玩转ApsaraDB HBase内嵌Spark系列(1)--通过Apache Zeppelin 快速实现交互式查询2

前言

目前ApsaraDB HBase数据库内嵌了Spark引擎，Spark对外提供了两种访问方式：

- LivyServer服务：用来提交生产作业，包括PySpark、scala/java的jar包作业、Streaming作业、生产级别离线SQL作业；
- ThriftServer SQL服务：对外提供了JDBC接口以及Beeline命令行的方式提交SQL;

不过在pyspark、scala/java等作业开发阶段，用户期望有一个所见即所得的交互式开发测试环境，这时候我们可以使用Apache Zeppelin对接LivyServer来搭建交互式工作台:

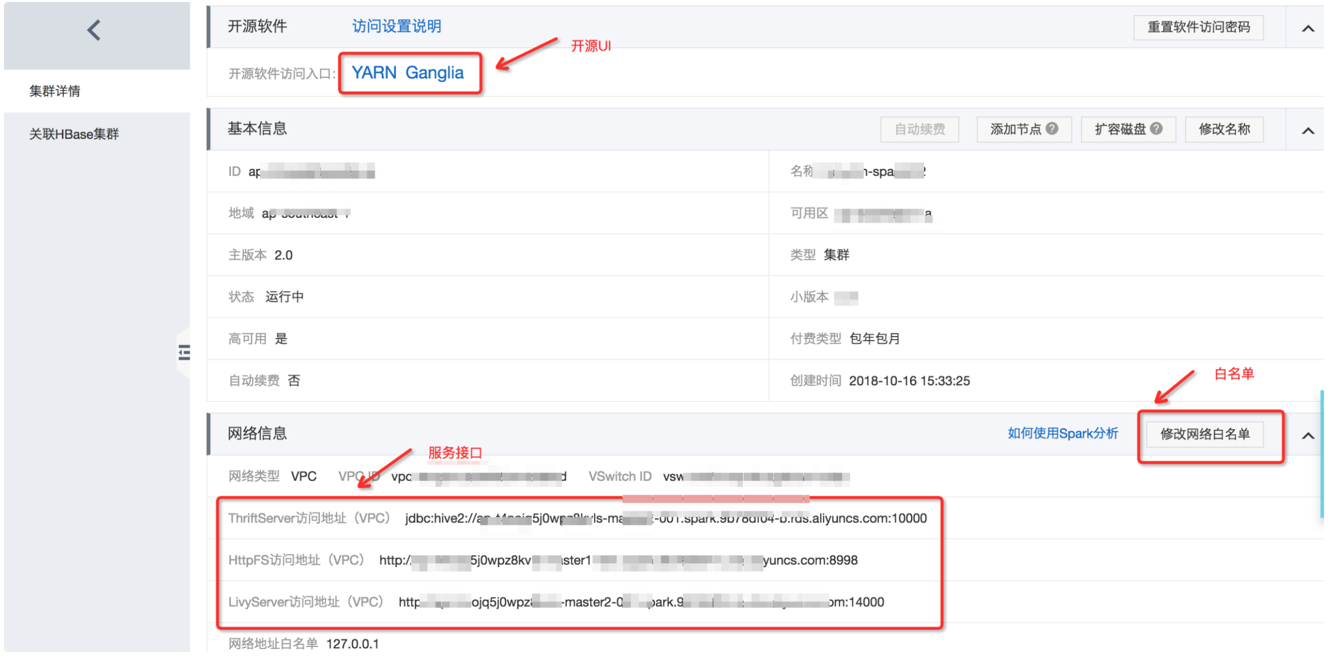
- Zeppelin简介：Zeppelin是一个Web笔记形式的交互式数据查询分析工具，可以在线用scala、SQL、python对Spark的数据进行查询分析并生成报表。 Zeppelin也可以支持其他引擎，比如JDBC系列的引擎、hbase、phoenix等。

Zeppelin服务搭建

1. 准备一台搭建Apache Zeppelin的ECS 因为Apache Zeppelin本身是一个java服务，需要一台ECS搭建。该ECS需要和云HBase的Spark集群在同一个VPC网络中。
2. 下载zeppelin的bin-all版本的包：https://zeppelin.apache.org/download.html
3. 配置zeppelin的服务端口 复制conf/zeppelin-site.xml.template为conf/zeppelin-site.xml，并修改文件中的zeppelin.server.port参数为想要的zeppelin的服务端口
4. 启动zeppelin服务 bin/zeppelin-daemon.sh start
5. 使用ECS的公网IP以及zeppelin端口在浏览器访问zeppelin服务 eg: ip:port

Zeppelin服务连通Spark服务

1. 获取Spark服务的LivyServer地址 在云HBase控制台找到对应Spark集群的页面，从服务接口处，获得LivyServer的服务访问地址，eg:”http://ap-xxx-001.spark.9b78df04-b.rds.aliyuncs.com:8998”



开源软件　访问设置说明　重置软件访问密码

开源软件访问入口: YARN Ganglia

集群详情

关联HBase集群

基本信息　自动续费　添加节点　扩容磁盘　修改名称

ID ap[redacted]	名称 [redacted]-spa[redacted]
地域 ap[redacted]	可用区 [redacted]a
主版本 2.0	类型 集群
状态 运行中	小版本 [redacted]
高可用 是	付费类型 包年包月
自动续费 否	创建时间 2018-10-16 15:33:25

网络信息　如何使用Spark分析　修改网络白名单

网络类型 VPC　VPC ID vpc[redacted]d　VSwitch ID vsw[redacted]

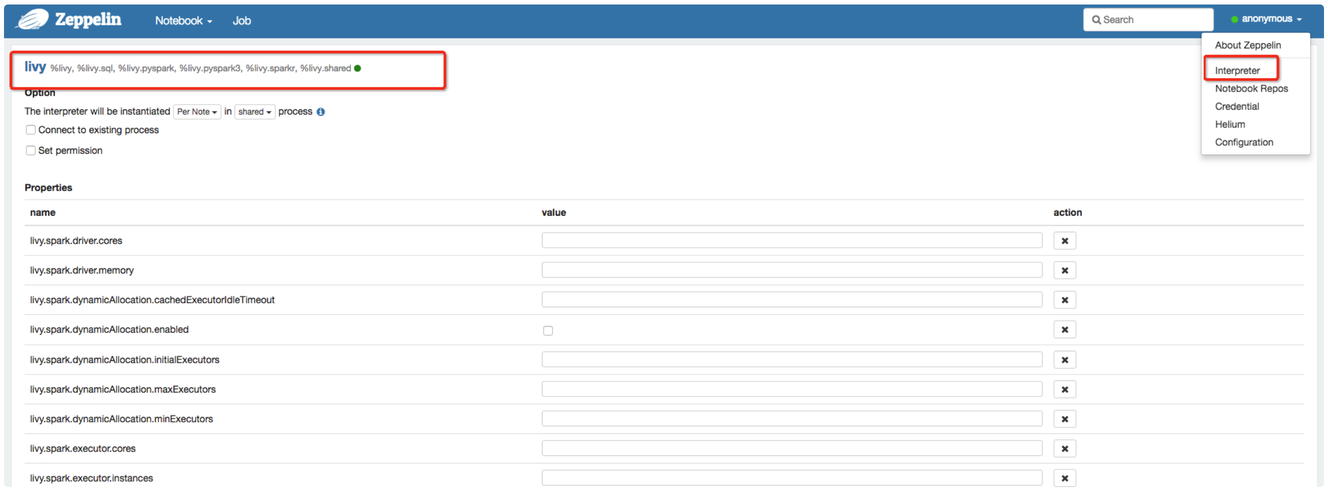
ThriftServer访问地址（VPC） jdbc:hive2://ap[redacted]5j0wp[redacted]ls-ma[redacted]-001.spark.9b78df04-b.rds.aliyuncs.com:10000

HttpFS访问地址（VPC） http:/[redacted]5j0wpz8kv[redacted]ster1[redacted]yuncs.com:8998

LivyServer访问地址（VPC） http[redacted]ojq5j0wpz[redacted]-master2-0[redacted]park.9[redacted]om:14000

网络地址白名单 127.0.0.1

2. 在zeppelin服务页面配置livy interpreter 将上面获取的LivyServer地址配置在zeppelin.livy.url条目中



livy %livy, %livy.sql, %livy.pyspark, %livy.pyspark3, %livy.sparkr, %livy.shared ●

Option

The interpreter will be instantiated Per Note in shared process

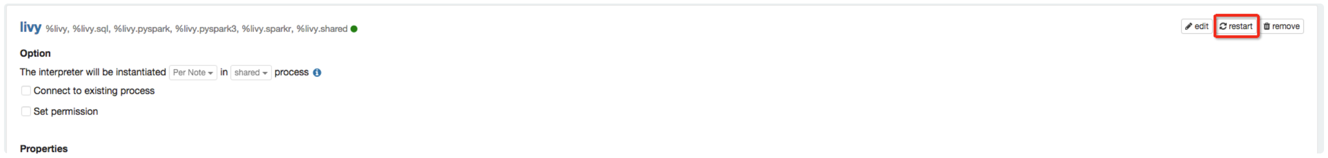
☐ Connect to existing process

☐ Set permission

Properties

name	value	action
livy.spark.driver.cores		×
livy.spark.driver.memory		×
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout		×
livy.spark.dynamicAllocation.enabled	☐	×
livy.spark.dynamicAllocation.initialExecutors		×
livy.spark.dynamicAllocation.maxExecutors		×
livy.spark.dynamicAllocation.minExecutors		×
livy.spark.executor.cores		×
livy.spark.executor.instances		×

3. 配置保存后点击”restart” livy interpreter



livy %livy, %livy.sql, %livy.pyspark, %livy.pyspark3, %livy.sparkr, %livy.shared ●

Option

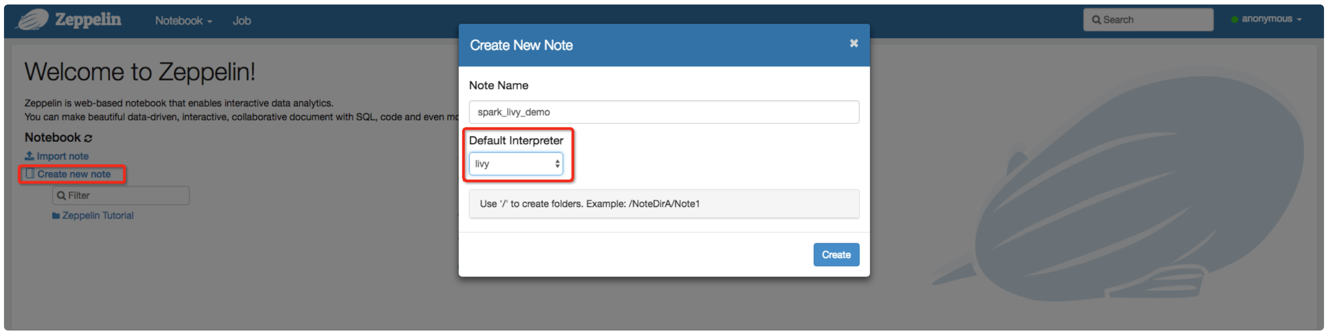
The interpreter will be instantiated Per Note in shared process

☐ Connect to existing process

☐ Set permission

Properties

4. 创建livy interpreter的notebook



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 mo

Notebook

Create New Note

Note Name: spark_livy_demo

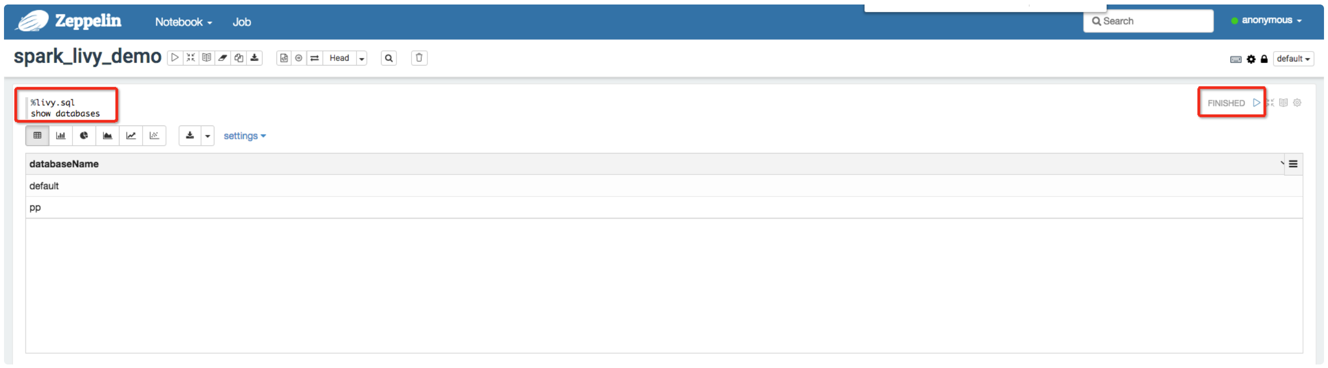
Default Interpreter: livy

Use '/' to create folders. Example: /NoteDirA/Note1

Create

%livy，%livy.sql，%livy.pyspark，%livy.pyspark3，%livy.sparkr

5. 测试服务是否正常 注意：每次启动首次运行notebook，因为要申请资源，需要等待一小会，后续不用等待。



spark_livy_demo

```
%livy.sql
show databases
```

FINISHED

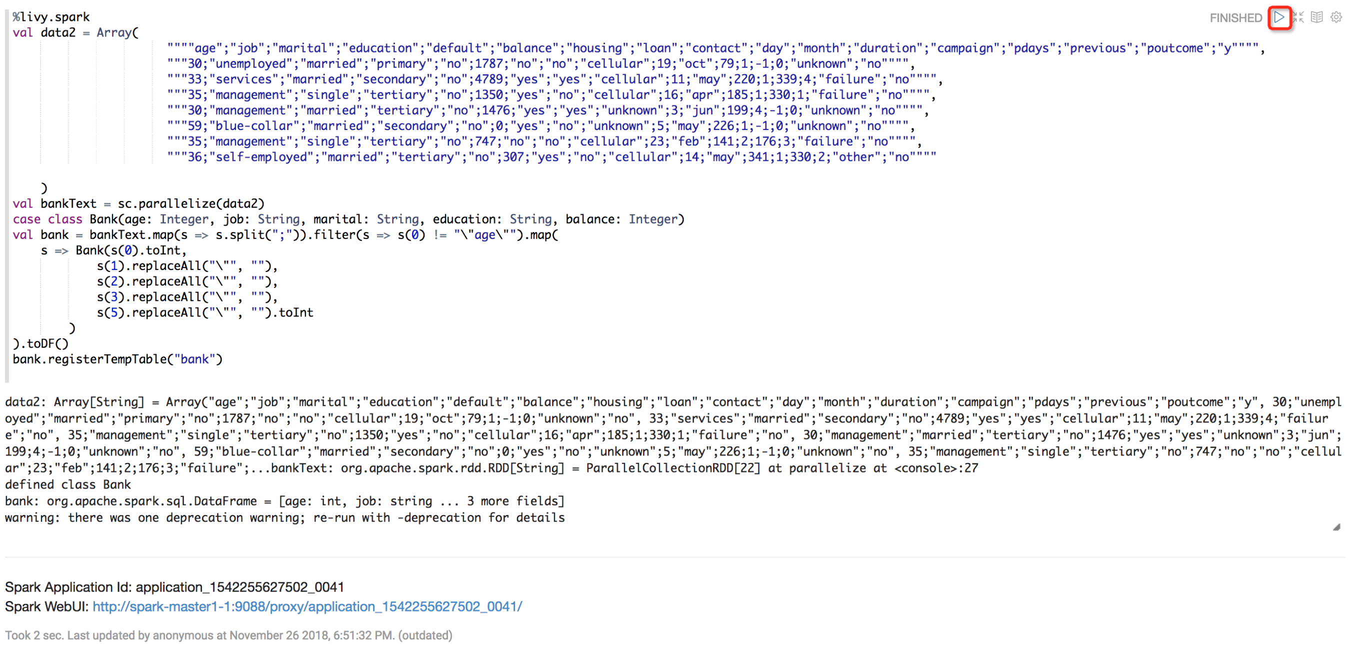
```
databaseName
default
pp
```

Zeppelin服务玩转Spark交互式查询

livy interpreter包含 %livy.spark，%livy.sql，%livy.pyspark，%livy.pyspark3，%livy.sparkr这几种解释器。其中%livy支持scala、java的code **注意：更多的zeppelin的例子参考aliyun-apsaradb-hbase-demo**

1. 使用%livy.spark准备一张临时数据表

```
%livy.spark
val data2 = Array(
                      """"age";"job";"marital";"education";"default";"balanc
e";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"pr
evious";"poutcome";"y"""",
                      """30;"unemployed";"married";"primary";"no";1787;"n
o";"no";"cellular";19;"oct";79;1;-1;0;"unknown";"no"""",
                      """33;"services";"married";"secondary";"no";4789;"ye
s";"yes";"cellular";11;"may";220;1;339;4;"failure";"no"""",
                      """35;"management";"single";"tertiary";"no";1350;"ye
s";"no";"cellular";16;"apr";185;1;330;1;"failure";"no"""",
                      """30;"management";"married";"tertiary";"no";1476;"ye
s";"yes";"unknown";3;"jun";199;4;-1;0;"unknown";"no"""",
                      """59;"blue-collar";"married";"secondary";"no";0;"ye
s";"no";"unknown";5;"may";226;1;-1;0;"unknown";"no"""",
                      """35;"management";"single";"tertiary";"no";747;"n
o";"no";"cellular";23;"feb";141;2;176;3;"failure";"no"""",
                      """36;"self-employed";"married";"tertiary";"no";30
7;"yes";"no";"cellular";14;"may";341;1;330;2;"other";"no""""

    )
val bankText = sc.parallelize(data2)
case class Bank(age: Integer, job: String, marital: String, education: String,
balance: Integer)
val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age\"").ma
p(
    s => Bank(s(0).toInt,
            s(1).replaceAll("\"", ""),
            s(2).replaceAll("\"", ""),
            s(3).replaceAll("\"", ""),
            s(5).replaceAll("\"", "").toInt
        )
).toDF()
bank.registerTempTable("bank")
```



```
%livy.spark
val data2 = Array(
                """"age";"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"poutcome";"y"""",
                """30;"unemployed";"married";"primary";"no";1787;"no";"no";"cellular";19;"oct";79;1;-1;0;"unknown";"no"""",
                """33;"services";"married";"secondary";"no";4789;"yes";"yes";"cellular";11;"may";220;1;339;4;"failure";"no"""",
                """35;"management";"single";"tertiary";"no";1350;"yes";"no";"cellular";16;"apr";185;1;330;1;"failure";"no"""",
                """30;"management";"married";"tertiary";"no";1476;"yes";"yes";"unknown";3;"jun";199;4;-1;0;"unknown";"no"""",
                """59;"blue-collar";"married";"secondary";"no";0;"yes";"no";"unknown";5;"may";226;1;-1;0;"unknown";"no"""",
                """35;"management";"single";"tertiary";"no";747;"no";"no";"cellular";23;"feb";141;2;176;3;"failure";"no"""",
                """36;"self-employed";"married";"tertiary";"no";307;"yes";"no";"cellular";14;"may";341;1;330;2;"other";"no""""

    )
val bankText = sc.parallelize(data2)
case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)
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
        )
).toDF()
bank.registerTempTable("bank")
```

```
data2: Array[String] = Array("age";"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"poutcome";"y", 30;"unempl
oyed";"married";"primary";"no";1787;"no";"no";"cellular";19;"oct";79;1;-1;0;"unknown";"no", 33;"services";"married";"secondary";"no";4789;"yes";"yes";"cellular";11;"may";220;1;339;4;"failur
e";"no", 35;"management";"single";"tertiary";"no";1350;"yes";"no";"cellular";16;"apr";185;1;330;1;"failure";"no", 30;"management";"married";"tertiary";"no";1476;"yes";"yes";"unknown";3;"jun";
199;4;-1;0;"unknown";"no", 59;"blue-collar";"married";"secondary";"no";0;"yes";"no";"unknown";5;"may";226;1;-1;0;"unknown";"no", 35;"management";"single";"tertiary";"no";747;"no";"no";"cellul
ar";23;"feb";141;2;176;3;"failure";...bankText: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[22] at parallelize at <console>:27
defined class Bank
bank: org.apache.spark.sql.DataFrame = [age: int, job: string ... 3 more fields]
warning: there was one deprecation warning; re-run with -deprecation for details
```

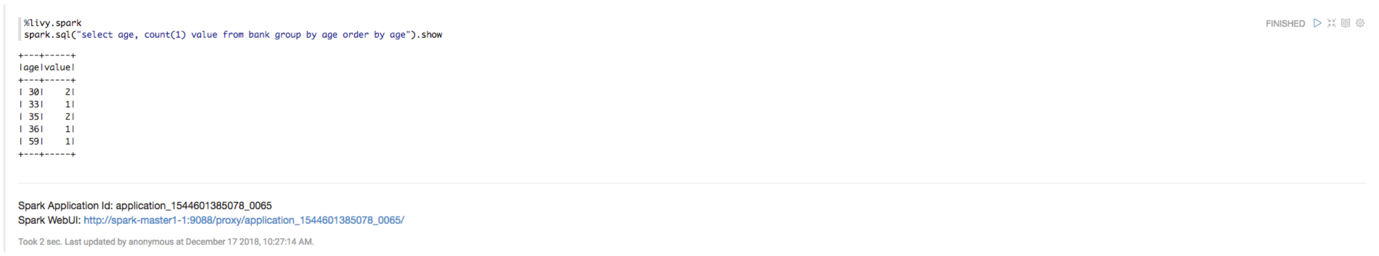
Spark Application Id: application_1542255627502_0041
Spark WebUI: http://spark-master1-1:9088/proxy/application_1542255627502_0041/

Took 2 sec. Last updated by anonymous at November 26 2018, 6:51:32 PM. (outdated)

2. 对于上面的临时表进行sql查询

```
%livy.spark
spark.sql("select age, count(1) value from bank group by age order by age").s
how
```

数据显示：



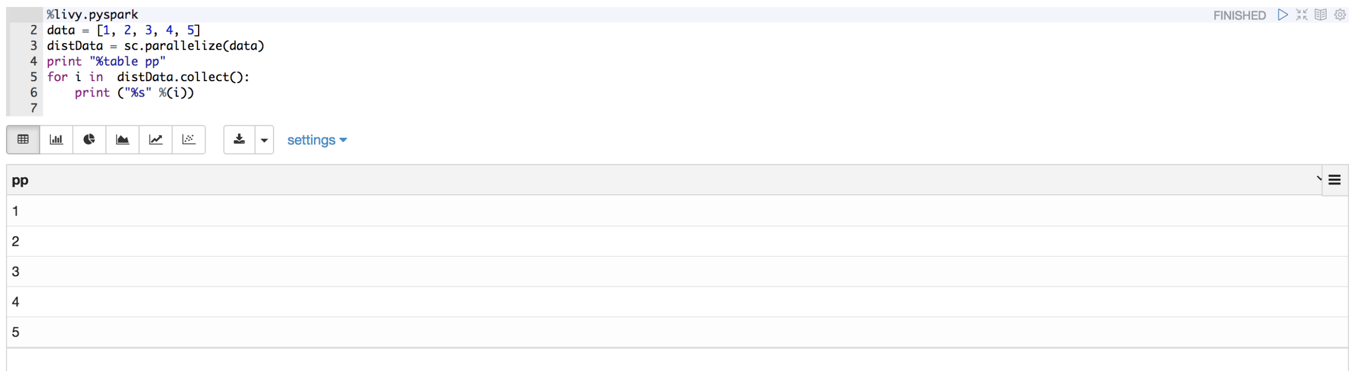
```
%livy.spark
spark.sql("select age, count(1) value from bank group by age order by age").show

+---+-----+
|age|value|
+---+-----+
| 30|    2|
| 33|    1|
| 35|    2|
| 36|    1|
| 59|    1|
+---+-----+
```

Spark Application Id: application_1544601385078_0065
Spark WebUI: http://spark-master1-1:9088/proxy/application_1544601385078_0065/

Took 2 sec. Last updated by anonymous at December 17 2018, 10:27:14 AM.

1. 使用%livy.pyspark来写pyspark代码

```
%livy.pyspark
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data)
print "%table pp"
for i in  distData.collect():
    print ("%s" %(i))
```



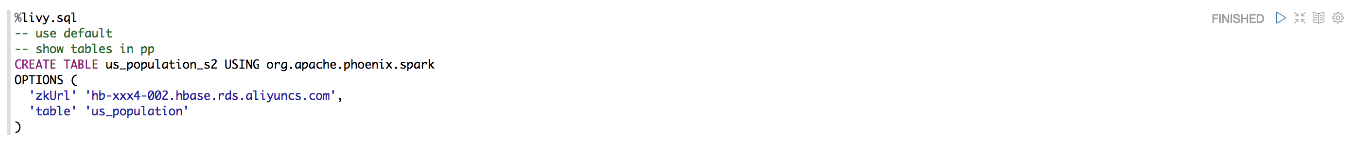
```
%livy.pyspark
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data)
print "%table pp"
for i in  distData.collect():
    print ("%s" %(i))
```

pp
1
2
3
4
5

Spark交互式分析HBase/phoenix表

1. 参考Phoniex的文档创建一张测试表us_population,然后使用Spark表us_population_s2关联该phoenix表

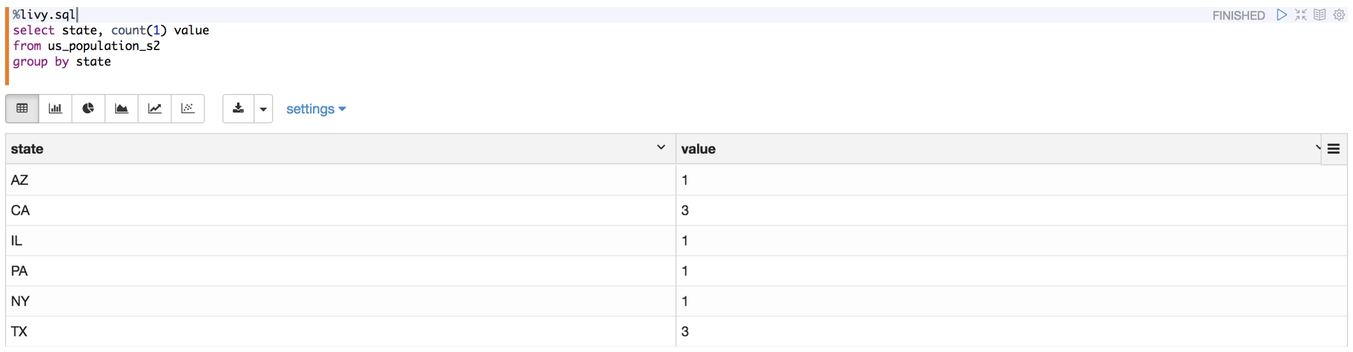
```
%livy.sql
CREATE TABLE us_population_s2 USING org.apache.phoenix.spark
OPTIONS (
  'zkUrl' 'hb-xxx-002.hbase.rds.aliyuncs.com',
  'table' 'us_population'
)
```



```
%livy.sql
-- use default
-- show tables in pp
CREATE TABLE us_population_s2 USING org.apache.phoenix.spark
OPTIONS (
  'zkUrl' 'hb-xxx4-002.hbase.rds.aliyuncs.com',
  'table' 'us_population'
)
```

2. 通过Spark表us_population_s2分析phoenix表us_population中的数据

```
select state, count(1) value
from us_population_s2
group by state
```



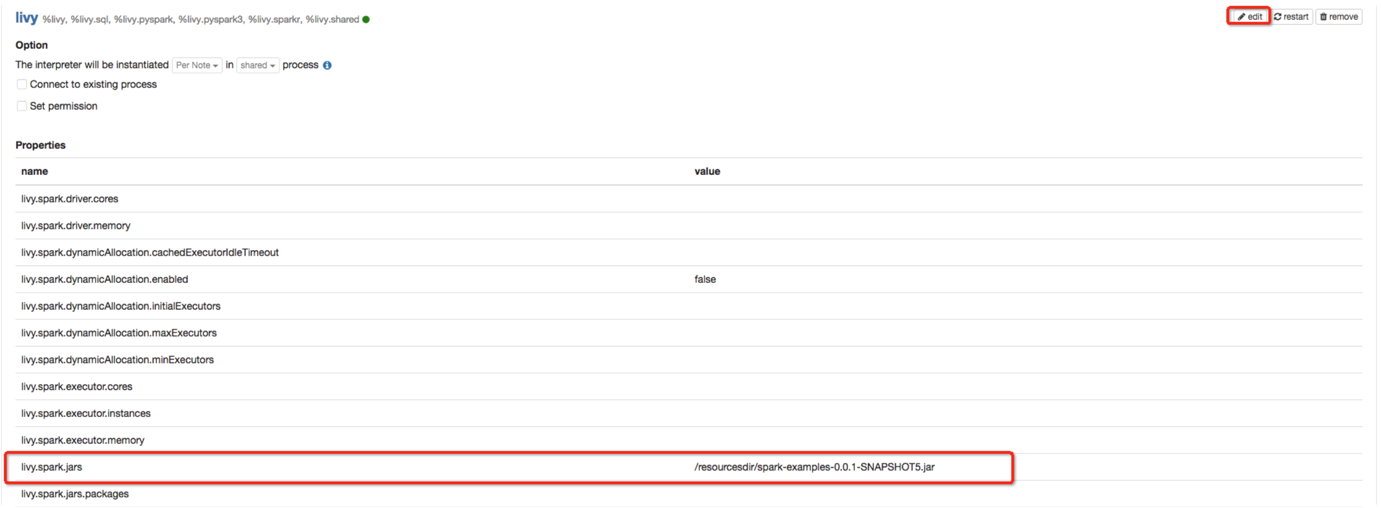
```
%livy.sql
select state, count(1) value
from us_population_s2
group by state
```

state	value
AZ	1
CA	3
IL	1
PA	1
NY	1
TX	3

Zeppelin中的livy interpreter添加外部三方包的依赖

1. 通过httpFS的方式，将jar包上传到对应的目录
2. 交互式的session中如果依赖三方库，可以通过下面的参数，添加步骤1对应目录的依赖库。然后重启

参数	spark-submit命令	value
livy.spark.jars	--jars	逗号隔开的路径
livy.spark.submit.pyFiles	--py-files	逗号隔开的路径



livy %livy, %livy.sql, %livy.pyspark, %livy.pyspark3, %livy.sparkr, %livy.shared ●

Option

The interpreter will be instantiated Per Note in shared process

Connect to existing process

Set permission

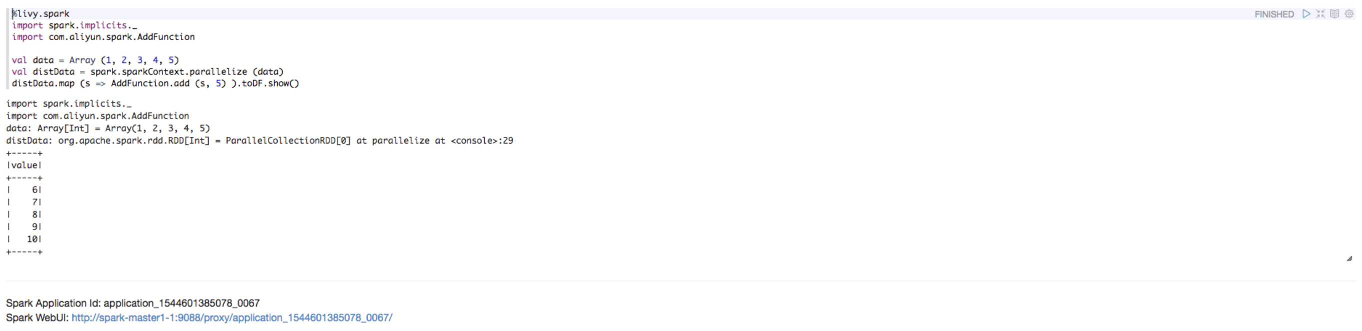
Properties

name	value
livy.spark.driver.cores	
livy.spark.driver.memory	
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout	
livy.spark.dynamicAllocation.enabled	false
livy.spark.dynamicAllocation.initialExecutors	
livy.spark.dynamicAllocation.maxExecutors	
livy.spark.dynamicAllocation.minExecutors	
livy.spark.executor.cores	
livy.spark.executor.instances	
livy.spark.executor.memory	
livy.spark.jars	/resourcesdir/spark-examples-0.0.1-SNAPSHOT5.jar
livy.spark.jars.packages	

1. 例子 a、从 https://github.com/aliyun/aliyun-apsaradb-hbase-demo/tree/master/spark/example-dependency 打包一个三方库的jar：example-dependency-0.0.1-SNAPSHOT.jar，其中包含com.aliyun.spark.AddFunction类 b、上传到对应spark集群的 /resourcesdir/example-dependency-0.0.1-SNAPSHOT.jar 目录 c、参考上面配

置livy.spark.jars 为/resourcesdir/example-dependency-0.0.1-SNAPSHOT.jar d、运行下面代码测试com.aliyun.spark.AddFunction 类已经导入

```
%livy.spark
import spark.implicits._
import com.aliyun.spark.AddFunction

val data = Array (1, 2, 3, 4, 5)
val distData = spark.sparkContext.parallelize (data)
distData.map (s => AddFunction.add (s, 5) ).toDF.show()
```



```
%livy.spark
import spark.implicits._
import com.aliyun.spark.AddFunction

val data = Array (1, 2, 3, 4, 5)
val distData = spark.sparkContext.parallelize (data)
distData.map (s => AddFunction.add (s, 5) ).toDF.show()
```

```
import spark.implicits._
import com.aliyun.spark.AddFunction
data: Array[Int] = Array(1, 2, 3, 4, 5)
distData: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:29
+-----+
|value|
+-----+
|    6|
|    7|
|    8|
|    9|
|   10|
+-----+
```

Spark Application Id: application_1544601385078_0067
Spark WebUI: http://spark-master1-1:9088/proxy/application_1544601385078_0067/

欢迎加入社群交流

对Spark服务以及HBase有兴趣的用户可以加入钉钉群，每周有专家的技术分享及答疑：“HBase生态+Spark社区大群”申请加群：https://dwz.cn/Fvqv066s