Multi-node Spark & Zeppelin

Lab 7
November 23th, 2017

Jun Heo(j.heo@snu.ac.kr)
Computer Science and Engineering
Seoul National University

***Slide credits**: Jonghyun Bae, Jun Heo, Jae W. Lee*

Index

- Multi-node Setting
- HiBench
- Ganglia
- Zeppelin
- Exercise

Student's VM IP List

- **Virtual Machine IP List**
  - https://docs.google.com/spreadsheets/d/1X9Uavr2PACqgfLC3rOcQ7Gqo4-NQNKoBhvcaL_86g9E/edit#gid=0

Exercise Answer (1)

- **The longest delay in this dataset? (2)**

```
>>> tripGraph.edges.groupBy().max("delay")
```

- **The number of delayed versus on-time/early flights (2)**

```
>>> tripGraph.edges.filter(" delay <= 0 ").count()
>>> tripGraph.edges.filter(" delay > 0 ").count()
```

- **What flights departing Seattle are most likely to have significant delays? (2)**
  - Seattle == 'SEA'

```
>>> tripGraph.edges.filter("src = 'SEA' and delay > 0").groupBy("src",
"dst").avg("delay").sort(desc("avg(delay)")).show(5)
```

Exercise Answer (2)

- **Top 5 busiest airports (most flights in and out) (2)**
  - Use vertex degrees

```
>>> tripGraph.degrees.sort(desc("degree")).show(5)
```

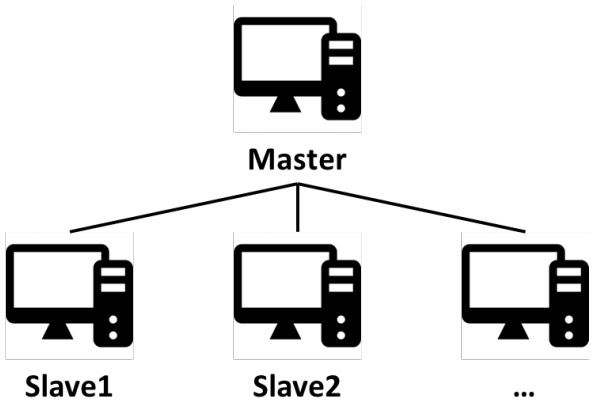
- **Airport ranking using PageRank (2)**
  - Reset probability=0.15, max iteration = 5

```
>>> ranks = tripGraph.pageRank(resetProbability=0.15, maxIter=5)
```

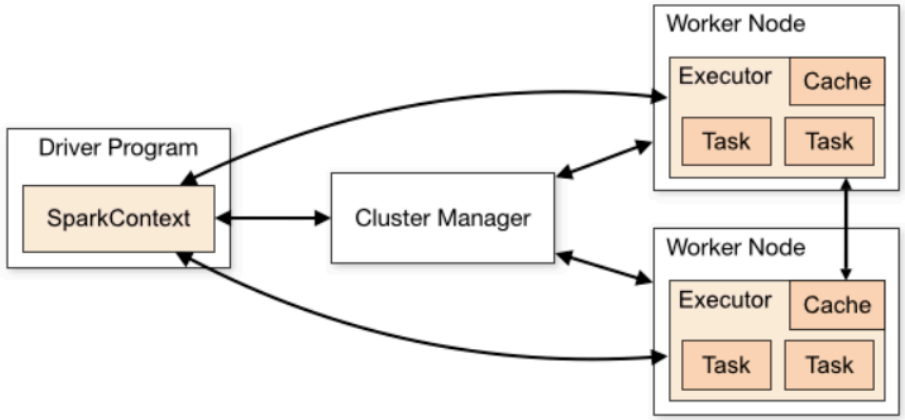
Multi-node Setting in Spark

Multi-node Setting: Overview

- **Cluster setup**
  - 1 master node + multiple slave nodes
    - Driver program
      - convert user program into tasks & schedule the tasks
    - Executors
      - run individual tasks in a given Spark job



General Spark Cluster



Multi-node Setting: Networking

- **Add new machine's hostname and IP**
  - xxx.xxx.xxx.xxx: virtual machine IP
    - ex) 172.31.1.25
  - HostName: host name of virtual machine
    - ex) ip-172-31-1-25
    - Master and slave

```
ubuntu@ip-x-x-x-x:~$ sudo vim /etc/hosts
```

/etc/hosts

```
127.0.0.1 localhost
xxx.xxx.xxx.xxx   Master_Host_Name
xxx.xxx.xxx.xxx   Slave_Host_Name
```

Multi-node Setting: SSH Access (1)

- **Set ubuntu password**

```
ubuntu@ip-x-x-x-x:~$ sudo passwd ubuntu
Enter new UNIX password:
Enter new UNIX password
passwd: password updated successfully
```

Multi-node Setting: SSH Access (2)

- **Password authentication yes**

```
ubuntu@ip-x-x-x-x:~$ sudo vim /etc/ssh/sshd_config
```

/etc/ssh/sshd_config

```
# Change to no to disable tunnelled clear text
PasswordAuthentication yes
```

Multi-node Setting: SSH Access (3)

- **Create public key**

```
ubuntu@ip-x-x-x-x:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
/home/ubuntu/.ssh/id_rsa already exists.
Overwrite (y/n)? y
```

Multi-node Setting: SSH Access (4)

- **Copy public key to new virtual machine**

```
ubuntu@ip-x-x-x-x:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub ubuntu@ip-x-x-x-x
ubuntu@ip-x-x-x-x's password:
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'ubuntu@ip-x-x-x-x'"
and check to make sure that only the key(s) you wanted were added.
```

- **Check network connection**

```
ubuntu@ip-x-x-x-x:~$ ssh ip-x-x-x-x
```

Multi-node setting: Hadoop Configuration (1)

- **Add new machine's IP (Master node only)**
  - ex) ip-172-31-15-72

```
ubuntu@ip-x-x-x-x:~$ vim $HADOOP_HOME/etc/hadoop/slaves
```

$HADOOP_HOME/etc/hadoop/slaves

```
Master_Host_Name
Slave_Host_Name
```

Multi-node setting: Hadoop Configuration (2)

- **Set hadoop configuration (new machine)**

```
ubuntu@ip-x-x-x-x:~$ vim $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

$HADOOP_HOME/etc/hadoop/hadoop-env.sh

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

Multi-node setting: Hadoop Configuration (3)

- **Set hadoop configuration (new machine)**

```
ubuntu@ip-x-x-x:~$ vim $HADOOP_HOME/etc/hadoop/core-site.xml
```

$HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
    <property>
        <name>fs.defaultFS<name>
        <value>hdfs://Master_Host_Name:9000</value>
    </property>
</configuration>
```

Multi-node setting: Hadoop Configuration (4)

- **Set hadoop configuration (new machine)**

```
ubuntu@ip-x-x-x-x:~$ vim $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
    <property>
        <name>dfs.replication<name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir<name>
        <value>file:/home/ubuntu/hadoop-2.7.4/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir<name>
        <value>file:/home/ubuntu/hadoop-2.7.4/hdfs/datanode</value>
    </property>
<configuration>
```

Multi-node setting: Hadoop Configuration (5)

- **Format name node (Master node only)**

```
ubuntu@ip-x-x-x-x:~/hadoop-2.7.4$ rm –rf hdfs
ubuntu@ip-x-x-x-x:~/hadoop-2.7.4$ bin/hdfs namenode -format
```

Multi-node setting: Hadoop Configuration (6)

- **Start hadoop**

```
ubuntu@ip-x-x-x-x:~/hadoop-2.7.4$ sbin/start-dfs.sh
```

- **Validation**

```
ubuntu@ip-x-x-x-x:~/hadoop-2.7.4$ jps
# Master node
10690 NameNode
10860 DataNode
11119 SecondaryNameNode
# Slave node
2458 DataNode
```

Multi-node setting: Spark Configuration (1)

- **Set spark configuration (new machine)**

```
ubuntu@ip-x-x-x-x:~$ vim $SPARK_HOME/conf/spark-env.sh
```

$SPARK_HOME/conf/spark-env.sh

```
export SPARK_WORKER_CORES=1
export SPARK_WORKER_MEMORY=10g
export SPARK_WORKER_INSTANCES=1
export SPARK_LOCAL_IP=Current_Machine_IP
export SPARK_MASTER_HOST=MASTER_IP
```

Multi-node setting: Spark Configuration (2)

- **Set spark configuration (new machine)**

```
ubuntu@ip-x-x-x-x:~$ vim $SPARK_HOME/conf/spark-defaults.conf
```

$SPARK_HOME/conf/spark-defaults.conf

```
spark.master            spark://MASTER_IP:7077
spark.driver.memory          2g
spark.executor.memory       10g
```

Multi-node setting: Spark Configuration (3)

- **Add new machine IP (Master node only)**

```
ubuntu@ip-x-x-x-x:~$ vim $SPARK_HOME/conf/slaves
```

$SPARK_HOME/conf/slaves

```
Master_IP
Slave_IP
```

Multi-node setting: Spark Configuration (4)

- **Start-all**

```
ubuntu@ip-x-x-x-x:~/spark-2.1.0$ sbin/start-all.sh
```

- **Validation**

```
ubuntu@ip-x-x-x-x:~/spark-2.1.0$ jps
# Master node
10690 NameNode
10860 DataNode
11119 SecondaryNameNode
11373 Master
11512 Worker
# Slave node
2458 DataNode
2690 Worker
```

Intel HiBench

- **Big data benchmark suite**
  - helps evaluate different big data frameworks in terms of speed, throughput and system resource utilizations
- **Types of workloads (19)**
  - Micro benchmark: sort, wordcount, tersort, …
  - Machine learning: bayesian classification, k-means clustering, …
  - SQL: scan, join, aggreagate
  - Websearch benchmark: pagerank, nutch indexing
  - …
- **https://github.com/intel-hadoop/HiBench**

HiBench: Installation (1)

- **Download HiBench**

```
ubuntu@ip-x-x-x-x:~$ git clone https://github.com/intel-hadoop/HiBench.git
```

- **Build**

```
ubuntu@ip-x-x-x-x:~$ sudo apt-get install maven
ubuntu@ip-x-x-x-x:~$ mvn -Psparkbench -Dspark=2.1 -Dscala=2.11 clean package
```

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] hibench ............................................ SUCCESS [4.622s]
[INFO] hibench-common ..................................... SUCCESS [1:49.776s]
[INFO] HiBench data generation tools ...................... SUCCESS [36.920s]
[INFO] sparkbench ......................................... SUCCESS [0.006s]
[INFO] sparkbench-common .................................. SUCCESS [41.949s]
[INFO] sparkbench micro benchmark ......................... SUCCESS [5.658s]
[INFO] sparkbench machine learning benchmark .............. SUCCESS [16.643s]
[INFO] sparkbench-websearch ............................... SUCCESS [4.170s]
[INFO] sparkbench-graph ................................... SUCCESS [14.790s]
[INFO] sparkbench-sql ..................................... SUCCESS [8.787s]
[INFO] sparkbench-streaming ............................... SUCCESS [6.965s]
[INFO] sparkbench project assembly ........................ SUCCESS [14.749s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4:25.234s
[INFO] Finished at: Tue Nov 21 12:47:17 UTC 2017
[INFO] Final Memory: 61M/450M
[INFO] ------------------------------------------------------------------------
ubuntu@ip-172-31-1-25:~/HiBench$
```

HiBench: Configuration (2)

- **Hadoop configuration in HiBench**

```
ubuntu@ip-x-x-x-x:~$ vim $HIBENCH_HOME/conf/hadoop.conf
```

$HIBENCH_HOME/conf/hadoop.conf

```
hibench.hadoop.home        /home/ubuntu/hadoop-2.7.4
hibench.hdfs.master        hdfs://Master_IP:9000
```

HiBench: Configuration (3)

- **Spark configuration in HiBench**

```
ubuntu@ip-x-x-x-x:~$ vim $HIBENCH_HOME/conf/spark.conf
```

$HIBENCH_HOME/conf/spark.conf

```
hibench.spark.home       /home/ubuntu/spark-2.1.0
hibench.spark.master     spark://Master_IP:7077
spark.executor.memory    10g
spark.driver.memory       2g
```

HiBench: Configuration (4)

- **HiBench configuration**

```
ubuntu@ip-x-x-x-x:~$ vim $HIBENCH_HOME/conf/hibench.conf
```

$HIBENCH_HOME/conf/hibench.conf

```
hibench.scale.profile       small
```

HiBench: WordCount (1)

- **Prepare workload's input**

```
ubuntu@ip-x-x-x-x:~/HiBench$
bin/workloads/micro/wordcount/prepare/prepare.sh
...
start HadoopPrepareWordcount bench
...
finish HadoopPrepareWordcount bench
```

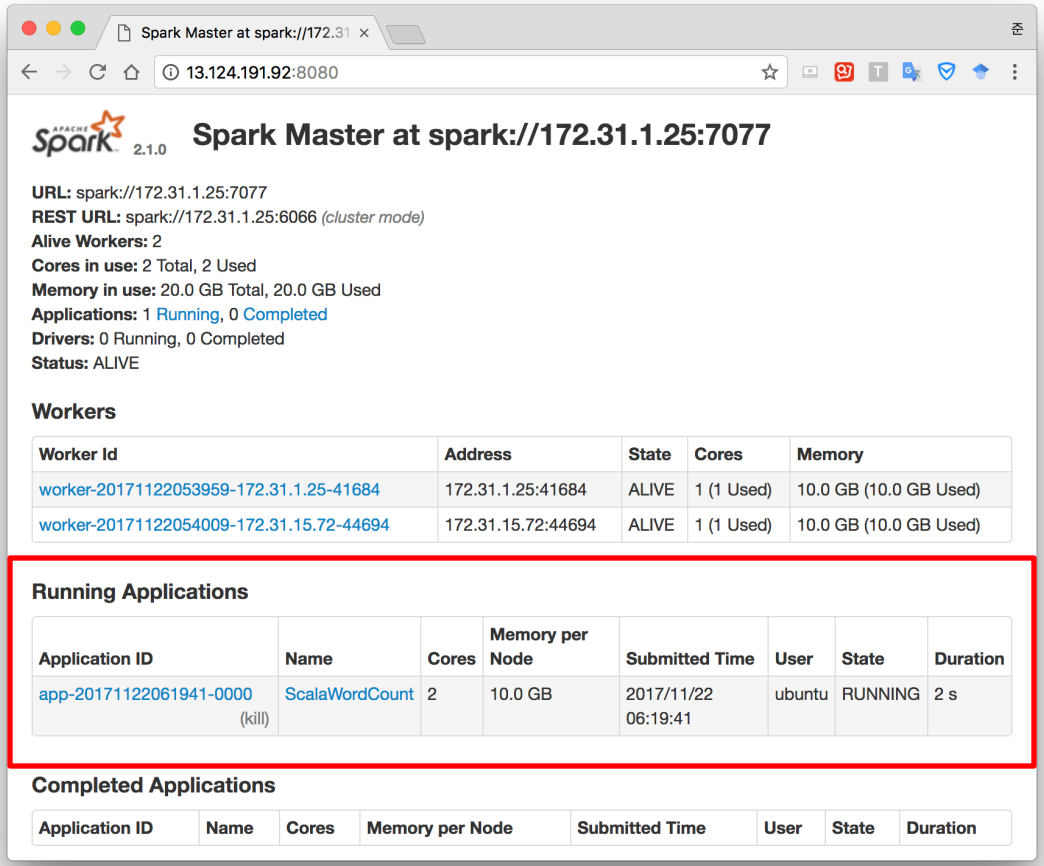
HiBench: WordCount (2)

- **Run workload**

```
ubuntu@ip-x-x-x-x:~/HiBench$ bin/workloads/micro/wordcount/spark/run.sh
...
start ScalaSparkWordcount bench
...
finish ScalaSparkWordcount bench
```

HiBench: WordCount (3)

- **Check running application**
  - http://Master_IP:8080



Profiling Tool: Ganglia

Ganglia: Installation

- **Install ganglia and related packages**

```
ubuntu@ip-x-x-x-x:~$ sudo apt-get install ganglia-monitor rrdtool gmetad
ganglia-webfrontend
```

- **Copy apache.conf**

```
ubuntu@ip-x-x-x-x:~$ sudo cp /etc/ganglia-webfrontend/apache.conf
/etc/apache2/sites-enabled/ganglia.conf
```

Ganglia: Configuration

- **Gmetad configuration**

```
ubuntu@ip-x-x-x-x:~$ vim /etc/ganglia/gmetad.conf
```

/etc/ganglia/gmetad.conf

```
data_source "my cluster" 60 Master_IP
```

Ganglia: Configuration

Gmond configuration

```
ubuntu@ip-x-x-x-x:~$ vim /etc/ganglia/gmond.conf
```

/etc/ganglia/gmond.conf

```
udp_send_channel {
    host = 172.31.1.25
    #mcast_join = 239.2.11.71
    port = 8649
    ttl = 1 }
...
udp_recv_channel {
    #mcast_join = 239.2.11.71
    port = 8649
    #bind = 239.2.11.71
```

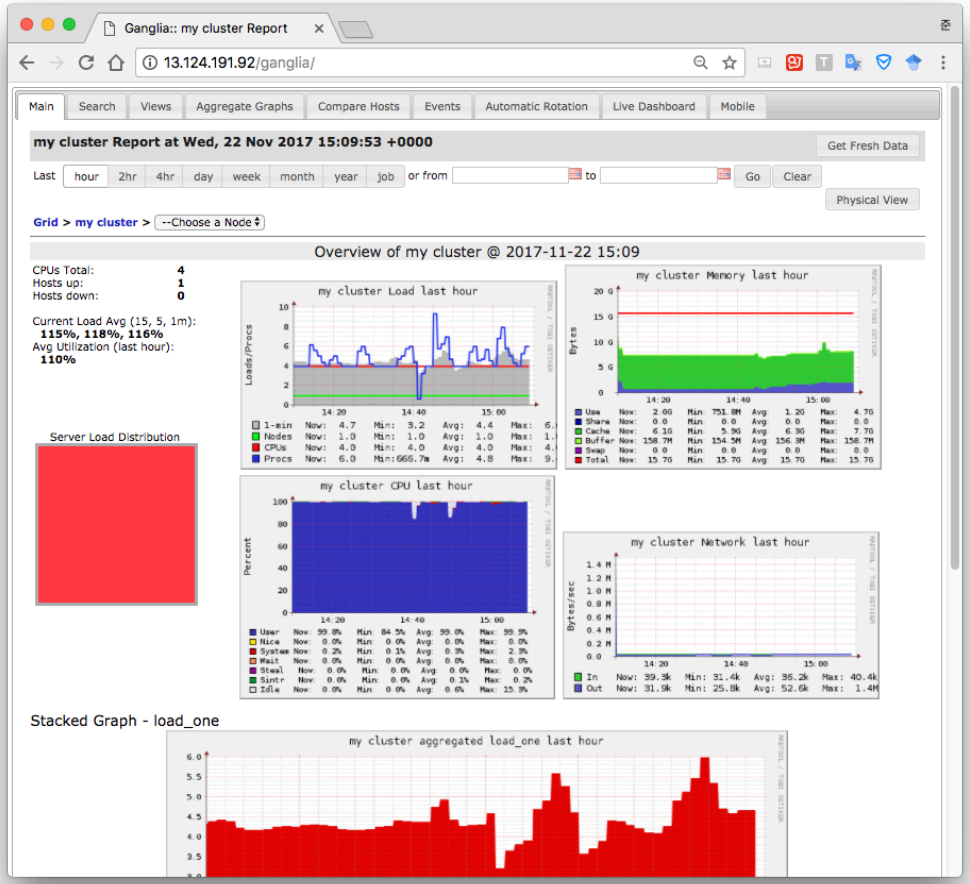
Ganglia

- **Restart ganglia-monitor, gmetad, and apache**

```
ubuntu@ip-x-x-x-x:~$ sudo service ganglia-monitor restart && sudo service gmetad restart
&& sudo service apache2 restart
```

Ganglia: Web UI

- **Start web UI**
  - http://Master_IP/ganglia



Zeppelin

Zeppelin: Installation

- **Download and unzip zeppelin.tgz (Master node)**

```
ubuntu@ip-x-x-x-x:~$ wget
http://apache.mirror.cdnetworks.com/zeppelin/zeppelin-0.7.3/zeppelin-0.7.3-bin-all.tgz
...
ubuntu@ip-x-x-x-x:~$ tar xvf zeppelin-0.7.3-bin-all.tgz
...
```

Zeppelin: Settings

- **Zeppelin configuration**

```
ubuntu@ip-x-x-x-x:~$ vim $ZEPPELIN_HOME/conf/zeppelin-env.sh
```

```
$ZEPPELIN_HOME/conf/zeppelin-env.sh
```

```
export ZEPPELIN_PORT=8082
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

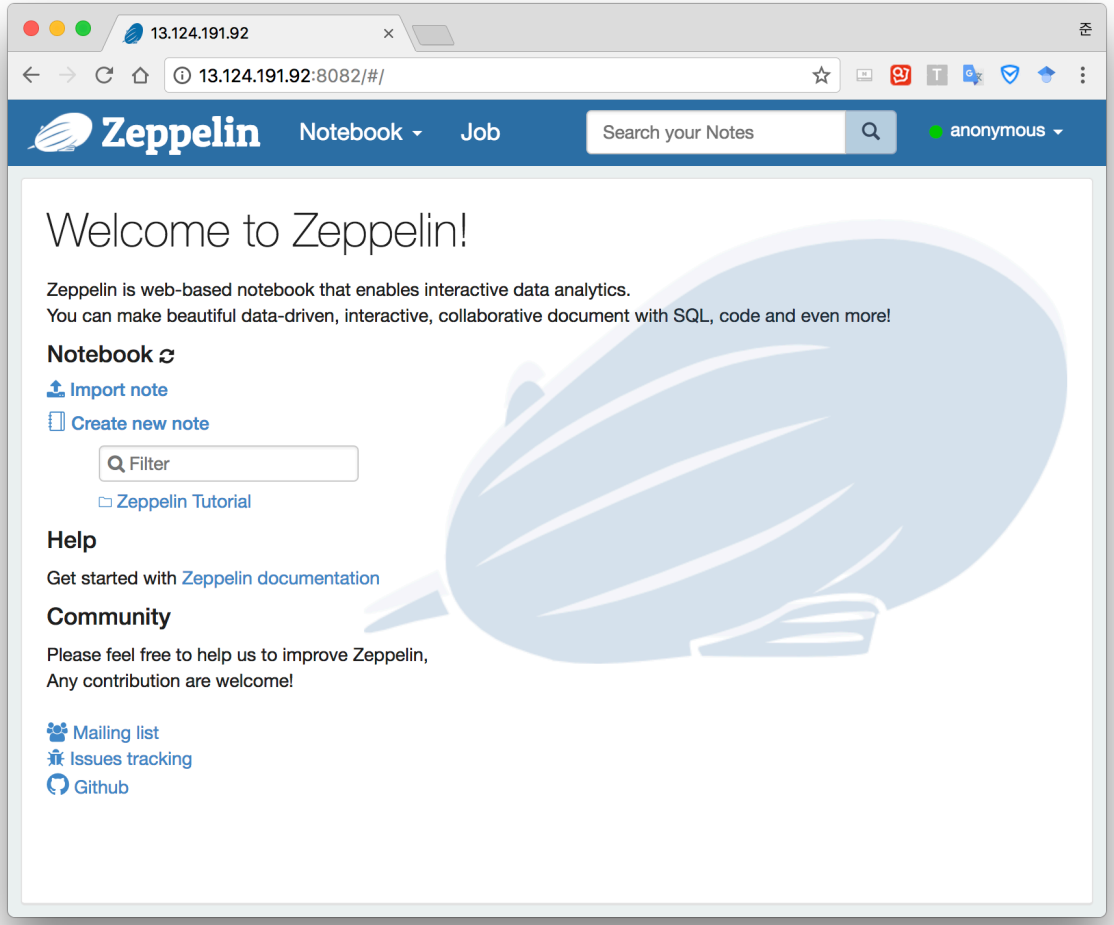
Zeppelin: Starting Daemon

- **Start Zeppelin daemon program**

```
ubuntu@ip-x-x-x-x:~$ bin/zeppelin-daemon.sh start
Zeppelin start       [OK]
```

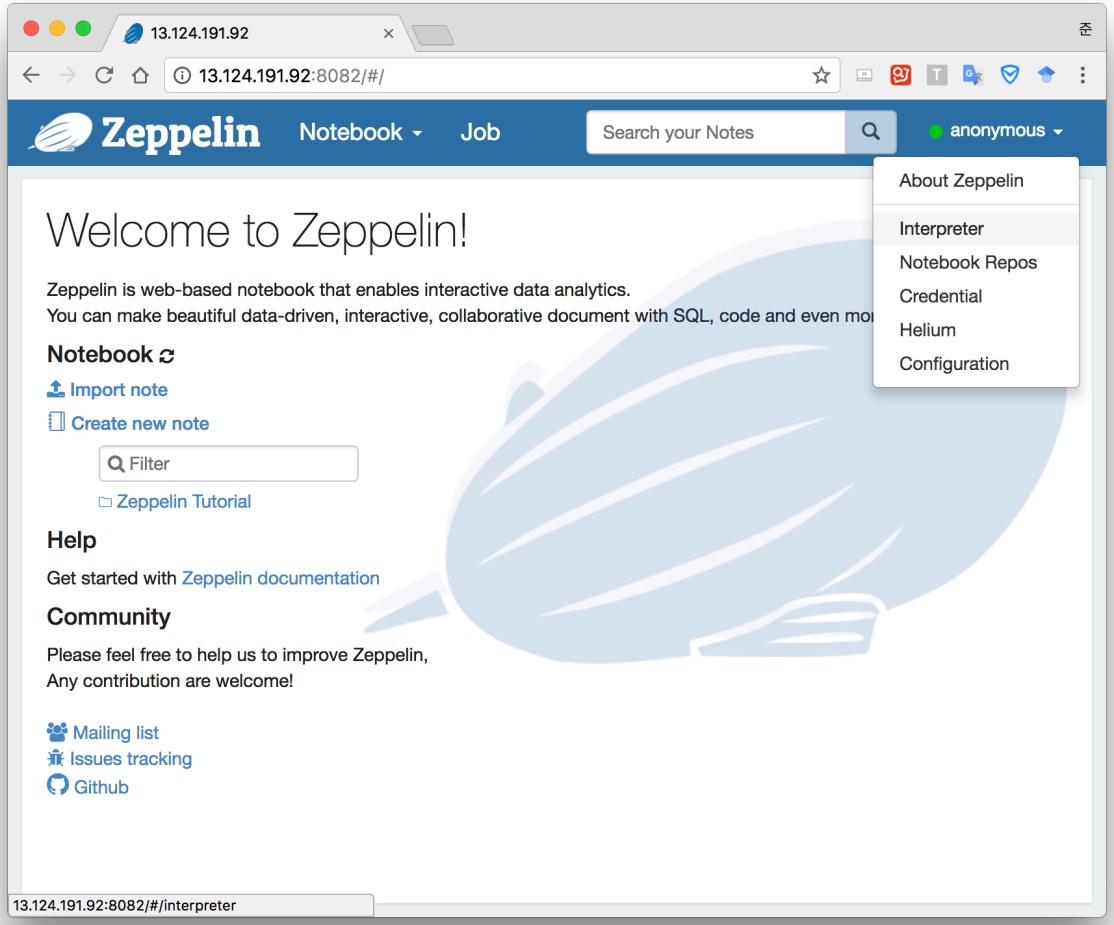
Zeppelin: Web UI

- **http://Master_IP:8082**



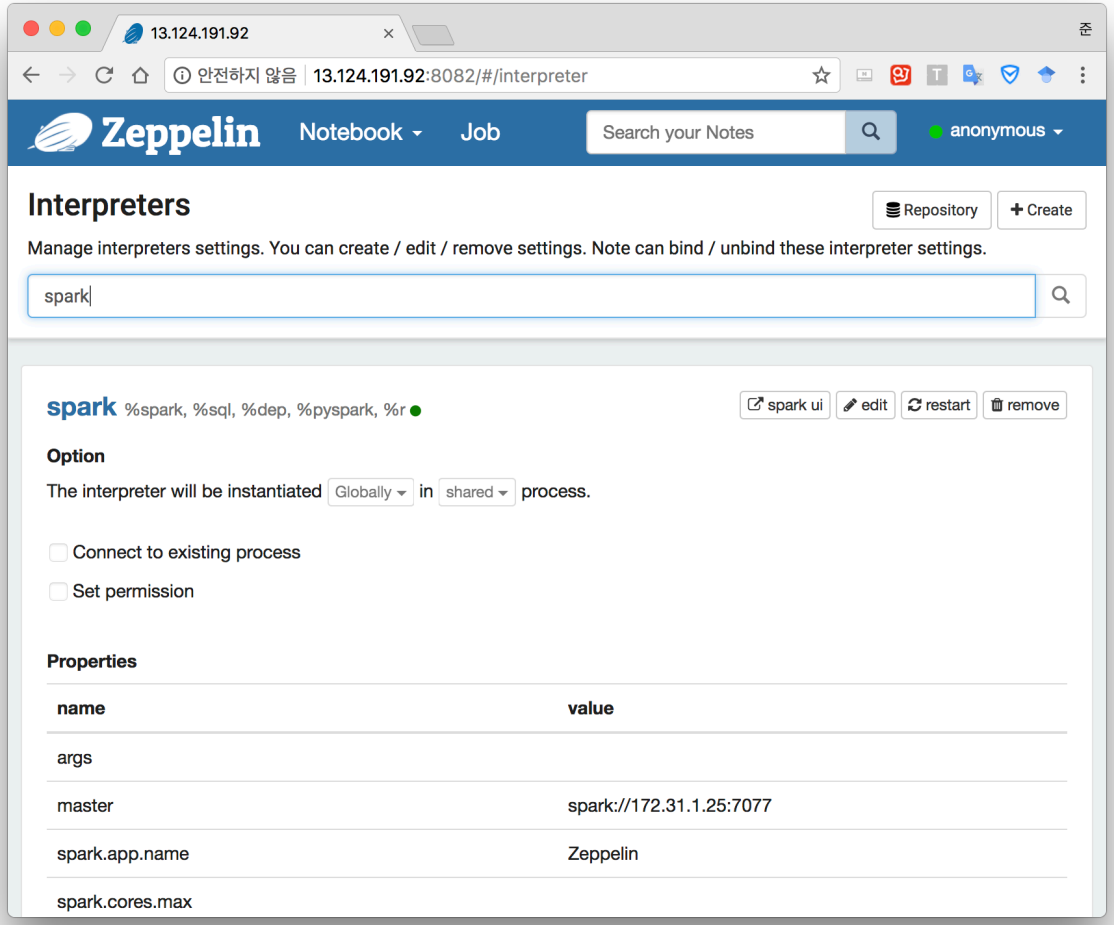
Zeppelin: Interpreter Setting (1)

- **Need to set interpreter configuration (Spark)**



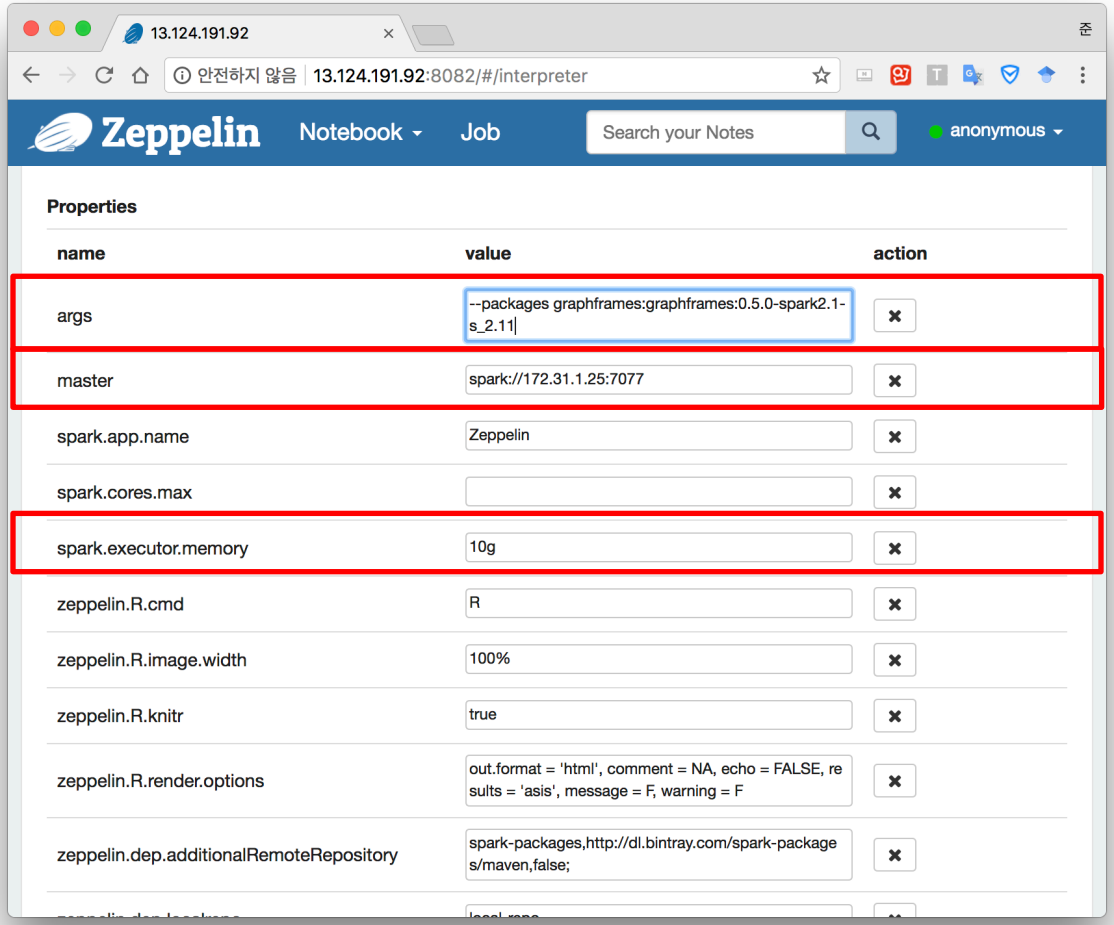
Zeppelin: Interpreter Setting (2)

- **Find spark interpreter**



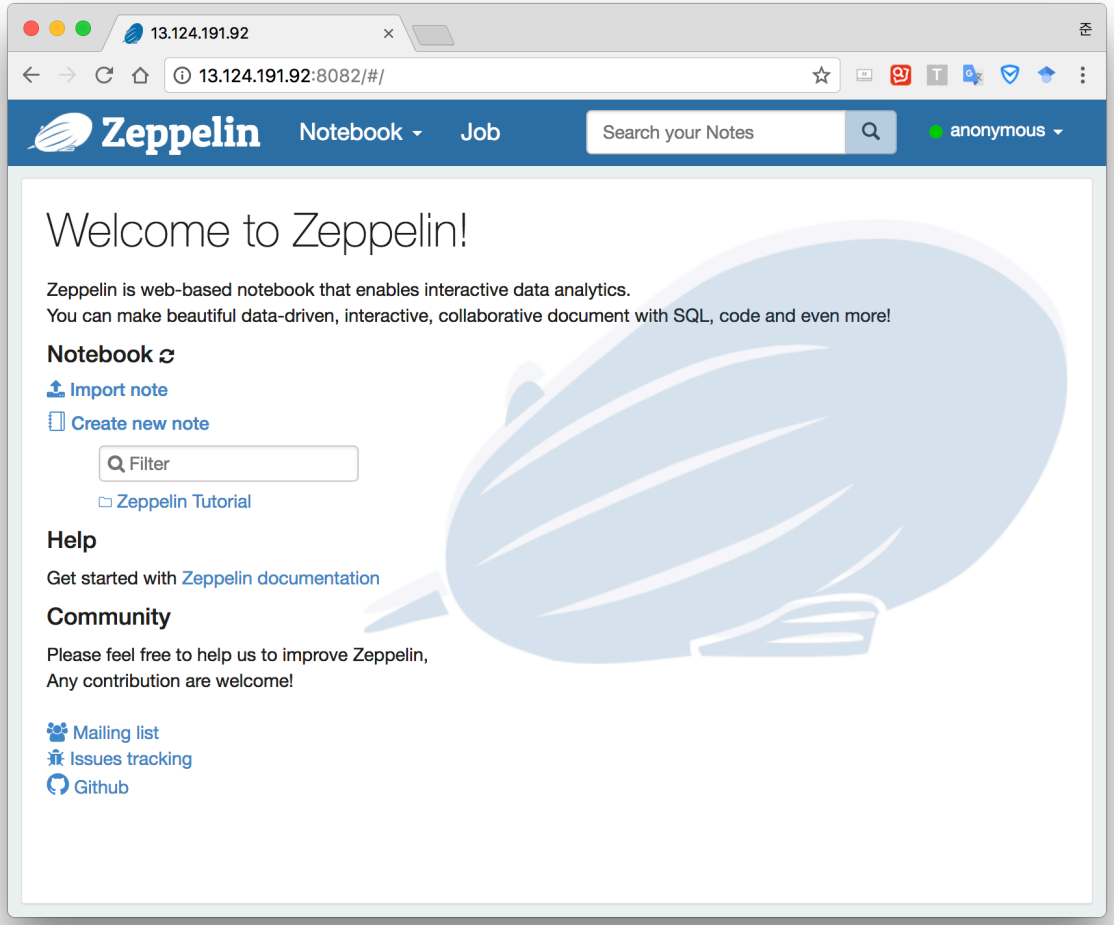
Zeppelin: Interpreter Setting (3)

- **Set parameters related to spark**



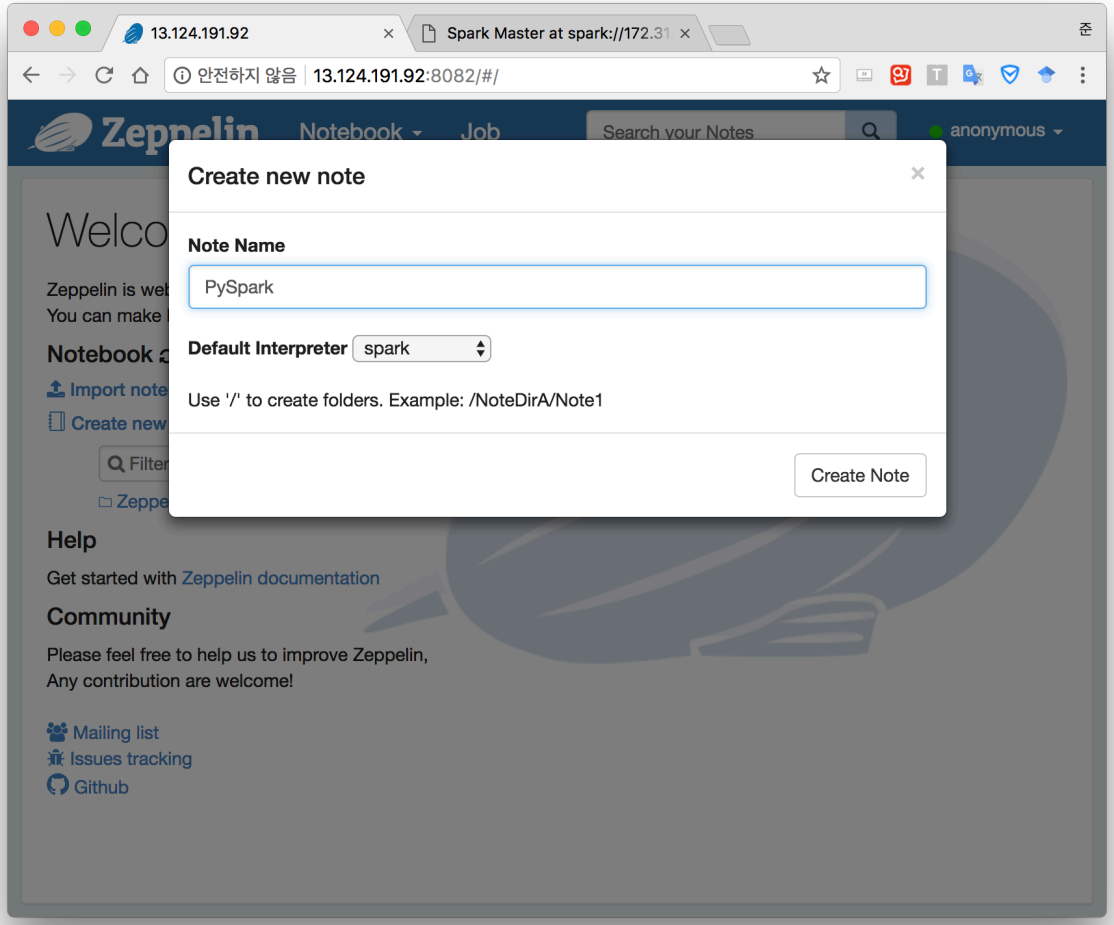
Zeppelin: Creating New Note (1)

- **Create new note**



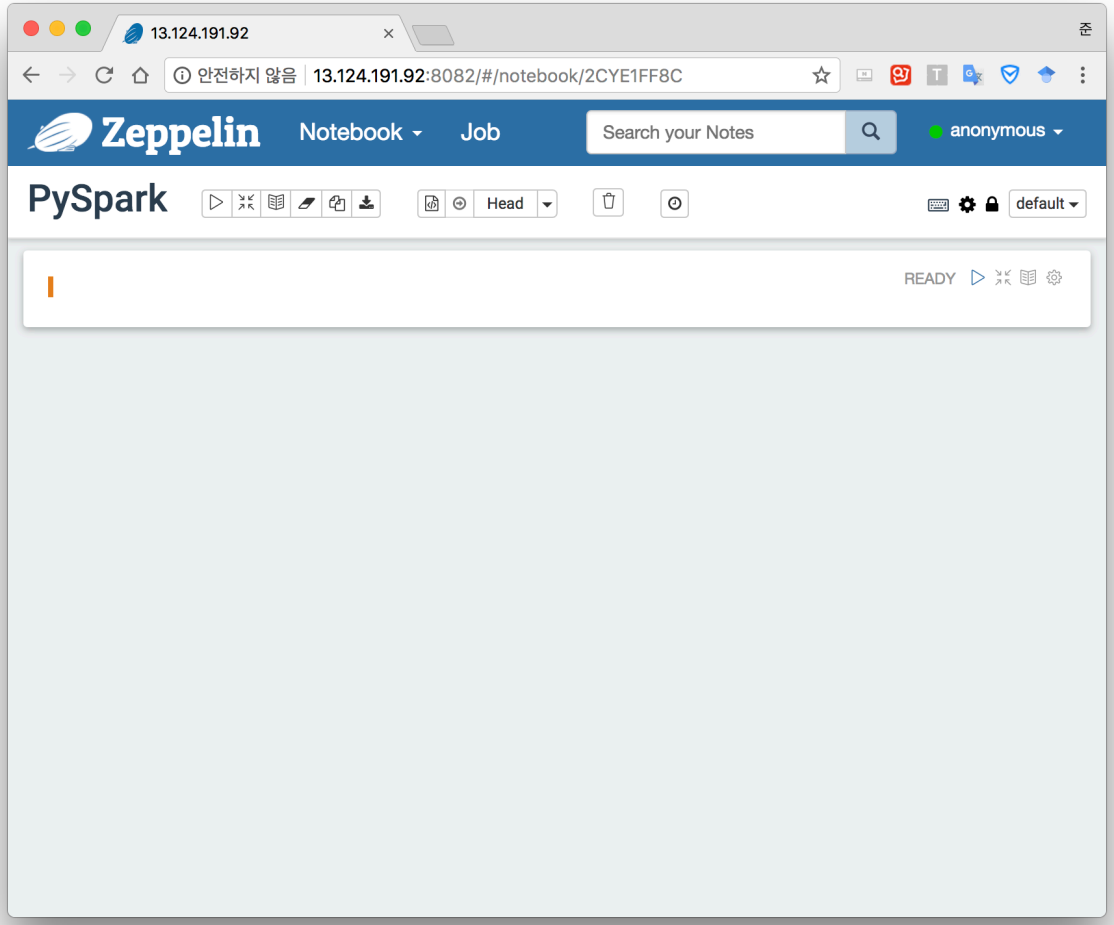
Zeppelin: Creating New Note (2)

- **Set note name**



Zeppelin: Creating New Note (3)

- **Zeppelin note window**



Zeppelin: Visualization Example (1)

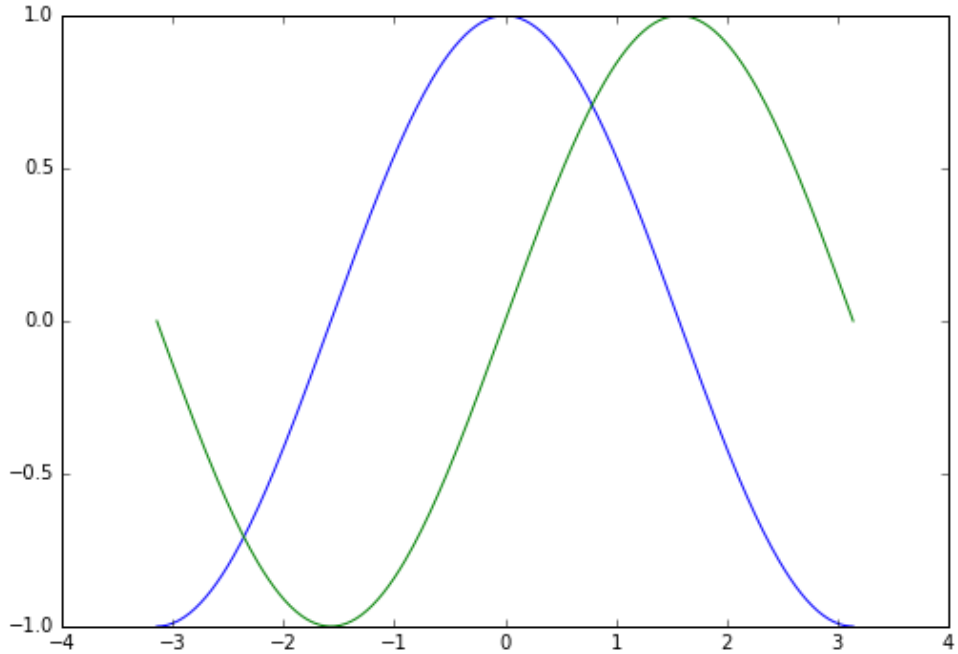
- **Import matplotlib**

```
%pyspark
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import rand
```

Zeppelin: Visualization Example (2)

- **Simple Graph**

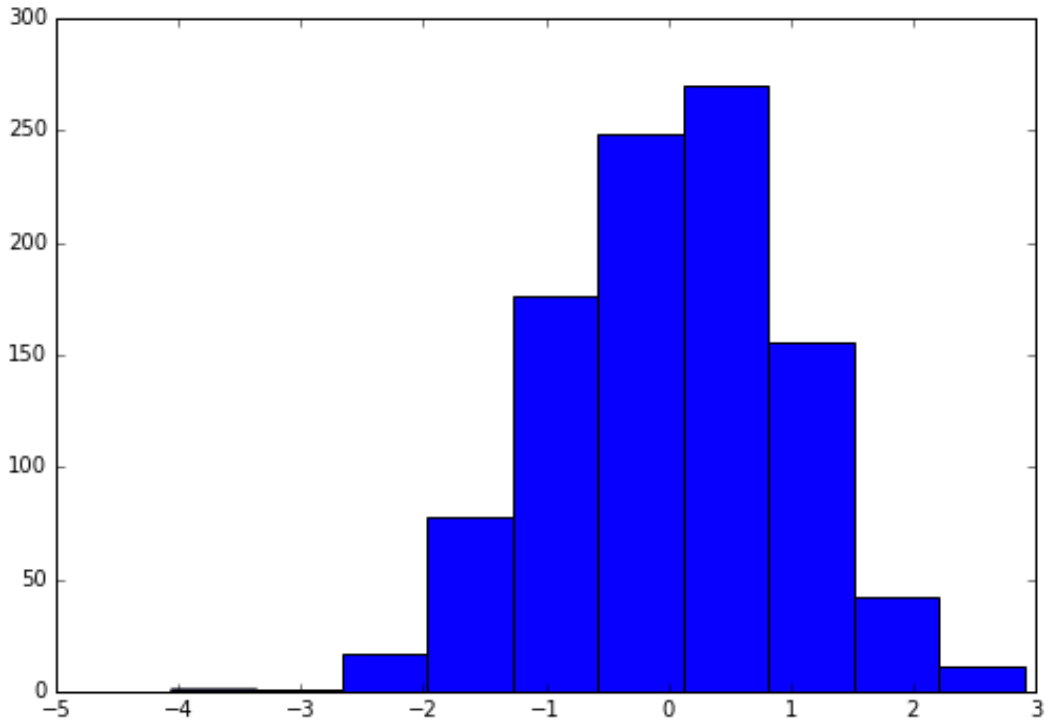
```
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.plot(X,C)
plt.plot(X,S)
```



Zeppelin: Visualization Example (3)

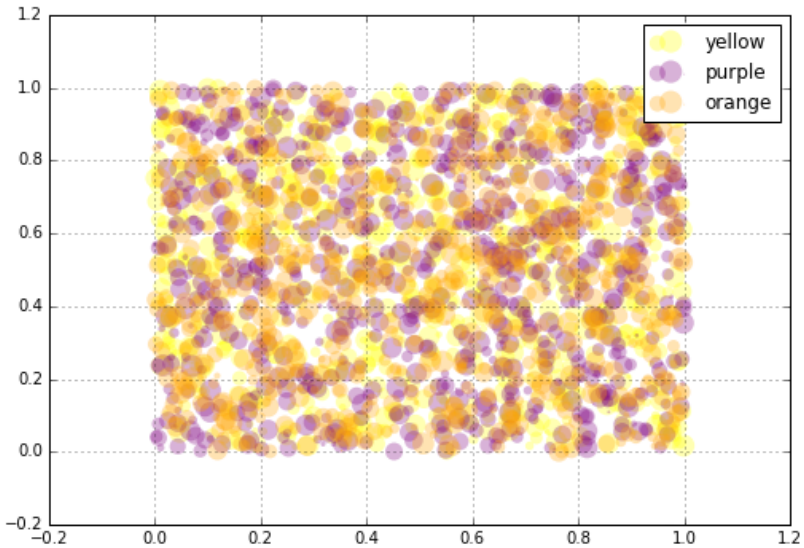
- **Histogram**

```
gaussian_numbers = np.random.randn(1000)
plt.hist(gaussian_numbers)
```



Zeppelin: Visualization Example (4)

- **Scatter**




```
fig, ax = plt.subplots()
for color in ['yellow', 'purple', 'orange']:
    n = 750
    x, y = rand(2, n)
    scale = 200.0 * rand(n)
    ax.scatter(x, y, c=color, s=scale, label=color, alpha=0.3, edgecolors='none')
ax.legend()
ax.grid(True)
plt.show()
```

http://matplotlib.org/examples/lines_bars_and_markers/scatter_with_legend.html

Exercise (1)

- **SQL with Zeppelin**
  - Make dataFrame
    - data-07.txt (need to save this file in zeppelin_home and spark_home)
    - https://drive.google.com/drive/folders/0B91DOcPTZ5DzWWplT1N3OEdDWVE
    - schema = ["custID","gender","state","cardholder","balance","numTrans"]
  - Questions
    - 1) Count frequencies of gender column (1=male, 2=female)
    - 2) Computes statistics for numeric and string columns
      - Hint: http://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html
      - Column: cardholder, balance, numTrans
      - Result: count, mean, stddev, min, max

Exercise (1)

- **SQL with Zeppelin**
  - Questions
    - 3) Create correlations matrix
      - Column: cardholder, balance, numTrans
      - 3 by 3 matrix
      - Hint: http://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html
    - 4) Draw histogram (balance)
      - API: select, rdd, flatMap
      - Hint: http://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html
      - Hint: http://spark.apache.org/docs/2.1.0/api/python/pyspark.html
      - Draw a figure by using other program (ex. Excel, …)
  - Submit screenshot of result

Exercise (2)

- **Analyze workloads in HiBench**
  - Workloads
    - WordCount, PageRank, TeraSort
    - Data size = large
  - Question
    - What are the bottlenecks in each workload?
    - cpu-bound / memory-bound
    - Submit screenshot and the reason
    - Attach a reason for thinking like that (one sentence)

- **Notification**
  - E-mail: heojun18@gmail.com
  - Deadline: 11/26, PM 23:59:59

Exercise (3)

- **Before starting the assignment, you should check the list below**
  - Check all configurations (hadoop, spark, hibench)
    - hdfs://x.x.x.x:9000 & spark://x.x.x.x:7077
      - x.x.x.x should always be the same between configurations (localhost / IP)
      - IP is 172.x.x.x
  - If there are some errors, you should check **spacing words** or **spelling mistake**
  - Please attach a screenshot to your email to get a quick response
    - heojun18@gmail.com

Appendix