Developer Training for Spark and Hadoop I: Hands-On Exercises

General Notes

Cloudera's training courses use a Virtual Machine running the CentOS Linux distribution. This VM has CDH (Cloudera's Distribution, including Apache Hadoop) installed in Pseudo-Distributed mode. Pseudo-Distributed mode is a method of running Hadoop whereby all Hadoop daemons run on the same machine. It is, essentially, a cluster consisting of a single machine. It works just like a larger Hadoop cluster, the only key difference (apart from speed, of course!) being that the block replication factor is set to 1, since there is only a single DataNode available.

Getting Started

1. Before starting the exercises, run the course setup script in a terminal window:

```
$ $DEV1/scripts/training_setup_dev1.sh
```

   This script will enable services and set up any data required for the course. You must run this script before starting the Hands-On Exercises.

Working with the Virtual Machine

1. The VM is set to automatically log in as the user `training`. Should you log out at any time, you can log back in as the user `training` with the password `training`.

2. Should you need it, the root password is `training`. You may be prompted for this if, for example, you want to change the keyboard layout. In general, you should not need this password since the `training` user has unlimited sudo privileges.

3. In some command-line steps in the exercises, you will see lines like this:

```
$ hdfs dfs -put shakespeare \
/user/training/shakespeare
```

The dollar sign (**$**) at the beginning of each line indicates the Linux shell prompt. The actual prompt will include additional information (e.g., `[training@localhost workspace]$`) but this is omitted from these instructions for brevity.

The backslash (`\`) at the end of the first line signifies that the command is not completed, and continues on the next line. You can enter the code exactly as shown (on two lines), or you can enter it on a single line. If you do the latter, you should *not* type in the backslash.

Points to note during the exercises

1. The main directory for the exercises for this course is `$DEV1/exercises` (`~/training_materials/dev1/exercises`). Each directory under that one corresponds to an exercise or set of exercises – this is referred to in the instructions as "the exercise directory". Any scripts or files required for the exercise (other than data) are in the exercise directory.

2. Within each exercise directory you may find the following subdirectories:

    a. `solution` – Solution code for each exercise.

    b. `stubs` – A few of the exercises depend on provided starter files containing skeleton code.

    c. Maven project directories – For exercises for which you must write Scala classes, you have been provided with preconfigured Maven project directories. Within these projects are two packages: `stubs`, where you will do your work using starter skeleton classes; and `solution`, containing the solution class.

3. Data files used in the exercises are in `$DEV1DATA` (`~/training_materials/data`). Usually you will upload the files to HDFS before working with them.

4. As the exercises progress, and you gain more familiarity with Hadoop and Spark, we provide fewer step-by-step instructions; as in the real world, we merely give you a requirement and it's up to you to solve the problem! You should feel free to refer to the solutions provided, ask your instructor for assistance, or consult with your fellow students!

5. There are additional bonus exercises for some of the exercises. If you finish the main exercise, please attempt the additional steps.

Catching Up

Most of the exercises in this course build on prior exercises. If you are unable to complete an exercise (even using the provided solution files), or if you need to catch up to the current exercise, run the provided catch up script:

```
$ $DEV1/scripts/catchup.sh
```

The script will prompt for which exercise you are starting; it will set up all the data required as if you had completed all the exercises.

Warning: If you run the catch up script, you will lose all your work! (e.g. All data will be deleted from HDFS, tables deleted from Hive, etc.)

Hands-On Exercise: Access HDFS with Command Line and Hue

Files and Data Used in This Exercise:

Data files (local):

`$DEV1DATA/kb/*`

`$DEV1DATA/base_stations.tsv`

In this exercise you will practice working with HDFS, the Hadoop Distributed File System.

You will use the HDFS command line tool and the Hue File Browser web-based interface to manipulate files in HDFS.

Set Up Your Environment

1. Before starting the exercises, be sure you have run the course setup script in a terminal window. (You only need to run this script once; if you ran it earlier, you do not need to run it again.)

```
$ $DEV1/scripts/training_setup_dev1.sh
```

Explore the HDFS Command Line Interface

HDFS is already installed, configured, and running on your virtual machine.

The simplest way to interact with HDFS is by using the `hdfs` command. To execute file system commands within HDFS, use the `hdfs dfs` command.

1. Open a terminal window (if one is not already open) by double-clicking the Terminal icon on the desktop.

2. Enter:

```
$ hdfs dfs -ls /
```

   This shows you the contents of the root directory in HDFS. There will be multiple entries, one of which is `/user`. Individual users have a "home" directory under this directory, named after their username; your username in this course is `training`, therefore your home directory is `/user/training`.

3. Try viewing the contents of the `/user` directory by running:

```
$ hdfs dfs -ls /user
```

   You will see your home directory in the directory listing.

4. List the contents of your home directory by running:

```
$ hdfs dfs -ls /user/training
```

   There are no files yet, so the command silently exits. This is different than if you ran `hdfs dfs -ls /foo`, which refers to a directory that doesn't exist and which would display an error message.

   Note that the directory structure in HDFS has nothing to do with the directory structure of the local filesystem; they are completely separate namespaces.

Upload Files to HDFS

Besides browsing the existing filesystem, another important thing you can do with the HDFS command line interface is to upload new data into HDFS.

5. Start by creating a new top level directory for exercises. You will use this directory throughout the rest of the course.

```
$ hdfs dfs -mkdir /loudacre
```

6. Change directories to the local filesystem directory containing the sample data we will be using in the course.

```
$ cd $DEV1DATA
```

If you perform a regular Linux `ls` command in this directory, you will see several files and directories used in this class. One of the data directories is `kb`. This directory holds Knowledge Base articles that are part of Loudacre's customer service website.

7. Insert this directory into HDFS:

```
$ hdfs dfs -put kb /loudacre/
```

This copies the local `kb` directory and its contents into a remote HDFS directory named `/loudacre/kb`.

8. List the contents of the new HDFS directory now:

```
$ hdfs dfs -ls /loudacre/kb
```

You should see the KB articles that were in the local directory.

Relative paths

In HDFS, any relative (non-absolute) paths are considered relative to your home directory. There is no concept of a "current" or "working" directory as there is in Linux and similar file systems.

9. Practice uploading a directory, then remove it, as it is not actually needed for the exercises.

```
$ hdfs dfs -put $DEV1DATA/calllogs /loudacre/
$ hdfs dfs -rm -r /loudacre/calllogs
```

View HDFS files

Now view some of the data you just copied into HDFS.

10. Enter:

```
$ hdfs dfs -cat /loudacre/kb/KBDOC-00289.html | tail \
-n 20
```

This prints the last 20 lines of the article to your terminal. This command is handy for viewing HDFS data. An individual file is often very large, making it inconvenient to view the entire file in the terminal. For this reason, it's often a good idea to pipe the output of the `fs -cat` command into `head`, `tail`, `more`, or `less`.

11. To download a file to work with on the local filesystem use the `hdfs dfs -get` command. This command takes two arguments: an HDFS path and a local path. It copies the HDFS contents into the local filesystem:

```
$ hdfs dfs -get \
/loudacre/kb/KBDOC-00289.html ~/article.html
$ less ~/article.html
```

12. There are several other operations available with the `hdfs dfs` command to perform most common filesystem manipulations: `mv`, `cp`, `mkdir`, etc.

    In the terminal window, enter:

```
$ hdfs dfs
```

    You see a help message describing all the file system commands provided by HDFS.

    Try playing around with a few of these commands if you like.

Use the Hue File Browser to browse, view and manage files

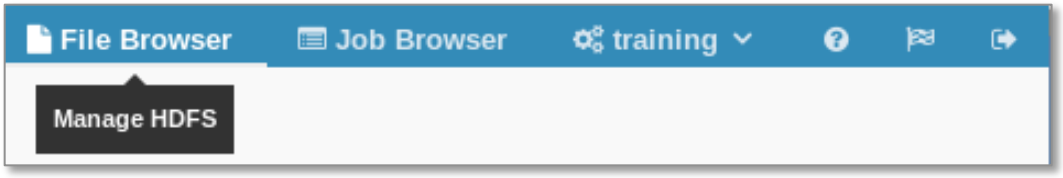
13. Start Firefox on the VM (using the shortcut provided on your desktop or task).

14. Click the Hue bookmark, or visit `http://localhost:8888`.

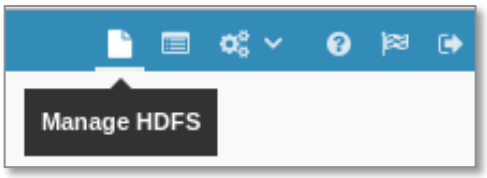
15. Because this is the first time anyone has logged into Hue on this server, you will be prompted to create a new user account. Enter username **`training`** and password **`training`**, then click **Create Account**. (If prompted you may click "Remember Password")

    - **Note**: When you first log in to Hue you may see a misconfiguration warning. This is because not all the services Hue depends on are running on the course VM. You can disregard the message.

16. Hue has many useful features, many of which will be covered later in the course. For now, to access HDFS, click **File Browser** in the Hue menu bar. (The mouse-over text is "Manage HDFS").



- **Note**: If your Firefox window is too small to display the full menu names, you will see just the icons instead.



17. By default, the contents of your HDFS home directory (`/user/training`) display. In the directory path name, click the leading slash (**/**) to view the HDFS root directory.

18. The contents of the root directory display, including the **loudacre** directory you created earlier. Click on that directory to see the contents.

19. Click on the name of the **kb** directory to see the knowledge base articles you uploaded.

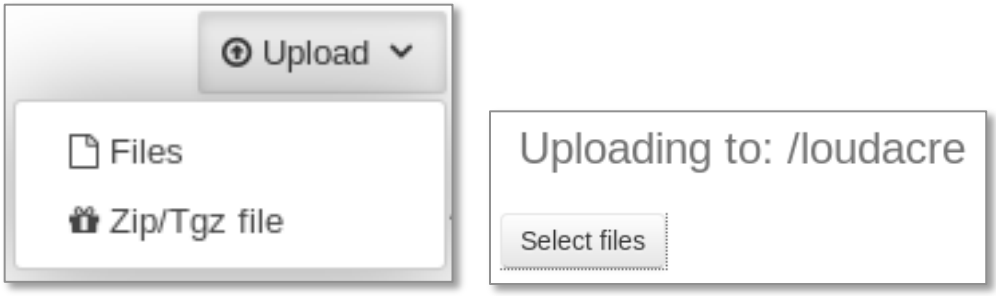
20. View one of the files by clicking on the name of any one of the articles.

21. In the file viewer, the contents of the file are displayed on the right. In this case, the file is fairly small, but typical files in HDFS are very large, so rather than displaying the entire contents on one screen, Hue provides buttons to move between pages.

22. Return to the directory review by clicking **View file location** in the Action panel on the left.

23. Click the up arrow ( ) to return to the `/loudacre` base directory.

24. To upload a file, click the **Upload** button. You can choose to upload a plain file, or to upload a zipped file (which will be automatically unzipped after upload). In this case, select **Files**, then click **Select Files**.



25. A Linux file browser appears. Browse to `/home/training/training_materials/data`.

26. Choose `base_stations.tsv` and click the Open button.

27. When the file has uploaded, it will be displayed in the directory. Click the checkbox next to the file's icon, then click the **Actions** button to see a list of actions that can be performed on the selected file(s).

28. *Optional*: explore the various file actions available. When you've finished, select any unneeded files you have uploaded and click the **Move to trash** button to delete.

This is the end of the Exercise

Hands-On Exercise: Run a YARN Job

Files and Data Used in this Exercise

Exercise directory: `$DEV1/exercises/yarn`

Dataset (HDFS): `/loudacre/kb`

In this exercise you will submit an application to the YARN cluster, and monitor the application using both the Hue Job Browser and the YARN Web UI.

The application you will run is provided for you. It is a simple Spark application written in Python that counts the occurrence of words in Loudacre's customer service Knowledge Base (which you uploaded in the last exercise). The focus of this exercise is not on what the application does, but on how YARN distributes tasks in a job across a cluster, and how to monitor an application and view its log files.

Explore the YARN cluster

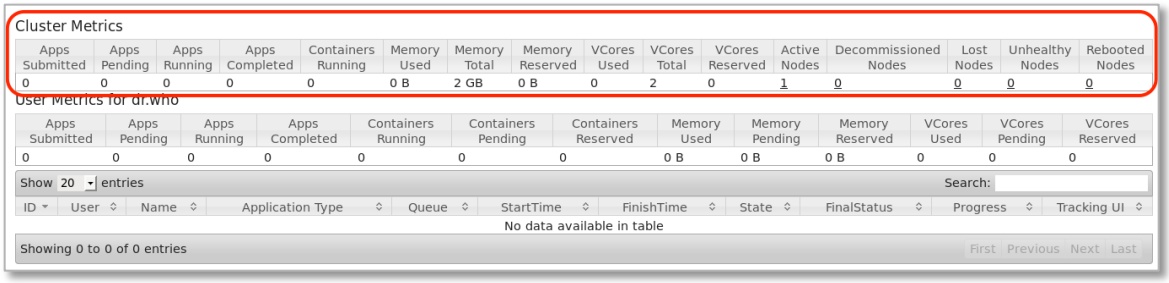
1. Visit the YARN Resource Manager (RM) UI in Firefox using the provided bookmark, or by going to URL `http://localhost:8088`.

   No jobs are currently running so the current view shows the cluster "at rest".

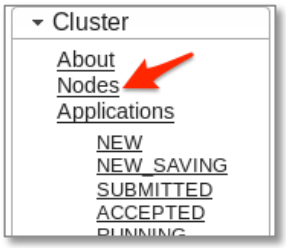
Who is Dr. Who?

You may notice that YARN says you are logged in as `dr.who`. This is what is displayed when user authentication is disabled for the cluster, as it is on the training VM. If user authentication were enabled, you would have to log in as a valid user to view the YARN UI, and your actual user name would be displayed, together with user metrics such as how many applications you had run, how much system resources your applications used and so on.

2. Take note of the values in the Cluster Metrics section, which displays number of applications running currently, previously run or waiting to run; the amount of memory used and available; how many worker nodes are in the cluster; etc.

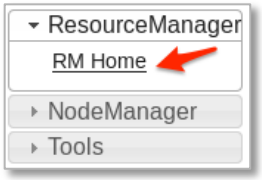


3. Click on the **Nodes** link in the Cluster menu on the left. The bottom section will display a list of worker nodes in the cluster. The pseudo-distributed cluster used for training has only a single node, which is running on the local machine. In the real world, this list would show multiple worker nodes.



4. Click on the **Node HTTP Address** to open the Node Manager UI on that specific node. This displays statistics about the selected node, including amount of available memory, currently running applications (none, currently) and so on.

5. To return to the Resource Manager, expand ResourceManager → RM Home on the left.



Submit an application to the YARN cluster

6. In a terminal window, change to the exercise directory:

```
$ cd $DEV1/exercises/yarn
```

7. Run the example wordcount.py program on the YARN cluster to count the frequency of words in the knowledge dataset:

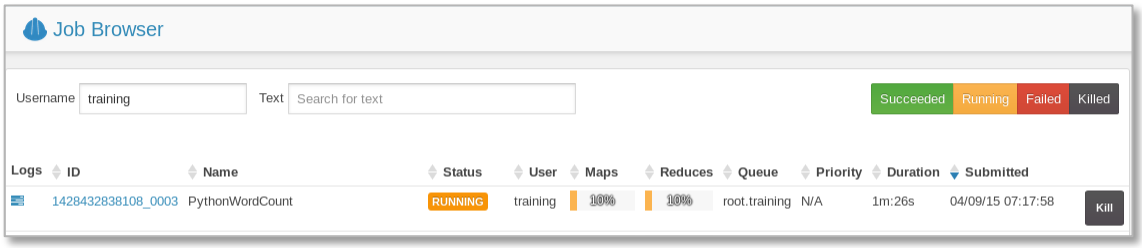
```
$ spark-submit --master yarn-cluster \
  wordcount.py /loudacre/kb/*
```

Later in the course you will learn how to write and submit your own Spark applications, as you are doing here, including what the various options mean. For now, focus on learning about the YARN UI.

View the application in the Hue Job Browser

8. Go to Hue in Firefox, and select the Job Browser. (Depending on the width of your browser, you may see the whole label, or just the icon.)

9. The Job Browser displays a list of currently running and recently completed applications. (If you don't see the application you just started, wait a few seconds, the page will automatically reload; it can take some time for the application to be accepted and start running.)

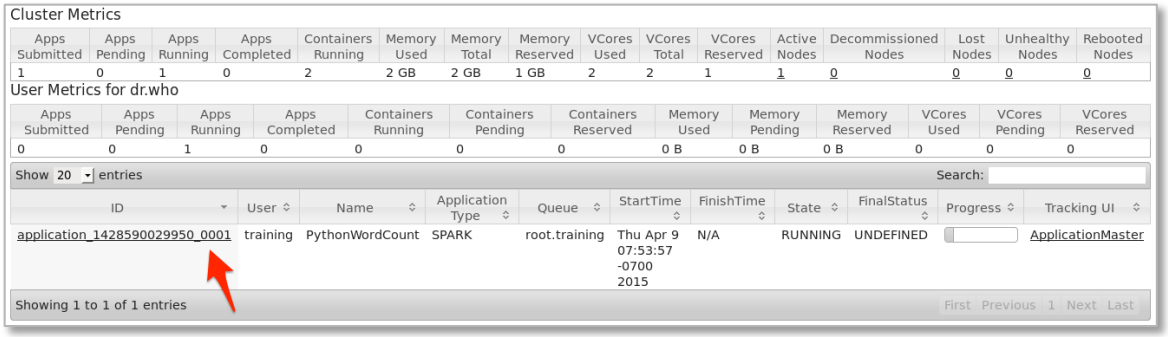


10. This page allows you to click the application ID to see details of the running application, or to kill a running job. (Don't do that now though!)

View the application in the YARN UI

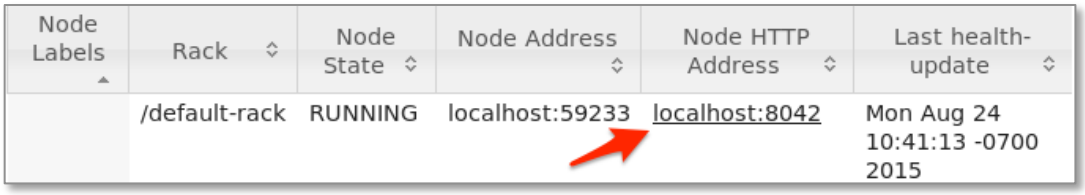
To get a more detailed view of the cluster, use the YARN UI.

11. Reload the YARN RM page in Firefox. You will now see the application you just started in the bottom section of the RM home page.

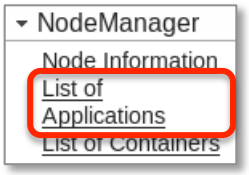


12. As you did in the first exercise section, select **Nodes**.

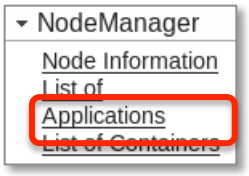
13. Select the **Node HTTP Address** to open the Node Manager UI.



14. Now that an application is running, you can click on **List of Applications** to see the application you submitted.



15. If your application is still running, try clicking on **List of Containers**.



This will display the containers the Resource Manager has allocated on the selected node for the current application. (No containers will show if no applications are running; if you missed it because the application completed, you can run the application again.)

This is the end of the Exercise

Hands-On Exercise: Import Data from MySQL Using Sqoop

Files and Data Used in this Exercise

Exercise directory: `$DEV1/exercises/sqoop`

MySQL Database: `loudacre`

MySQL Tables: `accounts`, `webpage`

In this exercise, you will import tables from MySQL into HDFS with Sqoop.

Import the accounts table from MySQL

You can use Sqoop to look at the table layout in MySQL. With Sqoop, you can also import the table from MySQL to HDFS.

1. Open a new terminal window if necessary.
2. Run the `sqoop help` command to familiarize yourself with the options in Sqoop:

```
$ sqoop help
```

3. List the tables in the `loudacre` database:

```
$ sqoop list-tables \
--connect jdbc:mysql://localhost/loudacre \
--username training --password training
```

4. Run the `sqoop import` command to see its options:

```
$ sqoop import --help
```

5. Use Sqoop to import the `accounts` table in the `loudacre` database and save it in HDFS under `/loudacre`:

```
$ sqoop import \
  --connect jdbc:mysql://localhost/loudacre \
  --username training --password training \
  --table accounts \
  --target-dir /loudacre/accounts \
  --null-non-string '\\N'
```

   The `--null-non-string` option tells Sqoop to represent `null` values as `\N`, which makes the imported data compatible with Hive and Impala.

6. *Optional*: While the Sqoop job is running, try viewing it in the Hue Job Browser or YARN Web UI, as you did in the previous exercise.

View the imported data

Sqoop imports the contents of the specified tables to HDFS. You can use the hdfs command line or the Hue File Browser to view the files and their contents.

7. List the contents of the `accounts` directory:

```
$ hdfs dfs -ls /loudacre/accounts
```

   - **Note**: Output of Hadoop processing jobs is saved as one or more numbered "partition" files. Partitions are covered later in the course.

8. Use either the Hue File Browser or the HDFS `tail` command to view the last part of the file for each of the MapReduce partition files, e.g.:

```
$ hdfs dfs -tail /loudacre/accounts/part-m-00000
$ hdfs dfs -tail /loudacre/accounts/part-m-00001
$ hdfs dfs -tail /loudacre/accounts/part-m-00002
$ hdfs dfs -tail /loudacre/accounts/part-m-00003
```

9. The first six digits in the output are the account ID. Take note of highest account ID because you will use it in the next step.

Import incremental updates to accounts

As Loudacre adds new accounts, the account data in HDFS must be updated as accounts are created. You can use Sqoop to append these new records.

10. Run the `add_new_accounts.py` script to add the latest accounts to MySQL.

```
$ $DEV1/exercises/sqoop/add_new_accounts.py
```

11. Incrementally import and append the newly added accounts to the `accounts` directory. Use Sqoop to import on the last value on the `acct_num` column largest account ID:

```
$ sqoop import \
 --connect jdbc:mysql://localhost/loudacre \
 --username training --password training \
 --incremental append \
 --null-non-string '\\N' \
 --table accounts \
 --target-dir /loudacre/accounts \
 --check-column acct_num \
 --last-value <largest_acct_num>
```

Note: replace *<largest_acct_num>* with the largest account number.

12. List the contents of the `accounts` directory to verify the Sqoop import:

```
$ hdfs dfs -ls /loudacre/accounts
```

13. You should see three new files. Use Hadoop's `cat` command to view the entire contents of these files.

```
$ hdfs dfs -cat /loudacre/accounts/part-m-0000[456]
```

Import webpage data using an alternate field delimiter

14. We also want to import the webpage table to HDFS. But first look at a few records in that table using the `sqoop eval` command.

```
$ sqoop eval \
  --query "SELECT * FROM webpage LIMIT 10" \
  --connect jdbc:mysql://localhost/loudacre \
  --username training --password training
```

Notice that the values in the last column contain commas. By default, `sqoop` uses commas as field separators, but because the data itself uses commas, we can't do that this time.

15. Use sqoop to import the webpage table, but use the tab character (`\t`) instead of the default (comma) as the field terminator.

```
$ sqoop import \
  --connect jdbc:mysql://localhost/loudacre \
  --username training --password training \
  --table webpage \
  --target-dir /loudacre/webpage \
  --fields-terminated-by "\t"
```

16. Using Hue or the hdfs command line, review the data files imported to the `/loudacre/webpage` directory. Take note of the structure of the data; you will use this data in the next exercises.

Bonus Exercise

Use Sqoop to import only accounts where the person lives in California (state = "CA") and has an active account (acct_close_dt IS NULL).

Note: This bonus exercise must output to a different HDFS directory. Otherwise, the original table will be overwritten and you will need to delete the directory and import the table again.

If you need a hint or would like to check your work, you can find a solution in the `$DEV1/exercises/sqoop/solution/bonus-sqoop-query.sh` file.

This is the end of the Exercise

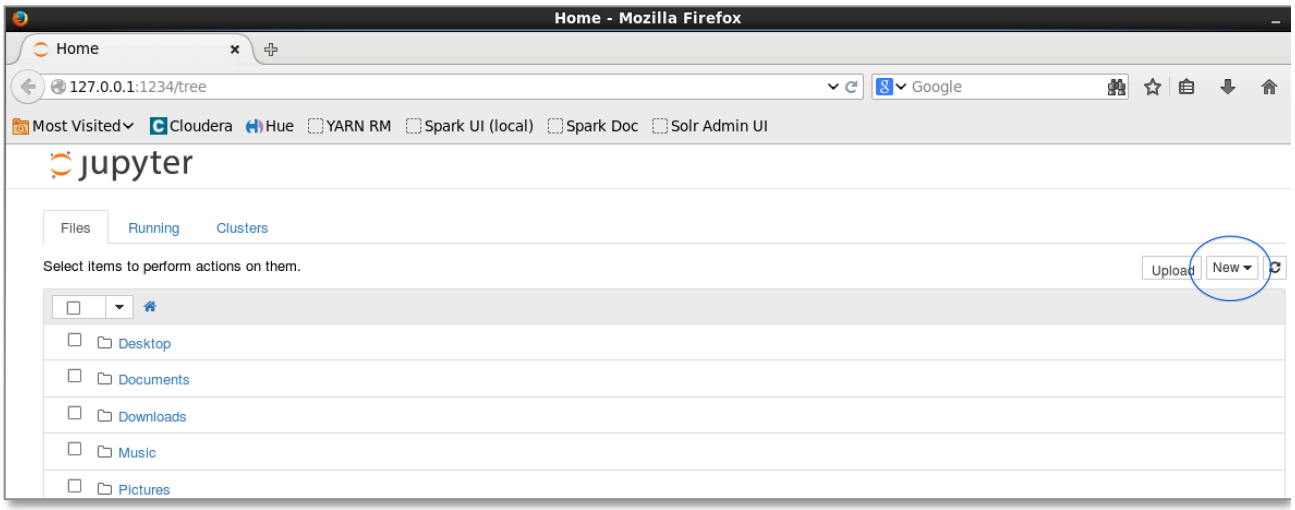
Appendix A: Enabling iPython Notebook

iPython Notebook is installed on the VM for this course. To use it instead of the command-line version of iPython, follow these steps:

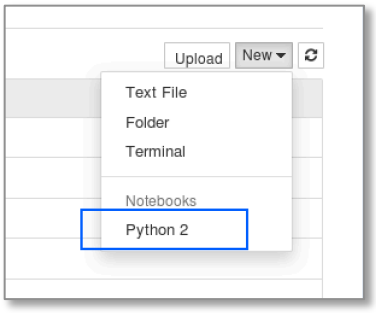
1. Open the following file for editing: `/home/training/.bashrc`
2. Uncomment out the following line (remove the leading #).

```
# export PYSPARK_DRIVER_PYTHON_OPTS='notebook ……..jax'
```

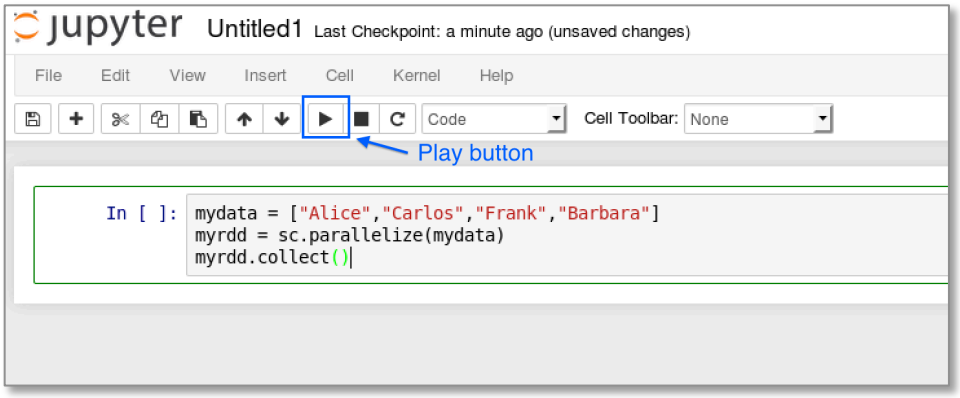
3. Save the file.
4. Open a new terminal window. (Must be a new terminal so it loads your edited `.bashrc` file).
5. Enter `pyspark` in the terminal. This will cause a browser window to open, and you should see the following web page:



6. On the right hand side of the page select **Python 2** from the **New** menu



7. Enter some spark code such as the following and use the play button to execute your spark code.



8. Notice the output displayed.