一、Livy安装配置

1. 下载Livy

http://livy.incubator.apache.org/

2. 下载后解压

3. 设置环境变量

https://github.com/cloudera/livy#building-livy

```
export SPARK_HOME=/usr/lib/spark
export HADOOP_CONF_DIR=/etc/hadoop/conf
```

生产环境的配置：

```
export SPARK_HOME=/opt/cloudera/parcels/SPARK2-2.2.0.cloudera2-1.cdh5.12.0.p0.232957/lib/spark2
export HADOOP_CONF_DIR=/etc/hadoop/conf
```

4. 修改livy的配置

在conf目录下面， 根据模板生成配置文件。

```
cp livy.conf.template livy.conf
```

```
# What spark master Livy sessions should use.
livy.spark.master = local

# What spark deploy mode Livy sessions should use.
# livy.spark.deploy-mode =

# List of local directories from where files are allowed to be added to user sessions. By
# default it's empty, meaning users can only reference remote URIs when starting their
# sessions.
# livy.file.local-dir-whitelist =

```

5. 启动livy

```
./bin/livy-server
```

livy启动后， 默认的端口是8998

二、Livy的使用--Session

（1）、查看当前的会话：

```
curl localhost:8998/sessions
```

$ curl localhost:8998/sessions
{"from":0,"total":0,"sessions":[]}

（2）、创建会话

```
$ curl -X POST --data '{"kind": "spark"}' -H "Content-Type:application/json" localhost:8998/sessions
```

[hadoop@spark123 ~]$ **curl -X POST --data '{"kind": "spark"}' -H "Content-Type:application/json" localhost:8998/sessions**
{"id":0,"appId":null,"owner":null,"proxyUser":null,"state":"starting","kind":"spark","appInfo":
{"driverLogUrl":null,"sparkUiUrl":null},"log":["stdout: ","\nstderr: "]}

创建会话后查看会话：
$ **curl localhost:8998/sessions**
{"from":0,"total":1,"sessions":[{"id":0,"appId":null,"owner":null,"proxyUser":null,"state":"idle","kind":"spark","appInfo":
{"driverLogUrl":null,"sparkUiUrl":null},"log":["18/04/08 20:56:22 INFO executor.Executor: Starting executor ID driver on host localhost","18/04/08 20:56:22 INFO executor.Executor: Using REPL class URI: http://192.168.9.13:39096","18/04/08 20:56:22 INFO util.Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 34000.","18/04/08 20:56:22 INFO netty.NettyBlockTransferService: Server created on 34000","18/04/08 20:56:22 INFO storage.BlockManagerMaster: Trying to register BlockManager","18/04/08 20:56:22 INFO storage.BlockManagerMasterEndpoint: Registering block manager localhost:34000 with 511.1 MB RAM, BlockManagerId(driver, localhost, 34000)","18/04/08 20:56:22 INFO storage.BlockManagerMaster: Registered BlockManager","18/04/08 20:56:23 INFO scheduler.EventLoggingListener: Logging events to hdfs://spark123:8020/system/spark/sparklogs/local-1523192182028","18/04/08 20:56:23 INFO driver.SparkEntries: Spark context finished initialization in 3203ms","18/04/08 20:56:25 INFO driver.SparkEntries: Created SQLContext."]}]}

（3）、查看某个会话的状态
查看id为0的会话状态：

```
curl localhost:8998/sessions/0
```

使用python格式化查看：
$ **curl localhost:8998/sessions/0 | python -m json.tool**
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
106  1167  106  1167    0     0  13464      0 --:--:-- --:--:-- --:--:-- 46680
{
```

```
    "appId": null,
    "appInfo": {
        "driverLogUrl": null,
        "sparkUiUrl": null
    },
    "id": 0,
    "kind": "spark",
    "log": [
        "18/04/08 20:56:22 INFO executor.Executor: Starting executor ID driver on host localhost",
        "18/04/08 20:56:22 INFO executor.Executor: Using REPL class URI: http://192.168.9.13:39096",
        "18/04/08 20:56:22 INFO util.Utils: Successfully started service
'org.apache.spark.network.netty.NettyBlockTransferService' on port 34000.",
        "18/04/08 20:56:22 INFO netty.NettyBlockTransferService: Server created on 34000",
        "18/04/08 20:56:22 INFO storage.BlockManagerMaster: Trying to register BlockManager",
        "18/04/08 20:56:22 INFO storage.BlockManagerMasterEndpoint: Registering block manager localhost:34000
with 511.1 MB RAM, BlockManagerId(driver, localhost, 34000)",
        "18/04/08 20:56:22 INFO storage.BlockManagerMaster: Registered BlockManager",
        "18/04/08 20:56:23 INFO scheduler.EventLoggingListener: Logging events to
hdfs://spark123:8020/system/spark/sparklogs/local-1523192182028",
        "18/04/08 20:56:23 INFO driver.SparkEntries: Spark context finished initialization in 3203ms",
        "18/04/08 20:56:25 INFO driver.SparkEntries: Created SQLContext."
    ],
    "owner": null,
    "proxyUser": null,
    "state": "idle"
}
```


（4）、提交任务

```
curl localhost:8998/sessions/0/statements -X POST -H 'Content-Type: application/json' -d '{"code":"1 + 1"}'
```

返回提交的任务id：

$ **curl localhost:8998/sessions/0/statements -X POST -H 'Content-Type: application/json' -d '{"code":"1 + 3"}'**

{"id":1,"code":"1 + 3","state":"waiting","output":null,"progress":0.0}

根据任务ID查看执行结果：

查询会话id为0， 任务id为1的执行结果：

```
curl localhost:8998/sessions/0/statements/1
```

$ **curl localhost:8998/sessions/0/statements/1**

{"id":1,"code":"1 + 3","state":"available","output":{"status":"ok","execution_count":1,"data":{"text/plain":"res1: Int = 4\n"}},"progress":1.0}

提交Spark任务：

统计RDD的数据集的数量：

```
curl localhost:8998/sessions/0/statements -X POST -H 'Content-Type: application/json' -d
'{"code":"sc.makeRDD(List(1,2,3,4)).count"}'
```

```
curl localhost:8998/sessions/0/statements/4
```

curl localhost:8998/sessions/0/statements/4
{"id":4,"code":"sc.makeRDD(List(1,2,3,4)).count","state":"available","output":{"status":"ok","execution_count":4,"data":{"text/plain":**"res4: Long = 4**\n"}},"progress":1.0}

（5）、共享变量
会话1创建一个rdd：

```
curl localhost:8998/sessions/0/statements -X POST -H 'Content-Type: application/json' -d '{"code":"val
rdd = sc.makeRDD(List((\"a\",1),(\"b\",2)))"}'
```

会话2统计rdd的count：

```
curl localhost:8998/sessions/0/statements -X POST -H 'Content-Type: application/json' -d
'{"code":"rdd.count"}'
```

curl localhost:8998/sessions/0/statements/9
{"id":9,"code":"rdd.count","state":"available","output":{"status":"ok","execution_count":9,"data":{"text/plain":**"res6: Long = 2\n**"}},"progress":1.0}

（6）、删除会话

```
curl localhost:8998/sessions/0 -X DELETE
```

（7）、Spark任务提交到YARN的配置参数

```
curl -X POST --data '{"kind": "spark", "numExecutors": 3, "executorMemory": "2G"}' -H "Content-Type:
application/json" localhost:8998/sessions
```

三、Livy的使用--Batch

这里修改livy的配置文件， 将提交模式修改为yarn：

livy.spark.master =yarn

默认是yarn client模式， 如果要修改为cluster模式， 修改livy.spark.deploy-mode 配置。

```
curl -X POST --data '{"file": "/tmp/myspark-1.0-SNAPSHOT.jar","jars":["/tmp/mysql-connector-java-5.1.44-bin.jar"], "className": "spark10.TestMyLivy"}' -H "Content-Type: application/json" localhost:8998/batches
```

```
curl -X POST --data '{"file": "/tmp/myspark-1.0-SNAPSHOT.jar","jars":["/tmp/mysql-connector-java-5.1.44-bin.jar"], "className": "spark10.TestMyLivy","conf":{"spark.master":"yarn","spark.submit.deployMode":"cluster"}}' -H "Content-Type: application/json" localhost:8998/batches
```

四、 使用python的rest方式

（1）、解压、编译、安装setuptools。

```
# tar -xzvf setuptools-0.6c11.tar.gz
# cd setuptools-0.6c11
# python setup.py build
# python setup.py install
```

（2）、安装pip
上传pip-9.0.1.tar.gz，解压后后安装

```
# tar -xzvf pip-9.0.1.tar.gz
# cd pip-9.0.1
# python setup.py install
```

（3）、安装request

```
pip install requests-2.13.0-py2.py3-none-any.whl
```

Here's a step-by-step example of interacting with Livy in Python with the Requests library. By default Livy runs on port 8998 (which can be changed with the `livy.server.port` config option). We'll start off with a Spark session that takes Scala code:

```
pip install requests
```

```
import json, pprint, requests, textwrap
host = 'http://localhost:8998'
data = {'kind': 'spark'}
headers = {'Content-Type': 'application/json'}
```

```
r = requests.post(host + '/sessions', data=json.dumps(data), headers=headers)
r.json()

{u'state': u'starting', u'id': 0, u'kind': u'spark'}
```

Once the session has completed starting up, it transitions to the idle state:

```
session_url = host + r.headers['location']
r = requests.get(session_url, headers=headers)
r.json()

{u'state': u'idle', u'id': 0, u'kind': u'spark'}
```

Now we can execute Scala by passing in a simple JSON command:

```
statements_url = session_url + '/statements'
data = {'code': '1 + 1'}
r = requests.post(statements_url, data=json.dumps(data), headers=headers)
r.json()

{u'output': None, u'state': u'running', u'id': 0}
```

If a statement takes longer than a few milliseconds to execute, Livy returns early and provides a statement URL that can be polled until it is complete:

```
statement_url = host + r.headers['location']
r = requests.get(statement_url, headers=headers)
pprint.pprint(r.json())

{u'id': 0,
  u'output': {u'data': {u'text/plain': u'res0: Int = 2'},
              u'execution_count': 0,
              u'status': u'ok'},
  u'state': u'available'}
```

That was a pretty simple example. More interesting is using Spark to estimate Pi. This is from the Spark Examples:

```
data = {
  'code': textwrap.dedent("""
    val NUM_SAMPLES = 100000;
    val count = sc.parallelize(1 to NUM_SAMPLES).map { i =>
      val x = Math.random();
      val y = Math.random();
      if (x*x + y*y < 1) 1 else 0
    }.reduce(_ + _);
    println(\"Pi is roughly \" + 4.0 * count / NUM_SAMPLES)
    """)
}

r = requests.post(statements_url, data=json.dumps(data), headers=headers)
pprint.pprint(r.json())

statement_url = host + r.headers['location']
r = requests.get(statement_url, headers=headers)
```

```
pprint.pprint(r.json())

{u'id': 1,
 u'output': {u'data': {u'text/plain': u'Pi is roughly 3.14004\nNUM_SAMPLES: Int = 100000\ncount: Int = 78501'},
             u'execution_count': 1,
             u'status': u'ok'},
 u'state': u'available'}
```

Finally, close the session:

```
session_url = 'http://localhost:8998/sessions/0'
requests.delete(session_url, headers=headers)

<Response [204]>
```