Spark summit EUROPE 2015

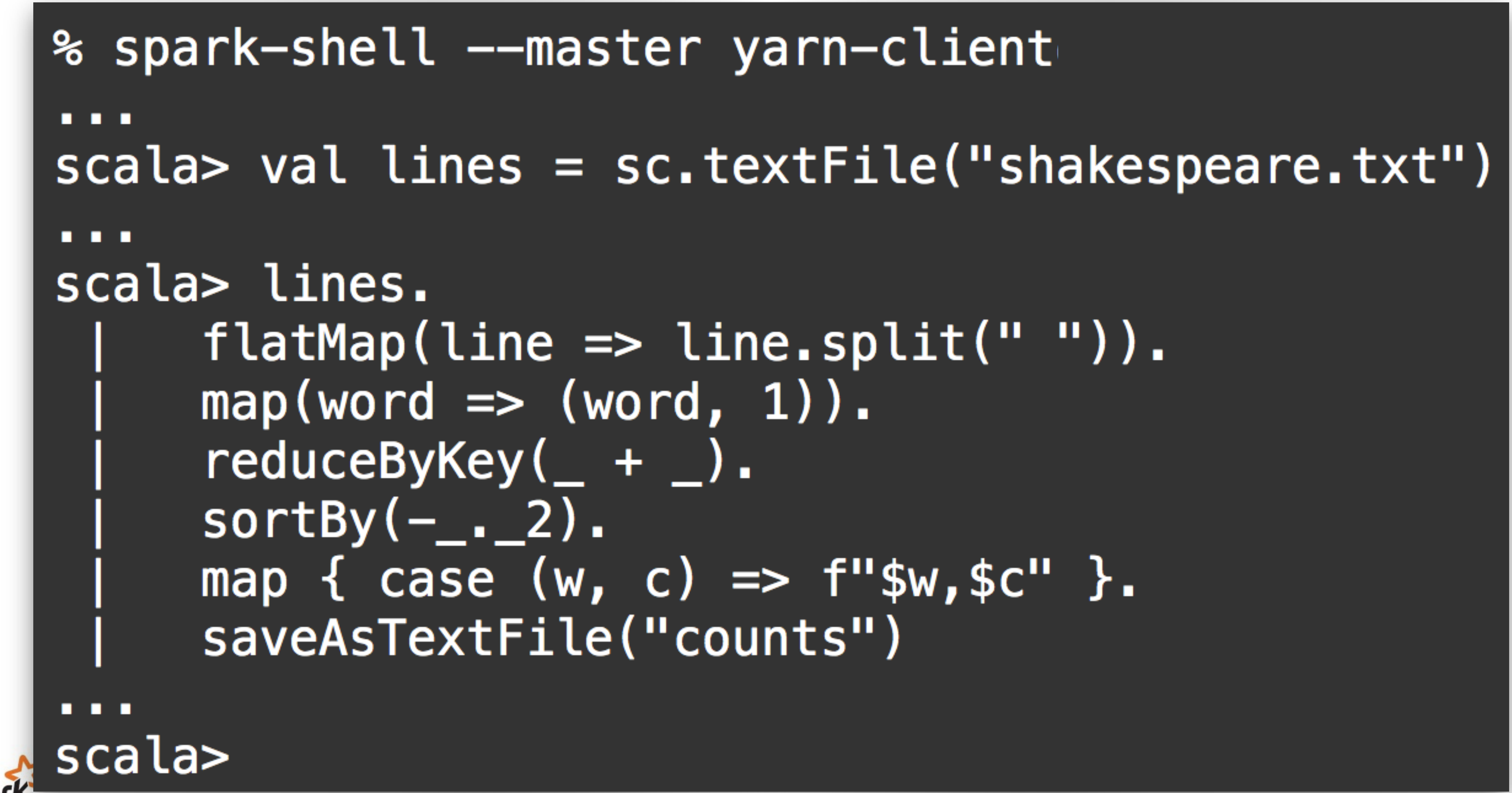
DATA SCIENCE AND ENGINEERING AT SCALE

AMSTERDAM • OCTOBER 27-29

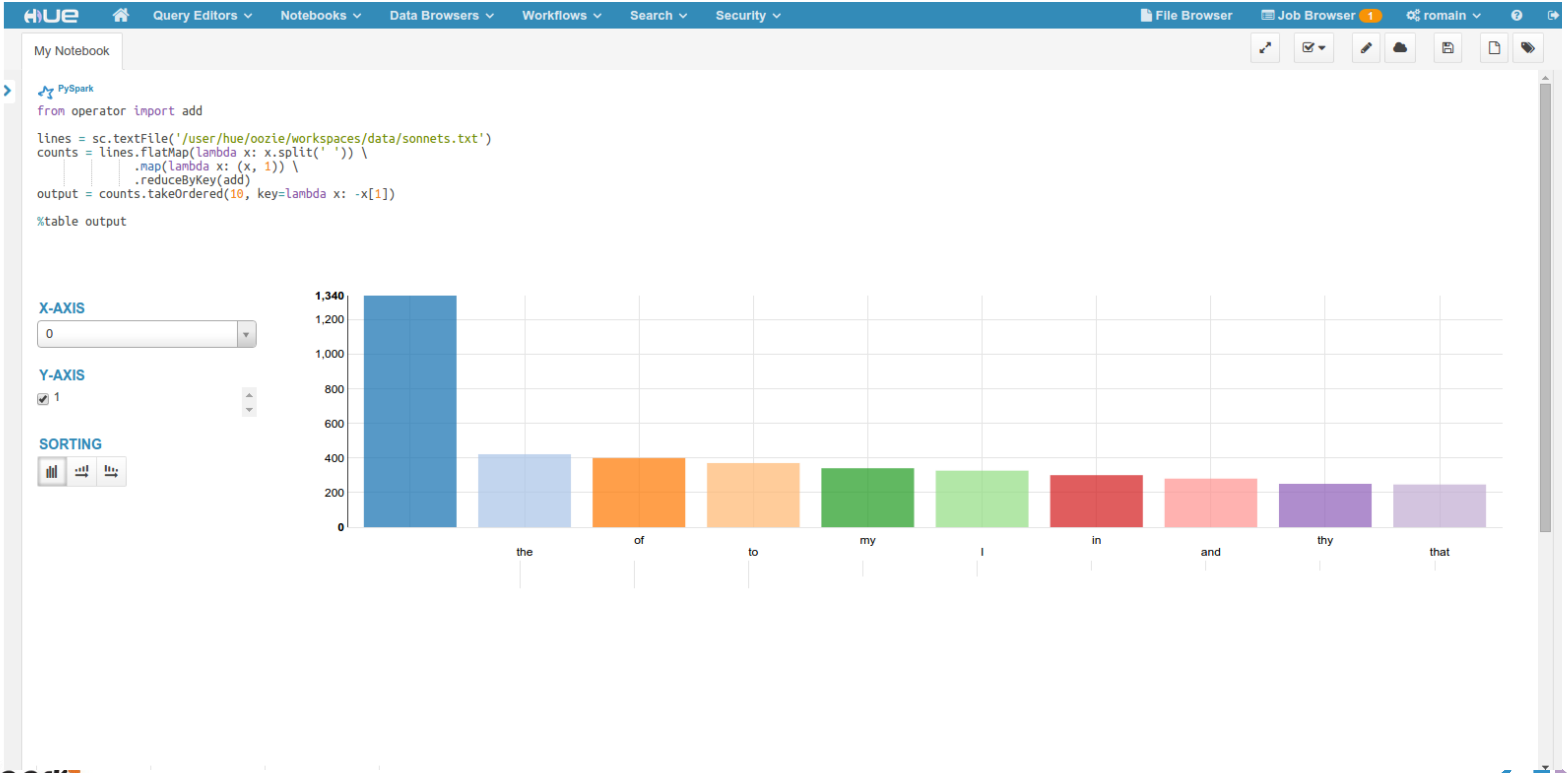
BUILDING A REST JOB SERVER FOR INTERACTIVE SPARK AS A SERVICE

Romain Rigaux - Cloudera
Erick Tryzelaar - Cloudera

WHY?



```
% spark-shell --master yarn-client
...
scala> val lines = sc.textFile("shakespeare.txt")
...
scala> lines.
     |    flatMap(line => line.split(" ")).
     |    map(word => (word, 1)).
     |    reduceByKey(_ + _).
     |    sortBy(-_._2).
     |    map { case (w, c) => f"$w,$c" }.
     |    saveAsTextFile("counts")
...
scala>
```



```
from operator import add

lines = sc.textFile('/user/hue/oozie/workspaces/data/sonnets.txt')
counts = lines.flatMap(lambda x: x.split(' ')) \
              .map(lambda x: (x, 1)) \
              .reduceByKey(add)
output = counts.takeOrdered(10, key=lambda x: -x[1])

%table output
```

X-AXIS

0

Y-AXIS

☑ 1

SORTING

WHY SPARK
AS A SERVICE?

NOTEBOOKS

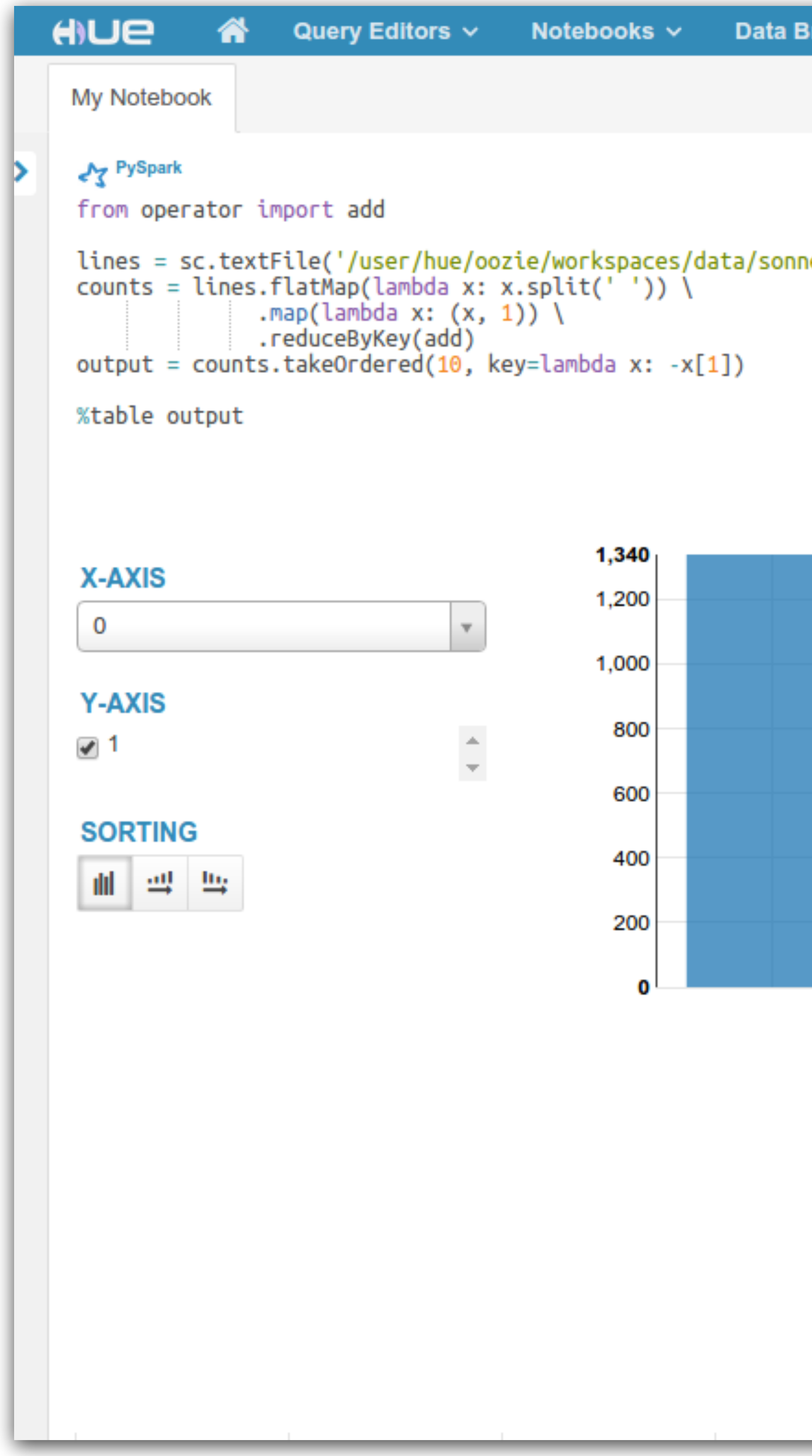
EASY ACCESS FROM ANYWHERE

SHARE SPARK CONTEXTS AND RDDs

BUILD APPS

SPARK MAGIC

...

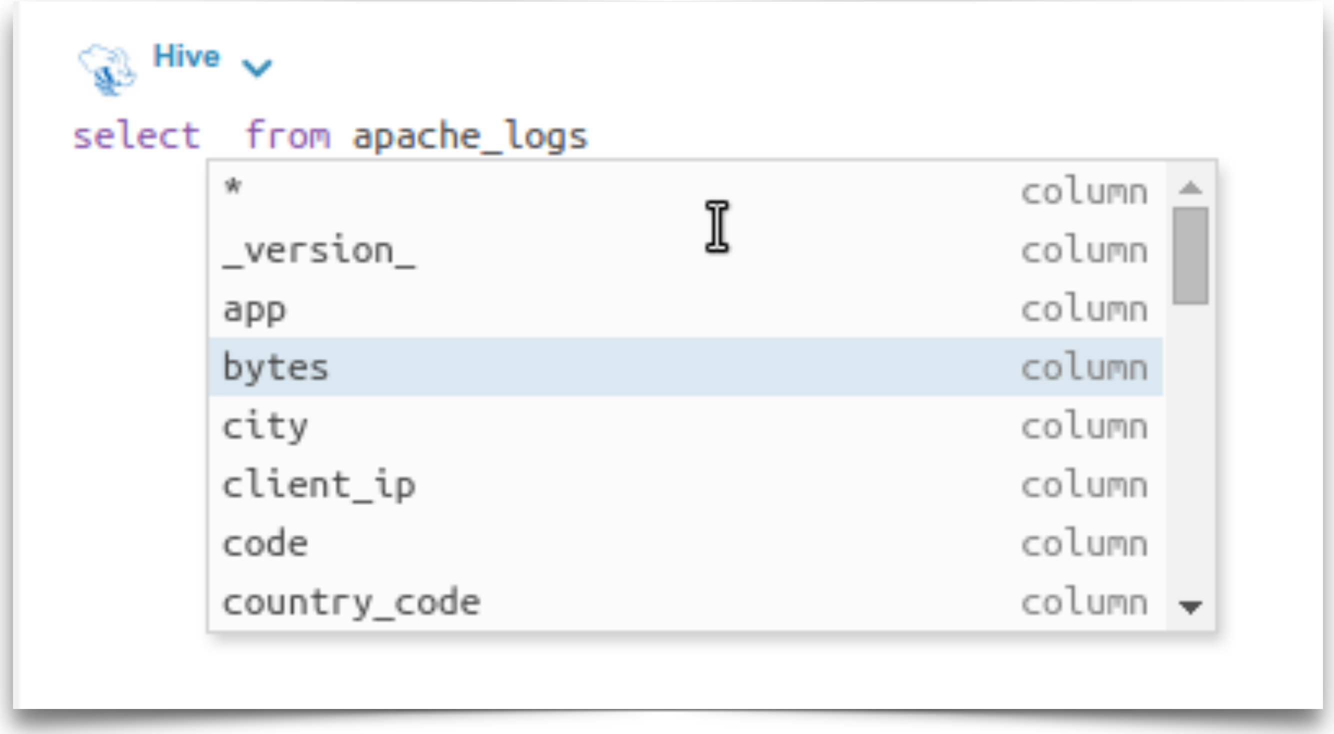


```
from operator import add

lines = sc.textFile('/user/hue/oozie/workspaces/data/sonn
counts = lines.flatMap(lambda x: x.split(' ')) \
              .map(lambda x: (x, 1)) \
              .reduceByKey(add)
output = counts.takeOrdered(10, key=lambda x: -x[1])

%table output
```

X-AXIS
0

Y-AXIS
☑ 1

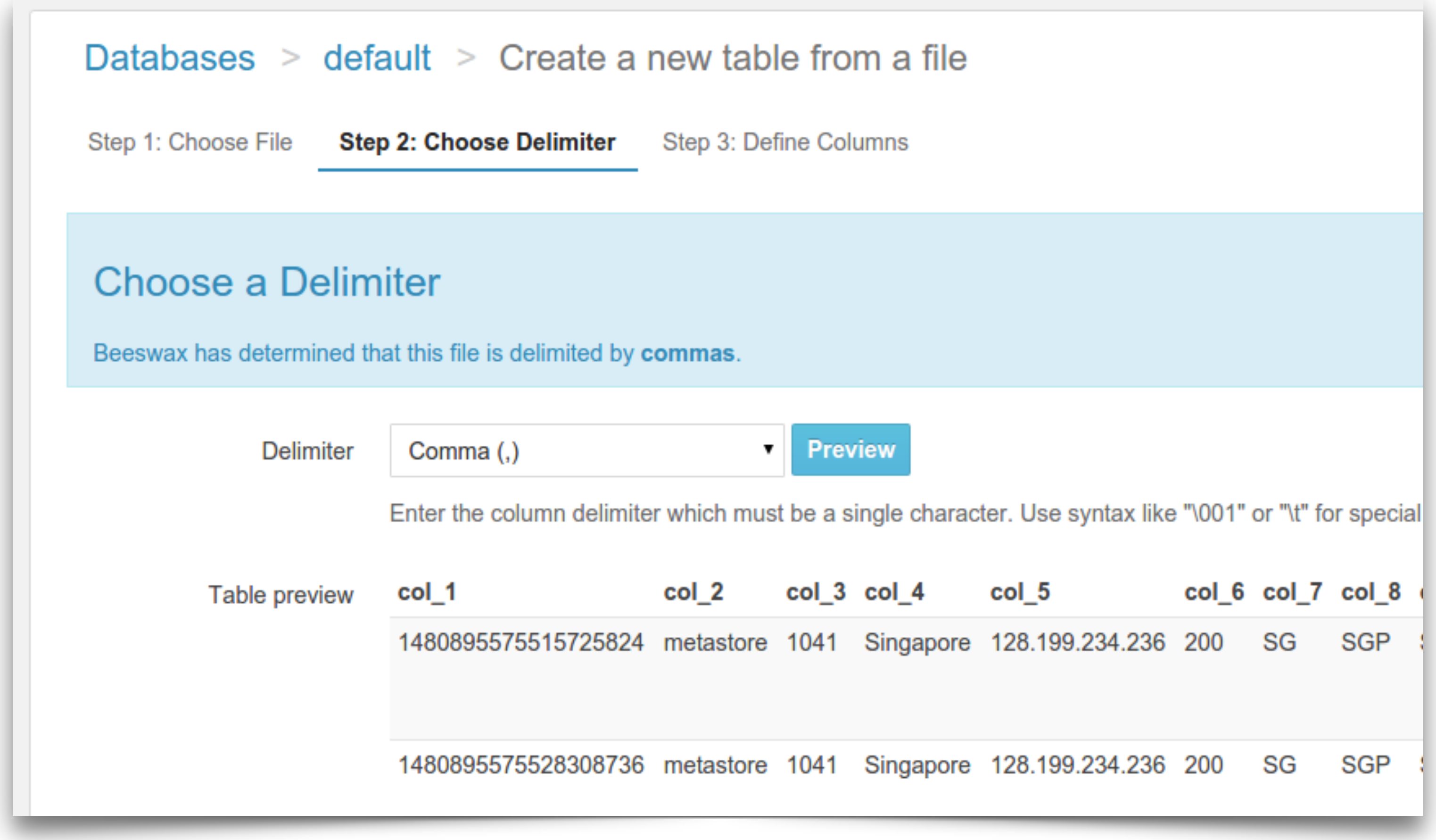
WHY SPARK
IN HUE?



Hive

```
select  from apache_logs
```

*	column
version	column
app	column
bytes	column
city	column
client_ip	column
code	column
country_code	column



Databases > default > Create a new table from a file

Choose a Delimiter

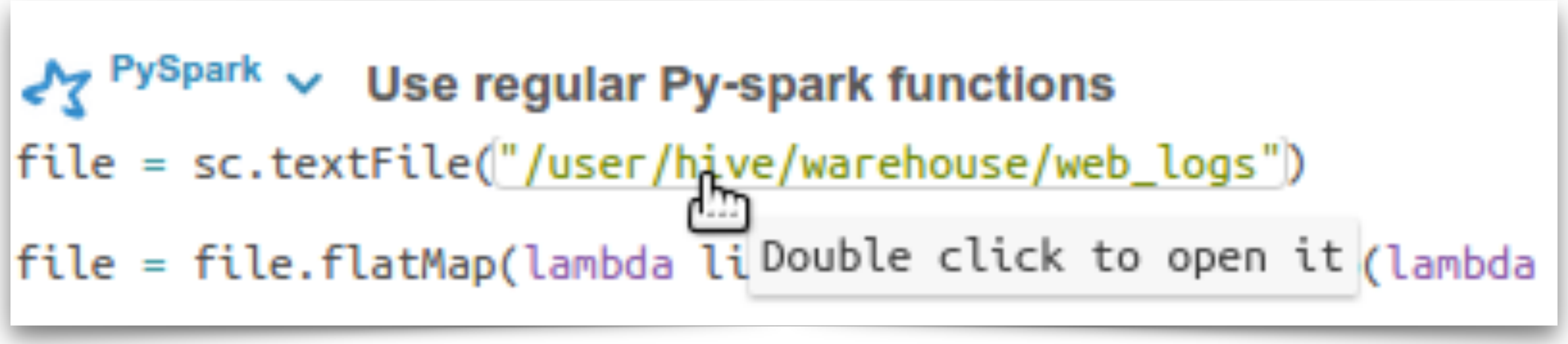
Beeswax has determined that this file is delimited by **commas**.

Delimiter: Comma (,) ▾

Enter the column delimiter which must be a single character. Use syntax like "\001" or "\t" for special

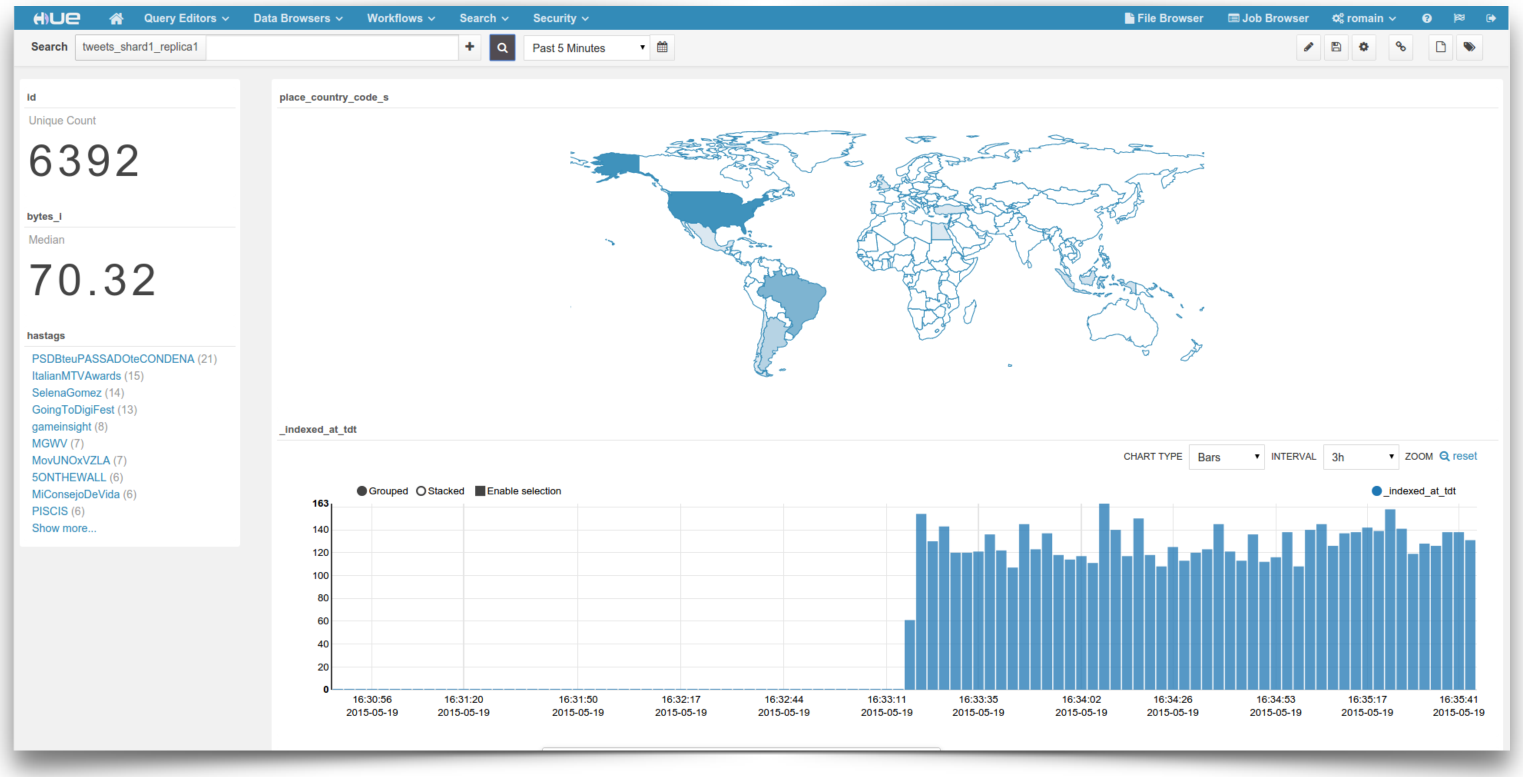
Table preview

col_1	col_2	col_3	col_4	col_5	col_6	col_7	col_8
1480895575515725824	metastore	1041	Singapore	128.199.234.236	200	SG	SGP
1480895575528308736	metastore	1041	Singapore	128.199.234.236	200	SG	SGP



PySpark **Use regular Py-spark functions**

```
file = sc.textFile("/user/hive/warehouse/web_logs")

file = file.flatMap(lambda li Double click to open it (lambda
```



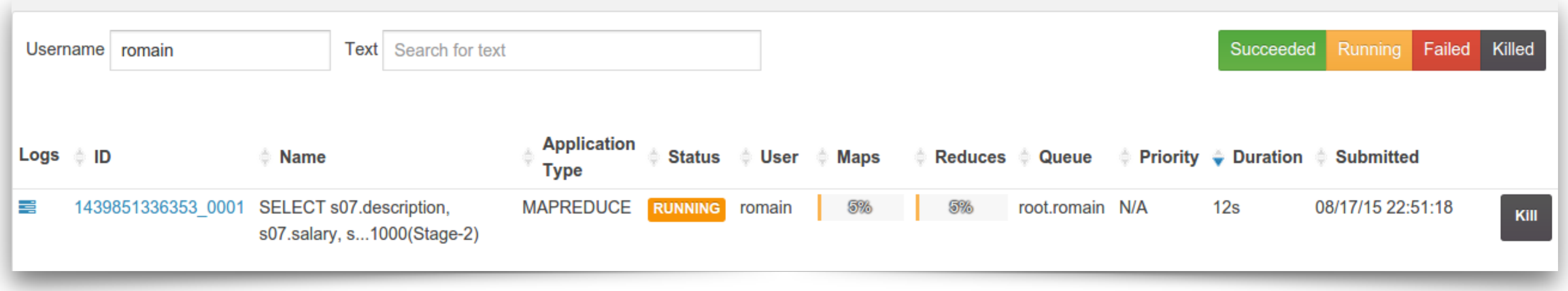
Search: tweets_shard1_replica1 Past 5 Minutes

id — Unique Count: 6392

bytes_i — Median: 70.32

hastags

- PSDBteuPASSADOteCONDENA (21)
- ItalianMTVAwards (15)
- SelenaGomez (14)
- GoingToDigiFest (13)
- gameinsight (8)
- MGWV (7)
- MovUNOxVZLA (7)
- 5ONTHEWALL (6)
- MiConsejoDeVida (6)
- PISCIS (6)
- Show more...



Username: romain Text: Search for text

Logs	ID	Name	Application Type	Status	User	Maps	Reduces	Queue	Priority	Duration	Submitted	
☰	1439851336353_0001	SELECT s07.description, s07.salary, s...1000(Stage-2)	MAPREDUCE	RUNNING	romain	5%	5%	root.romain	N/A	12s	08/17/15 22:51:18	Kill

MARRIED WITH FULL HADOOP ECOSYSTEM

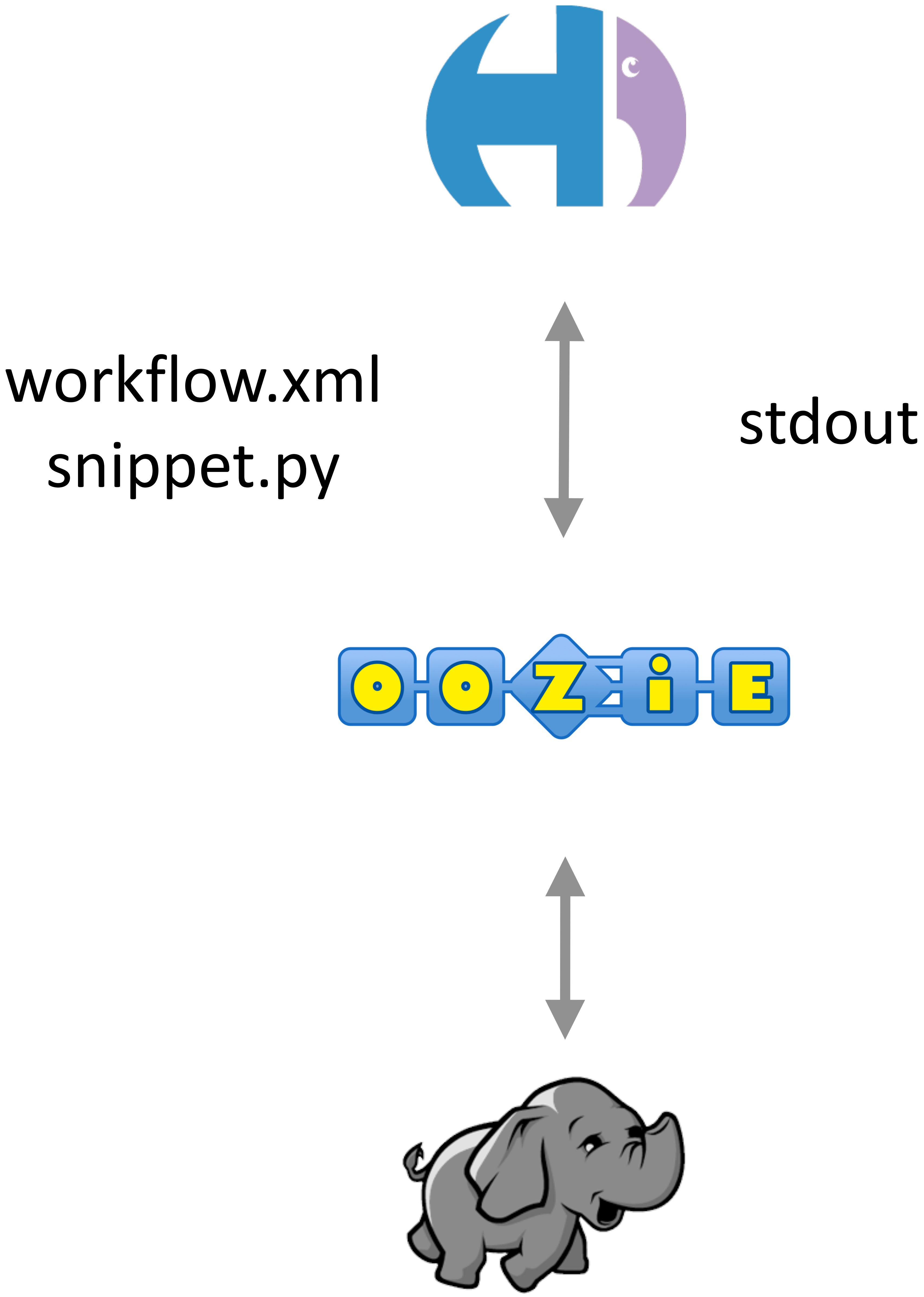
HISTORY
V1: OOZIE

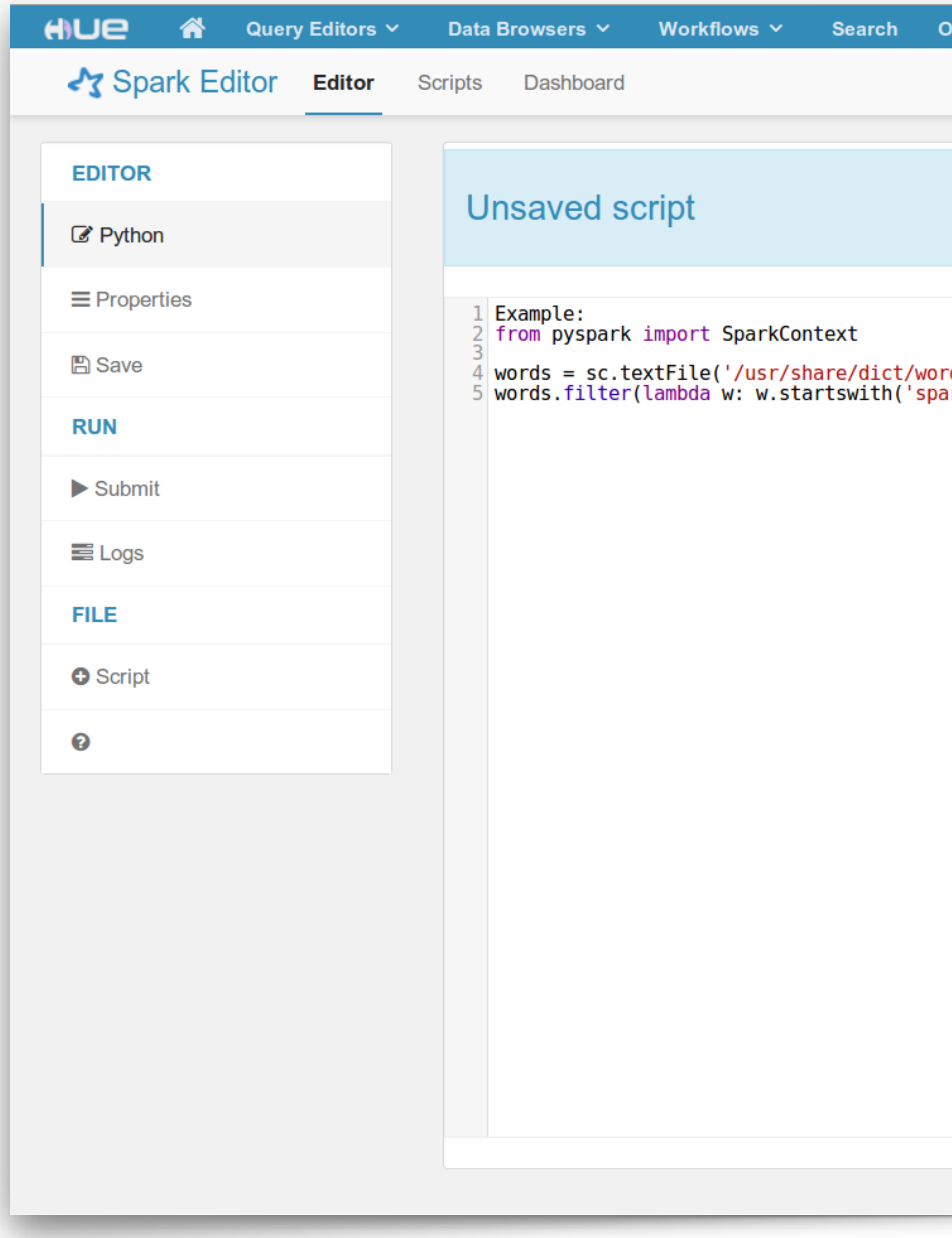
THE GOOD

- It works
- Code snippet

THE BAD

- Submit through Oozie
- Shell action
- Very Slow
- Batch





Unsaved script

```
Example:
from pyspark import SparkContext

words = sc.textFile('/usr/share/dict/wor
words.filter(lambda w: w.startswith('spa
```

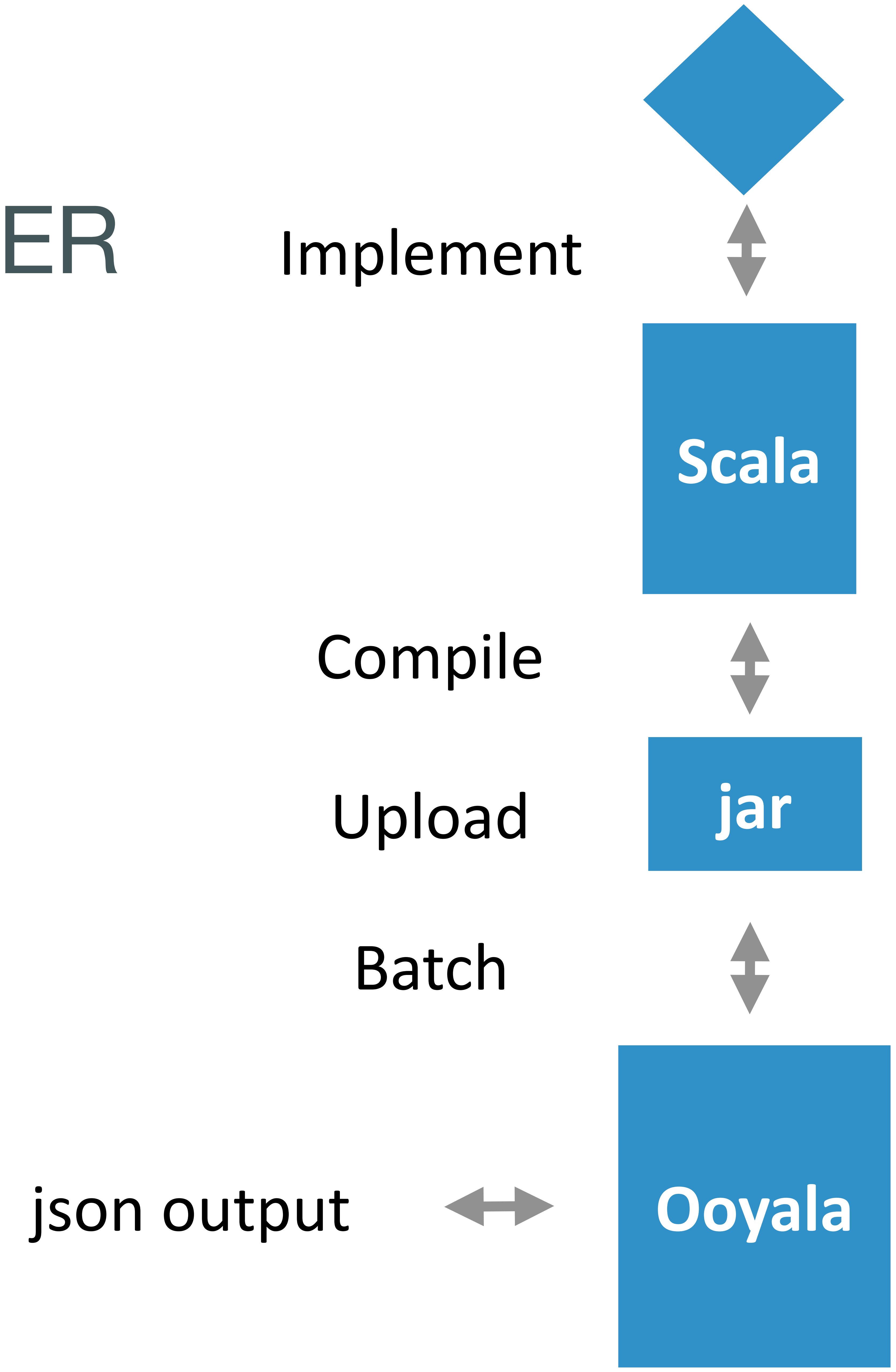
HISTORY
V2: SPARK IGNITER

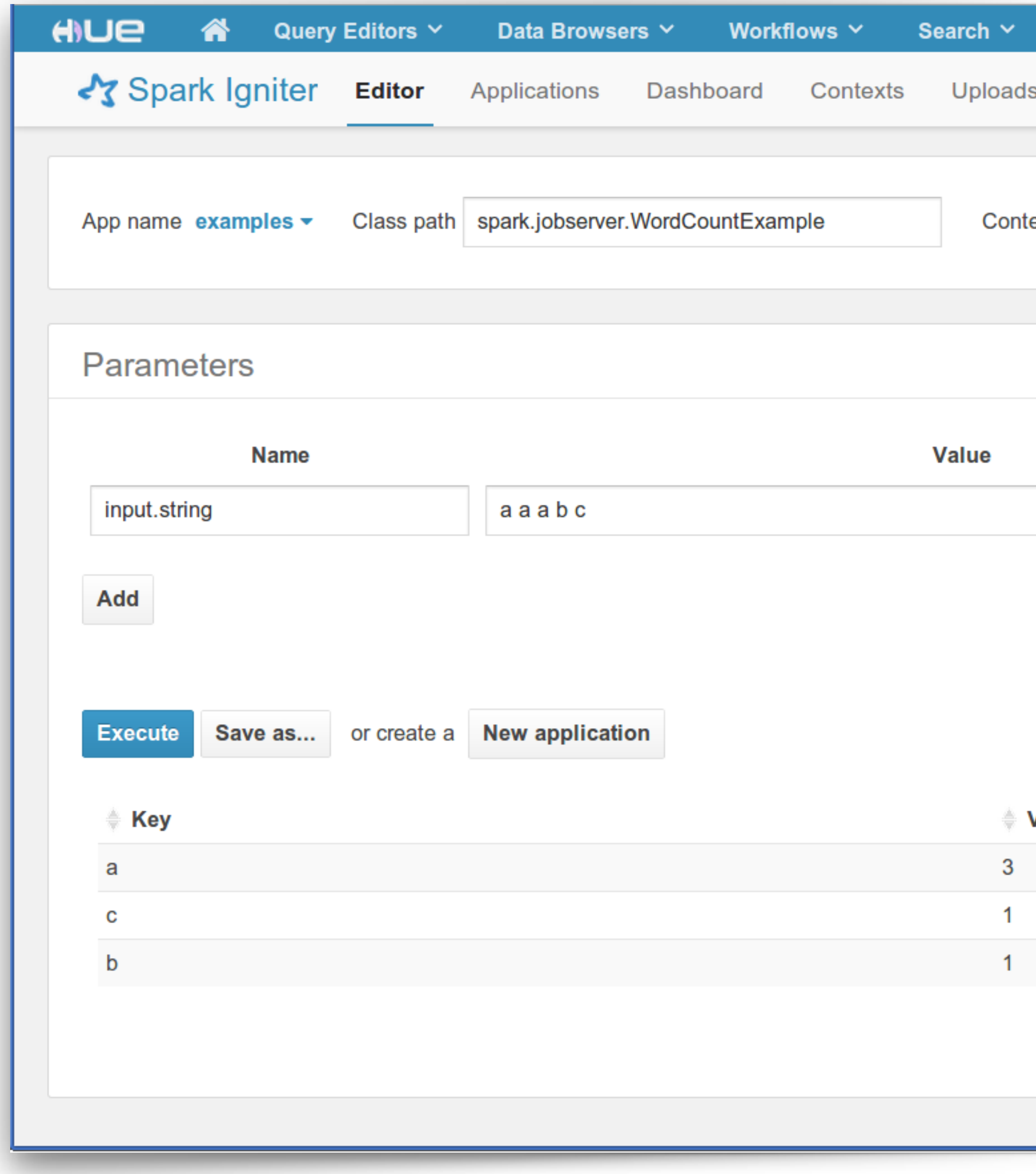
THE GOOD

- It works better

THE BAD

- Compiler Jar
- Batch only, no shell
- No Python, R
- Security
- Single point of failure





App name examples Class path spark.jobserver.WordCountExample Conte

Parameters

Name	Value
input.string	a a a b c

Add

Execute Save as... or create a New application

Key	V
a	3
c	1
b	1

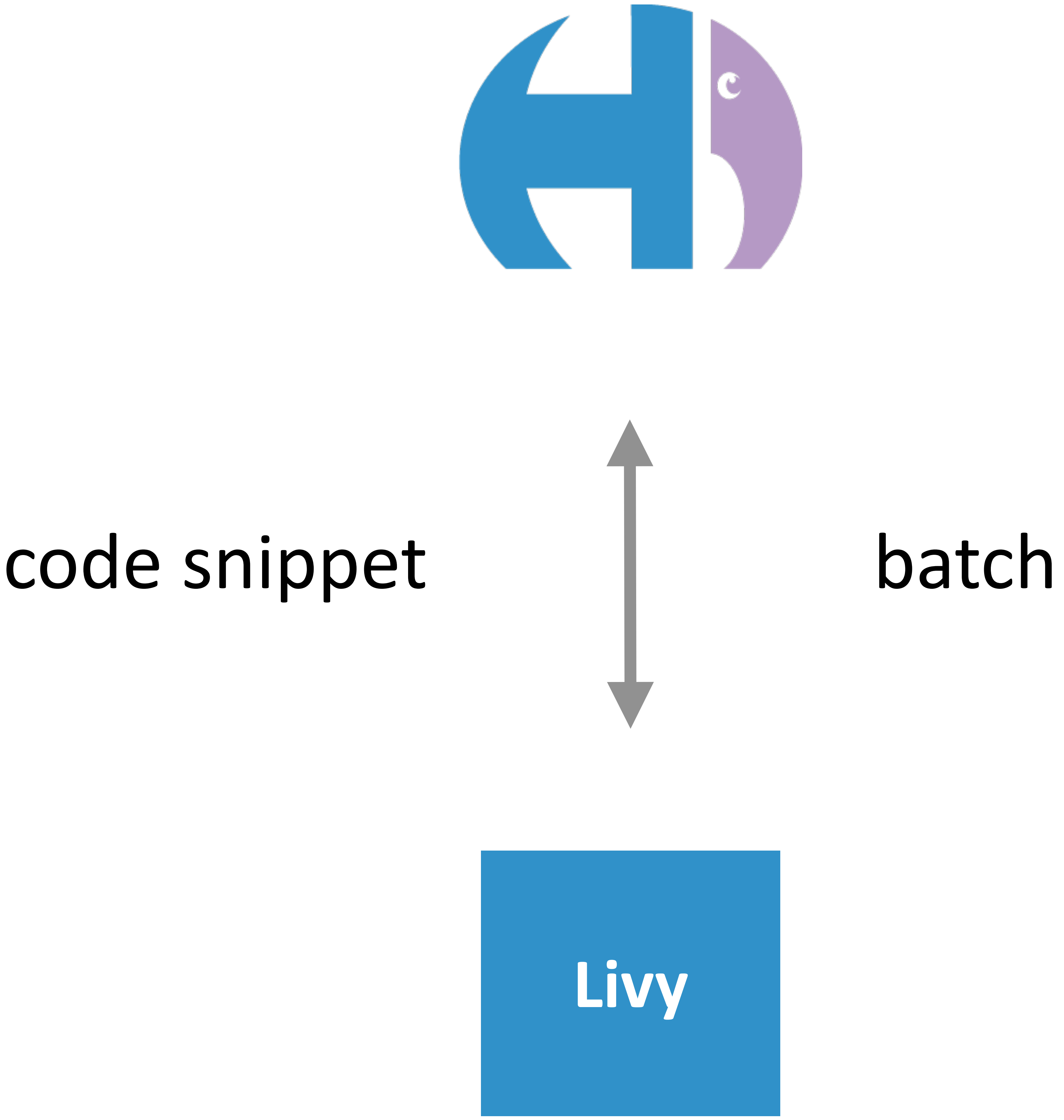
HISTORY
V3: NOTEBOOK

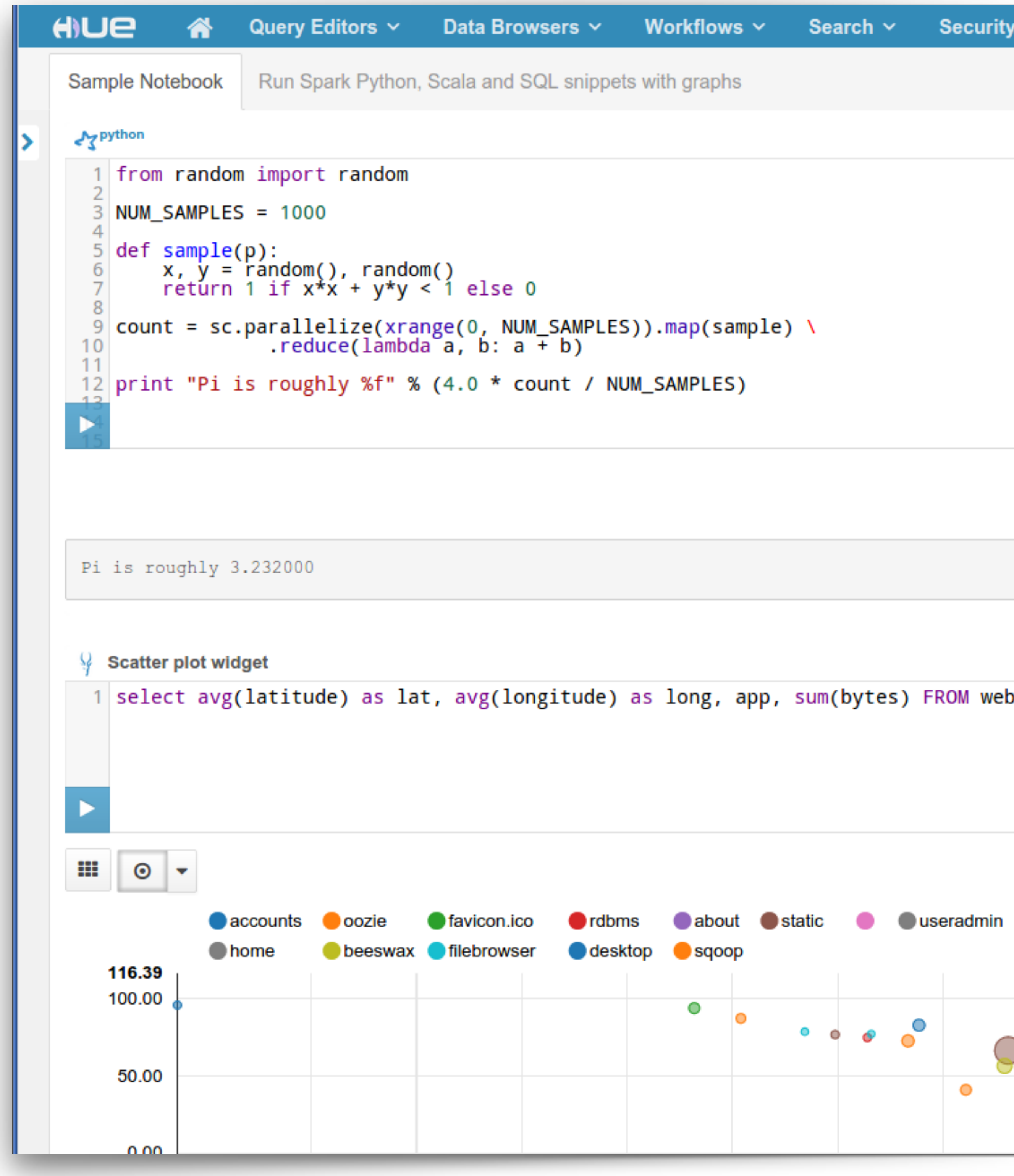
THE GOOD

- Like spark-submit / spark shells
- Scala / Python / R shells
- Jar / Python batch Jobs
- Notebook UI
- YARN

THE BAD

- Beta?





Run Spark Python, Scala and SQL snippets with graphs

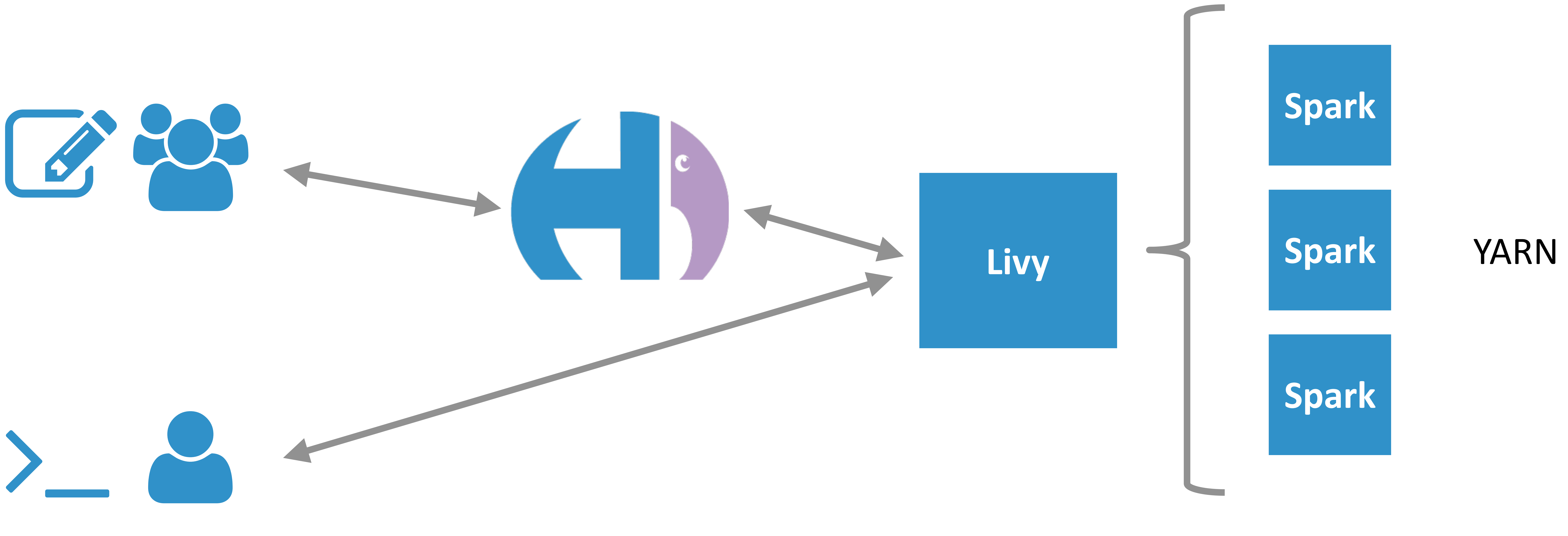
```
from random import random

NUM_SAMPLES = 1000

def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(xrange(0, NUM_SAMPLES)).map(sample) \
             .reduce(lambda a, b: a + b)

print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

```
Pi is roughly 3.232000
```

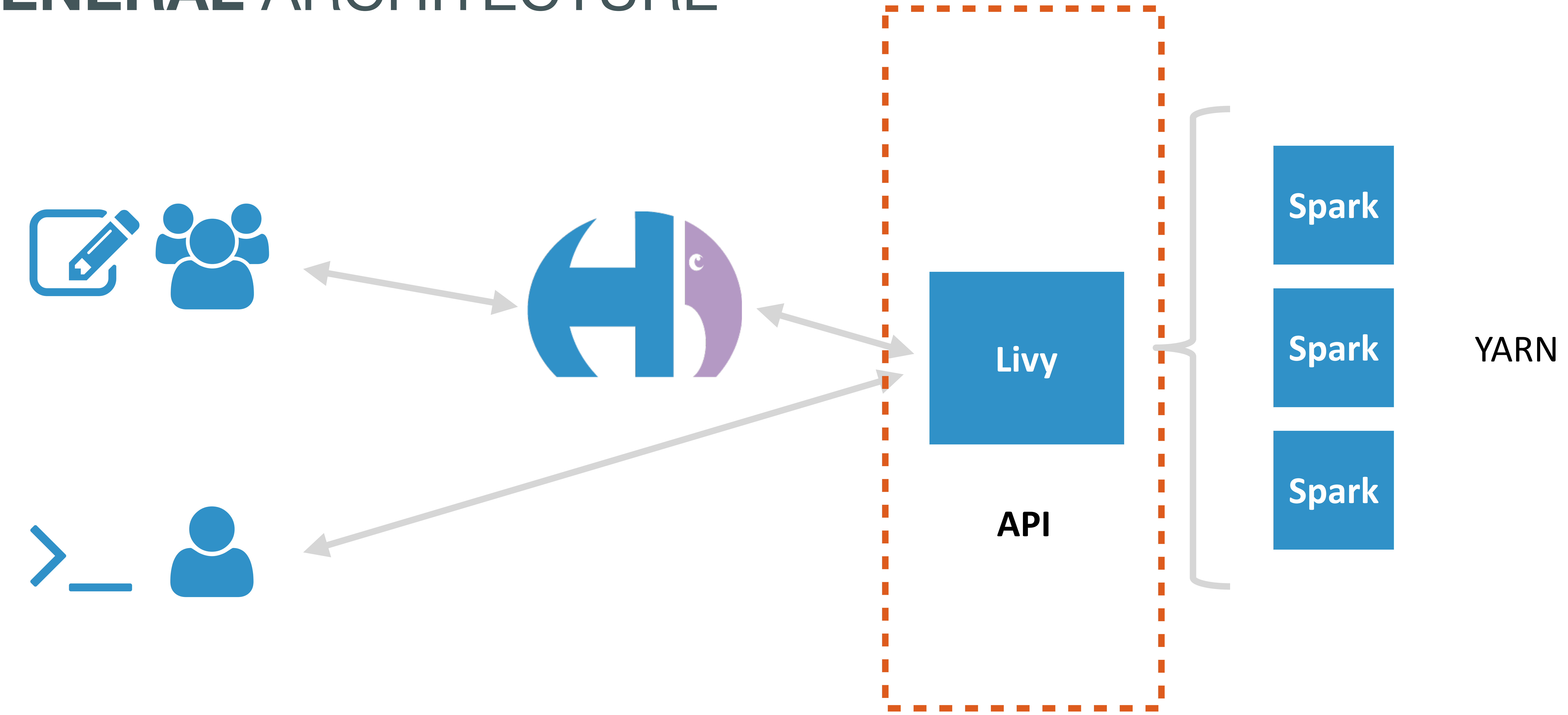
Scatter plot widget

```
select avg(latitude) as lat, avg(longitude) as long, app, sum(bytes) FROM web
```

GENERAL ARCHITECTURE



GENERAL ARCHITECTURE

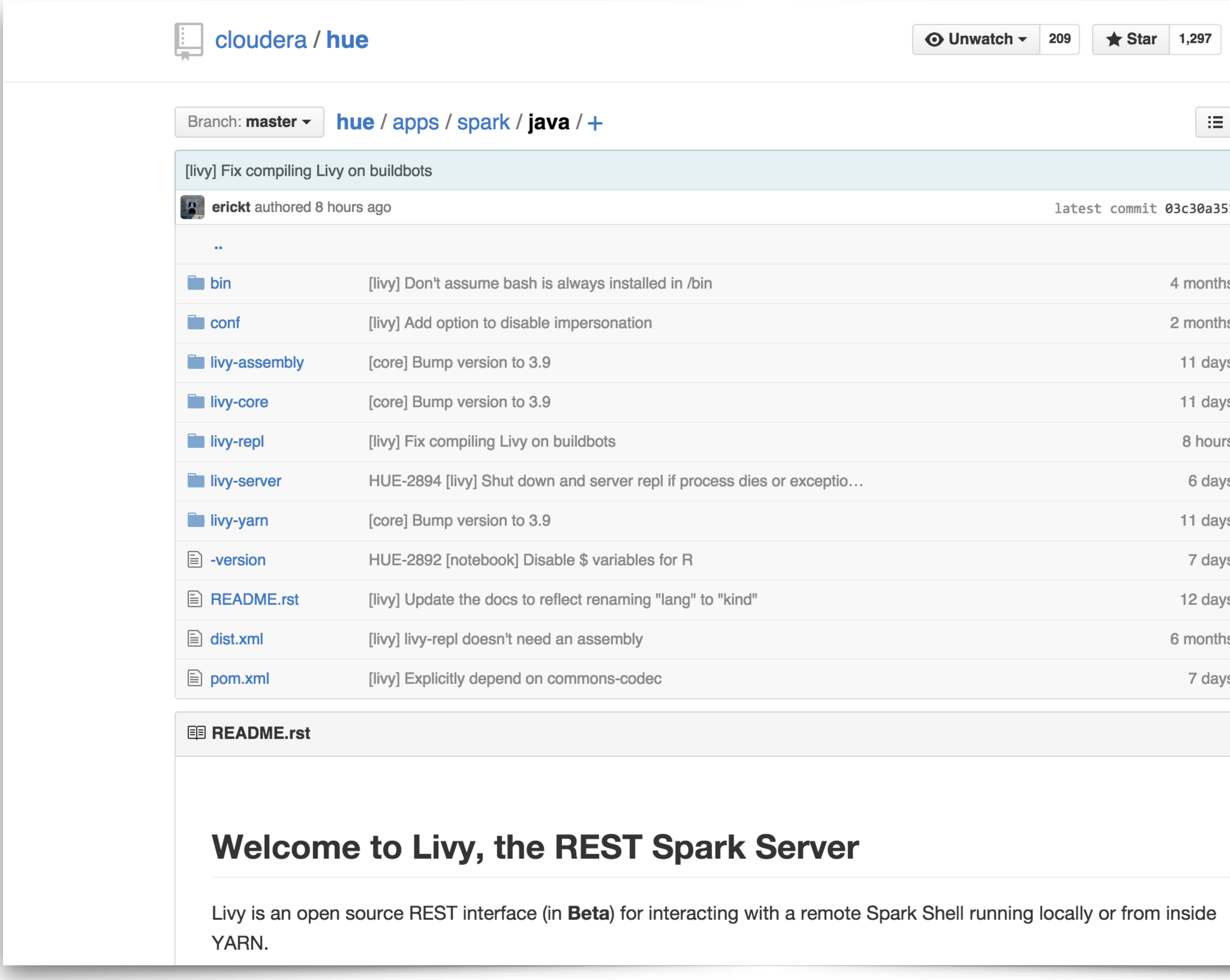


LIVY SPARK SERVER

LIVY
SPARK SERVER

- REST Web server in Scala for Spark submissions
- Interactive Shell Sessions or Batch Jobs
- Backends: Scala, Java, Python, R
- No dependency on Hue

- Open Source: https://github.com/cloudera/hue/tree/master/apps/spark/java
- Read about it: http://gethue.com/spark/



[livy] Fix compiling Livy on buildbots

erickt authored 8 hours ago latest commit 03c30a35

..

bin	[livy] Don't assume bash is always installed in /bin	4 months
conf	[livy] Add option to disable impersonation	2 months
livy-assembly	[core] Bump version to 3.9	11 days
livy-core	[core] Bump version to 3.9	11 days
livy-repl	[livy] Fix compiling Livy on buildbots	8 hours
livy-server	HUE-2894 [livy] Shut down and server repl if process dies or exceptio…	6 days
livy-yarn	[core] Bump version to 3.9	11 days
.version	HUE-2892 [notebook] Disable $ variables for R	7 days
README.rst	[livy] Update the docs to reflect renaming "lang" to "kind"	12 days
dist.xml	[livy] livy-repl doesn't need an assembly	6 months
pom.xml	[livy] Explicitly depend on commons-codec	7 days

README.rst

Welcome to Livy, the REST Spark Server

Livy is an open source REST interface (in **Beta**) for interacting with a remote Spark Shell running locally or from inside YARN.

ARCHITECTURE

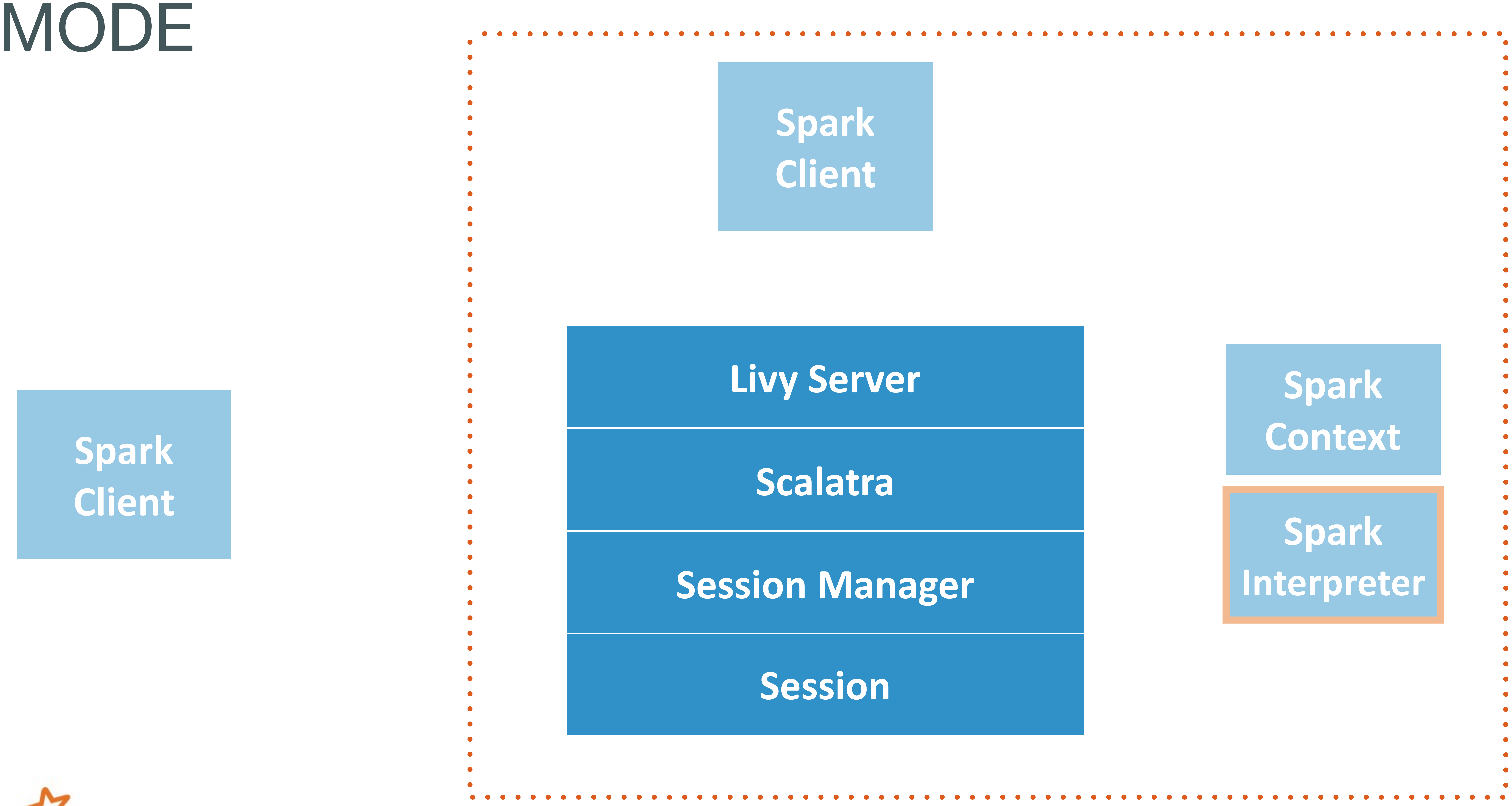
- Standard web service: wrapper around spark-submit / Spark shells
- YARN mode, Spark drivers run inside the cluster (supports crashes)
- No need to inherit any interface or compile code
- Extended to work with additional backends

LIVY WEB SERVER
ARCHITECTURE

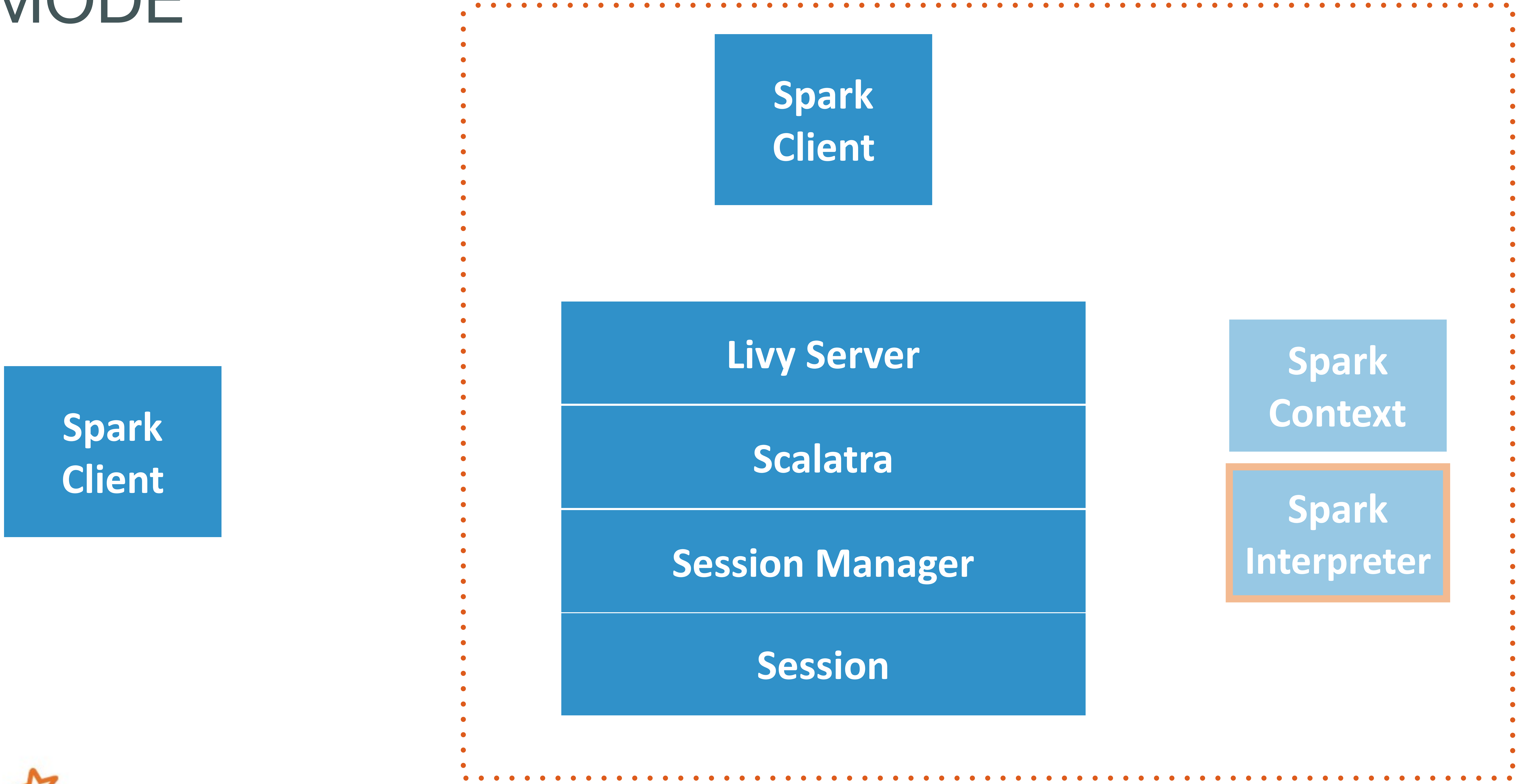
LOCAL “DEV” MODE

YARN MODE

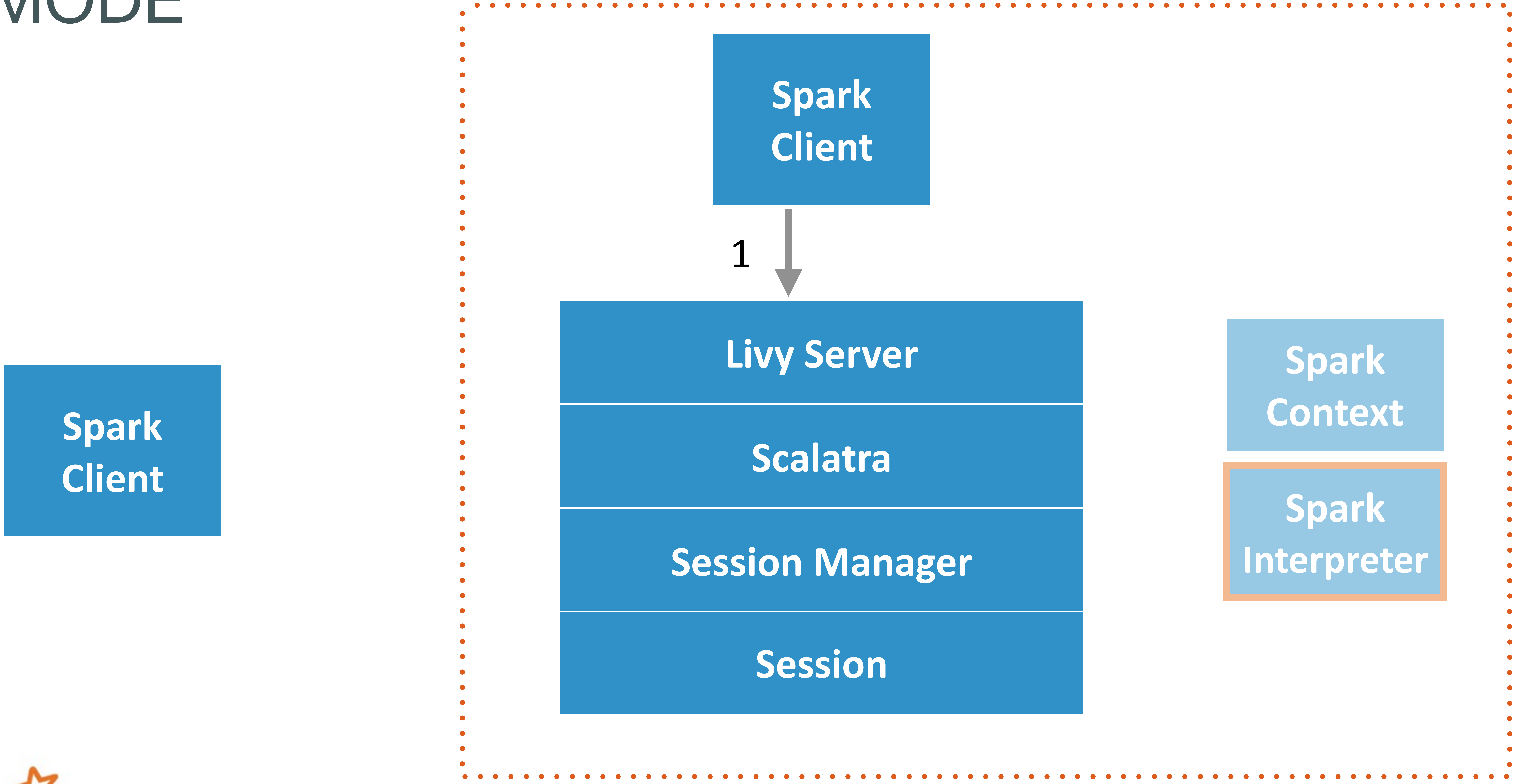
LOCAL MODE


Spark Client
Spark Client
Livy Server
Scalatra
Session Manager
Session
Spark Context
Spark Interpreter

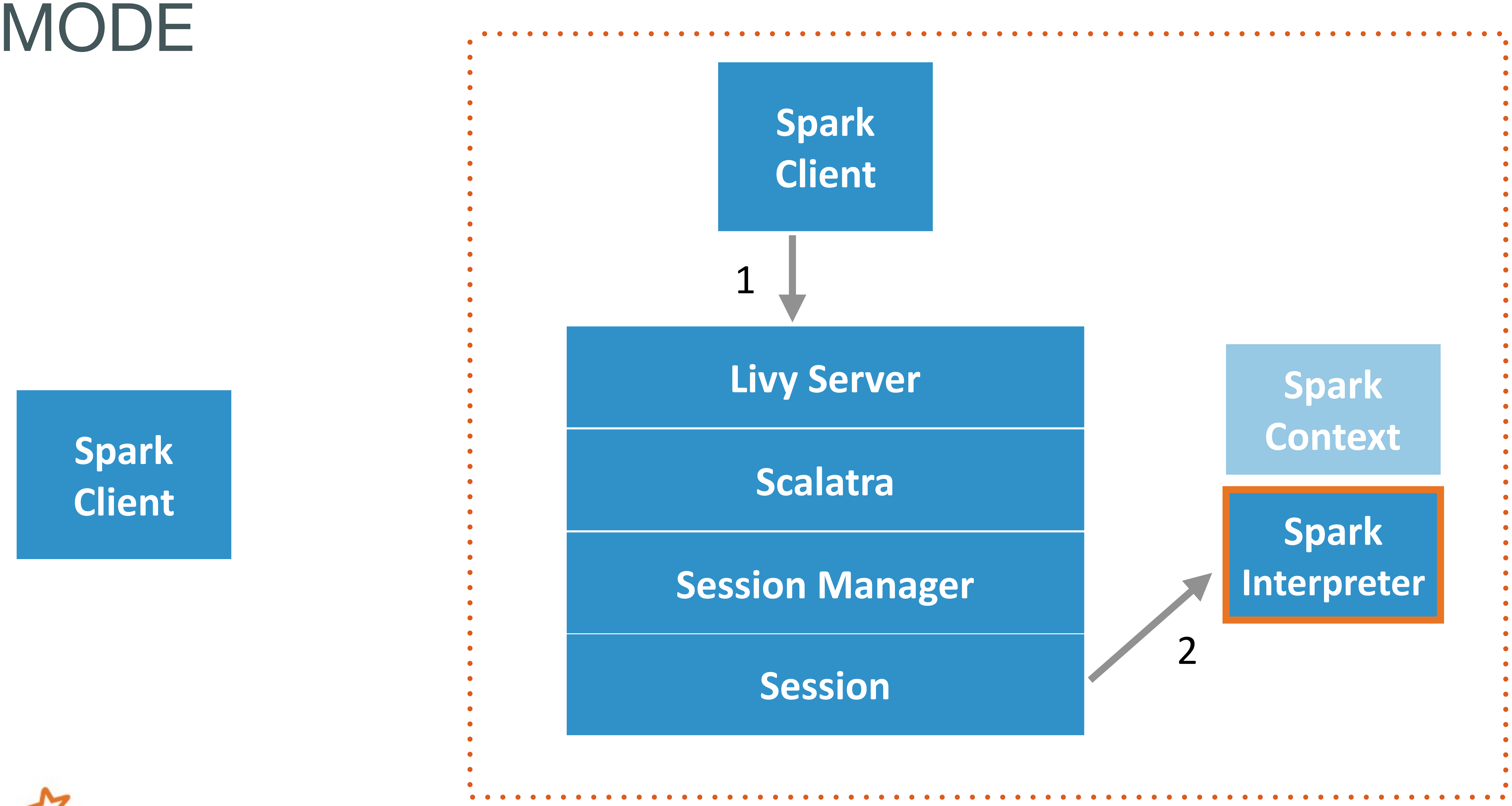
LOCAL MODE


Spark Client
Spark Client
Livy Server
Scalatra
Session Manager
Session
Spark Context
Spark Interpreter

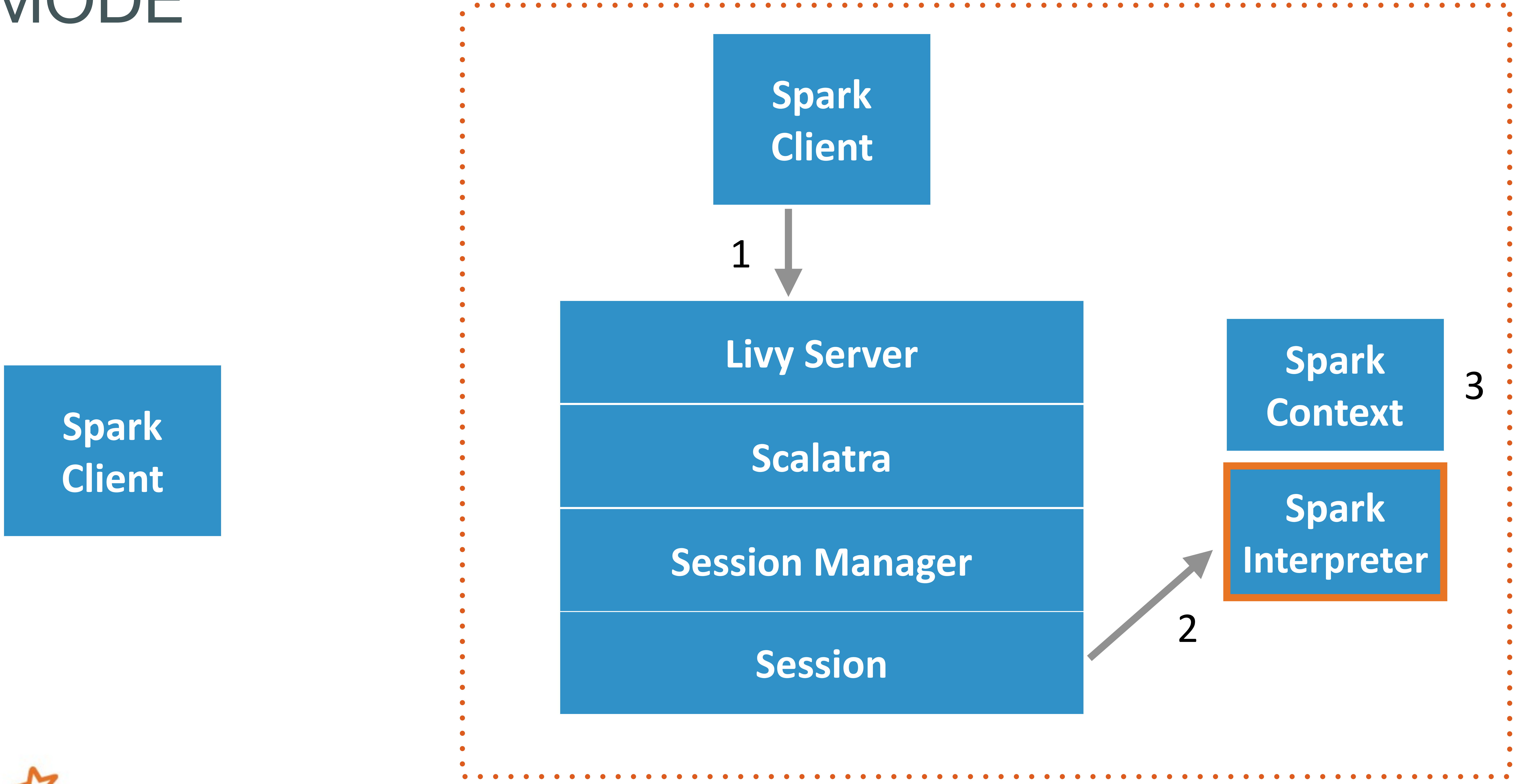
LOCAL MODE


Spark Client
Spark Client
1
Livy Server
Scalatra
Session Manager
Session
Spark Context
Spark Interpreter

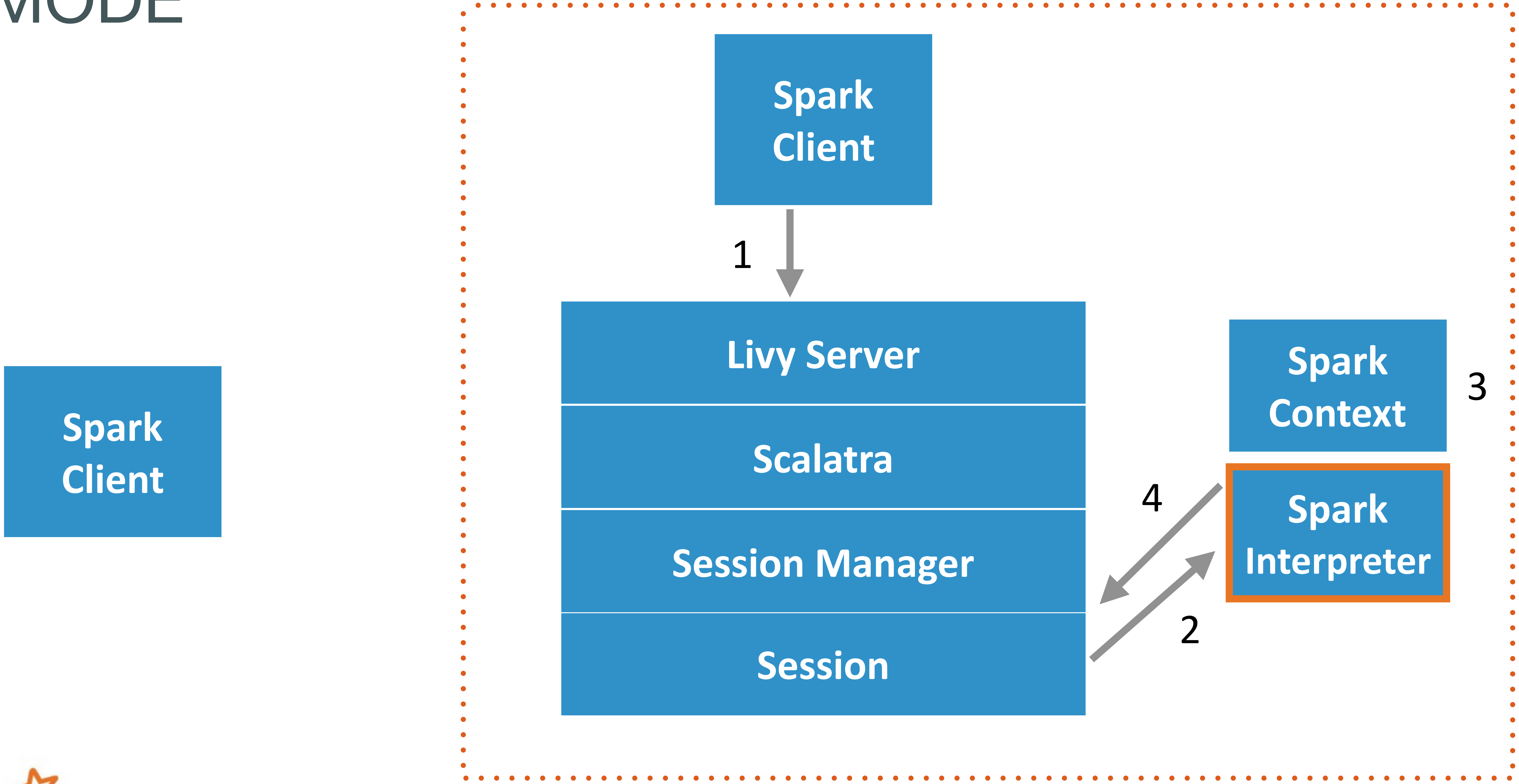
LOCAL MODE


Spark Client
Spark Client
1
Livy Server
Scalatra
Session Manager
Session
2
Spark Context
Spark Interpreter

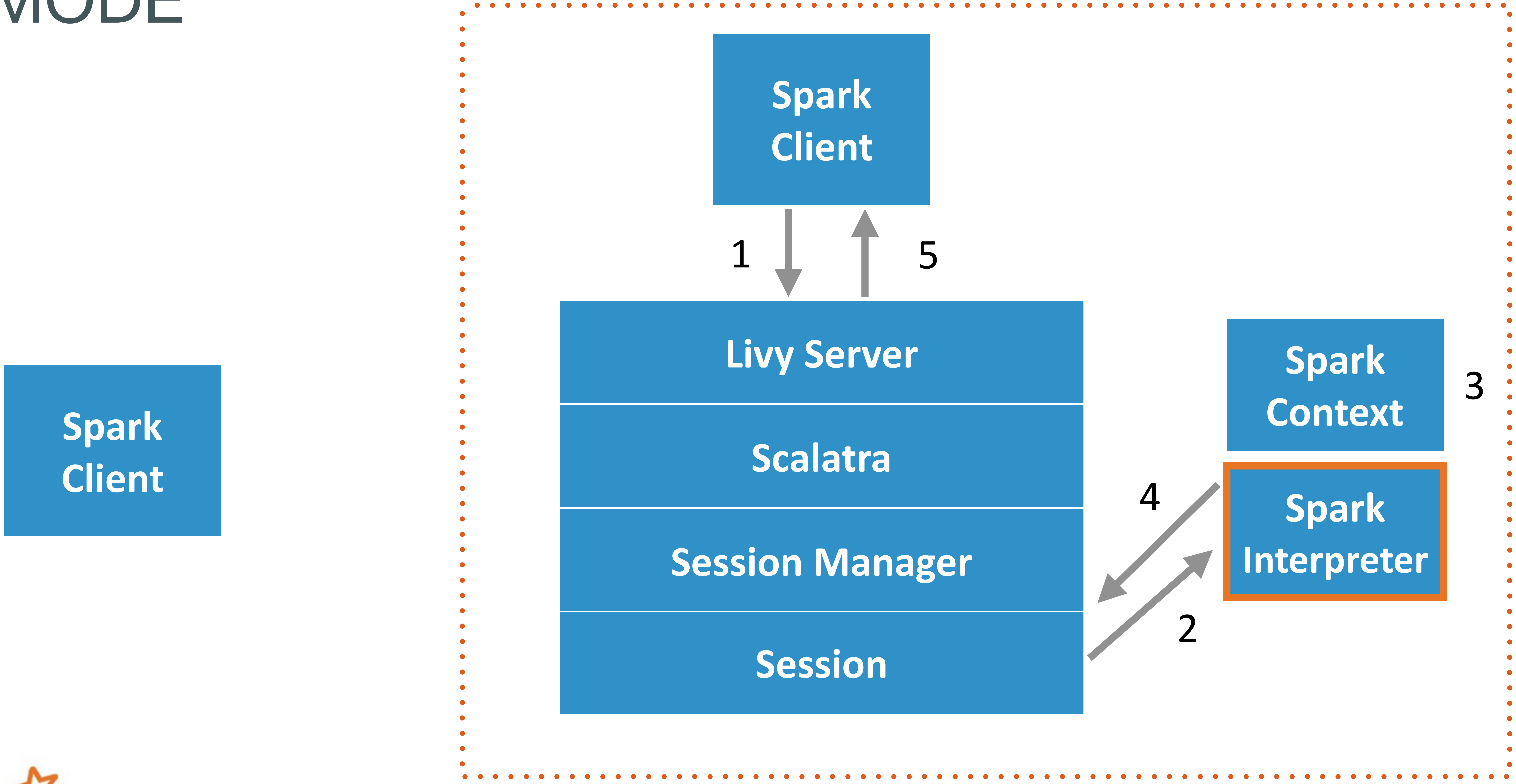
LOCAL MODE


Spark Client
Spark Client
1
Livy Server
Scalatra
Session Manager
Session
2
Spark Context
3
Spark Interpreter

LOCAL MODE


Spark Client
Spark Client
1
Livy Server
Scalatra
Session Manager
Session
Spark Context
3
4
Spark Interpreter
2

LOCAL MODE

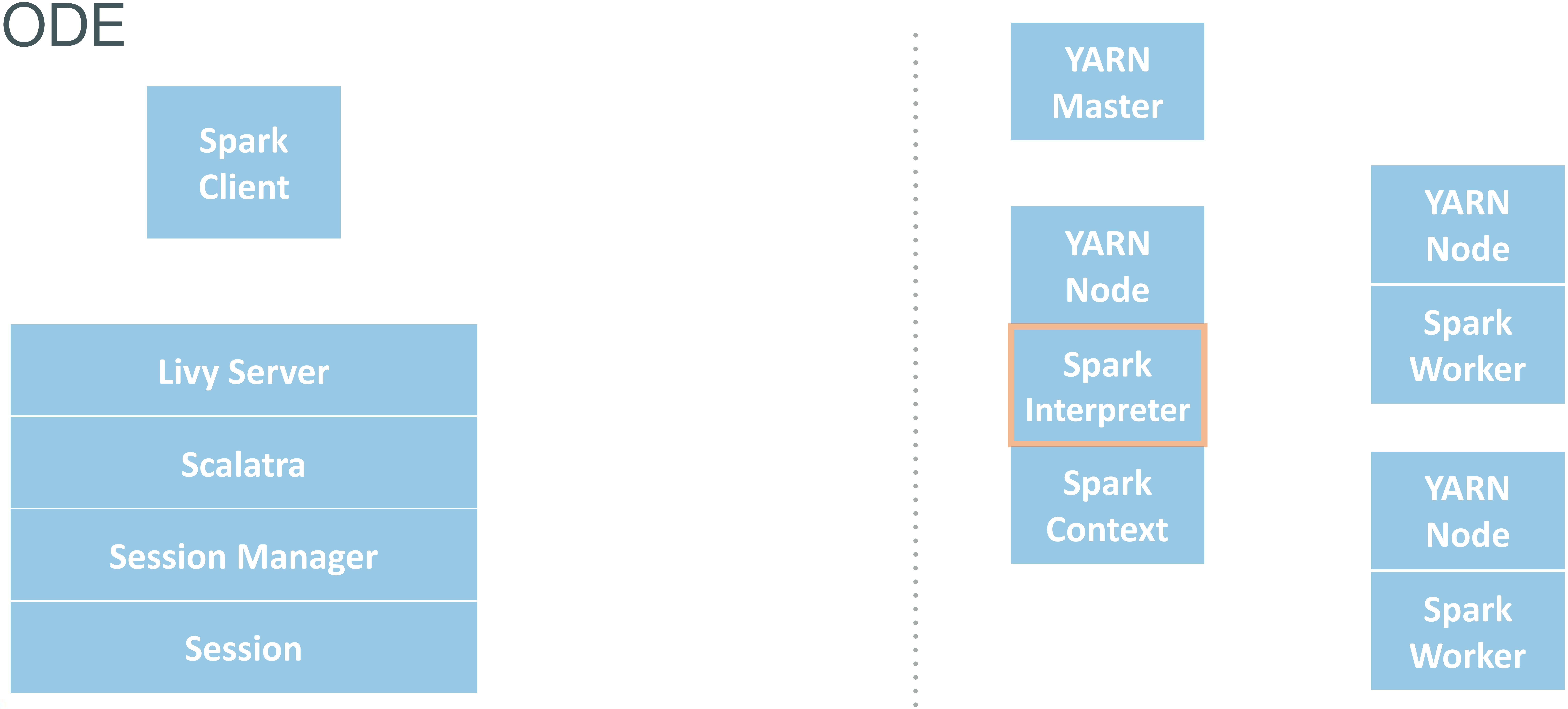

Spark Client
Spark Client
1
5
Livy Server
Scalatra
Session Manager
Session
Spark Context
3
4
Spark Interpreter
2

YARN-CLUSTER MODE

PRODUCTION

SCALABLE

YARN-CLUSTER MODE



Spark Client

Livy Server

Scalatra

Session Manager

Session

YARN Master

YARN Node

Spark Interpreter

Spark Context

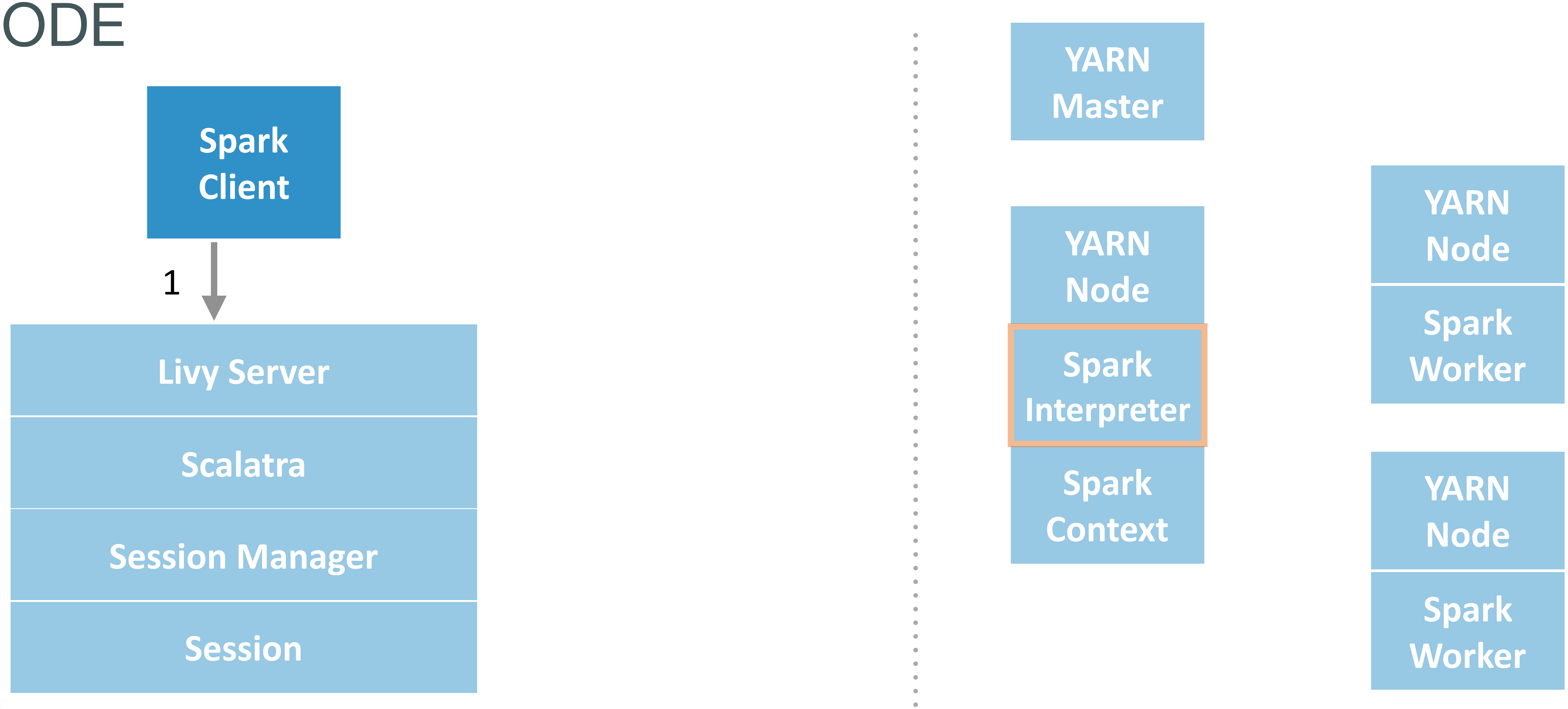
YARN Node

Spark Worker

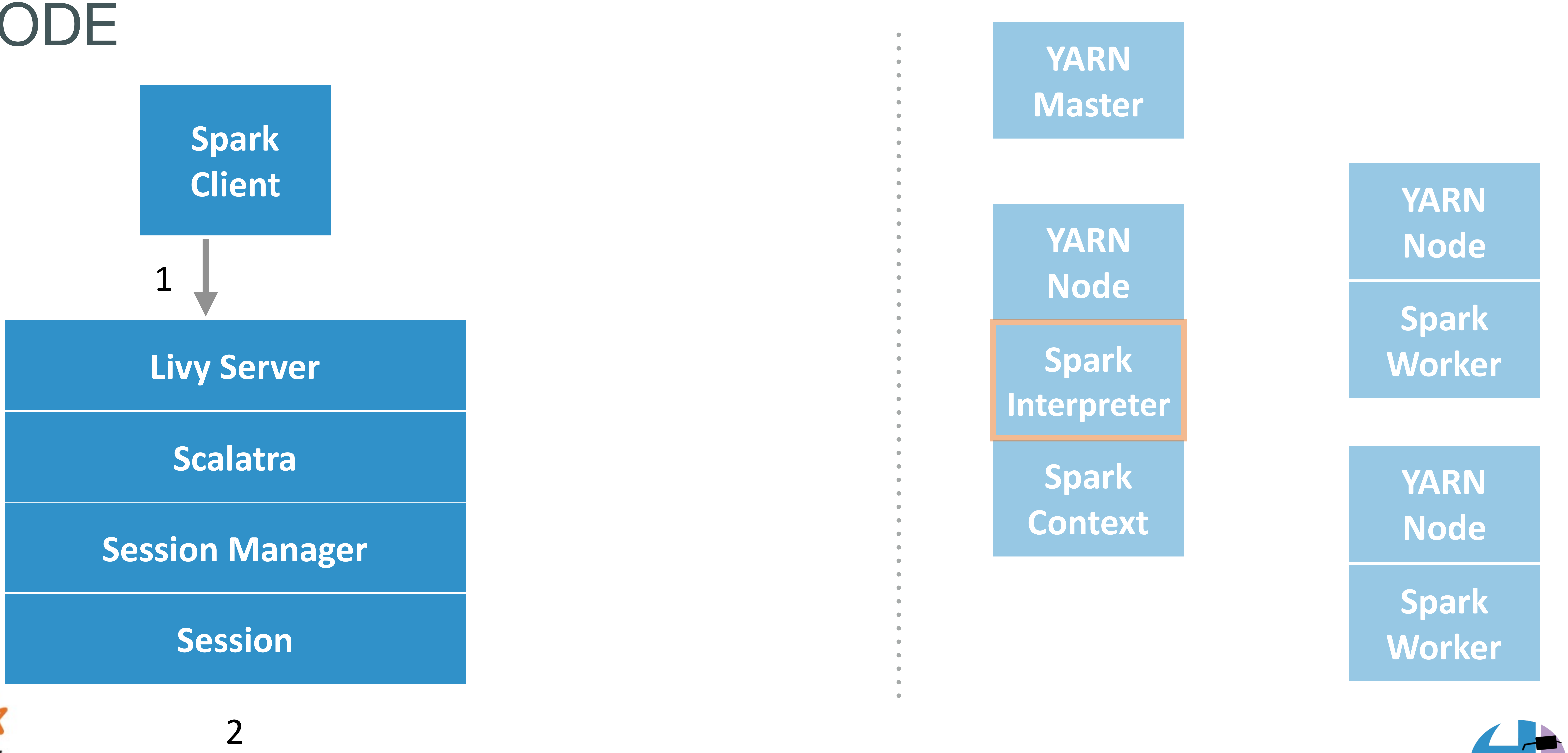
YARN Node

Spark Worker

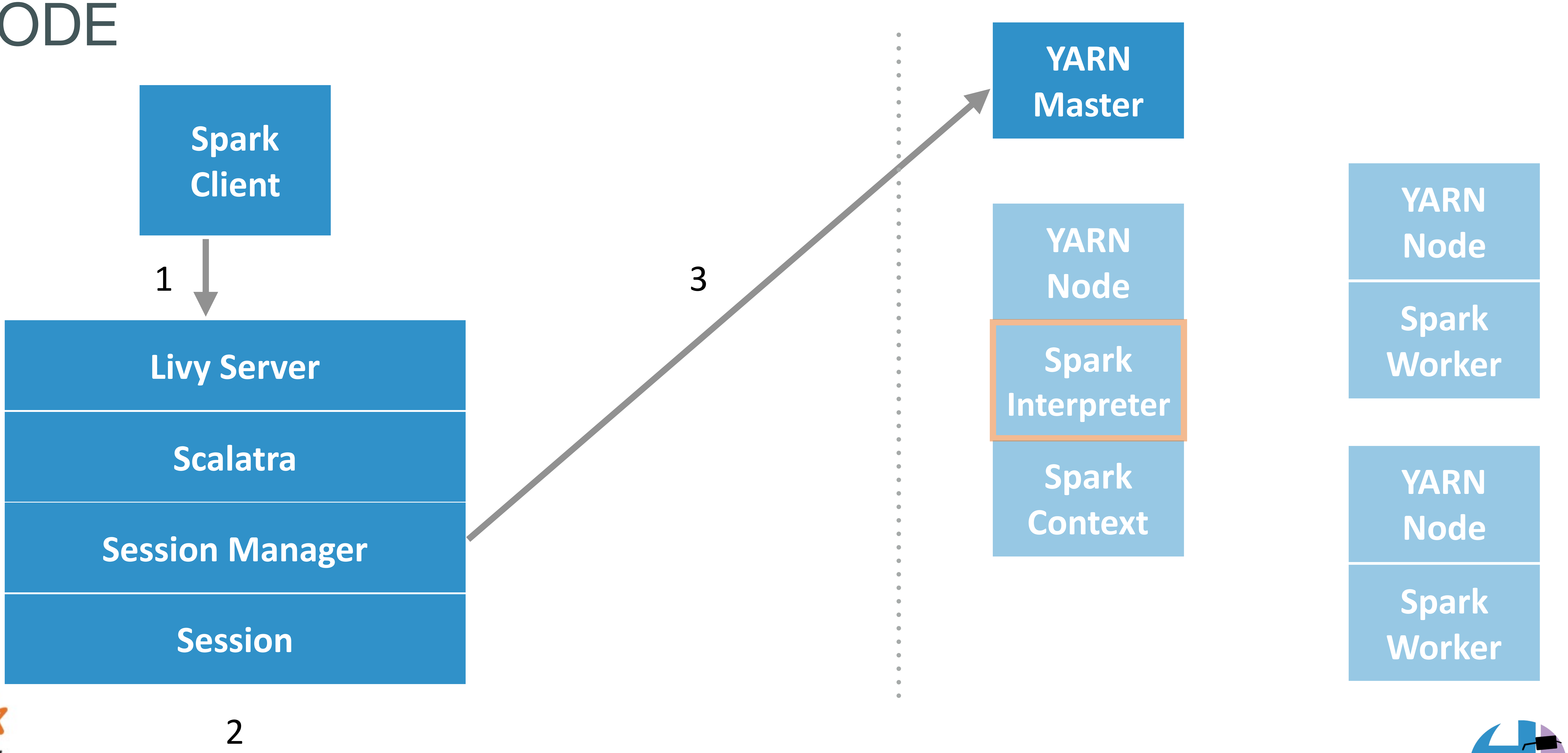
YARN-CLUSTER
MODE



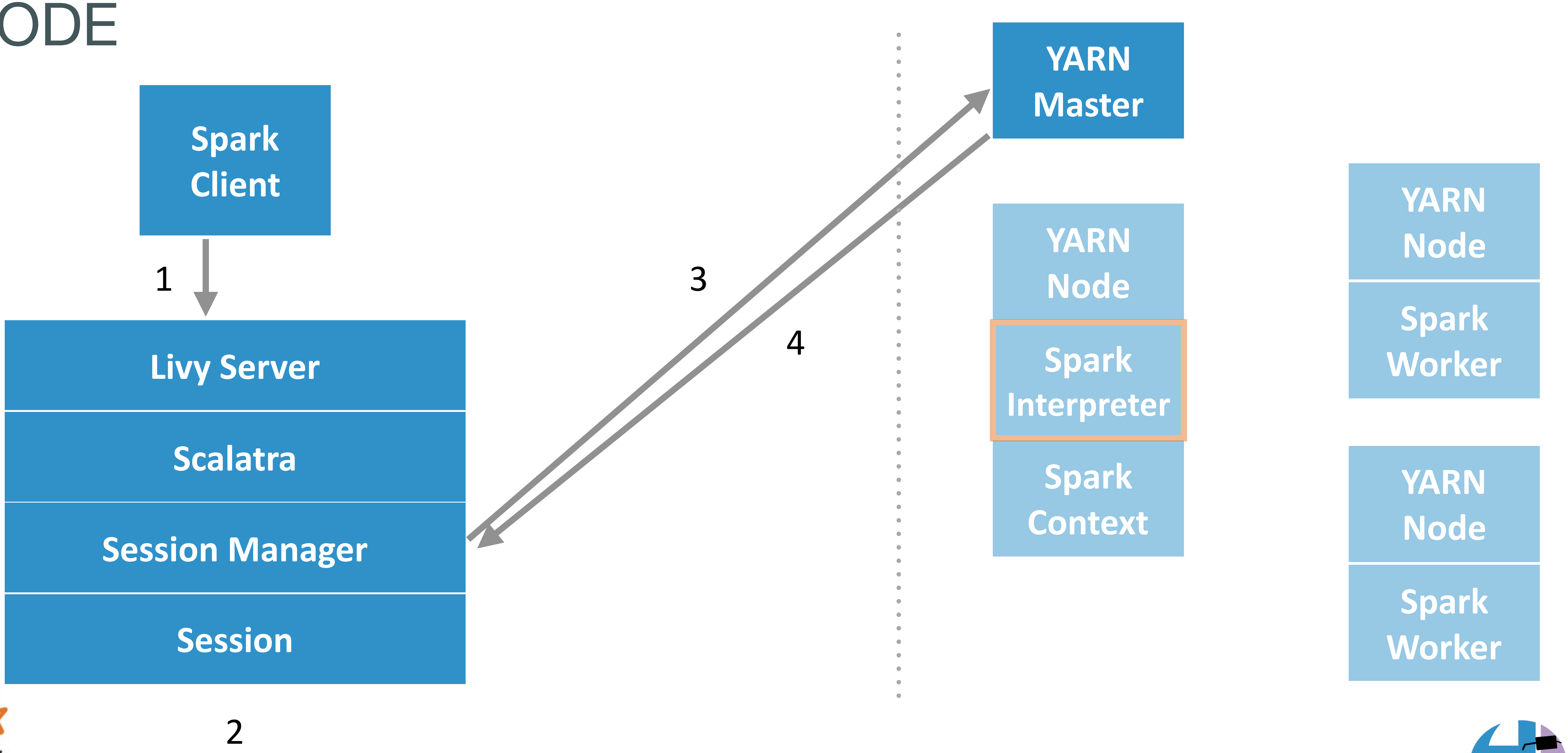
YARN-CLUSTER MODE



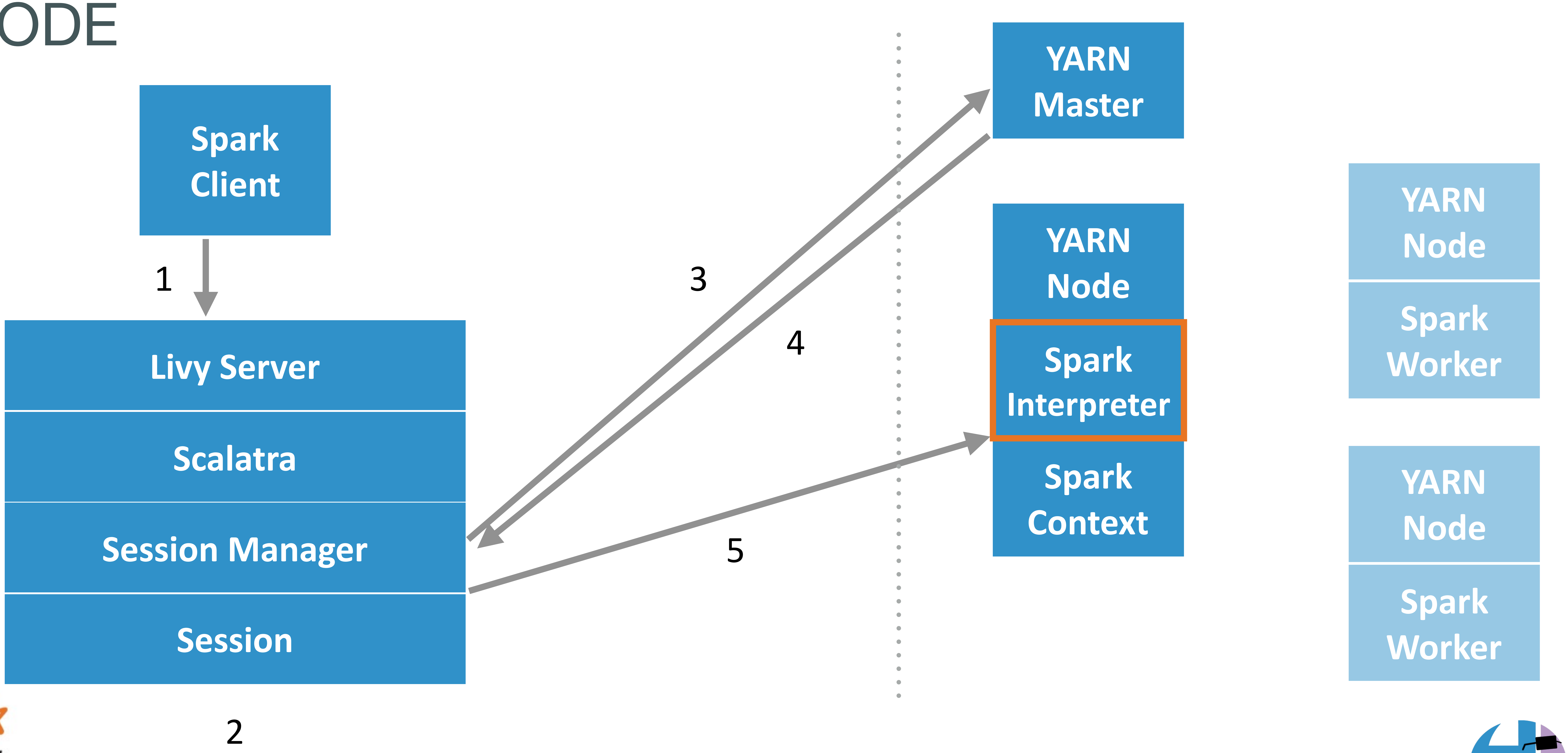
YARN-CLUSTER MODE



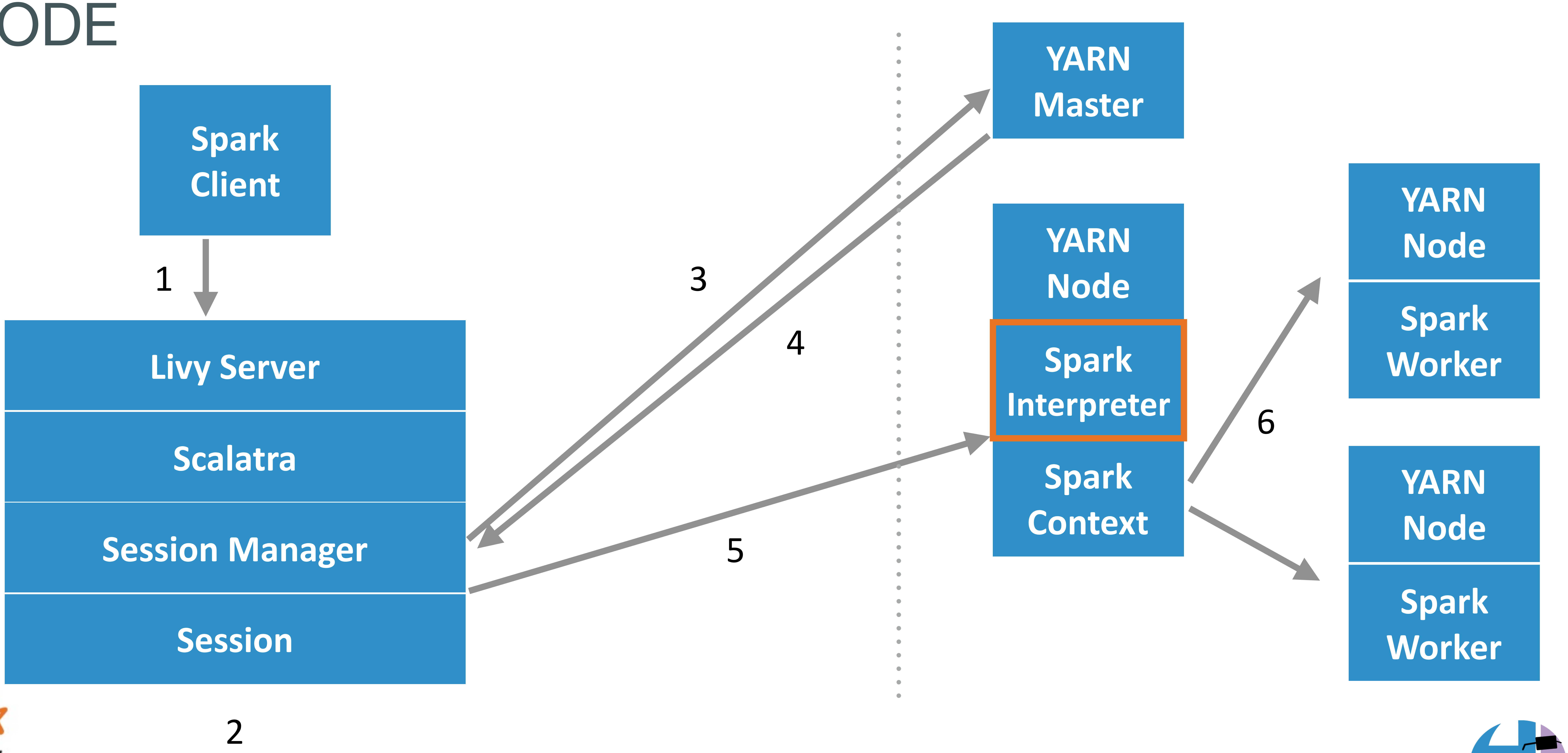
YARN-CLUSTER MODE



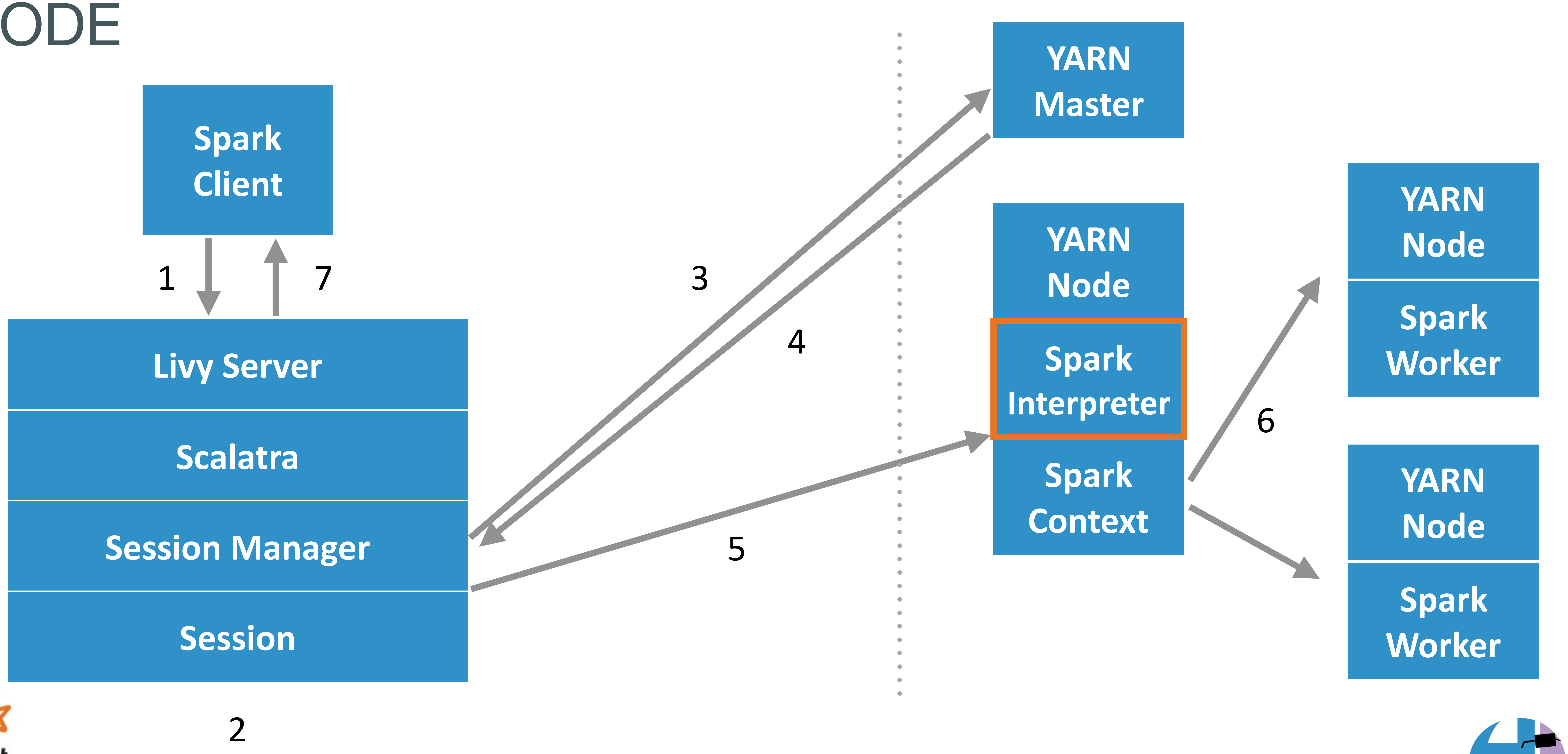
YARN-CLUSTER MODE



YARN-CLUSTER MODE



YARN-CLUSTER MODE

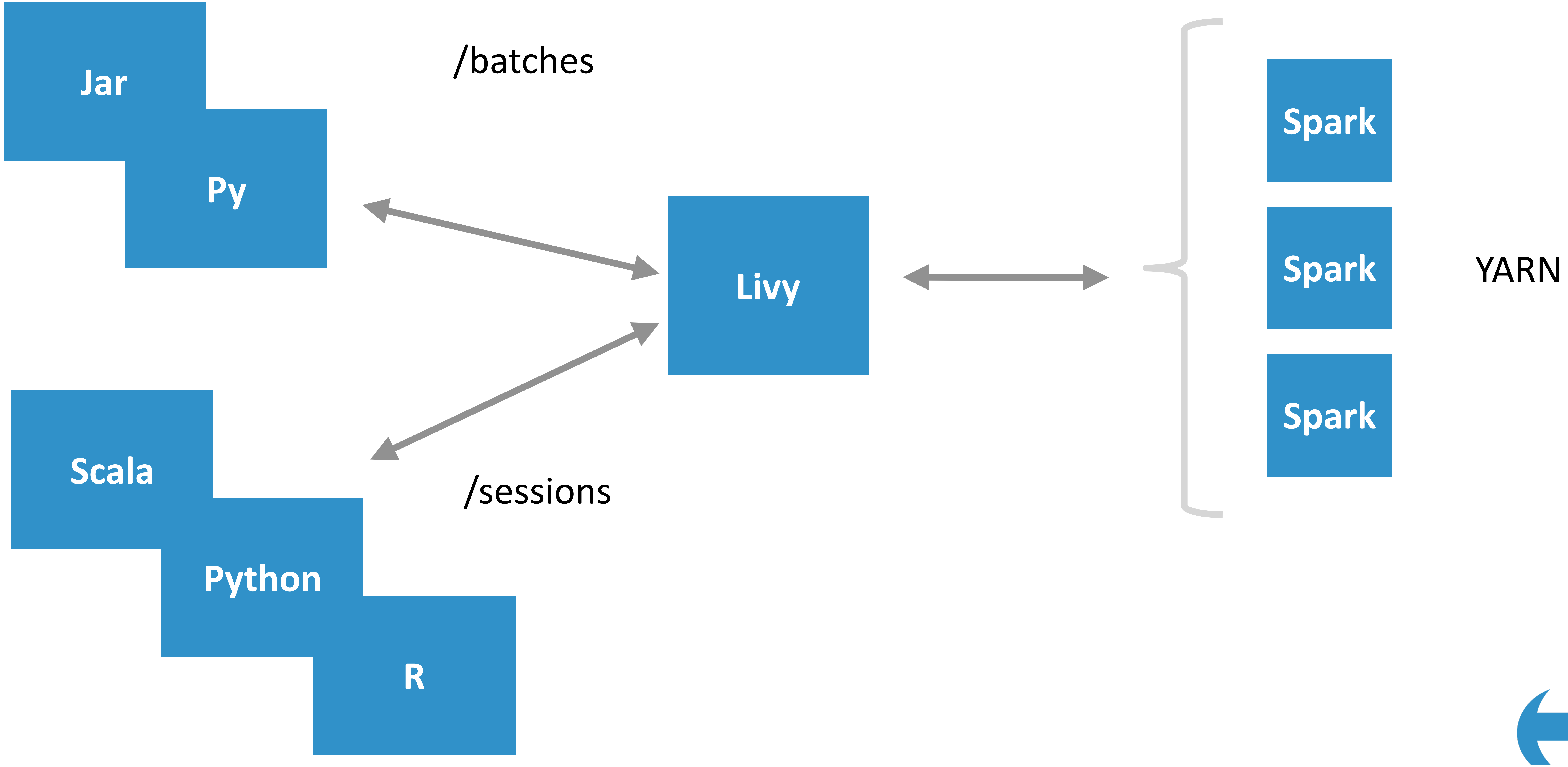


SESSION CREATION AND EXECUTION

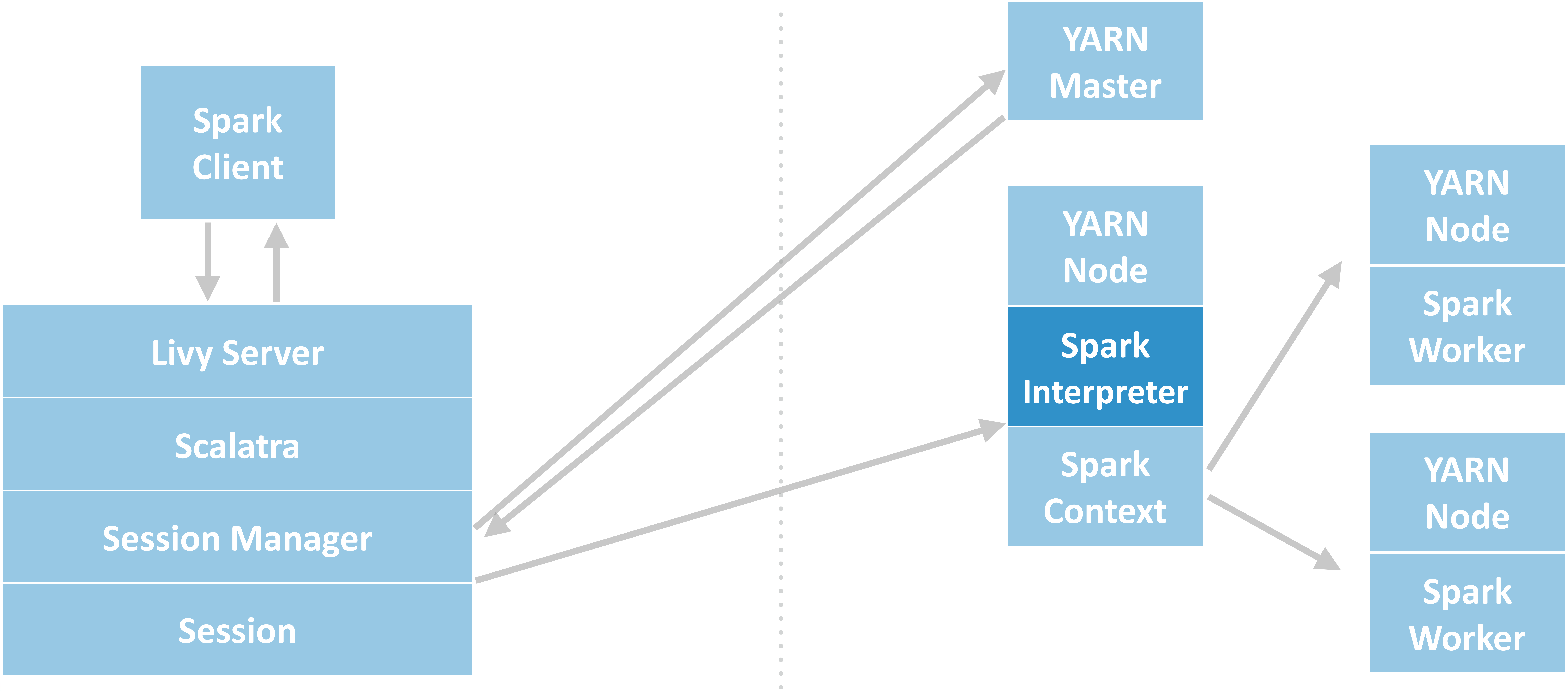
```
% curl -XPOST localhost:8998/sessions \
  -d '{"kind": "spark"}'
{
  "id": 0,
  "kind": "spark",
  "log": [...],
  "state": "idle"
}
```

```
% curl -XPOST localhost:8998/sessions/0/statements -d '{"code": "1+1"}'
{
  "id": 0,
  "output": {
    "data": { "text/plain": "res0: Int = 2" },
    "execution_count": 0,
    "status": "ok"
  },
  "state": "available"
}
```

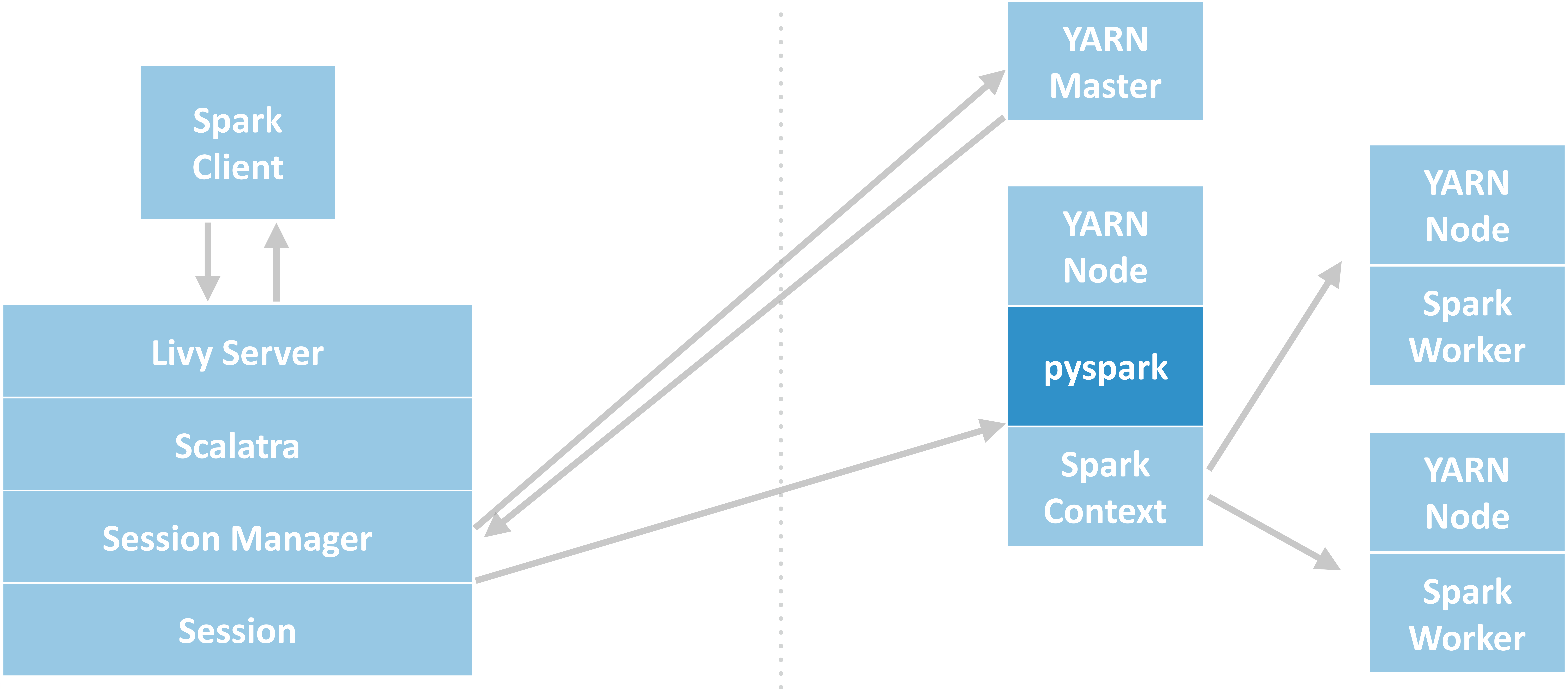
BATCH OR INTERACTIVE



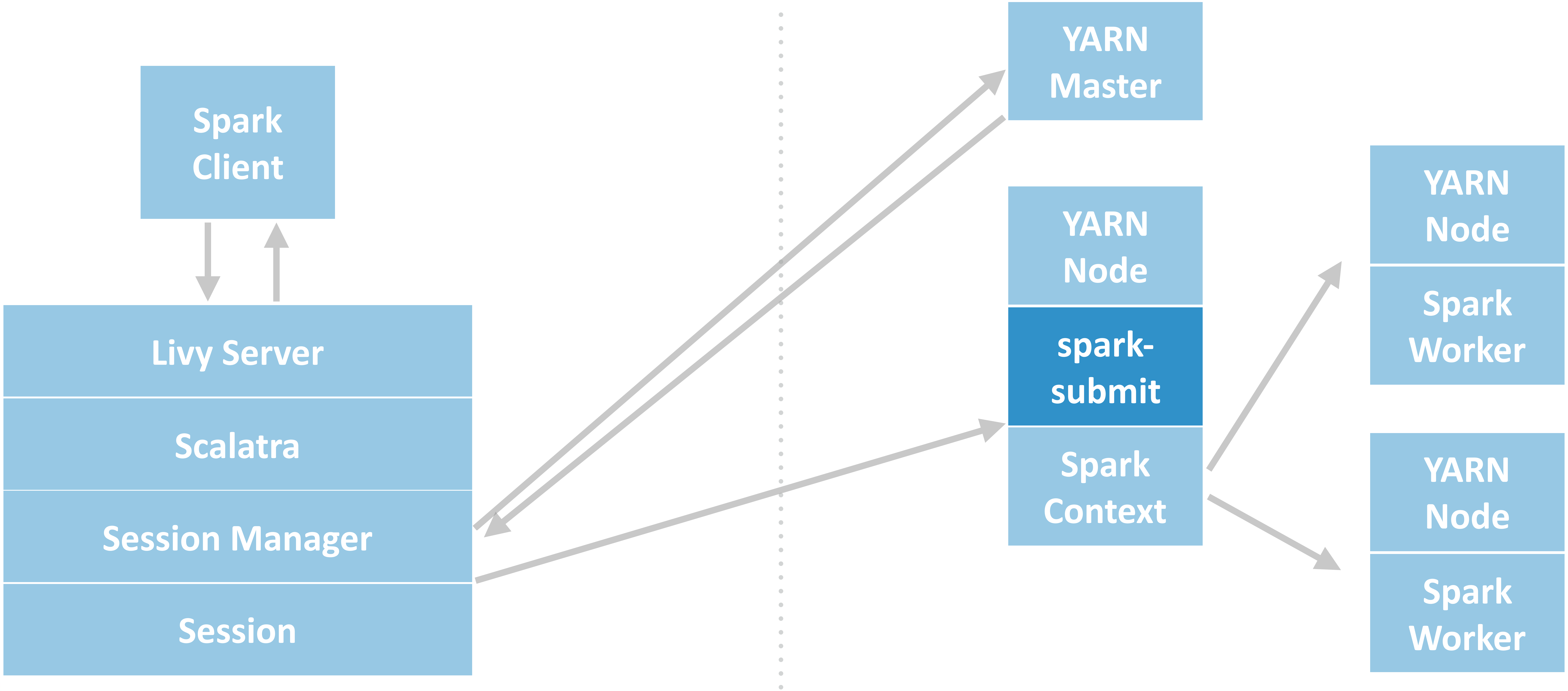
SHELL OR BATCH?



SHELL


Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Master
YARN Node
pyspark
Spark Context
YARN Node
Spark Worker
YARN Node
Spark Worker

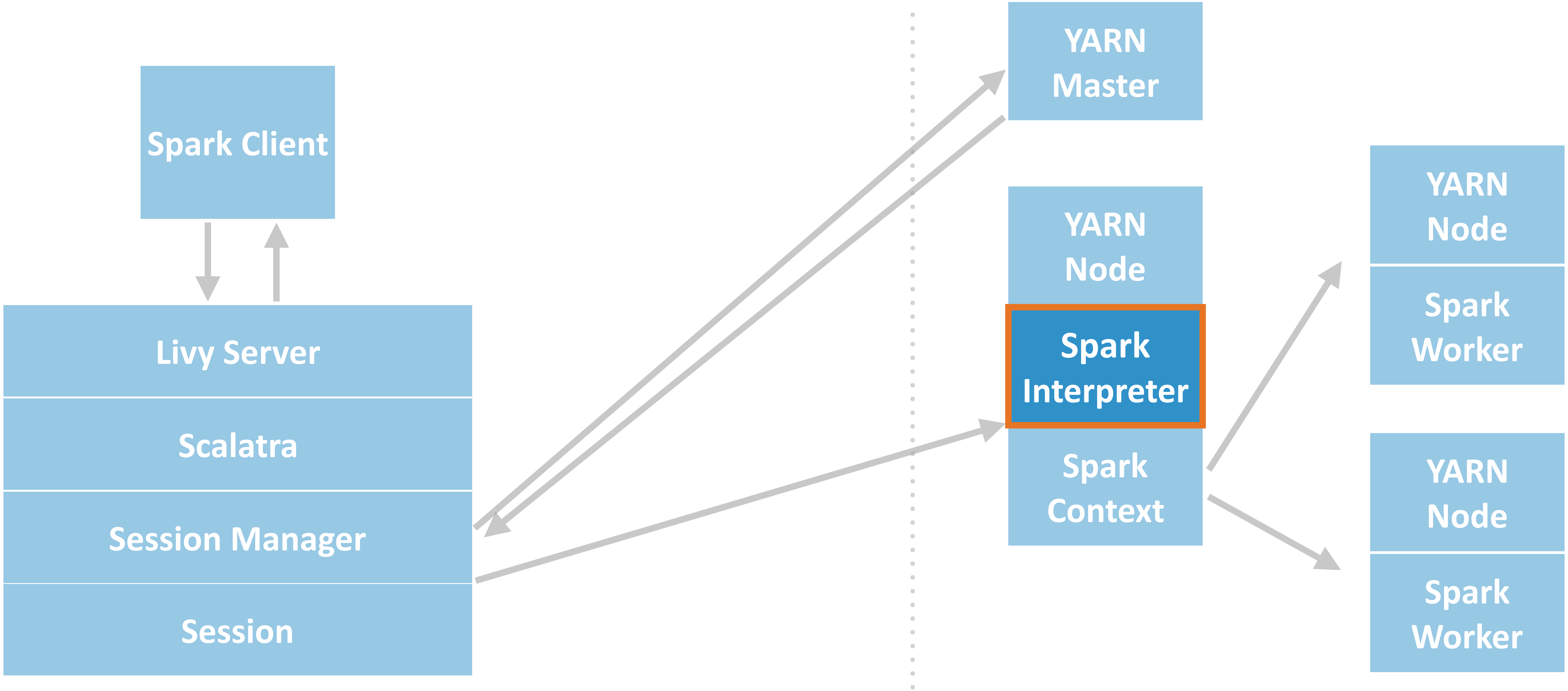
BATCH


Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Master
YARN Node
spark-submit
Spark Context
YARN Node
Spark Worker
YARN Node
Spark Worker

LIVY INTERPRETERS

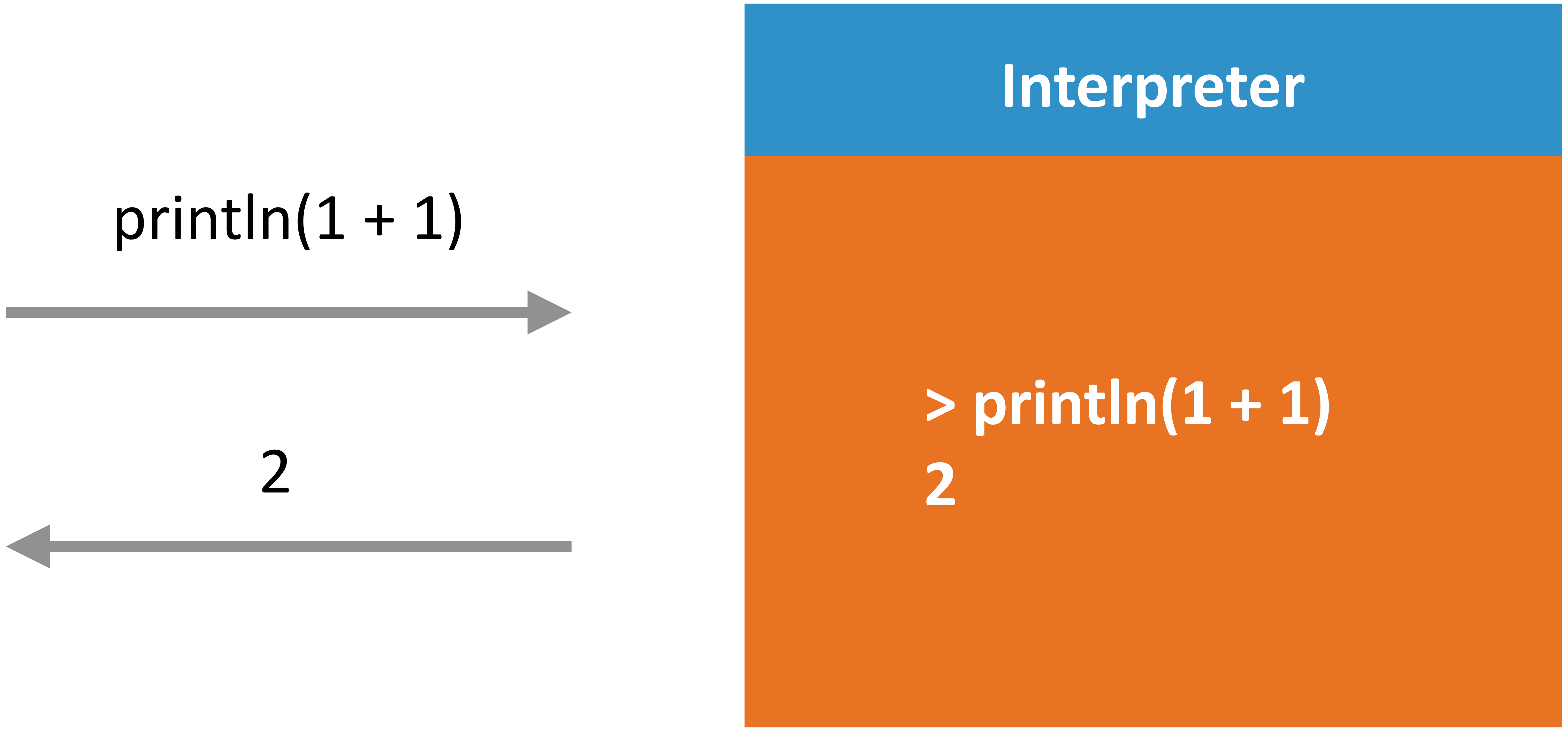
Scala, Python, R…

REMEMBER?


Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Master
YARN Node
Spark Interpreter
Spark Context
YARN Node
Spark Worker
YARN Node
Spark Worker

INTERPRETERS

- Pipe stdin/stdout to a running shell
- Execute the code / send to Spark workers
- Perform magic operations
- One interpreter per language
- “Swappable” with other kernels (python, spark..)



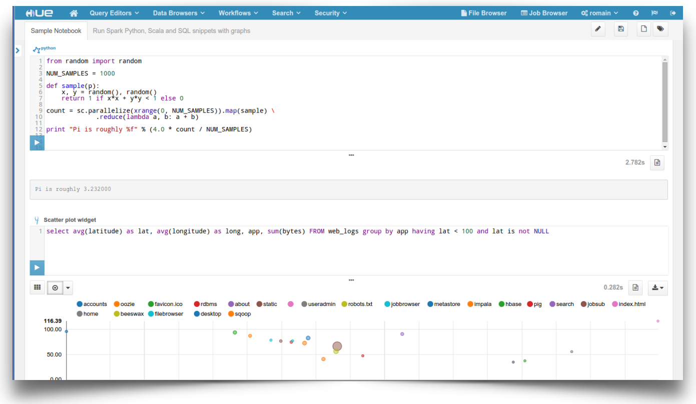
println(1 + 1)

2

Interpreter

```
> println(1 + 1)
2
```

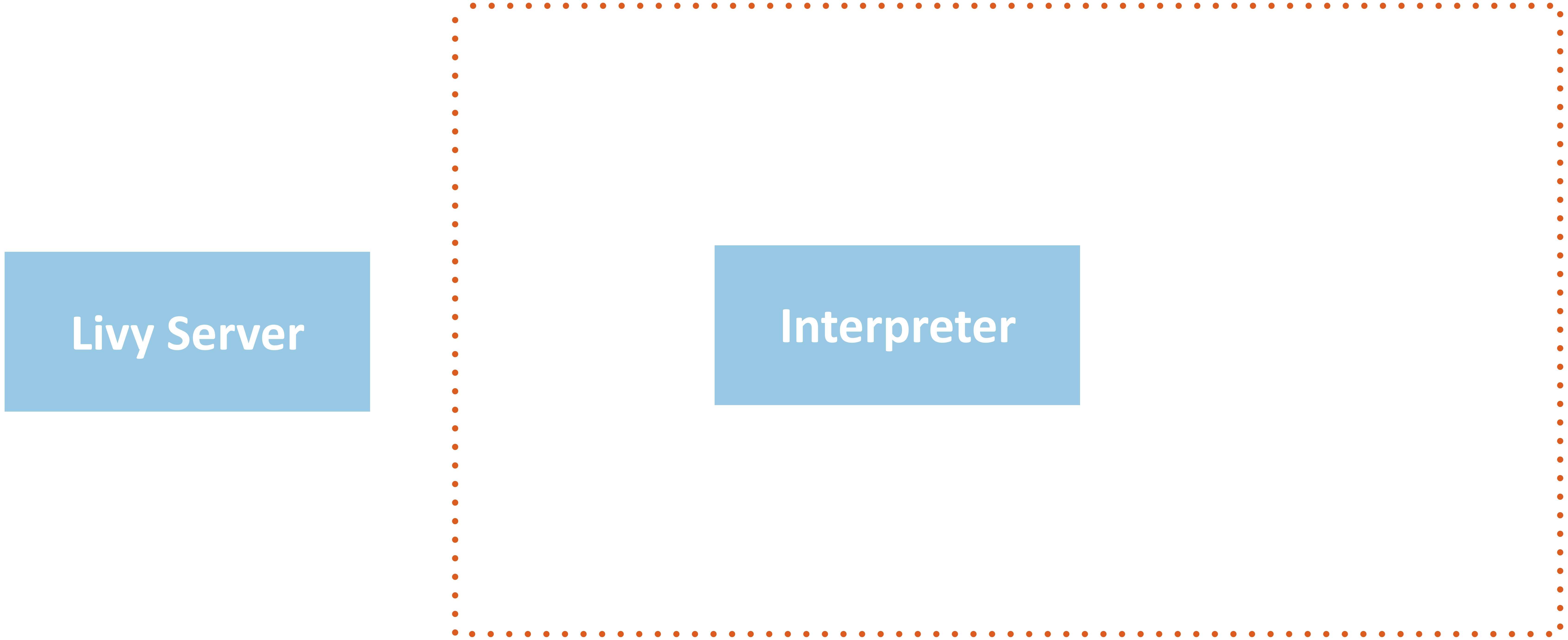
INTERPRETER FLOW



```
from random import random

NUM_SAMPLES = 1000

def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(xrange(0, NUM_SAMPLES)).map(sample) \
             .reduce(lambda a, b: a + b)

print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

```
Pi is roughly 3.232000
```

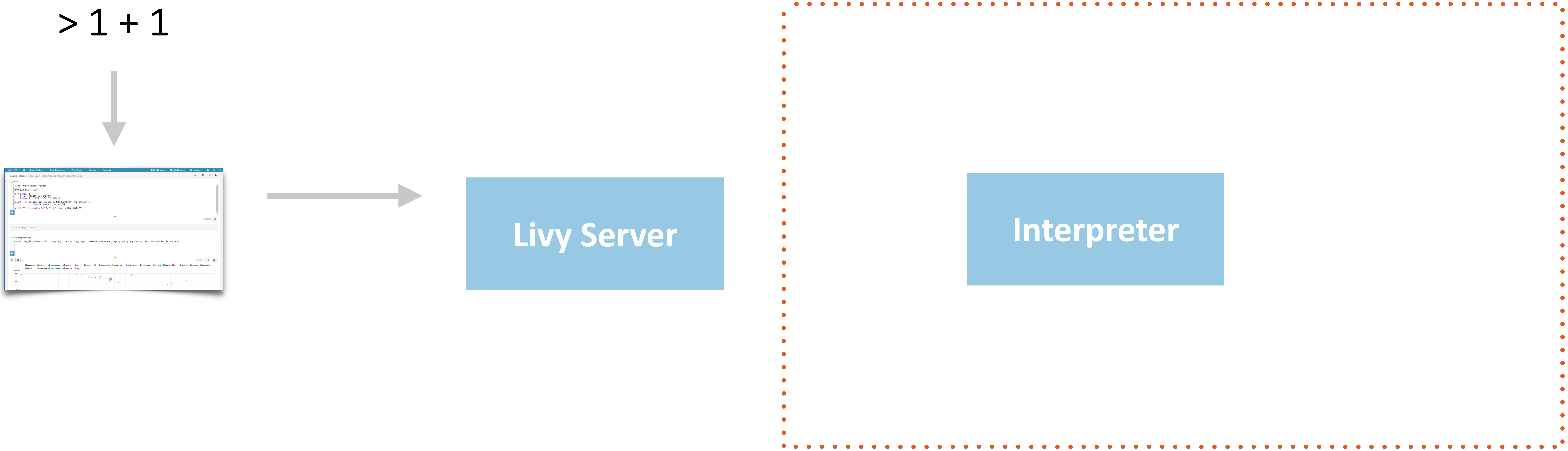
```
select avg(latitude) as lat, avg(longitude) as long, app, sum(bytes) FROM web_logs group by app having lat < 100 and lat is not NULL
```



Livy Server

Interpreter

INTERPRETER FLOW

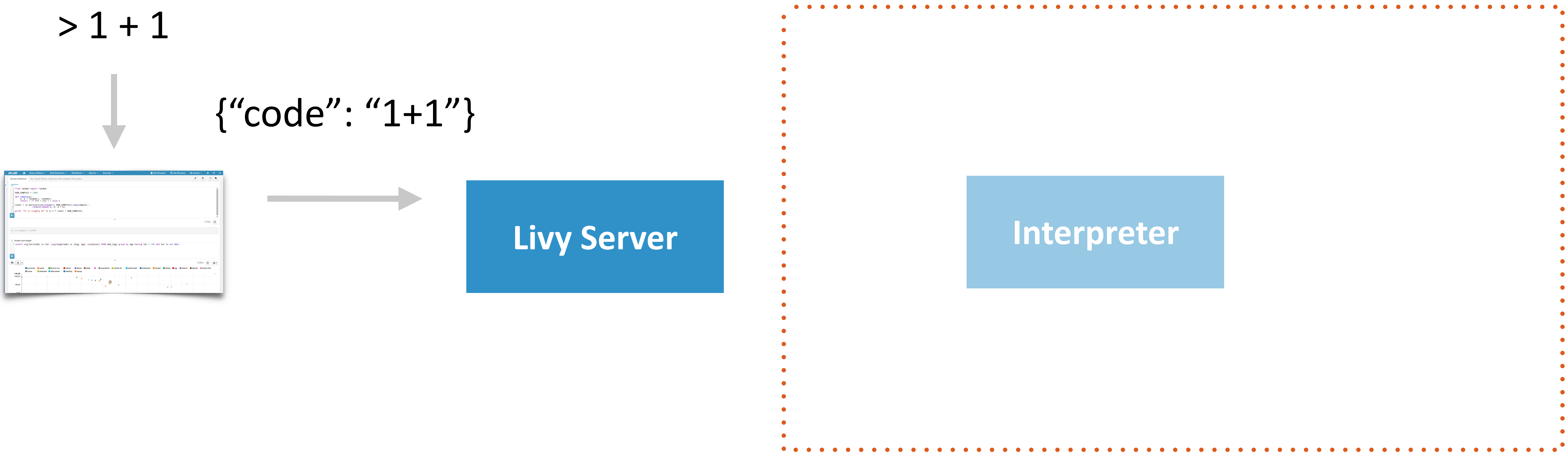


> 1 + 1

Livy Server

Interpreter

INTERPRETER FLOW



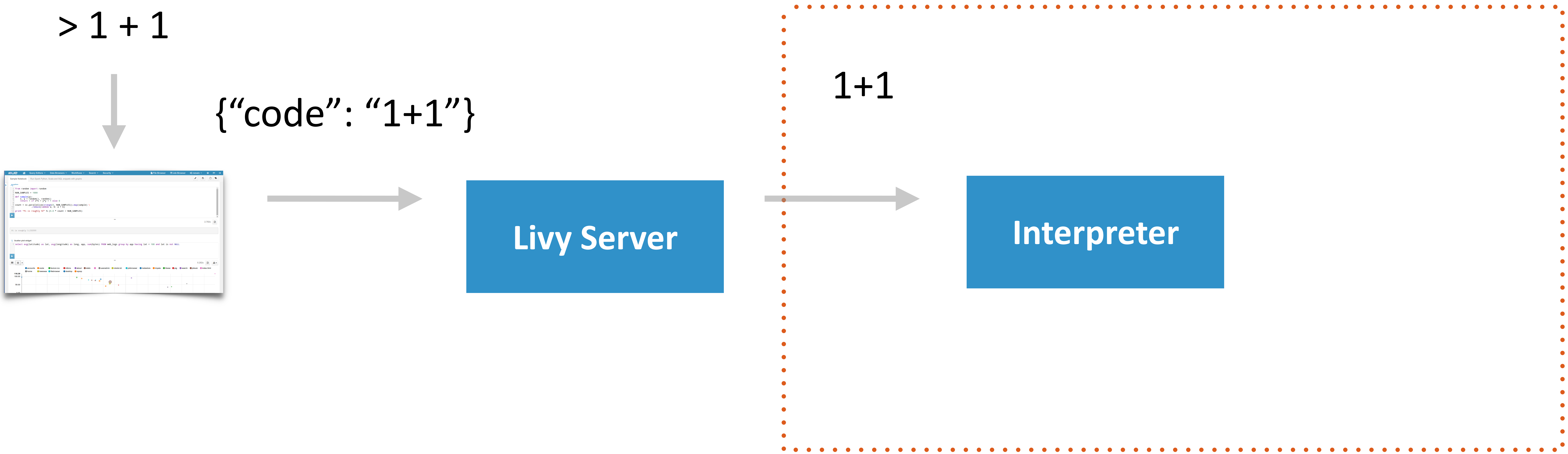
> 1 + 1

{“code”: “1+1”}

Livy Server

Interpreter

INTERPRETER FLOW



> 1 + 1

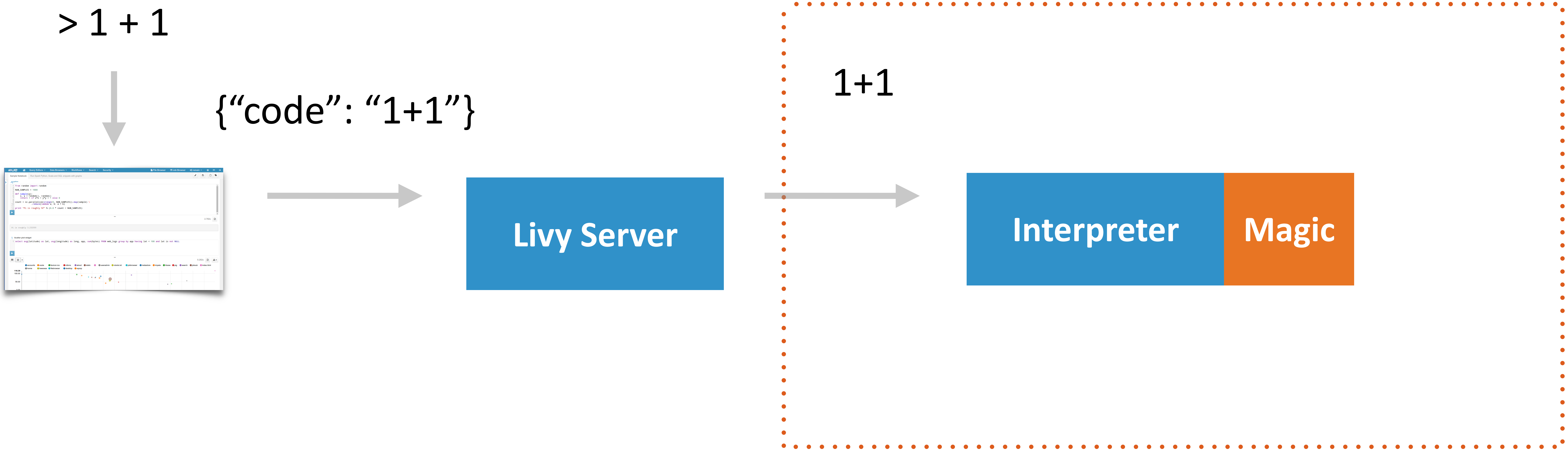
{“code”: “1+1”}

Livy Server

1+1

Interpreter

INTERPRETER FLOW



> 1 + 1

{"code": "1+1"}

1+1

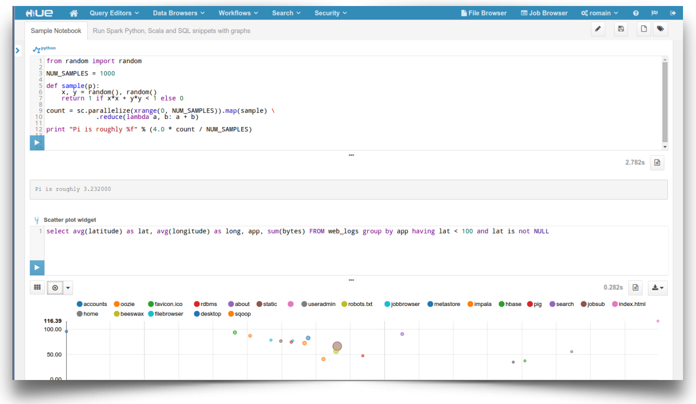
Livy Server

Interpreter

Magic

INTERPRETER FLOW

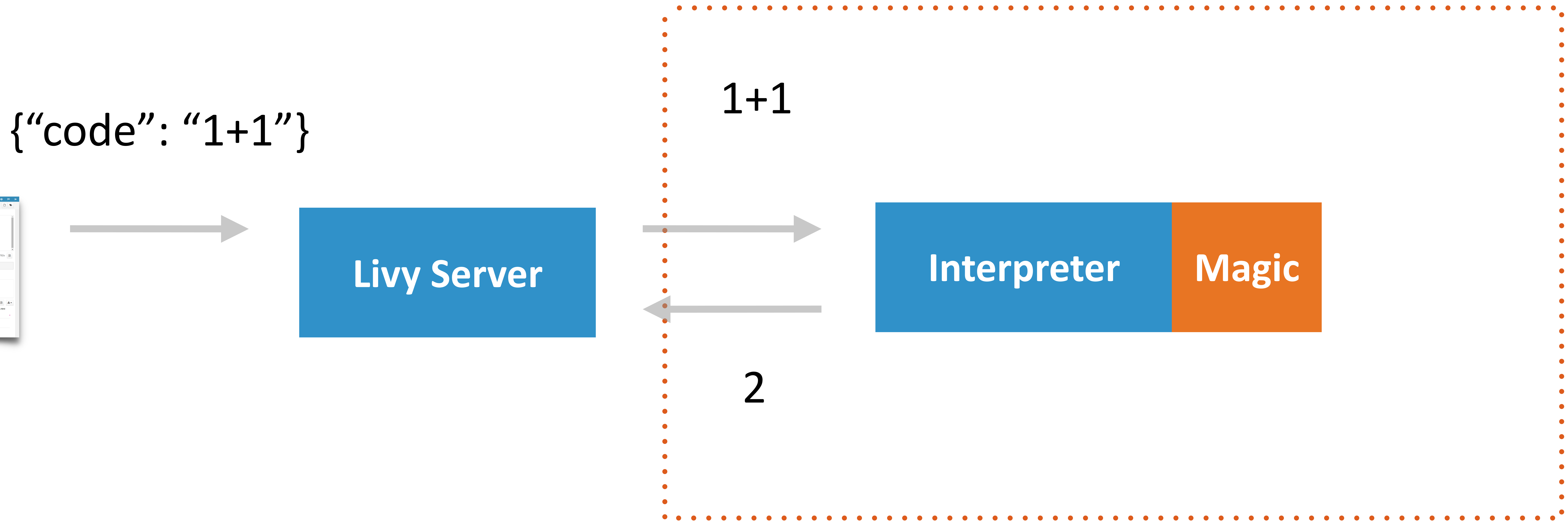
> 1 + 1



```
from random import random

NUM_SAMPLES = 1000

def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(xrange(0, NUM_SAMPLES)).map(sample) \
             .reduce(lambda a, b: a + b)

print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

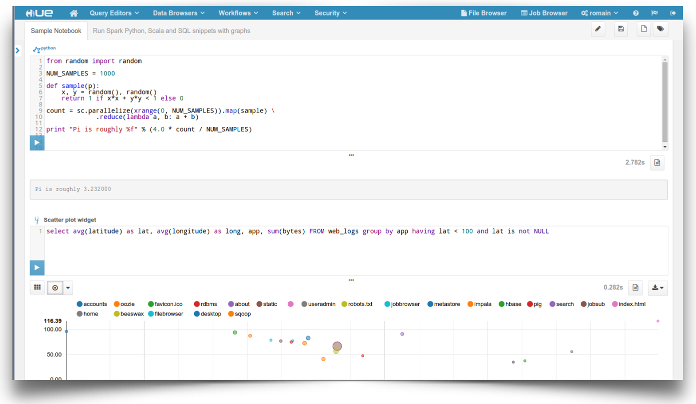
```
Pi is roughly 3.232000
```

```
select avg(latitude) as lat, avg(longitude) as long, app, sum(bytes) FROM web_logs group by app having lat < 100 and lat is not NULL
```



INTERPRETER FLOW

> 1 + 1



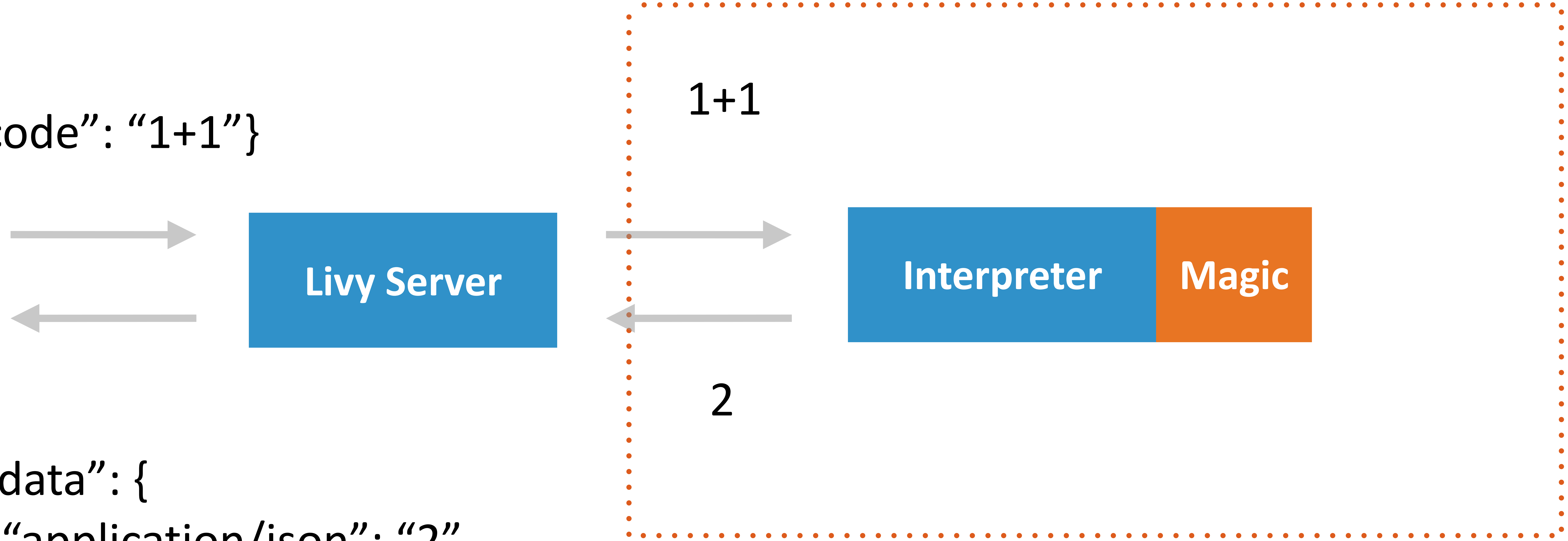
```
from random import random

NUM_SAMPLES = 1000

def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(xrange(0, NUM_SAMPLES)).map(sample) \
            .reduce(lambda a, b: a + b)

print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

2.782s

```
Pi is roughly 3.232000
```

```
select avg(latitude) as lat, avg(longitude) as long, app, sum(bytes) FROM web_logs group by app having lat < 100 and lat is not NULL
```

0.282s



{“code”: “1+1”}

Livy Server

1+1

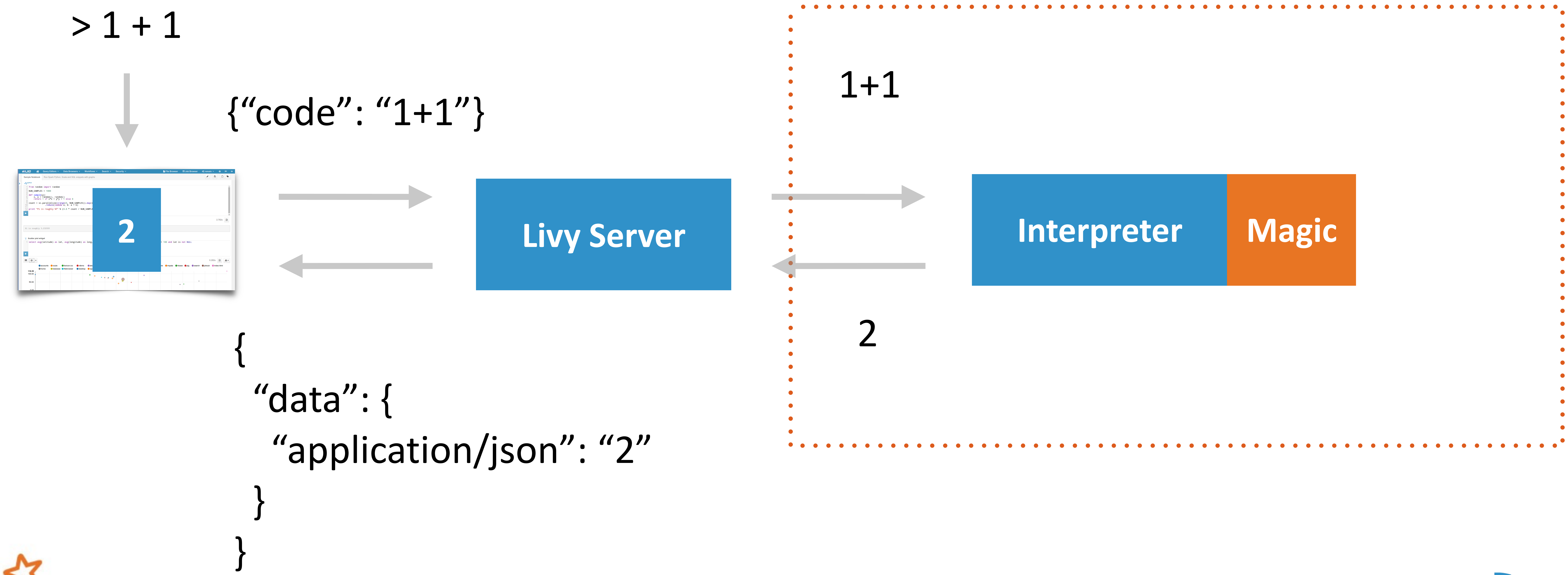
Interpreter

Magic

2

```
{
  “data”: {
    “application/json”: “2”
  }
}
```

INTERPRETER FLOW



> 1 + 1

2

{“code”: “1+1”}

Livy Server

1+1

Interpreter

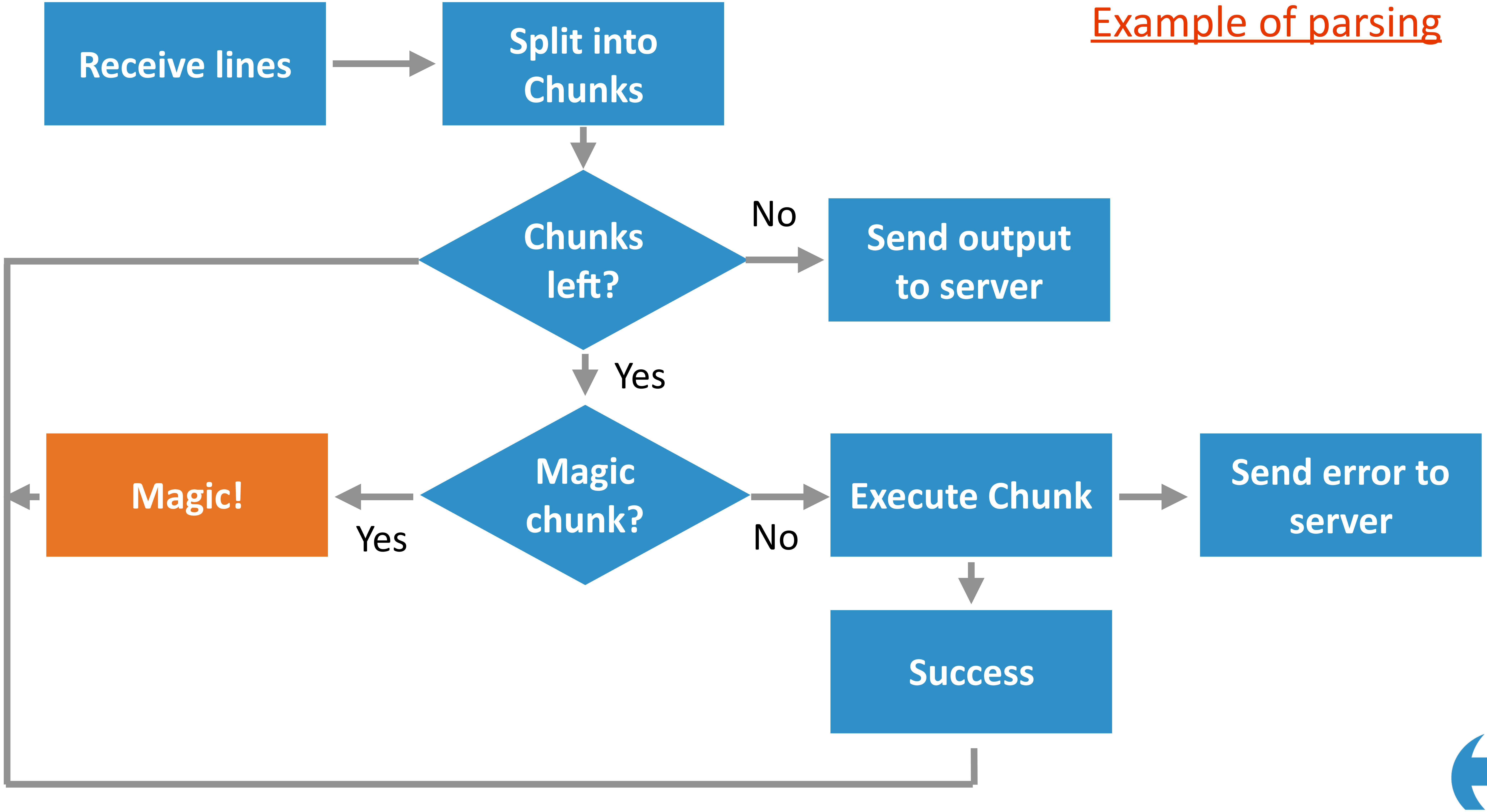
Magic

2

```
{
  “data”: {
    “application/json”: “2”
  }
}
```

INTERPRETER FLOW CHART

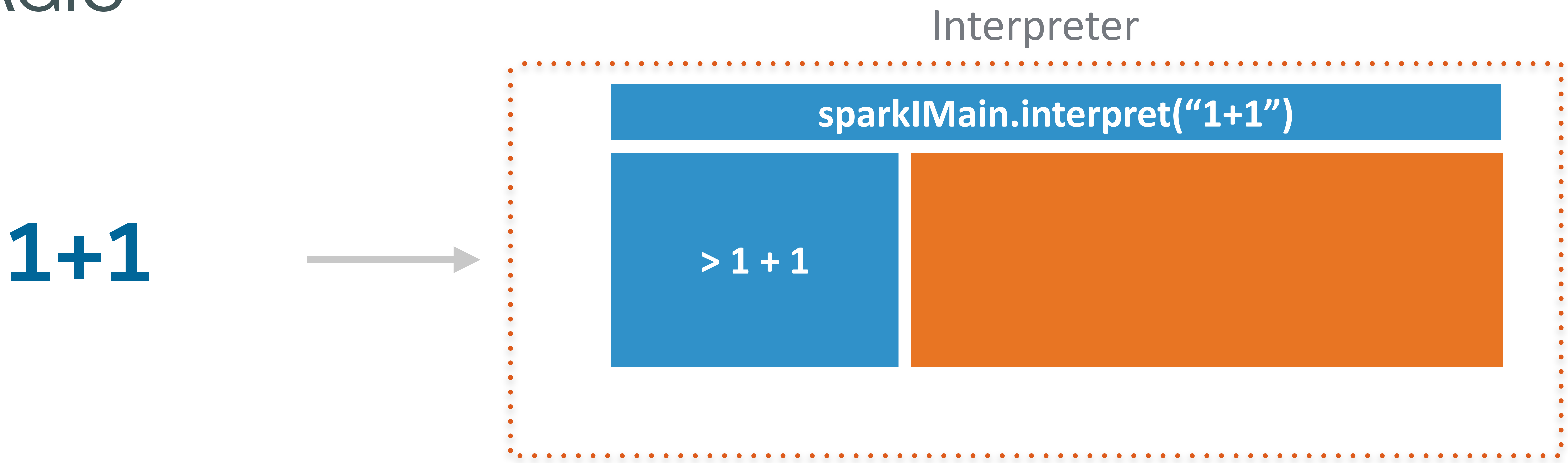
Example of parsing



INTERPRETER MAGIC

- table
- json
- plotting
- …

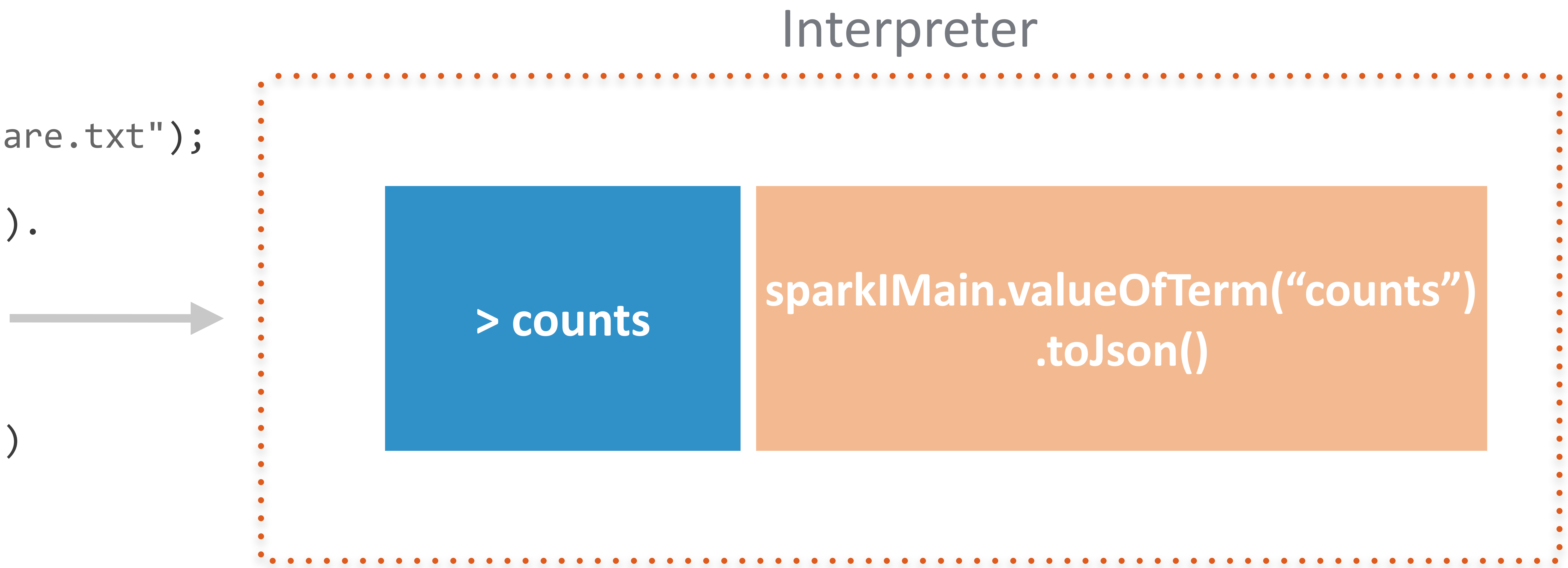
NO MAGIC



```
{
  "id": 0,
  "output": {
    "application/json": 2
  }
}
```

JSON MAGIC

```
val lines = sc.textFile("shakespeare.txt");
val counts = lines.
  flatMap(line => line.split(" ")).
    map(word => (word, 1)).
    reduceByKey(_ + _).
    sortBy(-_._2).
    map { case (w, c) =>
    Map("word" -> w, "count" -> c)
    }

%json counts
```



Interpreter

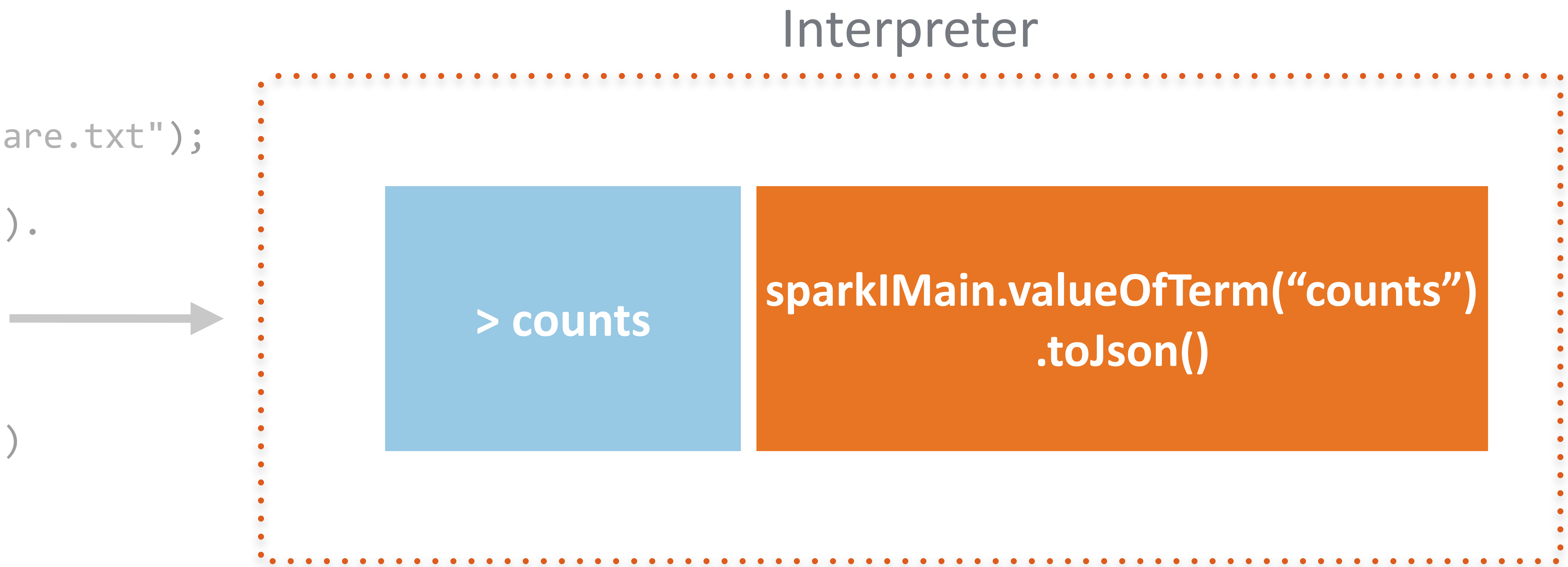
> counts

sparkIMain.valueOfTerm("counts")
.toJson()

```
[('', 506610), ('the', 23407), ('I', 19540)... ]
```

JSON MAGIC

```
val lines = sc.textFile("shakespeare.txt");
val counts = lines.
  flatMap(line => line.split(" ")).
    map(word => (word, 1)).
    reduceByKey(_ + _).
    sortBy(-_._2).
    map { case (w, c) =>
    Map("word" -> w, "count" -> c)
    }

%json counts
```



Interpreter

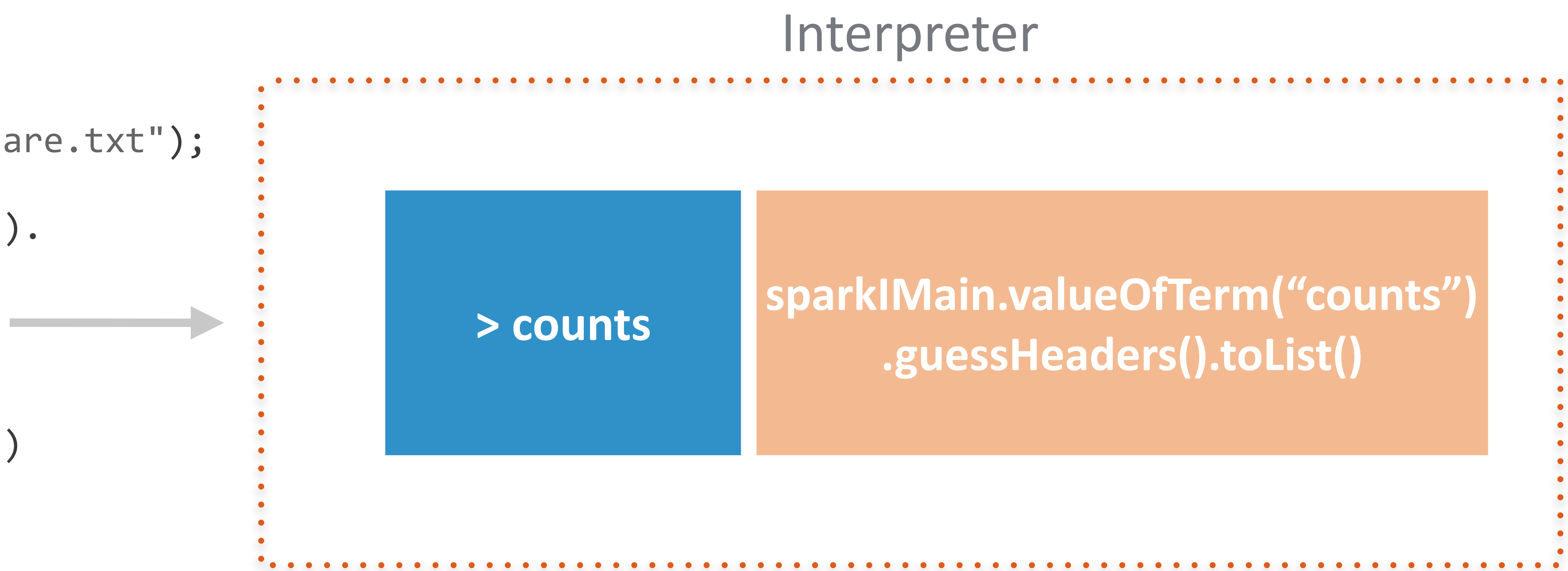
> counts

sparkIMain.valueOfTerm("counts")
.toJson()

```
{
  "id": 0,
  "output": {
    "application/json": [
      { "count": 506610, "word": "" },
      { "count": 23407, "word": "the" },
      { "count": 19540, "word": "I" },
      ...
    ]
  ...
}
```

TABLE MAGIC

```
val lines = sc.textFile("shakespeare.txt");
val counts = lines.
  flatMap(line => line.split(" ")).
    map(word => (word, 1)).
    reduceByKey(_ + _).
    sortBy(-_._2).
    map { case (w, c) =>
    Map("word" -> w, "count" -> c)
    }

%table counts
```



Interpreter

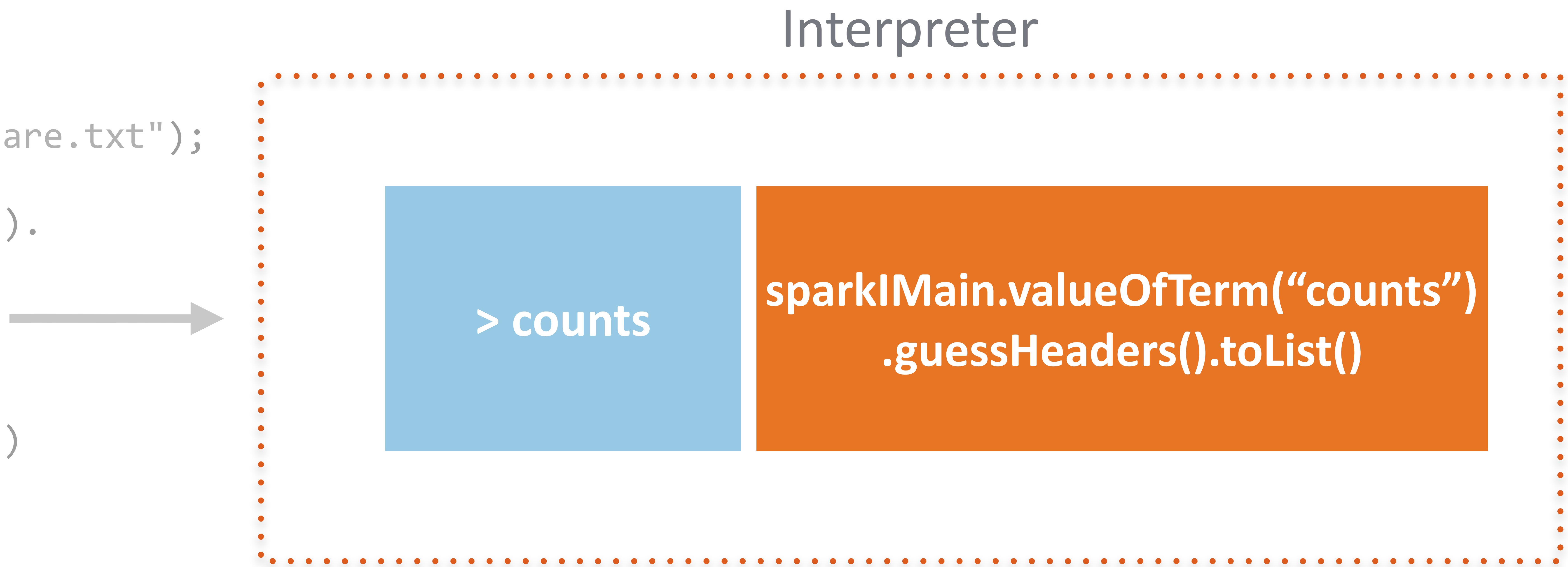
> counts

sparkIMain.valueOfTerm(“counts”)
.guessHeaders().toList()

```
[('', 506610), ('the', 23407), ('I', 19540)... ]
```

TABLE MAGIC

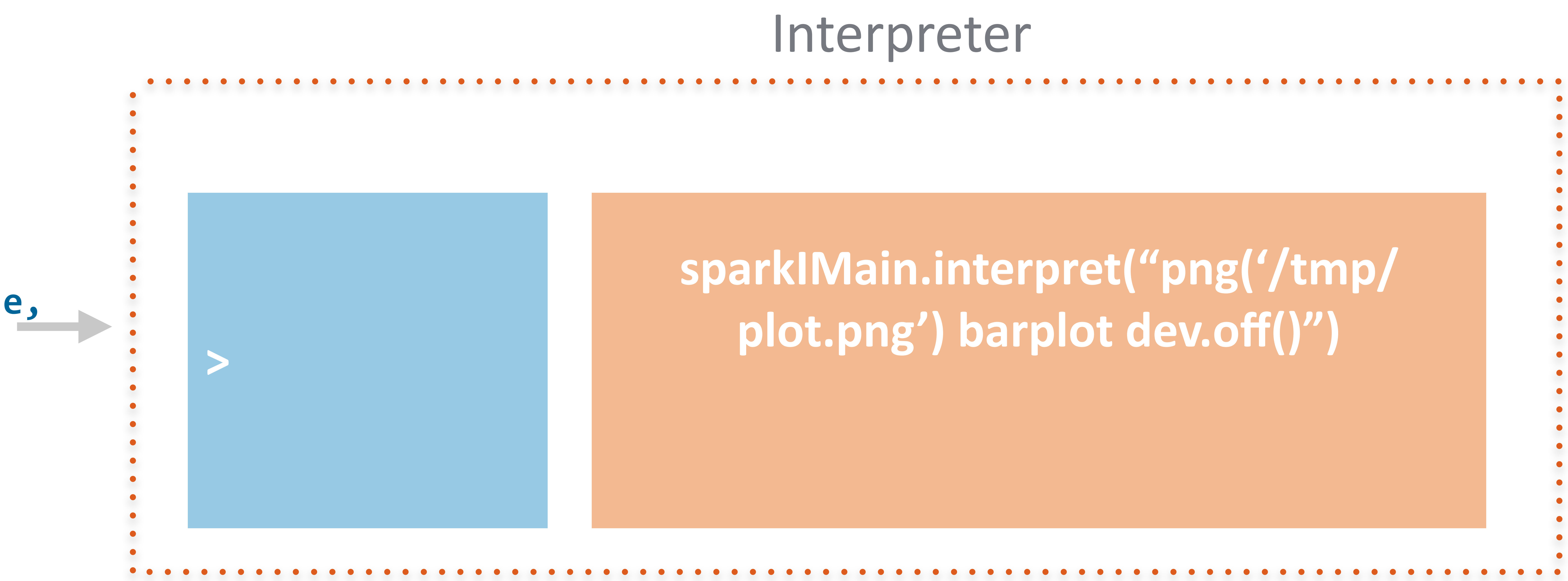
```
val lines = sc.textFile("shakespeare.txt");
val counts = lines.
  flatMap(line => line.split(" ")).
    map(word => (word, 1)).
    reduceByKey(_ + _).
    sortBy(-_._2).
    map { case (w, c) =>
    Map("word" -> w, "count" -> c)
    }

%table counts
```



```
"application/vnd.livy.table.v1+json": {
  "headers": [
    { "name": "count", "type": "BIGINT_TYPE" },
    { "name": "name", "type": "STRING_TYPE" }
  ],
  "data": [
    [ 23407, "the" ],
    [ 19540, "I" ],
    [ 18358, "and" ],
          ...
  ]
}
```

PLOT MAGIC

```
...
barplot(sorted_data
$count,names.arg=sorted_data$value,
main="Resource hits", las=2,
col=colfunc(nrow(sorted_data)),
ylim=c(0,300))
```



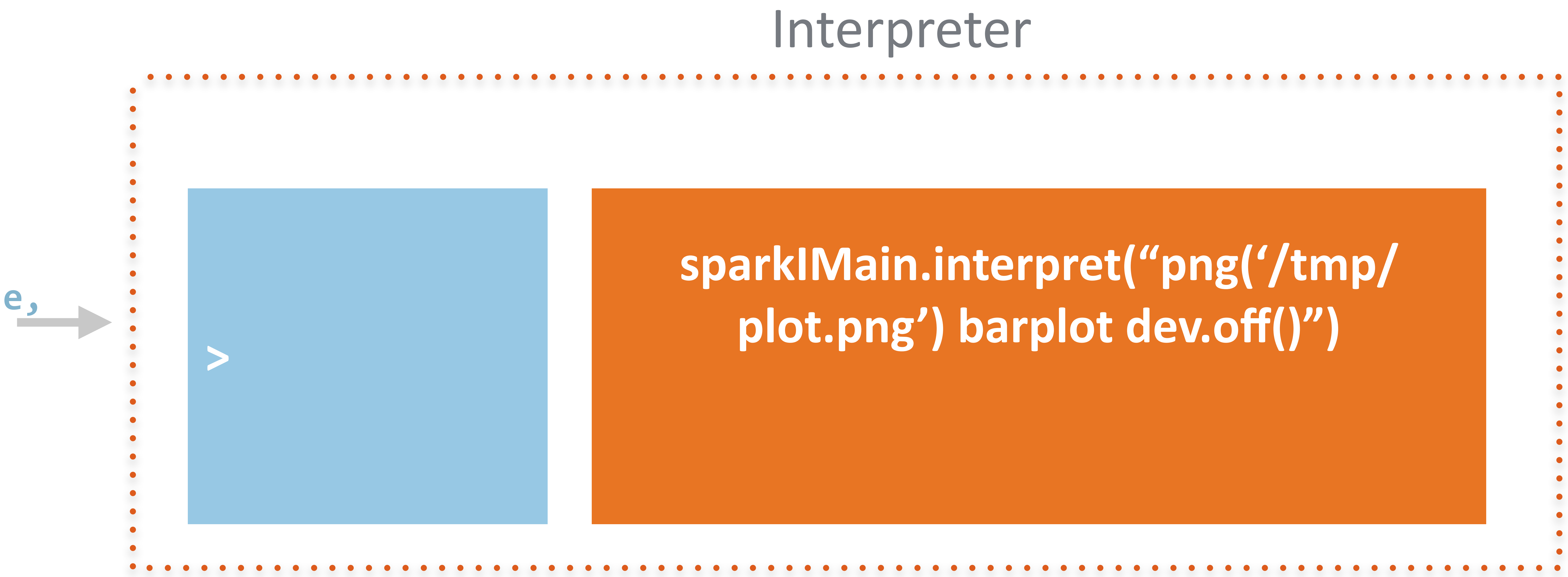
Interpreter

>

sparkIMain.interpret(“png(‘/tmp/plot.png’) barplot dev.off()”)

PLOT MAGIC

```
...
barplot(sorted_data
$count,names.arg=sorted_data$value,
main="Resource hits", las=2,
col=colfunc(nrow(sorted_data)),
ylim=c(0,300))
```



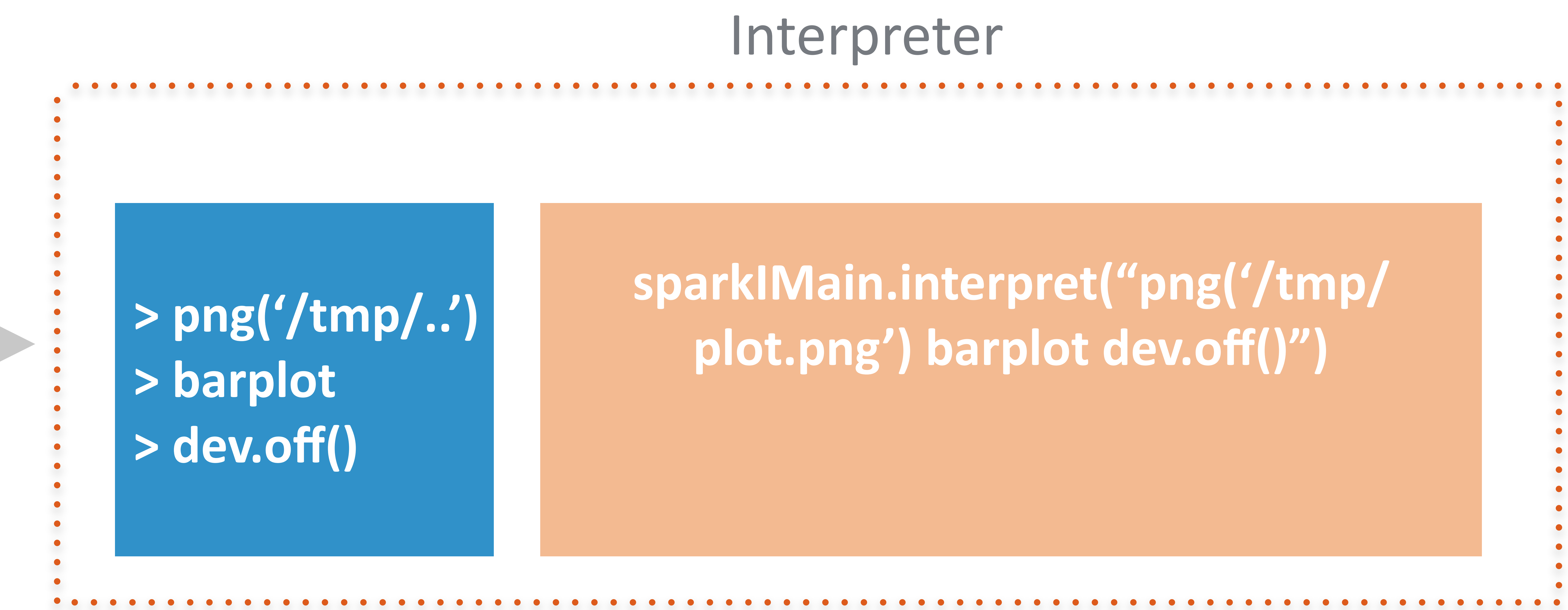
Interpreter

>

sparkIMain.interpret("png('/tmp/plot.png') barplot dev.off()")

PLOT MAGIC

```
...
barplot(sorted_data
$count,names.arg=sorted_data$value,
main="Resource hits", las=2,
col=colfunc(nrow(sorted_data)),
ylim=c(0,300))
```



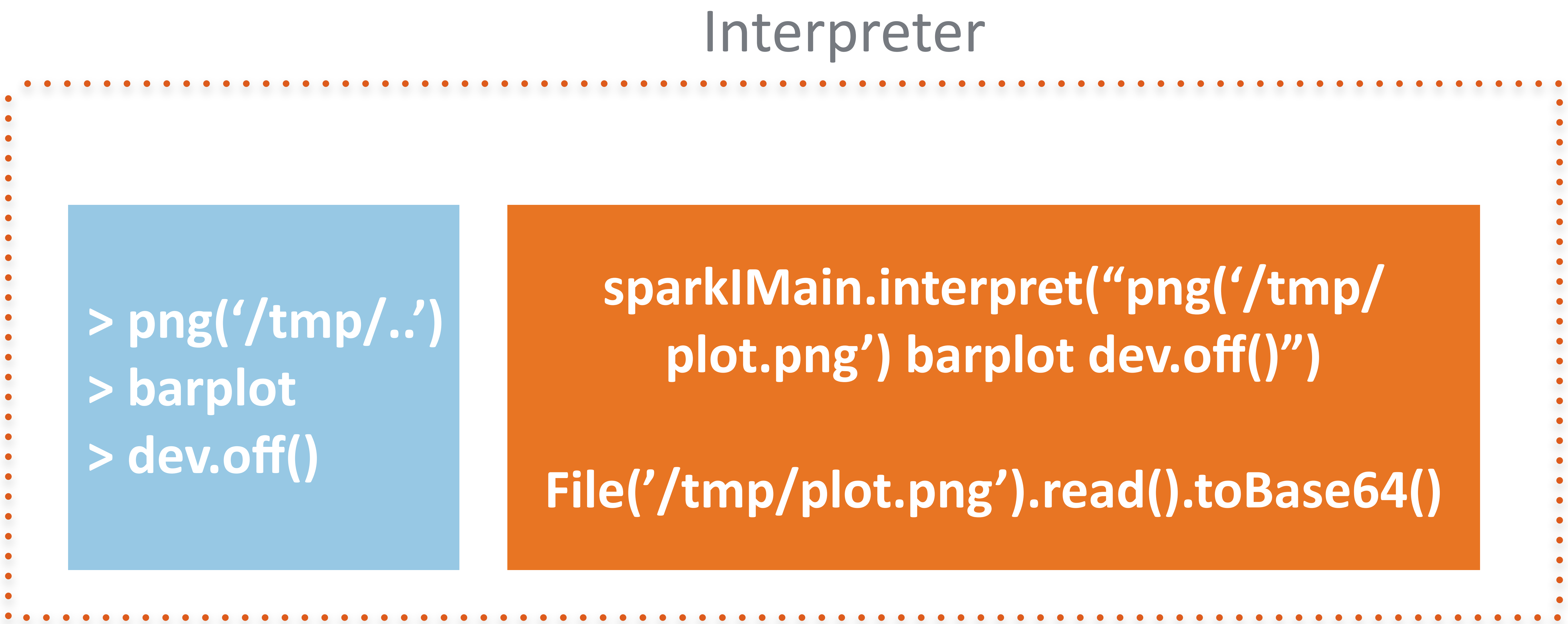
Interpreter

> png(‘/tmp/..’)
> barplot
> dev.off()

sparkIMain.interpret(“png(‘/tmp/
plot.png’) barplot dev.off()”)

PLOT MAGIC

```
...
barplot(sorted_data
$count,names.arg=sorted_data$value,
main="Resource hits", las=2,
col=colfunc(nrow(sorted_data)),
ylim=c(0,300))
```



Interpreter

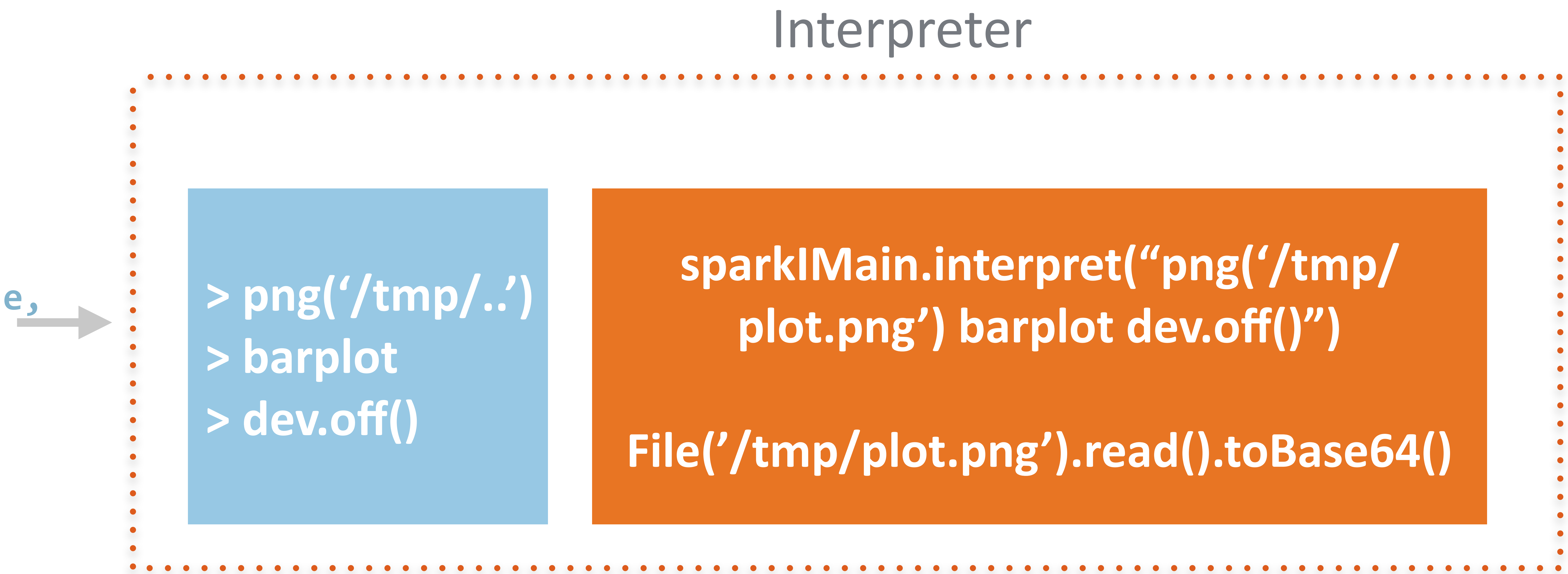
> png('/tmp/..')
> barplot
> dev.off()

sparkIMain.interpret("png('/tmp/plot.png') barplot dev.off()")

File('/tmp/plot.png').read().toBase64()

PLOT MAGIC

```
...
barplot(sorted_data
$count,names.arg=sorted_data$value,
main="Resource hits", las=2,
col=colfunc(nrow(sorted_data)),
ylim=c(0,300))
```

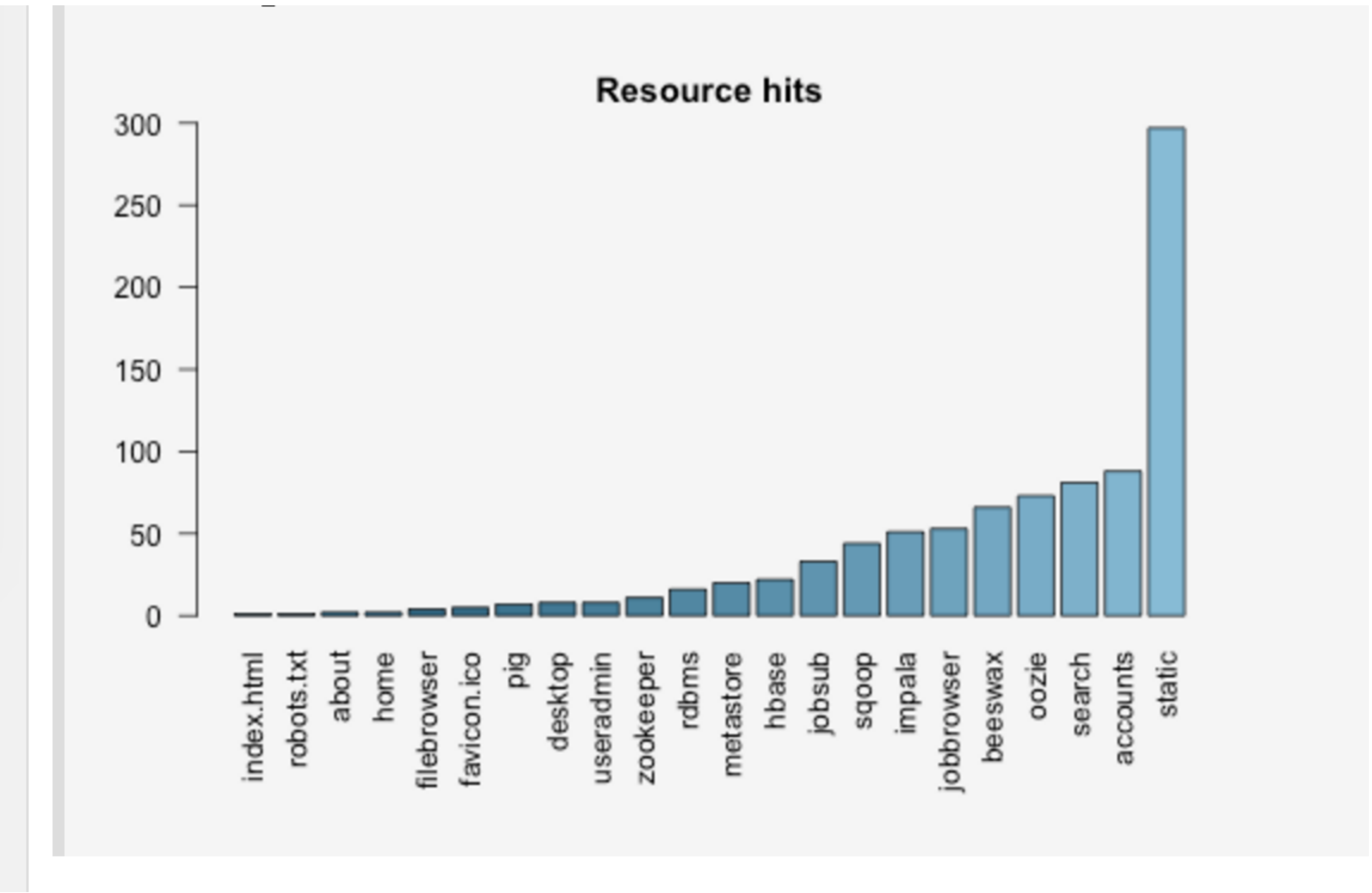


Interpreter

> png(‘/tmp/..’)
> barplot
> dev.off()

sparkIMain.interpret(“png(‘/tmp/plot.png’) barplot dev.off()”)

File(’/tmp/plot.png’).read().toBase64()



```
{
  "data": {
    "image/png": "iVBORw0KGgoAAAANSUhE
     ...
    }
  ...
}
```

PLUGGABLE INTERPRETERS

- Pluggable Backends
- Livy's Spark Backends
  - Scala
  - pyspark
  - R
- IPython / Jupyter support coming soon

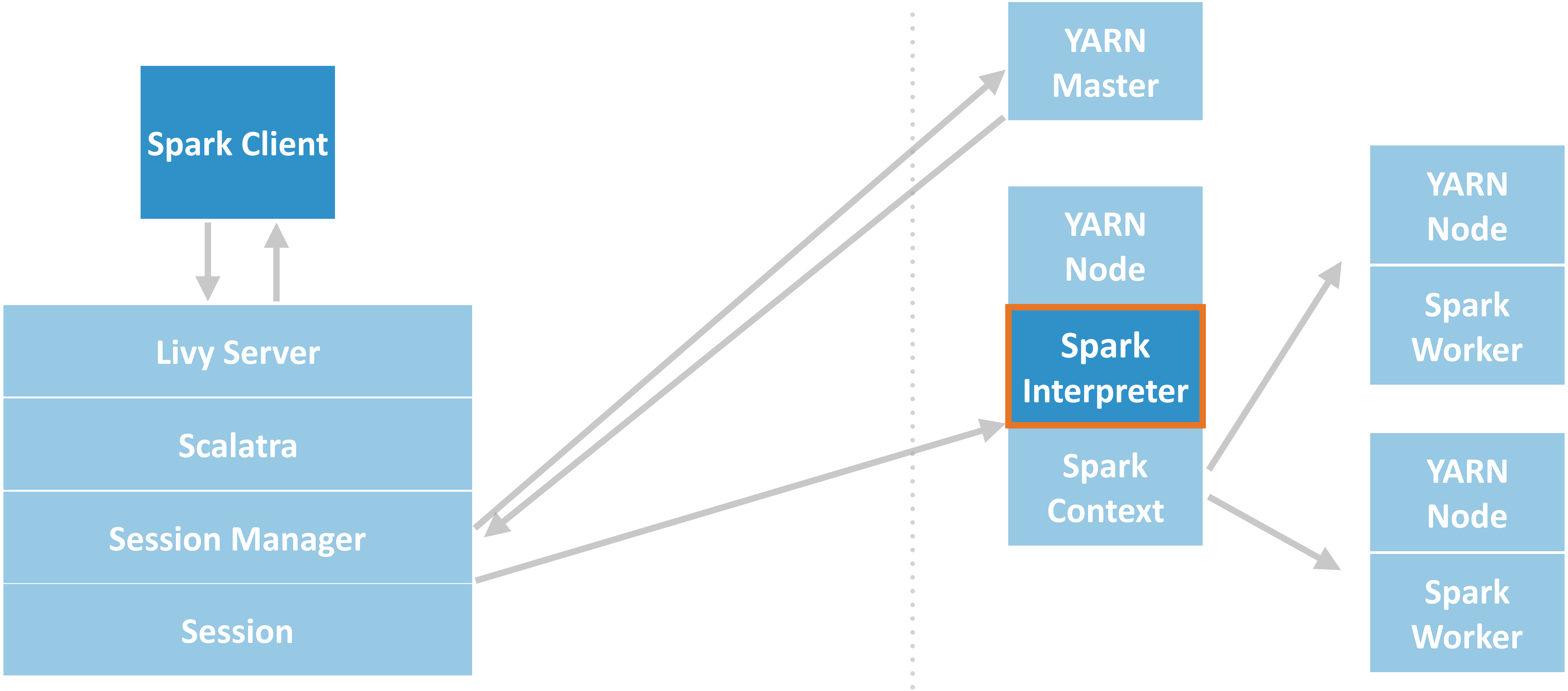
JUPYTER BACKEND

- Re-using it
- Generic Framework for Interpreters
- 51 Kernels

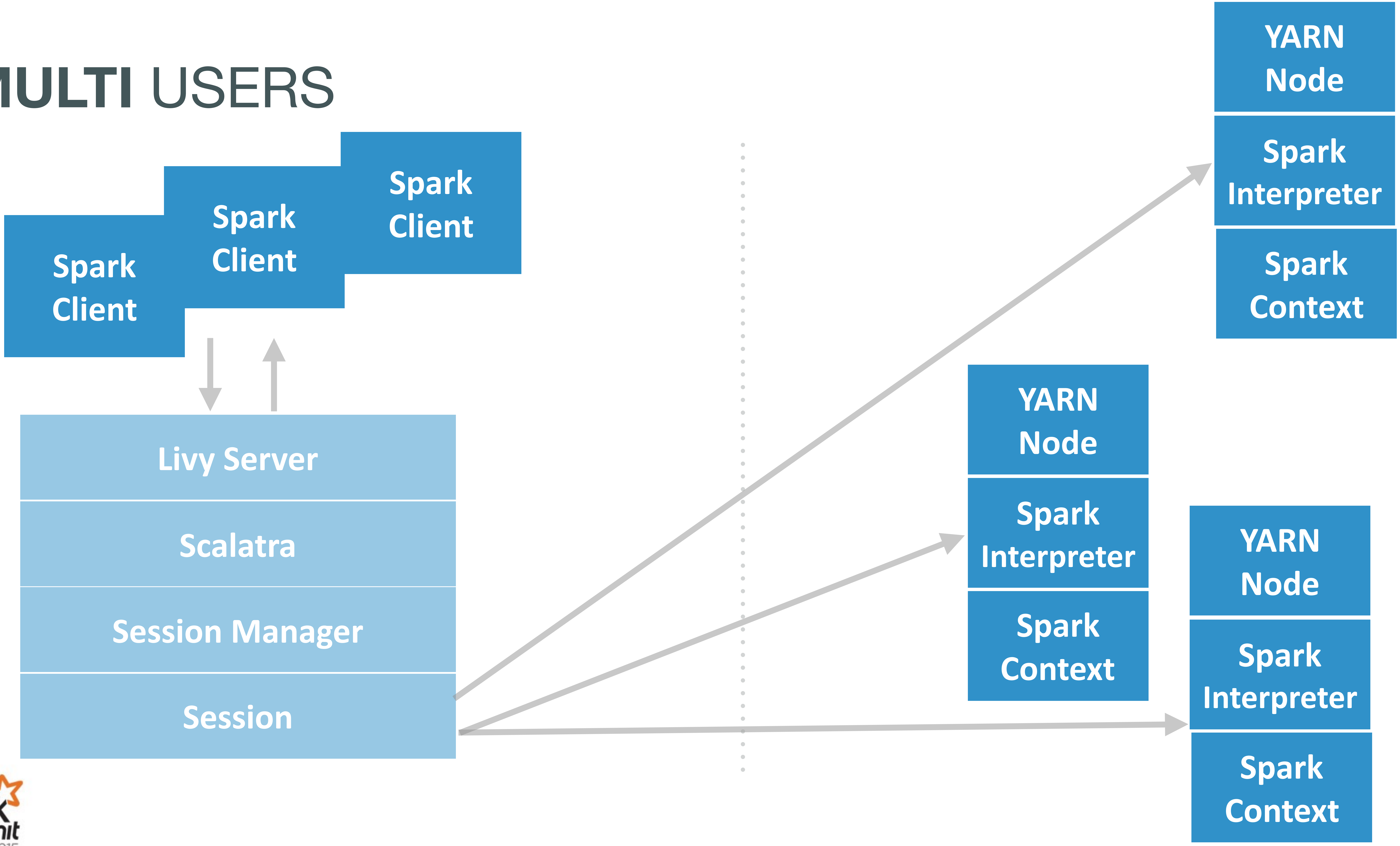
jupyter

SPARK AS A SERVICE

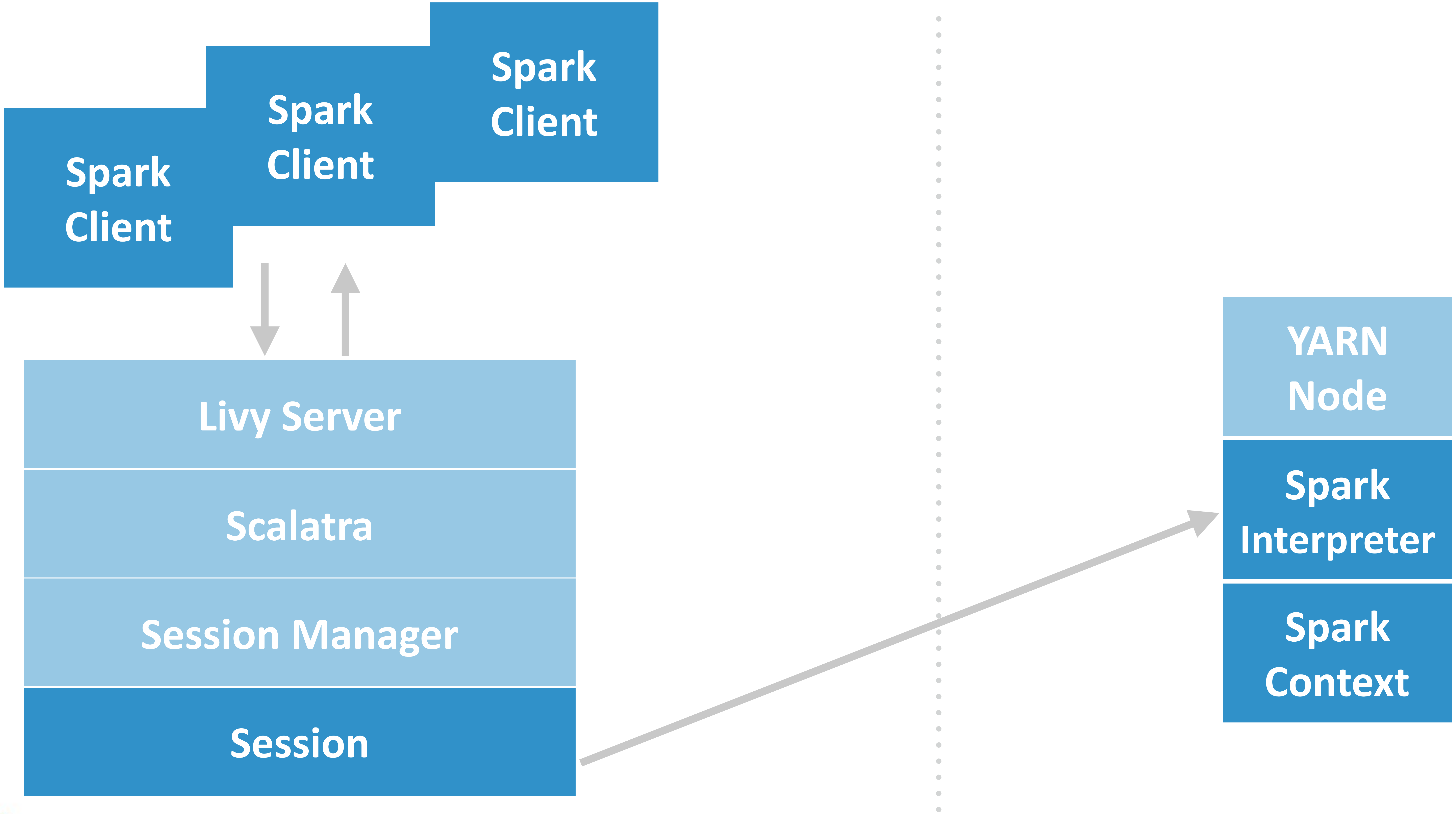
REMEMBER AGAIN?



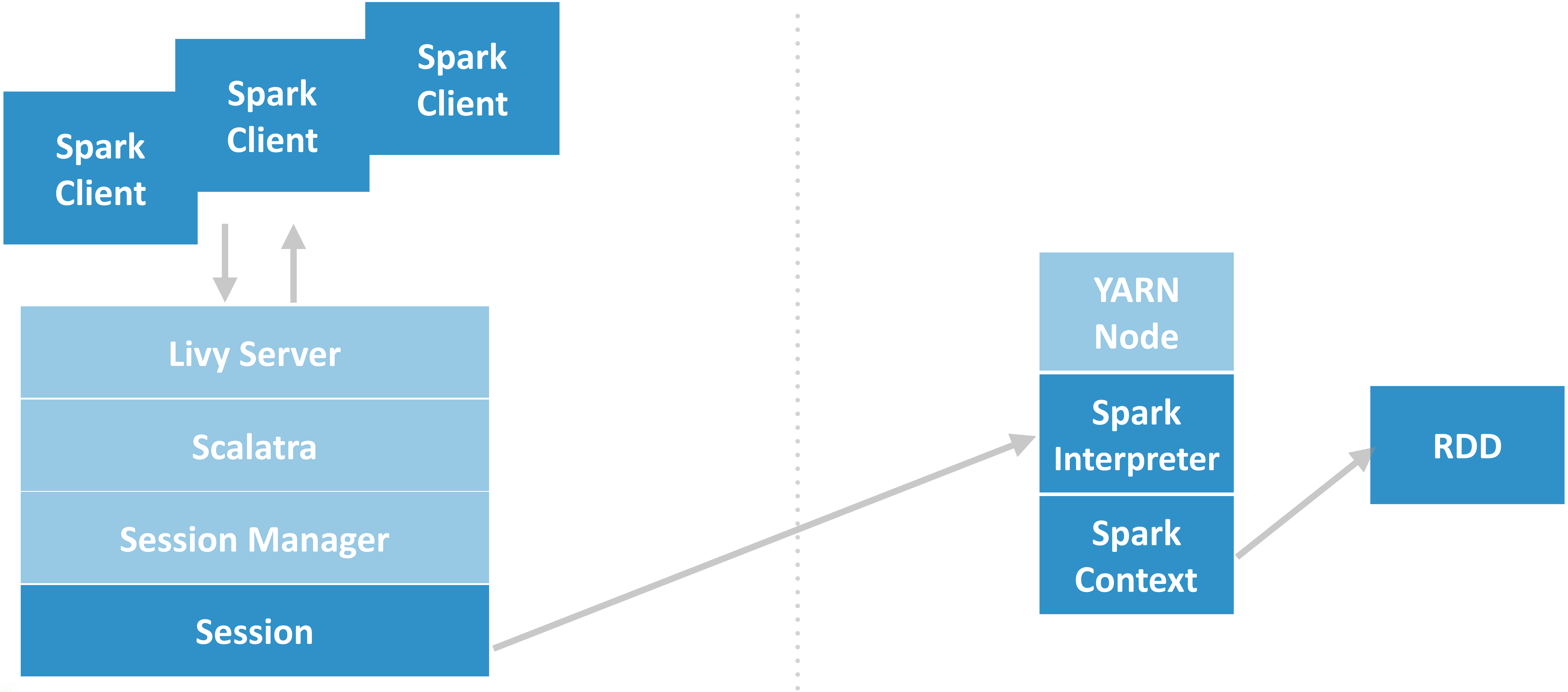
MULTI USERS


Spark Client
Spark Client
Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Node
Spark Interpreter
Spark Context
YARN Node
Spark Interpreter
Spark Context
YARN Node
Spark Interpreter
Spark Context

SHARED CONTEXTS?

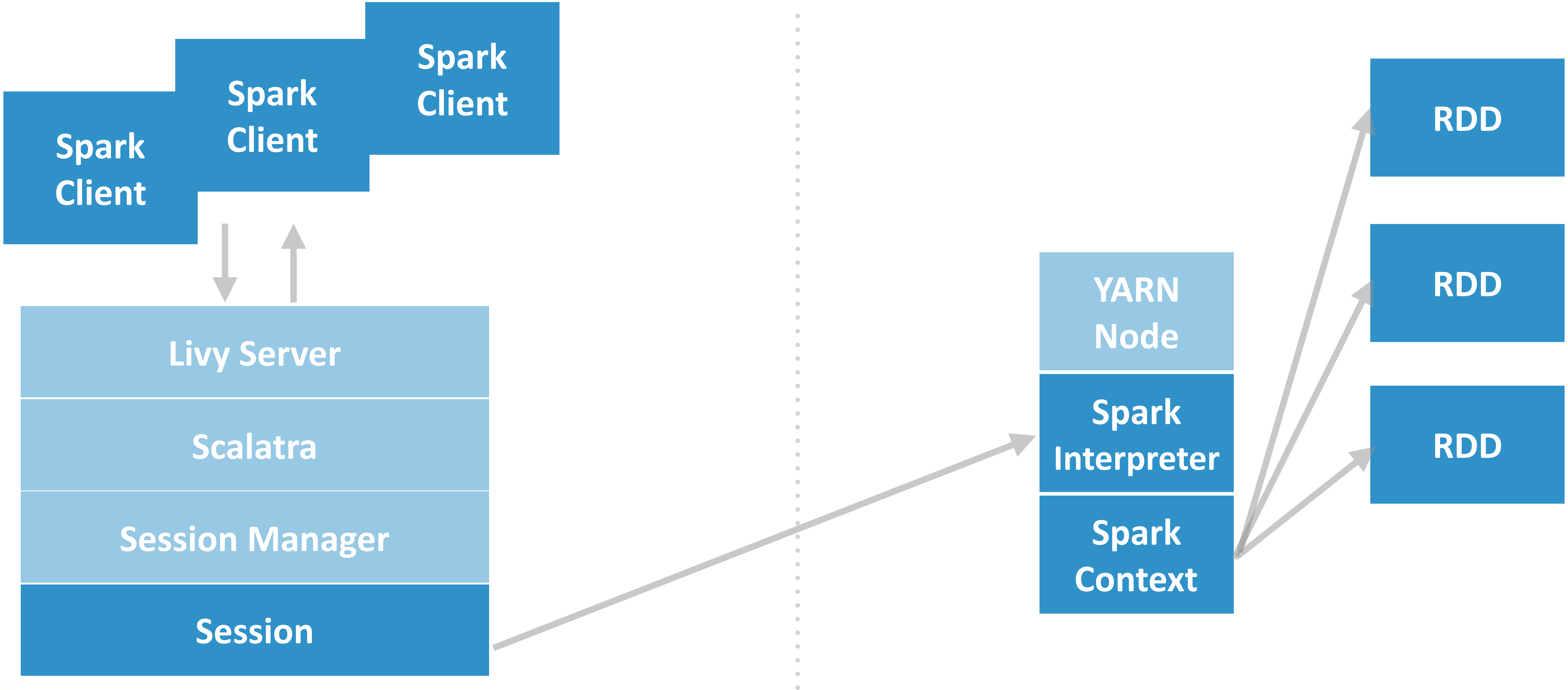


SHARED RDD?

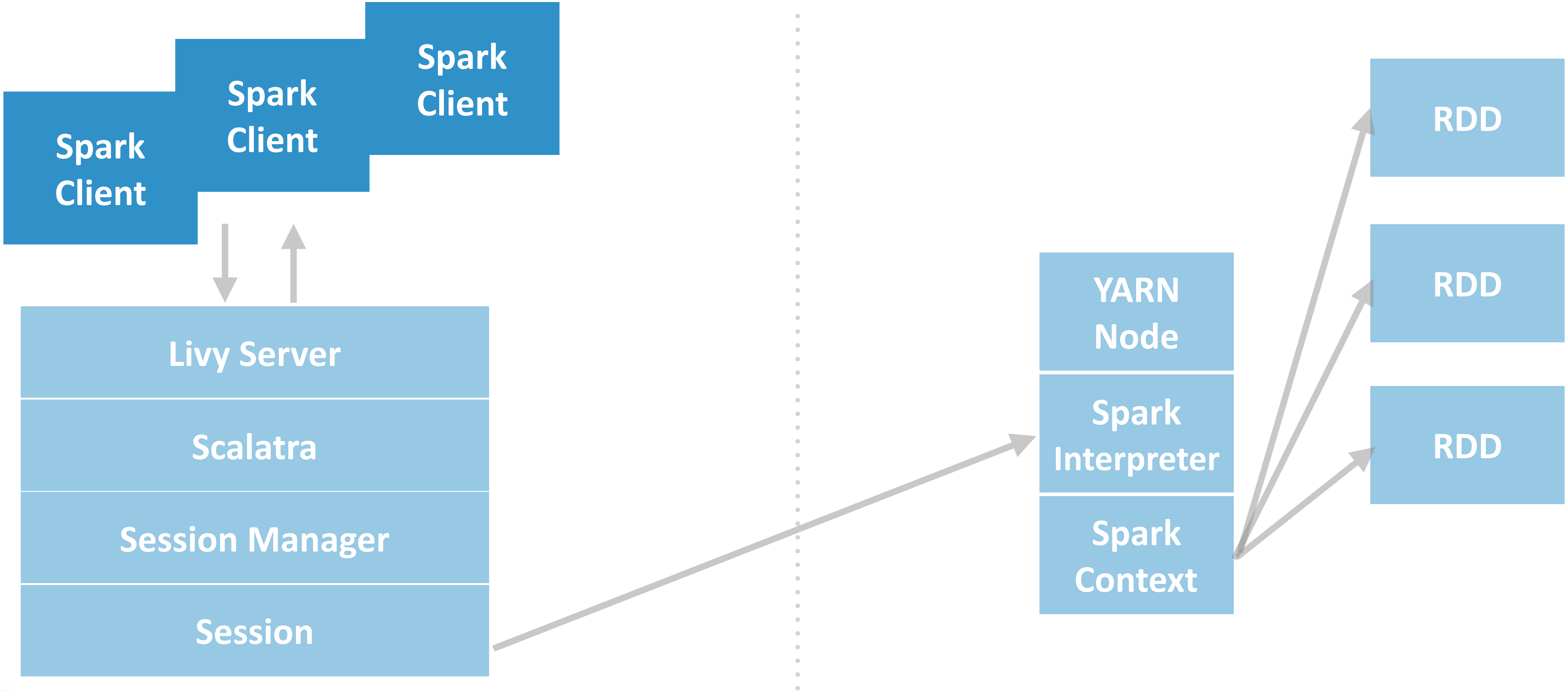


Spark Client
Spark Client
Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Node
Spark Interpreter
Spark Context
RDD

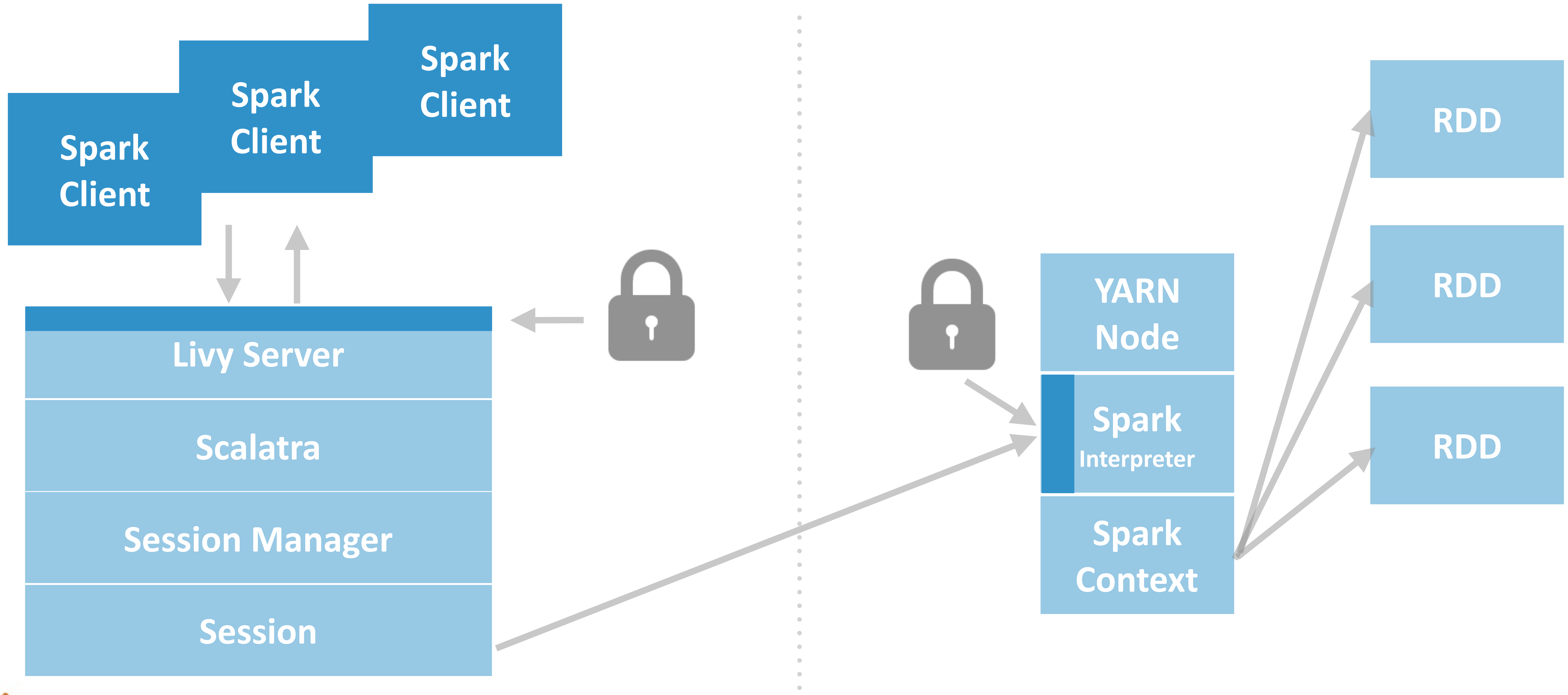
SHARED RDDS?


Spark Client
Spark Client
Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Node
Spark Interpreter
Spark Context
RDD
RDD
RDD

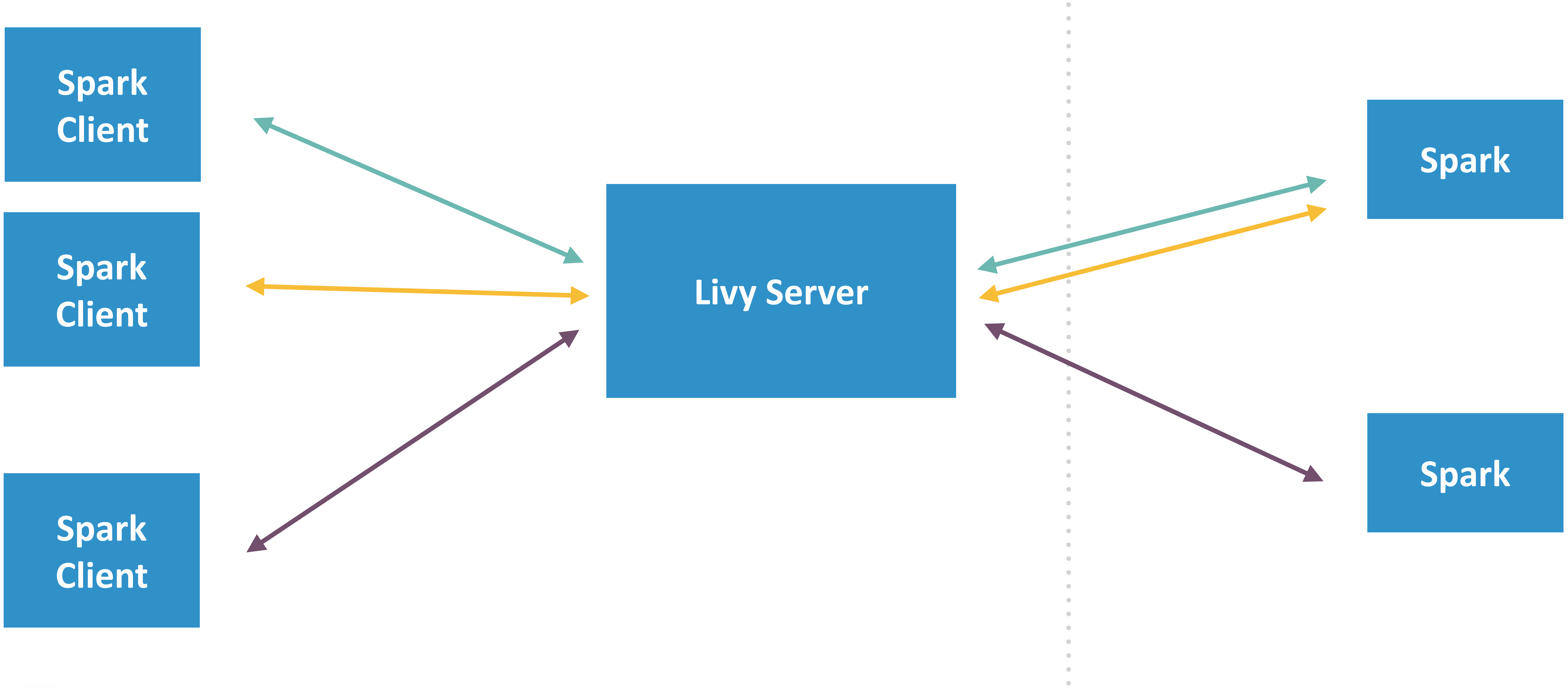
SECURE IT?


Spark Client
Spark Client
Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Node
Spark Interpreter
Spark Context
RDD
RDD
RDD

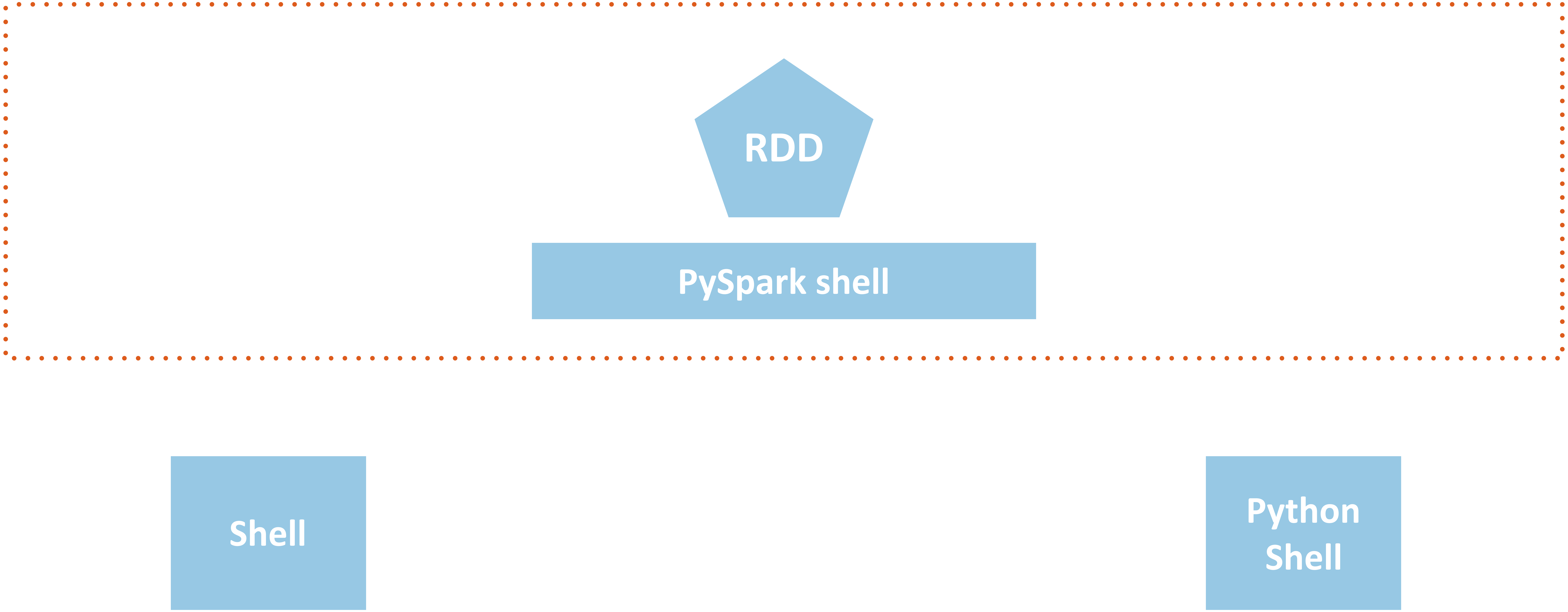
SECURE IT?


Spark Client
Spark Client
Spark Client
Livy Server
Scalatra
Session Manager
Session
YARN Node
Spark Interpreter
Spark Context
RDD
RDD
RDD

SPARK AS SERVICE



SHARING RDDS

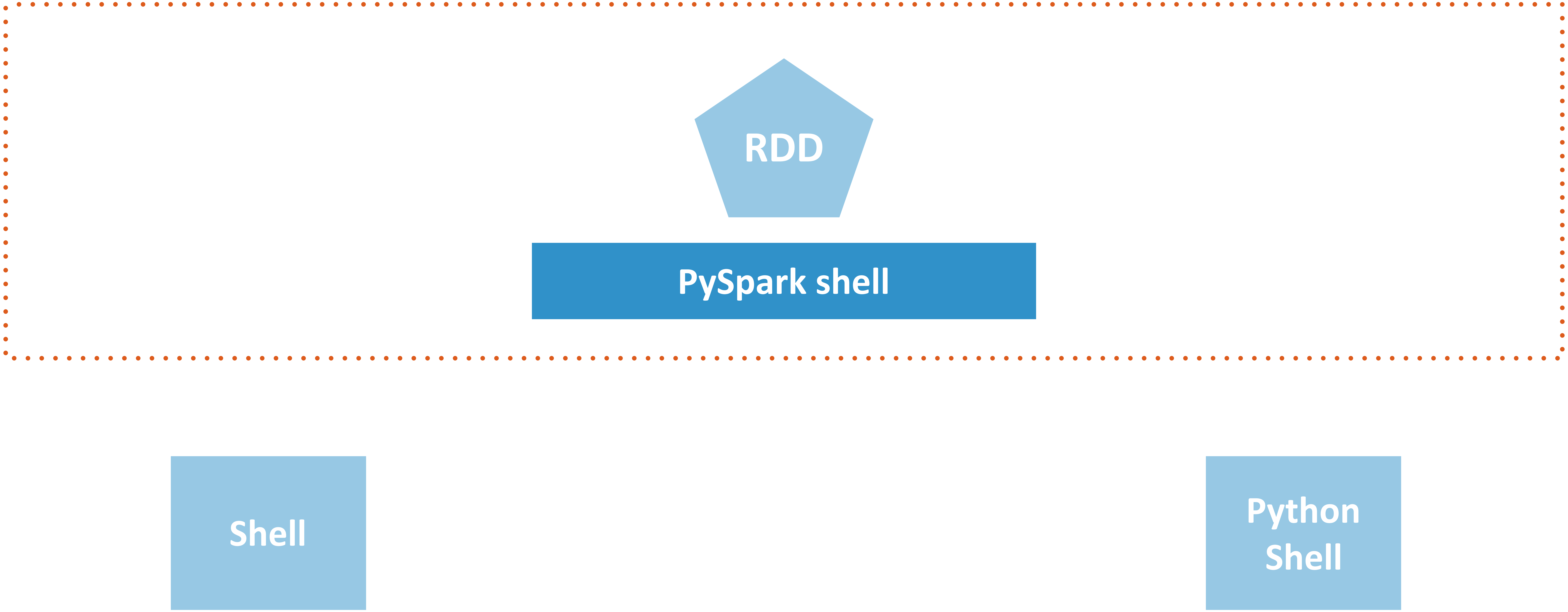


RDD

PySpark shell

Shell

Python Shell



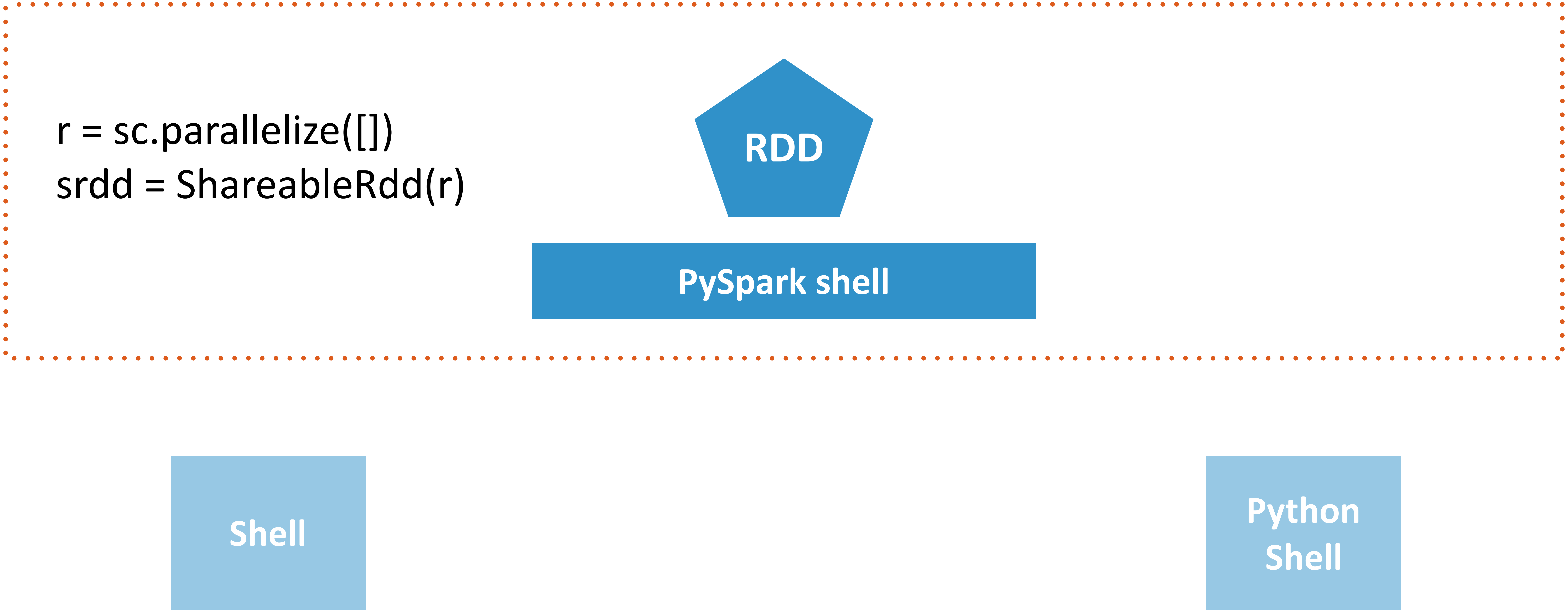
RDD

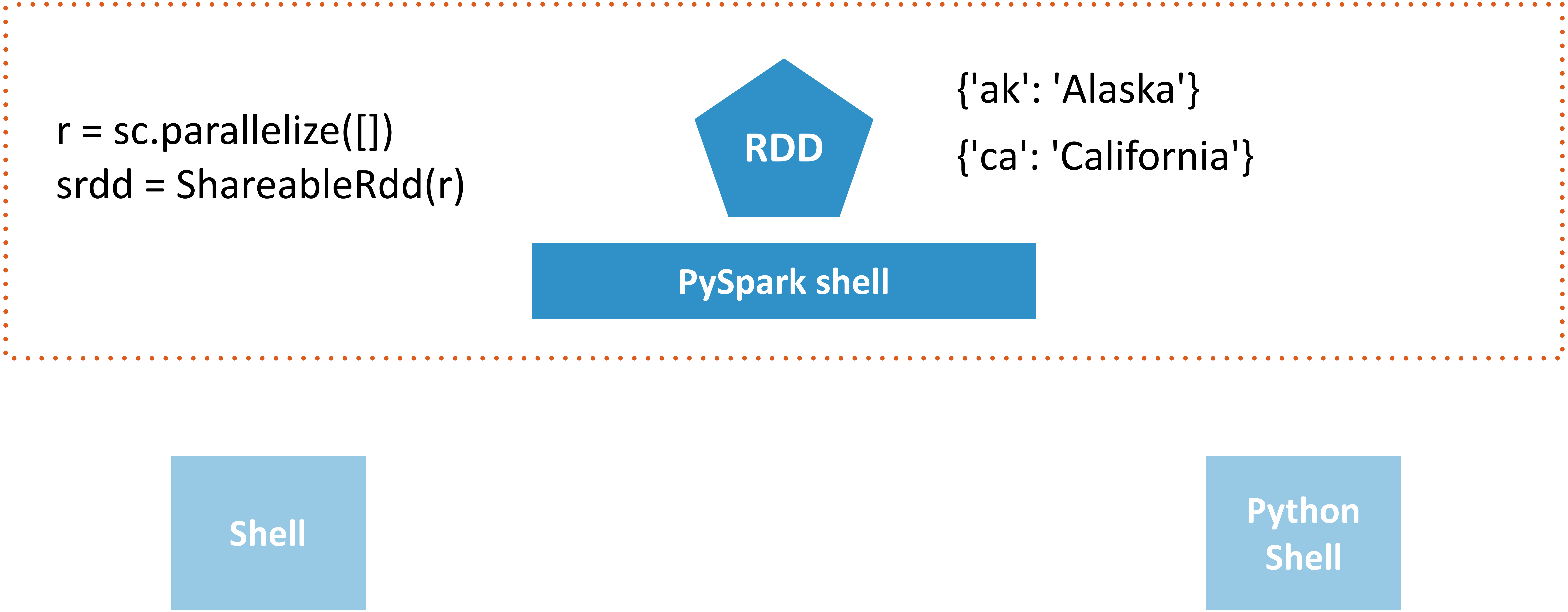
PySpark shell

Shell

Python Shell

```
r = sc.parallelize([])
srdd = ShareableRdd(r)
```





```
r = sc.parallelize([])
srdd = ShareableRdd(r)
```

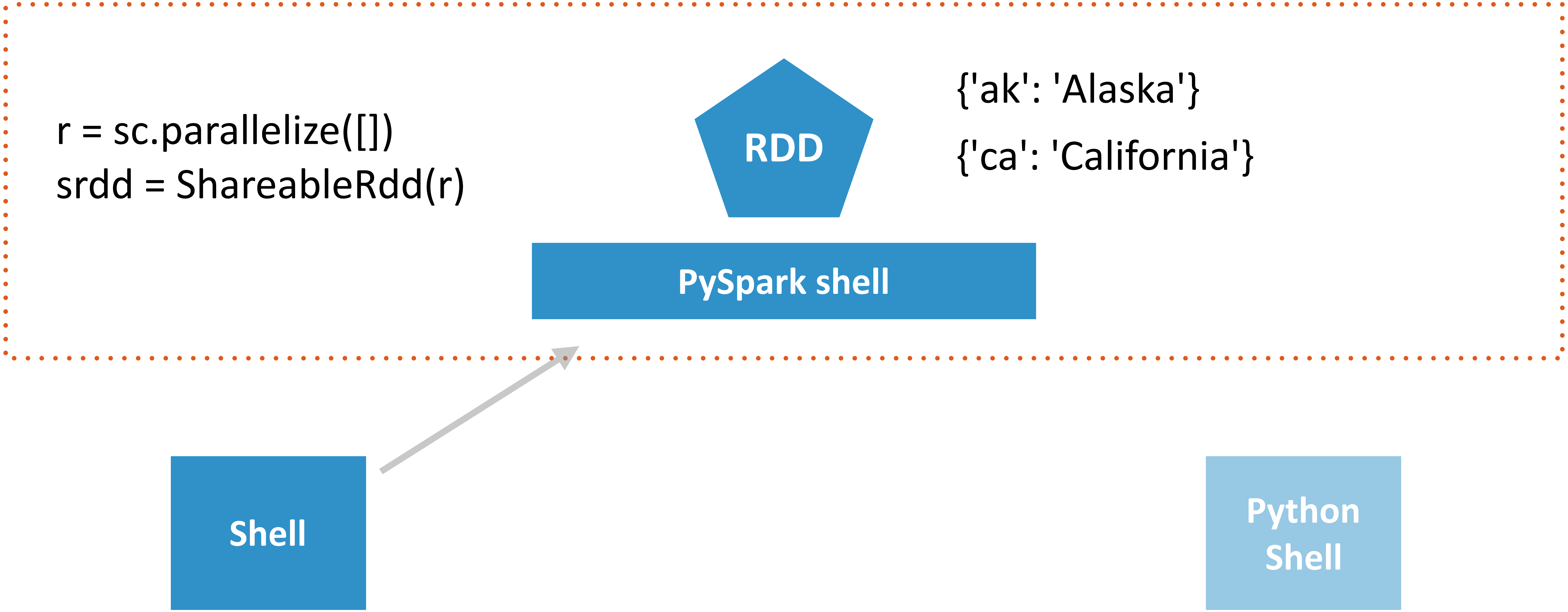
RDD

```
{'ak': 'Alaska'}
{'ca': 'California'}
```

PySpark shell

Shell

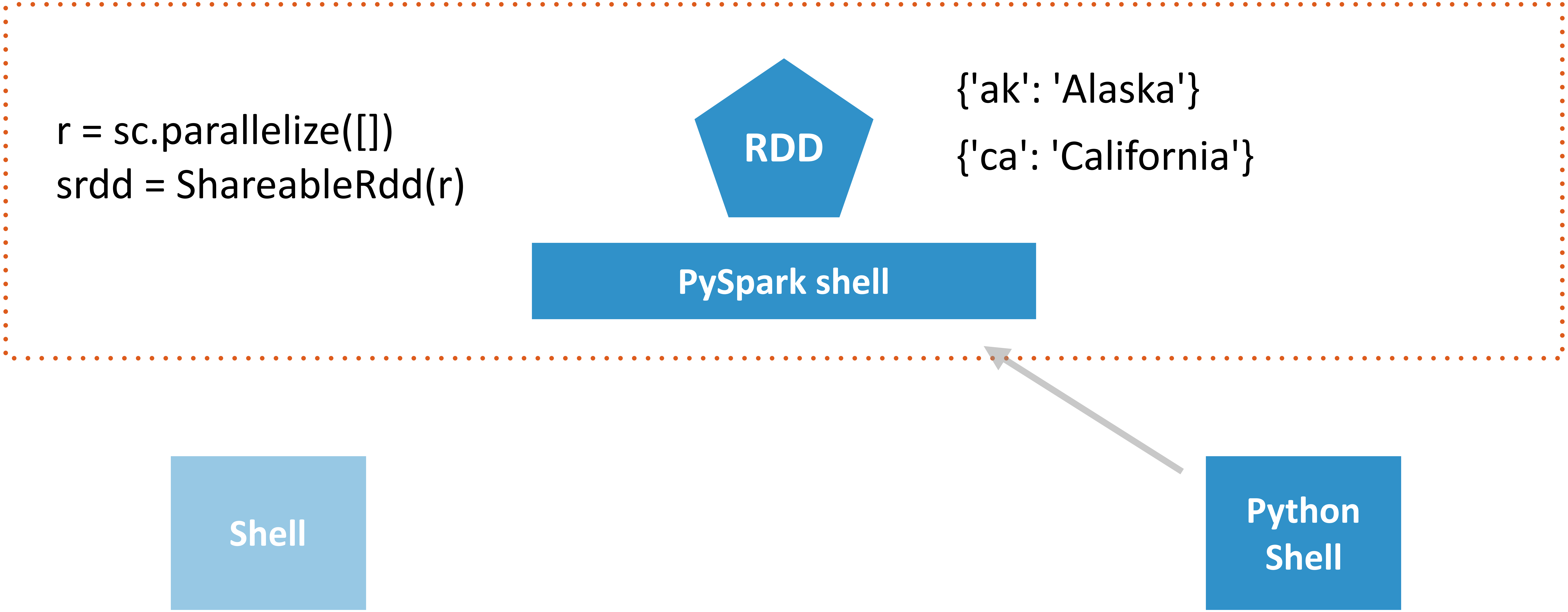
Python Shell



```
r = sc.parallelize([])
srdd = ShareableRdd(r)
```

```
{'ak': 'Alaska'}
{'ca': 'California'}
```

```
curl -XPOST /sessions/0/statement {
  'code': srdd.get('ak')
}
```



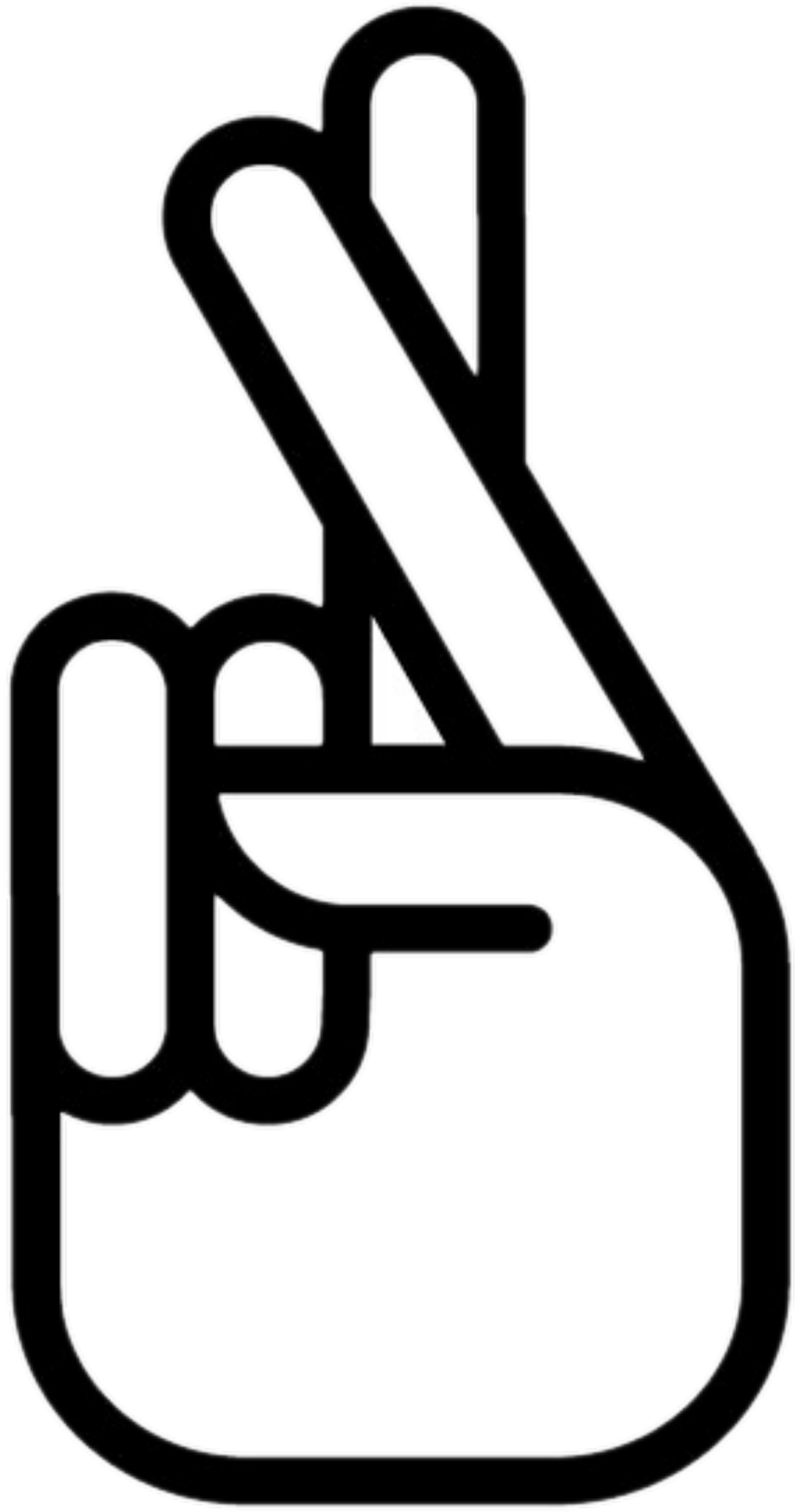
```
r = sc.parallelize([])
srdd = ShareableRdd(r)
```

```
{'ak': 'Alaska'}
{'ca': 'California'}
```

```
curl -XPOST /sessions/0/statement {
  'code': srdd.get('ak')
}
```

```
states = SharedRdd('host/sessions/0', 'srdd')
states.get('ak')
```

DEMO TIME



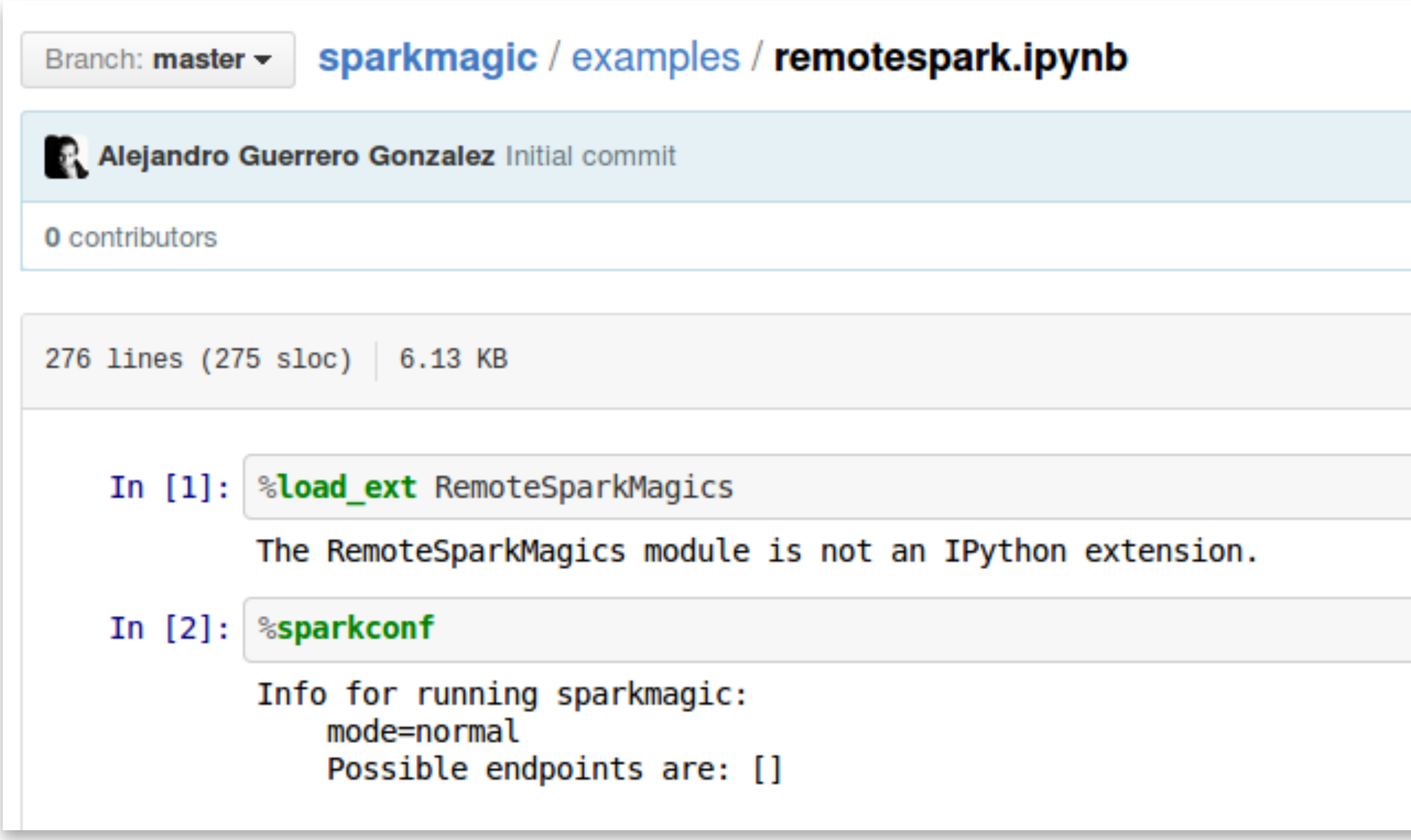
https://github.com/romainr/hadoop-tutorials-examples/tree/master/notebook/shared_rdd

SECURITY

- SSL Support
- Persistent Sessions
- Kerberos

SPARK MAGIC

- From Microsoft
- Python magics for working with remote Spark clusters
- Open Source: https://github.com/jupyter-incubator/sparkmagic



Branch: master
sparkmagic / examples / remotespark.ipynb
Alejandro Guerrero Gonzalez Initial commit
0 contributors

```
276 lines (275 sloc) | 6.13 KB

In [1]: %load_ext RemoteSparkMagics
        The RemoteSparkMagics module is not an IPython extension.

In [2]: %sparkconf
        Info for running sparkmagic:
            mode=normal
            Possible endpoints are: []
```

FUTURE

- Move to ext repo?
- Security
- iPython/Jupyter backends and file format
- Shared named RDD / contexts?
- Share data
- Spark specific, language generic, both?
- Leverage Hue 4

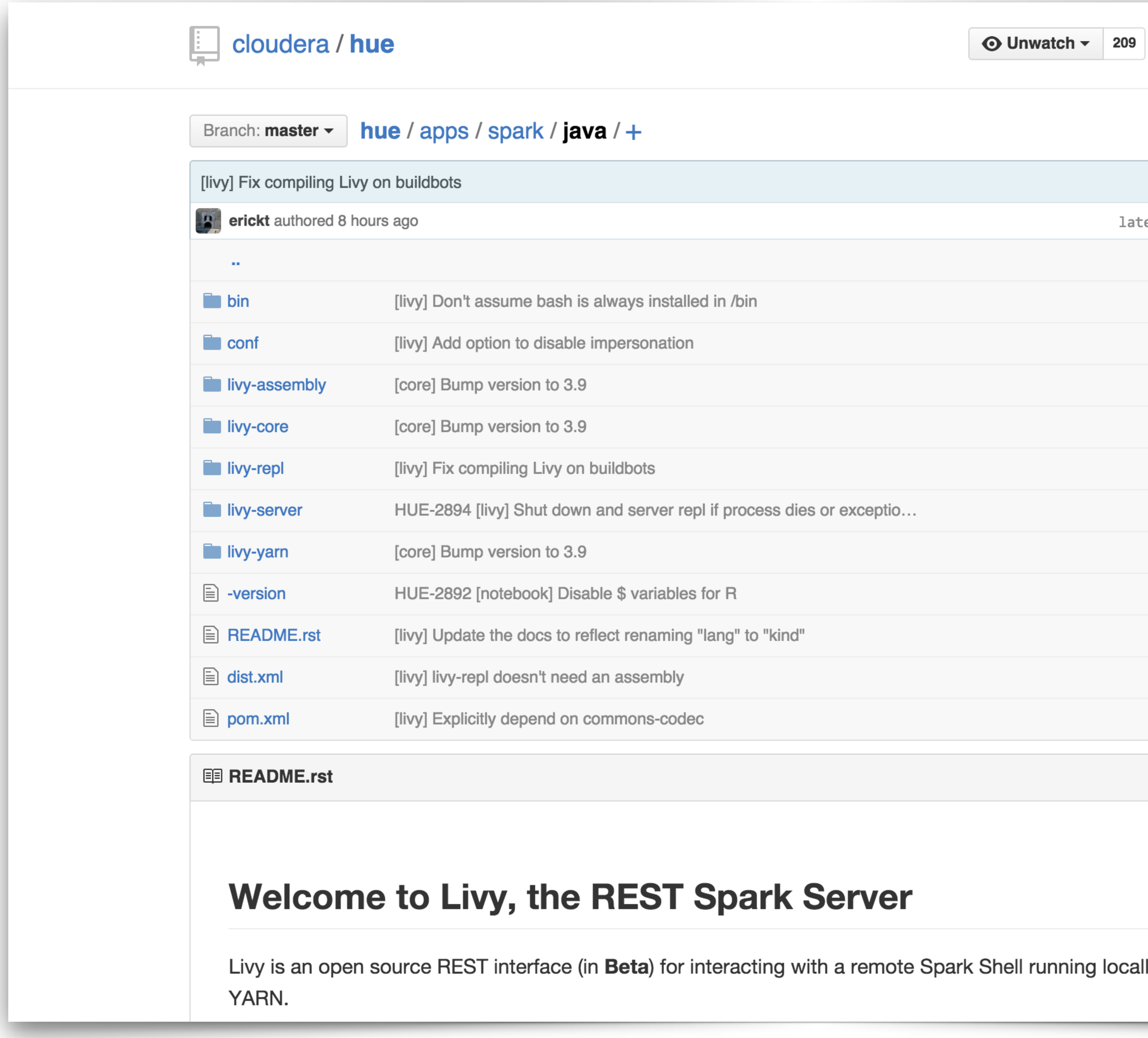
https://issues.cloudera.org/browse/HUE-2990

LIVY'S
CHEAT SHEET

- Open Source: https://github.com/cloudera/hue/tree/master/apps/spark/java
- Read about it: http://gethue.com/spark/

- Scala, Java, Python, R
- Type Introspection for Visualization
- YARN-cluster or local modes
- Code snippets / compiled
- REST API

- Pluggable backends
- Magic keywords
- Failure resilient
- Security



Branch: master

hue / apps / spark / java / +

[livy] Fix compiling Livy on buildbots

erickt authored 8 hours ago late

..

bin	[livy] Don't assume bash is always installed in /bin
conf	[livy] Add option to disable impersonation
livy-assembly	[core] Bump version to 3.9
livy-core	[core] Bump version to 3.9
livy-repl	[livy] Fix compiling Livy on buildbots
livy-server	HUE-2894 [livy] Shut down and server repl if process dies or exceptio…
livy-yarn	[core] Bump version to 3.9
-version	HUE-2892 [notebook] Disable $ variables for R
README.rst	[livy] Update the docs to reflect renaming "lang" to "kind"
dist.xml	[livy] livy-repl doesn't need an assembly
pom.xml	[livy] Explicitly depend on commons-codec

README.rst

Welcome to Livy, the REST Spark Server

Livy is an open source REST interface (in **Beta**) for interacting with a remote Spark Shell running locall
YARN.

BEDANKT!

WEBSITE

http://gethue.com

LEARN

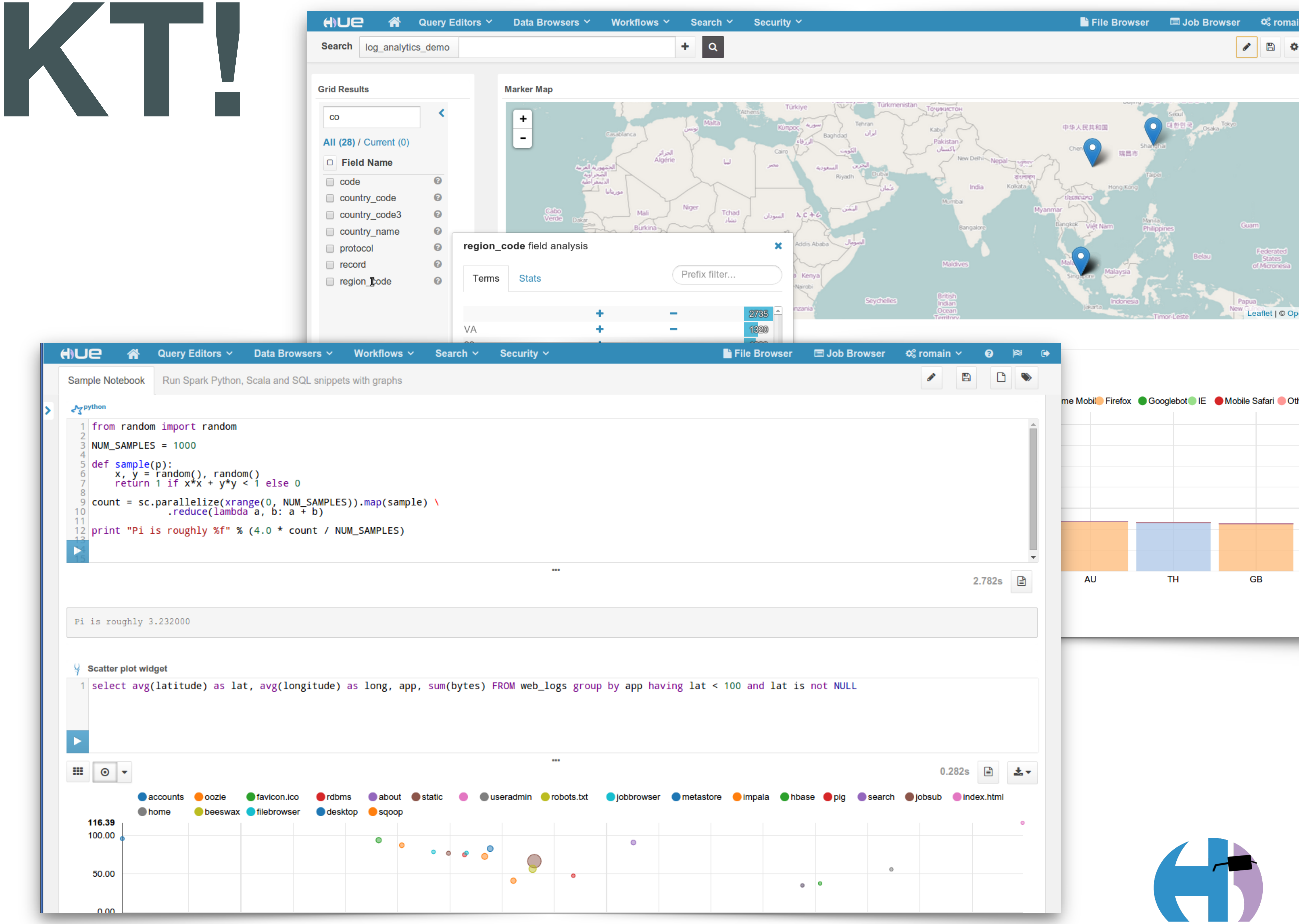
http://learn.gethue.com

TWITTER

@gethue

USER GROUP

hue-user@



```python
from random import random

NUM_SAMPLES = 1000

def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(xrange(0, NUM_SAMPLES)).map(sample) \
             .reduce(lambda a, b: a + b)

print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

```
Pi is roughly 3.232000
```

```sql
select avg(latitude) as lat, avg(longitude) as long, app, sum(bytes) FROM web_logs group by app having lat < 100 and lat is not NULL
```