

**Document Title:** Cloudera Data Science lab guide

**Date:** October 2018

**Version:** 2.4

**Authors:** Colm Moynihan, Toby Ferguson, Andre Molenaar

## Cloudera Data Science Workbench Labs

### Table of Contents

<b>Introduction</b>	<b>2</b>
Lab 1 – Self registration onto CDSW on AWS	3
Lab 2 – Creating a new project	5
Lab 3 - Visualization and Sharing	17
Lab 4 - Hadoop Integration	21
Lab 5 - Pushing the Boundaries	23
Lab 6 - SparklyR	26
Lab 6 - Shiny	28
Lab 8 - Scala	31
Lab 9 - Project Creation using Local Files	36
Lab 10 - Scheduling Jobs	38
Lab 11 – Experiments	45
Lab 12 – Working with Models	52
Lab 13 – Face recognition with Python	63
Lab 14 – Optional: Install CDSW in your Cluster	66
<b>Troubleshooting</b>	<b>77</b>
DNS Issue	77
Spark R backend might have failed	77
<b>Appendix</b>	<b>78</b>
Cloudera Documentation	78
CDSW User Guide	78

## 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

[Forgot Password?](#)

[Sign Up for a New Account](#)

Enter in your details.

- Your Full Name = (example: Andre Molenaar)
- Username is firstname initial + surname = (example: amolenaar)
- 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-w-cloudera.eu/cmoynihan>

colm@cloudera.com

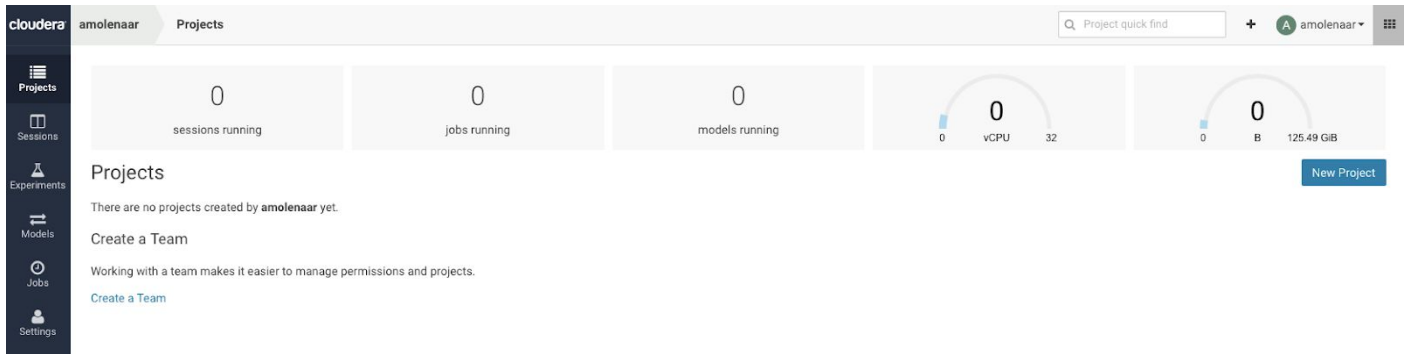
.....

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

**Experiments** - batch experiments

**Models** - build, deploy, and manage models as REST APIs to serve predictions

**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

<https://github.com/andremolenaar/CDSW-Demo-Short>

Project\_name = your user name and labs

You should see this

## Create a New Project

### Project Name

Andre\_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/andremolenaar/CDSW-Demo-Short>

Create Project

The screenshot shows the Cloudera Data Science Workbench (CDSW) interface. On the left is a dark sidebar with navigation icons for Account, Overview, Sessions, Experiments, Models, Jobs, Files, Team, and Settings. The main content area is titled 'CDSW Demo Test' and contains three sections: 'Models' (with a message that no models exist), 'Jobs' (with a message that no jobs exist), and 'Files'. The 'Files' section displays a list of files and folders:

<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"/>	5_shiny.R
<input type="checkbox"/>	README.md
<input type="checkbox"/>	server.R
<input type="checkbox"/>	spark-defaults.conf
<input type="checkbox"/>	ui.R
<input type="checkbox"/>	utils.py

On the left hand side panel you will see a new menu items, among them Team where you can add team members to your project. Ask your neighbor for his or her username, and start typing in the search box.

As an example

The screenshot shows the 'Add' user interface. A search box contains the text 'fi' and an 'Add' button. Below the search box is a list of users:

- Filippo (flambiente)
- Sofie (Gundersen)
- Sofia Thorén (sofiathoren)

To the right of the user list is a table with a 'Permission' column:

	Permission
	Admin

You can add anybody to your project. If you cannot find your neighbor, you can use the 'admin' user to share your project with.

# Cloudera and Deloitte Tech Summit - Data Science Labs

## Collaborators

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

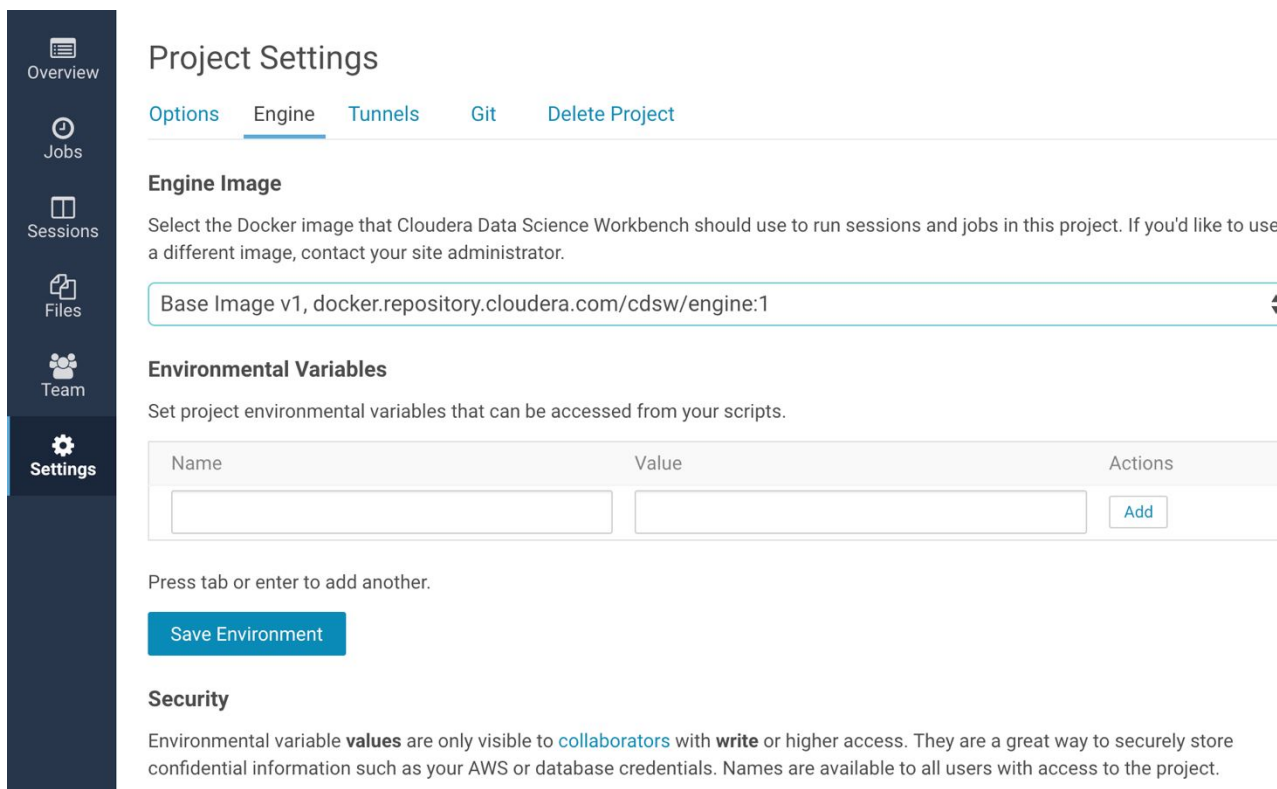
### Add Collaborator

Search by name, username, or email...

Collaborator	Permission
 amolenaar	Admin
 admin	Viewer <input type="button" value="change"/> <input type="button" value="delete"/>

Granting write or admin permission to other users may have security impact since it gives them full access to your project files and running sessions. Write or admin permission should only be granted to trusted users.

Click on the Settings icon and then the Engine tab:



The screenshot shows the Cloudera Project Settings interface. On the left is a dark sidebar with navigation icons for Overview, Jobs, Sessions, Files, Team, and Settings (which is highlighted). The main content area is titled 'Project Settings' and has tabs for Options, Engine (selected), 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 for Name, Value, and Actions. The table currently has one empty row and an 'Add' button. 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.

Under Project Settings you will see:

Options - Project Name and Description, Private or Public

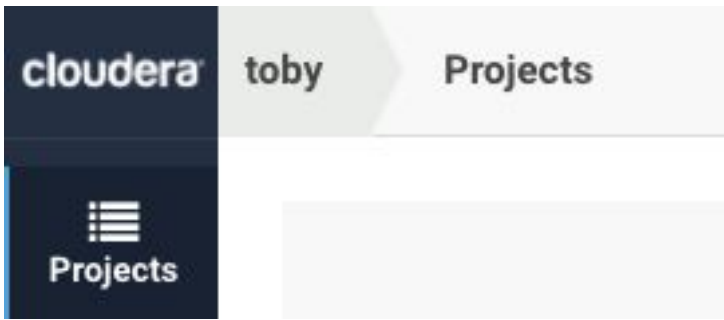
Engine - Engine image and environment variables

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

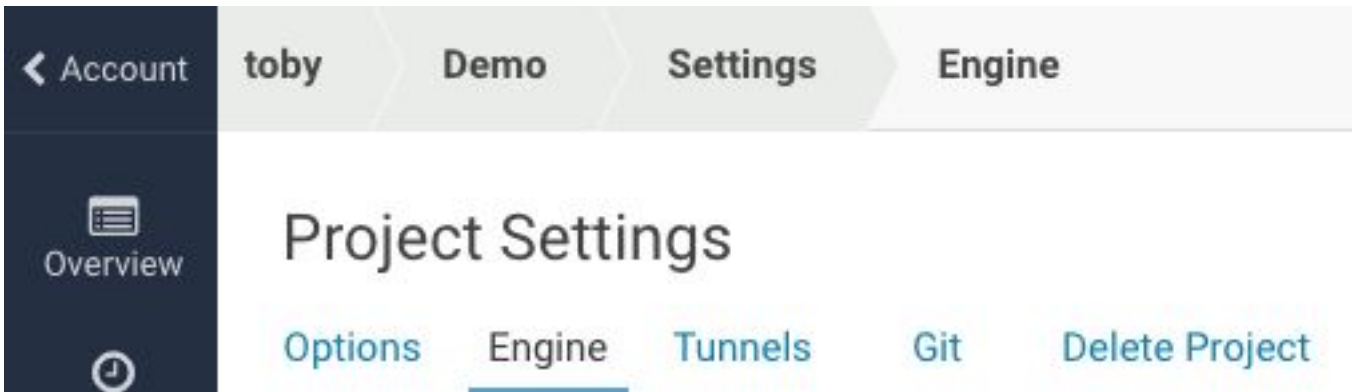
And this is where if you have to (please don't!) **Delete Project**

**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:

The screenshot shows the Cloudera Data Science Workbench interface for a project named 'Andre\_CDSW\_Demos'. The user 'amolenaar' is logged in. The interface includes a sidebar with navigation options: Overview, Sessions, Experiments, Models, Jobs, Files, Team, and Settings. The main content area is divided into sections: 'Models' (no models yet), 'Jobs' (no jobs yet), and 'Files'. The 'Files' section contains a table of files with columns for Name, Size, and Last Modified. Below the files table, there is a section titled 'Cloudera Data Science Workbench demos' with a 'Workbench' subsection. A red circle highlights the 'Open Workbench' button in the top right corner of the interface.

Name ^v	Size	Last Modified
data	-	6 minutes ago
1_python.py	3.01 kB	6 minutes ago
2_pyspark.py	1.60 kB	6 minutes ago
3_tensorflow.py	3.39 kB	6 minutes ago
4_sparklyr.R	2.50 kB	6 minutes ago
5_shiny.R	769 B	6 minutes ago
README.md	1.74 kB	6 minutes ago
server.R	536 B	6 minutes ago
spark-defaults.conf	74 B	6 minutes ago
ui.R	689 B	6 minutes ago
utils.py	1.09 kB	6 minutes ago


**Cloudera Data Science Workbench demos**  
Basic tour of Cloudera Data Science Workbench.

**Workbench**  
There are 4 scripts provided which walk through the interactive capabilities of Cloudera Data Science Workbench.

- 1 **Basic Python visualizations (Python 2)**. Demonstrates:

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:

The screenshot displays the Cloudera Data Science Workbench interface. On the left is a file browser with a 'refresh' icon at the top, showing a directory structure including 'data', 'README.md', 'server.R', 'spark-defaults.conf', 'ui.R', and 'utils.py'. The central pane is a code editor showing the content of 'README.md', which includes sections for 'Workbench', 'Jobs', and 'Setup instructions'. The right pane is a 'Start New Session' configuration panel with a yellow warning box, 'Engine Image - Configure' section, 'Select Engine Kernel' options (Python 2, Python 3, Scala, R), and 'Select Engine Profile' dropdown set to '1 vCPU / 2 GiB Memory'. At the bottom of the right pane are 'Launch Session' and 'Run Experiment...' buttons.

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](#).

### Engine Image - [Configure](#)

Base Image v5 - [docker.repository.cloudera.com/cdsw/engine:5](https://docker.repository.cloudera.com/cdsw/engine:5)

### Select Engine Kernel

- Python 2
- Python 3
- Scala
- R

### Select Engine Profile

1 vCPU / 2 GiB Memory



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.

The screenshot shows the Cloudera Data Science Workbench interface. On the left is a file explorer with a tree view containing files like 1\_python.py, 2\_pyspark.py, 3\_tensorflow.py, 4\_sparklyr.R, 5\_shiny.R, data, README.md, server.R, spark-defaults.conf, ui.R, and utils.py. The main editor displays the README.md file with the following content:

```
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
6
7 1. **Basic Python visualizations (Python 2).** Demonstrates:
8 - Markdown via comments
9 - Jupyter-compatible visualizations
10 - Simple console sharing
11
12 2. **PySpark (Python 2).** Demonstrates:
13 - Easy connectivity to (kerberized) Spark in YARN client mode.
14 - Access to Hadoop HDFS CLI (e.g. `hdfs dfs -ls /`).
15
16 3. **TensorFlow (Python 2).** Demonstrates:
17 - Ability to install and use custom packages (e.g. `pip search tensorflow`)
18
19 4. **R on Spark via Sparklyr (R).** Demonstrates:
20 - Use familiar dplyr with Spark using `sparklyr` (http://spark.rstudio.com)
21
22 5. **Advanced visualization with Shiny (R).** Demonstrates:
23 - Use of 'shiny' to provide interactive graphics inside CDSW
24
25 ## Jobs
26 We recommend setting up a "**Nightly Analysis**" job to illustrate how data
27
28 ## Setup instructions
29 Note: You only need to do this once.
30
31 1. In a Python 2 Session:
32 ``` Python
33 !pip2 install --upgrade dask
34 !pip2 install --upgrade keras
35 !pip2 install --upgrade matplotlib==2.0.0.
36 !pip2 install --upgrade pandas_highcharts
37 !pip2 install --upgrade protobuf
38 !pip2 install --upgrade tensorflow==1.3.0.
39 !pip2 install --upgrade seaborn
40 ```
41 Note, you must then stop the session and start a new Python session in order
42
43 2. In an R Session:
44 ``` R
45 install.packages('sparklyr')
46 install.packages('plotly')
47 install.packages('nycflights13')
48 install.packages('Lahman')
49 install.packages('mgcv')
50 install.packages('shiny')
51 ```
52
53 3. Stop all sessions, then proceed.
```

On the right, an 'Untitled Session' terminal window is open, showing the text: 'By test - Python 2 Session - 1 vCPU / 2 GiB Memory - just now'. Below this, the 'Getting Started' section is visible, explaining that the left side is the editor and the right side is the input prompt, and providing instructions on how to install packages and execute code.

8. Look at the line 30 to 36 of README.md:

```
!pip2 install --upgrade dask
!pip2 install --upgrade keras
!pip2 install --upgrade matplotlib==2.0.0.
!pip2 install --upgrade pandas_highcharts
!pip2 install --upgrade protobuf
!pip2 install --upgrade tensorflow==1.3.0.
!pip2 install --upgrade seaborn
```

9. 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

10. Make a block selection of these lines, and select 'Run Line(s)':

```

28 1. In a Python 2 Session:
29   ...Python
30 !pip2 install --upgrade dask
31 !pip2 install --upgrade keras
32 !pip2 install --upgrade matplotlib==2.0.0.
33 !pip2 install --upgrade pandas_highcharts
34 !pip2 install --upgrade protobuf
35 !pip2 install --upgrade tensorflow==1.3.0.
36 !pip2 install --upgrade seaborn
37   ...
38 Note, you must then stop the session and start
39
40 2. In an R Session:
41   ...R

```

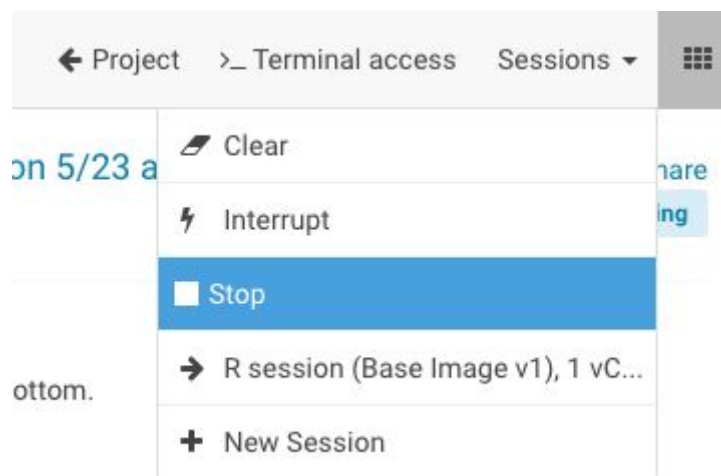
Run Line(s)	⌘Enter
Select All	⌘A

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

12. 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.

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

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



ter on Mac or **Ctrl-Enter** on Windows. You can also

15. 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](#).

### Engine Image - Configure

Base Image v5 - [docker.repository.cloudera.com/cdsw/engine:5](https://docker.repository.cloudera.com/cdsw/engine:5)

### Select Engine Kernel

- Python 2
- Python 3
- Scala
- R

### Select Engine Profile

2 vCPU / 4 GiB Memory

Launch Session

or

Run Experiment..

16. This time we're going to execute lines 42 through 47 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:

```
40 2. In an R Session:  
41 ```R  
42 install.packages('sparklyr')  
43 install.packages('plotly')  
44 install.packages("nycflights13")  
45 install.packages("Lahman")  
46 install.packages("mgcv")  
47 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!

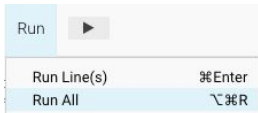


## Lab 3 - Visualization and Sharing

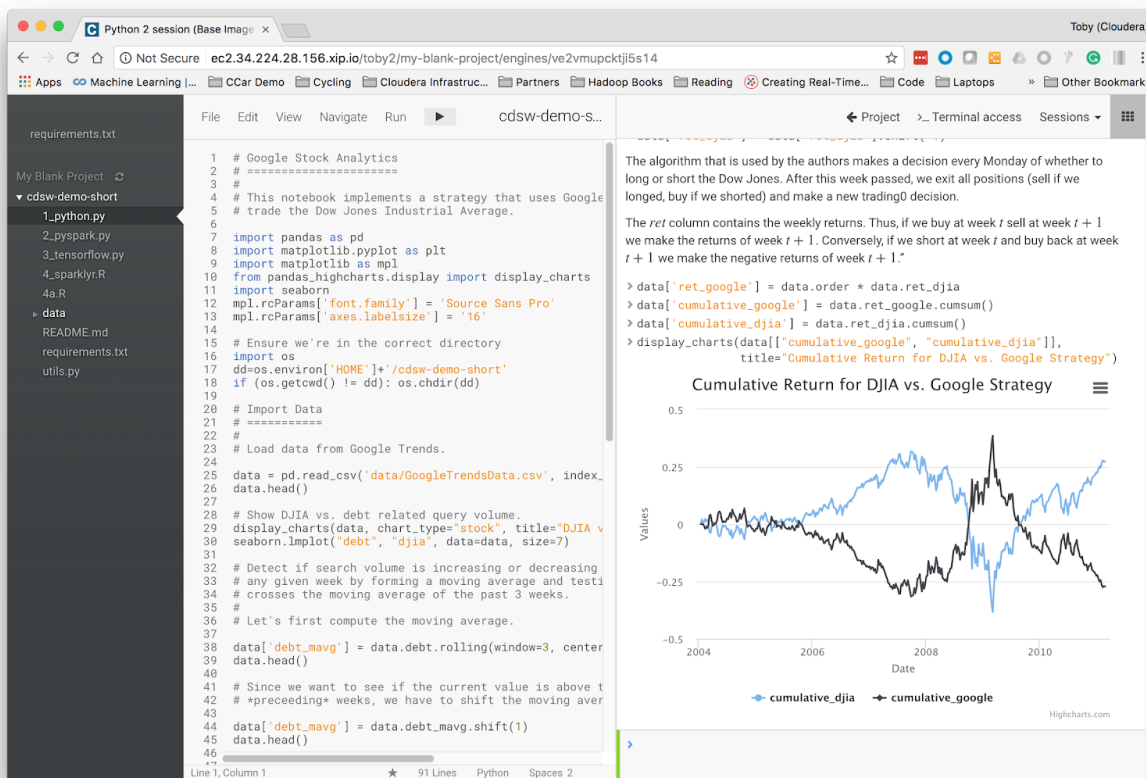
Check that you have NO sessions running - if any sessions are running then stop them now. You've setup your environment and those sessions can be safely disposed of.

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!).



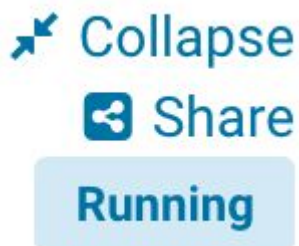
4. You should end up with some nice graphs in the output window:



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 ...



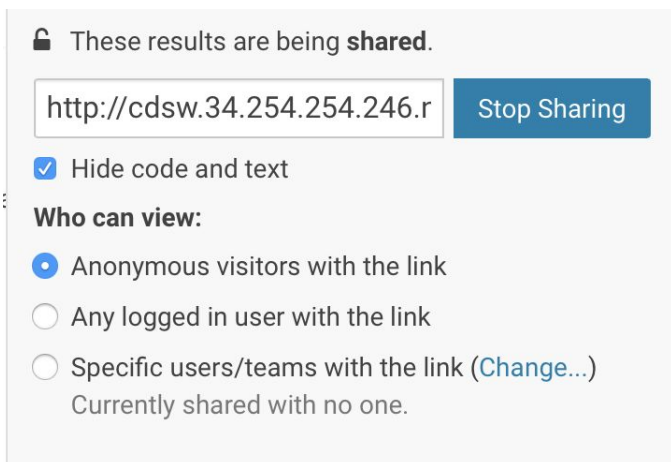
9. Scroll to the top of the window and you'll see this on the far right (the exact layout depends upon the real estate available – you might have to expand your browser window to see the following links and they might be laid out vertically or horizontally):



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 'Share with Others' (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!)

Python 2 session (Base Image v2), 1 vCPU / 2 GiB Memory, on 9/6 at 9:26

Expand

Running

By Colm — Python 2 Session (Base Image v2) — 4 minutes ago for running

## Google Stock Analytics

This notebook implements a strategy that uses Google Trends data to trade the Dow Jones Industrial Average.

Ensure we're in the correct directory

### Import Data

Load data from Google Trends.

	djia	debt
Date		
2004-01-14	10485.18	0.210000
2004-01-22	10528.66	0.210000
2004-01-28	10702.51	0.210000
2004-02-04	10499.18	0.213333
2004-02-11	10579.03	0.200000

Show DJIA vs. debt related query volume.



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!

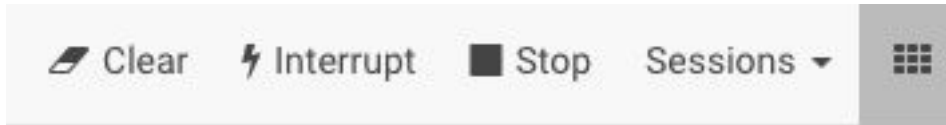
## Lab 4 - Hadoop Integration

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.

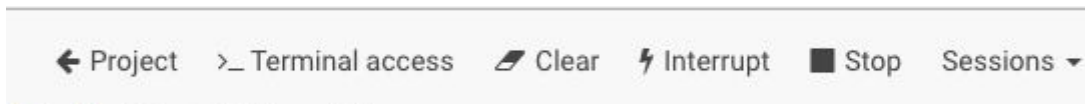
1. Clean up your Python session by hitting the 'clear' button and the 'expand' link (if available)



6 at 10:51 

 Expand  Share Running

2. Select the 2\_pyspark.py file
3. If you don't have a Python 2 workbench session open yet, Launch a Python 2 session 1 vCPU / 2 GB RAM
4. Run line 34:
5. `!hdfs dfs -put -f $HOME/data/kmeans_data.txt /user/cdsw`
6. Execute the 2\_pyspark.py file in your already running Python session
7. **Question:** What did it do?
8. **Question:** What kind of thing is the variable 'data'? (try typing 'data' into the console and seeing what gets printed out.
9. Open a terminal using the 'terminal' icon in the top right:



10. 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!)

```
Welcome to Cloudera Data Science Workbench

Kernel: python2

Project workspace: /home/cdsw

No Kerberos principal or Hadoop username detected

Runtimes:
  R: R version 3.4.1 (--) -- "Single Candle"
  Python 2: Python 2.7.11
  Python 3: Python 3.6.1
  Java: java version "1.8.0_121"

Git origin: https://github.com/andremolenaar/CDSW-Demo-Short

cdsw@6mzzmuu4vvgsvbjy:~$ hdfs dfs -ls
Found 5 items
drwx-----   - cdsw cdsw           0 2018-10-05 13:00 .Trash
drwxr-xr-x    - cdsw cdsw           0 2018-10-12 11:23 .sparkStaging
-rw-r--r--    3 cdsw cdsw          71 2018-10-12 11:22 kmeans_data.txt
drwxr-xr-x    - cdsw cdsw           0 2018-10-12 11:18 models
drwxr-xr-x    - cdsw cdsw           0 2018-09-04 13:52 output
cdsw@6mzzmuu4vvgsvbjy:~$
```

or

```
> !hdfs dfs -ls

Found 5 items
drwx-----   - cdsw cdsw           0 2018-10-05 13:00 .Trash
drwxr-xr-x    - cdsw cdsw           0 2018-10-12 11:23 .sparkStaging
-rw-r--r--    3 cdsw cdsw          71 2018-10-12 11:22 kmeans_data.txt
drwxr-xr-x    - cdsw cdsw           0 2018-10-12 11:18 models
drwxr-xr-x    - cdsw cdsw           0 2018-09-04 13:52 output

>|
```

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 2 through 6 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

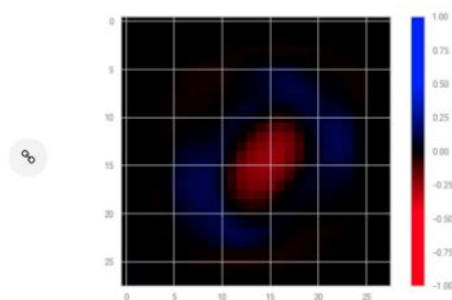
Examine layers

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



At the bottom of the output, there is a link present called 'Open Tensorboard'.

The screenshot shows a Jupyter Notebook interface with a code editor on the left and a TensorBoard visualization on the right. The code in the notebook is as follows:

```
1 |
2 | import tensorflow as tf
3 | import numpy as np
4 | import matplotlib
5 | from matplotlib import pyplot as plt
6 | import utils
7 |
8 | ### Import MNIST data
9 | from tensorflow.examples.tutorials.mnist import input_data
10 | mnist = input_data.read_data_sets('data/MNIST', one_hot=True)
11 |
12 | ### View Data
13 | for i in xrange(0, 3):
14 |     tmp = mnist.train.images[i]
15 |     tmp = tmp.reshape((28,28))
16 |     plt.imshow(tmp, cmap = plt.cm.Greys)
17 |     plt.show()
18 |
19 | ### Parameters
20 | learning_rate = 0.01
21 | training_epochs = 5
22 | batch_size = 100
23 | display_step = 1
24 | logs_path = './tmp/tensorboard'
25 |
26 | ### Cleanup old logs
27 | if tf.gfile.Exists(logs_path):
28 |     tf.gfile.DeleteRecursively(logs_path)
29 |     tf.gfile.MakeDirs(logs_path)
30 |
31 | ### Model
32 | # Use a single-layer perceptron as example $pred = softmax(W x+b)$
33 | x = tf.placeholder('float', [None, 784], name='data')
34 | y = tf.placeholder('float', [None, 10], name='label')
35 |
36 | # Model bias and weight variables: W, b
37 | W = tf.Variable(tf.zeros([784,10]), name='weights')
38 | b = tf.Variable(tf.zeros([10]), name='bias')
39 |
40 | # Put the model ops into scopes for tensorboard
41 | with tf.name_scope('Model'):
42 |     logits = tf.matmul(x,W)+b
43 |     pred = tf.nn.softmax(logits)
44 |     with tf.name_scope('Loss'):
45 |         cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y))
46 |     optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
47 |     with tf.name_scope('evaluation'):
48 |         corr_pred = tf.equal( tf.argmax(pred,1), tf.argmax(y,1) )
49 |         acc = tf.reduce_mean(tf.cast(corr_pred, 'float'))
50 |
51 |     init = tf.global_variables_initializer()
52 |
53 |
54 | ### Summaries
55 | # Create *summary* ops to monitor the cost/accuracy
56 |
57 | loss_summary = tf.summary.scalar('loss', cost)
58 | accu_summary = tf.summary.scalar('accuracy', acc)
59 |
60 | merged_summary_op = tf.summary.merge([loss_summary, accu_summary])
61 |
```

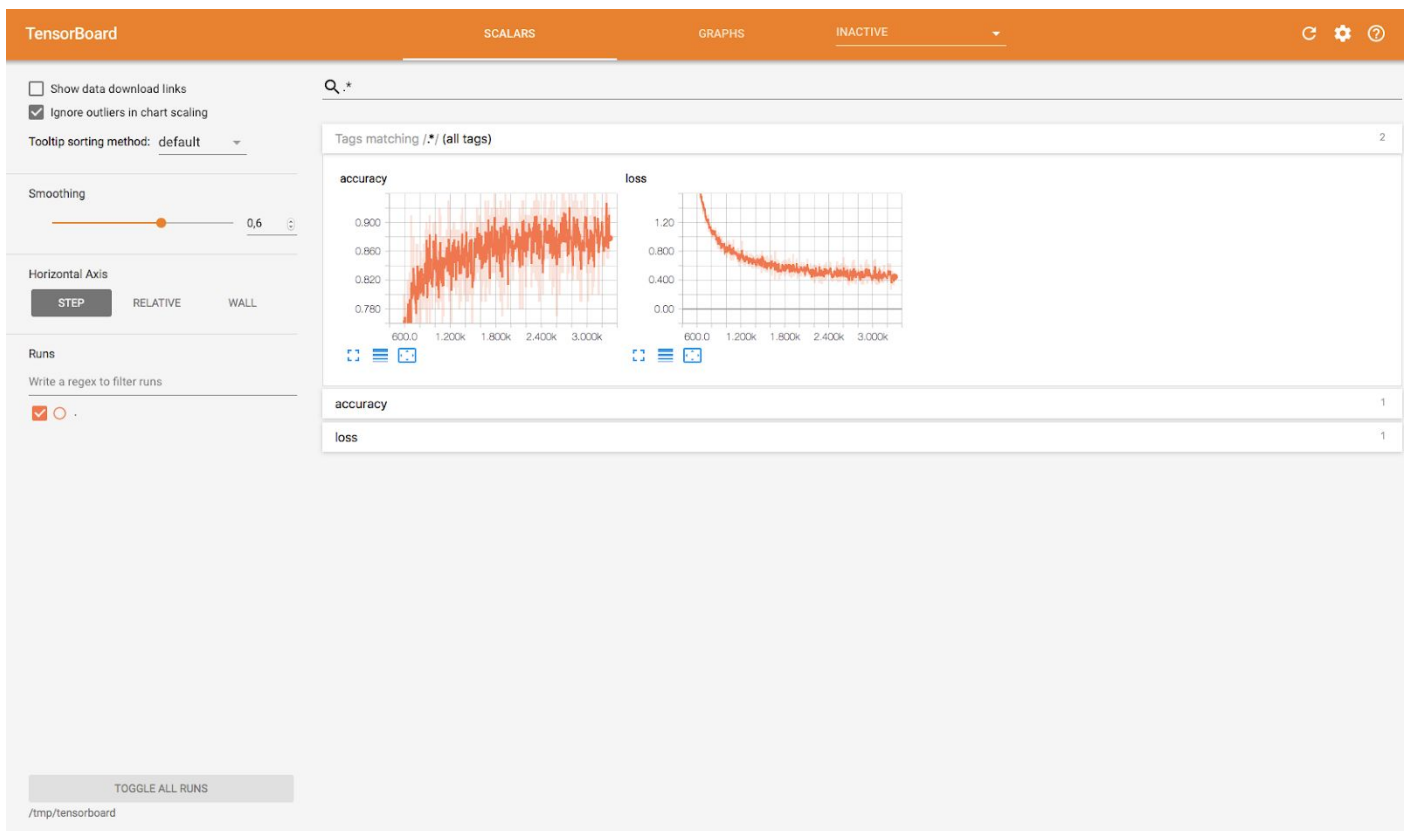
The TensorBoard visualization on the right shows two heatmaps of handwritten digits. The top heatmap is labeled 'Digit 4' and shows a digit '4' on a grid. The bottom heatmap shows another digit. The color scale ranges from -1.00 (dark blue) to 1.00 (dark red).

Below the heatmaps, the TensorBoard interface shows the following text:

```
> utils.start_tensorboard(logs_path, iframe=False)
TensorBoard 0.1.8 at http://6mzmmu4vvgsvbjy:8080 (Press CTRL+C to quit)
Starting TensorBoard at http://6mzmmu4vvgsvbjy.cds.w.52.211.207.216.nip.io...
Open TensorBoard
```

Click on the link. A separate tab will open in your browser, where the Tensorflow Tensorboard is present.





If you want, you can browse a bit through the Tensorboard application. This application is available as long as you keep your workbench session running.

When you are finished with TensorBoard, close the browser tab and open the tab with the Data Science Workbench.

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](#).

#### Engine Image - [Configure](#)

Base Image v5 - [docker.repository.cloudera.com/cdsw/engine:5](https://docker.repository.cloudera.com/cdsw/engine:5)

#### Select Engine Kernel

- Python 2
- Python 3
- Scala
- R

#### Select Engine Profile

2 vCPU / 4 GiB Memory

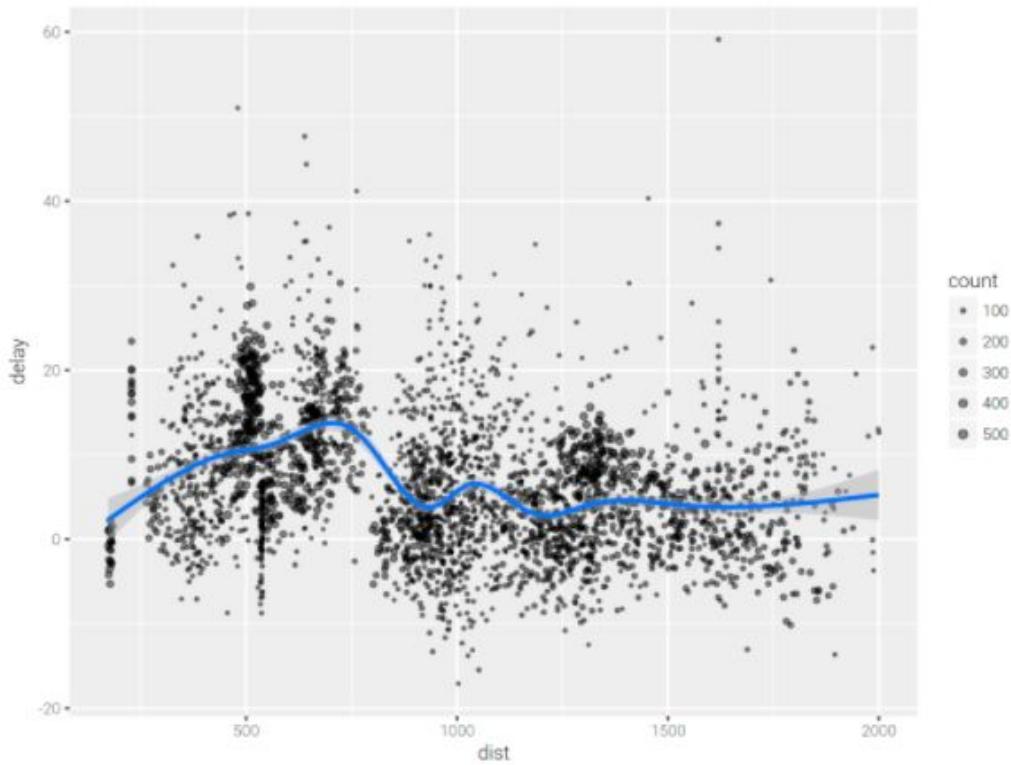
**Launch Session**

**Run Experiment..**

### 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):

```
'geom_smooth()' using method = 'gam'
```



## 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!

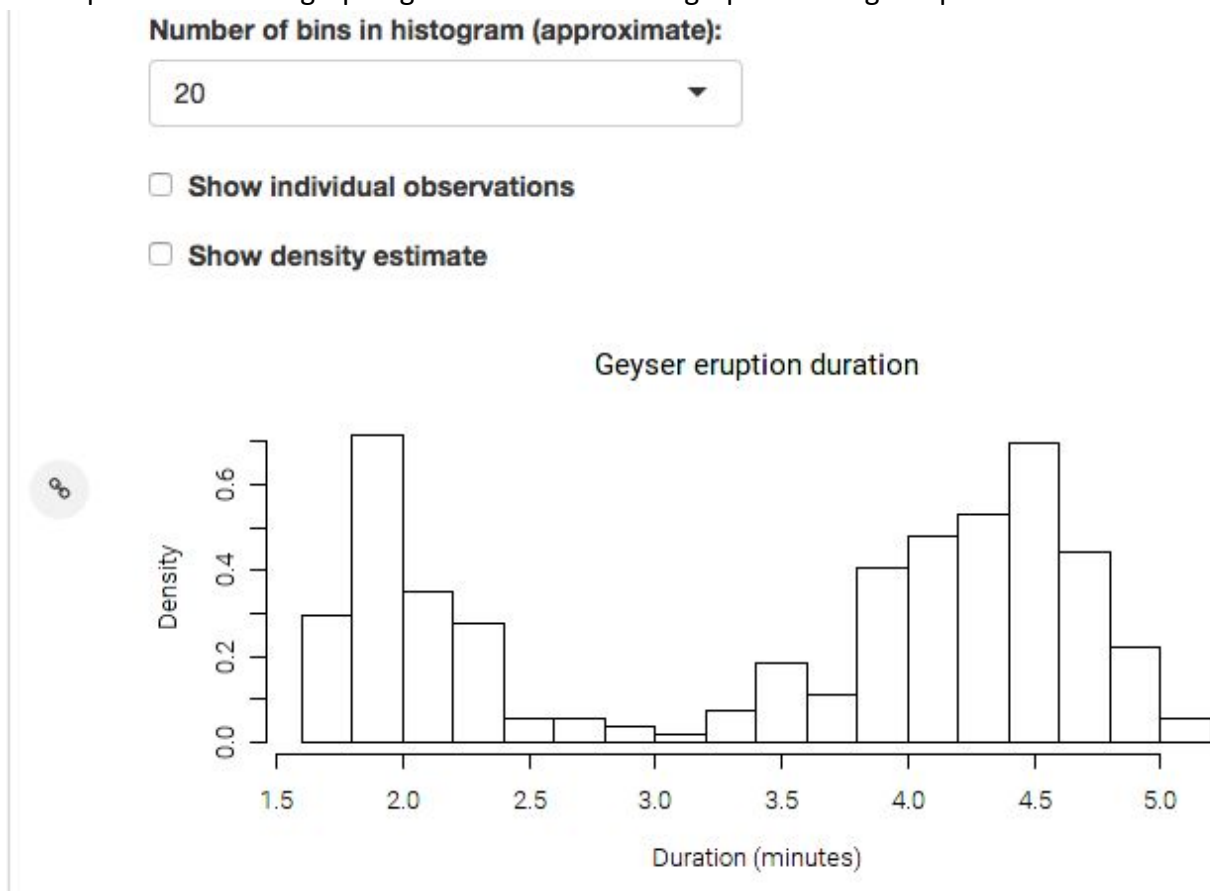
### Lab 6 - Shiny

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!



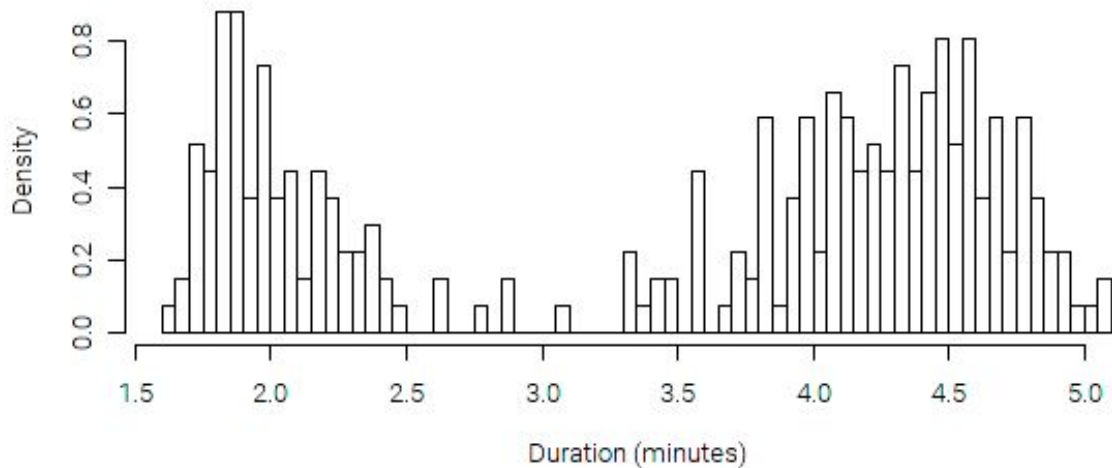
2. You can also change the number of bins in histogram to 50

Number of bins in histogram (approximate):

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?

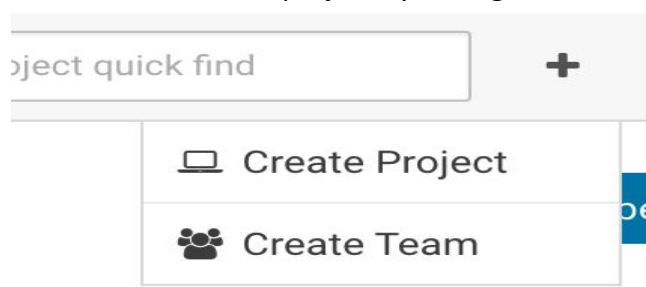
### Lab 8 - Scala

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:



The screenshot shows the Cloudera IDE interface for a project named 'Scala'. The top navigation bar includes 'Account', 'amolenaar', and 'Scala'. A search bar contains 'Project quick find'. On the right, there are buttons for 'Fork' and 'Open Workbench'. The left sidebar contains navigation icons for Overview, Sessions, Experiments, Models, Jobs, Files, Team, and Settings. The main content area is divided into sections: 'Models' (no models yet), 'Jobs' (no jobs yet), and 'Files'. The 'Files' section shows a table of files:

Name ^	Size	Last Modified
data	-	just now
examples	-	just now
auction-analysis.scala	2.12 kB	just now
log4j.properties	21 B	just now
pi.scala	837 B	just now
README.md	2.05 kB	just now
spark-defaults.conf	88 B	just now
wordcount.scala	562 B	just now

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

### Select Engine Kernel

- Python 2
- Python 3
- Scala
- R

### Select Engine Profile

1 vCPU / 2 GiB Memory

**Launch Session** or **Run Experiment..**

7. 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:
8. Open Terminal
9. `hdfs dfs -put -f data /tmp`

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

10. Execute one or more of the various scala files from the examples folder in the Workbench
11. Open example -> kmeans.scala and execute Run All

# Cloudera and Deloitte Tech Summit - Data Science Labs

```
File Edit View Navigate Run Run Line(s) Run All KMeansModel
```

```
1 import org.apache.spark.mllib.clustering.KMeansModel}
2 import org.apache.spark.mllib.clustering.KMeansModel}
3 import org.apache.spark.mllib.clustering.KMeansModel}
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
```

```
> import sys.process._
load local data to hdfs
> "hdfs dfs -put data/kmeans_data.txt /tmp" !
example kmeans clustering script
> val data = sc.textFile("/tmp/kmeans_data.txt")
> val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble))).cache()
Cluster the training data set into two classes with KMeans
> val numClusters = 2
> val numIterations = 20
> val clusters = KMeans.train(parsedData, numClusters, numIterations)
[Stage 0:>
Evaluate clustering by computing Within Set Sum of Squared Errors
> val WSSSE = clusters.computeCost(parsedData)
> println(s"Within Set Sum of Squared Errors = $WSSSE")
Within Set Sum of Squared Errors = 0.1199999999999994547
Save the model
> val output = "output/KMeansExample/KMeansModel"
> FileUtils.deleteQuietly(new File(output))
false
> clusters.save(sc, output)
[Stage 12:==
example of loading and predicting on the model we created
> val sameModel = KMeansModel.load(sc, output)
> sameModel.clusterCenters.zipWithIndex.foreach { case (center, idx) =>
  println(s"Cluster Center ${idx}: ${center}")
}
Cluster Center 0: [9.1,9.1,9.1]
Cluster Center 1: [0.1,0.1,0.1]
> sameModel.predict(Vectors.dense(7,5,6))
0
```

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:1489)
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:

The screenshot shows the Cloudera Data Science Workbench (CDSW) interface for a project named 'My Blank Project'. The interface includes a left-hand navigation menu with options: Account, Overview, Jobs, Sessions, Files, Team, and Settings. The main content area shows the project name 'My Blank Project' with a lock icon, a '0 Fork' button, and an 'Open Workbench' button. Below this, there is a 'Jobs' section with a message: 'This project has no jobs yet. Create a new job to document your analytics pipelines.' The 'Files' section features a table with columns for Name, Size, and Last Modified. The table lists two folders: 'cdsw-demo-short' and 'R', both created yesterday. There are also buttons for 'Download', '+ New', and 'Upload'. At the bottom, a message states: 'This project doesn't contain a README.md file. Consider adding one that describes your project.'

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

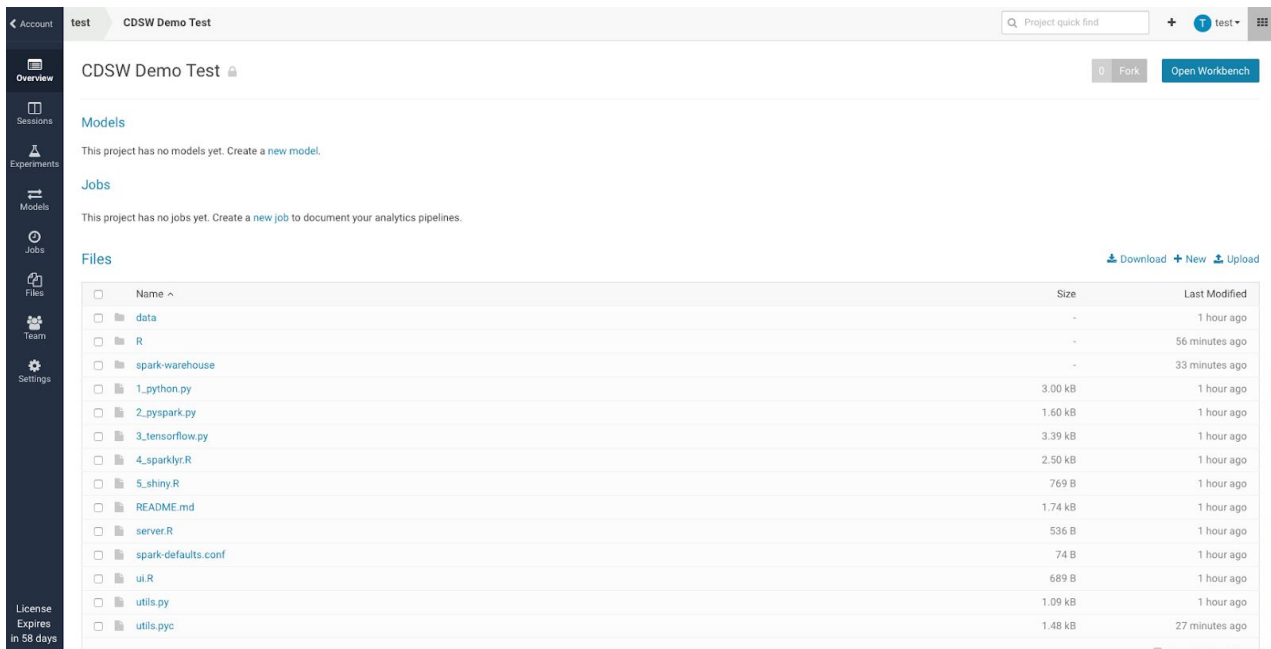
**Question:** With the combination of git, blank projects and uploading from a local file system on your laptop do you feel pretty confident you can get the data and code you want into the CDSW environment?

## Lab 10 - Scheduling Jobs

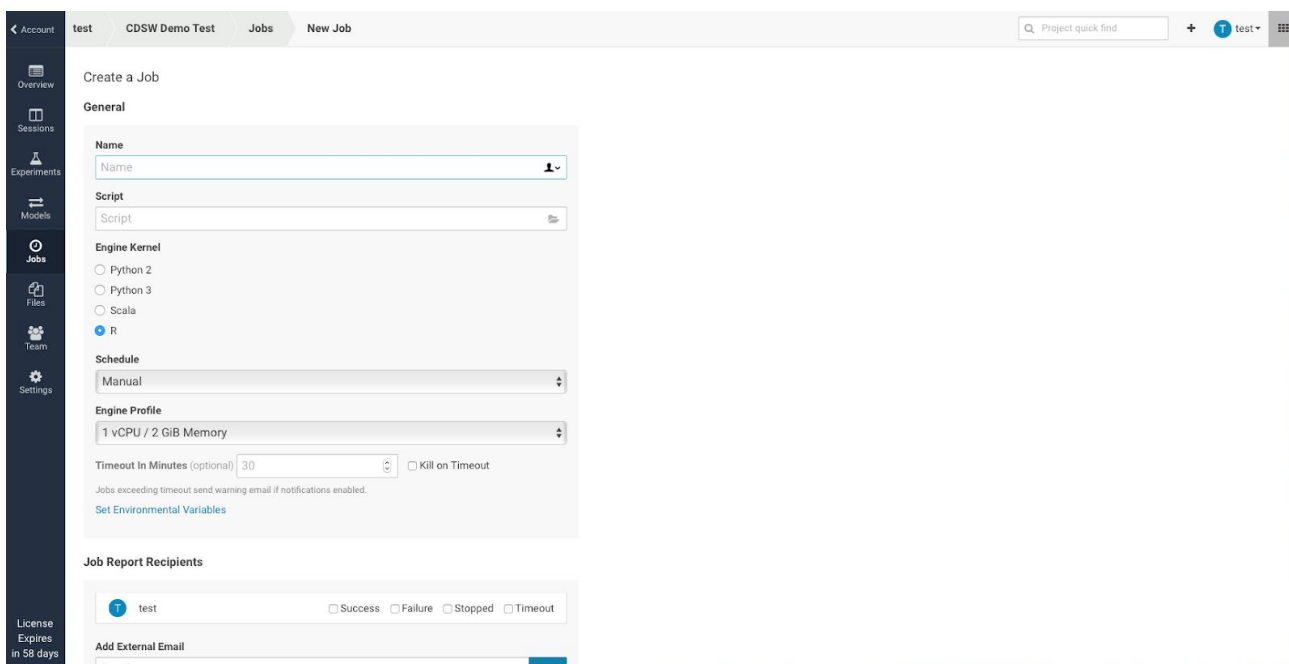
Its often the case that you need to execute tasks on a periodic basis, and to execute one or more tasks once some other task has succeeded. Obviously there are sophisticated workflow engines but for simple workflows CDSW has a jobs system built in.

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:



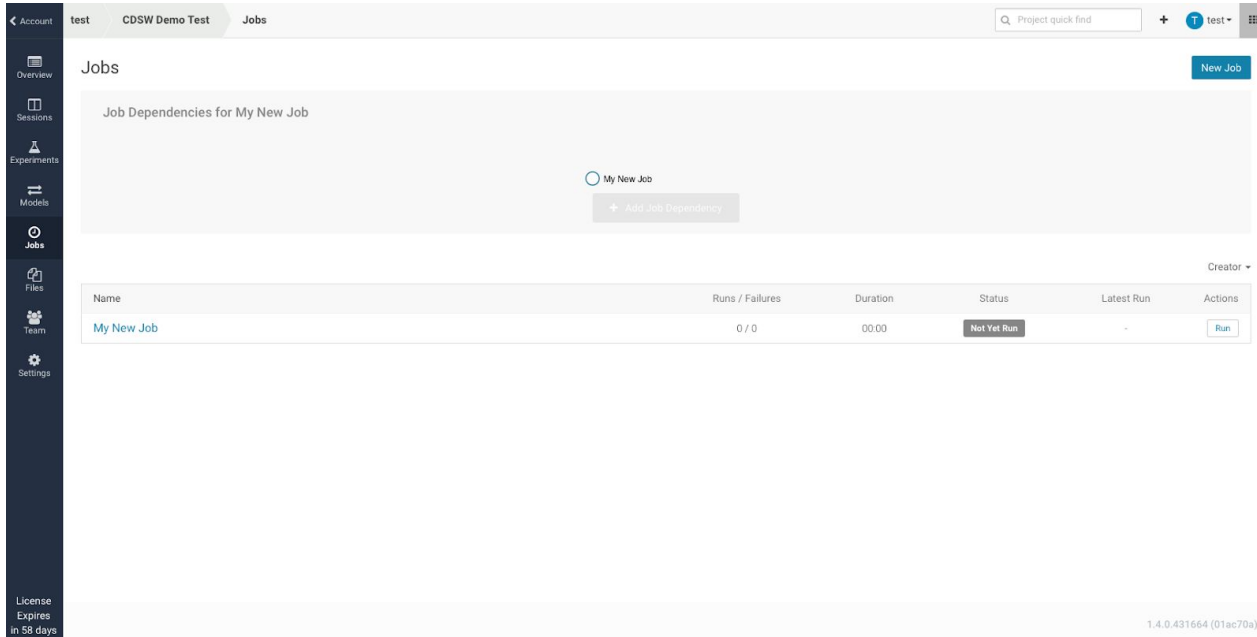
2. You need to be in a project to create a Job
3. Select the 'new job' link in the middle of the page, and you'll get to the following screen (there are other ways of getting to this next screen - its an exercise for the student to figure out what they might be):



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:



- 6. Here you can see that you have a job ('My New Job'). It's never been run, and it has no dependencies.
- 7. 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

The screenshot shows the 'General' configuration page for a Cloudera job. The fields are as follows:

- Name:** Job 2
- Script:** 2\_pyspark.py
- Engine Kernel:** Python 2 (selected), Python 3, Scala, R
- Schedule:** Dependent (dropdown), My New Job (dropdown)
- Engine Profile:** 1 vCPU / 2 GiB Memory (dropdown)
- Timeout In Minutes (optional):** 30 (input field), Kill on Timeout (checkbox)

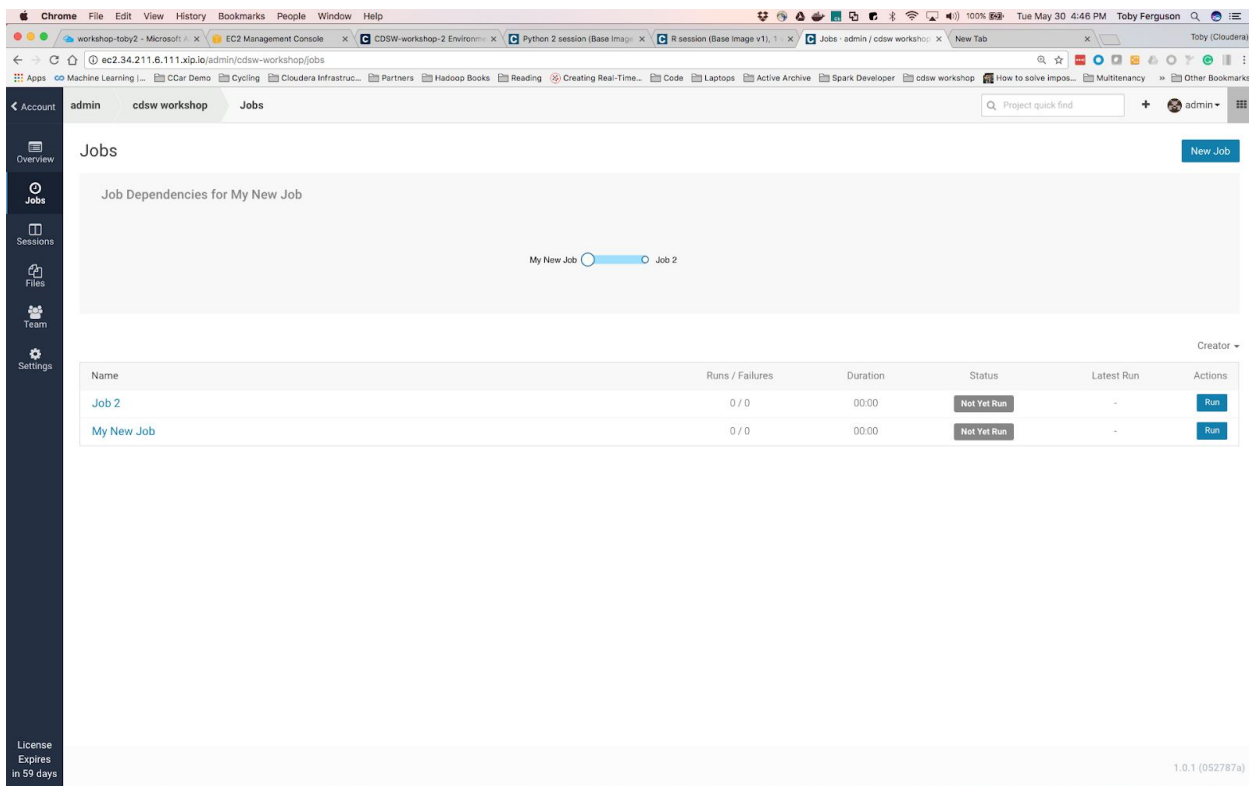
Jobs exceeding timeout send warning email if notifications enabled.

[Set Environmental Variables](#)

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



# Cloudera and Deloitte Tech Summit - Data Science Labs




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**  
R Job

**Script**  
4\_sparklyr.R 

**Engine Kernel**  
 Python 2  
 Python 3  
 Scala  
 R

**Schedule**  
Dependent   
My New Job 

**Engine Profile**  
1 vCPU / 2 GiB Memory 

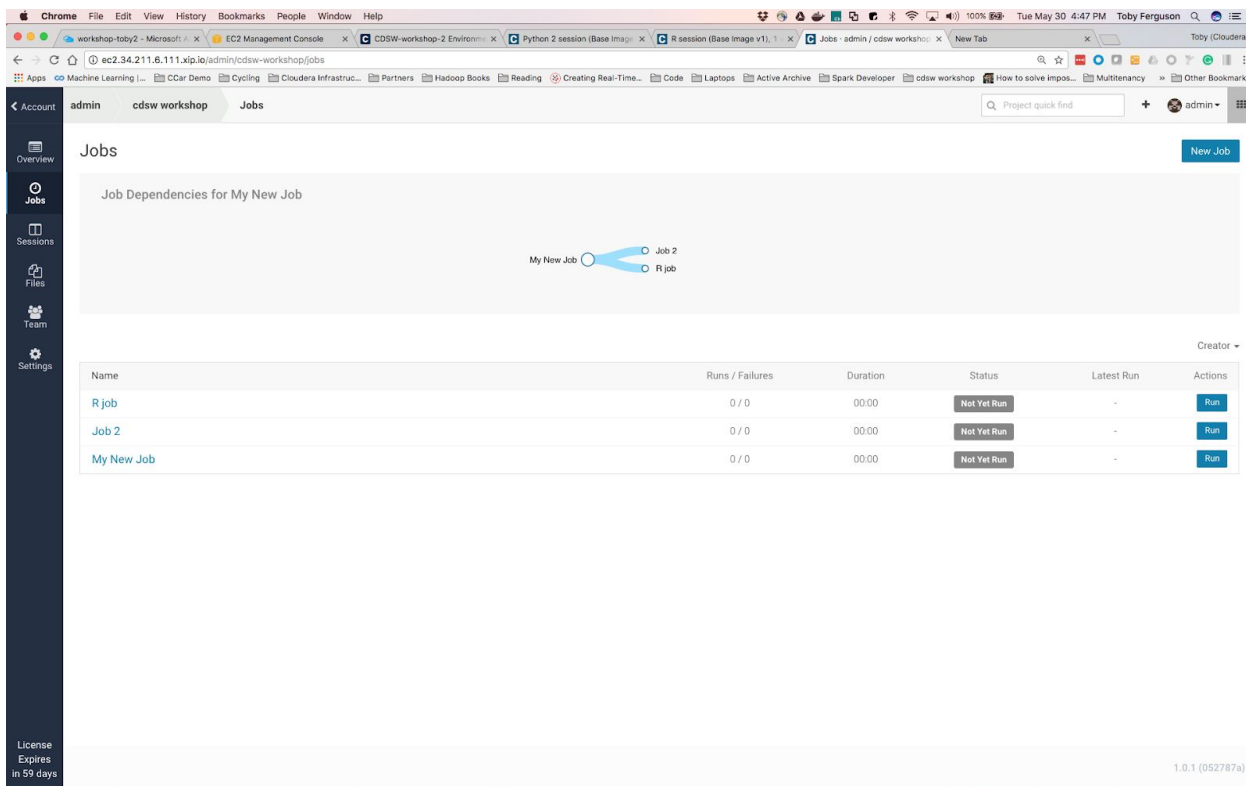
**Timeout In Minutes** (optional)   Kill on Timeout

Jobs exceeding timeout send warning email if notifications enabled.

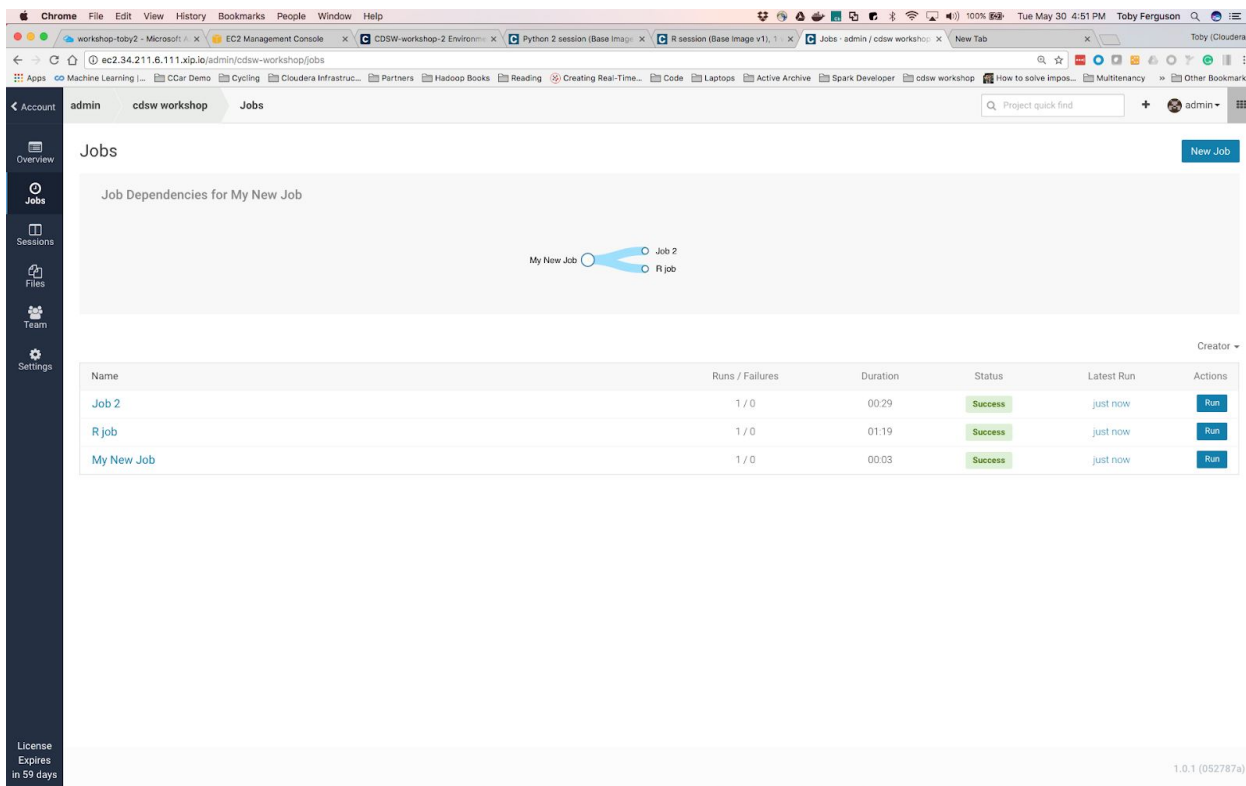
[Set Environmental Variables](#)

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

# Cloudera and Deloitte Tech Summit - Data Science Labs



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:



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 – Experiments

Starting with version 1.4, Cloudera Data Science Workbench allows data scientists to run batch experiments that track different versions of code, input parameters, and output (both metrics and files).

### Challenge

As data scientists iteratively develop models, they often experiment with datasets, features, libraries, algorithms, and parameters. Even small changes can significantly impact the resulting model. This means data scientists need the ability to iterate and repeat similar experiments in parallel and on demand, as they rely on differences in output and scores to tune parameters until they obtain the best fit for the problem at hand. Such a training workflow requires versioning of the file system, input parameters, and output of each training run.

Without versioned experiments you would need intense process rigor to consistently track training artifacts (data, parameters, code, etc.), and even then it might be impossible to reproduce and explain a given result. This can lead to wasted time/effort during collaboration, not to mention the compliance risks introduced.

### Solution

Starting with version 1.4, Cloudera Data Science Workbench uses experiments to facilitate ad-hoc batch execution and model training. Experiments are batch executed workloads where the code, input parameters, and output artifacts are versioned. This feature also provides a lightweight ability to track output data, including files, metrics, and metadata for comparison.

### Concepts

The term experiment refers to a non interactive batch execution script that is versioned across input parameters, project files, and output. Batch experiments are associated with a specific project (much like sessions or jobs) and have no notion of scheduling; they run at creation time. To support versioning of the project files and retain run-level artifacts and metadata, each experiment is executed in an isolated container.

### Lifecycle of an Experiment



## Step 1: Create a new project

Go to the homepage of your Data Science workbench, and create a 'New' project.

Call the new repository something like Experiments and Models.

Create the repository as a clone of the github repository:

<https://github.com/andremolenaar/dsfortelcoCDSW.git>

### Create a New Project

#### Project Name

Experiments and Models

#### Project Visibility

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

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

#### Initial Setup

Blank

Template

Local

Git

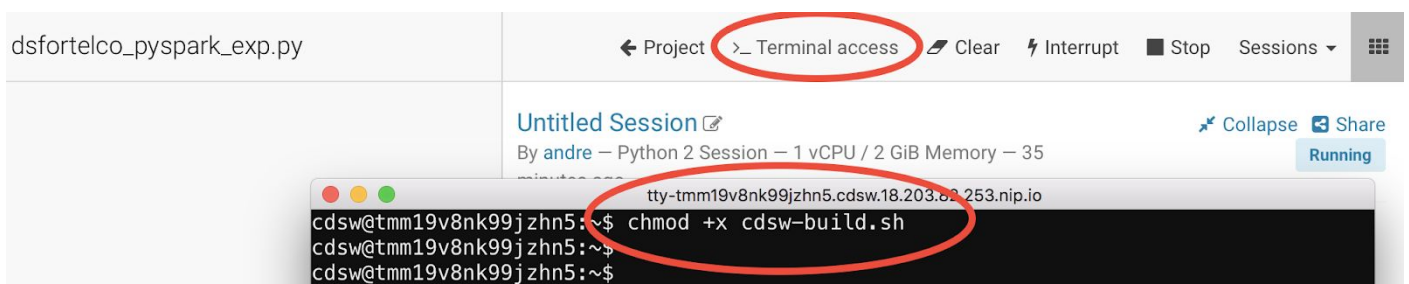
<https://github.com/andremolenaar/dsfortelcoCDSW.git>

Create Project

Start a workbench with a Python 2 and 2 GB of memory.

When the workbench is available, open a terminal window and make the `cdsw-build.sh` program executable. Use the following command to do that:

```
chmod +x cdsw-build.sh
```



## Step 2: Examin `dsfortelco_sklearn_exp.py`

Open the file “dsfortelco\_sklearn\_exp.py”. This is a python program that builds a churn model to predict customer churn (the likelihood that this customer is going to stop his subscription with his telecom operator). There is a dataset available on hdfs (/tmp/churn\_all.csv), with customer data, including a churn indicator field.

The program is going to build a churn prediction model using the Random Forest algorithm. Random forests are ensembles of decision trees. Random forests are one of the most successful machine learning models for classification and regression. They combine many decision trees in order to reduce the risk of overfitting. Like decision trees, random forests handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.

spark.mllib supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features. spark.mllib implements random forests using the existing decision tree implementation. Please see the decision tree guide for more information on trees.

The Random Forest algorithm expects a couple of parameters:

- numTrees: Number of trees in the forest.  
Increasing the number of trees will decrease the variance in predictions, improving the model’s test-time accuracy.  
Training time increases roughly linearly in the number of trees.
- maxDepth: Maximum depth of each tree in the forest.  
Increasing the depth makes the model more expressive and powerful. However, deep trees take longer to train and are also more prone to overfitting.  
In general, it is acceptable to train deeper trees when using random forests than when using a single decision tree. One tree is more likely to overfit than a random forest (because of the variance reduction from averaging multiple trees in the forest).

In the dsfortelco\_pyspark\_exp.py program, these parameters can be passed to the program at runtime. In the lines 38 and 39, these parameters are passed to python variables:

```
param_numTrees=int(sys.argv[1])  
param_maxDepth=int(sys.argv[2])
```

Also note that at the lines 69 and 70, the quality indicator for the Random Forest model, are written back to the Data Science Workbench repository:

```
cdsw.track_metric("auroc", auroc)  
cdsw.track_metric("ap", ap)
```

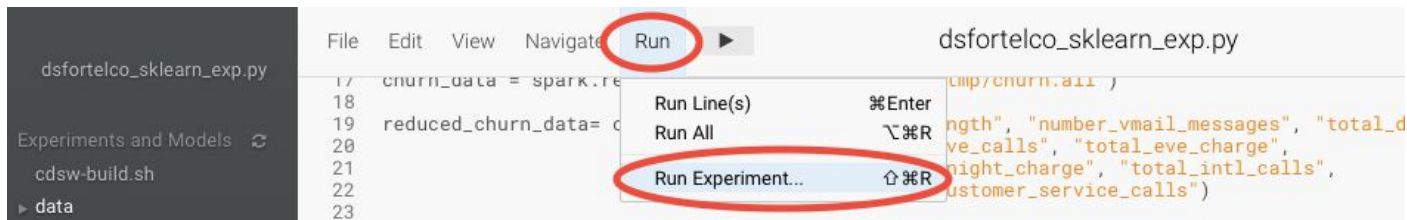
These indicators will show up later in the Experiments dashboard.

### **Step 3: Run the experiment for the first time**

Now, run the experiment using the following parameters:

```
numTrees = 40  
numDepth = 20
```

From the menu, select Run -> Experiments.



Specify the arguments for this run, by typing the numbers behind the arguments field. Note that these fields are separated by a space and that there is no comma (,)

A screenshot of the 'Run New Experiment' dialog box. It contains the following fields:

- Script:** dsfortelco\_sklearn\_exp.py
- Arguments:** 40 20
- Engine Kernel:** Python 2 (selected), Python 3, Scala, R
- Engine Profile:** 1 vCPU / 2 GiB Memory
- Comment:** First Random Forrest Experiment

At the bottom, there are 'Cancel' and 'Start Run' buttons.

Now, in the background, the Data Science Workbench environment will spin up a new docker container, where this program will run.

#### Step 4: Check the results for the first experiment

Go back to the 'Projects' page in CDSW, and hit the 'Experiments' button.



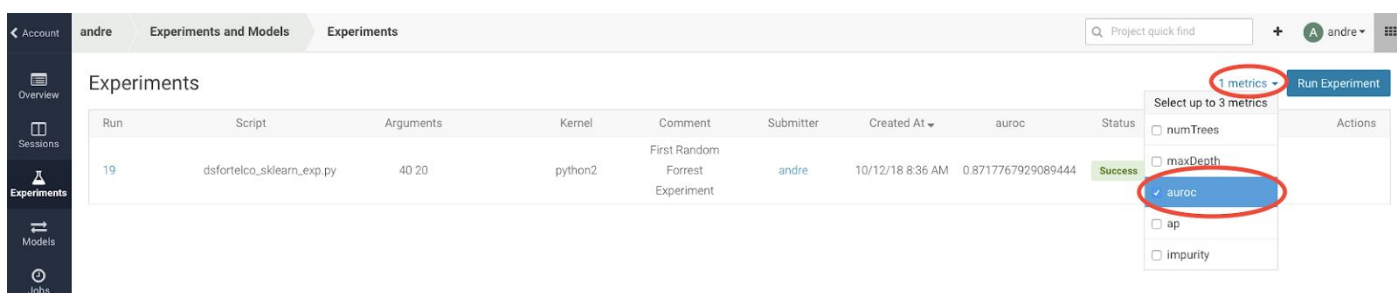
# Cloudera and Deloitte Tech Summit - Data Science Labs



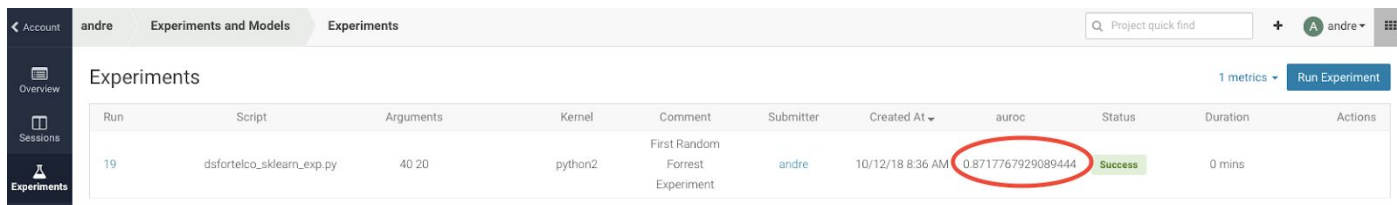
If the Status indicates 'Running', you have to wait till the run is completed.

In case the status is 'Build Failed' or 'Failed', check the log information. This is accessible by clicking on the run number of your experiments. There you can find the session log, as well as the build information.

In case your status indicates 'Success', you should be able to see the auroc (Area Under the Curve) model quality indicator. It might be that this value is hidden by the CDSW user interface. In that case, click on the '3 metrics' links, and select the auroc field. It might be needed to de-select some other fields, since the interface can only show 3 metrics at the same time.



When the auroc metric is selected, you will be able to see the value.



In this example, 0.871

Not bad, but maybe there are better hyper parameter values available.

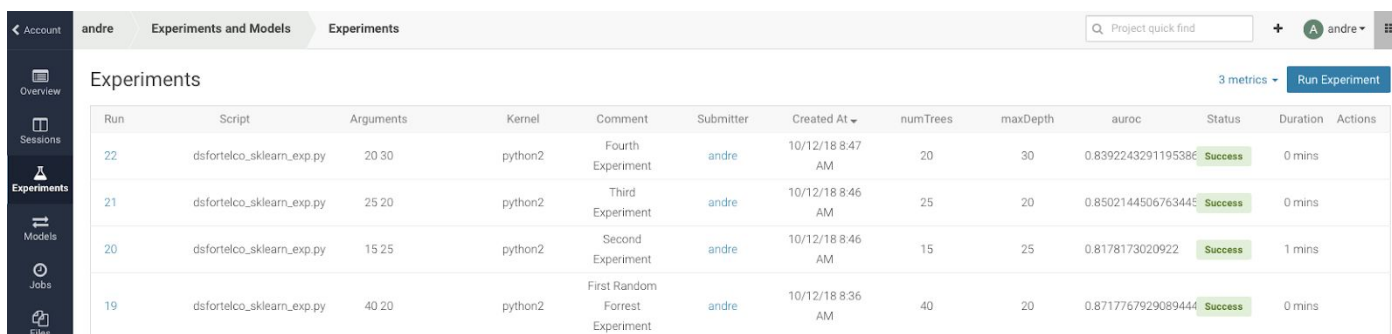
## Step 5: Re run the experiment several times

Now, re-run the experiment 3 more times and try different values for NumTrees and NumDepth. Try the following values:

NumTrees	NumDepth
15	25

25	20
Try something yourself	Try something yourself

When all runs have completed successfully, check which parameters had the best quality (best predictive value). This is represented by the highest 'area under the curve', auroc metric.

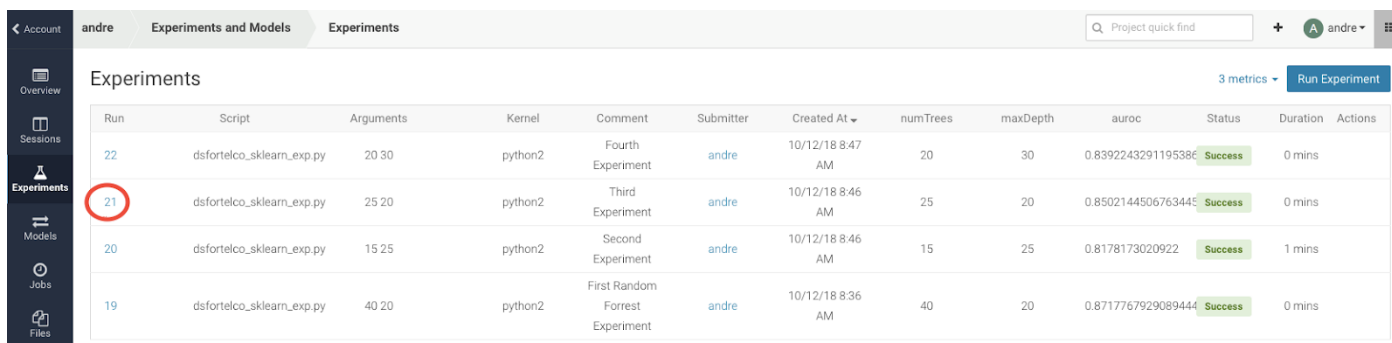


Run	Script	Arguments	Kernel	Comment	Submitter	Created At	numTrees	maxDepth	auroc	Status	Duration	Actions
22	dsfortelco_sklearn_exp.py	20 30	python2	Fourth Experiment	andre	10/12/18 8:47 AM	20	30	0.8392243291195386	Success	0 mins	
21	dsfortelco_sklearn_exp.py	25 20	python2	Third Experiment	andre	10/12/18 8:46 AM	25	20	0.8502144506763445	Success	0 mins	
20	dsfortelco_sklearn_exp.py	15 25	python2	Second Experiment	andre	10/12/18 8:46 AM	15	25	0.8178173020922	Success	1 mins	
19	dsfortelco_sklearn_exp.py	40 20	python2	First Random Forrest Experiment	andre	10/12/18 8:36 AM	40	20	0.8717767929089444	Success	0 mins	

In this example, run 21 had the highest auroc value, so that is the model that you would want to use for your business.

## Step 6: Save the best model to your environment

Select the run number with the best predictive value, in this example, run number 21.



Run	Script	Arguments	Kernel	Comment	Submitter	Created At	numTrees	maxDepth	auroc	Status	Duration	Actions
22	dsfortelco_sklearn_exp.py	20 30	python2	Fourth Experiment	andre	10/12/18 8:47 AM	20	30	0.8392243291195386	Success	0 mins	
21	dsfortelco_sklearn_exp.py	25 20	python2	Third Experiment	andre	10/12/18 8:46 AM	25	20	0.8502144506763445	Success	0 mins	
20	dsfortelco_sklearn_exp.py	15 25	python2	Second Experiment	andre	10/12/18 8:46 AM	15	25	0.8178173020922	Success	1 mins	
19	dsfortelco_sklearn_exp.py	40 20	python2	First Random Forrest Experiment	andre	10/12/18 8:36 AM	40	20	0.8717767929089444	Success	0 mins	

In the Overview screen of the experiment, you can see that the model in spark format, is captured in the file 'sklearn\_rf.pkl'. Select this file and hit the 'Add to Project' button. This will copy the model to your project directory.

← Account **andre** Experiments and Models Experiments 21 Overview

Overview Sessions Experiments Models Jobs Files Team Settings

**Run-21** [New Experiment](#)

Overview **Session** Build

**Configuration**

Script	dsfortelco_sklearn_exp.py
Arguments	25 20
Comment	Third Experiment
Build Snapshot	77e3b0ce1cf6b010c840e282aeda6d7ffd8dc248
Created At	10/12/18 8:46 AM
Submitter	andre

**Metrics**

numTrees	25
maxDepth	20
impurity	gini
auroc	0.8502144506763445
ap	0.680864067072068

**Output**

- sklearn\_rf.pkl

[Add to Project](#)

### **Lab 12 – Working with Models**

Starting with version 1.4, Cloudera Data Science Workbench allows data scientists to build, deploy, and manage models as REST APIs to serve predictions.

#### **Challenge**

Data scientists often develop models using a variety of Python/R open source packages. The challenge lies in actually exposing those models to stakeholders who can test the model. In most organizations, the model deployment process will require assistance from a separate DevOps team who likely have their own policies about deploying new code.

For example, a model that has been developed in Python by data scientists might be rebuilt in another language by the devops team before it is actually deployed. This process can be slow and error-prone. It can take months to deploy new models, if at all. This also introduces compliance risks when you take into account the fact that the new re-developed model might not be even be an accurate reproduction of the original model.

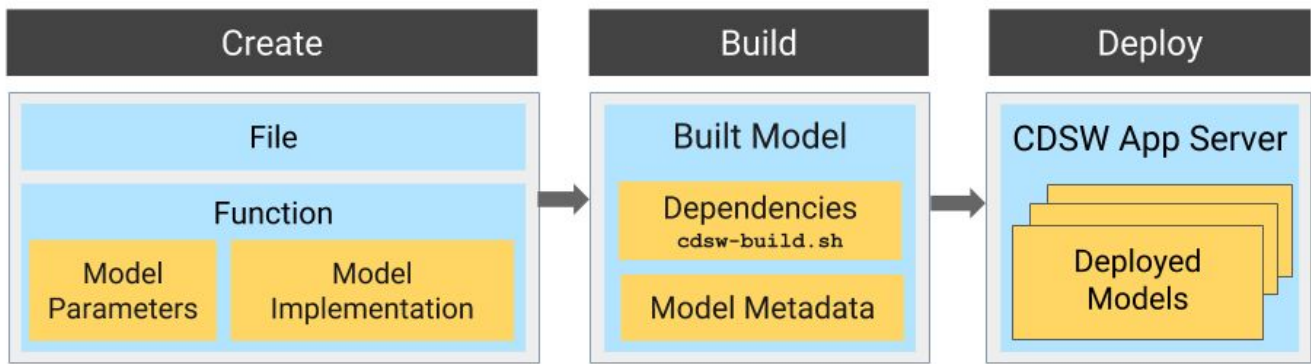
Once a model has been deployed, you then need to ensure that the devops team has a way to rollback the model to a previous version if needed. This means the data science team also needs a reliable way to retain history of the models they build and ensure that they can rebuild a specific version if needed. At any time, data scientists (or any other stakeholders) must have a way to accurately identify which version of a model is/was deployed.

#### **Solution**

Starting with version 1.4, Cloudera Data Science Workbench allows data scientists to build and deploy their own models as REST APIs. Data scientists can now select a Python or R function within a project file, and Cloudera Data Science Workbench will:

- Create a snapshot of model code, model parameters, and dependencies.
- Package a trained model into an immutable artifact and provide basic serving code.
- Add a REST endpoint that automatically accepts input parameters matching the function, and that returns a data structure that matches the function's return type.
- Save the model along with some metadata.
- Deploy a specified number of model API replicas, automatically load balanced.

Stages of the Model Deployment Process



**Step 1: Examine the program predic\_churn\_sklearn.py**

Open the project you created in the previous lab, and examine the file.

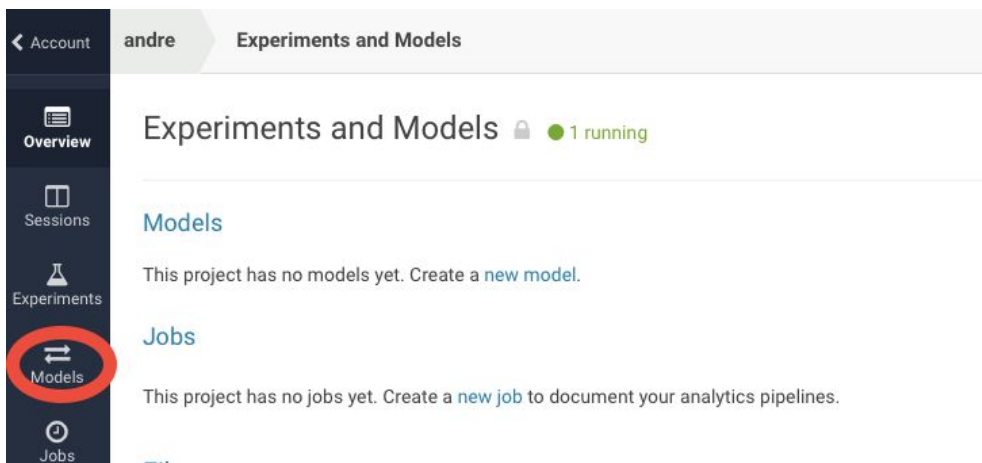
```
1 import pickle
2 import numpy as np
3
4 model = pickle.load(open("models/sklearn_rf.pkl", "rb"))
5
6 def predict(args):
7     account=np.array(args["feature"].split(",")).reshape(1, -1)
8     return {"result" : model.predict(account)[0]}
9
10
```

This PySpark program uses the pickle.load mechanism to deploy models.. The model it refers to the sklearn\_rf.pkl file, was saved in the previous lab from the experiment with the best predictive model.

There is a predict definition which is the function that calls the model, using features, and will return a result variable.

**Step 2: Deploy the model**

From the projects page of your project, select the 'Models' button.



Select 'New Model', and populate specify the following configuration:

Name: something like "My Churn Prediction Model"  
Description: Anything you want  
File: predict\_churn\_sklearn.py  
Function: predict  
Example Inp: {  
                  "feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9.4, 6, 3.43, 4"  
                  }  
Kernal: Python 2  
Engine: 1 vCPU / 2 GiB Memory  
Replicas: 1

## Create a Model

### General

**Name \***

**Description \***

### Build

**File \***

**Function \***

**Example Input** ⓘ  

```
{  
  "feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9.4, 6, 3.43, 4"  
}
```

**Example Output** ⓘ  

```
{ "result": "value" }
```

**Kernel**  
 Python 2  
 Python 3  
 R

**Comment**

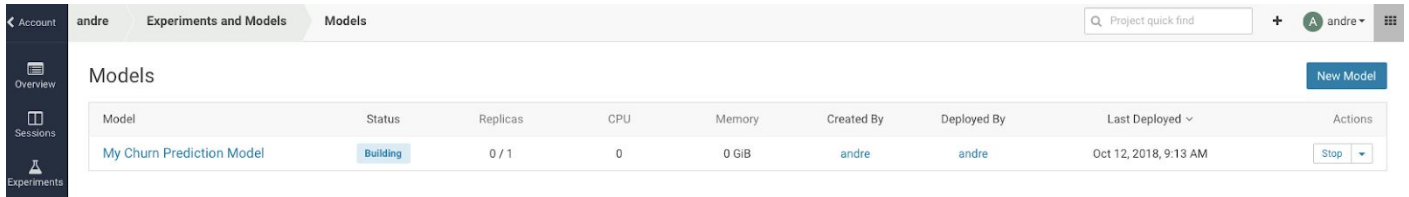
### Deployment

**Engine Profile**

**Replicas**

[Set Environmental Variables](#)

If all parameters are set, you can hit the 'Deploy Model' button. Wait till the model is deployed. This will take several minutes.

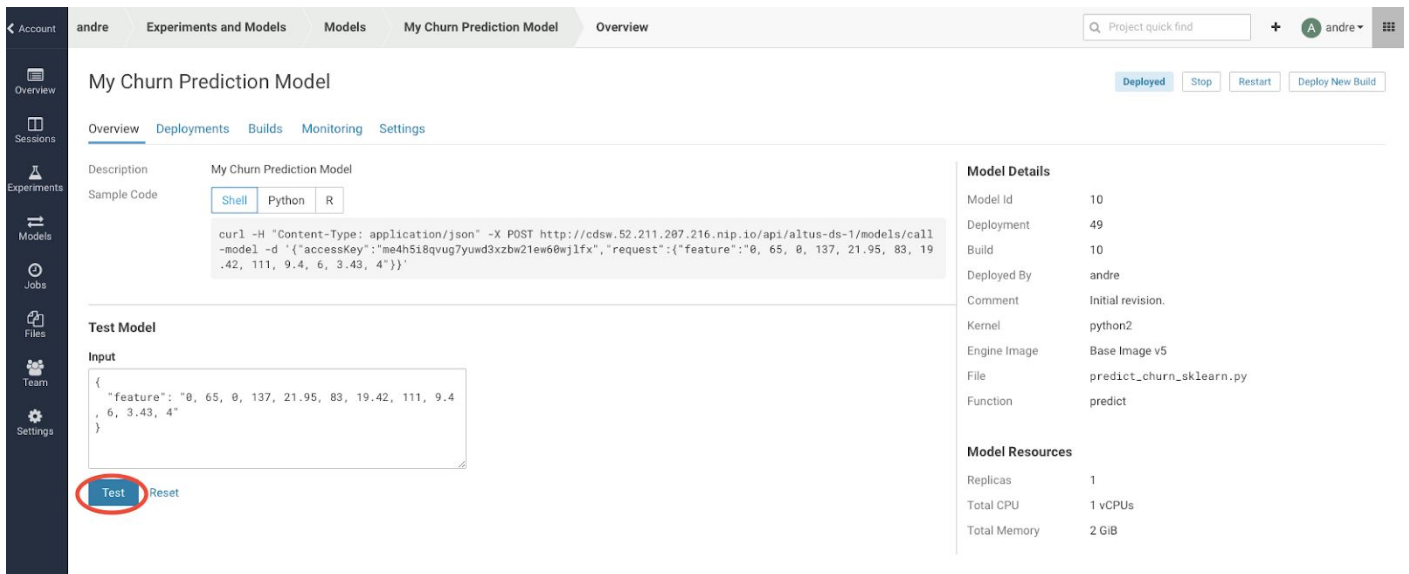


### Step 3: Test the deployed model

After the several minutes, your model should get to the 'Deployed' state.



Now, click on the Model Name link, to go to the Model Overview page. From the that page, hit the 'Test' button to check if the model is working.



If your model is working, you should receive an output similar like this:



### Test Model

#### Input

```
{  
  "feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9.4  
  , 6, 3.43, 4"  
}
```

Test

Reset

#### Result

Status	<span style="color: green;">●</span> success
Response	<pre>{   "result": 1 }</pre>
Replica ID	my-churn-prediction-model-10-49-599d6548d8-x9cmp

The green color with success is telling that our REST call to the model is technically working. And if you examine the response: {"result": 1}, it returns a 1, which mean that customer with these features is likely to churn.

Now, lets change the input parameters and call the predict function again. Put the following values in the Input field:

```
{  
  "feature": "0, 95, 0, 88, 26.62, 75, 21.05, 115, 8.65, 5, 3.32, 3"  
}
```

## Test Model

### Input

```
{
  "feature": "0, 95, 0, 88, 26.62, 75, 21.05, 115, 8.65, 5, 3.32, 3"
}
```

Test Reset

### Result

Status	● success
Response	{ "result": 0 }
Replica ID	my-churn-prediction-model-10-49-599d6548d8-x9cmp

With these input parameters, the model returns 0, which means that the customer is not likely to churn.

## Step 4: Model Administration

When a model is deployed, Cloudera Data Science Workbench allows you to specify a number of replicas that will be deployed to serve requests. For each active model, you can monitor its replicas by going to the model's Monitoring page. On this page you can track the number of requests being served by each replica, success and failure rates, and their associated stderr and stdout logs. Depending on future resource requirements, you can increase or decrease the number of replicas by re-deploying the model.

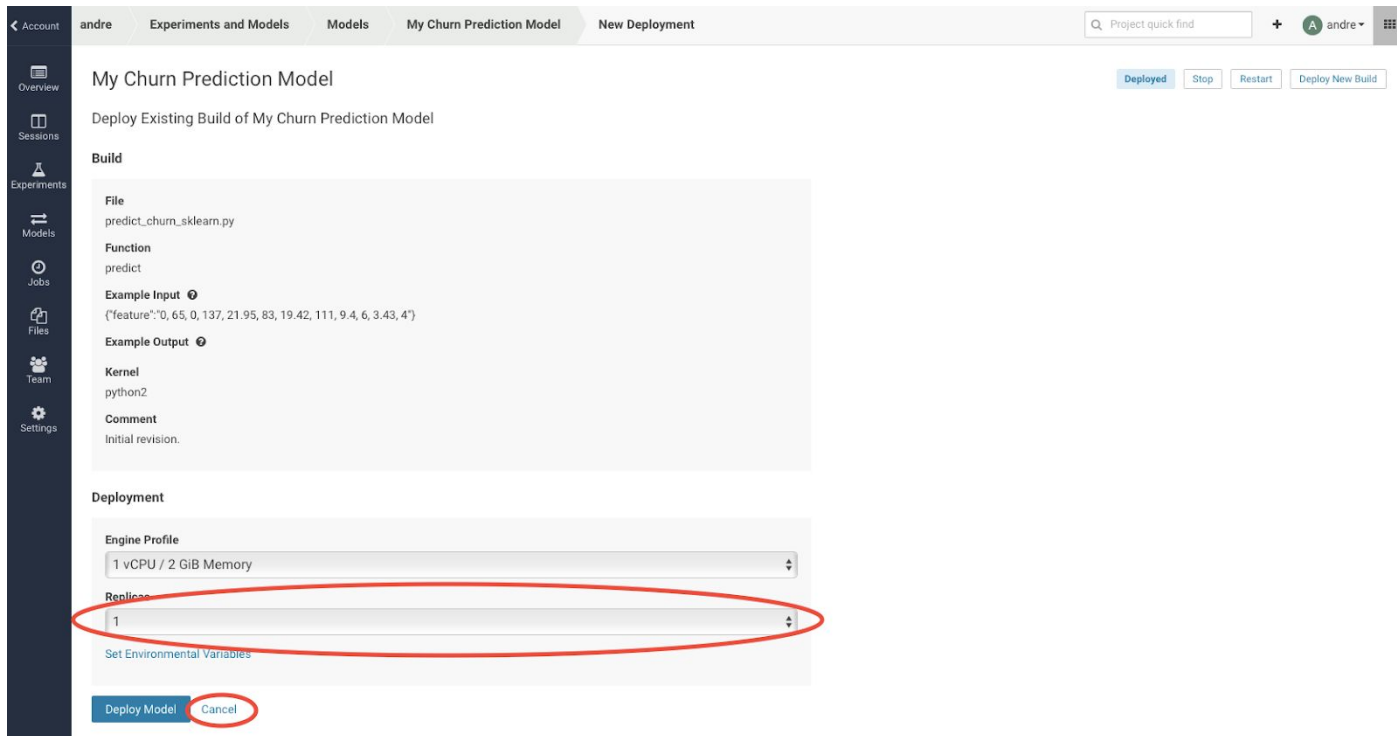
The screenshot shows the Cloudera Data Science Workbench interface. The top navigation bar includes 'Account', 'andre', 'Experiments and Models', 'Models', 'My Churn Prediction Model', and 'Deployments'. A search bar and user profile are on the right. The left sidebar contains navigation icons for Overview, Sessions, Experiments, Models, Jobs, Files, Team, and Settings. The main content area is titled 'My Churn Prediction Model' and has tabs for Overview, Deployments, Builds, Monitoring, and Settings. The 'Deployments' tab is active and shows a table with one deployment:

Id	Build	Status	Deployed At	Stopped At	Deployed By
49	1	Deployed	Oct 12, 2018, 9:15 AM		andre

Below the table, there are buttons for 'Deployed', 'Stop', 'Restart', and 'Deploy New Build'. On the right side, there is a 'Model' details panel with fields for Id, Name, Description, Build Number, UUID, File, Function, Kernel, Engine, and Deployment. A 'Re-deploy This Build' button is circled in red in the top right corner of the model details panel.

## Cloudera and Deloitte Tech Summit - Data Science Labs

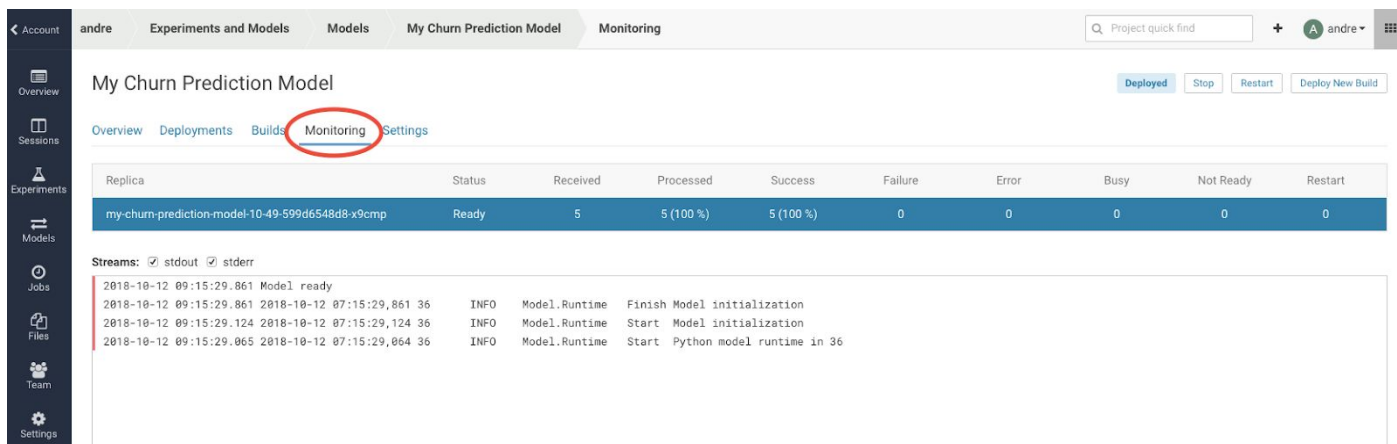
When you get to the re-deployment page, you can increase the number of replica's.



The screenshot shows the 'New Deployment' page for a model named 'My Churn Prediction Model'. The 'Build' section displays the file 'predict\_churn\_sklearn.py', the function 'predict', and example input/output. The 'Deployment' section shows the engine profile '1 vCPU / 2 GiB Memory' and the number of replicas set to '1'. The 'Deploy Model' button is highlighted with a red circle, and the 'Cancel' button is also highlighted with a red circle.

In order not to overload the cluster, hit the 'Cancel' button to return to the running model page.

Now, navigate to the 'Monitoring' tab.



The screenshot shows the 'Monitoring' page for the 'My Churn Prediction Model'. The 'Monitoring' tab is highlighted with a red circle. A table displays replica statistics, and a log stream is visible below.

Replica	Status	Received	Processed	Success	Failure	Error	Busy	Not Ready	Restart
my-churn-prediction-model-10-49-599d6548d8-x9cmp	Ready	5	5 (100 %)	5 (100 %)	0	0	0	0	0

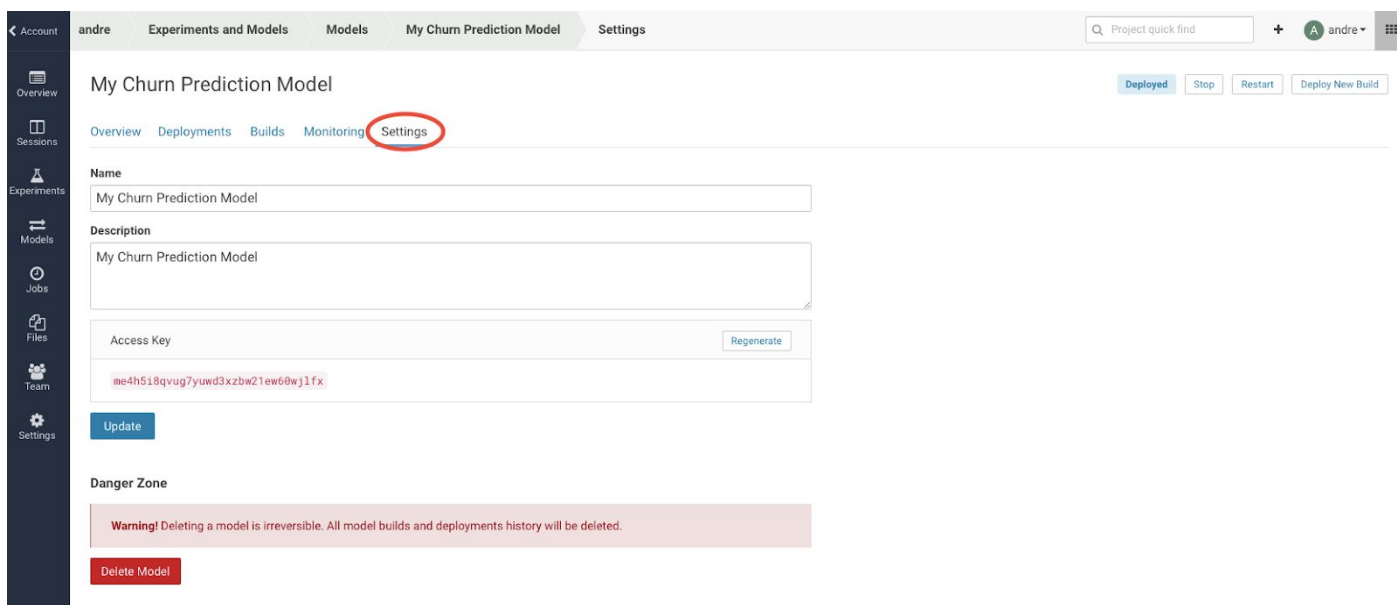
Streams:  stdout  stderr

```
2018-10-12 09:15:29.861 Model ready
2018-10-12 09:15:29.861 2018-10-12 07:15:29,861 36 INFO Model.Runtime Finish Model initialization
2018-10-12 09:15:29.124 2018-10-12 07:15:29,124 36 INFO Model.Runtime Start Model initialization
2018-10-12 09:15:29.065 2018-10-12 07:15:29,064 36 INFO Model.Runtime Start Python model runtime in 36
```

Several statistics of the model are displayed, like the number of times the model has been called, have been processed, etc.

Logfile information is also available here. The most recent logs are at the top of the pane (see image). stderr logs are displayed next to a red bar while stdout logs are by a green bar. Note that model logs and statistics are only preserved so long as the individual replica is active. When a replica restarts (for example, in case of bad input) the logs also start with a clean slate.

Now, navigate to the 'Settings' tab.

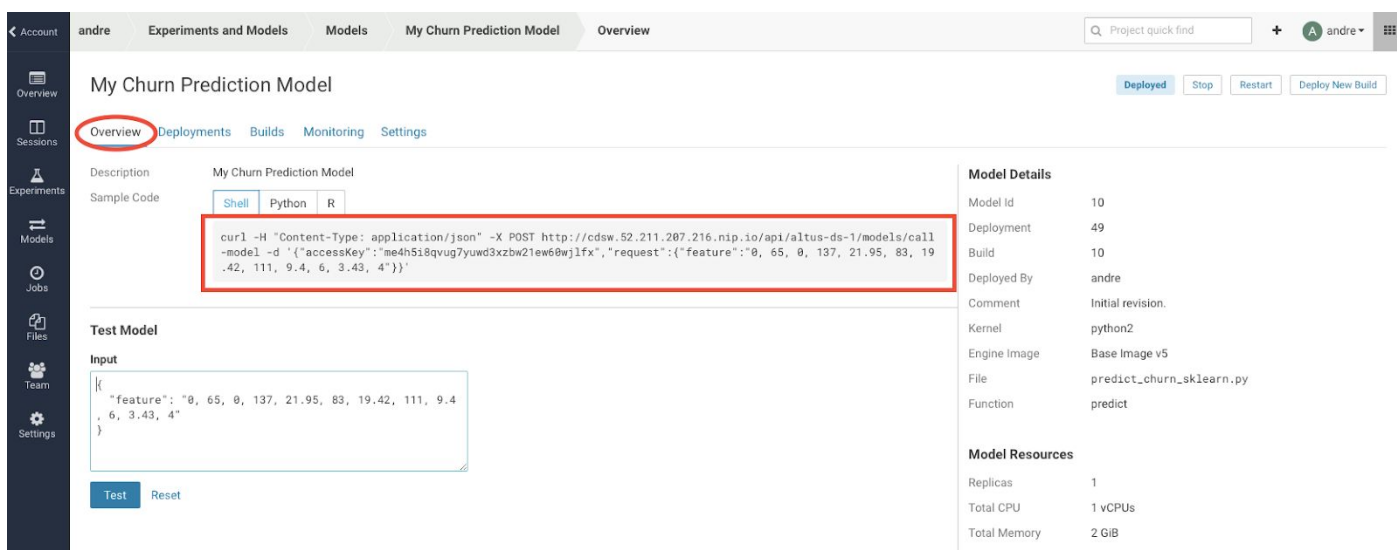


On the settings tab, you will be able to find the "Access Key" that is needed in order to call the model with a REST webservice call.

## Step 5: Test the rest service from a commandline.

The last step in this workshop, is to test the predict function from another (virtual) machine, using the "curl" tool.

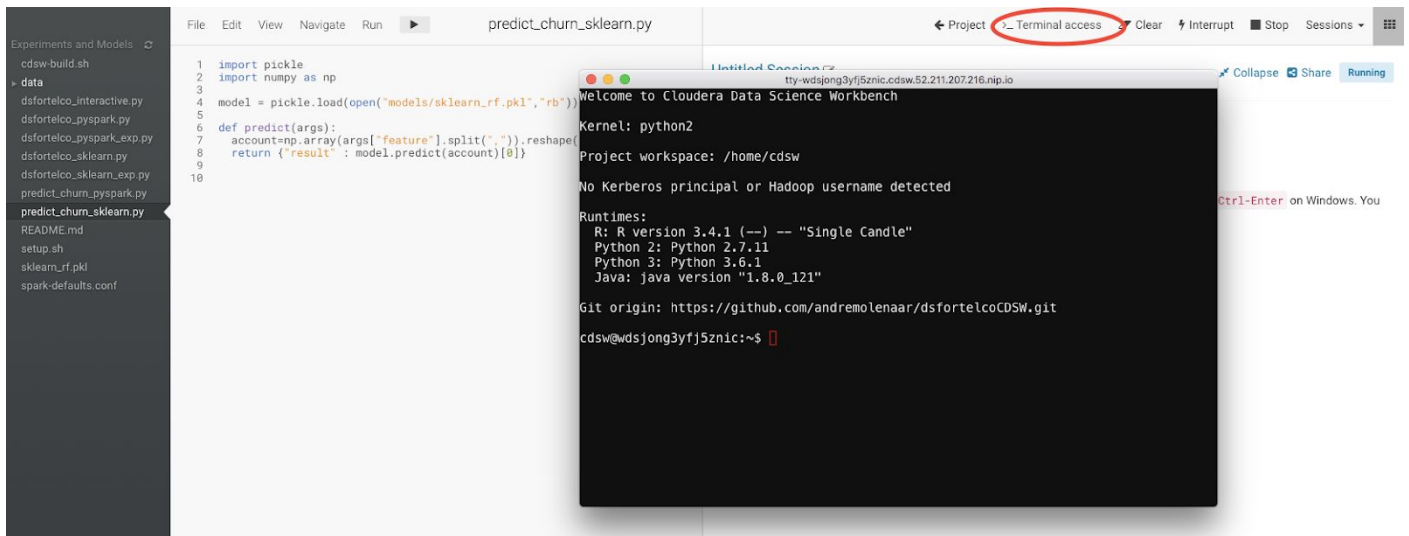
Navigate to the Overview tab of your running model.



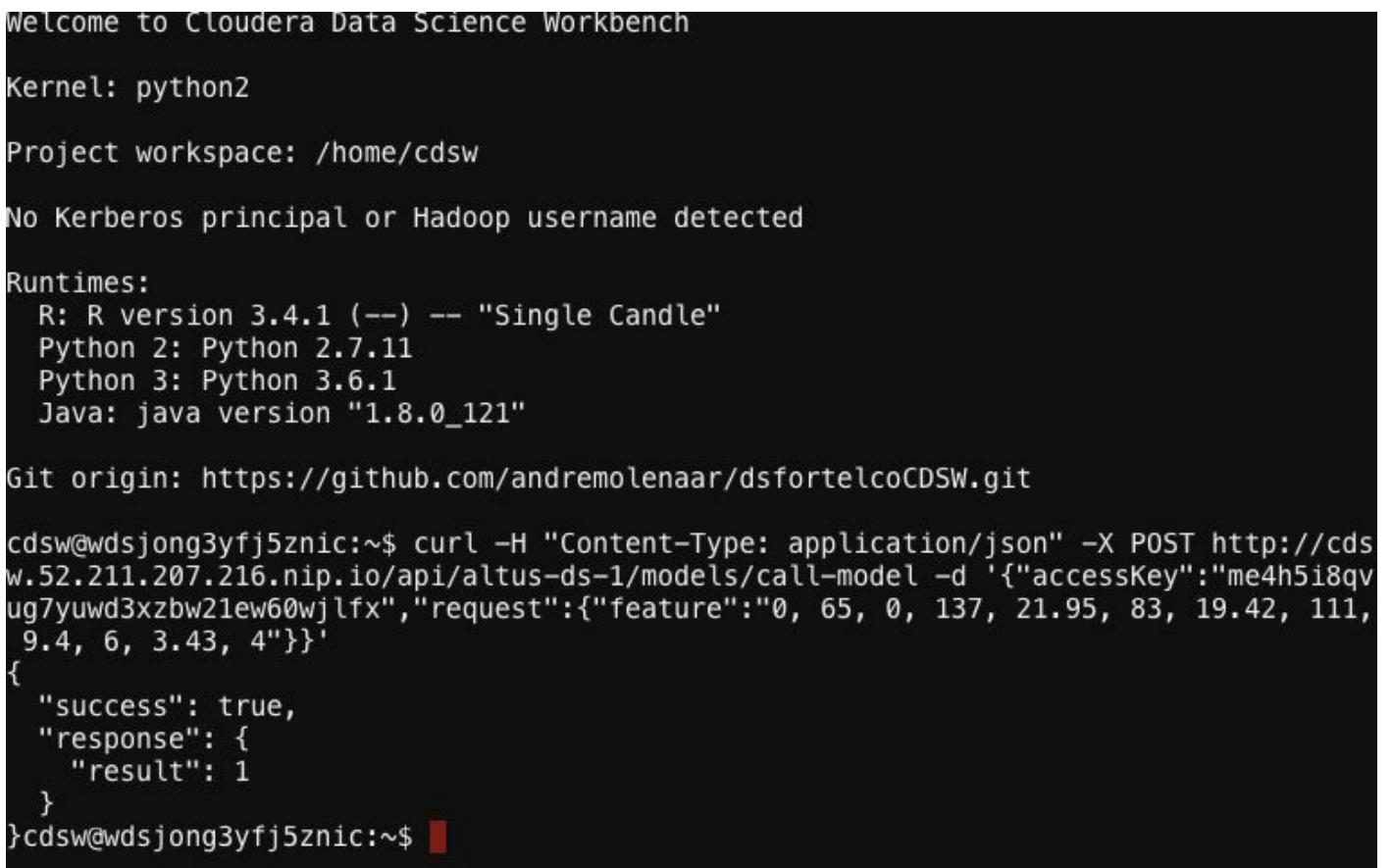
## Cloudera and Deloitte Tech Summit - Data Science Labs

Copy the whole shell statement, starting with 'curl -H ....'

Open a workbench session, running python 2 with 2 GB of memory. When the session is available, open a Terminal.



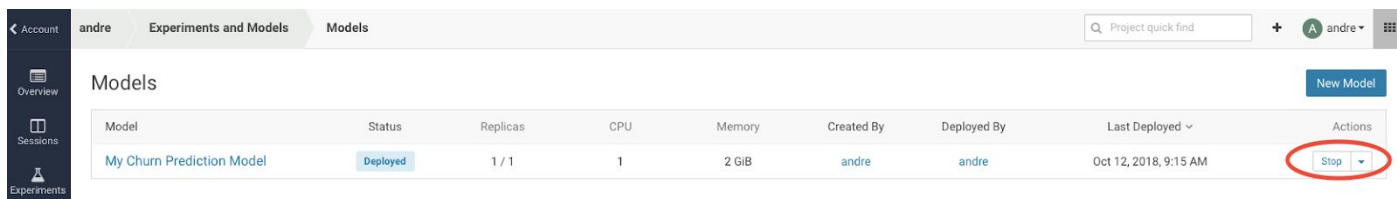
Now, paste the curl statement to the command prompt, and run the statement.



## Cloudera and Deloitte Tech Summit - Data Science Labs

The response shows that the model is still running and making predictions.

That completes our lab with models. Please, free up some resources for other people and new projects. So stop your workbench session. And from the Models page, also stop your deployed model.



The screenshot shows the Cloudera Databricks interface. The top navigation bar includes 'Account', 'andre', 'Experiments and Models', and 'Models'. A search bar for 'Project quick find' and a user profile 'andre' are also visible. The main content area is titled 'Models' and contains a table with the following data:

Model	Status	Replicas	CPU	Memory	Created By	Deployed By	Last Deployed	Actions
My Churn Prediction Model	Deployed	1 / 1	1	2 GiB	andre	andre	Oct 12, 2018, 9:15 AM	Stop

### **Lab 13 – 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 with a Python template, 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 5:**

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 6:**

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 7:**

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 8:**

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 9:

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)`

If you don't feel like coding, you can also clone the github repository [https://github.com/andremolenaar/face\\_recognition](https://github.com/andremolenaar/face_recognition)

## Lab 14 – Optional: Install CDSW in your Cluster

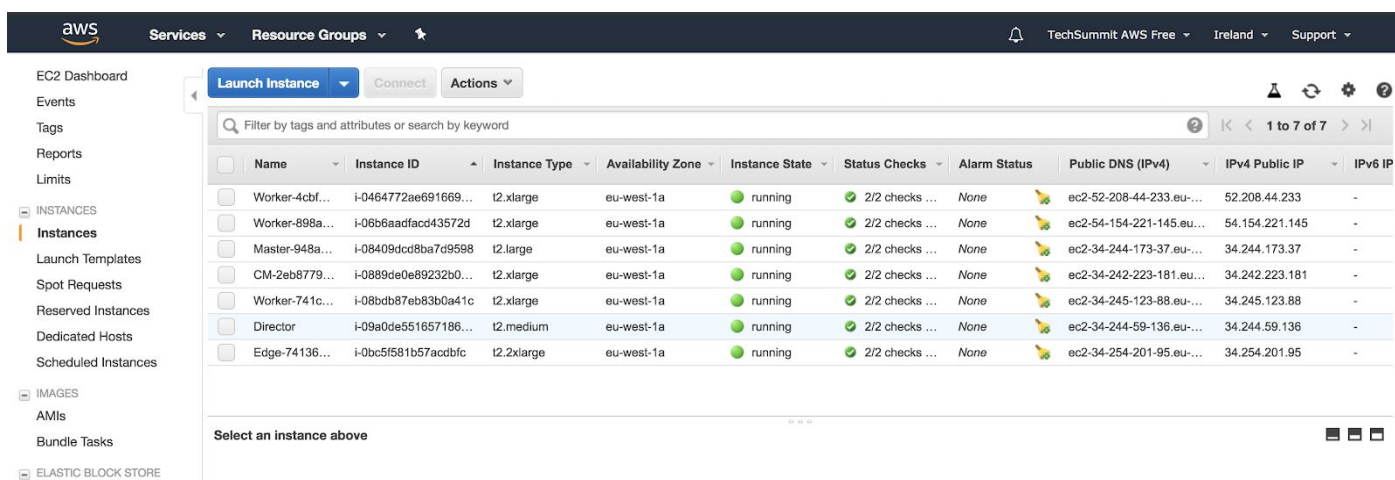
In this lab you'll learn how to:

1. Upgrade your cluster to run Spark
2. Install CDSW using Cloudera Manager
3. Configure CDSW

Please note that this lab is Optional. If you are not interested in installing and configuring the Data Science Workbench application yourself, you may proceed with Lab 1.

### Step 1: Make sure that you cluster is up-and running

Use your AWS console, to check that all instances needed for your cluster are available. You can access this page from your AWS Dashboard -> EC2 -> Instances. You should see something like this:



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IP
Worker-4cbf...	i-0464772ae691669...	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-52-208-44-233.eu...	52.208.44.233	-
Worker-898a...	i-06b6aadfacd43572d	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-54-154-221-145.eu...	54.154.221.145	-
Master-948a...	i-08409dcd8ba7d9598	t2.large	eu-west-1a	running	2/2 checks ...	None	ec2-34-244-173-37.eu...	34.244.173.37	-
CM-2eb8779...	i-0889de0e89232b0...	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-34-242-223-181.eu...	34.242.223.181	-
Worker-741c...	i-08bdb87eb83b0a41c	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-34-245-123-88.eu...	34.245.123.88	-
Director	i-09a0de551657186...	t2.medium	eu-west-1a	running	2/2 checks ...	None	ec2-34-244-59-136.eu...	34.244.59.136	-
Edge-74136...	i-0bc5f581b57acdbfc	t2.2xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-34-254-201-95.eu...	34.254.201.95	-

If your instances are not yet running, select them, and start them using the 'Actions' button.

ToDo: set the centos version (/etc/centos.version) to 7.4

ToDo: umount /dev/data0 , umount /dev/data1 on the edge node.

### Step 2: Check the health of your cluster

Now we need to find out the ip address of your Cloudera Manager instance, and login to the Cloudera Manager application.

In your EC2 dashboard, select the instance with the name that starts with 'CM'. In the properties panel below, copy the IPv4 Public IP address.

# Cloudera and Deloitte Tech Summit - Data Science Labs

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IP
Worker-4cbf...	i-0464772ae691669...	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-52-208-44-233.eu...	52.208.44.233	-
Worker-898a...	i-06b6aadfacd43572d	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-54-154-221-145.eu...	54.154.221.145	-
Master-948a...	i-08409dcd8ba7d9598	t2.large	eu-west-1a	running	2/2 checks ...	None	ec2-34-244-173-37.eu...	34.244.173.37	-
CM-2eb8779...	i-0889de0e89232b0...	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-34-242-223-181.eu...	34.242.223.181	-
Worker-741c...	i-08bdb87eb83b0a41c	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-34-245-123-88.eu...	34.245.123.88	-
Director	i-09a0de551657186...	t2.medium	eu-west-1a	running	2/2 checks ...	None	ec2-34-244-59-136.eu...	34.244.59.136	-
Edge-74136...	i-0bc5f581b57acd8fc	t2.xlarge	eu-west-1a	running	2/2 checks ...	None	ec2-34-254-201-95.eu...	34.254.201.95	-

Instance: **i-0889de0e89232b03b (CM-2eb8779d-7756-499b-a438-dc75ccc6a180)** Public DNS: **ec2-34-242-223-181.eu-west-1.compute.amazonaws.com**

**Description** | Status Checks | Monitoring | Tags | Usage Instructions

Instance ID	i-0889de0e89232b03b	Public DNS (IPv4)	ec2-34-242-223-181.eu-west-1.compute.amazonaws.com
Instance state	running	IPv4 Public IP	<b>34.242.223.181</b>
Instance type	t2.xlarge	IPv6 IPs	-
Elastic IPs		Private DNS	ip-10-0-0-52.eu-west-1.compute.internal
Availability zone	eu-west-1a	Private IPs	10.0.0.52
Security groups	SecurityGroup . view inbound rules . view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-0113f70c97b2316cf
AMI ID	CentOS Linux 7 x86_64 HVM EBS ENA 1804_2-b7ee8a69-ee97-4a49-9e68-afae216db2e-ami-55a2322a.4 (ami-4c457735)	Subnet ID	subnet-0ef455007fab3e61
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True
Key pair name	tech-summit	T2/T3 Unlimited	Disabled

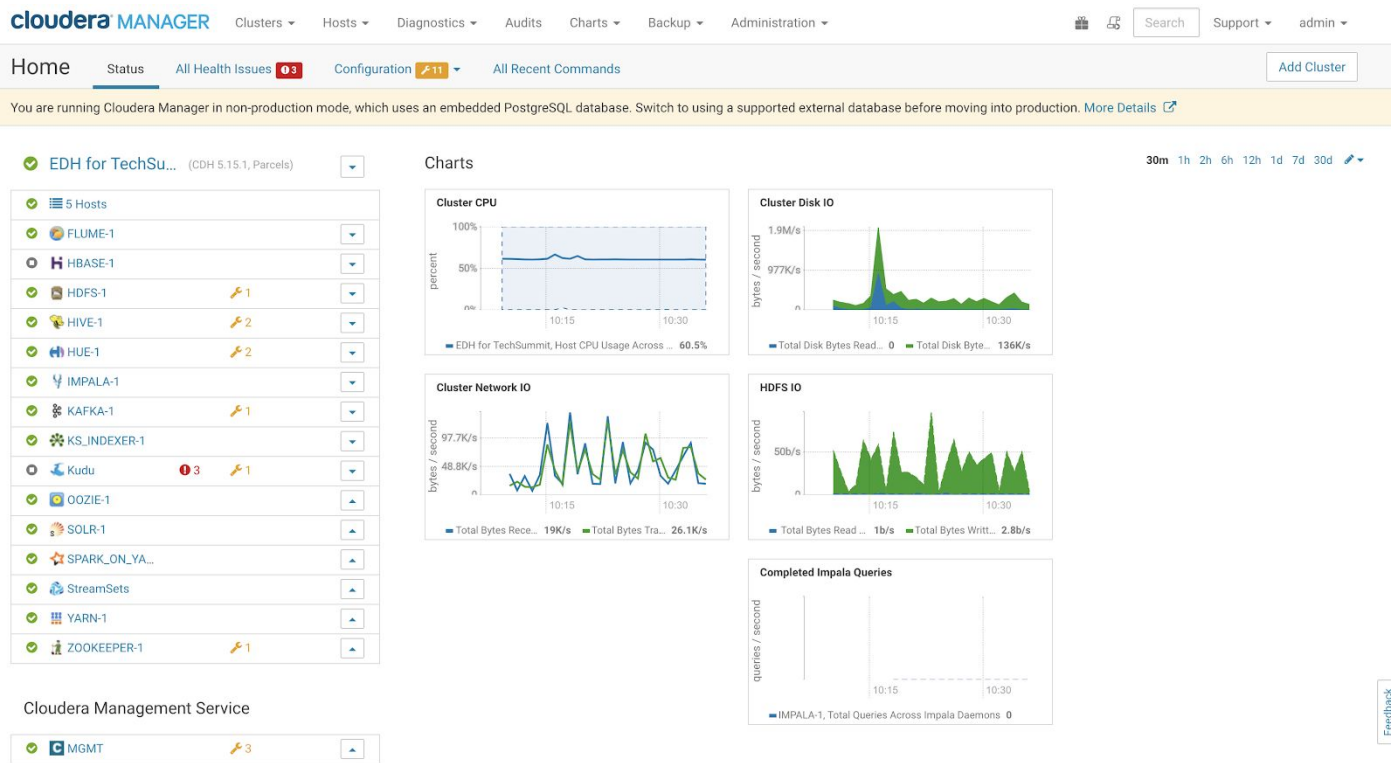
Now, open a new browser tab, and navigate to the Cloudera Manager login page. The link to your personal Cloudera Manager login is: <http://<CM-ip-address>:7180>

In this example, the Cloudera Manager login page is available at <http://34.242.223.181:7180>

Login to Cloudera Manager, using the credential you created earlier. If you followed the defaults, your user is 'admin' and your password is 'Cloudera\_123'.

As soon as you are logged-in, check if your all your services are running. You may need to restart some services. On AWS your nodes may get a new ip address after a restart. Restarting a service will in most cases resolve problems. You should end with a running cluster.

# Cloudera and Deloitte Tech Summit - Data Science Labs



## Step 3: Upgrade Spark to version 2.1

SSH into the Cloudera Manager host and download the CSD file (where you need to substitute the public IP address here below with yours). If you have a Windows machine, please use Putty to setup your ssh terminal. See the Appendix for Putty setup instructions.

```
ssh -i tech-summit centos@34.242.223.181
```

Install the wget download tool. If you attended the IoT workshop, you may skip this step:

```
sudo yum install wget
```

Download the Spark2.1 service descriptor:

```
wget
```

```
http://archive.cloudera.com/spark2/csd/SPARK2_ON_YARN-2.1.0.cloudera3.jar
```

Put the file in the right location with the right attributes:

```
sudo mv SPARK2_ON_YARN-2.1.0.cloudera3.jar /opt/cloudera/csd/  
sudo chmod 644 /opt/cloudera/csd/SPARK2_ON_YARN-2.1.0.cloudera3.jar  
sudo chown cloudera-scm:cloudera-scm  
/opt/cloudera/csd/SPARK2_ON_YARN-2.1.0.cloudera3.jar
```

Restart the Cloudera Manager Service:

```
sudo service cloudera-scm-server restart
```

# Cloudera and Deloitte Tech Summit - Data Science Labs

Open the Cloudera Manager and restart the Management Service:

The screenshot shows the Cloudera Manager interface. The top navigation bar includes 'Home', 'Status', 'All Health Issues' (with a red notification icon), 'Configuration', and 'All Recent Commands'. Below this, a yellow banner states: 'You are running Cloudera Manager in non-production mode, which uses an embedded PostgreSQL database. Switch to using a supported external database before moving into production. M'. The main content area is divided into a left sidebar and a right 'Charts' section. The sidebar lists various services: 5 Hosts, FLUME-1, HBASE-1, HDFS-1, HIVE-1, HUE-1, IMPALA-1, KAFKA-1, KS\_INDEXER-1, Kudu, OOZIE-1, SOLR-1, SPARK\_ON\_YARN-1, StreamSets, YARN-1, ZOOKEEPER-1, and MGMT. The MGMT service is selected, and a context menu titled 'MGMT Actions' is open over it. The 'Restart' option in this menu is circled in red. The 'Charts' section contains five graphs: 'Cluster CPU' (showing 60.5% usage), 'Cluster Disk IO' (showing 273b/s read and 382b/s write), 'Cluster Network IO' (showing 80.7K/s read and 125K/s write), 'HDFS IO' (showing 1b/s read and 95.2b/s write), and 'Completed Impala Queries' (showing 0 queries).

When Cloudera Manager has been restarted, navigate to the 'Parcel', search for the Spark2 parcel, and hit the download button.

The screenshot shows the 'Parcels' page in Cloudera Manager. The top navigation bar includes 'Home', 'Status', 'All Health Issues', 'Configuration', and 'All Recent Commands'. Below this, a yellow banner states: 'You are running Cloudera Manager in non-production mode, which uses an embedded PostgreSQL database. Switch to using a supported external database before moving into production. M'. The main content area is divided into a left sidebar and a right 'Parcels' section. The sidebar contains filters for 'ERROR STATUS', 'PARCEL NAME', and 'STATUS'. The 'PARCEL NAME' filter is expanded, and 'SPARK2' is selected and circled in red. The 'Parcels' section shows a table with the following data:

Parcel Name	Version	Status
SPARK2	2.1.0.cloudera3-1.cdh5.13.3.p0.569822	Available Remotely

The 'Download' button for the SPARK2 parcel is circled in red.

When the parcel is downloaded, hit the 'Distribute' button. And when the parcel is distributed, hit the 'Activate' button. When the service is activated, you should see something like this:

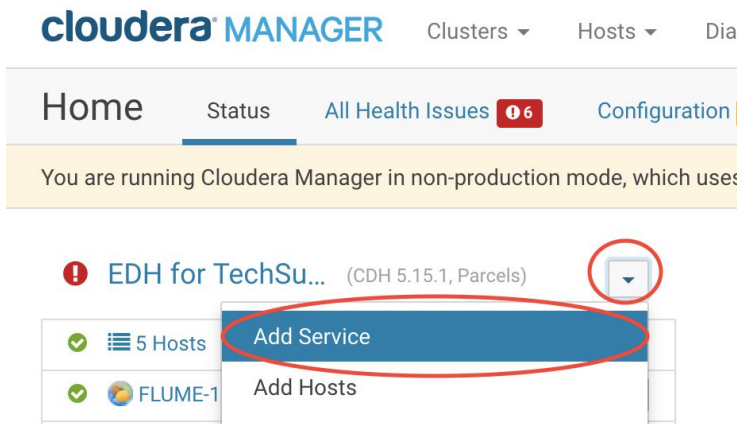
## Cloudera and Deloitte Tech Summit - Data Science Labs

### EDH for TechSummit

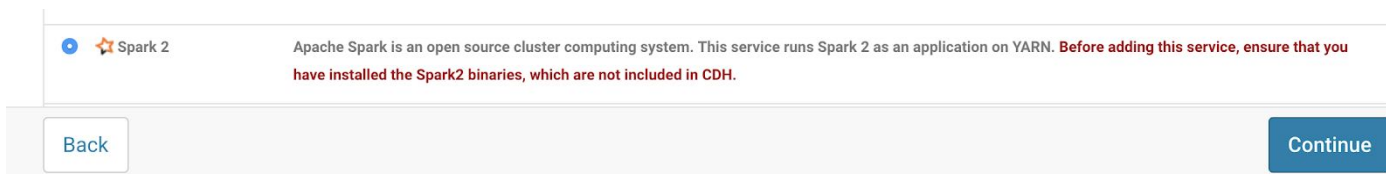
Parcel Name	Version	Status	
SPARK2	2.1.0.cloudera3-1.cdh5.13.3.p0.569822	Distributed, Activated	<a href="#">Deactivate</a>

The last step in the Spark2 setup is to add the Spark2 service to the cluster. You can do this through Cloudera Manager:

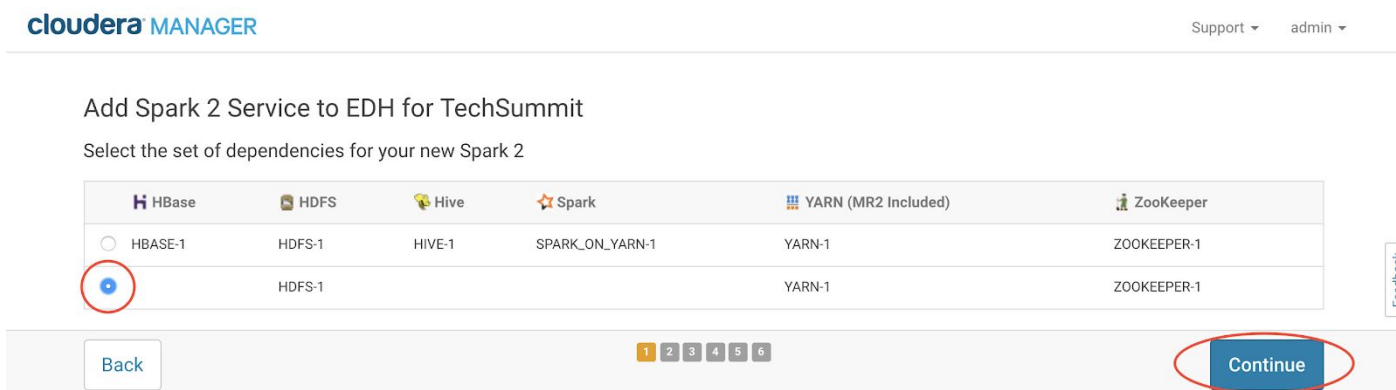
From the homepage, select the 'Add Service' option from the Cluster dropdown menu.



From the Service panel, select the 'Spark 2' service and hit Continue.



Select the dependencies for only HDFS-1 and YARN-1, and hit continue.



Specify the Cloudera Manager node as the Spark History Server node, and the Edge node as the Spark Gateway node. Then hit Continue.

# Cloudera and Deloitte Tech Summit - Data Science Labs

cloudera MANAGER Support admin

### Add Spark 2 Service to EDH for TechSummit

Assign Roles for Spark 2

You can customize the role assignments for your new service here, but note that if assignments are made incorrectly, such as assigning too many roles to a single host, performance will suffer.

You can also view the role assignments by host: [View By Host](#)

History Server x 1 New Gateway x 1 New

ip-10-0-0-220.eu-west-1.compute.int... ip-10-0-0-66.eu-west-1.compute.inter...

Feedback

1 2 3 4 5 6

[Back](#) [Continue](#)

Wait till the service is deployed and started. From your Cloudera Manager console you should see the service is added and started:

✓ EDH for TechSu... (CDH 5.15.1, Parcels) [dropdown]

✓ 5 Hosts [wrench icon] 1	[dropdown]
✓ FLUME-1	[dropdown]
⊖ HBASE-1	[dropdown]
✓ HDFS-1 [wrench icon] 1	[dropdown]
✓ HIVE-1 [wrench icon] 2	[dropdown]
✓ HUE-1 [wrench icon] 2	[dropdown]
✓ IMPALA-1	[dropdown]
✓ KAFKA-1 [wrench icon] 1	[dropdown]
✓ KS_INDEXER-1	[dropdown]
⊖ Kudu [wrench icon] 1	[dropdown]
✓ OOZIE-1	[dropdown]
✓ SOLR-1	[dropdown]
✓ SPARK_ON_YA...	[dropdown]
✓ Spark 2	[dropdown]
✓ StreamSets	[dropdown]
✓ YARN-1	[dropdown]
✓ ZOOKEEPER-1 [wrench icon] 1	[dropdown]

Step 3: Download the Cloudera Data Science Workbench parcel

SSH into the Cloudera Manager host and download the CSD file (where you need to substitute the public IP address here below with yours). If you have a Windows machine, please use Putty to setup your ssh terminal. See the Appendix for Putty setup instructions.

```
ssh -i tech-summit centos@34.242.223.181
```

## Cloudera and Deloitte Tech Summit - Data Science Labs

Download the Cloudera Data Science Workbench service descriptor:

```
wget
```

```
https://archive.cloudera.com/cdsw1/1.4.2/csd/CLLOUDERA\_DATA\_SCIENCE\_WORKBENCH-CDH5-1.4.2.jar
```

Put the file at the right location with the right attributes:

```
sudo mv CLOUDERA_DATA_SCIENCE_WORKBENCH-CDH5-1.4.2.jar /opt/cloudera/csd
sudo chmod 644
/opt/cloudera/csd/CLLOUDERA_DATA_SCIENCE_WORKBENCH-CDH5-1.4.2.jar
sudo chown cloudera-scm:cloudera-scm
/opt/cloudera/csd/CLLOUDERA_DATA_SCIENCE_WORKBENCH-CDH5-1.4.2.jar
```

Restart the Cloudera Manager Service:

```
sudo service cloudera-scm-server restart
```

Open the Cloudera Manager and restart the Management Service:

The screenshot shows the Cloudera Manager web interface. At the top, there are navigation tabs for Home, Status, All Health Issues (05), Configuration (11), and All Recent Commands. Below the navigation is a warning message: "You are running Cloudera Manager in non-production mode, which uses an embedded PostgreSQL database. Switch to using a supported external database before moving into production. MC".

The main content area is divided into two sections. On the left, there is a list of services under the heading "EDH for TechSu... (CDH 5.15.1, Parcels)". The "MGMT" service is selected, and a "MGMT Actions" menu is open, showing options: Start, Stop, Restart (highlighted with a red circle), Instances, Configuration, Add Role Instances, Rename, and Delete. Below the menu, there is a "View Maintenance Mode Status" button.

On the right, there are several monitoring charts:

- Cluster CPU:** A line chart showing CPU usage across the cluster. The y-axis is labeled "percent" and ranges from 0 to 100. The x-axis shows time from 10:45 to 11 AM. The legend indicates "EDH for TechSummit, Host CPU Usage Across ..." with a value of 60.5%.
- Cluster Disk IO:** A line chart showing disk I/O. The y-axis is labeled "bytes / second" and ranges from 0 to 391K/s. The x-axis shows time from 10:45 to 11 AM. The legend indicates "Total Disk Byte... 273b/s" and "Total Disk Byte... 382K/s".
- Cluster Network IO:** A line chart showing network I/O. The y-axis is labeled "bytes / second" and ranges from 0 to 97.7K/s. The x-axis shows time from 10:45 to 11 AM. The legend indicates "Total Bytes Re... 80.7K/s" and "Total Bytes Tra... 125K/s".
- HDFS IO:** A line chart showing HDFS I/O. The y-axis is labeled "bytes / second" and ranges from 0 to 50b/s. The x-axis shows time from 10:45 to 11 AM. The legend indicates "Total Bytes Read ... 1b/s" and "Total Bytes Wri... 95.2b/s".
- Completed Impala Queries:** A line chart showing the number of completed queries. The y-axis is labeled "queries / second" and ranges from 0 to 1. The x-axis shows time from 10:45 to 11 AM. The legend indicates "IMPALA-1, Total Queries Across Impala Daemons 0".

Step 4: Install the Cloudera Data Science Workbench Parcel

From the Cloudera Manager page, navigate to parcels, select the 'CDSW' parcel, and hit the Download button.



# Cloudera and Deloitte Tech Summit - Data Science Labs

cloudera MANAGER Clusters Hosts Diagnostics Audits Charts Backup Administration

Parcels Parcel Usage Configuration Check for New Parcels

Location: EDH for TechSummit (Available Remotely)

Filters: ERROR STATUS (Error: 0), PARCEL NAME (Clear)

Parcel Name	Version	Status	
CDSW	1.4.2.p1.624065	Available Remotely	Download

ACCUMULO: 2, CDH 5: 1, CDSW: 1, KAFKA: 2, KUDU: 1

When the parcel is downloaded, distribute it to your cluster, and then activate it. This will take a couple of minutes.

EDH for TechSummit

Parcel Name	Version	Status	
CDSW	1.4.2.p1.624065	Distributed, Activated	Deactivate

Step 4: Install the Cloudera Data Science Workbench on your Edge node

Now, install the Cloudera Data Science Workbench on the Edge node of your cluster. From your Cloudera Manager console, select the 'Add Service' action next to your cluster name.

cloudera MANAGER Clusters Hosts Diag

Home Status All Health Issues Configuration 12

You are running Cloudera Manager in non-production mode, which uses

EDH for TechSu... (CDH 5.15.1, Parcels) [Dropdown]

- 5 Hosts
- FLUME-1
- Add Service
- Add Hosts

Select the 'Cloudera Data Science Workbench service, and hit 'Continue'.

# Cloudera and Deloitte Tech Summit - Data Science Labs

## Add Service to EDH for TechSummit

Select the type of service you want to add.

Service Type	Description
<input type="radio"/> ADLS Connector	The ADLS Connector service provides key management for accessing Azure Data Lake Stores from CDH services.
<input type="radio"/> Accumulo	The Apache Accumulo sorted, distributed key/value store is a robust, scalable, high performance data storage and retrieval system. This service only works with releases based on Apache Accumulo 1.6 or later.
<input checked="" type="radio"/> Cloudera Data Science Workbench	Cloudera Data Science Workbench enables fast, easy, and secure self-service data science for the enterprise.

[Back](#) [Continue](#)

Set the HDFS, Spark 2, YARN-1 and ZOOKEEPER-1 as the dependencies and hiet 'Continue'.

## Add Cloudera Data Science Workbench Service to EDH for TechSummit

Select the set of dependencies for your new Cloudera Data Science Workbench

HBase	HDFS	Hive	Solr	Spark 2	Spark	YARN (MR2 Included)	ZooKeeper
<input checked="" type="radio"/>	HDFS-1			Spark 2		YARN-1	ZOOKEEPER-1
<input type="radio"/> HBASE-1	HDFS-1	HIVE-1	SOLR-1	Spark 2	SPARK_ON_YARN-1	YARN-1	ZOOKEEPER-1
<input type="radio"/> HBASE-1	HDFS-1		SOLR-1	Spark 2		YARN-1	ZOOKEEPER-1
<input type="radio"/> HBASE-1	HDFS-1	HIVE-1		Spark 2	SPARK_ON_YARN-1	YARN-1	ZOOKEEPER-1
<input type="radio"/> HBASE-1	HDFS-1			Spark 2		YARN-1	ZOOKEEPER-1
<input type="radio"/>	HDFS-1		SOLR-1	Spark 2		YARN-1	ZOOKEEPER-1

[Back](#) 1 2 3 4 5 6 [Continue](#)

Specify your cluster Edge node (the only node with 31.3 GB mem) as your CDSW Master node.

# Cloudera and Deloitte Tech Summit - Data Science Labs

1 Host Selected

Host	IP	OS	Roles	Memory	Roles	
1.compute.internal	ip-10-0-0-220.eu-west-1.compute.internal	10.0.0.220	/default	2	7.6 GiB	H M B NN SNN HMS HS2 HS HS ICS ISS LHBI
1.compute.internal	ip-10-0-0-243.eu-west-1.compute.internal	10.0.0.243	/default	4	15.5 GiB	M OS SS HS HS JHS RM S
<input checked="" type="checkbox"/>	ip-10-0-0-66.eu-west-1.compute.internal	10.0.0.66	/default	8	31.3 GiB	A H G G G KB G G G DC G M
1.compute.internal	ip-10-0-0-75.eu-west-1.compute.internal	10.0.0.75	/default	4	15.5 GiB	H RS DN ID TS NM

Displaying 1 - 5 of 5

Cancel **OK**

We have a small cluster, so we will not specify any CDSW worker nodes. Hit 'Continue'.

cloudera MANAGER Support admin

### Add Cloudera Data Science Workbench Service to EDH for TechSummit

Assign Roles for Cloudera Data Science Workbench

You can customize the role assignments for your new service here, but note that if assignments are made incorrectly, such as assigning too many roles to a single host, performance will suffer.

You can also view the role assignments by host: **View By Host**

Master x 1 New

Worker

**Continue**

Specify the following CDSW configuration parameters. Replace the tag <public-ip-cluster-edge-node> with the value of the public ip address of the edge node of your cluster. You can find this on your EC2 dashboard.

Cloudera Data Science Workbench Domain:	cdsw.<public-ip-cluster-edge-node>.nip.io
Master Nod IPv4 Address:	<public-ip-cluster-edge-node>
Install Required Packages:	selected
Docker Block Device:	/dev/xvdf /dev/xvdg

Hit the 'Continue' button when you have specified the values.

# Cloudera and Deloitte Tech Summit - Data Science Labs

The screenshot shows the Cloudera Manager interface for configuring a new service. The page title is "Add Cloudera Data Science Workbench Service to EDH for TechSummit". Under the "Review Changes" section, there are four configuration items:

- Cloudera Data Science Workbench Domain** (DOMAIN): Set to "cdsw.34.254.201.95.nip.io".
- Master Node IPv4 Address** (MASTER\_IP): Set to "34.254.201.95".
- Install Required Packages** (cdsw.install.required.packages): Checked for "Cloudera Data Science Workbench (Service-Wide)".
- Docker Block Device** (DOCKER\_BLOCK\_DEVICES): Set to "Docker Daemon Default Group" with two entries: "/dev/xvdf" and "/dev/xvdg".

At the bottom of the page, there is a "Back" button on the left and a "Continue" button on the right, which is circled in red. A progress indicator shows 5 steps, with the current step highlighted in yellow.

Cloudera Manager will now deploy CDSW to the edge node and start it. This will take several minutes.

From Cloudera Manager, select the newly created "Cloudera Data Science Workbench" service and monitor starting of the service.

The screenshot shows the Cloudera Manager monitoring page for the "Cloudera Data Science Workbench" service. The page includes a navigation bar with "Status", "Instances", "Configuration", "Commands", "Charts Library", "Audits", "CDSW Web UI", and "Quick Links".

**Health Tests:** Shows "CDSW Status" with the message "Cloudera Data Science Workbench is starting..." and a "Show 3 Good" link.

**Status Summary:** Shows a summary of the service's status.

**Health History:** A table showing the service's status over time:

Time	Status	Action
1:37:52 PM	CDSW Status Unknown	Show
1:37:22 PM	1 Became Bad 3 Became Good	Show
12:40:46 PM	4 Became Disabled	Show
12:40:41 PM	4 Became Unknown	Show

**Charts:** Two charts are displayed: "Informational Events" and "Important Events and Alerts". Both charts show a spike in events around 01:30.

## 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](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](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>

