

# Managing Jobs and Pipelines in Cloudera Machine Learning

Date published: 2020-07-16

Date modified: 2024-07-31



# 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>Creating a Job.....</b>	<b>4</b>
<b>Creating a Pipeline.....</b>	<b>5</b>
<b>Viewing Job History.....</b>	<b>6</b>
<b>Legacy Jobs API (Deprecated).....</b>	<b>6</b>

## Creating a Job

This topic describes how 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.

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

### For CML UI

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. In Run Job as, if the job is to run in a service account, select Service Account and choose the account from the dropdown menu.
6. In Script, 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 Project Files*.
7. In Arguments, enter arguments to provide to the script.

Use the environment variable `JOB_ARGUMENTS` to access the arguments in a runtime-agnostic way.

For non-PBJ Runtimes, the contents of this field are available as standard command line arguments.

8. Depending on the code you are running, select an Engine Kernel for the job from one of the following option: Python 3.
9. 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.

10. Select an Resource Profile to specify the number of cores and memory available for each session.
11. Enter an optional timeout value in minutes.
12. Click Set environment variables if you want to set any values to override the overall project environment variables.
13. 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.
14. Add any Attachments such as the console log to the job reports that will be emailed.

**15. Click Create Job.**

You can use the API v2 to schedule jobs from third party workflow tools. For details, see *Using the Jobs API* as well as the *CML APIv2* tab.

**For CML APIv2**

To create a job using the API, follow the code below:

```
job_body = cmlapi.CreateJobRequest()

# name and script
job_body.name = "my job name"
job_body.script = "pi.py"

# arguments
job_body.arguments = "arg1 arg2 \"all arg 3\""

# engine kernel
job_body.kernel = "python3" # or "r", or "scala"

# schedule
# manual by default
# for recurring/cron:
job_body.schedule = "* * * * 5" # or some valid cron string

# for dependent (don't set both parent_job_id and schedule)
job_body.parent_job_id = "abcd-1234-abcd-1234"

# resource profile (cpu and memory can be floating point for partial)
job_body.cpu = 1 # one cpu vcore
job_body.memory = 1 # one GB memory
job_body.nvidia_gpu = 1 # one nvidia gpu, cannot do partial gpus

# timeout
job_body.timeout = 300 # this is in seconds

# environment
job_body.environment = {"MY_ENV_KEY": "MY_ENV_VAL", "MY_SECOND_ENV_KEY":
"MY_SECOND_ENV_VAL"}

# attachment
job_body.attachments = ["report/1.txt", "report/2.txt"] # will attach /
home/cdsd/report/1.txt and /home/cdsd/report/2.txt to emails

# After setting the parameters above, create the job:
client = cmlapi.default_client("host", "api key")
client.create_job(job_body, project_id="id of project to create job in")
```

For some more examples of commands related to jobs, see: *Using the Jobs API*.

**Related Information**

[Managing Project Files](#)

[Using the Jobs API](#)

[Legacy Jobs API \(Deprecated\)](#)

## Creating a Pipeline

This topic describes how to create a scheduled pipeline of jobs within a project.

### About this task

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 Machine Learning 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.

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. 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 shows you how to view the history for jobs run within a project.

### 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.

## Legacy Jobs API (Deprecated)

This topic demonstrates how to use the legacy API to launch jobs.

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



**Note:** The Jobs API is now deprecated. See *CML API v2* and *API v2* usage for the successor API.

## Legacy API Key Authentication

Cloudera recommends using your legacy API key for requests instead of your actual username/password so as to avoid storing and sending your credentials in plaintext. The legacy 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. To look up your Cloudera Machine Learning legacy API key:

1. Sign in to Cloudera Machine Learning.
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 legacy 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
"<LEGACY_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 `<LEGACY_API_KEY_encoded_in_base64>`. For example, if your API key is uysgxtj7jzkps96njextnxxmq05usp0b, set Authorization to Basic `dXlzM3h0ajdqemtwczk2bmpleHRueHhtcTA1dXNwMG` `I6`.

## Starting a Job Run Using the API

Once a job has been created and configured through the Cloudera Machine Learning web application, you can start a run of the job through the legacy API. This will constitute sending a POST request to a job start URL of the form: `http://cdsw.example.com/api/v1/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start`.

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.

1. Log in to the Cloudera Machine Learning 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/v1/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/v1/projects/alice/sample-project/jobs/123/start \
--user "<API_KEY>:" --header "Content-type: application/json"
```

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

## Setting Environment 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.

```
{
```

```

    "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 CML variables.

### 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 (uysgxtj7jzkps96njextnxxmq05usp0b) to create an API request as follows:

```

curl -v -XPOST http://cdsw.example.com/api/v1/projects/alice/risk-analysis/jobs/208/start \
--user "uysgxtj7jzkps96njextnxxmq05usp0b:" --header "Content-type: application/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/v1/projects/alice/risk-analysis/jobs/21/start HTTP/1.1
> Host: cdsw.example.com
> Authorization: Basic dXlzMzZ3h0ajdqemtwczk2bnpIeHRueHhtcTAldXNwMGIG
> 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": "cw6wclmg0x482u0"
}

```

You can confirm that the job was started by going to the Cloudera Machine Learning 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 Machine Learning. Extending the Risk Analysis example from the previous section, the following sample Python code creates 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/start
HTTP status code 200
Engine ID r1lw5q3q589ryg9o
```

### Limitations

- Cloudera Machine Learning does not support changing your legacy API key, or having multiple API keys.
- Currently, you cannot create a job, stop a job, or get the status of a job using the Jobs API.

### Related Information

[API v2 usage](#)

[Basic Access Authentication](#)

[Creating a Pipeline](#)

[Environment Variables](#)

[CML API v2](#)