

Document Title: Cloudera Data Science lab guide

Date: September 2017

Version: 2.2

Authors: Colm Moynihan, Toby Ferguson

Cloudera Data Science Workbench Labs

Table of Contents

Introduction	2
Lab 1 – Self registration onto CDSW on AWS	3
Lab 2 – Creating a new project	5
Lab 3 - Visualization and Sharing	15
Lab 4 - Hadoop Integration	20
Lab 5 - Pushing the Boundaries	22
Lab 6 - SparklyR	24
Lab 6 - Shiny	27
Lab 8 - Scala	29
Lab 9 - Project Creation using Local Files	34
Lab 10 - Scheduling Jobs	36
Lab 11 – Face recognition with Python	44
Troubleshooting	47
DNS Issue	47
Spark R backend might have failed	47
Appendix	48
Cloudera Documentation	48
CDSW User Guide	48

Introduction

Cloudera Data Science workbench is a new product from Cloudera launched in May 2017. It is based on the acquisition of Sense.io that we made in March 2016. Cloudera has taken this product enhanced it and ensures that all workloads can be pushed down to Cloudera.

Accelerate data science from exploration to production using R, Python, Spark and more

For data scientists



Open data science, your way.

Use R, Python, or Scala with your favorite libraries and frameworks



No need to sample.

Directly access data in secure Hadoop clusters through Apache Spark and Apache Impala



Reproducible, collaborative research.

Share insights with your whole team

For IT professionals



Bring analysis to the data.

Give your data science team the freedom to work how they want, when they want



Secure by default.

Stay compliant with out-of-the-box support for full Hadoop security



Flexible deployment.

Run on-premises or in the cloud

Cloudera Data Science workbench supports the R, Python, Scala programming languages. That capability could certainly be useful to Cloudera; the software could enable companies to make the most of their data scientists, who can then be more efficient with their use of company time and infrastructure.



Programming language and software environment for statistical computing and graphics.

Best known in: **Academia and statistics community.**



High-level programming language for general-purpose programming.

Best known in: **Machine learning and data engineering community.**



General-purpose functional programming language with a strong static type system.

Best known in: **Data engineering community, due to Spark.**

Cloudera's goal with Cloudera Data Science workbench is to Help more data scientists use the power of Hadoop, make it easy and secure to add new users, use cases.

Why Hadoop for Data Science well here are the reasons:

High volume, low cost shared storage – More data more kinds of data

Parallel compute local to the data – more experiments, better results

Scalable, fault tolerant – easy to scale out, not just scale up

Flexible multipurpose data platform – easier path to production

Superior flexibility and price / performance to any other data platform

Lab 1 – Self registration onto CDSW on AWS

In this lab you'll learn how to:

- Signup
- Login to a Cloudera Data Science Workbench instance
- Navigate the Cloudera Data Science Workbench application

First thing you need to do is register onto Cloudera Data Science Workbench

URL: *Ask for details or see projector*

Select the link **"Sign Up for a New Account"**

Sign In to Data Science Workbench

Sign In

[Forgot Password?](#)

[Sign Up for a New Account](#)

Enter in your details.

- Your Full Name = Colm Moynihan
- Username is firstname initial + surname = cmoynihan
- email is needed
- Password = Cloudera123

Sign Up for Data Science Workbench

Colm Moynihan

cmoynihan

Your personal profile will be located at:

<http://cdsw.partner1.cds-sw-cloudera.eu/cmoynihan>

colm@cloudera.com

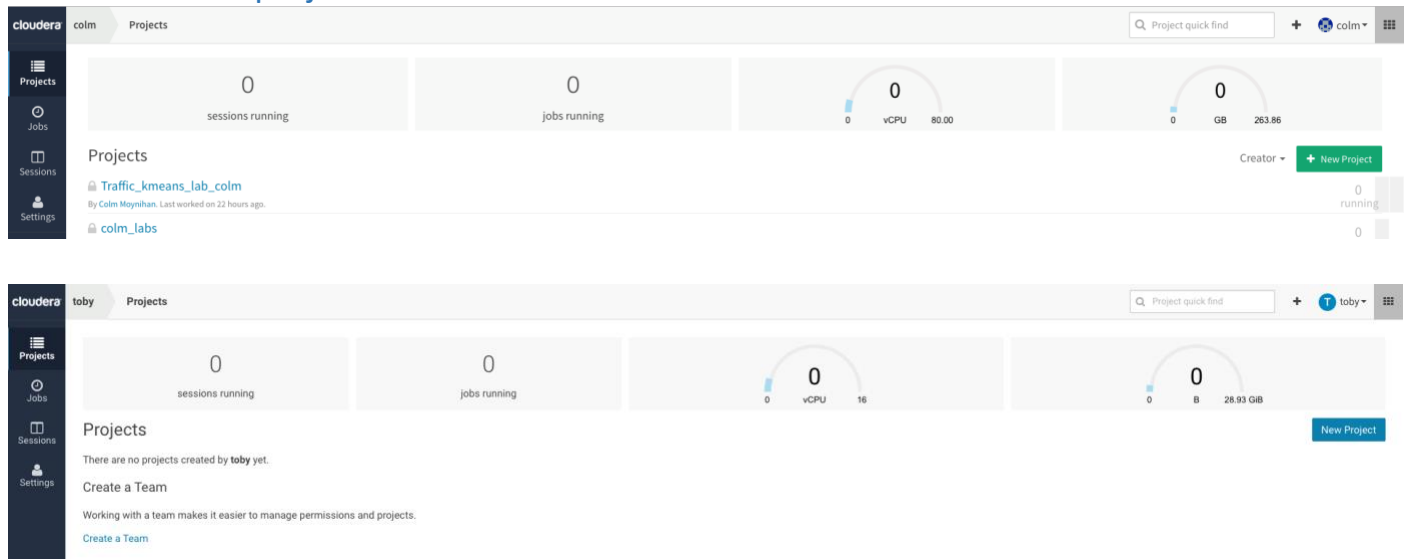
.....

Sign Up

Already have an account? [Sign In!](#)

Lab 2 – Creating a new project

You will see the project window as follows



You have on the left hand panel

Projects - where you create data science projects

Jobs - Run and schedule jobs and add dependencies

Sessions - Python, Scala or R sessions

Settings - User, Hadoop Authentication, SSH Keys and permission settings

In the top right hand corner you have

Search bar - for search for projects

+ adding new projects or new teams

User name - Account settings and Sign out - Same as settings in home screen

Let's create a new Project



Copy and paste this GitHub URL into the Git tab

<http://github.com/TobyHFerguson/cdsd-demo-short.git>

Project_name = your user name and labs
You should see this

Create a New Project

Project Name

Colm_CDSW_Demos

Project Visibility

- ☒ **Private** - Only added collaborators can view the project.
- ☐ **Public** - All authenticated users can view this project.

Initial Setup

Blank

Template

Local

Git

`https://github.com/TobyHFerguson/cdsw-demo-short.git`

Give Cloudera Data Science Workbench access to this SSH repo using your [public key](#).

Create Project

Account cmoynihan Colm_CDSW_Demos

Colm_CDSW_Demos

Jobs

This project has no jobs yet. Create a [new job](#) to document your analytics pipelines.

Files

<input type="checkbox"/>	Name ^
<input type="checkbox"/>	data
<input type="checkbox"/>	1_python.py
<input type="checkbox"/>	2_pyspark.py
<input type="checkbox"/>	3_tensorflow.py
<input type="checkbox"/>	4_sparklyr.R
<input type="checkbox"/>	4a.R
<input type="checkbox"/>	README.md
<input type="checkbox"/>	utils.py

On the left hand side panel you will see a new menu item Team where you can add team members to your project.

As an example

Collaborators

This project is **private**. Only collaborators can view and edit this project. [Change Settings](#).

Add Collaborator

Search by name, username, or email... [Add](#)

Collaborator	Permission
cmoynihan	Admin
Admin	Admin change delete

I have added Admin as an Administrator – this is optional; we just want to show you one of CDSW's sharing features.

Click on the Settings icon and then the Engine tab:

The screenshot shows the 'Project Settings' page with the 'Engine' tab selected. The left sidebar contains navigation links: Overview, Jobs, Sessions, Files, Team, and Settings (highlighted). The main content area has tabs for Options, Engine, Tunnels, Git, and Delete Project. Under the 'Engine Image' section, there is a text input field containing 'Base Image v1, docker.repository.cloudera.com/cdsw/engine:1'. Below this is the 'Environmental Variables' section, which includes a table with columns 'Name', 'Value', and 'Actions'. The table is currently empty, and there is an 'Add' button in the 'Actions' column. A 'Save Environment' button is located below the table. The 'Security' section at the bottom explains that environmental variable values are only visible to collaborators with write or higher access.

Project Settings

Options Engine Tunnels Git Delete Project

Engine Image

Select the Docker image that Cloudera Data Science Workbench should use to run sessions and jobs in this project. If you'd like to use a different image, contact your site administrator.

Base Image v1, docker.repository.cloudera.com/cdsw/engine:1

Environmental Variables

Set project environmental variables that can be accessed from your scripts.

Name	Value	Actions
<input type="text"/>	<input type="text"/>	<button>Add</button>

Press tab or enter to add another.

Save Environment

Security

Environmental variable **values** are only visible to **collaborators** with **write** or higher access. They are a great way to securely store confidential information such as your AWS or database credentials. Names are available to all users with access to the project.

Under Project Settings you will see:

Options - Project Name and Description, Private or Public

Environment - Spark config and environment variables

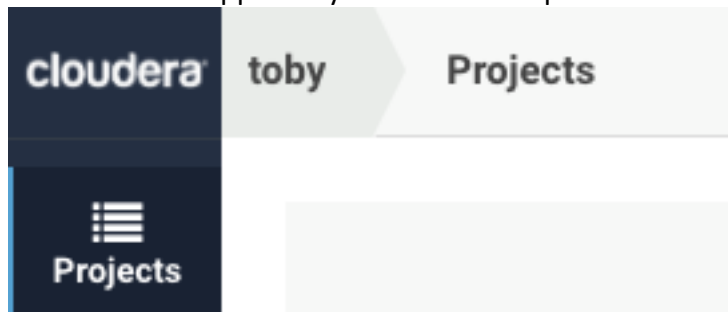
Tunnels - SSH tunnels allow you to easily connect to firewalled resources such as databases or Hadoop

Git - **Git Webhooks** To enable automatic git pulls to the underlying filesystem of your project

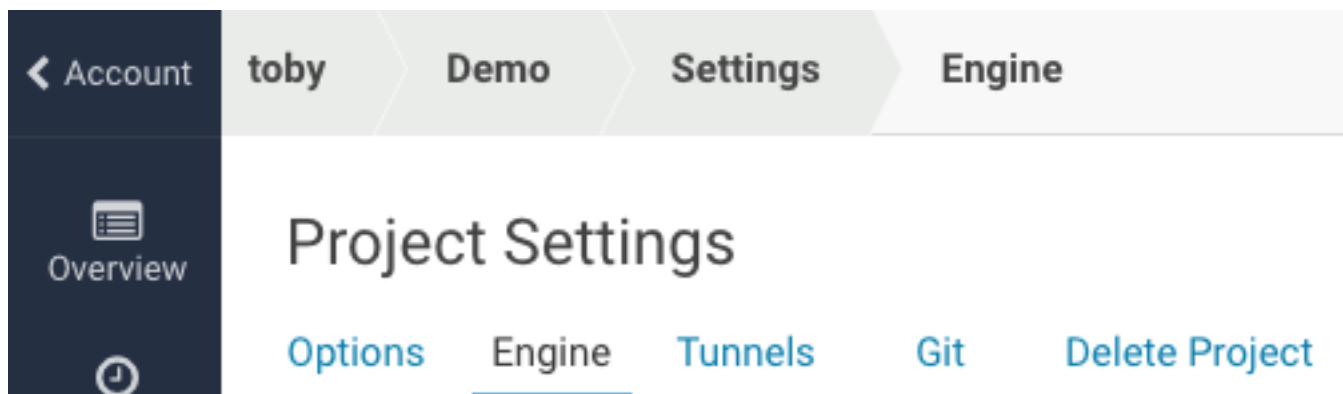
And this is where if you have to (please don't!) **Delete Project**

At the top left hand corner

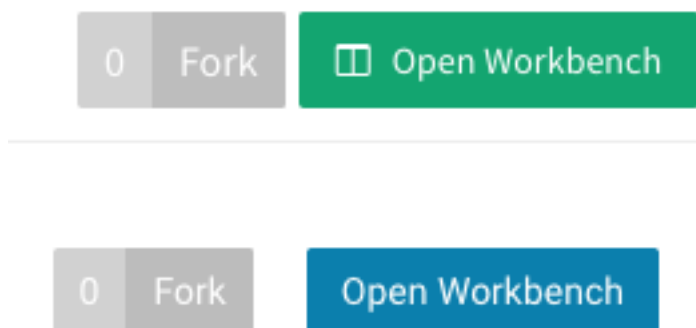
Cloudera - will appear if you are at the top level



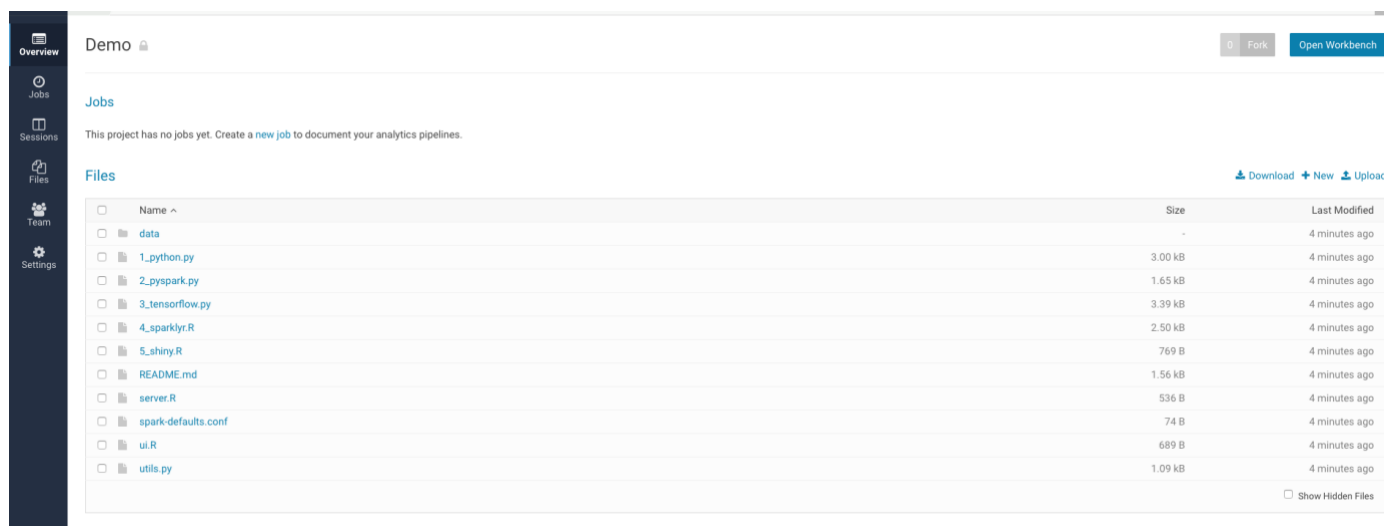
< **Account** - will appear if you are at the project level



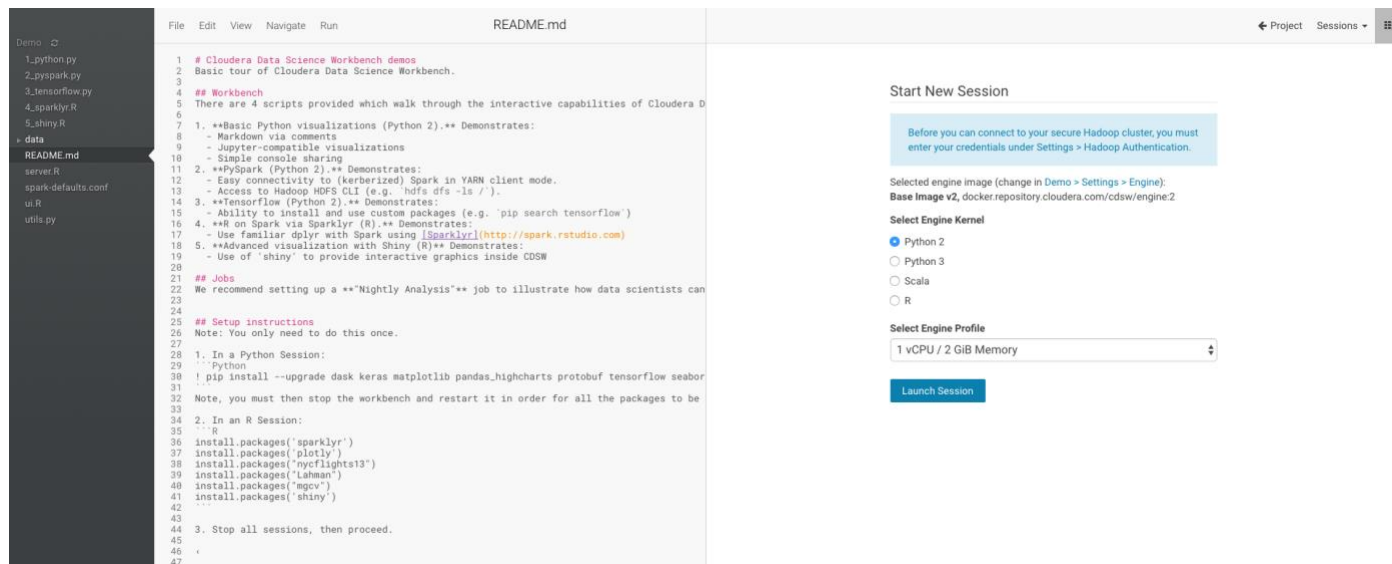
Click on Overview
Click on Open Workbench




Launch a Python Session as shown below:



1. This shows you the files that you have in your project.
2. Hit 'Open Workbench' and you go to the Workbench. Select the README.md on the left hand side and you should see something like this:



3. On the far left is a file browser (note the little 'refresh' icon at the top: )
4. In the middle is an editor open on the file that you selected - in this case the README.md file.
5. On the right is a Session Start tile - in this case it's waiting for you to select an engine to run (so far your project has file space but no compute sessions).

6. Select the Python 2 kernel; select the 1 vCPU/2GiB Engine (and use this size engine for all your Python sessions in this workshop) and then the Launch Session button.

Start New Session

Before you can connect to your secure Hadoop cluster, you must enter your credentials under [Settings > Hadoop Authentication](#).

Selected engine image (change in [Colm_CDSW_Demos > Settings > Engine](#)):

Base Image v2, docker.repository.cloudera.com/cdsw/engine:2

Select Engine Kernel

☒ Python 2

☐ Python 3

☐ Scala

☐ R

Select Engine Profile

1 vCPU / 2 GiB Memory

Launch Session

7. This will startup a Python engine and the right hand side will become two tiles. The top one is an output tile and the bottom one (with a red, then green left hand border) is a shell input window.

```
1 # Cloudera Data Science Workbench demos
2 Basic tour of Cloudera Data Science Workbench.
3
4 ## Workbench
5 There are 4 scripts provided which walk through the interactive capabilities of Cloudera D
6
7 1. **Basic Python visualizations (Python 2)** Demonstrates:
8   - Markdown via comments
9   - Jupyter-compatible visualizations
10  - Simple console sharing
11 2. **PySpark (Python 2)** Demonstrates:
12  - Easy connectivity to (kerberized) Spark in YARN client mode.
13  - Access to Hadoop HDFS CLI (e.g. 'hdfs dfs -ls /').
14 3. **Tensorflow (Python 2)** Demonstrates:
15  - Ability to install and use custom packages (e.g. 'pip search tensorflow')
16 4. **R on Spark via Sparklyr (R)** Demonstrates:
17  - Use familiar dplyr with Spark using [Sparklyr](http://spark.rstudio.com)
18 5. **Advanced visualization with Shiny (R)** Demonstrates:
19  - Use of 'shiny' to provide interactive graphics inside CDSW
20
21 ## Jobs
22 We recommend setting up a **Nightly Analysis** job to illustrate how data scientists can
23
24
25 ## Setup instructions
26 Note: You only need to do this once.
27
28 1. In a Python Session:
29 ::python
30 ! pip install --upgrade dask keras matplotlib pandas_highcharts protobuf tensorflow seaborn
31
32 Note, you must then stop the workbench and restart it in order for all the packages to be
33
34 2. In an R Session:
35 ::R
36 install.packages('sparklyr')
37 install.packages('plotly')
38 install.packages('nyerflights13')
39 install.packages('Lahman')
40 install.packages('mgcv')
41 install.packages('shiny')
42
43
44 3. Stop all sessions, then proceed.
45
46
47
```

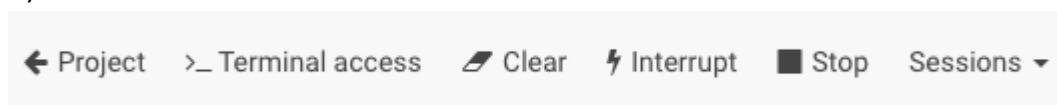
8. Look at Line 30 of README.md:
9. `! pip install --upgrade dask keras matplotlib pandas_highcharts protobuf tensorflow seaborn`
10. Question: What does this do? If you know Python it should be obvious; if you don't, then let me tell you: this is an execution of pip (a Python package manager), which will tell the system to install or upgrade the listed python packages
11. Put your cursor at end of line 30 and select 'Run Line(s)':

```
! pip install --upgrade dask keras matplotlib pandas_highcharts protobuf tensorflow seaborn
Note, you must then stop the workbench and restart it in order for all the packages to be
```

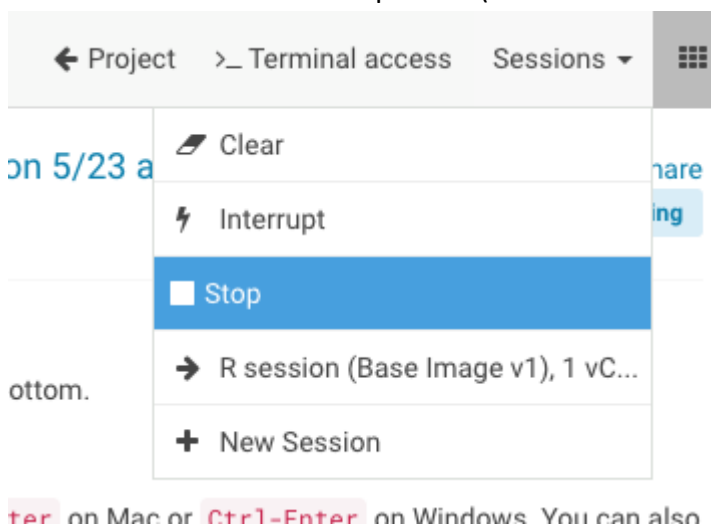
12. The cursor on the left should turn to red and, after a few seconds, you should see output like this:

13. What's happening here is that the code in the file is being executed in the console (did you notice the left hand edge turned red?) and now you can see the output in the right hand screen.

14. Stop this session. The Stop button is either on the top menu bar (when there's sufficient room for it):



15. Or it's in the the Session drop down (when there's little room for buttons):



16. Now launch an R session. **Use a 2vCPU, 4GiB Engine.** Use this size engine for **all** your R sessions during this workshop.

Start New Session

Before you can connect to your secure Hadoop cluster, you must enter your credentials under Settings > Hadoop Authentication.

Selected engine image (change in [Demo > Settings > Engine](#)):
Base Image v2, docker.repository.cloudera.com/cdsw/engine:2

Select Engine Kernel

- ☐ Python 2
- ☐ Python 3
- ☐ Scala
- ☒ R

Select Engine Profile

2 vCPU / 4 GiB Memory

Launch Session

17. This time we're going to execute lines 37 through 42 in the R session (these numbers could be off by 1 or so ... look at the images below and figure out what you need to select). Select and highlight just these lines then select Run Lines to execute as in prior step:

```
35 2. In an R Session:
36 ```R
37 install.packages('sparklyr')
38 install.packages('plotly')
39 install.packages("nycflights13")
40 install.packages("Lahman")
41 install.packages("mgcv")
42 install.packages('shiny')
```

```
install.packages('sparklyr')
install.packages('plotly')
install.packages("nycflights13")
install.packages("Lahman")
install.packages("mgcv")
install.packages('shiny')
```

Run Line(s)

⌘Enter

Select All

⌘A

This process will take 2 or 3 minutes because code is being downloaded, compiled and installed into your project's workbench.

Note that this workbench is independent of any other project's workbench. You want to try out different and conflicting libraries? Go for it. Just start another project, open the workbench, pick the libraries you want, install them and off you go. Just like on your laptop, but this is managed, secure, stable, won't get stolen from a taxi, is always available and easily shared!

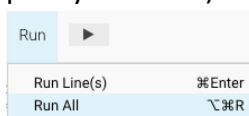
All of this isolation is achieved by mounting filesystems (one or more per project) into docker containers (one per engine). The details don't matter, except to note that now you can really go mad and try out lots of different and conflicting permutations without having to go down the complex path of virtual environments etc. It's all been done for you!

Lab 3 - Visualization and Sharing

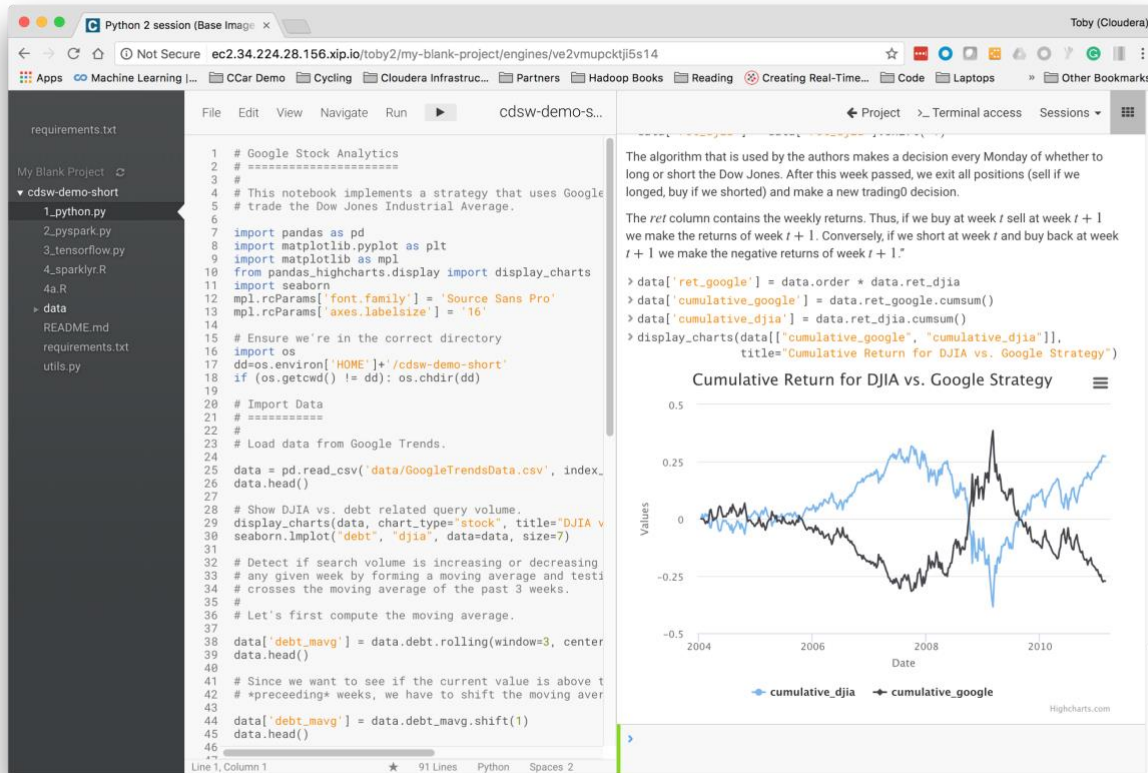
Check that you have NO sessions running - if any sessions are running then stop them now. You've setup your environment and those sessions can be safely disposed of.

Data Science is often about visualizing ideas, and then sharing them to persuade others to take action. CDSW lets you use the visualization tools you'd use naturally, and adds a neat twist to the whole idea of sharing. Let's get started:

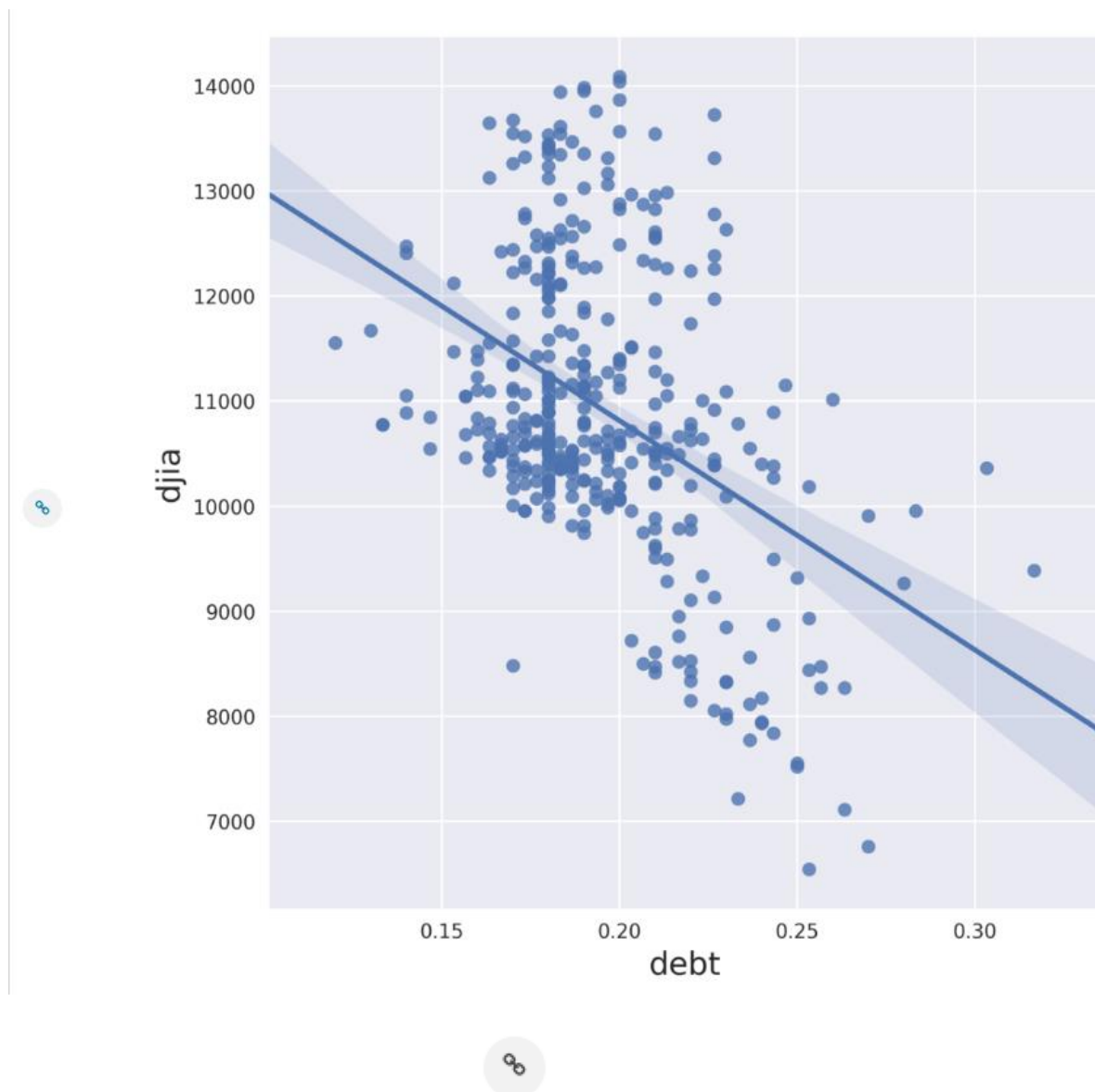
1. Start up a new Python 2.0 session (1vCPU, 2GiB) in the same manner you did before.
2. Select 1_python.py in the file browser
3. Run the entire file (multiple ways of doing that - try to figure out more than one way. It should be pretty obvious!).



4. You should end up with some nice graphs in the output window:



5. You can see that CDSW is very similar to a notebook, supporting the same visualization tools. However, unlike a notebook, it doesn't use cells: instead it uses markup in the source file, and an output window. Furthermore, that window has some interesting properties ...
6. Scroll up to find this diagram:



- On the left is a little chain link button:
- Click on it and you'll see beneath the chart some html that can be used to embed that chart into a web site:

Copy and paste to embed this cell in your website!

width (optional)

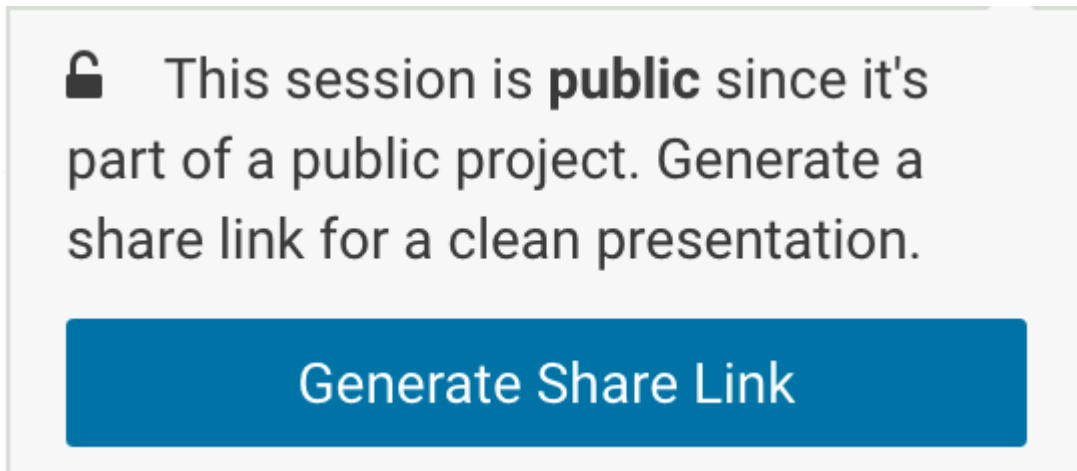
height (optional)

```
<div id='sense-widget-s7dkodjq' class="sense-embed-container" style="width:100%;"><script id='embedding-script-21' src='http://consoles.cds.w.52.214.150.212.nip.io/js/sense-embed.js' data-cell-url='http://livelog.cds.w.52.214.150.212.nip.io/topics/xeupfj2oeyaktdpa_output/21' data-auth-token='eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhdXRoIjpp7InJlYWQlOlsieGV1cGZqMm9leWFrdGRwYV8qIl0sIndyaXRlIjpbdata-assets-cdn-root='http://consoles.cds.w.52.214.150.212.nip.io/0/7/xeupfj2oeyaktdpa/' data-widget='sense-widget-s7dkodjq' data-width="" data-height=""></script></div>
```

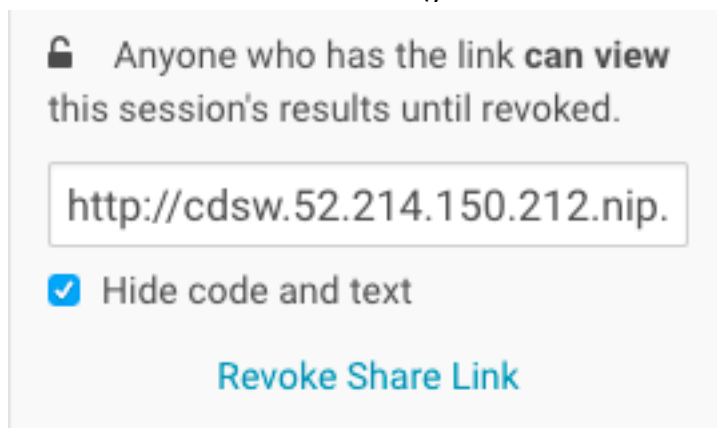
9. Scroll to the top of the window and you'll see this on the far right (the exact layout depends upon the real estate available – you might have to expand your browser window to see the following links and they might be laid out vertically or horizontally):



10. Hit the 'collapse' link and see the difference in the output window.
11. Question: What difference did you see? How might this be used? Is it useful?
12. Notebooks have great output, but how do you share what they show you? CDSW solves this by simply providing a link to the output that you can send to anyone and they can see the output. Try it:
13. Select the 'Share' link:

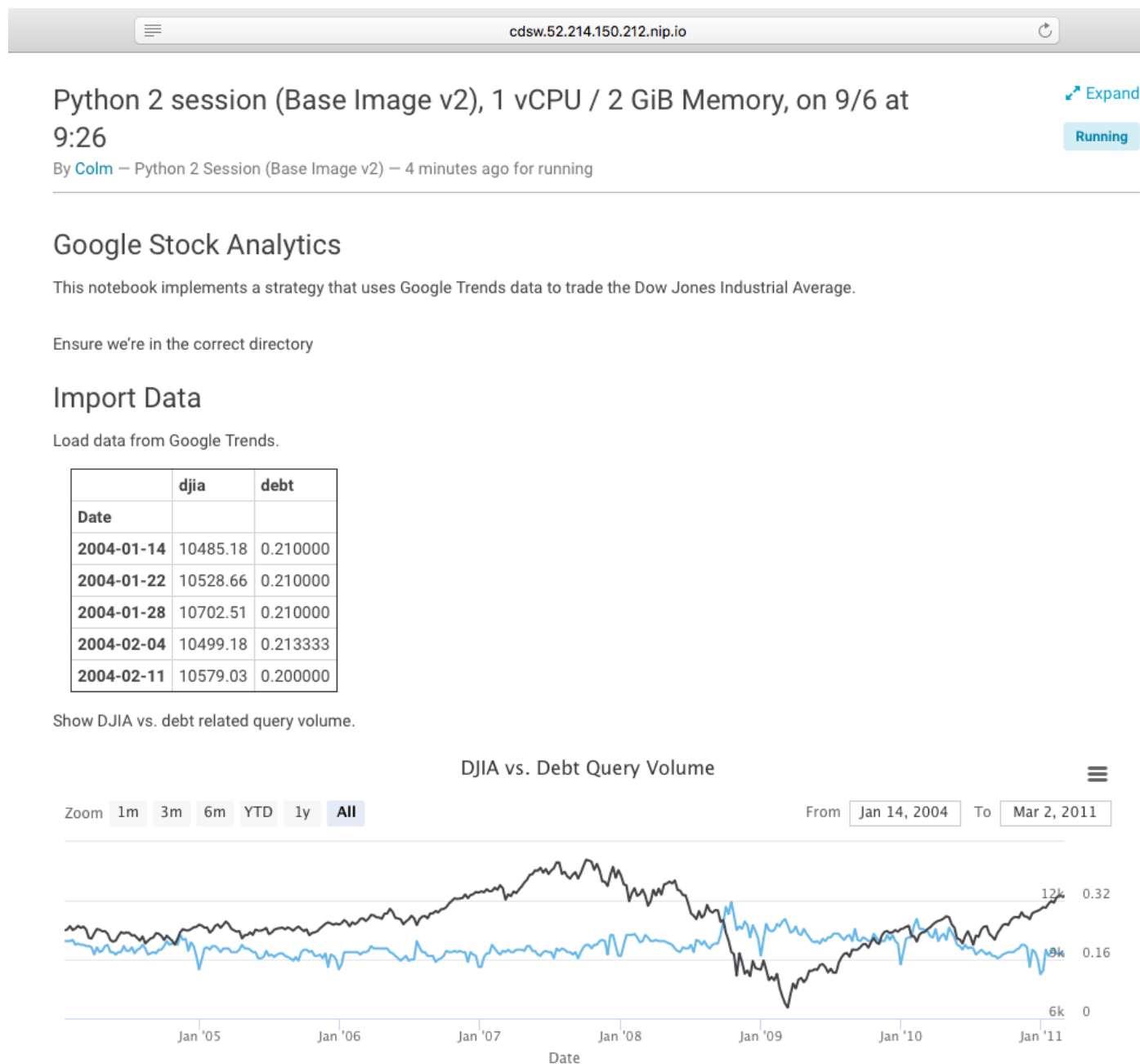


14. And then 'Generate Share Link' (your URL will be similar, but different from this one):



15. Cut and paste that link and put it into some other browser (best to be a completely different browser than the one you're logged in with, but not that important)

16. You should see that you have access to almost the same output window (this new one doesn't have this share link!)



So we've demonstrated how CDSW is like a notebook, but is perhaps more powerful, and has great sharing capability. Let's go on to see about integration with Hadoop!

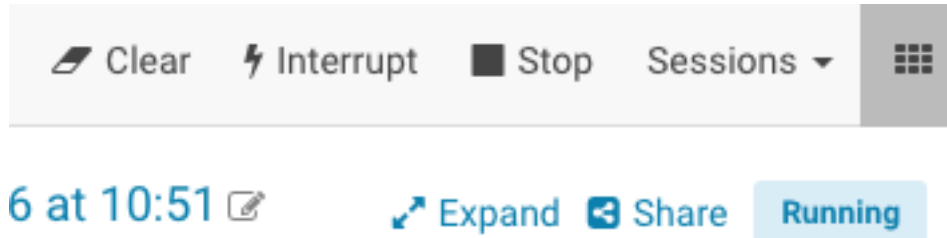
Lab 4 - Hadoop Integration

In this lesson we'll see two mechanisms for integrating with Hadoop:

1. Filesystem - storing data in Hadoop itself using HDFS
2. Computation - executing code on the Hadoop cluster via Spark

Execute the following instructions.

1. Clean up your Python session by hitting the 'clear' button and the 'expand' link (if available)



2. Select the 2_pyspark.py file
3. Launch a Python 2 session 1 vCPU / 2 GB RAM
4. Run line 34:
5. `!hdfs dfs -put -f $HOME/cdsd-demo-short/data/kmeans_data.txt /user/$HADOOP_USER_NAME`
6. If you get a **permission denied like this:**

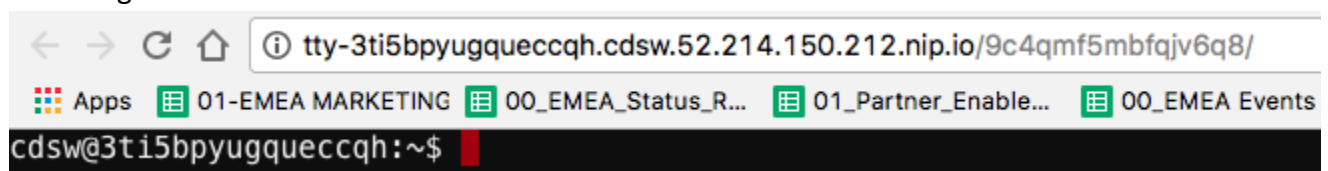


then do the following:

select Terminal

>_ Terminal access

You will get



Enter in the following command

export HADOOP_USER_NAME=hdfs

```
~$ export HADOOP_USER_NAME=hdfs
```

You will then create a directory in the /user/ folder

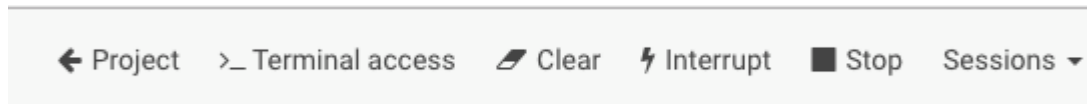
please use **your own username**: should be first name initial plus surname (I'll use my name – please use yours!)

```
$ hdfs dfs -mkdir /user/cmoynihan
```

And then enter – replacing cmoynihan with your username:

```
$ hdfs dfs -chown cmoynihan:cmoynihan /user/cmoynihan
```

7. Execute the 2_pyspark.py file in your already running Python session
8. **Question:** What did it do?
9. **Question:** What kind of thing is the variable 'data'? (try typing 'data' into the console and seeing what gets printed out.
10. Open a terminal using the 'terminal' icon in the top right:



11. Execute 'hdfs dfs -ls' to see the data file in the hadoop file system (or, to show off, execute '! hdfs dfs -ls' in the python console to do the same thing!)

```
> ! hdfs dfs -ls
Found 2 items
drwxr-xr-x  - hdfs_super supergroup      0 2017-09-06 13:06 .sparkStaging
-rw-r--r--  3 hdfs_super supergroup     71 2017-09-06 13:06 kmeans_data.txt

> ! hdfs dfs -ls
```

If everything went correctly you'll see that we demonstrate:

- Natural integration with HDFS - it's just a path to a file!
- Natural parallel computation across the cluster using Spark

Lab 5 - Pushing the Boundaries

So you've just heard that the latest thing from Google is [Tensorflow](#) and you're keen to get started. You're going to need to install some custom packages and, more importantly, connect a program providing a web interface so that you can view the results. Your IT department isn't going to help you - you're going to want to do this experiment on your own.

Fortunately CDSW enables you to install custom libraries. This cluster might be managed by IT but you can still get your libraries in there to do the work you need ...

We've done the hard work for you - take a look:

1. Select 3_tensorflow.py
2. Lines 6 through 13 show the imports of the various libraries (we installed them in Step 3)
3. Run 3_tensorflow.py in your current Python session (you might like to 'clear' your output screen first)- the input handwritten number images are shown, along with some images of feature maps for particular numbers

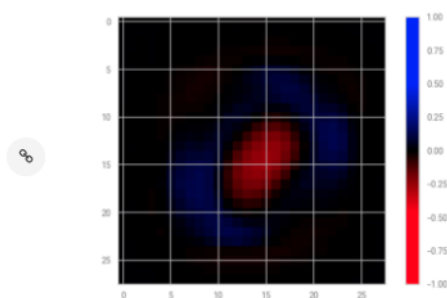
Examine layers

A red/black/blue colormap

```
> cdict = {'red': [(0.0, 1.0, 1.0),
                  (0.25, 1.0, 1.0),
                  (0.5, 0.0, 0.0),
                  (1.0, 0.0, 0.0)],
          'green': [(0.0, 0.0, 0.0),
                   (1.0, 0.0, 0.0)],
          'blue': [(0.0, 0.0, 0.0),
                   (0.5, 0.0, 0.0),
                   (0.75, 1.0, 1.0),
                   (1.0, 1.0, 1.0)]}

> redblue = matplotlib.colors.LinearSegmentedColormap('red_black_blue', cdict, 256)
> wts = W.eval(sess)
> for i in range(0,5):
    im = wts.flatten()[i::10].reshape((28,-1))
    plt.imshow(im, cmap = redblue, clim=(-1.0, 1.0))
    plt.colorbar()
    print("Digit %d" % i)
    plt.show()
```

Digit 0



Key takeaways: You were able to install and use custom third party libraries, as well as run an application that you can connect to from an external application

STOP all your Python sessions now (you should only have one but sometimes people get carried away). This will help ensure there're plenty of resources for you and the others in the workshop.

Lab 6 - SparklyR

We've focused on python integration, but just to show we can do similar things with R, let's take a look at the R programs and execute them.

This lab requires that R is setup correctly. We provided instructions [earlier](#), but if you skipped them then do this:

Ensure that you've started up an R session (**2vCPU, 4GiB engine**) and executed lines 37 through 42 from the README.md file. You'll only have to do this once for this project so if you've already done it don't do it again.

1. Create a new R Session.

Start New Session

Before you can connect to your secure Hadoop cluster, you must enter your credentials under Settings > Hadoop Authentication.

Selected engine image (change in [Demo > Settings > Engine](#)):

Base Image v2, docker.repository.cloudera.com/cdsw/engine:2

Select Engine Kernel

- ☐ Python 2
- ☐ Python 3
- ☐ Scala
- ☒ R

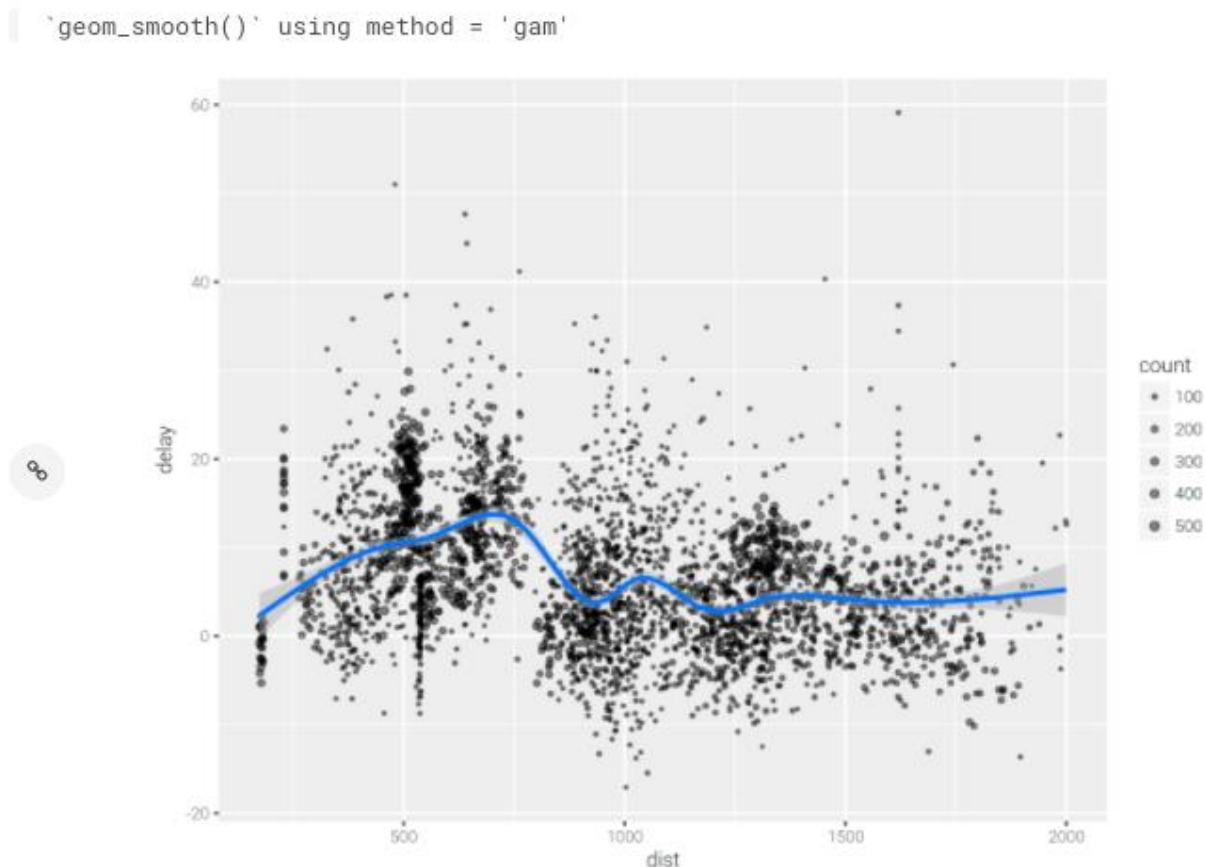
Select Engine Profile

2 vCPU / 4 GiB Memory

Launch Session

2. Select (and run) 4_sparklyr.R

The right hand side output window should (eventually) look like this (more or less - depending on your screen real-estate):



Machine Learning

You can orchestrate machine learning algorithms in a Spark cluster via the machine learning functions within sparklyr. connect to a set of high-level APIs built on top of DataFrames that help you create and tune machine learning workflow

In this example we'll use `ml_linear_regression` to fit a linear regression model. We'll use the built-in `mtcars` dataset, and predict a car's fuel consumption (`mpg`) based on its weight (`wt`) and the number of cylinders the engine contains (`cyl`). each case that the relationship between `mpg` and each of our features is linear.

copy mtcars into spark

```
> mtcars_tbl <- copy_to(sc, mtcars, overwrite = TRUE)
```

transform our data set, and then partition into 'training', 'test'

```
> partitions <- mtcars_tbl %>%
```

```
>
```

Have a read through the Machine Learning section

3. Can you figure out some of the things it's doing? If you know R, and if you know sparklyr, then you can get detailed; if you don't know R then simply 'collapse' the output and see if you can make sense of the analysis without looking at any code ... hopefully you can!

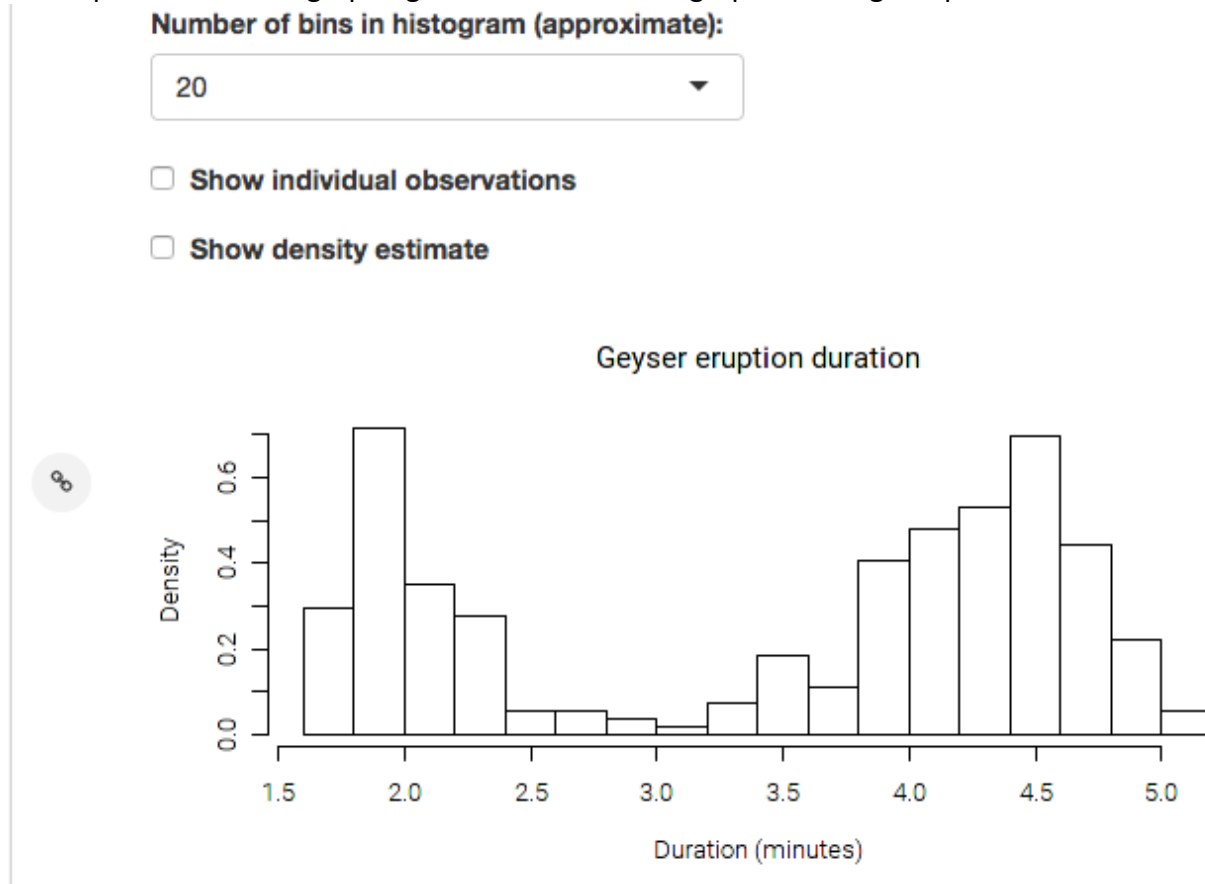
Lab 6 - Shiny

R has a great interactive experience using the shiny package. In this lab we'll create an interactive histogram and you can work with it to find out the frequency distribution of the period between Yellowstone Geyser eruptions!

1. clear the R session screen and then run the 5_shiny.R application

This will start up a shiny server in the local context (line 16), and connect it to the server/ui code (in server.R and ui.R, respectively).

Your users are then presented with a nice little application where they can experiment with changing some of the parameters and graphing features of R's base graphics histogram plot!



2. You can also change the number of bins in histogram to 50

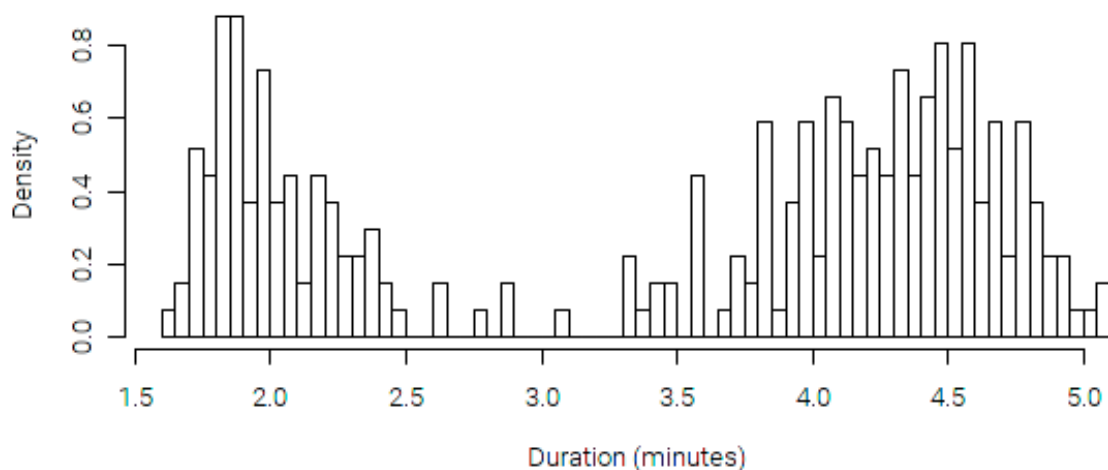
Number of bins in histogram (approximate):

50

☐ Show individual observations

☐ Show density estimate

Geyser eruption duration



Try generating a share link and opening up the share in another browser window - amazingly enough each browser share is independent, allowing your users to share the same underlying experience, but with their individual data inputs!

When you've finished here then stop the R session just to free up some resources.

Have you Stopped the R Session ?

Question: Is this kind of interactivity with data likely to change the way your business users understand and appreciate the work of the Data Scientists?

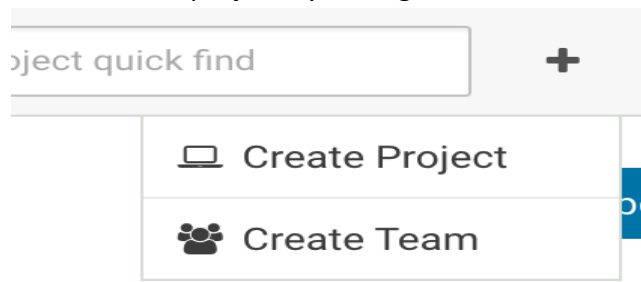
Lab 8 - Scala

In this lab we show how you can use the 'Template' mechanism to get started with a simple Scala example. Note that the built in templates and example code aren't written with multiple users in mind, so you might see file access and permission errors due to the fact that other students might've created or deleted files before you!:

1. Navigate to the project space by selecting "project":



2. Create a new project by hitting the '+' button on the top right and selecting 'create project':



3. In the Create new Project window that comes up provide a name for your new project ('Scala', for example), and then choose the Scala template in the Initial Setup drop down menu:

Create a New Project

Project Name

Scala

Project Visibility

☒ **Private** - Only added collaborators can view the project.

☐ **Public** - All authenticated users can view this project.

Initial Setup

Blank

Template

Local

Git

Scala

Templates include example code to help you get started.

Create Project

4. Create the project. You'll see the File Browser view onto the project:

Scala

Jobs

This project has no jobs yet. Create a [new job](#) to document your analytics pipelines.

Files

Name ^	Size	Last Modified
data	-	just now
examples	-	just now
auction-analysis.scala	2.03 kB	just now
pi.scala	837 B	just now
wordcount.scala	476 B	just now

[Download](#) [New](#) [Upload](#)

☐ Show Hidden Files

This project doesn't contain a README.md file. Consider adding one that describes your project.

- Hit 'Open Workbench' in the top right and let's go run some Scala code:
- Start a Scala session

Select Engine Kernel

- ☐ Python 2
- ☐ Python 3
- ☒ Scala
- ☐ R

Select Engine Profile

1 vCPU / 2 GiB Memory

Launch Session

- The Scala example project includes its own data set that needs to be moved into HDFS, since that is where the scala code expects to find it. Open a terminal and execute the following shell commands to do this. Note how you have ready access to your own project's data (purely local to you) and the secure (and massive) HDFS cluster:
- Open Terminal
- `hdfs dfs -put -f data /tmp`

```
cdsw@bm8gfdncr5ziertv:~$ hdfs dfs -put -f data /tmp
```

- If you get a permission error (because the file has already been created):
- Use `hdfs dfs -put -f data /tmp_cmoynihan`
- Where cmoynihan is your username
- Execute one or more of the various scala files from the examples folder in the Workbench

14. Open example -> kmeans.scala and execute Run All

File	Edit	View	Navigate	Run	examples/kmeans.scala	Project	Terminal access	Clear
1	import org.apache.spa			Run Line(s)	KMeansModel}			
2	import org.apache.spa			Run All				
3	import org.apache.com							
4	import java.io.File							
5	import sys.process._							
6								
7	//Load local data to hdfs							
8	"hdfs dfs -put data/kmeans_data.txt /tmp" !							
9								
10	//example kmeans clustering script							
11	val data = sc.textFile("/tmp/kmeans_data.txt")							
12	val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble))).cache()							
13								
14	// Cluster the training data set into two classes with KMeans							
15	val numClusters = 2							
16	val numIterations = 20							
17	val clusters = KMeans.train(parsedData, numClusters, numIterations)							
18								
19	// Evaluate clustering by computing Within Set Sum of Squared Errors							
20	val WSSSE = clusters.computeCost(parsedData)							
21	println(s"Within Set Sum of Squared Errors = \$WSSSE")							
22								
23	// Save the model							
24	val output = "output/KMeansExample/KMeansModel"							
25	FileUtils.deleteQuietly(new File(output))							
26	clusters.save(sc, output)							
27								
28	//example of loading and predicting on the model we created							
29	val sameModel = KMeansModel.load(sc, output)							
30	sameModel.clusterCenters.zipWithIndex.foreach { case (center, idx) =>							
31	println(s"Cluster Center \${idx}: \${center}")							
32	}							
33	sameModel.predict(Vectors.dense(7,5,6))							
34								

If you get a `FileAlreadyExistsException` executing 'clusters.save(sc,output)' then you can ignore the error and finish this lab by just executing the lines after that one:


```
> clusters.save(sc, output)
✖ Name: org.apache.hadoop.mapred.FileAlreadyExistsException
Message: Output directory hdfs://ip-10-0-100-220.eu-west-1.compute.internal:8020/user/hdfs_super/output/KM
StackTrace:   at org.apache.hadoop.mapred.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:131)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopDataset$1.apply$mcV$sp(PairRDDFunctions.scala:1096)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopDataset$1.apply(PairRDDFunctions.scala:1096)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopDataset$1.apply(PairRDDFunctions.scala:1096)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
              at org.apache.spark.rdd.RDD.withScope(RDD.scala:362)
              at org.apache.spark.rdd.PairRDDFunctions.saveAsHadoopDataset(PairRDDFunctions.scala:1096)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopFile$4.apply$mcV$sp(PairRDDFunctions.scala:1035)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopFile$4.apply(PairRDDFunctions.scala:1035)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopFile$4.apply(PairRDDFunctions.scala:1035)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
              at org.apache.spark.rdd.RDD.withScope(RDD.scala:362)
              at org.apache.spark.rdd.PairRDDFunctions.saveAsHadoopFile(PairRDDFunctions.scala:1035)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopFile$1.apply$mcV$sp(PairRDDFunctions.scala:961)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopFile$1.apply(PairRDDFunctions.scala:961)
              at org.apache.spark.rdd.PairRDDFunctions$$anonfun$saveAsHadoopFile$1.apply(PairRDDFunctions.scala:961)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
              at org.apache.spark.rdd.RDD.withScope(RDD.scala:362)
              at org.apache.spark.rdd.PairRDDFunctions.saveAsHadoopFile(PairRDDFunctions.scala:960)
              at org.apache.spark.rdd.RDD$$anonfun$saveAsTextFile$1.apply$mcV$sp(RDD.scala:1468)
              at org.apache.spark.rdd.RDD$$anonfun$saveAsTextFile$1.apply(RDD.scala:1468)
              at org.apache.spark.rdd.RDD$$anonfun$saveAsTextFile$1.apply(RDD.scala:1468)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
              at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
              at org.apache.spark.rdd.RDD.withScope(RDD.scala:362)
              at org.apache.spark.rdd.RDD.saveAsTextFile(RDD.scala:1468)
              at org.apache.spark.mllib.clustering.KMeansModel$SaveLoadV1_0$.save(KMeansModel.scala:128)
              at org.apache.spark.mllib.clustering.KMeansModel.save(KMeansModel.scala:94)

> val sameModel = KMeansModel.load(sc, output)
> sameModel.clusterCenters.zipWithIndex.foreach { case (center, idx) =>
  println(s"Cluster Center ${idx}: ${center}")
}
Cluster Center 0: [0.1,0.1,0.1]
Cluster Center 1: [9.1,9.1,9.1]

> sameModel.predict(Vectors.dense(7,5,6))
1
```

Question: How will you use templates when demonstrating CDSW to your friends and colleagues?

Remember to stop your scala Session

Lab 9 - Project Creation using Local Files

1. Create a new project, naming it 'Local', and select the 'Local tab':

Create a New Project

Project Name

Local

Project Visibility

- ☒ **Private** - Only added collaborators can view the project.
- ☐ **Public** - All authenticated users can view this project.

Initial Setup

Blank

Template

Local

Git

Upload .zip or .tar.gz

Upload folder

Or Drag and Drop Files Here

Create Project

2. Try adding a file or folder and then create your project
3. If you have some code you want to try then select that - otherwise just note that the project was created from the file(s) you selected.
4. You might've noticed in the File browser window that the ability to upload and download files is there for a project, no matter how you started the project:

[← Account](#)

toby2

My Blank Project

Overview

Jobs

Sessions

Files

Team

Settings

My Blank Project

0 Fork

Open Workbench

Jobs

This project has no jobs yet. Create a [new job](#) to document your analytics pipelines.

Files

Download New Upload

<input type="checkbox"/>	Name ^	Size	Last Modified
<input type="checkbox"/>	cdsw-demo-short	-	yesterday
<input type="checkbox"/>	R	-	yesterday

☐ Show Hidden Files

This project doesn't contain a README.md file. Consider adding one that describes your project.

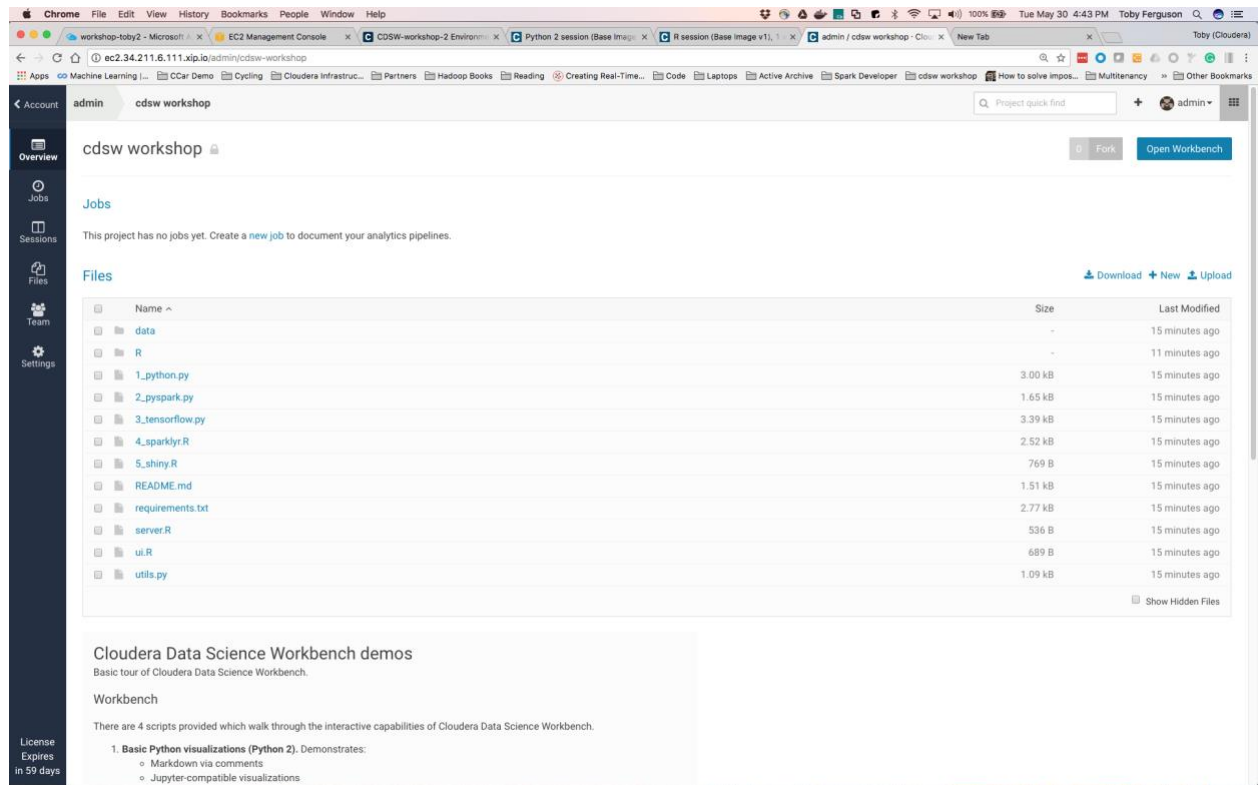
Question: With the combination of git, blank projects and uploading from a local file system on your laptop do you feel pretty confident you can get the data and code you want into the CDSW environment?

Lab 10 - Scheduling Jobs

Its often the case that you need to execute tasks on a periodic basis, and to execute one or more tasks once some other task has succeeded. Obviously there are sophisticated workflow engines but for simple workflows CDSW has a jobs system built in.

This lab goes through the mechanics of creating a simple multi-step job process.

1. Open up your 'cdsw workshop' project to get to this screen:



2. You need to be in a project to create a Job
3. Select the 'new job' link in the middle of the page, and you'll get to the following screen (there are other ways of getting to this next screen - its an exercise for the student to figure out what they might be):

The screenshot shows the 'Create a Job' page in the Cloudera Data Science Workshop. The 'General' section contains the following fields:

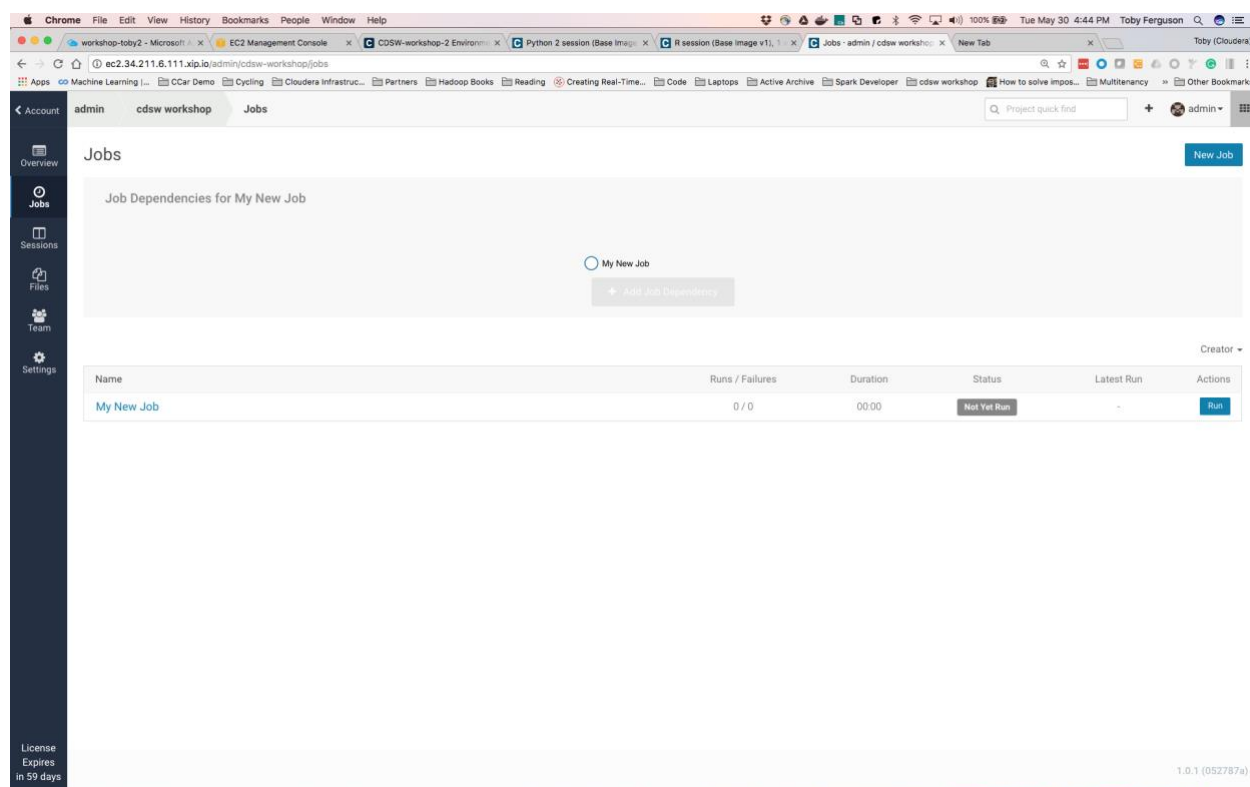
- Name:** A text input field.
- Script:** A text input field.
- Engine Kernel:** A dropdown menu with options: Python 2 (selected), Python 3, Scala, and R.
- Schedule:** A dropdown menu with the option: Manual.
- Engine Profile:** A dropdown menu with the option: 1 vCPU (burstable), 2 GiB memory.
- Timeout In Minutes (optional):** A text input field with the value 30.
- Kill on Timeout:** A checkbox.

The 'Job Report Recipients' section shows a list of recipients with checkboxes for Success, Failure, Stopped, and Timeout. The first recipient is Toby Ferguson. There is also an 'Add External Email' field with an 'Add' button.

4. Create a job that will be triggered manually and will execute the 1_python.py. Here are the parameters to do that:

Name	My New Job
Script	1_python.py
Engine Kernel	Python 2

5. Leave everything else as **default**. Scroll down and hit 'Create Job'. You should get to this screen:



- Here you can see that you have a job ('My New Job'). It's never been run, and it has no dependencies.
- Let's make other jobs depend on this one: Click the '+ Add Job Dependency' grayed out button and add a new job that has a dependency on 'My New Job'. The parameters are:


Name	Job 2
Script	2_pyspark.py
Engine Kernel	Python 2

Create a Job

General

Name

Script

2_pyspark.py 

Engine Kernel


☒ Python 2


☐ Python 3

☐ Scala


☐ R

Schedule

Dependent 

My New Job 

Engine Profile

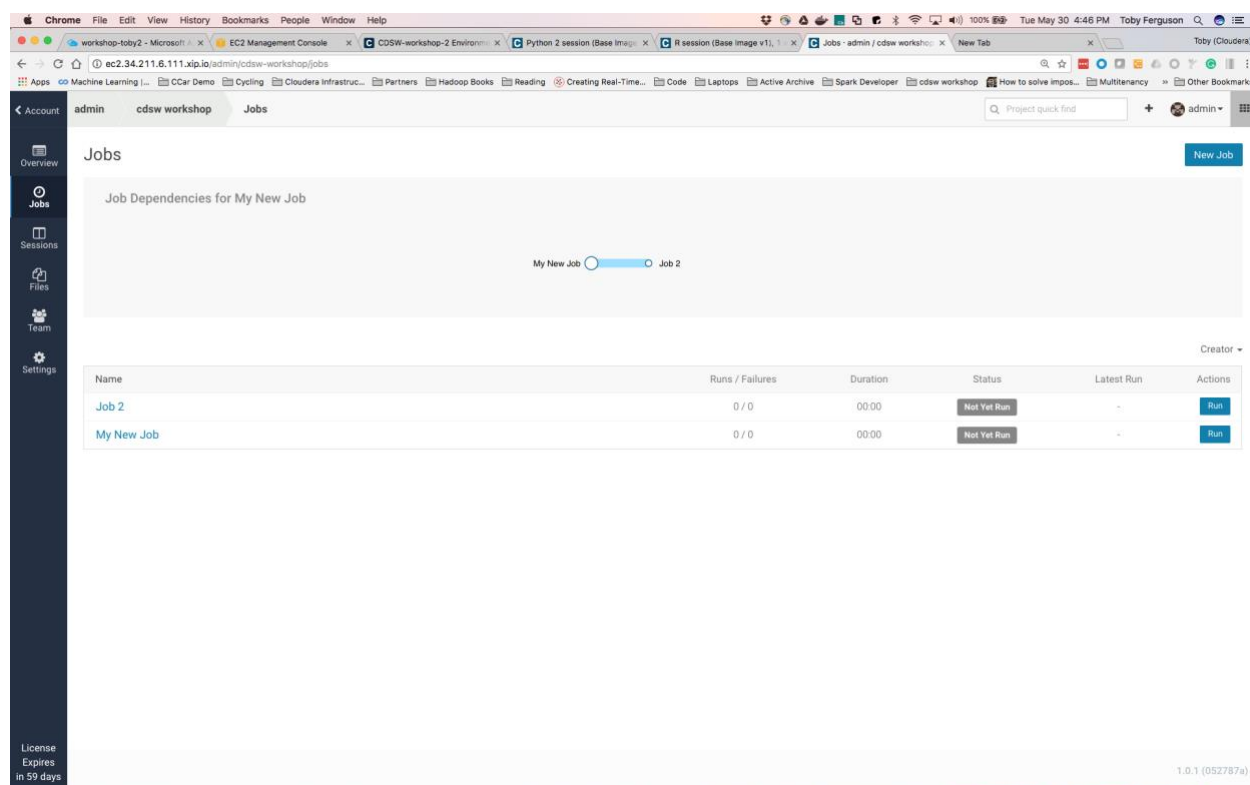
1 vCPU / 2 GiB Memory 

Timeout In Minutes (optional) ☐ Kill on Timeout

Jobs exceeding timeout send warning email if notifications enabled.

[Set Environmental Variables](#)

8. Scroll down and 'Create Job'. You'll now see a page like this:




9. So here we can see that 'Job 2' depends upon 'My New Job' (although you can run each manually, if you so choose).
10. Lets add another job that will run in parallel with Job2:
11. Click 'New Job' in the top right corner and create another job that depends upon 'My New Job'. The parameters you'll need are:

Name	R Job
Script	4_sparklyr.R
Engine Kernel	R
Schedule	Dependent / My New Job

General

Name

Script

Engine Kernel

☐ Python 2

☐ Python 3

☐ Scala

☒ R

Schedule

Dependent

My New Job

Engine Profile

1 vCPU / 2 GiB Memory

Timeout In Minutes (optional) ☐ Kill on Timeout

Jobs exceeding timeout send warning email if notifications enabled.

[Set Environmental Variables](#)

12. Create the job and you'll see this:

The screenshot shows the Cloudera Jobs page. At the top, there's a 'Jobs' section with a 'New Job' button. Below it, a diagram shows 'My New Job' as a dependency for 'Job 2' and 'R job'. A table lists the jobs:

Name	Runs / Failures	Duration	Status	Latest Run	Actions
R job	0 / 0	00:00	Not Yet Run	-	Run
Job 2	0 / 0	00:00	Not Yet Run	-	Run
My New Job	0 / 0	00:00	Not Yet Run	-	Run

13. Lets run it all - hit the 'Run' button next to 'My New Job' (bottom of the list of jobs). You should see the job get scheduled, run, complete, and then the next two jobs should likewise get scheduled, run and complete:

The screenshot shows the Cloudera Jobs page after running the jobs. The table now shows the following status:

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Job 2	1 / 0	00:29	Success	just now	Run
R job	1 / 0	01:19	Success	just now	Run
My New Job	1 / 0	00:03	Success	just now	Run

Question: How will a job scheduler reduce the effort required for you to build simple pipelines?

Question: What other facilities surrounding a job did we not explain? What do you think those other parameters might do?

Lab 11 – Face recognition with Python

In this lab, you will work with images in Python. Using a predefined library, you will try to locate faces on a photo. And once a face is found, you will try to match a specific face on the photo. The instructions for this lab are less detailed, so you might need to browse back in this document (or use google) to find the exact syntax for some statements.

The instructions for this lab will be using a new library, `face_recognition`.

Step 1:

Create a new project in CDSW, and start a Python workbench with 16GB of RAM

Step 2:

Install the `face_recognition` library using the “pip install” command.

As soon as the library is installed, stop your CDSW session. As soon as it is stopped, open a new CDSW session with 4 GB of memory, to allow other students resources to install the library.

Also, install the `opencv-python` library, to use some graphic processing capabilities.

Step 3:

Open a command prompt from your CDSW workbench, and copy all .jpg files from the hdfs directory `/tmp/photos` to the home directory of your CDSW session.

To copy files, you can use the command:

```
hdfs dfs -get
```

Step 4:

Try to display a photo using the `Image` command. Before you can use this command, you need to import some display libraries, with the statement:

```
from IPython.display import Image, display
```

As soon as the library is imported, try to display a photo. Use the command:

```
display(Image("<filename>"))
```

In the directory with .jpg files, search for the image with your name. This should be your photo as was found on LinkedIn.

Step 4:

Detect the exact location of the face on the image, using the `face_recognition` library.

First, load the library:

```
import face_recognition
```

Now you can use the following statement to find where the face is located on the photo:

```
my_photo = face_recognition.load_image_file("<filename>")
my_face_locations = face_recognition.face_locations(my_photo)
```

Check if your algorithm has detected a face, by showing the value.

```
my_face_locations
```

You will see a list of tuples, with the locations of where a face can be found on the photo. Most probably, you will only see 1 tuple, with 1 face on the photo. The output should be something like this:

```
[(32L, 107L, 94L, 45L)]
```

This is the representation of 1 tuple, with the values [(top, right, bottom, left)], which represent the pixel in the top right corner, as well as the bottom left corner. The face on the photo is located in the square between these 2 corners.

Step 5:

Cut the face out of the picture. To do this, use the corners found in step 4. First, extract the corners from the array of tuples. You can do this with the following command:

```
top, right, bottom, left = my_face_locations[0]
```

Now we can extract the face out of the photo, using the following statement:

```
from PIL import Image, ImageDraw
my_face=my_photo[top:bottom, left:right]
my_face_img=Image.fromarray(my_face)
```

And now visualise the image, to check if it worked.

We use some libraries from PIL for that, so we need to import that first.

```
display(my_face_img)
```

You should see the face only now.

Step 6:

Use the face_recognition library to encode the face to a 'match_code'.

```
my_face_encoding = face_recognition.face_encodings(my_photo)[0]
```

The face encoding is an array of numbers that represent the face from the picture. This array is used for matching to find a face that is similarly looking.

Step 7:

Encode the faces on group photo.

First, show the group photo with the statement:

```
from IPython.display import Image, display
display(Image(filename="teamphoto.jpg"))
```

Now, find the face locations on the group photo:

```
group_photo = face_recognition.load_image_file("teamphoto.jpg")
```

Step 8:

Write a loop that makes an encoding of each face in the photo, and that matches the face on the group photo with the encoding created in step 6.

A piece of example code is here:

```
group_photo = face_recognition.load_image_file("teamphoto.jpg")
group_face_locations = face_recognition.face_locations(group_photo)
print len(group_face_locations)
for face_location in group_face_locations:
    top, right, bottom, left = face_location
    print top, right, bottom, left
```

Your hints are:

- Find the number of faces in the group photo. Use the statement from Step 4.
- Create a loop to an encoding for each face. Encoding is done using the statement in Step 6.
- Match the encoding of my_photo and the face of the group_photo.
- `face_recognition.compare_faces([my_face_encoding], group_face_encoding)`
- If matching is True, then draw a box around the face on the group photo.
- Draw a box statement:
- `import cv2`
- `cv2.rectangle(<image>, (left, top), (right, bottom), (0,0,255), 2)`

Troubleshooting

DNS Issue

If you see something like this:

Python 2 session (Base Image v1), 1 vCPU (burstable), 2 GiB memory, on 6/2 at 13:38   Collapse  Share Running
By [Toby Ferguson](#) — Python 2 Session — just now

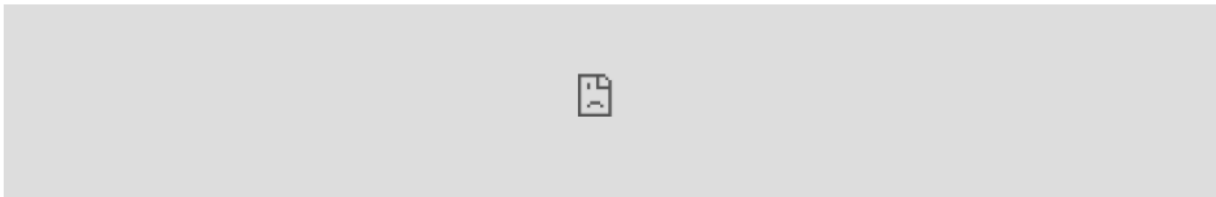
Getting Started

This is your **Python 2 session**. Your **editor** is on the left and your **input prompt** is on the bottom.

To install a package type: `!pip install [package_name]` at the input prompt.

To execute code from the editor, select the code and execute it with `Command-Enter` on Mac or `Ctrl-Enter` on Windows. You can also enter code at the prompt below.

Use `?command` to get help on a particular command.



Then just refresh your browser.

On Windows you might have to flush your dns cache. To do that open up a command prompt and then execute

```
ipconfig /flushdns
```

The root cause of this is that the DNS database on our DNS provider (xip.io) are being actively updated but that takes time (a few seconds) and you're requesting a DNS record that hasn't yet been added. So requesting again after a little while is a sensible thing to do ... and might need to be repeated, depending on how quickly the records are updated.

Spark R backend might have failed

```
✖ Error in invoke_method.spark_shell_connection(sc, TRUE, class, method, :  
  No status is returned. Spark R backend might have failed.
```

Or:

```
✖ Error in invoke_method.spark_shell_connection(sc, TRUE, class, method, :  
  No status is returned. Spark R backend might have failed.  
⚠ Engine exhausted available memory, consider a larger engine size.
```

Then you likely chose a 2G engine - stop that session and start another one, this time with a 4G session

At times we are seeing connections problems with the dynamic DNS configuration we are using specifically for this lab setup. If you experience ongoing connection problems, try stopping your sessions and restarting. That appears to re-establish connections through the DNS.

Appendix

Cloudera Documentation

<http://www.cloudera.com/documentation.html>

Cloudera Data Science Workbench

<https://www.cloudera.com/products/data-science-and-engineering/data-science-workbench.html>

CDSW User Guide

https://www.cloudera.com/documentation/data-science-workbench/latest/topics/cdsw_user_guide.html

Troubleshooting Guide

https://www.cloudera.com/documentation/data-science-workbench/latest/topics/cdsw_troubleshooting.html

Recordings

Part 1 - Introduction

<https://www.cloudera.com/content/dam/www/marketing/resources/webinars/introducing-cloudera-data-science-workbench-part1-recorded-webinar.png.landing.html>

Part 2 – A Visual Dive into machine Learning

<https://www.cloudera.com/content/dam/www/marketing/resources/webinars/part-2-visual-dive-into-machine-learning-and-deep-learning.png.landing.html>

Part 3 – Data Science Models into production from beginner to end

<https://www.cloudera.com/content/dam/www/marketing/resources/webinars/models-in-production-a-look-from-beginning-to-end-part3.png.landing.html>