Cloudera Data Science Workbench

# Managing Jobs and Pipelines in Cloudera Data Science Workbench

**Date published: 2020-02-28**
**Date modified:**

## CLOUDƎRA

# Legal Notice

# Contents

# Managing Jobs and Pipelines in Cloudera Data Science Workbench

Cloudera Data Science Workbench allows you to automate analytics workloads with a built-in job and pipeline scheduling system that supports real-time monitoring, job history, and email alerts.

A job automates the action of launching an engine, running a script, and tracking the results, all in one batch process. Jobs are created within the purview of a single project and can be configured to run on a recurring schedule. You can customize the engine environment for a job, set up email alerts for successful or failed job runs, and email the output of the job to yourself or a colleague.

As data science projects mature beyond ad hoc scripts, you might want to break them up into multiple steps. For example, a project may include one or more data acquisition, data cleansing, and finally, data analytics steps. For such projects, Cloudera Data Science Workbench allows you to schedule multiple jobs to run one after another in what is called a pipeline, where each job is dependent on the output of the one preceding it.

## Creating a Job

Jobs are created within the scope of a project. When you create a job, you will be asked to select a script to run as part of the job, and create a schedule for when the job should run. Optionally, you can configure a job to be dependent on another existing job, thus creating a pipeline of tasks to be accomplished in a sequence.

### About this task

**Note:** The script files and any other job dependencies must exist within the scope of the same project.

### Procedure

1. Navigate to the project for which you want to create a job.
2. On the left-hand sidebar, click Jobs.
3. Click New Job.
4. Enter a Name for the job.
5. Select a script to run for this job by clicking on the folder icon. You will be able to select a script from a list of files that are already part of the project. To upload more files to the project, see Managing Files.
6. (Optional) Specify command-line arguments that are needed by the scripts that are running within your job in the Arguments field.

   **Note:** This feature is supported only for scripts written in Python and R. It is not supported for Scala.

7. Depending on the code you are running, select an Engine Kernel for the job from one of the following options: Python 2, Python 3, R, or Scala.

   **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 section, while ML Runtimes allow Editor, Kernel, Variant, and Version selection. Resource Profile list is applicable for both ML Runtimes and Legacy Engines.

8. Select a Schedule for the job runs from one of the following options.

   - Manual - Select this option if you plan to run the job manually each time.
   - Recurring - Select this option if you want the job to run in a recurring pattern every X minutes, or on an hourly, daily, weekly or monthly schedule. Set the recurrence interval with the drop-down buttons.

     As an alternative, select Use a cron expression to enter a Unix-style cron expression to set the interval. The expression must have five fields, specifying the minutes, hours, day of month, month, and day of week. If the cron expression is deselected, the schedule indicated in the drop-down settings takes effect.
   - Dependent - Use this option when you are building a pipeline of jobs to run in a predefined sequence. From a dropdown list of existing jobs in this project, select the job that this one should depend on. Once you have configured a dependency, this job will run only after the preceding job in the pipeline has completed a successful run.

9. Select an Engine Profile to specify the number of cores and memory available for each session.

10. Enter an optional timeout value in minutes.

11. To override the overall project environment variables, under Environmental Variables enter the name and value of your new variable and click Add.

    You can also delete an existing environment variable by selecting it and clicking Delete.

12. Specify a list of Job Report Recipients to whom you can send email notifications with detailed job reports for job success, failure, or timeout. You can send these reports to yourself, your team (if the project was created under a team account), or any other external email addresses.

13. Add any Attachments such as the console log to the job reports that will be emailed.

14. Click Create Job.

    Starting with version 1.1.x, you can use the Jobs API to schedule jobs from third partly workflow tools. For details, see Cloudera Data Science Workbench Jobs API.

# Creating a Pipeline

The Jobs overview presents a list of all existing jobs created for a project along with a dependency graph to display any pipelines you've created. Job dependencies do not need to be configured at the time of job creation.

## About this task

Pipelines can be created after the fact by modifying the jobs to establish dependencies between them. From the job overview, you can modify the settings of a job, access the history of all job runs, and view the session output for individual job runs.

Let's take an example of a project that has two jobs, Read Weblogs and Write Weblogs. Given that you must read the data before you can run analyses and write to it, the Write Weblogs job should only be triggered after the Read Weblogs job completes a successful run. To create such a two-step pipeline:

## Procedure

1. Navigate to the project where the Read Weblogs and Write Weblogs jobs were created.

2. Click Jobs.

3. From the list of jobs, select Write Weblogs.

4. Click the Settings tab.

5. Click on the Schedule dropdown and select Dependent. Select Read Weblogs from the dropdown list of existing jobs in the project.

6. Click Update Job.

## Viewing Job History

This topic describes how to view your job history.

### Procedure

1. Navigate to the project where the job was created.
2. Click Jobs.
3. Select the relevant job.
4. Click the History tab. You will see a list of all the job runs with some basic information such as who created the job, run duration, and status. Click individual runs to see the session output for each run.

# Cloudera Data Science Workbench Jobs API

Cloudera Data Science Workbench exposes a REST API that allows you to schedule jobs from third-party workflow tools. You must authenticate yourself before you can use the API to submit a job run request. The Jobs API supports HTTP Basic Authentication, accepting the same users and credentials as Cloudera Data Science Workbench.

## API Key Authentication

Cloudera recommends using your API key for requests instead of your actual username/password so as to avoid storing and sending your credentials in plaintext. The API key is a randomly generated token that is unique to each user. It must be treated as highly sensitive information because it can be used to start jobs via the API.

### About this task

To look up your Cloudera Data Science Workbench API key:

### Procedure

1. Sign in to Cloudera Data Science Workbench.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click Settings.
4. Select the API Key tab.

   The following example demonstrates how to construct an HTTP request using the standard basic authentication technique. Most tools and libraries, such as Curl and Python Requests, support basic authentication and can set the required Authorization header for you. For example, with curl you can pass the API Key to the --user flag and leave the password field blank.

   ```
   curl -v -XPOST http://cdsw.example.com/api/v1/<path_to_job> --user
    "<API_KEY>:"
   ```

   To access the API using a library that does not provide Basic Authentication convenience methods, set the request's Authorization header to Basic *<API_KEY_encoded_in_base64>* . For example, if your API key is uysgxtj7jzkps96njextnxxmq05usp0b, set Authorization to Basic dXlzZ3h0ajdqemtwczk2bmpleHRueHhtcTA1dX NwMGI6.

# Starting a Job Run Using the API

Once a job has been created and configured through the Cloudera Data Science Workbench web application, you can start a run of the job through the API. This will constitute sending a POST request to a job start URL of the form: http ://cdsw.example.com/api/v2/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start.

### Before you begin

To construct a request, use the following steps to derive the username, project name, and job ID from the job's URL in the web application.

### Procedure

1. Log in to the Cloudera Data Science Workbench web application.
2. Switch context to the team/personal account where the parent project lives.
3. Select the project from the list.
4. From the project's Overview, select the job you want to run.

   This will take you to the job Overview page. The URL for this page is of the form: http://cdsw.example.com/< $USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>.

5. Use the $USERNAME, $PROJECT_NAME, and $JOB_ID parameters from the job Overview URL to create the following job start URL:

```
http://cdsw.example.com/api/v2/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<
$JOB_ID>/start
```

For example, if your job Overview page has the URL http://cdsw.example.com/alice/sample-project/jobs/123, then a sample POST request would be of the form:

```
curl -v -XPOST http://cdsw.example.com/api/v2/projects/alice/sample-proj
ect/jobs/123/start     \
--user "<API_KEY>:" --header "Content-type: application/json"
```

```
{
    "environment": {
        "ENV_VARIABLE": "value 1",
        "ANOTHER_ENV_VARIABLE": "value 2"
    }
}
```

**Note:** The request must have the Content-Type header set to application/json, even if the request body is empty.

# Setting Environmental Variables

You can set environment variables for a job run by passing parameters in the API request body in a JSON-encoded object with the following format.

### About this task

You can set environment variables for a job run by passing parameters in the API request body in a JSON-encoded object with the following format.

```
{
    "environment": {
        "ENV_VARIABLE": "value 1",
        "ANOTHER_ENV_VARIABLE": "value 2"
    }
```

```
}
```

The values set here will override the defaults set for the project and the job in the web application. This request body is optional and can be left blank.

Be aware of potential conflicts with existing defaults for environment variables that are crucial to your job, such as PATH and the CDSW * variables.

## Sample Job Run

This topic provides a sample job run.

As an example, let's assume user Alice has created a project titled Risk Analysis. Under the Risk Analysis project, Alice has created a job with the ID, 208. Using curl, Alice can use her API Key (uysgxtj7jzkps96njextnxxmq05u sp0b) to create an API request as follows:

```
curl -v -XPOST http://cdsw.example.com/api/v2/projects/alice/risk-analysis/j
obs/208/start \
--user "uysgxtj7jzkps96njextnxxmq05usp0b:" --header "Content-type: applicati
on/json" \
--data "{\"environment\": {\"START_DATE\": \"2017-01-01\", \"END_DATE\":
\"2017-01-31\"}}"
```

In this example, START_DATE and END_DATE are environment variables that are passed as parameters to the API request in a JSON object.

In the resulting HTTP request, curl automatically encodes the Authorization request header in base64 format.

```
* Connected to cdsw.example.com (10.0.0.3) port 80 (#0)
* Server auth using Basic with user 'uysgxtj7jzkps96njextnxxmq05usp0b'
> POST /api/v2/projects/alice/risk-analysis/jobs/21/start HTTP/1.1
> Host: cdsw.example.com
> Authorization: Basic dXlzZ3h0ajdqemtwczk2bmpleHRueHhtcTA1dXNwMGI6
> User-Agent: curl/7.51.0
> Accept: */*
> Content-type: application/json
>
< HTTP/1.1 200 OK
< Access-Control-Allow-Origin: *
< Content-Type: application/json; charset=utf-8
< Date: Mon, 10 Jul 2017 12:00:00 GMT
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
<
{
    "engine_id": "cwg6wclmg0x482u0"
}
```

You can confirm that the job was started by going to the Cloudera Data Science Workbench web application.

# Starting a Job Run Using Python

To start a job run using Python, Cloudera recommends using Requests, an HTTP library for Python; it comes with a convenient API that makes it easy to submit job run requests to Cloudera Data Science Workbench.

Extending the Risk Analysis example from the previous section, the following sample Python code will create an HTTP request to run the job with the job ID, 208.

Python 2

```
# example.py

import requests
import json

HOST = "http://cdsw.example.com"
USERNAME = "alice"
API_KEY = "uysgxtj7jzkps96njextnxxmq05usp0b"
PROJECT_NAME = "risk-analysis"
JOB_ID = "208"

url = "/".join([HOST, "api/v1/projects", USERNAME, PROJECT_NAME, "jobs",
JOB_ID, "start"])
job_params = {"START_DATE": "2017-01-01", "END_DATE": "2017-01-31"}
res = requests.post(
    url,
    headers = {"Content-Type": "application/json"},
    auth = (API_KEY,""),
    data = json.dumps({"environment": job_params})
)

print "URL", url
print "HTTP status code", res.status_code
print "Engine ID", res.json().get('engine_id')
```

When you run the code, you should see output of the form:

```
python example.py
```

```
URL http://cdsw.example.com/api/v1/projects/alice/risk-analysis/jobs/208/sta
rt
HTTP status code 200
Engine ID r11w5q3q589ryg9o
```