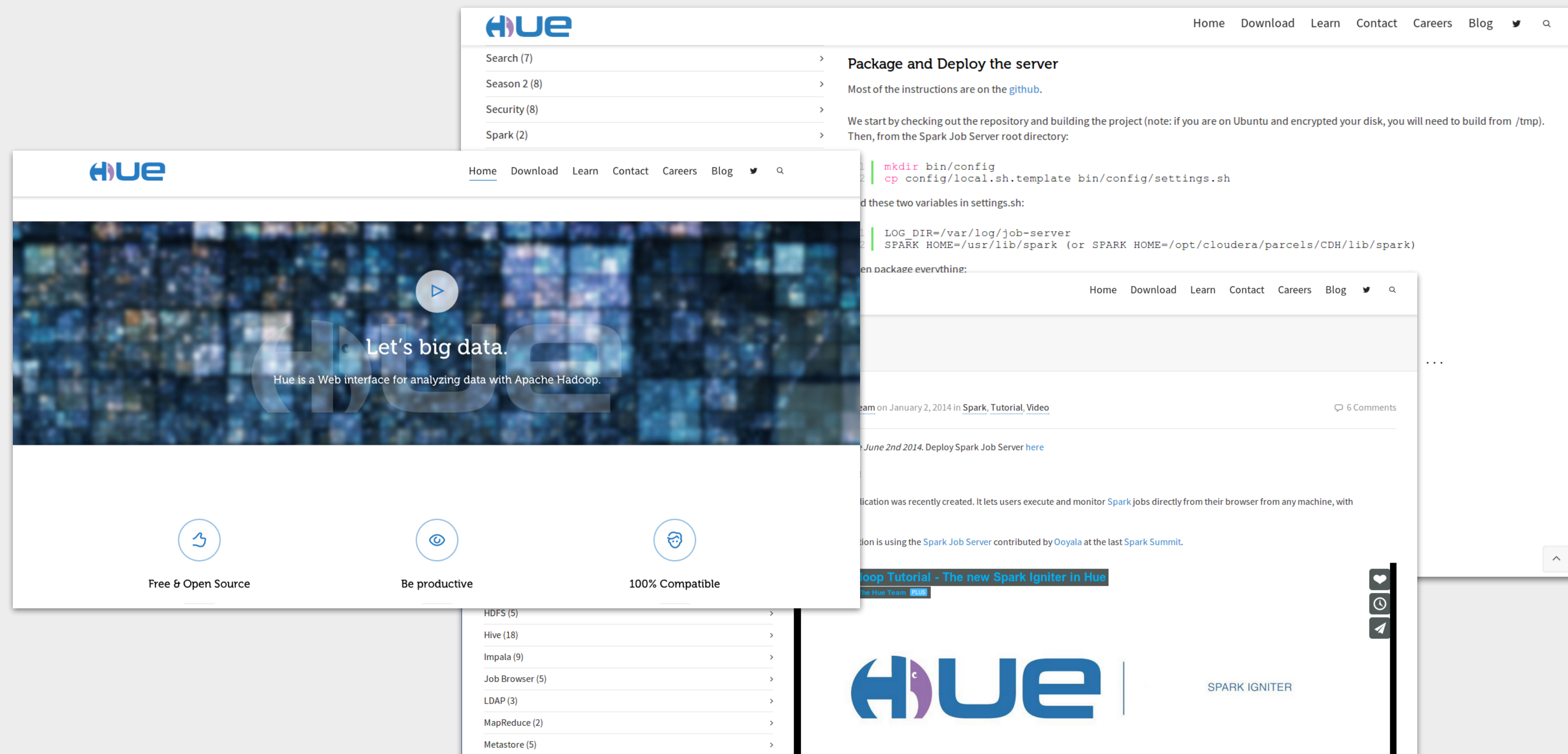
TWITTER

[@gethue](https://twitter.com/gethue)

USER GROUP

hue-user@

cloudera®



<http://gethue.com/get-started-with-spark-deploy-spark-server-and-compute-pi-from-your-web-browser/>

