# Airline Demo Setup Instructions

## Cluster setup

We will setup an HDP 2.2 cluster with 5 machines, 1 of which is an edge node (does not act as a datanode or worker node) dedicated to host IPython and RCloud. But it has all the Hadoop client bits so it can connect to the cluster

1. Setup the HDP cluster, use Ambari 1.7 to install HDP bits and configure
2. Setup Hue (helps with viewing files in the file browser)

Let’s assume our cluster boxes are named data1…data4 and master is the machine dedicated to IPython/RCloud (32-64GB, 8-16 cores)

1. On master, create user “demo”
2. Using Hue create an HDFS user “demo”
3. Setup JAVA\_HOME

## Python and IPython notebook

On HDP5 we will install IPython and a bunch of data-science packages that we will need. We install all of this under user “demo” (and not system-wide) since IPython requires Python 2.7.3 or later, and typical Centos machines have only 2.6.6, and upgrading in root causes issues with YUM. So the following sequence is rather complex, but necessary.

1. As root, prepare the system (run the below commands line by line or in a script, don’t copy paste all lines in one go):

yum update

yum install –y gcc, gcc-c++ gcc-gfortran

yum install -y zlib-devel bzip2-devel openssl-devel xz-libs wget

yum install -y sqlite-devel lrzsz freetype-devel libpng-devel

yum install -y geos geos-devel cairo-devel libXt-devel

yum install –y boost-devel libcurl libcurl-devel

yum groupinstall -y development tools

yum install –y lapack lapack-devel blas blas-devel readline-devel

yum install java-1.7.0-openjdk-devel

2. We need to compile Python with SQLite3, so as user “demo”:

cd /home/demo

wget <http://www.sqlite.org/2014/sqlite-autoconf-3080600.tar.gz>

tar xvfz sqlite-autoconf-3080600.tar.gz

cd sqlite-autoconf-3080600

./configure --prefix=/home/demo/.sqlite3

make && make install

3. Install Python 2.7.8 as user demo

cd /home/demo

wget https://www.[python.org/ftp/python/2.7.8/Python-2.7.8.tar.xz](http://python.org/ftp/python/2.7.8/Python-2.7.8.tar.xz)

xz -d Python-2.7.8.tar.xz && tar -xvf Python-2.7.8.tar

cd Python-2.7.8

./configure --prefix=/home/demo/.python LDFLAGS='-L/home/demo/sqlite-autoconf-3080600/.libs' CPPFLAGS='-I/home/demo/sqlite-autoconf-3080600/'

make && make altinstall

4. Install Python’s package management: easy\_install and pip

cd /home/demo

wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez\_setup.py

.python/bin/python2.7 [ez\_setup.py](http://ez_setup.py/)

.python/bin/easy\_install-2.7 pip

5. Install and activate VirtualEnv

.python/bin/pip2.7 install virtualenv

.python/bin/virtualenv-2.7 pyenv

chmod +x ./pyenv/bin/activate

source ./pyenv/bin/activate

Consider adding “source /home/demo/pyenv/bin/activate” to /home/demo/.bashrc so the python environment is setup automatically on login.

6. Install some data-science related Python packages (can take several minutes)

pip install numpy scipy pandas scikit-learn rpy2

7. Install matplotlib (for graphics in Python)

* cd /home/demo
* wget <http://sourceforge.net/projects/freetype/files/freetype2/2.5.0/freetype-2.5.0.tar.gz>
* tar xvfz freetype-2.5.0.tar.gz

As “root” execute these commands:

* cd /home/demo/freetype-2.5.0
* ./configure --prefix=/usr --disable-static && make
* make install && install -v -m755 -d /usr/share/doc/freetype-2.5.0 && cp -v -R docs/\* /usr/share/doc/freetype-2.5.0

As “demo” execute this command:

* cd /home/demo
* source ./pyenv/bin/activate
* pip install matplotlib

8. Install ipython

easy\_install ipython

9. Install PYDOOP – package to enable Hadoop access from Python.

We will use the new version of Pydoop – 1.0rc1

git clone <https://github.com/crs4/pydoop.git>

cd pydoop

To support HDP 2.2, in the file pydoop/hadoop\_utils.py, replace the function \_hadoop2\_jars with:

def \_hadoop2\_jars(hadoop\_home):

jars = \_jars\_from\_dirs([os.path.join(hadoop\_home, d) for d in (

'',

'client',

)])

jar\_root\_cldr = os.path.join(hadoop\_home, 'share', 'hadoop')

cldr\_jars = \_jars\_from\_dirs([os.path.join(jar\_root\_cldr, d) for d in (

'hdfs',

'common',

os.path.join('common', 'lib'),

'mapreduce',

'yarn', # hadoop >= 2.2.0

)])

jars.extend(cldr\_jars)

return jars

Ensure JAVA\_HOME and HADOOP\_HOME are set (consider adding to /home/demo/.bashrc)

export HADOOP\_HOME=/usr/hdp/current/hadoop-client

export JAVA\_HOME=<java home dir>

Now build PyDoop:

python setup.py build

python [setup.py](http://setup.py/) install --skip-build

10. Setup default IPython notebook profile:

* pip install tornado pyzmq ipython pygments matplotlib jinja2
* ipython profile create default

Edit the file ~/.ipython/profile\_default/ipython\_notebook\_config.py, and add the following:

c = get\_config()

c.IPKernelApp.pylab = 'inline'

c.NotebookApp.ip = '\*'

c.NotebookApp.open\_browser = False

c.NotebookApp.port = 9999

11. Install Spark 1.2 tech preview - follow instructions here: <http://hortonworks.com/hadoop-tutorial/using-apache-spark-hdp/>

Due to the integration with ISpark, you would need to download the source as follows:

Git clone <https://github.com/hortonworks/spark>.git

Git clone <https://github.com/hortonworks/spark-native-yarn.git> (2.2-maint branch)

Make sure you are on 2.2-maint branch in both sub-folders.

Git branch 2.2-maint

Then apply patch <https://issues.apache.org/jira/browse/SPARK-4923> inside the spark folder, and compile spark from scratch, using this command:

patch -p0 -i SPARK-4923\_\_Maven\_build\_should\_keep\_publishing\_spark-repl.patch

HADOOP\_VERSION=2.6.0 TEZ\_VERSION=0.5.2 SPARK\_VERSION=1.2.1-SNAPSHOT sh make-distribution.sh --HDP --tgz -Phive -Phive-thriftserver -Pyarn -Pyarn-history -Phadoop-2.4 -Dhadoop.version=2.6.0  -Dhbase.profile=hadoop2 –DskipTests

12. Setup Scala / ISpark notebook profile:

Following <https://github.com/tribbloid/ISpark>, download ISpark from the link, unzip under the folder name “ISpark”, and do the following:

Due to the Spark patch, you need to change ISpark/core/pom.xml to use the SNAPSHOT jar instead of the regular one for spark-repl and spark-sql.

Then compile:

mvn package

source ./pyenv/bin/activate

ipython profile create spark

Edit the file ~/.ipython/profile\_spark/ipython\_config.py, and add the following:

import os

SPARK\_HOME = os.environ['SPARK\_HOME']

MASTER = 'yarn-client'

c.KernelManager.kernel\_cmd = [SPARK\_HOME+"/bin/spark-submit",

"--master", MASTER,

"--class", "org.tribbloid.ispark.Main",

"--executor-memory", "2G",

"/home/demo/ISpark/core/target/ispark-core-assembly-0.1.0-SNAPSHOT.jar",

"--profile", "{connection\_file}",

"--interp", "Spark", "--parent"]

c.NotebookApp.ip = '\*'

c.NotebookApp.open\_browser = False

13. Enable PySpark from IPYTHON

Add the following to a new file at ~/.ipython/profile\_spark/startup/00-pyspark-setup.py

import os

import sys

spark\_home = os.environ.get(‘SPARK\_HOME’, None)

sys.path.insert(0, os.path.join(spark\_home, ‘python’))

sys.path.insert(0, os.path.join(spark\_home, ‘python/lib/py4j-0.8.1-src.zip’))

execfile(os.path.join(spark\_home, ‘python/pyspark/shell.py’))

Make sure you have the following in .bashrc:

export YARN\_CONF\_DIR=/etc/hadoop/conf

export HADOOP\_HOME=/usr/lib/hadoop

export SPARK\_HOME=/path/to/spark (e.g. /home/demo/spark-1.1.0)

14. Upload sample notebooks provided with this lab to /home/demo and ensure that util.py exists in the folder that contains the notebooks

15. To run the IPython server, use “ipython notebook”. Make sure it stays up even after disconnecting from SSH, e.g., using Screen.

To run the ISpark version, use “ipython notebook --profile spark”  
*Note: you may get a permission denied error for /root. To workaround it, do a recursive chmod on /root so the demo user can read it*

## R, RStudio server and RCloud

On the same HDP5 node we now install R, some R packages, and RStudio Server.

1. Set environment variables for R:

Add following to /etc/profile.d/r.sh:

. /usr/lib/hadoop/libexec/hadoop-config.sh --config /etc/hadoop/conf

export HADOOP\_CMD=/usr/bin/hadoop

export HADOOP\_STREAMING=/usr/lib/hadoop/contrib/streaming/hadoop-streaming.jar

2. On each node perform the following (is there a way to automate this?):

Install R and packages:

yum install R

Rscript -e 'install.packages(c("Rcpp", "RJSONIO", "bitops", "digest", "functional", "reshape2", "stringr", "plyr", "caTools", "rJava", "Hmisc", "plyr", "dplyr", "devtools", "Rook", "R.methodsS3"), repos="[http://cran.us.r-project.org](http://cran.us.r-project.org/)");'

Rscript -e 'install.packages(c("nnet", "randomForest", "rpart", "C50", "gbm", "e1071", "glmnet", "bnlearn", "cluster", "bigrf", "biclust"), repos="[http://cran.us.r-project.org](http://cran.us.r-project.org/)");'

Install RMR and RHDFS:

download RMR v3.2 and RHDFS v1.0.8 from RHadoop website: <https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads>

R CMD INSTALL rmr2\_3.2.0.tar.gz

export HADOOP\_CMD=/usr/bin/hadoop

R CMD INSTALL rhdfs\_1.0.8.tar.gz

Enable RPY2:

R CMD Java reconfigure -e

3. Install sparkR (Optional)

<https://github.com/amplab-extras/SparkR-pkg/blob/master/README.md>

Install Scala 2.10.4

Then from R console:

library(devtools)

install\_github("amplab-extras/SparkR-pkg", subdir="pkg")

## RStudio Server installation

Follow instructions here:

http://www.rstudio.com/products/rstudio/download-server/

Then run “rstudio-server start”

## RCloud installation

Rcloud is a relatively new open-source project out of AT&T labs. It enables an IPython-like environment for R, and also includes a nice GIT backend for artifact storage and versioning.

You can read more details here: <http://stats.research.att.com/RCloud/>

To install on HDP5 follow these instructions (following what’s provided here: https://github.com/att/rcloud/blob/develop/doc/INSTALL.md):

1. Download RCloud release 1.0 (not the dev branch) from Github: <https://github.com/att/rcloud>
2. Unzip into a folder named RCloud
3. In R type:
   1. install.packages("Cairo")
   2. install.packages("rjson")
   3. install.packages("github",,"[http://rforge.net](http://rforge.net/)")
   4. install.packages("rcloud.support", repos=c("http://RForge.net", "http://R.research.att.com"), type="source")
4. Using your Github account, create a GITHub application named “RCLOUD” that you will use for authentication. Make sure you specify the “application URL” as HDP5 FQN (let’s assume it’s hdp5.cloud.hortonworkjs.com), and “callback URL” is (under the same assumption) with the addition of login\_successful.R, in other words: “hdp5.cloud.hortonworks.com/login\_successful.R”
5. Create a file under “rcloud/conf” called “rcloud.conf” and put in it the following lines:

github.client.id: your.20.character.client.id

github.client.secret: your.40.character.client.secret

github.base.url: <https://github.com/>

github.api.url: <https://api.github.com/>

github.gist.url: <https://gist.github.com/>

Cookie.Domain: hdp5.cloud.hortonworks.com

1. Run “./scripts/fresh\_start.sh”

## Data Ingest

1. Under user “demo” in HDFS create folder named “airline”.  
Ensure that the demo user has permissions to this folder in HDFS

2. Download airline delay dataset files for 2007 and 2008 from <http://stat-computing.org/dataexpo/2009/the-data.html>, and ingest them under /user/demo/airline/delay - (2007.csv and 2008.csv)

3. Download weather data for 2007 and 2008 from <http://www.ncdc.noaa.gov/cdo-web/datasets/>, click on “daily summaries”, then “FTP”, and “by year”.

4. download 2007.csv.gz and 2008.csv.gz, then ingest the unzipped csv files under /user/demo/airline/weather - (2007.csv and 2008.csv)