

The Cloudera logo, featuring the word "CLOUDERA" in a bold, orange, sans-serif font.

# CDE Workshop

## Student Guide

<b>Introduction</b>	<b>3</b>
<b>Prerequisites</b>	<b>4</b>
Verify access to the environment	4
Download the resources	5
Update the files	6
<b>Lab 1 - Walkthrough of CDE Data Service</b>	<b>6</b>
ACCESSING THE VIRTUAL CLUSTER	10
Step 1 : Select the appropriate CDE Service	10
Step 2 : Virtual cluster selection	10
<b>Lab 2 - Create and trigger ad-hoc Spark jobs</b>	<b>11</b>
Resource Creation	11
Job Creation	13
Triggering the jobs	16
<b>Lab 3 - Add schedule to the ad-hoc Spark jobs</b>	<b>19</b>
<b>Lab 4 - Orchestrate a set of jobs using Airflow</b>	<b>21</b>
<b>Lab 5 - Install and Configure CDE CLI</b>	<b>25</b>
For Mac users:	25
For Windows users:	29
<b>Lab 6 - Run jobs using CDE CLI</b>	<b>32</b>
Run a spark-scala job using CLI	32
<b>Lab 7 - Data Lineage and Auto-Scaling</b>	<b>33</b>
Data Lineage using Atlas	33
Auto-scaling in CDE	36

## Introduction

---

This document aims to introduce to our partners the features of **CDE**, the Data Engineering Data Service of Cloudera Data Platform (**CDP**). During the course of this workshop, you will experience how simple it is to run and orchestrate spark jobs with the help of auto-scaling infrastructure. We will use Airflow for orchestrating the various jobs.

In this workshop:

- You will be given an active CDE service running in an existing/registered CDP environment in a given tenant.
- You will have a virtual cluster with the given configuration that serves as compute for the spark workload.
- You will run sample spark jobs as ad-hoc jobs.
  - PySpark
  - Spark-scala
- You will run the same spark jobs as part of a schedule.
- With the help of Airflow, you will orchestrate a set of Spark jobs and trigger them as a flow.
- You will use CDE CLI to trigger the jobs from terminal/powershell.
- You will see the data lineage using Atlas and witness the auto-scaling capabilities of the CDE Data service.

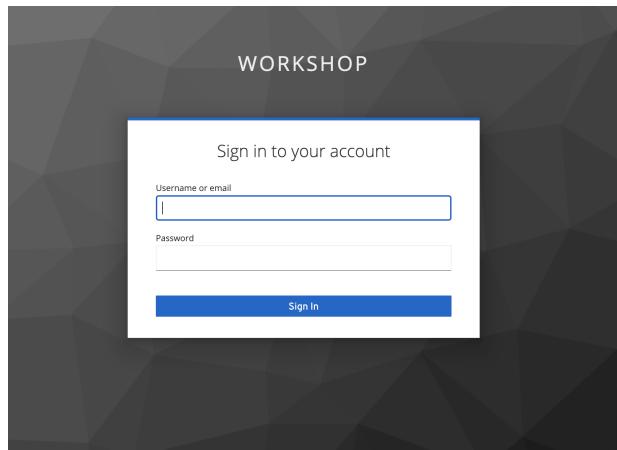
# Prerequisites

---

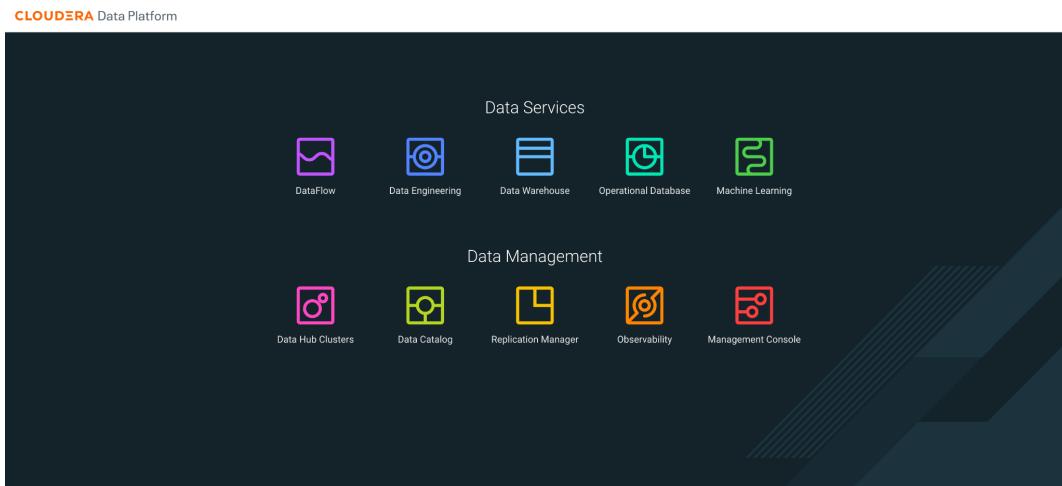
## Verify access to the environment

- Open the shared link and login with the credentials assigned to you.

<Will be shared by the instructor at the start>



- You should land on the CDP Console as shown below.

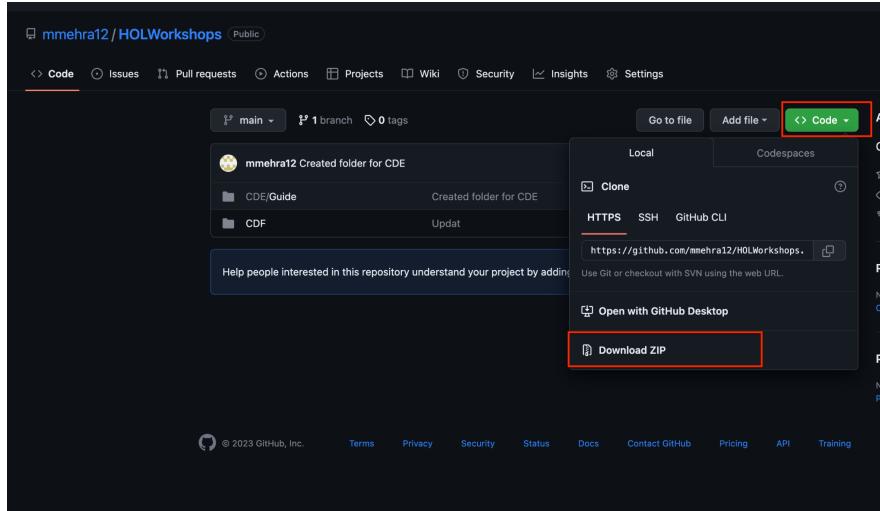


## Download the resources

There are two ways in which you can access the scripts/resources.

- Download the zip file from the GitHub repository.

<https://github.com/mmehra12/HOLWorkshops>

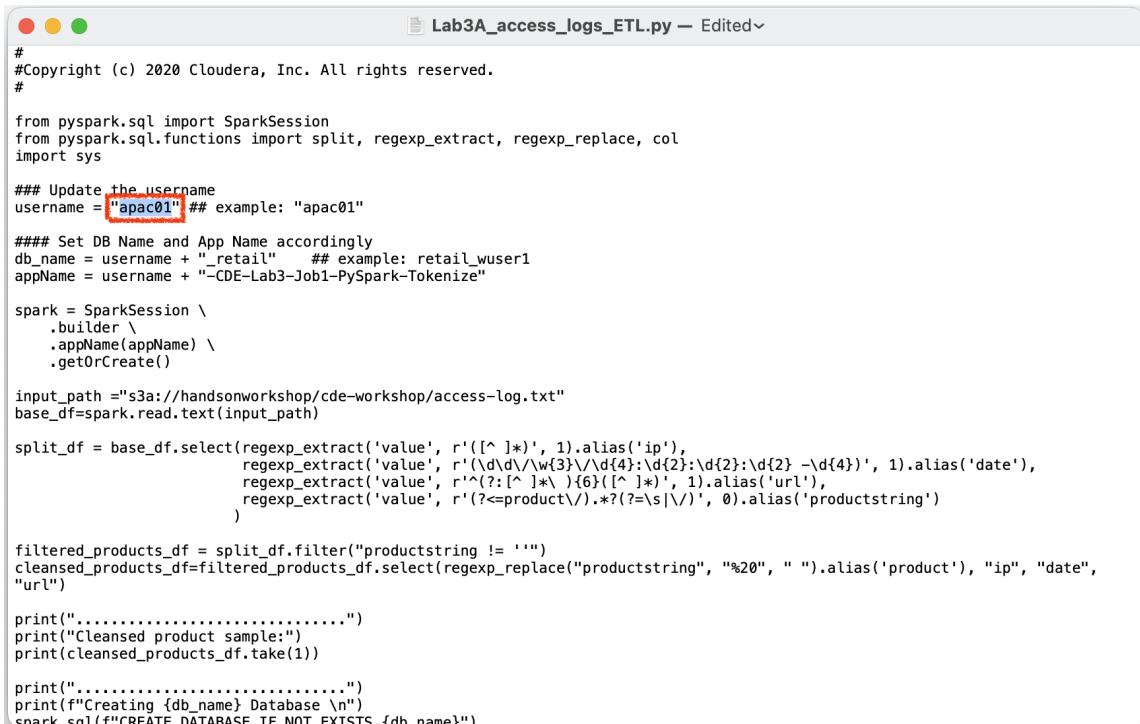


After decompressing the ZIP file the folder structure should look something like this

- The resources were also sent to you on your registered email an hour before the event. Please download the zip file attached to the email.  
After decompressing the ZIP file the folder structure should look something like this

## Update the files

- Go through each script and update the necessary values as mentioned in the script.
  - For all the scripts, update the username field with the username that you have been assigned to. You will find this at the starting of the script itself.



```
#Copyright (c) 2020 Cloudera, Inc. All rights reserved.
#
from pyspark.sql import SparkSession
from pyspark.sql.functions import split, regexp_extract, regexp_replace, col
import sys
### Update the username
username = "apac01" ## example: "apac01"
#### Set DB Name and App Name accordingly
db_name = username + "_retail" ## example: retail_wuser1
appName = username + "-CDE-Lab3-Job1-PySpark-Tokenize"
spark = SparkSession \
    .builder \
    .appName(appName) \
    .getOrCreate()
input_path ="s3a://handsonworkshop/cde-workshop/access-log.txt"
base_df=spark.read.text(input_path)
split_df = base_df.select(regexp_extract('value', r'([^\s]+)', 1).alias('ip'),
                         regexp_extract('value', r'(\d\d\d\w{3}\d{4}\d{2}\d{2}\d{4})', 1).alias('date'),
                         regexp_extract('value', r'(^|\s){6}([^\s]+)', 1).alias('url'),
                         regexp_extract('value', r'(?=<product\b).*(?=\s|/)', 0).alias('productstring'))
filtered_products_df = split_df.filter("productstring != ''")
cleansed_products_df=filtered_products_df.select(regexp_replace("productstring", "%20", " ").alias('product'), "ip", "date",
"url")
print(".....")
print("Cleansed product sample:")
print(cleansed_products_df.take(1))
print(".....")
print(f"Creating {db_name} Database \n")
spark.sql(f"CREATE DATABASE IF NOT EXISTS {db_name}")
```

# Lab 1 - Walkthrough of CDE Data Service

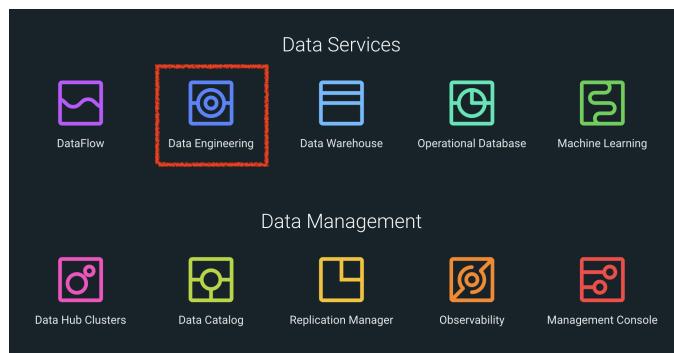
Cloudera Data Engineering (CDE) is a serverless service for Cloudera Data Platform that allows you to submit jobs to auto-scaling virtual clusters.

The CDE service involves several components:

- **Environment**
  - A logical subset of your cloud provider account including a specific virtual network.
- **CDE Data Service**
  - The long-running Kubernetes cluster and services that manage the virtual clusters. The CDE service must be enabled in an environment before you can create any virtual clusters.
- **Virtual Cluster**
  - An individual auto-scaling cluster with defined CPU and memory ranges. Virtual Clusters in CDE can be created and deleted on demand. Jobs are associated with clusters.
- **Job**
  - Application code along with defined configurations and resources. Jobs can be run on demand or scheduled.
- **Resource**
  - A defined collection of files such as a Python file or application JAR, dependencies, and any other reference files required for a job.
- **Job run**
  - An individual job run.

The above components can be accessed in the following ways:

- Go to the CDP console and click on Data Engineering.

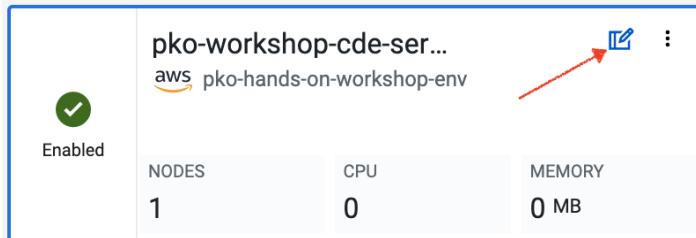


- You will see the CDE homepage

Note : While the page loads take a look at the next setup, we can come back to this later

- We should have a CDE service running which we will use for this workshop. To check this Select the **ADMINISTRATION** option on the left menu on your screen. You should be able to see all the CDE Service and their status.

- On the **CDE service pko-workshop-cde-service**, click on the pencil icon and observe the configuration and other details related to the service.



Administration / Service / pko-workshop-cde-service

**pko-workshop-cde-service**

VERSION: 1.18.1-h3-b6 CLUSTER ID: cluster-l87grkbn CREATED BY: Manick Mehra NODES: 1 / 50 CPU: 0 / 400 MEMORY: 0 MB / 1600 GB

DATA LAKE: pko-workshop-dl ENVIRONMENT: pko-hands-on-workshop-env

GRAFANA CHARTS RESOURCE SCHEDULER

Configuration Charts Logs Access Diagnostics

**Edit**

**Environment:** pko-hands-on-workshop-env

**Workload Type:** General - Small (EBS)

**EBS Size:** 100 GB

**Capacity & Costs:**

- Autoscale Range:** On-demand Instances (1 to 50)
- Use Spot Instances:** (unchecked)

**Network & Storage:**

- Enable Public Loadbalancer:** (checked)

**API server Authorized IP Ranges:** 0.0.0.0/0

- Click on each tab and go through all the details related to the CDE service.
- Once done, click on the **Home** on the left tab to go back to the CDE home page. This page shows us the active CDE services and the associate clusters. Let's start with accessing the virtual cluster that is assigned to you.

## ACCESSING THE VIRTUAL CLUSTER

Step 1 : Select the appropriate CDE Service

Go to the Administration page and select your CDE Service (In our case **partner-hol-cde-service**)

The screenshot shows the Cloudera Data Engineering Administration interface. On the left sidebar, 'Administration' is selected. The main area lists several CDE services:

- cde-test**: Failed, cdp-azure-demo-cdpwipdec
- cdecdp**: Failed, cdpoc
- cdetest**: Failed, cdpazure
- meta-cde**: Enabled, aws meta-workshop. Nodes: 4, CPU: 0, Memory: 0 MB.
- pko-workshop-cde-service...**: Enabled, aws pko-hands-on-workshop-env. Nodes: 1, CPU: 0, Memory: 0 MB. This service is highlighted with a blue border.
- pse-workshop**

Step 2 : Virtual cluster selection

- Select the CDE Service and click on the virtual cluster that was assigned to you.

The screenshot shows the Cloudera Data Engineering Administration interface. The 'Administration' section on the left has 'Administration' selected. The main area displays the 'pko-workshop-cde-service...' cluster in detail:

- aws-cdtest-operation...**: Failed, aws aws-cd-test-operations
- cde-test**: Failed, cdp-azure-demo-cdpwipdec
- cdecdp**: Failed, cdpoc
- cdetest**: Failed, cdpazure
- meta-cde**: Enabled, aws meta-workshop. Nodes: 4, CPU: 0, Memory: 0 MB.
- hostcz-virtual-cluster**: Running, pko-workshop-cde-service. Nodes: 0, CPU: 0, Memory: 0 MB, Jobs: 0.
- cluster07**: Failed, pko-workshop-cde-service. Nodes: 0, CPU: 0, Memory: 0 MB, Jobs: 0.
- pko-workshop-cde-service...**: Enabled, aws pko-hands-on-workshop-env. Nodes: 1, CPU: 0, Memory: 0 MB. This cluster is highlighted with a red box.

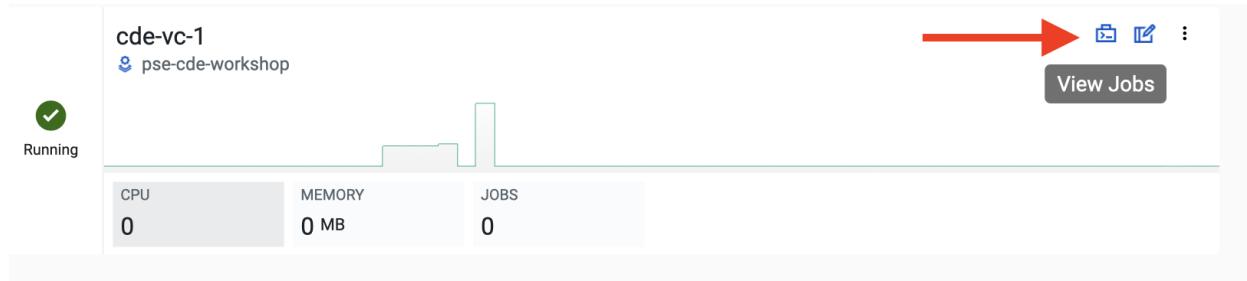
## Lab 2 - Create and trigger ad-hoc Spark jobs

In this lab, we will create spark jobs and run them on an ad-hoc basis, i.e., without any schedule. As part of this lab, we have taken two simple use-cases that can be addressed with the help of Spark jobs.

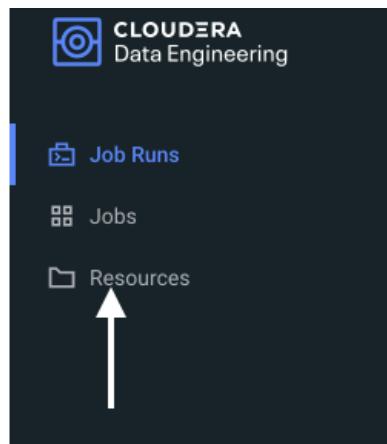
1. Log Data Cleansing using Spark
2. Analyze the Paycheck Protection Program Data
  - a. Report 1: Breakdown of all cities in Texas that retained jobs
  - b. Report 2: Breakdown of company type that retained jobs
3. PySpark job to enrich your data using an existing data warehouse

### Resource Creation

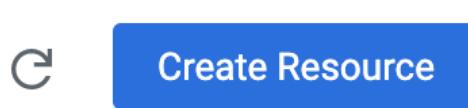
- On the virtual cluster **Cluster Name** : <username>-virtual-cluster [ Virtual cluster created in Lab 1] tab, click on view jobs. This will open a new page with details of the Job Runs, Jobs, and Resources.



- In the left pane, click on the **Resources** tab.



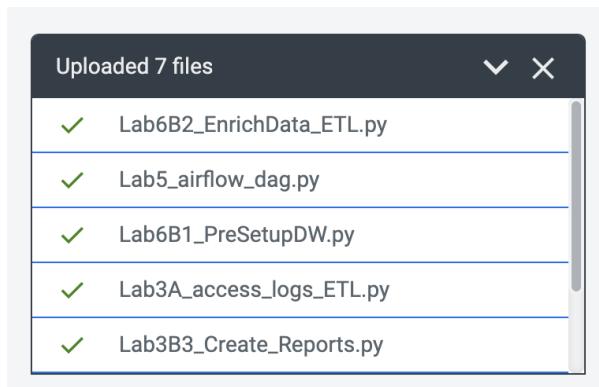
- You will get the **Resources** page to the right. Click on **Create Resource**.



- Give a unique name(username-resources) and create the resource. This acts as your repository for storing all the scripts and dependencies.
- Once it is created, you will get an option to upload the files as shown below.

Drop files here or click on the "Upload Files" button to select them from your computer  
**Upload Files**

- Click on **Upload Files** and select all the scripts downloaded from the prerequisites step. (**Please upload only .py files**)
- You will get a pop-up with all the files uploaded to your resource.



- Click on **Resources** at the top, select your resource and validate if all the five .py files are present in your resource. We are now ready to create jobs using these resources.

Resources / pkatti-resources / Files



## Job Creation

- We will now create the first job with the script ***Lab3A\_access\_logs\_ETL.py***.
- In the left pane, click on **Jobs**.
- You will get the **Jobs** page to the right. Click on **Create Job**.



- Select job type as **Spark**.
- Please give the job names as mentioned below.

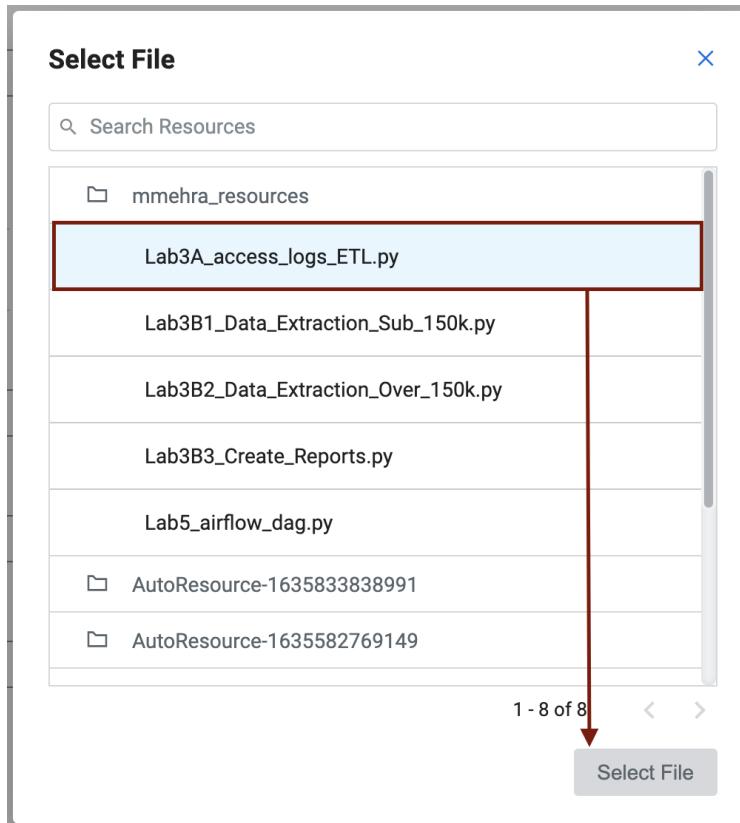
<username>\_<script\_name\_without\_py\_extension>

Eg:- For wuser1, job1 name would be **wuser1\_Lab3A\_access\_logs\_ETL**

The screenshot shows the 'Job Details' section of the CDE interface. It includes fields for 'Job Type' (with 'Spark 2.4.7' selected), 'Name' (containing 'wuser01\_Lab3A\_access\_logs\_ETL'), and other configuration options.

- As this is a shared environment, please name the jobs with your username so that it helps in differentiating yours from others' jobs.
- In **Application File**, click on **Select from Resource** and select the file ***Lab3A\_access\_logs\_ETL.py*** from your resource(<username>-resources).

The screenshot shows the 'Application File' section of the CDE interface. It has two radio button options: 'File' (selected) and 'URL'. Below them is a button labeled 'Upload or Select from Resource', with 'Select from Resource' being highlighted by a red box.



- Ignore the remaining configuration options. Do not enable the schedule now. This is how it should finally look like.

Jobs / Create Job

Job Type \*

Spark 2.4.7  Airflow

Name \*

mmehra\_Lab3A\_access\_logs\_ETL

Application File

File  URL

Lab3A\_access\_logs\_ETL.py

Arguments (Optional)

Argument

Configurations (Optional)

config\_key config\_value

Python Version

Python 3  Python 2

Python Environment

Select Python Environment

Advanced Options

Upload additional files, customize no. of executors, driver and executor cores and memory

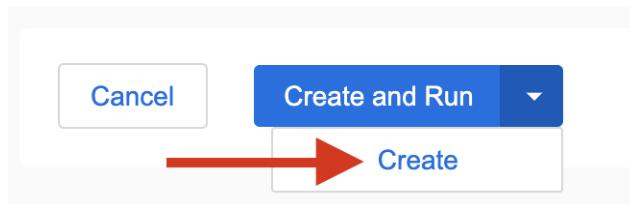
Schedule

Turn on to schedule Job, enable catchup and jobs dependants

Cancel Create and Run ▾

Create

- Click on the drop down option and click on **Create**. (do not click Create and Run)



- Similarly, create three other jobs with the same naming conventions. Please refer to the table below to confirm you are creating exactly the same.

For wuser1:

Jobs	Job Name	Script Used
Job1	wuser1_Lab3A_access_logs_ETL	Lab3A_access_logs_ETL.py
Job2	wuser1_Lab3B1_Data_Extraction_Sub_150k	Lab3B1_Data_Extraction_Sub_150k.py
Job3	wuser1_Lab3B2_Data_Extraction_Over_150k	Lab3B2_Data_Extraction_Over_150k.py
Job4	wuser1_Lab3B3_Create_Report	Lab3B3_Create_Report.py

- Create these jobs as **ad-hoc** jobs i.e., without any schedule.
- Once done, click on the **Jobs** tab and enter your username in the search bar and press **ENTER**. You should see four jobs as shown below with your username.

Status	Job	Type	Schedule	Modified On	Actions
Idle	wuser01_Lab3B3_Create_Report	Spark	Ad-Hoc	Nov 3, 2021, 12:05:56 PM	⋮
Idle	wuser01_Lab3B2_Data_Extraction_Over_150k	Spark	Ad-Hoc	Nov 3, 2021, 12:05:34 PM	⋮
Idle	wuser01_Lab3B1_Data_Extraction_Sub_150k	Spark	Ad-Hoc	Nov 3, 2021, 12:05:09 PM	⋮
Idle	wuser01_Lab3A_access_logs_ETL	Spark	Ad-Hoc	Nov 3, 2021, 12:04:40 PM	⋮

- Observe the type of the job is set to Spark and for schedule, it is Ad-hoc.

## Triggering the jobs

- You need to trigger the jobs in the following order
  - JOB 1 : wuser1\_Lab3A\_access\_logs\_ETL
  - JOB 2 : wuser1\_Lab3B1\_Data\_Extraction\_Sub\_150k
  - JOB 3 : wuser1\_Lab3B2\_Data\_Extraction\_Over\_150k
  - JOB 4 : wuser1\_Lab3B3\_Create\_Report (Run once JOB 2 and JOB 3 have completed successfully)

**NOTE : JOB 1, JOB 2 and JOB 3 can be triggered one after the other.  
JOB 4 should be executed after the successful completion of JOB 2 and  
JOB 3**

- To trigger the job, go to the **Jobs** tab, click on the 3-dotted icon, and click on **Run Now**.

The screenshot shows the 'Jobs' page in the Cloudera Data Engineering interface. A search bar at the top contains 'wuser'. Below it is a filter section labeled 'Filter By: Status' and 'Type'. A 'Create Job' button is on the right. The main table lists four jobs:

Status	Job	Type	Schedule	Modified On	Actions
○	wuser01_Lab3B3_Create_Reports	Spark	Ad-Hoc	Nov 3, 2021, 12:05:56 PM	⋮
○	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	Ad-Hoc	Nov 3, 2021, 12:05:34 PM	⋮
○	wuser01_Lab3B1_Data_Extraction_Su...	Spark	Ad-Hoc	Nov 3, 2021, 12:05:09 PM	⋮
○	wuser01_Lab3A_access_logs_ETL	Spark	Ad-Hoc	Nov 3, 2021, 12:04:40 PM	⋮

A context menu is open over the fourth job ('wuser01\_Lab3A\_access\_logs\_ETL'). The 'Run Now' option is highlighted with a red box. Other options in the menu include 'Add Schedule', 'Clone', 'Configuration', and 'Delete'.

- To check the job logs, click on **Job Runs** and select the **ID** against the job that you have triggered.

The screenshot shows the 'Job Runs' page in the Cloudera Data Engineering interface. The left sidebar has 'Job Runs' selected. The main area shows a table of job runs:

Status	Run ID	Job	Type	User	Duration	Start Time	Actions
○	47	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	wuser01		Nov 3, 2021, 12:09:05 PM	⋮
○	46	wuser01_Lab3B1_Data_Extraction_Su...	Spark	wuser01		Nov 3, 2021, 12:09:04 PM	⋮
○	45	wuser01_Lab3A_access_logs_ETL	Spark	wuser01		Nov 3, 2021, 12:08:59 PM	⋮

The screenshot shows the 'Job Runs' page in the Cloudera Data Engineering interface. The left sidebar has 'Job Runs' selected. The main area shows a table of job runs:

Status	Run ID	Job	Type	User	Duration	Start Time	Actions
○	47	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	wuser01	2.4 MIN	Nov 3, 2021, 12:09:05 PM	⋮
○	46	wuser01_Lab3B1_Data_Extraction_Su...	Spark	wuser01	1.8 MIN	Nov 3, 2021, 12:09:04 PM	⋮
○	45	wuser01_Lab3A_access_logs_ETL	Spark	wuser01	1.9 MIN	Nov 3, 2021, 12:08:59 PM	⋮

The screenshot shows the CDE Job Runs page with the following details:

- Status:** Succeeded
- Job:** wuser01\_Lab3B2\_Data\_Extr... [Atlas](#)
- Lineage:** Atlas
- Duration:** 2.4 min
- Start Time:** Nov 3, 2021, 12:09:05 PM

Below the main summary, there are tabs for Trends, Configuration, Logs, and Spark UI. The Trends tab is selected, displaying a histogram of job run durations. A search bar for "Search by Run Id" is present. At the bottom, a table lists job runs with columns: Status, Run ID, Duration, Executor Memory, Drivers Cores, Executor Cores, User, Start Time, and Actions. The first row in the table corresponds to the job shown at the top.

- For simplifying the job selection, you can choose the **User** filter and add your username and hit enter. You will see the list of jobs triggered by you.

## Job Runs

The screenshot shows the search interface for job runs. A red arrow points to the "User" dropdown menu, which is currently set to "User". Other filter options visible include "Run Id", "Status", "Job Name", and "User". A search bar labeled "Search Job Runs" is also present.

- Navigate to different tabs in the job run page and you will see all that you need to observe for the run of a Spark job.

The screenshot shows the CDE Job Run page for a specific job run. The "Trends" tab is selected, while other tabs for Configuration, Logs, and Spark UI are available. The main content area is currently blank.

## Lab 3 - Add schedule to the ad-hoc Spark jobs

In this lab, we will add a schedule to a job created as part of the previous lab.

- We will add a schedule to the job **Lab3A\_access\_logs\_ETL** (in your case it will be <username>\_Lab3A\_access\_logs\_ETL)
- Go to **Jobs** tab, click on the 3-dotted icon next to the job **Lab3A\_access\_logs\_ETL** and select **Add schedule**.

The screenshot shows the Cloudera Data Engineering interface. On the left is a sidebar with 'Home', 'Jobs' (which is selected), 'Job Runs', 'Resources', and 'Administration'. The main area is titled 'hostcz-virtual-cluster / Jobs'. It has a search bar, filter options for 'Status' and 'Type', and a 'Create Job' button. Below is a table with columns: Status, Job, Type, Schedule, Modified On, and Actions. Four jobs are listed: 'hostcz\_Lab3B3\_Create\_Reports' (Spark, Ad-Hoc, May 9, 2023, 4:11:27 PM), 'hostcz\_Lab3B2\_Data\_Extraction\_Over\_150k' (Spark, Ad-Hoc, May 9, 2023, 4:11:01 PM), 'hostcz\_Lab3B1\_Data\_Extraction\_Sub\_150k' (Spark, Ad-Hoc, May 9, 2023, 4:10:32 PM), and 'hostcz\_Lab3A.access\_logs.ETL' (Spark, Ad-Hoc, May 9, 2023, 3:21:27 PM). A red arrow points to the three-dot menu icon next to the last job. A context menu is open over this job, with 'Add Schedule' highlighted by a red box.

- You will land in the **Job Schedule** page. Click on **Create a Schedule**.

The screenshot shows the 'Jobs / wuser01\_Lab3A.access\_l...' page. At the top, there's a status bar with 'vc1'. Below is a table with 'Status' (Ad-Hoc) and 'Runs' (1). The 'Schedule' tab is selected. A message box says 'This job currently does not have a schedule.' At the bottom right is a blue 'Create a Schedule' button.

- Choose the **Cron Expression** option and enter the cron expression as given below.

**\*/10 \* \* \* \*** → This means that the job is scheduled to run every 10 minutes.

Jobs / skillupuser\_Lab3A\_acces...

Status: Ad-Hoc | Runs: 0

Schedule Tab (highlighted)

Cron Expression: \*/10 \* \* \* \*

Start Date: Thursday, May 11, 2023 at 11:35:28 AM

End Date: Friday, May 12, 2023 at 11:35:28 AM

Scheduling Configurations:

- Enable Catchup: Kick off Job Runs for intervals that has not been run along with the current interval
- Depends on Previous: If the job runs are dependant, then the catchup will occur serially

Actions: Cancel, Add Schedule

**Job Schedule has been updated**

- You can repeat the same process for the other jobs as well.
- JOB 1 : Run every 10 mins  
 JOB 2 : Run every 10 mins  
 JOB 3 : Run every 10 mins  
 JOB 4 : Run every 30 mins
- Wait for the scheduler to trigger the jobs at the scheduled time and observe the results and logs for the same.
  - Once done, please pause the schedule for all the jobs for which it was added by following the below steps.**
  - Go to the Jobs tab, click on the 3-dotted icon next to the job and select **Pause schedule**. [ Do this for all jobs ]

Jobs

Status	Job	Type	Schedule	Modified On	Actions
○	skillupuser_Lab3A_access_logs_ETL	Spark	*/10 * * * *	May 11, 2023, 11:36:58 AM	⋮
○	hostcz_Lab3B1_Create_Reports	Spark	*/10 * * * *	May 10, 2023, 11:36:58 AM	⋮
○	hostcz_Lab3B2_Data_Extraction_Over_150k	Spark	*/10 * * * *	May 10, 2023, 11:36:58 AM	⋮
○	hostcz_Lab5_airflow_dag	Airflow	*/10 * * * *	May 9, 2023, 12:31:33 PM	⋮
○	hostcz_Lab3A_access_logs_ETL	Spark	Ad-Hoc	May 9, 2023, 12:31:33 PM	⋮
○	hostcz_Lab3B1_Data_Extraction_Sub_150k	Spark	Ad-Hoc	May 9, 2023, 12:31:33 PM	⋮

The schedule for skillupuser\_Lab3A\_access\_logs\_ETL job has been paused.

## Lab 4 - Orchestrate a set of jobs using Airflow

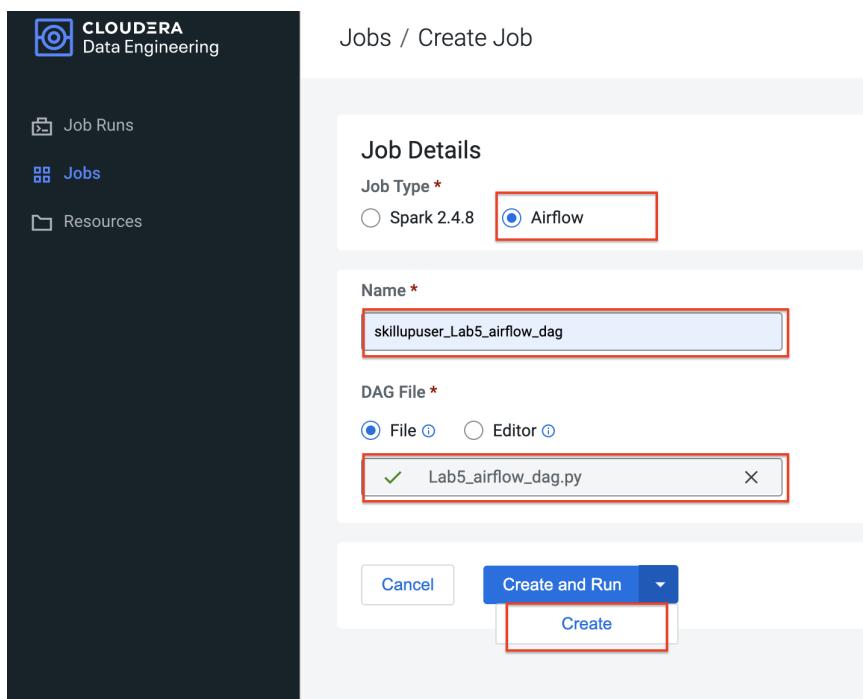
In this lab, we will create a flow with the help of a dag file that uses the jobs created in Lab3. Thus, you will be able to complete subsequent labs only if you have completed Lab3 successfully.

- Go to Jobs tab, click on **Create Job** and choose Airflow in Job type.
- Give the job name as below and upload the *Lab5\_airflow\_dag.py* file from the resources.

JOB NAME : <username>\_Lab5\_airflow\_dag

Example : For user **wuser01** the job name will be, **wuser01\_Lab5\_airflow\_dag**

- Click on **Create**.



- Go to **Jobs** tab and observe the airflow job created with the schedule mentioned in the dag file.

### Job

Jobs		
Status	Job	Type
	hostcz_Lab5_airflow_dag	Airflow

## DAG File

```

## Update the username
owner = "<ENTER YOUR USER NAME HERE>" # Example: "apac01"

DAG_name = owner + "_Airflow_Dag"
job_name_1 = owner + "_Lab3B1_Data_Extraction_Sub_150k"
job_name_2 = owner + "_Lab3B2_Data_Extraction_Over_150k"
job_name_3 = owner + "_Lab3B3_Create_Reports"

default_args = {
    'owner': owner,
    'retry_delay': timedelta(seconds=5),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 0
}

dag = DAG(
    DAG_name,
    default_args=default_args,
    start_date=datetime(2023, 5, 3),
    end_date=datetime(2023, 5, 18),
    schedule_interval='*/20 * * * *',
    catchup=False,
    is_paused_upon_creation=False
)

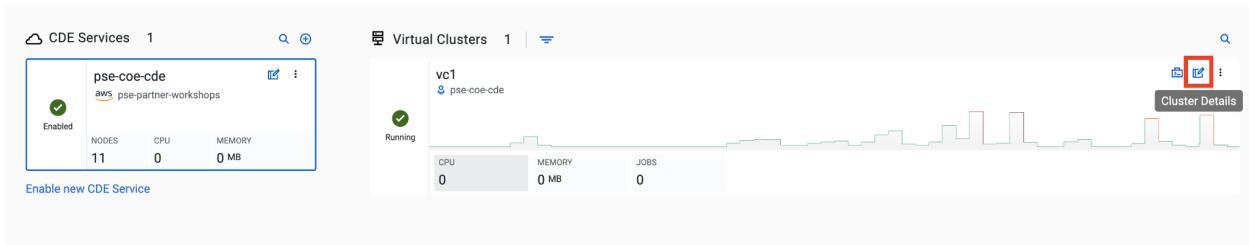
start = DummyOperator(task_id='start', dag=dag)

Data_Extraction_Sub_150k = CDEJobRunOperator(
    task_id=job_name_1,
    retries=3,
    dag=dag,
    job_name=job_name_1
)

```

- Go to the Virtual Cluster you are using and click on **Cluster Details**.

Overview



- Click on **Airflow UI** and observe the schedule created for your job.

Overview / vc1

**vc1**

Running

VERSION 1.12.0-b119 VC ID dex-app-hg98mkj CREATED BY Panag Katti CPU 3 MEMORY 4 GB JOBS 0

ENVIRONMENT DATA LAKE

**AIRFLOW UI**

Configuration Charts Logs

CDE Service pse-partner-workshops

Autoscale Max Capacity 30

CPU 30 600

Memory (GB) 508 2400

Driver and Executors will run on By default Driver would run on on-demand and executors on spot

Spark Version Spark 2.4.7

Enable Airflow Job Authoring UI (Technical Preview)

DAGs

All 3 Active 3 Paused 0

Filter DAGs by tag Search DAGs

DAG Owner Runs Schedule Last Run Recent Tasks Actions Links

wuser01\_Airflow\_Dag wuser01 0000C \*5\*\*\*

DAG: wuser01\_Airflow\_Dag schedule: \*5\*\*\*

Tree View Graph View Calendar View Task Duration Task Tries Landing Times Gantt Details Code

2021-11-03T07:39:37Z Runs 25 Update No DAG runs yet.

CDEJobRunOperator DummyOperator

Auto-refresh

```

graph TD
    start((start)) --> Lab3B1[wuser01_Lab3B1_Data_Extraction_Sub_150k]
    Lab3B1 --> Lab3B2[wuser01_Lab3B2_Data_Extraction_Over_150k]
    Lab3B2 --> Lab3B3[wuser01_Lab3B3_Create_Reports]
    Lab3B3 --> end((end))

```

- Once the job has run successfully, we need to edit the job to **pause** the schedule.
- Click on the Jobs tab and locate the airflow job that you have just created.
- Next to the job, click on the 3 dots and click on **Pause Schedule**.

The screenshot shows the CDE Jobs interface. At the top, there's a search bar, filter dropdowns for Status and Type, and a 'Create Job' button. Below is a table with columns: Status, Job, Type, Schedule, Modified On, and Actions. The 'Actions' column contains three dots, Run Now, Pause Schedule, Configuration, and Delete. The 'Pause Schedule' option is highlighted with a red box. A green success message box is overlaid on the interface, stating: 'The schedule for hostcz\_Lab5\_airflow\_dag job has been paused.'

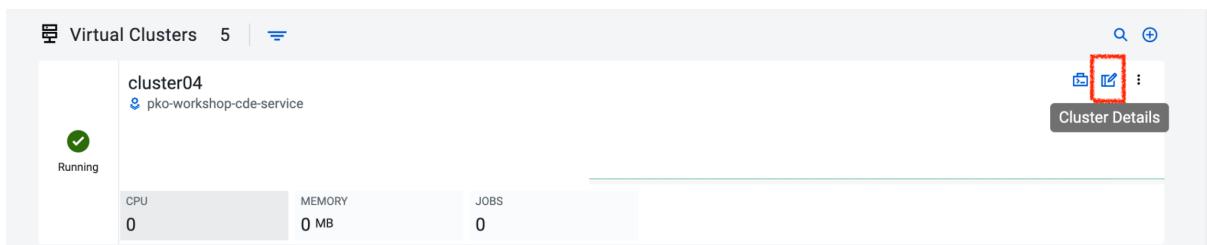
Status	Job	Type	Schedule	Modified On	Actions
○	hostcz_Lab5_airflow_dag	Airflow	*/20 * * * *	May 11, 2023, 11:43:07 AM	⋮ Run Now Pause Schedule Configuration Delete
○	skillupuser_Lab3A_access_logs_ETL	Spark	*/10 * * * *	May 11, 2023	⋮
○	hostcz_Lab3B3_Create_Reports	Spark	*/10 * * * *	May 10, 2023	⋮ Configuration Delete
○	hostcz_Lab3B2_Data_Extraction_Over_150k	Spark	*/10 * * * *	May 10, 2023, 4:01:34 AM	⋮

- You can go to the AirFlow UI again and see that the Job is now in Paused State

The screenshot shows the AirFlow UI with a single DAG listed: 'apac52\_Airflow\_Dag'. Above the DAG name, a status message 'DAG is Paused' is displayed. The DAG icon is greyed out, indicating it is currently paused.

## Lab 5 - Install and Configure CDE CLI

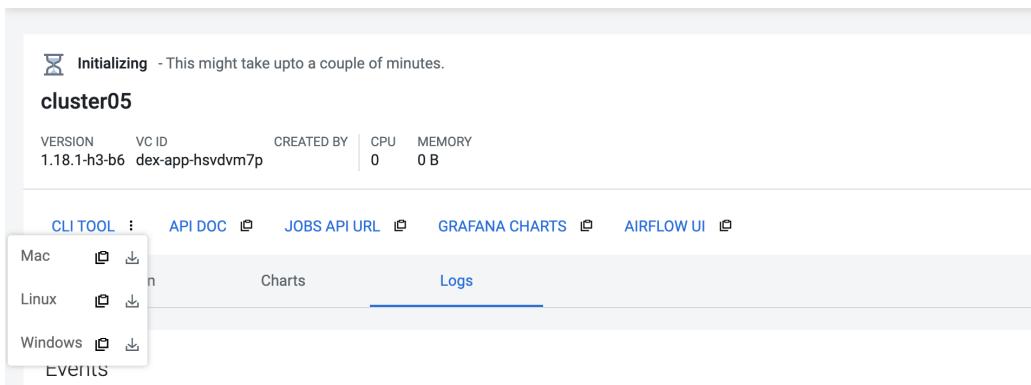
- In this lab, we will use the CDE CLI to create and run a spark job. This way, you can use the rich api's of CDE CLI to integrate any of your applications to communicate with the CDE service.
- The CLI executable can be downloaded from the virtual cluster.
  - **Step 1 :** Go to the **Cluster Details** of the virtual cluster where you are creating your job



The screenshot shows the Cloudera Data Engine (CDE) interface. On the left, there's a sidebar with a search icon and a plus sign. The main area is titled "Virtual Clusters" and shows 5 clusters. One cluster, "cluster04", is highlighted and has a green checkmark icon next to it, indicating it is running. Below the cluster name, it says "pko-workshop-cde-service". To the right of the cluster details, there are three icons: a magnifying glass, a gear, and a three-dot menu. Below these icons is a "Cluster Details" button, which is also highlighted with a red box. At the bottom of the cluster card, there are metrics: CPU 0, MEMORY 0 MB, and JOBS 0.

- **Step 2 :** Click on CLI TOOL to download the executable based on your operating system.

Administration / Virtual Cluster / cluster05



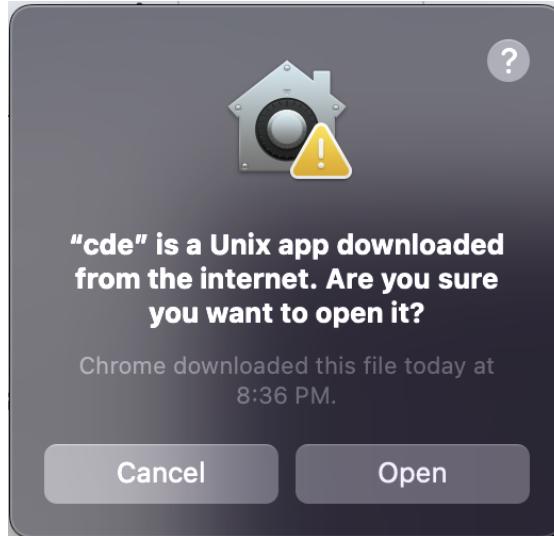
The screenshot shows the detailed view for cluster05. At the top, there's a message: "Initializing - This might take upto a couple of minutes." Below that, the cluster details are shown: VERSION 1.18.1-h3-b6, VC ID dex-app-hsvdvm7p, CREATED BY, CPU 0, and MEMORY 0 B. There are tabs for "CLI TOOL", "API DOC", "JOBS API URL", "GRAFANA CHARTS", "AIRFLOW UI", and "AIRFLOW UI". Under the "CLI TOOL" tab, there are three dropdown menus for "Mac", "Linux", and "Windows", each with a "Download" button. Below these dropdowns are two tabs: "Charts" and "Logs", with "Logs" being the active tab. At the bottom, there are tabs for "Events" and "Logs".

### For Mac users:

- Make sure that the `cde` file is executable by running the below command.

```
chmod +x /path/to/cde
```

- Go to the folder where the executable is present. Right click and select “Open with” -> Terminal . You will get the below message



- Click on Open
- Once done, you will get the following window and message

```
--credentials-profile string    CDP credentials profile name (default "defa
ult")
  -h, --help                      help for cde
  --hide-progress-bars            hide progress bars for file uploads
  --region string                 CDP Control Plane region ("us-west-1", "eu-
1" or "ap-1") (default "us-west-1")
  --skip-credentials-file        skip CDP credentials file discovery
  --tls-ca-certs string          additional PEM-encoded CA certificates
  --tls-insecure                  skip verification of API server TLS certifi
cate
  --user string                   CDP user to authenticate as
  --vcluster-endpoint string     CDE virtual cluster endpoint
  -v, --verbose                   verbose logging
  --version                       version for cde

Use "cde [command] --help" for more information about a command.

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

- To validate the installation, run the below command from the terminal.  
./cde --help

```
mehra@Downloads % chmod +x cde
[mehra@ Downloads % ./cde --help
Cloudera Data Engineering

Usage:
  cde [command]

Available Commands:
  airflow      Airflow commands
  backup       Create and Restore CDE backups
  credential   Manage CDE credentials
  help         Help about any command
  job          Manage CDE jobs
  resource     Manage CDE resources
  run          Manage CDE runs
  spark        Spark commands

Flags:
  --access-key-id string      access key identifier
  --access-key-secret string   access key secret
  --auth-cache-file string    token file cache location (default "$USERCACHE/token-cache")
  --auth-no-cache              do not cache authentication tokens
  --auth-pass-file string    authentication password file location
  --cdp-endpoint string       CDP API endpoint (default depends on CDP Control Plane region)
  --credentials-file string   CDP credentials file location
  --credentials-profile string CDP credentials profile name (default "default")
  -h, --help                  help for cde
  --hide-progress-bars        hide progress bars for file uploads
  --region string             CDP Control Plane region ("us-west-1", "eu-1" or "ap-1") (default "us-west-1")
  --skip-credentials-file     skip CDP credentials file discovery
  --tls-ca-certs string      additional PEM-encoded CA certificates
  --tls-insecure               skip verification of API server TLS certificate
  --user string                CDP user to authenticate as
  --vcluster-endpoint string  CDE virtual cluster endpoint
  -v, --verbose                verbose logging
  --version                   version for cde

Use "cde [command] --help" for more information about a command.
[mehra@ Downloads % ]
```

- If you get the output as shown above, then the installation is completed successfully. We now need to configure the CLI to connect to our virtual cluster.
- For configuring the CDE CLI, we create a new file and add the cluster details and use it as an environment variable for connecting to the CDE virtual cluster.
- Create a file as config.yaml and add the following details.

```
touch config.yaml
```

```
[mehra@MacBook-Pro workshop % touch config.yaml
[mehra@MacBook-Pro workshop % ls
  cde      config.yaml
[mehra@MacBook-Pro workshop % ]
```

```
vi config.yaml
```

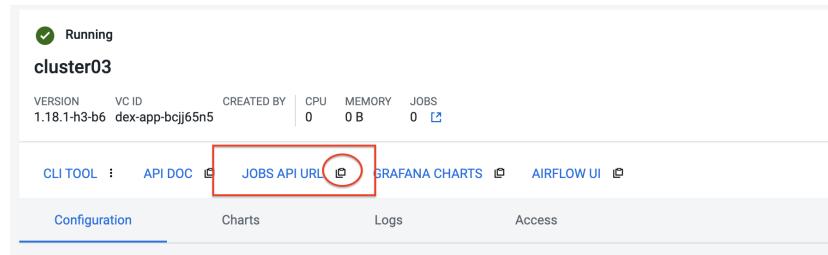
```
user: <CDP_user>
vcluster-endpoint: <CDE_virtual_cluster_endpoint>
```

Here, **user** is the username you have been mapped in the excel sheet.

**vcluster-endpoint** can be obtained from the Virtual Cluster that is assigned to you. Go to the Virtual Cluster “Cluster Details”



Click on the copy icon next to JOBS API URL to copy the vcluster-endpoint



The terminal window displays the copied JOBS API URL: user: wuser01  
vcluster-endpoint: https://hg98mkgj.cde-7fsd4m65.pse-part.dp5i-5vkq.cloudera.site/dex/api/v1

- Save config.yaml
- Run the below command to validate the configuration. Upon running it, you will be asked to provide the API password. Please enter the password as mentioned in the excel sheet.

```
./cde job list
```

- Once you enter the password, you should see all the jobs present in the virtual cluster.

```
mmehra@ Downloads % ./cde job list
API User Password: ----- Enter your workload password
[{
  "name": "apac52_Lab3A_access_logs_ETL",
  "type": "spark",
  "created": "2023-05-09T14:05:57Z",
  "modified": "2023-05-09T14:05:57Z",
  "retentionPolicy": "keep_indefinitely",
  "mounts": [
    {
      "resourceName": "apac52_resources"
    }
  ],
  "spark": {
    "file": "Lab3A_access_logs_ETL.py",
    "driverMemory": "1g",
    "driverCores": 1,
    "executorMemory": "1g",
    "executorCores": 1,
    "conf": {
      "dex.safariEnabled": "false",
      "spark.pyspark.python": "python3"
    },
    "logLevel": "INFO"
  },
  "schedule": {
    "enabled": false,
    "user": "host_mmehra"
  }
},
{
  "name": "apac52_Lab3B1_Data_Extraction_Sub_150k",
  "type": "spark",
  "created": "2023-05-09T14:06:40Z",
  "modified": "2023-05-09T14:29:43Z",
  "retentionPolicy": "keep_indefinitely",
  "mounts": [
    {
      "resourceName": "apac52_resources"
    }
  ],
  "spark": {
    "file": "Lab3B1_Data_Extraction_Sub_150k.py",
    "driverMemory": "1g",
    "driverCores": 1,
    "executorMemory": "1g",
    "executorCores": 1,
    "conf": {
      "dex.safariEnabled": "false",
      "spark.pyspark.python": "python3"
    },
    "logLevel": "INFO"
  },
  "schedule": {
    "enabled": false,
    "user": "host_mmehra",
    "cronExpression": "* /2 * * * *",
    "start": "2023-05-09T14:25:52.455Z",
    "end": "2023-05-10T14:25:52.455Z"
  }
},
{
  "name": "apac52_Lab3B2_Data_Extraction_Over_150k",
  "type": "spark",
  "created": "2023-05-09T14:07:10Z",
  "modified": "2023-05-09T14:36:32Z",
  "retentionPolicy": "keep_indefinitely",
}
```

- If you get any error related to the certificate, please add the flag to skip tls verification.

```
./cde job list --tls-insecure
```

- This marks the end of installation and configuration of CDE CLI. Now, head over to the next lab to trigger the jobs from CLI.

## For Windows users:

- Open Powershell and navigate to the folder where you have downloaded the cde.exe file.
- You can use the below command to navigate.

```
cd C:\Users\<path-to-cde.exe folder>
```

- Run the below command to start the cde cli. It will be executed in the background.

```
start .\cde.exe
```

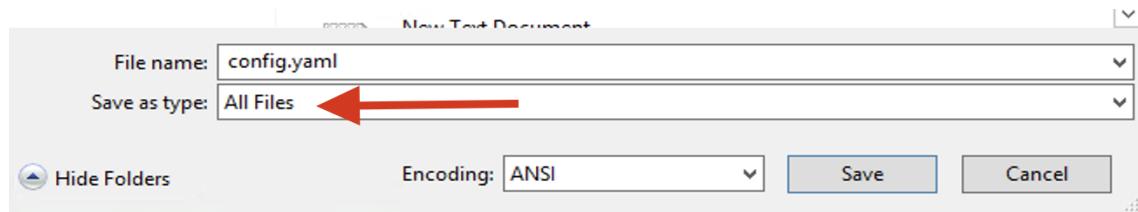
```
PS C:\Users\cdp_windows> cd C:\Users\cdp_windows\Desktop
PS C:\Users\cdp_windows\Desktop> ls

    Directory: C:\Users\cdp_windows\Desktop

Mode                LastWriteTime      Length Name
----                -----
-a---        10/28/2021  8:37 AM     38056448 cde.exe
-a---        10/28/2021  8:52 AM       106 config.yaml
-a---        10/28/2021  9:00 AM         0 he.txt
-a---        10/28/2021  8:51 AM         0 New Text Document.txt

PS C:\Users\cdp_windows\Desktop> start .\cde.exe
PS C:\Users\cdp_windows\Desktop> _
```

- Create a new text file and name it as *config.yaml*. Please note that while saving, choose the format as **All Files** and NOT as **Text Documents**.



- Add the following lines in this file.

```
user: <CDP_user>
vcluster-endpoint: <CDE_virtual_cluster_endpoint>
```

Here, **user** is the username you have been mapped in the excel sheet. For the **vcluster-endpoint** get in touch with the instructor.

[Can be obtained from your virtual cluster]

- Open Powershell and run the below command to create an environment variable.

```
$env:CDE_CONFIG = "C:\Users\<path-to-config.yaml>"
```

- Run the below command for validation. You should see the path-to-config.yaml as the output.

```
ls $env:CDE_CONFIG
```

```
PS C:\Users\cdp_windows\Desktop> start .\cde.exe
PS C:\Users\cdp_windows\Desktop> $env:CDE_CONFIG = "C:\Users\cdp_windows\Desktop\config.yaml"
PS C:\Users\cdp_windows\Desktop> ls env:CDE_CONFIG
```

Name	Value
CDE_CONFIG	C:\Users\cdp_windows\Desktop\config.yaml

```
PS C:\Users\cdp_windows\Desktop>
```

- Run the below command to validate the configuration. Upon running it, you will be asked to provide the API password. Please enter the workload password as mentioned in the excel sheet.

```
.\cde job list
```

```
PS C:\Users\cdp_windows\Desktop> .\cde job list
API User Password: _
```

- If you get the below error related to certificate, please follow the next step to skip tls verification.

```
x509: certificate signed by unknown authority
[Errno 1] _ssl.c:501: error:1409008B:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
```

- Run the below command with the tls flag and enter the API password.

```
.\cde job list --tls-insecure
```

```
PS C:\Users\cdp_windows\Desktop> .\cde job list --tls-insecure
WARN: Plaintext or insecure TLS connection requested, take care before continuing. Continue? yes/no [no]: yes
API User Password: _
```

- Once you enter the password, you should see all the jobs present in the virtual cluster.
- This marks the end of installation and configuration of CDE CLI. Now, head over to the next lab to trigger the jobs from CLI.

## Lab 6 - Run jobs using CDE CLI

You can use the CLI to create and update jobs, view job details, manage job resources, run jobs, and so on. Please use the link below to read more about the usage of CLI to manage CDE jobs.

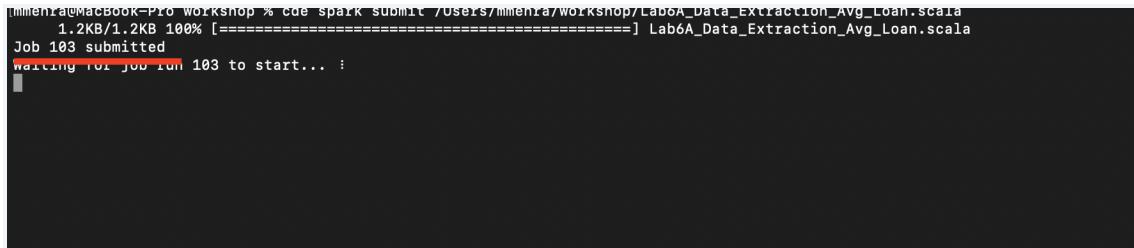
<https://docs.cloudera.com/data-engineering/cloud/cli-access/topics/cde-cli-manage-jobs.html>

### Run a spark-scala job using CLI

As a first exercise in this lab, we will trigger a spark-scala job using the CDE CLI. Please note that you don't have to build a jar to submit the job to CDE.

- Locate and get the path of the script *Lab6A\_Data\_Extraction\_Avg\_Loan.scala* downloaded from the prerequisites step.
- Run the below command to submit this job to CDE.

```
./cde spark submit  
/path/to/Lab6A_Data_Extraction_Avg_Loan.scala
```



A terminal window showing the command being run and its output. The command is `./cde spark submit /path/to/Lab6A_Data_Extraction_Avg_Loan.scala`. The output shows the file being uploaded (1.2KB/1.2KB 100%), the job being submitted (Job 103 submitted), and the job status (Waiting for job run 103 to start...).

- Go to CDE UI and click on Job Runs. You will see a job submitted with the name `cli-submit-<username>-<temp-resource-id>`

Run ID	Job ↑	Type	User	Duration
103	<a href="#">cli-submit-wuser01-1635928294681</a>	Spark	wuser01	1.3 MIN

- You can observe the logs and SparkUI for this Job Run.
- Please note that you are not creating this as a job in CDE. It will be an ad-hoc run without the need of registering it as a job.

## Lab 7 - Data Lineage and Auto-Scaling

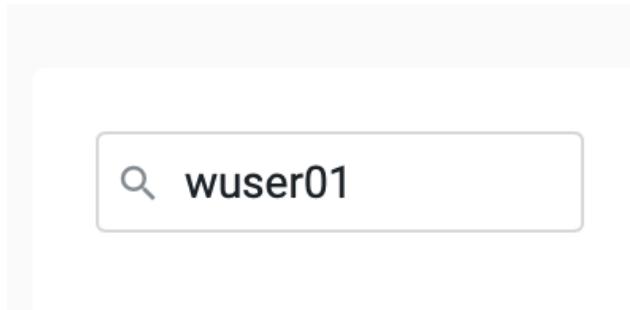
---

In this lab, you will go through the data lineage of the two use cases that we worked on. Additionally, you will also see the auto-scaling capabilities of CDE service with the rising demand for compute resources.

### Data Lineage using Atlas

- In the CDE UI, click on the Jobs tab. Go to the job `<username>_Lab3B3_Create_Report`s that you have created in the Lab2.
- To get the jobs, please filter the jobs with your username.

### Jobs



- In Run History tab, click on the successful Run ID i.e., the one with the green tick mark.

Status Ad-Hoc	Runs 3
------------------	-----------

Run History   Configuration   Schedule

Duration

Search by Run Id

Status	Run ID	Duration	Executor Memory	Drivers Cores	Executor Cores
✗	33	19 sec	1G	1	1
✗	32	44 sec	1G	1	1
✓	14	5.6 min	1G	1	1

(Note the ID will be different for you from the one you see in the screenshot)

- Click on **Atlas** under Lineage.

## Job Runs / 14

The screenshot shows a UI component with two sections. On the left, under 'Status', there is a green checkmark icon followed by the word 'Success'. A large red arrow points from this section to the right. On the right, under 'Lineage', the word 'Atlas' is displayed in blue text.

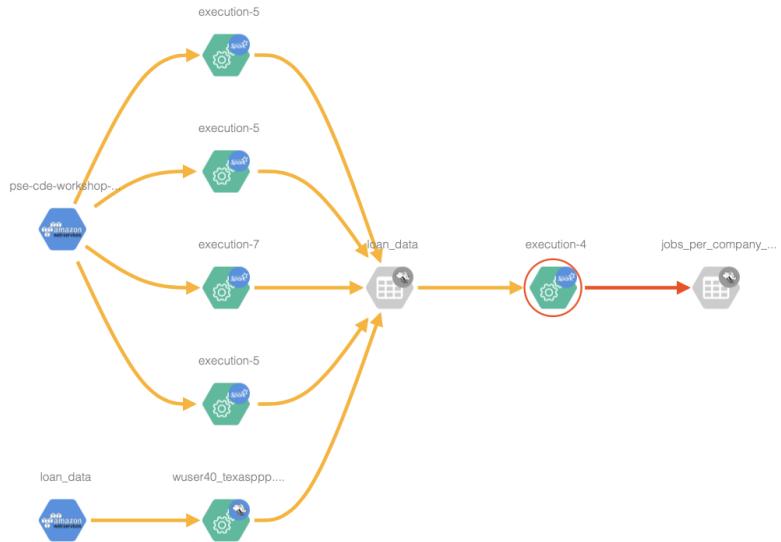
- Click on the execution that you see in the list.

Name	Owner	Description	Type	Classifications	Term
execution-4			spark_process	<a href="#">+</a>	<a href="#">+</a>

- Click on **Lineage** to observe the Data Lineage for this job.

The screenshot shows a navigation bar with several tabs: 'Properties', 'Lineage' (which is highlighted in green), 'Relationships', 'Classifications', and 'Audits'. To the left of the tabs, the word 'Terms:' is followed by a green square button containing a white plus sign.

○ Current Entity ⏳ In Progress → Lineage → Impact

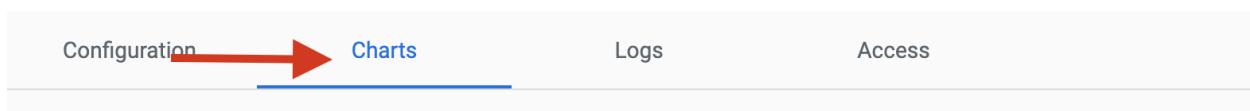


- Click on each entity to understand how the data is flowing from source to consumption.

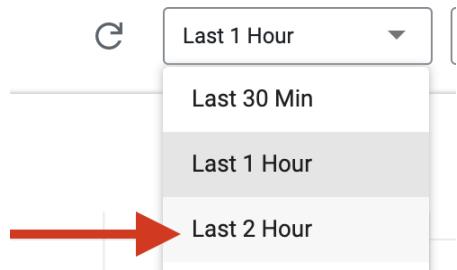
## Auto-scaling in CDE

- As a last step, we want you to witness the auto-scaling capabilities of CDE. At the start of the lab, you might have noticed the cpu and memory consumption of the virtual cluster. Please check the dashboard now to see how it has scaled up based on the demand experienced.
- On the CDE home page, click on the **Cluster Details** on the virtual cluster.
- Click on the **Charts** tab.

CLI TOOL : API DOC JOBS API URL | GRAFANA CHARTS | AIRFLOW UI



- Set the filter to **Last 2 Hour** and observe the varying load on cpu and memory.



- Click on **Grafana Charts** to view another set of metrics of the virtual cluster.

CLI TOOL : [API DOC](#) [JOBS API URL](#) **GRAFANA CHARTS** | [AIRFLOW UI](#)

- This marks the end of the overall CDE Hands-on Workshop session.

**THANK YOU VERY MUCH FOR YOUR PARTICIPATION**