

Accessing the Cloudera Data Engineering service using the API

Date published: 2020-07-30

Date modified: 2024-11-12



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 Cloudera Data Engineering API.....	4
Getting a Cloudera Data Engineering API access token.....	4
Using an access token in Cloudera Data Engineering API calls.....	5
Managing Cloudera Data Engineering job resources using the API.....	6
Creating a Cloudera Data Engineering resource using the API.....	6
Deleting a Cloudera Data Engineering resource using the API.....	7
Using custom operators and libraries for Apache Airflow using API.....	7
Adding or updating custom operators and libraries using API.....	8
Deleting custom operators and libraries using API.....	10
Troubleshooting custom operators and libraries.....	10
Viewing logs for custom operators and libraries using API.....	10
Cancel Maintenance using API.....	11
Managing workload secrets with Cloudera Data Engineering Spark Jobs using the API.....	11
Creating a workload secret for Cloudera Data Engineering Spark Jobs using API.....	11
Listing an existing workload secret for Cloudera Data Engineering Spark Jobs using API.....	12
Deleting a workload secret for Cloudera Data Engineering Spark Jobs using API.....	13
Linking a workload secret to the Cloudera Data Engineering Spark Job definitions using API.....	13
Using the workload secret in the Spark application code.....	14
Creating a Cloudera Data Engineering job using the API.....	15
Listing Cloudera Data Engineering jobs using the API.....	16
Getting Cloudera Data Engineering job information using the API.....	16
Adding a database and a Storage Account deployed in private DNS zones using the API.....	17
Managing VC-level Spark configurations using the API.....	18

Using the Cloudera Data Engineering API

Cloudera Data Engineering (CDE) provides a robust API for integration with your existing continuous integration/continuous delivery platforms.

The Cloudera Data Engineering service API is documented in Swagger. You can view the API documentation and try out individual API calls by accessing the API DOC link in any virtual cluster:

1. In the Cloudera Data Platform (CDP) console, click the Data Engineering tile. The CDE Home page displays.
2. In the Virtual Clusters section, click the Cluster Details icon in any of the listed virtual clusters.
3. Click API DOC.

Related Information

[Using CLI-API to Automate Access to Cloudera Data Engineering](#)

[Using Cloudera Data Engineering CLI](#)

Getting a Cloudera Data Engineering API access token

Cloudera Data Engineering uses JSON Web Tokens (JWT) for API authentication. To interact with a virtual cluster using the API, you must obtain an access token for that cluster.

Before you begin



Important: The user interface for CDE 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view CDE Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Determine the authentication endpoint for your virtual cluster:

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera Data Platform (CDP) management console.
2. In the CDE Services column, select the environment containing the virtual cluster you want to interact with.
3. In the Virtual Clusters column on the right, click the Cluster Details icon on the virtual cluster you want to interact with.
4. Click the link under GRAFANA CHARTS. The hostname of the URL in your browser is the base URL, and /gateway/authtoken/knoxtoken/api/v1/token is the endpoint. For example:

```
https://service.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/gateway/authtoken/knoxtoken/api/v1/token
```

Procedure

1. From the client you want to use to access the API, run `curl -u <workload_user> <auth_endpoint>`. Enter your workload password when prompted.

For example:

```
curl -u csso_psherman https://service.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/gateway/authnkn/knoxtoken/api/v1/token
```

The user account is your [CDP workload user](#).

2. In the output, the `access_token` value is the JWT. For convenience, copy it and set it as an environment variable:

```
export CDE_TOKEN=<access_token>
```

Alternatively, you can set the token in a single step using `jq` to extract the token:

```
export CDE_TOKEN=$(curl -s -u <workload_user> <auth_endpoint> | jq -r '.access_token')
```

What to do next

See [Using an access token in Cloudera Data Engineering API calls](#) for instructions on using the token in API calls.

Using an access token in Cloudera Data Engineering API calls

Cloudera Data Engineering (CDE) uses JSON Web Tokens (JWT) for API authentication.

Before you begin

Get an access token and save it as an environment variable as described in [Getting a Cloudera Data Engineering API access token](#).



Important: The user interface for CDE 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view CDE Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API. The API URL for managing jobs is different from the URL used to obtain the access token.
 - a) In the Cloudera Data Platform (CDP) console, click the Data Engineering tile and click Overview
 - b) In the CDE Services column, select the environment containing the virtual cluster you want to interact with using the API.
 - c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to interact with using API.
 - d) Click JOBS API URL to copy the link to your clipboard.
For example: `https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1`

2. When you make an API call, include the JWT as a bearer token. For example, to list all jobs associated with the virtual cluster, assuming you have saved the token as an environment variable named CDE_TOKEN:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -H "Content-Type: application/json" -X GET "https://pmjkrng5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/jobs"
```

Managing Cloudera Data Engineering job resources using the API

A *resource* in Cloudera Data Engineering (CDE) is a named collection of files referenced by a job. The files can include application code, configuration files, and even Python virtual environment specifications (requirements.txt). These resources can be managed using the CDE API.

Creating a Cloudera Data Engineering resource using the API

A *resource* in Cloudera Data Engineering (CDE) is a named collection of files referenced by a job. The files can include application code, configuration files, and even Python virtual environment specifications (requirements.txt). You can create a resource using the CDE API.

Before you begin

As with all API calls, make sure you have a valid access token. For instructions, see [Getting a Cloudera Data Engineering API access token](#).

Procedure

1. Create a JSON file describing the resource, using the structure for the type of resource you want to create.

The JSON payload for a resource is structured as follows:

files resource type

```
{
  "name": "<resourceName>",
  "type": "files",
  "retentionPolicy": "keep_indefinitely"
}
```

python-env resource type

```
{
  "name": "<resourceName>",
  "type": "python-env",
  "retentionPolicy": "keep_indefinitely",
  "pythonEnv": {
    "pythonVersion": "python3"
  }
}
```

custom-runtime-image resource type

```
{
  "customRuntimeImage": {
    "credential": "string",
    "engine": "string",
    "image": "string",
  }
}
```

```

    "modified": "string"
  },
  "name": "<resourceName>",
  "retentionPolicy": "keep_indefinitely",
  "type": "custom-runtime-image"
}

```

2. Create the resource by submitting a POST request to the resources endpoint. The JSON filename is referenced using the @/path/to/filename.json convention. In this example, the JSON filename is fileResource.json, and describes a files type resource named example-job-files.

```

curl <jobs_api_url>/resources
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "Content-Type: application/json" \
-X POST -d @${HOME}/fileResource.json

```

3. Verify that the resource was creating by requesting the resource details from the /resources/<resourceName> endpoint:

```

curl <jobs_api_url>/example-job-files \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "Content-Type: application/json" \
-X GET

```

Deleting a Cloudera Data Engineering resource using the API

A *resource* in Cloudera Data Engineering (CDE) is a named collection of files referenced by a job. The files can include application code, configuration files, and even Python virtual environment specifications (requirements.txt). You can delete a resource using the CDE API.

Before you begin

As with all API calls, make sure you have a valid access token. For instructions, see [Getting a Cloudera Data Engineering API access token](#).

Procedure

1. Make sure that you no longer need the resource before deleting it.
2. Delete the resource by sending a DELETE request to the /resources/<resourceName> endpoint.
For example, to delete a resource named example-job-files:

```

curl <jobs_api_url>/example-job-files \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "Content-Type: application/json" \
-X DELETE

```

Using custom operators and libraries for Apache Airflow using API

You can install and use custom python packages for Airflow with Cloudera Data Engineering (CDE). Cloudera provides access to the open source packages that you can use for your Airflow jobs using the CDE API.

Related Information

[Using custom operators and libraries for Apache Airflow using CDE UI](#)

Adding or updating custom operators and libraries using API

You can add or update custom python packages for Airflow with Cloudera Data Engineering (CDE). Cloudera provides access to the open source packages that you can use for your Airflow jobs using the API.

About this task

While you can install the operator, if additional runtime dependencies are required such as additional setup with binaries on the path, environment configuration like Kerberos and Cloud credentials, and so on, then the operator will not work.



Important: You must be in a maintenance session to set up a custom Python environment for Airflow. You can cancel a maintenance session before or after any of these steps in the workflow. After the maintenance session is cancelled, all objects related to the session are deleted.

1. Start a maintenance session by calling the `/admin/airflow/env/maintenance`. It is a synchronous call and the session is created upon success.

```
curl -X POST \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

2. Create and define the pip repositories by calling the `/admin/airflow/env/maintenance/repos`. This is a sync operation.

The JSON payload to create a job is structured as follows:

```
{
  "extraPipRepositories": [
    {
      "caCerts": "string",
      "credential": {
        "password": "string",
        "username": "string"
      },
      "skipCertValidation": false,
      "url": "string"
    }
  ],
  "pipRepository": {
    "caCerts": "string",
    "credential": {
      "password": "string",
      "username": "string"
    },
    "skipCertValidation": false,
    "url": "string"
  }
}
```

For example:

```
curl -X POST \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/repos" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json" \
  -H "Content-Type: application/json" \
  -d '{
```



```

"pipRepository": {
  "url": "https://pypi.org/simple/"
}
},
curl -X GET \
"https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/status" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"

```

3. Build the Python environment by calling `/admin/airflow/env/maintenance/build`.

- a. To monitor results, issue `GET /admin/airflow/env/maintenance/status`.

Required build starting states	Success state	Failure state
pip-repos-defined	built	build-failed
built		
build-failed		

The payload to create a job for a multi-part/form input is as follows:

```

# requirements.txt should be a file on the local file system and a well
# formatted python requirements.txt

curl -X POST \
"https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/build" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json" \
--form file="@requirements.txt"

curl -X GET \
"https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/status" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"

```

4. Activate the Python environment by calling `/admin/airflow/env/maintenance/activate`.

- a. To monitor results, issue `GET /admin/airflow/env/maintenance/status`.

Required activation starting states	Success state	Failure state
built	N/A, request should fail with 404, since maintenance is done	activation-failed
activation-failed	<code>/admin/airflow/env/status</code> should have the “activated” status and the description of your environment should be correct and updated	

For example:

```

curl -X POST \
"https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/activate" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"

```

```
curl -X GET \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/status" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

You can now use your DAGs with custom code.

Deleting custom operators and libraries using API

If you no longer need your custom operators and libraries, you can reset the runtime environment to CDE default environment by deleting the currently active Python environment. To reset the CDE Airflow Python environment, complete the following steps.



Note: Ensure that no jobs are using the packages which are meant to be deleted, otherwise they will stop working in the Airflow instance. Also, no other maintenance operations like delete operation, updating the python environment is allowed while the delete operation is in progress.

To delete an environment, call DELETE on /admin/airflow/env:

```
curl -X DELETE \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

To ensure that the environment has been successfully deleted, the currently active environment status must be checked and the request must fail with a 404. This operation may take a few minutes. This indicates that the environment is not currently in use. For example:

```
curl -X GET \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

Troubleshooting custom operators and libraries

Learn the different ways that you can troubleshoot the setup of your Airflow Python environment in Cloudera Data Engineering (CDE).

Viewing logs for custom operators and libraries using API

You can view the logs of a maintenance session and logs of an environment. You can view the maintenance logs when a maintenance is ongoing and if fails at some step. You can view logs of an active environment if some jobs behave unexpectedly on an active Airflow Python environment.

View logs of a maintenance session:



Note: Logs are cumulative and would contain all actions done within a maintenance session. Therefore, each attempt log (upload retry) will end up in the logs.

```
curl -X GET \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/logs" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

View the logs of an environment:

```
curl -X GET \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/logs" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

Cancel Maintenance using API

You cancel the maintenance of an Airflow Python environment setup and all the objects related to the session will be discarded and deleted.

To cancel a maintenance, call DELETE on /admin/airflow/env/maintenance:

```
curl -X DELETE \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

To ensure that the maintenance has been cancelled, the maintenance status must be checked and the request must fail with a 404. This may take seconds or up to a few minutes. This indicates that no maintenance session is ongoing. For example:

```
curl -X GET \
  "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/admin/airflow/env/maintenance/status" \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  -H "accept: application/json"
```

Managing workload secrets with Cloudera Data Engineering Spark Jobs using the API

This API provides a secure way to create and store workload secrets for Cloudera Data Engineering (CDE) Spark Jobs. This is a more secure alternative to storing credentials in plain text embedded in your application or job configuration.

Creating a workload secret for Cloudera Data Engineering Spark Jobs using API

Creating a workload secret for Cloudera Data Engineering (CDE) Spark Jobs using API provides a secure way to create workload secrets. This is a more secure alternative to storing credentials in plain text embedded in your application or job configuration.

Procedure

1. Create Workload Credentials with name workload-cred:-

```
curl -k -X "POST" 'https://<dex-vc-host>/dex/api/v1/credentials' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  --data '{
```

```
"workloadCred": {
  "aws-secret": "secret123",
  "db-pass": "dbpass123"
},
"name": "workload-cred-1",
"type": "workload-credential",
"description": "workload credential description"
}'
```

2. Edit the existing workload credentials with new data. For example, edit workload-cred-1:

```
curl -k -X "PATCH" 'https://<dex-vc-host>/dex/api/v1/credentials/workload-cred-1' \
-H 'Accept: application/json' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${CDE_TOKEN}" \
--data '{
  "workloadCred": {
    "aws-secret": "newsecret123",
    "db-pass": "newdbpass123"
  }
}'
```

Listing an existing workload secret for Cloudera Data Engineering Spark Jobs using API

You can list an existing secret for Cloudera Data Engineering (CDE) Spark Jobs using API.

Procedure

1. List the existing workload credentials in the virtual cluster by creating a filter with type that is equal to workload-credential

```
curl -k -X "GET" 'https://<dex-vc-host>/dex/api/v1/credentials?filter=type%5Bcredential%5D' \
-H 'Accept: application/json' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${CDE_TOKEN}"
```

Response:

```
{
  "credentials": [
    {
      "name": "workload-cred-1",
      "type": "workload-credential",
      "description": "workload credential description",
      "created": "2022-10-18T07:26:41Z",
      "modified": "2022-10-18T07:26:41Z"
    }
  ],
  "meta": {
    "hasNext": false,
    "limit": 20,
    "offset": 0,
    "count": 1
  }
}
```

}

2. Get the existing specific workload credential. For example, workload-cred-1:

```
curl -k -X "GET" 'https://<dex-vc-host>/dex/api/v1/credentials/workload-cred-1' \
-H 'Accept: application/json' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${CDE_TOKEN}"
```

Response:

```
{
  "name": "workload-cred-1",
  "type": "workload-credential",
  "description": "workload credential description",
  "created": "2022-10-18T07:26:41Z",
  "modified": "2022-10-18T07:26:41Z"
}
```

Deleting a workload secret for Cloudera Data Engineering Spark Jobs using API

You can delete an existing secret for Cloudera Data Engineering (CDE) Spark Jobs using API.

Procedure

Delete the workload-credential. For example, delete workload-cred-1:

```
curl -k -X "DELETE" 'https://<dex-vc-host>/dex/api/v1/credentials/workload-cred-1' \
-H 'Accept: application/json' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${CDE_TOKEN}"
```

Linking a workload secret to the Cloudera Data Engineering Spark Job definitions using API

Once you've created a workload secret, you can link it to the Cloudera Data Engineering (CDE) Spark Job definitions that you created using API.

Procedure

1. Use 1 or more workload-credentials in a new Spark job. For example, workload-cred-1 and workload-cred-2 in the job below:

```
curl -k -X "POST" 'https://<dex-vc-host>/dex/api/v1/jobs' \
-H 'Accept: application/json' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${CDE_TOKEN}" \
--data '{
  "name": "test-workload-job",
  "spark": {
    "className": "org.apache.spark.examples.SparkPi",
```

```

    "file": "spark-examples_2.11-2.4.8.7.2.12.0-195.jar"
  },
  "mounts": [
    {
      "resourceName": "spark-jar",
      "dirPrefix": ""
    }
  ],
  "type": "spark",
  "workloadCredentials": ["workload-cred-1", "workload-cred-2"]
}'

```

2. Use 1 or more workload-credentials in an existing Spark job. For example, workload-cred-1 and workload-cred-2 in the job below:

```

curl -k -X "PATCH" 'https://<dex-vc-host>/dex/api/v1/jobs/test-workload-
job' \
-H 'Accept: application/json' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${CDE_TOKEN}" \
--data '{
  "workloadCredentials": ["workload-cred-1", "workload-cred-2"]
}'

```

Using the workload secret in the Spark application code

To use the workload secret credentials, you can read the credentials that are mounted into the Spark drivers and executors as read-only files.

Procedure

The workload secrets are mounted into the Spark drivers and executors in this path: `/etc/dex/secrets/<workload-credential-name>/<credential-key-1>` `/etc/dex/secrets/<workload-credential-name>/<credential-key-2>`

Example workload credentials:

The workload-credential was created with the payload below.

```

{
  "workloadCred": {
    "aws-secret": "secret123",
    "db-pass": "dbpass123"
  },
  "name": "workload-cred-1",
  "type": "workload-credential",
  "description": "workload credential description"
}

```

The secrets can be read as local files from the paths below within the Spark drivers and executors:

`/etc/dex/secrets/workload-cred-1/aws-secret`

`/etc/dex/secrets/workload-cred-1/db-pass`

Example of a Pyspark application code to read a secret:

```

from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Sample DB Connection") \

```

```
.getOrCreate()

# read the password from the local file
dbPass=open("/etc/dex/secrets/workload-cred-1/db-pass").read()

# use the password in a jdbc connection
jdbcDF= spark.read \
```

Creating a Cloudera Data Engineering job using the API

You can create a job in Cloudera Data Engineering (CDE) using the CDE jobs API endpoint.

Before you begin

Request an access token and save it as an environment variable to use in API calls. For instructions, see [Getting a Cloudera Data Engineering API access token](#).



Important: The user interface for CDE 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view CDE Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

If you're using secrets or if you want to access workload secrets, refer to [Managing workload secrets with Cloudera Data Engineering using the API](#) linked below.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API:
 - a) In the Cloudera Data Platform (CDP) console, click the Data Engineering tile and click Overview
 - b) In the CDE Services column, select the environment containing the virtual cluster you want to interact with using the API.
 - c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to interact with.
 - d) Click JOBS API URL to copy the URL.
For example: `https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1`
2. Submit the job creation request using the API to the `/jobs` endpoint.
The JSON payload to create a job is structured as follows:

```
{
  "name": "demoJob",
  "spark": {
    "className": "com.example.demoJobMainClass",
    "file": "local:/path/to/demoJobJar"
  },
  "type": "spark"
}
```

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" <jobs_api_url>/jobs \
-H "Content-Type: application/json" \
```

```
-X POST -d '{"name":"demoJob","spark":{"className":"com.example.demoJobMainClass","file":"local:/path/to/demoJobJar"},"type":"spark"}'
```

3. Verify the job was created. You can view job details using the `/jobs/<jobName>` endpoint:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -H "Content-Type: application/json" -X GET "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/jobs/demoJob"
```

Listing Cloudera Data Engineering jobs using the API

You can list Cloudera Data Engineering (CDE) jobs using the API by issuing a GET request to the `/jobs` endpoint.

Before you begin

Get an access token and save it as an environment variable as described in [Getting a Cloudera Data Engineering API access token](#).



Important: The user interface for CDE 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view CDE Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API:
 - a) In the Cloudera Data Platform (CDP) console, click the Data Engineering tile and click Overview
 - b) In the CDE Services column, select the environment containing the virtual cluster you want to interact with using the API.
 - c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to interact with.
 - d) Click JOBS API URL to copy the URL.
For example: `https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1`
2. Issue a GET request to the `/jobs` endpoint:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -X GET "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/jobs"
```

Getting Cloudera Data Engineering job information using the API

You can get Cloudera Data Engineering (CDE) job information using the API by issuing a GET request to the `/jobs/<job_name>` endpoint.

Before you begin

Get an access token and save it as an environment variable as described in [Getting a Cloudera Data Engineering API access token](#).



Important: The user interface for CDE 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view CDE Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API:
 - a) In the Cloudera Data Platform (CDP) console, click the Data Engineering tile and click Overview
 - b) In the CDE Services column, select the environment containing the virtual cluster you want to interact with using the API.
 - c) Click JOBS API URL to copy the URL.
For example: `https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1`
2. Issue a GET request to the `/jobs/<jobname>` endpoint:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -X GET "https://pmjkrn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/jobs/SparkPi"
```

Configuring an existing private DNS zone for a database and a Storage Account File Share using the API (Technical Preview)

Learn about how to configure existing private DNS zones for a database and a Storage Account File Share using the API.

Before you begin

To learn about the CLI option and about the prerequisites:

- for the database, see [Configuring an existing private DNS zone for a database](#)
- for the Storage Account File Share, see [Configuring an existing private DNS zone for a Storage Account File Share](#)

Procedure

To configure the private DNS zones for the database and the Storage Account File Share, specify the full resource ID of the private DNS zones where you want to configure them.

To configure the Azure private DNS zone IDs for the database and the Storage Account File Share, in `config.properties` of the cluster endpoint, post the full resource IDs in the following parameters respectively:

- For the database, use `azure.database.privateDNSZoneId`
- For the Storage Account, use `azure.fileshare.privateDNSZoneId`

Request example

Example for the base URL, which changes according to the region: <https://console.us-west-1.cdp.cloudera.com>

```
curl -H "Cookie: cdp-session-token=${CST}" '[***BASE-URL***]/dex/api/v1/cluster' \
-H "Content-Type: application/json" \
-H 'accept: application/json' \
-X POST -d '{
  "name": "[***NAME***]",
  "env": "[***ENVIRONMENT-NAME***]",
  "config": {
    "properties": {
      "loadbalancer.internal": "true",
      "dbus-wxm-client.enabled": "false",
      "kubernetes.api.allowList": "",
      "loadbalancer.allowList": "",
      "kubernetes.api.proxyCIDRList": "",
      "subnets": "",
      "loadBalancerSubnets": "",
      "default-vc.create": "true",
      "cde.deployPreviousVersion": "false",
      "privateNetwork.enabled": "true",
      "azure.fileshare.privateDNSZoneId": "[***AZURE-PRIVATEDNSZONE-RESOURCE-ID-FOR-STORAGE-ACCOUNT-FILE-SHARE***]",
      "azure.database.privateDNSZoneId": "[***AZURE-PRIVATEDNSZONE-RESOURCE-ID-FOR-DATABASE***]"
    },
    "resources": {
      "instance_type": "Standard_D8s_v4",
      "min_spot_instances": "0",
      "max_spot_instances": "0",
      "min_instances": "0",
      "max_instances": "50",
      "initial_instances": "0",
      "initial_spot_instances": "0",
      "root_vol_size": "100",
      "allp_instance_group_details": {
        "instance_type": "Standard_D8s_v4",
        "min_spot_instances": "0",
        "max_spot_instances": "0",
        "min_instances": "0",
        "max_instances": "50",
        "initial_instances": "0",
        "initial_spot_instances": "0",
        "root_vol_size": "100"
      }
    },
    "tags": {}
  },
  "skipValidation": false
}'
```

Managing Virtual Cluster-level Spark configurations using the API (Technical Preview)

Learn about how to manage Virtual Cluster (VC)-level Spark configurations using the API.

Before you begin

For information about managing Virtual Cluster-level Spark configurations using the CLI, see [Managing Virtual Cluster-level Spark configurations](#).

Creating a VC instance with VC-level Spark configurations

On the control plane side, to create a Virtual Cluster (VC) instance with VC-level Spark configurations, run the following command.



Note: The Spark configurations defined in the `sparkConfigs` field apply to all the Spark jobs within the VC.

```
curl -H "Cookie: cdp-session-token=${CST}" '[***BASE-URL***]/dex/api/v1/cluster/[***CLUSTER-ID***]/instance' \
-H "Content-Type: application/json" \
-H 'accept: application/json' \
-X POST -d '{
  "name": "vc-spark-configs-cli",
  "config": {
    "sparkConfigs": {
      "spark.executor.instances": "4",
      "is.config.vc": "true",
      "is.config.job": "false"
    },
    "resources": {
      "cpu_requests": "20",
      "mem_requests": "80Gi"
    }
  }
}'
```

Example for the base URL, which changes according to the region: <https://console.us-west-1.cdp.cloudera.com>

Payload for creating the Spark configurations for a VC

```
{
  "name": "vc-spark-configs-cli",
  "config": {
    "sparkConfigs": {
      "spark.executor.instances": "4",
      "is.config.vc": "true",
      "is.config.job": "false"
    },
    "resources": {
      "cpu_requests": "20",
      "mem_requests": "80Gi"
    }
  }
}
```

If successful, the 200, OK response is received.

Updating the Spark configurations for a VC instance

On the control plane side, to update the Spark configurations for a VC instance, run the following command.



Note: The Spark configurations defined in the `sparkConfigs` field override the existing `sparkConfigs`.

```
curl -H "Cookie: cdp-session-token=${CST}" '[***BASE-URL***]/dex/api/v1/cluster/[***CLUSTER-ID***]/instance/[***INSTANCE-ID***]' \
-H "Content-Type: application/json" \
-H 'accept: application/json' \
-X 'PATCH' -d '{
  "config": {
    "sparkConfigs": {
```

```

    "spark.executor.instances": "4",
    "is.config.vc": "true",
    "is.config.job": "false"
  }
},
},

```

Example for the base URL, which changes according to the region: <https://console.us-west-1.cdp.cloudera.com>

Payload for updating the Spark configurations

```

{
  "config": {
    "sparkConfigs": {
      "spark.executor.instances": "4",
      "is.config.vc": "true",
      "is.config.job": "false"
    }
  }
}

```

If successful, the 200, OK response is received.

Getting the Spark configuration information for the Job-runs of a VC instance

On the workload side, to get the Spark configuration information for the Job-runs of a VC instance, run the following command.

In the Job-Runs endpoint, the instanceSparkConfigs field contains the Spark configurations that are applied at VC-level to a particular virtual cluster instance.

1. To get the API URL of the VC you want to access, on the CDE UI, navigate to Administration Virtual Clusters and click the Cluster Details icon of the VC you want to interact with.
2. To copy the URL, click Actions COPY JOBS API URL .

```

curl -X GET '[[**JOBS-API-URL**]]/job-runs/[**JOB-RUN-ID**] ' \
-H 'accept: application/json' \
-H "Cookie: cde-csrf-token=${CSRF}" \
-H "Cookie: hadoop-jwt=${JWT}"

```

In the Spark.Conf field, the compiled list of configurations, which include the configurations applied at VC-level and at Job-level, is added with the default configurations of Spark. The configuration information is sent to Livy, which is a job submitter with precedence. The precedence hierarchy, from highest to lowest priority is: Job-Run, Job, and VC-level configurations.