Cloudera Data Engineering 1.5.5

Managing the Cloudera Data Engineering service using the CLI

Date published: 2020-07-30
Date modified: 2025-06-06

https://docs.cloudera.com/

Legal Notice

Contents

Configuring the CLI client

The CDE CLI client uses a configuration file, ~/.cde/config.yaml, to define the default Cloudera Data Engineering virtual cluster to interact with, as well as other configuration parameters.

Before you begin

Make sure that you have downloaded the CDE CLI client.

Procedure

1. Determine the virtual cluster endpoint URL.
   a) Navigate to the Cloudera Data Engineering Overview page.
   b) In the Environments column, select the environment containing the virtual cluster you want to access using the CLI.
   c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to access.
   d) Click JOBS API URL to copy the URL to your clipboard.

   Note: Currently, the URL copied to your clipboard begins with http://, not https://. To use the URL, you must manually change this to https://.

2. On the host with the CLI client, create or edit the configuration file at ~/.cde/config.yaml. You can create multiple profiles in the ~/.cde/config.yaml file and can be used while running commands. For more information, see Creating and using multiple profiles.

   Note: You can use a custom file location by setting the CDE_CONFIG environment variable. If you include “user: ” in your config.yaml, the CLI tool will not use the access token and will instead keep prompting for a password. If you want to use a credential file, then you must remove the specified user from the profile.

3. In the configuration file, specify the Cloudera user and virtual cluster endpoint as follows:

   ```
   user: <CDP_USER>
   vcluster-endpoint: <CDE_VIRTUAL_CLUSTER_ENDPOINT>
   ```

   Important: The CLI in this release does not support TLS validation. You must disable TLS validation by adding the following lines to the Cloudera Data Engineering configuration file:

   ```
   tls-insecure: true
   ```

   The connection still uses HTTPS, but the TLS certificate is not validated.

   The Cloudera user is your workload username.

4. Save the configuration file.
5. If you have not done so already, make sure that the cde file is executable by running chmod +x */PATH/TO/*cde.
6. Run cde job list to verify your configuration. Enter your workload password when prompted.

   Note: If the directory containing the cde file is not part of your PATH environment variable, you can either add it to your PATH environment variable or use the full path when running the command.

   You can also configure the CLI to use an access token so that you do not need to enter your password each time. For more information, see Cloudera Data Engineering CLI authentication.

What to do next

See CDE CLI configuration options for other configuration options.

Cloudera Data Engineering CLI configuration options

The Cloudera Data Engineering CLI can be configured using a configuration file, environment variables, or by command flags.

Configuration Option	Configuration File (~/.cde/config.yaml)	Environment Variable	Command Flag
User	user: <*USERNAME*>	CDE_USER=<*USER*>	--user <*USERNAME*>
Credentials file	credentials-file: </path/to/credentials>	CDE_CREDENTIALS_FILE=</path/to/credentials>	--credentials-file </path/to/credentials>
Skip credentials file detection	skip-credentials-file: true	CDE_SKIP_CREDENTIALS_FILE=true	--skip-credentials-file
Password file	auth-pass-file: <*PASSWORD_FILE*>	CDE_AUTH_PASS_FILE=<*PASSWORD_FILE*>	--auth-pass-file <*PASSWORD_FILE*>
Virtual cluster endpoint	vcluster-endpoint: <*VIRTUAL_CLUSTER*>	CDE_VCLUSTER_ENDPOINT=<*VIRTUAL_CLUSTER*>	--vcluster-endpoint <*VIRTUAL_CLUSTER*>
Disable authentication token caching	auth-no-cache: true	CDE_AUTH_NO_CACHE=true	--auth-no-cache
Authentication token cache file	auth-cache-file: <*TOKEN_CACHE_FILE*>	CDE_AUTH_CACHE_FILE=<*TOKEN_CACHE_FILE*>	--auth-cach-file <*TOKEN_CACHE_FILE*>
CDE configuration profile		CDE_CONFIG_PROFILE=<*PROFILE_NAME*>	--config-profile <*PROFILE_NAME*>

Creating and using multiple profiles using CDE CLI

You can now add a collection of CDE CLI configurations grouped together as profiles, to the config.yaml file. You can use these profiles while running commands. You can set the configurations either at a profile level or at a global level.

About this task

The CDE CLI client uses ~/.cde/config.yaml configuration file to define the default Cloudera Data Engineering virtual cluster to interact with and to define other configuration parameters. CDE CLI now supports the profiling option in the configuration file. A profile is a collection of configurations that you can apply to a CLI command. Multiple named profiles can be stored in the configuration file.

Before you begin

Make sure that you have downloaded the CLI client. For more information, see *Using the Cloudera Data Engineering command line interface*.

Procedure

1. Create or open the ~/.cde/config.yaml file.
2. Add profiles to the config.yaml file based on the following structure:

```
<Global Configurations>
profiles:
- name: <PROFILE NAME 1>
 <PROFILE CONFIGURATIONS>
- name: <PROFILE NAME 2>
```

```
 <PROFILE CONFIGURATIONS>
```

Example Configuration file:

```
vcluster-endpoint: https://g7f9bnv2.cde.dev.cldr.work/dex/api/v1
user: cdpuser

profiles:
- name: dev
  user: cdpuser1
  vcluster-endpoint: https://y86gbhv3.cde.dev.cldr.work/dex/api/v1

- name: test
  vcluster-endpoint: https://g7f9bnv2.cde8x.dev.cldr.work/dex/api/v1
```

- Global Configuration: These configurations are set at the global level. The configurations set here are used by default when a profile name is not specified or the configuration is not specified in the profile
- Profile Configuration: These configurations are set at the profile level and overrides the respective configurations set at the global level. You can select the profile either by using the environment variables, or command flags.

Note: Any profile mentioned under profiles overrides the keys of global configuration and does not replace all the configurations.

For example, if there are five parameters in global and only three are configured in the selected profile, the final configurations will be three from the selected profile and the remaining from global . In the above test profile example, it does not have the user configuration defined, so if the user selects the test profile, except vcluster-endpoint, which is set in the test profile, all the other configurations are used from global .

Global configuration is accessed as the default profile name. Hence, you cannot create a profile named default .

3. You can select the profile using the flag or environment variable.

   Flag

   ```
   cde job list --config-profile <PROFILE NAME>
   ```

   Environment variable

   ```
   export CDE_CONFIG_PROFILE=<PROFILE NAME>
   cde job list
   ```

4. [Optional] You can view the active profile in the configuration file.

   ```
   cde profile show-active
   ```

Cloudera Data Engineering CLI authentication

The Cloudera Data Engineering CLI tool supports both interactive and transparent authentication. For interactive authentication, if you have configured the CLI with your workload username, you are prompted for a password. For transparent authentication, the CDE CLI supports a password file, Cloudera access keys, and Cloudera credentials file.

The CDE CLI provides the following mechanisms for authentication:

- Cloudera access key stored in a credentials file
- Cloudera access key specified by CLI flag or environment variable
- Interactive prompt for workload password
- Workload password specified by CLI flag or environment variable

In all cases, the CLI uses the provided credentials to obtain an authentication token for the specified user, and caches it locally in a file on the machine where the CLI is running. You can disable caching of tokens entirely by using the --auth-no-cache CLI flag or the CDE_AUTH_NO_CACHE environment variable.

Important: The minimum required roles to obtain an access token are *DEUSER* and *ENVIRONMENTUSER*. *ENVIRONMENTADMIN* role is not required.

The cache file location is automatically determined based on the default system user cache:

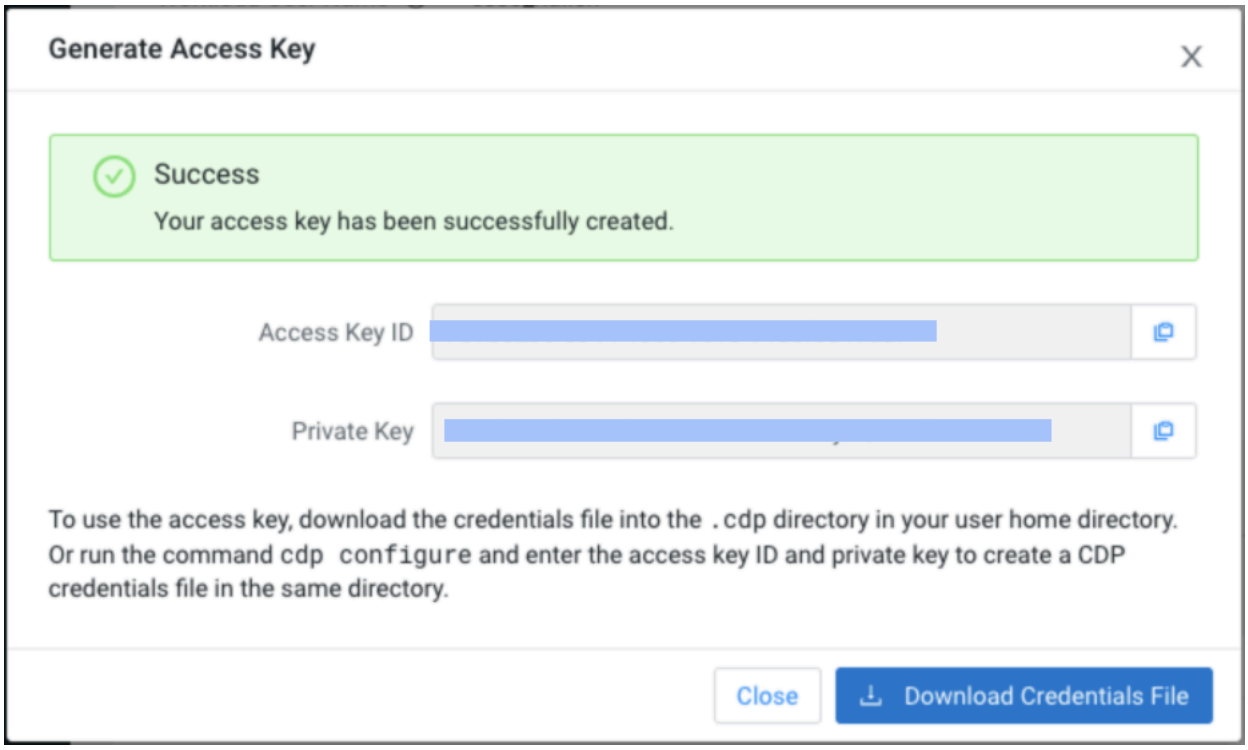
- Linux: $HOME/.cache/cloudera/cde or $XDG_CACHE_HOME/cloudera/cde/
- macOS: $HOME/Library/Caches/cloudera/cde/
- Windows: %LocalAppData%\cloudera\cde\

If you want to use a custom location, specify it with the --auth-cache-file flag or the CDE_AUTH_CACHE_FILE environment variable. You can use the special string $USERCACHE, which is expanded according to the default system user cache (as listed above, without the /cloudera/cde/ suffix).

Note: If you include `user:` in your config.yaml file, the CLI tool will not use the access token and will instead keep prompting for a password. If you want to use a credential file, then you must remove the specified user from the profile.

Cloudera credentials file

When you generate a Cloudera access key, you can download it to a credentials file:



The access key is only displayed and available for download when you first generate it. After you close the dialog, there is no way to recover the key.

Save or copy the credentials file to $HOME/.cdp/credentials on the machine where you are running the CDE CLI. Credentials stored in this file are automatically discovered by both the Cloudera Data Engineering and Cloudera CLIs. If a credentials file is found, authentication occurs transparently using the discovered Cloudera access key.

The CDE CLI automatically looks for a Cloudera access key in the following locations in the order given:

1. ./credentials
2. $HOME/.cde/credentials

3. /etc/cde/credentials
4. $HOME/.cdp/credentials

You can override this by using the --credentials-file </*PATH/TO/CREDENTIALS*> CLI flag to specify a different file location.

You can also skip credential discovery by using the --skip-credentials-file flag.

Cloudera access key

If you do not want to use the credentials file, you can specify the access key using environment variables or command line flags as follows:

Table 1: Cloudera access key environment variables and CLI flags

Parameter	Environment variable	CLI flag
Access key ID	CDE_ACCESS_KEY_ID=<*ACCESS_KEY_ID*>	--access-key-id <*ACCESS_KEY_ID*>
Access key secret	CDE_ACCESS_KEY_SECRET=<*ACCESS_KEY_SECRET*>	--access-key-secret string <*ACCESS_KEY_SECRET*>

Along with the above flags, CDE CLI expects Cloudera endpoint URL to be configured. Cloudera Endpoint URL is same as the Cloudera on premises console URL. You can configure this using environment variables or command line flags as follows:

Table 2: Cloudera endpoint environment variables and CLI flags

Parameter	Environment variable	CLI flag
Cloudera Endpoint	CDE_CDP_ENDPOINT=<*CDP_ENDPOINT*>	--cdp-endpoint <*CDP_ENDPOINT*>

Workload password prompt

When the CLI requires a new token for a virtual cluster, you are prompted for the password for the workload user, identified by the --user CLI flag or the CDE_USER environment variable.

The workload password, for both human and machine users, can be set using the Cloudera User Management console. For more information, see Managing user access and authorization.

Workload password file

If you do not want to be prompted for your workload password, you can provide a password file. A password file is a file containing your workload password, and nothing else.

Note: When using a password file, the CLI strips one trailing newline character. If your password actually includes a newline character at the end, add an extra newline at the end of the file.

You can specify the password file by using an environment variable or a command line flag as follows:

Environment variable

CDE_AUTH_PASS_FILE=</*PATH/TO/PASSWORD/FILE*>

Command line flag

--auth-pass-file </*PATH/TO/PASSWORD/FILE*>

Cloudera Data Engineering CLI TLS configuration

Important: You can choose one of the following:

- If you want to disable TLS validation, add the following lines to the Cloudera Data Engineering configuration file (~/.cde/config.yaml):

```
tls-insecure: true
```

- Else, configure the CDE CLI using TLS with secure mode as per Configure CDE CLI with TLS CA certificates.

All Cloudera Data Engineering virtual cluster endpoints are configured with TLS. In non-production or on-premises environments the TLS certificates are usually signed by a non-production or non-public certificate authority (CA). In these cases, without additional configuration, the CLI tool fails because it attempts to validate the API server's TLS certificate. The CLI provides a TLS configuration when using non-public/non-production CAs.

Specify a file containing the PEM encoded public certificate(s) of the signing CA in one of the following ways:

- add the --tls-ca-certs *[***/PATH/TO/CA.PEM***]* flag on the command line
- define the tls-ca-certs: *[***/PATH/TO/CA.PEM***]* variable in the ~/.cde/config.yaml configuration file
- set the CDE_TLS_CA_CERTS environment variable

Replace *[***/PATH/TO/CA.PEM***]* with the path to a valid ca.pem file.

For on cloud, certificates are issued and signed by LetsEncrypt:

Note: LetsEncrypt Production CA Chain is part of the standard CA bundle therefore you do not need to add it on Linux or macOS. It is however, mandatory on Windows, where you have to concatenate the following into a single CA file:

- https://letsencrypt.org/certs/lets-encrypt-x3-cross-signed.pem.txt
- https://letsencrypt.org/certs/trustid-x3-root.pem.txt

For internal or on-premises environments you need to obtain your CA certificates through your internal process.

Note:

If using the CLI on Windows, ensure you use path styles such as C:\Users\janeblogs\.cde\ca.pem when referencing local files.

Configuring TLS CA certificates using Cloudera Data Engineering CLI

The CDE CLI is currently configured using insecure mode. You can manually generate the CA certificates for Cloudera Data Engineering domains and configure the CLI in a secured mode.

About this task

This procedure must be performed in your Cloudera cluster only when you configure CDE CLI with TLS for the first time.

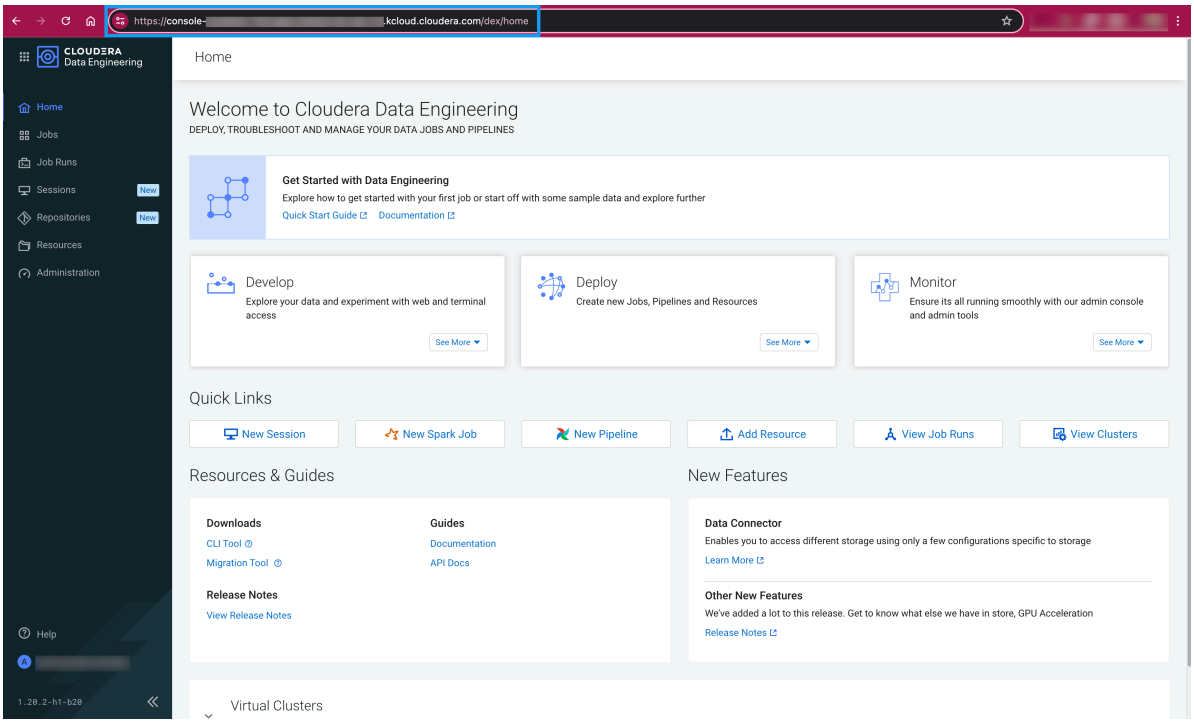
Before you begin

Ensure that the following conditons are met:

- A Cloudera Data Engineering Service and a virtual cluster are installed.
- CDE CLI for the correct host machine is downloaded.
- You are using OpenSSL to generate certificates for the domains.

Procedure

1. Generate certificate for the Cloudera domain.
   a) Get the domain information from the Cloudera Data Engineering landing page URL.

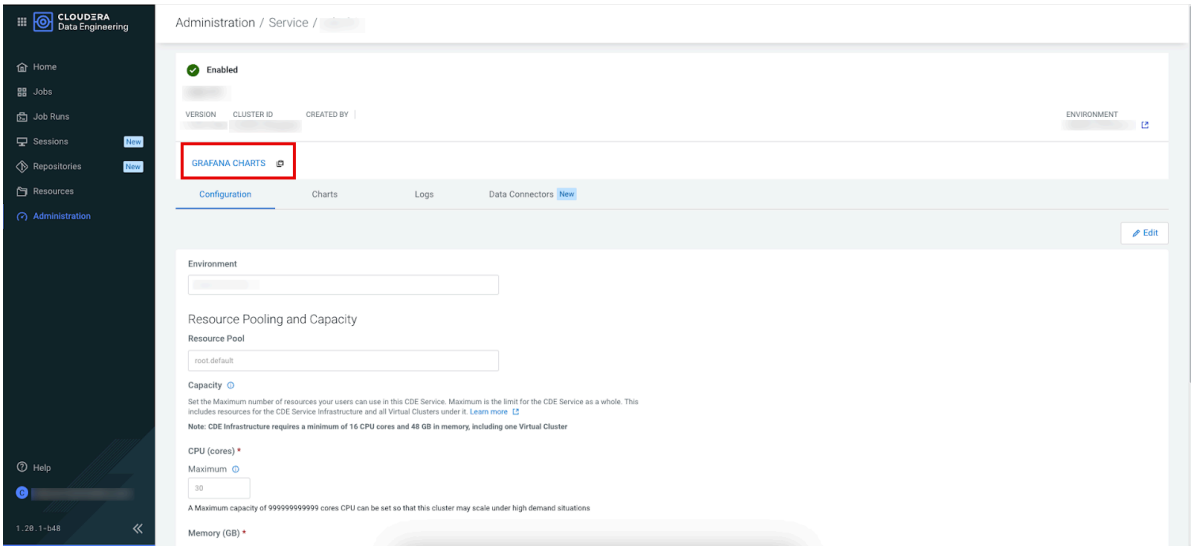


   b) Run the following command using the URL obtained from the preceding step:

```
openssl s_client -showcerts -servername [***CLOUDERA DATA ENGINEERING
 LANDING PAGE URL***] -connect [***CLOUDERA DATA ENGINEERING LANDING
 PAGE URL***]:443 </dev/null
```

   c) Copy both the certificates from the logs of the command in the preceding step.
   d) Paste the certificates one after the other with a line space between them and save the file as cdp.pem.

2. Generate certificate for the Cloudera Data Engineering domain.
   a) Go to the cluster details page in Cloudera Data Engineering and copy the URL of the Grafana charts.



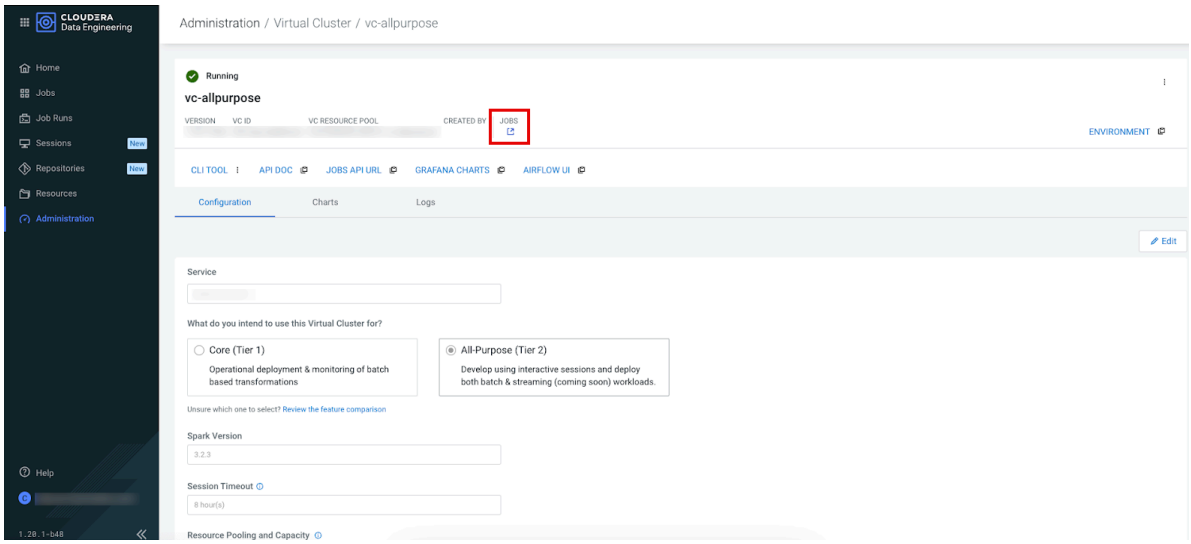
   The URL of the Grafana charts looks like https://*[***GRAFANA CHARTS DOMAIN***]*/grafana/d/sK1XDusZz/kubernetes. Copy the Grafana charts domain from that URL.

   b) Run the following command using the Grafana charts domain obtained from the preceding step:

```
openssl s_client -showcerts -servername [***GRAFANA CHARTS DOMAIN***] -
connect [***GRAFANA CHARTS DOMAIN***]:443 </dev/null
```

   c) Copy both the certificates from the logs of the command in the preceding step.
   d) Paste the certificates one after the other with a line space between them and save the file as cde.pem.

3. Generate certificates for the virtual cluster domain.
   a) Go to the virtual cluster details page and click Jobs.



   The Jobs URL looks like https://*[***VIRTUAL CLUSTER DOMAIN***]*/dex/ui/. Copy the *[***VIRTUAL CLUSTER DOMAIN***]* from that URL.
   b) Run the following command using the Virtual Cluster domain obtained from the preceding step:

```
openssl s_client -showcerts -servername [***VIRTUAL CLUSTER DOMAIN***] -
connect [***VIRTUAL CLUSTER DOMAIN***]:443 </dev/null
```

   c) Copy both the certificates from the logs of the command in the preceding step.
   d) Paste the certificates one after the other with a line space between them in the vc.pem file and save the file.
4. Run the following command to combine all the certificate files into a single file named all_certs.pem:

```
cat cdp.pem cde.pem vc.pem >> all_certs.pem
```

5. Add the certificates PEM file details in the config.yml file using the tls-ca-certs key.
   The config.yml file should look as follows:

```
vcluster-endpoint: [***YOUR VIRTUAL CLUSTER ENDPOINT***]
tls-ca-certs: [***PATH TO ALL_CERTS.PEM FILE***]
```

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

Cloudera Data Engineering provides a secure way to create and store workload secrets for Cloudera Data Engineering 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 CLI

You can create workload secrets using interactive mode or using a JSON file. You can use the `--workload-cred-json-file` and the `--workload-cred-key` flags along with the `--type` flag supporting workload-credential.

Before you begin

Make sure that you have downloaded the CLI client. For more information, see Using the Cloudera Data Engineering command line interface .

For Interactive mode

- Specify --workload-cred-key when prompted for secret values multiple times. The values which are sensitive are read as a hidden password field interactively from the CLI.

```
./cde credential create --name <workload-credential-name> --type workloa
d-credential --workload-cred-key <workload-credential_key> --workload-cr
ed-key <workload-credential_key>
```

For example:

```
./cde credential create --name workload-cred-1 --type workload-credentia
l --workload-cred-key db-pass --workload-cred-key aws-secret

Enter Secret value for Workload Cred key "db-pass" :
Re-enter Secret value for Workload Cred key "db-pass" :
Enter Secret value for Workload Cred key "aws-secret" :
Re-enter Secret value for Workload Cred key "aws-secret" :
```

For JSON file

1. Create a JSON file with workload secret keys.

```
sample.json file

{
    "<workload-credential-key>": "<secret_value_of_key>",
    "<workload-credential-key>": "<secret_value_of_key>"
  }
```

For example:

```
sample.json file

{
    "aws-secret": "secret123",
    "db-pass": "dbpass123"
  }
```

2. Run the following command to create the workload secret:

```
./cde credential create  --name <workload-credential-name> --type worklo
ad-credential --workload-cred-json-file <workload-credential-json-file-n
ame>
```

For example:

```
./cde credential create  --name workload-cred-1 --type workload-credenti
al --workload-cred-json-file sample.json
```

Updating a workload secret for Cloudera Data Engineering Spark Jobs using CLI

You can update an existing secret to use it with the Cloudera Data Engineering Spark Jobs.

For Interactive mode

- Specify --workload-cred-key when prompted for secret values multiple times. The values which are sensitive are read as a hidden password field interactively from the CLI.

```
./cde credential update --name <workload-credential-name> --workload-cre
d-key <workload-credential_key> --workload-cred-key <workload-credential
_key>
```

For example:

```
./cde credential update --name workload-cred-1 --workload-cred-key db-pa
ss --workload-cred-key aws-secret --workload-cred-key api-token
Enter Secret value for Workload Cred key "dbPassword" :
Re-enter Secret value for Workload Cred key "dbPassword" :
Enter Secret value for Workload Cred key "aws-secret" :
Re-enter Secret value for Workload Cred key "aws-secret" :
```

For JSON file

1. Update the JSON file with workload secret keys.

   For example:

```
sample.json file – file name

{
    "aws-secret": "secret123",
    "db-pass": "dbpass123"
  }
```

2. Run the following command to create the workload secret with updated parameters:

```
./cde credential update --name <workload-credential-name> --type workloa
d-credential --workload-cred-json-file <workload-credential-json-file-na
me>
```

For example:

```
./cde credential update --name workload-cred-2 --workload-cred-json-file
 sample.json
```

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

After you create a workload secret, you can link it to the Cloudera Data Engineering Spark Job definitions that you created using CLI.

```
./cde job create --name <workload-credential-name> --type <workload-credenti
al_type> --application-file <application-file-name> --workload-credential <w
orkload-credential-1> --workload-credential <workload-credential-2>
```

For example:

```
./cde job create --name test-workload-job --type spark --mount-1-resource te
st-workload --application-file test-workload-cred.py --workload-credential w
orkload-cred-1 --workload-credential workload-cred-2
```

Using a workload secret in 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.

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/secr
ets/<workload-credential-name>/<credential-key-2>
```

Example workload credentials to use in the application code:

The workload credential is created with the command below.

```
./cde credential create --name workload-cred-1 --type workload-credential --
workload-cred-key db-pass --workload-cred-key aws-secret
```

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 \
         .jdbc("jdbc:postgresql:dbserver", "schema.tablename",
         properties={"user": "username", "password": dbPass})
```

Listing an existing workload secret to the Cloudera Data Engineering Spark Job

You can list an existing secret for Cloudera Data Engineering Spark Jobs using CLI.

```
./cde credential list  --filter 'type[eq]workload-credential'
```

Example output:

```
[
    {
      "name": "workload-cred-1",
```

```
      "type": "workload-credential",
      "description": "workload credential description",
      "created": "2022-12-18T07:26:41Z",
      "modified": "2022-12-18T07:26:41Z"
    }
  ]
```

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

You can delete an existing secret for Cloudera Data Engineering Spark Jobs using CLI.

About this task

```
./cde credential delete --name <workload-credential-name>
```

For example:

```
./cde credential delete --name workload-cred-1
```