

Cloudera Data Science Workbench

Using the Workbench

Date published: 2020-02-28

Date modified:

CLOUDERA

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using the Workbench.....	4
Start a New Session.....	4
Run Code.....	9
Code Autocomplete.....	10
Project Code Files.....	10
Access the Terminal.....	10
Stop a Session.....	10
Workbench editor file types.....	11
Jupyter Magic Commands.....	11
Python.....	12
Scala.....	12

Using the Workbench

The workbench console provides an interactive environment tailored for data science, supporting R, Python and Scala.

The Workbench currently supports R, Python, and Scala engines. You can use these engines in isolation, as you would on your laptop, or connect to your CDH cluster using Cloudera Distribution of Apache Spark 2 and other libraries.

The workbench UI includes four primary components:

- An editor where you can edit your scripts.
- A console where you can track the results of your analysis.
- A command prompt where you can enter commands interactively.
- A terminal where you can use a Bash shell.

The screenshot displays the Cloudera Data Science Workbench interface. On the left, a file explorer shows a project named 'Python Template Project' with files 'analysis.py', 'README.md', and 'seaborn-data'. The main editor shows a Python script named 'analysis.py' with the following code:

```

2 # -----
3
4 import pandas as pd
5 import seaborn as sns
6 # -----
7 # Basic Data Manipulation
8 # -----
9
10 # Use the seaborn tips dataset to generate a best fitting linear regression line
11
12 tips = sns.load_dataset("tips")
13 sns.set(font="DejaVu Sans")
14 sns.jointplot("total_bill", "tip", tips, kind="reg").fig.suptitle("Tips Regression", y=
15
16 # Examine the difference between smokers and non smokers
17 sns.lmplot("total_bill", "tip", tips, col="smoker").fig.suptitle("Tips Regression - cat
18
19 # Explore the dataframe in grid output
20 pd.options.display.html.table_schema = True
21 tips.head()
22
23 # Using IPython's Rich Display System
24 # -----
25 #
26 # IPython has a [rich display system](bit.ly/HHP0ac) for
27 # interactive widgets.
28
29 from IPython.display import IFrame
30 from IPython.core.display import display
31
32 # Define a google maps function.
33 def gmaps(query):
34     url = "https://maps.google.com/maps?q={}&output=embed".format(query)
35     display(IFrame(url, "788px", "458px"))
36
37 gmaps("Golden Gate Bridge")
38
39 # Worker Engines
40 # -----
41 #
42 # You can launch worker engines to distribute your work across a cluster.
43 # Uncomment the following to launch two workers with 2 cpu cores and 0.5GB
44 # memory each.
45
46 # import cdsw
47 # workers = cdsw.launch_workers(n=2, cpu=0.2, memory=0.5, code="print 'Hello from a CDS
48

```

On the right, a terminal window titled 'Python Template Session' shows the execution of the script. The terminal output includes the same code as the editor, followed by the execution of the jointplot function. Below the code, a data visualization titled 'Tips Regression' is displayed. The visualization consists of a scatter plot of 'total_bill' vs 'tip' with a blue regression line, a marginal histogram of 'total_bill' on the top, and a marginal histogram of 'tip' on the right. The Pearson correlation coefficient is shown as $\text{pearsonr} = 0.68; p = 6.7e-34$.

Annotations in the image point to the 'Project file system' (file explorer), 'Terminal access to running engine' (terminal window), and 'Interactive command prompt' (the prompt at the bottom of the terminal).

Typically, you would use the following steps to run a project in the workbench:

Start a New Session

The first step to run a project in the workbench is to start a new session.

About this task



Note: Shell start up files are not run during session start up. CDSW sessions are not bash shells, so shell start up files such as `bashrc`, `zsh`, and `ksh` are not run. If you want to set an environment variable, you can use the CDSW environment variables feature. This will ensure the environment variable is injected in all contexts: sessions, terminals, experiments, models, jobs, etc. If you want to run more complicated code during startup (for example, conditional statements), consider using `PYTHONSTARTUP`, see [Startup.py](#) or `Rprofile`, see [Managing R with .Rprofile](#), [.Renviron](#), [Rprofile.site](#), [Renviron.site](#), [rsession.conf](#), and [repos.conf](#).

Procedure

1. Navigate to your project's Overview page.
2. Click New Session.

The information presented on this page will depend on which default engine you have chosen for your project: Runtime or Legacy Engines. You can change the default engine later in this task.

The **Start A New Session** dialog box opens.

3. Check the settings for your session:

If your project is using ML Runtimes, you will see the following settings:

Start A New Session

Not authenticated to Hadoop
Before you can connect to your secure Hadoop cluster, you must enter your credentials under [Settings > Hadoop Authentication](#)

Session Name

Untitled Session

Runtime

Editor ⓘ Please select one ▼

Kernel ⓘ Please select one ▼

Edition ⓘ Please select one ▼

Version Please select one ▼

Runtime Image

Resource Profile

1 vCPU / 2 GiB Memory ▼

Cancel Start Ses

Editor

Selects the Editor; currently only Workbench is supported and therefore the selector is static.

Kernel

Selects the Kernel, for example Python 3.7, R4.0.

Edition

Selects the Runtime Edition. Initially only Standard variants are supported.

Version

Selects the ML Runtimes version.



Note: The selector options only consider the configurations supported by the actual deployments and certain selections will automatically limit others. For example, certain versions are only relevant for Python or certain editors are supported only with certain kernels.

If your project is using Legacy Engines, you will see the following settings:

Start A New Session



Not authenticated to Hadoop

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

Session Name

Engine

Editor ⓘ**Kernel** ⓘ

Engine Image - [Configure](#) Base Image v13 - docker.repository.cloudera.com/cdsw/engine:13

Resource Profile

Editor

Selects the Editor; currently only Workbench is supported and therefore the selector is static.

Kernel

Selects the Kernel. Initially only Python Runtimes are supported.

Engine Image

Selects the engine image. Click Configure to display the Project Setting > Advanced window to modify your environment variables and shared memory limit.

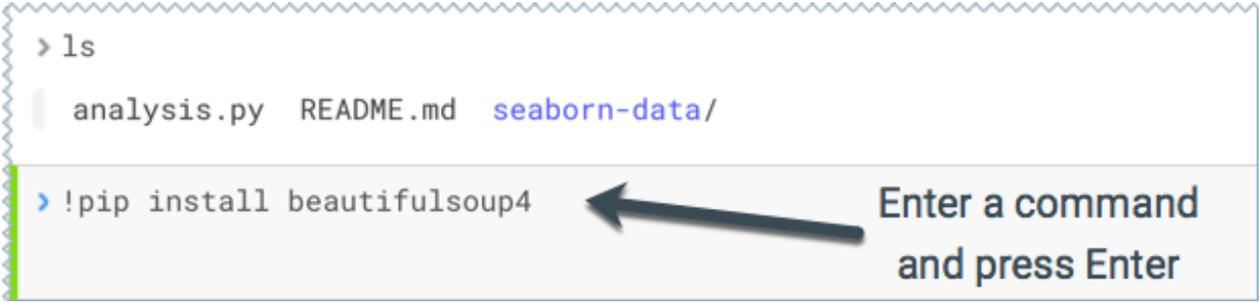
4. If your project is using Legacy Engines, you can modify the engine image used by this session:
 - a) By Engine Image, click Configure.
Cloudera Machine Learning displays the Project Settings page.
 - b) Select the Runtime/Engine tab.
 - c) Next to Default Engine, select ML Runtime or Legacy Engine.
 - d) Click Save Engine.
5. Specify your Resource Profile.
This attribute will define how many vCPUs and how much memory will be reserved to run the workload (for example, session including the runtime itself). The minimum configuration is 1vCPU and 2 GB memory.
6. Click Start Session.
The command prompt at the bottom right of your browser window will turn green when the engine is ready. Sessions typically take between 10 and 20 seconds to start.

Run Code

You can enter and run code at the command prompt or the editor. The editor is best for code you want to keep, while the command prompt is best for quick interactive exploration.

About this task

Command Prompt - The command prompt functions largely like any other. Enter a command and press Enter to run it. If you want to enter more than one line of code, use Shift+Enter to move to the next line. The output of your code, including plots, appears in the console.



```
> ls
analysis.py  README.md  seaborn-data/
> !pip install beautifulsoup4
```

Enter a command and press Enter

If you created your project from a template, you should see project files in the editor. You can open a file in the editor by clicking the file name in the file navigation bar on the left.

Editor - To run code from the editor:

Procedure

1. Select a script from the project files on the left sidebar.
- 2.

To run the whole script click  on the top navigation bar, or, highlight the code you want to run and press Ctrl+Enter (Windows/Linux) or cmd+Enter (macOS).

When doing real analysis, writing and executing your code from the editor rather than the command prompt makes it easy to iteratively develop your code and save it along the way.

If you require more space for your editor, you can collapse the file list by double-clicking between the file list pane and the editor pane. You can hide the editor using editor's View menu.

- Alternatively you can stop a session by typing the following command:

R

```
quit()
```

Python

```
exit
```

Scala

```
quit()
```

Sessions automatically stop after an hour of inactivity.

Workbench editor file types

The default workbench editor supports the following file types:

- Text
- CSS
- HTML
- JavaScript
- JSON
- PHP
- Scala
- C++
- C#
- CLike
- Java
- CoffeeScript
- R
- Julia
- Ruby
- Clojure
- Perl
- Python
- SASS
- Lua
- SQL
- Diff
- Markdown
- YAML
- Haxe

Jupyter Magic Commands

Cloudera Data Science Workbench's Scala and Python kernels are based on Jupyter kernels. Jupyter kernels support varying magic commands that extend the core language with useful shortcuts. This section details the magic commands (magics) supported by Cloudera Data Science Workbench.



Note: Jupyter magic commands apply only to legacy engine projects.

Line magics begin with a single %: for example, %timeit. Cell magics begin with a double %%: for example, %%bash.

Python

The examples below show how to retrieve the password from an environment variable and use it to connect.

You can access data using [pyodbc](#) or [SQLAlchemy](#).

```
# pyodbc lets you make direct SQL queries.
!wget https://pyodbc.googlecode.com/files/pyodbc-3.0.7.zip
!unzip pyodbc-3.0.7.zip
!cd pyodbc-3.0.7;python setup.py install --prefix /home/cdsw
import os

# See http://www.connectionstrings.com/ for information on how to construct
  ODBC connection strings.
db = pyodbc.connect("DRIVER={PostgreSQL Unicode};SERVER=localhost;PORT=54
32;DATABASE=test_db;USER=cdswuser;OPTION=3;PASSWORD=%s" % os.environ["POSTGR
ESQL_PASSWORD"])
cursor = cnxn.cursor()
cursor.execute("select user_id, user_name from users")

# sqlalchemy is an object relational database client that lets you make data
base queries in a more Pythonic way.
!pip install sqlalchemy
import os
import sqlalchemy
from sqlalchemy.orm import sessionmaker
from sqlalchemy import create_engine
db = create_engine("postgresql://cdswuser:%s@localhost:5432/test_db" % os.
environ["POSTGRESQL_PASSWORD"])
session = sessionmaker(bind=db)
user = session.query(User).filter_by(name='ed').first()
# python-oracledb can be used to connect directly to Oracle databases witho
ut need to install oracle drivers
# See https://python-oracledb.readthedocs.io/en/latest/user_guide/installa
tion.html#quickstart
!pip install oracledb
import oracledb
import os

un = os.environ.get('PYTHON_USERNAME')
pw = os.environ.get('PYTHON_PASSWORD')
cs = os.environ.get('PYTHON_CONNECTSTRING')
with oracledb.connect(user=un, password=pw, dsn=cs) as connection:
    with connection.cursor() as cursor:
        sql = """select sysdate from dual"""
        for r in cursor.execute(sql):
            print(r)
```

Scala

Cloudera Data Science Workbench's Scala kernel is based on Apache Toree.

It supports the line magics documented in the Apache Toree [magic tutorial](#).