Chapter 28. April 2019

Welcome to the April 2019 edition of DataStax Developer's Notebook (DDN). This month we answer the following question(s);

I work on my laptop using Python, R, and Jupyter Notebooks, doing analytics for my company. I've been digging on the analytics topics built inside DataStrax Enterprise (pre-installed, data co-located with the analytics routines), but don't see how I can make use of any of this. Can you help ?

Excellent question ! Isn't Python and R the number one data science software pairing on the planet ? (Regardless, it must be in the top two.) DataStax Enterprise ships pre-configured with a Python interpreter, and R. For R, don't know why, you must install one new external library.

We do have a bias towards Spark/R, since Spark/R seemed to lead in the area of open source parallel capable routines. (Speed, performance, better documentation.)

Jupyter is also an excellent choice, especially if you're Python focused. We've not yet looked at the Anaconda distribution of Jupyter, which seems very promising. We'll show a Jupyter install, but may ourselves stick with Apache Zeppelin, since Zeppelin seems to come pre-installed/pre-configured for so many more languages and options out of the box. (Less work for us.)

Software versions

The primary DataStax software component used in this edition of DDN is DataStax Enterprise (DSE), currently release 6.7. All of the steps outlined below can be run on one laptop with 16 GB of RAM, or if you prefer, run these steps on Amazon Web Services (AWS), Microsoft Azure, or similar, to allow yourself a bit more resource.

For isolation and (simplicity), we develop and test all systems inside virtual machines using a hypervisor (Oracle Virtual Box, VMWare Fusion version 8.5, or similar). The guest operating system we use is Ubuntu Desktop version 18.04, 64 bit.

28.1 Terms and core concepts

As stated above, ultimately the end goal is to run R analytics inside DataStax Enterprise (DSE). The advantages of running R inside DSE, versus on the laptop, are many:

- While your laptop many have 8-16 cores, and 32-64 GB of RAM, the DSE server will likely have much more resource. DSE could be running 100+ CPUs, and a TB or more of RAM.

  DSE will also have many disk drives and controllers (parallel disk I/O), and not the smaller/single drive and I/O bus on a workstation or laptop.
- Also, workload isolation; you can send an analytics job to the DSE server, and while processing, your laptop will still be responsive.
- Lastly, the data is already loaded inside DSE; you wont have to lift the data off of the server, network it to the laptop, yadda.

The net of the above results in massive performance gains. Our favorite definition of the term, machine learning, is; a routine whose accuracy improves given more data. Using DSE, you can process way more data, and get more accurate results much faster.

DSE command line

DataStax Enterprise (DSE) ships with many command line interpreters built in, pre-configured, ready to use.

Running the "dse --" command will produce a usage message. Use this step if you forget what the specific (other) options to the "dse" command are.

To run the DSE Python/Spark command line interpreter, run,

```
dse pyspark

         recs_x = sc.parallelize( (
            (111, "Bob"  ),
            (222, "Ted"  ),
            (333, "Alice")
            ) )
         recs_x.count()
         exit()
```

The above is a full Python interpretive environment, and the answer received before exit is, 3.

To run the DSE Spark/R command line interpreter, run,

```
dse sparkR --total-executor-cores 1
```

If you've done nothing to your DSE to configure R, you will receive an error above similar to,

```
The log file is at /root/.sparkR.log
env: 'R': No such file or directory
```

And the "dse sparkR" command will terminate. Per this DataStax version 6.7 documentation Url,

```
https://docs.datastax.com/en/dse/6.7/dse-admin/datastax_enterprise/s
park/sparkROverview.html
```

The procedure to install the missing R libraries is,

```
echo "deb http://cran.rstudio.com/bin/linux/ubuntu trusty/" >>
/etc/apt/sources.list
gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9 && \
   gpg -a --export E084DAB9 | sudo apt-key add -
apt-get update
apt-get install r-base


#  didn't work,
      ...
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:


The following packages have unmet dependencies:
   r-base : Depends: r-recommended (= 3.4.4-1ubuntu1) but it is not
going to be installed
E: Unable to correct problems, you have held broken packages.
```

For our tests, we were on Ubuntu version (Ubuntu 18.04.1 LTS), and an alpha release of DSE version 6.8, and the above (install) sequence failed. We were able to install via this sequence of commands,

```
From,

https://askubuntu.com/questions/218708/installing-latest-version-of-
r-base

          apt-get -y update

          apt-get -y install r-base

          apt-get -y install r-base-dev
```

The above did leave us with a JVM version mismatch (the DSE and R JVMs were off by 0.0.2), which produced warnings, but no errors.

To install-verify, we ran a,

```
dse sparkR --total-executor-cores 1

   help.start()     // Starts a Web browser

   demo()

   demo("scoping")


   q()
```

Demoing R on top of DSE

To demonstrate, and prove that out DSE Spark/R environment is running, we ran the following as shown in Example 28-1. A code review follows.

Example 28-1   Multiple languages below ! Example to run Spark/R.

```
DROP KEYSPACE IF EXISTS ks_28;
      //
   CREATE KEYSPACE ks_28 WITH REPLICATION =
      {'class': 'SimpleStrategy', 'replication_factor': 1};
         //
   USE ks_28;

   CREATE TABLE t1
      (
      col1     INT PRIMARY KEY,
      col2     INT,
      col3     INT,
      col4     INT
      );
   INSERT INTO t1 (col1, col2, col3, col4)
      VALUES (1, 1, 1, 1);
   INSERT INTO t1 (col1, col2, col3, col4)
      VALUES (2, 2, 2, 2);
   INSERT INTO t1 (col1, col2, col3, col4)
      VALUES (3, 3, 3, 3);
   INSERT INTO t1 (col1, col2, col3, col4)
      VALUES (4, 4, 4, 4);

         //

   dse fs "cp 'file:///mnt/hgfs/My.20/MyShare_1/18 DSE Developers Notebook/28
Jupyter/04_sample.csv' ."

         //

   df_01 <-sql("SELECT * FROM ks_28.t1 WHERE col1 > 2")
   head(df_01)

   df_02 <- read.df("dsefs:///04_sample.csv",
      header = "false",
      delimiter = "|",
      source = "csv")
   head(df_02)
```

Relative to Example 28-1, the following is offered:

- The first block is run inside CQLSH or similar, and makes a keyspace and table for our testing/use. We insert 4 rows of data.
- The second block, denoted by the single line "dse fs" command, copies a ASCII text file into the shared HDFS compatible filesystem named, "dsefs".

  The file has (n) rows, and 4 columns, delimited by ASCII 127, the vertical bar, or pipe symbol.
- And lastly, R commands.

  We demonstrate the ability to read from a DSE table.

  And we demonstrate reading from a DSEFS CSV style file.

Jupyter, (or at least a mention of it)

To install Jupyter, you have two primary paths-

```
   which python

         /usr/bin/python
```

```
python --version
   Python 2.7.15rc1

https://jupyter.org/install
   |
   +--->  https://www.anaconda.com/distribution/#download-section
      "64 bit command line installer
   https://docs.anaconda.com/anaconda/install/linux/
.  Comes as a *.sh  ,  run that
   Stopped here-
```

The Anaconda (path) install is much more robust, includes more out-of-the-box capabilities, yadda. there is a free ware version of Anaconda, but honestly we stopped here is favor if true open source.

A simpler Jupyter install using Python/PIP (PIP: Python installer program).

```
From,
      https://jupyter.org/install


      python -m pip install --upgrade pip
      python -m pip install jupyter


      dse exec jupyter notebook --allow-root
```

This simpler Jupyter install leaves you with the ability to:

– Run Python
– Run Shell
– And markdown

Very easy (but also tedious ?), you then proceed to install 20 or more packages manually that include function similar to:

– Format SQL style results in an (HTML table)
– Read from given data source, including Cassandra or DSE
– Other

Note: Apache Zeppelin has a distribution where these twenty or so extra packages (interpreters) come pre-installed, pre-configured, including results formatting, Cassandra, yadda.

Apache Zeppelin

As stated above, apache Zeppelin has a distribution that comes pre-installed, pre-configured, with much more out of the box, including Python.

The first time you try to run Spark/R, you will receive an error.

```
%spark.r


   df_01 <-sql("SELECT * FROM ks_28.t1 WHERE col1 > 2")
   head(df_01)


   //  java.lang.RuntimeException: Error in library("knitr"):
   //     there is no package called ‘knitr’ at
   //
org.apache.zeppelin.spark.ZeppelinR.request(ZeppelinR.java:259)
   //     at
org.apache.zeppelin.spark.ZeppelinR.eval(ZeppelinR.java:185)
   //     at
org.apache.zeppelin.spark.SparkRInterpreter.open(SparkRInterpreter.j
ava:112
   //        ...
```

To repair the above, enter the DSE Spark/R command line interpreter,

```
dse sparkR --total-executor-cores 1


         install.packages("knitr")
```

And then inside Apache Zeppelin,

```
%spark.r
```

```
df_01 <-sql("SELECT * FROM ks_28.t1 WHERE col1 > 2")
head(df_01)

//    col1 col2 col3 col4
// 1  3    3    3    3
// 2  4    4    4    4
```

28.2 Complete the following

At this point in this document we detailed how to configure R for use inside DSE, and documented a round-trip; reading from both DSE tables and the Hadoop HDFS compatible parallel filesystem inside DSE.

Go forth, and use R inside DSE.

28.3 In this document, we reviewed or created:

This month and in this document we detailed the following:

– Configuring R for use inside DSE

– Using R, how to read from a DataStax table, and how to read from DSEFS.

Persons who help this month.

Kiyu Gabriel, Alex Ott, and Jim Hatcher.

Additional resources:

Free DataStax Enterprise training courses,

```
https://academy.datastax.com/courses/
```

Take any class, any time, for free. If you complete every class on DataStax Academy, you will actually have achieved a pretty good mastery of DataStax Enterprise, Apache Spark, Apache Solr, Apache TinkerPop, and even some programming.

This document is located here,

```
https://github.com/farrell0/DataStax-Developers-Notebook
https://tinyurl.com/ddn3000
```