

## Experiments

Date published: 2020-02-28

Date modified: 2020-12-15

# CLOUDERA

# 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

<b>Experiments.....</b>	<b>4</b>
Purpose.....	4
Concepts.....	4
Running an Experiment (Quick Start).....	5
Tracking Metrics.....	8
Saving Files.....	8
Disabling the Experiments Feature.....	8
Limitations.....	9
Debugging Issues with Experiments.....	9

## 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).

Demo: Watch the following video for a quick demonstration of the steps described in this topic: [Experiments with Cloudera Data Science Workbench](#)

## Purpose

As data scientists iteratively develop models, they often experiment with datasets, features, libraries, algorithms, and parameters.

### Challenge

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.

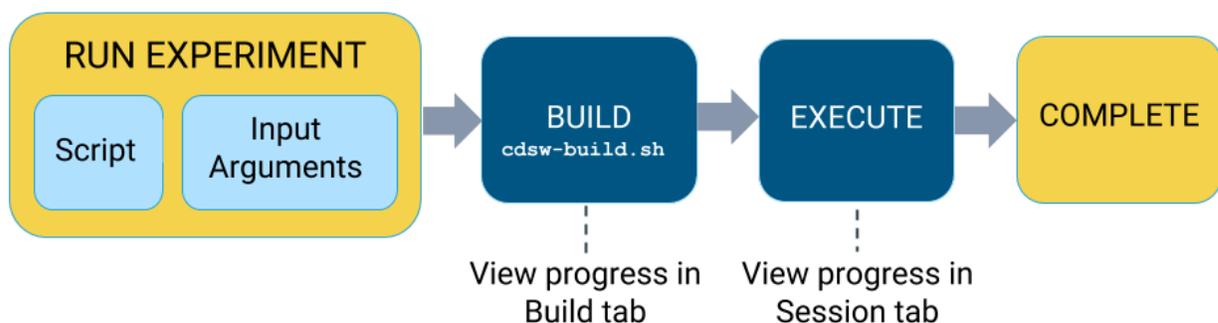
### Solution

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



## Running an Experiment (Quick Start)

The following steps describe how to launch an experiment from the Workbench console. In this example we are going to run a simple script that adds all the numbers passed as arguments to the experiment.

### Procedure

1. Go to the project Overview page.
2. Click Open Workbench.
3. Create/modify any project code as needed. You can also launch a session to simultaneously test code changes on the interactive console as you launch new experiments.

As an example, you can run this Python script that accepts a series of numbers as command-line arguments and prints their sum.

add.py

```
import sys
import cdsw

args = len(sys.argv) - 1
sum = 0
x = 1

while (args >= x):
    print ("Argument %i: %s" % (x, sys.argv[x]))
    sum = sum + int(sys.argv[x])
    x = x + 1

print ("Sum of the numbers is: %i." % sum)
```

4. To test the script, launch a Python session and run the following command from the workbench command prompt:

```
!python add.py 1 2 3 4
```

5. Click Run Experiment. If you're already in an active session, click **Run Run Experiment** . Fill out the following fields:

- Script - Select the file that will be executed for this experiment.
- Arguments - If your script requires any command line arguments, enter them here.



**Note:** Arguments are not supported with Scala experiments.

- Engine Kernel and Resource Profile - Select the kernel and computing resources needed for this experiment.



**Note:** The list of options here is specific to the default engine you have specified in your Project Settings: ML Runtimes or Legacy Engines. Engines allow kernel selection, while ML Runtimes allow Editor, Kernel, Variant, and Version selection. Resource Profile list is applicable for both ML Runtimes and Legacy Engines.

For this example we will run the add.py script and pass some numbers as arguments.

### Run New Experiment ✕

**Script**

**Arguments** ?

**Engine Kernel**

Python 2

Python 3

Scala

R

**Engine Profile**

**Comment**

Cancel Start Run

6. Click Start Run.

- To track progress for the run, go back to the project Overview. On the left navigation bar click Experiments. You should see the experiment you've just run at the top of the list. Click on the Run ID to view an overview for each individual run. Then click Build.

On this Build tab you can see realtime progress as Cloudera Data Science Workbench builds the Docker image for this experiment. This allows you to debug any errors that might occur during the build stage.

```

Run-4 New Experiment

Overview Session Build

Sending build context to Docker daemon 14.34 MB

Step 1/5 : FROM docker.repository.cloudera.com/cdsw/engine:5
----> da62f78351e0
Step 2/5 : COPY sources /home/cdsw
----> dc0d62362524
Removing intermediate container 52edb8ecef1f
Step 3/5 : WORKDIR /home/cdsw
----> 604125f1bd47
Removing intermediate container 934e47683660
Step 4/5 : RUN chown -R 8536:8536 /home/cdsw
----> Running in 80097e84eb11

```

- Once the Docker image is ready, the run will begin execution. You can track progress for this stage by going to the Session tab.

For example, the Session pane output from running `add.py` is:

```

Run-4 New Experiment

Overview Session Build

> import sys
> import cdsw
> args = len(sys.argv) - 1
> sum = 0
> x = 1
> while (args >= x):
  print ("Parameter %i: %s" % (x, sys.argv[x]))
  sum = sum + int(sys.argv[x])
  x = x + 1
Parameter 1: 18
Parameter 2: 90
Parameter 3: 34
> print ("Sum of the numbers is: %i." % sum)
Sum of the numbers is: 142.

```

- (Optional) The `cdsw` library that is bundled with Cloudera Data Science Workbench includes some built-in functions that you can use to compare experiments and save any files from your experiments.

For example, to track the sum for each run, add the following line to the end of the `add.py` script.

```
cdsw.track_metric("Sum", sum)
```

This will be tracked in the Experiments table:

Run	Script	Arguments	Kernel	Comment	Submitter	Created At	Sum	Status	Duration
12	add.py	1 2 3 4 5	python3		admin	6/11/18 1:48 PM		Scheduling	
11	add.py	66 42 97 104 2004	python3		admin	6/11/18 1:44 PM	2313	Success	0 mins
10	add.py	4 60 72	python3		admin	6/11/18 1:43 PM	136	Success	0 mins
9	add.py	22 96 78	python3		admin	6/11/18 1:43 PM	196	Success	0 mins

## Tracking Metrics

The `cdsw` library includes a `track_metric` function that can be used to log up to 50 metrics associated with a run, thus allowing accuracy and scores to be tracked over time.

The function accepts input in the form of key value pairs.

```
cdsw.track_metric(key, value)
```

Python

```
cdsw.track_metric("R_squared", 0.79)
```

R

```
cdsw::track.metric("R_squared", 0.62)
```

These metrics will be available on the project's Experiments tab where you can view, sort, and filter experiments on the values. The table on the Experiments page will allow you to display only three metrics at a time. You can select which metrics are displayed from the metrics dropdown.



**Note:** This function is not supported with Scala experiments.

## Saving Files

Cloudera Data Science Workbench allows you to select which artifacts you'd like to access and evaluate after an experiment is complete. These artifacts could be anything from a text file to an image or a model that you have built through the run.

The `cdsw` library includes a `track_file` function that can be used to specify which artifacts should be retained after the experiment is complete.

Python

```
cdsw.track_file('model.pkl')
```

R

```
cdsw::track.file('model.pkl')
```

Specified artifacts can be accessed from the run's Overview page. These files can also be saved to the top-level project filesystem and downloaded from there.



**Note:** This function is not supported with Scala experiments.

## Disabling the Experiments Feature

### Before you begin

Required Role: Site Administrator

### About this task



**Important:** The feature flag mentioned here only hides the Experiments feature from the UI. It will not stop any experiments that have already been queued for execution.

To disable this feature on your Cloudera Data Science Workbench deployment:

### Procedure

1. Log in to Cloudera Data Science Workbench.
2. Click Admin Settings .
3. Under the Feature Flags section, disable the Enable users to run experiments. checkbox.

## Limitations

Experiments have the following limitations.

- Experiments do not store snapshots of project files. You cannot automatically restore code that was run as part of an experiment.
- Experiments will fail if your project filesystem is too large for the Git snapshot process. As a general rule, any project files (code, generated model artifacts, dependencies, etc.) larger than 50 MB must be part of your project's .gitignore file so that they are not included in snapshots for experiment builds.
- Experiments cannot be deleted. As a result, be conscious of how you use the `track_metrics` and `track_file` functions.
  - Do not track files larger than 50MB.
  - Do not track more than 100 metrics per experiment. Excessive metric calls from an experiment may cause Cloudera Data Science Workbench to stop responding.
- The Experiments table will allow you to display only three metrics at a time. You can select which metrics are displayed from the metrics dropdown. If you are tracking a large number of metrics (100 or more), you might notice some performance lag in the UI.
- Arguments are not supported with Scala experiments.
- The `track_metrics` and `track_file` functions are not supported with Scala experiments.
- The UI does not display a confirmation when you start an experiment or any alerts when experiments fail.

## Debugging Issues with Experiments

This topic lists some common issues to watch out for during an experiment's build and execution process.

### Experiment spends too long in Scheduling/Built stage

If your experiments are spending too long in any particular stage, check the resource consumption statistics for the cluster. When the cluster starts to run out of resources, often experiments (and other entities like jobs, models) will spend too long in the queue before they can be executed.

Resource consumption by experiments (and jobs, sessions) can be tracked by site administrators on the Admin Activity page.

### Experiment fails in the Build stage

During the build stage Cloudera Data Science Workbench creates a new Docker image for the experiment. You can track progress for this stage on each experiment's Build page. The build logs on this page should help point you in the right direction.

Common issues that might cause failures at this stage include:

- Lack of execute permissions on the build script itself.
- Inability to reach the Python package index or R mirror when installing packages.
- Typo in the name of the build script (`cdsw-build.sh`). Note that the build process will only execute a script called `cdsw-build.sh`; not any other bash scripts from your project.

- Using pip3 to install packages in cdsw-build.sh, but selecting a Python 2 kernel when you actually launch the experiment. Or vice versa.

**Experiment fails in the Execute stage**

Each experiment includes a Session page where you can track the output of the experiment as it executes. This is similar to the output you would see if you test the experiment in the workbench console. Any runtime errors will display on the Session page just as they would in an interactive session.