Cloudera Data Engineering 1.5.0

Using spark-submit drop-in migration tool for migrating Spark workloads to CDE

Date published: 2020-07-30
Date modified: 2023-01-24

Legal Notice

Contents

Using spark-submit drop-in migration tool for migrating Spark workloads to CDE

Cloudera Data Engineering (CDE) provides a command line tool cde-env to help migrate your CDP Spark workloads running on CDP Private Cloud Base (spark-on-YARN) to CDE without having to completely rewrite your existing spark-submit command-lines.

Supported platforms

You can use the migration tool in the following platforms:

- Linux
- MacOS

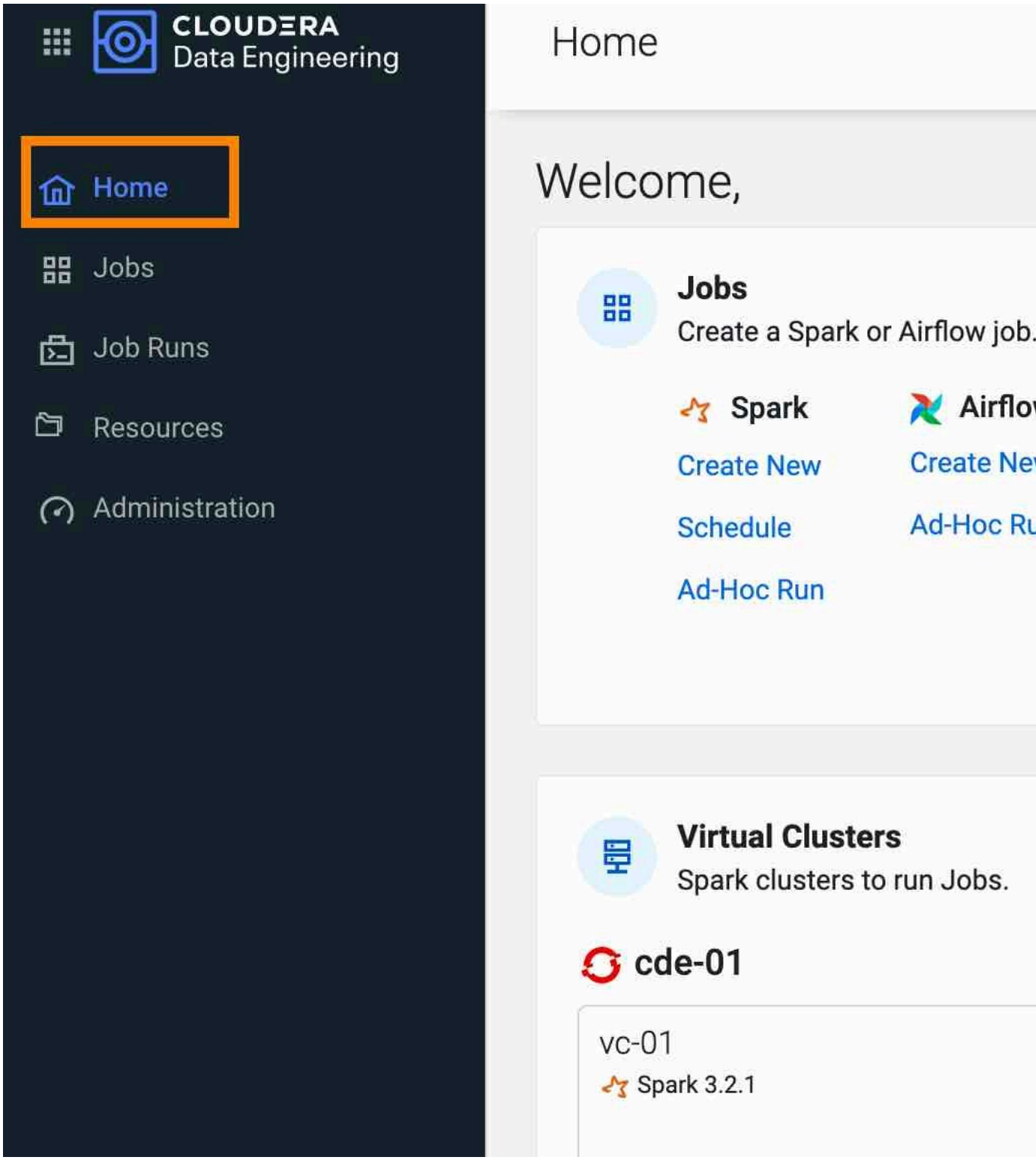
Downloading the cde-env tool

You can use the cde-env tool to migrate your spark-on-YARN workloads to CDE:

Procedure

1. in the Cloudera Data Platform (CDP) console, click the Data Engineering tile. The CDE Home page displays.

2. In the Home page, click the CDE-Env Tool link under Docs & Downloads to download the migration tool.



3. Unzip the archive.

   This is a temporary installation package and can be saved in any location. The folder contains README.md, cde, cde-env.sh, spark-submit-cde, and spark3-submit-cde files.

Installing the cde-env tool

You can install the cde-env tool as an Administrator or as a normal user. Cloudera recommends you to install the tool as an Administrator in the /opt/cloudera/bin folder so that all the users in the host can access the tool.

For Administrator

1. Install the tool by copying the required binary and script files to the /opt/cloudera/bin folder so that the migration tool can run in the current gateway host.

```
$ ./cde-env.sh enable-spark-submit-proxy -f private
```

2. Enable the installed binary and script files by granting access to those files for all users on the host using one of the following options.

   - Link the existing /usr/bin/spark-submit and /usr/bin/spark3-submit.

```
$ ln -s /opt/cloudera/bin/cde /usr/bin/cde
$ ln -s /opt/cloudera/bin/cde-env.sh /usr/bin/cde-env.sh
$ ln -s -f /opt/cloudera/bin/spark-submit /usr/bin/spark-submit
$ ln -s -f /opt/cloudera/bin/spark3-submit /usr/bin/spark3-submit
```

   or

   - Update PATH to point to the new installation location at the host level.

```
$ export PATH=/opt/cloudera/bin:$PATH
```

For User

1. Install the tool on the host by running the following command:

   By default, the binary and script files will be installed in the $HOME/bin folder. You can change the location by replacing the $HOME/bin folder to the target folder in the following command.

   Linux

```
$ sed -i "s#CLOUDERA_BIN=/opt/cloudera/bin#CLOUDERA_BIN=$HOME/bin#g"
cde-env.sh && ./cde-env.sh enable-spark-submit-proxy -f private
```

   MacOS

```
$ sed -i '' "s#CLOUDERA_BIN=/opt/cloudera/bin#CLOUDERA_BIN=$HOME/bin#g"
cde-env.sh && ./cde-env.sh enable-spark-submit-proxy -f private
```

2. Update PATH to give access to those binary and script files.

```
$ export PATH=$HOME/bin:$PATH
```

Configuring the cde-env tool

The CDE env-tool uses the ~/.cde/config.yaml configuration file to manage jobs in CDE virtual clusters. You must manually edit the ~/.cde/config.yaml file and update the profiles with the required information.

For more information, see Creating and using multiple profiles.

Prerequisites

You must obtain the virtual cluster endpoint URL, CDP endpoint URL, and generate user keys for each user whose Spark jobs you are migrating over to CDE.

Procedure

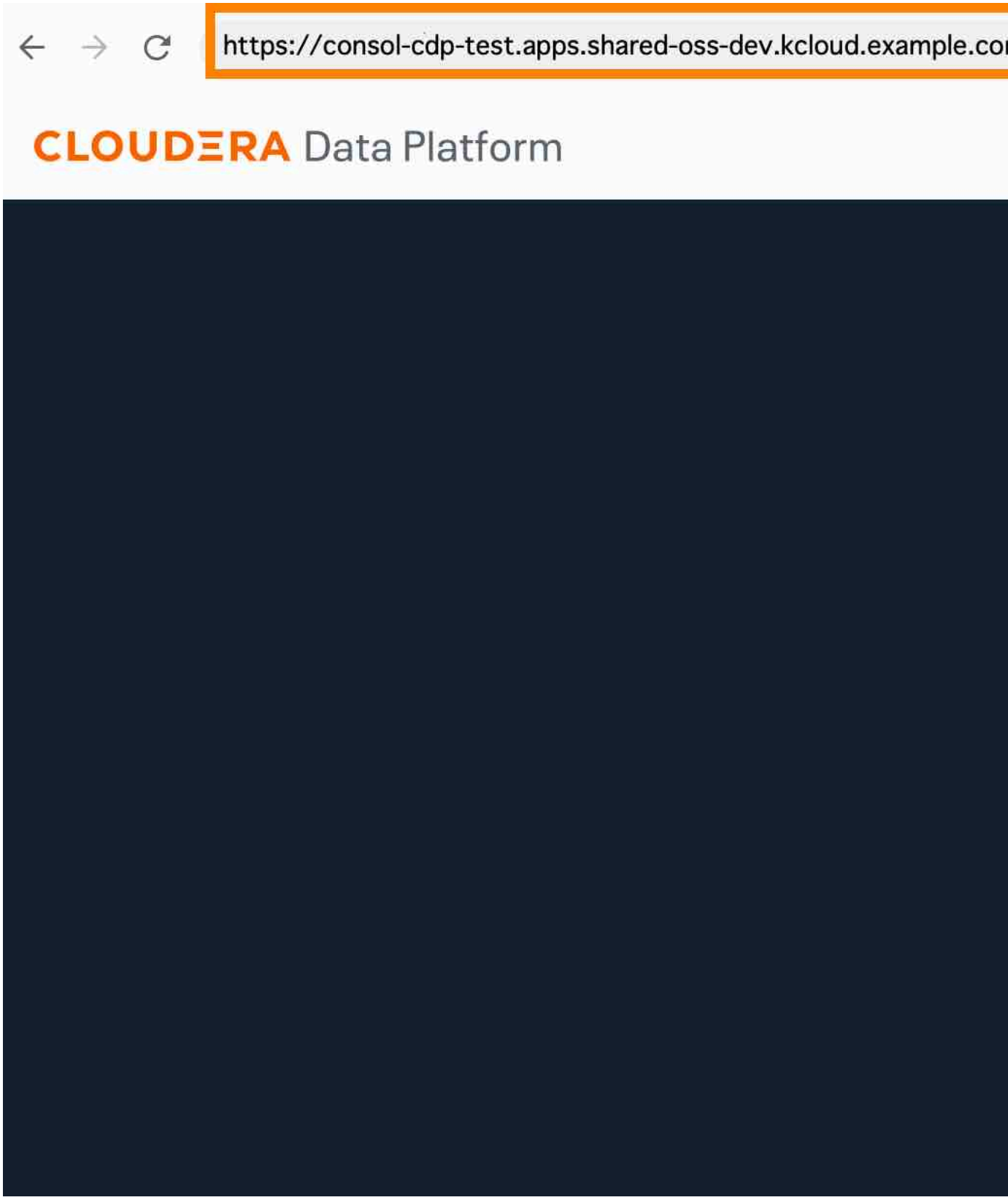
1. **Virtual Cluster Endpoint URL**: Determine the virtual cluster endpoint URL.
   a. Navigate to the Cloudera Data Engineering Home page.
   b. In the Environments column, select the environment containing the virtual cluster you want to use.
   c. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to use to migrate your spark jobs to.
   d. Click JOBS API URL to copy the URL to your clipboard.
   e. Paste the URL into a text editor to identify the endpoint host. For example, the URL is similar to the following:

      ```
      https://dfdj6kgx.cde-2cdxw5x5.ecs-demo.example.com/dex/api/v1
      ```

      In the above example, the endpoint host is

      ```
      dfdj6kgx.cde-2cdxw5x5.ecs-demo.example.com
      ```

2. **CDP Endpoint URL**: Copy the CDP console URL.



3. **Access Key**: Generate access key for each user whose Spark jobs you are migrating over to CDE:
   a. Sign in to the Cloudera Data Platform console as an Administrator.
   b. In the Cloudera Data Platform Home page, click Management Console.
   c. On the left navigation menu, click Users.
   d. On the Users page, click the name of the user or the machine user account for which you want to generate an access key.
   e. On the user account page, go to the Access Keys section and click Generate Access Key.

      Cloudera Data Platform creates the key and displays the information on the screen.

Adding profile for each user

Edit the ~/.cde/config.yaml file in the same host where the cde-env tool is installed and update the profiles with the required information. 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.

Edit the ~/.cde/config.yaml file to add the allow-all-spark-submit-flags: true parameter and update the profiles.

```
# ~/.cde/config.yaml

allow-all-spark-submit-flags: true
credentials-file: <credentials-location>
cdp-endpoint: <CDP-endpoint>
tls-insecure:true

profiles:
- name: <Profile Name1>
  vcluster-endpoint: <VC-endpoint>

- name: <Profile Name2>
  vcluster-endpoint: <VC-endpoint>
```

Example configuration file:

```
# ~/.cde/config.yaml

allow-all-spark-submit-flags: true
credentials-file: /home/cdpuser1/.cde/credentials
cdp-endpoint: https://console-xhu-141.apps.shared-os-dev-01.kcloud.example
.com
tls-insecure:true

profiles:
- name: vc-2
  vcluster-endpoint: https://5b27g4jm.cde-x6j2nh5j.apps.apps.shared-osdev-0
1.kcloud.example.com/dex/api/v1/

- name: spark3-1
  vcluster-endpoint: https://7j92n8q4.cde-smstx27m.apps.apps.shared-osdev-0
1.kcloud.example.com/dex/api/v1/
```

Using the cde-env tool

You can run the spark-submit command after installing the cde-env tool. You can use the cde-env tool even without administrator privileges.

You can activate the specific profile you want to use by running the following command:

```
$ cde-env.sh activate -p <profile-name>
```

For example, to alternatively run the the same spark-submit command either against YARN or one of the CDE virtual cluster, you can activate the relevant profile:

1. Run spark jobs on YARN by activating the yarn profile. yarn is a reserved profile name here.

   ```
   $ cde-env.sh activate -p yarn
   ```

2. Run spark jobs on a CDE virtual cluster configured by CDE CLI profile named vc-1:

   ```
   $ cde-env.sh activate -p vc-1
   ```

Switching profiles back and forth this way lets you run the same spark-submit command either against YARN or one of the CDE virtual clusters.

Run sample spark-submit command

After you activate the profile using the cdp-env tool, you can run your spark-submit commands on CDE without completely rewriting your existing spark-on-yarn command lines.

- Sample spark-submit commands you can run on the CDE workloads.

```
$ spark-submit \
--name pt_rpt_streams \
--master=yarn --deploy-mode=cluster \
--driver-memory 4G \
--executor-memory 4G --executor-cores 3 \
--num-executors 4 \
--files "$HOME/spark-sql.py" \
--conf "spark.executor.extraJavaOptions=-
Djava.security.auth.login.config=/home/hdpsparkprd/spark-
hdpsparkprdkeytab-jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf -
Djavax.security.auth.useSubjectCredsOnly=true" \
--conf "spark.driver.extraJavaOptions=-Djava.security.auth.login.config=/
home/hdpsparkprd/spark-hdpsparkprdkeytab-jaas.conf
 -Djava.security.krb5.conf=/etc/krb5.conf -
Djavax.security.auth.useSubjectCredsOnly=true" \
--co
nf "spark.io.compression.codec=org.apache.spark.io.LZ4CompressionCodec" \
$HOME/spark-sql.py
```

```
$ spark3-submit \
--name pt_rpt_streams \
--master=yarn --deploy-mode=cluster \
--driver-memory 4G \
--executor-memory 4G --executor-cores 3 \
--num-executors 4 \
--files "$HOME/spark-sql.py" \
--conf "spark.executor.extraJavaOptions=-
Djava.security.auth.login.config=/home/hdpsparkprd/spark-
hdpsparkprdkeytab-jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf -
Djavax.security.auth.useSubjectCredsOnly=true" \
--conf "spark.driver.extraJavaOptions=-Djava.security.auth.login.config=/
home/hdpsparkprd/spark-hdpsparkprdkeytab-jaas.conf
 -Djava.security.krb5.conf=/etc/krb5.conf -
Djavax.security.auth.useSubjectCredsOnly=true" \
--c
onf "spark.io.compression.codec=org.apache.spark.io.LZ4CompressionCodec" \
```

```
$HOME/spark-sql.py
```

- Sample spark-submit commands with an inline profile configuration you can run on the CDE workloads.

```
$ CDE_CONFIG_PROFILE=yarn \
spark-submit \
--name pt_rpt_streams --master=yarn \
--deploy-mode=cluster --driver-memory 4G \
--executor-memory 4G --executorcores 3 \
--num-executors 4 --files "$HOME/spark-sql.py" \
--conf "spark.executor.extraJavaOptions=-
Djava.security.auth.login.config=/home/hdpsparkprd/spark-
hdpsparkprdkeytab-jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf -
Djavax.security.auth.useSubjectCredsOnly=true" \
--conf "spark.driver.extraJavaOptions=-Djava.security.auth.login.config=/
home/hdpsparkprd/spark-hdpsparkprdkeytab-jaas.conf
 -Djava.security.krb5.conf=/etc/krb5.conf -
Djavax.security.auth.useSubjectCredsOnly=true" \
--c
onf "spark.io.compression.codec=org.apache.spark.io.LZ4CompressionCodec" \
$HOME/spark-sql.py
```

```
$ CDE_CONFIG_PROFILE=vc-1 \
spark3-submit \
--name pt_rpt_streams \
--master=yarn --deploy-mode=cluster \
--driver-memory 4G --executor-memory 4G \
--executor-cores 3 --num-executors 4 \
--files "$HOME/spark-sql.py" \
--conf "spark.executor.extraJavaOptions=-
Djava.security.auth.login.config=/home/hdpsparkprd/spark-
hdpsparkprdkeytab-jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf -
Djavax.security.auth.useSubjectCredsOnly=true" \
--conf
"spark.driver.extraJavaOptions=-Djava.security.auth.login.config=/home/
hdpsparkprd/spark-hdpsparkprdkeytab-jaas.conf -Djava.security.krb5.conf=/
etc/krb5.conf -Djavax.security.auth.useSubjectCredsOnly=true" \
--c
onf "spark.io.compression.codec=org.apache.spark.io.LZ4CompressionCodec" \
$HOME/spark-sql.py
```

Known Issues and Limitations

This page lists the current known issues and limitations that you might run into while using the cde-env tool.

- Limited to spark-submit commands and does not include spark-shell, pyspark, and sparksql.
- When activating a profile using the cde-env.sh script, there is no validation yet on whether such profile exists. However, if a profile does not exist, it will display an error when running the spark-submit command.
- The following spark-submit flags are not yet supported in CDE:
  - --archives
  - --exclude-packages
  - --driver-class-path
  - --driver-library-path
  - --driver-java-options

  Users are instead suggested to create CDE jobs to handle the above mentioned scenarios.